Advanced machine learning techniques based on DCA and applications to predictive maintenance

Hoang Phuc Hau Luu

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Techniques avancées d'apprentissage automatique basées sur DCA et applications à la maintenance prédictive

THÈSE

présentée et soutenue publiquement le 21 octobre 2022

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

LUU Hoang Phuc Hau

Composition du jury

Président :	Anne BOYER	Professeur, Université de Lorraine
Rapporteurs :	Emilio CARRIZOSA	Professeur, Université de Séville
	Jalal FADILI	Professeur, ENSI de Caen
Examinateurs :	Stéphane CANU	Professeur, INSA Rouen Normandie
	Xuan Vinh DOAN	Reader, Université de Warwick
	Yann GUERMEUR	Directeur de Recherche, LORIA
Invité :	Tao PHAM DINH	Professeur émérite, INSA Rouen Normandie
Directrice de thèse :	Hoai An LE THI	Professeur, Université de Lorraine
Co-directeur de thèse :	Hoai Minh LE	Professeur, Université de Lorraine

Laboratoire de Génie Informatique, de Production et de Maintenance — EA 3096

Mis en page avec la classe thesul.

Remerciements

Cette thèse n'aurait pas été possible sans les encouragements, les conseils et le soutien de mes conseillers, de mon comité de thèse, de mes collègues, de mes amis et de ma famille. Trois années ont passé avec beaucoup de souvenirs que je n'oublierais jamais. La thèse a été réalisée au LGIPM de l'Université de Lorraine pendant une période de turbulence et d'incertitude causée par la pandémie de COVID-19. La thèse touche maintenant à sa fin et il est temps de dire merci.

Tout d'abord, je tiens à remercier ma directrice de thèse, le professeur Le Thi Hoai An, Université de Lorraine, pour ses conseils qui sont essentiels à la réalisation de ma thèse. Je lui suis reconnaissant pour sa patience, sa confiance et son attention pendant ma thèse de doctorat et aussi pendant la période où j'ai travaillé au LGIPM en tant qu'étudiant stagiaire. Tout au long de ces années éprouvantes, elle m'a donné une liberté totale pour poursuivre une variété de sujets de recherche intéressants. Ses commentaires constructifs sur mon travail m'ont appris à mener des recherches significatives et ont grandement amélioré mes capacités de réflexion critique.

Je suis reconnaissant à mon co-directeur de thèse, le professeur Le Hoai Minh, Université de Lorraine, pour ses encouragements, son soutien et ses discussions utiles qui m'ont aidé à améliorer mes recherches sous de nombreux aspects. Ses conseils sont essentiels pour me permettre de mener à bien cette thèse. Il m'a également donné d'excellentes conditions de travail et des outils pour mes recherches.

Je suis reconnaissant au professeur Pham Dinh Tao, l'INSA de Rouen, qui m'a offert d'innombrables discussions qui ont réellement inspiré ma passion pour les mathématiques. Ces discussions instructives ont considérablement élargi mes idées sur l'optimisation, en particulier la programmation non convexe. Je le remercie également pour ses encouragements durant la période difficile de la pandémie.

Je tiens à remercier le professeur Jalal Fadili, ENSI de Caen, et le professeur Emilio Carrizosa, Université de Séville, pour avoir accepté d'être rapporteurs de ma thèse. J'apprécie le temps précieux que vous avez consacré à la lecture de ma thèse.

Je suis reconnaissant au professeur Stéphane Canu, INSA Rouen Normandie, au professeur Anne Boyer, Université de Lorraine, au professeur Xuan Vinh Doan, Université de Warwick, au professeur Yann Guermeur, LORIA, d'avoir accepté d'être membres du comité de ma thèse.

Je tiens à remercier le professeur Duong Minh Duc, Université des sciences - VNUHCM, pour sa remarquable contribution au programme PUF et son inspiration.

Je voudrais remercier le professeur Huynh Van Ngai, Université de Quy Nhon, pour ses commentaires perspicaces qui m'ont aidé à améliorer l'aspect théorique de la thèse.

Je suis très reconnaissant du temps et du soutien crucial fournis par le professeur Loic Colson, Université de Lorraine, et le professeur Laurent Vigneron, Université de Lorraine, en tant que mon comité de suivi.

Je suis reconnaissant au professeur Pascal Omnes, Université Sorbonne Paris Nord, qui m'a apporté un soutien complet pendant le programme du Master 2 du PUF. Ce fut une période difficile mais il m'a facilité la tâche.

Une gratitude toute particulière va au professeur Marc Peigné (Université de Tours), au professeur Kilian Raschel (Université de Tours), au professeur François James (Université d'Orléans), au professeur Jing-Rebecca Li (École polytechnique), au professeur Mounir Haddou (INSA Rennes), au professeur

Laurence Halpern et au professeur Juliette Ryan (Université Sorbonne Paris Nord), qui ont apporté des changements fondamentaux dans mon point de vue sur les mathématiques appliquées.

Je suis reconnaissant au professeur Pham The Bao, Université de Saigon, de m'avoir enseigné la vision par ordinateur et le traitement d'images, qui ont servi de base à mon entrée dans le domaine de l'intelligence artificielle. C'est également lui qui m'a enseigné la rédaction scientifique et m'a beaucoup aidé lorsque je travaillais au laboratoire IPIC (traitement de l'image et informatique intelligente). En outre, je tiens à remercier tous les membres du laboratoire IPIC, Vu, Van, Hai, Thi, Nhu, Thuy, Tran, et bien d'autres, pour les moments agréables que nous avons passés ensemble.

Mes sincères remerciements vont également à tous les membres du LGIPM, en particulier, Vinh Thanh, Tran Bach, Aurelié, Sarah Blind, Viet Anh, Phuong Anh, Sara Samir, Pham Tuan, Tuyet Trinh, Khac Linh, Truc Vy, Prof. Xuan Vinh, Gaëlle, Mme Greppi, Mme Wiemert. J'apprécie votre gentillesse et votre amabilité, qui ont contribué à faire de notre séjour au LGIPM un merveilleux souvenir. En particulier, j'ai passé la majorité de mon temps au LGIPM avec Thanh, Tuan, Trinh, et Linh, qui m'ont beaucoup soutenu pendant ma thèse.

Je tiens à remercier mes anciens professeurs de mathématiques, Xuan Phong, Thai Duy, Van Khoi, Ngoc Quynh, Ngoc Phuc, Van Minh et bien d'autres, pour avoir nourri mon intérêt pour les mathématiques et les sciences lorsque j'étais lycéen.

D'autres amis à qui je dois des remerciements sincères sont Viet Cuong, Cong Duc, Trung Hau, Kim Trinh, Quoc Viet, Huu Viet, Trung Tin, Hong, Thanh Tin, Duc Vinh, and Nga. Merci pour votre compagnie et vous allez tous me manquer.

Je suis reconnaissant à Amélie de m'avoir appris le français et pour sa générosité et son amitié.

Je tiens à exprimer ma sincère gratitude à Thach Son et Van Thanh en particulier pour m'avoir accueilli pendant mes jours à Paris.

Je suis reconnaissant à mes amis proches, Khang Truong, Thinh Bui, Khai Hoan, Minh Tri, Son Huynh, et Hai pour tous les souvenirs que nous avons eu pendant les jours de collège. Avoir votre compagnie à ce moment le plus merveilleux de ma vie est une bénédiction.

Je voudrais remercier mes camarades de classe de 12T2 (Tê Tu) au lycée Thoai Ngoc Hau pour les merveilleux moments que nous avons passés ensemble. Mes remerciements vont également à tous mes amis des classes 14TTH1, 14TTH1TN, et PUF pour tout ce que nous avons vécu.

Je tiens à exprimer ma gratitude particulière à ma petite amie, Diem Phuong, pour son amour et sa gentillesse. Merci d'être toujours à mes côtés, qu'il pleuve ou qu'il vente.

Je tiens à exprimer ma profonde gratitude à ma famille, qui m'a apporté un soutien indéfectible et des encouragements incessants tout au long de ma vie. Mon père, un apprenant passionné, qui a abandonné l'école en septième année à cause de la pauvreté, a tout sacrifié pour soutenir mon éducation. J'espère que les études que j'ai menées jusqu'à présent permettront de réaliser en partie son rêve d'apprendre. En outre, je suis reconnaissant à tous les membres de ma famille élargie, notamment Tri, Quoc, Nhanh, Thuan An, Le, Huy, Thuong, et bien d'autres.

Je tiens à remercier la France d'avoir été si gentille avec moi. Je me suis senti aimé et soutenu tout au long de ces années éprouvantes.

Enfin, je voudrais exprimer ma gratitude à Thay Thich Nhat Hanh, un moine bouddhiste simple et humble, qui a apporté la paix intérieure dans le monde entier.

Je dédie cette thèse
à mon père, Luu Van Hieu,
et à ma mère, Hoang Thi Do,
qui me soutiennent toujours dans ma vie.

LUU Hoang Phuc Hau

Né le 18 mai 1996 au Vietnam
E-mail: hoang-phuc-hau.luu@univ-lorraine.fr , et hoangphuchau.luu@gmail.com
Adresse professionnelle: Bureau UM-AN1-34, LGIPM - Université de Lorraine, 3 rue Augustin Fresnel, BP 45112, 57073 METZ Cedex 03, France

Situation Actuelle

Depuis 10/2019	Doctorant au LGIPM (Laboratoire de Génie Informatique, de Production et de Maintenance - EA 3096), Université de Lorraine, France Encadré par Prof. Hoai An Le Thi and Prof. Hoai Minh Le Sujet de thèse: **Techniques avancées d'apprentissage automatique basées sur DCA et applications à la maintenance prédictive**

Experience Professionnelle

4/2019 - 7/2019	Stagiaire au laboratoire LGIPM, Université de Lorraine, France Mémoire: **Online DCA for stochastic learning**
9/2018 - 12/2018	Assistant d'enseignement à l'Université Nationale du Vietnam Ho Chi Minh Ville, Université des Sciences
6/2016 - 8/2016	Stagiaire au laboratoire Nakagawa, Université d'agriculture et de technologie de Tokyo, Tokyo, Japon Mémoire: **Applying language model for handwriting recognition**

Diplôme et Formation

2019 - present	Doctorant en Informatique, LGIPM, Université de Lorraine, Metz, France
2018 - 2019	Master 2 en Mathématiques Appliquées, Université Sorbonne Paris Nord, Paris, France
2014 - 2018	Diplôme Universitaire, programme d'honneur en Mathématiques, Université Nationale du Vietnam Ho Chi Minh Ville, Université des Sciences

Publications

Referred international journals

[J1] H. A. Le Thi, H. P. H. Luu, & T. Pham Dinh (2022). Online Stochastic DCA with applications to Principal Component Analysis. **Accepted for publication**, *IEEE Transactions on Neural Networks and Learning Systems*. The corresponding arXiv preprint `arXiv:2108.02300`.

[J2] H. A. Le Thi, V. N. Huynh, T. Pham Dinh, & H. P. H. Luu (2022). Stochastic Difference-of-Convex Algorithms for Solving nonconvex optimization problems. *SIAM Journal on Optimization*, 32(3), 2263-2293. `https://doi.org/10.1137/20M1385706`

[J3] H. A. Le Thi, H. P. H. Luu, H. M. Le, & T. Pham Dinh (2022). Stochastic DCA with Variance Reduction and applications in Machine Learning. *Journal of Machine Learning Research*, 23(206):1–44. `http://jmlr.org/papers/v23/21-1146.html`

[J4] H. A. Le Thi, T. T. T. Nguyen, & H. P. H. Luu (2022). A DC programming approach for solving a centralized group key management problem. *Journal of Combinatorial Optimization*, 1-29. `https://doi.org/10.1007/s10878-022-00862-1`

[J5] H. P. H. Luu, H. M. Le, & H. A. Le Thi (2022). Markov Chain Stochastic DCA and Applications in Deep Learning with PDEs Regularization. **To be submitted**.

Referred international conference papers

[C1] T. T. T. Nguyen, H. P. H. Luu & H. A. Le Thi (2022). Solving a Centralized Dynamic Group Key Management Problem by an Optimization Approach. In: *Le Thi, H.A., Pham Dinh, T., Le, H.M. (eds) Modelling, Computation and Optimization in Information Systems and Management Sciences. MCO 2021. Lecture Notes in Networks and Systems*, vol 363. Springer, Cham.

[C2] V. T. Pham, H. P. H. Luu, & H. A. Le Thi (2022). A block coordinate DCA approach for large-scale kernel SVM. In: *Computational Collective Intelligence. ICCCI 2022. Lecture Notes in Artificial Intelligence.* Springer, Cham.

Book chapters

H. A. Le Thi, T. Pham Dinh, H. P. H. Luu, & H. M. Le, Deterministic and stochastic DCA for DC programming, *Handbook of Engineering Statistics 2nd edition*, In press, Springer.

Communications in national / international conferences

H. P. H. Luu, H. A. Le Thi, & T. Pham Dinh (2021). Acceleration techniques in DC programming. *World Congress on Global Optimization*, Athens, Greece.

Contents

List of Figures

List of Tables

Abbreviations and Notations

The key abbreviations and notations used in this thesis are listed as follows.

DC	Difference of convex functions
DCA	DC algorithm
SGD	stochastic gradient descent
prox	proximal
SVRG	stochastic variance reduced gradient
RNN	recurrent neural network
LSTM	long short-term memory
FC	fully-connected
drop	dropout
RUL	remaining useful life
SoH	state-of-health
$\mathbb{R}$	set of real numbers
$\mathbb{R}^n$	set of real vectors of size n
$\nabla f(x)$	gradient of f at x
$\partial f(x)$	subdifferential of f at x
$\mathrm{dom}\, f$	effective domain of a function f
$\mathbb{E}(X)$	expectation of the random variable X
$\mathbb{E}(X\|\mathcal{P})$	conditional expectation of X given sigma-algebra $\mathcal{P}$
$\propto$	is proportional to
$\mathcal{N}(0, I)$	standard Gaussian distribution
$P(x, dy)$	Markov transition kernel
O	big O notation

Introduction générale

Cadre général et nos motivations

L'intelligence artificielle, avec l'apprentissage automatique et l'apprentissage profond en tête, a entraîné des changements fondamentaux dans de nombreux aspects de la vie. Le mécanisme sous-jacent qui permet l'apprentissage est l'optimisation. Il existe deux grands thèmes d'optimisation : déterministe et stochastique. Tous les algorithmes qui adhèrent à l'approche d'optimisation déterministe supposent qu'il n'y a pas d'aléa/incertitude dans le problème et la conception algorithmique. De nos jours, avec la prolifération des données, qui est une bonne condition nécessaire à l'intelligence artificielle pilotée par les données, l'extensibilité des données et l'incertitude rendent le paradigme déterministe impraticable. C'est alors que l'approche stochastique entre en jeu. L'optimisation stochastique permet à l'incertitude d'apparaître dans le problème pour modéliser des quantités inconnues (programmation stochastique) et permet également aux algorithmes d'utiliser des approximations stochastiques peu coûteuses pour résoudre le problème d'extensibilité (algorithmes stochastiques). Un programme stochastique prend généralement la forme suivante

$$\min_{x \in S} F(x) = \mathbb{E}(f(x, Z)), \tag{1}$$

où Z est une variable aléatoire dont la distribution, P_Z, est inconnue en général. En général, on ne peut accéder à Z que par l'intermédiaire de ses réalisations i.i.d. (indépendantes et identiquement distribuées) $Z_1, Z_2, \ldots, Z_N$ (appelées échantillons). Dans un scénario plus difficile, ces échantillons ne sont plus i.i.d., c'est-à-dire qu'ils sont corrélés les uns aux autres, et leurs distributions sont différentes de la distribution de Z, mais convergent vers celle-ci. En particulier, une chaîne de Markov $Z_1 \to Z_2 \to \ldots \to Z_N \to \ldots$ avec une distribution d'équilibre de P_Z est un exemple typique de ce dernier cas. Dans le cadre (1), la distribution de Z ne dépend pas de la variable d'optimisation (la décision) x, on parle alors d'incertitude exogène. Dans le cas où la décision x affecte la distribution de Z, nous sommes face à une incertitude endogène, un objet bien plus compliqué que le premier.

Comme les modèles d'apprentissage automatique ont tendance à être de plus en plus compliqués pour capturer la structure complexe des données, les problèmes d'optimisation qui en résultent sont plus susceptibles d'être non convexes. Bien qu'il existe une littérature importante sur l'optimisation convexe dans des contextes déterministes et stochastiques, la recherche sur l'optimisation stochastique non convexe reste limitée. Pour contribuer à cette direction de recherche, dans cette thèse, l'optimisation stochastique non convexe est le principal sujet d'étude. La structure non convexe étudiée est la DC (différence de fonctions convexes). Comme son nom l'indique, une fonction est dite DC si elle peut être

exprimée comme une différence de deux fonctions convexes, ce qui donne le programme DC suivant

$$\min_{x\in\mathbb{R}^n}\{f(x) := g(x) - h(x)\},$$

où $g, h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ sont convexes, propres, semicontinus inférieurs. La classe des programmes DC est fermée sous l'opérateur habituel considéré en optimisation et elle est assez large pour englober la plupart des fonctions objectives du monde réel [76]. La programmation DC est une approche fondamentale de l'optimisation non convexe, non seulement en raison de sa polyvalence mais aussi de sa maniabilité, grâce à un algorithme bien établi connu sous le nom de DCA (algorithme DC) [103]. Le DCA [103, 76], introduit par Pham Dinh Tao, puis développé par lui et Le Thi Hoai An, ainsi que quelques collaborateurs, est une technique efficace pour résoudre les programmes DC, en particulier ceux avec des décompositions DC adaptées. La programmation DC et la DCA ont été appliquées avec succès dans divers domaines scientifiques, par exemple, le transport et la logistique, la fouille de données et l'apprentissage automatique, la biologie et la chimie computationnelles, les réseaux de communication, la finance et l'économie, le traitement d'images (voir [76] pour un aperçu détaillé). La principale force de DCA réside dans sa flexibilité, qui est due au fait qu'une fonction DC possède une infinité de décompositions DC, chacune d'entre elles donnant lieu à un DCA correspondant (c'est pourquoi DCA est généralement qualifié de philosophie plutôt que d'algorithme spécifique). L'objectif de cette thèse est d'étendre la programmation DC et la DCA du cadre déterministe au cadre stochastique. Dans ce dernier cadre, le principe DCA conseille comment traiter la non-convexité, alors que d'autres techniques stochastiques doivent être utilisées en plus pour traiter l'incertitude.

De nos jours, la maintenance est au centre de toute activité industrielle pour assurer la stabilité, l'efficacité et la productivité des entreprises. À l'ère de l'industrie intelligente, une politique de maintenance correcte peut faire une grande différence dans la compétitivité d'une entreprise. Grâce à sa capacité à prévoir les défauts de fonctionnement et à estimer l'état de santé des actifs, la maintenance prédictive peut minimiser les risques, les coûts de maintenance et les temps d'arrêt tout en améliorant la productivité. Pour mettre en œuvre la maintenance prédictive, il faut soit une connaissance considérable du domaine pour modéliser précisément le mécanisme physique sous-jacent de la dégradation, soit utiliser l'intelligence artificielle pour prédire les défauts directement à partir des données des capteurs. Les connaissances d'experts n'étant pas disponibles dans la plupart des cas, cette dernière approche est préférable grâce à sa facilité de déploiement, à la technologie de pointe des capteurs qui permet une collecte de données de haute qualité, et aux améliorations rapides de l'apprentissage automatique et de l'apprentissage profond. Le problème de la maintenance prédictive, lorsqu'on utilise l'approche axée sur les données, est essentiellement un problème de prédiction de séries temporelles, où l'objectif est de prédire les quantités d'intérêt telles que la durée de vie utile restante d'un moteur. Les réseaux neuronaux profonds sont des candidats intéressants pour les prédicteurs, car la relation entre les données du capteur et les quantités d'intérêt peut être hautement non linéaire. Compte tenu de la topologie d'un réseau neuronal, l'ajustement des paramètres internes du réseau pour capturer une telle relation est fondamentalement un problème d'optimisation (hautement) non convexe.

Nos contributions

L'objectif principal de la thèse est de concevoir et d'étudier des algorithmes stochastiques innovants basés sur la programmation DC et DCA afin de relever les nouveaux défis de l'apprentissage automatique, en particulier l'apprentissage profond. D'une part, les algorithmes sont conçus pour plusieurs classes de programmes d'optimisation stochastique non convexes, avec des analyses de convergence fermes fournies. Ils sont, d'autre part, utilisés pour résoudre des problèmes d'apprentissage automatique tels que l'analyse en composantes principales, la sélection de variables de groupe dans la régression logistique multiclasse et l'entraînement de réseaux neuronaux profonds via la régularisation des EDP (équations différentielles partielles). En tant qu'application industrielle, les méthodes proposées seront appliquées à la maintenance prédictive via une approche d'apprentissage profond.

Sur le plan théorique, nous concevons et étudions un certain nombre d'algorithmes stochastiques pour résoudre des programmes DC stochastiques. Tout d'abord, nous considérons un programme DC avec une structure à somme élevée. La structure à somme élevée se présente généralement comme le problème de minimisation du risque empirique (ERM) du programme stochastique (1) utilisant un grand ensemble d'échantillons i.i.d.. Autrement dit, l'espérance de (1) est approximée par la méthode de Monte-Carlo. Dans une telle situation, le nombre de sommets est équivalent au nombre de points de données utilisés, ce qui est énorme pour les ensembles de données modernes. Une approche naturelle est d'approximer itérativement les quantités pertinentes de grandes sommes (telles que le sous-gradient de la deuxième composante DC) par certaines contreparties stochastiques dans un style mini-batch. Cependant, la variance de cette approche est élevée, ce qui fait que la séquence générée fluctue à proximité des points critiques mais ne converge pas vers ces points critiques. Nous proposons donc deux nouveaux schémas DCA stochastiques, DCA-SVRG et DCA-SAGA, qui combinent des techniques de réduction de la variance et étudient les deux stratégies d'échantillonnage, à savoir l'échantillonnage avec et sans remplacement. La convergence presque sûre des algorithmes proposés vers les points critiques DC est établie, et la complexité des méthodes est examinée. Deuxièmement, nous étudions comment résoudre directement les programmes DC stochastiques avec une variable aléatoire arbitraire Z (sous la forme de (1) où $f(\cdot, z)$ sont des fonctions DC). En comparaison, la première approche résolvant le problème ERM aura quelque chose à voir avec une sorte d'écart de généralisation. Nous essayons maintenant - au moins théoriquement - de combler ce fossé. La situation d'intérêt est i.i.d. les échantillons de la distribution de Z arrivent séquentiellement (données en continu), donc on n'a pas un grand nombre d'échantillons au début et l'algorithme proposé devrait être capable d'apprendre en ligne. Nous concevons un algorithme appelé DCA stochastique en ligne et deux variantes pour résoudre ce problème d'apprentissage statistique. Étant donné que la taille du mini-batch augmente à un rythme approprié au fil des itérations, la convergence presque certaine vers les points critiques DC est assurée. Troisièmement, nous étudions une classe de programmes DC stochastiques dans lesquels l'incertitude endogène est présente et les échantillons i.i.d. sont *indisponibles*. Au lieu de cela, nous supposons que seules les chaînes de Markov ergodiques suffisamment rapides pour atteindre les distributions cibles sont accessibles. Ce contexte est valable car le bruit markovien se produit dans une variété de contextes, y compris l'inférence bayésienne, l'apprentissage par renforcement et l'optimisation stochastique dans des espaces hautement dimensionnels ou combinatoires. Nous concevons ensuite un algorithme stochastique appelé DCA stochastique par chaîne de Markov (MCSDCA) et fournissons l'analyse de convergence au sens asymptotique et non asymptotique. De plus, nous étendons notre théorie pour inclure les chaînes de

Markov *temps-inhomogène*. Il s'agit d'une extension importante puisque certaines chaînes de Markov basées sur la diffusion qui jouent un rôle clé dans l'échantillonnage sont homogènes en temps. Enfin, nous montrons comment décomposer une large classe de structures de régularisation basées sur les EDP en décompositions DC qui peuvent être utilisées pour entraîner des réseaux neuronaux profonds. Grâce à la structure DC découverte, la MCSDCA est appliquée spécifiquement à l'apprentissage profond. Il existe deux réalisations de la MCSDCA dans cette application. La première est appelée MCSDCA-odLD (MCSDCA overdamped Langevin dynamics) où les chaînes de Markov sont construites sur la base de la diffusion de Langevin suramplifiée avec une discrétisation Euler-Maruyama. La seconde est appelée MCSDCA-udLD (MCSDCA underdamped Langevin dynamics) où les chaînes de Markov sont développées sur la base de la diffusion de Langevin sous-amortie avec un nouveau schéma de discrétisation introduit dans [25].

Sur le plan pratique, les méthodes proposées ont été appliquées à un certain nombre de problèmes d'apprentissage automatique/apprentissage profond, où leurs vertus ont été justifiées. Le DCA-SVRG et le DCA-SAGA sont appliqués à trois problèmes d'apprentissage automatique : l'analyse en composantes principales non négatives, la sélection de variables de groupe dans la régression logistique multiclasse et la régression linéaire éparse. Deux versions du DCA-SVRG (échantillonnage avec/sans remplacement) et une version du DCA-SAGA (échantillonnage sans remplacement) figurent parmi les meilleurs gagnants dans les trois problèmes. Plus important encore, les avantages de DCA-SVRG et DCA-SAGA en tant qu'algorithmes stochastiques ont été démontrés par rapport au DCA déterministe, et les mérites des techniques de réduction de variance utilisées (estimateurs SAGA et SVRG) ont été démontrés par rapport au DCA stochastique proposé dans [73] employant l'estimateur SAG. Deuxièmement, les comportements pratiques de la DCA stochastique en ligne sont étudiés en détail par le biais du problème théorique de l'analyse en composantes principales. En particulier, étant une méthode en ligne qui utilise de nouveaux échantillons frais à chaque itération, la DCA stochastique en ligne possède une capacité d'adaptation lorsqu'il y a un changement abrupt dans la distribution de Z. Cette remarque est justifiée expérimentalement. Troisièmement, et surtout, les modèles MCSDCA-odLD et MCSDCA-udLD sont utilisés pour entraîner avec succès des réseaux neuronaux profonds, tels que les réseaux neuronaux à action directe, les réseaux neuronaux récurrents et la mémoire à long terme, pour des tâches de maintenance prédictive, telles que la prédiction de la durée de vie utile restante (RUL) et l'estimation de l'état de santé (SoH)/capacité. D'une part, le MCSDCA-odLD a montré sa force dans le problème de prédiction de la durée de vie utile résiduelle, où il surpasse ou rivalise avec un certain nombre d'optimiseurs de base dans l'apprentissage profond, notamment Adam, Adagrad, RMSprop et SGD. D'autre part, le MCSDCA-udLD a démontré qu'il avait une capacité d'accélération supérieure à celle du MCSDCA-odLD dans la tâche d'estimation de la capacité. Étant donné que la dynamique de Langevin sous-amortie présente un certain nombre d'avantages par rapport à la dynamique de Langevin suramortie, notamment l'accélération, ce comportement correspond à nos attentes initiales. Le MCSDCA-udLD bat également les optimiseurs de base mentionnés pour la tâche d'estimation de la capacité. Dans l'ensemble, les deux méthodes proposées ont entraîné efficacement les réseaux neuronaux profonds, car les valeurs prédites de RUL/capacité correspondent étroitement aux valeurs réelles.

Organisation de la thèse

La thèse se compose de six chapitres. Le chapitre 1 contient des informations préliminaires sur la programmation DC et la DCA. Dans le chapitre 2, nous étudions une classe de programmes DC dont les fonctions objectives ont une structure à somme élevée, et nous proposons le DCA-SVRG et le DCA-SAGA. Dans le chapitre 3, nous considérons les programmes DC stochastiques, où nous concevons la DCA stochastique en ligne et ses variantes. La DCA stochastique par chaîne de Markov est développée dans le chapitre 4 pour une classe de programmes DC stochastiques endogènes avec des échantillons non-i.i.d.. Le chapitre 5 traite des applications de la maintenance prédictive. Enfin, le chapitre 6 conclut la thèse.

Chapter 1

Preliminaries

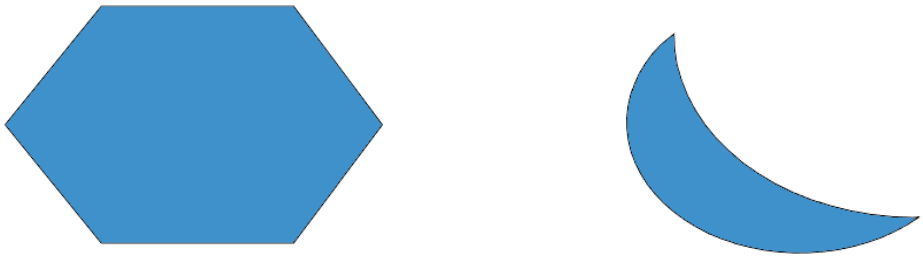
1.1 Convex and variational analysis

To begin, we review some notions from Convex Analysis and Nonsmooth Analysis that will be useful later (see, e.g., [112, 113, 26, 114, 15, 91]).

Firstly, for $x \in \mathbb{R}^n$ and $r > 0$, we denote by $B(x, r)$ the open ball of radius r and center x,

$$B(x, r) = \{y \in \mathbb{R}^n : \|y - x\| < r\}.$$

A set $C \subset \mathbb{R}^n$ is said to be *convex* if $\lambda x + (1 - \lambda)y \in C$, for all $x, y \in C, \lambda \in [0, 1]$. That is, for every couple of points in C, the *line segment* connecting these two points is still contained in C. The weighted sum $\lambda x + (1 - \lambda)y$ with $\lambda \in [0, 1]$ is called a *convex combination* of x and y. More generally, for a finite set of points $x_1, x_2, \ldots, x_k$ and $\lambda_1, \lambda_2, \ldots, \lambda_k \geq 0$ with $\lambda_1 + \lambda_2 + \ldots + \lambda_k = 1$, we call $\lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_k x_k$ a convex combination of $x_1, x_2, \ldots, x_k$. It can be shown that a set is convex if and only if it contains every convex combination of its points. Figure 1.1 illustrates a convex set and a nonconvex set.



(a) a convex set (b) a nonconvex set

Figure 1.1: Examples on convex and nonconvex sets

For a nonconvex set C, one can convexify it by defining a set of all convex combinations of C. This set

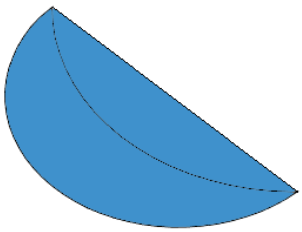


Figure 1.2: Convex hull

is called a convex hull of C, denoted by $\operatorname{conv} C$,

$$\operatorname{conv} C := \{\lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_k x_k : x_i \in C, \lambda_i \geq 0, \forall i = \overline{1,k}, \lambda_1 + \lambda_2 + \ldots + \lambda_k = 1\}.$$

The convex hull of C is the smallest convex set that contains C. For example, Figure 1.2 depicts the convex hull of the nonconvex set presented in Figure 1.1 (b).

If we relax the condition $\lambda \in [0,1]$ in the definition of convex set, we obtain the notion of affine set: a set $C \subset \mathbb{R}^n$ is *affine* if the line passing through any two distinct points in C remains in C, i.e., for $x_1, x_2 \in C$ and $\lambda \in \mathbb{R}$, $\lambda x_1 + (1-\lambda)x_2 \in C$. As a matter of fact, a nonempty affine set is always parallel to a unique linear subspace of $\mathbb{R}^n$. One then defines the dimension of an affine set to be the dimension of the corresponding parallel linear subspace.

Likewise, in the above definition of convex hull, if the condition $\lambda_1, \lambda_2, \ldots, \lambda_k \geq 0$ is relaxed, we obtain the notion of affine hull,

$$\operatorname{aff} C := \{\lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_k x_k : x_i \in C, \forall i = \overline{1,k}, \lambda_1 + \lambda_2 + \ldots + \lambda_k = 1\}.$$

The affine hull of C is the smallest affine set containing C.

Let Ω be a subset of $\mathbb{R}^n$, we call Ω a *cone* if $\lambda x \in \Omega$ whenever $\lambda \geq 0$ and $x \in \Omega$. Let $C \subset \mathbb{R}^n$ be a convex set and $\bar{x} \in C$. The *normal cone* of C at $\bar{x}$ is defined as follows

$$N_C(\bar{x}) := \{v \in \mathbb{R}^n : \langle v, x - \bar{x} \rangle \leq 0, \forall x \in C\}.$$

For a set C, we define the *relative interior* of C, denoted by $\operatorname{ri} C$, as follows:

$$\operatorname{ri} C := \{x \in C : B(x,r) \cap \operatorname{aff} C \subset C \text{ for some } r > 0\}.$$

When the affine hull of C is full dimension, i.e., $\operatorname{aff} C = \mathbb{R}^n$, the notion of relative interior coincides with the notion of interior (denoted by int).

For a nonempty convex set $C \subset \mathbb{R}^n$, the above notion of relative interior is reduced to

$$\operatorname{ri} C := \{x \in C : \forall y \in C, \exists \lambda > 1 \text{ such that } \lambda x + (1-\lambda) y \in C\}.$$

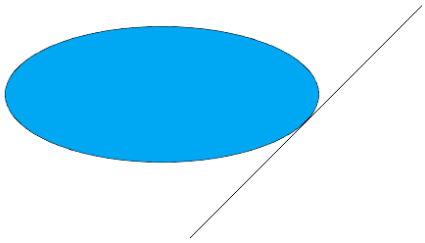


Figure 1.3: Supporting hyperplane of a convex set

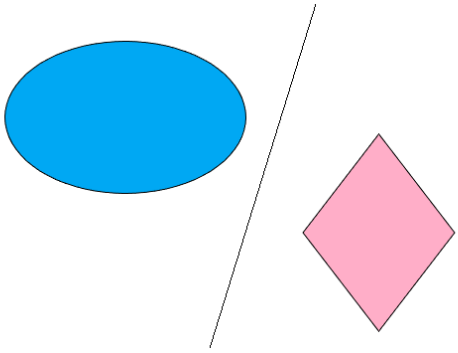


Figure 1.4: Separate two disjoint convex sets by a hyperplane

In plain language, this means that x belongs to the relative interior of a convex set C if one can prolong (a bit) the line segment connecting an arbitrary point $y \in C$ to x at the endpoint x without leaving the set C. The relative interior of a convex set is a convex set. Moreover, for two convex sets C and D, the inclusion $C \subset D$ does not necessarily imply $\operatorname{ri} C \subset \operatorname{ri} D$, unless $\operatorname{aff} C = \operatorname{aff} D$.

An important geometrical characteristic of a convex set is that a convex set "bends outwards". Due to this characteristic, it is possible to support a convex set - at each boundary point of it - by a hyperplane, i.e., a set of the form $\{x \in \mathbb{R}^n : \langle v, x\rangle = \alpha\}$ with $v \neq 0$ (see Figure 1.3) and it is possible to separate two disjoint convex sets by a hyperplane (see Figure 1.4). Formal statements for these properties can be found, for example, in [15].

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be an extended-real-valued function. The effective domain of f is defined as

$$\operatorname{dom} f := \{x \in \mathbb{R}^n : \quad f(x) < +\infty\}.$$

The function f is said to be proper if $\operatorname{dom} f \neq \emptyset$.

For $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, the *epigraph* of f is defined as

$$\operatorname{epi} f = \{(x, t) : x \in \operatorname{dom} f, f(x) \leq t\}.$$

Meanwhile, the *graph* of f, $\operatorname{gph} f$, is given by

$$\operatorname{gph} f = \{(x, t) : x \in \operatorname{dom} f, f(x) = t\}.$$

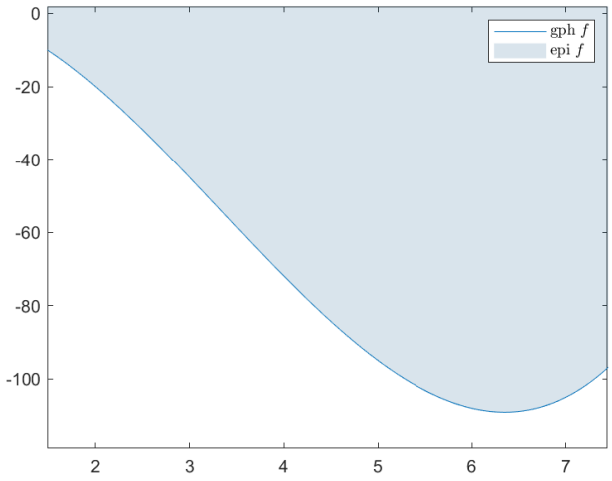


Figure 1.5: Graph and epigraph of a function f

The function f is said to be convex if its epigraph is a convex set in $\mathbb{R}^{n+1}$. Or equivalently,

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y), \text{ for all } x, y \in \mathbb{R}^n, \lambda \in [0,1]. \tag{1.1}$$

Furthermore, f is called ρ-convex for some $\rho \geq 0$, if $f - (\rho/2)\|\cdot\|^2$ is a convex function. Or equivalently, for all $x, y \in \mathbb{R}^n, \lambda \in [0,1]$,

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) - \frac{\rho}{2}\lambda(1-\lambda)\|x-y\|^2. \tag{1.2}$$

The supremum of all $\rho \geq 0$ such that the inequality (1.2) holds is called the convex modulus of f, denoted by $\rho(f)$ or ρ_f. If $\rho(f) > 0$, f is said to be strongly convex. Note that, the notion of convexity (and strong convexity) can be defined locally over a nonempty convex set contained in $\operatorname{dom} f$ as follows: f is convex on C if the inequality (1.1) holds true for all $x, y \in C, \lambda \in [0,1]$.
A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is Lipschitz continuous on $\Omega \subset \operatorname{dom} f$ if there is a constant $L \geq 0$ such that

$$|f(x) - f(y)| \leq L\|x-y\|, \quad \forall x, y \in \Omega.$$

Meanwhile, f is said to be locally Lipschitz continuous around $\bar{x} \in \operatorname{dom} f$ if there exists $L_{\bar{x}} \geq 0$ and $\delta_{\bar{x}} > 0$ such that

$$|f(x) - f(y)| \leq L_{\bar{x}}\|x-y\|, \quad \forall x, y \in B(\bar{x}, \delta_{\bar{x}}).$$

If f is a convex function, it is locally Lipschitz continuous in the interior of its domain, $\operatorname{int}(\operatorname{dom} f)$.
Let $\Omega \subset \operatorname{dom} f$ be an open set and suppose that f is differentiable on Ω, f is said to have L-Lipschitz continuous gradient for some $L > 0$ (L-smooth) on Ω if

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|, \quad \forall x, y \in \Omega.$$

Let $x \in \mathrm{dom}(f)$, a vector $z \in \mathbb{R}^n$ is called a subgradient of f at x if

$$f(y) - f(x) \geq \langle z, y - x \rangle, \quad \forall y \in \mathbb{R}^n.$$

The set of all subgradients of f at x is called the subdifferential of f at x, denoted by $\partial f(x)$,

$$\partial f(x) := \{z \in \mathbb{R}^n : f(y) - f(x) \geq \langle z, y - x \rangle, \quad \forall y \in \mathbb{R}^n\}. \tag{1.3}$$

By definition, the subdifferential of f at x, if nonempty, offers affine minorants to f at x (affine functions whose graphs are below the graph of f and touch the graph of f at x), $y \mapsto f(x) + \langle z, y - x \rangle$, for $z \in \partial f(x)$. Figure 1.6 illustrates affine minorants of f.

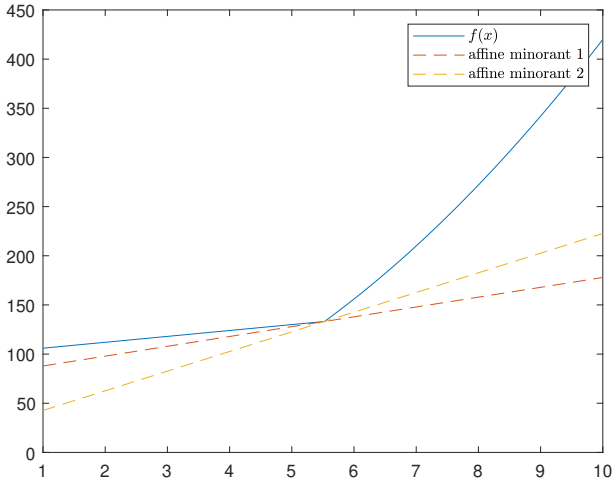


Figure 1.6: Affine minorants

And, f is said to be subdifferentiable at x if $\partial f(x) \neq \emptyset$. Also, by definition,

$$\mathrm{dom}\, \partial f = \{x \in \mathbb{R}^n : \partial f(x) \neq \emptyset\}.$$

For a proper convex function f, the following relations hold

$$\mathrm{ri}(\mathrm{dom}\, f) \subset \mathrm{dom}\, \partial f \subset \mathrm{dom}\, f.$$

For $\epsilon \geq 0$, the ϵ-subdifferential (which is a relaxation of the notion of subdifferential) of f at x is defined as follows

$$\partial_\epsilon f(x) := \{z \in \mathbb{R}^n : f(y) - f(x) \geq \langle z, y - x \rangle - \epsilon, \quad \forall y \in \mathbb{R}^n\}.$$

While the the concepts of subgradient and subdifferential do not assume that f is convex, they do not make much sense for a nonconvex function f as the subdifferential of a nonconvex function can easily be an empty set. Therefore, when it comes to subgradient and subdifferential, one usually refers to a convex function (in such a case, $\mathrm{dom}\, \partial f \supset \mathrm{ri}(\mathrm{dom}\, f)$). Furthermore, for $x \in \mathrm{dom}\, f$ where f is convex,

the subdifferential $\partial f(x)$ can be characterized in terms of normal cone of $\operatorname{epi} f$ as follows,

$$\partial f(x) = \{v \in \mathbb{R}^n : (v, -1) \in N_{\operatorname{epi} f}((x, f(x)))\}.$$

In particular, if f is the indicator function of a convex set C,

$$f(x) = \chi_C(x) := \begin{cases} 0 & \text{if } x \in C, \\ +\infty & \text{otherwise,} \end{cases}$$

the above relation implies $\partial f(x) = N_C(x)$.
The function f is said to be lower semicontinuous at $x \in \mathbb{R}^n$ if

$$f(x) \leq \liminf_{y \to x} f(y).$$

Moreover, f is called lower semicontinuous on $\mathbb{R}^n$ if f is lower semicontinuous at every point of $\mathbb{R}^n$. It should be noted that a function is lower semicontinuous if and only if its epigraph is closed.
For a proper function f and $x^* \in \operatorname{dom} f$, x^* is called a global minimizer of f if $f(x^*) \leq f(x), \forall x \in \mathbb{R}^n$; Meanwhile, x^* is called a local minimizer (resp. strict local minimizer) of f if there is $r > 0$ such that $f(x^*) \leq f(x), \forall x \in B(x^*, r)$ (resp. $f(x^*) < f(x), \forall x \in B(x^*, r) \setminus \{x^*\}$). For a convex function, a local minimizer is also a global minimizer. This fundamental fact differentiates the level of difficulty between convex optimization and nonconvex optimization, where the latter does not guarantee the equivalence between locality and globality. It follows straightforwardly from the definition of subgradient that: x^* is a global minimizer of f if and only if $0 \in \partial f(x^*)$.
The conjugate of a function f is defined as

$$f^*(y) := \sup\{\langle x, y\rangle - f(x) : x \in \mathbb{R}^n\}, \quad \text{for } y \in \mathbb{R}^n.$$

It follows directly from the definition of f^* that

$$f^*(y) + f(x) \geq \langle x, y\rangle, \quad \forall x, y \in \mathbb{R}^n,$$

which is usually referred to as *Young's inequality*. An interesting property of conjugation is that the conjugate of the conjugate (known as biconjugate) of a proper, convex, lower semicontinuous function is itself, i.e., $f^{**} = f$, which is known as the Fenchel-Moreau theorem. Hereafter, we denote the class of proper, convex, lower semicontinuous functions by $\Gamma_0(\mathbb{R}^n)$.
A direct implication of the Fenchel-Moreau theorem is the following equivalence. Let $f \in \Gamma_0(\mathbb{R}^n)$, then

$$y \in \partial f(x) \Leftrightarrow x \in \partial f^*(y). \tag{1.4}$$

In summary, the following theorem encompasses essential properties of functions in $\Gamma_0(\mathbb{R}^n)$.

Theorem 1. *Let $f \in \Gamma_0(\mathbb{R}^n)$. The following properties hold true.*
1. $f^* \in \Gamma_0(\mathbb{R}^n)$, $f^{**} = f$.
2. $y \in \partial f(x) \Leftrightarrow x \in \partial f^*(y) \Leftrightarrow f(x) + f^*(y) = \langle x, y\rangle$. Meanwhile, the inequality $f(x) + f^*(y) \geq \langle x, y\rangle$ always holds true for all $x, y \in \mathbb{R}^n$.

3. f is continuous on $\operatorname{int}(\operatorname{dom} f)$.

4. The subdifferential mapping ∂f is closed: for two sequences $\{x^k\}$ and $\{y^k\}$ such that $y^k \in \partial f(x^k)$ and $x^k \to x^*$, $y^k \to y^*$, then $y^* \in \partial f(x^*)$.

5. Let S be a compact set of $\operatorname{int}(\operatorname{dom} f)$*, then $\partial f(S) := \cup\{\partial f(x) : x \in S\}$ is a compact set, and let*

$$\gamma := \sup\{\|y\| : y \in \partial f(S)\} < +\infty.$$

In this case f is Lipschitz continuous on S with

$$|f(u) - f(x)| \leq \gamma \|u - x\|, \forall (u, x) \in S \times S$$

6. $0 \in \partial f(x^*)$ if and only if x^* is a solution of the convex program

$$\inf\{f(x) : x \in \mathbb{R}^n\}.$$

As mentioned earlier, although the notion of subdifferential (1.3) can be defined without the requirement of convexity, it is less meaningful in the nonconvex setting since this subdifferential is easily an empty set for nonconvex functions. Therefore, one needs some alternatives. We next recall some different notions of subdifferentials and stationarity for nonsmooth, nonconvex functions. For a lower semicontinuous function, the *Fréchet subdifferential* of f at $x \in \operatorname{dom} f$ is defined as

$$\partial^F f(x) = \left\{ z \in \mathbb{R}^n : \ \liminf_{h \to 0} \frac{f(x+h) - f(x) - \langle z, h \rangle}{\|h\|} \geq 0 \right\}.$$

For $x \notin \operatorname{dom} f$, we set $\partial^F f(x) = \emptyset$. Note that, even if $x \in \operatorname{dom} f$, $\partial^F f(x)$ can be empty, e.g., take $f(u) = -|u|$ where $u \in \mathbb{R}$. The set $\partial^F f(x)$ is a closed and convex set. However, the Fréchet subdifferential is not a closed mapping, hence it is computationally unstable [11]. For this reason, one considers the following "limiting version" of Fréchet subdifferential, called *limiting subdifferential*. By definition, the limiting subdifferential of f at $x \in \operatorname{dom} f$ is

$$\partial f(x) = \{z \in \mathbb{R}^n : \ \exists (x_k, f(x_k)) \to (x, f(x)), \ z_k \in \partial^F f(x_k), \ \ z_k \to z\},$$

and we put $\partial f(x) = \emptyset$ if $x \notin \operatorname{dom} f$. Now, the limiting subdifferential is a closed mapping. However, the set $\partial f(x)$ is closed but not necessarily convex.

If f is locally Lipschitz at $x \in \mathbb{R}^n$, then the *Clarke directional derivative* $f^C(x, \cdot)$ at x and the *Clarke subdifferential* $\partial^C f(x)$ are given respectively by

$$f^C(x; d) = \limsup_{(t,u) \to (0^+, x)} \frac{f(u + td) - f(u)}{t}$$
$$\text{and } \partial^C f(x) = \{y \in \mathbb{R}^n : \langle y, d \rangle \leq f^C(x; d), \forall d \in \mathbb{R}^n\}.$$

A point $x \in \mathbb{R}^n$ is called a Fréchet (resp. limiting/ Clarke) stationary point for the function f, if $0 \in \partial^F f(x)$ (resp. $0 \in \partial f(x)$/ $0 \in \partial^C f(x)$).

The following inclusions describe the relations between Fréchet, limiting, and Clarke subdifferentials:

$$\partial^F f(x) \subset \partial f(x) \subset \partial^C f(x).$$

When f is a convex function, the Fréchet, limiting, and Clarke subdifferential coincide with the Convex Analysis subdifferential (1.3).

Let f be a proper function and $x \in \operatorname{dom} f$, the directional derivative of f at x along a direction d is defined by

$$f'(x;d) = \lim_{t \downarrow 0} \frac{f(x+td) - f(x)}{t},$$

given that the limit exists. A point $x \in \operatorname{dom} f$ is d-stationary point of f iff $f'(x;d) \geq 0$ for all $d \in \mathbb{R}^n$.

If f is convex, if $x \in \operatorname{int}(\operatorname{dom} f)$, $f'(x,d)$ exists as a real number, for all direction $d \in \mathbb{R}^n$. Moreover, for $x \in \operatorname{dom} f$, the set $\partial f(x)$ can be characterized using directional derivative as follows: $y \in \partial f(x)$ iff $f'(x;d) \geq \langle d, y \rangle$ for all $d \in \mathbb{R}^n$.

1.2 DC programming and DCA

Recall that $\Gamma_0(\mathbb{R}^n)$ denotes the convex cone of all proper, lower semicontinuous, convex functions on $\mathbb{R}^n$. The standard DC program takes the form [103]

$$(P_{dc}) \qquad \alpha := \inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^n\}$$

where $g, h \in \Gamma_0(\mathbb{R}^n)$. Such a function f is called DC, $g - h$ is DC decomposition while g and h are DC components of f. It is worth noting that a DC function has infinitely many DC decompositions. We may assume that g and h are strongly convex without losing generality since f can be recast as the difference of two strongly convex functions as follows

$$f = \left(g + \frac{\rho}{2}\|\cdot\|^2\right) - \left(h + \frac{\rho}{2}\|\cdot\|^2\right), \quad \text{with } \rho > 0.$$

The class of DC functions is quite rich so as to encompass most real-world optimization problems [76]. Moreover, it is possible to uniformly approximate any continuous function over a compact set by a sequence of DC functions [3]. The class of DC functions is closed under usually operators considered in optimization, e.g., linear combination, maximum, or minimum of a finite number of DC functions are again DC functions. However, the class of DC functions is unstable under pointwise convergence and the supremum operator (of an infinite number of DC functions, of course) [28].

A DC program with a convex constraint $x \in C$ can be written in form of the standard DC program (P_{dc}) by integrating the indicator function of C, χ_C, into the first DC component, $\hat{g} = g + \chi_C$. When a DC function is minimized over nonconvex DC constraints, the optimization problem is no longer a standard DC program. Rather, it is referred to as a general DC program [68, 104]. For general DC programs with DC constraints, there are two approaches to convert them to standard DC programs: applying penalty technique or convexifying DC constraints [68, 104]. In this thesis, we are mostly concerned with standard DC programs.

The dual DC program of (P_{dc}) is defined by

$$(D_{dc}) \qquad \beta := \inf\{h^*(y) - g^*(y) : y \in \mathbb{R}^n\} \qquad .$$

It can be verified that $\alpha = \beta$ and there is the perfect symmetry of the DC duality: the dual of (D_{dc}) is exactly (P_{dc}).

For a DC function $f = g - h$, a *global minimizer* of f is characterized as follows [103, Theorem 1]: x^* is the global minimizer of f if and only if $\partial_\epsilon h(x^*) \subset \partial_\epsilon g(x^*), \quad \forall \epsilon \geq 0$. It is fascinating that - for DC programming - globality can be expressed in terms of ϵ-subdifferentials. In practice, however, the condition $\partial_\epsilon h(x^*) \subset \partial_\epsilon g(x^*)$ for all $\epsilon \geq 0$ is almost impossible to achieve as it requires the inclusion to hold for all $\epsilon \geq 0$. The foregoing inclusion is naturally relaxed to hold solely for $\epsilon = 0$, giving rise to the notion of *strong (DC) criticality*: A point x^* is called strongly (DC) critical for $f = g - h$ if $\emptyset \neq \partial h(x^*) \subset \partial g(x^*)$ (note that ∂_0 coincides with the standard notion of subdifferential in convex analysis). This requirement - however - is also very hard to develop corresponding iterative algorithms. Therefore, the notion of *(DC) criticality* was invented as a further relaxation of strong DC criticality: a point x^* is called a (DC) critical point of $f = g - h$ if $0 \in \partial g(x^*) - \partial h(x^*)$, or equivalently, $\emptyset \neq \partial g(x^*) \cap \partial h(x^*)$. When h is differentiable at x^*, DC criticality becomes strong DC criticality. As a convex function is differentiable almost everywhere, a DC critical point is very likely to be strongly DC critical. However, the notion of DC criticality is now dependent on DC decompositions of f as a result of this relaxing process. Because a DC function can be decomposed in infinitely many ways, to be better understood, the above definitions of DC criticality should be related to other generalized subdifferentials of f in nonsmooth nonconvex analysis, such as Fréchet, or Clarke subdifferentials. For instance, the following inclusions give a link between DC criticality and Clarke/Fréchet stationarity

$$\partial^C f(x) \subseteq \partial g(x) - \partial h(x), \quad \partial^F f(x) \subseteq \partial g(x) - \partial h(x). \tag{1.5}$$

The first (resp. second) equality is verified *if either g or h* (resp. h) is differentiable at x. On the other hand, strong DC criticality coincides with d-stationarity on $\mathrm{ri}(\mathrm{dom}\, g) \cap \mathrm{ri}(\mathrm{dom}\, h)$ [69, 76].

DC criticality and strong DC criticality are the necessary conditions for local optimality. The sufficient conditions for local optimality are presented as follows [103]. Let x^* be a point admitting a neighborhood U such that $\partial h(x) \cap \partial g(x^*) \neq \emptyset, \forall x \in U \cap \mathrm{dom}\, g$, then x^* is a local minimizer of f. On the other hand, if $x^* \in \mathrm{int}(\mathrm{dom}\, h)$ satisfies $\partial h(x^*) \subset \mathrm{int}(\partial g(x^*))$, then x^* is a strict local minimizer of f.

In practice, to measure the goodness of the found solution, one also uses the notions of *ϵ-criticality* and *nearly ϵ-criticality* [134]. A point x^* is called ϵ-critical of (P_{dc}) if $\mathrm{dist}(\partial g(x^*), \partial h(x^*)) \leq \epsilon$, while it is called nearly ϵ-critical if there exists a point $\bar{x}$ such that $\|\bar{x} - x^*\| = O(\epsilon)$ and $\mathrm{dist}(\partial g(\bar{x}), \partial h(\bar{x})) \leq \epsilon$, where $\mathrm{dist}(A, B)$ denotes the distance between two sets A and B.

The DCA [103, 76, 75] is an effective method for solving DC programs, especially when the DC decomposition is tailored. Iteratively linearizing the second DC component, $h_k(x) = h(x^k) + \langle x - x^k, y^k \rangle$, where $y^k \in \partial h(x^k)$, and solving the resultant convex program to yield x^{k+1}, is essentially how DCA works. The DCA subsequentially converges to DC critical points while guaranteeing a decrease in the objective value across iterations.

Standard DCA.

Initialization: Let $x^0 \in \operatorname{dom} \partial h$ and $k = 0$.

repeat

Step 1: Compute the subgradient $y^k \in \partial h(x^k)$.

Step 2: Solve the convex program $x^{k+1} \in \arg\min\{g(x) - h_k(x) : x \in \mathbb{R}^n\}$.

Step 3: $k = k + 1$.

until Stopping criterion.

The reader is referred to [103, 75, 75] for the DCA's convergence properties and the theoretical underpinning of DC programming. Some essential properties of the sequence generated by DCA are presented in Theorem 2.

Theorem 2. *The sequence $\{x^k\}$ generated by DCA has the following properties:*

1. *The sequence $\{(g-h)(x^k)\}$ is decreasing,*

$$(g-h)(x^{k+1}) \leq (g-h)(x^k) - \frac{\rho(g)+\rho(h)}{2}\|x^{k+1}-x^k\|^2, \quad \forall k \in \mathbb{N}.$$

2. *If $(g-h)(x^{k+1}) = (g-h)(x^k)$, then x^k and x^{k+1} are critical points of (P_{dc}):*

$$y^k \in \partial g(x^k) \cap \partial h(x^k) \text{ and } y^k \in \partial g(x^{k+1}) \cap \partial h(x^{k+1}).$$

3. *If $\rho(g) + \rho(h) > 0$ then the series $\sum_{k=1}^{\infty} \|x^{k+1}-x^k\|^2$ is convergent, and*

$$\min\{\|x^{l+1}-x^l\| : l = 0, 1, \ldots, k\} \leq \frac{\sqrt{2}(f(x^0)-\alpha)^{1/2}}{(\rho(g)+\rho(h))^{1/2}(k+1)^{1/2}},$$

   so, $\min\{\|x^{l+1}-x^l\| : l = 0, 1, \ldots, k\} = O(k^{-1/2})$.

4. *If the optimal value α of the problem (P_{dc}) is finite and the sequences $\{x^k\}$ and $\{y^k\}$ are bounded, then every limit point of $\{x^k\}$ is a critical point of (P_{dc}), and every limit point of $\{y^k\}$ is a critical point of (D_{dc}).*

We observe that item 3 of Theorem 2 guaranteeing $\sum_{k=1}^{\infty} \|x^{k+1}-x^k\|^2 < +\infty$ is one step away from the convergence of the whole sequence $\{x^k\}$. Indeed, if a stronger condition $\sum_{k=1}^{\infty} \|x^{k+1}-x^k\| < +\infty$ holds, it follows that $\{x^k\}$ is a Cauchy sequence, so it is convergent.

We visualize how DCA works through the following simple example. Let us consider a DC function $f = g - h$ where $g = x^2 + 2x$ and $h = 5|x| + 0.005(x+1)^4$. Figure 1.7 depicts the graphs of f and its two DC components. With the mentioned DC decomposition, DCA iteratively constructs a sequence of convex surrogates (by linearizing h at the current point and then minimizing the convex surrogate to get a next point). Figures 1.8 illustrates the sequence of convex surrogates generated by DCA, starting from two different initial points.

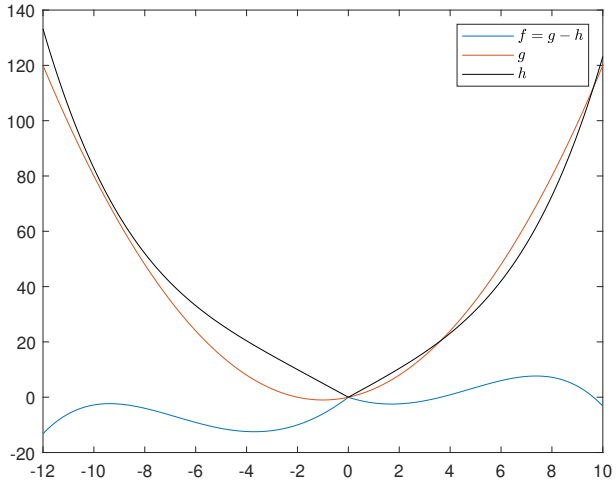
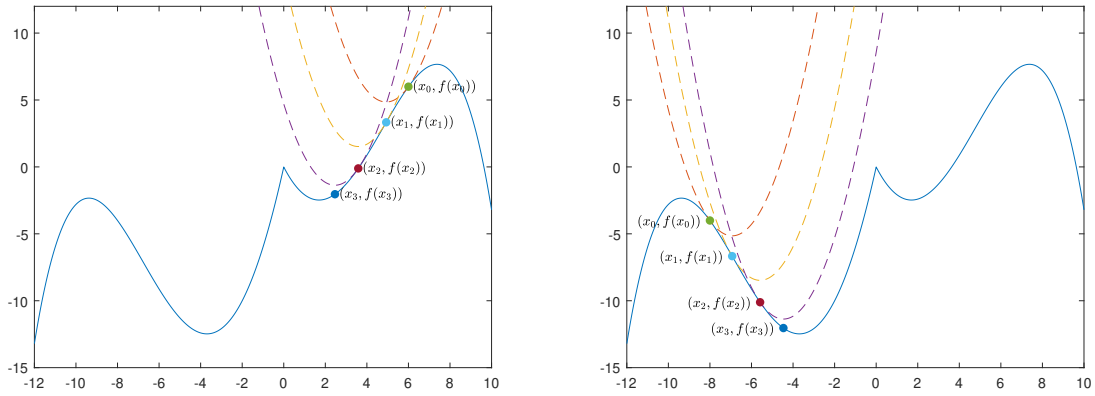


Figure 1.7: DC decomposition of $f = g - h$, where $g = x^2 + 2x$ and $h = 5|x| + 0.005(x + 1)^4$.



(a) Starting point $x_0 = 6$

(b) Starting point $x_0 = -8$

Figure 1.8: How DCA works

A duality point of view

Step 1 of the DCA, $y^k \in \partial h(x^k)$, is equivalent to $x^k \in \partial h^*(y^k)$ thanks to (1.4), or

$$h^*(z) \geq h^*(y^k) + \langle x^k, z - y^k \rangle, \quad \forall z \in \mathbb{R}^n, \tag{1.6}$$

which is further equivalent to

$$y^k \in \arg\min\{h^*(z) - \langle x^k, z \rangle : z \in \mathbb{R}^n\}.$$

Likewise, Step 2 of the DCA, $x^{k+1} \in \arg\min\{g(x) - \langle y^k, x \rangle\}$, is equivalent to $y^k \in \partial g(x^{k+1})$. Thanks to (1.4), it is further equivalent to

$$x^{k+1} \in \partial g^*(y^k). \tag{1.7}$$

From (1.6) and (1.7), we have the following observation: if $\{(x^k, y^k)\}$ is the sequence generated by DCA applied to the primal DC program (P_{dc}), then $\{(y^k, x^{k+1})\}$ is the sequence generated by DCA applied to the dual DC program (D_{dc}). Consequently, for each convergence property of the primal sequence $\{x^k\}$, there is a corresponding convergence property for the dual sequence $\{y^k\}$.

The convergence of DCA for subanalytic functions

It is noteworthy that Theorem 2 only guarantees the *subsequence convergence* of the sequence generated by DCA. Recent advances in DC programming and DCA have shed some light on the convergence of the whole sequence for subanalytic functions, as well as the convergence rates of the sequence to DC critical points. Such convergence results will be presented in the remainder of this introductory chapter. First, let us review some subanalytic set and function concepts [83, 10].

1. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called real-analytic if for each point $p \in \mathbb{R}^n$, there is a convergent series that converges to f in some neighborhood of p.

2. A subset C of $\mathbb{R}^n$ is said to be semianalytic if for each point of $\mathbb{R}^n$ there exists a neighborhood V such that $C \cap V$ is of the following form:

$$C \cap V = \bigcup_{i=1}^{p} \bigcap_{j=1}^{q} \{x \in V : \ f_{ij}(x) = 0, \ g_{ij}(x) > 0\},$$

   where $f_{ij}, \ g_{ij} : V \to \mathbb{R}$ $(1 \leq i \leq p, 1 \leq j \leq q)$ are real-analytic functions.

3. A subset C of $\mathbb{R}^n$ is called subanalytic if for each point of $\mathbb{R}^n$ there exists a neighborhood V such that

$$C \cap V = \{x \in \mathbb{R}^n : \exists \, y \in \mathbb{R}^m, (x, y) \in D\},$$

   where D is a bounded semianalytic subset of $\mathbb{R}^n \times \mathbb{R}^m$ for some $m \geq 1$.

4. A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is said to be subanalytic if its graph $\operatorname{gph} f$ is a subanalytic subset of $\mathbb{R}^{n+1}$.

Subanalytic functions enjoy the following Łojasiewicz subgradient inequality.

Theorem 3. *(Theorem 3.1 [10]) Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a subanalytic function such that its domain $\operatorname{dom} f$ is closed and $f \mid_{\operatorname{dom} f}$ is continuous and let x_0 be a Fréchet critical point of f. Then, there exist $\theta \in [0, 1)$, $L > 0$ and a neighborhood V of x_0 such that the following inequality holds,*

$$|f(x) - f(x_0)|^\theta \leq L\|x^*\| \quad \forall \, x \in V, \ x^* \in \partial^F f(x),$$

where a convention $0^0 = 1$ is used.

The number θ is called a *Łojasiewicz exponent* of the function f at the critical point x_0.
Under subanalytic data, Theorem 4 establishes the convergence of the whole sequence generated by DCA, and Theorem 5 provides the convergence rate of such a sequence to its limit point (the limit point is also a DC critical point), where the convergence rate depends on the Łojasiewicz exponent of f at that limit point (see [69] for more details).

Theorem 4. *Let us consider DC problem (P_{dc}) with $\alpha \in \mathbb{R}$. Suppose that the sequences $\{x^k\}$ and $\{y^k\}$ are defined by the DCA. Suppose that the DC function $f := g - h$ is subanalytic such that $\operatorname{dom} f$ is closed; $f\,|_{\operatorname{dom} f}$ is continuous and that around every critical point of (P_{dc}), either g or h is differentiable with locally Lipschitz derivative. Assume that $\rho := \rho(g) + \rho(h) > 0$, where $\rho(g)$ and $\rho(h)$ are modulus of the strong convexity of g and h, respectively. If either the sequence $\{x^k\}$ or $\{y^k\}$ is bounded, then $\{x^k\}$ and $\{y^k\}$ are convergent to critical points of (P_{dc}) and (D_{dc}), respectively.*

Theorem 5. *Suppose that the assumptions of Theorem 4 are satisfied. Let x^∞ be the limit point of $\{x^k\}$, where the Łojasiewicz exponent of the function f at x^∞ is denoted by $\theta \in [0, 1)$. Then, there exist constants $\tau_1, \tau_2 > 0$ such that, for all $k \in \mathbb{N}$,*

$$\|x^k - x^\infty\| \leq \sum_{j=k}^{\infty} \|x^j - x^{j+1}\| \leq \tau_1 \|x^k - x^{k-1}\| + \tau_2 \|x^k - x^{k-1}\|^{\frac{1-\theta}{\theta}}.$$

As a consequence, one has

- *If $\theta \in (1/2, 1)$, then $\|x^k - x^\infty\| \leq ck^{\frac{1-\theta}{1-2\theta}}$ for some $c > 0$.*
- *If $\theta \in (0, 1/2]$, then $\|x^k - x^\infty\| \leq cq^k$ for some $c > 0$ and $q \in (0, 1)$.*
- *If $\theta = 0$, then $\{x^k\}$ is convergent in a finite number of steps.*

Chapter 2

Stochastic DCA with Variance Reduction

Abstract. In this chapter, we develop stochastic DCA schemes for tackling a class of structured DC programs. Because the standard DCA requires full information of (sub)gradients, which can be costly in large-scale contexts, stochastic methods depend instead on stochastic information. Stochastic estimations, on the other hand, generate additional variance terms, making stochastic algorithms unstable. Therefore, we incorporate several variance reduction strategies into our design, such as SVRG and SAGA. The proposed algorithms' almost sure convergence to critical points is established, and the algorithms' complexity are examined. We apply our algorithms to three important machine learning tasks to investigate their efficiency: nonnegative principal component analysis, group variable selection in multiclass logistic regression, and sparse linear regression. Numerical studies have demonstrated the effectiveness of our proposed algorithms over existing cutting-edge stochastic approaches for addressing nonconvex large-sum problems.

2.1 Introduction

The following nonconvex optimization problem is of our interest

$$(P) \quad \min \{F(x) = G(x) - H(x) + r_1(x) - r_2(x) : x \in \mathbb{R}^n\},$$

where $H := \frac{1}{N}\sum_{i=1}^{N} h_i$, $G, h_i : \mathbb{R}^n \to \mathbb{R}$ and $r_1, r_2 : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ are convex, lower semicontinuous functions while each h_i has Lipschitz continuous gradient (with a common constant L) on the effective domain of r_1. The optimal value α of the problem (P) is assumed to be finite, i.e., $\alpha > -\infty$, which implies $\operatorname{dom} r_1 \subset \operatorname{dom} r_2$. Moreover, we assume that $\operatorname{dom} r_1 \subset \operatorname{dom} \partial r_2$.

This type of problem is common in many areas, particularly machine learning, where $G - H$ represents the data-fitting term (the loss function) and $r_1 - r_2$ represents the regularization term to encourage some desired properties on the found solutions (for example, low-rank or sparsity) or to model constraints on x. Furthermore, in the age of big data, optimization models are expected to take the large-sum structure into account so as to represent the high volume of data. As a result, the loss function frequently has a (nonconvex) large-sum structure.

The material of this chapter is extracted from the following work:
H. A. Le Thi, H. P. H. Luu, H. M. Le, & T. Pham Dinh (2022). Stochastic DCA with Variance Reduction and Applications in Machine Learning. *Journal of Machine Learning Research*, 23(206):1–44.

In terms of the regularizer, we investigate two cases of r_2: r_2 is convex (the regularizer is a DC function), or r_2 is a composite function defined by $r_2(x) = \sum_{i=1}^{m} l_i(p_i(x))$ with $l_i : \mathbb{R} \to \mathbb{R}$ being convex, decreasing and $p_i : \mathbb{R}^n \to \mathbb{R}$ being convex. In the second case, r_2, in general, is no longer convex. Most nonconvex sparsity-promoting regularizers (usually used to approximate the ℓ_0 norm) are known to have the above composite form of r_2 [71, 97], for instance, log-sum penalty [18], smoothly clipped absolute deviate (SCAD) [44], capped ℓ_1 [139], minimax concave penalty [137], (nonconcave) piecewise linear function [67].

The problem (P) catches our attention because of two prominent features that play an important role in machine learning and big data analytics: the (nonconvex) DC structure of the objective (on both loss and regularizer) and the large-sum structure of H. Indeed, as machine learning models tend to be more complicated, nonconvex optimization is then indispensable. The nonconvexity is represented here by the DC structure, which is applicable to a wide range of nonconvex optimization problems [76]. Furthermore, as previously stated, the large-sum structure is one of the most commonly seen forms in practice for modeling massive data-driven problems. The large-sum structure, for example, emerges naturally in empirical risk minimization (ERM) in stochastic programming. That is, for instance, the stochastic problem

$$\min_{x \in D} \mathbb{E}(f(x, \xi)),$$

where ξ is a random variable, can be approximated by

$$\text{(ERM)} \quad \min_{x \in D} \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i) + \lambda \Omega(x),$$

where $\{\xi_1, \xi_2, \dots \xi_N\}$ is a set of i.i.d. realizations of ξ, and Ω is the regularization term. In particular, when Ω is a convex function, D is a convex set, and $f(\cdot, \xi_i)$ has L-Lipschitz gradients for all i, the problem (ERM) falls into the spectrum of the problem (P) with $G(x) = (L/2)\|x\|^2$, $h_i(x) = (L/2)\|x\|^2 - f(x, \xi_i)$, $i = 1, 2, \dots, N$, and $r_1(x) = \lambda\Omega(x) + \chi_D(x), r_2(x) = 0$. Many important applications such as LASSO, principal component analysis, logistic regression, etc., can be expressed in this form.

Related works and our motivation The two previously mentioned aspects of the problem (P) also pose a significant challenge. On the one hand, there are very few efficient and scalable algorithms for dealing with nonconvex problems, while on the other hand, deterministic algorithms struggle to manage the large-sum structure. As a result, combining stochastic techniques for dealing with the large-sum structure with an powerful algorithm for nonconvex programming could be an efficient strategy. A stochastic algorithm often incorporates some stochastic approximation techniques based on the framework of a deterministic algorithm. This deterministic frame is critical to the overall quality of a stochastic algorithm and should thus be well suited to the problem's structure. And the stochastic approximation used should be inexpensive and has small noise.

The literature on stochastic optimization in the convex context is vast. In particular, the problem (ERM) with $f(\cdot, \xi)$ and Ω being convex, $D = \mathbb{R}^n$, has been studied extensively. Since the influential idea of stochastic gradient descent, which can be traced back to the seminal work [109], many variants have been developed, for example, stochastic dual coordinate descent [121], stochastic average gradi-

ent (SAG) [119], stochastic variance reduced gradient (SVRG) [59], SAGA [31], StochAstic Recursive grAdient algoritHm (SARAH) [94], etc. Other works studied the (strongly) convex sum function while relaxing the convexity of each component function $f(\cdot, \xi)$ [2, 120].

The number of research on nonsmooth, nonconvex large-sum problems remains small. The (ERM) problem with L-smooth $f(\cdot, \xi)$ and (possibly nonsmooth) convex regularizer Ω (which is a special case of (P)) is probably the most prevalent model that has been studied via the proximal-based approach [58, 101]. Among stochastic methods for solving the (ERM), the prox-SVRG and prox-SAGA [58] are the most related to our approach since they use the SVRG and SAGA estimators, respectively. Another work based on the Majorization-Minimization (MM) approach was proposed in [85] which studied a large-sum of nonconvex functions where each function admit surrogates with L-smooth error (i.e. the difference between a function and its surrogates is L-smooth).

The existing body of research for nonsmooth DC large-sum problems is even more limited. Some recent publications [72, 73, 134] developed stochastic methods based on DCA (DC Algorithm). The first stochastic DCA was presented in [72] for a large-sum of (nonconvex) L-smooth functions with a DC regularizer, and it was later expanded to a more general problem where the L-smooth condition was relaxed in [73]. The stochastic DCA has been constructed in these studies based on the SAG estimator [119], which is a variance reduction estimator.

Another approach was [134] in which the data-fitting term is large-sum DC and the regularizer is nonconvex, nonsmooth whose proximal operator can be efficiently computed. The author approximated the regularizer by its Moreau envelope, resulting in a DC program, in which stochastic DCA schemes were developed. In essence, the proposed algorithms can be viewed as the standard DCA where the (large-sum) convex subproblems are solved by stochastic algorithms (e.g., Adagrad [37], SVRG [59]) to a certain degree of precision.

Based on the above discussion, we are inspired to develop novel stochastic algorithms based on DCA for solving (P). In addition, several research studies have demonstrated the theoretical and practical value of stochastic variance-reduced gradients provided by SVRG and SAGA. Consequently, combining them with the DCA could be an effective strategy.

Our contribution We present two stochastic DCA schemes, DCA-SVRG and DCA-SAGA, that incorporate SVRG and SAGA. We investigate two typically used sampling strategies in our design: sampling with replacement and sampling without replacement. Sampling is called *with replacement* if one sample selected at random from the population is returned to the population, then the next sample is selected in the same manner; hence, it allows repetition. Meanwhile, the sampling *without replacement* strategy sequentially samples from the population, with each picked unit not being placed back. Algorithms for tackling large-sum problems in the literature primarily use the former sampling method due to its useful property: each sample is independent of the others. However, in practice, the latter is a much more natural method to apply and can occasionally produce significantly superior outcomes (as demonstrated by our numerical tests in Section 2.5). As a result, we give analyses for both sampling procedures.

The convergence of the DCA-SVRG and DCA-SAGA has been thoroughly investigated. The proposed algorithms' almost sure convergence to critical points is established: each accumulating point of the sequence generated by DCA-SVRG and DCA-SAGA is shown to be a DC critical point of

$F = (G + r_1) - (H + r_2)$. Furthermore, if r_2 is L-smooth, we obtain the $O(k^{-1/2})$ convergence rate with respect to the measure of proximity to criticality, where k is the number of iterations. Consequently, to find ϵ-criticality, DCA-SVRG and DCA-SAGA have the complexity of $O(N^{2/3}/\epsilon^2)$ and $O(N + N^{3/4}/\epsilon^2)$ (respectively) in terms of gradient evaluations. Another contribution is the novel arguments used in the analysis of DCA-SAGA, where we introduce an elegant idea of how to escape the non-independence (which is the main difficulty in our analysis) and provide good quantitative links between $\mathbb{E}\|x^t - \alpha_i^t\|^2$ and $\mathbb{E}\|x^t - x^{t-1}\|^2, \mathbb{E}\|x^{t-1} - x^{t-2}\|^2$, etc., where $\{\alpha_i^t\}_{i=1}^N$ are SAGA-introduced local auxiliary variables that are used to build variance reduction terms and are updated progressively. These new arguments should be helpful in analyzing future SAGA-type algorithms.

Moreover, we provide additional convergence analysis to deal with the composite structure of r_2. By introducing new optimization variables, $z_i = p_i(x)$ with $i = \overline{1, m}$, this problem can be reformulated as a DC program in the form of (P) with respect to a pair of old and new variables. Our goal here is to provide a more practical convergence result by not simply treating this pair of variables as a whole. Instead, we still concentrate on the primary role of the variable x. This special treatment provides us with more flexible results that can be used in practice while avoiding an undesirable situation when using the original convergence analysis of the proposed algorithms (discussed in Section 2.4).

Finally, we perform numerical experiments to thoroughly investigate the behaviors of the proposed methods. Nonnegative principal component analysis, group variable selection in multiclass logistic regression, and sparse linear regression are the real-world problems under consideration.

Comparison with related works In contrast to [72, 73], we apply the SVRG and SAGA estimators in the construction of convex subproblems, whereas [72, 73] used the SAG estimator. Because of its averaging feature, the SAG appears to be quite "conservative": at each iteration, it computes the average of b new gradients and $N - b$ kept gradients (b is the minibatch size and N is the number of functions in the large sum) to form a stochastic direction used to construct a subproblem. As a result, if b is small in proportion to N, the old information dominates the new information, causing this stochastic direction to shift slowly from iteration to iteration. In the convex optimization scenario, numerical tests in the publication [31] have shown that SVRG and SAGA have some advantages over SAG. As a result, SVRG and SAGA are likely to maintain their advantages in the context of DC programming and DCA.

On the other hand, our proposed algorithms are considerably different from [134]: their algorithms can be regarded as the deterministic DCA in which stochastic convex solvers are utilized to solve convex subproblems, whereas our techniques stochastically build convex subproblems. Furthermore, in their study, the authors used the SVRG algorithm to solve their subproblem, resulting in the SSDC-SVRG algorithm. Our DCA-SVRG distinguishes itself from the SSDC-SVRG by the fact that the SVRG estimator is utilized as an outer technique to compute stochastic gradients in the DCA-SVRG, whereas the SVRG is used as an inner convex solver in the SSDC-SVRG.

Concerning the proximal-SVRG and the prox-SAGA in [58] for solving the (ERM) problem where each $f(\cdot, \xi_i)$ is L-smooth and Ω is convex, which is a special case of (P): the DCA-SVRG via the option *with replacement* applied to this particular problem with G, h_i, r_1, r_2 defined above recovers the prox-SVRG. Meanwhile, the DCA-SAGA via the option *with replacement* does not correspond to the prox-SAGA, but rather a related algorithm, where the "stochastic variance-reduced gradient" of prox-SAGA, $\tilde{\nabla}_{\text{prox-SAGA}}$, is drifted by another unbiased variance-reduction term: $\tilde{\nabla}_{\text{DCA-SAGA}} = \tilde{\nabla}_{\text{prox-SAGA}} - \text{drift}$,

where

$$\text{drift} = \frac{L}{N}\sum_{i=1}^{N}\alpha_i^t - \frac{L}{b}\sum_{i\in I}\alpha_i^t,$$

with I being a set of random indexes. Furthermore, in the prox-SAGA step of updating the table of gradients, the authors do not update directly on I. Instead, they draw another independent set of indexes J, which mostly serves to facilitate their theoretical analysis. This procedure additionally consumes - in the worst scenario - b gradients computation, where b is the minibatch size. In contrast, the DCA-SAGA does not need to employ another set J, rather, we just update the table of gradients using I directly, omitting the extra computations.

2.2 Control variates

We begin with an introduction to control variates, a fundamental concept underlying recent advances in variance reduction techniques in machine learning. The following discussion is a variant of the discussion in [31]. The method of control variates [64, 65] is a variance reduction technique that is widely used in Monte Carlo simulation. Let μ be an unknown parameter of interest and X be an unbiased estimator of μ, $\mathbb{E}(X) = \mu$. In order for X to be a good estimator of μ, the variance of X, $\mathrm{Var}(X)$, should be small. Suppose there is a random variable Y with *known* mean $\mathbb{E}(Y) = \tau$. Consider a random variable defined as follows

$$Z = X + c(\tau - Y),$$

where $c > 0$ is some constant. It is observed that Z is an unbiased estimator of μ since $\mathbb{E}(Z) = \mathbb{E}(X) = \mu$. The variance of Z is given by

$$\mathrm{Var}(Z) = \mathrm{Var}(X) + c^2\,\mathrm{Var}(Y) - 2c\,\mathrm{Cov}(X, Y). \tag{2.1}$$

Hence, if X and Y are highly correlated in the sense that either $2\,\mathrm{Cov}(X, Y) > \mathrm{Var}(Y)$ (then $\mathrm{Var}(Z) < \mathrm{Var}(X)$ for $c \leq 1$) or one can choose small c such that $c^2\,\mathrm{Var}(Y) < 2c\,\mathrm{Cov}(X, Y)$ (note that c^2 is one degree of smallness away from c, so this inequality is possible when $\mathrm{Cov}(X, Y) > 0$), all of which lead to $\mathrm{Var}(Z) < \mathrm{Var}(X)$.

When X and Y have positive correlation, as a well-known fact, the optimal value of c minimizing the variance of Z is (the minimizer of the parabola (2.1))

$$c^* = \frac{\mathrm{Cov}(X, Y)}{\mathrm{Var}(Y)}.$$

With such a value of c,

$$\mathrm{Var}(Z) = \mathrm{Var}(X) - \frac{\mathrm{Cov}(X, Y)^2}{\mathrm{Var}(Y)} = \big(1 - \mathrm{Corr}(X, Y)^2\big)\,\mathrm{Var}(X) < \mathrm{Var}(X),$$

where $\mathrm{Corr}(X, Y)$ is the correlation between X and Y,

$$\mathrm{Corr}(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)\,\mathrm{Var}(Y)}}.$$

However, such a perfect c^* is hard to compute in practice. Therefore, the best scenario is that X, Y are highly correlated, and then Z serves as a variance-reduced unbiased estimator for μ even when c is not very carefully chosen.

In machine learning, the quantity of interest is often the gradient of a large-sum objective function, say, at some point x^k,

$$\mu := \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x^k).$$

A natural estimator for μ is $X = \frac{1}{b}\sum_{i\in I} \nabla f_i(x^k)$, where I is a set of b indexes randomly drawn from $\{1, 2, \ldots, N\}$. Now, let $\tilde{x}$ be point near to x^k, the random variable $Y := \frac{1}{b}\sum_{i\in I} \nabla f_i(\tilde{x})$ is expected to be highly correlated with X since x^k and $\tilde{x}$ are not too far from each other so they can share some common geometry of the objective. Therefore, the following unbiased estimator

$$Z := \frac{1}{b}\sum_{i\in I} \nabla f_i(x^k) - \frac{1}{b}\sum_{i\in I} \nabla f_i(\tilde{x}) + \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\tilde{x})$$

is expected to have smaller variance than that of X. The question is that what is the point of using Z when we have to compute the expensive full gradient $N^{-1}\sum_{i=1}^{N} \nabla f_i(\tilde{x})$? The point is that once $N^{-1}\sum_{i=1}^{N} \nabla f_i(\tilde{x})$ is computed, it can be used in the update of several x^k, in which only $2b$ gradient computations are needed for each update. Then, $\tilde{x}$ is updated, from time to time, accordingly to the progress of the sequence $\{x^k\}$ so as to make sure it is relevant in terms of variance reduction. This leads to the creation of a technique known as stochastic variance reduction gradient (SVRG) [59].

Another technique named SAGA [31] for using control variates in computing μ is to utilize a table of references $\{\alpha_1^k, \alpha_2^k, \ldots, \alpha_N^k\}$ where each α_i^k is among the set $\{x^0, x^1, \ldots, x^{k-1}\}$. The random variable $Y := b^{-1}\sum_{i\in I} \nabla f_i(\alpha_i^k)$ is expected to be highly correlated with X, so the following unbiased estimator

$$Z := \frac{1}{b}\sum_{i\in I} \nabla f_i(x^k) - \frac{1}{b}\sum_{i\in I} \nabla f_i(\alpha_i^k) + \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\alpha_i^k)$$

is expected to have small variance. The key is that one always keeps the table of gradients $\{\nabla f_i(\alpha_i^k)\}_{i=1}^N$ in the memory so that $N^{-1}\sum_{i=1}^{N} \nabla f_i(\alpha_i^k)$ is ready to use, and the table is updated progressively to ensure the high correlation between X and Y.

2.3 Stochastic DCA with Variance Reduction

Throughout this section, we study the problem (P) with r_2 being a convex function.

2.3.1 The first stochastic DCA: DCA-SVRG

In this part, we design the first stochastic DCA, DCA-SVRG, to solve the problem (P). The approach can be thought of as a combination of deterministic DCA and SVRG-style gradient updating. That is, in the deterministic DCA, we substitute the gradient of H with the "stochastic variance-reduced gradient" of H. The technique is epoch-based, with the full gradient of H computed at the start of each epoch and used as a variance-reduction term inside that epoch.

To build the stochastic variance reduction gradient of H, we can sample in one of two ways: with or without replacement. If sampling is with replacement, a set of indexes $I_b = \{i_1, i_2, \ldots, i_b\}$ is, at each iteration, randomly chosen from $\{1, 2, \ldots, N\}$ where each i_j is independent of the others. Otherwise, the repetition in I_b is not permitted.

Algorithm 1 DCA-SVRG

Initialization: $\tilde{x}^0 \in \operatorname{dom} r_1$, inner-loop length M, minibatch size b, $k = 0$, option (either *with replacement* or *without replacement*).
repeat
  Compute the full gradient $\tilde{\nu}^k = \frac{1}{N}\sum_{i=1}^{N} \nabla h_i(\tilde{x}^k)$ and set $x_0^{k+1} = \tilde{x}^k$.
  for $j = 0 : M-1$ **do**
    if option is *with replacement* **then**
      Randomly choose with replacement the set I_b of b elements of $[N]$.
    else
      Randomly choose without replacement the set I_b of b elements of $[N]$.
    end if
    Compute the "stochastic variance reduced gradient" t_j^{k+1} by

$$t_j^{k+1} = \frac{1}{b}\sum_{i\in I_b} \nabla h_i(x_j^{k+1}) + \tilde{\nu}^k - \frac{1}{b}\sum_{i\in I_b} \nabla h_i(\tilde{x}^k).$$

    Compute $y_j^{k+1} \in \partial r_2(x_j^{k+1})$ and let $z_j^{k+1} = t_j^{k+1} + y_j^{k+1}$.
    Solve the convex problem $x_{j+1}^{k+1} \in \arg\min\{G(x) + r_1(x) - \langle z_j^{k+1}, x\rangle : x \in \mathbb{R}^n\}$.
  end for
  Set $\tilde{x}^{k+1} = x_M^{k+1}$, and $k = k+1$.
until Stopping criterion.

On the other hand, since a critical point x^* of $F = (G + r_1) - (H + r_2)$ satisfies $\operatorname{dist}(\partial(G + r_1)(x^*), \partial(H + r_2)(x^*)) = 0$, we define the measure of proximity to criticality as follows

$$d_K = \min_{k=0,1,\ldots,K-1, j=0,1,\ldots,M-1} \mathbb{E}\operatorname{dist}(\partial(H + r_2)(x_{j+1}^{k+1}), \partial(G + r_1)(x_{j+1}^{k+1})).$$

For brevity, we denote $\bar{x}^k = \{x_0^k, x_1^k, \ldots, x_{M-1}^k\}$ and $\bar{y}^k = \{y_0^k, y_1^k, \ldots, y_{M-1}^k\}$ for all $k \in \mathbb{N}^*$. Let $\mathcal{P}_j^{k+1}$ be the σ-algebra defined as

$$\mathcal{P}_j^{k+1} = \sigma(\bar{x}^1, \ldots, \bar{x}^k, x_0^{k+1}, x_1^{k+1}, \ldots, x_j^{k+1}, \bar{y}^1, \ldots, \bar{y}^k, y_0^{k+1}, y_1^{k+1}, \ldots, y_j^{k+1}).$$

With a suitable selection of the minibatch size b and the inner-loop length M, we derive the following convergence results.

Theorem 6. *If the minibatch size b and the inner-loop length M satisfy*

$$\frac{M}{\sqrt{b}} \leq \frac{1}{4\sqrt{e-1}} \frac{\rho_{G+r_1} + \rho_H + \rho_{r_2}}{L},$$

then

1. *The sequence $\{F(\tilde{x}^k)\}$ converges almost surely.*

2. *$\sum_{k=0}^{\infty} \sum_{j=0}^{M-1} \mathbb{E}\|x_{j+1}^{k+1} - x_j^{k+1}\|^2 < +\infty$.*

3. *Suppose the sequence $\{y_j^k\}$ is bounded almost surely, then every limit point of $\{x_j^k\}$ is a critical point of $F = (G + r_1) - (H + r_2)$ almost surely.*

4. *If r_2 has Lipschitz continuous gradient over $\operatorname{dom} r_1$, then $d_K = O(1/\sqrt{K})$. Moreover, by choosing $b = \lfloor N^{2/3} \rfloor$, $M = \lfloor \mu\sqrt{b} \rfloor$, where*

$$\mu = \frac{1}{4\sqrt{e-1}} \frac{\rho_{G+r_1} + \rho_H + \rho_{r_2}}{L},$$

   the complexity in terms of the number of gradient evaluations to obtain ϵ-criticality in expectation is $O(N^{2/3}/\epsilon^2)$; Meanwhile, the complexity in terms of the number of convex subproblems being solved is $O(1/\epsilon^2)$.

Because the proofs for the two sampling choices largely overlap, we shall prove Theorem 6 concurrently for both options with and without replacement. We will emphasize parts of the proof that must be studied differently for each option. Before proving Theorem 6, we first introduce the following lemma for the option sampling without replacement.

Lemma 1. *For the option of sampling without replacement, the following inequality holds*

$$\mathbb{E}(\|t_j^{k+1} - \nabla H(x_j^{k+1})\|^2 | \mathcal{P}_j^{k+1}) \leq \frac{L^2}{b}\left(1 - \frac{b-1}{N-1}\right)\|x_j^{k+1} - \tilde{x}^k\|^2 \leq \frac{L^2}{b}\|x_j^{k+1} - \tilde{x}^k\|^2.$$

Proof of Lemma 1. We have,

$$\begin{aligned}
&\mathbb{E}\left(\left\|\frac{1}{b}\sum_{i\in I_b}\nabla h_i(x_j^{k+1}) - \frac{1}{b}\sum_{i\in I_b}\nabla h_i(\tilde{x}^k) + \nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\right\|^2 | \mathcal{P}_j^{k+1}\right) \\
&= \mathbb{E}_{I_b}\left\|\frac{1}{b}\sum_{i\in I_b}\nabla h_i(x_j^{k+1}) - \frac{1}{b}\sum_{i\in I_b}\nabla h_i(\tilde{x}^k) + \nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\right\|^2 \\
&= \sum_{I\subset[N],|I|=b} \mathbb{P}(I_b = I)\left\|\frac{1}{b}\sum_{i\in I}\nabla h_i(x_j^{k+1}) - \frac{1}{b}\sum_{i\in I}\nabla h_i(\tilde{x}^k) + \nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\right\|^2,
\end{aligned}$$

where the probability $\mathbb{P}(I_b = I) = 1/C_N^b$.

For a fixed I, we compute

$$\begin{aligned}R &= \left\|\frac{1}{b}\sum_{i\in I}\nabla h_i(x_j^{k+1}) - \frac{1}{b}\sum_{i\in I}\nabla h_i(\tilde{x}^k) + \nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\right\|^2\\ &= \frac{1}{b^2}\sum_{i\in I}\|\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k) + \nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\|^2\\ &+ \frac{1}{b^2}\sum_{i\neq l,i,l\in I}\langle\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k) + \nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1}),\\ &\qquad \nabla h_l(x_j^{k+1}) - \nabla h_l(\tilde{x}^k) + \nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\rangle := \frac{1}{b^2}(R_1 + R_2).\end{aligned}$$

$$\begin{aligned}R_1 &= \sum_{i\in I}\|\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k)\|^2 + b\|\nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\|^2\\ &+ 2\langle\nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1}), \sum_{i\in I}(\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k))\rangle,\\ R_2 &= A_b^2\|\nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\|^2\\ &+ \langle\nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1}), \sum_{i\neq l,i,l\in I}(\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k) + \nabla h_l(x_j^{k+1}) - \nabla h_l(\tilde{x}^k))\rangle\\ &+ \sum_{i\neq l,i,l\in I}\langle\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k), \nabla h_l(x_j^{k+1}) - \nabla h_l(\tilde{x}^k)\rangle.\end{aligned}$$

Therefore,

$$\begin{aligned}R =&\frac{1}{b^2}\sum_{i\in I}\|\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k)\|^2 + \frac{2}{b}\langle\nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1}), \sum_{i\in I}(\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k))\rangle\\ &+ \|\nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\|^2 + \frac{1}{b^2}\sum_{i\neq l,i,l\in I}\langle\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k), \nabla h_l(x_j^{k+1}) - \nabla h_l(\tilde{x}^k)\rangle.\end{aligned}$$

On the other hand, we have the following computations

$$\begin{aligned}&\sum_{I\subset[N],|I|=b}\sum_{i\in I}\|\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k)\|^2 = C_{N-1}^{b-1}\sum_{i=1}^{N}\|\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k)\|^2,\\ &\sum_{I\subset[N],|I|=b}\langle\nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1}), \sum_{i\in I}(\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k))\rangle\\ &= -C_{N-1}^{b-1}\cdot N\cdot\|\nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\|^2,\\ &\sum_{I\subset[N],|I|=b}\sum_{i\neq l,i,l\in I}\langle\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k), \nabla h_l(x_j^{k+1}) - \nabla h_l(\tilde{x}^k)\rangle\\ &= C_{N-2}^{b-2}\sum_{i\neq j}\langle\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k), \nabla h_l(x_j^{k+1}) - \nabla h_l(\tilde{x}^k)\rangle.\end{aligned}$$

Therefore,

$$\begin{aligned}
&\mathbb{E}_{I_b}\left\|\frac{1}{b}\sum_{i\in I_b}\nabla h_i(x_j^{k+1}) - \frac{1}{b}\sum_{i\in I_b}\nabla h_i(\tilde{x}^k) + \nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\right\|^2 \\
&= \frac{1}{bN}\sum_{i=1}^{N}\|\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k)\|^2 - \|\nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\|^2 \\
&+ \frac{b-1}{bN(N-1)}\sum_{i\neq l}\langle \nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k), \nabla h_l(x_j^{k+1}) - \nabla h_l(\tilde{x}^k)\rangle \\
&= \frac{1}{bN}\left(1 - \frac{b-1}{N-1}\right)\sum_{i=1}^{N}\|\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k)\|^2 + \left(\frac{N(b-1)}{b(N-1)} - 1\right)\|\nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\|^2 \\
&\leq \frac{L^2}{b}\left(1 - \frac{b-1}{N-1}\right)\|x_j^{k+1} - \tilde{x}^k\|^2.
\end{aligned}$$

□

Now we prove Theorem 6 as follows.

Proof of Theorem 6. 1. Since H is ρ_H-convex and by the definition of x_{j+1}^{k+1},

$$\begin{aligned}
H(x_{j+1}^{k+1}) &\geq H(x_j^{k+1}) + \langle \nabla H(x_j^{k+1}), x_{j+1}^{k+1} - x_j^{k+1}\rangle + \frac{\rho_H}{2}\|x_{j+1}^{k+1} - x_j^{k+1}\|^2, \\
r_2(x_{j+1}^{k+1}) &\geq r_2(x_j^{k+1}) + \langle y_j^{k+1}, x_{j+1}^{k+1} - x_j^{k+1}\rangle + \frac{\rho_{r_2}}{2}\|x_{j+1}^{k+1} - x_j^{k+1}\|^2.
\end{aligned}$$

It follows from the definition of x_{j+1}^{k+1} that

$$(G + r_1)(x_j^{k+1}) \geq (G + r_1)(x_{j+1}^{k+1}) + \langle z_j^{k+1}, x_j^{k+1} - x_{j+1}^{k+1}\rangle + \frac{\rho_{G+r_1}}{2}\|x_j^{k+1} - x_{j+1}^{k+1}\|^2.$$

These inequalities imply

$$F(x_{j+1}^{k+1}) \leq F(x_j^{k+1}) + \langle x_{j+1}^{k+1} - x_j^{k+1}, t_j^{k+1} - \nabla H(x_j^{k+1})\rangle - \frac{\rho}{2}\|x_{j+1}^{k+1} - x_j^{k+1}\|^2,$$

where $\rho = \rho_H + \rho_{r_2} + \rho_{G+r_1}$. Let $\gamma > 0$ that will be determined later. By applying Schwartz inequality and AM-GM inequality,

$$F(x_{j+1}^{k+1}) \leq F(x_j^{k+1}) + \frac{1}{2\gamma}\|t_j^{k+1} - \nabla H(x_j^{k+1})\|^2 - \frac{\rho - \gamma}{2}\|x_{j+1}^{k+1} - x_j^{k+1}\|^2.$$

By taking conditional expectation with respect to $\mathcal{P}_j^{k+1}$,

$$\begin{aligned}
\mathbb{E}(F(x_{j+1}^{k+1})|\mathcal{P}_j^{k+1}) \leq F(x_j^{k+1}) &- \frac{\rho - \gamma}{2}\mathbb{E}(\|x_{j+1}^{k+1} - x_j^{k+1}\|^2|\mathcal{P}_j^{k+1}) \\
&+ \frac{1}{2\gamma}\mathbb{E}(\|t_j^{k+1} - \nabla H(x_j^{k+1})\|^2|\mathcal{P}_j^{k+1}).
\end{aligned}$$

If option is sampling with replacement, the set I_b consists of independent indexes, we therefore evaluate

$\mathbb{E}(\|t_j^{k+1} - \nabla H(x_j^{k+1})\|^2|\mathcal{P}_j^{k+1})$ as follows,

$$\begin{aligned}
&\mathbb{E}(\|t_j^{k+1} - \nabla H(x_j^{k+1})\|^2|\mathcal{P}_j^{k+1}) \\
&= \mathbb{E}_{I_b}\left(\left\|\frac{1}{b}\sum_{i\in I_b}\nabla h_i(x_j^{k+1}) - \frac{1}{b}\sum_{i\in I_b}\nabla h_i(\tilde{x}^k) + \nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\right\|^2\right) \\
&= \frac{1}{b}\mathbb{E}_i\left(\left\|\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k) + \nabla H(\tilde{x}^k) - \nabla H(x_j^{k+1})\right\|^2\right), \text{ where } i \overset{\text{uni}}{\sim} [N] \\
&\leq \frac{1}{b}\mathbb{E}_i\|\nabla h_i(x_j^{k+1}) - \nabla h_i(\tilde{x}^k)\|^2 \leq \frac{L^2}{b}\|x_j^{k+1} - \tilde{x}^k\|^2. \qquad (2.2)
\end{aligned}$$

Together with Lemma 1, the following inequality holds for two options

$$\mathbb{E}(\|t_j^{k+1} - \nabla H(x_j^{k+1})\|^2|\mathcal{P}_j^{k+1}) \leq \frac{L^2}{b}\|x_j^{k+1} - \tilde{x}^k\|^2. \qquad (2.3)$$

Therefore,

$$\mathbb{E}(F(x_{j+1}^{k+1})|\mathcal{P}_j^{k+1}) \leq F(x_j^{k+1}) + \frac{L^2}{2b\gamma}\|x_j^{k+1} - \tilde{x}^k\|^2 - \frac{\rho-\gamma}{2}\mathbb{E}(\|x_{j+1}^{k+1} - x_j^{k+1}\|^2|\mathcal{P}_j^{k+1}).$$

Consider the sequence of Lyapunov functions $V_j^{k+1} = F(x_j^{k+1}) + c_j\|x_j^{k+1} - \tilde{x}^k\|^2$, where $\{c_j\}$ are non-negative numbers determined later (the idea of such sequence of Lyapunov functions is adopted from [58]). We have

$$\begin{aligned}
&\mathbb{E}(V_{j+1}^{k+1}|\mathcal{P}_j^{k+1}) = \mathbb{E}(F(x_{j+1}^{k+1}) + c_{j+1}\|x_{j+1}^{k+1} - \tilde{x}^k\|^2|\mathcal{P}_j^{k+1}) \\
&\leq \mathbb{E}(F(x_{j+1}^{k+1}) + c_{j+1}(1+\beta)\|x_{j+1}^{k+1} - x_j^{k+1}\|^2 \\
&\quad + c_{j+1}(1+1/\beta)\|x_j^{k+1} - \tilde{x}^k\|^2|\mathcal{P}_j^{k+1}), \text{ where } \beta > 0 \\
&\leq F(x_j^{k+1}) + \left(\frac{L^2}{2b\gamma} + c_{j+1}\left(1 + \frac{1}{\beta}\right)\right)\|x_j^{k+1} - \tilde{x}^k\|^2 \\
&\quad + \left(c_{j+1}(1+\beta) - \frac{\rho-\gamma}{2}\right)\mathbb{E}(\|x_{j+1}^{k+1} - x_j^{k+1}\|^2|\mathcal{P}_j^{k+1}). \qquad (2.4)
\end{aligned}$$

To obtain V_j^{k+1} in the right-hand side of (2.4), we choose the sequence $\{c_j\}$ in such a way $c_M = 0$ and $c_j = \frac{L^2}{2b\gamma} + c_{j+1}\left(1 + \frac{1}{\beta}\right)$ if $j < M$. These relations yield $c_j = \frac{\beta L^2}{2b\gamma}\left(\left(1 + \frac{1}{\beta}\right)^{M-j} - 1\right)$. Next, to achieve the descent property on the Lyapunov sequence, i.e. $\mathbb{E}(V_{j+1}^{k+1}|\mathcal{P}_j^{k+1}) < V_j^{k+1}$, we want $c_{j+1}(1+\beta) + \frac{\gamma}{2} \leq \frac{\rho}{4}, \forall j = \overline{0, M-1}$, or equivalently,

$$(1+\beta)\frac{\beta L^2}{2b\gamma}\left(\left(1 + \frac{1}{\beta}\right)^{M-j-1} - 1\right) + \frac{\gamma}{2} \leq \frac{\rho}{4}, \forall j = \overline{0, M-1},$$

which is further equivalent to

$$(1+\beta)\frac{\beta L^2}{2b\gamma}\left(\left(1 + \frac{1}{\beta}\right)^{M-1} - 1\right) + \frac{\gamma}{2} \leq \frac{\rho}{4}. \qquad (2.5)$$

By choosing $\beta = M-1$, and noticing that $\left(1+\frac{1}{M-1}\right)^{M-1} \leq e$ and $(1+\beta)\beta < (1+\beta)^2 = M^2$, (2.5) holds if the following stronger inequality holds

$$\frac{M^2L^2}{2b\gamma}(e-1) + \frac{\gamma}{2} \leq \frac{\rho}{4}. \tag{2.6}$$

Now by choosing $\gamma = \frac{ML\sqrt{e-1}}{\sqrt{b}}$ to optimize the LHS of (2.6), the inequality (2.6) becomes $\frac{M}{\sqrt{b}} \leq \frac{\rho}{4L\sqrt{e-1}}$. Consequently, if the minibatch size b and the inner-loop length M satisfy this inequality, together with (2.4) we get

$$\mathbb{E}(V_{j+1}^{k+1}|\mathcal{P}_j^{k+1}) \leq V_j^{k+1} - \frac{\rho}{4}\mathbb{E}(\|x_{j+1}^{k+1} - x_j^{k+1}\|^2|\mathcal{P}_j^{k+1}). \tag{2.7}$$

From the definitions of V_j^{k+1} and $\mathcal{P}_j^{k+1}$, it is observed that V_j^{k+1} is $\mathcal{P}_j^{k+1}$-measurable, for $k \in \mathbb{N}, j \in \{0,\ldots,M-1\}$. Moreover, it can be verified that $\mathcal{P}_{M-1}^{k+1} \subset \mathcal{P}_0^{k+2}, \forall k \in \mathbb{N}$. Therefore, we have the following filtration

$$\underbrace{\mathcal{P}_0^1}_{\mathcal{Q}_0} \subset \underbrace{\mathcal{P}_1^1}_{\mathcal{Q}_1} \subset \ldots \subset \underbrace{\mathcal{P}_{M-1}^1}_{\mathcal{Q}_{M-1}} \subset \underbrace{\mathcal{P}_0^2}_{\mathcal{Q}_M} \subset \underbrace{\mathcal{P}_1^2}_{\mathcal{Q}_{M+1}} \subset \ldots,$$

in which the following sequence is an adapted process:

$$\underbrace{V_0^1}_{W_0}, \underbrace{V_1^1}_{W_1}, \ldots, \underbrace{V_{M-1}^1}_{W_{M-1}}, \underbrace{V_0^2}_{W_M}, \underbrace{V_1^2}_{W_{M+1}}, \ldots$$

On the other hand, it is observed that $V_M^{k+1} = V_0^{k+2}, \forall k \in \mathbb{N}$. So by plugging $j = M-1$ to (3.15), we obtain

$$\mathbb{E}(V_0^{k+2}|\mathcal{P}_{M-1}^{k+1}) \leq V_{M-1}^{k+1} - \frac{\rho}{4}\mathbb{E}(\|x_M^{k+1} - x_{M-1}^{k+1}\|^2|\mathcal{P}_{M-1}^{k+1}). \tag{2.8}$$

From (3.15), (2.8), and the fact that $V_j^{k+1} \geq F(x_j^{k+1}) \geq \alpha, \forall k \in \mathbb{N}, j \in \{0,1,\ldots,M-1\}$, we can apply the supermartingale convergence theorem [8]: there exists a random variable V^∞ such that the sequence $\{W_k\}$ converges to V^∞ almost surely. As a consequence, the sequence $\{F(\tilde{x}^k)\}$ converges to V^∞ almost surely since $V_0^{k+1} = F(x_0^{k+1}) = F(\tilde{x}^k)$.

2. From (3.15), we obtain $\frac{\rho}{4}\mathbb{E}\|x_{j+1}^{k+1} - x_j^{k+1}\|^2 + \mathbb{E}(V_{j+1}^{k+1}) \leq \mathbb{E}(V_j^{k+1})$. From this inequality and the fact that $\mathbb{E}(V_0^1) = \mathbb{E}(F(\tilde{x}^0)) = F(\tilde{x}^0) < +\infty$, by induction, one obtains $\mathbb{E}(V_j^{k+1}) < +\infty$, for all k, j. By telescoping with j and k,

$$\frac{\rho}{4}\sum_{k=0}^{\infty}\sum_{j=0}^{M-1}\mathbb{E}\|x_{j+1}^{k+1} - x_j^{k+1}\|^2 \leq \mathbb{E}(V_0^1) - \alpha < +\infty. \tag{2.9}$$

3. From the evaluation (2.3), we obtain

$$\mathbb{E}\left\|t_j^{k+1} - \nabla H(x_j^{k+1})\right\|^2 \leq \frac{L^2(M-1)}{b}\left(\mathbb{E}\|x_1^{k+1} - x_0^{k+1}\|^2 + \ldots + \mathbb{E}\|x_{M-1}^{k+1} - x_{M-2}^{k+1}\|^2\right),$$

which implies

$$\sum_{k=0}^{\infty}\sum_{j=0}^{M-1}\mathbb{E}\left\|t_j^{k+1}-\nabla H(x_j^{k+1})\right\|^2<+\infty. \tag{2.10}$$

Let $S_1=\left(t_j^{k+1}-\nabla H(x_j^{k+1})\to 0,\forall j=\overline{0,M-1}\right)$. It follows from (2.10) that $\mathbb{P}(S_1)=1$. On the other hand, let $S_2=\left(x_{j+1}^{k+1}-x_j^{k+1}\to 0,\forall j=\overline{0,M-1}\right)$, it follows from (2.9) that $\mathbb{P}(S_2)=1$. Furthermore, $\mathbb{P}(S_3)=1$, where $S_3=(y_j^k$ is bounded, $\forall j=\overline{1,M-1})$.

Now, consider an event in $S_1\cap S_2\cap S_3$ which gives rise to a realization $\{x_j^k\}$ (here, $\{x_j^k\}$ is a sequence of real numbers rather than a sequence of random variables). Let x^* be a limit point of $\{x_j^k\}$. Note that the sequence $\{x_j^k\}$ generated by the algorithm is expressed explicitly in the following order

$$x_0^1,x_1^1,\ldots,x_{M-1}^1,x_M^1\equiv x_0^2,x_1^2,\ldots,x_{M-1}^2,x_M^2\equiv x_0^3,x_1^3,\ldots$$

We observe that a subsequence of $\{x_j^k\}$ has the following form $\{x_{v(l)}^{u(l)}\}_{l\in\mathbb{N}}$, where $u:\mathbb{N}\to\mathbb{N}^*$ and $v:\mathbb{N}\to\{0,1,\ldots,M-1\}$ satisfy the following relation: for every $i<t$ in $\mathbb{N}$, either $u(i)<u(t)$ or $u(i)=u(t)$ and $v(i)<v(t)$. Since x^* is a limit point of $\{x_j^k\}$, there exists a subsequence $\{x_{v(l)}^{u(l)}\}$ converging to x^* as $l\to\infty$, which implies $x_{v(l)+1}^{u(l)}\to x^*$. As a consequence, $t_{v(l)}^{u(l)}\to\nabla H(x^*)$. By passing to a subsequence if necessary, we assume that $y_{v(l)}^{u(l)}$ converges to y^*. Since $y_{v(l)}^{u(l)}\in\partial r_2\left(x_{v(l)}^{u(l)}\right)$, we obtain $y^*\in\partial r_2(x^*)$ thanks to the closedness of the graph of the subdifferential operator.

By the definition of $x_{v(l)+1}^{u(l)}$, we obtain $z_{v(l)}^{u(l)}\in\partial(G+r_1)\left(x_{v(l)+1}^{u(l)}\right)$. Since the graph of the subdifferential operator is closed, by letting $l\to\infty$ we derive $\nabla H(x^*)+y^*\in\partial(G+r_1)(x^*)$, hence x^* is a critical point of $F=(G+r_1)-(H+r_2)$.

4. We first evaluate $\mathbb{E}\,\mathrm{dist}(\nabla H(x_{j+1}^{k+1})+\nabla r_2(x_{j+1}^{k+1}),\partial(G+r_1)(x_{j+1}^{k+1}))$. Since $z_j^{k+1}\in\partial(G+r_1)(x_{j+1}^{k+1})$, we obtain

$$\begin{aligned}
&\mathbb{E}\,\mathrm{dist}(\nabla H(x_{j+1}^{k+1})+\nabla r_2(x_{j+1}^{k+1}),\partial(G+r_1)(x_{j+1}^{k+1}))\\
&\le\mathbb{E}\|\nabla H(x_{j+1}^{k+1})-t_j^{k+1}\|+\mathbb{E}\|\nabla r_2(x_{j+1}^{k+1})-\nabla r_2(x_j^{k+1})\|\\
&\le\mathbb{E}\|\nabla H(x_{j+1}^{k+1})-\nabla H(x_j^{k+1})\|+\mathbb{E}\|\nabla H(x_j^{k+1})-t_j^{k+1}\|+\mathbb{E}\|\nabla r_2(x_{j+1}^{k+1})-\nabla r_2(x_j^{k+1})\|\\
&\le(L+L_{r_2})\mathbb{E}\|x_{j+1}^{k+1}-x_j^{k+1}\|+\left(\mathbb{E}\|t_j^{k+1}-\nabla H(x_j^{k+1})\|^2\right)^{\frac{1}{2}}\\
&\le(L+L_{r_2})\mathbb{E}\|x_{j+1}^{k+1}-x_j^{k+1}\|+\frac{L}{\sqrt{b}}\left(\mathbb{E}\|x_j^{k+1}-\tilde{x}^k\|^2\right)^{\frac{1}{2}}.
\end{aligned}\tag{2.11}$$

Furthermore, we have

$$\begin{aligned}
&\|x_j^{k+1}-\tilde{x}^k\|^2=\|x_j^{k+1}-x_0^{k+1}\|^2\\
&\le(M-j+M-j-1+\ldots+M-1)\times\\
&\quad\left(\frac{1}{M-j}\|x_j^{k+1}-x_{j-1}^{k+1}\|^2+\ldots+\frac{1}{M-1}\|x_1^{k+1}-x_0^{k+1}\|^2\right).
\end{aligned}$$

Therefore,

$$\sum_{j=1}^{M-1}\left(\mathbb{E}\|x_j^{k+1}-\tilde{x}^k\|^2\right)^{\frac{1}{2}} \leq \sum_{j=1}^{M-1}\left(\sum_{r=1}^{j}(M-r)\right)^{\frac{1}{2}}\left(\sum_{r=1}^{j}\frac{1}{M-r}\mathbb{E}\|x_r^{k+1}-x_{r-1}^{k+1}\|^2\right)^{\frac{1}{2}}$$
$$\leq \left(\sum_{j=1}^{M-1}\sum_{r=1}^{j}(M-r)\right)^{\frac{1}{2}}\left(\sum_{j=1}^{M-1}\sum_{r=1}^{j}\frac{1}{M-r}\mathbb{E}\|x_r^{k+1}-x_{r-1}^{k+1}\|^2\right)^{\frac{1}{2}}$$
$$= \left(\frac{(M-1)M(2M-1)}{6}\right)^{\frac{1}{2}}\left(\sum_{r=1}^{M-1}\mathbb{E}\|x_r^{k+1}-x_{r-1}^{k+1}\|^2\right)^{\frac{1}{2}}. \tag{2.12}$$

It follows from (2.11) and (2.12) that

$$\sum_{k=0}^{K-1}\sum_{j=0}^{M-1}\mathbb{E}\,\mathrm{dist}(\nabla H(x_{j+1}^{k+1})+\nabla r_2(x_{j+1}^{k+1}),\partial(G+r_1)(x_{j+1}^{k+1})) \leq (L+L_{r_2})\sum_{k=0}^{K-1}\sum_{j=0}^{M-1}\mathbb{E}\|x_{j+1}^{k+1}-x_j^{k+1}\|$$
$$+\frac{L}{\sqrt{b}}\left(\frac{(M-1)M(2M-1)}{6}\right)^{\frac{1}{2}}\sum_{k=0}^{K-1}\left(\sum_{r=1}^{M-1}\mathbb{E}\|x_r^{k+1}-x_{r-1}^{k+1}\|^2\right)^{\frac{1}{2}}$$
$$\leq \sqrt{K}\left((L+L_{r_2})\sqrt{M}+\frac{L}{\sqrt{b}}\left(\frac{(M-1)M(2M-1)}{6}\right)^{\frac{1}{2}}\right)\left(\sum_{k=0}^{K-1}\sum_{j=0}^{M-1}\mathbb{E}\|x_{j+1}^{k+1}-x_j^{k+1}\|^2\right)^{\frac{1}{2}}.$$

As a consequence, the iteration convergence rate is given by $d_K = O(1/\sqrt{K})$.

Now we derive the complexity to find ϵ-criticality as follows. After K iterations, the output x_a is chosen uniformly from $\{x_{j+1}^{k+1}\}_{j=0,\ldots,M-1}^{k=0,\ldots,K-1}$, it holds, for some $C>0$,

$$KM\cdot\mathbb{E}\,\mathrm{dist}(\nabla H(x_a)+\nabla r_2(x_a),\partial(G+r_1)(x_a))$$
$$\leq \sqrt{K}\left((L+L_{r_2})\sqrt{M}+\frac{L}{\sqrt{b}}\left(\frac{(M-1)M(2M-1)}{6}\right)^{\frac{1}{2}}\right)C.$$

Let's choose $M=\lfloor\mu\sqrt{b}\rfloor$, where

$$\mu = \frac{1}{4\sqrt{e-1}}\frac{\rho_{G+r_1}+\rho_H+\rho_{r_2}}{L}$$

one gets

$$KM\cdot\mathbb{E}\,\mathrm{dist}(\nabla H(x_a)+\nabla r_2(x_a),\partial(G+r_1)(x_a)) \leq C\sqrt{K}\left((L+L_{r_2})\sqrt{M}+\frac{L}{2\sqrt{b}}M^{3/2}\right)$$
$$\leq C\sqrt{K}\left((L+L_{r_2})\sqrt{M}+\frac{L\mu}{2}\sqrt{M}\right) \leq C\sqrt{MK}\left(L+L_{r_2}+\frac{L\mu}{2}\right).$$

Therefore,

$$\mathbb{E}\,\mathrm{dist}(\nabla H(x_a)+\nabla r_2(x_a),\partial(G+r_1)(x_a)) = O\left(\frac{1}{\sqrt{KM}}\right).$$

By choosing $b = \lfloor N^{2/3} \rfloor$, the algorithm needs $K = O(\frac{1}{\epsilon^2 N^{1/3}})$ outer iterations to attain an ϵ-DC critical point. The total number of gradient evaluations is then:

$$K\left(N + 2bM\right) \leq K(N + 2N^{2/3}\mu N^{1/3}) = KN(1 + 2\mu) = O\left(\frac{N^{2/3}}{\epsilon^2}\right).$$

On the other hand, there are KM convex subproblems to be solved, leading to the complexity of $O(1/\epsilon^2)$ in terms of the number of convex subproblems.

□

Remark 1. *About the constraint between b and M, $\frac{M}{\sqrt{b}} \leq \frac{1}{4\sqrt{e-1}}\frac{\rho}{L}$:*

i. If the DC decomposition admits a large number $\frac{\rho}{L}$, we have great flexibility to choose b and M.

ii. If the DC problem has a very small value of $\frac{\rho}{L}$, we can adjust the DC decomposition to increase $\frac{\rho}{L}$. Indeed, to this end, we add $\frac{\gamma}{2}\|\cdot\|^2$ ($\gamma > 0$) to both DC components, $F = \left(G + r_1 + \frac{\gamma}{2}\|\cdot\|^2\right) - \left(H + r_2 + \frac{\gamma}{2}\|\cdot\|^2\right)$. With this new DC decomposition ($\bar{G} = G, \bar{H} = H, \bar{r_1} = r_1 + \frac{\gamma}{2}\|\cdot\|^2, \bar{r_2} = r_2 + \frac{\gamma}{2}\|\cdot\|^2$), the larger value is obtained since

$$\frac{\bar{\rho}}{\bar{L}} = \frac{\rho + 2\gamma}{L} \to +\infty \text{ as } \gamma \to +\infty.$$

However, adopting the above technique should be done with caution because adding a strongly convex term to both DC components increases the "gap" between the second DC component and its linear minorant, which can lead to bad approximations.

2.3.2 The second stochastic DCA: DCA-SAGA

In this part, we propose a new stochastic DCA named DCA-SAGA presented in Algorithm 2. The new algorithm, like DCA-SVRG, is a combination of deterministic DCA with stochastic gradient update in the SAGA style. That is, the deterministic DCA's gradient of H is replaced by the stochastic gradient provided by SAGA.

We denote $d_T = \min_{t=0,1,\ldots,T-1} \mathbb{E}\,\mathrm{dist}(\partial(H + r_2)(x^{t+1}), \partial(G + r_1)(x^{t+1}))$. For brevity, we denote $\bar{\alpha}^t = \{\alpha_1^t, \alpha_2^t, \ldots, \alpha_N^t\}, \forall t \in \mathbb{N}$. We denote the sequence of increasing σ-algebra $\{\mathcal{P}_t\}$ as $\mathcal{P}_t = \sigma(x^0, x^1, \ldots, x^t, y^0, y^1, \ldots, y^t, \bar{\alpha}^0, \bar{\alpha}^1, \ldots, \bar{\alpha}^t)$. The convergence results of DCA-SAGA is described as follows.

Theorem 7. *If the minibatch size b satisfies $\frac{N\sqrt{N+b}}{b^2} \leq \frac{\rho_H + \rho_{r_2} + \rho_{G+r_1}}{4L}$ for the option of sampling with replacement or $\frac{N\sqrt{N+1}}{b^2} \leq \frac{\rho_H + \rho_{r_2} + \rho_{G+r_1}}{4L}$ for the option of sampling without replacement, then*

1. $\sum_{t=0}^{\infty} \mathbb{E}\|x^{t+1} - x^t\|^2 < \infty$.
2. $\sum_{t=0}^{\infty} \sum_{i=1}^{N} \mathbb{E}\|x^t - \alpha_i^t\|^2 < \infty$.
3. *The sequence $\{F(x^t)\}$ converges almost surely.*
4. *Suppose the sequence $\{y^t\}$ is bounded almost surely. Then, every limit point of $\{x^t\}$ is a critical point of $F = (G + r_1) - (H + r_2)$ almost surely.*

Algorithm 2 DCA-SAGA

Initialization: $x^0 \in \operatorname{dom} r_1$, set $\alpha_i^0 = x^0, \forall i = \overline{1,N}, t = 0$, option (either *with replacement* or *without replacement*).
Compute the full gradient $\nu^0 = \frac{1}{N}\sum_{i=1}^N \nabla h_i(\alpha_i^0)$.
repeat
  if option is *with replacement* **then**
    Randomly choose with replacement the set I_t of b elements of $[N]$.
  else
    Randomly choose without replacement the set I_t of b elements of $[N]$.
  end if
  Compute the "stochastic variance reduced gradient" v^t,

$$v^t = \frac{1}{b}\sum_{i\in I_t} \nabla h_i(x^t) + \nu^t - \frac{1}{b}\sum_{i\in I_t} \nabla h_i(\alpha_i^t).$$

  Compute $y^t \in \partial r_2(x^t)$, and let $z^t = v^t + y^t$.
  Solve the convex program $x^{t+1} \in \arg\min\{G(x) + r_1(x) - \langle z^t, x\rangle : x \in \mathbb{R}^n\}$.
  Update $\alpha_j^{t+1} = \begin{cases} x^t, & \text{if } j \in I_t \\ \alpha_j^t, & \text{otherwise} \end{cases}$
  Update the "full gradient" $\nu^{t+1} = \nu^t - \frac{1}{N}\sum_{i\in I_t} \nabla h_i(\alpha_i^t) + \frac{1}{N}\sum_{i\in I_t} \nabla h_i(\alpha_i^{t+1})$.
  Update $t = t + 1$.
until Stopping criterion.

5. *If r_2 has Lipschitz continuous gradient over $\operatorname{dom} r_1$, then $d_T = O(1/\sqrt{T})$. Furthermore, by choosing*

$$b = \left\lceil \frac{\sqrt[4]{2}}{\sqrt{\mu}} N^{3/4} \right\rceil, \text{ where } \mu = \frac{\rho_H + \rho_{r_2} + \rho_{G+r_1}}{4L},$$

the complexity in terms of number of gradient evaluations to obtain ϵ-criticality in expectation is $O(N + N^{3/4}/\epsilon^2)$; Meanwhile, the complexity in terms of number of convex subproblems solved is $O(1/\epsilon^2)$.

Proof of Theorem 7. First, we prove the above properties for the sampling with replacement option. The proof for the sampling without replacement option is then outlined, where the key different arguments are highlighted.

Consider the sampling with replacement option.

1. Since H is ρ_H-convex, r_2 is ρ_{r_2}-convex, and by the definition of x^{t+1},

$$\begin{aligned} H(x^{t+1}) &\geq H(x^t) + \langle \nabla H(x^t), x^{t+1} - x^t\rangle + \frac{\rho_H}{2}\|x^{t+1} - x^t\|^2. \\ r_2(x^{t+1}) &\geq r_2(x^t) + \langle y^t, x^{t+1} - x^t\rangle + \frac{\rho_{r_2}}{2}\|x^{t+1} - x^t\|^2, \\ (G + r_1)(x^{t+1}) &\leq (G + r_1)(x^t) + \langle z^t, x^{t+1} - x^t\rangle - \frac{\rho_{G+r_1}}{2}\|x^{t+1} - x^t\|^2. \end{aligned}$$

Combining these inequalities,

$$F(x^{t+1}) \leq F(x^t) + \langle x^{t+1} - x^t, v^t - \nabla H(x^t)\rangle - \frac{\rho}{2}\|x^{t+1} - x^t\|^2, \tag{2.13}$$

where $\rho = \rho_H + \rho_{r_2} + \rho_{G+r_1}$. By taking conditional expectation with respect to $\mathcal{P}_t$,

$$\mathbb{E}(F(x^{t+1})|\mathcal{P}_t) \leq F(x^t) + \mathbb{E}(\langle x^{t+1} - x^t, v^t - \nabla H(x^t)\rangle|\mathcal{P}_t) - \frac{\rho}{2}\mathbb{E}(\|x^{t+1} - x^t\|^2|\mathcal{P}_t),$$

or can be rewritten as follows to address the expectation is taken over the randomness of I_t,

$$\mathbb{E}_{I_t}(F(x^{t+1})) \leq F(x^t) + \mathbb{E}_{I_t}\left(\langle x^{t+1} - x^t, v^t - \nabla H(x^t)\rangle\right) - \frac{\rho}{2}\mathbb{E}_{I_t}\|x^{t+1} - x^t\|^2,$$

which implies

$$\mathbb{E}_{I_t}(F(x^{t+1})) \leq F(x^t) + \frac{1}{2\gamma}\mathbb{E}_{I_t}\|v^t - \nabla H(x^t)\|^2 - \frac{\rho - \gamma}{2}\mathbb{E}_{I_t}\|x^{t+1} - x^t\|^2 \tag{2.14}$$

where $\gamma > 0$ that will be determined later to gain some advantages.

We now evaluate $\mathbb{E}_{I_t}\|v^t - \nabla H(x^t)\|^2$ as follows,

$$\begin{aligned}
\mathbb{E}_{I_t}\|v^t - \nabla H(x^t)\|^2 &= \mathbb{E}_{I_t}\left\|\frac{1}{b}\sum_{i\in I_t}\nabla h_i(x^t) + \nu^t - \frac{1}{b}\sum_{i\in I_t}\nabla h_i(\alpha_i^t) - \nabla H(x^t)\right\|^2 \\
&= \frac{1}{b}\mathbb{E}_i\left\|\nabla h_i(x^t) - \nabla h_i(\alpha_i^t) + \nu^t - \nabla H(x^t)\right\|^2 \text{ where } i \overset{\text{uni}}{\sim} \{1, 2, \ldots, N\} \\
&\leq \frac{1}{b}\mathbb{E}_i\|\nabla h_i(x^t) - \nabla h_i(\alpha_i^t)\|^2 \leq \frac{L^2}{b}\mathbb{E}_i\|x^t - \alpha_i^t\|^2 = \frac{L^2}{bN}\sum_{i=1}^N\|x^t - \alpha_i^t\|^2.
\end{aligned} \tag{2.15}$$

Therefore,

$$\mathbb{E}_{I_t}(F(x^{t+1})) \leq F(x^t) - \frac{\rho - \gamma}{2}\mathbb{E}_{I_t}\|x^{t+1} - x^t\|^2 + \frac{L^2}{2\gamma bN}\sum_{i=1}^N\|x^t - \alpha_i^t\|^2.$$

We define the sequence of Lyapunov functions $V^t = F(x^t) + c_t\sum_{i=1}^N\|x^t - \alpha_i^t\|^2$, where the sequence $\{c_t\}$ will be determined later. For all $\beta > 0$,

$$\begin{aligned}
\mathbb{E}_{I_t}(V^{t+1}) &= \mathbb{E}_{I_t}\left(F(x^{t+1}) + c_{t+1}\sum_{i=1}^N\|x^{t+1} - \alpha_i^{t+1}\|^2\right) \\
&\leq \mathbb{E}_{I_t}\left(F(x^{t+1}) + c_{t+1}(\beta+1)\sum_{i=1}^N\|x^{t+1} - x^t\|^2 + c_{t+1}(1+1/\beta)\sum_{i=1}^N\|x^t - \alpha_i^{t+1}\|^2\right) \\
&\leq \mathbb{E}_{I_t}\left(F(x^t) + \left(Nc_{t+1}(\beta+1) - \frac{\rho-\gamma}{2}\right)\|x^{t+1} - x^t\|^2\right. \\
&\qquad \left. + \frac{L^2}{2\gamma bN}\sum_{i=1}^N\|x^t - \alpha_i^t\|^2 + c_{t+1}(1+1/\beta)\sum_{i\notin I_t}\|x^t - \alpha_i^t\|^2\right)
\end{aligned}$$

$$
\begin{aligned}
&= F(x^t) + \left(Nc_{t+1}(\beta+1) - \frac{\rho-\gamma}{2}\right)\mathbb{E}_{I_t}\|x^{t+1}-x^t\|^2 \\
&\quad + \left(\frac{L^2}{2\gamma bN} + c_{t+1}(1+1/\beta)\right)\sum_{i=1}^{N}\|x^t-\alpha_i^t\|^2 - c_{t+1}\left(1+\frac{1}{\beta}\right)b\mathbb{E}_i\|x^t-\alpha_i^t\|^2 \\
&= F(x^t) + \left(Nc_{t+1}(\beta+1) - \frac{\rho-\gamma}{2}\right)\mathbb{E}_{I_t}\|x^{t+1}-x^t\|^2 \\
&\quad + \left(\frac{L^2}{2\gamma bN} + c_{t+1}\left(1+\frac{1}{\beta}\right)\left(1-\frac{b}{N}\right)\right)\sum_{i=1}^{N}\|x^t-\alpha_i^t\|^2.
\end{aligned}
$$

Similar to the proof of DCA-SVRG, we set $c_t = \frac{L^2}{2\gamma bN} + c_{t+1}\left(1+\frac{1}{\beta}\right)\left(1-\frac{b}{N}\right)$, or equivalently, $c_{t+1} = \left(c_t - \frac{L^2}{2\gamma bN}\right)\frac{\beta N}{(\beta+1)(N-b)}$. We obtain by recursion that

$$
c_t = \left(c_0 - \frac{L^2\beta}{2\gamma b(\beta b+b-N)}\right)\left(\frac{\beta N}{(\beta+1)(N-b)}\right)^t + \frac{L^2\beta}{2\gamma b(\beta b+b-N)}.
$$

From the above recursion, if we choose the initial value $c_0 = \frac{L^2\beta}{2\gamma b(\beta b+b-N)}$, we will get $c_0 = c_1 = c_2 = \ldots = c_t = \ldots$ Next, we choose β such that $\beta b+b-N > 0 \Leftrightarrow \beta > \frac{N-b}{b}$ $(\star)$ to make the sequence $\{c_t\}$ nonnegative. With this choice, we obtain $\mathbb{E}_{I_t}(V^{t+1}) \leq V^t + \left(Nc_{t+1}(\beta+1) - \frac{\rho-\gamma}{2}\right)\mathbb{E}_{I_t}\|x^{t+1}-x^t\|^2$.

In order to obtain the "decreasing property" of the sequence $\{V^t\}$, we want the term $Nc_{t+1}(\beta+1) - \frac{\rho-\gamma}{2}$ to be negative. More specifically, we want $Nc_{t+1}(\beta+1) + \frac{\gamma}{2} \leq \frac{\rho}{4}$, or equivalently,

$$
N(\beta+1)\frac{L^2\beta}{2\gamma b(\beta b+b-N)} + \frac{\gamma}{2} \leq \frac{\rho}{4}. \tag{2.16}
$$

Now by choosing $\gamma > 0$ to make the LHS of (2.16) as small as possible (AM-GM inequality), $\gamma = L\sqrt{\frac{\beta(\beta+1)N}{b(\beta b+b-N)}}$, we need $\sqrt{\frac{N(\beta+1)L^2\beta}{b(\beta b+b-N)}} \leq \frac{\rho}{4}$, or equivalently

$$
\sqrt{\frac{N(\beta+1)\beta}{b(\beta b+b-N)}} \leq \frac{\rho}{4L}. \tag{2.17}
$$

From $(\star)$, we choose $\beta = N/b$. The inequality (2.17) becomes

$$
\frac{N\sqrt{N+b}}{b^2} \leq \frac{\rho}{4L}. \tag{2.18}
$$

Consequently, if b satisfies (2.18), we obtain $\mathbb{E}_{I_t}(V^{t+1}) \leq V^t - \frac{\rho}{4}\mathbb{E}_{I_t}\|x^{t+1}-x^t\|^2$. Since $c_t \geq 0$, we obtain $V^t \geq F(x^t) \geq \alpha > -\infty$. By applying supermartingale convergence theorem, we obtain: there exits V^∞ such that $V^t \to V^\infty$ a.s. Next, we have $\frac{\rho}{4}\mathbb{E}\|x^{t+1}-x^t\|^2 \leq \mathbb{E}(V^t) - \mathbb{E}(V^{t+1})$. By induction based on this inequality, we obtain $\mathbb{E}(V^t)$ is finite for all t. By telescoping the above inequality, we derive $\frac{\rho}{4}\sum_{t=0}^{\infty}\mathbb{E}\|x^{t+1}-x^t\|^2 \leq \mathbb{E}(V^0) - \alpha < +\infty$.

2. We now evaluate the term $\mathbb{E}\|x^t-\alpha_i^t\|^2$. This is a key for us to establish the convergence to critical points. By the definition, we can write $\alpha_i^t = x^{t-1}\cdot 1_{\{i\in I_{t-1}\}} + \alpha_i^{t-1}\cdot 1_{\{i\notin I_{t-1}\}}$. Similarly, we can track back one further step $\alpha_i^t = x^{t-1}\cdot 1_{\{i\in I_{t-1}\}} + x^{t-2}\cdot 1_{\{i\in \bar{I}_{t-1}\cap I_{t-2}\}} + \alpha_i^{t-2}\cdot 1_{\{i\in \bar{I}_{t-1}\cap \bar{I}_{t-2}\}}$, where $\bar{I}$ is

the complement of I, $\bar{I} = \{1, 2, \ldots, N\} \setminus I$.
By doing so repeatedly,

$$\begin{aligned}\alpha_i^t =& x^{t-1} \cdot 1_{\{i \in I_{t-1}\}} + x^{t-2} \cdot 1_{\{i \in \bar{I}_{t-1} \cap I_{t-2}\}} + x^{t-3} \cdot 1_{\{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2} \cap I_{t-3}\}} \\ &+ \ldots + x^0 \cdot 1_{\{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2} \cdots \cap \bar{I}_1\}}.\end{aligned}$$

It is worth noting that $\{i \in I_{t-1}\}, \{i \in \bar{I}_{t-1} \cap I_{t-2}\}, \{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2} \cap I_{t-3}\}, \ldots, \{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2} \cdots \cap \bar{I}_1\}$ form a partition of the sample space Ω, and we obtain

$$\alpha_i^t = \begin{cases} x^{t-1} \text{ on } \{i \in I_{t-1}\} \\ x^{t-2} \text{ on } \{i \in \bar{I}_{t-1} \cap I_{t-2}\} \\ x^{t-3} \text{ on } \{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2} \cap I_{t-3}\} \\ \ldots \\ x^0 \text{ on } \{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2} \cdots \cap \bar{I}_1\}. \end{cases}$$

Therefore,

$$\begin{aligned}&\mathbb{E}\|x^t - \alpha_i^t\|^2 = \int_\Omega \|x^t - \alpha_i^t\|^2 d\mathbb{P} \\ &= \int_{\{i \in I_{t-1}\}} \|x^t - x^{t-1}\|^2 d\mathbb{P} + \int_{\{i \in \bar{I}_{t-1} \cap I_{t-2}\}} \|x^t - x^{t-2}\|^2 d\mathbb{P} \\ &+ \int_{\{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2} \cap I_{t-3}\}} \|x^t - x^{t-3}\|^2 d\mathbb{P} + \ldots + \int_{\{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2} \cdots \cap \bar{I}_1\}} \|x^t - x^0\|^2 d\mathbb{P}. \end{aligned} \tag{2.19}$$

Our main aim is to prove $\sum_{t=1}^{\infty} \mathbb{E}\|x^t - \alpha_i^t\|^2 < +\infty$.
Here we have several remarks playing as guiding light in our proof:

- We already proved $\sum_{t=0}^{\infty} \mathbb{E}\|x^{t+1} - x^t\|^2 < +\infty$, therefore, we will find links between $\mathbb{E}\|x^t - \alpha_i^t\|^2$ and $\mathbb{E}\|x^t - x^{t-1}\|^2, \mathbb{E}\|x^{t-1} - x^{t-2}\|^2, \ldots$

- Our main challenge when dealing with the RHS of (2.19) is that it is very hard to compute explicitly those integrals since, for instance, $\|x^t - x^{t-1}\|^2$ and $\{i \in \bar{I}_{t-1}\}$ are not independent.

- We know that if a random variable X is independent to the set of events B, then we have the property $\int_B X d\mathbb{P} = \mathbb{E}(X)\mathbb{P}(B)$. We will therefore evaluate in such a way that after our evaluations, by grouping integrals with the common function integrated, the combined region is independent to that function. For example, after some evaluations, we obtain something of the form

  $$\int_{A_1} X d\mathbb{P} + \int_{A_2} X d\mathbb{P} + \ldots + \int_{A_m} X d\mathbb{P} = \int_{A_1 \cup A_2 \cup \ldots \cup A_m} X d\mathbb{P},$$

  where $A_1, A_2, \ldots, A_m$ are pair-wise disjoint. Here our aim is to have $A_1 \cup A_2 \cup \ldots \cup A_m$ being independent to X, so we can use the mentioned property.

With these meta-ideas kept in mind, we are back to the proof. We denote $\{\epsilon_i\}$ the sequence as $\epsilon_i = \frac{1}{i^2}$. The sequence $\{\epsilon_i\}$ plays a crucial role in our proof. Moreover, we denote $A = \sum_{i=1}^{\infty} \epsilon_i = \sum_{i=1}^{\infty} \frac{1}{i^2} <$

$+\infty$. We define another sequence $\{\theta_i\}$ as $\theta_i = \frac{\epsilon_i}{A}$, which implies $\sum_{i=1}^{\infty} \theta_i = 1$. We will evaluate $\mathbb{E}\|x^t - \alpha_i^t\|^2, (t \geq 2)$ based on (2.19) as follows. Cauchy-Schwartz inequality implies

$$\int_{\{i\in\bar{I}_{t-1}\cap I_{t-2}\}} \|x^t - x^{t-2}\|^2 d\mathbb{P} \leq \frac{1}{\theta_1} \int_{\{i\in\bar{I}_{t-1}\cap I_{t-2}\}} \|x^t - x^{t-1}\|^2 d\mathbb{P} + \frac{1}{1-\theta_1} \int_{\{i\in\bar{I}_{t-1}\cap I_{t-2}\}} \|x^{t-1} - x^{t-2}\|^2 d\mathbb{P},$$

$$\int_{\{i\in\bar{I}_{t-1}\cap\bar{I}_{t-2}\cap I_{t-3}\}} \|x^t - x^{t-3}\|^2 d\mathbb{P} \leq \frac{1}{\theta_1} \int_{\{i\in\bar{I}_{t-1}\cap\bar{I}_{t-2}\cap I_{t-3}\}} \|x^t - x^{t-1}\|^2 d\mathbb{P} + \frac{1}{\theta_2} \int_{\{i\in\bar{I}_{t-1}\cap\bar{I}_{t-2}\cap I_{t-3}\}} \|x^{t-1} - x^{t-2}\|^2 d\mathbb{P} + \frac{1}{1-\theta_1-\theta_2} \int_{\{i\in\bar{I}_{t-1}\cap\bar{I}_{t-2}\cap I_{t-3}\}} \|x^{t-2} - x^{t-3}\|^2 d\mathbb{P},$$

$\ldots$

$$\int_{\{i\in\bar{I}_{t-1}\cap\ldots\cap\bar{I}_1\}} \|x^t - x^0\|^2 d\mathbb{P} \leq \frac{1}{\theta_1} \int_{\{i\in\bar{I}_{t-1}\cap\ldots\cap\bar{I}_1\}} \|x^t - x^{t-1}\|^2 d\mathbb{P} + \frac{1}{\theta_2} \int_{\{i\in\bar{I}_{t-1}\cap\ldots\cap\bar{I}_1\}} \|x^{t-1} - x^{t-2}\|^2 d\mathbb{P} + \ldots + \frac{1}{\theta_{t-1}} \int_{\{i\in\bar{I}_{t-1}\cap\ldots\cap\bar{I}_1\}} \|x^2 - x^1\|^2 d\mathbb{P} + \frac{1}{1-\theta_1-\theta_2-\ldots-\theta_{t-1}} \int_{\{i\in\bar{I}_{t-1}\cap\ldots\cap\bar{I}_1\}} \|x^1 - x^0\|^2 d\mathbb{P}.$$

By adding (2.19) and these inequalities, then grouping integrals sharing common the function integrated, namely $\|x^t - x^{t-1}\|^2, \|x^{t-2} - x^{t-1}\|^2, \ldots, \|x^1 - x^0\|^2$, and noticing

$$\{i \in \bar{I}_{t-1} \cap I_{t-2}\} \cup \{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2} \cap I_{t-3}\} \ldots \cup \{i \in \bar{I}_{t-1} \cap \ldots \bar{I}_1\} = \{i \in \bar{I}_{t-1}\},$$
$$\{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2} \cap I_{t-3}\} \cup \ldots \cup \{i \in \bar{I}_{t-1} \cap \ldots \bar{I}_1\} = \{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2}\},$$
$\ldots$

we get

$$\mathbb{E}\|x^t - \alpha_i^t\|^2 \leq \int_{\{i\in I_{t-1}\}} \|x^t - x^{t-1}\|^2 d\mathbb{P} + \frac{1}{\theta_1} \int_{\{i\in\bar{I}_{t-1}\}} \|x^t - x^{t-1}\|^2 d\mathbb{P} + \frac{1}{1-\theta_1} \int_{\{i\in\bar{I}_{t-1}\cap I_{t-2}\}} \|x^{t-1} - x^{t-2}\|^2 d\mathbb{P} + \frac{1}{\theta_2} \int_{\{i\in\bar{I}_{t-1}\cap\bar{I}_{t-2}\}} \|x^{t-1} - x^{t-2}\|^2 d\mathbb{P} + \ldots + \frac{1}{1-\theta_1-\ldots-\theta_{t-2}} \int_{\{i\in\bar{I}_{t-1}\cap\ldots\cap I_1\}} \|x^2 - x^1\|^2 d\mathbb{P} + \frac{1}{\theta_{t-1}} \int_{\{i\in\bar{I}_{t-1}\cap\ldots\cap\bar{I}_1\}} \|x^2 - x^1\|^2 d\mathbb{P} + \frac{1}{1-\theta_1-\theta_2-\ldots-\theta_{t-1}} \int_{\{i\in\bar{I}_{t-1}\cap\ldots\cap\bar{I}_1\}} \|x^1 - x^0\|^2 d\mathbb{P}.$$

By observing that $1 \leq \frac{1}{\theta_1}, \quad \frac{1}{1-\theta_1} \leq \frac{1}{\theta_2}, \quad \ldots, \quad \frac{1}{1-\theta_1-\ldots-\theta_{t-2}} \leq \frac{1}{\theta_{t-1}}, \frac{1}{1-\theta_1-\ldots-\theta_{t-1}} \leq \frac{1}{\theta_t}$, we have

$$\mathbb{E}\|x^t - \alpha_i^t\|^2 \leq \frac{1}{\theta_1}\int_\Omega \|x^t - x^{t-1}\|^2 d\mathbb{P} + \frac{1}{\theta_2}\int_{\{i\in\bar{I}_{t-1}\}} \|x^{t-1} - x^{t-2}\|^2 d\mathbb{P}$$
$$+ \ldots + \frac{1}{\theta_{t-1}}\int_{\{i\in\bar{I}_{t-1}\cap\ldots\cap\bar{I}_2\}} \|x^2 - x^1\|^2 d\mathbb{P} + \frac{1}{\theta_t}\int_{\{i\in\bar{I}_{t-1}\cap\ldots\cap\bar{I}_1\}} \|x^1 - x^0\|^2 d\mathbb{P}.$$

By the independence of each pair $\|x^{t-1}-x^{t-2}\|^2$ and $\{i \in \bar{I}_{t-1}\}$,..., $\|x^2-x^1\|^2$ and $\{i \in \bar{I}_{t-1}\cap\ldots\cap\bar{I}_2\}$, $\|x^1 - x^0\|^2$ and $\{i \in \bar{I}_{t-1} \cap \ldots \cap \bar{I}_1\}$,

$$\begin{aligned}
\mathbb{E}\|x^t - \alpha_i^t\|^2 &\leq \frac{1}{\theta_1}\mathbb{E}\|x^t - x^{t-1}\|^2 + \frac{1}{\theta_2}\mathbb{E}\|x^{t-1} - x^{t-2}\|^2\mathbb{P}(\{i \in \bar{I}_{t-1}\}) + \ldots \\
&+ \frac{1}{\theta_{t-1}}\mathbb{E}\|x^2 - x^1\|^2\mathbb{P}(\{i \in \bar{I}_{t-1} \cap \ldots \cap \bar{I}_2\}) + \frac{1}{\theta_t}\mathbb{E}\|x^1 - x^0\|^2\mathbb{P}(\{i \in \bar{I}_{t-1} \cap \ldots \cap \bar{I}_1\}) \\
&= \frac{1}{\theta_1}\mathbb{E}\|x^t - x^{t-1}\|^2 + \frac{1}{\theta_2}\mathbb{E}\|x^{t-1} - x^{t-2}\|^2\left(\frac{N-1}{N}\right)^b + \ldots \\
&+ \frac{1}{\theta_{t-1}}\mathbb{E}\|x^2 - x^1\|^2\left(\frac{N-1}{N}\right)^{b(t-2)} + \frac{1}{\theta_t}\mathbb{E}\|x^1 - x^0\|^2\left(\frac{N-1}{N}\right)^{b(t-1)}.
\end{aligned}$$

For short, let us denote $\lambda = \left(\frac{N-1}{N}\right)^b < 1$. The inequality can be written as follows

$$\mathbb{E}\|x^t - \alpha_i^t\|^2 \leq \sum_{k=1}^{t} \frac{\lambda^{t-k}}{\theta_{t-k+1}}\mathbb{E}\|x^k - x^{k-1}\|^2.$$

Therefore,

$$\begin{aligned}
\sum_{t=1}^{\infty}\mathbb{E}\|x^t - \alpha_i^t\|^2 &\leq \sum_{t=1}^{\infty}\sum_{k=1}^{t}\frac{1}{\theta_{t-k+1}}\lambda^{t-k}\mathbb{E}\|x^k - x^{k-1}\|^2 \\
&= \sum_{k=1}^{\infty}\sum_{t=k}^{\infty}\frac{1}{\theta_{t-k+1}}\lambda^{t-k}\mathbb{E}\|x^k - x^{k-1}\|^2 = \sum_{k=1}^{\infty}\mathbb{E}\|x^k - x^{k-1}\|^2\sum_{t=k}^{\infty}\frac{\lambda^{t-k}}{\theta_{t-k+1}} \\
&= \sum_{k=1}^{\infty}\mathbb{E}\|x^k - x^{k-1}\|^2\sum_{t=0}^{\infty}\frac{\lambda^t}{\theta_{t+1}} = A\left(\sum_{t=0}^{\infty}\frac{\lambda^t}{\epsilon_{t+1}}\right)\sum_{k=1}^{\infty}\mathbb{E}\|x^k - x^{k-1}\|^2 \\
&= A\left(\sum_{t=0}^{\infty}(t+1)^2\lambda^t\right)\sum_{k=1}^{\infty}\mathbb{E}\|x^k - x^{k-1}\|^2 < +\infty,
\end{aligned}$$

since the power series $u(x) = \sum_{t=0}^{\infty}(t+1)^2\lambda^t$ has the convergence radius 1.

3. As a consequence of properties 1,2 above, we derive that $\{F(x^t)\}$ converges to V^∞ almost surely.
4. If follows from (2.15) and property *2* that $\mathbb{E}\left(\sum_{k=1}^{\infty}\|v^k - \nabla H(x^k)\|^2\right) < +\infty$. Let

$$S = \left(v^k - \nabla H(x^k) \to 0, x^k - x^{k+1} \to 0, \{y^k\} \text{ is bounded}\right).$$

It is evident that $\mathbb{P}(S) = 1$. We consider a fixed event in S to obtain a realization $\{x^k\}$. Let x^* be a limit point of $\{x^k\}$, there exists a subsequence $\{x^{l_k}\}$ such that $x^{l_k} \to x^*$, implying $x^{l_k+1} \to x^*$. Since H is L-smooth, we obtain $\nabla H(x^{l_k}) \to \nabla H(x^*)$, which implies, $v^{l_k} \to \nabla H(x^*)$. Since $\{y^{l_k}\}$ is bounded, by passing to a subsequence if necessary, we assume that $y^{l_k} \to y^*$, therefore, $y^* \in \partial r_2(x^*)$. On the

other hand, we have $z^{l_k} \in \partial(G + r_1)(x^{l_k+1})$. By the closedness of the graph subdifferential operator, we obtain $\nabla H(x^*) + y^* \in \partial(G + r_1)(x^*)$. Therefore, x^* is a critical point of $F = (G + r_1) - (H + r_2)$.

5. Since $z^t \in \partial(G + r_1)(x^{t+1})$,

$$\begin{aligned}
&\operatorname{dist}\left(\nabla H(x^{t+1}) + \nabla r_2(x^{t+1}), \partial(G + r_1)(x^{t+1})\right) \leq \|\nabla H(x^{t+1}) + \nabla r_2(x^{t+1}) - v^t - y^t\| \\
&\leq \|\nabla H(x^{t+1}) - \nabla H(x^t)\| + \|\nabla H(x^t) - v^t\| + \|\nabla r_2(x^{t+1}) - \nabla r_2(x^t)\| \\
&\leq (L + L_{r_2})\|x^{t+1} - x^t\| + \|\nabla H(x^t) - v^t\|.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\mathbb{E}\operatorname{dist}\left(\nabla H(x^{t+1}) + \nabla r_2(x^{t+1}), \partial(G + r_1)(x^{t+1})\right) \\
&\leq (L + L_{r_2})\mathbb{E}\|x^{t+1} - x^t\| + \left(\mathbb{E}\|\nabla H(x^t) - v^t\|^2\right)^{\frac{1}{2}} \\
&\leq (L + L_{r_2})\left(\mathbb{E}\|x^{t+1} - x^t\|^2\right)^{\frac{1}{2}} + \frac{L}{\sqrt{bN}}\left(\sum_{i=1}^{N}\mathbb{E}\|x^t - \alpha_i^t\|^2\right)^{\frac{1}{2}}.
\end{aligned}$$

Summing these inequalities for $t = 0$ to $T - 1$,

$$\begin{aligned}
&\sum_{t=0}^{T-1}\mathbb{E}\operatorname{dist}(\nabla H(x^{t+1}) + \nabla r_2(x^{t+1}), \partial(G + r_1)(x^{t+1})) \\
&\leq (L + L_{r_2})\sum_{t=0}^{T-1}\left(\mathbb{E}\|x^{t+1} - x^t\|^2\right)^{\frac{1}{2}} + \frac{L}{\sqrt{bN}}\sum_{t=0}^{T-1}\left(\sum_{i=1}^{N}\mathbb{E}\|x^t - \alpha_i^t\|^2\right)^{\frac{1}{2}} \\
&\leq (L + L_{r_2})\sqrt{T}\left(\sum_{t=0}^{T-1}\mathbb{E}\|x^{t+1} - x^t\|^2\right)^{\frac{1}{2}} + \frac{L}{\sqrt{bN}}\sqrt{T}\left(\sum_{t=0}^{T-1}\sum_{i=1}^{N}\mathbb{E}\|x^t - \alpha_i^t\|^2\right)^{\frac{1}{2}} \\
&\leq (L + L_{r_2})\sqrt{T}C + \frac{L\sqrt{T}C}{\sqrt{b}} \leq (2L + L_{r_2})\,C\sqrt{T}, \qquad \text{for some } C > 0.
\end{aligned}$$

It follows that the iteration convergence rate is $d_T = O(1/\sqrt{T})$.
If the output x_a is chosen randomly from $\{x^t\}_{t=1}^T$,

$$\mathbb{E}\operatorname{dist}(\nabla H(x_a) + \nabla r_2(x_a), \partial(G + r_1)(x_a)) = O\left(\frac{1}{\sqrt{T}}\right).$$

Therefore, to get ϵ-criticality, the algorithm needs $T = O(1/\epsilon^2)$ iterations. From the condition for b, if we choose

$$b = \left\lceil \frac{\sqrt[4]{2}}{\sqrt{\mu}}N^{3/4} \right\rceil, \text{ where } \mu = \frac{\rho_H + \rho_{r_2} + \rho_{G+r_1}}{4L},$$

one obtains the complexity of $O(N + N^{3/4}/\epsilon^2)$ in terms of the number of gradient evaluations and the complexity of $O(1/\epsilon^2)$ in terms of the number of convex subproblems.

Sketched proof for the sampling without replacement option.

1. We arrive at the inequality (2.14) by following the arguments offered in the previous proof. We

then introduce the following lemma whose proof is similar to that of Lemma 1.

Lemma 2. *The following inequality holds,*

$$\mathbb{E}_{I_t}\|v^t - \nabla H(x^t)\|^2 \leq \frac{L^2}{bN}\left(1 - \frac{b-1}{N-1}\right)\sum_{i=1}^{N}\|x^t - \alpha_i^t\|^2.$$

Similar to the arguments above, by introducing the Lyapunov sequence $V^t = F(x^t) + c_t\sum_{i=1}^{N}\|x^t - \alpha_i^t\|^2$ with $c_t = \frac{\beta L^2(N-b)}{2\gamma b(N-1)(b-N+\beta b)}$, where $\beta = N/b$, and γ is determined by the AM-GM inequality as above, we want

$$\frac{N\sqrt{N^2-b^2}}{b^2\sqrt{N-1}} \leq \frac{\rho}{4L}. \tag{2.20}$$

Consequently, if $\frac{N\sqrt{N+1}}{b^2} \leq \frac{\rho}{4L}$, (2.20) holds and we obtain that $\{V^t\}$ converges almost surely and $\sum_{t=0}^{\infty}\mathbb{E}\|x^{t+1} - x^t\|^2 < \infty$.

2. The arguments above can be employed with the following notice: $\mathbb{P}(\{i \in \bar{I}_{t-1}\}) = 1 - b/N, \mathbb{P}(\{i \in \bar{I}_{t-1} \cap \bar{I}_{t-2}\}) = (1-b/N)^2$, etc.

3,4,5. These properties are established similar to the option of sampling with replacement.

□

Remark 2. *i. We can apply the same technique discussed in Remark 1 to improve the number $\frac{\rho}{L}$ of the problem if necessary.*

ii. The analysis providing links between $\mathbb{E}\|x^t - \alpha_i^t\|^2$ and $\sum_{t=1}^{\infty}\mathbb{E}\|x^{t+1} - x^t\|^2$ is quite novel in the literature, which is expected to be useful in analyzing SAGA-type algorithms. The analysis offers an elegant solution to escape the "non-independence" of the function and the region of an integral.

Remark 3. *Several comparisons to related algorithms:*

i. From the convergence of the standard DCA [103], one can verify that to find an ϵ-criticality, under the L-smooth assumption of H and r_2, it requires $O(1/\epsilon^2)$ iterations. As a result, the standard DCA has the complexity of $O(N/\epsilon^2)$ in terms of gradient evaluations and $O(1/\epsilon^2)$ in terms of the number of convex subproblems. While the proposed DCA-SVRG and DCA-SAGA enjoy the same complexity as the standard DCA in terms of the number of convex subproblems, they significantly outperform the standard DCA in terms of gradient evaluations, DCA-SVRG: $O(N^{2/3}/\epsilon^2)$, DCA-SAGA: $O(N + N^{3/4}/\epsilon^2)$.

ii. [134] considered the following large-sum problem

$$\min_{x\in\mathbb{R}^n} F(x) = G(x) - H(x) + r(x) := \frac{1}{\bar{N}}\sum_{i=1}^{\bar{N}} g_i(x) - \frac{1}{N}\sum_{j=1}^{N} h_j(x) + r(x),$$

where g_i, h_j are convex and the regularizer r is convex. The authors also examine the case that r is nonconvex (not necessarily DC) where the proximal mapping can be computed efficiently. For the sake of simplicity, we only discuss the case where r is convex (note that their complexity for the case of nonconvex r is obviously worse than in the case where r is convex). When each h_j is L-smooth, the best gradient complexity to find (nearly) ϵ-criticality is given in [134, Corollary 9], which is $O((\bar{N}+N)/\epsilon^2)$. Obviously, this complexity takes the cost of solving each large-sum convex subproblem by the SVRG into account. In order to compare this complexity to our algorithms, we must express it as the number of

gradient evaluations of H and the number of convex subproblems being solved. To obtain a (nearly) ϵ-critical point, they need $O(1/\epsilon^2)$ stages in total [134, Lemma 3 and Corollary 9]. Each stage consumes a full gradient of H and requires solving inexactly a convex subproblem, resulting in the complexity of $O(N/\epsilon^2)$ in terms of gradient evaluations and $O(1/\epsilon^2)$ in terms of the number of convex subproblems. Our algorithms enjoy superior gradient complexity, while the complexity in terms of convex subproblems is somewhat incomparable since their algorithmic design also takes into account the inexactness of solving each such problem, while we do not quantify this issue here.

2.4 Additional convergence analysis for the composite structure of r_2

In this section, we consider $r_2(x) = \sum_{i=1}^m l_i(p_i(x))$ where $l_i : \mathbb{R} \to \mathbb{R}$ is convex and decreasing, $p_i : \mathbb{R}^n \to \mathbb{R}$ is convex and hence continuous, for $i = \overline{1,m}$. The optimization problem (P) becomes

$$(Q) \quad \alpha = \min \left\{ F(x) = G(x) - \frac{1}{N} \sum_{i=1}^N h_i(x) + r_1(x) - \sum_{i=1}^m l_i(p_i(x)) \right\}.$$

By denoting $\Omega = \{(x,z) \in \mathbb{R}^n \times \mathbb{R}^m : p_i(x) \leq z_i, \forall i = \overline{1,m}\}$, we can solve the following problem instead

$$(Q') \quad \alpha = \min \left\{ \varphi(x,z) = G(x) - \frac{1}{N} \sum_{i=1}^N h_i(x) + r_1(x) + \chi_\Omega(x,z) - \sum_{i=1}^m l_i(z_i) \right\},$$

since if (x^*, z^*) is the optimal solution of (Q'), x^* is the optimal solution of (Q). We notice that the problem (Q') with respect to the optimization variables (x,z) takes the form of (P). As a result, we can use the proposed DCA-SVRG and DCA-SAGA to solve (Q'). However, because x and z do not play the same role, the convergence analysis of DCA-SVRG and DCA-SAGA should be modified for this case. We can see that the modulus of convexity of G and H with respect to (x,z) is 0 due to the newly introduced optimization variable z, which is undesirable. By Remark 1, we can add the convex term $\frac{\gamma}{2}\|(x,z)\|^2$ to both DC components. However, this technique has the risk of resulting in bad approximations. Therefore, we still want to use the modulus of strong convexity of G and H with respect to x only in our analysis, which necessitates adapting the convergence analysis of DCA-SVRG and DCA-SAGA to this case. For convenience of presentation, we provide in detail the DCA-SVRG scheme applied to (Q') and state the convergence results with its sketched proof. For DCA-SAGA applied to (Q'), only present convergence results are presented.

The DCA-SVRG scheme applied to (Q') is given in the Algorithm 3, where we denote $r_3(z) = \sum_{i=1}^m l_i(z_i)$.

Henceforth, we still denote ρ_H, ρ_{G+r_1} the modulus of strong convexity of H and $G + r_1$ and L the Lipschitz constant of ∇h_i with respect to x only. We derive the following convergence results.

Theorem 8. *If the minibatch size b and the inner-loop length M satisfy*

$$\frac{M}{\sqrt{b}} \leq \frac{1}{4\sqrt{e-1}} \frac{\rho_{G+r_1} + \rho_H}{L},$$

Algorithm 3 DCA-SVRG applied to (Q')

Initialization: $\tilde{x}^0 \in \operatorname{dom} r_1$, inner-loop length M, minibatch size b, $k = 0$, option (either *with replacement* or *without replacement*).
repeat
Compute the full gradient $\tilde{\nu}^k = \frac{1}{N}\sum_{i=1}^N \nabla h_i(\tilde{x}^k)$ and set $x_0^{k+1} = \tilde{x}^k$, $z_0^{k+1} = (z_{0,1}^{k+1}, z_{0,2}^{k+1}, \ldots, z_{0,m}^{k+1})^\top$ with $z_{0,r}^{k+1} = p_r(x_0^{k+1})$.
for $j = 0 : M-1$ **do**
if option is *with replacement* **then**
Randomly choose with replacement the set I_t of b elements of $[N]$.
else
Randomly choose without replacement the set I_t of b elements of $[N]$.
end if
Compute the "stochastic variance reduced gradient" t_j^{k+1} by

$$t_j^{k+1} = \frac{1}{b}\sum_{i\in I_b} \nabla h_i(x_j^{k+1}) + \tilde{\nu}^k - \frac{1}{b}\sum_{i\in I_b} \nabla h_i(\tilde{x}^k).$$

Compute $y_j^{k+1} \in \partial r_3(z_j^{k+1})$.
Solve the convex problem

$$(x_{j+1}^{k+1}, z_{j+1}^{k+1}) = \arg\min\{G(x) + r_1(x) + \chi_\Omega(x,z) - \langle t_j^{k+1}, x\rangle - \langle y_j^{k+1}, z\rangle\}$$

end for
Set $\tilde{x}^{k+1} = x_M^{k+1}$, and $k = k+1$.
until Stopping criterion.

then

1. *The sequence $\{F(\tilde{x}^k)\}$ converges almost surely.*

2. $\sum_{k=0}^{\infty}\sum_{j=0}^{M-1} \mathbb{E}\|x_{j+1}^{k+1} - x_j^{k+1}\|^2 < +\infty$.

3. *Suppose the sequence $\{y_j^k\}$ is bounded almost surely, let x^* be a limit point of $\{x_j^k\}$, then (x^*, z^*) is a critical point of $F = (G+r_1+\chi_\Omega)-(H+r_3)$ almost surely, where $z^* = (p_1(x^*), p_2(x^*), \ldots, p_m(x^*))^\top$.*

Proof of Theorem 8. 1, 2. The convex subproblem can be solved as follows

$$x_{j+1}^{k+1} = \arg\min\left\{G(x) + r_1(x) - \langle t_j^{k+1}, x\rangle - \sum_{r=1}^m y_{j,r}^{k+1} p_r(x)\right\} \tag{2.21}$$

$$z_{j+1,r}^{k+1} = p_r(x_{j+1}^{k+1}), \quad \forall r = \overline{1,m}. \tag{2.22}$$

It follows from (2.21) that $t_j^{k+1} \in \partial\left(G + r_1 - \sum_{r=1}^m y_{j,r}^{k+1} p_r\right)(x_{j+1}^{k+1})$, which implies

$$\begin{aligned} G(x_{j+1}^{k+1}) + r_1(x_{j+1}^{k+1}) \leq & G(x_j^{k+1}) + r_1(x_j^{k+1}) + \langle t_j^{k+1}, x_{j+1}^{k+1} - x_j^{k+1}\rangle \\ & - \sum_{r=1}^m y_{j,r}^{k+1}(p_r(x_j^{k+1}) - p_r(x_{j+1}^{k+1})) - \frac{\rho_{G+r_1}}{2}\|x_{j+1}^{k+1} - x_j^{k+1}\|^2. \end{aligned} \tag{2.23}$$

From the convexity of H and r_3, we obtain

$$H(x_{j+1}^{k+1}) \geq H(x_j^{k+1}) + \langle \nabla H(x_j^{k+1}), x_{j+1}^{k+1} - x_j^{k+1} \rangle + \frac{\rho_H}{2} \|x_{j+1}^{k+1} - x_j^{k+1}\|^2, \tag{2.24}$$

$$r_3(z_{j+1}^{k+1}) \geq r_3(z_j^{k+1}) + \langle y_j^{k+1}, z_{j+1}^{k+1} - z_j^{k+1} \rangle. \tag{2.25}$$

From (2.23), (2.24) and (2.25),

$$F(x_{j+1}^{k+1}) \leq F(x_j^{k+1}) + \langle t_j^{k+1} - \nabla H(x_j^{k+1}), x_{j+1}^{k+1} - x_j^k \rangle - \frac{\rho_{G+r_1} + \rho_H}{2} \|x_{j+1}^{k+1} - x_j^{k+1}\|^2,$$

by noting that $\varphi(x_{j+1}^{k+1}, z_{j+1}^{k+1}) = F(x_{j+1}^{k+1}), \varphi(x_j^{k+1}, z_j^{k+1}) = F(x_j^{k+1})$.

Here, by considering the Lyapunov sequence $V_j^{k+1} = F(x_j^{k+1}) + c_j \|x_j^{k+1} - \tilde{x}^k\|^2$ similar to the proof of Theorem 6, we arrive at the conclusion 1 and 2.

3. Let $S = \left(t_j^{k+1} - \nabla H(x_j^{k+1}) \to 0, x_{j+1}^{k+1} - x_j^{k+1} \to 0, \{y_j^k\} \text{ is bounded} \right)$. We see that $\mathbb{P}(S) = 1$ and let $\{x_j^k\}$ be a realization with respect to a fixed event in S. Let x^* be a limit point of $\{x_j^k\}$, there exists a subsequent $x_{v(l)}^{u(l)} \to x^*$, which further implies $x_{v(l)+1}^{u(l)} \to x^*$. From (2.22) and the continuity of p_r, we obtain $z_{v(l)}^{u(l)} \to (p_1(x^*), \ldots, p_m(x^*))^\top$. Without loss of generality, we assume that $y_{v(l)}^{u(l)} \to y^*$, which yields $y^* \in \partial r_3((p_1(x^*), \ldots, p_m(x^*))^\top)$. On the other hand, $t_{v(l)}^{u(l)} \to \nabla H(x^*)$. It follows from

$$t_{v(l)}^{u(l)} \in \partial \left(G + r_1 - \sum_{r=1}^{m} y_{v(l),r}^{u(l)} p_r \right) (x_{v(l)+1}^{u(l)})$$

that $\nabla H(x^*) \in \partial(G + r_1 - \sum_{r=1}^m y_r^* p_r)(x^*)$. Therefore,

$$\begin{pmatrix} \nabla H(x^*) \\ y^* \end{pmatrix} \in \partial(G(x) + r_1(x) + \chi_\Omega(x, z))(x^*, z^*),$$

which implies $\partial(G(x) + r_1(x) + \chi_\Omega(x, z))(x^*, z^*) \cap \partial(H(x) + r_3(z))(x^*, z^*) \neq \emptyset$.

□

Similarly, we have the following convergence results when applying DCA-SAGA to minimize the function $\varphi(x, z)$.

Theorem 9. *If the minibatch size b satisfies $\frac{N\sqrt{N+b}}{b^2} \leq \frac{\rho_H + \rho_{G+r_1}}{4L}$ for the option of sampling with replacement or $\frac{N\sqrt{N+1}}{b^2} \leq \frac{\rho_H + \rho_{G+r_1}}{4L}$ for the option of sampling without replacement, then*

1. $\sum_{t=0}^{\infty} \mathbb{E}\|x^{t+1} - x^t\|^2 < \infty$.
2. $\sum_{t=0}^{\infty} \sum_{i=1}^{N} \mathbb{E}\|x^t - \alpha_i^t\|^2 < \infty$.
3. *The sequence $\{F(x^t)\}$ converges almost surely.*
4. *Suppose the sequence $\{y^t\}$ is bounded almost surely, let x^* be a limit point of $\{x^t\}$, then (x^*, z^*) is a critical point of $F = (G+r_1+\chi_\Omega) - (H+r_3)$ almost surely, where $z^* = (p_1(x^*), p_2(x^*), \ldots, p_m(x^*))^\top$.*

2.5 Applications

In this section, we apply DCA-SVRG and DCA-SAGA to three important machine learning problems to investigate their practical behaviors: nonnegative principal component analysis, group variable selection in multi-class logistic regression, and sparse linear regression.
All numerical experiments in this section are performed on a Processor Intel(R) core(TM) i7-8700, CPU @ 3.20GHz, RAM 16 GB.

2.5.1 Nonnegative principal component analysis

Principal component analysis is arguably one of the most effective tools for reducing dimensionality. PCA has a long history of success in a wide range of applied sciences, including neuroscience, medicine, psychology, material science, scientometry, astronomy, geography, and social sciences.

We conduct numerical experiments on a type of PCA structure known as Non-negative component analysis (NN-PCA). The term "non-negative" implies that we limit the search domain to be the non-negative portion of the search space. This restriction implicitly assumes that we know some additional information about the PCA solution: the solution belongs to the non-negative orthant. As discussed in [90], this additional information offers some theoretical advantages. The NN-PCA has the following form

$$\min_{\|x\|\leq 1, x\geq 0} -\frac{1}{2N}\sum_{i=1}^{N}\langle x, z_i\rangle^2, \tag{2.26}$$

where $\{z_i\}_{i=1}^N$ is the given dataset.

DCA-SVRG and DCA-SAGA applied to (2.26)

We apply DCA-SVRG and DCA-SAGA to (2.26) with the following DC decomposition $G(x) = \frac{\rho}{2}\|x\|^2$, $h_i(x) = \frac{\rho}{2}\|x\|^2 + \frac{1}{2}\langle x, z_i\rangle^2, r_1(x) = \chi_S(x), r_2(x) = 0$, where $S = \{x \in \mathbb{R}^n : x \geq 0, \|x\| \leq 1\}$, and $\rho > 0$. In our experiments, we fix $\rho = 1$.
With this setting, we implement two versions of DCA-SAGA and two versions of DCA-SVRG, namely, DCA-SAGA-v1 (sampling with replacement) and DCA-SAGA-v2 (sampling without replacement), DCA-SVRG-v1 (sampling with replacement) and DCA-SVRG-v2 (sampling without replacement), respectively.

Numerical experiments

Datasets We use standard machine learning datasets in LIBSVM [2], namely, `a9a` (32561×123), `aloi` (108000×128), `cifar10` (50000×3072), `SensIT Vehicle` (78823×100), `connect-4` (67557×126), `letter` (15000×16), `mnist` (60000×780), `protein` (17766×357), `shuttle` (43500×9), `YearPredictionMSD` (463715×90). Each sample of these datasets is normalized as $\|z_i\| = 1$.

[2] The datasets can be downloaded from `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

Comparative algorithms We implement stochastic DCA (abbreviated SDCA) [73] as a main competitor to justify the merits of the variance-reduction estimators used. For a more in-depth comparison, we further implement prox-SGD and prox-SAGA [58], prox-SARAH [101], prox-SpiderBoost [129]. We do not compare with the prox-SVRG here because, as previously discussed, the DCA-SVRG-v1 applied to the above setting recovers a version of the prox-SVRG.

For proximal-type algorithms (prox-SGD, prox-SAGA, prox-SARAH, prox-SpiderBoost), the convex regularizer is set to be $\chi_S(x)$, while each L-smooth component of the large sum is set as $f_i(x) = -\frac{1}{2}\langle x, z_i\rangle^2$ (f_i is 1-smooth).

Experiment setups and results We study the evolution process of the objective in relation to the computational resource used. This resource is measured by the number of stochastic first-order oracle calls (SFO). That is, one SFO call is counted each time we access one stochastic gradient (the gradient of one summand) of the large sum. We set the fixed budget of SFO calls to be $15N$. We do, however, allow all algorithms to use some additional SFO calls to complete their final iteration.

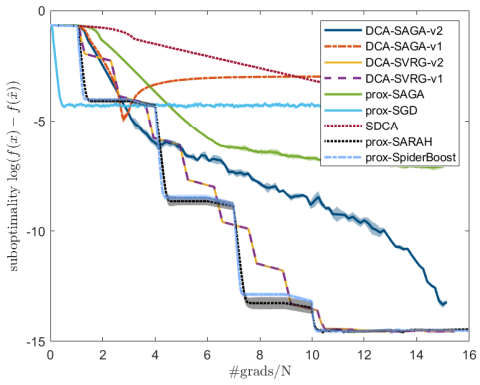
We observe that each function $x \mapsto \frac{1}{2}\langle x, z_i\rangle^2$ is 1-smooth. Based on theoretical convergence results, we choose hyperparameters for the algorithms as follows. The minibatch size b is chosen as $\lfloor N^{\frac{2}{3}}\rfloor, \lfloor N^{\frac{2}{3}}\rfloor, \lceil 2^{\frac{5}{4}}N^{\frac{3}{4}}\rceil, \lceil 2\sqrt{N\sqrt{N+1}}\rceil$ for DCA-SVRG-v1, DCA-SVRG-v2, DCA-SAGA-v1, DCA-SAGA-v2, respectively. We set the inner loop length M for DCA-SVRG-v1 and DCA-SVRG-v2 to be $\lfloor \frac{1}{4\sqrt{e-1}}\sqrt{b}\rfloor$. On the other hand, we use a neutral minibatch size of $10\%N$ which usually yields good results [73] for the SDCA. For the prox-SGD, $\eta = 1/(2L)$ [49] and we choose a neutral minibatch size of 500. For the prox-SAGA, we set $\eta = 1/(5L)$ and the minibatch size $b = \lfloor N^{\frac{2}{3}}\rfloor$ [58]. The hyperparamters of prox-SARAH are chosen as follows [101, Theorem 8]: the minibatch size $b = \lfloor\sqrt{n}\rfloor$, the inner loop length $M = \lfloor N/b\rfloor$, the step size η and the averaging parameter γ:

$$\eta = \frac{2\sqrt{\omega M}}{4\sqrt{\omega M}+1}, \gamma = \frac{1}{L\sqrt{\omega M}}, \text{ where } \omega := \frac{3(N-b)}{2b(N-1)}.$$

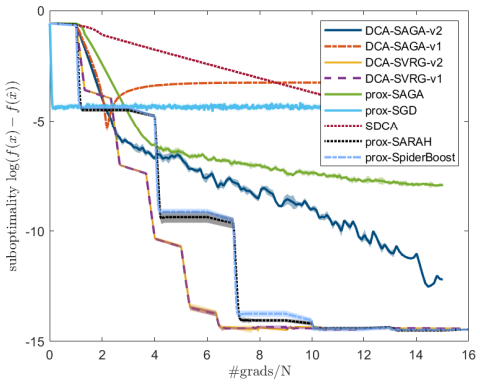
Lastly, for the prox-SpiderBoost, we choose $\eta = 1/(2L)$ and the inner loop length and the mini-batch size $M = b = \lfloor\sqrt{N}\rfloor$ [129, Theorem 1].

We report the suboptimality of the objective, $f(x) - f(\hat{x})$, where $\hat{x}$ is the "optimal" solution found by running the deterministic DCA 500 iterations (which means that this deterministic DCA uses $500N$ SFO calls) 10 times with different initial points. The final optimal value $f(\hat{x})$ is the smallest value found by the deterministic DCA and all the competitive algorithms in this experiment. In order to increase visibility, we plot this suboptimality on the logarithmic scale of base 10. Figure 2.1 shows the results over 10 random runs of all comparative algorithms. The bold lines depict the mean curves, while the shaded regions represent the mean absolute deviation (MAD). Note that, in the normal scale, one might plot $x \pm \mathrm{MAD}(x)$; Meanwhile, in the log scale, a reasonable way is to plot $\log_{10}(x) \pm (1/\ln(10))\,\mathrm{MAD}(x)/x \approx \log_{10}(x) \pm 0.434\,\mathrm{MAD}(x)/x$ (see, e.g., [123]).

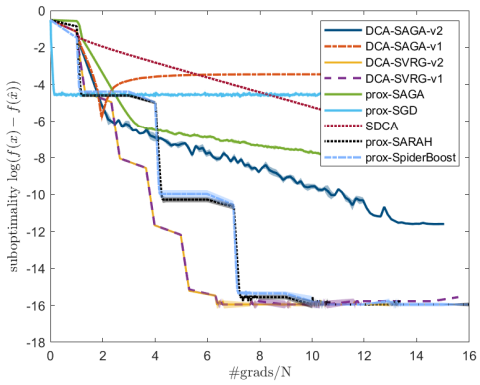
Comments Figure 2.1 shows that two versions of DCA-SVRG, prox-SARAH, and prox-SpiderBoost perform very well and typically take the lead, with suboptimality around 10^{-15} the majority of cases. It is then followed by the sampling without replacement version of DCA-SAGA (DCA-SAGA-v2) which obtains a good suboptimality (usually less than 10^{-10}). While the performance of DCA-SVRG-v1 and



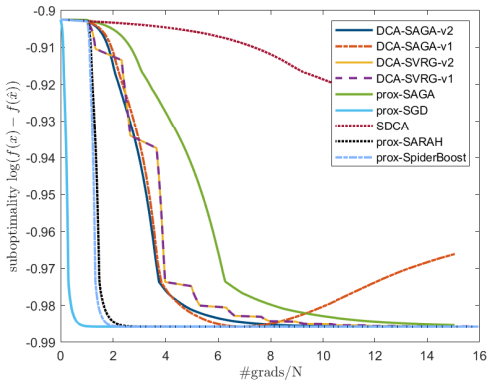
(a) a9a



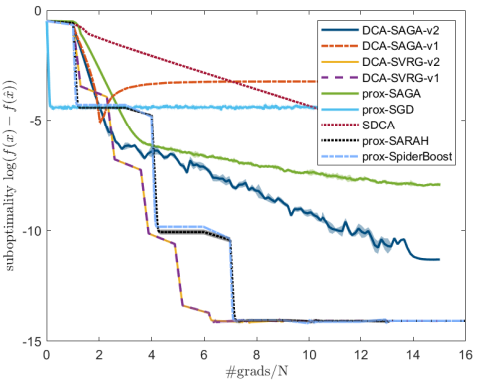
(b) aloi



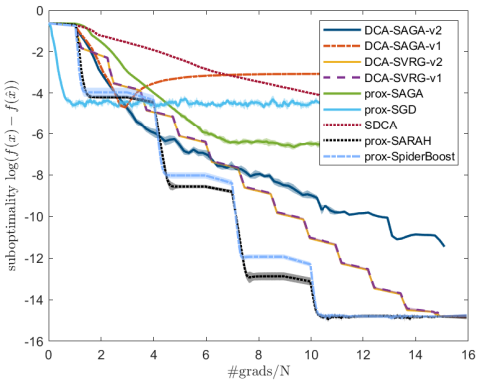
(c) cifar10



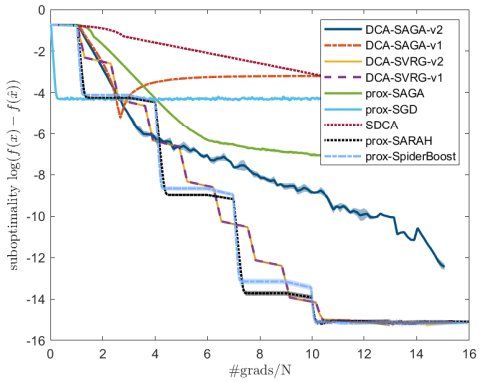
(d) SensIT Vehicle



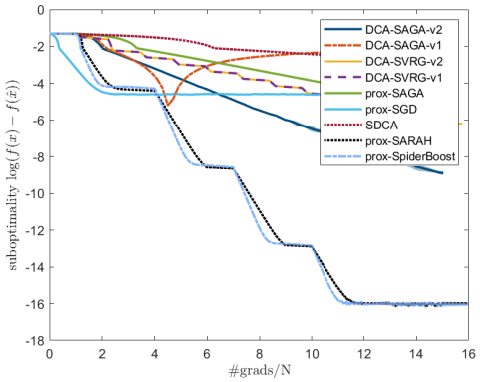
(e) connect-4



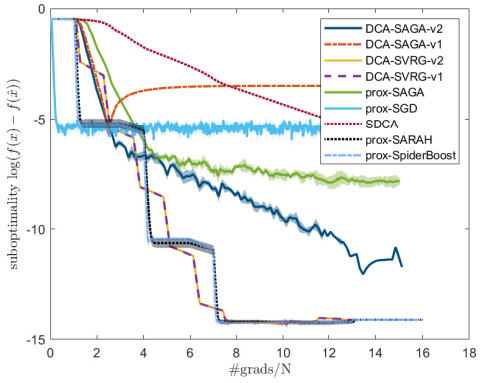
(f) letter



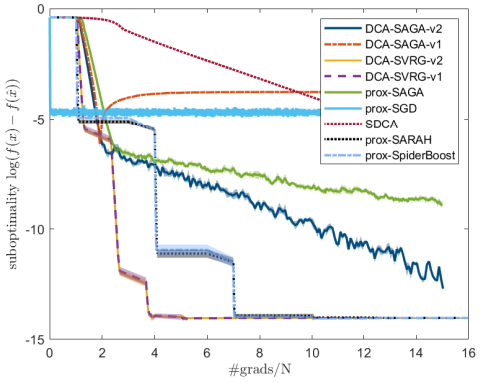
(g) `mnist`



(h) `protein`



(i) `shuttle`



(j) `YearPredictionMSD`

Figure 2.1: The suboptimality of all algorithms over 10 datasets

DCA-SVRG-v2 are nearly identical, DCA-SAGA-v1 performs significantly worse than DCA-SAGA-v2, indicating that sampling strategies are of decisive importance on the performance of DCA-SAGA. In this experiment, the prox-SAGA also obtains good results with suboptimality less than 10^{-5} across most datasets. The SDCA performs marginally worse than prox-SAGA, whereas prox-SGD fluctuates around 10^{-5} and cannot reduce suboptimality any further.

2.5.2 Group Variables Selection in Multi-class Logistic Regression

Logistic regression is without a doubt one of the most successful classification tools. Logistic regression has a wide range of applications including cancer detection [62], social sciences [63], medical [4], etc. For high-dimensional data, on the other hand, feature selection is typically used to select relevant features and encourage weights corresponding to irrelevant features to go to 0.

Let $\{(x_i, y_i) : i = 1, 2, \ldots, N\}$ be a training set with feature vectors x_i and the labels $y_i \in \{1, 2, \ldots, Q\}$ where Q is the number of classes. Let W be the $d \times Q$ matrix with columns $W_{:,1}, W_{:,2}, \ldots, W_{:,Q}$ and $b = (b_1, b_2, \ldots, b_Q)$. The conditional probability of class y given the observation $X = x$ can be modeled by

$$p(Y = y|X = x) = \frac{\exp\left(b_y + W_{:,y}^\top x\right)}{\sum_{k=1}^{Q} \exp\left(b_k + W_{:,k}^\top x\right)}.$$

To find (W, b), one minimizes the following negative log-likelihood function over the whole training set

$$\mathcal{L}(W, b) := \frac{1}{N} \sum_{i=1}^{N} \ell(x_i, y_i, W, b)$$

where $\ell(x_i, y_i, W, b) = -\log p(Y = y_i|X = x_i)$. On the other hand, to select relevant features, one employs the following $\ell_{q,0}$-norm of W

$$\|W\|_{q,0} = |\{j \in \{1, 2, \ldots, d\} : \|W_{j,:}\|_q \neq 0\}|,$$

which leads to the following optimization problem

$$\min_{W,b} \left\{ \frac{1}{N} \sum_{i=1}^{N} \ell(x_i, y_i, W, b) + \lambda \|W\|_{q,0} \right\}. \tag{2.27}$$

In practice, the discontinuous $\ell_{q,0}$-norm is usually approximated by some continuous function. In this work, we approximate $\ell_{q,0}$ with the following nonconvex functions, which have been shown to be efficient in a variety of problems such as individual variable selection in SVM [16], sparse optimal scoring problem [78], sparse covariance matrix estimation problem [105],

$$\begin{aligned} \text{Exponential:} \quad & \eta_\alpha^{\exp}(s) = 1 - \exp(-\alpha s), \\ \text{Capped} - \ell_1 : \quad & \eta_\alpha^{\text{cap}-\ell_1}(s) = \min\{1, \alpha s\}. \end{aligned}$$

The corresponding approximation problem of (2.27) takes the form

$$\min_{W,b}\left\{\frac{1}{N}\sum_{i=1}^{N}\ell(x_i,y_i,W,b)+\lambda\sum_{j=1}^{d}\eta_\alpha(\|W_{j,:}\|_q)\right\}. \tag{2.28}$$

DCA-SVRG and DCA-SAGA applied to (2.28)

It is worth noting that an L-smooth function φ admits the following DC decomposition

$$\varphi(x)=\frac{\gamma}{2}\|x\|^2-\left(\frac{\gamma}{2}\|x\|^2-\varphi(x)\right) \tag{2.29}$$

whenever $\gamma \geq L$. We can show that $\ell(x_i,y_i,W,b)$ is $\frac{2\sqrt{2}}{3\sqrt{3}}\sqrt{Q}(\|x_i\|^2+1)$-smooth, so the DC decomposition (2.29) can be used. On the other hand, η_α is concave and increasing. Therefore, the problem (2.28) takes the form of (Q) with

$$\begin{aligned}&G(W,b)=\frac{\gamma}{2}\|(W,b)\|^2,\ h_i(W,b)=\frac{\gamma}{2}\|(W,b)\|^2-\ell(x_i,y_i,W,b),\\&r_1(W,b)=0, l_i(s)=-\lambda\eta_\alpha(s),\ p_i(W,b)=\|W_{i,:}\|_q, \forall i=\overline{1,d},\end{aligned}$$

where $\gamma \geq \frac{2\sqrt{2}}{3\sqrt{3}}\sqrt{Q}(\max_{i=\overline{1,N}}\|x_i\|^2+1)$. We can apply DCA-SVRG and DCA-SAGA to solve (2.28). In general, the convex subproblem of DCA-SVRG and DCA-SAGA requires the computing of the proximal operator of $r\|\cdot\|_q$ with $q\in\{1,2\}$, where the proximal operator of a function f is defined by

$$\operatorname{prox}_f(x)=\arg\min_y\left\{\frac{1}{2}\|x-y\|^2+f(y)\right\}.$$

It is worth noting that, these proximal operators can be computed explicitly as follows

$$\begin{aligned}\operatorname{prox}_{r\|\cdot\|_2}(x)&=\begin{cases}\left(1-\dfrac{r}{\|x\|_2}\right)x & \text{if } \|x\|_2\geq r,\\ 0 & \text{otherwise}\end{cases}\\ \operatorname{prox}_{r\|\cdot\|_1}(x)&=\max(|x|-r,0)\circ\operatorname{sign}(x).\end{aligned}$$

Therefore, the corresponding DCA-SVRG and DCA-SAGA schemes applied to (2.28) is explicit, i.e., no convex solver is needed for solving convex subproblems.

In this experiment, we found that DCA-SAGA with replacement yields poor performance and is not numerically stable. Hence, we only study the behaviors of three algorithms: DCA-SAGA without replacement, DCA-SVRG-v1 (with replacement), and DCA-SVRG-v2 (without replacement).

Numerical experiments

Datasets We use standard machine learning datasets obtained from LIBSVM for multiclass classification, namely, `connect-4` (67557×126, 3 classes), `dna` (2000×180, 3 classes), `letter` (15000×16, 26 classes), `SensIT Vehicle` (78823×100, 3 classes), `Sensorless` (58509×48, 11 classes), `shuttle` (43500×9, 7 classes). All datasets are normalized by MinMaxScaler to scale all features into the range $[0,1]$.

Comparative algorithms The DCA and the SDCA are chosen as comparative algorithms in this experiment.

Experiment setups and results We set the budget of SFO calls at $20N$. Throughout this experiment, regularization parameter λ and the number α that controls the tightness of the $\ell_{q,0}$ approximation are set to 1 and 0.5, respectively. As mentioned, it can be demonstrated that each function $\ell(x_i, y_i, W, b)$ is L-smooth, where $L = \frac{2\sqrt{2}}{3\sqrt{3}}\sqrt{Q}(\max_{i=\overline{1,N}} \|x_i\|^2 + 1)$. To ensure the convexity of two DC components, we need to set $\gamma \geq L$. In our experiment, we fix $\gamma = 2L$.

By the convergence analysis above and the numerical experiments in [73], we choose hyperparameters for algorithms as follows. The minibatch size b is chosen as $\left\lceil 2\sqrt{N\sqrt{N+1}} \right\rceil$, $\lfloor N^{\frac{2}{3}} \rfloor$, $\lfloor N^{\frac{2}{3}} \rfloor$, $\lfloor 10\% N \rfloor$ for DCA-SAGA, DCA-SVRG-v1, DCA-SVRG-v2, SDCA, respectively. The inner loop length M of DCA-SVRG-v1 and DCA-SVRG-v2 is set to be $\lfloor \frac{1}{4\sqrt{e-1}}\sqrt{b} \rfloor$.

Figures 2.2, 2.3, 2.4, and 2.5 report the mean curves of the objective (averaged over 10 runs) and the mean absolute deviation of five algorithms. To improve visibility, the MAD is scaled up by a factor of 20.

Comments It can be seen that DCA-SAGA consistently outperforms other algorithms in terms of convergence rate and the quality of solutions. The two versions of DCA-SVRG are then considered to be the second-best, where the performances of these two versions are almost identical. Over all tested cases, these patterns are present consistently. The sampling strategies are crucial for the performance of DCA-SAGA in this experiment once more: while sampling with replacement makes DCA-SAGA numerically unstable, sampling without replacement makes DCA-SAGA the best algorithm among all competitors. We can see that the SDCA performs the worst in this experiment because it decreases the objective value in a quite slow rate, partially illuminating the averaging feature's conservative nature of the SAG estimator. The performance of the standard DCA is relatively similar to that of the SDCA. The gain of DCA-SAGA over the SDCA varies from 0.71% up to 60.15% and the gain of DCA-SVRG-v1 over the SDCA ranges from 0.56% to 54.73%.

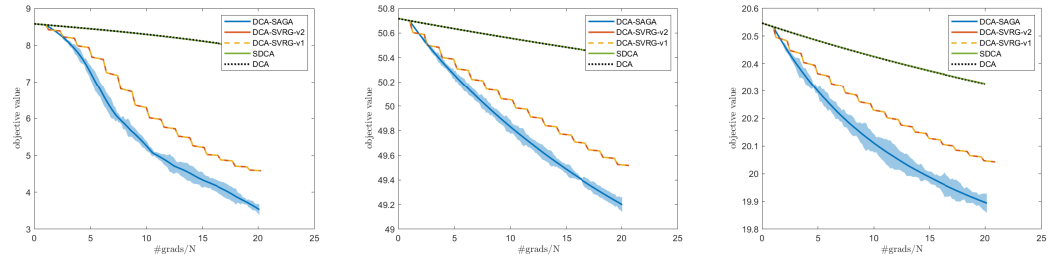
2.5.3 Sparse Linear Regression

Sparse linear regression is a linear regression model that aims to eliminate features that are redundant, noisy, or irrelevant. Given a dataset $\{(x_i, y_i)\}_{i=1}^N$, sparse linear regression attempts to fit $\langle \beta, x_i \rangle \approx y_i$ with a sparse solution β in order to select right features, particularly in the high-dimensional regime. Formally, the optimization problem associated with sparse linear regression is given by

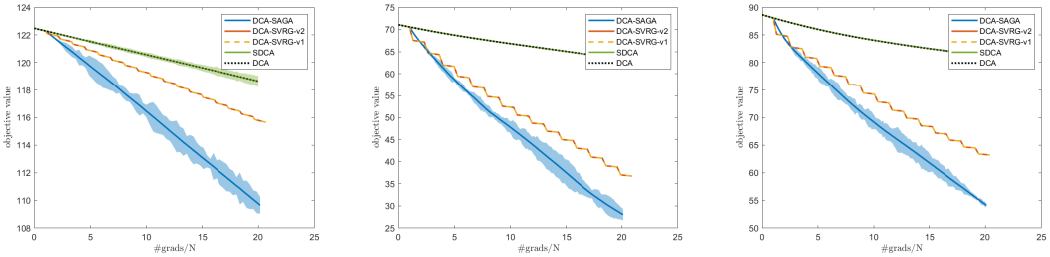
$$\min_{\beta} \frac{1}{2N} \sum_{i=1}^{N} (y_i - \langle \beta, x_i \rangle)^2 + \lambda \|\beta\|_0, \tag{2.30}$$

where $\lambda > 0$ is a parameter that makes a trade-off between the data-fitting term and the sparsity-encouraging term.

As previously discussed, the discontinuous ℓ_0-norm is typically approximated by a continuous one to make the problem tractable. In this section, we focus on the Capped-ℓ_1 that is used to approximate

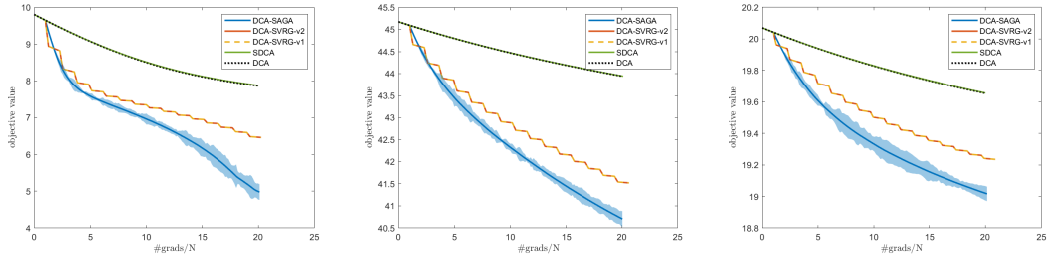


(a) `shuttle` (b) `Sensorless` (c) `letter`

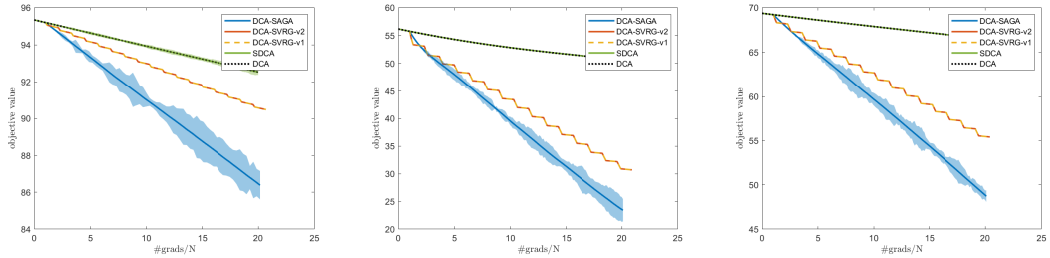


(d) `dna` (e) `SensIT Vehicle` (f) `connect-4`

Figure 2.2: The (averaged) objective value of five algorithms for the case $\eta^{\exp}$ and $q = 1$.

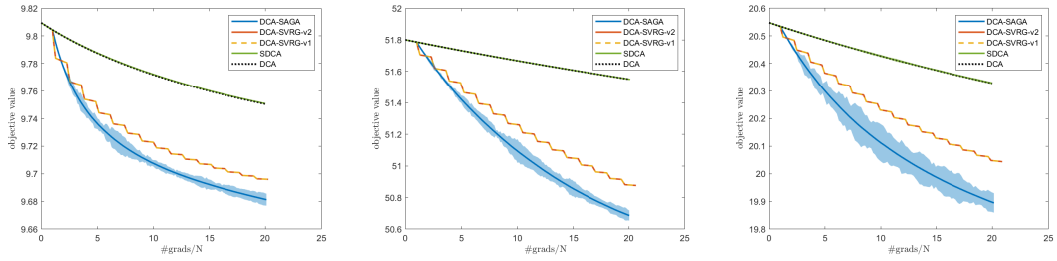


(a) `shuttle` (b) `Sensorless` (c) `letter`

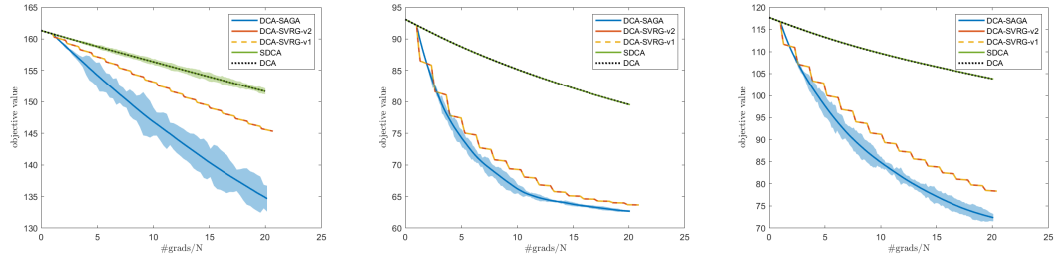


(d) `dna` (e) `SensIT Vehicle` (f) `connect-4`

Figure 2.3: The (averaged) objective value of five algorithms for the case $\eta^{\exp}$ and $q = 2$.

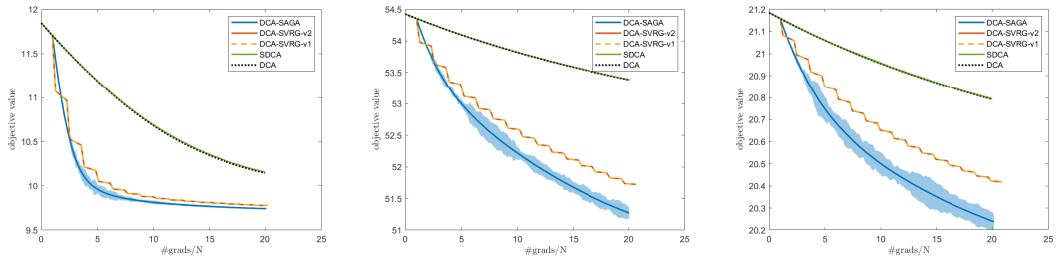


(a) `shuttle` (b) `Sensorless` (c) `letter`

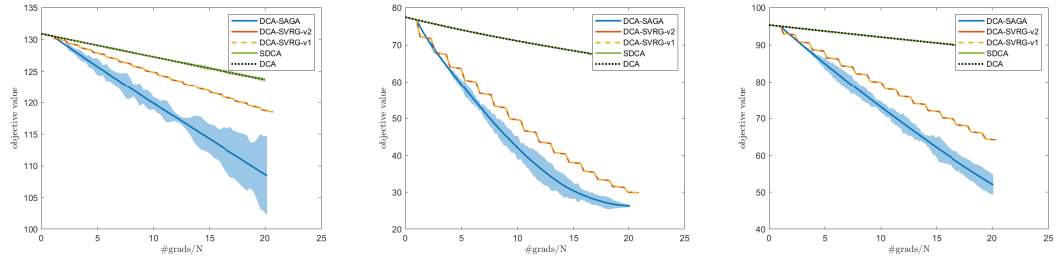


(d) `dna` (e) `SensIT Vehicle` (f) `connect-4`

Figure 2.4: The (averaged) objective value of five algorithms for the case $\eta^{\text{cap}-\ell_1}$ and $q = 1$.



(a) `shuttle` (b) `Sensorless` (c) `letter`



(d) `dna` (e) `SensIT Vehicle` (f) `connect-4`

Figure 2.5: The (averaged) objective value of five algorithms for the case $\eta^{\text{cap}-\ell_1}$ and $q = 2$.

the ℓ_0-norm to give rise to the following approximation problem

$$\min_{\beta} F(\beta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \langle \beta, x_i\rangle)^2 + \lambda \sum_{j=1}^{m} \min(1, \alpha|\beta_j|). \tag{2.31}$$

DCA-SVRG and DCA-SAGA applied to (2.31)

Firstly, each function $\beta \mapsto \frac{1}{2}(y_i - \langle \beta, x_i\rangle)^2$ is $\|x_i\|^2$-smooth, so it admits the DC decomposition (2.29). On the other hand, the Capped-ℓ_1 function is DC with the DC decomposition $\min(1, \alpha|\beta|) = \tilde{g}(\beta) - \tilde{h}(\beta)$, where $\tilde{g}(\beta) = 1 + \alpha|\beta|$, and $\tilde{h}(\beta) = \max(1, \alpha|\beta|)$. Therefore, the problem (2.31) falls into our framework with the following setting

$$\begin{aligned} G(\beta) &= \frac{\gamma}{2}\|\beta\|^2, h_i(\beta) = \frac{\gamma}{2}\|\beta\|^2 - \frac{1}{2}\langle \beta, x_i\rangle^2 + y_i\langle \beta, x_i\rangle, \\ r_1(\beta) &= \lambda\alpha\|\beta\|_1, r_2(\beta) = \lambda \sum_{j=1}^{m} \max(1, \alpha|\beta_j|), \end{aligned}$$

where $\gamma \geq \max_{i=\overline{1,N}} \|x_i\|^2$. When applying DCA-SAGA and DCA-SVRG to this setting, the subproblem is nothing but the proximal operator of ℓ_1, which has a closed-form solution.

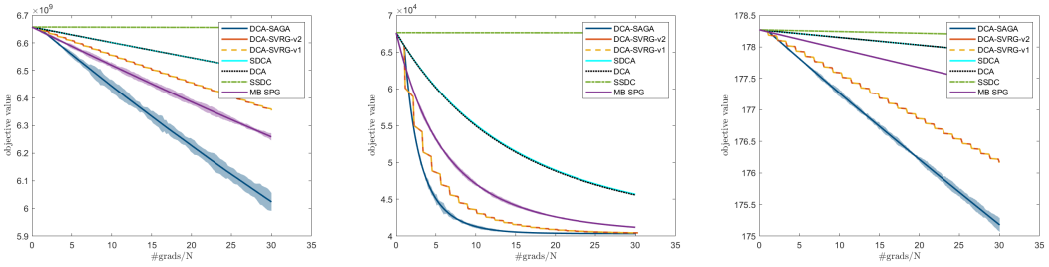
On the other hand, we discovered that DCA-SAGA with replacement performs poorly and is numerically unstable in this experiment, which is a phenomenon that also occurred in the previous experiments on group variables selection in multi-class logistic regression. As a result, we only take into account DCA-SAGA without replacement, DCA-SVRG-v1, and DCA-SVRG-v2.

Numerical experiments

Datasets We use regression datasets, namely, `cadata` (20640×8), `YearPredictionMSD` (463715×90) from LIBSVM, and SGEMM GPU kernel performance dataset, `sgemm_product` (241600 × 14) [96]. Each dataset is standardized to have mean (μ) of 0 and standard deviation (σ) of 1.

Comparative algorithms We compare our proposed algorithms with the standard DCA, the SDCA, the stochastic stagewise DC (SSDC) proposed in [134], and the mini-batch stochastic proximal gradient (MB-SPG) [133].

Experiment setups and results The budget of SFO calls is set to be $30N$. The regularization parameter λ and the parameter α of the Capped-ℓ_1 are both set to be 1 throughout this experiment. To guarantee the convexity of DC components, we choose $\gamma = \max_{i=\overline{1,N}} \|x_i\|^2$. On the other hand, the minibatch size b is chosen as $\lfloor N^{\frac{2}{3}} \rfloor$, $\lfloor N^{\frac{2}{3}} \rfloor$, $\lceil 2\sqrt{2}\sqrt{N\sqrt{N+1}} \rceil$, $\lfloor 10\%N \rfloor$ for DCA-SVRG-v1, DCA-SVRG-v2, DCA-SAGA, and SDCA, respectively. For two versions of DCA-SVRG, we set $M = \lfloor \frac{1}{8\sqrt{e-1}}\sqrt{b} \rfloor$. About the SSDC, the nonconvex regularizer Capped-ℓ_1 is replaced by the Monreau envelope r_μ [134, Sect. 4]. Moreover, the proximal operator of the Capped-ℓ_1 can be efficiently computed (see, e.g., [51]), which allows the SSDC to work. The parameter μ controlling the Monreau envelope is required to match the (square) order of the final error ϵ [134, Theorem 10c], so it is set to be a relatively small number, $\mu = 10^{-6}$. Note that, since the convex subproblem has a closed-form solution and the first DC component is not given as a large sum, no stochastic convex solver is required for the SSDC in this case.



(a) `cadata` (b) `sgemm_product` (c) `YearPredictionMSD`

Figure 2.6: The performance of seven algorithms on the sparse linear regression problem

On the other hand, as remarked in [133], the use of the Moreau envelope could be a bad idea as it introduces the approximation error while slowing down the convergence. Therefore, we further implement the mini-batch stochastic proximal gradient (MB-SPG) [133] that uses directly the proximal operator of the Capped-ℓ_1. The minibatch size of MB-SPG is of order $\mathcal{O}(1/\epsilon^2)$ where ϵ is the expected final error, so it is proper to set the minibatch size at $\lfloor 10\%N \rfloor$. The constant c (see [133]) is in $(0, 0.5)$, so we set it at 0.25 which is a neutral number between the two extremes; Meanwhile, as discussed earlier, the Lipschitz smoothness parameter can be chosen as $\max_{i=\overline{1,N}} \|x_i\|^2$.

We report the mean curve and the mean absolute deviation (over 10 runs) of the objective value of five comparative algorithms in Figure 2.6, where the mean absolute deviation is scaled up by a factor of 20 for visibility.

Comments It is observed that, once again, DCA-SAGA is the best algorithm over all tested cases where it decreases the objective value in a fast rate to obtain good solutions. Meanwhile, the performances of DCA-SVRG-v1 and DCA-SVRG-v2 are almost identical and are the second best on `sgemm_product` and `YearPredictionMSD` datasets. On the other hand, the MB-SPG is the second best on the `cadata` dataset and pushes the DCA-SVRGs to the third place in this case. The gain of DCA-SAGA over DCA-SVRG-v1 are 5.22%, 0.24%, and 0.56% on `cadata`, `sgemm_product`, and `YearPredictionMSD`, respectively. On the other hand, the graphs of DCA and SDCA are also almost the same on all three datasets. The gain of DCA-SVRG-v1 over SDCA on `cadata`, `sgemm_product`, and `YearPredictionMSD` are 2.11%, 11.45%, and 0.98%, respectively. Finally, in this experiment, while sampling with replacement makes DCA-SAGA numerically unstable, sampling without replacement consistently makes DCA-SAGA the best algorithm over all considered tested cases. Finally, the SSDC performs quite poorly in this experiment. Although it still manages to decrease the objective function, this decrease is insufficient to be visible in the decreasing magnitude of other comparative algorithms. This can be attributed to the slow convergence caused by the use of the Moreau envelope.

2.6 Conclusion

In this chapter, we introduced two stochastic DCA methods with integrated variance reduction techniques, DCA-SVRG and DCA-SAGA, for tackling a wide class of nonconvex DC problems, including the large-sum case, where both the data-fitting and regularization terms are given as (nonsmooth) DC functions. The DCA framework addresses the DC structure, while the unbiased estimators SVRG and

SAGA address the large-sum structure. In this study, the benefits of DC programming and DCA are multiple.

DC programming and DCA give a broad overview of the problems under consideration and advise on how to use the structural information in these problems. They create a flexible deterministic frame around which stochastic approaches are developed to deal with the stochastically problematic aspects of the problems. Our proposed methods inherit both the advantages and the inherent limitations of the stochastic estimators used. The DCA-SVRG is an epoch-based algorithm with the benefit of requiring just one gradient vector in memory to produce the stochastic variance reduction term. However, the pivot point $\tilde{x}^k$ is not changed within its corresponding epoch, which appears to be a flaw of this approach. Indeed, when the points x_j^{k+1} in an epoch proceed far away from its pivot $\tilde{x}^k$, $\nabla h_i(x_j^{k+1})$ and $\nabla h_i(\tilde{x}^k)$ (i is a random index) are no longer highly correlated, undermining the effect of the variance reduction term. In contrast, the progressive updating of the variance reduction term across iterations allows DCA-SAGA to partially address this issue. However, DCA-SAGA has a limitation of its own, which is the memory need for storing a table of gradients. Although one can retain a vector of scalars only in some specific problems, this issue persists generally and is the algorithm's bottleneck. We have observed that a biased estimator named SARAH [94] can potentially address both of the drawbacks of SVRG and SAGA: it progressively updates the variance reduction term within each epoch and does not need the storage of the gradient table. This observation stimulates our future work on the DCA-SARAH combination. Additionally, we have researched both sampling with and without replacement in the step of selecting a subset of the large sum. The independence of the selected elements is enjoyed by sampling with replacement, making convergence analysis easier. The latter strategy, however, is more widely used and prevents some low-probability bad events that occur in sampling with replacement (high repetition leading to bad approximations). In our numerical experiments on three problems, the results of DCA-SAGA with the two sampling options are considerably different from one another: sampling without replacement makes DCA-SAGA more stable and makes it the best method in two of the studied problems. On the other hand, DCA-SVRG's performances using the two sampling options are almost identical.

From a theoretical standpoint, we establish the almost sure convergence of the proposed algorithms to DC critical points. Furthermore, for the class of problem (P) where r_2 in the regularizer $r(x) = r_1(x) - r_2(x)$ has a composite form, we provide additional convergence analysis to facilitate parameters selection in practice. By exploiting the special composite structure of r_2, we reformulate this problem as a DC program. We then provide new relationships between the parameters of the proposed algorithms (minibatch size, inner-loop length) and the parameters of the problems (strong convexity, Lipschitz smoothness, dataset size), which ensures the almost sure convergence property to DC critical points. The key benefit of the extra analysis is that it enables more relaxing convergence criteria than those that would be obtained by just applying the full previous convergence analysis to the reformulated problem.

Overall, the proposed algorithms appear to outperform several state-of-the-art stochastic methods for large-sum nonconvex problems in three experiments. These experiments also demonstrate the advantages of the SAGA and SVRG estimators within the proposed algorithms, as well as the limitations of the SAG estimator used in SDCA.

Chapter 3

Online Stochastic DCA

Abstract. In this chapter, we propose stochastic DCA schemes for solving stochastic DC programs whose involved random variable can have an arbitrary distribution, with the focus on handling streaming data. The context of streaming data is pervasive in real-world applications, necessitating a fast and scalable approach. The proposed algorithms' convergence is thoroughly investigated using tools from modern convex analysis and martingale theory. Finally, we study several aspects of the proposed algorithms on an important machine learning problem: the expected problem in Principal Component Analysis.

3.1 Introduction

Stochastic optimization is a means for resolving many challenges in the age of big data, such as data processing, storage bottlenecks, noisy measurements, high dimensionality, and so on. A typical stochastic optimization problem takes the form

$$\min\{F(w) := \mathbb{E}(f(w, Z)) : w \in S \subset \mathbb{R}^m\}, \tag{3.1}$$

where Z is a random vector with unknown distribution in general, and f is a loss function defined on $\mathbb{R}^m$. There is an extensive literature studying this problem in the convex setting, i.e. when $f(\cdot, Z)$ is convex for all Z, and S is a convex set. When either $f(\cdot, Z)$ or S is nonconvex (resp. $f(\cdot, Z)$ is nonsmooth), (3.1) is a nonconvex (resp. nonsmooth) stochastic optimization problem. Similar to deterministic optimization, the transition from convex to nonconvex settings presents challenges. Nonsmooth and nonconvex stochastic optimization, which is significantly more difficult than convex and/or smooth optimization, plays a pivotal role in a variety of practical applications. Nevertheless, techniques for nonsmooth and nonconvex stochastic optimization are still limited.

The material of this chapter is extracted from the following works:
H. A. Le Thi, H. P. H. Luu, & T. Pham Dinh (2022). Online Stochastic DCA with applications to Principal Component Analysis. **Accepted for publication**, *IEEE Transactions on Neural Networks and Learning Systems*. arXiv:2108.02300.

H. A. Le Thi, T. Pham Dinh, H. P. H. Luu, & H. M. Le, Deterministic and stochastic DCA for DC programming, *Handbook of Engineering Statistics 2nd edition*, In press, Springer.

In this chapter, we are concerned with stochastic DC programs, the major class of nonconvex stochastic optimization, which have the form of (3.1) with f being a DC function,

$$\min\{F(w) = \mathbb{E}(g(w, Z) - h(w, Z)) : w \in S\}, \tag{3.2}$$

where Z is a random vector determined in some complete probability space $(\Omega, \mathcal{M}, \mathbb{P})$ ($Z : \Omega \to \mathbb{R}^n$), while $S \subset \mathbb{R}^m$ is a nonempty, compact, and convex set, $g(\cdot, Z)$ and $h(\cdot, Z)$ are lower semi-continuous convex functions for all Z, satisfying some technical conditions described later. The framework of the problem (3.2) is very general in two respects. Firstly, the underlying distribution of Z is arbitrary, which makes it capable of treating any random variable involved. As a special case, when Z is uniformly distributed over a finite set $\{z_1, z_2, \ldots, z_N\}$, we obtain a large-sum problem which has been addressed in Chapter 2,

$$\min_{w \in S} \left\{ F(w) = \frac{1}{N} \sum_{i=1}^{N} g(w, z_i) - \frac{1}{N} \sum_{i=1}^{N} h(w, z_i) \right\}. \tag{3.3}$$

Note that, as mentioned in Chapter 2, for an arbitrary random variable Z, if $z_1, z_2, \ldots, z_N$ are i.i.d. realizations of Z, the problem (3.3) is considered as the ERM problem of (3.2).

Secondly, in (3.2), g and h are possibly nonsmooth, resulting in a very large class of stochastic nonsmooth, nonconvex DC programs that encompasses the vast majority of real-world problems [76].

Our research is primarily inspired by the role of stochastic DC programs in machine learning, as well as new issues emerging from optimization problems in this field. To begin, let us note that machine learning is a mine of DC programs, and that many machine learning problems can be expressed in the form of the stochastic DC program (3.2). Indeed, formally, a machine learning task results in an optimization problem that involves minimizing a loss function $f(w, Z)$ on a training set of samples, measuring the goodness of the prediction model parameterized by w using the data point Z. A regularizer is frequently added to the loss to promote certain types of solution structure, such as sparsity or low rank. When the loss function and the regularizer are both DC, this learning problem is a DC program. A typical instance of DC regularizers arises in learning with sparsity where the zero-norm (the zero-norm of a vector is the number of its nonzero components) is used to model sparsity. The main approach to deal with the zero-norm is nonconvex approximation, in which the zero-norm is replaced by its nonconvex approximation functions, and it is demonstrated in [77] that all existing nonconvex approximations (e.g., capped ℓ_1, log-sum penalty, minimax concave penalty, etc.) are DC. There are various examples of DC loss functions. In supervised learning, for example, classification or regression problems such as robust logistic regression [99], truncated linear regression [22], robust support vector machines [141, 27], robust support vector regression [127], phase retrieval [30], the nonconvex subproblem of Uzawa Algorithm for Neyman-Pearson classification [46], or again neural network with ReLU activations [29] (combined with popularly used losses including cross-entropy, Huber loss, square error, LogCosh loss), etc. Several other unsupervised/semi-supervised learning problems possessing DC structures are as follows: Principal Component Analysis [90], t-Distributed Stochastic Neighbor Embedding [74], Positive Unlabeled learning [134], Semi-supervised support vector machines [66], etc.
If we consider the preceding learning problems in the context of stochastic data, i.e., Z is a random vector, then we have to minimize the expectation loss $F(w) = \mathbb{E}(f(w, Z))$ (and possibly with a regu-

larizer), and this optimization problem takes the form of (3.2).

Our contributions. To address the primary difficulty of the problem (3.2), which is its nonconvexity, we study the DCA approach. Moreover, DCA is also introduced in an online manner to handle streaming data. Our contributions are as follows.

First, we develop a general online stochastic DCA scheme for solving the problem (3.2) (where the problem setting will be described in more detail in section 3.3). We combine DCA with Stochastic Approximations (SA) in the online manner: at each iteration, new data is used to construct a stochastic approximation of G and the one of a subgradient of H, and the solution is updated via the DCA's principle. Since the update steps of the proposed algorithms require *new fresh samples* from the distribution of Z, we refer to our algorithm as *online stochastic DCA* (abbreviated osDCA). The proposed osDCA enjoyes a double benefit from being an online algorithm: it is suitable to perform *streaming data* coming from an *unknown distribution*.

Second, we propose two additional osDCA schemes for situations in which we can compute directly (i.e. the SA procedure is not used) the function $G(w) := \mathbb{E}(g(w, Z))$ (the second osDCA) or $H(w) := \mathbb{E}(h(w, Z))$ (the third osDCA). These two algorithms have some advantages over the first osDCA as they seek to use the additional information of data to make better updates (which are less noisy than the first one) at the expense of computing the exact expectations. Nevertheless, while the first osDCA can be applied to any data, the latter two usually necessitate extra knowledge about Z. A typical case is that the distribution of some partial measurement of Z, $\mu(Z)$, is known and either $g(w, \cdot)$ or $h(w, \cdot)$ is a function with respect to μ (see Remark 6).

Third, we rigorously study the convergence analysis of the three proposed methods. For the first algorithm, we establish the subsequential convergence to critical points with probability one. In addition, the proposed algorithm and its convergence analysis can be extended to a more general setting which is $F(w) = \mathbb{E}(g(w, Z) - h(w, \tilde{Z}))$, where Z and $\tilde{Z}$ are two different random vectors. The extension aims to address optimization problems involving with two parallel streams of data. For the second and the third algorithms, the subsequent convergence to DC critical points is also established with milder conditions than those of the first osDCA scheme.

Fourth, we study the practical performance of the proposed algorithms via the expected problem of principal component analysis. Numerical experiments have been conducted carefully on a variety of real-world as well as synthetic datasets to study the proposed algorithms' behaviors in different respects.

3.2 Related works

Stochastic optimization has been studied thoroughly for convex problems since the seminal work [109], where the well-known Stochastic Gradient Descent (SGD) was introduced, which really opened a door in numerical optimization for large-scale problems [13, 12]. In nonconvex setting, stochastic algorithms remain rare, and most of them require the objective to be smooth or partially smooth (some components of the objective are smooth). Overall, there are four approaches to tackle nonconvex stochastic problems. The first, inspired by the aforementioned SGD, includes stochastic (proximal) (sub)gradient-based methods designed primarily for smooth or weakly convex objective functions [48, 9, 30]. The second is stochastic MM (Majorization-Minimization) for partially smooth objective [84, 108], which minimizes a stochastic convex surrogate iteratively to obtain an updated optimization variable. Here the authors con-

sidered either smooth error function or smooth surrogate, which implies - in case of DC programs - one DC component should be smooth. The third is stochastic Successive Convex Approximation [136, 135] (mainly for smooth objective functions) that is similar to stochastic MM where the sequence of approximation functions are convex but not necessary upper bounds of sample objective functions. The fourth is stochastic DCA for stochastic DC programs - a substantially large class to cover almost all real-world nonconvex optimization problems [76]. Initial works in this approach include [72, 73, 82, 95, 134] that consider some special classes of DC problems such as large-sum and/or (partially) smooth, as well as [70] working on a very general class of stochastic nonsmooth DC programs. To extend beyond the DC programming framework, [88] used Moreau envelope which is a DC function to approximate a nonsmooth, nonconvex regularizer, and then developed a stochastic DCA for solving the resulting problem. It should be noted that, as indicated in [76], while the (stochastic) MM proposes a general idea to majorize the objective function, (stochastic) DCA gives the simplest and the most closed convex surrogate thanks to DC structures of the objective. Furthermore, usual choices of surrogates of MM result in DCA versions [76].

So far, there are very few algorithms for the general setting (3.2). To our knowledge, the paper [70] is the first work dealing with (3.2) where both DC components are nonsmooth. In that article, several stochastic DCA (SDCA) schemes are proposed in the aggregated update style. That is, all past information (sample realizations) is used to construct subproblems. These algorithms therefore need to store all samples in the computer memory during the computational process. In the present work, we investigate online stochastic DCA for the general problem (3.2) to deal with fast streaming data where we do not need to store samples all the time. Furthermore, thanks to the online mechanism, our proposed algorithms have the adaptive ability which is a great advantage over the SDCA schemes proposed in [70]. Numerical experiments justify this claim.

3.3 Proposed methods

This section develops osDCA schemes for solving the problem (3.2) which can be described as follows.

3.3.1 Problem setting

Let P_Z be the probability distribution of Z on $\mathbb{R}^n$ and $\Xi = \operatorname{supp}(P_Z)$ be the support of P_Z. By definition, a point $x \in \mathbb{R}^n$ is in $\operatorname{supp}(P_Z)$ if $P_Z(N_x) > 0$, for all neighborhood N_x of x. Since a measure "lives" in its support, we only need to work in Ξ instead of $\mathbb{R}^n$. For instance, a Dirac measure δ_a concentrating at a single point a admits a support containing only one point a; a discrete measure $\mu = \sum_{i=1}^{\infty} \beta_i \delta_{a_i}$ with $\beta_i > 0$ admits a support $\{a_1, a_2, \ldots\}$. A basic property of Ξ is that it is closed in $\mathbb{R}^n$. Moreover, $P_Z(\Xi^{\complement}) = 0$ since $\mathbb{R}^n$ is the topological Hausdoff space and P_Z is a Radon measure in $\mathbb{R}^n$. Therefore, only the values of g and h on $S \times \Xi$ matter. For simplicity of presentation, we assume that $\operatorname{dom} g = \operatorname{dom} h = S \times \Xi$. That is, the value of g and h outside $S \times \Xi$ is set to $+\infty$. This implies that $\operatorname{dom} G = \operatorname{dom} H = S$ (recall that $G(w) := \mathbb{E}(g(w, Z))$ and $H(w) := \mathbb{E}(h(w, Z))$). Here we use the convention $+\infty - (+\infty) = +\infty$. Moreover, g and h are assumed to be bounded below and Borel measurable, and $g(w, Z), h(w, Z)$ are integrable for all $w \in S$. It is noted that the Borel σ-algebra on $\mathbb{R} \cup \{+\infty\}$ is generated by the order topology of $\mathbb{R} \cup \{+\infty\}$. As $g(\cdot, z)$ and $h(\cdot, z)$ are convex, lower semi-continuous for all $z \in \Xi$, so are G and H. In the sequel we will consider the problem (3.2) in its

DC form

$$\min\{F(w) = G(w) - H(w) : w \in S\}. \tag{3.4}$$

Moreover, we need some mild assumptions as follows.

Assumption 1. *i. For all* $z \in \Xi$, $\operatorname{dom} \partial h(\cdot, z) = S$.
ii. $\bar{\rho} := \rho_H + \inf_{z\in\Xi} \rho(g(\cdot, z)) > 0$.
iii. There exists a Borel measurable selector τ *such that*

$$\forall w \in S, z \in \Xi, \tau(w, z) \in \partial_w h(w, z),$$

where τ *is* L^2 *uniformly bounded in the sense that there exists a Borel measurable function* $\tilde{\tau}$ *such that* $\tilde{\tau}(Z)^2$ *is integrable and* $\forall w \in S, z \in \Xi, \|\tau(w, z)\| \le \tilde{\tau}(z)$.
iv. $\sup_{w\in S} |F(w)| < +\infty$.

Remark 4. *It is observed that the assumptions i), iii) and iv) are mild. On another hand, thanks to the regularization technique introduced in [103], Assumption 1(ii) is easily fulfilled by adding an* L_2 *regularizer to both DC components.*

Next, to measure the complexity of a class of functions, we adopt the following notion of Rademacher average (or Rademacher complexity), see, e.g. [42].

Rademacher average/complexity
For a set of points $\{z_1, z_2, \ldots, z_l\} := z^l$ in Ξ, the Rademacher average $R_l(g, z^l)$ is defined as

$$R_l(g, z^l) = \mathbb{E}_\sigma \sup_{w\in S} \left| \frac{1}{l} \sum_{i=1}^{l} \sigma_i g(w, z_i) \right|,$$

where $\sigma_i' s$ are i.i.d. random numbers such that $\sigma_i \in \{\pm 1\}$ with $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = 1/2$.
The Rademacher average of a family of functions $\{g(\cdot, z) : z \in \Xi\}$, denoted by $R_l(g, \Xi)$, is defined as

$$R_l(g, \Xi) = \sup_{z_1\in\Xi, z_2\in\Xi, \ldots, z_l\in\Xi} R_l(g, z^l).$$

Assumption 2. *i. There exists a Borel measurable function* $\tilde{g} : \mathbb{R}^n \to \mathbb{R}$ *such that* $\tilde{g}(Z)$ *is integrable and*

$$|g(w, z)| \le \tilde{g}(z), \ \forall w \in S, z \in \Xi.$$

ii. $R_k(g, \Xi) \le N_g/k^\alpha$ *with* $N_g > 0$ *and* $\alpha > 0$.

It is noteworthy that Assumption 2(ii) holds for various cases described as follows [42].
Case 1. Holder functions $g(\cdot, z), z \in \Xi$: Let D be the length of a cube in $\mathbb{R}^m$ containing the compact set S. Suppose that $\exists M, L > 0$ and $\gamma \in (0, 1]$ such that

1. $|g(w, z)| \le M, \forall w \in S, z \in \Xi$,

2. $|g(x, z) - g(y, z)| \le L\|x - y\|^\gamma, \forall x, y \in S, z \in \Xi$.

Then, for any $\alpha \in (0, 1/2)$, $R_k(g, \Xi) \leq N_g/k^\alpha$, where

$$N_g = LD^\gamma m^{\frac{\gamma}{2}} + \frac{M\sqrt{m}}{\sqrt{\gamma(1-2\alpha)e}}.$$

Case 2. Holder functions $g(w, \cdot), w \in S$: Suppose that Ξ is compact, let D be the length of a cube in $\mathbb{R}^n$ that contains Ξ. Suppose that there exists $M, L, \gamma > 0$ such that
1. $|g(w, z)| \leq M, \forall w \in S, z \in \Xi$.
2. $|g(w, u) - g(w, v)| \leq L\|u - v\|^\gamma, \forall w \in S, u, v \in \Xi$.
Then $R_k(g, \Xi) \leq N_g/k^\alpha$ where $N_g = M + LD^\gamma n^{\frac{\gamma}{2}}$ and $\alpha = \gamma/(2\gamma + n)$.
Case 3. Discrete set Ξ: Suppose that the number of elements of Ξ is finite, say $|\Xi| = N_\Xi$. Furthermore, assume that there exists $M > 0$ such that $|g(w, z)| \leq M,\ \forall w \in S, z \in \Xi$. Then, $R_k(g, \Xi) \leq M\sqrt{N_\Xi/k}$, hence, $\alpha = 1/2$.

It turns out that Assumption 2 is not strong; hence a class of functions meeting the criteria is wide to cover many problems arising in practice. In three cases of Rademacher complexity presented above, though α in case 2 can be very small in the high-dimension regime, which makes our next algorithm impractical, the other two cases have $\alpha = 1/2$ or arbitrarily near to $1/2$, which are appropriate sample rates in practice. It should be stressed that the Rademacher complexity measures the richness of a class of functions. Therefore, roughly speaking, the function g must be quite "simple" in this Rademacher sense. This criterion naturally fulfills our demand to control the variability of stochastic approximations made on g.

3.3.2 Online Stochastic DCA schemes

We introduce an osDCA scheme described in Algorithm 4.

Algorithm 4 Online Stochastic DCA

Initialization: Choose $w^0 \in S$ and a sequence of sample sizes $\{n_k\}$, set $k = 0$.
repeat

1. Draw independently n_k samples $Z_{k,1}, \ldots, Z_{k,n_k}$ from the distribution of Z in such a way that they are also independent of the past.

2. Compute $t^k = \dfrac{1}{n_k} \sum_{i=1}^{n_k} \tau(w^k, Z_{k,i})$.

3. Compute w^{k+1}, an optimal solution to the problem

$$\min \left\{ \frac{1}{n_k} \sum_{i=1}^{n_k} g(w, Z_{k,i}) - \langle t^k, w \rangle : w \in \mathbb{R}^m \right\}. \tag{3.5}$$

4. Set $k = k + 1$.

until Stopping criterion.

Algorithm 4 is well defined with probability 1. To be more specific, the set of events that makes Algorithm 4 work is $\mathcal{V} = \cap_{k=1}^\infty \cap_{i=1}^{n_k} (Z_{k,i} \in \Xi)$ and hence $\mathbb{P}(\mathcal{V}) = 1$. We denote $Z_k = Z_{k,1:n_k}$ and $\mathcal{P}_k = \sigma(Z_0, Z_1, \ldots, Z_{k-1}, w^0, w^1, \ldots, w^k)$ which is the σ-algebra generated by associated random variables. We observe that $\{w^k\}_{k=0}^\infty$ is a predictable process and $\{t^k\}_{k=0}^\infty$ is an adapted process with respect to the filtration $\{\mathcal{P}_{k+1}\}_{k=0}^\infty$. The convergence results of Algorithm 4 are presented in Theorem 10.

Theorem 10. *Under Assumptions 1 and 2, let $\beta = \min\{\alpha, 1\}$, if the sequence of sample sizes $\{n_k\}$ satisfies $\sum_{k=1}^{\infty} n_k^{-\beta} < +\infty$, the iterations of Algorithm 4 satisfy:*

1. There exists F^∞ integrable such that $F(w^k) \to F^\infty$ a.s.

2. $\sum_{k=1}^{\infty} \|w^{k+1} - w^k\|^2 < +\infty$ a.s.

3. There exists a measurable set $\mathcal{L} \subset \Omega$ with $\mathbb{P}(\mathcal{L}) = 1$ such that for each $\omega \in \mathcal{L}$, every limit point of $\{w^k(\omega)\}$ is a critical point of $F = G - H$.

Proof. 1. Let $\nu(w) := \mathbb{E}(\tau(w, Z))$. It follows from the definition of the selector τ that: for every $w \in S$,

$$h(w', Z) \geq h(w, Z) + \langle \tau(w, Z), w' - w \rangle, \quad \forall w' \in \mathbb{R}^m. \tag{3.6}$$

By taking expectation on both sides of (3.6), we obtain

$$H(w') \geq H(w) + \langle \nu(w), w' - w \rangle, \quad \forall w' \in \mathbb{R}^m,$$

or $\nu(w) \in \partial H(w)$. Since $\nu(w^k) \in \partial H(w^k)$,

$$H(w^{k+1}) \geq H(w^k) + \langle \nu(w^k), w^{k+1} - w^k \rangle + \frac{\rho_H}{2} \|w^{k+1} - w^k\|^2. \tag{3.7}$$

On the other hand, it follows from definition of w^{k+1} that

$$\begin{aligned} \frac{1}{n_k} \sum_{i=1}^{n_k} g(w^k, Z_{k,i}) \geq & \frac{1}{n_k} \sum_{i=1}^{n_k} g(w^{k+1}, Z_{k,i}) + \langle t^k, w^k - w^{k+1} \rangle \\ & + \frac{\inf_{z \in \Xi} \rho(g(\cdot, z))}{2} \|w^{k+1} - w^k\|^2. \end{aligned} \tag{3.8}$$

From (3.7) and (3.8), we obtain

$$\begin{aligned} & \frac{1}{n_k} \textstyle\sum_{i=1}^{n_k} g(w^{k+1}, Z_{k,i}) \leq H(w^{k+1}) + \frac{1}{n_k} \sum_{i=1}^{n_k} g(w^k, Z_{k,i}) \\ & -H(w^k) - \frac{\bar{\rho}}{2} \|w^{k+1} - w^k\|^2 + \langle t^k - \nu(w^k), w^{k+1} - w^k \rangle, \end{aligned} \tag{3.9}$$

with $\bar{\rho} = \rho_H + \inf_{z \in \Xi} \rho(g(\cdot, z))$. By taking conditional expectation with respect to $\mathcal{P}_k$ both sides of (3.9), we obtain

$$\begin{aligned} & \mathbb{E}(F(w^{k+1}) - F(w^k) | \mathcal{P}_k) \leq \mathbb{E}(\langle t^k - \nu(w^k), w^{k+1} - w^k \rangle | \mathcal{P}_k) \\ & + \mathbb{E}\left(G(w^{k+1}) - \frac{1}{n_k} \sum_{i=1}^{n_k} g(w^{k+1}, Z_{k,i}) | \mathcal{P}_k \right) \\ & - \frac{\bar{\rho}}{2} \mathbb{E}(\|w^{k+1} - w^k\|^2 | \mathcal{P}_k). \end{aligned} \tag{3.10}$$

By applying Schwartz inequality and Holder inequality,

$$\begin{aligned} & \mathbb{E}(\langle t^k - \nu(w^k), w^{k+1} - w^k \rangle | \mathcal{P}_k) \\ & \leq \mathbb{E}(\|t^k - \nu(w^k)\|^2 | \mathcal{P}_k)^{\frac{1}{2}} \mathbb{E}(\|w^{k+1} - w^k\|^2 | \mathcal{P}_k)^{\frac{1}{2}}. \end{aligned} \tag{3.11}$$

By using AM-GM inequality, we obtain

$$\begin{aligned}&\mathbb{E}(\|t^k-\nu(w^k)\|^2|\mathcal{P}_k)^{\frac{1}{2}}\mathbb{E}(\|w^{k+1}-w^k\|^2|\mathcal{P}_k)^{\frac{1}{2}}\\&\leq\frac{1}{2\bar{\rho}}\mathbb{E}(\|t^k-\nu(w^k)\|^2|\mathcal{P}_k)+\frac{\bar{\rho}}{2}\mathbb{E}(\|w^{k+1}-w^k\|^2|\mathcal{P}_k).\end{aligned}\tag{3.12}$$

It follows from the independence of $Z_{k,i}$ and $Z_{k,j}$ $(i\neq j)$ that

$$\mathbb{E}\left(\|t^k-\nu(w^k)\|^2|\mathcal{P}_k\right)=\frac{1}{n_k^2}\sum_{i=1}^{n_k}\mathbb{E}\left(\|\tau(w^k,Z_{k,i})-\nu(w^k)\|^2|\mathcal{P}_k\right).$$

We observe that

$$\begin{aligned}&\mathbb{E}\left(\|\tau(w^k,Z_{k,i})-\nu(w^k)\|^2|\mathcal{P}_k\right)\\&=\mathbb{E}_Z\left(\|\tau(w^k,Z)\|^2\right)-\|\nu(w^k)\|^2=\mathbb{V}_Z(\tau(w^k,Z)).\end{aligned}$$

Thus, $\mathbb{E}\left(\|t^k-\nu(w^k)\|^2|\mathcal{P}_k\right)=\dfrac{1}{n_k}\mathbb{V}_Z(\tau(w^k,Z))$.
Combining this and (3.10), (3.11), (3.12), and we obtain

$$\begin{aligned}&\mathbb{E}(F(w^{k+1})-F(w^k)|\mathcal{P}_k)\leq\frac{\mathbb{V}_Z(\tau(w^k,Z))}{2\bar{\rho}\times n_k}\\&+\mathbb{E}\left(G(w^{k+1})-\frac{1}{n_k}\sum_{i=1}^{n_k}g(w^{k+1},Z_{k,i})|\mathcal{P}_k\right).\end{aligned}\tag{3.13}$$

Next, we make an upper bound on the right-hand side of (3.13). Firstly, the (nonnegative) term $\mathbb{V}_Z(\tau(w^k,Z))$ is bounded above by $\mathbb{E}(\tilde{\tau}(Z)^2)$. Secondly, we show that

$$\mathbb{E}\left(\sup_{w\in S}\left|G(w)-\frac{1}{n_k}\sum_{i=1}^{n_k}g(w,Z_{k,i})\right|\right)\leq 2R_{n_k}(g,\Xi).\tag{3.14}$$

To prove (3.14), let us first introduce "ghost samples" $Z'_{k,1},Z'_{k,2},\ldots,Z'_{k,n_k}$ (similar to the arguments in [14]) that are independent of all $Z_{k,i}$ and identically distributed with Z. By Jensen's inequality, we get

$$\begin{aligned}&\left|\frac{1}{n_k}\sum_{i=1}^{n_k}g(w,Z_{k,i})-\mathbb{E}(g(w,Z))\right|\\&\leq\mathbb{E}\left(\left|\frac{1}{n_k}\sum_{i=1}^{n_k}\left(g(w,Z_{k,i})-g(w,Z'_{k,i})\right)\right||Z_{k,i},i=\overline{1,n_k}\right).\\&\text{Thus, }\mathbb{E}\left(\sup_{w\in S}\left|\frac{1}{n_k}\sum_{i=1}^{n_k}g(w,Z_{k,i})-\mathbb{E}(g(w,Z))\right|\right)\\&\leq\mathbb{E}\left(\sup_{w\in S}\left|\frac{1}{n_k}\sum_{i=1}^{n_k}g(w,Z_{k,i})-\frac{1}{n_k}\sum_{i=1}^{n_k}g(w,Z'_{k,i})\right|\right).\end{aligned}$$

Now let $\sigma_1,\sigma_2,\ldots,\sigma_{n_k}$ be independent random variables with $\mathbb{P}(\sigma_i=1)=\mathbb{P}(\sigma_i=-1)=\frac{1}{2}$ in such a

way that they are also independent of $Z_{k,i}$ and $Z'_{k,i}$. Then,

$$\begin{aligned}
&\mathbb{E}\left(\sup_{w\in S}\left|\frac{1}{n_k}\sum_{i=1}^{n_k} g(w,Z_{k,i}) - \frac{1}{n_k}\sum_{i=1}^{n_k} g(w,Z'_{k,i})\right|\right)\\
&= \mathbb{E}\left(\sup_{w\in S}\left|\frac{1}{n_k}\sum_{i=1}^{n_k} \sigma_i(g(w,Z_{k,i}) - g(w,Z'_{k,i}))\right|\right)\\
&\leq 2\mathbb{E}\left(\mathbb{E}_\sigma\left(\sup_{w\in S}\frac{1}{n_k}\left|\sum_{i=1}^{n_k}\sigma_i g(w,Z_{k,i})\right|\right)\right)\\
&= 2\mathbb{E}(R_{n_k}(g,Z^{n_k})) \leq 2\mathbb{E}(R_{n_k}(g,\Xi)) = 2R_{n_k}(g,\Xi).
\end{aligned}$$

Now, we establish the almost sure convergence of the sequence $\{F(w^k)\}$ as follows. Assumption 1(iv) implies that there exists R such that $F(w) \geq R, \forall w \in S$. Let $D(w) = F(w) - R \geq 0$ and $S_k = [\mathbb{E}(D(w^{k+1}) - D(w^k)|\mathcal{P}_k) > 0]$. Since S_k is $\mathcal{P}_k$-measurable and by using (3.13), (3.14) , we obtain

$$\begin{aligned}
&\sum_{k=1}^{\infty}\mathbb{E}(1_{S_k}(D(w^{k+1}) - D(w^k)))\\
&= \sum_{k=1}^{\infty}\mathbb{E}\left(\mathbb{E}(1_{S_k}(D(w^{k+1}) - D(w^k))|\mathcal{P}_k)\right)\\
&\leq \frac{1}{2\bar{\rho}}\sum_{k=1}^{\infty}\frac{\mathbb{E}(\mathbb{V}_Z(\tau(w^k,Z)))}{n_k} + 2\sum_{k=1}^{\infty}R_{n_k}(g,\Xi)\\
&\leq \frac{\mathbb{E}(\tilde{\tau}(Z)^2)}{2\bar{\rho}}\sum_{k=1}^{\infty}\frac{1}{n_k} + 2N_g\sum_{k=1}^{\infty}\frac{1}{n_k^\alpha} < +\infty.
\end{aligned}$$

It follows from semimartingale convergence theorem [89] that there exists D^∞ integrable such that $D(w^k) \to D^\infty$ a.s., which implies $F(w^k) \to F^\infty = D^\infty + R$ a.s.

2. By AM-GM inequality, $\langle t^k - \nu(w^k), w^{k+1} - w^k\rangle \leq \frac{1}{\bar{\rho}}\|t^k - \nu(w^k)\|^2 + \frac{\bar{\rho}}{4}\|w^{k+1} - w^k\|^2$.

Combining this inequality with (3.9), we get

$$\begin{aligned}
&\frac{\bar{\rho}}{4}\|w^{k+1} - w^k\|^2 \leq F(w^k) - F(w^{k+1}) + G(w^{k+1}) - G(w^k)\\
&- \frac{1}{n_k}\sum_{i=1}^{n_k}\left(g(w^{k+1},Z_{k,i}) - g(w^k,Z_{k,i})\right) + \frac{1}{\bar{\rho}}\|t^k - \nu(w^k)\|^2.
\end{aligned}$$

By applying Lebesgue dominated convergence theorem [17, Theorem 4.2], we get

$$\begin{aligned}
&\frac{\bar{\rho}}{4}\mathbb{E}\left(\sum_{k=1}^{\infty}\|w^k - w^{k+1}\|^2\right) \leq \mathbb{E}(F(w^1)) - \mathbb{E}(F^\infty)\\
&+ \frac{M}{\bar{\rho}}\sum_{k=1}^{\infty}\frac{1}{n_k} + 2N_g\sum_{k=1}^{\infty}\frac{1}{n_k^\alpha} < \infty.
\end{aligned}$$

Therefore, $\sum_{k=1}^{\infty}\|w^k - w^{k+1}\|^2 < +\infty$ a.s.

3. Denote by $G_k(w) = \frac{1}{n_k}\sum_{i=1}^{n_k} g(w,Z_{k,i})$. It follows from $t^k \in \partial G_k(w^{k+1})$ and $\nu(w^k) \in$

$\partial H(w^k)$ that

$$\langle w^{k+1}, t^k\rangle = G_k(w^{k+1}) + G_k^*(t^k),$$
$$\langle \nu(w^k), w^k\rangle = H(w^k) + H^*(\nu(w^k)).$$

Then we obtain

$$G_k(w^k) - H(w^k) \geq H^*(\nu(w^k)) - G_k^*(t^k) + \langle t^k - \nu(w^k), w^k\rangle$$
$$\geq G_k(w^{k+1}) - H(w^{k+1}) + \langle t^k - \nu(w^k), w^k - w^{k+1}\rangle,$$

which implies

$$G_k(w^k) - H(w^k) - H^*(\nu(w^k)) + G_k^*(t^k) \to 0, \tag{3.15}$$

since $G_k(w^k) - H(w^k) \to F^\infty$, $t^k - \nu(w^k) \to 0$, and $G_k(w^{k+1}) - H(w^{k+1}) \to F^\infty$.

Hence, (3.15) implies $G(w^k) + G_k^*(t^k) - \langle w^k, \nu(w^k)\rangle \to 0$.

On another hand, from the definition of the conjugate functions G^* and G_k^* we get

$$|G_k^*(t^k) - G^*(t^k)| \leq \sup_{x\in S} |G_k(x) - G(x)| \to 0.$$

Thus $G(w^k) + G^*(t^k) - \langle w^k, \nu(w^k)\rangle \to 0$ a.s. Now let $\mathcal{L}$ be an intersection of sets with probability 1 gained from all almost surely true statements from the beginning of the proof, we have $\mathbb{P}(\mathcal{L}) = 1$ since there are at most countably finite statements. Let $\omega \in \mathcal{L}$, we have $\{w^k(\omega)\}$ and $\{\nu(w^k(\omega))\}$ are bounded. Let $w^* \in S$ be a limit point of $\{w^k(\omega)\}$, there exists a subsequence $\{w^{k_j}(\omega)\}$ such that $w^{k_j}(\omega) \to w^*$. By extracting a subsequence of $\{\nu(w^{k_j}(\omega))\}$ if necessary, we can assume that $\nu(w^{k_j}(\omega)) \to \nu^*$, which implies $t^{k_j}(\omega) \to \nu^*$. Therefore, $G(w^{k_j}(\omega)) + G^*(t^{k_j}(\omega)) \to \langle w^*, \nu^*\rangle$. By letting $j \to +\infty$ and noting that $\theta(w, z) = G(w) + G^*(z)$ is lower semicontinuous, we obtain $G(w^*) + G^*(\nu^*) \leq \langle w^*, \nu^*\rangle$. On the other hand, according to Young's inequality, $G(w^*) + G^*(\nu^*) \geq \langle w^*, \nu^*\rangle$. Therefore, $G(w^*) + G^*(\nu^*) = \langle w^*, \nu^*\rangle$. In other words, $\nu^* \in \partial G(w^*)$. Furthermore, for each $w \in S$, it follows from $\nu(w^{k_j}(\omega)) \in \partial H(w^{k_j}(\omega))$ that $H(w) \geq H(w^{k_j}(\omega)) + \langle \nu(w^{k_j}(\omega)), w - w^{k_j}(\omega)\rangle$, which implies $H(w) \geq H(w^*) + \langle \nu^*, w - w^*\rangle$. Therefore, $\nu^* \in \partial H(w^*)$, and we conclude that w^* is a critical point of $F = G - H$ since $\partial G(w^*) \cap \partial H(w^*) \neq \emptyset$. □

Remark 5. *(i) The algorithm only uses samples at the current time to update the solution (past samples are no longer used). Therefore, even if the distribution of Z changes at a certain time (suppose that, due to some real-world events, Z becomes Z' at the iteration k), the algorithm will automatically solve the problem (3.2) with Z being replaced by Z'. Indeed, the current solution w^k can be considered as the initial point for restart, the algorithm continues operating based on new samples from the distribution of Z'. Theorem 10 is still valid, and the subsequential convergence with probability one to DC critical points of the DC problem associated with the new distribution is guaranteed. This is indeed an advantage of the osDCA. In contrast, intuitively, stochastic algorithms using aggregated update (still using old samples to compute the current solution) barely have this kind of adaptivity. We will conduct numerical experiments to study this aspect.*
(ii) Our algorithm and the convergence analysis can be extended to deal with the more general problem whose the random variables inside the first and the second DC components are not necessarily the

same, i.e., $F(w) = \mathbb{E}(g(w,Z)) - \mathbb{E}(h(w,\tilde{Z}))$. *At the iteration* k, *we approximate values of* G *and the subgradients of* H *by using* n_k *independent random samples from the distribution of* Z *and* $\tilde{n}_k$ *independent random samples from the distribution of* $\tilde{Z}$, *respectively. The sample size sequences* $\{n_k\}$ *and* $\{\tilde{n}_k\}$ *need to increase in such a way that* $\sum_{k=1}^{\infty} n_k^{-\alpha} < \infty$ *and* $\sum_{k=1}^{\infty} \tilde{n}_k^{-1} < \infty$.

Next, we will discuss two scenarios where one can directly compute (without stochastically approximation) values of G or H. Since the information of G (resp. H) can be achieved, we will modify Algorithm 4 to exploit this advantage. Note that these two schemes are not special cases of Algorithm 4, but they will coincide with Algorithm 4 in some cases (Remark 6).

G can be directly computed without approximation In this case, G does not need to be stochastically approximated, we replace the approximation of G in step 3 of Algorithm 4 by its true value, which results in Algorithm 5.

Algorithm 5 Online Stochastic DCA with exact G

Algorithm 4, in which the problem (3.5) in step 3 is replaced by $\min\left\{G(w) - \langle t^k, w\rangle : w \in \mathbb{R}^m\right\}$.

With this algorithm, we obtain stronger convergence results since G is computed exactly. Note that, in the convergence results of Algorithm 4, we impose Assumption 2 in order to control the variance of the stochastic estimator of G. To study the convergence of Algorithm 5, we do not need such an assumption. Furthermore, in Assumption 1, we replace the convexity condition $\rho_H + \inf_{z\in\Xi} \rho(g(\cdot,z)) > 0$ by the weaker one $\rho_H + \rho_G > 0$, which gives rise to a milder assumption called Assumption 1'. We obtain Theorem 11 whose proof is similar to that of Theorem 10.

Theorem 11. *Under Assumption 1', if the sequence of sample sizes* $\{n_k\}$ *satisfies* $\sum_{k=1}^{\infty} n_k^{-1} < +\infty$, *then the iterations of Algorithm 5 satisfy:*

1. There exists F^∞ *integrable such that* $F(w^k) \to F^\infty$ *a.s.*

2. $\sum_{k=1}^{\infty} \|w^{k+1} - w^k\|^2 < +\infty$ *a.s.*

3. There exists a measurable set $\mathcal{L} \subset \Omega$ *with* $\mathbb{P}(\mathcal{L}) = 1$ *such that for each* $\omega \in \mathcal{L}$, *every limit point of* $\{w^k(\omega)\}$ *is a critical point of* $F = G - H$.

H can be directly computed without approximation In this case, we replace the stochastic estimator of the subgradient of H in Algorithm 4 by the true subgradient of H to obtain the following algorithm.

Algorithm 6 Online Stochastic DCA with exact H

Algorithm 4, in which the step 2 is replaced by
2. Compute $t^k \in \partial H(w^k)$.

Since we work directly on H, we replace Assumption 1(i) by $\operatorname{dom}\partial H = S$. Likewise, Assumption 1(iii) is replaced by: there exists $M > 0$ such that $\forall w \in S, \forall t \in \partial H(w) : \|t\| \leq M$. These modifications bring about a new set of assumptions called Assumption 1". We obtain the following convergence results whose proof is similar to that of Algorithm 10.

Theorem 12. *Under Assumptions 1" and 2, if the sequence of sample sizes* $\{n_k\}$ *satisfies* $\sum_{k=1}^{\infty} n_k^{-\alpha} < +\infty$, *the iterations of Algorithm 6 satisfy:*

1. There exists F^∞ *integrable such that* $F(w^k) \to F^\infty$ *a.s.*

2. $\sum_{k=1}^{\infty} \|w^{k+1} - w^k\|^2 < +\infty$ *a.s.*

3. There exists a measurable set $\mathcal{L} \subset \Omega$ *with* $\mathbb{P}(\mathcal{L}) = 1$ *such that for each* $\omega \in \mathcal{L}$*, every limit point of* $\{w^k(\omega)\}$ *is a critical point of* $F = G - H$.

Remark 6. *(i) In practice, thanks to the flexibility of DC decompositions, one can usually formulate the problem* (3.1) *as a stochastic DC program,* $f(w,z) = g(w,z) - h(w,z)$*, where either* g *or* h *does not depend on* z*. When* $g(w,z)$ *does not depend on* z*, Algorithm 5 coincides with Algorithm 4. For example, if the functions* $f(\cdot,z)$ *are* L*-smooth with the same constant* L *for all* $z \in \Xi$*,* f *has the following DC decomposition:* $f(w,z) = (L/2)\|w\|^2 - \big((L/2)\|w\|^2 - f(w,z)\big)$*. Likewise, when* $h(w,z)$ *does not depend on* z*, Algorithm 6 and Algorithm 4 coincide. For instance, suppose that there exists a convex function* $\varphi(w)$ *such that* $f(w,z) + \varphi(w)$ *are convex for all* $z \in \Xi$ *(in particular, when* $\varphi(w) = (\kappa/2)\|w\|^2$*,* $f(\cdot,z)$ *are weakly convex),* f *has the following DC decomposition:* $f(w,z) = (f(w,z) + \varphi(w)) - \varphi(w)$.

(ii) Apart from the use of the above appropriate DC decompositions so that G *or* H *is a deterministic function and thereafter Algorithm 5 or 6 is applicable, the exact computation of* G *and* H *requires additional knowledge about* Z*. For instance, suppose that the distribution of* Z *is unknown but the distribution of its measurement,* $\mu(Z)$*, is known (*P_μ*). For example,* μ *can be the magnitude* $\|Z\|$*, a partial observation* Z^* *(*Z^* *is a part of the vector* Z*), the difference* $Z_1 - Z_2$ *(where* $Z = (Z_1, Z_2)$*), etc. In such cases, if* $g(w,\cdot)$ *is a function with respect to* μ*, i.e.,* $g(w,z) = u(w,\mu(z))$*, the expectation* $\mathbb{E}(g(w,z)) = \int u(w,\mu)dP_\mu$ *which is possibly computed. Similar arguments apply to* $h(w,z)$.

(iii) In big data analytics, large-sum problems play a key role. We consider the following large-sum objective function

$$F(w) = \sum_{i=1}^{N} \alpha_i f_i(w) = \sum_{i=1}^{N} \alpha_i g_i(w) - \sum_{i=1}^{N} \alpha_i h_i(w),$$

where g_i, h_i *are convex,* $\alpha_i \geq 0$ *for all* $i = \overline{1,N}$ *and* $\sum_{i=1}^{N} \alpha_i = 1$*. The function* F *can be rewritten as* $F(w) = \mathbb{E}(g_I(w)) - \mathbb{E}(h_I(w))$*, where* I *is a random index with* $\mathbb{P}(I = i) = \alpha_i$*. In this case, the distribution of* I *is known completely. However, as* N *can be very large, we may still need to apply osDCA schemes. Furthermore, since* I *is known, we have full freedom to choose Algorithm 4, Algorithm 5, or Algorithm 6 to apply, which leads to - in general - three distinctive algorithms. The practical trade-off between these algorithms would be which DC component (or none of them) is cheaper to be computed directly.*

3.4 Applications: solving the Expected PCA

In this section, we will apply osDCA schemes to the expected problem of PCA to study the generalization capacity of the proposed methods.

3.4.1 osDCA schemes for solving Expected PCA

We consider the following expected problem of PCA (denoted by E-PCA) as follows [90],

$$\min\{-\frac{1}{2}\mathbb{E}(\langle w, Z\rangle^2) : \|w\| \leq 1, \quad \text{(E-PCA)}$$

where Z is a normalized random vector, i.e. $\|Z\| = 1$, with unknown distribution. The situation in which we are interested is the data obtained online. The problem (E-PCA) can be considered as the theoretical problem of the classic PCA (and - vice versa - the classic PCA is the empirical problem of (E-PCA)). The problem (E-PCA) aims to generalize the compressing capacity of the classical PCA on unseen data.

Firstly, we see that the problem (E-PCA) is nonconvex and it can be formulated as a DC problem of the form (3.4), where

$$G(w) = \frac{\lambda}{2}\|w\|^2, H(w) = \mathbb{E}\left(\frac{\lambda}{2}\|w\|^2 + \frac{1}{2}\langle w, Z\rangle^2\right), \tag{3.16}$$

$S = \{w \in \mathbb{R}^m : \|w\| \leq 1\}$ and $\lambda > 0$. Although we have a very natural DC decomposition with $G(w) = 0, H(w) = \mathbb{E}\left(\frac{1}{2}\langle w, Z\rangle^2\right)$, here we add $\frac{\lambda}{2}\|w\|^2$ to both DC components to fulfill to Assumption 1(ii). Since the values G are directly obtained without approximation, Algorithm 4 coincides with Algorithm 5. We call this scheme osDCA-1, where the k-th iteration is described as follows.

1. Receive n_k samples $Z_{k,1}, \ldots, Z_{k,n_k}$.
2. Compute $t^k = \lambda w^k + \frac{1}{n_k}\sum_{i=1}^{n_k}\langle w^k, Z_{k,i}\rangle Z_{k,i}$.
3. Update $w^{k+1} = \begin{cases} \lambda^{-1}t^k \text{ if } \|t^k\| \leq \lambda \\ t^k/\|t^k\| \text{ otherwise.} \end{cases}$

Secondly, it is well-known that if a function θ has L-Lipschitz continuous gradient, then $(L/2)\|\cdot\|^2 - \theta$ and $(L/2)\|\cdot\|^2 + \theta$ are convex. Therefore, we have another DC decomposition for the problem (E-PCA) as follows,

$$\begin{aligned} G(w) &= \mathbb{E}\left(\tfrac{L}{2}\|w\|^2 - \tfrac{1}{2}\langle w, Z\rangle^2\right), \\ H(w) &= \mathbb{E}\left(\tfrac{L}{2}\|w\|^2 + \tfrac{1}{2}\langle w, Z\rangle^2\right). \end{aligned} \tag{3.17}$$

Since G and H remain unknown, we apply Algorithm 4 to this DC problem. Obviously the family $\{g(\cdot, z) : \|z\| = 1\}$ is uniformly Lipschitz and uniformly bounded by a constant, therefore, the rate α in Assumption 2 can be chosen arbitrarily in $(0, 1/2)$. With this setup, we obtain a second scheme called osDCA-2 whose the k-th iteration is described as follows.

1. Receive n_k samples $Z_{k,1}, Z_{k,2}, \ldots, Z_{k,n_k}$.
2. Compute the stochastic gradient

$$t^k = Lw^k + \frac{1}{n_k}\sum_{i=1}^{n_k}\langle w^k, Z_{k,i}\rangle Z_{k,i}.$$

3. Solve the following convex program to get w^{k+1},

$$\min_{w\in S}\left\{\frac{L}{2}\|w\|^2 - \frac{1}{2n_k}\sum_{i=1}^{n_k}\langle w, Z_{k,i}\rangle^2 - \langle t^k, w\rangle\right\}. \tag{3.18}$$

The problem (3.18) is convex and can be solved by an existing convex optimization solver. Thanks to the efficiency of DCA we use again it to solve (3.18), via the following "false" DC decomposition

$$\tilde{g}(w) = \frac{L}{2}\|w\|^2, \tilde{h}(w) = \frac{1}{2n_k}\sum_{i=1}^{n_k}\langle w, Z_{k,i}\rangle^2 + \langle t^k, w\rangle.$$

Table 3.1: Datasets' information

Dataset	Features	Train-Validation split	Test set
letter	16	15000:2500	2500
YearPredictionMSD	90	463715:25815	25815
SensIT Vehicle	100	78823:9852	9853
shuttle	9	43500:7250	7250
covtype	54	500000:40506	40506
SUSY	18	4900000:50000	50000
codrna	8	59535:78706	78707
madelon	500	2000:300	300
mushrooms	112	6000:1000	1124
satimage	36	4435:1000	1000
a7a	123	12880:1610	1610
a9a	123	26049:3256	3256

We get a simple DCA scheme where the solutions of convex subproblems have closed-form. The (deterministic) DCA takes the current solution $u^0 = w^k$ as the initial point, then operates until the stopping criterion which is $\|u^{l+1} - u^l\| < \epsilon$ is met, where $\epsilon > 0$ is the error tolerance.

3.4.2 Numerical experiments

Datasets

The numerical experiments are conducted on standard machine learning datasets on LIBSVM. The information of the used datasets is described in Table 3.1. The samples of each dataset are normalized as $\|z_i\| = 1$.

Furthermore, to test the adaptive ability of osDCA schemes, we generate a synthetic dataset that consists of two subdatasets (training set (200000×500), test set (500000×500)) and (training set (200000×500), test set (200000×500)), in which the generating mechanism is described in Subsection 3.4.2.

Comparative algorithms

We compare our algorithms with two versions of Projected Stochastic Subgradient method (PSS) [30] - an online algorithm for weakly convex objective functions, four Stochastic DCA schemes (SDCA) [70] proposed for nonconvex, nonsmooth DC programs, and two benchmark algorithms for solving online PCA including the AdaOja [53] and MaxAP [33].

Experiment setup and results

The numerical experiments comprise of five parts. The first experiment is a comparison between algorithms, the second experiment studies our algorithms' behaviors when the DC decomposition of the problem varies, the third experiment compares between convex solvers for solving subproblems, the fourth experiment studies the adaptive capacity of osDCA schemes, and the fifth experiment compares directly the differences in the practical performance of the three proposed osDCA schemes.

Experiment 1. We compare osDCA schemes with two versions of PSS (constant stepsize policy and diminishing stepsize policy), four SDCA schemes, AdaOja, and MaxAP. Firstly, we run cross-validation 10 times on each dataset to find the best hyperparamters for the comparative algorithms. To be specific, for the PSS with constant stepsize, we cross-validate to find the best stepsizes from $\{0.001, 0.005, 0.01, 0.015, 0.02\}$; Meanwhile, for the PSS with adaptive stepsize $\alpha_k = c/k$, c is searched from $\{4, 5, 6, \ldots, 15\}$. On the other hand, the AdaOja automatically tunes its stepsize based on the gradient information. The MaxAP uses diminishing stepsize $\alpha_k = c/k$ where c is chosen from $\{1, 2, \ldots, 10\}$. For the four SDCA schemes, it should be stressed that SDCA1 and SDCA3 require the first DC component of the objective to be explicitly defined, meanwhile, SDCA2 and SDCA4 can handle the unknown first DC component. Therefore, we apply SDCA1 and SDCA3 to (3.16) with $\lambda = 10^{-6}$ (small λ yields good results [70]); meanwhile, SDCA2 and SDCA4 are applied to (3.17) where $L = 1$. We use the sequence of equal weights for all four SDCA schemes. On the other hand, based on the theoretical analysis, the parameters of osDCA schemes are chosen as follows. For the osDCA-1, we choose the sequence of sample sizes as $n_k = k^2$, and $\lambda = 1$ which is a neutral number. For the osDCA-2, the sequence of sample sizes is chosen as $n_k = k^3$, the Lipschitz smoothness constant $L = 1$ and the tolerance error in solving subproblems $\epsilon = 10^{-5}$.

Next, the training dataset and the validation dataset are merged together and are fed to all algorithms (one pass - to meet the online setting context and guarantee the independence of samples). As a preprocessing step, each training dataset is randomly shuffled before each run. The starting points are also randomly initialized in S. The performance of our algorithms are measured on the test set to guarantee their generalization capability. To enhance visualization, we report the suboptimality graph $F(w_n) - F(w^*)$ averaging over 20 runs, where $F(w^*)$ is the optimal value found on each test set by computing the largest eigenvalue of the covariance matrix. Furthermore, we classify osDCA-1, SDCA1, SDCA3 in one group and osDCA-2, SDCA2, SDCA4 in another group (since the former three use the DC decomposition (3.16) and the latter three use (3.17)) to plot them in two different figures.

All experiments are performed on a PC Intel(R) Core(TM) i7-8700 CPU @3.20GHz of 16 GB RAM.

Figures 3.1 and 3.2 illustrate the performance of all algorithms (we only display six out of twelve datasets). Moreover, to show the final effect of the learned system, we report in Table 3.2 the average reconstruction errors (on test sets) when reconstructing compressed data from the learned eigenspace.

Comparisons between osDCA schemes and PSS, AdaOja, MaxAP. Our algorithms take a very short amount of time to pass through the training sets while obtaining really small suboptimality values, say $10^{-4} \sim 10^{-5}$ in most of the cases. In contrast, the PSS with constant stepsize and the AdaOja converge slower and usually can not obtain the comparable objective values. On the other hand, PSS with diminishing stepsize performs very well and obtains similar suboptimality to osDCA schemes, where the differences are negligible. Lastly, the MaxAP performs quite poorly in this experiment.

Comparisons between osDCA and SDCA. About the solutions' quality, osDCA schemes and SDCA1, SDCA3, SDCA4 are similar in all cases; Meanwhile, the SDCA2 is slighly worse in some cases (e.g. letter, SensIT Vehicle). On the other hand, the osDCA-2 usually obtains such solutions at a faster rate than SDCA2 and SDCA4.

Overall, among the top six best algorithms (osDCA-1,2, SDCA1,3,4, PSS diminishing stepsize), time to pass through the entire training datasets (only considered in datasets where the computational time is significant) of osDCA-1 is the shortest, osDCA-2 is the second-best. In comparison, SDCA1,

Table 3.2: Average reconstruction errors of all studied algorithms

	osDCA-1	osDCA-2	SDCA1	SDCA2	SDCA3	SDCA4	PSS-dimi	PSS-const	AdaOja	MaxAP
letter	**0.4412**	**0.4412**	**0.4412**	0.4413	**0.4412**	**0.4412**	**0.4412**	0.4413	0.4423	0.4490
YearPredictionMSD	**0.1782**	**0.1782**	**0.1782**	**0.1782**	**0.1782**	**0.1782**	**0.1782**	0.1783	0.1783	0.1785
SensIT Vehicle	**0.7054**	**0.7054**	**0.7054**	0.7059	**0.7054**	**0.7054**	**0.7054**	**0.7054**	0.7058	0.7212
shuttle	**0.1130**	**0.1130**	**0.1130**	**0.1130**	**0.1130**	**0.1130**	**0.1130**	**0.1130**	0.1131	0.1140
covtype	**0.1096**	**0.1096**	**0.1096**	**0.1096**	**0.1096**	**0.1096**	**0.1096**	0.1097	0.1097	**0.1096**
SUSY	**0.4550**	**0.4550**	**0.4550**	**0.4550**	**0.4550**	**0.4550**	**0.4550**	0.4551	**0.4550**	0.4566
codrna	**0.0151**	**0.0151**	**0.0151**	**0.0151**	**0.0151**	**0.0151**	**0.0151**	0.0152	0.0152	0.0157
madelon	**0.0037**	**0.0037**	**0.0037**	**0.0037**	**0.0037**	**0.0037**	**0.0037**	0.0038	0.0038	0.0047
mushrooms	**0.6028**	0.6030	0.6032	0.6043	0.6031	0.6031	0.6029	0.6044	0.6069	0.6082
satimage	**0.6271**	0.6307	0.6337	0.6322	0.6319	0.6319	0.6313	0.6307	0.6277	0.6600
a7a	0.5491	**0.5490**	**0.5490**	0.5491	**0.5490**	**0.5490**	**0.5490**	0.5496	0.5506	0.5582
a9a	**0.5478**	**0.5478**	**0.5478**	0.5479	**0.5478**	**0.5478**	**0.5478**	0.5479	0.5489	0.5567

which is the fastest algorithm among all SDCA schemes, is $2.8 \sim 7.7$ times slower than osDCA-1; PSS with diminishing stepsize is $2.9 \sim 7.1$ times slower than osDCA-1.

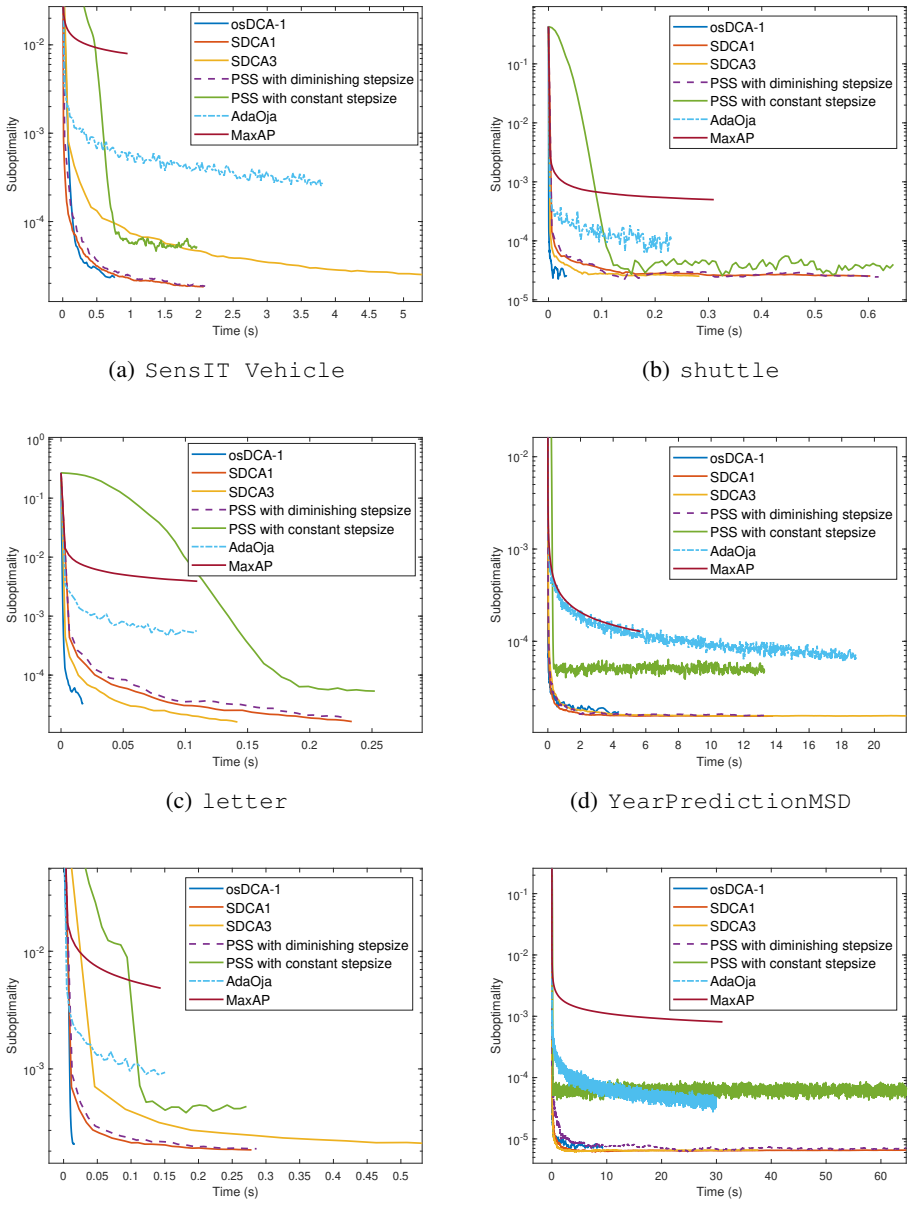
Furthermore, it is well-known that there are two main factors needed to be carefully considered when designing any DCA (or its variants), namely the DC decomposition of the problem and the convex solver for subproblems. Therefore, we consider the following experiments to study our proposed algorithms' behaviors within these two mentioned perspectives.

Experiment 2. Our aim is to study the behavior of osDCA-1 when λ varies (change the DC decomposition of the problem). It is observed that, to surely fulfill the strong convexity condition $\rho_G + \rho_H > 0$, we add the regularization term $\lambda\|\cdot\|^2$ to both G and H components. A natural question raised is that: suppose H is already strongly convex, will we obtain some "optimal" performance if we do not use this regularization term? This curiosity motivates us to perform the osDCA-1 scheme with DC decomposition $g(w,z) = 0, h(w,z) = \frac{1}{2}\langle w,z\rangle^2$. Before presenting the experimental results, let us discuss a little bit about the condition $\rho_H > 0$ in this case. We know that this condition does not always hold and it is equivalent to $\mathbb{E}(ZZ^\top)$ being positive definite. By definition, the positive definiteness of $\mathbb{E}(ZZ^\top)$ is equivalent to $\mathbb{E}\left((w^\top Z)^2\right) > 0, \forall w \neq 0$. Therefore, this condition is violated if there exists $w_0 \neq 0$ such that $\mathbb{E}((w_0^\top Z)^2) = 0$, or equivalently $w_0^T Z = 0$ almost surely. In other words, the condition $\rho_H > 0$ does not hold if there is a perfectly linear dependence between features of the random vector Z.

Figure 3.3 shows the behaviors of osDCA-1 with different $\lambda > 0$ and an extreme case where $\lambda = 0$ on the `YearPredictionMSD` dataset. We observe that, the optimal performance of osDCA-1 is achieved at some moderate values of λ, say, from 1 to 5. Besides, the quality of the performance is not monotone with respect to λ. With large value of λ, osDCA-1 somehow gets stuck at the beginning. The performance of osDCA-1 is gradually improved as λ decreases up to a certain value, and then the performance slightly deteriorates as λ continues to approach 0.

Experiment 3. We study the performance of osDCA-2 with different convex solvers for subproblems. To be specific, beside the (deterministic) DCA used in the osDCA-2 scheme, we want to use the industrial CPLEX for solving the convex subproblems. Figure 3.4 shows the difference between osDCA-2 using deterministic DCA and CPLEX for solving convex subproblems. It is observed that while the suboptimality values of these two algorithms are similar, osDCA-2 using DCA for the convex subproblem is faster than osDCA-2 with CPLEX.

Experiment 4. We study the adaptive capacity of osDCA schemes compared with SDCA schemes when there is an abrupt change in the distribution of Z. We describe the context of the problem as

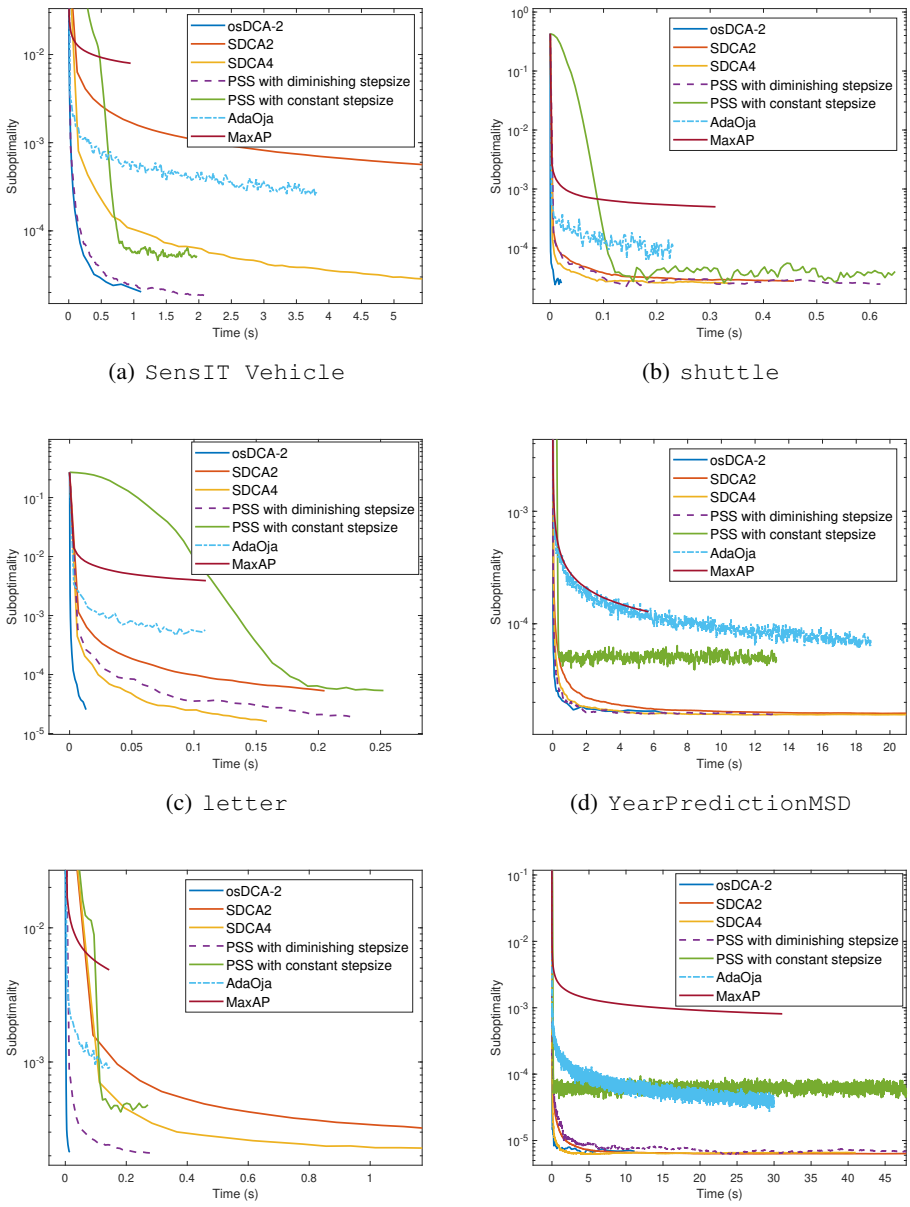


(a) `SensIT Vehicle` (b) `shuttle`

(c) `letter` (d) `YearPredictionMSD`

(e) `a7a` (f) `SUSY`

Figure 3.1: The performance of osDCA-1 compared with SDCA1, SDCA3, two versions of PSS, AdaOja, and MaxAP



(a) `SensIT Vehicle` (b) `shuttle`

(c) `letter` (d) `YearPredictionMSD`

(e) `a7a` (f) `SUSY`

Figure 3.2: The performance of osDCA-2 compared with SDCA2, SDCA4, two versions of PSS, AdaOja, and MaxAP

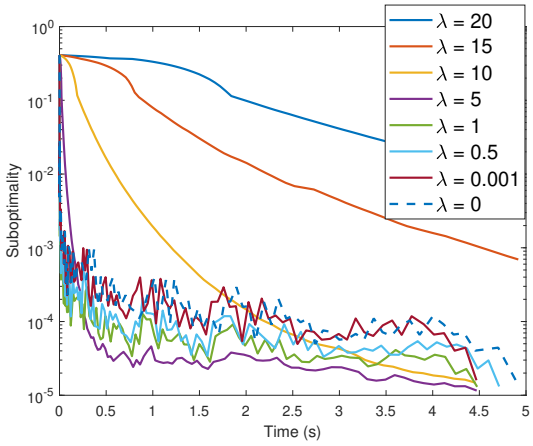
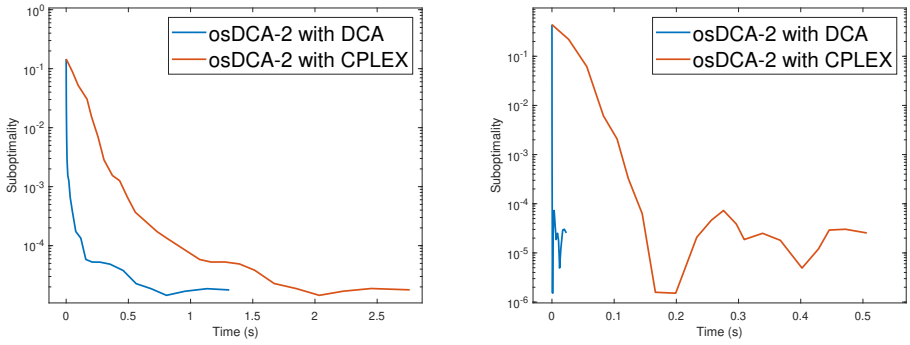
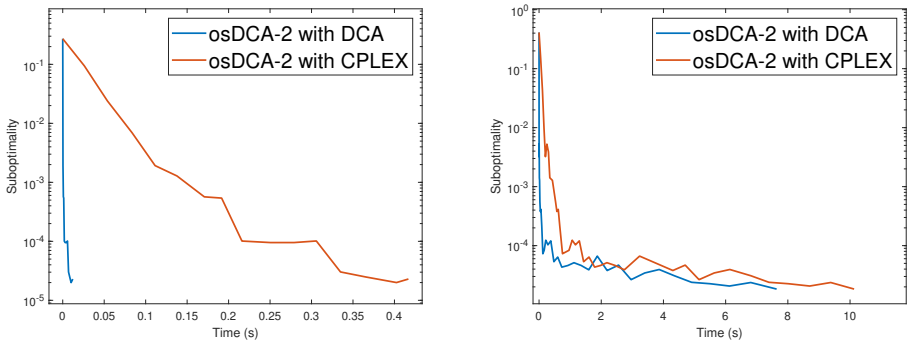


Figure 3.3: Performance (one run) of osDCA-1 when $\lambda \geq 0$ varies



(a) `SensIT Vehicle` (b) `shuttle`



(c) `letter` (d) `YearPredictionMSD`

Figure 3.4: The performance (one run) of osDCA-2 with two different convex solvers: DCA and CPLEX

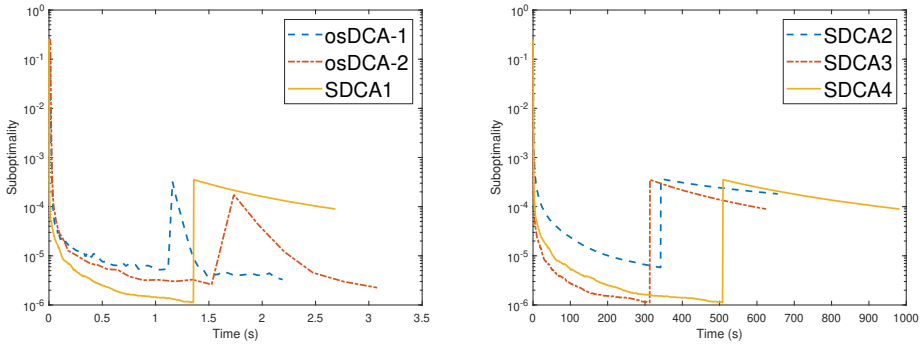
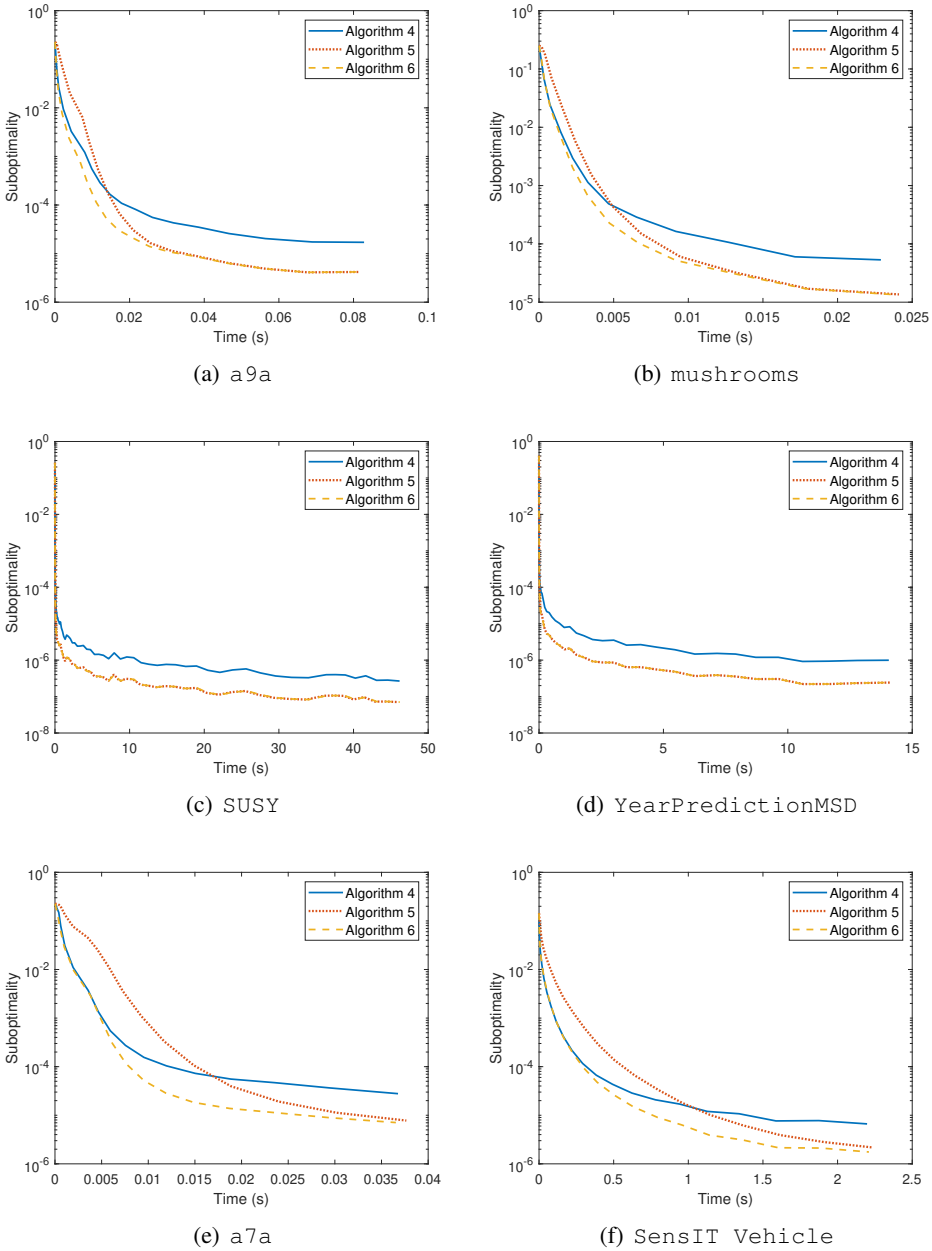


Figure 3.5: The adaptive ability of osDCA schemes over SDCA schemes

follows. We are receiving streaming data from an unknown distribution (the data is - in fact - realizations of Z). At a certain time, suppose that there is a real-world event that makes the distribution of Z change (Z becomes some Z'). We do not know this event (and hence, the change of Z is also unknown to us) and continue to receive streaming data from the changed distribution. From that time, we want to solve (3.2) with Z being replaced by Z' since the new random variable Z' is more relevant than Z.

To this end, we generate a synthetic dataset as follows. The dataset consists of two subdatasets representing data collected before and after the abrupt change. The first subdataset includes a training set (200000×500) and a test set (500000×500) that are generated from multivariate normal distribution with a mean vector 0 and a positive definite covariance matrix $\Sigma = \Lambda^\top\Lambda + I$ where Λ is randomly generated. Then, we change Λ and generate the second subdataset consisting of a training set (200000×500) and a test set (200000×500). All data is then normalized as $\|z_i\| = 1$. We concatenate two training sets to create one unified training set in order to feed to the algorithms. Before the change, we measure the performance of each algorithm on the first test set, and after the change, we use the second test set. Figure 3.5 shows the average results of 20 runs, here we separate the results into two subfigures because the running times of SDCA2, SDCA3, SDCA4 are remarkably longer than osDCA-1, osDCA-2, and SDCA1. The numerical results confirm the adaptive capacity of osDCA schemes over SDCA schemes. Indeed, after the abrupt change, osDCA schemes quickly regain suboptimality values that are as good as the ones obtained before the change. Meanwhile, SDCA schemes barely adapt to the change and decrease the suboptimality slowly.

Experiment 5. Finally, we compare the practical performance of the three proposed algorithms in a case on which they are all applicable, i.e. G and H can be exactly computed. Let us suppose that the random variable Z is known to have a discrete distribution over a set Ξ of N elements, $\mathbb{P}(Z = z_i) = \alpha_i$ for all $z_i \in \Xi$, where $\alpha_i > 0$ and $\sum_{i=1}^{N} \alpha_i = 1$. We use the DC decomposition (3.17) to test the performance of the three algorithms. Since α_i are known, we can compute $G(w)$ and $\nabla H(w)$ exactly. The probability vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)$ is randomly generated. Figure 3.6 shows the difference on the practical performance of our three proposed algorithms. We observe that Algorithms 5 and 6 always obtain better solutions in comparison to Algorithm 4. This result can be explained as Algorithms 5 and 6 use more information about the DC decomposition than Algorithm 4 which approximates both DC components at each iteration.



(a) a9a (b) mushrooms

(c) SUSY (d) YearPredictionMSD

(e) a7a (f) SensIT Vehicle

Figure 3.6: Comparison between the three proposed osDCA schemes

3.5 Conclusion

We have designed three online stochastic algorithms based on the DCA to handle stochastic nonsmooth, nonconvex DC programs. The first scheme stochastically approximates both DC components while the other two are designed for the context that one of two DC components can be directly computed. The convergence properties of the proposed algorithms are rigorously studied, and the almost sure convergence to critical points is established. As online stochastic algorithms, the osDCA schemes gain a competitive edge when dealing with streaming data. The benefits of osDCA schemes include remedying storage burden and the ability to adapt to new changes of data distribution, as well as the efficiency and flexibility of DCA. On the other hand, it is well-known that the variance of stochastic estimators of online stochastic algorithms is high, which creates difficulties in the convergence analysis, especially in nonconvex and nonsmooth settings. Our algorithms' convergence results hold thanks to the increase of sample sizes. Moreover, the rate of this increase is determined based on the Rademacher complexity of the family of functions $\{g(\cdot, z) : z \in \Xi\}$. Nevertheless, such complexity is not always easy to compute. In future works, we would like to improve this condition and provide a better rate. To study the practical behaviors of the proposed algorithms, we conduct numerical experiments on the expected problem of PCA with streaming data from an unknown distribution. The numerical experiments justify the efficiency of proposed algorithms. Indeed, the proposed osDCA schemes obtain good solutions within a short time. In addition, the adaptive capacity of osDCA schemes have been confirmed: after a change of the data distribution, our algorithms quickly adapt to the new distribution. As a comparison, SDCA schemes do not have this ability. Further experimental insights confirm the important role of DC decompositions and convex solvers for subproblems in DCA. It should be noted that DCA plays a double roles in these osDCA schemes, as it has also been used for solving the convex subproblems. And numerical experiments showed that the (deterministic) DCA is a very efficient and robust convex solver. Finally, via a direct comparison between the three proposed algorithms, it seems consistent that Algorithms 5 and 6 using the exact information of G or H (when possible) improve the performance of Algorithm 4.

Chapter 4

Markov chain Stochastic DCA and applications in Deep learning

Abstract. This chapter studies a broad class of nonsmooth nonconvex stochastic DC algorithms with *endogenous uncertainty* and the absence of i.i.d. (independent and identically distributed) samples. Instead, we assume that only Markov chains with sequences of distributions that converge to the target distributions can be accessed. Markovian noise occurs in a variety of contexts, including Bayesian inference, reinforcement learning, and stochastic optimization in high-dimensional or combinatorial spaces, therefore this setting is legitimate. We then develop a stochastic algorithm termed Markov chain stochastic DCA (MCSDCA). The convergence analysis is established in both asymptotic and nonasymptotic senses. The MCSDCA is then applied to deep learning via PDEs regularization, where two realizations of MCSDCA are constructed, namely MCSDCA-odLD and MCSDCA-udLD, based on overdamped and underdamped Langevin dynamics, respectively.

4.1 Introduction

We study the following nonsmooth nonconvex stochastic DC program

$$\min_{x\in\mathbb{R}^n}\{F(x) := G(x) - H(x)\}, \tag{4.1}$$

where G and H are convex, and a subgradient of H is given in the expectation form

$$\mathbb{E}_{P(\xi|x)}(v(x,\xi)) \in \partial H(x)$$

involving with the *endogenous uncertainty* of ξ (its distribution is determined by the optimization variable x). Furthermore, we suppose that i.i.d. samples from $P(\xi|x)$ are *unavailable*, but a Markov chain with an equilibrium distribution $P(\xi|x)$ can be constructed. Departing from the classical settings where the objective is convex and/or L-smooth and the uncertainty is *exogenous* (not affected by the optimization variable) with i.i.d. samples at hand, our setting has (direct or indirect) implications in challenging

The material of this chapter is developed based on the following work:
H. P. H. Luu, H. M. Le, & H. A. Le Thi. Markov Chain Stochastic DCA and Applications in Deep Learning with PDEs Regularization. **To be submitted**.

domains of machine learning such as variational inference, local entropy minimization, Boltzmann machine, as well as operational research where the uncertainty is influenced by the decision. The considered problem can be categorized as a nonconvex stochastic program with (endogenous) correlated noise. To our knowledge, there is no existing work in the literature that formally studies this problem.

Besides the important role of DC programming, our motivation to study this problem is twofold. Firstly, endogenous uncertainty occurs naturally in a variety of applications and is more difficult to deal with than exogenous uncertainty. In general, transitioning from exogeneity to endogeneity results in the loss of convexity [39]. While the majority of stochastic optimization research focuses on exogenous uncertainty, endogenous uncertainty has received little attention. Our work attempts to discuss this topic to a certain extent. Third, while the availability of i.i.d. samples, on which classic approaches like Sample Average Approximation (SAA) and Stochastic Approximation (SA) rely, is a common assumption in stochastic optimization, this is not always the case. Such samples are frequently unfeasible or prohibitively expensive in simulation, for example, in high-dimensional or combinatorial space [38, 124], and also in the context of Bayesian posterior sampling. Working with a sequence of dependent samples, namely a Markov chain, becomes more viable. Dimension is less of an issue with this approach because some strategies, such as Gibbs sampling, can break down high-dimensional target densities into low-dimensional conditionals. However, because of the autocorrelation between the chain elements, Markov sampling has a higher variance than i.i.d. sampling. In terms of modeling, data obtained in the real world can be highly correlated, therefore modeling them as i.i.d. data is unreasonable. Dependent samples (Markovian or general ergodic data) are utilized for various special classes of stochastic programming under exogenous uncertainty, such as least-square regression [93] and convex/L-smooth objective [38, 124, 35].

Examples of problem (4.1) The following exogenous stochastic DC program has got a lot of attention recently and has been studied in Chapter 3,

$$\min F(x) = G(x) - H(x) := \mathbb{E}_{P(\xi)}(g(x,\xi)) - \mathbb{E}_{P(\xi)}(h(x,\xi)) \text{ s.t. } x \in X, \tag{4.2}$$

where $g(\cdot,\xi), h(\cdot,\xi)$ are convex functions and X is a convex set. Obviously, (4.2) is a special case of (4.1) with $\mathbb{E}_{P(\xi)}(v(x,\xi)) \in \partial H(x)$, where $v(x,\xi)$ is a subgradient of $h(x,\xi)$.

We next consider the endogenous stochastic program $\min_\theta \mathbb{E}_{P(\xi|\theta)}(f(\theta,\xi))$, where θ represents the hyperparameters of a family of distributions or a general decision variable. Suppose that $P(\xi|\theta)$ is a continuous distribution with a density $p(\xi|\theta)$. If - for each ξ - the function $f(\theta,\xi)p(\xi|\theta)$ admits the DC decomposition $f(\theta,\xi)p(\xi|\theta) = g(\theta,\xi)p(\xi|\theta) - h(\theta,\xi)p(\xi|\theta)$, then the above problem is DC: $\mathbb{E}_{P(\xi|\theta)}(f(\theta,\xi)) = \mathbb{E}_{P(\xi|\theta)}(g(\theta,\xi)) - \mathbb{E}_{P(\xi|\theta)}(h(\theta,\xi))$.

Suppose $h(\theta,\xi), p(\xi|\theta)$ are differentiable. The gradient of the second DC component is given by

$$\nabla \mathbb{E}_{P(\xi|\theta)}(h(\theta,\xi)) = \nabla \int h(\theta,\xi)p(\xi|\theta)d\xi = \mathbb{E}_{P(\xi|\theta)}\left(\nabla_\theta h(\theta,\xi) + h(\theta,\xi)\nabla_\theta \log p(\xi|\theta)\right).$$

Our contributions To begin, we develop a new stochastic algorithm called Markov chain stochastic DCA (MCSDCA) to handle a class of stochastic (nonsmooth) DC programs with endogenous uncertainty in the absence of i.i.d. samples. Asymptotic and non-asymptotic convergence analyses are established using the theory of the general state-space Markov chain and modern convex analysis. On

the non-asymptotic side, the algorithm - in expectation - promises to locate (nearly) ϵ-critical points with reasonable convergence rates if either G or H is L-smooth. On the asymptotic side, we get almost sure convergence to critical points where the L-smooth assumption of both G and H are relaxed, but this requires a greater level of accuracy when solving convex subproblems as well as more Markov chain iterations than in the non-asymptotic case. Second, we broaden our study to include Markov chains that are not always time-homogeneous. This is significant because some diffusion-based Markov chains that play a key role in sampling are time-inhomogeneous. Then, we show how to decompose a broad class of PDE-based regularization structures into DC decompositions that may be utilized to train deep neural networks. In the future, this could help to stimulate the use of the DCA-based approach to solve other nonconvex regularized deep learning challenges. Finally, thanks to the discovered DC structure, the MCSDCA is applied specifically to deep learning. There are two realizations of MCSDCA in this application. The first one is named MCSDCA-odLD (MCSDCA overdamped Langevin dynamics) where Markov chains are constructed based on the overdamped Langevin diffusion with Euler-Maruyama discretization. The second one is named MCSDCA-udLD (MCSDCA underdamped Langevin dynamics) where Markov chains are developed based on the underdamped Langevin diffusion with a novel discretization scheme introduced in [25]. We also discuss the implementation aspect of the MCSDCA-udLD in the context of deep neural networks (which is more complicated than the MCSDCA-odLD). Numerical experiments on standard neural networks including feed-forward neural networks, LSTM (long short-term memory), and RNN (recurrent neural networks) are postponed to the next chapter where we will study time series prediction problems arising in predictive maintenance.

4.2 Markov chain Stochastic DCA

In this section, all Markov chains are understood as time-homogeneous. Additional analysis for time-inhomogeneous Markov chains will be presented in Section 4.3.

4.2.1 Markov chain in general state space

Let $\mathcal{X}$ be some state space (possibly uncountable) and π be a distribution on $\mathcal{X}$. Let $\{Z_k : k \geq 0\}$ be a time-homogeneous $\mathcal{X}$-valued Markov chain associated with a Markov transition kernel $P(x, dy)$. We call π an invariant (or stationary) distribution of the Markov chain $\{Z_k\}$ if for all $x, y \in \mathcal{X}$,

$$\int_{x\in\mathcal{X}} \pi(dx)P(x, dy) = \pi(dy). \tag{4.3}$$

The distribution of Z_0 is called the initial distribution of the chain. The relation (4.3) literally reads: if the Markov chain starts from its invariant distribution, the distribution of all later elements will remain invariant. We denote $P^n(x, A) := \mathbb{P}(Z_n \in A|Z_0 = x)$ the n-step transition law of the chain. For a fixed x, $P^n(x, dy)$ describes the distribution of the n-th state of the Markov chain, Z_n, given the initial state $Z_0 = x$. We denote by $\mathbb{E}_x$ (resp. $\mathbb{E}_\pi$) the expectation given the chain starts at $Z_0 = x$ (resp. $Z_0 \sim \pi$). The invariant distribution π of a chain is called an *equilibrium distribution* if for π-almost all x, $\lim_{n\to\infty} P^n(x, A) = \pi(A)$, for all measurable sets A. If π is an invariant distribution of $\{Z_k\}$, under mild assumptions (irreducible, aperiodic), π is the unique invariant distribution and also an equilibrium distribution of the chain. In fact, a stronger convergence holds under the total variation (TV) distance

[125]. In other words, for n being large enough, Z_n looks similar to a sample drawn from π (but still different, nevertheless). In this research, we want to measure how fast this convergence occurs. We employ the notion of geometric ergodicity introduced as follows (see, e.g., [5]). For a π-integrable function f, we denote $\pi(f) := \int f(x)\pi(dx)$. For convenience, sometimes we write $\langle \pi, f \rangle$ with the same meaning as $\pi(f)$. Let $W : \mathcal{X} \to [1, +\infty)$, the W-norm of a function $h : \mathcal{X} \to \mathbb{R}$ is defined as $\|h\|_W = \sup_{x\in\mathcal{X}}\{|h(x)|/W(x)\}$. For two measures μ and ν on $\mathcal{X}$ with $\mu(W), \nu(W) < +\infty$, we define the W-norm of $\mu - \nu$ as $\|\nu - \mu\|_W = \sup_{\|f\|_W \leq 1} |\mu(f) - \nu(f)|$.

Geometric ergodicity Let $W : \mathcal{X} \to [1, +\infty)$ be a measurable function. The Markov chain associated with the Markov transition kernel P admits a unique invariant measure π with $\pi(W) < \infty$ and there exist $\varsigma > 0, 0 < \rho < 1$ such that for all $x \in \mathcal{X}, n \in \mathbb{N}$,

$$\|P^n(x, dy) - \pi(dy)\|_W \leq \varsigma W(x)\rho^n.$$

4.2.2 Problem's setting and assumptions

Firstly, we suppose that the optimal value (called F^*) of (4.1) is finite. Let Ξ be a (general) state space equipped with the underlying σ-algebra $\mathcal{M}$. For each $x \in \mathbb{R}^n$, $P(\xi|x)$ specifies a probability distribution over Ξ. Consider the problem (4.1), for each x, one subgradient of $\partial H(x)$ is of the form $\mathbb{E}_{P(\xi|x)}(v(x, \xi))$ for some $\mathbb{R}^n$-valued function $v = (v_1, v_2, \ldots, v_n)^\top$. We assume that one can construct a geometrically ergodic Markov chain (corresponding to a Markov kernel $P_x(\xi, d\xi')$) whose equilibrium distribution is $P(\xi|x)$. To be specific, the Markov chain is geometrically ergodic w.r.t. to some $W_x : \Xi \to [1, +\infty), \varsigma_x > 0, \rho_x \in (0, 1)$. We further assume some uniform boundedness as:

$$\begin{aligned}
&\sup_x \varsigma_x < +\infty, \ \sup_x \rho_x < 1, \\
&\sup_x \|v_j(x, \cdot) - \langle P(\cdot|x), v_j(x, \cdot)\rangle\|_{(W_x)^{1/2}} < +\infty \ \text{ for } j = \overline{1, n}, \ \sup_x \langle P(\cdot|x), W_x\rangle < +\infty.
\end{aligned}$$

Finally, we assume that $P(\xi|x)$ and $P_x(\xi, d\xi')$ have densities with respect to some common underlying measure on $(\Xi, \mathcal{M})$. It should be stressed that this assumption is very weak: it holds in most cases, where the underlying measure is usually either the counting measure (for countable state space) or the Lebesgue measure (for continuous state space).

4.2.3 Algorithmic design

To solve the program (4.1), we devise a Markov chain Stochastic DCA, whose major characteristics are as follows. Generally, at iteration k, a geometrically ergodic Markov chain $\xi_0 \to \xi_1 \to \xi_2 \to \ldots$ whose equilibrium distribution is $P(\xi|x^k)$ is simulated. Then, to address the burn-in period, some early samples are possibly discarded. The remaining samples are used to construct a *biased estimator* of a subgradient of H. Finally, the algorithm stochastically linearizes the second DC component and solves the resulting convex program up to a certain level of accuracy.

For technical convenience, the starting point of the Markov chain at iteration k, ξ_0^k, is chosen such that there is no randomness of ξ_0^k given x^k, i.e., $\xi_0^k = \varphi_k(x^k)$ where φ_k is a deterministic function.

Algorithm 7 Markov chain Stochastic DCA

Initialization. A starting point x^0, a sequence of positive numbers $\{\gamma_k\}$, a sequence of Markov chains' length $\{n_k\}$, the number of burn-in samples b, a symmetric positive definite matrix $A \succeq I$, set $k = 0$.

repeat

Run a Markov chain $\xi_0^k \to \xi_1^k \to \ldots \to \xi_{n_k-1}^k$ whose equilibrium distribution is $P(\xi|x^k)$.

Compute $y^k = \dfrac{1}{n_k - b} \sum_{i=b}^{n_k-1} v(x^k, \xi_i^k)$.

Solve the following convex problem up to a (stochastic) level of accuracy ϵ_k to obtain x^{k+1},

$$\min_x \left\{ G(x) - \langle x, y^k \rangle + \frac{\gamma_k}{2} \|x - x^k\|_A^2 \right\}. \tag{4.4}$$

$k = k + 1$.

until Stopping criterion

Remark 7. *1) The number b (burn-in) is the number of samples we wait before the Markov chain enters the sampling period where it produces samples looking similar to those drawn from the target distribution.*

2) The convex problem (4.4) *is only needed to be solved up to the (stochastic) level of accuracy ϵ_k, i.e.,*

$$G(x^{k+1}) - \langle x^{k+1}, y^k \rangle + \frac{\gamma_k}{2} \|x^{k+1} - x^k\|_A^2 \leq \inf_x \left\{ G(x) - \langle x, y^k \rangle + \frac{\gamma_k}{2} \|x - x^k\|_A^2 \right\} + \epsilon_k.$$

This feature is desirable in case G is complicated or even stochastic by nature.

4.2.4 Convergence results

Let $\mathcal{F}_k$ denote the information up to time k, $\mathcal{F}_k = \sigma(\{\xi_j^i\}_{j=0:n_i}^{i=0:k-1}, x^0, x^1, \ldots, x^k, \epsilon_1, \epsilon_2, \ldots, \epsilon_{k-1})$ which is the σ-algebra generated by the associated random variables. Firstly, the non-asymptotic convergence is presented in Theorem 13.

Theorem 13 (Non-asymptotic convergence)**.** *If the sequences of $\{n_k\}, \{\gamma_k\}$ are chosen in such a way that $n_k = \bar{b} + \lfloor k^\lambda \rfloor$, $\gamma_k = \gamma k^\beta$ for some $\bar{b} \geq b, \lambda > 0, \beta \in [0, 1), \gamma > 0$. Let $\nu > \beta$, suppose that - at iteration k - we solve the convex subproblem* (4.4) *up to the level of accuracy $k^{-\nu}$, i.e., $\mathbb{E}(\epsilon_k|\mathcal{F}_k) \leq k^{-\nu}$. Moreover, we assume that the sequences*

$$\{\mathbb{E}F(x^k)\}_k, \ \left\{ \frac{\mathbb{E}[W_{x^k}(\varphi_k(x^k))]}{n_k - b} \right\}_k \text{ are bounded.}$$

After T iterations, let κ be sampled from $\{1, 2, \ldots, T\}$ with probability $\mathbb{P}(\kappa = k) \propto k^\alpha$ for some $\alpha \geq \max\{2\beta + \lambda, \beta + \nu\}$, then

1) If G is L-smooth, x^κ is an ϵ-critical point in expectation with the following rate

$$\mathbb{E}\,\mathrm{dist}(\nabla G(x^\kappa), \partial H(x^\kappa)) = O\left(\frac{1}{T^{\min\{\frac{1-\beta}{2}, \frac{\lambda}{2}, \frac{\nu-\beta}{2}\}}} \right).$$

2) If H is L-smooth, x^κ is a nearly ϵ-critical point in expectation with the following rate:

$$\text{there exists } z^\kappa: \quad \mathbb{E}\|z^\kappa - x^\kappa\|_A = O\left(\frac{1}{T^{\min\{\frac{\beta+1}{2}, \frac{2\beta+\lambda}{2}, \frac{\beta+\nu}{2}\}}}\right),$$
$$\text{and} \quad \mathbb{E}\,\mathrm{dist}(\partial G(z^\kappa), \nabla H(z^\kappa)) = O\left(\frac{1}{T^{\min\{\frac{1-\beta}{2}, \frac{\lambda}{2}, \frac{\nu-\beta}{2}\}}}\right),$$

where z^κ can be defined explicitly as follows

$$z^\kappa = \arg\min\left\{G(x) - \langle \mathbb{E}_{P(\xi|x^\kappa)}(v(x^\kappa, \xi)), x\rangle + \frac{\gamma_\kappa}{2}\|x - x^\kappa\|_A^2\right\}.$$

To prove Theorem 13, we need the following two lemmas.

Lemma 3. *Let $\mathcal{X}$ be a general state space and $\{Z_k\}$ be a time-homogeneous $\mathcal{X}$-valued Markov chain (corresponding to a Markov transition kernel $P(x, dy)$) with an equilibrium distribution π. Assume that $\pi, P(x, dy)$ have densities with respect to some common underlying distribution. Let $W : \mathcal{X} \to [1, \infty)$ such that $\pi(W) < +\infty$. Suppose the Markov chain is geometrically egordic w.r.t. $W, \rho \in (0, 1), \varsigma > 0$. Let h be a function such that $\pi(|h|) < +\infty$ and denote $\tilde{h} = h - \pi(h)$. Then*

1) $|\mathbb{E}_x[\tilde{h}(Z_0)\tilde{h}(Z_s)]| \le \sqrt{2}\varsigma^{1/2}\rho^{s/2}W(x)\|\tilde{h}\|^2_{W^{1/2}}, \quad \forall x \in \mathcal{X}$.

2) $|\mathbb{E}_\pi[\tilde{h}(Z_0)\tilde{h}(Z_s)]| \le \sqrt{2}\varsigma^{1/2}\rho^{s/2}\pi(W)\|\tilde{h}\|^2_{W^{1/2}}$.

3) $|\mathbb{E}_x[\tilde{h}(Z_k)\tilde{h}(Z_{k+s})]| \le \sqrt{2}\varsigma^{1/2}\rho^{s/2}\pi(W)\|\tilde{h}\|^2_{W^{1/2}} + \sqrt{2}\varsigma^{3/2}\|\tilde{h}\|^2_{W^{1/2}}W(x)\rho^{k+s/2}, \quad \forall x \in \mathcal{X}$.

Proof of Lemma 3. The proof is standard (for example, see [5]), but let us include it here (and make it clearer) for the sake of completeness.

1) Since $\pi(\tilde{h}) = 0$, we have

$$\begin{aligned}|\mathbb{E}_x[\tilde{h}(Z_0)\tilde{h}(Z_s)]| &= \left|\int_{\mathcal{X}} \tilde{h}(x)\tilde{h}(y)P^s(x, dy) - \tilde{h}(x)\int_{\mathcal{X}} \tilde{h}(y)\pi(dy)\right| \\ &\le \|\tilde{h}\|_{W^{1/2}}W^{1/2}(x)\left|\int_{\mathcal{X}} \tilde{h}(y)P^s(x, dy) - \int_{\mathcal{X}} \tilde{h}(y)\pi(dy)\right|.\end{aligned}$$

Since $P^s(x, dy)$ and π have densities that - with a slight abuse of notation - we still call $P^s(x, \cdot)$ and π, respectively. Continuing the evaluation,

$$\begin{aligned}&|\mathbb{E}_x[\tilde{h}(Z_0)\tilde{h}(Z_s)]| \le \|\tilde{h}\|_{W^{1/2}}W^{1/2}(x)\left|\int_{\mathcal{X}} \tilde{h}(y)(P^s(x, y) - \pi(y))dy\right| \\ &\le \|\tilde{h}\|^2_{W^{1/2}}W^{1/2}(x)\left|\int_{\mathcal{X}} W^{1/2}(y)(P^s(x, y) - \pi(y))dy\right| \\ &\le \|\tilde{h}\|^2_{W^{1/2}}W^{1/2}(x)\left(\int_{\mathcal{X}} W(y)|P^s(x, y) - \pi(y)|dy\right)^{1/2}\left(\int_{\mathcal{X}} |P^s(x, y) - \pi(y)|dy\right)^{1/2} \\ &\le \sqrt{2}\|\tilde{h}\|^2_{W^{1/2}}W^{1/2}(x)\left(\int_{\mathcal{X}} W(y)|P^s(x, y) - \pi(y)|dy\right)^{1/2} \\ &\le \sqrt{2}\|\tilde{h}\|^2_{W^{1/2}}W^{1/2}(x)\left(\int_{\mathcal{X}} u(y)(P^s(x, y) - \pi(y))dy\right)^{1/2} \\ &\le \sqrt{2}\|\tilde{h}\|^2_{W^{1/2}}W^{1/2}(x)\|P^s(x, dy) - \pi(dy)\|^{1/2} \le \sqrt{2}\varsigma^{1/2}\rho^{s/2}\|\tilde{h}\|^2_{W^{1/2}}W(x),\end{aligned}$$

where

$$u(y) = \begin{cases} W(y) & \text{if } P^s(x,y) - \pi(y) \geq 0 \\ -W(y) & \text{otherwise.} \end{cases}$$

2) In property 1, the initial state Z_0 is fixed to be x, while $Z_0 \sim \pi$ in this case. We notice that

$$\begin{aligned} |\mathbb{E}_\pi[\tilde{h}(Z_0)\tilde{h}(Z_s)]| &= |\mathbb{E}_\pi\left(\mathbb{E}(\tilde{h}(Z_0)\tilde{h}(Z_s)|Z_0)\right)| \leq \mathbb{E}_\pi|\mathbb{E}_{Z_0}(\tilde{h}(Z_0)\tilde{h}(Z_s))| \\ &\leq \sqrt{2}\varsigma^{1/2}\rho^{s/2}\|\tilde{h}\|^2_{W^{1/2}}\mathbb{E}_\pi(W(Z_0)) = \sqrt{2}\varsigma^{1/2}\rho^{s/2}\|\tilde{h}\|^2_{W^{1/2}}\pi(W). \end{aligned}$$

3) This property evaluates the (auto)correlation between two arbitrary elements of the chain given the initial state. We have

$$\begin{aligned} &\left|\mathbb{E}_x[\tilde{h}(Z_k)\tilde{h}(Z_{k+s})] - \mathbb{E}_\pi[\tilde{h}(Z_k)\tilde{h}(Z_{k+s})]\right| \\ &= \left|\mathbb{E}_x\left(\mathbb{E}(\tilde{h}(Z_k)\tilde{h}(Z_{k+s})|Z_k)\right) - \mathbb{E}_\pi\left(\mathbb{E}(\tilde{h}(Z_k)\tilde{h}(Z_{k+s})|Z_k)\right)\right| \quad \triangleright \text{ law of total expectation} \\ &= \left|\mathbb{E}_x\left(\tilde{h}(Z_k)\mathbb{E}(\tilde{h}(Z_{k+s})|Z_k)\right) - \mathbb{E}_\pi\left(\tilde{h}(Z_k)\mathbb{E}(\tilde{h}(Z_{k+s})|Z_k)\right)\right| \quad \triangleright \text{pulling out what is known} \\ &= \left|\mathbb{E}_x\left(\tilde{h}(Z_k)\mathbb{E}_{Z_k}(\tilde{h}(Z_s))\right) - \mathbb{E}_\pi\left(\tilde{h}(Z_k)\mathbb{E}_{Z_k}(\tilde{h}(Z_s))\right)\right| \quad \triangleright \text{ due to the homogeneity of the chain} \\ &= \left|\int_y \tilde{h}(y)\mathbb{E}_y(\tilde{h}(Z_s))P^k(x,y)dy - \int_y \tilde{h}(y)\mathbb{E}_y(\tilde{h}(Z_s))\pi(y)dy\right| \quad \triangleright \ Z_0 \sim \pi \text{ then } Z_k \sim \pi \\ &= \left|\int_y \mathbb{E}_y(\tilde{h}(y)\tilde{h}(Z_s))(P^k(x,y) - \pi(y))dy\right| \\ &\leq \sqrt{2}\varsigma^{1/2}\rho^{s/2}\|\tilde{h}\|^2_{W^{1/2}}\|P^k(x,dy) - \pi(dy)\|_W \quad \triangleright \text{ thanks to property 1} \\ &\leq \sqrt{2}\varsigma^{3/2}\rho^{k+s/2}\|\tilde{h}\|^2_{W^{1/2}}W(x) \quad \triangleright \text{ due to geometric ergodicity.} \end{aligned}$$

On another hand, by noticing that $\mathbb{E}_\pi(\tilde{h}(Z_k)\tilde{h}(Z_{k+s})) = \mathbb{E}_\pi(\tilde{h}(Z_0)\tilde{h}(Z_s))$ since the chain starts from stationarity and it is homogeneous, the proof is complete thanks to property 2.

□

Lemma 4. *Under the same settings as in Lemma 3, the following inequality holds*

$$\mathbb{E}_x\left(\frac{1}{N-b}\sum_{i=b}^{N-1} h(Z_i) - \pi(h)\right)^2 \leq \frac{1+\sqrt{\rho}}{1-\sqrt{\rho}}\frac{\sqrt{2\varsigma}\|\tilde{h}\|^2_{W^{1/2}}}{N-b}\left(\frac{\varsigma W(x)}{(1-\rho)(N-b)} + \pi(W)\right).$$

Proof of Lemma 4. Expanding

$$\begin{aligned}
&\mathbb{E}_x\left(\frac{1}{N-b}\sum_{i=b}^{N-1}h(Z_i)-\pi(h)\right)^2\\
&=\frac{1}{(N-b)^2}\sum_{i=b}^{N-1}\mathbb{E}_x(h(Z_i)-\pi(h))^2+\frac{2}{(N-b)^2}\sum_{i=b}^{N-2}\sum_{j=i+1}^{N-1}\mathbb{E}_x(h(Z_i)-\pi(h))(h(Z_j)-\pi(h)))\\
&=\frac{1}{(N-b)^2}\underbrace{\sum_{i=b}^{N-1}\mathbb{E}_x\tilde{h}(Z_i)^2}_{\mathcal{P}\sim\text{variance}}+\frac{2}{(N-b)^2}\underbrace{\sum_{i=b}^{N-2}\sum_{j=i+1}^{N-1}\mathbb{E}_x\tilde{h}(Z_i)\tilde{h}(Z_j)}_{\mathcal{Q}\sim\text{autocovariance}}.
\end{aligned}$$

By using Lemma 3, item 3,

$$\begin{aligned}
\mathcal{P}&\leq\sum_{i=b}^{N-1}\left(\sqrt{2}\varsigma^{1/2}\pi(W)\|\tilde{h}\|^2_{W^{1/2}}+\sqrt{2}\varsigma^{3/2}\|\tilde{h}\|^2_{W^{1/2}}W(x)\rho^i\right)\\
&\leq\sqrt{2}\varsigma^{1/2}\pi(W)\|\tilde{h}\|^2_{W^{1/2}}(N-b)+\sqrt{2}\varsigma^{3/2}\|\tilde{h}\|^2_{W^{1/2}}W(x)\frac{\rho^b}{1-\rho},
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{Q}&\leq\sum_{i=b}^{N-2}\sum_{j=i+1}^{N-1}\left(\sqrt{2}\varsigma^{1/2}\rho^{(j-i)/2}\pi(W)\|\tilde{h}\|^2_{W^{1/2}}+\sqrt{2}\varsigma^{3/2}\|\tilde{h}\|^2_{W^{1/2}}W(x)\rho^{i+(j-i)/2}\right)\\
&=\sqrt{2}\varsigma^{1/2}\pi(W)\|\tilde{h}\|^2_{W^{1/2}}\sum_{i=b}^{N-2}\sum_{j=i+1}^{N-1}\rho^{(j-i)/2}+\sqrt{2}\varsigma^{3/2}\|\tilde{h}\|^2_{W^{1/2}}W(x)\sum_{i=b}^{N-2}\sum_{j=i+1}^{N-1}\rho^{i+(j-i)/2}.
\end{aligned}$$

Let $A=\sqrt{2}\varsigma^{1/2}\pi(W)\|\tilde{h}\|^2_{W^{1/2}}, B=\sqrt{2}\varsigma^{3/2}\|\tilde{h}\|^2_{W^{1/2}}W(x)$, by standard calculations, it holds

$$\mathcal{Q}\leq\frac{B}{(1-\sqrt{\rho})^2(1+\sqrt{\rho})}\rho^{(2b+1)/2}+\frac{A\sqrt{\rho}(N-b)}{1-\sqrt{\rho}}.$$

Putting together these evaluations, we obtain

$$\begin{aligned}
&\mathbb{E}_x\left(\frac{1}{N-b}\sum_{i=b}^{N-1}h(Z_i)-\pi(h)\right)^2\\
&\leq\frac{1}{(N-b)^2}\left[A(N-b)+B\frac{\rho^b}{1-\rho}+\frac{2B}{(1-\sqrt{\rho})^2(1+\sqrt{\rho})}\rho^{(2b+1)/2}+\frac{2A\sqrt{\rho}(N-b)}{1-\sqrt{\rho}}\right]\\
&=\frac{B\rho^b}{(1-\sqrt{\rho})^2(N-b)^2}+\frac{A(1+\sqrt{\rho})}{(1-\sqrt{\rho})(N-b)}\\
&\leq\frac{B}{(1-\sqrt{\rho})^2(N-b)^2}+\frac{A(1+\sqrt{\rho})}{(1-\sqrt{\rho})(N-b)}\\
&\leq\frac{1+\sqrt{\rho}}{1-\sqrt{\rho}}\frac{\sqrt{2\varsigma}\|\tilde{h}\|^2_{W^{1/2}}}{N-b}\left(\frac{\varsigma W(x)}{(1-\rho)(N-b)}+\pi(W)\right).
\end{aligned}$$

□

We now prove Theorem 13 as follows.

Proof of Theorem 13. Firstly, we denote

$$z^k = \arg\min \left\{ G(x) - \langle \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)), x\rangle + \frac{\gamma_k}{2}\|x - x^k\|_A^2 \right\}.$$

By the first-order optimality condition of z^k,

$$0 \in \partial G(z^k) - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)) + \gamma_k A(z^k - x^k),$$

or equivalently,

$$\mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)) - \gamma_k A(z^k - x^k) \in \partial G(z^k).$$

Consequently,

$$G(x^k) \geq G(z^k) + \langle \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)) - \gamma_k A(z^k - x^k), x^k - z^k\rangle$$

or

$$G(x^k) - \langle \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)), x^k\rangle \geq G(z^k) - \langle \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)), z^k\rangle + \gamma_k\|z^k - x^k\|_A^2. \tag{4.5}$$

By the definition of x^{k+1},

$$G(z^k) - \langle y^k, z^k\rangle + \frac{\gamma_k}{2}\|z^k - x^k\|_A^2 + \epsilon_k \geq G(x^{k+1}) - \langle y^k, x^{k+1}\rangle + \frac{\gamma_k}{2}\|x^{k+1} - x^k\|_A^2. \tag{4.6}$$

From (4.5) and (4.6),

$$\begin{aligned} G(x^k) - \langle y^k, z^k\rangle - \langle \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)), x^k\rangle + \frac{\gamma_k}{2}\|z^k - x^k\|_A^2 + \epsilon_k \geq G(x^{k+1}) \\ - \langle \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)), z^k\rangle - \langle y^k, x^{k+1}\rangle + \gamma_k\|z^k - x^k\|_A^2 + \frac{\gamma_k}{2}\|x^{k+1} - x^k\|_A^2, \end{aligned}$$

or equivalently,

$$\begin{aligned} &G(x^k) - \langle y^k, z^k - x^{k+1}\rangle + \langle \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)), z^k - x^k\rangle + \epsilon_k \\ &\geq G(x^{k+1}) + \frac{\gamma_k}{2}\|z^k - x^k\|_A^2 + \frac{\gamma_k}{2}\|x^{k+1} - x^k\|_A^2. \end{aligned}$$

By the convexity of H,

$$\begin{aligned}
&\langle \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)), z^k - x^k\rangle + \langle y^k, x^{k+1} - z^k\rangle \\
&= \langle \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)), x^{k+1} - x^k\rangle + \langle \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi)), z^k - x^{k+1}\rangle + \langle y^k, x^{k+1} - z^k\rangle \\
&\leq H(x^{k+1}) - H(x^k) + \langle x^{k+1} - z^k, y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\rangle \\
&\leq H(x^{k+1}) - H(x^k) + \frac{\gamma_k}{8}\|x^{k+1} - z^k\|^2 + \frac{2}{\gamma_k}\|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\|^2 \\
&\leq H(x^{k+1}) - H(x^k) + \frac{\gamma_k}{8}\|x^{k+1} - z^k\|_A^2 + \frac{2}{\gamma_k}\|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\|^2 \\
&\leq H(x^{k+1}) - H(x^k) + \frac{\gamma_k}{4}\|x^{k+1} - x^k\|_A^2 + \frac{\gamma_k}{4}\|z^k - x^k\|_A^2 + \frac{2}{\gamma_k}\|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\|^2.
\end{aligned}$$

Therefore, we arrive at

$$\frac{\gamma_k}{4}\|z^k - x^k\|_A^2 + \frac{\gamma_k}{4}\|x^{k+1} - x^k\|_A^2 \leq F(x^k) - F(x^{k+1}) + \frac{2}{\gamma_k}\|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\|^2 + \epsilon_k. \tag{4.7}$$

Expanding

$$\begin{aligned}
\|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\|^2 &= \left\| \frac{1}{n_k - b}\sum_{i=b}^{n_k-1} v(x^k,\xi_i^k) - \mathbb{E}_{P(\xi|x^k)}v(x^k,\xi)\right\|^2 \\
&= \sum_{j=1}^{n}\left(\frac{1}{n_k - b}\sum_{i=b}^{n_k-1} v_j(x^k,\xi_i^k) - \mathbb{E}_{P(\xi|x^k)}v_j(x^k,\xi)\right)^2.
\end{aligned}$$

For $j \in \{1, 2, \ldots, n\}$, by applying Lemma 4,

$$\begin{aligned}
&\mathbb{E}_{\xi_0^k}\left(\frac{1}{n_k - b}\sum_{i=b}^{n_k-1} v_j(x^k,\xi_i^k) - \mathbb{E}_{P(\xi|x^k)}v_j(x^k,\xi)\right)^2 \\
&\leq \frac{1 + \sqrt{\rho_{x^k}}}{1 - \sqrt{\rho_{x^k}}}\frac{\sqrt{2\varsigma_{x^k}}}{n_k - b}\|\tilde{v}_j(x^k,\cdot)\|_{W_{x^k}^{1/2}}^2\left(\frac{\varsigma_{x^k}W_{x^k}(\varphi_k(x^k))}{(1-\rho_{x^k})(n_k - b)} + \langle P(\cdot|x^k), W_{x^k}\rangle\right).
\end{aligned}$$

By the boundedness assumptions, it holds, for some universal $B > 0$ independent of k,

$$\mathbb{E}\left(\frac{1}{n_k - b}\sum_{i=b}^{n_k-1} v_j(x^k,\xi_i^k) - \mathbb{E}_{P(\xi|x^k)}v_j(x^k,\xi)\right)^2 \leq \frac{B}{n_k - b}. \tag{4.8}$$

Denoting $\bar{n}_k = n_k - b$, from (4.7) and (4.8), for some $C > 0$,

$$\frac{\gamma_k}{4}\mathbb{E}\|z^k - x^k\|_A^2 \leq \mathbb{E}F(x^k) - \mathbb{E}F(x^{k+1}) + \frac{C}{\gamma_k\bar{n}_k} + \mathbb{E}(\epsilon_k). \tag{4.9}$$

Let $w_k = k^\alpha$ (where $\alpha > 0$), (4.9) is equivalent to

$$\frac{w_k}{4}\mathbb{E}\|z^k - x^k\|_A^2 \leq \frac{w_k}{\gamma_k}(\mathbb{E}F(x^k) - \mathbb{E}F(x^{k+1})) + \frac{Cw_k}{\gamma_k^2 \bar{n}_k} + \frac{w_k \mathbb{E}(\epsilon_k)}{\gamma_k}.$$

Taking sum from $k = 1$ to T,

$$\sum_{k=1}^{T} \frac{w_k}{4}\mathbb{E}\|z^k - x^k\|_A^2 \leq \sum_{k=1}^{T} \frac{w_k}{\gamma_k}(\mathbb{E}F(x^k) - \mathbb{E}F(x^{k+1})) + \sum_{k=1}^{T} \frac{Cw_k}{\gamma_k^2 \bar{n}_k} + \sum_{k=1}^{T} \frac{w_k \mathbb{E}(\epsilon_k)}{\gamma_k}.$$

The rest of the proof follows the arguments in [134], which - in turn - are based on [23]. We have the following evaluations

$$\begin{aligned}
&\sum_{k=1}^{T} \frac{w_k}{\gamma_k}(\mathbb{E}F(x^k) - \mathbb{E}F(x^{k+1})) = \sum_{k=1}^{T} \frac{w_k}{\gamma_k}\mathbb{E}F(x^k) - \sum_{k=1}^{T} \frac{w_k}{\gamma_k}\mathbb{E}F(x^{k+1}) \\
&= \sum_{k=1}^{T} \left(\frac{w_{k-1}}{\gamma_{k-1}}\mathbb{E}F(x^k) - \frac{w_k}{\gamma_k}\mathbb{E}F(x^{k+1})\right) + \sum_{k=1}^{T} \left(\frac{w_k}{\gamma_k} - \frac{w_{k-1}}{\gamma_{k-1}}\right)\mathbb{E}F(x^k) \\
&= \frac{w_0}{\gamma_0}\mathbb{E}F(x^1) - \frac{w_T}{\gamma_T}\mathbb{E}F(x^{T+1}) + \sum_{k=1}^{T} \left(\frac{w_k}{\gamma_k} - \frac{w_{k-1}}{\gamma_{k-1}}\right)\mathbb{E}F(x^k) \\
&= \sum_{k=1}^{T} \left(\frac{w_k}{\gamma_k} - \frac{w_{k-1}}{\gamma_{k-1}}\right)(\mathbb{E}F(x^k) - \mathbb{E}F(x^{T+1})) \\
&\leq \sum_{k=1}^{T} \underbrace{\left(\frac{w_k}{\gamma_k} - \frac{w_{k-1}}{\gamma_{k-1}}\right)}_{\geq 0}(\mathbb{E}F(x^k) - F^*) \leq \sum_{k=1}^{T} \left(\frac{w_k}{\gamma_k} - \frac{w_{k-1}}{\gamma_{k-1}}\right) D = D\frac{w_T}{\gamma_T},
\end{aligned}$$

for some $D > 0$. Therefore,

$$\sum_{k=1}^{T} \frac{w_k}{4}\mathbb{E}\|z^k - x^k\|_A^2 \leq D\frac{w_T}{\gamma_T} + C\sum_{k=1}^{T} \frac{w_k}{\gamma_k^2 \bar{n}_k} + \sum_{k=1}^{T} \frac{w_k \mathbb{E}(\epsilon_k)}{\gamma_k}. \tag{4.10}$$

By denoting $\bar{w}_k = \dfrac{w_k}{w_1 + w_2 + \ldots + w_T}$, (4.10) becomes

$$\sum_{k=1}^{T} \frac{\bar{w}_k}{4}\mathbb{E}\|z^k - x^k\|_A^2 \leq D\frac{\bar{w}_T}{\gamma_T} + C\sum_{k=1}^{T} \frac{\bar{w}_k}{\gamma_k^2 \bar{n}_k} + \sum_{k=1}^{T} \frac{\bar{w}_k \mathbb{E}(\epsilon_k)}{\gamma_k}.$$

Case 1: G is L-smooth.

From the first-order optimality,

$$0 = \nabla G(z^k) - \mathbb{E}_{P(\xi|x^k)}(v(x^k, \xi)) + \gamma_k A(z^k - x^k).$$

Therefore, by noting there exists $c > 0$ such that $A \preceq cI$,

$$\begin{aligned}
&d(\nabla G(x^k), \partial H(x^k)) \leq \|\nabla G(x^k) - \mathbb{E}_{P(\xi|x^k)}(v(x^k, \xi))\| \\
&\leq \|\nabla G(x^k) - \nabla G(z^k)\| + \|\nabla G(z^k) - \mathbb{E}_{P(\xi|x^k)}(v(x^k, \xi))\| \\
&\leq L\|x^k - z^k\| + \gamma_k \sqrt{(z^k - x^k)^\top A^2 (z^k - x^k)} \\
&\leq L\|x^k - z^k\|_A + \gamma_k \sqrt{c} \sqrt{(z^k - x^k)^\top A (z^k - x^k)} \\
&= L\|x^k - z^k\|_A + \gamma_k \sqrt{c} \|x^k - z^k\|_A = (L + \gamma_k \sqrt{c}) \|x^k - z^k\|_A,
\end{aligned}$$

which implies

$$\mathbb{E}d(\nabla G(x^k), \partial H(x^k)) \leq (L + \gamma_k \sqrt{c}) \mathbb{E}\|x^k - z^k\|_A.$$

Since $\mathbb{P}(\kappa = k) = \bar{w}_k$,

$$\begin{aligned}
&\mathbb{E}d(\nabla G(x^\kappa), \partial H(x^\kappa)) = \sum_{k=1}^{T} \bar{w}_k \mathbb{E}d(\nabla G(x^k), \partial H(x^k)) \\
&\leq (L + \gamma_T \sqrt{c}) \sum_{k=1}^{T} \bar{w}_k \mathbb{E}\|x^k - z^k\|_A \\
&\leq (L + \gamma_T \sqrt{c}) \sum_{k=1}^{T} \bar{w}_k \left(\mathbb{E}\|x^k - z^k\|_A^2 \right)^{1/2} \\
&\leq (L + \gamma_T \sqrt{c}) \left(\sum_{k=1}^{T} \bar{w}_k \right)^{1/2} \left(\sum_{k=1}^{T} \bar{w}_k \mathbb{E}\|x^k - z^k\|_A^2 \right)^{1/2} \\
&\leq (L + \gamma_T \sqrt{c}) \left(4D \frac{\bar{w}_T}{\gamma_T} + 4C \sum_{k=1}^{T} \frac{\bar{w}_k}{\gamma_k^2 \bar{n}_k} + 4 \sum_{k=1}^{T} \frac{\bar{w}_k \mathbb{E}(\epsilon_k)}{\gamma_k} \right)^{1/2}. \qquad (4.11)
\end{aligned}$$

The next evaluations will employ the standard calculus: $\sum_{s=1}^{S} s^\alpha \geq \int_0^S x^\alpha dx = \dfrac{1}{\alpha + 1} S^{\alpha+1}$.

Now by plugging $\gamma_k = \gamma \cdot k^\beta$, $w_k = k^\alpha$, $\bar{n}_k = (\bar{b} - b) + \lfloor k^\lambda \rfloor \geq \lfloor k^\lambda \rfloor$ to (4.11), where $\gamma > 0$, $0 \leq \beta < 1$, $\alpha \geq \max\{2\beta + \lambda, \beta + \nu\}$, $\lambda > 0$, and noticing that $\mathbb{E}(\epsilon_k | \mathcal{F}_k) \leq k^{-\nu}$ and $\nu > \beta$, it holds

$$\mathbb{E}d(\nabla G(x^\kappa), \partial H(x^\kappa)) \leq (L + \gamma_T\sqrt{c}) \left(4D\frac{\bar{w}_T}{\gamma_T} + 4C\sum_{k=1}^{T} \frac{\bar{w}_k}{\gamma_k^2 \bar{n}_k} + 4\sum_{k=1}^{T} \frac{\bar{w}_k \mathbb{E}(\epsilon_k)}{\gamma_k} \right)^{1/2}$$

$$\leq 2(L + \gamma \cdot T^\beta\sqrt{c}) \left(D\frac{T^\alpha}{\gamma \cdot T^\beta \sum_{k=1}^{T} k^\alpha} + \frac{C}{\sum_{k=1}^{T} k^\alpha} \sum_{k=1}^{T} \frac{k^\alpha}{\gamma^2 k^{2\beta} \lfloor k^\lambda \rfloor} + \frac{1}{\sum_{k=1}^{T} k^\alpha} \sum_{k=1}^{T} \frac{k^\alpha \mathbb{E}(\epsilon_k)}{\gamma k^\beta} \right)^{1/2}$$

$$\leq 2(L + \gamma \cdot T^\beta\sqrt{c}) \left(\frac{D(\alpha+1)}{\gamma \cdot T^{\beta+1}} + \frac{C(\alpha+1)}{\gamma^2 T^{\alpha+1}} \sum_{k=1}^{T} \frac{k^{\alpha-2\beta}}{c_\lambda k^\lambda} + \frac{\alpha+1}{\gamma T^{\alpha+1}} \sum_{k=1}^{T} k^{\alpha-\beta-\nu} \right)^{1/2}, \ c_\lambda > 0$$

$$\leq 2(L + \gamma \cdot T^\beta\sqrt{c}) \left(\frac{D(\alpha+1)}{\gamma \cdot T^{\beta+1}} + \frac{C(\alpha+1)}{\gamma^2 c_\lambda T^{\alpha+1}} T^{\alpha-2\beta-\lambda+1} + \frac{\alpha+1}{\gamma T^{\alpha+1}} T^{\alpha-\beta-\nu+1} \right)^{1/2}$$

$$= 2(L + \gamma \cdot T^\beta\sqrt{c}) \left(\frac{D(\alpha+1)}{\gamma \cdot T^{\beta+1}} + \frac{C(\alpha+1)}{\gamma^2 c_\lambda T^{2\beta+\lambda}} + \frac{\alpha+1}{\gamma T^{\beta+\nu}} \right)^{1/2}.$$

Therefore, x^κ is an ϵ-critical point in expectation with the following rate

$$\mathbb{E}d(\nabla G(x^\kappa), \partial H(x^\kappa)) = O\left(\frac{1}{T^{\min\{\frac{1-\beta}{2}, \frac{\lambda}{2}, \frac{\nu-\beta}{2}\}}} \right).$$

Case 2: H is L-smooth.

$$d(\partial G(z^k), \nabla H(z^k)) \leq d(\partial G(z^k), \nabla H(x^k)) + \|\nabla H(x^k) - \nabla H(z^k)\|$$
$$\leq \gamma_k \sqrt{(z^k - x^k)^\top A^2 (z^k - x^k)} + L\|x^k - z^k\|$$
$$\leq \gamma_k\sqrt{c}\|z^k - x^k\|_A + L\|x^k - z^k\|_A \leq (L + \gamma_k\sqrt{c})\|z^k - x^k\|_A.$$

Similarly, x^κ is a nearly ϵ-critical point in expectation with the following rate

$$\mathbb{E}d(\partial G(z^\kappa), \nabla H(z^\kappa)) = O\left(\frac{1}{T^{\min\{\frac{1-\beta}{2}, \frac{\lambda}{2}, \frac{\nu-\beta}{2}\}}} \right)$$

where

$$\mathbb{E}\|z^\kappa - x^\kappa\|_A = O\left(\frac{1}{T^{\min\{\frac{\beta+1}{2}, \frac{2\beta+\lambda}{2}, \frac{\beta+\nu}{2}\}}} \right).$$

□

We next obtain the almost sure convergence results stated in Theorem 14.

Theorem 14 (Asymptotic convergence)**.** *Let $\gamma > 0$ and set $\gamma_k = \gamma$, for all $k \in \mathbb{N}$. If the sequence $\{n_k\}$ is chosen such that*

$$\sum_{k=1}^{\infty} \frac{1}{n_k - b} < +\infty, \ \textit{and it is further supposed that } \exists R > 0 : \forall k, \frac{W_{x^k}(\varphi_k(x^k))}{n_k - b} \leq R \ \textit{a.s.},$$

and the sequence of random errors in solving convex subproblems satisfies $\sum_{k=1}^{\infty} \mathbb{E}(\epsilon_k) < +\infty$, then

1) The sequence of the objective values $\{F(x^k)\}$ is convergent almost surely.

2) $\sum_{k=1}^{\infty} \|x^{k+1} - x^k\|_A^2 < +\infty$ *almost surely.*

3) Assume that $\{x^k\}$ *and* $\{y^k\}$ *are bounded almost surely, then, almost surely, every limit point of* $\{x^k\}$ *is a critical point of* F.

Proof of Theorem 14. 1) It follows from (4.7) that

$$\begin{aligned}
&\frac{\gamma}{4}\|z^k - x^k\|_A^2 + \frac{\gamma}{4}\mathbb{E}(\|x^{k+1} - x^k\|_A^2|\mathcal{F}_k) \leq F(x^k) - \mathbb{E}(F(x^{k+1})|\mathcal{F}_k) \\
&+ \frac{2}{\gamma}\mathbb{E}\left(\|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\|^2|\mathcal{F}_k\right) + \mathbb{E}(\epsilon_k|\mathcal{F}_k) \\
&= F(x^k) - \mathbb{E}(F(x^{k+1})|\mathcal{F}_k) + \frac{2}{\gamma}\mathbb{E}_{\xi_0^k}\left(\|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\|^2\right) + \mathbb{E}(\epsilon_k|\mathcal{F}_k).
\end{aligned}$$

As a result, for some $C > 0$,

$$\mathbb{E}(F(x^{k+1})|\mathcal{F}_k) + \frac{\gamma}{4}\mathbb{E}(\|x^{k+1} - x^k\|_A^2|\mathcal{F}_k) \leq F(x^k) + \frac{C}{\gamma(n_k - b)} + \mathbb{E}(\epsilon_k|\mathcal{F}_k). \tag{4.12}$$

Noting that the condition $\sum_{k=1}^{\infty}\mathbb{E}(\epsilon_k) < +\infty$ implies $\sum_{k=1}^{\infty}\mathbb{E}(\epsilon_k|\mathcal{F}_k) < +\infty$ a.s. By supermartingale theorem [110, Theorem 1], $\{F(x^k)\}$ is convergent almost surely.

2) From (4.12),

$$\frac{\gamma}{4}\mathbb{E}(\|x^{k+1} - x^k\|_A^2) \leq \mathbb{E}(F(x^k)) - \mathbb{E}(F(x^{k+1})) + \frac{C}{\gamma(n_k - b)} + \mathbb{E}(\epsilon_k).$$

It follows that

$$\mathbb{E}\left(\sum_{k=1}^{\infty}\|x^{k+1} - x^k\|_A^2\right) < +\infty,$$

implying

$$\sum_{k=1}^{\infty}\|x^{k+1} - x^k\|_A^2 < +\infty \text{ a.s.}$$

3) It has been established from the proof of Theorem 1 that

$$\mathbb{E}\|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\|^2 \leq \frac{E}{n_k - b}, \quad \text{for some } E > 0.$$

As a consequence,

$$\sum_{k=1}^{\infty}\mathbb{E}\|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\|^2 < +\infty,$$

which yields

$$\|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k,\xi))\| \to 0 \text{ a.s.}$$

We denote a set of events

$$\mathcal{S} := \left\{ \sum_{k=1}^{\infty} \|x^{k+1} - x^k\|_A^2 < \infty, \|y^k - \mathbb{E}_{P(\xi|x^k)}(v(x^k, \xi))\| \to 0, \epsilon_k \to 0, \{x^k\}, \{y^k\} \text{ are bounded} \right\}.$$

Obviously, $\mathbb{P}(\mathcal{S}) = 1$. Now fix an event $\omega \in \mathcal{S}$, giving rise to realization sequences of $\{x^k\}$ and $\{y^k\}$, in which - for simplicity - we still call these realizations $\{x^k\}$ and $\{y^k\}$. Let x^* be a limit point of $\{x^k\}$, there exists a subsequence $\{x^{l_k}\}$ converging to x^*. Since $\|x^{l_k+1} - x^{l_k}\| \to 0$, we also have $x^{l_k+1} \to x^*$. From the definition of x^{l_k+1},

$$G(x^{l_k+1}) - \langle y^{l_k}, x^{l_k+1} \rangle + \frac{\gamma}{2}\|x^{l_k+1} - x^{l_k}\|_A^2 \leq G(u) - \langle y^{l_k}, u \rangle + \frac{\gamma}{2}\|u - x^{l_k}\|_A^2 + \epsilon_{l_k}, \forall u \in \mathbb{R}^n.$$

By passing to subsequence if necessary, we can assume that $y^{l_k} \to y^*$. Taking $\liminf$ on two sides of the above inequality and noting that G is lower semicontinuous,

$$G(x^*) - \langle y^*, x^* \rangle \leq G(u) - \langle y^*, u \rangle + \frac{\gamma}{2}\|u - x^*\|_A^2.$$

Hence,

$$y^* \in \partial \left(G + \frac{\gamma}{2}\| \cdot - x^*\|_A^2 \right)(x^*),$$

which implies

$$y^* \in \partial G(x^*) + \gamma A(x^* - x^*) = \partial G(x^*).$$

On the other hand, from the definition of $\mathcal{S}$, $\|y^{l_k} - \mathbb{E}_{P(\xi|x^{l_k})}(v(x^{l_k}, \xi))\| \to 0$. As a result,

$$\mathbb{E}_{P(\xi|x^{l_k})}(v(x^{l_k}, \xi)) \to y^*.$$

In combination with $\mathbb{E}_{P(\xi|x^{l_k})}(v(x^{l_k}, \xi)) \in \partial H(x^{l_k})$, it holds $y^* \in \partial H(x^*)$ thanks to the closedness of the subdifferential map. Therefore, as $\partial G(x^*) \cap \partial H(x^*) \neq \emptyset$, x^* is a critical point of $F = G - H$.

□

4.3 Additional analysis for the time-inhomogeneous case

In this section, we consider the case where each Markov chain can be time-inhomogeneous. In practice, time-inhomogeneous Markov chains play an important role as they offer more flexibility and sometimes give acceleration to the equilibrium distribution in comparison with the time-homogeneous case.

Different from the time-homogeneous case, the transition law of a time-inhomogeneous Markov chain can vary over time, i.e., the chain is characterized by a sequence of Markov kernels. We adapt the notion of geometric ergodicity to the time-inhomogeneous case as follows.

Let $\mathcal{X}$ be a general state space and π be a distribution on $\mathcal{X}$. Let $\{Z_k : k \geq 0\}$ be a time-inhomogeneous $\mathcal{X}$-valued Markov chain. We denote $P_k^{k+s}(x, dy)$ the probability distribution of Z_{k+s} given $Z_k = x$. When $k = 0$, we simply write $P^s(x, dy)$. The Markov chain is said to be geometrically

ergodic to π with respect to $W : \mathcal{X} \to [1, +\infty)$ (where $\pi(W) < +\infty$) and $\varsigma > 0, 0 < \rho < 1$ if

$$\|P_k^{k+s}(x, dy) - \pi(dy)\|_W \leq \varsigma W(x)\rho^s, \quad \forall x \in \mathcal{X}, \text{ and } \forall k, s \in \mathbb{N}.$$

Apart from the Markov chains being now time-inhomogeneous and geometrically ergodic in the above sense, other settings in subsection 4.2.2 remain unchanged. Furthermore, the convergence results stated in Theorems 13 and 14 are still valid in this new setting.

Theorem 15. *With the above setting, Theorems 13 and 14 hold true.*

Proof. In essence, it is sufficient to verify property 3) of Lemma 3. Everything else remains the same.

We prove:

$$\left|\mathbb{E}_x\left(\tilde{h}(Z_k)\tilde{h}(Z_{k+s})\right)\right| \leq \sqrt{2}\varsigma^{3/2}\|\tilde{h}\|^2_{W^{1/2}}W(x)\rho^{k+s/2} + \sqrt{2}\varsigma^{1/2}\rho^{s/2}\|\tilde{h}\|^2_{W^{1/2}}\pi(W).$$

It is observed that

$$\begin{aligned}
\mathbb{E}_x\left(\tilde{h}(Z_k)\tilde{h}(Z_{k+s})\right) &= \mathbb{E}\left(\tilde{h}(Z_k)\tilde{h}(Z_{k+s})|Z_0 = x\right) \\
&= \mathbb{E}\left(\mathbb{E}\left(\tilde{h}(Z_k)\tilde{h}(Z_{k+s})|Z_k, Z_0 = x\right)|Z_0 = x\right) \\
&= \mathbb{E}\left(\mathbb{E}\left(\tilde{h}(Z_k)\tilde{h}(Z_{k+s})|Z_k\right)|Z_0 = x\right) \\
&= \mathbb{E}\left(\tilde{h}(Z_k)\mathbb{E}\left(\tilde{h}(Z_{k+s})|Z_k\right)|Z_0 = x\right) \\
&= \mathbb{E}\left(\tilde{h}(Z_k)\int \tilde{h}(z)P_k^{k+s}(Z_k, dz)|Z_0 = x\right) \\
&= \int \tilde{h}(y)\int \tilde{h}(z)P_k^{k+s}(y, dz)P^k(x, dy) \\
&= \int\int \tilde{h}(y)\tilde{h}(z)P_k^{k+s}(y, z)P^k(x, y)dzdy. \qquad (4.13)
\end{aligned}$$

On the other hand,

$$\begin{aligned}
&\left|\int\int \tilde{h}(y)\tilde{h}(z)P_k^{k+s}(y, z)P^k(x, y)dzdy - \int\int \tilde{h}(y)\tilde{h}(z)P_k^{k+s}(y, z)\pi(y)dzdy\right| \\
&= \left|\int\int \tilde{h}(y)\tilde{h}(z)P_k^{k+s}(y, z)(P^k(x, y) - \pi(y))dzdy\right| \\
&\leq \underbrace{\left|\int\int \tilde{h}(y)\tilde{h}(z)\left(P_k^{k+s}(y, z) - \pi(z)\right)\left(P^k(x, y) - \pi(y)\right)dzdy\right|}_{\mathcal{A}} \\
&\quad + \underbrace{\left|\int\int \tilde{h}(y)\tilde{h}(z)\pi(z)\left(P^k(x, y) - \pi(y)\right)dzdy\right|}_{\mathcal{B}}. \qquad (4.14)
\end{aligned}$$

We prove $\mathcal{B} = 0$. Indeed,

$$\begin{aligned}
&\int\int \tilde{h}(y)\tilde{h}(z)\pi(z)\left(P^k(x,y)-\pi(y)\right)dzdy \\
&= \int\int \tilde{h}(y)\tilde{h}(z)\pi(z)P^k(x,y)dzdy - \int\int \tilde{h}(y)\tilde{h}(z)\pi(z)\pi(y)dzdy \\
&= \int \tilde{h}(y)P^k(x,y)\underbrace{\int \tilde{h}(z)\pi(z)dz}_{=0}dy - \int \tilde{h}(y)\pi(y)\underbrace{\int \tilde{h}(z)\pi(z)dz}_{=0}dy = 0.
\end{aligned}$$

We evaluate $\mathcal{A}$ as follows

$$\begin{aligned}
\mathcal{A} &\le \int\int \left|\tilde{h}(y)\tilde{h}(z)\left(P_k^{k+s}(y,z)-\pi(z)\right)\left(P^k(x,y)-\pi(y)\right)\right|dzdy \\
&\le \|\tilde{h}\|_{W^{1/2}}^2 \int\int W^{1/2}(y)W^{1/2}(z)\left|P_k^{k+s}(y,z)-\pi(z)\right|\cdot\left|P^k(x,y)-\pi(y)\right|dzdy \\
&\le \|\tilde{h}\|_{W^{1/2}}^2 \int W^{1/2}(y)\left|P^k(x,y)-\pi(y)\right|\underbrace{\int W^{1/2}(z)\left|P_k^{k+s}(y,z)-\pi(z)\right|dz}_{\Delta}dy.
\end{aligned}$$

It follows from Hölder's inequality that

$$\begin{aligned}
\Delta &\le \left(\int W(z)\left|P_k^{k+s}(y,z)-\pi(z)\right|dz\right)^{1/2}\left(\int \left|P_k^{k+s}(y,z)-\pi(z)\right|dz\right)^{1/2} \\
&\le \sqrt{2}\|P_k^{k+s}(y,dz)-\pi(dz)\|_W^{1/2} \le \sqrt{2}\varsigma^{1/2}\rho^{s/2}W(y)^{1/2}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathcal{A} &\le \sqrt{2}\varsigma^{1/2}\|\tilde{h}\|_{W^{1/2}}^2\rho^{s/2}\int W(y)\left|P^k(x,y)-\pi(y)\right|dy \\
&\le \sqrt{2}\varsigma^{1/2}\|\tilde{h}\|_{W^{1/2}}^2\rho^{s/2}\left\|P^k(x,dy)-\pi(dy)\right\|_W \\
&\le \sqrt{2}\varsigma^{3/2}\|\tilde{h}\|_{W^{1/2}}^2 W(x)\rho^{k+s/2}.
\end{aligned} \tag{4.15}$$

From (4.13), (4.14), (4.15),

$$\left|\mathbb{E}_x\left(\tilde{h}(Z_k)\tilde{h}(Z_{k+s})\right)\right| \le \sqrt{2}\varsigma^{3/2}\|\tilde{h}\|_{W^{1/2}}^2 W(x)\rho^{k+s/2} + \underbrace{\left|\int\int \tilde{h}(y)\tilde{h}(z)P_k^{k+s}(y,z)\pi(y)dzdy\right|}_{\diamond}. \tag{4.16}$$

On another hand,

$$\begin{aligned}
\diamond &= \left|\int \tilde{h}(y)\pi(y)\int \tilde{h}(z)P_k^{k+s}(y,z)dzdy\right| \\
&= \left|\int \tilde{h}(y)\pi(y)\left(\int \tilde{h}(z)P_k^{k+s}(y,z)dz - \int \tilde{h}(z)\pi(z)dz\right)dy\right| \\
&\le \int |\tilde{h}(y)|\pi(y)\underbrace{\left|\int \tilde{h}(z)P_k^{k+s}(y,z)dz - \int \tilde{h}(z)\pi(z)dz\right|}_{\boxtimes}dy.
\end{aligned}$$

It holds

$$\begin{aligned}
\boxtimes &\leq \int |\tilde{h}(z)| \left| P_k^{k+s}(y,z) - \pi(z) \right| dz \\
&\leq \|\tilde{h}\|_{W^{1/2}} \int W^{1/2}(z) \left| P_k^{k+s}(y,z) - \pi(z) \right| dz \\
&\leq \sqrt{2}\|\tilde{h}\|_{W^{1/2}} \left(\int W(z) \left| P_k^{k+s}(y,z) - \pi(z) \right| dz \right)^{1/2} \\
&\leq \sqrt{2}\|\tilde{h}\|_{W^{1/2}} \left\| P_k^{k+s}(y,dz) - \pi(dz) \right\|_W^{1/2} \\
&\leq \sqrt{2}\varsigma^{1/2}\|\tilde{h}\|_{W^{1/2}} W(y)^{1/2} \rho^{s/2}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\diamond &\leq \int |\tilde{h}(y)| \pi(y) \sqrt{2}\varsigma^{1/2}\|\tilde{h}\|_{W^{1/2}} W(y)^{1/2} \rho^{s/2} dy \\
&\leq \sqrt{2}\varsigma^{1/2}\rho^{s/2}\|\tilde{h}\|_{W^{1/2}} \int |\tilde{h}(y)| \pi(y) W(y)^{1/2} dy \\
&\leq \sqrt{2}\varsigma^{1/2}\rho^{s/2}\|\tilde{h}\|_{W^{1/2}}^2 \int W(y)\pi(y) dy \\
&= \sqrt{2}\varsigma^{1/2}\rho^{s/2}\|\tilde{h}\|_{W^{1/2}}^2 \pi(W). \qquad (4.17)
\end{aligned}$$

From (4.16) and (4.17), we conclude that

$$\left| \mathbb{E}_x \left(\tilde{h}(Z_k)\tilde{h}(Z_{k+s}) \right) \right| \leq \sqrt{2}\varsigma^{3/2}\|\tilde{h}\|_{W^{1/2}}^2 W(x)\rho^{k+s/2} + \sqrt{2}\varsigma^{1/2}\rho^{s/2}\|\tilde{h}\|_{W^{1/2}}^2 \pi(W).$$

□

4.4 Applications to PDEs regularization in Deep learning

4.4.1 Regularization based on PDEs and a DC structure of the regularized problem

PDE-based regularization techniques have recently risen to prominence as a method for training deep neural networks with good generalization [21, 20]. Consider the deep learning optimization problem,

$$\min f(x),$$

where f is a large sum of compositions between a neural network and a loss function, and x represents the network's trainable parameters. Rather than immediately minimizing f, which can be quite rugged and only represents the training data, it is preferable to minimize its smoothed form $\tilde{f}$, which seeks to capture the primary trend of f and is expected to generalize well on unseen data [21, 20]. The following Hamilton-Jacobi equation, presented by [21], can be used to obtain such a smooth function

$$u_t + \frac{1}{2}|\nabla u|^2 = 0 \quad \text{in} \quad \mathbb{R}^n \times (0, +\infty),$$

and its "viscous" version (for $\epsilon > 0$),

$$u_t + \frac{1}{2}|\nabla u|^2 = \frac{\epsilon}{2}\Delta u \quad \text{in} \quad \mathbb{R}^n \times (0, +\infty),$$

where the initial condition of both PDEs is set as $u(x, 0) = f(x)$. The variables x and t represent space and time, respectively. The fundamental idea is to let the systems evolve up to time t, then minimize $u(x, t)$ instead of minimizing f, which is the initial condition (at time 0) of the above PDEs.

The former PDE gives rise to the viscous solution, which is the inf-convolution of f, bringing down local maxima while enlarging local minima, whereas the latter implies local entropy which favors local minima in wide valleys while suppressing those in sharp thin valleys. The local entropy, which is the viscous approximation of the inf-convolution, is the focus of this research. The local entropy plays a particularly important role in deep learning. In the context of training deep neural networks, Chaudhari et al. [20] argued that local minima lying in large flat region generalize better than those located in sharp valleys (although this argument is still controversial [34]). They also empirically verified that standard optimizers such as Adam or SGD applied to the original loss surface usually end up at solutions lying in flat regions (by checking the spectrum of the Hessian), explaining why these optimizers usually generalize well. Therefore, they suggested the construction of the local entropy that proactively modifies the original loss surface to favor flat-region local minima. The local entropy is a DC function that comes within the scope of this study. In fact, the DC structure can be found in a broader class of regularization structures based on a more general Hamilton-Jacobi equation

$$u_t + \frac{1}{2}\langle \nabla u, \mathcal{H}^{-1}\nabla u\rangle = 0 \quad \text{in } \mathbb{R}^n \times (0, +\infty),$$

with the initial condition $u(x, 0) = f(x)$, where $\mathcal{H}$ is a symmetric positive definite matrix. According to Hopf-Lax formula [43], the above PDE has the following viscous solution:

$$u(x, t) = \inf_{x'} \left\{ f(x') + \frac{1}{2t}\langle x' - x, \mathcal{H}(x' - x)\rangle \right\}. \tag{4.18}$$

We next present a fact that explains why the local entropy is a viscous approximation to the inf-convolution ($\epsilon \to 0^+$, the local entropy tends to the inf-convolution) and allows us to construct an approximation for the more general structure (4.18).

Fact [50]: if $\mathcal{F} \in C(\mathbb{R}^n)$ is superlinear, then $\lim_{\epsilon\to 0^+} \left(\int_{\mathbb{R}^n} e^{-\frac{\mathcal{F}(v)}{\epsilon}} dv\right)^{\epsilon} = e^{-\inf \mathcal{F}}$.

So, let $\mathcal{F}(x') = f(x') + \frac{1}{2t}\langle x' - x, \mathcal{H}(x' - x)\rangle$ which is superlinear if f is bounded below, the above fact implies $\tilde{u}(x, t) := -\epsilon \log \int_{\mathbb{R}^n} \exp\left(\frac{1}{\epsilon}\left[-f(x') - \frac{1}{2t}\langle x' - x, \mathcal{H}(x' - x)\rangle\right]\right) dx' \xrightarrow{\epsilon\to 0^+} u(x, t)$.

When we replace $\mathcal{H} = I$ for the above limit, we get the fact that the local entropy tends to the inf-convolution as $\epsilon \to 0^+$, which is usually referred to as *viscosity vanishing* in the PDE literature.

We point out a DC structure of $\tilde{u}(x, t)$ as follows,

$$\tilde{u}(x, t) = \underbrace{\frac{1}{2t}x^\top \mathcal{H}x}_{G(x)} - \underbrace{\epsilon \log \int_{\mathbb{R}^n} \exp\left(-\frac{1}{\epsilon}\left[f(x') + \frac{1}{2t}x'^\top \mathcal{H}x' - \frac{1}{t}x^\top \mathcal{H}x'\right]\right) dx'}_{H(x)}. \tag{4.19}$$

It is obvious that G is convex, while the convexity of H is verified as follows.

Verification of the convexity of H. Denote $v(x,x') = \exp\left(-\frac{1}{\epsilon}\left[f(x') + \frac{1}{2t}x'^{\top}\mathcal{H}x' - \frac{1}{t}x^{\top}\mathcal{H}x'\right]\right)$.

We compute the partial derivative of H,

$$\begin{aligned}\frac{\partial H}{\partial x_i} &= \epsilon\frac{\frac{\partial}{\partial x_i}\int v(x,x')dx'}{\int v(x,x')dx'} = \epsilon\frac{\int v(x,x')(\frac{1}{\epsilon t}[\mathcal{H}x']_i)dx'}{\int v(x,x')dx'}\\ &= \frac{1}{t}\frac{\int v(x,x')[\mathcal{H}x']_i dx'}{\int v(x,x')dx'} = \frac{1}{t}\mathbb{E}_{p(x'|x)}[\mathcal{H}x']_i,\end{aligned}$$

where $p(x'|x) \propto v(x,x')$. Therefore, $\nabla H(x) = \frac{1}{t}\mathbb{E}_{p(x'|x)}(\mathcal{H}x')$.

For $i,j \in \{1,2,\ldots,n\}$, the second order derivative is given by

$$\begin{aligned}&\frac{\partial^2 H}{\partial x_j \partial x_i} = \frac{1}{t}\frac{\frac{\partial}{\partial x_j}\int v(x,x')[\mathcal{H}x']_i dx' \int v(x,x')dx' - \int v(x,x')[\mathcal{H}x']_i dx' \frac{\partial}{\partial x_j}\int v(x,x')dx'}{\left(\int v(x,x')dx'\right)^2}\\ &= \frac{1}{t}\frac{\int v(x,x')\frac{1}{\epsilon t}[\mathcal{H}x']_j[\mathcal{H}x']_i dx' \int v(x,x')dx' - \int v(x,x')[\mathcal{H}x']_i dx' \int v(x,x')\frac{1}{\epsilon t}[\mathcal{H}x']_j dx'}{\left(\int v(x,x')dx'\right)^2}\\ &= \frac{1}{\epsilon t^2}\frac{\int v(x,x')[\mathcal{H}x']_j[\mathcal{H}x']_i dx' \int v(x,x')dx' - \int v(x,x')[\mathcal{H}x']_i dx' \int v(x,x')[\mathcal{H}x']_j dx'}{\left(\int v(x,x')dx'\right)^2}\\ &= \frac{1}{\epsilon t^2}\left(\mathbb{E}_{p(x'|x)}[\mathcal{H}x']_i[\mathcal{H}x']_j - \mathbb{E}_{p(x'|x)}[\mathcal{H}x']_i\mathbb{E}_{p(x'|x)}[\mathcal{H}x']_j\right).\end{aligned}$$

We prove the Hessian matrix $\mathbf{H} = \left(\frac{\partial^2 H}{\partial x_j \partial x_i}\right)_{i,j}$ is positive semidefinite. It amounts to showing $z^{\top}\mathbf{H}z \geq 0$, for all $z \in \mathbb{R}^n$, or equivalently,

$$\sum_{i,j} z_i z_j \mathbb{E}_{p(x'|x)}[\mathcal{H}x']_i[\mathcal{H}x']_j \geq \sum_{i,j} z_i z_j \mathbb{E}_{p(x'|x)}[\mathcal{H}x']_i\mathbb{E}_{p(x'|x)}[\mathcal{H}x']_j.$$

The above inequality is subsequently rewritten as follows

$$\mathbb{E}_{p(x'|x)}\left(\sum_{i,j} z_i z_j [\mathcal{H}x']_i[\mathcal{H}x']_j\right) \geq \left(\sum_{i=1}^{n} z_i \mathbb{E}_{p(x'|x)}[\mathcal{H}x']_i\right)^2,$$

and then

$$\mathbb{E}_{p(x'|x)}\left(\sum_{i=1}^{n} z_i[\mathcal{H}x']_i\right)^2 \geq \left[\mathbb{E}_{p(x'|x)}\left(\sum_{i=1}^{n} z_i[\mathcal{H}x']_i\right)\right]^2,$$

which is true thanks to Hölder's inequality.

□

4.4.2 Langevin dynamics

The gradient of the H is given by $\nabla H(x) = \frac{1}{t}\mathbb{E}_{p(x'|x)}(\mathcal{H}x')$, as shown previously. As a result, the MCSDCA for minimizing $\tilde{u}(x,t)$ requires - at each iteration - a simulation of a Markov chain with the

target distribution $p(x'|x) \propto \exp\left(-\frac{1}{\epsilon}\left[f(x') + \frac{1}{2t}x'^{\top}\mathcal{H}x' - \frac{1}{t}x^{\top}\mathcal{H}x'\right]\right)$. To this goal, we use Langevin dynamics, which was originally developed to model the dynamics of molecular systems. Let us consider the general target distribution $\pi(x) \propto \exp(-\beta U(x))$ where β is referred to as the inverse temperature.

The *overdamped Langevin diffusion* is of the form

$$dX_t = -\nabla U(X_t)dt + \sqrt{2\beta^{-1}}dB_t$$

where B_t is standard Brownian motion [92]. The overdamped Langevin diffusion, under suitable conditions, admits $\pi(x)$ as its unique invariant measure and the convergence to $\pi(x)$ happens at the geometric rate [111, 40]. To be implementable, this diffusion must be discretized. If the Euler-Maruyama discretization is employed, we obtain the following Markov chain named Unadjusted Langevin Algorithm

$$X_{k+1} = X_k - \eta\nabla U(X_k) + \sqrt{2\eta\beta^{-1}}W_k, \tag{4.20}$$

where W_k is the standard Gaussian noise. Some discretization error occurs but it is not quantified here. Furthermore, for computational feasibility, the full gradient of ∇U required in (4.20) is replaced by the mini-batch gradient, resulting in the stochastic gradient Langevin dynamics [130].

On the other hand, the *underdamped Langevin diffusion* takes the following form

$$\begin{aligned} dV_t &= -\mu V_t dt - \nabla U(X_t)dt + \sqrt{2\mu\beta^{-1}}dB_t \\ dX_t &= V_t dt \end{aligned} \tag{4.21}$$

where μ is the friction coefficient. Basically, this diffusion lifts the original space to a higher dimensional space (2x higher). Under mild conditions, the diffusion process (X_t, V_t) admits

$$\pi(x, v) \propto \exp\left(-\beta\left(\frac{1}{2}\|v\|^2 + U(x)\right)\right)$$

as the invariant measure and the convergence to $\pi(x, v)$ is exponentially fast [19]. The marginal distribution of the first component X is then $\pi(x) \propto \exp(-\beta U(x))$, which is exactly the target distribution we want to sample from. Furthermore, it is noteworthy that

$$\begin{aligned} \mathbb{E}_{\pi(x)}(F(x)) &= \int_x F(x)\pi(x)dx = \int_x F(x)\int_v \pi(x, v)dvdx \\ &= \int_x\int_v F(x)\pi(x, v)dvdx = \mathbb{E}_{\pi(x,v)}(F(x)). \end{aligned}$$

Therefore, instead of constructing a Markov chain with target distribution $\pi(x)$, one can consider a Markov chain with the target $\pi(x, v)$, namely, $(X^0, V^0) \to (X^1, V^1) \to \ldots \to (X^{k-1}, V^{k-1})$, then

$$\mathbb{E}_{\pi(x)}(F(x)) = \mathbb{E}_{\pi(x,v)}(F(x)) \approx \frac{1}{k}\sum_{i=0}^{k-1} F(X^i).$$

Now the question is what is a good discretization scheme for (4.21). Although the simple Euler-Maruyama discretization scheme can again be applied to this diffusion (resulting in the so-named inertial

Langevin dynamics), it also generates large error [56]. In this work, we make use of the state-of-the-art discretization scheme introduced in [25]. Basically, let's say with the specific friction $\mu = 2$, given the current state (X^i, V^i), the next state is sampled from a multivariate normal distribution with mean and covariance as follows (here the expectation is understood as the conditional expectation given (X^i, V^i))

$$\begin{aligned}
&\mathbb{E}(V^{i+1}) = V^i e^{-2\delta} - \frac{1}{2}(1 - e^{-2\delta})\nabla U(X^i),\\
&\mathbb{E}(X^{i+1}) = X^i + \frac{1}{2}(1 - e^{-2\delta})V^i - \frac{1}{2}\left(\delta - \frac{1}{2}(1 - e^{-2\delta})\right)\nabla U(X^i),\\
&\mathbb{E}\left[(X^{i+1} - \mathbb{E}(X^{i+1}))(X^{i+1} - \mathbb{E}(X^{i+1}))^\top\right] = \frac{1}{\beta}\left[\delta - \frac{1}{4}e^{-4\delta} - \frac{3}{4} + e^{-2\delta}\right]\cdot I_{d\times d},\\
&\mathbb{E}\left[(V^{i+1} - \mathbb{E}(V^{i+1}))(V^{i+1} - \mathbb{E}(V^{i+1}))^\top\right] = \frac{1}{\beta}(1 - e^{-4\delta})\cdot I_{d\times d},\\
&\mathbb{E}\left[(X^{i+1} - \mathbb{E}(X^{i+1}))(V^{i+1} - \mathbb{E}(V^{i+1}))^\top\right] = \frac{1}{2\beta}[1 + e^{-4\delta} - 2e^{-2\delta}]\cdot I_{d\times d},
\end{aligned} \tag{4.22}$$

where $\delta \in (0, 1)$ is the step size. Again, in the context of big data, the full gradient $\nabla U(X^i)$ is replaced by the stochastic gradient $\tilde{\nabla} U(X^i)$ computed over a mini-batch, resulting in a stochastic version of the above scheme (see [25][Subsection 2.2.1.]).

4.4.3 MCSDCA-odLD and MCSDCA-udLD

In this subsection, by putting necessary components together, we derive two specific MCSDCA schemes for training deep neural networks. In this practical setting, we simply choose the matrix $A = I$ in (4.4), and the matrix $\mathcal{H} = I$ in (4.18).

The first scheme, MCSDCA-odLD, is obtained straightforwardly by integrating the overdamped Langevin dynamics (4.20) into the inner loop of Algorithm 7. Note that with (we drop the outer iteration index for simplicity)

$$p(x'|x) \propto \exp\left(-\frac{1}{\epsilon}\left[f(x') + \frac{1}{2t}x'^\top x' - \frac{1}{t}x^\top x'\right]\right),$$

the stochastic version of the Langevin dynamics (4.20) reads

$$x'_{i+1} \sim x'_i - \eta\left(\tilde{\nabla} f(x'_i) + \frac{1}{t}(x'_i - x)\right) + \sqrt{2\eta\epsilon}\mathcal{N}(0, I),$$

where $\mathcal{N}(0, I)$ is the standard Gaussian distribution. Furthermore, the starting state of the Markov chain $\{x'_i\}$ is simply set to be x (the current point of the outer loop). The detailed MCSDCA-odLD is presented in Algorithm 8. It is worth noting that, with the mentioned specific settings, MCSDCA-odLD can be viewed as a sibling of the Entropy-SGD developed in [20, 21]. Only two distinctions exist: (1) the MCSDCA-odLD takes a traditional approach to the Markov chain burn-in period (discard early samples) while the Entropy-SGD employs exponential averaging, (2) the step size of the MCSDCA-odLD (in general, MCSDCA-odLD does not have a step size, but when applied to the local entropy with the given DC decomposition, it appears to have a step size) is specified by the DC decomposition (and partially by users via $\{\gamma_k\}$) and is always smaller than t, while the step size of the Entropy-SGD is

specified arbitrarily by users.

Algorithm 8 MCSDCA overdamped Langevin dynamics

Initialization. A starting point x^0, a sequence of positive numbers $\{\gamma_k\}$, a sequence of Markov chains' length $\{n_k\}$, the number of burn-in samples b, $\epsilon > 0$, $\eta > 0$, $t > 0$, set $k = 0$,

repeat

Set the starting state of a Markov chain $x_0^k := x^k$.

for $i = 0$ to $n_k - 2$ **do**

1. Receive a minibatch of data $D_{\text{minibatch}}$.
2. Compute a stochastic gradient $\tilde{\nabla} f(x_i^k)$ of f using $D_{\text{minibatch}}$.
3. Sample the next state of the Markov chain as

$$x_{i+1}^k \sim x_i^k - \eta \left(\tilde{\nabla} f(x_i^k) + \frac{1}{t}(x_i^k - x^k) \right) + \sqrt{2\eta\epsilon}\mathcal{N}(0, I)$$

where $\mathcal{N}(0, I)$ is the standard Gaussian distribution.

end for

Compute $y^k = \dfrac{1}{n_k - b} \sum_{i=b}^{n_k - 1} x_i^k$.

Solve the following convex problem,

$$x^{k+1} = \arg\min_x \left\{ \frac{1}{2t}\|x\|^2 - \frac{1}{t}\langle x, y^k \rangle + \frac{\gamma_k}{2}\|x - x^k\|^2 \right\}, \tag{4.23}$$

which has the closed-form solution

$$x^{k+1} = \frac{t\gamma_k}{1 + t\gamma_k} x^k + \frac{1}{1 + t\gamma_k} y^k.$$

Set $k = k + 1$.

until Stopping criterion

The second scheme, named MCSDCA-udLD, is obtained by using the underdamped Langevin dynamics (4.22) to approximately sample from $p(x'|x)$. It then reads,

$$\begin{aligned}
&\mathbb{E}(v'_{i+1}) = v'_i e^{-2\delta} - \frac{1}{2}(1 - e^{-2\delta}) \left(\nabla f(x'_i) + \frac{1}{t}(x'_i - x) \right) \\
&\mathbb{E}(x'_{i+1}) = x'_i + \frac{1}{2}(1 - e^{-2\delta}) v'_i - \frac{1}{2} \left(\delta - \frac{1}{2}(1 - e^{-2\delta}) \right) \left(\nabla f(x'_i) + \frac{1}{t}(x'_i - x) \right) \\
&\mathbb{E}\left[(x'_{i+1} - \mathbb{E}(x'_{i+1}))(x'_{i+1} - \mathbb{E}(x'_{i+1}))^\top \right] = \epsilon \left[\delta - \frac{1}{4}e^{-4\delta} - \frac{3}{4} + e^{-2\delta} \right] \cdot I_{d\times d} := c_1 \cdot I_{d\times d} \\
&\mathbb{E}\left[(v'_{i+1} - \mathbb{E}(v'_{i+1}))(v'_{i+1} - \mathbb{E}(v'_{i+1}))^\top \right] = \epsilon(1 - e^{-4\delta}) \cdot I_{d\times d} := c_2 \cdot I_{d\times d} \\
&\mathbb{E}\left[(x'_{i+1} - \mathbb{E}(x'_{i+1}))(v'_{i+1} - \mathbb{E}(v'_{i+1}))^\top \right] = \frac{\epsilon}{2}[1 + e^{-4\delta} - 2e^{-2\delta}] \cdot I_{d\times d} := c_3 \cdot I_{d\times d}.
\end{aligned}$$

The question is how to efficiently sample (x'_{i+1}, v'_{i+1}) with the above mean and covariance matrix. In the context of deep learning, x'_{i+1} represents all trainable parameters of a neural network, and so does v'_{i+1}. Therefore, the size of x'_{i+1} and v'_{i+1} is enormous. Furthermore, in deep learning frameworks, the structure of trainable parameters is not given as a vector. Rather, it is a list containing of many matrices and vectors, where each matrix (resp. vector) represents a weight (resp. bias) of a particular layer.

```
[Parameter containing:
tensor([[ 0.0478,  0.0519, -0.0146,  0.0574, -0.0137,  0.0126],
        [-0.0304,  0.0367,  0.0551, -0.0459,  0.0543,  0.0117],
        [ 0.0462,  0.0085,  0.0301, -0.0088,  0.0482,  0.0092],
        ...,
        [ 0.0476, -0.0113,  0.0411,  0.0144,  0.0376,  0.0609],
        [ 0.0142, -0.0232, -0.0062,  0.0176,  0.0105, -0.0171],
        [ 0.0490,  0.0376, -0.0388, -0.0051, -0.0060, -0.0392]],
       requires_grad=True), Parameter containing:
tensor([[ 0.0091, -0.0192, -0.0323,  ..., -0.0479, -0.0030, -0.0579],
        [-0.0568,  0.0552, -0.0048,  ...,  0.0015,  0.0114,  0.0589],
        [ 0.0290,  0.0134, -0.0126,  ...,  0.0008, -0.0281, -0.0494],
        ...,
        [-0.0101,  0.0558,  0.0292,  ...,  0.0475, -0.0080, -0.0537],
        [ 0.0581, -0.0445, -0.0067,  ...,  0.0494,  0.0268, -0.0079],
        [ 0.0289,  0.0508, -0.0365,  ...,  0.0474, -0.0219,  0.0177]],
       requires_grad=True), Parameter containing:
tensor([ 0.0175, -0.0124,  0.0202, -0.0041,  0.0160, -0.0234,  0.0466,  0.0074,
        -0.0541, -0.0249, -0.0204, -0.0466, -0.0550,  0.0311,  0.0062, -0.0455,
        -0.0331,  0.0579,  0.0466,  0.0091,  0.0603,  0.0582, -0.0053,  0.0415,
        ...,
         0.0595, -0.0397,  0.0166,  0.0358,  0.0620, -0.0002,  0.0142, -0.0082,
        -0.0569,  0.0318,  0.0033, -0.0513, -0.0536, -0.0465,  0.0362,  0.0408,
         0.0502, -0.0601, -0.0159,  0.0498, -0.0578,  0.0025, -0.0575, -0.0429],
       requires_grad=True), Parameter containing:
tensor([-0.0488,  0.0050, -0.0270,  0.0120, -0.0516,  0.0308,  0.0458,  0.0533,
        -0.0119, -0.0449,  0.0344,  0.0353, -0.0282,  0.0331,  0.0120, -0.0199,
         0.0373,  0.0312,  0.0162, -0.0360, -0.0088,  0.0543, -0.0002, -0.0606,
        ...,
        -0.0108,  0.0105, -0.0566, -0.0507,  0.0044, -0.0189,  0.0197,  0.0061,
         0.0348,  0.0564, -0.0327,  0.0009,  0.0319, -0.0591,  0.0191,  0.0286,
        -0.0011, -0.0432,  0.0521, -0.0139,  0.0492, -0.0347, -0.0303,  0.0451],
       requires_grad=True), Parameter containing:
tensor([[-2.3901e-02,  4.4087e-02,  1.9745e-02, -3.8896e-02,  5.6917e-02,
         -4.5016e-03,  3.0661e-03,  3.8288e-02,  5.2530e-02, -6.2331e-02,
         -1.4310e-03, -8.5702e-03, -3.4372e-02, -6.1862e-02, -2.0792e-03,
         ...,
         -4.7404e-02,  1.6868e-03,  7.5514e-03,  5.0420e-03, -1.6819e-02,
         -1.2975e-02,  4.6078e-02,  1.0787e-02,  9.3726e-03, -6.1640e-02,
          1.9391e-02]], requires_grad=True), Parameter containing:
tensor([-0.0212], requires_grad=True)]
```

Figure 4.1: Data structure of trainable parameters of a recurrent neural network

Figure 4.1 illustrates a typical data structure of trainable parameters of a recurrent neural network.

Clearly, it is not a wise choice to flatten these matrices and concatenate them together to form a super long vector, then construct a super large covariance matrix to do sampling. It is easier to sample layer by layer. We have several observations and remarks:

- The elements (the features) of x'_{i+1} are independent from each other, and so are the elements of v'_{i+1}. There is only correlation between each element of x'_{i+1} and v'_{i+1} at *the same location*, e.g., the element at row 2, column 3 of the weight matrix of the 4-th layer of x'_{i+1} is correlated with the element at row 2, column 3 of the weight matrix of the 4-th layer of v'_{i+1}.

- Conditional distribution of a multivariate normal distribution is again multivariate normal. To be specific, let (X, Y) be a pair of random vectors, whose joint distribution of multivariate normal with mean $\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$ and covariance matrix $\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$. Then, the conditional distribution of X given $Y = a$ is multivariate normal $\mathcal{N}(\bar{\mu}, \bar{\Sigma})$, where $\bar{\mu}$ and $\bar{\Sigma}$ are given as follows (see, e.g., [41, Section 3.4])

$$\begin{aligned} \bar{\mu} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(a - \mu_2), \\ \bar{\Sigma} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}. \end{aligned}$$

Therefore, instead of sampling (x'_{i+1}, v'_{i+1}) simultaneously, we will sample it sequentially and this can be done in a layer-by-layer manner. Firstly, we sample v'_{i+1} from the normal multivariate with mean

$\mathbb{E}(v'_{i+1}) = v'_i e^{-2\delta} - \frac{1}{2}(1-e^{-2\delta})\left(\nabla f(x'_i) + \frac{1}{t}(x'_i - x)\right)$ and covariance matrix $c_2 \cdot I$, or equivalently,

$$v'_{i+1} \sim v'_i e^{-2\delta} - \frac{1}{2}(1-e^{-2\delta})\left(\nabla f(x'_i) + \frac{1}{t}(x'_i - x)\right) + \sqrt{c_2}\mathcal{N}(0, I).$$

Next, we sample $x'_{i+1}|v'_{i+1}$. Thanks to the second remark above, $x'_{i+1}|v'_{i+1} \sim \mathcal{N}(\bar{\mu}, \bar{\sigma})$ with

$$\begin{aligned}\bar{\mu} &= \mathbb{E}(x'_{i+1}) + c_3 c_2^{-1}(v'_{i+1} - \mathbb{E}(v'_{i+1})),\\ \bar{\Sigma} &= (c_1 - c_3^2 c_2^{-1})I.\end{aligned}$$

Consequently, $x'_{i+1}|v'_{i+1} \sim \bar{\mu} + \sqrt{(c_1 - c_3^2 c_2^{-1})}\mathcal{N}(0, I)$. Finally, as discussed earlier, for computational feasibility, the gradient $\nabla f(x'_i)$ in the previous computations is replaced by the stochastic gradient $\tilde{\nabla} f(x'_i)$ computed over a mini-batch. The detailed MCSDCA-udLD is presented in Algorithm 9.

Algorithm 9 MCSDCA underdamped Langevin dynamics

Initialization. A starting point x^0, a sequence of positive numbers $\{\gamma_k\}$, a sequence of Markov chains' length $\{n_k\}$, the number of burn-in samples b, $\epsilon > 0$, $\delta > 0$, $t > 0$, set $k = 0$.
repeat
Set the starting state of the Markov chain $x_0^k := x^k$, and $v_0^k := 0$.
for $i = 0$ to $n_k - 2$ **do**
1. Receive a minibatch of data $D_{\text{minibatch}}$.
2. Compute a stochastic gradient $\tilde{\nabla} f(x_i^k)$ of f using $D_{\text{minibatch}}$.
3. Compute the conditional expectation of v_{i+1}^k and x_{i+1}^k given x_i^k and v_i^k as follows

$$\begin{aligned}\mathbb{E}(v_{i+1}^k) &= e^{-2\delta} v_i^k - \frac{1}{2}(1-e^{-2\delta})\left(\tilde{\nabla} f(x_i^k) + \frac{1}{t}(x_i^k - x^k)\right),\\ \mathbb{E}(x_{i+1}^k) &= x_i^k + \frac{1}{2}(1-e^{-2\delta})v_i^k - \frac{1}{2}\left(\delta - \frac{1}{2}(1-e^{-2\delta})\right)\left(\tilde{\nabla} f(x_i^k) + \frac{1}{t}(x_i^k - x^k)\right).\end{aligned}$$

4. Sample $v_{i+1}^k \sim \mathbb{E}(v_{i+1}^k) + \sqrt{c_2}\mathcal{N}(0, I)$.
5. Sample $x_{i+1}^k | v_{i+1}^k \sim \mathbb{E}(x_{i+1}^k) + c_3 c_2^{-1}(v_{i+1}^k - \mathbb{E}(v_{i+1}^k)) + \sqrt{(c_1 - c_3^2 c_2^{-1})}\mathcal{N}(0, I)$.
end for
Compute $y^k = \frac{1}{n_k - b}\sum_{i=b}^{n_k-1} x_i^k$.
Solve the following convex problem,

$$x^{k+1} = \arg\min_x \left\{\frac{1}{2t}\|x\|^2 - \frac{1}{t}\langle x, y^k\rangle + \frac{\gamma_k}{2}\|x - x^k\|^2\right\}, \tag{4.24}$$

which has the closed-form solution

$$x^{k+1} = \frac{t\gamma_k}{1 + t\gamma_k}x^k + \frac{1}{1 + t\gamma_k}y^k.$$

$k = k + 1$.
until Stopping criterion

Technical note. In implementation, the step of computing $y^k = \frac{1}{n_k - b}\sum_{i=b}^{n_k-1} x_i^k$ in Algorithms 8 and 9 is integrated into the inner loop, say, y^k is computed progressively.

4.5 Conclusion

In this chapter, we present a Markov chain stochastic DCA for solving (nonsmooth) endogenous stochastic DC programs using only Markovian data. There are strong guarantees in locating DC critical points of the proposed algorithm when both asymptotic and non-asymptotic convergence results are proved. As an important extension, we give additional analysis to time-inhomogeneous Markov chains, proving that the previously proved convergence results still hold in this scenario. This study should open the way for a more complete approach with firm convergence analysis to a larger class of nonconvex stochastic algorithms with diverse intrinsic noise properties. We next use PDEs regularization to apply the proposed technique to deep learning optimization problems, yielding two MCSDCA realizations, MCSDCA-olLD and MCSDCA-ulLD, which are based on the overdamped and underdamped Langevin dynamics, respectively. In the next chapter, these two MCSDCA realizations will be used to tackle time series forecasting problems utilizing some different deep neural network structures.

Chapter 5

Predictive Maintenance: a deep learning approach

Abstract. In this chapter, we study predictive maintenance - one of the key components in prognostics and health management. Predictive maintenance, with its ability to foresee operation failures and estimate assets' health, is able to reduce risks, maintenance costs and downtime, while increasing productivity. We investigate two major predictive maintenance problems: remaining useful life prediction and capacity/state-of-health estimation. We adopt the data-driven approach where deep neural networks are used to develop prediction models thanks to their strength in learning highly nonlinear relations while minimizing human effort in extracting features. Feed-forward, recurrent, and long short-term memory neural networks are the structures being used in our research. We then train these neural networks using the MCSDCA-odLD and MCSDCA-udLD presented in Chapter 4. Numerical experiments show the virtue of the two methods in comparison with other baseline optimizers in deep learning. The prediction results closely match the true remaining useful life and capacity values.

5.1 Introduction

Today, maintenance is at the heart of every industrial operation to ensure companies' stability, efficiency, and productivity. The right choice of maintenance policy can make a huge difference in the age of smart industry, enhancing significantly the competitiveness of companies. Going through a long history, maintenance has evolved from its original form of fixing assets once broken down to integrating "intelligence" to predict the time that failures occur, hence optimizing the maintenance schedule. Generally, there are three main themes of maintenance: reactive, preventive, and predictive [79]. Reactive maintenance, also known as run-to-failure maintenance, is the classical approach which intervenes only when a machine exhibits abnormal behavior. The primal advantage of such a maintenance strategy is the lower beginning expenditures and manpower requirements. For simple devices such as cell phones that do not serve a critical function, it is sufficient to employ reactive maintenance. However, when it comes to sophisticated machinery systems such as industrial plants, breakdowns mean major production challenges and

The numerical experiments of this chapter are partially extracted from the following work:
H. P. H. Luu, H. M. Le, & H. A. Le Thi. Markov Chain Stochastic DCA and Applications in Deep Learning with PDEs Regularization. **To be submitted**.

safety hazards, thereby diminishing quickly the advantages of reactive maintenance. The next stepping stone in the timeline of maintenance is preventive maintenance, where the main philosophy is to maintain machines before malfunctions occur. It follows a set schedule based on expected statistical information of equipment (provided by manufacturers) to carry out maintenance even if the equipment is operating normally. It helps us to gain proactivity in prioritizing the maintenance schedules. The last milestone is predictive maintenance, which refers to a class of techniques that actively predict the status of operating equipment based on actual (and time-based) data in order to determine when maintenance should be performed. This type of maintenance is probably the most advancing and sophisticated, which is backed by the cutting-edge technologies such as machine learning, deep learning, high-sensitivity sensor, and so on.

Predictive maintenance, once properly implemented, can have decisive advantages over reactive and preventive approaches. On the one hand, reactive maintenance, while simple to deploy, is cost-inefficient and unsuitable for industries where safety is of utmost priority such as transportation, health, and aerospace. Predictive maintenance addresses these limitations by actively maintaining equipment based on early alert detection, preventing catastrophic failures, reducing unplanned downtime, improving safety and liability, and prolonging equipment lifespan. On the other hand, while preventive maintenance is also capable of preventing disastrous crashes and enhancing safety, it is not cost-effective as equipment possibly still operates regularly for quite some time after the scheduled repair. Predictive maintenance, on the contrary, can potentially save a lot of unnecessary costs by making "just-in-time" decisions based on current equipment data, which is more reliable and informative than average population statistics used in preventive maintenance.

In predictive maintenance, two top-priority problems are remaining useful life (RUL) prediction and state-of-health (SoH) estimation. The RUL of a machine is the length of time in which the machine is expected to continue functioning as intended. The RUL, once accurately predicted, can aid engineers with crucial information to schedule maintenance. Meanwhile, the SoH indicates the current health condition of a battery (or a cell) in comparison with the initial health status. The SoH estimation provides important information in battery-powered systems, enabling smart replacement decisions possible.

Predictive maintenance methodologies, for RUL prediction and SoH estimation, can be categorized into model-based and data-driven methods. The model-based approach [126, 61, 60] requires extensive domain knowledge and the understanding of the underlying physical system to accurately model the degradation. For example, partial differential equations have been used to describe the electrochemical reactions that happen inside the lithium-ion batteries [140]. Once the system degradation is modeled precisely, this approach can be very accurate and reliable [106]. However, this modeling procedure is quite demanding, and in many circumstances, such knowledge is unavailable. Furthermore, because the model is not always linear, precise analysis of the solutions can be challenging at times [45]. Meanwhile, the data-driven approach [6, 36, 36, 132, 116, 138, 98] treats the underlying mechanism as a blackbox and learns to predict the relevant values directly from time-based data. In essence, with this approach, the two mentioned problems (RUL, SoH) are characterized as time series forecasting,

$$\boxed{\text{Time series data}} \rightarrow \boxed{\text{Machine learning/Deep learning model}} \rightarrow \boxed{\text{RUL/SoH}}$$

where the intermediate machine learning/deep learning model attempts to capture the relationship be-

tween up-to-date time series data and the corresponding RUL/SoH. This approach has lately acquired popularity due to its ease of implementation and the allowance of the lack of prior domain expertise. Because of the rapid advancement of machine learning and artificial intelligence, the data-driven approach is now prevalent in practically all domains of applied sciences, particularly predictive maintenance. A considerable amount of high-quality data is usually required for the training process in order to produce a successful model. This demand may be addressed thanks to high-precision sensors and cutting-edge internet-of-things technologies.

We further classify the data-driven approach into two categories: classical machine learning such as support vector machine [6], hidden Markov model [36], relevance vector machine [81, 142], or deep learning using deep neural networks as prediction functions [132, 116, 138, 98]. In recent years, the latter category, deep learning, is becoming increasingly popular, owing to the capacity of deep neural networks in learning highly nonlinear relationships without requiring much effort to extract information manually. Deep neural networks, in other words, can automatically extract features from raw data and use them to produce accurate predictions. On the other hand, some network topologies such as recurrent neural network (RNN) and long short-term memory (LSTM) are specialized for time series forecasting, making them much more prevalent in the context of predictive maintenance. Furthermore, one of the key strengths of deep neural networks is their flexibility: one can design different prediction models by simply modifying the model settings such as the number of hidden layers, the number of neurons in each layer, the activation functions, and so on. This increases the likelihood of obtaining a model that fits the data at hand.

Contribution We study two important problems in predictive maintenance, which are remaining useful life prediction and state-of-heath/capacity estimation. Two typical datasets used in our study are the turbofan engine degradation simulation dataset (CMAPSS) and the Li-ion battery aging dataset, both of which are provided by NASA (The National Aeronautics and Space Administration). In light of the preceding discussion, we are motivated to adopt the data-driven approach and rely on the well-established strength of deep neural networks. Three typical network structures are investigated in our study, namely, feed-forward neural networks, recurrent neural networks, and long short-term memory. It is worth noting that training a neural network is essentially about optimizing the loss function. We adopt the PDEs-based approach introduced in Chapter 4 to transform the original optimization problems into their smooth versions, in which some latent DC structures are observed. The neural networks are then trained using our proposed MCSDCA-odLD and MCSDCA-udLD algorithms. Numerical findings demonstrate the merits of our methods as they typically achieve lower generalization errors than several baseline optimizers in deep learning, including Adam, Adagrad, RMSProp, and SGD. Overall, the learned systems are capable of capturing the true RUL and the capacity of the turbofan engines and the lithium-ion batteries, respectively.

Our case studies are optimization-oriented, meaning that we focus on the optimization aspect of training deep neural networks to confirm the learning capability of the two proposed algorithms, MCSDCA-odLD and MCSDCA-udLD, over other state-of-the-art optimizers in deep learning.

5.2 Time series forecasting

A time series, denoted by $\{y_1, y_2, \ldots, y_k, \ldots\}$, is a sequence of data points that show up in time order. The data points are often temporally correlated to one another. Time series forecasting, in general, is the prediction of the time series values at some point in the future based on the data gathered up to the present time. There are several types of time series forecasting. To begin, it is preferable to utilize the time series' historical values to forecast its future value,

$$\hat{y}_{k+s} = f(y_k, y_{k-1}, \ldots, y_{k-r}),$$

where s is the look-ahead timestamp we want to predict at (known as the forecast horizon), r is the number of look-back timestamps we want to use to produce the prediction, and f is a function to be chosen in order to make accurate predictions.

Some research also considered making predictions using data observed from timestamps with unequal lengths [107, 1]

$$\hat{y}_{k+s} = f(y_{k-l_1}, y_{k-l_2}, \ldots, y_{k-l_r}), \tag{5.1}$$

where $l_1, l_2, \ldots, l_r$ indicate varying intervals of time between observations.

When several exogenous time series $\{u_i^1\}_{i\in\mathbb{N}}, \{u_i^2\}_{i\in\mathbb{N}}, \ldots, \{u_i^T\}_{i\in\mathbb{N}}$ are available to aid the prediction, one has the following formulation

$$\hat{y}_{k+s} = f(y_k, y_{k-1}, \ldots, y_{k-r}, u_k^1, u_{k-1}^1, \ldots, u_{k-r}^1, u_k^2, u_{k-1}^2, \ldots, u_{k-r}^2 \ldots u_k^T, u_{k-1}^T, \ldots, u_{k-r}^T). \tag{5.2}$$

In some cases, the true value of the time series $\{y_k\}_{k\in\mathbb{N}}$ is unknown at the prediction time and only exogenous time series $\{u_i^1\}_{i\in\mathbb{N}}, \{u_i^2\}_{i\in\mathbb{N}}, \ldots, \{u_i^T\}_{i\in\mathbb{N}}$ are at disposal, the formula becomes

$$\hat{y}_{k+s} = f(u_k^1, u_{k-1}^1, \ldots, u_{k-r}^1, u_k^2, u_{k-1}^2, \ldots, u_{k-r}^2 \ldots u_k^T, u_{k-1}^T, \ldots, u_{k-r}^T). \tag{5.3}$$

In this latter case, s can be 0: one uses exogenous time series to forecast the unknown value of the time series of interest at the current moment. Usually, the function f is chosen in a certain family of functions parameterized by w, $\{f_w\}_{w\in\mathcal{W}}$. The problem then becomes choosing the best w that fits the above regression as well as possible.

In the context of predictive maintenance, $\{u_i^1\}_{i\in\mathbb{N}}, \{u_i^2\}_{i\in\mathbb{N}}, \ldots, \{u_i^T\}_{i\in\mathbb{N}}$ are often the time series collected by sensors, and the target time series $\{y_k\}_{k\in\mathbb{N}}$ is the variable of interest such as the remaining useful life or the state-of-health/capacity.

Methods for time series forecasting can be classified into three categories: naïve forecasting methods, statistical forecasting methods, and machine learning forecasting methods [87].

Naïve forecasting methods As the name suggests, this approach is the simplest family of methods for forecasting. Regarding the averaging method, the forecasts of all future values are equal to the average historical data (see, e.g., [57]),

$$\hat{y}_{k+s} = \frac{y_k + y_{k-1} + \ldots + y_1}{k}.$$

Hence, observations are considered to be of equal relevance for the prediction. Another naïve method is to forecast all future values to be the last observed value,

$$\hat{y}_{k+s} = y_k.$$

Between these two extremes, one can also assign higher weights to the latter observations than the former ones,

$$\hat{y}_{k+s} = \alpha y_k + \alpha(1-\alpha)y_{k-1} + \alpha(1-\alpha)^2 y_{k-2} + \ldots + (1-\alpha)^{k-1} y_1.$$

In order to capture the trend of the time series, Holt's linear trend method [55] takes into account, beside the forecast equation, two auxiliary smoothing equations

$$\begin{aligned} \hat{y}_{t+s} &= l_t + sb_t \\ l_t &= \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \end{aligned}$$

where the latter two equations are to capture the level and the trend, respectively, and $\alpha, \beta \in [0,1]$ are smoothing parameters. To capture the seasonality, the above Holt's method was extended to the Holt-Winters' seasonal method [131].

The application of this approach is limited, for example, it is inapplicable when the history of the time series is missing, as in formula (5.3).

Statistical forecasting methods This class of methods, which has a strong mathematical foundation, mainly uses linear regression to forecast. Some prominent methods include: AutoRegression (AR), Moving Average (MA), and AutoRegressive Integrated Moving Average (ARIMA) (see, e.g., [57]).

Autoregression uses a linear combination of past values to forecast. An autoregression model of order p can be written as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t,$$

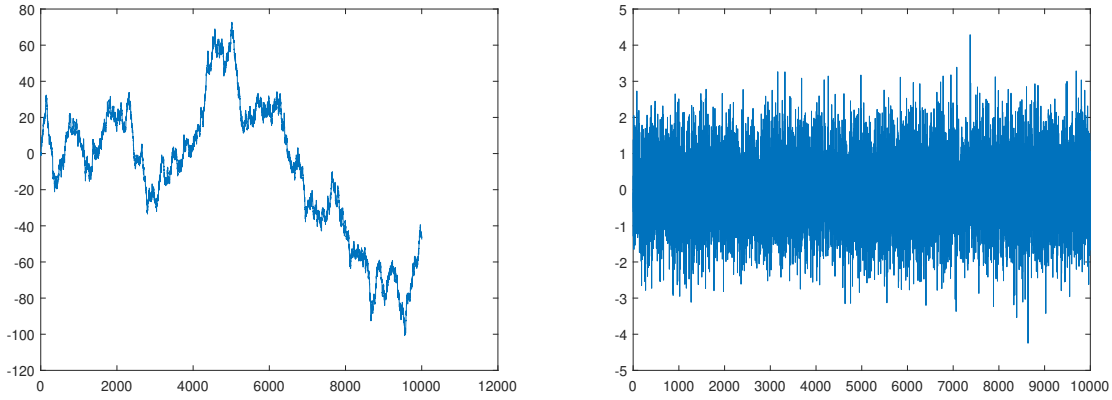
where ϵ_t is Gaussian noise and $\phi_1, \phi_2, \ldots, \phi_p$ are regression parameters.

Moving average model, on the other hand, uses a linear combination of past forecast errors

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q},$$

where $\epsilon_t, \epsilon_{t-1}, \ldots, \epsilon_{t-q}$ are Gaussian noise and $\theta_1, \theta_2, \ldots, \theta_q$ are regression parameters.

Autoregression and Moving average are closely related. Indeed, when we enroll the autoregression formula, it can be written in form of a linear combination of errors. For example, a simple autoregression



(a) original series

(b) differenced series

Figure 5.1: Differencing

model $y_t = \alpha y_{t-1} + \epsilon_t$ can be written as

$$y_t = \alpha^{t-1} y_1 + \alpha^{t-2} \epsilon_2 + \alpha^{t-3} \epsilon_3 + \ldots + \epsilon_t.$$

Another method is AutoRegressive integrated moving average (ARIMA), which combines the AR, MA and *differencing*. In time series analysis, differencing is a technique to make a time series stationary. Given a time series $\{y_t\}_t$, the differenced time series $\{y'_t\}_t$ is defined by

$$y'_t = y_t - y_{t-1}.$$

Differencing can remove trends, as illustrated in Figure 5.1.

Furthermore, one can apply differencing again to $\{y'_t\}_t$ to obtain a twice-differenced series

$$y''_t = y'_t - y'_{t-1},$$

and so on to obtain $\{y_t^{(3)}\}_t$, $\{y_t^{(4)}\}_t$, etc.

An ARIMA model takes the following form

$$y_t^{(d)} = c + \phi_1 y_{t-1}^{(d)} + \ldots + \phi_p y_{t-p}^{(d)} + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q} + \epsilon_t,$$

where p, q are the orders of the autoregressive part and the moving average part, respectively, d is the degree of differencing used.

The mentioned methods mainly perform forecasting based on historical data of the same time series. In case one has to predict based on multiple exogenous time series, the following linear regression model can be used,

$$y_t = \beta_0 + \beta_1 u_t^1 + \beta_2 u_t^2 + \ldots + \beta_T u_t^T + \epsilon_t,$$

where $\beta_0, \beta_1, \ldots, \beta_T$ are the model parameters, and ϵ_t is Gaussian noise.

Machine leaning approach Machine learning techniques have less constraints in terms of linearity and stationarity than statistical forecasting techniques [24]. The machine learning regression approach for time series prediction includes support vector regression, decision tree, random forest, and deep neural networks such as the vanilla feed-forward neural network, convolutional neural network, recurrent neural network, and long short-term memory. The ability to learn highly nonlinear regression is the approach's key strength, allowing it to cope with difficult time series prediction applications. With this approach, the prediction function f can be a decision tree, a random forest, a neural network, and so on. The next section introduces three common neural networks that will be employed in this study.

5.3 Deep neural networks and optimization problems

In this section, we give a brief introduction to three types of neural networks used in our research: feed-forward neural network, recurrent neural network, and long short-term memory. For each neural network, we clearly state the associated optimization problem which we aim to solve.

5.3.1 Deep feed-forward neural networks

Deep feed-forward neural networks, or multilayer perceptrons, are the classic deep learning models that are formed by stacking layers together. The phrase "feed-forward" alludes to the fact that information goes through the function from input to hidden layers to output, and there is no feedback connection where the output of a model is fed back to itself [52]. The term deep learning comes from the fact that there are many hidden layers (refers to as the depth of a network) between the input and the output that can extract meaningful features from raw data.

Figure 5.2 presents a typical vanilla feed-forward neural network [6], knowing that there exist other advanced structures, e.g., convolutional neural networks using special layers such as convolution and pooling. Since each neuron is connected to all neurons in the previous layer, the feed-forward neural network in 5.2 is also called fully-connected (FC) neural network.

The connection between two consecutive layers is in fact an affine transformation followed by a non-linear transformation (each neuron is activated by a function σ), resulting in the following mathematical expression,

$$\begin{aligned} \text{input}_{\text{layer}_k} &= W_{k-1}\text{output}_{\text{layer}_{k-1}} + b_{k-1}, \\ \text{output}_{\text{layer}_k} &= \sigma(\text{input}_{\text{layer}_k}). \end{aligned}$$

Hence, a feed-forward neural network is a function of the following composition form

$$\hat{y}(x) = W_l\sigma\left(W_{l-1}\sigma\left(W_{l-2}\sigma\left(\cdots(W_0x + b_0)\cdots\right) + b_{l-2}\right) + b_{l-1}\right) + b_l.$$

Given a training dataset $\{(x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N)\}$, one tries to adjust the internal parameters of the network to match the network output $\hat{y}$ to the true corresponding label, yielding the following optimization problem

[6] This figure is produced by using `https://alexlenail.me/NN-SVG/`.

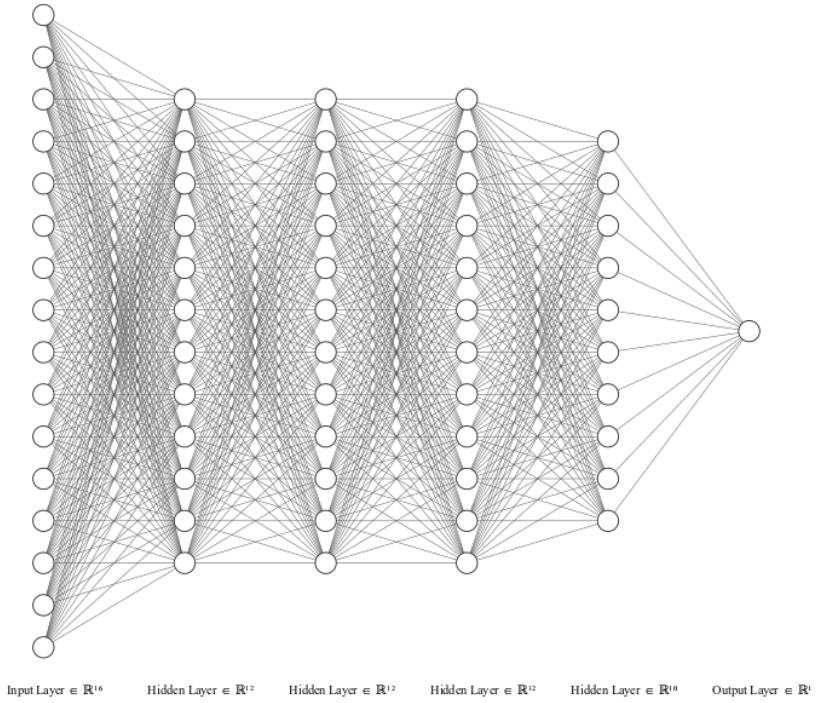


Figure 5.2: Feed-forward neural network

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(W_l \sigma\left(W_{l-1} \sigma\left(W_{l-2} \sigma\left(\cdots\left(W_0 x^i + b_0\right)\cdots\right) + b_{l-2}\right) + b_{l-1}\right) + b_l, y^i\right),$$

where $\Theta = (W_0, b_0, W_1, b_1, \ldots, W_l, b_l)$ is the optimization variable.

5.3.2 Recurrent neural networks

The "feed-forward" feature of the classic neural networks appears to be a limitation in problems where the sequential correlation of data is of major importance. In such a case, it is desirable to have a feedback connection. To this end, RNN maintains a hidden variable that is transferred across time steps. Hence, now the input to be fed into an RNN cell at time t is not just the current data x_t, but also the previous hidden state h_{t-1}, leading to the following update of the hidden state

$$h_t = \sigma(W_{xh} x_t + W_{hh} h_{t-1} + b_h)$$

and the output

$$o_t = W_{hq} h_t + b_q,$$

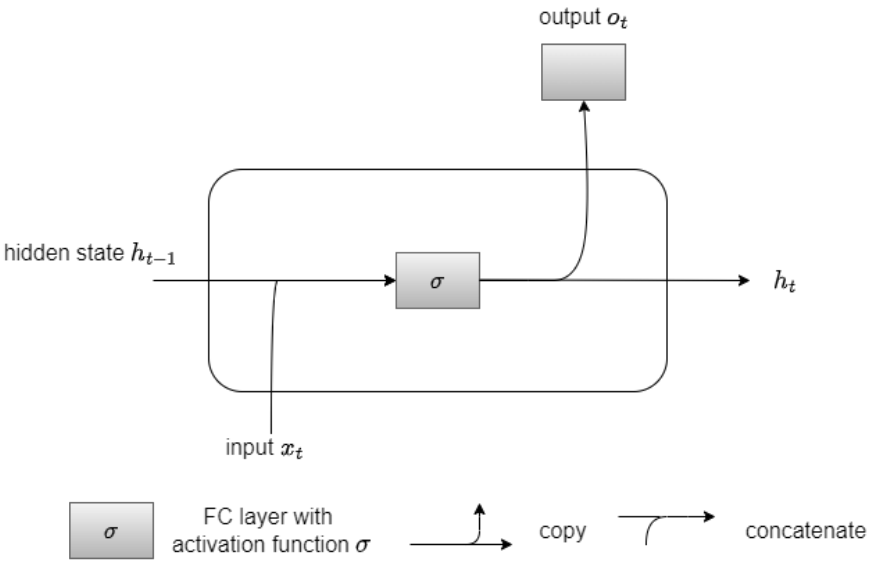


Figure 5.3: RNN cell

where $W_{xh}, W_{hh}, b_h, W_{hq}, b_q$ are trainable parameters. Figure 5.3 depicts a typical RNN cell [7], where FC stands for fully-connected.

Given a sequence of training samples $\{x_1, x_2, \ldots, x_K\}$ and a corresponding groundtruth label y, we want to build a prediction for y based on the last output of the RNN. To this end, we stack on top of o_t a fully-connected layer with an output size compatible with y,

$$\hat{y} = W_{fc} o_t + b_{fc},$$

resulting the loss of $\mathcal{L}(y, \hat{y})$. We can unroll the above recurrent structure to give rise to the analytical form of $\hat{y}$,

$$\hat{y} = W_{fc}\left(W_{hq}\sigma\left(W_{xh}x_K + W_{hh}\sigma\left(\cdots\sigma(W_{xh}x_1 + W_{hh}h_0 + b_h)\cdots\right) + b_h\right) + b_q\right) + b_{fc},$$

where h_0 is some starting hidden state, which is usually chosen to be a zero vector.

Given a training dataset consisting of a number of training sequences with their associated labels,

$$\mathcal{S} = \{(\{x_1^1, x_2^1, \ldots, x_K^1\}, y^1), (\{x_1^2, x_2^2, \ldots, x_K^2\}, y^2), \ldots, (\{x_1^N, x_2^N, \ldots, x_K^N\}, y^N)\},$$

the optimization problem reads

$$\min_{\Theta} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(W_{fc}\left(W_{hq}\sigma\left(W_{xh}x_K^i + W_{hh}\sigma\left(\cdots\sigma(W_{xh}x_1^i + W_{hh}h_0 + b_h)\cdots\right) + b_h\right) + b_q\right) + b_{fc}, y^i)$$

where $\Theta = (W_{xh}, W_{hh}, b_h, W_{hq}, b_q, W_{fc}, b_{fc})$ is the optimization variable.

[7] This figure is produced by using `https://app.diagrams.net/`.

5.3.3 Long short-term memory

An LSTM cell [54, 47] is made up of three gates: forget, input, and output, and it maintains a hidden state that is recurrently updated. At each time step, the LSTM gates are fed with the current input data x_t and the previous hidden state h_{t-1} via three fully-connected layers activated by σ,

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \end{aligned}$$

where $W_{xi}, W_{xf}, W_{xo}, W_{hi}, W_{hf}, W_{ho}, b_i, b_f, b_o$ are parameters to be learned. Similarly, the candidate memory cell, $\tilde{c}_t$, is computed by passing h_{t-1} and x_t through another fully-connected layer with $\tanh$ activation, which ends up being scaled to $(-1, 1)$,

$$\tilde{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c),$$

where W_{xc}, W_{hc}, b_c are again parameters to be learned. Then, the LSTM calculates the actual memory c_t (that will be transmitted to the next time step) based on the previous memory c_{t-1} forgotten some amount of old information addressed by the forget gate f_t, and the candidate memory $\tilde{c}_t$ governed by the input gate i_t to control how much new information is taken into consideration,

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t.$$

The new memory c_t going through the $\tanh$ activation together with the output gate o_t are used to update the hidden state

$$h_t = o_t \odot \tanh(c_t).$$

Finally, c_t and h_t are broadcast to the next time step to continue the recurrent update. Figure 5.4 depicts an LSTM cell with all of the components listed [8].

In summary, the complete equations for an LSTM cell are described as follows

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \\ \tilde{c}_t &= \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \\ h_t &= o_t \odot \tanh(c_t). \end{aligned}$$

Given a sequence of training sample $\{x_1, x_2, \ldots, x_K\}$ and the label y, similar to the RNN, we stack

[8] This figure is produced by using `https://app.diagrams.net/`.

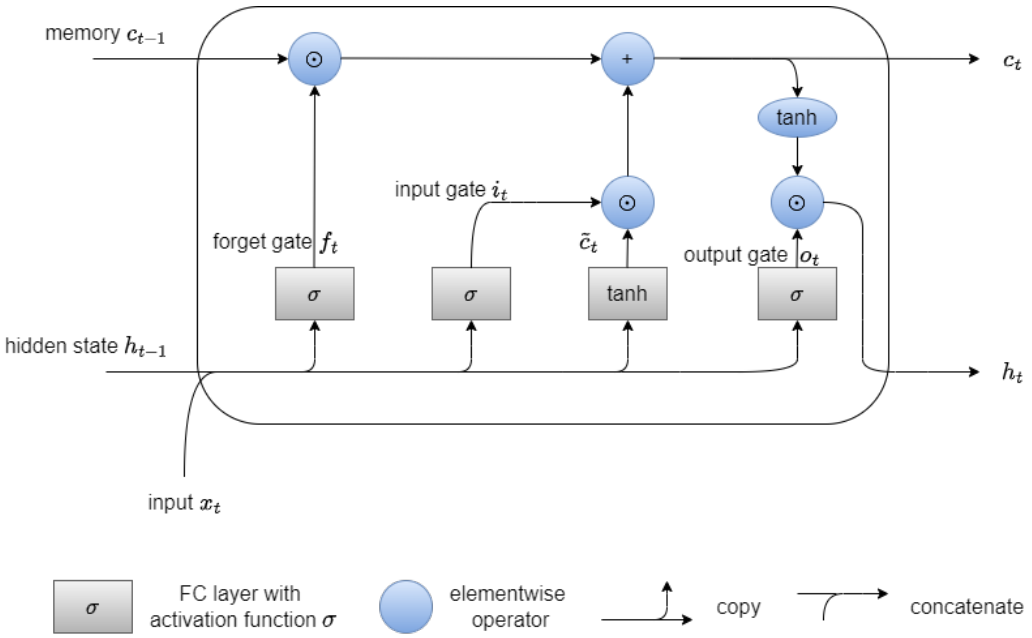


Figure 5.4: LSTM cell

on top of o_t a fully-connected layer with output size compatible with y,

$$\hat{y} = W_{fc} o_t + b_{fc},$$

resulting in the loss of $\mathcal{L}(y, \hat{y})$. Again, to have a glance at the optimization being studied, we unroll back to two time steps (it is very complicated to trace back further) the above recurrent relations to giving rise to the following complicated analytical form of $\hat{y}$,

$$\begin{aligned}\hat{y} =& b_{fc} + W_{fc}\,[\sigma(W_{xo}x_K + W_{ho}\sigma(W_{xo}x_{K-1} + W_{ho}o_{K-2} \odot \tanh(c_{K-2}) + b_o) \\ & \odot \tanh(\sigma(W_{xf}x_{K-1} + W_{hf}h_{K-2} + b_f) \odot c_{K-2} \\ & + \sigma(W_{xi}x_{K-1} + W_{hi}h_{K-2} + b_i) \odot \tanh(W_{xc}x_{K-1} + W_{hc}h_{K-2} + b_c)) + b_o)]\end{aligned}$$

knowing that $h_{K-2}, c_{K-2}, o_{K-2}$ will be traced back all the way down to h_0, c_0 (note that o_1 is defined via h_0). The starting initial hidden state h_0 and the initial memory c_0 are usually set to be zero vectors.

Given a training dataset consisting of a number of training sequences with their associated labels,

$$\mathcal{S} = \{(\{x_1^1, x_2^1, \ldots, x_K^1\}, y^1), (\{x_1^2, x_2^2, \ldots, x_K^2\}, y^2), \ldots, (\{x_1^N, x_2^N, \ldots, x_K^N\}, y^N)\},$$

the optimization problem reads

$$\begin{aligned}\min_{\Theta} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(b_{fc} + W_{fc}\,\big[\sigma(W_{xo}x_K^i + W_{ho}\sigma(W_{xo}x_{K-1}^i + W_{ho}o_{K-2} \odot \tanh(c_{K-2}) + b_o) \\ \odot \tanh(\sigma(W_{xf}x_{K-1}^i + W_{hf}h_{K-2} + b_f) \odot c_{K-2} \\ + \sigma(W_{xi}x_{K-1}^i + W_{hi}h_{K-2} + b_i) \odot \tanh(W_{xc}x_{K-1}^i + W_{hc}h_{K-2} + b_c)) + b_o)\big]\,, y^i),\end{aligned}$$

here $\Theta = (W_{xi}, W_{hi}, b_i, W_{xf}, W_{hf}, b_f, W_{xo}, W_{ho}, b_o, W_{xc}, W_{hc}, b_c, W_{fc}, b_{fc})$ is the optimization variable to be learned.

5.4 Remaining useful life prediction

5.4.1 Data exploration and preprocessing

One of the utmost important tasks in predictive maintenance is to predict the remaining useful life (RUL) as the RUL provides critical information so that one can arrange an appropriate maintenance schedule. We study the turbofan engine degradation simulation dataset (CMAPSS) [117], where the engine model is of the 90000 lb thrust class [118]. A turbofan engine is an engine that generates thrust and thereby propels an airplane through the air. The CMAPSS dataset contains four subdatasets (FD001, FD002, FD003, FD004), and each - in turn - contains multiple sensor data of hundreds of engines. The information of the CMAPSS dataset is presented in Table 5.1, where an operating condition is a combination of three settings (flight altitude, Mach number, and throttling parser angle), and the considered fault modes include high-pressure compressor (HPC) degradation and fan degradation.

Dataset	Operating conditions	Fault modes	Number of training engines	Number of test engines
FD001	1	1	100	100
FD002	6	1	260	259
FD003	1	2	100	100
FD004	6	2	248	249

Table 5.1: CMAPSS dataset information

Sensorial data of each engine is presented in form of 21 time series, where the actual physical meaning of each time series is unknown. In the beginning, each engine operates normally, and then gradually develops fault over time, where there is one fault mode for the datasets FD001 and FD002, and two fault modes for FD003 and FD004. In the training sets, the engines are let to run to failure, i.e., the moment when the engines are considered as being unhealthy. On the other hand, the sensors recorded in the test sets terminated sometime before the failure, and the goal is to predict the RUL of each engine of the test set. Among 21 sensors, $s_1, s_5, s_6, s_{10}, s_{16}, s_{18}, s_{19}$ contain no useful trend for predicting the RUL. Hence, only 14 time series are used for the prediction. Figures 5.5 and 5.6 depict the sensor 7 and sensor 4 of all engines of dataset FD001. The trends of these sensors are clearly observed: as the engines progressively operate towards the end of their useful life, the values of sensor 7 tend to decrease, while the values of sensor 4 exhibit an increasing trend.

Exponential smoothing Exponential moving average (EMA) is a well-known method to smooth time series. Given a time series $\{x_n\}$, EMA constructs a smoothed (filtered) time series $\{s_n\}$ as follows

$$\begin{aligned} s_0 &= x_0 \\ s_n &= \alpha x_n + (1-\alpha) s_{n-1}, \end{aligned}$$

where $\alpha \in (0, 1)$ is a parameter controlling the smoothness. If α is close to 0, the filtered time series is nearly constant and the smoothing effect is strong. On the other hand, if α is close to 1, $\{s_n\}$ is similar

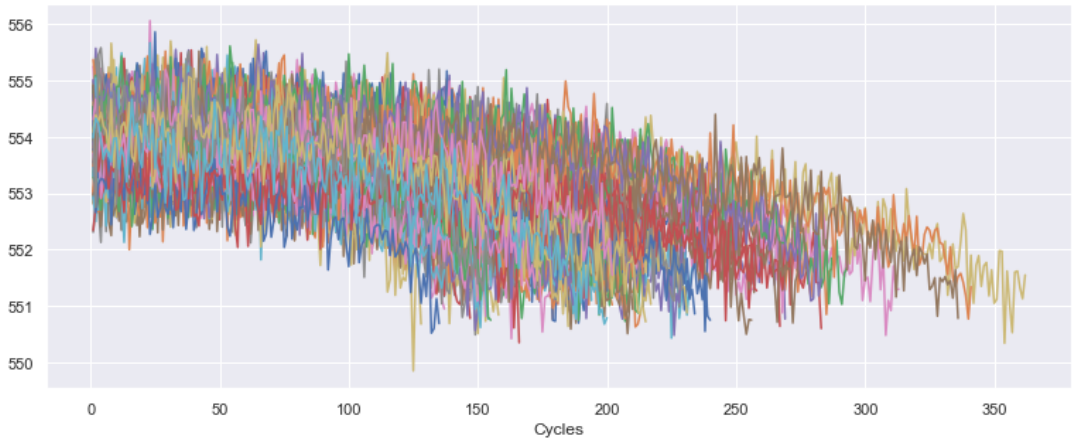
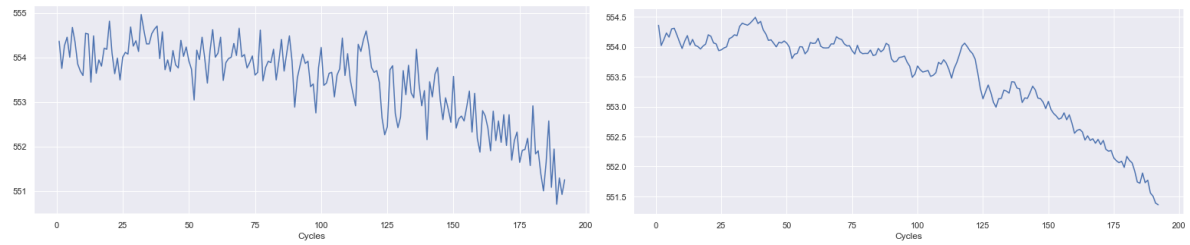


Figure 5.5: Sensor 7, dataset FD001

Figure 5.6: Sensor 4, dataset FD001



(a) original time series (b) smoothed time series

Figure 5.7: Smoothing effect ($\alpha = 0.2$) on sensor 7 of engine number 1

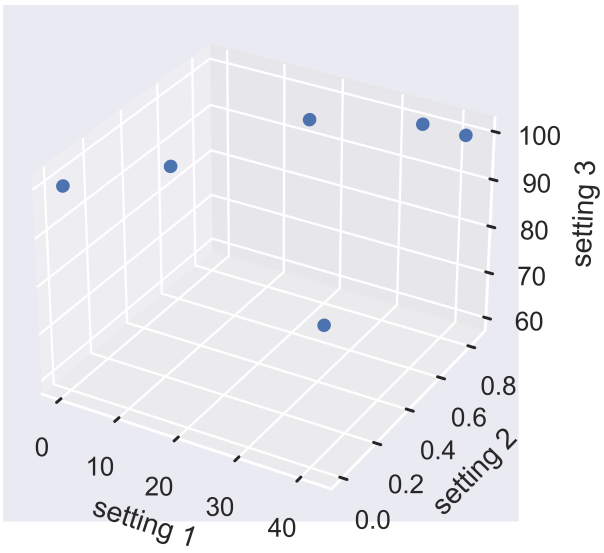


Figure 5.8: Six operating conditions of dataset FD004

to the original time series $\{x_n\}$. Figure 5.7 illustrates the smooth effect when using EMA with $\alpha = 0.2$.

Operating conditions While aero-engines in datasets FD001 and FD003 operate under one unique condition, aero-engines in datasets FD002 and FD004 operate under six conditions (up to a certain noise), which is much more challenging. Figure 5.8 depicts six operating conditions showing up as data points of three settings are clustered into six different groups when we use 3D scatter plot. These six operating conditions change alternatively from cycle to cycle of each engine. It is known that the variance introduced by the operating conditions overwhelms the one caused by the progressing degradation trends [100]. One of the methods to resolve this problem is to use standard scaler within each condition (see, e.g., [100]).

Truncated RUL As the initial wear and tear of each engine are unknown, instead of using the normal linear RUL, we use the truncated RUL for the training data (see, e.g., [116]). The truncated RUL is defined as truncated_RUL $= \min(\text{RUL}, T)$, where T is the truncation level.

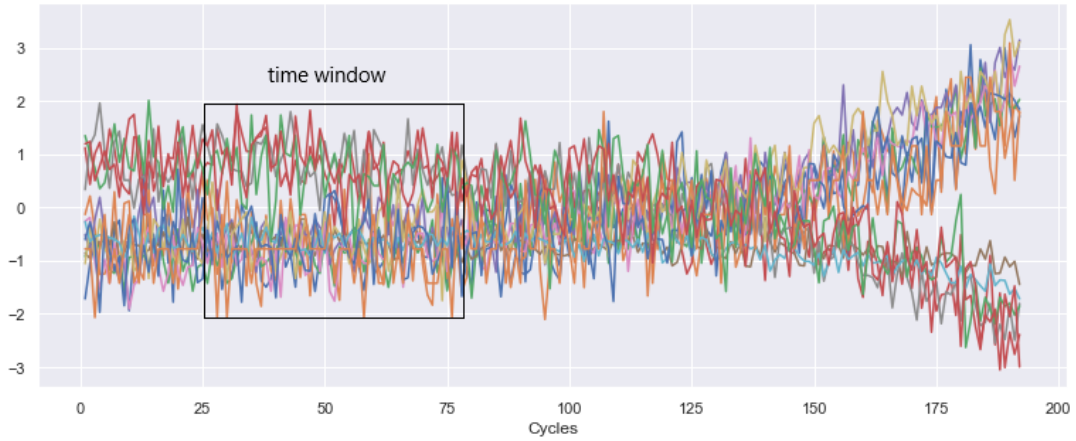


Figure 5.9: Time window for engine 1, dataset FD001

Time window for the time series prediction problem As discussed early in Section 5.2, for a time series prediction problem, one usually uses a time window of collected data from the current time back to a fixed interval of time in the past, altogether to predict the target value. Hence, the data extracted from multiple time series within a time window serves as the input of a neural network, while the RUL at the corresponding timestamp serves as the target value. Figure 5.9 illustrates one typical time window of data extracted from engine 1.

5.4.2 Experimental setups

In this research, we focus on the optimization aspect of training deep neural networks. In other words, we want to study the learning capacity of the two proposed algorithms, MCSDCA-odLD and MCSDCA-udLD, in comparison with other benchmark optimizers in deep learning. When comparing optimizers, it's common to employ pre-existing network topologies that are known to have good overall (train, test) performance on the considered datasets. Then, with the same random initialization weight, all optimizers are utilized to train these specified networks. We follow the same spirit here, but instead of relying on pre-existing structures for the CMAPSS dataset, we first find satisfactory model configurations (number of neurons per layer, dropout probability [122]). To that purpose, we use a random search to find the best standard fully-connected neural network and LSTM for each dataset, as well as certain other hyperparameters such as the RUL truncation level and the window length (of time series). On the acquired models, we then conduct a comparison study between MCSDCA, SGD, Adam, Adagrad, and RMSprop, with the training process stated below. Furthermore, in this experimental study, we found that MCSDCA-odLD is a good proxy for the MCSDCA as it performs very well. As the MCSDCA-odLD is already efficient, the MCSDCA-udLD, which is expected to accelerate the MCSDCA-odLD, does not show its advantage in this case. However, in the next experiment on battery capacity estimation when the MCSDCA-odLD is slow to converge, the MCSDCA-udLD significantly accelerates the learning speed. Therefore, we preserve the MCSDCA-udLD for the next experiment, while the MCSDCA-odLD will be used as a good proxy for the MCSDCA in this experiment.

Model configurations We use the following fully-connected and LSTM neural networks topologies:

FD001:

$$\begin{aligned}\text{Fully-connected NN: } &\text{input}_{378} \to \text{fc}_{64}(\text{sigmoid}) \to \text{drop}_{0.3} \to \text{fc}_{128}(\text{sigmoid}) \to \text{drop}_{0.3} \\ &\to \text{fc}_{256}(\text{sigmoid}) \to \text{drop}_{0.3} \to \text{fc}_1 \\ \text{LSTM: } &\text{input}_{24\times 14} \to \text{lstm}_{64} \to \text{drop}_{0.2} \to \text{fc}_1\end{aligned}$$

FD002:

$$\begin{aligned}\text{Fully-connected NN: } &\text{input}_{252} \to \text{fc}_{256}(\text{sigmoid}) \to \text{drop}_{0.5} \\ &\to \text{fc}_{512}(\text{sigmoid}) \to \text{drop}_{0.5} \to \text{fc}_{1024}(\text{sigmoid}) \to \text{drop}_{0.5} \to \text{fc}_1 \\ \text{LSTM: } &\text{input}_{18\times 14} \to \text{lstm}_{128} \to \text{drop}_{0.5} \to \text{fc}_1\end{aligned}$$

FD003:

$$\begin{aligned}&\text{Fully-connected NN: } \text{input}_{210} \to \text{fc}_{256}(\text{tanh}) \to \text{fc}_{512}(\text{tanh}) \to \text{fc}_{1024}(\text{tanh}) \to \text{fc}_1 \\ &\text{LSTM: } \text{input}_{20\times 14} \to \text{lstm}_{128} \to \text{drop}_{0.3} \to \text{fc}_1\end{aligned}$$

FD004:

$$\begin{aligned}\text{Fully-connected NN: } &\text{input}_{252} \to \text{fc}_{64}(\text{sigmoid}) \to \text{drop}_{0.3} \to \text{fc}_{128}(\text{sigmoid}) \to \text{drop}_{0.3} \\ &\to \text{fc}_{256}(\text{sigmoid}) \to \text{drop}_{0.3} \to \text{fc}_1 \\ \text{LSTM: } &\text{input}_{18\times 14} \to \text{lstm}_{256} \to \text{fc}_1\end{aligned}$$

Comparative algorithms We use Adam, Adagrad, RMSprop and SGD as baseline algorithms.

Training procedure For the baseline algorithms (Adam, Adagrad, RMSprop, SGD), we design a staircase schedule for the learning rate: the initial learning rate is set as Adam(0.001), Adagrad(0.01), RMSprop(0.01), SGD(0.01) and the learning rate is dropped by a factor of 2 after each 10 epochs. For the MCSDCA-odLD, motivated from [20], we set time $1/t = 10^{-4} \cdot 1.001$ and $\epsilon = 10^{-8}$. One can also slowly decrease time as $1/t = 10^{-4} \cdot 1.001^k$ w.r.t. iteration k (known as "scoping" [20]). However, in this work, we just set $k = 1$. Moreover, based on the non-asymptotic convergence analysis, the hyperparameters are chosen as follows. The number of Langevin steps is $n_k = n_{\text{init}} + \lfloor k^\lambda \rfloor$ with $n_{\text{init}} = 20, \lambda = 0.1$. We discard 10 samples from each Markov chain. Because $1/t$ shows up in both G and ∇H, it is conventional to set $\gamma_k = \frac{1}{t}\tilde{\gamma}_k$. We choose sequence $\{\tilde{\gamma}_k\}$ as $\tilde{\gamma}_k = 10^{-5}(k+1)^{0.1}$. The step size of the Langevin dynamics (4.20) is set at 0.001. For all algorithms, we set the maximum number of epochs to be 40, resulting in the backprop call budget of $40N$, where N is number of training samples. If an algorithm exhausts its computing budget, it will be terminated automatically. At each run, we randomly partition the training dataset into training (90%) and validation (10%), where the validation set is utilized for early stopping: if the validation loss does not improve after 4 consecutive epochs, the algorithm is stopped.

5.4.3 Experimental results

Tables 5.2, 5.3, 5.4, and 5.5 report the average results on the datasets FD001, FD002, FD003, and FD004, respectively. The MCSDCA-odLD outperforms other baselines on the fully-connected NN in three criteria for the dataset FD001, while RMSprop outperforms other baselines on the LSTM model in terms of generalization errors. On the LSTM models for the datasets FD002 and FD003, MCSDCA-odLD achieves the best generalization performance. Meanwhile, Adagrad wins on the dataset FD003 (albeit the test error margin between MCSDCA-odLD and Adagrad is narrow: 14.95 and 14.94), MCSDCA-odLD wins on FD002, and Adagrad wins on FD003. MCSDCA-odLD produces the best train, validation, and test results on the fully-connected NN for the dataset FD004, which is the most complex (6 operating conditions and 2 fault modes) dataset among the four datasets. Meanwhile, Adam comes in second place, catching up to MCSDCA-odLD's performance on the validation and test sets. MCSDCA-odLD is the sole winner on the LSTM.

The training and validation curves (the original objective, plotted against the number of backprop calls - agnostic to less relevant factors, and the training curves are produced by taking the average of all mini-batch training losses thus far) displayed in Figures 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16 and 5.17 show that MCSDCA-odLD exhibits a strong decrease like RMSprop and SGD at the very beginning of the training process, but is much better than SGD afterwards, and is less prone to underfitting and overfitting than RMSprop. On both types of networks, the MCSDCA-odLD prediction results are quite close to the true RUL on the test sets (Figures 5.18, 5.19, 5.20, and 5.21). Overall, MCSDCA-odLD obtains the lowest test errors in 5 of 8 case studies.

Algorithm	Fully-connected NN			LSTM		
	Train	Validation	Test	Train	Validation	Test
MCSDCA-odLD	**14.61**	**11.54**	**12.75**	17.35	15.14	14.76
Adam	21.24	12.11	13.10	29.71	15.03	14.45
Adagrad	20.60	12.81	13.60	33.92	23.98	18.92
RMSprop	16.71	13.19	15.66	16.17	**14.67**	**14.42**
SGD	25.82	19.07	21.73	**15.95**	15.59	15.40

Table 5.2: Results on dataset FD001

Algorithm	Fully-connected NN			LSTM		
	Train	Validation	Test	Train	Validation	Test
MCSDCA-odLD	**20.22**	**19.32**	21.88	21.18	**19.37**	**21.47**
Adam	21.13	19.61	21.87	25.22	20.67	23.35
Adagrad	21.18	20.63	**21.02**	26.78	21.76	23.21
RMSprop	35.65	27.16	29.04	**19.89**	20.05	22.41
SGD	49.81	49.44	54.18	29.99	27.06	30.27

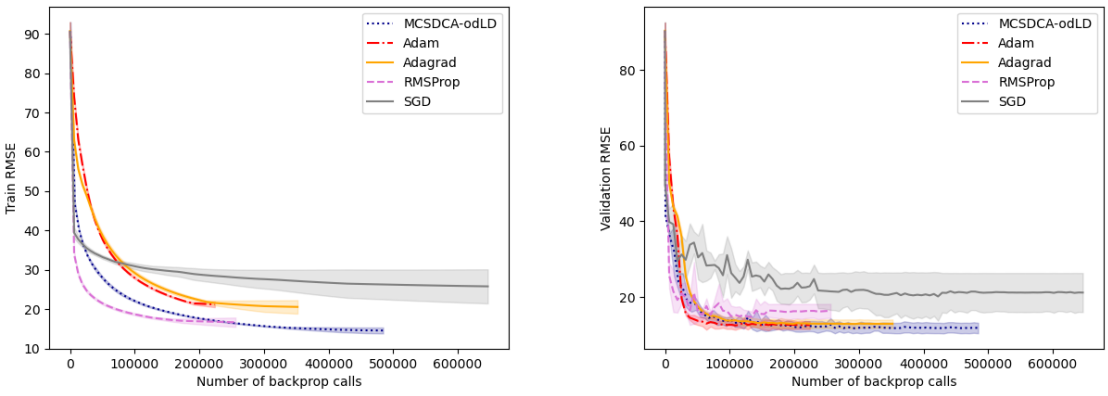
Table 5.3: Results on dataset FD002

Algorithm	Fully-connected NN			LSTM		
	Train	Validation	Test	Train	Validation	Test
MCSDCA-odLD	14.74	13.25	14.95	14.92	**12.34**	**15.38**
Adam	15.58	13.43	15.13	23.68	13.18	16.83
Adagrad	**14.56**	**13.08**	**14.94**	23.46	14.56	18.55
RMSprop	20.14	15.60	18.03	**14.37**	13.69	16.51
SGD	NaN	NaN	NaN	18.10	16.49	20.77

Table 5.4: Results on dataset FD003

Algorithm	Fully-connected NN			LSTM		
	Train	Validation	Test	Train	Validation	Test
MCSDCA-odLD	**19.24**	**18.91**	**22.74**	**21.82**	**18.10**	**22.23**
Adam	22.08	**18.91**	**22.74**	25.84	22.03	25.28
Adagrad	23.53	20.76	25.14	26.41	22.59	24.63
RMSprop	19.58	20.38	24.34	22.84	20.24	24.45
SGD	21.60	20.78	25.15	36.09	33.02	39.04

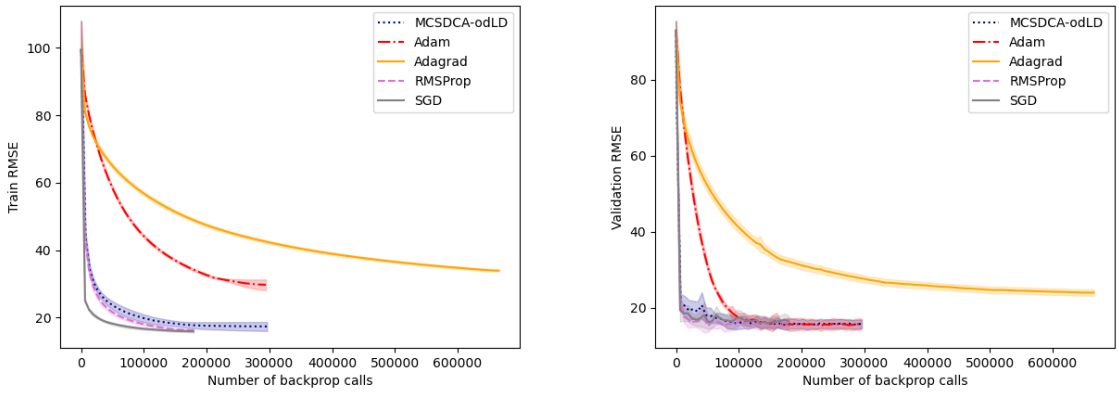
Table 5.5: Results on dataset FD004



(a) Train

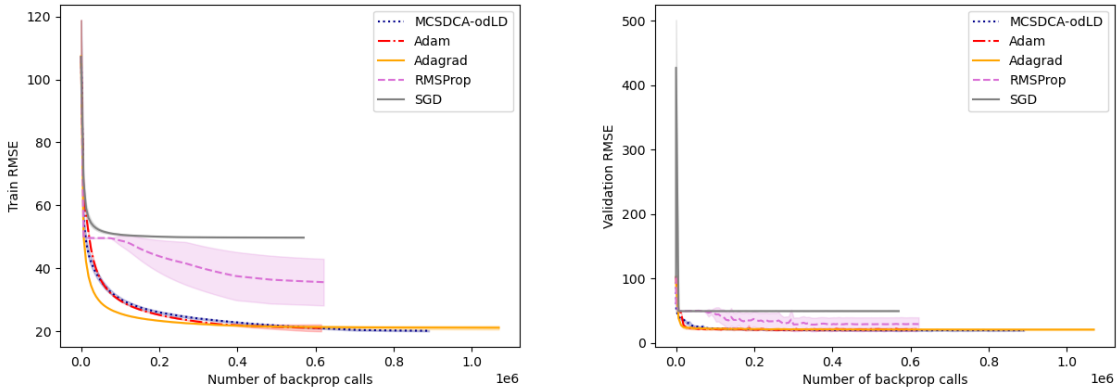
(b) Validation

Figure 5.10: Training and validation curves on the fully-connected NN, dataset FD001



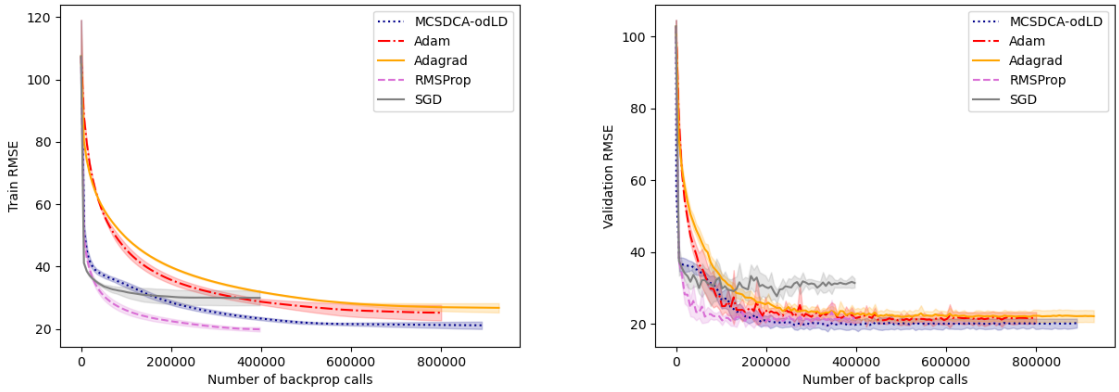
(a) Train (b) Validation

Figure 5.11: Training and validation curves on the LSTM, dataset FD001



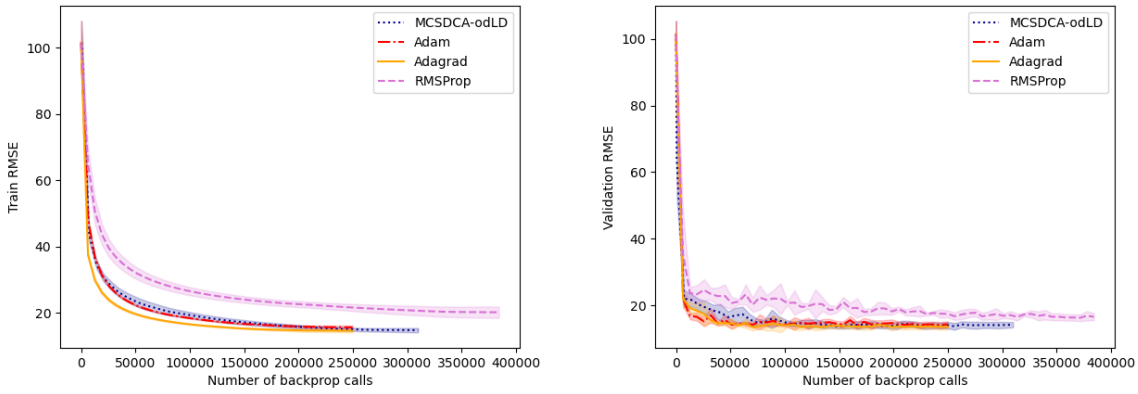
(a) Train (b) Validation

Figure 5.12: Training and validation curves on the fully-connected NN, dataset FD002



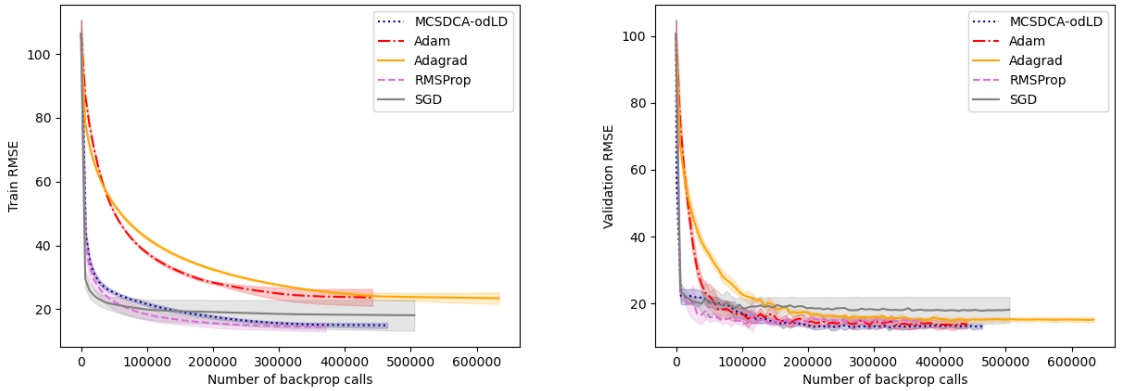
(a) Train (b) Validation

Figure 5.13: Training and validation curves on the LSTM, dataset FD002



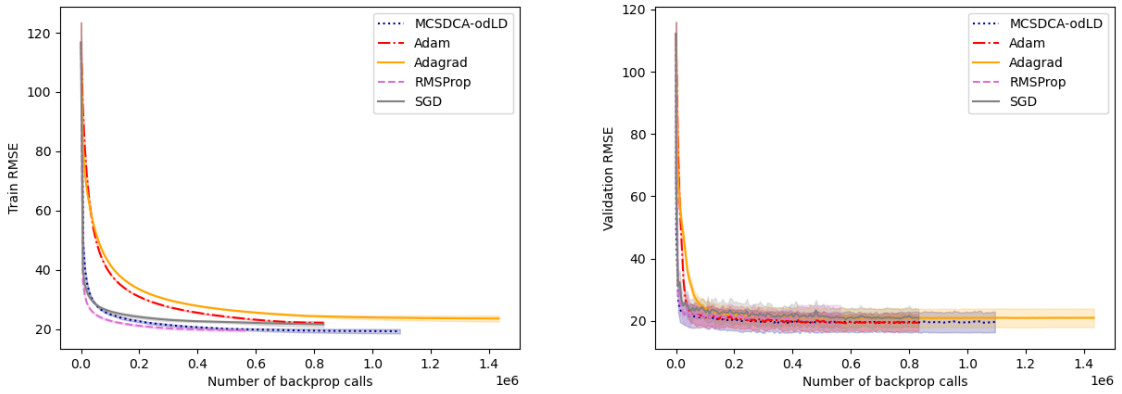
(a) Train (b) Validation

Figure 5.14: Training and validation curves on the fully-connected NN, dataset FD003



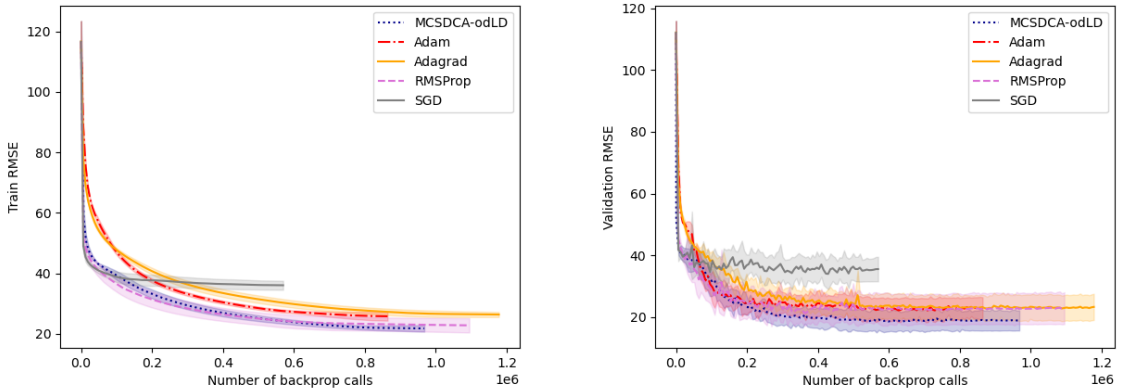
(a) Train (b) Validation

Figure 5.15: Training and validation curves on the LSTM, dataset FD003



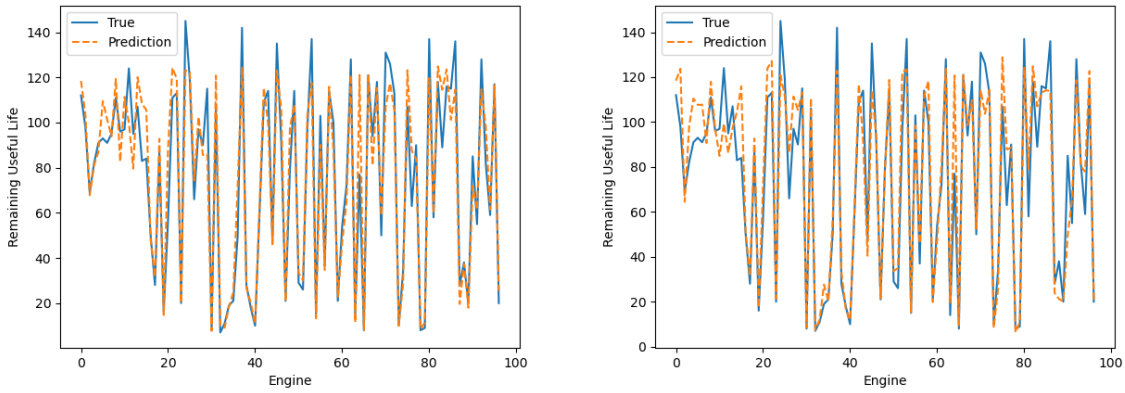
(a) Train (b) Validation

Figure 5.16: Training and validation curves on the fully-connected NN, dataset FD004



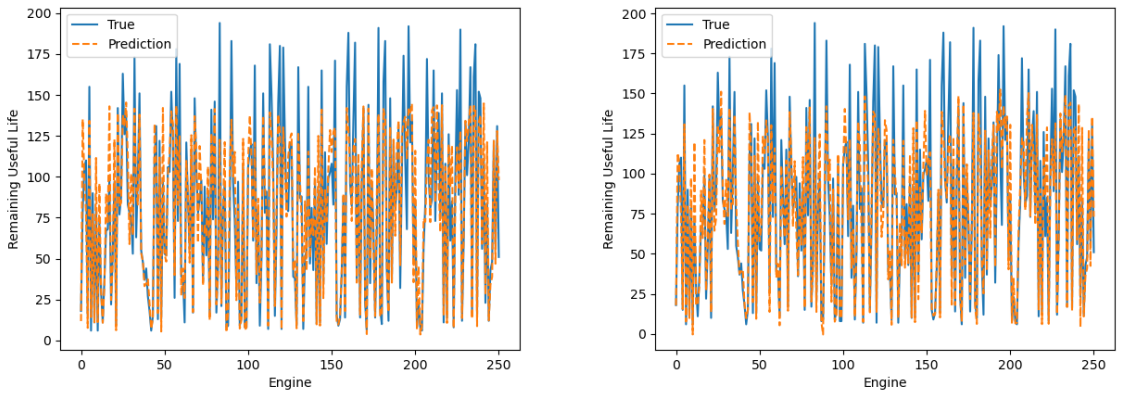
(a) Train (b) Validation

Figure 5.17: Training and validation curves on the LSTM, dataset FD004



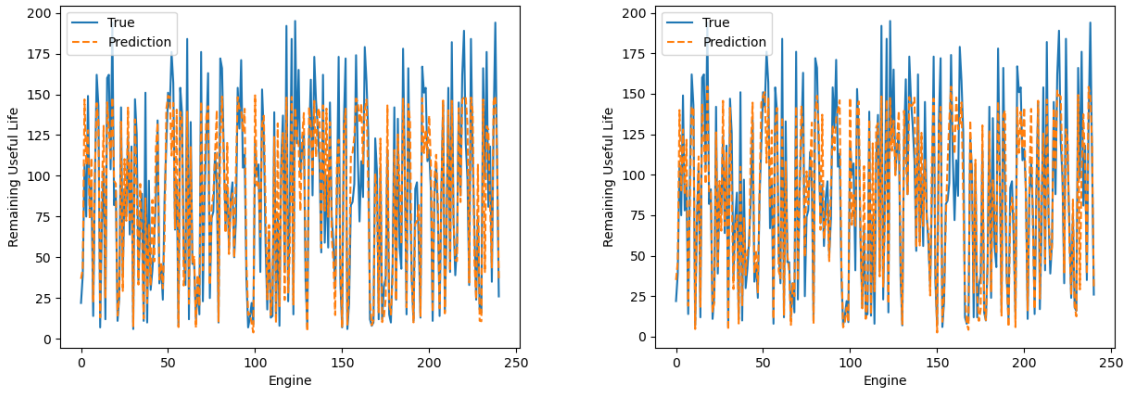
(a) Fully-connected NN (b) LSTM

Figure 5.18: Prediction results by MCSDCA-odLD on the test set of FD001



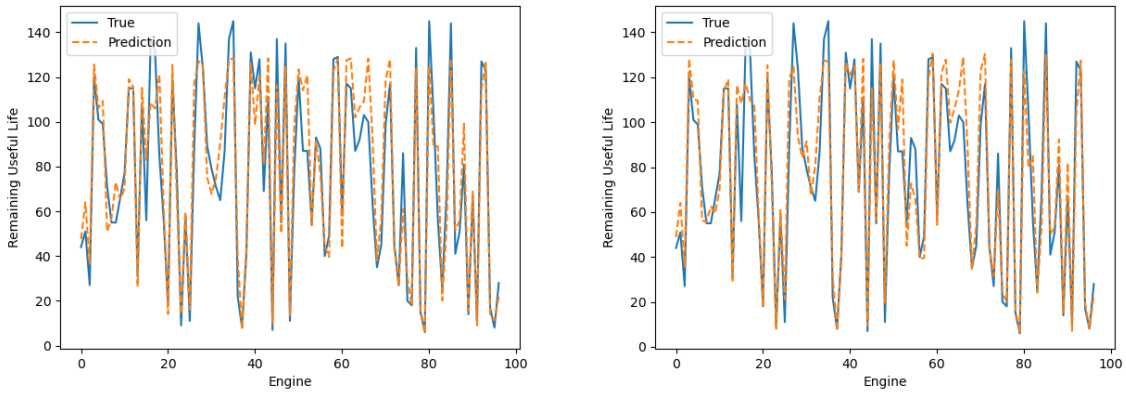
(a) Fully-connected NN (b) LSTM

Figure 5.19: Prediction results by MCSDCA-odLD on the test set of FD002



(a) Fully-connected NN (b) LSTM

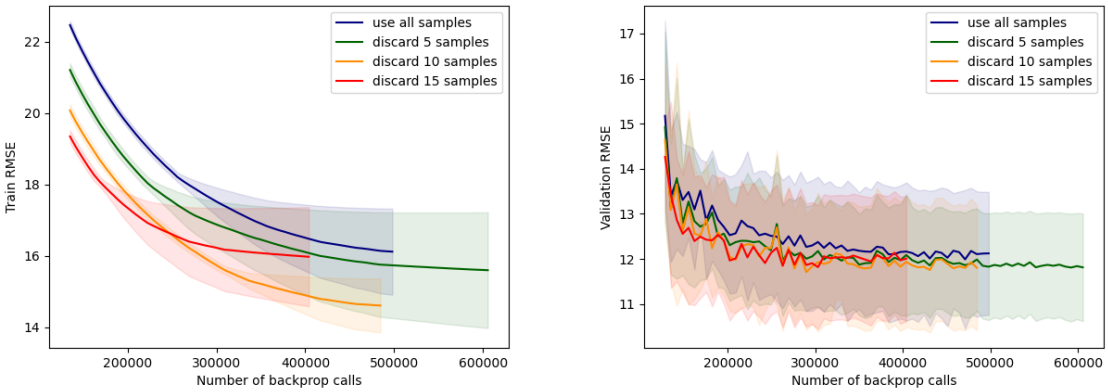
Figure 5.21: Prediction results by MCSDCA-odLD on the test set of FD004



(a) Fully-connected NN (b) LSTM

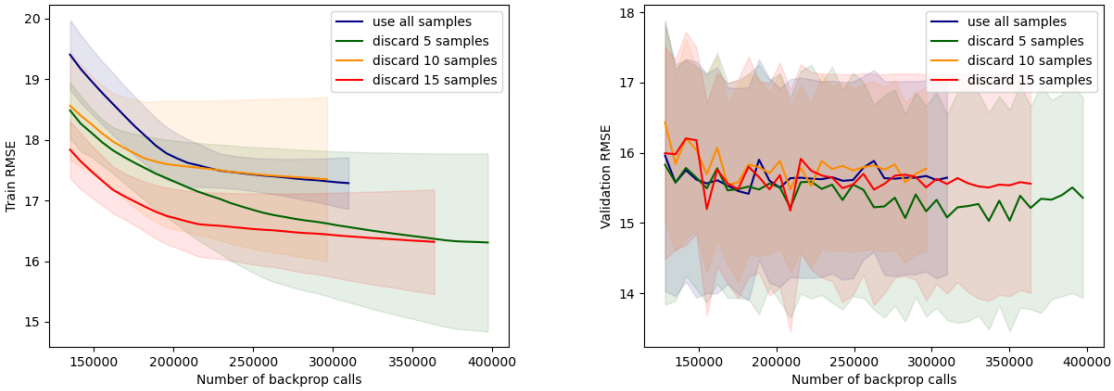
Figure 5.20: Prediction results by MCSDCA-odLD on the test set of FD003

Effects of the burn-in period In this experiment, we study how the amount of omitted samples affects the MCSDCA's overall performance. To accomplish this, we either maintain all samples or discard 5, 10, 15 samples from each Markov chain. Figures 5.22, 5.23, 5.24, 5.25, 5.26, 5.27, 5.28, 5.29 show that discarding more samples seems to improve training convergence but also increases the risk of overfitting.



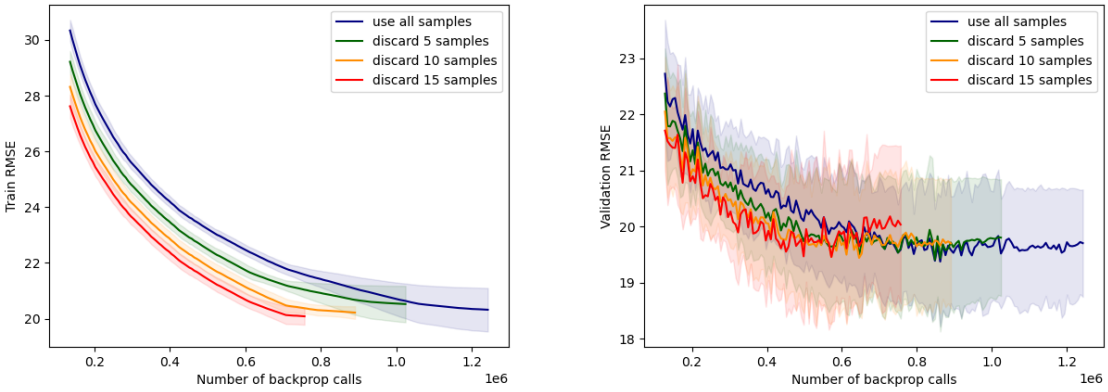
(a) Train (b) Validation

Figure 5.22: Burn-in effect on the fully-connected NN, dataset FD001



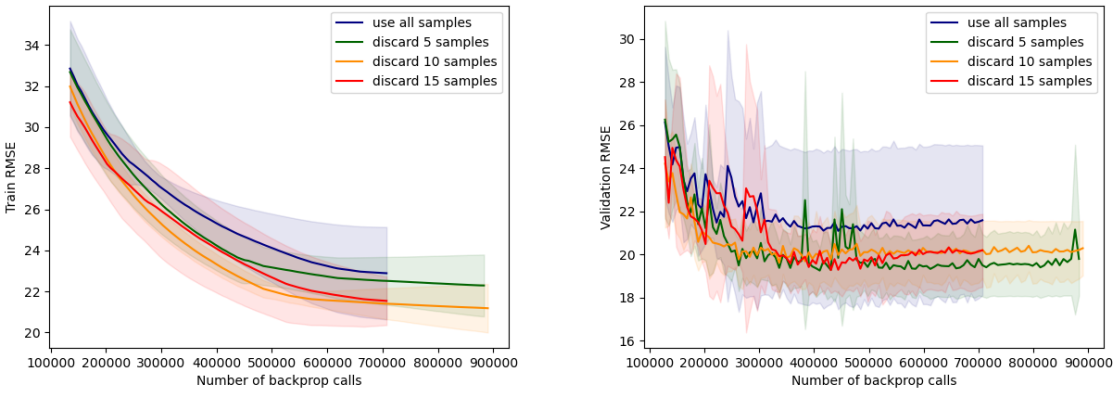
(a) Train (b) Validation

Figure 5.23: Burn-in effect on the LSTM, dataset FD001



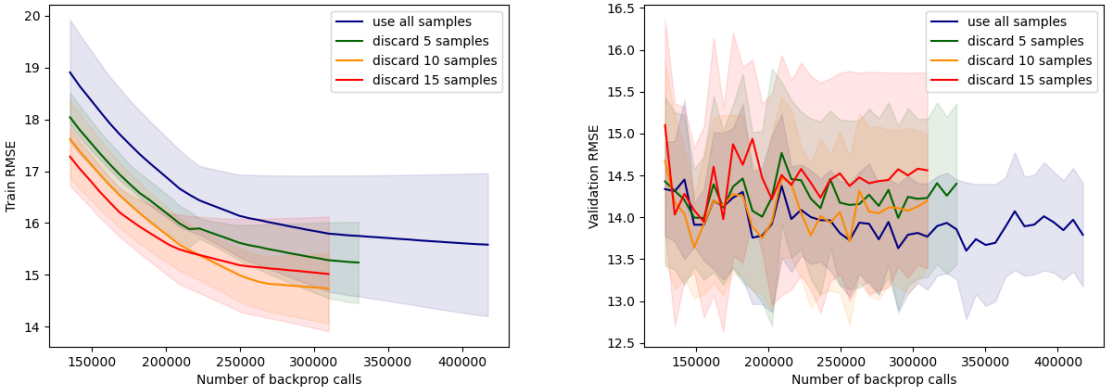
(a) Train (b) Validation

Figure 5.24: Burn-in effect on the fully-connected NN, dataset FD002



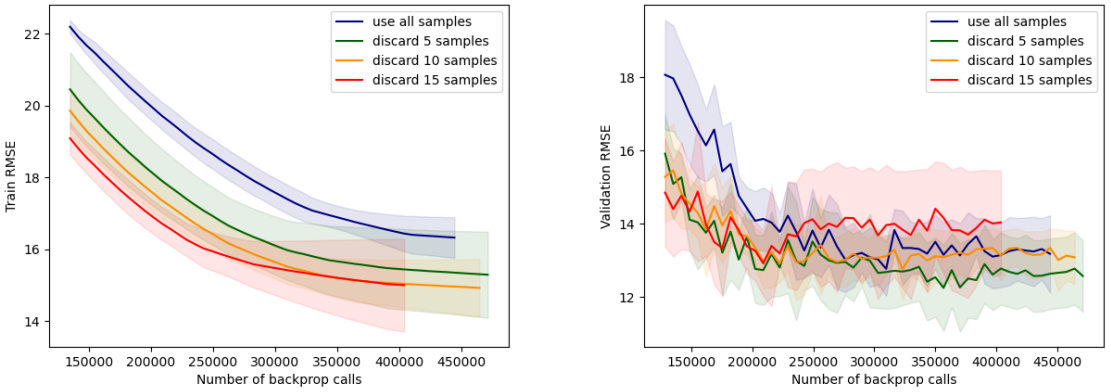
(a) Train (b) Validation

Figure 5.25: Burn-in effect on the LSTM, dataset FD002



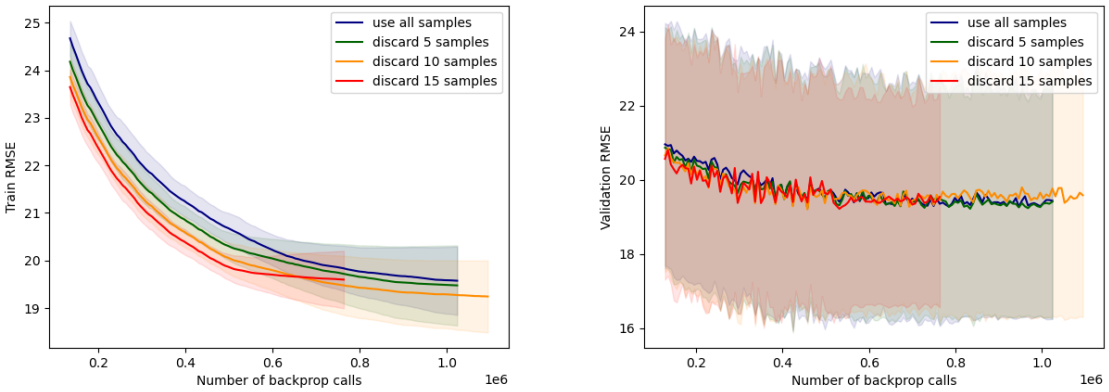
(a) Train (b) Validation

Figure 5.26: Burn-in effect on the fully-connected NN, dataset FD003



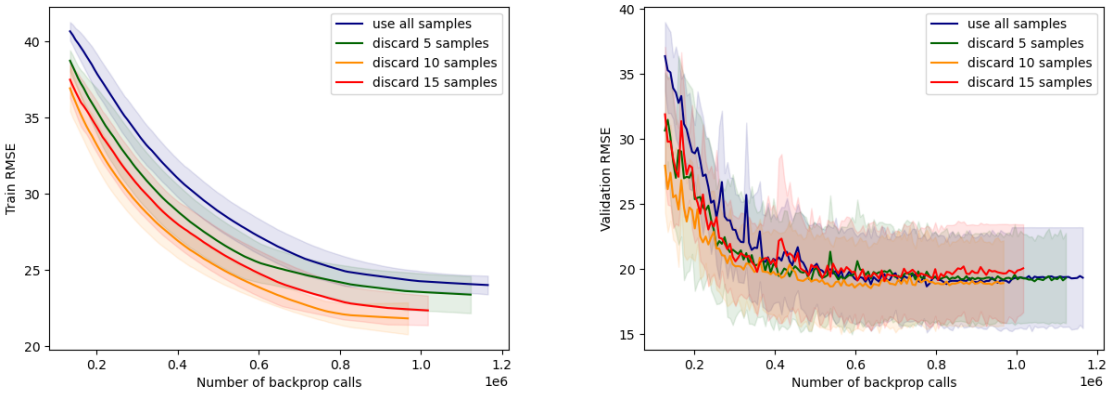
(a) Train (b) Validation

Figure 5.27: Burn-in effect on the LSTM, dataset FD003



(a) Train (b) Validation

Figure 5.28: Burn-in effect on the fully-connected NN, dataset FD004



(a) Train (b) Validation

Figure 5.29: Burn-in effect on the LSTM, dataset FD004

5.5 State-of-health/capacity estimation

5.5.1 Data exploration and preprocessing

State-of-health (SoH) estimation plays a prominent role in many predictive maintenance paradigms as it indicates the status of the batteries that power the whole system and channels if the batteries' health has crossed the safety threshold so that they will be replaced. In this work, we conduct a case study in forecasting the SoH of Lithium-ion batteries. Lithium-ion battery technology is used as cutting-edge technology in various commercial applications, including electric vehicles, consumer electronics [7, 128], and aerospace/aeronautics industries [86]. Javelin anti-tank missile, a symbol of Ukraine's resistance in the 21st century, is powered by lithium-ion batteries [32]. We use the satellite lithium-ion battery degradation dataset [115]. The SoH at time t of a lithium-ion battery is defined as

$$\text{SoH}(t) = \frac{\text{Capacity at time } t}{\text{Initial capacity}}.$$

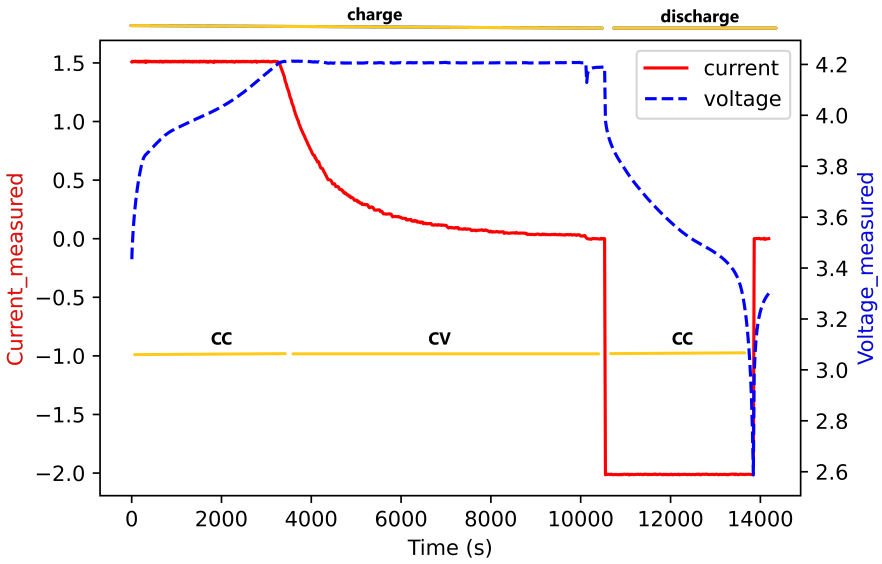


Figure 5.30: One cycle of charge-discharge

As the initial capacity is given by the manufacturers, the prediction task is amount to estimating the capacity of the battery. Usually, the battery is marked as reaching its end of life if the SoH is under 70%.

The dataset contains four batteries, namely, B0005, B0006, B0007, and B0018. Each battery underwent many repeated charge-discharge cycles at room temperature (24 Celsius degrees), resulting in the accelerated aging. For each cycle, batteries are charged in a constant current (CC) mode (1.5 A) until the battery voltage hit 4.2 V, then it is switched to the constant voltage (CV) mode until the current falls to 20 mA. Batteries are discharged at the constant current (CC) level of -2 A until the voltage drops to 2.7 V, 2.5 V, 2.2 V, and 2.5 V for batteries 5, 6, 7, and 18, respectively. Figure 5.30 illustrates a charge and discharge process.

Along the process, the battery terminal voltage, the battery output current, the battery temperature, the charger measurement current, the charger measurement voltage and the cycle time are recorded. During the discharge process, the actual capacity of the batteries is also provided, which serves as the groundtruth label in our study. Figure 5.31 shows the degradation of batteries, where the capacity value at each timestamp is what we want to accurately estimate.

Furthermore, additional impedance measurements obtained by electrochemical impedance spectroscopy (EIS) frequency swept from 0.1 Hz to 5 kHz are provided but are not taken into account here.

As the charge and discharge operations are repeated, distinct patterns can be seen in critical measurements such as battery current and voltage profiles, as well as battery surface temperature, as illustrated in Figures 5.32, 5.33, and 5.34. Figure 5.34, for example, shows that the temperature of an old battery reaches a peak when charged and discharged quicker than a fresh battery.

Data normalization and extraction We use the following data fields: Voltage measured, Current measured, Current load, Voltage load, Temperature measured, Time. Data in each field is then scaled to the range $(0, 1)$ by minmax scaler. As one cycle contains a large number of data points, we uniformly extract 15 points for each charge and discharge phase. For example, Figure 5.35 shows data points extracted from one discharge voltage profile.

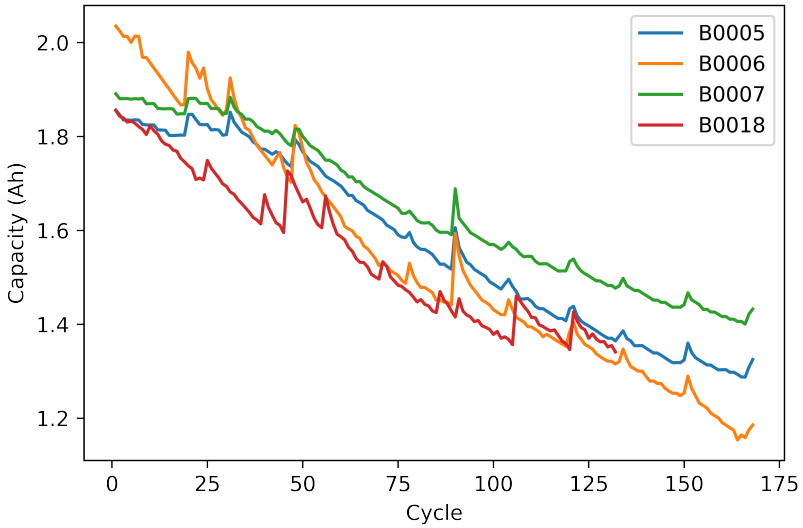
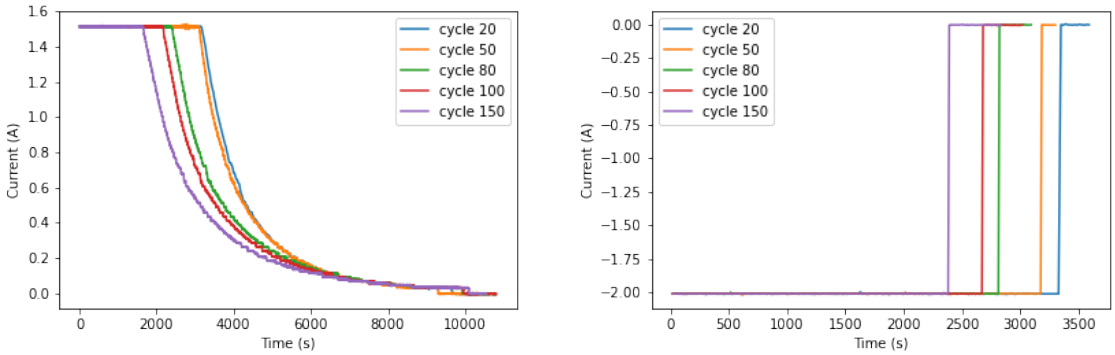
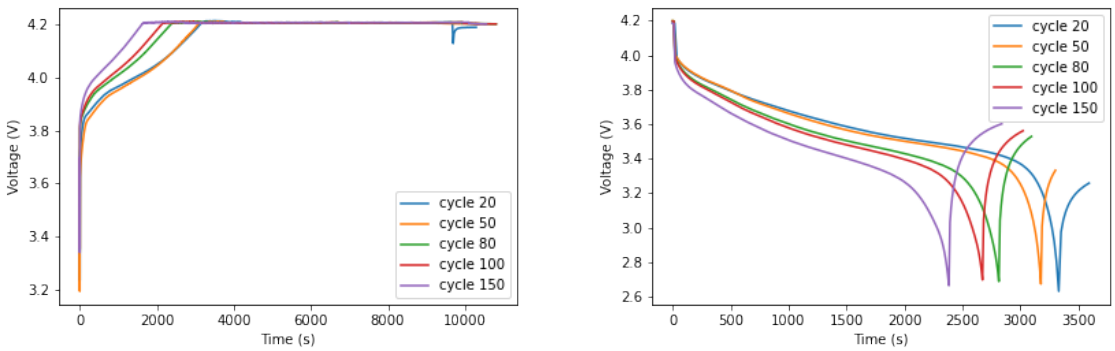


Figure 5.31: Capacity degradation



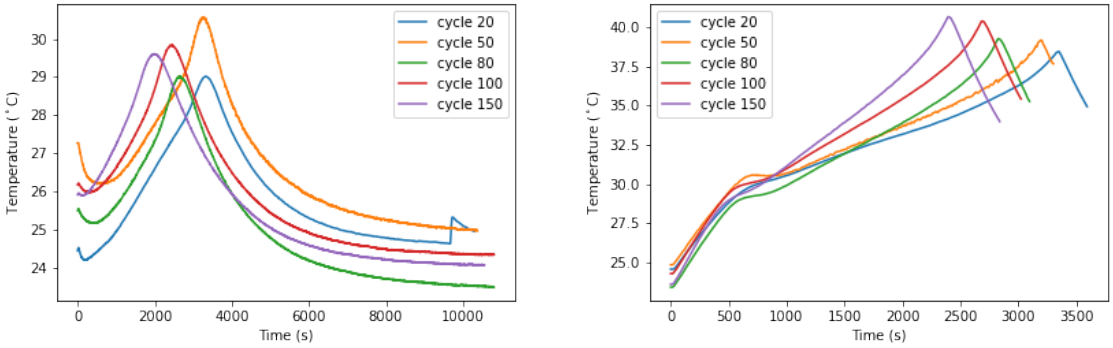
(a) Charging (b) Discharging

Figure 5.32: Current profiles change as the battery B005 ages



(a) Charging (b) Discharging

Figure 5.33: Voltage profiles change as the battery B005 ages



(a) Charging (b) Discharging

Figure 5.34: Temperature profiles change as the battery B005 ages

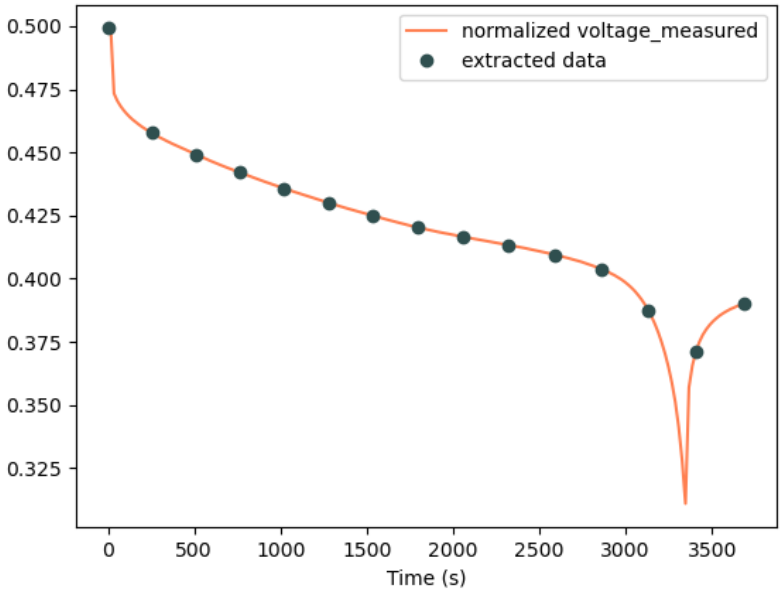


Figure 5.35: Extracted data from one discharge voltage profile

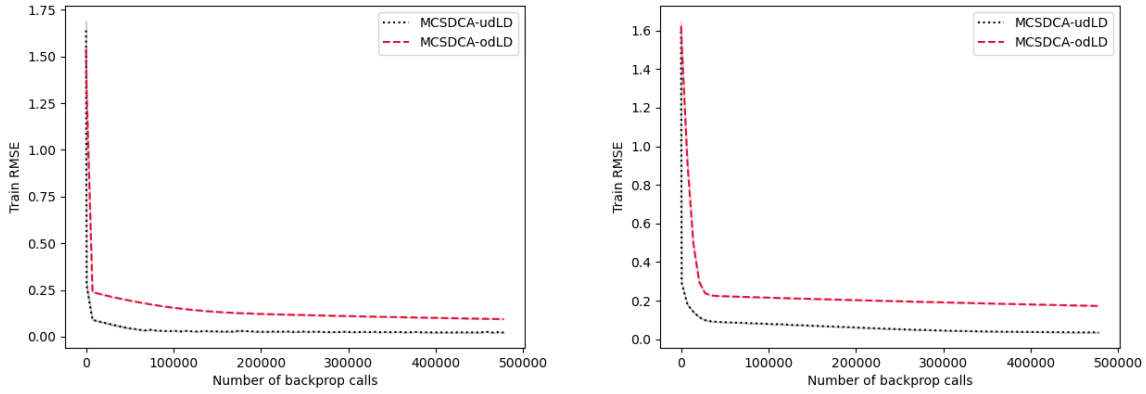
5.5.2 Experimental setups

We use standard RNN and LSTM with the following network configurations:

$$\text{RNN: } \text{input}_{6\times y} \rightarrow \text{rnn}_{256} \rightarrow \text{fc}_1$$
$$\text{LSTM: } \text{input}_{6\times y} \rightarrow \text{lstm}_{256}(\text{tanh}) \rightarrow \text{fc}_1$$

where $y = 15$ if either charge or discharge data is used, while $y = 30$ if both charge and discharge phases are used.

In this experiment, we observed that the MCSDCA-odLD exhibits a slow convergence. On the contrary, the MCSDCA-udLD significantly speeds up the training process, which is exactly what we expect the MCSDCA-ulLD to behave. For example, we can see an impressive performance of the MCSDCA-udLD in comparison with the MCSDCA-odLD.



(a) RNN (b) LSTM

Figure 5.36: The MCSDCA-udLD over the MCSDCA-odLD

Hence, in this experiment, we choose the MCSDCA-udLD as a good proxy for the MCSDCA.

Training procedure We have four batteries, B0005, B0006, B0007, and B0018, and we alternate between using three as the training set and one as the test set. This yielded four train-test pairings. Moreover, we either use only the charge or discharge data, or both in our experiment. For each train-test pair, we execute all algorithms at the same random initialization, and the results are averaged over five random runs. A staircase schedule for the learning rate is designed for the baseline algorithms (Adam, Adagrad, RMSprop, SGD): the initial learning rate is set as Adam(0.001), Adagrad(0.01), RMSprop(0.01), SGD(0.01), and the learning rate is lowered by a factor of 2 after each 10 epochs. For the MCSDCA-udLD, all parameters that are shared by the MCSDCA-odLD (ϵ, time t, number of Langevin steps, number of discarded samples, $\{\gamma_k\}$) are set to the same values as in Subsection 5.4.2. The MCSDCA-udLD has its own parameter, δ, which is now set to 0.1. We set the maximum number of training epochs for all methods to 1000 due to the modest size of the collected datasets, resulting in a backprop call budget of $1000N$, where N is the number of training samples. If an algorithm exceeds its computing budget, it will be automatically terminated.

5.5.3 Experimental results

Tables 5.6, 5.7, and 5.8 report the average training and test results when using charge, discharge, and both charge-discharge profiles, respectively. When only charge data is used, the MCSDCA-udLD obtains the best train and test results on both neural networks for the test battery B0006, on the RNN with the test battery B0005, and the best test result on the LSTM with the test battery B0018. Meanwhile, Adam wins the LSTM for the test battery B0005 and achieves the best test error on the RNN for the test battery B0007. Adagrad and RMSprop gain the best test results on the LSTM, test battery B0007 and the RNN, test battery B0018, respectively. Regarding the results using only discharge data, the MCSDCA-udLD is the sole winner on both types of neural networks for the test batteries B0005 and B0007. Meanwhile, Adam achieves the best test results on both types of neural networks for the test batteries B0006 and B0018, where the MCSDCA-udLD appears to be overfitting. Finally, the results using both charge and discharge data reveal that the MCSDCA-udLD outperform all competitive optimizers on all test batteries and two neural networks.

In comparison between the prediction results obtained using charge/discharge or both forms of data, the MCSDCA-udLD achieves much superior results when discharge data is used (only discharge, or both types of data), and underfits when just charge data is used. In general, the MCSDCA-udLD works best on the RNN when both data types are used; however, on the LSTM, the MCSDCA-udLD performs best when only discharge data is used.

Figures 5.37, 5.38, 5.39 show the training curves of all considered algorithms. As the dataset sizes are modest, these training curves are generated by computing the full-batch training loss at the present solution, which correctly captures the training process. It is observed that it is the only competition between the MCSDCA-udLD and Adam when other optimizers are left behind. In comparison between Adam and MCSDCA-udLD, Adam decreases the objective faster than MCSDCA-udLD at the beginning. Then, the MCSDCA-udLD, in most cases, can catch up with Adam at some point and then strongly surpasses Adam till the end of the training process.

Test battery	Algorithm	RNN		LSTM	
		Train	Test	Train	Test
B0005	MCSDCA-udLD	**0.0493**	**0.0554**	0.1311	0.2039
	Adam	0.0874	0.0961	**0.1267**	**0.1573**
	Adagrad	0.2011	0.1901	0.1923	0.1915
	RMSprop	0.202	0.1908	0.1891	0.1764
	SGD	0.2019	0.197	0.2177	0.1968
B0006	MCSDCA-udLD	**0.0468**	**0.1142**	**0.1228**	**0.2101**
	Adam	0.0643	0.1475	0.1486	0.213
	Adagrad	0.1731	0.2575	0.1716	0.2457
	RMSprop	0.1745	0.2563	0.1746	0.2566
	SGD	0.185	0.2558	0.1896	0.2666
B0007	MCSDCA-udLD	**0.0531**	0.2427	**0.1129**	0.4984
	Adam	0.0804	**0.101**	0.1205	0.3056
	Adagrad	0.2035	0.1896	0.2006	**0.1739**
	RMSprop	0.2064	0.1822	0.2059	0.1806
	SGD	0.2066	0.2267	0.2145	0.2253
B0018	MCSDCA-udLD	**0.0456**	0.188	0.1242	**0.1415**
	Adam	0.071	0.1994	**0.1106**	0.2765
	Adagrad	0.2048	0.1839	0.2015	0.1525
	RMSprop	0.2024	**0.1506**	0.2015	0.1546
	SGD	0.2092	0.1787	0.2212	0.1797

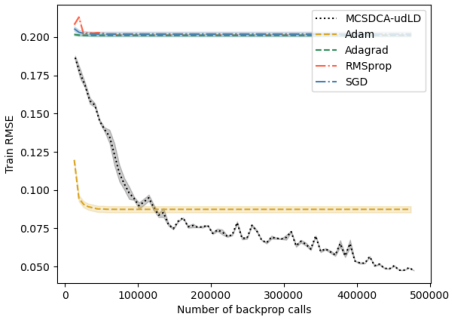
Table 5.6: Results using only charge data

Test battery	Algorithm	RNN		LSTM	
		Train	Test	Train	Test
B0005	MCSDCA-udLD	**0.0223**	**0.0531**	**0.0351**	**0.0612**
	Adam	0.0287	0.0568	0.0634	0.0879
	Adagrad	0.1964	0.1861	0.0937	0.1006
	RMSprop	0.1976	0.1896	0.1074	0.1351
	SGD	0.0903	0.1015	0.1936	0.1850
B0006	MCSDCA-udLD	**0.0261**	0.0653	**0.0345**	0.0616
	Adam	0.0282	**0.0541**	0.0706	**0.0603**
	Adagrad	0.1677	0.2451	0.0972	0.0966
	RMSprop	0.1742	0.2557	0.1035	0.1127
	SGD	0.0952	0.1292	0.1743	0.2586
B0007	MCSDCA-udLD	**0.0274**	**0.0706**	**0.0407**	**0.0432**
	Adam	0.0423	0.1235	0.0618	0.1198
	Adagrad	0.1657	0.1395	0.0888	0.1059
	RMSprop	0.2058	0.1806	0.1130	0.0867
	SGD	0.0878	0.0970	0.2031	0.1845
B0018	MCSDCA-udLD	**0.0145**	0.1315	**0.0298**	0.0987
	Adam	0.0312	**0.0563**	0.0516	**0.0871**
	Adagrad	0.2023	0.1538	0.0817	0.1246
	RMSprop	0.2068	0.1568	0.1050	0.1378
	SGD	0.0795	0.0997	0.2041	0.1409

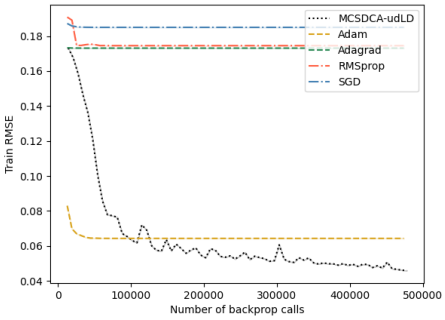
Table 5.7: Results using only discharge data

Test battery	Algorithm	RNN		LSTM	
		Train	Test	Train	Test
B0005	MCSDCA-udLD	**0.0262**	**0.0292**	**0.0594**	**0.0666**
	Adam	0.0576	0.047	0.0699	0.0719
	Adagrad	0.1908	0.1866	0.1224	0.1243
	RMSprop	0.202	0.1908	0.1178	0.1299
	SGD	0.1336	0.1357	0.2128	0.1958
B0006	MCSDCA-udLD	**0.0231**	**0.0495**	**0.0596**	**0.0657**
	Adam	0.0608	0.0836	0.1134	0.1404
	Adagrad	0.1713	0.2531	0.1279	0.1723
	RMSprop	0.1746	0.2564	0.124	0.1417
	SGD	0.1348	0.1876	0.1911	0.2636
B0007	MCSDCA-udLD	**0.0272**	**0.0456**	0.0605	**0.1718**
	Adam	0.0622	0.071	**0.0595**	0.1755
	Adagrad	0.1819	0.1459	0.0944	0.1992
	RMSprop	0.2064	0.1822	0.0937	0.1846
	SGD	0.1597	0.1547	0.2153	0.2317
B0018	MCSDCA-udLD	**0.0291**	**0.0649**	**0.0485**	**0.0977**
	Adam	0.0511	0.1206	0.055	0.1035
	Adagrad	0.2062	0.1528	0.0948	0.1257
	RMSprop	0.2088	0.1572	0.1364	0.1353
	SGD	0.1261	0.1304	0.2225	0.1643

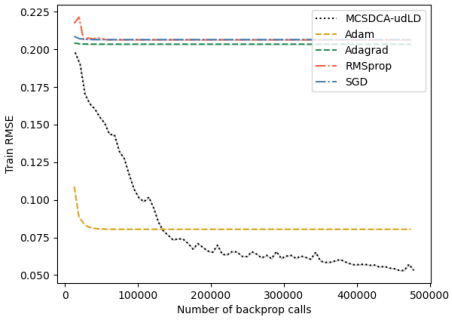
Table 5.8: Results using both charge and discharge data



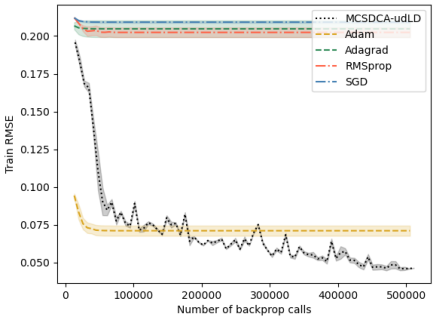
(a) RNN, Battery B0005



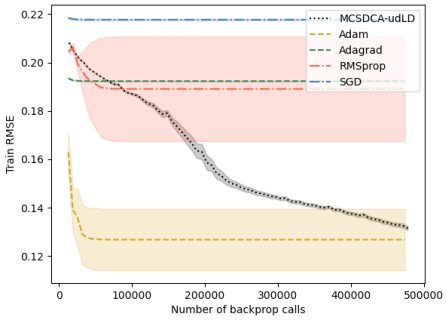
(b) RNN, Battery B0006



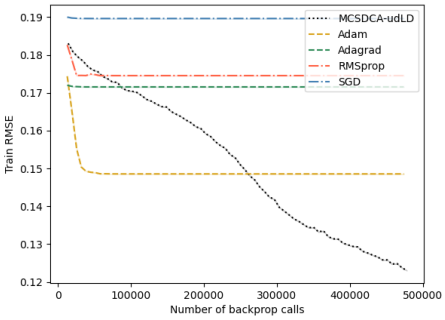
(c) RNN, Battery B0007



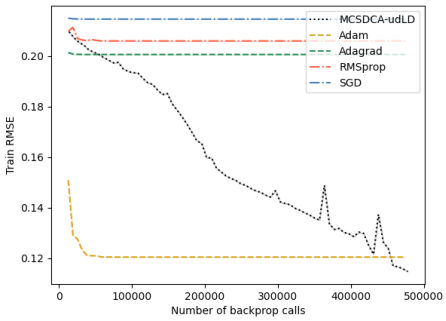
(d) RNN, Battery B0018



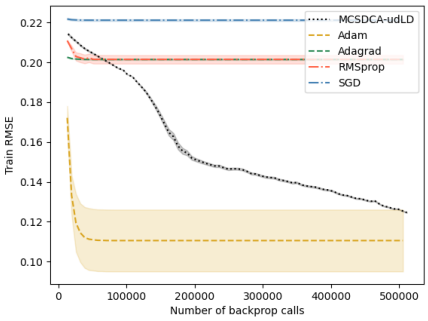
(e) LSTM, Battery B0005



(f) LSTM, Battery B0006

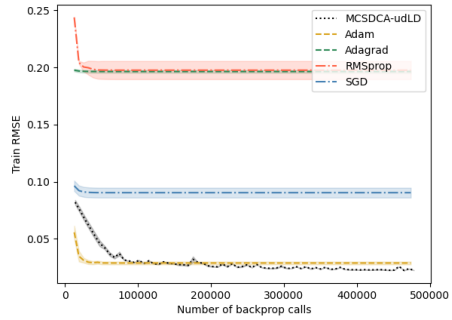


(g) LSTM, Battery B0007

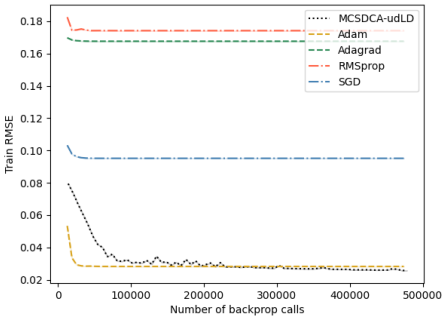


(h) LSTM, Battery B0018

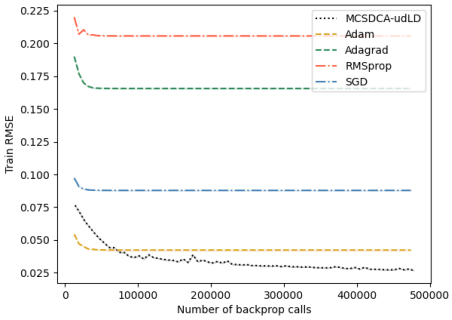
Figure 5.37: Training curves, charge data



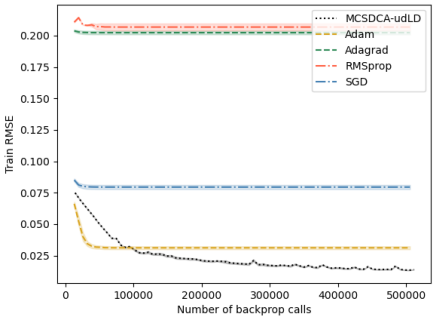
(a) RNN, Battery B0005



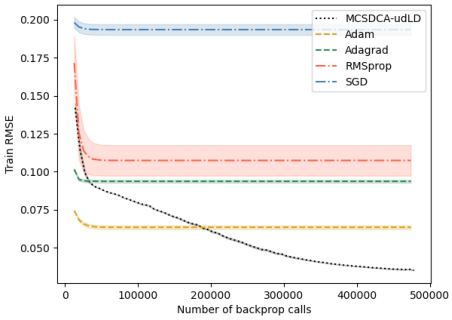
(b) RNN, Battery B0006



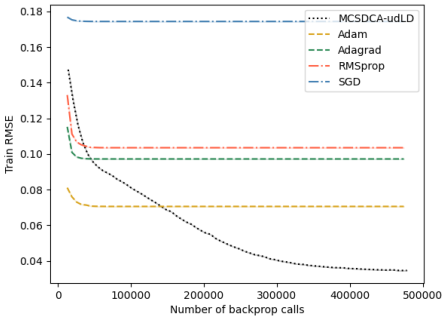
(c) RNN, Battery B0007



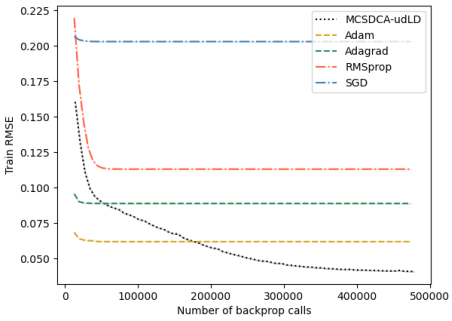
(d) RNN, Battery B0018



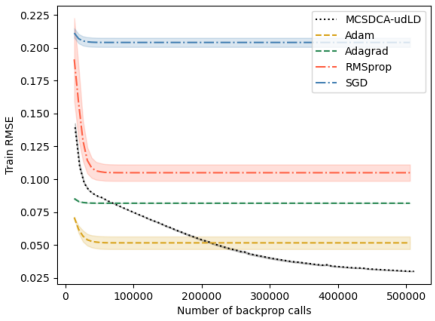
(e) LSTM, Battery B0005



(f) LSTM, Battery B0006

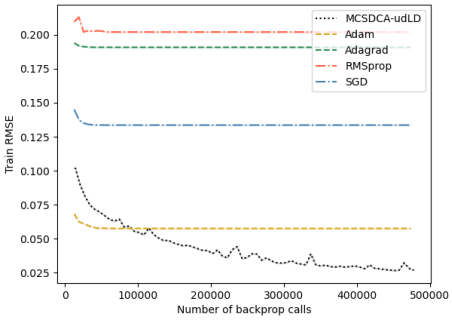


(g) LSTM, Battery B0007

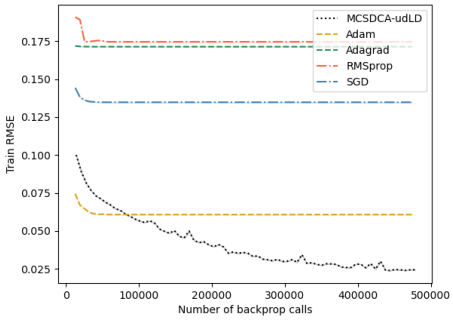


(h) LSTM, Battery B0018

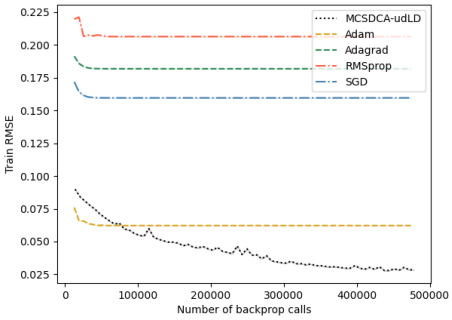
Figure 5.38: Training curves, discharge data



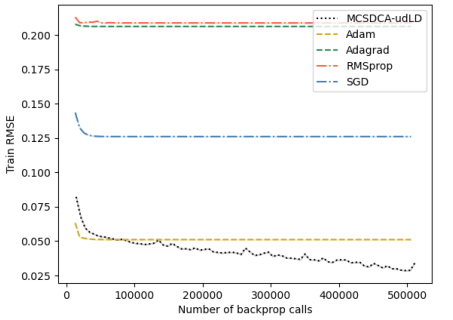
(a) RNN, Battery B0005



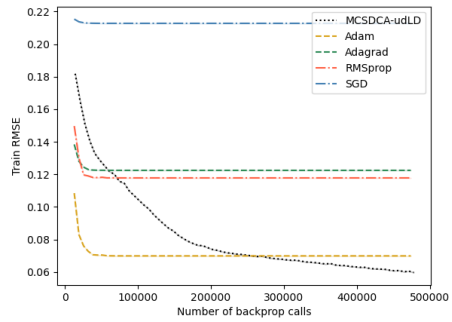
(b) RNN, Battery B0006



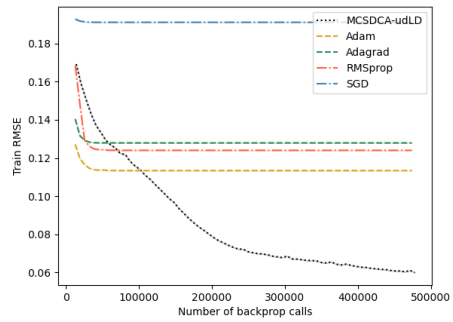
(c) RNN, Battery B0007



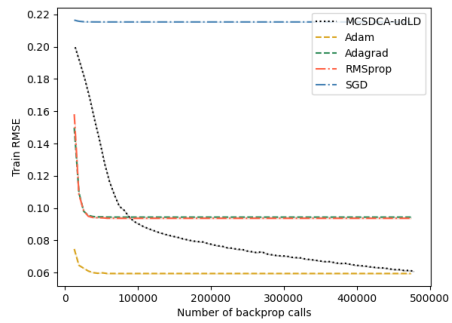
(d) RNN, Battery B0018



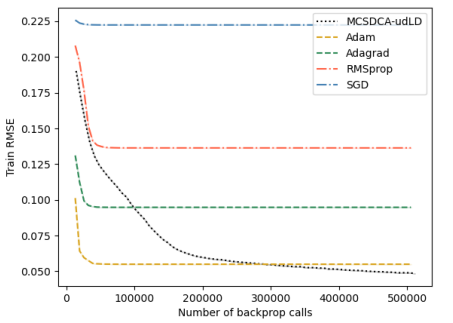
(e) LSTM, Battery B0005



(f) LSTM, Battery B0006



(g) LSTM, Battery B0007

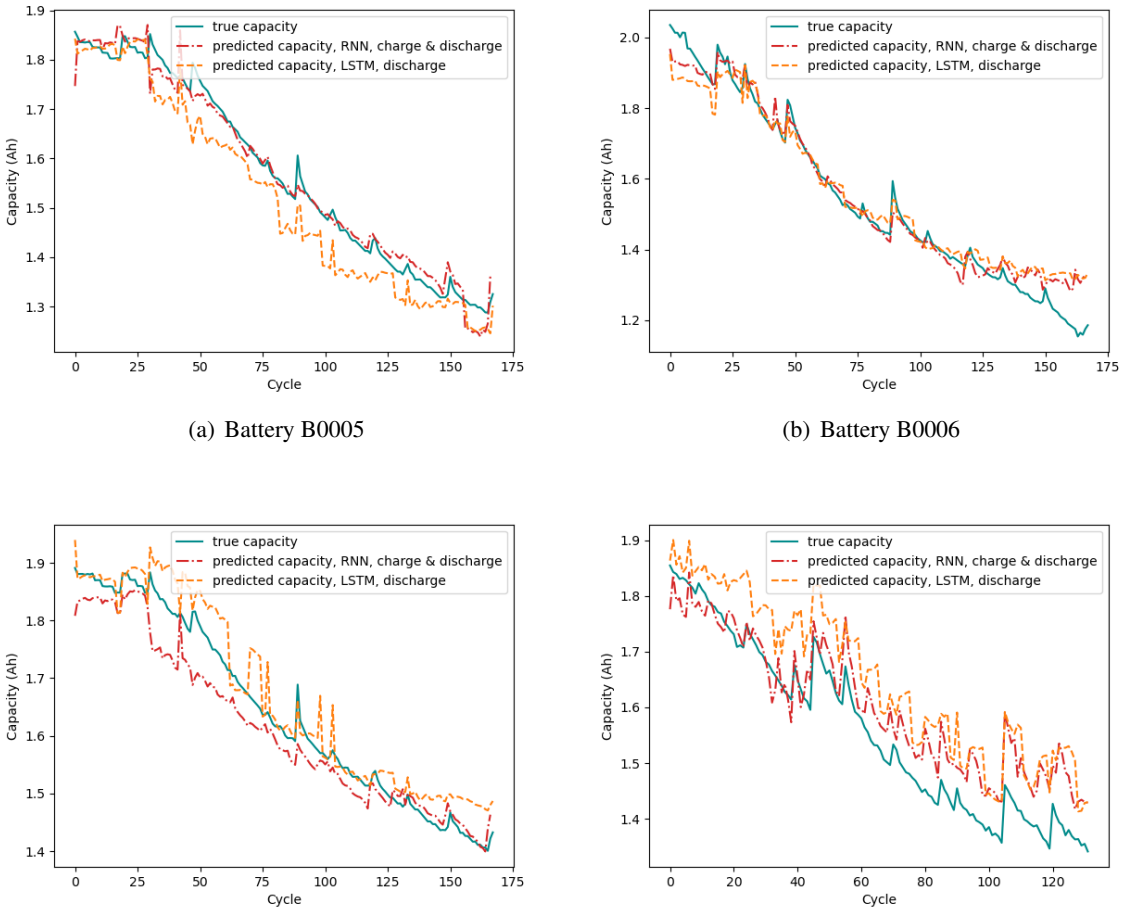


(h) LSTM, Battery B0018

Figure 5.39: Training curves, both charge and discharge data

Figure 5.40 shows the prediction results obtained by the MCSDCA-udLD on four datasets. It is

observed that the predicted capacity follows closely the trend of the true capacity, and the RNN produces better prediction than that of the LSTM.



(a) Battery B0005

(b) Battery B0006

(c) Battery B0007

(d) Battery B0018

Figure 5.40: Prediction results, models trained by MCSDCA-udLD

5.6 Conclusion

In this chapter, we have studied predictive maintenance via a deep learning approach, with a particular emphasis on the optimization aspect of training deep neural networks. Two case studies have been carried out, which are the RUL prediction and SoH estimation. The study confirms the training ability of the two proposed algorithms, MCSDCA-odLD and MCSDCA-udLD, on the diverse structures of neural works. Furthermore, the MCSDCA-udLD has been shown to accelerate remarkably the training speed in cases where the MCSDCA-odLD appears slow to converge.

Chapter 6

Conclusion and perspectives

In this thesis, we have studied three classes of DC programs: large-sum, stochastic with exogenous uncertainty with i.i.d. samples, stochastic that involves endogenous uncertainty with non-i.i.d. samples. For each class of programs, we proposed the corresponding stochastic DCA schemes to handle the associated challenges. Several applications in machine learning have been used to verify the effectiveness of the proposed methods. Especially, with the predictive maintenance applications, the MCSDCA-odLD and MCSDCA-udLD have been successful in training a number of neural networks for the associated prediction tasks.

For solving the class of large-sum DC programs, we have proposed two stochastic DCA schemes, DCA-SVRG and DCA-SAGA, integrating variance reduction techniques. The theoretical convergence analysis is studied intensively under both sampling strategies, namely, with and without replacement. The numerical experiments have shown the virtues of the proposed methods via three machine learning problems: nonnegative principal component analysis, group variable selection in multiclass logistic regression, and sparse linear regression. Our proposed algorithms inherit the virtues as well as the unavoidable limitations of the stochastic estimators used. The proposed methods outperform the standard DCA in terms of gradient complexity, theoretically and experimentally, and they also outperform experimentally the stochastic DCA proposed in [73] that uses the SAG estimator. However, the SVRG estimator has a limitation that is the variance reduction term inside each epoch is not updated, which undermines the high correlation requirement of the control variate principle. On the other hand, the SAGA, in general, suffers from the storage burden. Our perspective is to use another estimator called SARAH [94] that can possibly address the above drawbacks. Similar to SVRG, SARAH is an epoch-based estimator where the full gradient is computed at the beginning of each epoch. However, unlike SVRG where the variance-reduction term is fixed within each epoch, SARAH progressively updates that term as

$$\nu^k = \nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^{k-1}) + \nu^{k-1}.$$

This update comes at the cost of the gradient estimator no longer being unbiased, which is likely to be more difficult to theoretically analyze. However, the performance of such a biased estimator is pretty well, as evidenced by [94, 102] and our experiments in Chapter 2. This may inspire our future work on the combination between DCA and SARAH.

For solving the class of stochastic DC programs with exogenous uncertainty and i.i.d. samples com-

ing as a stream, we have designed the online stochastic DCA and its two variants. The convergence properties of the proposed algorithms are rigorously studied, where we established the almost sure convergence to critical points. In addition to the efficiency and flexibility of DCA, the advantages of osDCA schemes include the reduction of storage burden and the capacity to cope with the changes in the data distribution (if any). As a limitation, our main algorithm, in its most general form, requires the knowledge on the Rademacher complexity of the family of functions $\{g(\cdot, z) : z \in \Xi\}$, which is not always easy to compute. The proposed algorithms' practical behaviors have been thoroughly studied on the expected problem of PCA. The numerical experiments validate the effectiveness of the proposed algorithms, in which they yield good results in a short period of time. Furthermore, the adaptability of osDCA schemes has been demonstrated: when the data distribution changes, our algorithms swiftly adapt to the new distribution. Additional experimental findings confirm the importance of the DC decomposition and the convex solver (for solving subproblems) in DCA. In future work, we will extend our study to include (exogenous) stochastic DC programs with stochastic DC constraints. To the best of our knowledge, the most recent advances in the field of stochastic programming with stochastic constraints are studies for L-smooth functions [136, 80]. Meanwhile, in the context of deterministic optimization, *general DC programs* (DC objective and DC constraints) have been extensively researched, and the *general DCA* has been established [68, 104]. We aim to investigate some stochastic versions of the general DCA for solving the mentioned class of stochastic DC programs with stochastic DC constraints.

For solving a class of stochastic DC programs where endogenous uncertainty is involved and i.i.d. samples are unavailable, we have designed a Markov chain stochastic DCA that uses only Markovian data. The proposed algorithm is guaranteed to find DC critical points in both asymptotic and non-asymptotic senses. As an important extension, we provide additional analysis to time-inhomogeneous Markov chains, demonstrating that previously established convergence results still hold true in this case. We next apply the proposed algorithm to deep learning via PDEs regularization, yielding two MCSDCA realizations, MCSDCA-odLD and MCSDCA-udLD, which are based on overdamped and underdamped Langevin dynamics, respectively. The efficiency of these two specific algorithms have been shown in the predictive maintenance applications via the deep learning approach. Two case studies are carried out: RUL prediction and SoH estimate. Given several suitable neural network topologies, the study confirms the learnability of the MCSDCA-odLD and MCSDCA-udLD. More importantly, the MCSDCA-udLD has exhibited acceleration in cases where the MCSDCA-odLD appears to be slow to converge. This finding matches our expectation as the underdamped Langevin dynamics, once tailored, should show acceleration over the overdamped Langevin dynamics. However, as a disadvantage, the proposed methods have not yet been able to address the general class of stochastic programs with endogenous uncertainty in which the analytical form of the family of distributions is unknown since it is nearly impossible to highlight a DC decomposition in such a case. This class of problems is much beyond the scope of DC programming and requires a brand new methodology. Furthermore, there is a gap between theory and practice that will be a subject in our future work: the discretization errors (when discretizing the overdamped and underdamped Langevin diffusions) and the stochastic noise induced by the use of mini-batch not yet to be quantified.

Bibliography

[1] C. H. Aladag, U. Yolcu, E. Egrioglu, and A. Z. Dalar. A new time invariant fuzzy time series forecasting method based on particle swarm optimization. *Applied Soft Computing*, 12(10):3291–3299, 2012.

[2] Z. Allen-Zhu and Y. Yuan. Improved svrg for non-strongly-convex or sum-of-non-convex objectives. In *International conference on machine learning*, pages 1080–1089. PMLR, 2016.

[3] M. Bačák and J. M. Borwein. On difference convexity of locally lipschitz functions. *Optimization*, 60(8-9):961–978, 2011.

[4] S. C. Bagley, H. White, and B. A. Golomb. Logistic regression in the medical literature:: Standards for use and reporting, with particular attention to one medical domain. *Journal of clinical epidemiology*, 54(10):979–985, 2001.

[5] D. Belomestny, L. Iosipoi, E. Moulines, A. Naumov, and S. Samsonov. Variance reduction for markov chains with application to mcmc. *Statistics and Computing*, 30(4):973–997, 2020.

[6] T. Benkedjouh, K. Medjaher, N. Zerhouni, and S. Rechak. Remaining useful life estimation based on nonlinear feature reduction and support vector regression. *Engineering Applications of Artificial Intelligence*, 26(7):1751–1760, 2013.

[7] M. Berecibar, I. Gandiaga, I. Villarreal, N. Omar, J. Van Mierlo, and P. Van den Bossche. Critical review of state of health estimation methods of li-ion batteries for real applications. *Renewable and Sustainable Energy Reviews*, 56:572–587, 2016.

[8] D. Bertsekas, A. Nedić, and A. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific optimization and computation series. Athena Scientific, 2003.

[9] D. P. Bertsekas and J. N. Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10(3):627–642, 2000.

[10] J. Bolte, A. Daniilidis, and A. Lewis. The łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17(4):1205–1223, 2007.

[11] J. Bolte, A. Daniilidis, A. Lewis, and M. Shiota. Clarke subgradients of stratifiable functions. *SIAM Journal on Optimization*, 18(2):556–572, 2007.

[12] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[13] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60(2):223–311, 2018.

[14] S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: a survey of some recent advances. *ESAIM: probability and statistics*, 9:323–375, 2005.

[15] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[16] P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, volume 98, pages 82–90. Citeseer, 1998.

[17] H. Brezis. *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media, 2010.

[18] E. J. Candes, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905, 2008.

[19] Y. Cao, J. Lu, and L. Wang. On explicit l2-convergence rate estimate for underdamped langevin dynamics. *arXiv preprint arXiv:1908.04746*, 2019.

[20] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

[21] P. Chaudhari, A. Oberman, S. Osher, S. Soatto, and G. Carlier. Deep relaxation: partial differential equations for optimizing deep neural networks. *Research in the Mathematical Sciences*, 5(3):1–30, 2018.

[22] G. Chen, D. Zeng, and M. R. Kosorok. Personalized dose finding using outcome weighted learning. *Journal of the American Statistical Association*, 111(516):1509–1521, 2016.

[23] Z. Chen, Z. Yuan, J. Yi, B. Zhou, E. Chen, and T. Yang. Universal stagewise learning for non-convex problems with convergence on averaged solutions. In *7th International Conference on Learning Representations*. OpenReview.net, 2019.

[24] C. Cheng, A. Sa-Ngasoongsong, O. Beyca, T. Le, H. Yang, Z. Kong, and S. T. Bukkapatnam. Time series forecasting for nonlinear and non-stationary processes: a review and comparative study. *Iie Transactions*, 47(10):1053–1071, 2015.

[25] X. Cheng, N. S. Chatterji, P. L. Bartlett, and M. I. Jordan. Underdamped langevin mcmc: A non-asymptotic analysis. In *Conference on learning theory*, pages 300–323. PMLR, 2018.

[26] F. H. Clarke. *Optimization and nonsmooth analysis*. John Wiley & Sons, 1983.

[27] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *Proceedings of the 23rd international conference on Machine learning*, pages 201–208, 2006.

[28] R. Correa, M. A. Lopez, and P. Pérez-Aros. Necessary and sufficient optimality conditions in dc semi-infinite programming. *SIAM Journal on Optimization*, 31(1):837–865, 2021.

[29] Y. Cui, J.-S. Pang, and B. Sen. Composite difference-max programs for modern statistical estimation problems. *SIAM Journal on Optimization*, 28(4):3344–3374, 2018.

[30] D. Davis and D. Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

[31] A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

[32] U. S. Department of the Army. Javelin—close combat missile system, medium. Technical report, 2013.

[33] M. Dhanaraj and P. P. Markopoulos. Stochastic principal component analysis via mean absolute projection maximization. In *IEEE Global Conference on Signal and Information Processing*, pages 1–5. IEEE, 2019.

[34] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[35] T. T. Doan, L. M. Nguyen, N. H. Pham, and J. Romberg. Convergence rates of accelerated markov gradient descent with applications in reinforcement learning. *arXiv preprint arXiv:2002.02873*, 2020.

[36] M. Dong and D. He. A segmental hidden semi-markov model (hsmm)-based diagnostics and prognostics framework and methodology. *Mechanical systems and signal processing*, 21(5):2248–2266, 2007.

[37] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[38] J. C. Duchi, A. Agarwal, M. Johansson, and M. I. Jordan. Ergodic mirror descent. *SIAM Journal on Optimization*, 22(4):1549–1578, 2012.

[39] J. Dupačová. Optimization under exogenous and endogenous uncertainty. In *Proceedings of the 24th International conference on Mathematical Methods in Economics*, page 131–136, 2006.

[40] A. Durmus and E. Moulines. Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, 2017.

[41] M. L. Eaton. Multivariate statistics: a vector space approach. *JOHN WILEY & SONS, INC., 605 THIRD AVE., NEW YORK, NY 10158, USA, 1983, 512*, 1983.

[42] Y. M. Ermoliev and V. I. Norkin. Sample average approximation method for compound stochastic optimization problems. *SIAM Journal on Optimization*, 23(4):2231–2263, 2013.

[43] L. C. Evans. *Partial differential equations*, volume 19. American Mathematical Soc., 2010.

[44] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

[45] D. Gao and M. Huang. Prediction of remaining useful life of lithium-ion battery based on multi-kernel support vector machine with particle swarm optimization. *Journal of Power Electronics*, 17(5):1288–1297, 2017.

[46] G. Gasso, A. Pappaioannou, M. Spivak, and L. Bottou. Batch and online learning algorithms for nonconvex neyman-pearson classification. *ACM transactions on intelligent systems and technology*, 2(3):1–19, 2011.

[47] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[48] S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[49] S. Ghadimi, G. Lan, and H. Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1-2):267–305, 2016.

[50] D. A. Gomes and E. Valdinoci. Entropy penalization methods for hamilton–jacobi equations. *Advances in Mathematics*, 215(1):94–152, 2007.

[51] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In *international conference on machine learning*, pages 37–45. PMLR, 2013.

[52] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[53] A. Henriksen and R. Ward. Adaoja: Adaptive learning rates for streaming PCA. *arXiv preprint arXiv:1905.12115*, 2019.

[54] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[55] C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004.

[56] Z. Hu, F. Huang, and H. Huang. Optimal underdamped langevin mcmc method. *Advances in Neural Information Processing Systems*, 34, 2021.

[57] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[58] S. J. Reddi, S. Sra, B. Poczos, and A. J. Smola. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In *Advances in Neural Information Processing Systems*, volume 29, pages 1145–1153, 2016.

[59] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

[60] M. Jouin, R. Gouriveau, D. Hissel, M.-C. Péra, and N. Zerhouni. Degradations analysis and aging modeling for health assessment and prognostics of pemfc. *Reliability Engineering & System Safety*, 148:78–95, 2016.

[61] M. Jouin, R. Gouriveau, D. Hissel, M.-C. Péra, and N. Zerhouni. Particle filter-based prognostics: Review, discussion and perspectives. *Mechanical Systems and Signal Processing*, 72:2–31, 2016.

[62] J. Kim, Y. Kim, and Y. Kim. A gradient-based optimization algorithm for lasso. *Journal of Computational and Graphical Statistics*, 17(4):994–1009, 2008.

[63] G. King and L. Zeng. Logistic regression in rare events data. *Political analysis*, 9(2):137–163, 2001.

[64] S. S. Lavenberg, T. L. Moeller, and P. D. Welch. Statistical results on control variables with application to queueing network simulation. *Operations Research*, 30(1):182–202, 1982.

[65] S. S. Lavenberg and P. D. Welch. A perspective on the use of control variables to increase the efficiency of monte carlo simulations. *Management Science*, 27(3):322–335, 1981.

[66] H. M. Le, H. A. Le Thi, and M. C. Nguyen. Sparse semi-supervised support vector machines by DC programming and DCA. *Neurocomputing*, 153:62–76, 2015.

[67] H. A. Le Thi. A new approximation for the ℓ_0-norm. *Research Report LITA EA 3097*, 2012.

[68] H. A. Le Thi, V. N. Huynh, and T. Pham Dinh. DC programming and DCA for general DC programs. In *Advanced Computational Methods for Knowledge Engineering*, pages 15–35. Springer, 2014.

[69] H. A. Le Thi, V. N. Huynh, and T. Pham Dinh. Convergence analysis of difference-of-convex algorithm with subanalytic data. *Journal of Optimization Theory and Applications*, 179(1):103–126, 2018.

[70] H. A. Le Thi, V. N. Huynh, T. Pham Dinh, and H. P. H. Luu. Stochastic difference-of-convex-functions algorithms for nonconvex programming. *SIAM Journal on Optimization*, 32(3):2263–2293, 2022.

[71] H. A. Le Thi, H. M. Le, and T. P. Dinh. Feature selection in machine learning: an exact penalty approach using a difference of convex function algorithm. *Machine Learning*, 101(1):163–186, 2015.

[72] H. A. Le Thi, H. M. Le, D. N. Phan, and B. Tran. Stochastic DCA for the large-sum of non-convex functions problem and its application to group variable selection in classification. In *International Conference on Machine Learning*, pages 3394–3403. PMLR, 2017.

[73] H. A. Le Thi, H. M. Le, D. N. Phan, and B. Tran. Stochastic DCA for minimizing a large sum of dc functions with application to multi-class logistic regression. *Neural Networks*, 132:220–231, 2020.

[74] H. A. Le Thi, H. M. Le, D. N. Phan, and B. Tran. Novel DCA based algorithms for a special class of nonconvex problems with application in machine learning. *Applied Mathematics and Computation*, 409:125904, 2021.

[75] H. A. Le Thi and T. Pham Dinh. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of operations research*, 133(1-4):23–46, 2005.

[76] H. A. Le Thi and T. Pham Dinh. DC programming and DCA: thirty years of developments. *Mathematical Programming, Special Issue dedicated to : DC Programming - Theory, Algorithms and Applications*, 169(1):5–68, 2018.

[77] H. A. Le Thi, T. Pham Dinh, H. M. Le, and X. T. Vo. DC approximation approaches for sparse optimization. *European Journal of Operational Research*, 244(1):26–46, 2015.

[78] H. A. Le Thi and D. N. Phan. Dc programming and dca for sparse optimal scoring problem. *Neurocomputing*, 186:170–181, 2016.

[79] C. Lee, Y. Cao, and K. H. Ng. Big data analytics for predictive maintenance strategies. In *Supply Chain Management in the Big Data Era*, pages 50–74. IGI Global, 2017.

[80] A. Liu, V. K. Lau, and B. Kananian. Stochastic successive convex approximation for non-convex constrained stochastic optimization. *IEEE Transactions on Signal Processing*, 67(16):4189–4203, 2019.

[81] D. Liu, J. Zhou, D. Pan, Y. Peng, and X. Peng. Lithium-ion battery remaining useful life estimation with an optimized relevance vector machine algorithm with incremental learning. *Measurement*, 63:143–151, 2015.

[82] J. Liu, Y. Cui, J.-S. Pang, and S. Sen. Two-stage stochastic programming with linearly bi-parameterized quadratic recourse. *SIAM Journal on Optimization*, 30(3):2530–2558, 2020.

[83] S. Łojasiewicz. Sur la géométrie semi-et sous-analytique. In *Annales de l'institut Fourier*, volume 43, pages 1575–1595, 1993.

[84] J. Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. *Advances in Neural Information Processing Systems*, 26:2283–2291, 2013.

[85] J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

[86] R. Marsh, S. Vukson, S. Surampudi, B. Ratnakumar, M. Smart, M. Manzo, and P. Dalton. Li ion batteries for aerospace applications. *Journal of power sources*, 97:25–27, 2001.

[87] S. Meisenbacher, M. Turowski, K. Phipps, M. Rätz, D. Müller, V. Hagenmeyer, and R. Mikut. Review of automated time series forecasting pipelines. *arXiv preprint arXiv:2202.01712*, 2022.

[88] M. Metel and A. Takeda. Simple stochastic gradient methods for non-smooth non-convex regularized optimization. In *International Conference on Machine Learning*, pages 4537–4545. PMLR, 2019.

[89] M. Metivier. *Semimartingales*. Walter de Gruyter, 2011.

[90] A. Montanari and E. Richard. Non-negative principal component analysis: Message passing algorithms and sharp asymptotics. *IEEE Transactions on Information Theory*, 62(3):1458–1484, 2015.

[91] B. S. Mordukhovich and N. M. Nam. An easy path to convex analysis and applications. *Synthesis Lectures on Mathematics and Statistics*, 6(2):1–218, 2013.

[92] P. Mörters and Y. Peres. *Brownian motion*, volume 30. Cambridge University Press, 2010.

[93] D. Nagaraj, X. Wu, G. Bresler, P. Jain, and P. Netrapalli. Least squares regression with markovian data: Fundamental limits and algorithms. *Advances in Neural Information Processing Systems*, 33:16666–16676, 2020.

[94] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pages 2613–2621. PMLR, 2017.

[95] A. Nitanda and T. Suzuki. Stochastic Difference of Convex Algorithm and its Application to Training Deep Boltzmann Machines. In *Artificial intelligence and statistics*, pages 470–478, Fort Lauderdale, FL, USA, 2017. PMLR.

[96] C. Nugteren and V. Codreanu. Cltune: A generic auto-tuner for opencl kernels. In *2015 IEEE 9th International Symposium on Embedded MulticoreMany-core Systems-on-Chip*, pages 195–202. IEEE, 2015.

[97] C. S. Ong and H. A. Le Thi. Learning sparse classifiers with difference of convex functions algorithms. *Optimization Methods and Software*, 28(4):830–854, 2013.

[98] K. Park, Y. Choi, W. J. Choi, H.-Y. Ryu, and H. Kim. Lstm-based battery remaining useful life prediction with multi-channel charging profiles. *IEEE Access*, 8:20786–20798, 2020.

[99] S. Y. Park and Y. Liu. Robust penalized logistic regression with truncated loss functions. *Canadian Journal of Statistics*, 39(2):300–323, 2011.

[100] G. D. Pasa, I. Medeiros, and T. Yoneyama. Operating condition-invariant neural network-based prognostics methods applied on turbofan aircraft engines. In *Annual conference of the phm society*, volume 11, pages 1–10, 2019.

[101] N. H. Pham, L. M. Nguyen, D. T. Phan, and Q. Tran-Dinh. Proxsarah: An efficient algorithmic framework for stochastic composite nonconvex optimization. *Journal of Machine Learning Research*, 21(110):1–48, 2020.

[102] N. H. Pham, L. M. Nguyen, D. T. Phan, and Q. Tran-Dinh. ProxSARAH: An efficient algorithmic framework for stochastic composite nonconvex optimization. *Journal of Machine Learning Research*, 21(110):1–48, 2020.

[103] T. Pham Dinh and H. A. Le Thi. Convex analysis approach to DC programming: theory, algorithms and applications. *Acta mathematica vietnamica*, 22(1):289–355, 1997.

[104] T. Pham Dinh and H. A. Le Thi. Recent advances in DC programming and DCA. *Transactions on computational intelligence*, 8342:1–37, 2014.

[105] D. N. Phan, H. A. Le Thi, and T. Pham Dinh. Sparse covariance matrix estimation by dca-based algorithms. *Neural computation*, 29(11):3040–3077, 2017.

[106] Y. Qian, R. Yan, and R. X. Gao. A multi-time scale approach to remaining useful life prediction in rolling bearing. *Mechanical Systems and Signal Processing*, 83:549–567, 2017.

[107] E. Ramasso, M. Rombaut, and N. Zerhouni. Joint prediction of continuous and discrete states in time-series based on belief functions. *IEEE transactions on cybernetics*, 43(1):37–50, 2012.

[108] M. Razaviyayn, M. Sanjabi, and Z.-Q. Luo. A stochastic successive minimization method for nonsmooth nonconvex optimization with applications to transceiver design in wireless communication networks. *Mathematical Programming*, 157(2):515–545, 2016.

[109] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[110] H. Robbins and D. Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing methods in statistics*, pages 233–257. Elsevier, 1971.

[111] G. O. Roberts and R. L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

[112] R. T. Rockafellar. *Convex analysis*. Princeton university press, 1970.

[113] R. T. Rockafellar. *The theory of subgradients and its applications to problems of optimization: Convex and nonconvex functions*. Berlin Heldermann., 1981.

[114] R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

[115] B. Saha and K. Goebel. Battery data set. *NASA AMES prognostics data repository*, 2007.

[116] G. Sateesh Babu, P. Zhao, and X.-L. Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International conference on database systems for advanced applications*, pages 214–228. Springer, 2016.

[117] A. Saxena and K. Goebel. Turbofan engine degradation simulation data set. *NASA Ames Prognostics Data Repository*, pages 1551–3203, 2008.

[118] A. Saxena, K. Goebel, D. Simon, and N. Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, pages 1–9. IEEE, 2008.

[119] M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[120] S. Shalev-Shwartz. Sdca without duality, regularization, and individual convexity. In *International Conference on Machine Learning*, pages 747–754. PMLR, 2016.

[121] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(2), 2013.

[122] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[123] E. M. Stuve. Estimating and plotting logarithmic error bars, 2004.

[124] T. Sun, Y. Sun, and W. Yin. On markov chain gradient descent. *Advances in neural information processing systems*, 31, 2018.

[125] L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, pages 1701–1728, 1994.

[126] A. V. Virkar. A model for degradation of electrochemical devices based on linear non-equilibrium thermodynamics and its application to lithium ion batteries. *Journal of Power Sources*, 196(14):5970–5984, 2011.

[127] K. Wang and P. Zhong. Robust non-convex least squares loss function for regression with outliers. *Knowledge-Based Systems*, 71:290–302, 2014.

[128] Y. Wang, D. Yang, X. Zhang, and Z. Chen. Probability based remaining capacity estimation using data-driven and neural network model. *Journal of Power Sources*, 315:199–208, 2016.

[129] Z. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh. Spiderboost and momentum: Faster variance reduction algorithms. *Advances in Neural Information Processing Systems*, 32, 2019.

[130] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[131] P. R. Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

[132] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu. Remaining useful life estimation of engineered systems using vanilla lstm neural networks. *Neurocomputing*, 275:167–179, 2018.

[133] Y. Xu, R. Jin, and T. Yang. Non-asymptotic analysis of stochastic methods for non-smooth non-convex regularized problems. *Advances in Neural Information Processing Systems*, 32, 2019.

[134] Y. Xu, Q. Qi, Q. Lin, R. Jin, and T. Yang. Stochastic optimization for dc functions and non-smooth non-convex regularizers with non-asymptotic convergence, 2019.

[135] Y. Yang, G. Scutari, D. P. Palomar, and M. Pesavento. A parallel decomposition method for non-convex stochastic multi-agent optimization problems. *IEEE Transactions on Signal Processing*, 64(11):2949–2964, 2016.

[136] C. Ye and Y. Cui. Stochastic successive convex approximation for general stochastic optimization problems. *IEEE Wireless Communications Letters.*, 9(6):755–759, 2019.

[137] C.-H. Zhang et al. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942, 2010.

[138] J. Zhang, P. Wang, R. Yan, and R. X. Gao. Long short-term memory for machine remaining life prediction. *Journal of manufacturing systems*, 48:78–86, 2018.

[139] T. Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11(3), 2010.

[140] L. Zheng, L. Zhang, J. Zhu, G. Wang, and J. Jiang. Co-estimation of state-of-charge, capacity and resistance for lithium-ion batteries based on a high-fidelity electrochemical model. *Applied Energy*, 180:424–434, 2016.

[141] S. Zhou and W. Zhou. Unified svm algorithm based on ls-dc loss. *Machine Learning*, pages 1–28, 2021.

[142] Y. Zhou, M. Huang, Y. Chen, and Y. Tao. A novel health indicator for on-line lithium-ion batteries remaining useful life prediction. *Journal of Power Sources*, 321:1–10, 2016.

Résumé

L'optimisation stochastique revêt une importance majeure à l'ère du big data et de l'intelligence artificielle. Ceci est attribué à la prévalence de l'aléatoire/de l'incertitude ainsi qu'à la disponibilité toujours croissante des données, deux facteurs qui rendent l'approche déterministe infaisable. Cette thèse étudie l'optimisation stochastique non convexe et vise à résoudre les défis du monde réel, notamment l'extensibilité, variance élevée, l'incertitude endogène et le bruit corrélé. Le thème principal de la thèse est de concevoir et d'analyser de nouveaux algorithmes stochastiques basés sur la programmation DC (différence de fonctions convexes) et DCA (algorithme DC) pour répondre aux nouvelles problématiques émergeant dans l'apprentissage automatique, en particulier l'apprentissage profond. Comme application industrielle, nous appliquons les méthodes proposées à la maintenance prédictive où le problème central est essentiellement un problème de prévision de séries temporelles.

La thèse se compose de six chapitres. Les préliminaires sur la programmation DC et le DCA sont présentés dans le chapitre 1. Le chapitre 2 étudie une classe de programmes DC dont les fonctions objectives contiennent une structure de somme importante. Nous proposons deux nouveaux schémas DCA stochastiques, DCA-SVRG et DCA-SAGA, qui combinent des techniques de réduction de la variance et étudient deux stratégies d'échantillonnage (avec et sans remplacement). La convergence presque sûre des algorithmes proposés vers les points critiques DC est établie, et la complexité des méthodes est examinée. Le chapitre 3 étudie les programmes DC stochastiques généraux (la distribution de la variable aléatoire associée est arbitraire) où un flux d'échantillons i.i.d. (indépendants et identiquement distribués) de la distribution intéressée est disponible. Nous concevons des schémas DCA stochastiques dans le cadre en ligne pour résoudre directement ce problème d'apprentissage théorique. Le chapitre 4 considère une classe de programmes DC stochastiques où l'incertitude endogène est en jeu et où les échantillons i.i.d. ne sont pas disponibles. Au lieu de cela, nous supposons que seules les chaînes de Markov qui sont ergodiques assez rapidement vers les distributions cibles peuvent être accédées. Nous concevons ensuite un algorithme stochastique appelé DCA stochastique à chaînes de Markov (MCSDCA) et fournissons une analyse de convergence dans les sens asymptotique et non asymptotique. La méthode proposée est ensuite appliquée à l'apprentissage profond via la régularisation des EDP (équations différentielles partielles), ce qui donne deux réalisations de MCSDCA, MCSDCA-odLD et MCSDCA-udLD, respectivement, basées sur la dynamique de Langevin suramortie et sous-amortie. Les applications de maintenance prédictive sont abordées au chapitre 5. La prédiction de la durée de vie utile restante (RUL) et l'estimation de la capacité sont deux problèmes centraux étudiés, qui peuvent tous deux être formulés comme des problèmes de prédiction de séries temporelles utilisant l'approche guidée par les données. Les modèles MCSDCA-odLD et MCSDCA-udLD établis au chapitre 4 sont utilisés pour former ces modèles à l'aide de réseaux neuronaux profonds appropriés. En comparaison avec divers optimiseurs de base en apprentissage profond, les études numériques montrent que les deux techniques sont supérieures, et les résultats de prédiction correspondent presque aux vraies valeurs de RUL/capacité. Enfin, le chapitre 6 met un terme à la thèse.

Mots-clés: Maintenance Prédictive, Apprentissage profond, Programmation DC et DCA

Abstract

Stochastic optimization is of major importance in the age of big data and artificial intelligence. This is attributed to the prevalence of randomness/uncertainty as well as the ever-growing availability of data, both of which render the deterministic approach infeasible. This thesis studies nonconvex stochastic optimization and aims at resolving real-world challenges, including scalability, high variance, endogenous uncertainty, and correlated noise. The main theme of the thesis is to design and analyze novel stochastic algorithms based on DC (difference-of-convex functions) programming and DCA (DC algorithm) to meet new issues emerging in machine learning, particularly deep learning. As an industrial application, we apply the proposed methods to predictive maintenance where the core problem is essentially a time series forecasting problem.

The thesis consists of six chapters. Preliminaries on DC programming and DCA are presented in Chapter 1. Chapter 2 studies a class of DC programs whose objective functions contain a large-sum structure. We propose two new stochastic DCA schemes, DCA-SVRG and DCA-SAGA, that combine variance reduction techniques and investigate two sampling strategies (with and without replacement). The proposed algorithms' almost sure convergence to DC critical points is established, and the methods' complexity is examined. Chapter 3 studies general stochastic DC programs (the distribution of the associated random variable is arbitrary) where a stream of i.i.d. (independent and identically distributed) samples from the interested distribution is available. We design stochastic DCA schemes in the online setting to directly solve this theoretical learning problem. Chapter 4 considers a class of stochastic DC programs where endogenous uncertainty is in play and i.i.d. samples are *unavailable*. Instead, we assume that only Markov chains that are ergodic fast enough to the target distributions can be accessed. We then design a stochastic algorithm termed Markov chain stochastic DCA (MCSDCA) and provide the convergence analysis in both asymptotic and nonasymptotic senses. The proposed method is then applied to deep learning via PDEs (partial differential equations) regularization, yielding two MCSDCA realizations, MCSDCA-odLD and MCSDCA-udLD, respectively, based on overdamped and underdamped Langevin dynamics. Predictive maintenance applications are discussed in Chapter 5. The remaining useful life (RUL) prediction and capacity estimation are two central problems being investigated, both of which may be framed as time series prediction problems using the data-driven approach. The MCSDCA-odLD and MCSDCA-udLD established in Chapter 4 are used to train these models using appropriate deep neural networks. In comparison to various baseline optimizers in deep learning, numerical studies show that the two techniques are superior, and the prediction results nearly match the true RUL/capacity values. Finally, Chapter 6 brings the thesis to a close.

Keywords: Predictive Maintenance, Deep Learning, DC (Difference of Convex functions) programming and DCA (DC Algorithms)