THE UNIVERSITY
of EDINBURGH

Integrating Local Information for Inference and Optimization in Machine Learning

Zhanxing Zhu

Doctor of Philosophy

Institute for Adaptive and Neural Computation
School of Informatics
University of Edinburgh

2016

Abstract

In practice, machine learners often care about two key issues: one is how to obtain a more accurate answer with limited data, and the other is how to handle large-scale data (often referred to as "Big Data" in industry) for efficient inference and optimization. One solution to the first issue might be aggregating learned predictions from diverse local models. For the second issue, integrating the information from subsets of the large-scale data is a proven way of achieving computation reduction. In this thesis, we have developed some novel frameworks and schemes to handle several scenarios in each of the two salient issues.

For aggregating diverse models – in particular, aggregating probabilistic predictions from different models – we introduce a spectrum of compositional methods, *Rényi divergence aggregators*, which are maximum entropy distributions subject to biases from individual models, with the Rényi divergence parameter dependent on the bias. Experiments are implemented on various simulated and real-world datasets to verify the findings. We also show the theoretical connections between Rényi divergence aggregators and machine learning markets with isoelastic utilities.

The second issue involves inference and optimization with large-scale data. We consider two important scenarios: one is optimizing large-scale *C*onvex-*C*oncave *S*addle *P*oint problem with a *Sep*arable structure, referred as *Sep-CCSP*; and the other is large-scale Bayesian posterior sampling.

Two different settings of Sep-CCSP problem are considered, Sep-CCSP with *strongly convex* functions and *non-strongly convex* functions. We develop efficient stochastic coordinate descent methods for both of the two cases, which allow fast parallel processing for large-scale data. Both theoretically and empirically, it is demonstrated that the developed methods perform comparably, or more often, better than state-of-the-art methods.

To handle the scalability issue in Bayesian posterior sampling, the stochastic approximation technique is employed, i.e., only touching a small mini batch of data items to approximate the full likelihood or its gradient. In order to deal with subsampling error introduced by stochastic approximation, we propose a covariance-controlled adaptive Langevin thermostat that can effectively dissipate parameter-dependent noise while maintaining a desired target distribution. This method achieves a substantial speedup over popular alternative schemes for large-scale machine learning applications.

Lay Summary

One of the fundamental tasks in science is to learn from and make predictions on the observed data. For example, given many images, one might try to predict what objects are in each image. This important task often cares about two issues: one is to how to obtain more accurate predictions given limited data; and the other is to how to handle large-scale data (referred as "Big Data" problem in industry) for efficient learning. One solution to the first issue might be that we develop various models for the same task and then integrate the predictions from these local models. For the second issue, recent works show that one can use much smaller subsets of the large-scale data to approximate the full computation. In this thesis, we have developed several novel frameworks and schemes to handle some aspects of the two important issues.

For aggregating learned predictions from various models, we propose a method to consider the situation when the individual models are learned from a biased version of the original data. And we implement extensive experiments on various simulated and real-world data to demonstrate the effectiveness of our method.

For handling learning problem with large-scale data, two scenarios are considered: inference and optimization. In both of the cases, we develop several methods which only use random (much smaller) subsets of the original large-scale data to approximate full computation. And thus, efficient learning can be done in a reasonable time, but not sacrificing much accuracy. This strategy solves the computational bottle neck involved in learning with large-scale data. Additionally, the developed methods also allow parallel processing, and thus enables the possibilities for usage of modern computing clusters.

Acknowledgements

First I would express my great thanks to my supervisor Amos J. Storkey, who has been providing consistent guidance and insights to my three years of Ph.D research. I feel lucky to be supervised by him. He always gave me the freedom to pursue my own ideas, but at the same time challenged me and steer me in the right direction when necessary. His friendly, approachable nature made a pleasant work environment for me, while his incredible knowledge and insight never cease to amaze me.

I wish to thank the following people for their assistance during the three years: Mingjun Zhong for valuable discussions on various interesting research problems; Jinli Hu and Xiaocheng Shang for close collaboration in some of my work; Peter Orchard, my office mate, for some helpful suggestions during the first two years; XingXing Zhang, for intensive discussions on neural networks, and Iain Murray, for the insightful discussion on the scalable methods for sampling.

Edinburgh machine learning group is an extraordinary place to work. I have been surrounded by many smart and motivated researchers, and have enjoyed lots of fresh and exciting ideas during brainstorming and paper discussion sessions. I also gratefully acknowledge the financial support from China Scholarships Council/University of Edinburgh Scholarships.

I have made a bunch of amazing friends in Edinburgh, allowing me to enjoy a colourful life in this beautiful city. I always remember the happy moments with them: Xin He, Cheng Feng, Guoli Yang, Jinli Hu, Benigno Uría, Konstantinos Georgatzis, Pol Moreno, Krzysztof Geras, He Wang and Yichuan Zhang, etc. Particularly, I would thank those joyful badminton matches with Xuan Huang, Sohan Seth, Boli Zhang and Xuri Tang.

Most endeavours in life begin with the support of family. I would never have been in a position to finish this Ph.D without the incredible people that brought me up, provided for me, educated, and inspired me throughout my life. My deepest gratitude goes to my parents, my older brother and my fiancée, Ming. Thanks so much for their love and endless supports during these years.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Zhanxing Zhu*)

Table of Contents

List of Figures

List of Tables

Notation

Vectors and Matrices

a	Scalars are written in plain typeface.
$\mathbf{a}$	Vectors are lowercase letters written in bold.
$\mathbf{A}$	Matrices are uppercase letters written in bold.
a_i	Element i of vector $\mathbf{a}$.
A_{ij}	Element (i, j) of matrix $\mathbf{A}$.
$\mathbf{A}_{i:}$	Row i of matrix $\mathbf{A}$.
$\mathbf{A}_{:j}$	Column j of matrix $\mathbf{A}$.
$\mathbf{a}_i$	The i-th in a sequence of vectors.
$\mu_{\max}(\mathbf{A})$	The maximum singular value of matrix $\mathbf{A}$.
$\|\mathbf{a}\|_2$	Euclidean norm (L_2-norm) of vector $\mathbf{a}$.
$\|\mathbf{A}\|_2$	Spectral norm of matrix $\mathbf{A}$.
$diag(\mathbf{a})$	A diagonal matrix with vector $\mathbf{a}$ as its diagonal elements.

Mathematical Symbols

$\mathbb{R}$	The real numbers.
$\mathbb{R}_+$	The non-negative real numbers.
$\mathbf{A} \succ 0$	$\mathbf{A}$ is positive-definite.
$\mathbf{A} \succeq 0$	$\mathbf{A}$ is positive-semidefinite.
$\mathbb{E}[\mathbf{a}]$	Expectation of random variable $\mathbf{a}$.
$\mathrm{Cov}[\mathbf{a}]$	Covariance matrix of random variable $\mathbf{a}$.

Conventions

$\mathbf{X}$	Data.
N	Number of data points.
$\mathbf{I}$	Identity matrix with its size clear from context.
$\tilde{A}$	Approximation of the quantity A.
$\langle \mathbf{a}, \mathbf{b} \rangle$	The inner product, $\mathbf{a}^T \mathbf{b}$.

Abbreviation Reference

The following table contains each abbreviation used in this thesis, along with a page number of either the first use, or on which more information can be found.

Abbreviation	Page	Meaning
AdaSPDC	50	Adaptive Stochastic Primal-Dual Coordinate Descent
ADMM	66	Alternating Direction Method of Multipliers
CCAdL	6, 93	Covariance-Controlled Adaptive Langevin Thermostat
CCSP	46	Convex-Concave Saddle Point problem
CD	49	Coordinate Descent methods
DRBM	102	Discriminative Restricted Boltzmann Machine
ERM	4	Empirical Risk Minimization
ESS	82	Effective Sample Size
FISTA	74	Fast Iterative Shrinkage Thresholdinng Algorithm
FOBOS	78	FOrward-Backward Splitting
GSADMM	70	Gauss-Seidel ADMM
HMC	84	Hamiltonian Monte Carlo
ISTA	74	Iterative Shrinkage Thresholding Algorithm
MALA	84	Metropolis Adjusted Langevin Algorithm
MCMC	5, 80	Markov Chain Monte Carlo
MH	82	Metropolis-Hasting algorithm
MLM	31	Machine Learning Markets
OSGA	77	Optimal SubGradient Algorithm
PDCP	48	Primal-Dual method of Chambolle and Pock (2011)
PDMM	70	Parallel Direction Method of Multipliers
RPCA	69	Robust Principal Component Analysis
SAG	55	Stochastic Averaging Gradient

Abbreviation	Page	Meaning
SDCA	55	Stochastic Dual Coordinate Descent
SDE	84	Stochastic Differential Equation
Sep-CCSP	4, 47	Separable Convex-Concave Saddle Point problem
SGD	4, 87	Stochastic Gradient Descent
SGHMC	6, 89	Stochastic Gradient Hamiltonian Monte Carlo
SGLD	6, 88	Stochastic Gradient Langevin Dynamics
SGNHT	6, 90	Stochastic Gradient Nosé-Hoover Thermostat
SP-BCD	50, 63	Stochastic Parallel Block Coordinate Descent
SPDC	49	Stochastic Primal-Dual Coordinate Descent

Chapter 1

Introduction

The fundamental task of machine learning is to learn from and make predictions on observed data. Two challenges are of significant concern of machine learners': one is how to achieve more accurate predictions given limited data; and the other is how to handle rich and/or large-scale data to facilitate efficient optimization and inference. For the first issue, a proven strategy for obtaining a more accurate prediction is aggregating the learned predictions from diverse local models. For the second issue, the computational bottleneck of dealing with large-scale data can be solved by integrating the information from its smaller subsets, which has been proven to be capable of reducing computational burden dramatically. The philosophy behind the solutions to the two challenges is to *integrate local information in a systematic way*, which forms the basic methodology utilized in this thesis. The core of the thesis is to develop novel frameworks and schemes to handle several machine learning problems in each of the two salient issues. In the following, we will introduce the two issues in more details to motivate the work that is described in the thesis.

Aggregating Probabilistic Predictions

Decisions and predictions resulting from aggregating information in large groups of agents or models are generally better than those made by isolated individuals. The intuition behind this is that different local models can distill different aspects of knowledge from the data. Aggregating the knowledge is a way of achieving more accurate predictions. This has been demonstrated in various environments of machine learning competitions, including the Netflix Challenge (Green, 2006), the PASCAL Visual Object Classes challenge (Everingham et al., 2006)), and many challenges in the Kag-

gle challenge environment (Goldbloom, 2010). Many workshops (e.g. KDD) also run a variety of machine learning challenges. One of the most consistent take-home messages from all the challenges is that aggregation of individual entries provides a performance benefit. The final winning Netflix submission was itself a large-scale aggregation of 107 different methods (Robert M. Bell and Volinsky, 2010).

In addition, as the complexity of various machine learning tasks increases, aggregation of predictions from different agents or algorithms is becoming increasingly necessary in distributed, large-scale or crowdsourcing systems. Much previous focus is on aggregation of classifiers or point predictions. However, aggregation of probabilistic predictions (beliefs) is also of particular importance, especially where quantification of risk matters, generative models are required or where probabilistic information is critical for downstream analyses. This motivates our work in Chapter 2 that focuses on aggregation of probability distributions (including conditional distributions).

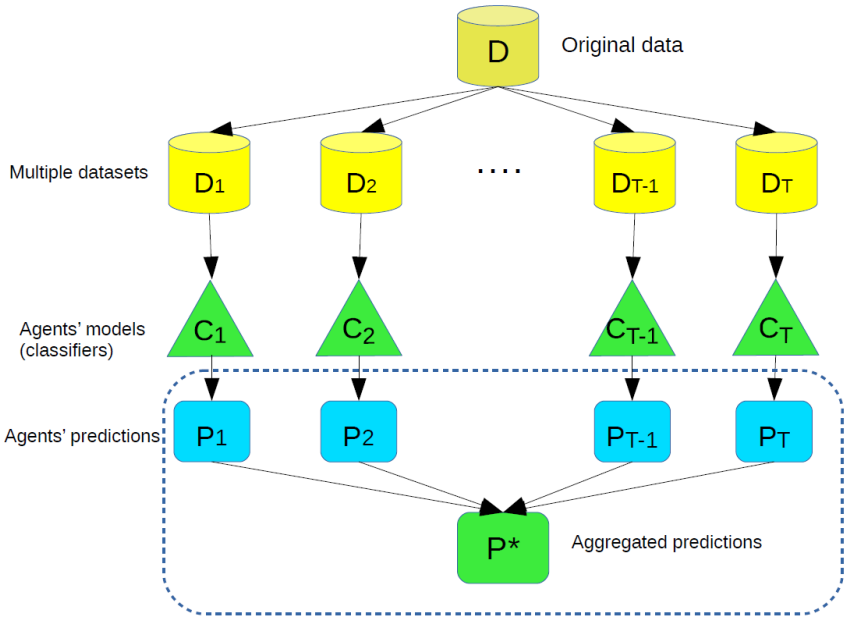


Figure 1.1: Sketch of aggregating probabilistic predictions form multiple agents. Individual agent obtains dataset D_i, which might be the exact copy of original data D or its biased version. Then, each agent builds its local model C_i to make (probabilistic) predictions P_i. The process inside the dashed rectangle represents the aggregation procedure to produce an aggregated prediction P^*. Note that we have no control over the individual agent's behaviour, which is favoured in distributed crowdsourcing systems.

Figure 1.1 provides a sketch of aggregating probabilistic predictions from multiple agents or algorithms. The data the agents observe is generated from a scenario that is the same as or similar (up to some bias) to the target scenario we care about.

A common case favoured in distributed crowdsourcing systems and many machine learning challenges is that there is no control over agents' behaviours, i.e., we do not control how they build their models and make their decisions or predictions. We wish to choose an aggregate distribution that has high log probability under data drawn from that target scenario.

Aggregating beliefs or probabilistic predictions has attracted significant attention both in the economics and the machine learning communities. In economics, aggregating beliefs from multiple agents is also referred as probabilistic opinion pooling (Dietrich and List, 2014), including two popular schemes: linear opinion pooling and logarithmic opinion pooling. Garg et al. (2004) generalized these pooling schemes into a divergence-based aggregation framework. However, the non-fully solved issue is that how to choose a particular type of divergence to accommodate different aggregation settings. To this end, in Chapter 2 we introduce a spectrum of compositional methods, *Rényi divergence aggregators*, that interpolate between log opinion pools and linear opinion pools. It is shown that these compositional methods are maximum entropy distributions for aggregating information from agents subject to individual biases, with the Rényi divergence parameter dependent on the bias. In the limit of no bias this reduces to the optimal limit of log opinion pools.

Another recent approach for aggregating probabilistic predictions uses information markets (Pennock and Wellman, 1997; Lay and Barbu, 2010; Storkey, 2011; Storkey et al., 2012) as an aggregation mechanism via the market price. In a machine learning market (one type of information markets), agents make utility maximizing decisions regarding trades in securities. These securities are tied to the random variables of the machine learning problem. For example they could be Arrow-Debreu securities defined on each possible predicted outcome. Given the trading desires of each agent, the equilibrium price in the market then defines a distribution that is an aggregation of the beliefs of different agents. Machine learning markets combine an incentivization mechanism (to ensure agents' actions reflect their beliefs) and a aggregation mechanism (via the trading process).

Despite the mechanism difference between traditional opinion pooling and information markets, it is interesting to investigate their theoretical connections. In Chapter 2, both theoretically and empirically, we show that Rényi divergence aggregators are directly implemented by machine learning markets with isoelastic utilities. The risk averseness of the isoelastic utility directly relates to the Rényi divergence parameter. This theoretical connection unifies the two streams of research efforts on aggregat-

ing probabilistic predictions, and provides a way of implementing Rényi divergence aggregators in incentivization market environments.

Large-scale Optimization and Inference

Data is booming exponentially nowadays, in both the quantity and dimension. The emergence of the "Big Data" era brings the computational challenge in many machine learning tasks, such as large-scale image classification, online recommendation offers in Amazon and Netflix with millions or even billions customers and products, and massive genomic data analysis, just to name a few. These motivate the developments of efficient methodologies for optimization and inference. This thesis is also dedicated to developing novel frameworks and schemes for handling large-scale optimization and inference.

The computational bottleneck with large-scale data often boils down to optimizing or evaluating the sum of a large number of separable functions (possibly with some constraints up to different models), where the separability occurs in terms of data points or variables we care about:

$$\sum_{i=1}^{N} f_i(\boldsymbol{\theta}; \mathbf{x}_i), \text{ when data points } \{\mathbf{x}_i\} \text{ are separable.}$$
$$\sum_{i=1}^{N} f_i(\boldsymbol{\theta}_i; \mathbf{X}), \text{ when variables } \{\boldsymbol{\theta}_i\} \text{ are separable.}$$

When N is large, say of millions or billions or even larger, iteratively optimizing or evaluating the sum is extremely costly. Historically, stochastic approximation has been a proven strategy used for computation reduction. The spirit of stochastic approximation is to only use much smaller subsets of the N terms to obtain an unbiased estimate of the original one and to integrate local approximation in a systematic way. Two representatives are stochastic gradient descent (SGD, Robbins and Monro (1951); Bottou (2010)) and stochastic coordinate descent methods (see Wright (2015) for a review).

In this thesis, we shall use the idea of stochastic approximation and develop novel approaches for several scenarios in machine learning, where the scalability issues are tamed successfully. We consider two important situations: one is optimizing large-scale Convex-Concave Saddle Point problem with a Separable structure, referred as *Sep-CCSP*; and the other is large-scale Bayesian posterior sampling.

Sep-CCSP solves a specific minmax problem, covering a wide range of important machine learning models, such as empirical risk minimization (ERM, Hastie et al.

(2009)) and linear constrained optimization (for instance, robust principal component analysis (Wright et al., 2009)). ERM can be formulated into Sep-CCSP by conjugate dual transformation of individual loss function per data item, while linear constrained optimization becomes Sep-CCSP by introducing Lagrangian multipliers for the linear constraints. Previous works (Chambolle and Pock, 2011; Zhang and Xiao, 2015) are either batch methods or not adaptive in choosing stepsizes. Chapter 3 aims to develop scalable methods for large-scale Sep-CCSP to facilitate efficient learning for these important machine learning models.

We consider Sep-CCSP with both strongly convex and general convex functions, respectively. In both cases, we design specific stochastic block coordinate descent methods with adaptively controlled stepsizes for large-scale Sep-CCSP, which achieve state-of-the-art convergence performance theoretically and empirically. In addition, these approaches are suitable for parallel processing, which allow the possibility of employing the power of modern computing clusters.

Another scenario we consider is large-scale Bayesian posterior sampling. Bayesian analysis gives us a simple recipe for learning from data: given a set of unknown parameters or latent variables $\boldsymbol{\theta}$ that are of interest, we specify a prior distribution $p(\boldsymbol{\theta})$ quantifying what we know about $\boldsymbol{\theta}$ before observing any data. Then we quantify how the observed data $\mathbf{X} = \{\mathbf{x}_i\}_{n=1}^N$ relates to $\boldsymbol{\theta}$ by specifying a likelihood function $p(\mathbf{X}|\boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{x}_i|\boldsymbol{\theta})$. Finally, we apply Bayes' rule to obtain the posterior distribution

$$p(\boldsymbol{\theta}|\mathbf{X}) = p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})/Z,$$

where Z is the normalization constant, $Z = \int p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$. Bayesian inference often involves computing the expectation of certain function with respect to the posterior distribution $p(\boldsymbol{\theta}|\mathbf{X})$. This integral over the parameter space distinguishes the Bayesian scheme of inference from other schemes based on optimization. This also endows Bayesian inference with the capability of avoiding overfitting. However, the integral over the (high-dimensional) parameter space rarely has analytical forms except for several simple prior and likelihood functions, e.g., conjugate families of distributions. Thus, we have to resort to some approximation, such as sampling from the posterior distribution and using the obtained samples to do Monte Carlo approximation of the integral.

Popular sampling procedures, such as Markov Chain Monte Carlo (MCMC, Neal (1993); Robert and Casella (2013)) techniques, have to evaluate the full likelihood or its gradient in each iteration. However, sampling easily becomes infeasible with

large-scale data, hence hindering its applicability.

In order to improve computational efficiency, a number of stochastic gradient methods (Welling and Teh, 2011; Chen et al., 2014; Ding et al., 2014) have been proposed in the setting of Bayesian sampling based on random (and much smaller) subsets to approximate the likelihood of the whole dataset, thus substantially reducing the computational cost in practice. Welling and Teh (2011) proposed the so-called Stochastic Gradient Langevin Dynamics (SGLD), combining the ideas of stochastic optimization (Robbins and Monro, 1951) and traditional Brownian dynamics, with a sequence of stepsizes decreasing to zero. SGLD generates samples from first order Brownian dynamics, and thus, with a fixed stepsize, one can show that it is unable to dissipate excess noise in gradient approximations while maintaining the desired invariant distribution (Chen et al., 2014). A Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) method was proposed by Chen et al. (2014), which relies on second order Langevin dynamics and incorporates a parameter-dependent diffusion matrix that is intended to effectively offset the stochastic perturbation of the gradient. However, it is difficult to accommodate the additional diffusion term in practice. Moreover, as pointed out in Ding et al. (2014) poor estimation of it may have a significant adverse influence on the sampling of the target distribution. Chapter 4 provides a comprehensive review over all these methods.

The "thermostat" idea, which is widely used in molecular dynamics (Frenkel and Smit, 2001; Leimkuhler and Matthews, 2015), was recently adopted in the Stochastic Gradient Nosé-Hoover Thermostat (SGNHT) by Ding et al. (2014) in order to adjust the kinetic energy during simulation in such a way that the canonical ensemble is preserved (i.e. so that a prescribed constant temperature distribution is maintained). However, SGNHT methods are designed based on the assumption of constant noise variance, which cannot handle general cases with non-constant noise. Chapter 4 aims to solve this critical issue, and we propose a Covariance-Controlled Adaptive Langevin thermostat (*CCAdL*), that can handle parameter-dependent noise, improving both robustness and reliability in practice, and which can effectively speed up the convergence to the desired invariant distribution in large-scale machine learning applications.

The work done in this thesis leverages with the approach of integrating local information in a systematic way, where two issues in machine learning are solved: aggregating probabilistic prediction, and large-scale optimization and inference. Employing this philosophy, we have developed approaches to tackle these challenging tasks. The thesis is structured as follows. Each chapter begins with an introduction and a review

of the background literature. In Chapter 2, we propose a spectrum of compositional methods, *Rényi divergence aggregators*, that interpolate between log opinion pools and linear opinion pools. And we show that this aggregator is directly implemented by machine learning markets with isoelastic utilities. We run various experiments to verify these findings. In Chapter 3, we introduce the Sep-CCSP problem, and show that it covers a wide range of important machine learning models, where both strongly convex and general convex functions are considered. We then develop efficient stochastic block coordinate descent methods for solving large-scale Sep-CCSP problem. Extensive applications and experiments are taken into consideration to demonstrate its effectiveness. In Chapter 4, we first review dynamics-based MCMC methods and their variants with stochastic gradients, and pointed out their advantage and shortcomings. Then we propose our Covariance-Controlled Adaptive Langevin thermostat (*CCAdL*) to handle the issue of non-constant noise variance introduced by noisy gradients. We conduct various experiments to show its superior performance compared with other state-of-the-art approaches. Finally, Chapter 5 summarizes the contributions of the thesis, where some potential research directions are also discussed.

Chapter 2

Aggregation of Probabilistic Predictions Under Bias

In this chapter, we focus on how to aggregate beliefs from multiple agents, i.e, learned probabilistic predictions from different models in machine learning context. In particular, the emphasis is aggregating probabilistic predictions on classification problems. Recently, trading in information markets, such as machine learning markets, has been shown to be an effective approach for aggregating the beliefs of different agents. In a machine learning context, aggregation commonly uses forms of linear opinion pools, or logarithmic (log) opinion pools. It is interesting to relate information market aggregation to the machine learning setting.

We introduce a spectrum of compositional methods, *Rényi divergence aggregators*, that interpolate between log opinion pools and linear opinion pools. It is shown that these compositional methods are maximum entropy distributions for aggregating information from agents subject to individual biases, with the Rényi divergence parameter dependent on the bias. In the limit of no bias this reduces to the optimal limit of log opinion pools. This relationship is demonstrated practically on both simulated and real datasets.

We then return to information markets and show that Rényi divergence aggregators are directly implemented by machine learning markets with isoelastic utilities, and so can result from autonomous self interested decision making by individuals contributing different predictors. The risk averseness of the isoelastic utility directly relates to the Rényi divergence parameter, and hence encodes how much an agent believes (s)he may be subject to an individual bias that could affect the trading outcome: if an agent believes (s)he might be acting on significantly biased information, a more risk averse

isoelastic utility is warranted.

This chapter is an extended work based on the following published paper, where AJS built the most of the theoretical framework; JH and ZZ contributed to the remaining theoretical aspects; and ZZ and AJS implemented all the experiments and analysis.

Storkey, A.J., Zhu, Z. and Hu, J.(2015). Aggregation Under Bias: Renyi Divergence Aggregation and its Implementation via Machine Learning Markets. In *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pages 560-574.

2.1 Motivation

Aggregation of predictions from different agents or algorithms is becoming increasingly necessary in distributed, large scale or crowdsourced systems. Much previous focus is on aggregation of classifiers or point predictions. However, aggregation of probabilistic predictions is also of particular importance, especially where quantification of risk matters, generative models are required or where probabilistic information is critical for downstream analyses. In this chapter we focus on aggregation of probability distributions (including conditional distributions).

The problem of probabilistic aggregation in machine learning can be cast as choosing a single aggregate distribution given no (or little) direct data, but given instead the beliefs of a number of independent agents. We have no control over what these agents do, other than that we know they *do* have direct access to data and we expect them to have obtained their beliefs using that data. The data the agents observe is generated from a scenario that is the same as or similar to the target scenario we care about. We wish to choose an aggregate distribution that has high log probability under data drawn from that target scenario.

One recent approach for aggregating probabilistic machine learning predictions uses information markets (Pennock and Wellman, 1997; Lay and Barbu, 2010; Storkey, 2011; Storkey et al., 2012) as an aggregation mechanism via the market price. In a machine learning market, agents make utility maximizing decisions regarding trades in securities. These securities are tied to the random variables of the machine learning problem. For example they could be Arrow-Debreu securities[1] defined on each possible predicted outcome. Given the trading desires of each agent, the equilibrium price in the market then defines a distribution that is an aggregation of the beliefs of different

[1] A canonical ArrowDebreu security is a security that pays one unit of numeraire if a predicted outcome is reached and zero otherwise.

agents. Machine learning markets combine an incentivization mechanism (to ensure agents' actions reflect their beliefs P_i) and a aggregation mechanism (via the trading process).

Understanding the relationship between individual actions and the aggregate market price is an interesting open question for information markets. In addition, finding efficient methods of arriving at market equilibria is key to their practical success. The main novel contributions of this chapter are

- Introducing the class of Rényi divergence based aggregators which interpolate between linear opinion pools and log opinion pools, and showing that they are the maximum entropy estimators for aggregation of beliefs potentially subject to bias. We also demonstrate this relationship practically via simulated and real problems.

- Directly relating Rényi divergence aggregators to machine learning markets with different isoelastic utilities, and showing that the risk averseness of the isoelastic utility relates to the Rényi divergence parameter that is used to control the assumed bias.

2.2 Background

Aggregation methods have been studied for some time, and have been discussed in a number of contexts. Aggregation methods differ from ensemble approaches (see e.g. (Dietterich, 2000)), as the latter also involves some control over the form of the individuals within the ensemble: with aggregation, the focus is entirely on the method of combination - there is no control over the individual agent beliefs. In addition, most aggregation methods focus on aggregating hard predictions (classifications, mean predictive values etc.) (Breiman, 1996; Domingos, 1997). Some, but not all of those are suitable for aggregation of probabilistic predictions (Dani et al., 2006; Ottaviani and Sørensen, 2007), where full predictive distributions are given. This issue has received significant attention in the context of aggregating Bayesian or probabilistic beliefs (West, 1984; Dietrich, 2010; Maynard-Reid and Chajewska, 2001; Pennock and Wellman, 1997; Storkey, 2011). Full predictive distributions are generally useful for a Bayesian analysis (where the expected loss function is computed from the posterior predictive distribution), in situations where full risk computations must be done, or simply to get the most information from the individual algorithms. Wolpert (1992)

describes a general framework for aggregation, where an aggregator is trained using the individual predictions on a held out validation set as inputs, and the true validation targets as outputs. This requires specification of the aggregation function. The work in this paper fits within this framework, with Rényi mixtures as the aggregator. In crowdsourcing settings, issues of reliability in different contexts come into play. Log opinion pools have been generalized to weighted log opinion pools using Bayesian approaches with an event-specific prior (Kahn, 2004). This emphasises that expert models can work with aggregators at many different levels, from individual data points to whole datasets within a corpus.

Recently, prediction markets, and methods derived from securities market settings (Storkey, 2011; Storkey et al., 2012; Lay and Barbu, 2010; Barbu and Lay, 2011; Pennock and Wellman, 1997; Dani et al., 2006; Chen and Wortman Vaughan, 2010), have provided a particular foundation for belief aggregation. That securities markets can perform belief aggregation was first discussed by Rubinstein (1974, 1975, 1976). Belief aggregation of this form is of importance in crowdsourcing settings, or settings combining information from different autonomous agents. In such settings, the beliefs of different agents can be subject to various biases.

One other area that aggregation has shown importance is in machine learning competitions, including the Netflix Challenge (Green, 2006), the PASCAL Visual Object Classes challenge (Everingham et al., 2006)), and many challenges set in the Kaggle challenge environment (Goldbloom, 2010). Many workshops (e.g. KDD) also run a variety of machine learning challenges. One of the most consistent take-home messages from all the challenges is that aggregation of individual entries provides a performance benefit. The final winning Netflix submission was itself a large scale aggregation of 107 different methods (Robert M. Bell and Volinsky, 2010).

In this study, we consider the most prominent aggregation methods that are relevant to a probabilistic aggregation setting. These vary from the most basic (simple averaging) to more complicated and computationally demanding. We organize these aggregation methods into two categories:

Simple Methods that require little or no optimization;

Learnt Methods where aggregation parameters such as weights are learnt on a validation dataset.

2.2.1 Simple Aggregation Methods

There are several types of simple belief aggregation methods.

2.2.1.1 Simple averaging

Simple averaging uses the mean of the beliefs from each model as the aggregate distribution. Thus, the aggregate distribution has the form

$$P(y|\mathbf{x}) = \frac{1}{N_A} \sum_{j=1}^{N_A} P_j(y|\mathbf{x}), \tag{2.1}$$

where N_A is the number of models to be aggregated, and $P_j(y|\mathbf{x})$ is predictive probability from jth model or agent on yth class (y is a discrete value to denote the class membership for classification problems), given the relevant covariate information $\mathbf{x}$.

2.2.1.2 Bayesian model averaging

Bayesian model averaging has been discussed in the context of aggregation methods (Dietterich, 2000; Domingos, 1997), but it is well known that it is inappropriate to use it as one (Minka, 2002). Bayesian model averaging is best thought of as a method for 'soft model selection'. It answers the question: "Given that all of the data so far was generated by exactly one of the hypotheses, what is the probability of observing the new pair $(y, \mathbf{x})$?" The soft weights in BMA only reflect a statistical inability to distinguish the hypothesis based on limited data. As more data arrives, the hypotheses become more distinguishable and Bayesian model averaging will always focus its weight on the most probable hypothesis, just as the posterior for the mean of a Gaussian focuses ever more narrowly on the sample mean. Invariably in large data settings, this results in a single agent having all the weight, and so it barely differs from the highest ranked individual model/agent. Bayesian model averaging uses

$$P(y|\mathbf{x}) \propto \sum_{j=1}^{N_A} P_j(y|\mathbf{x}, \mathcal{D}_{\text{tr}}) P(j), \tag{2.2}$$

where $\mathcal{D}_{\text{tr}}$ denotes the training data for training each classification model, and $P(j)$ is the model evidence.

2.2.1.3 Bayesian model averaging with power heuristics

Though Bayesian model averaging is inappropriate if we believe that all approaches are potential contributors to the final result, it is undoubtedly true that the likelihood

of the model does provide *some* useful evidence for the relative benefits of different predictors. One simple way of incorporating that information is a heuristic that uses an annealed likelihood as a weighting, and chooses an appropriate annealing power. Effectively this is a modified version of Bayesian model averaging in which the posterior weighting term is raised to some power $\alpha > 0$

$$P(y|\mathbf{x}) \propto \sum_{j=1}^{N_A} P_j(y|\mathbf{x}, \mathcal{D}_{\text{tr}}) \left(P(j)\right)^{\alpha}, \tag{2.3}$$

where we ignore the normalization constant in the R.H.S of the above expression.

2.2.2 Learnt Aggregation Methods

2.2.2.1 Linear Opinion Pool

Linear opinion pool combines the beliefs by a weighted arithmetic average,

$$\begin{aligned} P(y|\mathbf{x}) &= \sum_{j=1}^{N_A} w_j P_j(y|\mathbf{x}) \\ \text{s.t.} \quad & w_j \geq 0, \forall j; \quad \sum_{j=1}^{N_A} w_j = 1. \end{aligned} \tag{2.4}$$

The weight vector $\mathbf{w}$ can be solved by maximizing the log likelihood with simplex constraints, or alternatively via an expectation maximization procedure. By convexity, the solution of both approaches is equivalent.

2.2.2.2 Logarithmic Opinion Pool

Logarithmic opinion pool aggregates the beliefs by a weighted geometric average,

$$\begin{aligned} P(y|\mathbf{x}) &= \frac{1}{Z(\mathbf{w})} \prod_{j=1}^{N_A} P(y|\mathbf{x})^{w_j} \\ \text{s.t.} \quad & w_j \geq 0, \forall j; \quad \sum_{j=1}^{N_A} w_j = 1. \end{aligned} \tag{2.5}$$

where $Z(\mathbf{w}) = \sum_y \prod_{j=1}^{N_A} P(y|\mathbf{x})^{w_j}$ is the normalization constant. The logarithmic opinion pool is more problematic to work with due to the required computation of the normalization constant. However for a discrete space of y (i.e., $y \in \mathbf{Z}_+$, often denoting the class membership) this normalization constant can be computed, and hence an exact gradient of log likelihood with respect to the weights $\mathbf{w}$ can be found. Hence $\mathbf{w}$ can

be obtained using a standard gradient-based optimizer. Others (e.g. (Heskes, 1998)) have used various approximate schemes for log opinion pools. We focus on the exact setting here.

2.3 Problem Statement

The problem setting is as follows. We have a prediction problem to solve, in common with a number of agents. These agents have learnt probabilistic predictors on each of their own training datasets, using their own machine learning algorithms, and provide the predictions for the test scenario. We wish to combine the agents' predictions to make the best prediction we can for our setting. We don't have access to the training data the agents see, but are potentially given the held out performance of each agent on their training data, and we may have access to their predictions for a small validation set of our own data which we know relates to our domain of interest (the distribution of which we denote by P^G). We consider the case where it may be possible that the data individual agents see are different in distribution (i.e. biased) with respect to our domain of interest.

Our objective is to minimize the negative log likelihood for a model P for future data generated from an unknown data generating distribution P^G. This can be written as desiring $\arg\min_P KL(P^G||P)$, where KL denotes the Kullback-Leibler divergence[2]. However in an aggregation scenario, we do not have direct access to data that can be used to choose a model P by a machine learning method. Instead we have access to beliefs P_i from $i = 1,2,\ldots,N_A$ other agents, which *do* have direct access to some data, and we must use those agent beliefs P_i to form our own belief P.

We have no control over the agents' beliefs P_i, but we can expect that the agents have learnt P_i using some learning algorithm with respect to data drawn from individual data distributions P_i^G. Hence agents will choose P_i with low $KL(P_i||P_i^G)$ with respect to their individual data (we choose $KL(P_i||P_i^G)$, not $KL(P_i^G||P_i)$, for derivation convenience later), drawn from P_i^G. For example agents can choose their own posterior distributions P_i with respect to the data they observe.

We also assume that each P_i^G is "close" to the distribution P^G we care about. Where we need to be specific, we use the measure $KL(P_i^G||P^G)$ as the measure of closeness,

[2]In probability theory and information theory, the KullbackLeibler divergence is a measure of the difference between two probability distributions P and Q, e.g., for discrete distributions, $KL(P||Q) = \sum_i P(i)\log\frac{P(i)}{Q(i)}$

which is appropriate if P_i^G is obtained by sample selection bias (Storkey, 2009) from P^G. When $KL(P_i^G||P^G) = 0 \; \forall i$, it implies all the agents obtain the unbiased data samples from generating distribution P^G.

2.4 Weighted Divergence Aggregation

Weighted divergence-based aggregation was proposed in Garg et al. (2004). The idea was, given individual distributions P_i, to choose an aggregate distribution P given by

$$P = \arg\min_Q \sum_i w_i D(P_i, Q), \tag{2.6}$$

where w_i is a positive weight, $\sum_i w_i = 1$, and $D(P_i, Q)$ represents a choice of divergence between P_i and Q, where $D(A,B) \geq 0$, with equality iff $A = B$. This framework generalizes several popular opinion pooling methods, e.g., linear opinion pooling when $D(P_i, Q) = KL(P_i||Q)$, and log opinion pooling when $D(P_i, Q) = KL(Q||P_i)$, see Garg et al. (2004).

Weighted divergence aggregation is very general but we need to choose a particular form of divergence. In this study we analyse the family of Rényi divergences for weighted divergence aggregation. This choice is motivated by two facts:

- Rényi divergence aggregator is the maximum entropy aggregating distribution when the individual agent distributions are biased due to a sample selection mechanism.
- Rényi divergence aggregators are implemented by machine learning markets, and hence can result from autonomous self interested decision making by the individuals contributing different predictors without centralized imposition. Hence this approach can *incentivize* agents to provide their best information for aggregation.

In much of the analysis that follows we will drop the conditioning (i.e. write $P(y)$ rather than $P(y|\mathbf{x})$) for the sake of clarity, but without loss of generality as all results follow through in the conditional setting.

2.4.1 Weighted Rényi Divergence Aggregation

Here we introduce the family of weighted Rényi divergence methods.

Definition 1 (Rényi Divergence with $0 < \gamma < 1$)**.** *Let* y *be a random variable taking values* $y = 1, 2, \ldots, K$*. The Rényi divergence of order* γ *(*$0 < \gamma < 1$*) from a distribution* P *to a distribution* Q *is defined as*

$$D_\gamma^R[P||Q] = \frac{1}{\gamma - 1} \log \left(\sum_{y=1}^{K} P(y)^\gamma Q(y)^{1-\gamma} \right). \tag{2.7}$$

The Rényi divergence has two relevant special cases: $\lim_{\gamma \to 1}(1/\gamma) D_\gamma^R(P||Q) = KL(P||Q)$, and $\lim_{\gamma \to 0}(1/\gamma) D_\gamma^R(P||Q) = KL(Q||P)$ (which can be seen via L'hôpital's rule). We assume the value for the Rényi divergence for $\gamma = 1$ is defined by $KL(P||Q)$ via analytical continuation.

Definition 2 (Weighted Rényi Divergence Aggregation)**.** *The* weighted Rényi divergence aggregation *is a weighted divergence aggregation given by (2.6), where each divergence* $D(P_i, Q) = \gamma_i^{-1} D_{\gamma_i}^R[P_i||Q]$.

Note that each component i in (2.6) can have a Rényi divergence with an individualized parameter γ_i. Sometimes we will assume that all divergences are the same, and refer to a single $\gamma = \gamma_i \ \forall i$ used by all the components.

2.4.1.1 Properties

The following propositions outline some properties of weighted Rényi divergence aggregation.

Proposition 1. *Weighted Rényi divergence aggregation satisfies the implicit equation for* $P(y)$ *of*

$$P(y) = \frac{1}{Z} \sum_i w_i \gamma_i^{-1} \frac{P_i(y)^{\gamma_i} P(y)^{1-\gamma_i}}{\sum_{y'} P_i(y')^{\gamma_i} P(y')^{1-\gamma_i}}, \ \textit{s.t.} \ P(y) \geq 0, \forall y. \tag{2.8}$$

where $0 < \gamma_i < 1$*,* w_i *are given non-negative weights, and* $Z = Z(\{\gamma_i\}) = \sum_i w_i \gamma_i^{-1}$ *is a normalisation constant, and* $\{\gamma_i\}$ *is the set of Rényi divergence parameters.*

Proof. Using $D(P_i, Q) = \gamma_i^{-1} D_{\gamma_i}^R[P_i||Q]$ from (2.7) in Eq. (2.6), we need to solve the following constrained optimization problem,

$$\min_Q h(Q) = \sum_i \frac{w_i}{\gamma_i(\gamma_i - 1)} \log \left(\sum_{y=1}^{K} P_i(y)^{\gamma_i} Q(y)^{1-\gamma_i} \right)$$
$$\text{s.t.} \sum_{y=1}^{K} Q(y) = 1, \text{ and } Q(y) \geq 0, \forall y.$$

Firstly, we show that the function $h(Q)$ is a convex function so that Lagrangian multipliers can be applied. It is obvious that the function $g(Q) = \sum_{y=1}^{K} P(y)^{\gamma_i} Q(y)^{1-\gamma_i}$ is concave since each component function is concave. And the following function

$$f_i(x) = \frac{w_i}{\gamma_i(\gamma_i - 1)} \log(x), \quad 0 < \gamma_i < 1$$

is convex. The composition of the two functions is convex,

$$h_i(Q) = f_i(g(Q)). \tag{2.9}$$

Then the objective function $h(Q) = \sum_i h_i(Q)$ is a convex function.

Now introducing Lagrangian multiplier Z to handle the first constraint, we have

$$L(Q,Z) = \sum_i \frac{w_i}{\gamma_i(\gamma_i - 1)} \log \left(\sum_{y=1}^{K} P(y)^{\gamma_i} Q(y)^{1-\gamma_i} \right) + Z \left(\sum_{y=1}^{K} Q(y) - 1 \right) \tag{2.10}$$

Employing calculus of variations w.r.t. $Q(y)$ and setting the derivatives as zeroes for the optimal values $P(y)$, we can obtain K equations

$$\sum_i w_i \gamma_i^{-1} \frac{P_i(y)^{\gamma_i} P(y)^{-\gamma_i}}{\sum_{y'=1}^{K} P_i(y')^{\gamma_i} P(y')^{1-\gamma_i}} - Z = 0, \forall y = 1, \ldots, K. \tag{2.11}$$

Multiplying each equation with $P(y)$, we can easily find $Z = \sum_i w_i \gamma_i^{-1}$ by summing over all K equations. Then we insert Z back into Eq.(2.11) and obtain the final result Eq.(2.8). □

Proposition 2. *Weighted Rényi divergence aggregation interpolates between linear opinion pooling ($\gamma \to 1$) and log opinion pooling ($\gamma \to 0$).*

Proof. Set $\gamma_i = 1$ in Eq.(2.8) to obtain a standard linear opinion pool.

For log opinion pool, we firstly set all $\gamma_i = \gamma$, and we take $\gamma \to 0$. Using L'Hôspital's rule, we can easily obtain

$$\lim_{\gamma \to 0} D(P_i, Q) = \lim_{\gamma \to 0} \gamma_i^{-1} D_{\gamma_i}^{R}[P_i || Q] = \sum_y Q(y) \log \frac{Q(y)}{P_i(y)} = KL(Q \| P_i). \tag{2.12}$$

Then the Lagrangian of Eq.(2.10) becomes the following form,

$$L(Q,Z) = \sum_i w_i \sum_y Q(y) \log \frac{Q(y)}{P_i(y)} + Z(\sum_y Q(y) - 1). \tag{2.13}$$

Applying calculus of variations w.r.t. $Q(y)$ and setting the derivatives as zeroes for the optimal values $P(y)$, we can obtain

$$\sum_i w_i \left(\log \frac{P(y)}{P_i(y)} + 1 \right) + Z = 0. \tag{2.14}$$

After rearrangement, we obtain the form of the log opinion pool,

$$P(y) = \frac{1}{Z'(\mathbf{w})} \prod_i P_i(y)^{w_i}, \tag{2.15}$$

where $Z'(\mathbf{w}) = \sum_y \prod_i P_i(y)^{w_i}$ is the normalization constant.

□

In the next section we show that Rényi divergence aggregation provides the maximum entropy distribution for combining together agent distributions where the belief of each agent is subject to a particular form of bias. Two consequences that are worth alerting the reader to ahead of that analysis are:

1. If all agents form beliefs on data drawn from the same (unbiased) distribution then the maximum entropy distribution is of the form of a log opinion pool.

2. If all agents form beliefs on unrelated data then the maximum entropy distribution is of the form of a linear opinion pool.

2.5 Maximum Entropy Arguments

Consider the problem of choosing an aggregator distribution P to model an unknown target distribution P^G given a number of individual distributions P_i. These individual distributions are assumed to be learnt from data by a number of individual agents. We will assume the individual agents did not (necessarily) have access to data drawn from P^G, but instead the data seen by the individual agents was biased, and instead sampled from distribution P_i^G. In aggregating the agent beliefs, we neither know the target distribution P^G, nor any of the individual bias distributions P_i^G, but model them with P and Q_i respectively. To clarify all the involved distributions, we list them in Table 2.1.

Table 2.1: The involved distributions in maximum entropy arguments.

Role	Truth	Sample	Learned
Agents	P^G	P_i^G	P_i
Aggregator's model	P	Q_i	-

As far as the individual agents are concerned they train and evaluate their methods on their individual data, unconcerned that their domains were biased with respect to the domain we care about. We can think of this scenario as convergent dataset shift

(Storkey, 2009), where there is a shift from the individual training to a common test scenario. The result is that we are given information regarding the test log likelihood performance for each P_i in their own domains: $\sum_y P_i^G(y) \log P_i(y) = a_i$.

The individual agent data is biased, not unrelated, and so we make the assumption that the individual distributions P_i^G are related to P^G in some way. We assume that $KL(P_i^G||P^G)$ is subject to some bound (and call this the nearness constraint). Given this scenario, we aim to find maximum entropy distributions Q_i to model P_i^G that capture the performance of the individual distributions P_i. At the same time, we could enforce additional constraints to this maximum entropy optimization by relating Q_i with P. As we know the test performance of individual agent's model, we write this as the constraints (These are linear constraints since P_i is known.):

$$\sum_y Q_i(y) \log P_i(y) = a_i, \tag{2.16}$$

Note that each agent i tries its best to optimize the quantity $\sum_y Q_i(y) \log P_i(y)$, which might restrict the space of $Q_i(y)$ considerably.

The nearness constraints[3] for Q_i are written as

$$KL(Q_i||P) \leq A_i \tag{2.17}$$

$$\Rightarrow \sum_y Q_i(y) \log \frac{Q_i(y)}{P(y)} \leq A_i \text{ for some } P\ . \tag{2.18}$$

encoding that our model Q_i for P_i^G must be near to the model P for P^G. That is the KL divergence between the two distributions must be bounded by some value A_i. This nearness also characterizes the settings we considered, i.e., individual agent only has access to a biased version of the original data distribution P^G.

Given these constraints, the final maximum entropy optimization problem becomes,

[3]We could work with a nearness penalty of the same form rather than a nearness constraint. The resulting maximum entropy solution would be of the same form.

$$\begin{aligned}
&\min_{\{Q_i\},P} \sum_i \sum_y Q_i(y) \log Q_i(y) \\
&\text{s.t. } \sum_y Q_i(y) \log P_i(y) = a_i \\
&\sum_y Q_i(y) \log \frac{Q_i(y)}{P(y)} \leq A_i \\
&\sum_y Q_i(y) = 1 \forall i \\
&\sum_y P(y) = 1 \\
&Q_i(y) \geq 0, \forall y, i \\
&P(y) \geq 0, \forall y.
\end{aligned} \tag{2.19}$$

Introducing Lagrangian multipliers for the first four constraints, we have

$$\begin{aligned}
L(\{Q_i\},P) = &\sum_i \sum_y Q_i(y) \log Q_i(y) + \sum_i b_i (1 - \sum_y Q_i(y)) \\
&- \sum_i \lambda_i \left(\left[\sum_y Q_i(y) \log P_i(y) \right] - a_i \right) + c(1 - \sum_y P(y)) \\
&+ \sum_i \rho_i \left(\left[\sum_y Q_i(y) \log \frac{Q_i(y)}{P(y)} \right] - A_i + s_i \right)
\end{aligned} \tag{2.20}$$

where s_i are slack variables $s_i \geq 0$, and ρ_i, λ_i, b_i and c are Lagrange multipliers. This minimization chooses maximum entropy Q_i, while ensuring there is a distribution P for which the nearness constraints are met.

Taking derivatives with respect to $Q_i(y)$ and setting to zero gives

$$Q_i(y) = \frac{1}{Z_i} P(y)^{\frac{\rho_i}{1+\rho_i}} P_i(y)^{\frac{\lambda_i}{1+\rho_i}}, \text{ s.t.} Q_i(y) \geq 0, \forall y, i \tag{2.21}$$

where Z_i is a normalization constant.

Given these Q_i, we can find also find an optimal, best fitting P. Taking derivatives of the Lagrangian with respect to $P(y)$ and setting to zero gives

$$P(y) = \sum_i \frac{\rho_i}{\sum_{i'} \rho_{i'}} Q_i(y) = \sum_i w_i \frac{(P_i(y)^{\lambda_i})^{\gamma_i} P(y)^{1-\gamma_i}}{Z_i}, \text{ s.t. } P(y) \geq 0. \tag{2.22}$$

where $w_i = \rho_i / \sum_i' \rho_{i'}$, and $\gamma_i = 1/(1+\rho_i)$, and $Z_i = \sum_{y'} (P_i(y')^{\lambda_i})^{\gamma_i} P(y')^{1-\gamma_i}$. Comparing this with (2.8) we see that this form of maximum entropy distribution is equivalent to the Rényi divergence aggregator of annealed forms of P_i. The maximum entropy

parameters of the aggregator could not be obtained explicitly by solving for the constraints in our case. We could estimate them using additional validation data from $P(y)$ by maximum likelihood. Empirically we find that, if all the P_i are trained on the same data, or on data subject to sample-selection bias (rather than say an annealed form of the required distribution), then $\lambda_i \approx 1$.

Note that the parameter ρ_i controls the level of penalty there is for a mismatch between the biased distributions Q_i and the distribution P. If all the ρ_i are zero for all i then this penalty is removed and the Q_i can bear little resemblance to the P and hence to one another. In this setting (2.22) becomes a standard mixture and the aggregator is a linear opinion pool. If however ρ_i tends to a large value for all i, then the distributions Q_i are required to be much more similar. In this setting (2.22) becomes like a log opinion pool.

We have shown that the Rényi divergence aggregator is not an arbitary choice of aggregating distribution. Rather it is the maximum entropy aggregating distribution when the individual agent distributions are expected to be biased using a sample selection mechanism.

2.6 Optimization of Weighted Rényi Divergence Aggregators

Rényi divergence aggregators can be implemented with direct gradient based optimization, stochastic gradient methods, or using a variational optimization for the sum of weighted divergences. This latter approach is described here.

To obtain the optimal distribution from the weighted Rényi divergence criterion in Eq. (2.6) we introduce a group of variational distributions $Q_i(y)$, and apply Jensen's inequality to the sum of weighted Rényi divergence given by Definition 2. Then we can obtain its upper bound, derived as follows.

$$\sum_i w_i D(P_i, Q) = \sum_i \frac{w_i}{\gamma_i(\gamma_i - 1)} \log \left(\sum_{y=1}^{K} P_i(y)^{\gamma_i} Q(y)^{1-\gamma_i} \right) \tag{2.23}$$

$$= \sum_i \frac{w_i}{\gamma_i(\gamma_i - 1)} \log \left(\sum_{y=1}^{K} \frac{P_i(y)^{\gamma_i} Q(y)^{1-\gamma_i}}{Q_i(y)} Q_i(y) \right) \tag{2.24}$$

$$\leq \sum_i \frac{w_i}{\gamma_i(\gamma_i - 1)} \sum_{y=1}^{K} Q_i(y) \log \frac{P_i(y)^{\gamma_i} Q(y)^{1-\gamma_i}}{Q_i(y)} \tag{2.25}$$

Algorithm 2.1 Variational Optimization for Weighted Rényi Divergence Aggregators

1: **Initialize** w'_i such that $\sum_i w'_i = 1$, and $Q(y)$.
2: **while** not convergent **do**
3: Set $Q_i(y) \propto P_i(y)^{\gamma_i} Q(y)^{1-\gamma_i}$;
4: Compute $q_{in} = w'_i Q_i(y_n) / \sum_i w'_i Q_i(y_n)$;
5: Compute the mixture coefficients as $w'_i = \sum_n q_{in} / \sum_{in} q_{in}$;
6: Set $Q(y) \propto \sum_i w'_i \gamma_i Q_i(y)$.
7: **end while**
8: **Return** $P(y) = Q(y)$.

Note equality is obtained in (2.25) for $Q_i(y) \propto P_i(y)^{\gamma_i} Q(y)^{1-\gamma_i}$ according to the property of Jensen's inequality. Optimizing for Q gives $P(y) = Q_{\text{opt}}(y) = \sum_i w'_i Q_i(y)$ with $w'_i = w_i \gamma_i^{-1} / \sum_i w_i \gamma_i^{-1}$.

This directly leads to an iterative variational optimization algorithm that is guaranteed (using the same arguments as Expectation and Maximization (EM) procedure, and using the convexity to optimize (2.25)): iteratively set $Q_i(y) \propto P_i(y)^{\gamma_i} Q(y)^{1-\gamma_i}$, and then set $Q(y) \propto \sum_i w'_i Q_i(y)$. The optimization of the parameters w'_i also naturally fits within this framework. $Q(y)$ is a simple mixture of $Q_i(y)$. Hence given $Q_i(y)$, the optimal w'_i are given by the optimal mixture model parameters. These can be determined using a standard inner Expectation Maximization loop. In practice, we get faster convergence if we use a single loop. First set $Q_i(y) \propto P_i(y)^{\gamma_i} Q(y)^{1-\gamma_i}$. Second compute $q_{in} = w'_i Q_i(y_n) / \sum_i w'_i Q_i(y_n)$. Third set $w'_i = \sum_n q_{in} / \sum_{in} q_{in}$. Finally set $Q(y) \propto \sum_i w'_i \gamma_i Q_i(y)$. This is repeated until convergence. All constants of proportionality are given by normalisation constraints. Note that where computing the optimal Q may be computationally prohibitive, this process also gives rise to an approximate divergence minimization approach, where Q_i is constrained to a tractable family while the optimizations for Q_i are performed. We describe the variational optimization procedure in Algorithm 2.1.

In practice, we found that this variational optimization converges slower than gradient-based methods. Particularly for large-scale datasets, stochastic gradient methods performs better. Thus, in the following experiments, gradient-based methods will be used for optimization.

2.7 Experiments

To test the practical validity of the maximum entropy arguments, the following three tasks were implemented.

Algorithm 2.2 Generate test data for agents with different biases, and test aggregation methods.

Select a target discrete distribution $P^*(.)$ over K values. Choose N_A, the number of agents.

Sample i.i.d a small number N_{Va} of values from the target distribution to get a validation set $\mathcal{D}_{Va}$

Sample i.i.d a large number N of values $\{y_n; n = 1,2,3,\ldots,N\}$ from the target distribution to get the base set $\mathcal{D}$ from which agent data is generated.

Sample bias probabilities $f_i(y)$ for each agent to be used as a rejection sampler.

for annealing parameter $\beta = 0$ TO 4 **do**

for each agent i **do**

Anneal f_i to get $f_k^*(y) = f_k(y)^\beta ./ \max_y f_i(y)^\beta$.

For each data point y_i, reject it with probability $(1 - f_k^*(y_i))$.

Collect the first $10,000$ unrejected points, and set P_i to be the resulting empirical distribution.

This defines the distribution P_i for agent i given the value of β.

end for

Find aggregate $P(.)$ for different aggregators given agent distributions P_i using the validation dataset $\mathcal{D}_{Va}$ for any parameter estimation.

Evaluate the performance of each aggregator using the KL Divergence between the target distribution $P^*(.)$ and the aggregate distribution $P(.)$: $KL(P^*||P)$.

end for

2.7.1 Task 1: Aggregation on simulated data

We aim to test the variation of the aggregator performance as the bias of the agent datasets is gradually changed. This requires that the data does not dramatically change across tests of different biases. We tested this process using a number of bias generation procedures, all with the same implication in terms of results. We give details for data generation and testing for the simplest approach in Algorithm 2.2, and summarize that approach here. We use $N_A = 10$ agents, $K = 64$ possible classes, $N_{Va} = 100$ vali-

dation data points, and a discretized $N(32, 64/7)$ normal distribution as the target. For each agent i and each class y an $f_i(y)$ was sampled from a uniform distribution. Agent data was generated by sampling from the target distribution and then rejecting using an annealed rejection probability proportional to $f_i(y)^{\beta}$ for each class. Hence each agent had different sampling biases and the annealing amount β was varied between $\beta = 0$ – a uniform rejection where all agent had samples form the target distribution – to the case $\beta = 4$ where the agent distributions were very different.

The details of the data generation and testing is given in Algorithm 2.2. We used $N_A = 10$, $K = 64$, $N_{Va} = 100$, P^* was a discretized $N(32, 64/7)$, $f_i(y)$ $U([0,1])$ to generate the artificial data that gave the results displayed here. Equivalent results were found for all (non-trivial) parameter choices we tried, as well as using completely different data generation procedures generating biased agent data.

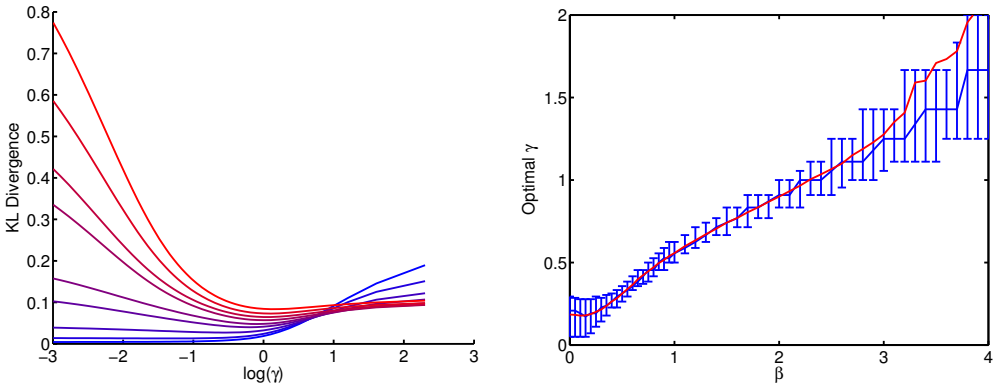
Figure 2.1(a) shows the test performance on different biases for different values of $\log(\gamma_i)$ in (2.22), where all γ_i are taken to be identical and equal to γ. Figure 2.1(b) shows how the optimal value of γ changes, as the bias parameter β changes. Parameter optimization was done using a conjugate gradient method. The cost of optimization for Rényi mixtures is comparable to that of log opinion pools.

2.7.2 Task 2: Aggregation on chords from Bach chorales

This task aims to accurately predict distributions of chords from Bach chorales (Bache and Lichman, 2013). The Bach chorales data was split equally and randomly into training and test distributions. Then training data from half of the chorales was chosen to be shared across all the agents. After that each agent received additional training data from a random half of the remaining chorales. The probabilistic model used by each agent is a mixture of Bernoulli's with a randomized number of mixture components between 5 and 100, and a random regularisation parameter between 0 and 1. 10 agents were used and after all 10 agents were fully trained. The Rényi mixture weights were optimized using the whole training dataset. Performance results were computed on the held-out test data.

Figure 2.2 shows the performance on the Bach chorales with 10 agents. Again in this real data setting, the Rényi mixtures show improved performance.

The two demonstrations show that when agents received a biased subsample of the overall data then Rényi-mixtures perform best as an aggregation method, in that they give the lowest KL divergence. As the bias increases, so the optimal value of γ



(a) (b)

Figure 2.1: (a) Task 1: Plot of the KL divergence against $\log\gamma$ for one dataset with $\beta = 0$ (lower lines, blue) through to $\beta = 4$ (upper lines, red) in steps of 0.5. Note that, unsurprisingly, more bias reduces performance. However the optimal value of γ (lowest KL), changes as β changes. for low values of β the performance of $\gamma = 0$ (log opinion pools) is barely distinguishable from other low γ values. Note that using a log opinion pool (low γ) when there is bias produces a significant hit on performance. (b) Task 1: Plot of the optimal γ (defining the form of Rényi mixture) for different values of β (determining the bias in the generated datasets for each agent). The red (upper) line is the mean, the blue line the median and the upper and lower bars indicate the 75th and 25th percentiles, all over 100 different datasets. For $\beta = 0$ (no bias) we have optimal aggregation with lower γ values, approximately corresponding to a log opinion pool. As β increases, the optimal γ gets larger, covering the full range of Rényi Mixtures.

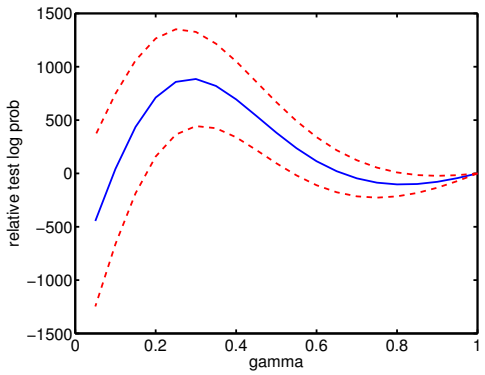


Figure 2.2: Task 2: test log probability results (relative to the log probability for a mixture) for the Bach chorales data for different values of γ, indicating the benefit of Rényi mixtures over linear ($\gamma = 1$) and log ($\gamma = 0$) opinion pools. Error bars are standard errors over 10 different allocations of chorales to agents prior to training.

increases. In the limit that the agents see almost the same data from the target distribution, Rényi-mixtures with small γ perform the best, and are indistinguishable from the $\gamma = 0$ limit. Rényi mixtures are equivalent to log opinion pools for $\gamma \to 0$.

2.7.3 Task 3: Aggregation on Kaggle competition

When all agents see the same data, the maximum-entropy aggregate is the log opinion pool. One setting of particular interest is in machine learning competitions, where the same training data is made publicly available to everyone. In this section we compare a number of aggregation methods on a real competition, and confirm that log opinion pools are the aggregation method of choice.

To analyze the use of combination methods in a realistic competition setting, we need data from an appropriate competitive setup. For this purpose we designed and ran a Kaggle-in-Class competition[4] described in this section. The competition consisted of a critical problem in low-level image analysis: the image coding problem, which is fundamental in image compression, infilling, super-resolution and denoising. We used data consisting of images from van Hateren's Natural Image Dataset[5] (Hateren and Schaaf, 1998). The data was preprocessed using Algorithm 2.3 to put it in a form suitable for a Kaggle competition, and ensure the data sizes were sufficient for use on student machines, and that submission files were suitable for uploading (this is the

[4] `https://inclass.kaggle.com/c/mlpr-challenge`
[5] `http://bethgelab.org/datasets/vanhateren/`

Algorithm 2.3 Competition Data Preparation

Load image data. Discretize to 64 gray scales. Put in INT8 format.

for j=1 to 140000 **do**

Pick random image and random pixel at least 40 pixels away from edge of image and find 35×30 patch including that pixel at the bottom-middle of the patch.

Record $x(j)$ =vectorisation of all pixels in patch 'before' that pixel in patch in raster-scan terms, $y(j)$ =grayscale value at chosen pixel,$i(j)$ =image number

end for

Produce three Matlab datasets. Set 1: x and y and i values in one `.mat` for 100000 training records. Set 2: x and i values in one `.mat` file for 40000 test records. Set 3: y values for the corresponding test cases, not publicly available.

reason for the 6 bit grayscale representation).

The competition problem was to infer $P(y|x,i)$, the predictive distribution for the grayscale value of the pixel at a given location, where y takes one of 64 possible values. The information given was the image number i and a raster scan x of an image patch above and up to a given pixel location (see Figure 2.3, which clarifies the form of the data). Patches were taken randomly from a large image corpus. Competitors were provided with three files specified in Algorithm 2.3. The competition submissions were unnormalized log probabilities at the test set points: $\log P(y = k|x,i)$, with one column for each k. The normalization was computed by the competition evaluation mechanism to prevent any room for cheating by false normalization. The test cases were split into a public set and a private set. The competitor was given the perplexity on the public set at submission time, but the final ranked ordering was on the private set. The perplexity is given by

$$\text{perplexity} = \exp\left(-\frac{1}{N_t}\sum_{j=1}^{N_t}\log(P(y_j = c_j|\mathbf{x}_j, i_j))\right), \tag{2.26}$$

where N_t is the number of test pixels and c_j is the true class the jth pixel belongs to, and $\mathbf{x}_j$ and i_j are the provided covariates. Note that perplexity is equivalent to test probability up to a monotonic transformation, and thus the lower the perplexity is, the better the model performs.

There were 46 competitors, with a total of 440 submissions. Some submissions were highly erroneous (submitting probabilities instead of log probabilities etc.), but competitors quickly fixed these issues for future submissions. A uniform prediction was used as a dummy baseline which has perplexity 64. We chose as agent distributions

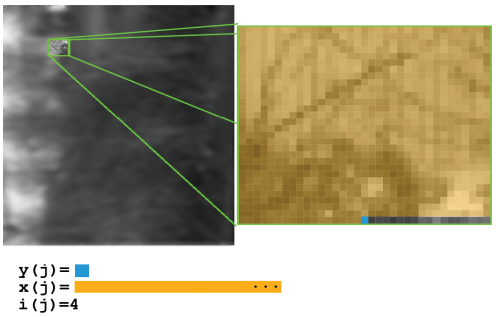


Figure 2.3: Competition form: competitors had to infer the probabilities for a pixel taking each of 64 values, given values for pixels above and to the left of that pixel. Note that i labels the original image source: iml00004.imk. The image patch is 35 pixels (horizontal) $\times$ 30 pixels (vertical).

the 269 submissions that had perplexity greater than 64.

2.7.3.1 Analysis of the Competition

The following aggregation methods were tested: weighted Rényi divergence aggregators, including linear opinion pools and log opinion pools, simple averaging of the top submissions (with an optimized choice of number), and a form of heuristic Bayesian model averaging, via an annealed likelihoood: $P(y|\cdot) \propto \sum_j P_j(y|\cdot)\,(P(j|\mathcal{D}_{\text{tr}}))^{\alpha}$, where α is an aggregation parameter choice[6]. The weighted Rényi divergence aggregators were optimized using stochastic gradient methods, until the change between epochs became negligible. The validation set ($20,000$ pixels) is used for learning the aggregation parameters. The test set (also $20,000$ pixels) is only used for the test results.

In order to show the generalization performance of all the aggregation methods, we split the private set into 10 subsets and apply each method to each subset to obtain the mean perplexity and the statistics for the difference in perplexity for all methods.

We present the test results of simple averaging in Table 2.2. Averaging inevitably depends on the number of on agents included in the average and so results are presented for various numbers of the best agents (in terms of ranked perplexity in the public set) on the private set. Despite the simplicity of a simple average, past experience has shown it to be remarkably effective: in most cases this can be understood in terms of bias variance tradeoff. Simple averaging, for example, is used in random forest ensemble methods.

[6]The heuristic model averaging, includes Bayesian Model Averaging as a special case. However we emphasize that Bayesian Model Averaging, though discussed in the context of aggregation Dietterich (2000); Domingos (1997), is not formulated as an aggregation method: it assumes only one of the submissions is actually correct Minka (2002).

Table 2.2: The perplexity on private set for simple averaging over different number of top agents. Simple averaging is easy to achieve, and requires nothing more that the public validation set to choose the numbers.

Top 5	Top 10	Top 15	Top 20	Top 25	Top 30	All
2.972 ± 0.074	2.931 ± 0.065	2.946 ± 0.064	2.958 ± 0.064	2.965 ± 0.063	2.974 ± 0.062	3.665 ± 0.071

The results for Bayesian model averaging (unsurprisingly, in the context of the discussion in Section 2.2.1.2) is identical to the top single classifier. For Bayesian averaging with power heuristics (using all the agents), the lowest perplexity achieved with this approach is $\underline{2.929}$ at $\alpha = 0.0049$. This approach involves no optimization passes, so is particularly simple to implement. However, it is dependent on hyperparameter selection.

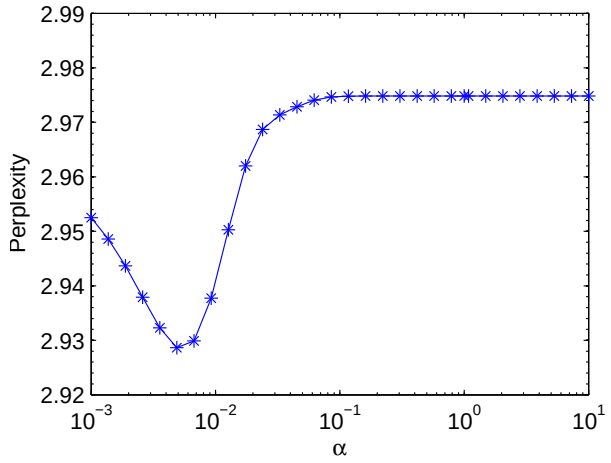


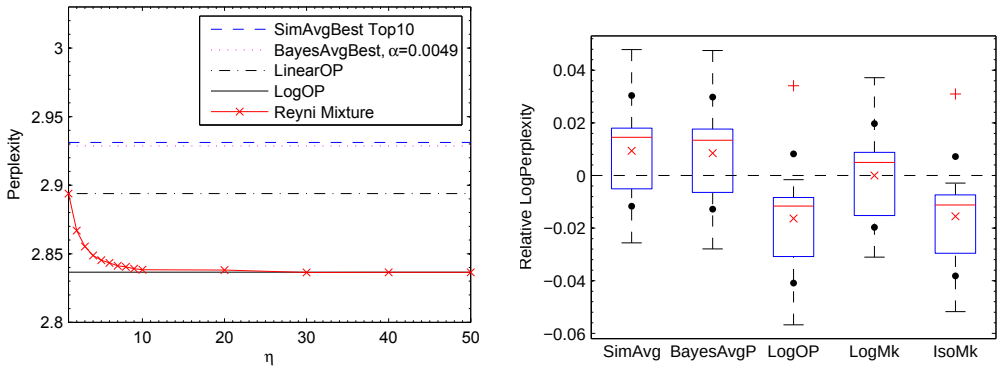
Figure 2.4: Heuristic 'Bayesian' Model Averaging with different α.

For Task 3, all agents see unbiased data and so we would expect log opinion pools to be optimal. The perplexity values as a function of $\eta = 1/\gamma$ for all the methods tested on the test set can be seen in Figure 2.5. The parameter-based pooling methods perform better than simple averages and all forms of heuristic model averaging as these are inflexible methods. There is a significant performance benefit of using logarithmic opinion pooling over linear pooling, and weighted Rényi divergence aggregators interpolate between the two opinion pooling methods. This figure empirically supports the maximum entropy arguments.

The mean perplexity values and standard deviation for all the methods tested can be seen in Figure 2.5 and Table 2.3. Table 2.3 also shows the difference in log perplexity between each approach and shows the estimated standard deviation of those

Methods:	SimpleAvgBest	Heuristic	LogOP	Reyni Mixture
Perplexity: (LinearOP: 2.894 $\pm$ 0.060)				
mean$\pm$std	2.931 $\pm$ 0.065	2.929 $\pm$ 0.067	2.837 $\pm$ 0.067	2.836 $\pm$ 0.061
Log Perplexity Difference from LinearOP:				
mean$\pm$std	0.013 $\pm$ 0.021	0.012 $\pm$ 0.021	-0.020 $\pm$ 0.024	-0.020 $\pm$ 0.021
p value	1	1	8.0×10^{-7}	5.3×10^{-7}

Table 2.3: Top: The perplexity on the test set for all the aggregation methods. The log opinion pool and the Rényi mixture for $\gamma = (1/30)$ are fairly equivalent. Bottom: Relative log perplexity using linear pool as the baseline, and corresponding p-value. Log opinion pools and Rényi mixture with sufficiently small γ perform significantly better than linear pooling. All perform better than the best individual competition entry (2.963).



(a) (b)

Figure 2.5: (a) Task 3: perplexity on the test set of all the compared aggregation methods against $\eta = 1/\gamma$. For each method, the best performance is plotted. Log opinion pools perform best as suggested by the maximum entropy arguments, and is statistically significantly better than the linear opinion pool($p = 8.0 \times 10^{-7}$). All methods perform better than the best individual competition entry (2.963). (b) Task 3: boxplot of relative log perplexity (using linear opinion pool as the baseline). The red 'x' represents the mean, and black dots are the standard deviation.

differences (log perplexity values appear approximately Gaussian), and corresponding single-tailed t-test sample probability (p-value) under the null assumption that the method is equivalent to the linear opinion pool. There is a statistically significant performance benefit of using logarithmic opinion pooling over linear pooling. The parameter-based pooling methods perform better than simple averages and all forms of heuristic model averaging as these are inflexible methods. All results have been

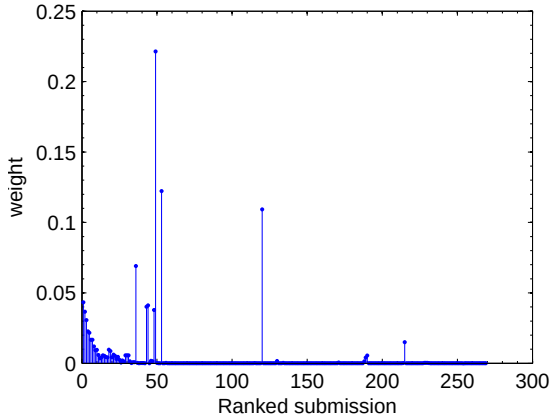


Figure 2.6: Weight distribution over 269 ranked submissions (agents) for the log opinion pool and Rényi mixture with $\eta = 1/\gamma = 30$.

tested for reproducibility using multiple initializations.

In this context, the 'value' of a submission is not the same as its performance: rather the importance of the contribution to the overall aggregate probability depends on how it is combined. A single good contribution, that is different from the others, is usually more valuable than a slightly higher scoring contribution that is very similar to all the others. Figure 2.6 shows the weights of the contributions for the log opinion pools and Rényi mixture with $\eta = 1/\gamma = 30$ (for $\eta \geq 30$ the results are sufficiently similar to the log opinion pool). Further analysis shows that the obvious spikes in weight are due to particular contributions that are noticeably different for the bulk of the high scoring contributions.

2.8 Machine Learning Markets and Rényi Divergence Aggregation

Machine learning markets (MLMs) with isoelastic utilities (Storkey et al., 2012) are an information market based aggregation method. Independent agents with different beliefs trade in a securities market. The equilibrium prices of the goods in that securities market can then be taken as an aggregate probability distribution, aggregating the individual agent beliefs.

Since machine learning markets are also designed for aggregating probabilistic predictions, it is interesting to explore the connection with weighted Rényi divergence aggregators introduced before. To this end, we first review the model details of MLMs and the reveal the connections between the two approaches.

2.8.1 Model Details of Machine Learning Markets

We now introduce the machine learning markets proposed by Storkey et al. (2012). This market mechanism is designed for aggregating probabilistic predictions from multiple agents through the equilibrium of the markets. Concretely, the market goods are enumerated by $k = 1, 2, \ldots, N_G$, corresponding to different outcomes of an event or a discrete random variable k. The good k will pay out one unit of currency if the outcome of the event is k. For each good k, the market has a commonly agreed price c_k ($0 < c_k < 1$), thus, the price vector $\mathbf{c} = (c_1, c_2, \ldots, c_{N_G})^T$. In MLMs, $\mathbf{c}$ is interpreted as the aggregated belief from the market agents trading in the market, $i = 1, 2, \ldots, N_A$. Each agent with wealth W_i invests an amount of money r_{ik} in good k, thus, the investment vector $\mathbf{r}_i = (r_{i1}, r_{i2}, \ldots, r_{iN_G})^T$. To facility belief aggregation for machine learning models, there are several assumptions in MLMs:

- No arbitrage assumption, which means that there is no possibilities for a risk free gain. This implies
$$\sum_k c_k = 1. \tag{2.27}$$
Suppose $\sum_k c_k \neq 1$, an agent can buy (or sell) one of each stock to achieve a sure return of 1 unit. The "sum-to-one" property also makes it possible to be interpreted as a probability distribution.

- All agents spend all their wealth:
$$\sum_k r_{ik} = W_i. \tag{2.28}$$
If any agent wants to keep a risk-free investment, that agent can simply purchase one of each good to achieve free risk.

- $\sum_i W_i = 1$, hence, $\sum_{ik} r_{ik} = 1$.

- We assume that the market is a close trading system, which implies that the wealth in the market must be conserved: the total payout were that item to occur matches the total original wealth.
$$\sum_{i=1}^{N_A} \frac{r_{ik}}{c_k} = \sum_{i=1}^{N_A} W_i = 1 \Rightarrow \sum_{i=1}^{N_A} r_{ik} = c_k. \tag{2.29}$$
Note that r_{ik}/c_k is the amount of good k bought by agent i and so is the amount received if the outcome is k.

Table 2.4: The investment function and market equilibrium for different types of agents

Type of agent	Utility function	Investment function	Market equilibrium
Exponential	$U(W) = -\exp(-W)$	$r^*_{ik} = \frac{W_i}{N_G} + c_k \log \frac{P_i(k)}{c_k} - \frac{1}{N_G}\sum_{k'} c_{k'} \log \frac{P_i(k')}{c_{k'}}$	$c_k = \frac{\prod_{i=1}^{N_A} P_i(k)^{1/N_A}}{\sum_{k'} \prod_{i=1}^{N_A} P_i(k')^{1/N_A}}$
Logarithmic	$U(W) = \log(W)$	$r^*_{ik} = W_i P_i(k)$	$c_k = \frac{\sum_i W_i P_i(k)}{\sum_i W_i}$
Isoelastic	$U(W) = \frac{W^{1-\eta}-1}{1-\eta}$	$r^*_{ik} = W_i \left(\frac{(c_k)^{\frac{\eta-1}{\eta}} (P_i(k))^{\frac{1}{\eta_i}}}{\sum_{k'} (c_{k'})^{\frac{\eta-1}{\eta}} (P_i(k'))^{\frac{1}{\eta}}} \right)$	$c_k = \sum_i W_i \frac{c_k (\frac{P_i(k)}{c_k})^{\frac{1}{\eta_i}})}{\sum_{k'} c_{k'} (\frac{P_i(k')}{c_{k'}})^{\frac{1}{\eta_i}}}$

Each agent has a utility function $U_i(W)$ and belief $\mathbf{P}_i$, with $P_i(k)$ denoting the probabilistic belief for that the outcome of the event will be k. Also, for each agent, $\sum_k P_i(k) = 1$.

We consider multiclass classification problem here. Each individual classifier plays as an agent in the market. And the market price $\mathbf{c}$ defines a probability distribution over possible outcomes, which can be used for multiclass prediction. In the following we will show how to calculate investment for each agent and how to compute the market equilibrium, respectively.

2.8.1.1 Investment Calculation

Given the market price $\mathbf{c}$ and current wealth W, each agent i tries to maximize their expected utility to determine how much they will invest on the goods.

$$\begin{aligned} \mathbf{r}^*_i = \mathbf{r}^*_i(W_i, \mathbf{c}) = \arg\max_{\mathbf{r}_i} & \sum_k P_i(k) U_i(\frac{r_{ik}}{c_k}) \\ \text{s.t.} & \sum_k r_{ik} = W_i. \end{aligned} \tag{2.30}$$

The solution of this constrained optimization (2.30) is

$$r^*_{ik} = c_k (U_i')^{-1} \left(\lambda_i(\mathbf{c}) \frac{c_k}{P_i(k)} \right), \tag{2.31}$$

where the U' is the derivative of U and $\lambda_i(\mathbf{c})$ is the a Lagrange multiplier such that $\sum_k = r^*_{ik} = W_i$ is satisfied. Generally, λ_i cannot be solved explicitly. However, for a number of utility functions, the investment functions are analytic. The third column of Table.2.4 gives the investment functions (i.e., the r^*_{ik} the optimal investment over the price vector $\mathbf{c}$) for some important utility functions.

2.8.1.2 Market Equilibrium

It is well known that the concavity of each agent's utility guarantees the existence of a fixed price point for which all the agents maximise their utility and the market con-

straints are satisfied. In MLMs, the fixed point price is unique. For the exponential and logarithmic agents, we have an analytic form of market equilibrium, but not for the isoelastic agents, see the fourth column of Table.2.4. Storkey et al. (2012) employed the minimisation of KL divergence between price vector $\mathbf{c}$ and $\sum_i \mathbf{r}_i$ to find the equilibrium iteratively for the isoelastic agents. The computational complexity of each pass of this algorithm is $O(N_A \times N_G)$, which is equivalent to a mixture model update. However, the wealth update needs to be done for every single data sample. For large-scale problem, this will introduce a heavy computational burden.

The equilibrium for sets of isoelastic agents with $\eta \neq 1$ cannot be obtained in an explicit form. However the form of solutions has some connections with existing models. For instance, considering the homogeneous market of isoelastic agents (i.e., all having the same η), we can have its equilibrium solution in a non-close form:

$$c_k = \left[\sum_i V_i P_i(k)^{\frac{1}{\eta}}\right]^{\eta} \tag{2.32}$$

where $V_i = W_i/Z_i$ and Z_i is the implicit solution to

$$Z_i = \sum_k \left[\sum_j \frac{W_j}{Z_j} P_j(k)^{\frac{1}{\eta}}\right]^{\eta-1} P_i(k)^{\frac{1}{\eta}}. \tag{2.33}$$

Eq. (2.32) is the same as the form of α-integration (Amari, 2007; Wu, 2009), but where V_i is defined implicitly in terms of a set of weights (or wealths) W_i as in Eq. (2.33). Figure 2.7 sketches the difference between an isoelastic market combination and a logarithmic combination (i.e., a standard mixture). In the isolelastic MLM, for $\eta > 1$, the individual beliefs are raised to a fractional power ("squashed") before being mixed, and are then 'unsquashed' again after mixing. The result of this is the areas of agreement between agents are emphasised relative to a standard mixture.

As described in Storkey et al. (2012), Eq. (2.32) can also be written into the following form

$$c_k = \sum_i W_i P_{ik}^{\eta}(\mathbf{c}) \tag{2.34}$$

where $P_{ik}^{\eta}(\mathbf{c})$ is defined as

$$P_{ik}^{\eta}(\mathbf{c}) = \frac{c_k \left(\frac{P_i(k)}{c_k}\right)^{1/\eta}}{\sum_{k'} c_{k'} \left(\frac{P_i(k')}{c_{k'}}\right)^{1/\eta}}. \tag{2.35}$$

It is easy to observe that the form expresses the equilibrium c_k as a weighted sum of the *effective beliefs* P_{ik}^{η} that are associated with each agent once the impact of the

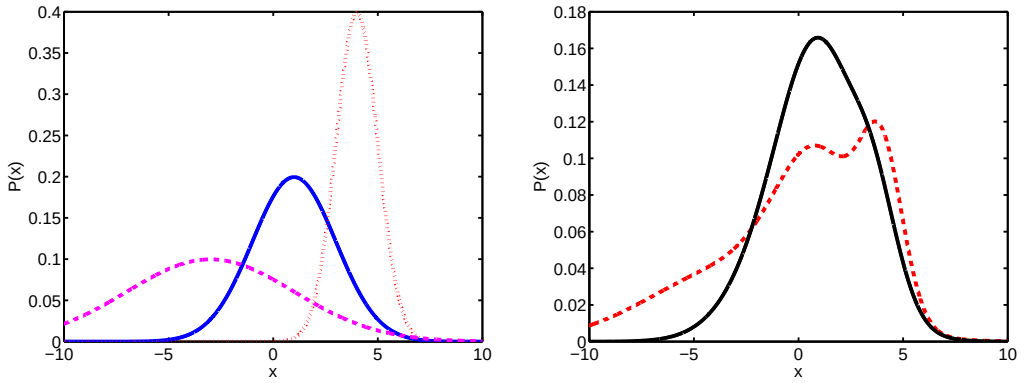


Figure 2.7: (a) Three different components (i.e. agent beliefs), each given weights W_i of 0.4, 0.4 and 0.2 from left to right. (b) The logarithmic (i.e. mixture) combination of these components (dashed) and the isoelastic ($\eta = 10$) combination (solid). Note the isoelastic combination puts more weight where the overlap of the different components are and down-weights the regions of disagreement or isolated components. Source from Storkey et al. (2012)

combination with rest of the market is taken into account. Each effective belief is weighted by the agent's wealth W_i before aggregation.

2.8.1.3 Market Training

Considering the classification problem, the aim of market training is to obtain a new wealth allocation for all the agents for belief aggregation in test phase. Intuitively, after market training, the agents (classifiers) with high accuracy will own more wealth than the agents with poor classification performance.

The individual belief $P_i(k)$ of each agent is obtained from their corresponding classification models. Each agent has a specific wealth W_i. The wealth and market price affect each other during the trading process. For each training point, all the agents make purchases based on their wealth and changing market price and finally the market reaches an equilibrium. When the true class of the current training point is revealed after trading, the payouts are made based on the investment of each agent on the correct class (goods), and then the wealth is updated for each agent. One pass for all the training points is called an epoch. This process should be repeated a number of epochs to reach a converged wealth allocation.

There are two types of wealth update mechanism: online mode and batch mode. Online update requires that the wealth is updated after passing each training data point,

Algorithm 2.4 MLM training and test

Input: D_{tr} (true classes included), D_{te}, utility type (with η), $p(D_{\text{tr}}|h)$ and $p(D_{\text{te}}|h)$.
Output: aggregated belief $\mathbf{c}$ for each test point.

MARKET TRAINING
repeat
  for each training point **do**
    Initialize: wealth W_i and prices c_k.
    Compute equilibrium and purchases by minimising KL divergence.
    if Online wealth update **then**
      Update the wealth based on the investment and payout.
    end if
    if Batch wealth update **then**
      Update the wealth after passing a batch of training points.
    end if
  end for
until Wealth W_i does not change.

MARKET TEST
for each test point **do**
  Initialize: goods prices c_k.
  Compute equilibrium c_k for each goods based on *the new wealth* from training phase.
end for

which is equivalent to Bayesian model updates. On the other hand, batch update is conducted after passing a batch of training points and it is shown that this type of update could avoid over penalising good predictors early on, and thus has a better aggregation performance.

The Algorithm 2.4 sketches how the market training and test are implemented.

2.8.2 Connection between MLMs and Rényi Divergence Aggregation

Following the notation and formalism in MLMs, agents indexed by i with belief $P_i(y)$, wealth W_i and utility function $U_i(.)$ trade in Arrow-Debreu securities derived from each possible outcome of an event. Given the agents maximize expected utility, the market equilibrium price of the securities $c(y)$ is used as an aggregate model $P(y) = c(y)$ of the agent beliefs. When each agent's utility is an isoelastic utility of the form $U_i(W) = W^{1-\eta_i}/(1-\eta_i)$ with a risk-averseness parameter η_i, the market equilibrium $P(y)$ is implicitly given by

$$P(y) = \sum_i \frac{W_i}{\sum_l W_l} \frac{P_i(y)^{\gamma_i} P(y)^{1-\gamma_i}}{\sum_{y'} P_i(y')^{\gamma_i} P(y')^{1-\gamma_i}} \tag{2.36}$$

with $\gamma_i = \eta_i^{-1}$ (generalising (10) in Storkey et al. (2012)). This shows the isoelastic market aggregator linearly mixes together components that are implicitly a weighted product of the agent belief and the final solution. Simple comparison of this market equilibrium with the Rényi Divergence aggregator (2.8) shows that the market solution and the Rényi divergence aggregator are of exactly the same form.

We conclude that a machine learning market implicitly computes a Rényi divergence aggregation via the actions of individual agents. The process of obtaining the market equilibrium is a process for building the Rényi Divergence aggregator, and hence machine learning markets provide a method of implementation of weighted Rényi divergence aggregators. The benefit of market mechanisms for machine learning is that they are *incentivized*. There is no assumption that the individual agents behave cooperatively, or that there is an overall controller who determines agents' actions. Simply, if agents choose to maximize their utility (under myopic assumptions) then the result is weighted Rényi Divergence aggregation.

In general, equilibrium prices are not necessarily straightforward to compute, but the algorithm in the implementation section provides one such method. As this iterates computing an interim P (corresponding to a market price) and an interim Q_i corresponding to agent positions given that price, the mechanism in this paper can lead to a form of tâtonnement[7] algorithm with a guaranteed market equilibrium – see e.g. (Cole and Fleischer, 2007).

The direct relationship between the risk averseness parameter for the isoelastic utilities and the bias controlling parameter of the Rényi mixtures ($\gamma_i = \eta_i^{-1}$) provides

[7]Tâtonnement is an iterative auction process by which an exchange equilibrium is assumed to be achieved.

an interpretation of the isoelastic utility parameter: if agents know they are reasoning with respect to a biased belief, then an isoelastic utility is warranted, with a choice of risk averseness that is dependent on the bias.

Storkey et al. (2012) show, on a basket of UCI datasets, that market aggregation with agents having isoelastic utilities performs better than simple linear opinion pools (markets with log utilities) and products (markets with exponential utilities) when the data agents see is biased. As such markets implement Rényi mixtures, this provides additional evidence that Rényi mixtures are appropriate when combining biased predictors.

2.9 Discussion

When agents are training and optimising on different datasets than one another, log opinion pooling is no longer a maximum entropy aggregator. Instead, under certain assumptions, the weighted Rényi divergence aggregator is the maximum entropy solution, and tests confirm this practically. The weighted Rényi divergence aggregator can be implemented using isoelastic machine learning markets.

Though there is some power in providing aggregated prediction mechanisms as part of competition environments, there is the additional question of the competition mechanism itself. With the possibility of using the market-based aggregation mechanisms, it would be possible to run competitions as prediction market or collaborative scenarios (Abernethy and Frongillo, 2011), instead of as winner takes all competitions. This alternative changes the social dynamics of the system and the player incentives, and so it is an open problem as to the benefits of this. We recognize the importance of such an analysis as an interesting direction for future work.

Chapter 3

Stochastic Methods for Separable Saddle Point Problems

In this chapter, we consider a general class of *C*onvex-*C*oncave *S*addle *P*oint problems with a *Sep*arable structure, and we refer to them as *Sep-CCSP* problems. This problem structure covers a wide range of important machine learning models, such as empirical risk minimization (ERM) and linear constrained optimization (for instance, robust principal component analysis). We aim at developing efficient methods for optimizing large-scale Sep-CCSP problems in the "Big Data" era to tame the computational bottleneck.

The key strategy we use is to incorporate stochastic block coordinate descent into the Sep-CCSP problems, as in the work by Zhang and Xiao (2015) i.e., in each iteration, only a random subset of coordinate blocks are updated. This strategy only uses local information from the chosen subset, and requires much less computation power than updating the entire coordinate blocks. We extend this by using adaptive stepsizes to accelerate the convergence speed.

Particularly, two different settings of Sep-CCSP problems are considered, Sep-CCSP with *strongly convex* functions and *non-strongly convex* functions. We develop efficient stochastic methods for both of the two cases, which allows fast parallel processing for large-scale data. Both theoretically and empirically, we demonstrate the developed methods perform comparably, or more often, better than state-of-the-art methods.

This chapter is an extended work based on two following papers, where ZZ was responsible for both the theoretical and empirical aspects, and AJS provided insightful suggestions for the two papers.

Zhu, Z. and Storkey, A.J.(2015). Adaptive Stochastic Primal-Dual Coordinate Descent for Separable Saddle Point Problems. In *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pages 645-658.

Zhu, Z. and Storkey, A.J.(2016). Stochastic Parallel Block Coordinate Descent for Large-scale Saddle Point Problems. In *30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, *accepted to appear*.

Terminology and Notations

Before introducing the Sep-CCSP problems, it is necessary to present the related terminology and notations that are commonly used in the optimization literature.

In this chapter, we denote $\mathbf{x} \in \mathbb{R}^{D_\mathbf{x}}$ as primal optimization variables and $\mathbf{y} \in \mathbb{R}^{D_\mathbf{y}}$ as dual variables.

1. **Convex function and strongly convex function.**

   Let $\mathcal{X}$ be a convex set in D-dimensional real space, a function $f\colon\ \mathcal{X} \to \mathbb{R}$ is *convex* if for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, and $0 \leq \alpha \leq 1$, we have

   $$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{x}') \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{x}') \tag{3.1}$$

   Geometrically, this inequality means that the line segment between $(\mathbf{x}, f(\mathbf{x}))$ and $(\mathbf{x}', f(\mathbf{x}'))$ lies above the graph of $f(\cdot)$. Suppose $f(\cdot)$ is differentiable, then $f(\cdot)$ is convex if and only if

   $$f(\mathbf{x}') \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T (\mathbf{x}' - \mathbf{x}) \tag{3.2}$$

   holds for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. This inequality shows that from *local information* about a convex function (i.e., its value and derivative at a point) we can derive *global information* (i.e., a global underestimator of it, $f(\mathbf{x}) + \nabla f(\mathbf{x})^T (\mathbf{x}' - \mathbf{x})$). This is perhaps the most important property of convex functions, and explain some of the remarkable properties convex functions and convex optimization problems.

   A differentiable function $f(\cdot)$ is called *strongly convex* with parameter $\lambda > 0$ if the following inequality holds for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$:

   $$f(\mathbf{x}') \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T (\mathbf{x}' - \mathbf{x}) + \frac{\lambda}{2} \|\mathbf{x}' - \mathbf{x}\|_2^2. \tag{3.3}$$

   This equality is also important for strongly convex functions since it can provide a global quadratic estimator of the function $f(\cdot)$.

2. **Lipschitz smoothness**.

   A function $f(\cdot)$ is *Lipschitz smooth* if its gradients are Lipschitz continuous with constant L, that is, for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{D_\mathbf{x}}$,

   $$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')\| \leq L \|\mathbf{x} - \mathbf{x}'\|. \tag{3.4}$$

Table 3.1: Some examples of conjugate functions.

Function	$f(\cdot)$	$f^*(\cdot)$
Affine	$f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} - b$	$f^*(\mathbf{y}) = \begin{cases} b, & \text{if } \mathbf{y} = \mathbf{a} \\ +\infty, & \text{otherwise.} \end{cases}$
Power	$f(x) = \frac{1}{a}\lvert x\rvert^a$, $1 < a < \infty$	$f^*(y) = \frac{1}{b}\lvert y\rvert^b$, where $1 < b < \infty$, and $\frac{1}{a} + \frac{1}{b} = 1$.
Quadratic	$f(\mathbf{x}) = \frac{1}{2}(x-b)^2$	$f^*(\mathbf{y}) = \frac{1}{2}y^2 + by$
Exponential	$f(x) = e^x$	$f^*(y) = \begin{cases} y\log(y) - y, & \text{if } y > 0 \\ 0, & \text{if } y = 0, \\ \infty, & \text{otherwise.} \end{cases}$
Smooth hinge	$f(x) = \begin{cases} 0 & \text{if } x \geq 1, \\ 1 - \frac{\gamma}{2} - x & \text{if } x \leq 1 - \gamma \\ \frac{1}{2\gamma}(1-x)^2 & \text{otherwise.} \end{cases}$	$f^*(y) = y + \frac{1}{2}y^2$, where $y \in [-1, 0]$.
Logistic	$f(x) = \log(1 + \exp(-x))$	$f^*(y_i) = -y\log(-y) + (1+y)\log(1+y)$, where $y \in (-1, 0)$.

3. **The conjugate function.**

   The function $f^*(\cdot)$ is called the *conjugate* of the function $f(\cdot)$ defined as

   $$f^*(\mathbf{y}) = \max_{\mathbf{x} \in \mathcal{X}} \langle \mathbf{y}, \mathbf{x} \rangle - f(\mathbf{x}), \tag{3.5}$$

   where $\langle \cdot, \cdot \rangle$ is the inner product of two vectors. Figure 3.1 illustrates the conjugate function in one-dimensional space. We see immediately that f^* is a convex function, since it is the pointwise supremum of a family of convex (indeed, affine) functions of $\mathbf{y}$. This is true whether or not f is convex.

   We present several examples of conjugate functions in Table 3.1, some of which will be used later.

   There is a direct and important relation: a function is strongly convex with constant λ if and only if its convex conjugate is Lipschitz smooth with constant $1/\lambda$ (Theorem 4.2.2 in Hiriart-Urruty and Lemaréchal (2001)).

4. **Proximal operators**.

   A *proximal operator* with parameter λ, $\mathbf{prox}_{\lambda f} : \mathbb{R}^D \to \mathbb{R}^D$ of f is defined by

   $$\mathbf{prox}_{\lambda f}(\mathbf{v}) = \text{argmin}_{\mathbf{x}} f(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{x} - \mathbf{v}\|_2^2. \tag{3.6}$$

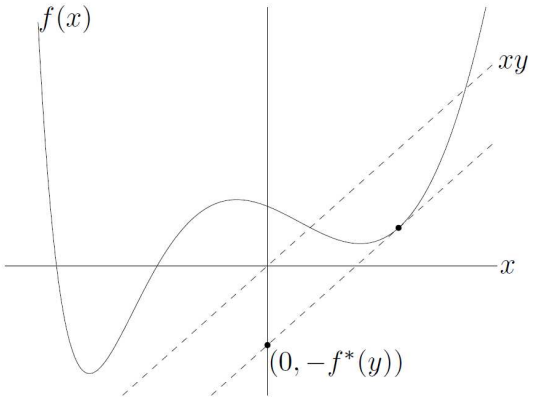


Figure 3.1: The conjugate function $f^*(y)$ is the maximum gap between the linear function yx and $f(x)$, as shown by the dashed line in the figure. If f is differentiable, this occurs at a point x where $\nabla f(x) = y$.

This definition indicates that $\mathbf{prox}_{\lambda f}(\mathbf{v})$ is a point that compromises between minimizing f and being near to $\mathbf{v}$.

Figure 3.2 illustrates the evaluation of a proximal operator. The thin black lines are level curves of a convex function f; the thicker black line indicates the boundary of its domain. Evaluating $\mathbf{prox}_f$ at the blue points moves them to the corresponding red points. The three points in the domain of the function stay in the domain and move towards the minimum of the function, while the other two move to the boundary of the domain and towards the minimum of the function. The penalty parameter λ controls the extent to which the proximal operator maps points towards the minimum of f, with larger values of λ associated with mapped points near the minimum, and smaller values giving a smaller movement towards the minimum.

We list the evaluation of several important proximal operators, which are used frequently in proximal algorithms, see Parikh and Boyd (2013) for a more comprehensive review.

Quadratic functions. If $f(\mathbf{x}) = (1/2)\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$, with $\mathbf{A} \in \mathbb{S}^n_+$, then

$$\mathbf{prox}_{\lambda f}(\mathbf{v}) = (\mathbf{I} + \lambda\mathbf{A})^{-1}(\mathbf{v} - \lambda b). \tag{3.7}$$

Particularly, if $f(\cdot) = (1/2)\|\cdot\|_2^2$, then

$$\mathbf{prox}_{\lambda f}(\mathbf{v}) = \frac{\mathbf{v}}{1+\lambda}, \tag{3.8}$$

is named as *shrinkage operator*.

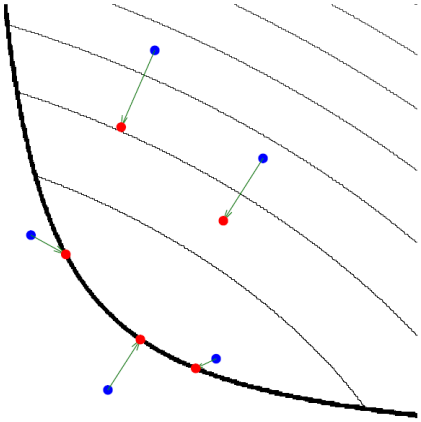


Figure 3.2: Evaluating a proximal operator at various points. The thin black lines are level curves of a convex function f; the thicker black line indicates the boundary of its domain. Evaluating $\mathbf{prox}_f$ at the blue points moves them to the corresponding red points. The three points in the domain of the function stay in the domain and move towards the minimum of the function, while the other two move to the boundary of the domain and towards the minimum of the function. The penalty parameter λ controls the extent to which the proximal operator maps points towards the minimum of f, with larger values of λ associated with mapped points near the minimum, and smaller values giving a smaller movement towards the minimum. Source from Parikh and Boyd (2013).

Logarithm functions. If $f(x) = -\log(x)$, then

$$\mathbf{prox}_{\lambda f}(v) = \frac{v + \sqrt{v^2 + 4v}}{2}. \tag{3.9}$$

l_1 norm. If $f(\mathbf{x}) = \|\mathbf{x}\|_1$, then each element of the proximal evaluation,

$$\left(\mathbf{prox}_{\lambda f}(\mathbf{v})\right)_i = \begin{cases} v_i - \lambda & v_i \geq \lambda \\ 0 & |v_i| \leq \lambda \\ v_i + \lambda & v_i \leq -\lambda \end{cases}, \tag{3.10}$$

which is known as (elementwise) *soft thresholding operator* and can be expressed more compactly as

$$\mathbf{prox}_{\lambda f}(\mathbf{v}) = (\mathbf{v} - \lambda)_+ - (-\mathbf{v} - \lambda)_+, \tag{3.11}$$

where the thresholding operator $((\mathbf{u})_+)_i = \max(u_i, 0)$.

Euclidean (l_2) norm. If $f(\mathbf{x}) = \|\mathbf{x}\|_2$, then

$$\mathbf{prox}_{\lambda f}(\mathbf{v}) = (1-\lambda/\|\mathbf{v}\|_2)_+\mathbf{v} = \begin{cases} (1-\lambda/\|\mathbf{v}\|_2)\mathbf{v} & \|\mathbf{v}\|_2 \geq \lambda \\ 0 & \|\mathbf{v}\|_2 < \lambda \end{cases}, \tag{3.12}$$

which is called *block soft thresholding operator*.

Elastic net. Elastic net regularization (Zou and Hastie, 2005) is a combination of l_1 and l_2 norm,

$$f(\mathbf{x}) = \|\mathbf{x}\|_1 + (\gamma/2)\|\mathbf{x}\|_2^2, \tag{3.13}$$

where $\gamma > 0$. Then

$$\mathbf{prox}_{\lambda f}(\mathbf{v}) = \frac{1}{1+\lambda\gamma}\mathbf{prox}_{\lambda\|\cdot\|_1}(\mathbf{v}), \tag{3.14}$$

i.e., soft thresholding followed by multiplicative shrinkage.

Sum of norms. Another important case is *sum-of-norms regularization*, used in group Lasso (Zhao et al., 2009; Jacob et al., 2009),

$$f(\mathbf{x}) = \sum_{g\in\mathcal{G}} \|\mathbf{x}_g\|_2, \tag{3.15}$$

where $\mathcal{G}$ is a partition of $[n]$. The its proximal operator has the form

$$(\mathbf{prox}_{\lambda f}(\mathbf{v}))_g = \left(1 - \frac{\lambda}{\|\mathbf{v}_g\|_2}\right)_+ \mathbf{v}_g \tag{3.16}$$

for all $g \in \mathcal{G}$.

Nuclear matrix norm. The *nuclear norm* or *trace norm* of a matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$ is the l_1 norm of its singular values,

$$f(\mathbf{X}) = \|\boldsymbol{\mu}(\mathbf{X})\|_1, \tag{3.17}$$

where $\boldsymbol{\mu}(\mathbf{X})$ denotes the vector including all the singular values of the matrix $\mathbf{X}$. The its proximal operator

$$\mathbf{prox}_{\lambda f}(\mathbf{A}) = \sum_{i=1}^{n} (\mu_i - \lambda)_+ \mathbf{u}_i\mathbf{v}_i^T, \tag{3.18}$$

where $\mathbf{A} = \sum_{i=1}^n \mu_i\mathbf{u}_i\mathbf{v}_i^T$ is the singular value decomposition of $\mathbf{A}$. This operation is called *singular value thresholding* since we soft threshold the singular values rather than matrix entries.

3.1 CCSP and Sep-CCSP Problems

The generic **convex-concave saddle point (CCSP)** problem is written as

$$\min_{\mathbf{x}\in\mathbb{R}^{D_\mathbf{x}}} \max_{\mathbf{y}\in\mathbb{R}^{D_\mathbf{y}}} \{L(\mathbf{x},\mathbf{y}) = f(\mathbf{x}) + \langle \mathbf{x}, \mathbf{K}\mathbf{y}\rangle - g^*(\mathbf{y})\}, \tag{3.19}$$

where we refer $\mathbf{x} \in \mathbb{R}^{D_\mathbf{x}}$ as primal variables and $\mathbf{y} \in \mathbb{R}^{D_\mathbf{y}}$ as dual variables. $f(\mathbf{x})$ is a proper convex function, $g^*(\cdot)$ is the convex conjugate of a convex function $g(\cdot)$, and matrix $\mathbf{K} \in \mathbb{R}^{D_\mathbf{x}\times D_\mathbf{y}}$. as Figure 3.3 depicts the CCSP problem in a two-dimensional case.

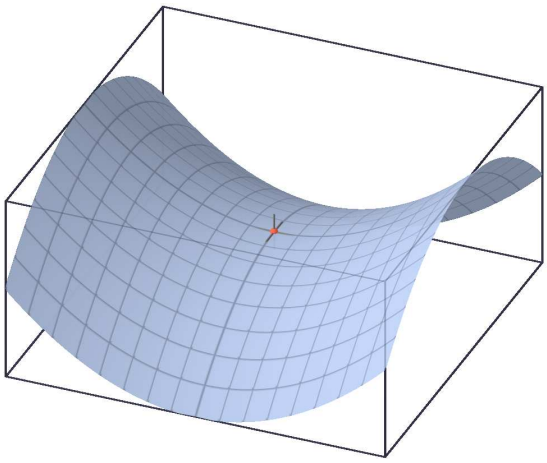


Figure 3.3: Illustration of Convex-Concave Saddle Point problem in a two-dimensional case. The red point is the saddle point $(\mathbf{x}^\star, \mathbf{y}^\star)$.

Many tasks in machine learning, computer vision and game theory, reduce to solving a problem of this form (Jacob et al., 2009; Hastie et al., 2009; Chambolle and Pock, 2011, 2014). As a result, this saddle problem has been widely studied (Zhu and Chan, 2008; Tseng, 2008; Esser et al., 2010; Chambolle and Pock, 2011; He and Yuan, 2012; He and Monteiro, 2014; Chambolle and Pock, 2014).

One important subclass of CCSP problem in (3.19) is where $f(\mathbf{x})$ and/or $g^*(\mathbf{y})$ exhibits an *block coordinate-wise separable structure*. We say $g^*(\mathbf{y})$ is *separable* when

$$g^*(\mathbf{y}) = \sum_{i=1}^{q} g_i^*(\mathbf{y}_i), \text{ with } \mathbf{y}_i \in \mathbb{R}^{D_{\mathbf{y}_i}}, \text{ and } \sum_{i=1}^{q} D_{\mathbf{y}_i} = D_\mathbf{y}. \tag{3.20}$$

Separability of $f(\mathbf{x})$ is defined likewise. We can also partition matrix $\mathbf{K}$ into q column blocks $\mathbf{K}_i \in \mathbb{R}^{D_\mathbf{x}\times D_{\mathbf{y}_i}}$, for $i = 1,\ldots,q$, and therefore,

$$\mathbf{K}\mathbf{y} = \sum_{i=1}^{q} \mathbf{K}_i \mathbf{y}_i, \tag{3.21}$$

resulting in the following problem

$$\min_{\mathbf{x}\in\mathbb{R}^{D_\mathbf{x}}}\max_{\mathbf{y}\in\mathbb{R}^{D_\mathbf{y}}}\left\{L(\mathbf{x},\mathbf{y})=f(\mathbf{x})+\sum_{i=1}^{q}\left(\langle\mathbf{x},\mathbf{K}_i\mathbf{y}_i\rangle-g^*(\mathbf{y}_i)\right)\right\}. \tag{3.22}$$

Or when $f(\mathbf{x})$ is separable,

$$\min_{\mathbf{x}\in\mathbb{R}^{D_\mathbf{x}}}\max_{\mathbf{y}\in\mathbb{R}^{D_\mathbf{y}}}\left\{L(\mathbf{x},\mathbf{y})=\sum_{j=1}^{q}\left(f(\mathbf{x}_j)+\left\langle\mathbf{y},\mathbf{K}_j^T\mathbf{x}_i\right\rangle\right)-g^*(\mathbf{y})\right\}. \tag{3.23}$$

We refer the problems in the form of (3.22) or (3.23) as **Sep**arable **C**onvex-**C**oncave **S**addle **P**oint (**Sep-CCSP**) problems. Although the problem (3.22) and (3.23) can be transformed between each other by changing role of primal and dual variable, in this chapter we still keep the role of primal and dual variable intact and write Sep-CCSP problem into two forms with separable primal and dual variables, respectively.

Sep-CCSP is a general form for many machine learning models. Now we introduce two important instantiations of the Sep-CCSP problem.

Instantiation 1: Separable function minimization with linear constraints. This optimization problem takes the form

$$\begin{aligned}&\min_{\mathbf{x}}\sum_{j=1}^{q}f_j(\mathbf{x}_j)\\&\text{s.t.}\sum_{j=1}^{q}\mathbf{A}_j\mathbf{x}_j=\mathbf{b},\end{aligned} \tag{3.24}$$

where $\mathbf{x}=[\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_q]^T$. After introducing the Lagrangian multiplier $\mathbf{y}$ for the linear constraint, we can reformulate the problem into the following Sep-CCSP problem with separable primal variables,

$$\min_{\mathbf{x}}\max_{\mathbf{y}}\left\{L(\mathbf{x},\mathbf{y})=\sum_{j=1}^{q}f_j(\mathbf{x}_j)+\left\langle\mathbf{y},\sum_{j=1}^{q}\mathbf{A}_j\mathbf{x}_j\right\rangle-\mathbf{y}^T\mathbf{b}\right\}. \tag{3.25}$$

A large number of machine learning problems can be cast as linearly constrained optimization problems of this form (Bertsekas and Tsitsiklis, 1989; Chen et al., 2001; Boyd et al., 2011; Chandrasekaran et al., 2012a; Wainwright, 2014), for instance, robust principal component analysis (RPCA, Wright et al. (2009); Candès et al. (2011)), and overlapping group Lasso problem (Zhao et al., 2009; Jacob et al., 2009).

Instantiation 2: Empirical risk minimization (ERM). ERM (Hastie et al., 2009)) of linear predictors with a convex regularization function $f(\mathbf{x})$:

$$\min_{\mathbf{x}\in\mathbb{R}^d}\left\{J(\mathbf{x})=\frac{1}{N}\sum_{i=1}^{N}\phi_i(\mathbf{a}_i^T\mathbf{x})+f(\mathbf{x})\right\}, \tag{3.26}$$

where $\mathbf{a}_1,\ldots,\mathbf{a}_n\in\mathbb{R}^D$ are the feature vectors of N data samples, $\phi_i(\cdot)$ corresponds the convex loss function w.r.t. the linear predictor $\mathbf{a}_i^T\mathbf{x}$. Many practical classification and regression models fall into this regularized ERM formulation, such as linear support vector machine (SVM), regularized logistic regression and ridge regression, see Hastie et al. (2009) for more details.

Reformulating the above regularized ERM by employing conjugate dual of the function $\phi_i(\cdot)$, i.e.

$$\phi_i(\mathbf{a}_i^T\mathbf{x})=\max_{y_i\in\mathbb{R}}\langle\mathbf{x},y_i\mathbf{a}_i\rangle-\phi_i^*(y_i), \tag{3.27}$$

leads directly to the following Sep-CCSP problem

$$\min_{\mathbf{x}\in\mathbb{R}^D}\max_{\mathbf{y}\in\mathbb{R}^N} f(\mathbf{x})+\frac{1}{n}\sum_{i=1}^{n}\left(\langle\mathbf{x},y_i\mathbf{a}_i\rangle-\phi_i^*(y_i)\right). \tag{3.28}$$

Comparing with the general form, we note that the matrix $\mathbf{K}_i$ in (3.22) is now a vector $\mathbf{a}_i$.

3.2 Primal-Dual Framework for CCSP and Sep-CCSP

The CCSP problem (3.19) has been investigated by several works (Zhu and Chan, 2008; Tseng, 2008; Esser et al., 2010; Chambolle and Pock, 2011; He and Yuan, 2012; He and Monteiro, 2014; Chambolle and Pock, 2014). The basic first-order primal-dual framework was formalized by Chambolle and Pock (2011). We refer this algorithm as PDCP. The key idea behind PDCP is to alternatively optimize with respect to primal and dual variables employing proximal algorithms, which can be summarized as follows.

With certain initialization $(\mathbf{x}^0,\mathbf{y}^0)$ and $\overline{\mathbf{x}}^0=\mathbf{x}^0$. Then update of PDCP in the $(t+1)$th iteration is as follows:

$$\mathbf{y}^{t+1}=\mathrm{argmin}_{\mathbf{y}}g^*(\mathbf{y})-\left\langle\overline{\mathbf{x}}^t,\mathbf{K}\mathbf{y}\right\rangle+\frac{1}{2\sigma}\|\mathbf{y}-\mathbf{y}^t\|_2^2 \tag{3.29}$$

$$\mathbf{x}^{t+1}=\mathrm{argmin}_{\mathbf{x}}f(\mathbf{x})+\left\langle\mathbf{x},\mathbf{K}\mathbf{y}^{t+1}\right\rangle+\frac{1}{2\tau}\|\mathbf{x}-\mathbf{x}^t\|_2^2 \tag{3.30}$$

$$\overline{\mathbf{x}}^{t+1}=\mathbf{x}^{t+1}+\theta(\mathbf{x}^{t+1}-\mathbf{x}^t), \tag{3.31}$$

where the subproblem (3.29) and (3.30) is exactly the proximal operator with respect to function $g^*(\mathbf{y}) - \langle \overline{\mathbf{x}}^t, \mathbf{K}\mathbf{y} \rangle$ and $f(\mathbf{x}) + \left\langle \mathbf{x}, \mathbf{K}\mathbf{y}^{t+1} \right\rangle$, with parameter σ and τ (i.e., the step sizes of primal and dual optimization). Eq. (3.31) is an extrapolation from $\mathbf{x}^t$ to $\mathbf{x}^{t+1}$, which is similar to Nesterov's acceleration technique, see Section 2.2 of Nesterov (2004) and Su et al. (2014).

When the parameter configuration satisfies $\tau\sigma \leq 1/\|\mathbf{K}\|^2$ and $\theta = 1$, PDCP could achieve $O(1/T)$ convergence rate for general convex function $f(\cdot)$ and $g^*(\cdot)$, where T is total number of iterations. When $f(\cdot)$ and $g^*(\cdot)$ are both strongly convex, a linear convergence rate can be achieved by adjusting stepsizes in each iteration.

PDCP is a batch method and is non-stochastic, i.e., it has to update all the dual coordinates in each iteration. When solving the Sep-CCSP problems with a number of coordinate blocks q (say q has a magnitude 10^5 or more), PDCP will be computationally intensive since it has to evaluate the proximal operators for all the coordinate blocks in each iteration.

3.2.1 Scalable Methods for Large-Scale Sep-CCSP

The key issue for large-scale Sep-CCSP problems is the computational bottleneck to handle a large number of coordinate blocks. Fortunately, current active research on coordinate descent methods (CD, Nesterov (2012a,b); Richtárik and Takáč (2012, 2014)) motivates developing scalable methods for large-scale Sep-CCSP problems. The basic idea of CD methods is simple but rather efficient and effective; one or several coordinates are updated in each iteration. In these works mentioned above, (stochastic) coordinate descent methods demonstrate their attractive benefits for efficiently solving large-scale and/or high-dimensional problems and are amenable to parallel optimization.

Due to the similarity between the problems addressed by those works and the Sep-CCSP problem, stochastic coordinate descent could be adopted as a key strategy for Sep-CCSP problems. For more details of coordinate descent methods, we suggest readers to see a comprehensive review by Wright (2015).

One representative work for handling large-scale Sep-CCSP problems is Stochastic Primal-Dual Coordinate Descent (SPDC) (Zhang and Xiao, 2015), which can be viewed as a stochastic variant of the batch method PDCP. SPDC adopts the stochastic coordinate descent into the PDCP, which updates a random subset of coordinates in each iteration to reduce the computation. However, SPDC has several limitations,

1. SPDC uses a conservative constant step size for primal and dual updates, which limits its convergence performance both theoretically and empirically;

2. SPDC assumes that both of $f(\mathbf{x})$ and $g^*(\mathbf{y})$ are *strongly convex*, and only can be applied to the problems similar to regularized ERM (3.28) with Lipschitz smooth loss functions (since the conjugate dual of Lipschitz smooth loss functions are strongly convex) and strongly convex regularization term;

3. SPDC only implements coordinate descent, and it is not clear how to implement block coordinate descent.

In this chapter, we develop two scalable methods to overcome SPDC's limitations for large-scale Sep-CCSP problems. The two proposed methods focus on different aspects of SPDC's limitations and introduce series of novel rules of updates.

One is named as **Ada**ptive **SPDC** (**AdaSPDC**) for Sep-CCSP problems with strongly convex functions (particularly for ERM problems with strongly convex functions), which is a non-trivial extension of SPDC. By carefully exploiting the structure of individual subproblem, we propose an adaptive (i.e., larger data-dependent) step size rule for both primal and dual updates according to the chosen subset of coordinate blocks in each iteration. Both theoretically and empirically, we show that AdaSPDC could yield a significantly better linear convergence rate than SPDC and other state-of-the-art methods.

Another proposed method, referred as **S**tochastic **P**arallel **B**lock **C**oordinate **D**escent (**SP-BCD**), particularly, deals with large-scale Sep-CCSP with general (*non-strongly convex*) functions. SP-BCD covers a wider range of machine learning applications than SPDC and AdaSPDC, since its capability of solving non-strongly convex functions enables it can be applied to separable function minimization problem with linear constraints 3.25. SP-BCD also exploits the structure of the matrix $\mathbf{K}$, and suggests a novel rule of step sizes, which can achieve better sublinear convergence rate without any strong convexity assumption. Various applications in the field of machine learning and economics are experimented that demonstrate SP-BCD's efficacy and efficiency.

In the following, we elaborate both of the two newly-established methods.

3.3 Adaptive Stochastic Primal-Dual Coordinate Descent

Since the proposed AdaSPDC mainly focuses on solving regularized ERM problems (3.28), we present a generalized version of problem (3.28),

$$\min_{\mathbf{x}\in\mathbb{R}^D}\max_{\mathbf{y}\in\mathbb{R}^N}\left\{L(\mathbf{x},\mathbf{y})=f(\mathbf{x})+\frac{1}{N}\sum_{i=1}^{N}\left(\langle\mathbf{x},\mathbf{A}_i\mathbf{y}_i\rangle-\phi_i^*(\mathbf{y}_i)\right)\right\}, \tag{3.32}$$

where the coordinates $\{y_i\}_{i=1}^N$ are generalized to block coordinates $\{\mathbf{y}_i\}_{i=1}^N$, and the feature vectors $\{\mathbf{a}_i\}_{i=1}^N$ are generalized into feature matrices $\{\mathbf{A}_i\}_{i=1}^N$.

We further assume that each $\phi_i^*(\mathbf{y}_i)$ is γ-strongly convex, and $f(\mathbf{x})$ is λ-strongly convex, i.e.,

$$\phi_i^*(\mathbf{y}_i')\geq\phi_i^*(\mathbf{y}_i)+\nabla\phi^*(\mathbf{y}_i)^T\left(\mathbf{y}_i'-\mathbf{y}_i\right)+\frac{\gamma}{2}\|\mathbf{y}_i'-\mathbf{y}_i\|_2^2,\quad\forall\mathbf{y}_i,\mathbf{y}_i'\in\mathbb{R}^{n_i}$$
$$f(\mathbf{x}')\geq f(\mathbf{x})+\nabla f(\mathbf{x})^T\left(\mathbf{x}'-\mathbf{x}\right)+\frac{\lambda}{2}\|\mathbf{x}_i'-\mathbf{x}_i\|_2^2,\quad\forall\mathbf{x},\mathbf{x}'\in\mathbb{R}^D.$$

Now we introduce AdaSPDC based on this Sep-CCSP form.

As a non-trivial extension of SPDC (Zhang and Xiao, 2015), our method AdaSPDC solves the Sep-CCSP problem by using an adaptive parameter configuration. Concretely, we optimize $L(\mathbf{x},\mathbf{y})$ in Eq. (3.32) by alternatively updating the dual and primal variables in a principled way. Thanks to the separable structure of $\phi^*(\mathbf{y})$, in each iteration we can randomly select m blocks of dual variables whose indices are denoted as S_t, and then we only update these selected blocks in the following way,

$$\mathbf{y}_i^{t+1}=\mathrm{argmin}_{\mathbf{y}_i}\left[\phi_i(\mathbf{y}_i)-\left\langle\overline{\mathbf{x}}^t,\mathbf{A}_i\mathbf{y}_i\right\rangle+\frac{1}{2\sigma_i}\|\mathbf{y}_i-\mathbf{y}_i^t\|_2^2\right],\ \text{if } i\in S_t. \tag{3.33}$$

For those coordinates in blocks not selected, $i\notin S_t$, we just keep $\mathbf{y}_i^{t+1}=\mathbf{y}_i^t$. By exploiting the structure of individual $\mathbf{A}_i$, we configure the step size parameter of the proximal term σ_i *adaptively*

$$\sigma_i=\frac{1}{2R_i}\sqrt{\frac{N\lambda}{m\gamma}}, \tag{3.34}$$

where $R_i=\|\mathbf{A}_i\|_2=\sqrt{\mu_{\max}\left(\mathbf{A}_i^T\mathbf{A}_i\right)}$, with $\|\cdot\|_2$ is the spectral norm of a matrix and $\mu_{\max}(\cdot)$ to denote the maximum singular value of a matrix.

Our step size is different from the one used in SPDC (Zhang and Xiao, 2015), where R is a constant $R=\max\{\|\mathbf{a}_i\|_2:i=1,\ldots,N\}$ (since SPDC only considers ERM problem, the matrix $\mathbf{A}_i$ is a feature vector $\mathbf{a}_i$).

Remark. Intuitively, R_i in AdaSPDC can be understood as the coupling strength between the i-th dual variable block and primal variable, measured by the spectral

norm of matrix $\mathbf{A}_i$. Smaller coupling strength allows us to use a larger step size for the current dual variable block without caring too much about its influence on primal variable, and vice versa. Compared with SPDC, our proposal of an adaptive coupling strength for the chosen coordinate block directly results in a larger step size, and thus helps to improve convergence speed.

In the stochastic dual update, we also use an intermediate variable $\overline{\mathbf{x}}^t$ as Eq. (3.31) in PDCP algorithm , and we will describe its update later.

Since we assume $f(\mathbf{x})$ is not separable, we update the primal variable as a whole,

$$\mathbf{x}^{t+1} = \operatorname{argmin}_{\mathbf{x}} \left[f(\mathbf{x}) + \left\langle \mathbf{x}, \mathbf{r}^t + \frac{1}{m} \sum_{j \in S_t} \mathbf{A}_j (\mathbf{y}_j^{t+1} - \mathbf{y}_j^t) \right\rangle + \frac{1}{2\tau^t} \|\mathbf{x} - \mathbf{x}^t\|_2^2 \right]. \quad (3.35)$$

The proximal parameter τ^t is also configured *adaptively*,

$$\tau^t = \frac{1}{2R_{\max}^t} \sqrt{\frac{m\gamma}{N\lambda}}, \quad (3.36)$$

where $R_{\max}^t = \max\{R_i | i \in S_t\}$, compared with constant R used in SPDC. To account for the incremental change after the latest dual update, an auxiliary variable $\mathbf{r}^t = \frac{1}{N}\sum_{i=1}^N \mathbf{A}_i \mathbf{y}_i^t$ is used and updated as follows

$$\mathbf{r}^{t+1} = \mathbf{r}^t + \frac{1}{N} \sum_{j \in S_t} \mathbf{A}_j \left(\mathbf{y}_j^{t+1} - \mathbf{y}_j^t \right). \quad (3.37)$$

Finally, we update the intermediate variable $\overline{\mathbf{x}}$, which implements an extrapolation step over the current $\mathbf{x}^{t+1}$ and can help to provide faster convergence rate (Nesterov, 2004; Chambolle and Pock, 2011).

$$\overline{\mathbf{x}}^{t+1} = \mathbf{x}^{t+1} + \theta^t (\mathbf{x}^{t+1} - \mathbf{x}^t), \quad (3.38)$$

where θ^t is configured adaptively as

$$\theta^t = 1 - \frac{1}{N/m + R_{\max}^t \sqrt{(N/m)/(\lambda\gamma)}}, \quad (3.39)$$

which is much smaller than the constant θ used in SPDC, $\theta_{\text{SPDC}} = 1 - \frac{1}{N/m + R\sqrt{(N/m)/(\lambda\gamma)}}$.

The whole procedure for solving Sep-CCSP problem (3.32) using AdaSPDC is summarized in Algorithm 3.1. There are several notable characteristics of AdaSPDC.

- Compared with SPDC, our method uses adaptive step size to obtain faster convergence (will be shown in Theorem 1), while the whole algorithm does not bring any other extra computational complexity. As demonstrated in the experiment Section 3.3.3, in many cases, AdaSPDC provides significantly better performance than SPDC.

Algorithm 3.1 AdaSPDC for Sep-CCSP problem (3.32)

Input: number of blocks picked in each iteration m and number of iterations T.
Initialize: $\mathbf{x}^0$, $\mathbf{y}^0$, $\overline{\mathbf{x}}^0 = \mathbf{x}^0$, $\mathbf{r}^0 = \frac{1}{N}\sum_{i=1}^{N} \mathbf{A}_i \mathbf{y}_i^0$
for $t = 0, 1, \ldots, T-1$ **do**
    Randomly pick a subset with size m from all the N coordinate blocks, denoted as S_t.
    According to the selected subset S_t, compute the adaptive parameter configuration of σ_i, τ^t and θ^t using Eq. (3.34), (3.36) and (3.39), respectively.
    for each selected block in parallel **do**
        Update the dual variable block using Eq.(3.33).
    end for
    Update primal variable using Eq.(3.35).
    Extrapolate primal variable block using Eq.(3.38).
    Update the auxiliary variable $\mathbf{r}$ using Eq.(3.37).
end for

- Since, in each iteration, a number of block coordinates can be chosen and updated independently (with independent evaluation of individual step size), this directly enables parallel processing, and hence use on modern computing clusters. The ability to select an arbitrary number of blocks can help to make use of all the computation structure available as effectively as possible. The property of arbitrary number of blocks selection was previously studied in detail by Richtárik and Takáč (2012); **?**.

3.3.1 Convergence Analysis for AdaSPDC

We characterise the convergence performance of AdaSPDC in the following theorem.

Theorem 1. *Assume that each $\phi_i^*(\cdot)$ is γ-strongly convex, and $g(\cdot)$ is λ-strongly convex, and given the parameter configuration in Eq. (3.34), (3.36) and (3.39), then after T iterations in Algorithm 3.1, AdaSPDC achieves the following convergence performance*

$$\begin{aligned} &\mathbb{E}\left[\left(\frac{1}{2\tau^T}+\lambda\right)\|\mathbf{x}^T-\mathbf{x}^\star\|_2^2\right]+\mathbb{E}\left[\|\mathbf{y}^T-\mathbf{y}^\star\|_{\mathbf{v}}^2\right] \\ &\leq\left(\prod_{t=1}^{T}\mathbb{E}[\theta^t]\right)\left(\left(\frac{1}{2\tau^0}+\lambda\right)\|\mathbf{x}^0-\mathbf{x}^\star\|_2^2+\|\mathbf{y}^0-\mathbf{y}^\star\|_{\mathbf{v}'}^2\right), \end{aligned} \tag{3.40}$$

where $(\mathbf{x}^\star, \mathbf{y}^\star)$ *is the optimal saddle point,* $\nu_i = \frac{1}{4m\sigma_i} + \frac{\gamma}{2m}$, $\nu_i' = \frac{1}{2m\sigma_i} + \frac{\gamma}{2m}$, *and* $\|\mathbf{y}^T - \mathbf{y}^\star\|_{\mathbf{v}}^2 = \sum_{i=1}^N \nu_i \|\mathbf{y}_i^T - \mathbf{y}_i^\star\|_2^2$.

Compared with the convergence rate of SPDC by Zhang and Xiao (2015), AdaSPDC achieves a much sharper rate since the term $\prod_{t=1}^T \mathbb{E}[\theta^t]$ will be much smaller than $\prod_{t=1}^T \theta_{\text{SPDC}}$. This is due to the usage of data-dependent stepsizes during each iteration.

Since the proof of the above is technical, we provide it in the Appendix B.

In our proof, given the proposed parameter θ^t, the critical point for obtaining a sharper linear convergence rate than SPDC is that we configure τ^t and σ_i as Eq. (3.36) and (3.34) to guarantee the positive definiteness of the following matrix in the t-th iteration,

$$\mathbf{P} = \begin{bmatrix} \frac{m}{2\tau^t}\mathbf{I} & -\mathbf{A}_{S_t} \\ -\mathbf{A}_{S_t}^T & \frac{1}{2\text{diag}(\boldsymbol{\sigma}_{S_t})} \end{bmatrix}, \tag{3.41}$$

where $\mathbf{A}_{S_t} = [\ldots, \mathbf{A}_i, \ldots] \in \mathbb{R}^{D \times mn_i}$ and $\text{diag}(\boldsymbol{\sigma}_{S_t}) = \text{diag}(\ldots, \sigma_i \mathbf{I}_{n_i}, \ldots)$ for $i \in S_t$. However, we found that the parameter configuration to guarantee the positive definiteness of $\mathbf{P}$ is not unique, and there exist other valid parameter configurations besides the proposed one in this work. We leave the further investigation on other potential parameter configurations as future work.

3.3.2 Further Comparison with SDPC

Compared with SPDC (Zhang and Xiao, 2015), AdaSPDC follows the similar primal-dual framework. The crucial difference between them is that AdaSPDC proposes a larger step size for both dual and primal updates, see Eq. (3.34) and (3.36) compared with SPDC's parameter configuration given in Eq.(10) in Zhang and Xiao (2015), where SPDC applies a large constant $R = \max\{\|\mathbf{a}_i\|_2 : i = 1, \ldots, n\}$ while AdaSPDC uses a more adaptive value of R_i and $R_{\max}^t$ for t-th iteration to account for the different coupling strength between the selected dual coordinate block and primal variable. This difference directly means that AdaSPDC can potentially obtain a significantly sharper linear convergence rate than SPDC, since the decay factor θ^t of AdaSPDC is smaller than θ in SPDC (Eq.(10) in Zhang and Xiao (2015)) , see Theorem 1 for AdaSPDC compared with SPDC (Theorem 1 in Zhang and Xiao (2015)). The empirical performance of the two algorithms will be demonstrated in the experimental Section 3.3.3.

To improve the algorithm performance, the authors of SPDC propose to non-uniformly sample the the dual coordinate to update in each iteration according to the norm of the each $\mathbf{a}_i$. However, as we show later in the empirical experiments, this non-uniform

sampling does not improve the convergence a lot for some datasets. By configuring the data-dependent step size explicitly, our method AdaSPDC provides a better solution for unnormalized data compared with SPDC, see Section 3.3.3 for more empirical evidence. Moreover, our method AdaSPDC can be potentially extended by non-uniformly sampling the dual coordinates, and we will leave this as future work.

Another difference is that SPDC only considers the regularized ERM task, i.e., only handling the case that each $\mathbf{A}_i$ is a feature vector $\mathbf{a}_i$, while AdaSPDC extends that $\mathbf{A}_i$ can be a matrix so that AdaSPDC can cover a wider range of applications than SPDC, i.e. in each iteration, a number of *block* coordinates could be selected while for SPDC only a number of coordinates are allowed.

3.3.3 Empirical Results

In this section, we apply AdaSPDC to several regularized empirical risk minimization problems. The experiments are conducted to compare our method AdaSPDC with other competitive stochastic optimization methods, including (1) **S**tochastic **D**ual **C**oordinate **A**scent (**SDCA**, Shalev-Shwartz and Zhang (2013)), a dual optimization method which stochastically update one coordinate in each iteration; (2) **S**tochastic **A**veraging **G**radient (**SAG**, Schmidt et al. (2013)), a primal optimization method by incorporating a memory of previous gradient values; (3) SPDC with uniform sampling (Zhang and Xiao, 2015) and (4) SPDC with non-uniform sampling (Zhang and Xiao, 2015). In order to provide a fair comparison with these methods, in each iteration only one dual coordinate (or data instance) is chosen, i.e., we run all the methods sequentially. To obtain results that are independent of the practical implementation of the algorithm, we measure the algorithm performance in term of objective suboptimality w.r.t. the effective passes to the entire data set.

Each experiment is run 10 times and the average results are reported to show statistical consistency. We present all the experimental results we have done for each application.

3.3.3.1 Ridge Regression

We firstly apply our method AdaSPDC into a simple ridge regression problem with synthetic data. The data is generated in the same way as Zhang and Xiao (2015); $N = 1000$ i.i.d. training points $\{\mathbf{a}_i, b_i\}_{i=1}^n$ are generated in the following manner,

$$b = \mathbf{a}^T \mathbf{x}^\diamond + \varepsilon, \quad \mathbf{a} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \quad \varepsilon \sim \mathcal{N}(0,1),$$

where $\mathbf{a} \in \mathbb{R}^D$ and $D = 1000$, and the elements of the vector $\mathbf{x}^\diamond$ are all ones. The covariance matrix $\mathbf{\Sigma}$ is set to be diagonal with $\Sigma_{jj} = j^{-2}$, for $j = 1, \ldots, D$. Then the ridge regression tries to solve the following optimization problem,

$$\min_{\mathbf{x}\in\mathbb{R}^D} \left\{ J(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2}(\mathbf{a}_i^T\mathbf{x} - b_i)^2 + \frac{\lambda}{2}\|\mathbf{x}\|_2^2 \right\}. \tag{3.42}$$

The optimal solution of the above ridge regression can be found as

$$\mathbf{x}^\star = \left(\mathbf{A}\mathbf{A}^T + n\lambda\mathbf{I}_D\right)^{-1}\mathbf{A}\mathbf{b}.$$

By employing the conjugate dual of quadratic loss (crossref, Eq. (3.27) and Table 3.1), we can reformulate the ridge regression as the following Sep-CCSP problem,

$$\min_{\mathbf{x}\in\mathbb{R}^D}\max_{\mathbf{y}\in\mathbb{R}^N} \frac{\lambda}{2}\|\mathbf{x}\|_2^2 + \frac{1}{N}\sum_{i=1}^{N}\left(\langle \mathbf{x}, y_i\mathbf{a}_i\rangle - \left(\frac{1}{2}y_i^2 + b_i y_i\right)\right). \tag{3.43}$$

It is easy to figure out that $f(\mathbf{x}) = \lambda/2\|\mathbf{x}\|_2^2$ is λ-strongly convex, and $\phi_i^*(y_i) = \frac{1}{2}y_i^2 + b_i y_i$ is 1-strongly convex.

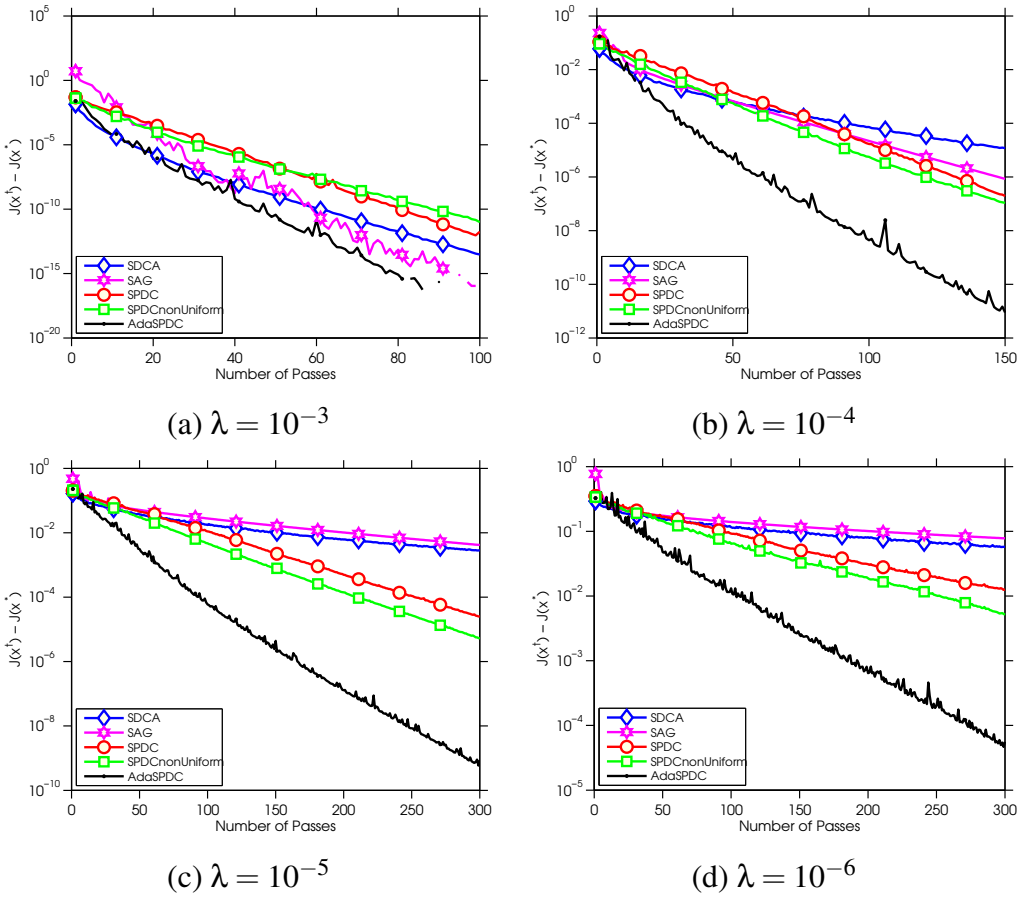
Thus, for ridge regression, the dual update in Eq. (3.33) and primal update in Eq. (3.35) of AdaSPDC have closed form solutions as below,

$$y_i^{t+1} = \frac{1}{1+1/\sigma_i}\left(\langle \overline{\mathbf{x}}^t, \mathbf{a}_i\rangle + b_i + \frac{1}{\sigma_i}y_i\right), \text{ if } i \in S_t$$
$$\mathbf{x}^{t+1} = \frac{1}{\lambda + 1/\tau^t}\left(\frac{1}{\tau^t}\mathbf{x}^t - \left(\mathbf{r}^t + \frac{1}{m}\sum_{j\in S_t}\mathbf{a}_j(y_j^{t+1} - y_j^t)\right)\right)$$

The algorithm performance is evaluated in term of objective suboptimality (measured by $J(\mathbf{x}^t) - J(\mathbf{x}^\star)$) w.r.t. number of effective passes to the entire datasets. Varying values of regularization parameter λ are experimented to demonstrate algorithm performance with different degree of ill-conditioning, $\lambda = \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$.

Figure 3.4 shows algorithm performance with different degrees of regularization. It is easy to observe that AdaSPDC converges substantially faster than other compared methods, particularly for less regularized (i.e. smaller λ) problems. Compared with SPDC and its variant with non-uniform sampling, the usage of adaptive step size in AdaSPDC significantly improves convergence speed. For instance, in the case with $\lambda = 10^{-6}$, AdaSPDC achieves 100 times better suboptimality than both SPDC and its variant SPDC with non-uniform sampling after 300 passes.

Note that the SDCA we implemented is the original version, and recently some extended versions of SDCA were developed and expected to have better performance, see **??**.



(a) $\lambda = 10^{-3}$ (b) $\lambda = 10^{-4}$

(c) $\lambda = 10^{-5}$ (d) $\lambda = 10^{-6}$

Figure 3.4: Ridge regression with synthetic data: comparison of convergence performance w.r.t. the number of passes. Problem size: $D = 1000, N = 1000$. We evaluate the convergence performance using objective suboptimality, $J(\mathbf{x}^t) - J(\mathbf{x}^\star)$.

Table 3.2: Benchmark datasets used in our experiments for binary classification.

Datasets	Number of samples N	Number of features D	Sparsity
w8a	49,749	300	3.9%
covertype	20,242	47,236	0.16%
url	2,396,130	3,231,961	0.0018%
quantum	50,000	78	43.44%
protein	145,751	74	99.21%

3.3.3.2 Binary Classification on Real-world Datasets

We now compare the performance of our method AdaSPDC with other competitive methods on several real-world data sets. Our experiments focus on the freely-available benchmark data sets for binary classification, whose detailed information are listed in

Table 4.2. The *w8a*, *covertype* and *url* data are obtained from the LIBSVM collection[1]. The *quantum* and *protein* data sets are obtained from KDD Cup 2004[2]. For all the datasets, each sample takes the form $(\mathbf{a}_i, b_i)$ with $\mathbf{a}_i$ is the feature vector and b_i is the binary label -1 or 1. We add a bias term to the feature vector for all the datasets. We aim to minimize the regularized empirical risk with following form

$$J(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\phi_i(\mathbf{a}_i^T\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{x}\|_2^2 \tag{3.44}$$

To provide a more comprehensive comparison between these methods, we experiment with two different loss function $\phi_i(\cdot)$, smooth Hinge loss (Shalev-Shwartz and Zhang, 2013) and logistic loss, described in the following.

3.3.3.2.1 Smooth Hinge loss

(with smoothing parameter $\gamma = 1$.)

$$\phi_i(z) = \begin{cases} 0 & \text{if } b_i z \geq 1, \\ 1 - \frac{\gamma}{2} - b_i z & \text{if } b_i z \leq 1-\gamma \\ \frac{1}{2\gamma}(1-b_i z)^2 & \text{otherwise.} \end{cases}$$

And its conjugate dual is

$$\phi_i^*(y_i) = b_i y_i + \frac{1}{2}y_i^2, \text{ with } b_i y_i \in [-1, 0].$$

We can observe that $\phi_i^*(y_i)$ is γ-strongly convex with $\gamma = 1$. The dual update of AdaSPDC for smooth Hinge loss is nearly the same with ridge regression except the necessity of projection into the interval $b_i y_i \in [-1, 0]$.

3.3.3.2.2 Logistic loss

$$\phi_i(z) = \log\left(1 + \exp(-b_i z)\right),$$

whose conjugate dual has the form

$$\phi_i^*(y_i) = -b_i y_i \log(-b_i y_i) + (1 + b_i y_i)\log(1 + b_i y_i) \text{ with } b_i y_i \in [-1, 0].$$

It is also easy to obtain that $\phi_i^*(y_i)$ is γ-strongly convex with $\gamma = 4$. Note that for logistic loss, the dual update in Eq. (3.33) does not have a closed form solution, and we can start from some initial solution and further apply several steps of Newton's update to obtain a more accurate solution.

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`
[2] `http://osmot.cs.cornell.edu/kddcup/datasets.html`

During the experiments, we observe that the performance of SAG is very sensitive to the choice of step size . To obtain best results of SAG, we try different candidates of step size in the interval $[1/16L, 1/L]$ and report the best result for each dataset, where L is Lipschitz constant of $\phi_i(\mathbf{a}_i^T\mathbf{x})$, $1/16L$ is the theoretical step size choice for SAG and $1/L$ is the suggested empirical choice by (Schmidt et al., 2013). For smooth Hinge loss, $L = \max_i\{\|\mathbf{a}_i\|_2, i = 1,\ldots,N\}$, and for logistic loss, $L = \frac{1}{4}\max_i\{\|\mathbf{a}_i\|_2, i = 1,\ldots,N\}$.

Figure 3.5 and Figure 3.6 depict the algorithm performance on the different methods with smooth Hinge loss and logistics loss, respectively. We compare all these methods with different values of $\lambda = \{10^{-5}, 10^{-6}, 10^{-7}\}$. Generally, our method AdaSPDC performs consistently better or at least comparably with other methods, and performs especially well for the tasks with small regularized parameter λ. For some datasets, such as covertype and quantum, SPDC with non-uniform sampling decreases the objective faster than other methods in early epochs, however, cannot achieve comparable results with other methods in later epochs, which might be caused by its conservative step size.

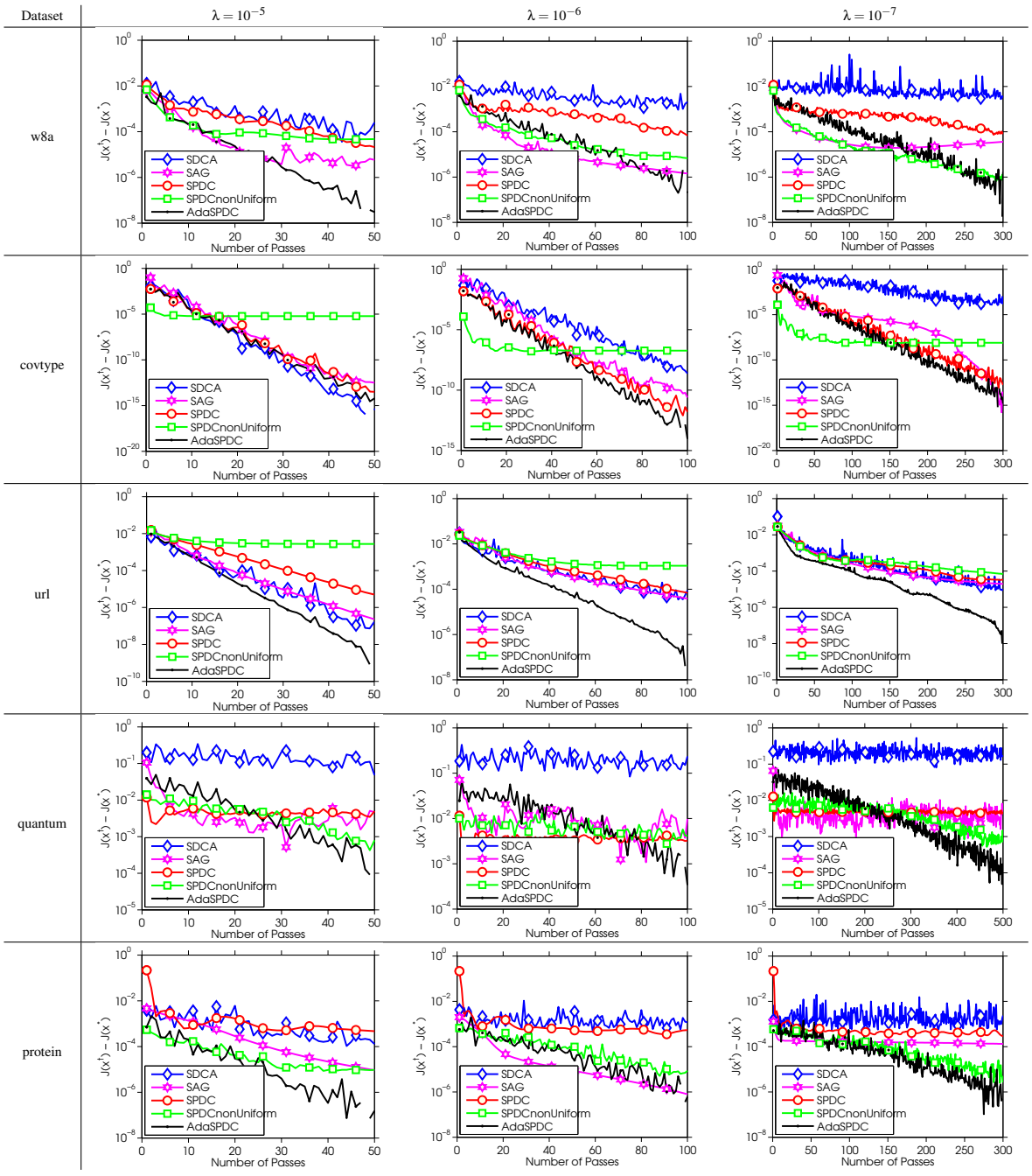


Figure 3.5: Comparison of algorithm performance with smooth Hinge loss.

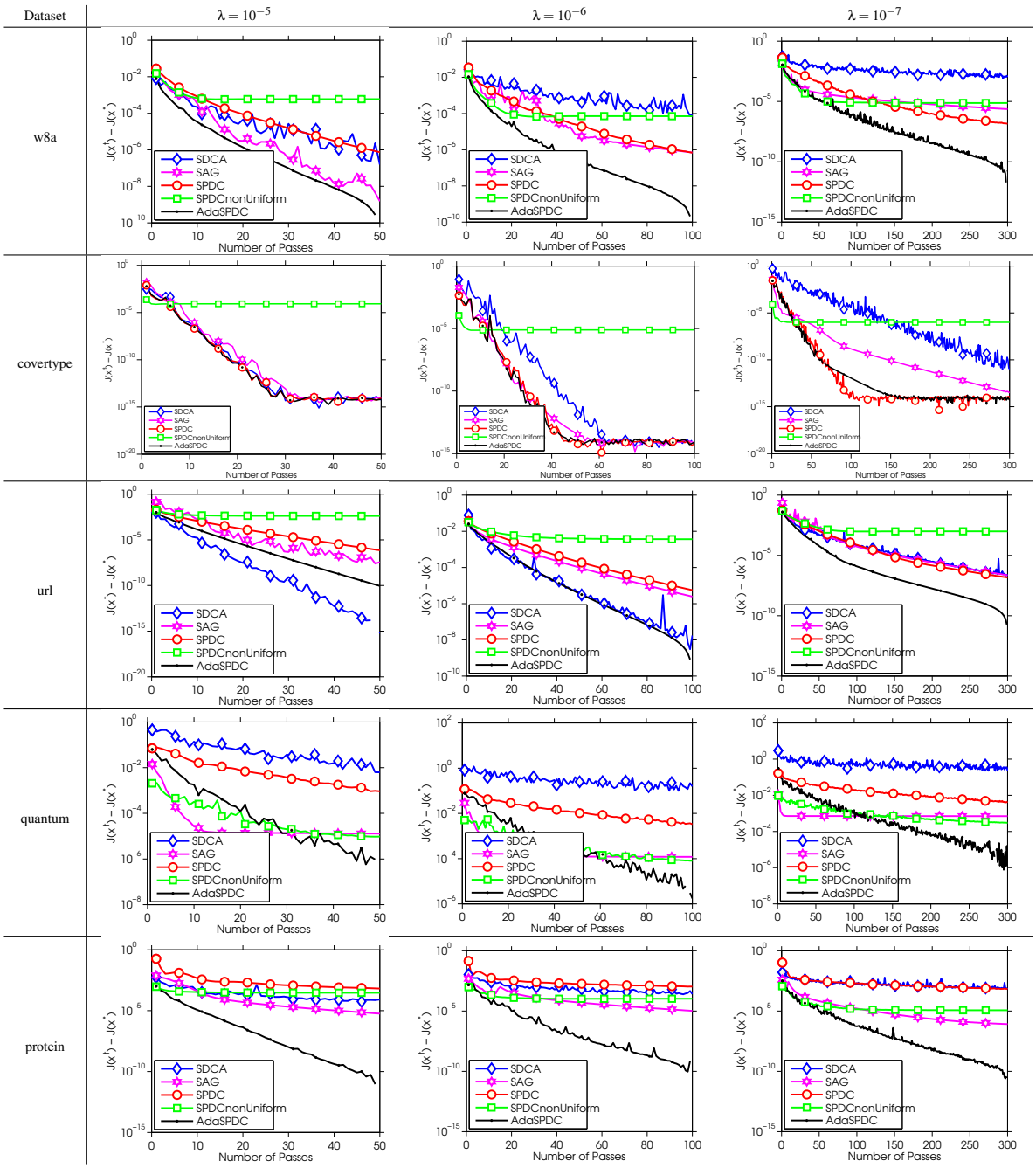


Figure 3.6: Comparison of algorithm performance with Logistic loss.

Interim Summary The proposed Adaptive Stochastic Primal-Dual Coordinate Descent (AdaSPDC) for Sep-CCSP problem is a non-trivial extension of a recent work SPDC (Zhang and Xiao, 2015). AdaSPDC uses an adaptive step size choices for both primal and dual updates in each iteration. The design of the step size for our method AdaSPDC explicitly and adaptively models the coupling strength between chosen block coordinates and primal variable through the spectral norm of each $\mathbf{A}_i$. We theoretically characterise that AdaSPDC holds a sharper linear convergence rate than SDPC. Additionally, we demonstrate the superiority of the proposed AdaSPDC method on ERM problems through extensive experiments on both synthetic and real-world data sets.

3.4 SP-BCD for General Sep-CCSP Problems

In last section, we introduce AdaSDPC and SPDC, which are particularly designed for Sep-CCSP problems with strongly convex functions. This part focuses on Sep-CCSP problems with general (*non-strongly convex*) functions that can be directly be applied to the problems of separable function minimization with linear constraints (3.23) and some ERM problems with non-strongly convex functions.

Considering the explicit forms of non-strongly convex Sep-CCSP problems for various applications, we focus on the following Sep-CCSP form with separable primal variables

$$\min_{\mathbf{x}\in\mathbb{R}^{D_\mathbf{x}}}\max_{\mathbf{y}\in\mathbb{R}^{D_\mathbf{y}}}\left\{L(\mathbf{x},\mathbf{y})=\sum_{j=1}^{q}\left(f(\mathbf{x}_j)+\left\langle\mathbf{y},\mathbf{A}_j\mathbf{x}_j\right\rangle\right)-g^*(\mathbf{y})\right\}, \tag{3.45}$$

where $\mathbf{x}_j\in\mathbb{R}^{D_{\mathbf{x}_j}}$, $\mathbf{A}_j\in\mathbb{R}^{D_\mathbf{y}\times D_{\mathbf{x}_j}}$ for $j=1,\ldots,q$, and $\sum_{j=1}^{q}D_{\mathbf{x}_j}=D_\mathbf{x}$. We do not enforce any strong convexity assumptions to $f_j(\cdot)$ and $g(\cdot)$.

Now we elaborate the novel method **S**tochastic **P**arallel **B**lock **C**oordinate **D**escent (**SP-BCD**) method for solving general (non-strongly convex) Sep-CCSP form (3.45). SP-BCD also incorporates stochastic coordinate descent, which is same as SPDC and AdaSPDC. The key difference is how to choose step size for primal and dual optimization. Concretely, due to the separable structure of $f(\mathbf{x})$, in each iteration we can randomly select m blocks of variables whose indices are denoted as S_t, and then we only update these selected blocks, given the current $y=y^t$, in the following way. If $j\in S_t$ then

$$\mathbf{x}_j^{t+1}=\mathrm{argmin}_{\mathbf{x}_j}f_j(\mathbf{x}_j)+\left\langle\mathbf{y}^t,\mathbf{A}_j\mathbf{x}_j\right\rangle+\frac{1}{2}\|\mathbf{x}_j-\mathbf{x}_j^t\|_{1/\boldsymbol{\tau}_j}^2, \tag{3.46}$$

otherwise, we just keep $\mathbf{x}_j^{t+1}=\mathbf{x}_j^t$. In the blockwise update, we add a proximal term to penalize the deviation from last update $\mathbf{x}_j^t$, i.e.,

$$\frac{1}{2}\|\mathbf{x}_j-\mathbf{x}_j^t\|_{1/\boldsymbol{\tau}_j}^2=\frac{1}{2}(\mathbf{x}_j-\mathbf{x}_j^t)^T\mathrm{diag}(1/\boldsymbol{\tau}_j)(\mathbf{x}_i-\mathbf{x}_j^t), \tag{3.47}$$

where $1/\boldsymbol{\tau}_j$ is a shorthand notation for the inverse of each element of the vector $\boldsymbol{\tau}_j\in\mathbb{R}^{D_{\mathbf{x}_j}}$, the diagonal matrix $\mathrm{diag}(1/\boldsymbol{\tau}_j)$ is applied for scaling each dimension of $\mathbf{x}_j$, and each $\boldsymbol{\tau}_j$ is a subvector of $\boldsymbol{\tau}=\left[\boldsymbol{\tau}_1^T,\ldots,\boldsymbol{\tau}_J^T\right]^T$. We configure the each dimension of $\boldsymbol{\tau}$ as

$$\tau_d=\frac{1}{\sum_{j=1}^{D_\mathbf{y}}|A_{jd}|},\quad d=1,2,\ldots,D_\mathbf{x}. \tag{3.48}$$

Employing the same intuition as AdaSPDC, τ_d in SP-BCD can be interpreted as the coupling strength between the d-th dimension of the primal variable $\mathbf{x}$ and dual variable

$\mathbf{y}$, measured by the L_1 norm of the vector $\mathbf{A}_{:,d}$ (i.e., the d-th column of matrix $\mathbf{A}$). Smaller coupling strength allows us to use smaller proximal penalty (i.e., larger step size) for updating the current primal variable block without caring too much about its influence on dual variable, and vice versa.

Then for those selected block variables, we use extrapolation technique given in Eq.(3.31) (Chambolle and Pock (2011), Chapter 2.2 in Nesterov (2004)) to yield an intermediate variable $\overline{\mathbf{x}}^{t+1}$ as follows,

$$\overline{\mathbf{x}}_j^{t+1} = \begin{cases} \mathbf{x}_j^{t+1} + \theta\left(\mathbf{x}_j^{t+1} - \mathbf{x}_j^t\right) & \text{if } j \in S_t \\ \overline{\mathbf{x}}_j^t & \text{otherwise,} \end{cases} \tag{3.49}$$

where $\theta = m/q$ to account for there being only m blocks out of q selected in each iteration.

Assuming $g^*(\mathbf{y})$ is not separable, we update the dual variable as a whole. A similar proximal term is added with the diagonal matrix $\mathrm{diag}(1/\boldsymbol{\sigma}^t)$:

$$\mathbf{y}^{t+1} = \mathrm{argmin}_{\mathbf{y}} g^*(\mathbf{y}) - \left\langle \mathbf{y}, \overline{\mathbf{r}}^t + \frac{q}{m}\sum_{j\in S_t} \mathbf{A}_j(\overline{\mathbf{x}}_j^{t+1} - \overline{\mathbf{x}}_j^t) \right\rangle + \frac{1}{2}\|\mathbf{y} - \mathbf{y}^t\|^2_{1/\boldsymbol{\sigma}^t}, \tag{3.50}$$

where $\overline{\mathbf{r}}^t = \sum_{j=1}^J \mathbf{A}_j \overline{\mathbf{x}}_j^t$. We configure the dual step size $\boldsymbol{\sigma}^t$ *adaptively* for each iteration,

$$\sigma_k^t = \frac{m}{q} \cdot \frac{1}{\sum_{j\in S_t} |A_{kj}|}, \quad k = 1, 2, \ldots, D_{\mathbf{y}}. \tag{3.51}$$

This configuration adaptively accounts for the coupling strength between the dual variable and the chosen primal variable blocks in S_t through measuring the structure of the matrix $\mathbf{A}$. Later we show that the usage of the proposed adaptive proximal penalty for both primal and dual update contributes to significantly improve the convergence performance for many machine learning applications.

Another crucial component of the dual update is the construction of the term $\overline{\mathbf{r}}^t + \frac{q}{m}\sum_{j\in S_t} \mathbf{A}_j(\overline{\mathbf{x}}_j^{t+1} - \overline{\mathbf{x}}_j^t)$, which is inspired by a recently proposed fast incremental gradient method for non-strongly convex functions, SAGA (Defazio et al., 2014). We use the combination of the cached sum of all $\mathbf{A}_j\overline{\mathbf{x}}_j^t$, i.e., $\overline{\mathbf{r}}^t$, and the newly updated sample average $\frac{1}{m}\sum_{j\in S_t} \mathbf{A}_j(\overline{\mathbf{x}}_j^{t+1} - \overline{\mathbf{x}}_j^t)$ to obtain a variance reduced estimation of $\mathbb{E}[\overline{\mathbf{r}}]$, which is essentially the spirit of SAGA. We refer the reader to see (Defazio et al., 2014, Section 3) for more details. After the dual update, $\overline{\mathbf{r}}^t$ is updated to $\overline{\mathbf{r}}^{t+1}$ using,

$$\overline{\mathbf{r}}^{t+1} = \overline{\mathbf{r}}^t + \sum_{j\in S_t} \mathbf{A}_j\left(\overline{\mathbf{x}}_j^{t+1} - \overline{\mathbf{x}}_j^t\right). \tag{3.52}$$

Algorithm 3.2 SP-BCD for Sep-CCSP problem (3.45)

Input: number of blocks picked in each iteration m, the configuration of parameter $\theta = m/q$, $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}^t$ as given in Eq. (3.48) and (3.51).
Initialize: $\mathbf{x}^0$, $\mathbf{y}^0$, $\bar{\mathbf{x}}^0 = \mathbf{x}^0$, $\bar{\mathbf{r}}^0 = \sum_{j=1}^{q} \mathbf{A}_j \bar{\mathbf{x}}_j^0$
for $t = 1, 2, \ldots, T$ **do**
    Randomly pick a subset with size m from all the q coordinate blocks, denoted as S_t.
    for each block in parallel **do**
        Update each primal variable block using Eq.(3.46).
        Extrapolate primal variable block using Eq.(3.49).
    end for
    Update dual variable using Eq.(3.50)
    Update $\bar{\mathbf{r}}^{t+1}$ using Eq.(3.52)
end for

The whole procedure for solving Sep-CCSP problem (3.45) using SP-BCD is summarized in Algorithm 3.2.

There are several notable characteristics of our algorithm:

- This algorithm is amenable to parallelism, which is suitable for modern computing clusters. Our method possesses one of key advantages of stochastic parallel coordinate descent method (Richtárik and Takáč, 2012): providing the flexibility that in each iteration the number of selected blocks can be optimized completely in parallel according to available number of machines or computational cores. This could make use of all the computational availability as effectively as possible.

- Each subproblem involves the evaluation of a scaled version of the proximal operators of $f_i(\mathbf{x}_i)$ and $g^*(\mathbf{y})$. And for many applications, the evaluation of proximal operators is well-studied and easy to implement in terms of computation complexity and scalability; see Parikh and Boyd (2013) for more details.

- The related non-stochastic primal-dual algorithms (Chambolle and Pock, 2011; He and Monteiro, 2014; Chambolle and Pock, 2014) need evaluation of the norm of $\mathbf{A}$. If the problem size is huge, the evaluation of the norm might be cumbersome and highly time-consuming. The parameter configuration in our algorithm shares the same spirit with Pock and Chambolle (2011), which avoids the norm

estimation, but still leads to a $O(1/T)$ convergence rate. When all the blocks are chosen in each iteration, SP-BCD is equivalent to (Pock and Chambolle, 2011).

- Compared with recent work by Zhang and Xiao (2015), we do not assume any smoothness or strong convexity of the function $f(\mathbf{x})$ and $g^*(\mathbf{y})$. This property enables the applicability of our method to a wider-range of applications, as demonstrated in Section 3.4.2.

- For the optimization problem with linear constraints, although an augmented Lagrangian framework, such as **A**lternating Direction **M**ethod of **M**ultipliers (**ADMM**, Boyd et al. (2011)), is capable of implementing an effective optimization in many problems, the selection of penalty parameter has a dramatic influence on its performance. The current rule of selecting the penalty parameter relies on various heuristics or exhaustive search, and no theoretical justifications exist. Our method SP-BCD avoids this issue.

3.4.1 Convergence Analysis for SP-BCD

For the convergence analysis of SP-BCD, we employ the following gap for any saddle point $(\mathbf{x}, \mathbf{y})$,

$$\mathcal{G}(\mathbf{x}', \mathbf{y}') \triangleq \max_{\mathbf{y}} L(\mathbf{x}', \mathbf{y}) - \min_{\mathbf{x}} L(\mathbf{x}, \mathbf{y}'), \tag{3.53}$$

This gap was also used and discussed by Chambolle and Pock (2011), which could practically measure the optimality of the algorithm for CCSP problems. Since we do not assume any strong convexity of function $f(\mathbf{x})$ or $g^*(\mathbf{y})$, there might exist multiple saddle points. Thus, it is not easy to measure the optimality by the distance to the saddle points as AdaSPDC. The following theorem establishes the convergence of SP-BCD. Note that since we do not assume strong convexity of functions $f_j(\cdot)$ and $g^*(\cdot)$, we cannot establish the linear convergence rate owned by SPDC and AdaSPDC, only sublinear convergence rate can be achieved.

Theorem 2. *Given that all $f_j(\mathbf{x}_j)$ and $g^*(\mathbf{y})$ are convex functions, and we set $\theta = m/q$, proximal parameters for primal and dual update as Eq.(3.48) and (3.51), respectively. Then for any saddle point $(\mathbf{x}, \mathbf{y})$, the expected gap decays as the following rate:*

$$\mathbb{E}\left[L\left(\sum_{t=1}^{T} \mathbf{x}^t/T, \mathbf{y}\right) - L\left(\mathbf{x}, \sum_{t=1}^{T} \mathbf{y}^t/T\right)\right] \leq \frac{1}{T} M(0),$$

where

$$M(0) = \frac{q}{m} \cdot \frac{1}{2}\|\mathbf{x}^0 - \mathbf{x}\|^2_{1/\boldsymbol{\tau}} + \frac{1}{2}\|\mathbf{y}^0 - \mathbf{y}\|^2_{1/\boldsymbol{\sigma}^0} - \left\langle \mathbf{y}^0 - \mathbf{y}, \mathbf{A}\left(\mathbf{x}^0 - \mathbf{x}\right)\right\rangle + \frac{q-m}{m}\left(f(\mathbf{x}^0) + \left\langle \mathbf{y}, \mathbf{A}\mathbf{x}^0\right\rangle - \left(f(\mathbf{x}) + \langle \mathbf{y}, \mathbf{A}\mathbf{x}\rangle\right)\right).$$

We can easily observe that the convergence rate of this gap imply the same primal/dual rate by transforming CCSP problem into pure primal/dual problem. The proof of the above theorem is presented in Appendix C.

Remark. For the parameter configuration of SP-BCD when $\theta = m/q$, the key point for obtaining the convergence of SP-BCD is that we select one particular configuration of $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}^t$ to guarantee the positive semidefiniteness of the following matrix,

$$\mathbf{P} = \begin{bmatrix} \operatorname{diag}(1/\boldsymbol{\tau}_{S_t}) & -\mathbf{A}_{S_t}^T \\ -\mathbf{A}_{S_t} & \frac{m}{q}\operatorname{diag}(1/\boldsymbol{\sigma}^t) \end{bmatrix} \succeq 0. \tag{3.54}$$

Under the chosen parameter configuration of $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}^t$ in SP-BCD, we can guarantee matrix $\mathbf{P}$ is diagonally dominant, directly leading positive semidefiniteness. However, the parameter configuration to make $\mathbf{P} \succeq 0$ is not unique. We find that other configurations are also valid, for instance, for each block j, $\boldsymbol{\tau}_j = \left(1/\|\mathbf{A}_j\|\right)\mathbf{I}$ and $\boldsymbol{\sigma} = \frac{m}{q}\sigma\mathbf{I}$, where $\sigma = 1/\max\{\|\mathbf{A}_j\|\}_{j=1}^q$. Different parameter configuration might provide some influence on the performance of the algorithm. We leave the comparison between them and further theoretical analysis as future work.

Implementation Details

For the selection of randomized blocks, before each pass (epoch) over all the blocks, we suggest firstly randomly permuting all the q blocks and then selecting m blocks cyclically. This way of selecting random blocks follows the custom of general stochastic gradient descent methods, and provides slightly better performance, in our experimental experience.

For some applications, such as Lasso in Section 3.4.2.2, each block variable could be a scalar. Then in each iteration, we just randomly select m coordinates to update.

3.4.2 Applications

In this section, we provide various examples of separable convex-concave saddle point problem from the area of machine learning applications. For each application, experi-

ments are conducted to compare our method SP-BCD with other competitive methods for that application. Note that in each application, we select different methods to compare with that have already shown strong performance in that particular scenario.
We run all the experiments in Matlab R2014a sequentially for demonstration with one Scientific Linux PC (quad-core) Intel(R) Core i5-2400 3.1 GHz CPU and 7.7 GB RAM. The parameters for each method for each different application will be specified below. Each experiment is run 10 times and the average results are reported to show statistical consistency. We present all the experimental results we have done for each application.

3.4.2.1 Matrix Decomposition

We consider a generic matrix decomposition problem with the form,

$$\begin{aligned} \min_{\{\mathbf{X}_j\}_{j=1}^q} \quad & \psi_1(\mathbf{X}_1) + \gamma_2\psi_2(\mathbf{X}_2) + \cdots + \gamma_q\psi_q(\mathbf{X}_q) \\ \text{s.t.} \quad & \sum_{j=1}^{q} \mathbf{X}_j = \mathbf{B}, \end{aligned} \tag{3.55}$$

where the matrix variables $\mathbf{X}_1, \ldots, \mathbf{X}_q \in \mathbb{R}^{s\times n}$, $\mathbf{B} \in \mathbb{R}^{s\times n}$ is a given data matrix and $\gamma_i > 0$ are trade-off parameters.

The goal of this optimization problem is to decompose a given matrix $\mathbf{B}$ into a sum of individual components $\mathbf{X}_j$ such that each of them is "simple" and "meaningful" in a sense described by its corresponding objective term $\psi(\cdot)$. This type of problems exhibit a variety of applications and have attracted substantial interests, see some of recent works for more details (Wright et al., 2009; Candès et al., 2011; Chandrasekaran et al., 2012a,b; Ma et al., 2013).

We list several choices of the objective terms, which are commonly used to enforce certain property for the matrix decomposition depending on different tasks.

- *Squared Frobenius norm.*

$$\psi(\mathbf{X}) = \frac{1}{2}\|\mathbf{X}\|_F^2 = \frac{1}{2}\sum_{ij} X_{ij}^2. \tag{3.56}$$

  This penalty is a classic least squares measure and enforces the entries of $\mathbf{X}$ to be small enough.

- *Entrywise l_1 norm.*

$$\psi(\mathbf{X}) = \|\mathbf{X}\|_1 = \sum_{ij} |X_{ij}|. \tag{3.57}$$

As a convex surrogate for the number of nonzero entries in $\mathbf{X}$, this norm encourages $\mathbf{X}$ to be sparse.

- *Sum-column-norm.*
$$\psi(\mathbf{X}) = \sum_j \|\mathbf{X}_{:j}\|_2, \tag{3.58}$$
where $\mathbf{X}_{:j}$ is the jth column of $\mathbf{X}$. This term enforces column sparsity in $\mathbf{X}$, i.e., choosing $\mathbf{X}$ with many zero columns. There is a corresponding row version.

- *Elementwise constraints.* In some cases, we would like to constrain some or all entries to lie in some set, i.e., $X_{ij} \in \mathcal{C}_{ij}$. For example, enforcing certain entries of the matrix to known values. This can be used to, e.g., require $\mathbf{X}$ to be diagonal (if $\mathbf{X}$ is square) or to follow some fixed sparsity pattern. Another example is the box constraint $\mathbf{X}_{ij} \in [l_{ij}, u_{ij}]$; a common special case is to require to be non-negative.

- *Nuclear norm.*
$$\psi(\mathbf{X}) = \|\mathbf{X}\|_*, \tag{3.59}$$
which encourages $\mathbf{X}$ to be low rank (as a convex surrogate for non-convex rank function). It can seen as the l_1 norm of the singular values of the matrix $\mathbf{X}$.

In the following, we introduce one particular interesting matrix decomposition problem, Robust Principal Component Analysis (RPCA) in computer vision (Wright et al., 2009; Candès et al., 2011), which involves some choices of the objective terms above. We then compare our method SP-BCD with other state-of-the-art approaches on this application.

Robust Principal Component Analysis (RPCA) is a variant of PCA to obtain a low rank and sparse decomposition of an observed data matrix $\mathbf{B}$ corrupted by noise (Wright et al., 2009; Candès et al., 2011), which could help to handle outliers existing in datasets. RPCA aims to solve the following optimization problem,

$$\min_{\mathbf{X}_1,\mathbf{X}_2,\mathbf{X}_3} \frac{1}{2}\|\mathbf{X}_1\|_{\mathrm{F}}^2 + \gamma_2\|\mathbf{X}_2\|_1 + \gamma_3\|\mathbf{X}_3\|_* \tag{3.60}$$

$$\text{s.t.} \quad \mathbf{B} = \mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3, \tag{3.61}$$

where $\mathbf{B} \in \mathbb{R}^{s\times n}$, $\mathbf{X}_1$ is a noise matrix, $\mathbf{X}_2$ is a sparse matrix, $\mathbf{X}_3$ is a low rank matrix, and $\|\cdot\|_*$ is the nuclear norm of a matrix.

We generate the observation matrix $\mathbf{B}$ in the same way as (Parikh and Boyd, 2014)[3]; chose $\mathbf{B} = \mathbf{L} + \mathbf{S} + \mathbf{V}$, where $\mathbf{L}$ is a rank r matrix, $\mathbf{S}$ is a sparse matrix, and $\mathbf{V}$ is a dense noise matrix. The matrix $\mathbf{L}$ is generated as $\mathbf{L} = \mathbf{L}_1\mathbf{L}_2$ with $\mathbf{L}_1 \in \mathbb{R}^{s\times r}$ and $\mathbf{L}_2 \in \mathbb{R}^{r\times n}$, where entries in both $\mathbf{L}_1$ and $\mathbf{L}_2$ were sampled independently from $\mathcal{N}(0,1)$. The matrix $\mathbf{S}$ was generated with density 0.05, with each non-zero entry sampled uniformly from $[-10,10]$. Each entry in $\mathbf{V}$ was sampled from $\mathcal{N}(0,10^{-3})$. We set $s = 2000$, $n = 5000$ and the rank is $r = 100$.

The regularization parameters are set as $\gamma_2 = 0.15\|\mathbf{B}\|_\infty$ and $\gamma_3 = 0.15\|\mathbf{B}\|$. Note that RPCA problem with this matrix size is non-trivial since there are in total 3×10^7 variables and 10^7 equality constraints to handle.

As a concrete example of separable function minimization with linear constraints, RPCA can be easily reformulated into Sep-CCSP form (3.45). The parameter configuration for SP-BCD with each different number of blocks m chosen from the possible $q = 3$ in each iteration can be obtained using Algorithm 3.2: (1) $m = 1$, $(\theta,\tau,\sigma^t) = (1/3,1,1)$; (2) $m = 2$, $(\theta,\tau,\sigma^t) = (2/3,1,1/2)$; (3) $m = 3$, $(\theta,\tau,\sigma^t) = (1,1,1/3)$.

Our method is compared with (1) ADMM implemented by Parikh and Boyd (2014); (2) Gauss-Seidel ADMM (GSADMM) (Hong and Luo, 2012), which solves the problem (3.24) in a cyclic block coordinate manner. However, GSADMM with multiple blocks is not well understood and there is no theory guarantee, and GSADMM has to be implemented sequentially and cannot be parallel; (3) PDCP (Chambolle and Pock, 2011), for which the recommended parameter configuration can be easily obtained as $(\theta,\tau,\sigma) = \left(1,1/\sqrt{3},1/\sqrt{3}\right)$; (4) Parallel Direction Method of Multipliers (PDMM, Wang et al. (2014)) with suggested parameters (see Eq.(8) in Wang et al. (2014)) and different numbers of blocks, $m = \{1,2,3\}$. For each of the three compared methods (ADMM, GSADMM and PDMM), we run extensive experiments using different penalty parameter ρ, and report the results for best performing ρ, despite the fact that knowledge of which ρ is optimal is not available to the algorithms a priori.

Figure 3.7 depicts the performance of the all the methods on evolution of the objective (in Eq.(3.60)) and the residual (i.e., the deviation from satisfied constraints measured by $\|\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3 - \mathbf{B}\|_F$) w.r.t. number of passes and consumed time. For the objective function, all the compared methods can quickly achieve the consensus value in 20 passes. The key difference of algorithm performance between them focuses on how fast they satisfy the constraint for Eq.(3.61). Our method SP-BCD with $m = 2$ is the fastest, achieving almost the same performance with GASDMM and fully par-

[3] `http://stanford.edu/~boyd/papers/prox_algs/matrix_decomp.html`

allelizable, while GSADMM can only be run sequentially. More details of algorithm performance are provided in Table 3.3.

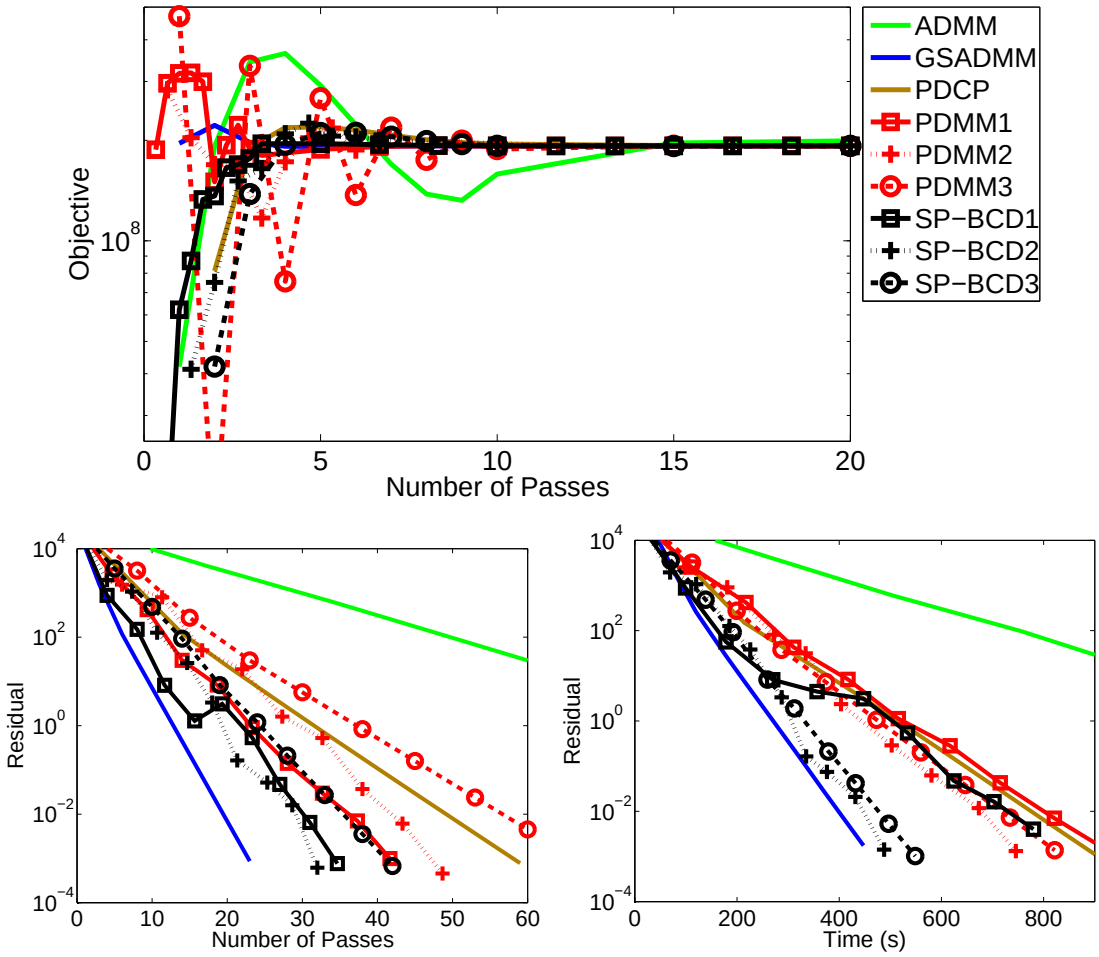


Figure 3.7: RPCA problem: comparison of our method and ADMM, GSADMM, PDCP and PDMM with $m = \{1, 2, 3\}$. The first column shows the evolution of objective function w.r.t. number of passes. The left and right panel of the second column depict the residual evolution (measured by $\|\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3 - \mathbf{B}\|_F$) as function of number of passes and consumed time, respectively. All the compared methods can quickly achieve the consensus objective value in 20 passes. The main difference is how fast they satisfy the constraint. SP-BCD with $m = 2$ is the fastest, achieving almost the same performance with GASDMM and fully parallelizable, while GSADMM can only be run sequentially.

Note that our method is also capable of handling two types of popular problems in economics, exchange and allocation (Parikh and Boyd, 2013, Chap 5.3 and 5.4), both of which share the similar structure with robust PCA.

Table 3.3: RPCA problem: performance of all compared methods. All the methods achieve the same objective value. Our method SP-BCD with $K = 2$ achieves nearly the same performance with GSADMM and can be fully parallelized, while GSADMM can only be run sequentially., Although PDMM2 obtains the lowest residual (measured by Frobenus Norm of deviation of satisfied constraints), it spends much longer time $750s$, compared with $492s$ for SP-BCD2. When we run the SP-BCD2 with the same amount of time as that of PDMM2, SP-BCD2 could achieve Frobenus Norm of residual as $\mathbf{2.36 \times 10^{-4}}$, which shows better performance than PDMM2.

Methods	Iteration	Time (s)	Frobenus Norm of residual (10^{-4})	Objective (10^8)
ADMM	149	2191	9.71	1.924
GSADMM	**23**	**448**	8.69	1.924
PDCP	59	911	7.80	1.924
PDMM1	125	927	9.92	1.924
PDMM2	73	750	**4.55**	1.924
PDMM3	67	834	8.56	1.924
SP-BCD1	104	784	7.63	1.924
SP-BCD2	48	**492**	**6.17**	1.924
SP-BCD3	42	553	6.72	1.924

- Exchange:

$$\begin{aligned} &\min_{\{\mathbf{x}_j\}_{j=1}^q} \sum_{j=1}^{q} f_j\left(\mathbf{x}_j\right), \\ &\text{s.t. } \sum_{j=1}^{q} \mathbf{x}_j = \mathbf{0}. \end{aligned} \tag{3.62}$$

- Allocation:

$$\begin{aligned} &\min_{\{\mathbf{x}_j\}_{j=1}^q} \sum_{j=1}^{q} f_j\left(\mathbf{x}_j\right) \\ &\text{s.t. } \mathbf{x}_j \geq \mathbf{0}, \quad j = 1, \ldots, q \\ &\sum_{j=1}^{q} \mathbf{x}_j = \mathbf{b}, \end{aligned} \tag{3.63}$$

where $f_j(\cdot)$ represents the cost function for subsystem (or agent) j, the components of the vectors $\mathbf{x}_j$ represent quantities of commodities that exchanged or allocated among q agents or subsystems. In exchange problem (3.62), the linear constraint $\sum_{j=1}^{q} \mathbf{x}_j = \mathbf{0}$ represents the equilibrium constraint that each commodity clears, or balances. In allocation problem (3.63), the constraint $\sum_{j=1}^{q} \mathbf{x}_j = \mathbf{b}$ simply means that the sum of all the allocated commodities should be equivalent to the total quantity $\mathbf{b}$.

Table 3.4: Lasso problem: performance of all compared methods. Problem size is described as: number of data samples N, number of features D, and d is number of non-zero entries in $\mathbf{x}_{\text{true}}$. For smaller sized problems, ADMM and SP-BCD are the fastest, achieving nearly the same performance in term of consumed time. For larger sized problems, SP-BCD is the fastest, since ADMM needs to solve a large-scale linear system in each iteration, involving a high computational burden.

Methods	N, D, d	Time (s)	Number of passes	$\|\mathbf{x}^* - \mathbf{x}_{\text{true}}\|_2$	Objective
ISTA	$1000, 5000, 100$	2.27	100	13.401	111.405
	$5000, 20000, 500$	45.67	100	25.552	448.351
FISTA	$1000, 5000, 100$	1.16	56	13.115	111.320
	$5000, 20000, 500$	19.00	49	25.207	448.271
ADMM	$1000, 5000, 100$	**0.69**	63	**13.088**	**111.318**
	$5000, 20000, 500$	19.83	51	**25.154**	**448.258**
PDCP	$1000, 5000, 100$	1.40	100	13.097	**111.318**
	$5000, 20000, 500$	26.80	100	**25.157**	448.263
SP-BCD	$1000, 5000, 100$	**0.70**	**30**	**13.088**	**111.318**
	$5000, 20000, 500$	**13.32**	**30**	**25.153**	**448.263**

3.4.2.2 Lasso

Lasso is an important l_1 regularized linear regression Hastie et al. (2009), which involves solving the following optimization problem,

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1, \tag{3.64}$$

where λ is a regularization parameter, $\mathbf{A} \in \mathbb{R}^{N\times D}$ is an observed feature matrix and each row of $\mathbf{A}$ is one data point. In typical applications, there are many more features than number of training examples, i.e., $N < D$. Lasso has been widely studied and applied, particularly in the analysis of biological data, where only a small fraction of a large number of possible factors are actually predictive of some outcome of interest (Hastie et al., 2009, Chap 18.4).

To put Lasso problem into Sep-CCSP form, we dualize the first quadratic loss function in Eq.(3.64), naturally yielding

$$\min_{\mathbf{x}\in\mathbb{R}^D} \max_{\mathbf{y}\in\mathbb{R}^N} \lambda\|\mathbf{x}\|_1 + \langle \mathbf{y}, \mathbf{A}\mathbf{x}\rangle - \sum_{i=1}^{N}\left(\frac{1}{2}y_i^2 + b_i y_i\right) \tag{3.65}$$

Since $\|\mathbf{x}\|_1$ is totally separable and non-strongly convex, we can apply our SP-BCD method to the above saddle point problem, i.e., in each iteration we randomly select

m coordinates of primal variable $\mathbf{x}$ to update. For the dual update, the corresponding problem has a simple close-formed solution that can be updated directly.

Due to the vast literature on optimization methods for the Lasso problem, we only choose several representative methods to compare with our method (There are several recently developed methods using pure coordinate decent, such as Richtárik and Takáč (2012, 2014), which are expected to have good convergence performance. We did not inclue here.) (1) ISTA (Iterative Shrinkage Thresholding Algorithm); (2) FISTA (Fast ISTA, Beck and Teboulle (2009)); (3) ADMM (Boyd et al., 2011, Chap 6.4), note that the formulation of ADMM for Lasso problem is different from Eq.(3.65). ADMM splits the loss function and regularization term using two separable variables, which needs to solve a linear system in each iteration. When the problem size is very large, the time complexity is high and even computationally inacceptable. (4) PDCP (Chambolle and Pock, 2011), which needs estimation of norm of matrix $\mathbf{A}$.

We generate the data in the same way as (Boyd et al., 2011, Chap 11.1): each element of matrix $\mathbf{A}$, $a_{ij} \sim \mathcal{N}(0,1)$ and then normalize the columns to have unit l_2 norm; a "true" value $\mathbf{x}_{\text{true}} \in \mathbb{R}^n$ has d nonzeros entries, each of which is sampled from $\mathcal{N}(0,1)$; the output $\mathbf{b} = \mathbf{A}\mathbf{x}_{\text{true}} + \varepsilon$, where $\varepsilon \sim \mathcal{N}\left(0, 10^{-3}\mathbf{I}\right)$. The regularization parameter is set as $\lambda = 0.1\|\mathbf{A}^T\mathbf{b}\|_\infty$. The implementation of ISTA, FISTA and ADMM is based on code by Parikh and Boyd (2013) [4]. The proximal parameter for these methods are set as 1. For our method SP-BCD, in each iteration we randomly choose $m = 100$ coordinates to run the experiments.

Table 3.4 reports the performance of all these methods on two problems with different sizes and sparsity. Figure 3.8 depicts the objective evolution as a function of number of passes and time for problem size: $N = 5000, N = 20000$ and number of non-zero entries of $\mathbf{x}_{\text{true}}$, $d = 500$. We can observe that SP-BCD uses the least number of passes and time to achieve same objective value with other methods. For smaller sized problems, ADMM also performs very well. However, when the problem size is rising, the computational burden from solving large linear systems becomes a serious issue for ADMM. The issue of scalability also influences the performance of PDCP since it needs the estimation of norm of matrix $\mathbf{A}$. Our method SP-BCD is not restricted heavily by a large problem size.

[4] `http://web.stanford.edu/~boyd/papers/prox_algs/lasso.html`

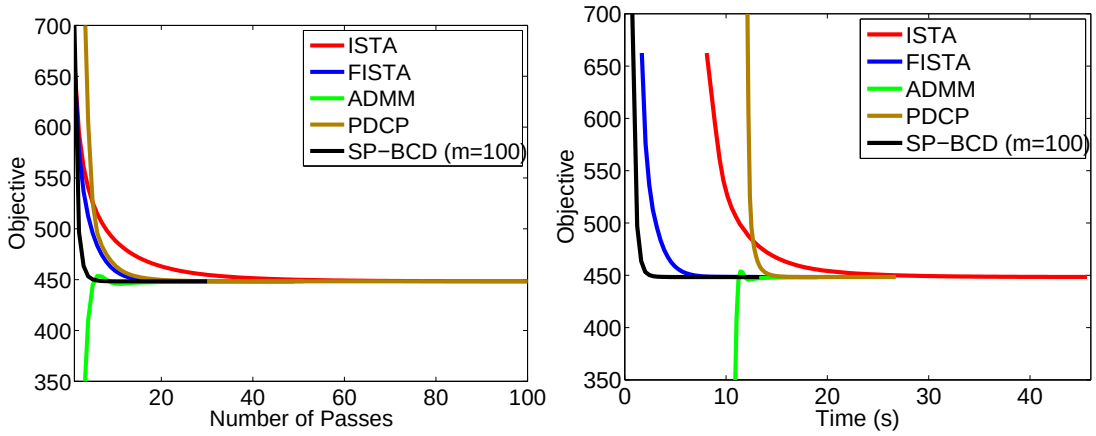


Figure 3.8: Lasso: comparison of convergence performance w.r.t. the number of passes and time. Problem size: $m = 5000, n = 20000$ and number of nonzero entries of $\mathbf{x}_{\text{true}}$, $d = 500$. SP-BCD uses least number of passes and time to achieve same objective value with other methods. ADMM needs to solve a large-scale linear system in each iteration. PDCP needs estimation of the norm of the large matrix $\mathbf{A}$. Both of them are hindered by the issue of scalability. Our method SP-BCD avoids this issue.

3.4.2.3 Feature Selection with Group Lasso

As an extension of Lasso, group Lasso is an structured regularized regression model for high-dimensional data, which can help to select key explanatory factors in a grouped manner (Yuan and Lin, 2006).

Given a training dataset consisting of N i.i.d. observations, $\{(\mathbf{a}_i, z_i)\}$, where $\mathbf{a}_i \in \mathbb{R}^D$ is a D-dimensional vector and $z_i \in \{-1, 1\}$ for the binary classification problem or $z_i \in \mathbb{R}$ for the regression problem. Suppose that these d features are divided into G disjoint groups with d_g, the number in g-th group. Hence, we can rewrite $\mathbf{a} = [\mathbf{a}_1^T, \mathbf{a}_2^T, \ldots, \mathbf{a}_G^T]^T$. When $d_g = 1$ for all groups, the data do not form a group in the feature space. Then group Lasso tries to find the optimal regression coefficient vector $\mathbf{x} \in \mathbb{R}^D$ by solving the following optimization problem,

$$\min_{\mathbf{x}} \lambda \sum_{g=1}^{G} \sqrt{d_g} \|\mathbf{x}_g\|_2 + \frac{1}{N} \sum_{i=1}^{N} g_i(\mathbf{a}_i^T \mathbf{x}, z_i), \tag{3.66}$$

where $\mathbf{x}$ is partitioned according to feature grouping, i.e., $\mathbf{x} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_G^T]^T$, the loss function $g_i(\mathbf{a}_i^T \mathbf{x}, z_i)$ should be convex, such as the squared loss, logit loss, or hinge loss. The regularization term is the sum of groupwise L_2-norm $\|\mathbf{x}_g\|_2$, and the trade-off constant λ is to balance between the loss and the regularization term. The value d_g accounts for the varying group sizes.

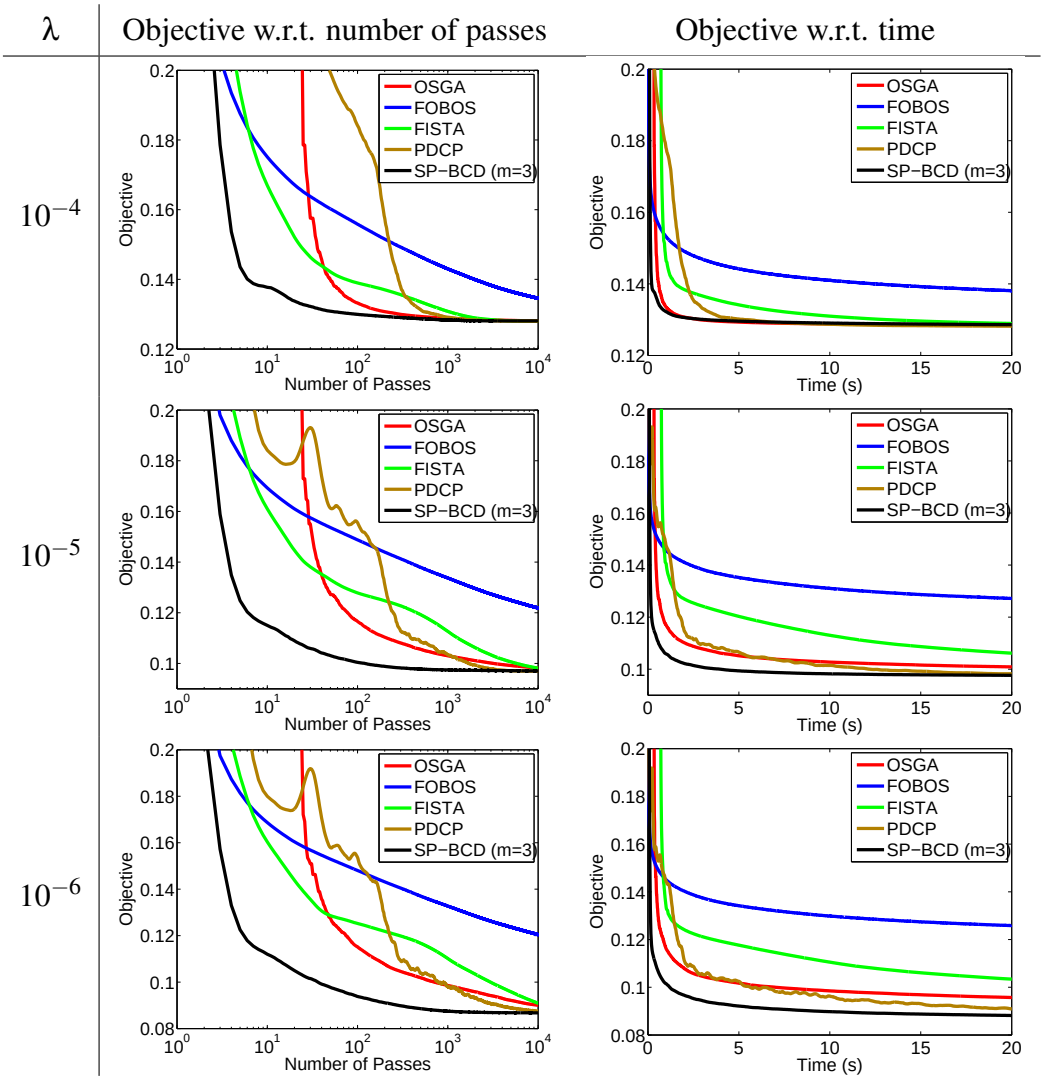
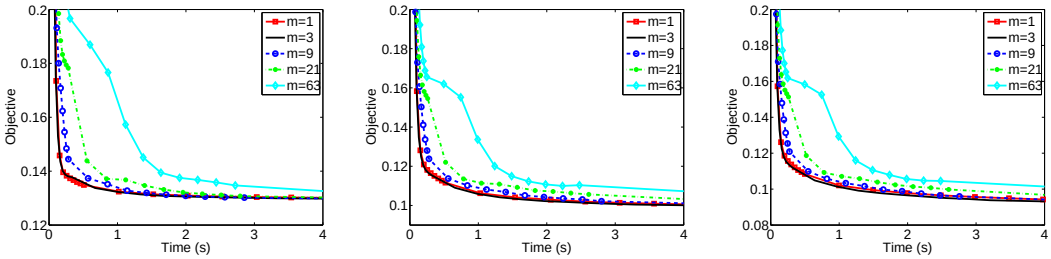


Figure 3.9: Group Lasso on MEMset dataset with different regularization parameter λ: comparison of our method SP-BCD ($m = 3$ blocks are chosen in each iteration) with OSGA, FOBOS, FISTA and PDCP. In all these test cases, SP-BCD demonstrates its superiority on both the number of passes and the consumed time. When the regularization is strong with large $\lambda = 10^{-4}$, all the methods tend to converge fast, but SP-BCD is the fastest one. PDCP performs poorly in the first hundreds or thousands of passes, since it only uses a constant step size $1/\|A\|$. Compared with PDCP, our method considers the structure of matrix $\mathbf{A}$ and scales each dimension of primal and dual updates, which can achieve better empirical performance.

We use hinge loss function $g_i(\mathbf{a}_i^T\mathbf{x}, z_i) = \max(0, 1 - z_i\mathbf{a}_i^T\mathbf{x})$ for demonstration, which is a non-smooth loss function. By employing the conjugate dual transformation of hinge loss,

$$g_i(\mathbf{a}_i^T\mathbf{x}, z_i) = \sup_{y_i \in [0,1]} \langle -y_i z_i \mathbf{a}_i, \mathbf{x} \rangle + y_i, \tag{3.67}$$

we can easily transform the group Lasso problem into the following saddle point prob-



(a) $\lambda = 10^{-4}$ (b) $\lambda = 10^{-5}$ (c) $\lambda = 10^{-6}$

Figure 3.10: SP-BCD for Group Lasso on MEMset dataset with different regularization parameter λ and different number chosen blocks m. The effect of m: a smaller number of blocks yields faster convergence, which shows the advantage of the flexible stochastic update of our method compared with Pock and Chambolle (2011).

lem,

$$\min_{\mathbf{x}} \max_{\mathbf{y}\in[0,1]^N} \lambda \sum_{g=1}^{G} \sqrt{d_g}\|\mathbf{x}_g\|_2 + \frac{1}{N}\langle -\sum_{i=1}^{N} y_i z_i \mathbf{a}_i, \mathbf{x}\rangle + \frac{1}{N}\sum_{i=1}^{N} y_i.$$

Note that $g^*(\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} y_i$ in not strongly convex, and then SP-BCD can be applied since it does not assume strong convexity. This reformulation of group Lasso makes both the dual and primal update extremely simple and efficient, both of which have closed-formed solution and can be easily derived.

In order to evaluate performance of our method for the group Lasso problem, we apply it to a real-world dataset for splice site detection, which plays an important role in gene finding. The MEMset Donor dataset[5] is used for the evaluation, which is widely used to demonstrate the advantages of the group Lasso models Meier et al. (2008); Roth and Fischer (2008). It contains a training set of 8,415 true and 179,438 false human donor sites. An additional test set consists of 4,208 true and 89,717 false donor site. From the original training set, we construct a balanced training set with $8,415$ true and $8,415$ false donar sites. Group lasso on this data with up to 2nd order interactions and up to 4 order interactions has been analyzed by Meier et al. (2008) and Roth and Fischer (2008), respectively. As shown in Roth and Fischer (2008), there is not much improvement using higher order interactions. Therefore we only consider all three-way and lower order interactions. This forms $G = 63$ groups or $D = 2604$-dimensional feature space with $\{7,21,35\}$ groups of $\{4,16,64\}$-dimensional coordinate block, respectively.

We compare our method SP-BCD with several recent developed competitive optimization methods for the non-smooth regularized problem: (1) OSGA (Optimal Sub-

[5] http://genes.mit.edu/burgelab/maxent/ssdata/

Gradient Algorithm, Neumaier (2014)), A fast subgradient algorithm with optimal complexity; (2) FOBOS (FOrward-Backward Splitting, Duchi and Singer (2009)); (3) FISTA (Beck and Teboulle, 2009), in which we use smoothing technique with make it applicable with chosen smoothing parameter $\varepsilon = 5 \times 10^{-4}$; (4) PDCP (Chambolle and Pock, 2011).

In this application, we evaluate the performance of these methods under different configurations of the regularization parameter $\lambda = \{10^{-4}, 10^{-5}, 10^{-6}\}$. Figure 3.9 compares our method SP-BCD (with $m = 3$) with other methods in terms of the evolution of the objective function in Eq.(3.66) both w.r.t. the number of passes and w.r.t time. We can observe that nearly all the compared methods could achieve sublinear convergence rate. In all these test cases, SP-BCD demonstrates its superiority on both number of passes and consumed time. When the regularization is strong with large $\lambda = 10^{-4}$, all the methods tend to converge fast, but SP-BCD is the fastest one. PDCP performs poorly in first hundreds or thousands of passes, since it only applies the constant step size $1/\|A\|$. Compared with PDCP, our method considers the structure of matrix $\mathbf{A}$ and scales each dimension of primal and dual updates, which can achieve better empirical performance.

In order to investigate the effect of the number of chosen blocks for SP-BCD, we implement it using different m values, $m = \{1, 3, 9, 21, 63\}$. The results are shown in Figure 3.10. In all the tested cases, a smaller number of blocks yields faster convergence, which shows the advantage of the flexible stochastic update of our method compared with Pock and Chambolle (2011). The approach of Pock and Chambolle (2011) is equivalent to SP-BCD when all blocks are chosen in each iteration, $m = 63$.

3.5 Discussion and Future Directions

This chapter introduces two novel methods for Sep-CCSP problems, AdaSPDC and SP-BCD. Now we provide a comprehensive comparison between the two propose methods for Sep-CCSP problems, which could guide its usage in practice and enlighten future work.

1. Both AdaSPDC and SP-BCD are based on a primal-dual framework for CCSP problems, which alternatively optimizes primal and dual variable through proximal algorithms. Both of them are stochastic coordinate descent methods designed for large-scale CCSP problems. Compared with Stochastic Gradient

Descent (SGD, a pure stochastic primal method) and Stochastic Dual Coordinate Descent (SDCA, a pure stochastic dual method), the primal-dual framework plays an "intermediate" role. This "intermediate" approach transforms the original problems, such as separable function minimization with linear constraints and regularized ERM, into the Sep-CCSP form, leading to simple and easy updates for the subproblems in each iteration and showing superior theoretical and empirical evidence through our algorithm design in AdaSDPC and SP-BCD.

2. The key difference between AdaSPDC and SP-BCD focuses on their assumptions that whether the separable functions $f(\mathbf{x})$ and $g^*(\mathbf{y})$ are strongly convex or not. AdaSPDC assumes that both $f(\mathbf{x})$ and $g(\mathbf{y})$ are strongly convex, particularly applicable to regularized ERM with Lipschitz smooth loss functions (since the conjugate dual of Lipschitz smooth loss functions are strongly convex) and strongly convex regularization term. Under the strong convexity assumption, AdaSPDC can achieve a sharper linear convergence rate that SPDC, as shown both theoretically and empirically. On the other hand, SP-BCD only assume $f(\mathbf{x})$ and $g*(\mathbf{y})$ are general convex functions, which are suitable for a wider range of applications, such as separable function minimization with linear constraints, and regularized ERM with non-smooth loss functions and/or general non-strongly convex regularization terms. Since without strong convexity assumption, SP-BCD can only achieve a sublinear convergence rate.

3. Both AdaSPDC and SP-BCD exploit the structure of connection matrix $\mathbf{K}$ between the primal and dual variable, and propose to use adaptive step sizes according to the randomly selected blocks in each iteration. However, due to different assumptions, AdaSPDC and SP-BCD use different stepsize rules.

As mentioned in Section 3.3.1 and 3.4.1, the parameter configuration of stepsizes in the proximal algorithms are not unique, and there exists other valid parameter configuration to induce the algorithm convergence. Thus, an immediate future research direction is to investigate other valid parameter configuration for both AdaSPDC and SP-BCD and compare their performance both theoretically and empirically.

Since Sep-CCSP has a wide range of applications in machine learning, computer vision and economics, etc, it is worthy of exploring more interesting applications in these areas besides the considered experiments in this chapter.

Chapter 4

Dynamics-based Methods for Large-scale Bayesian Sampling

Using Monte Carlo sampling for Bayesian posterior inference is a common approach in machine learning. Markov Chain Monte Carlo (MCMC) is a general and powerful framework, which allows sampling from a large class of target distributions, and which scales well with the dimensionality of the sample space. However, when faced with extremely *large-scale* data, traditional MCMC methods involve expensive computational cost due to its evaluation over the entire dataset in each iteration.

To handle the scalability issue in Bayesian sampling methods, this chapter explores series of dynamics-based sampling methods based on stochastic gradient. All of these methods employ the spirit of *integrating local information, i.e., only touching a small mini-batch of data items for every sample we generate*. In this setting, existing techniques rely on estimating the variance or covariance of the subsampling error, and assume constant variance. We propose a covariance-controlled adaptive Langevin thermostat that can effectively dissipate parameter-dependent noise while maintaining a desired target distribution. This method achieves a substantial speedup over popular alternative schemes for large-scale machine learning applications.

This chapter is an extended work based on the following published paper, where ZZ and SX initialized the idea and algorithm; SX contributed to proof of the theorem; ZZ implemented all the experiments and analysis; BL and AJS provided insightful suggestions for the paper.

Shang, X.*, Zhu, Z.*, Leimkuhler, B. and Storkey, A.J.(2015). Covariance-Controlled Adaptive Langevin Thermostat for Large-Scale Bayesian Sampling. In *Advances in Neural Information Processing Systems 28 (NIPS).* (* indicates the equal contribution,

the order was decided by lot.)

4.1 Problem Settings

Bayesian analysis gives us a simple recipe for learning from data: given a set of unknown parameters or latent variables $\boldsymbol{\theta} \in \mathbb{R}^D$ that are of interest, we specify a prior distribution $p(\boldsymbol{\theta})$ quantifying what we know about $\boldsymbol{\theta}$ before observing any data. Then we quantify how the observed data $\mathbf{X} = \{\mathbf{x}_i\}_{n=1}^N$ relates to $\boldsymbol{\theta}$ by specifying a likelihood function $p(\mathbf{X}|\boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{x}_i|\boldsymbol{\theta})$. Finally, we apply Bayes' rule to obtain the posterior distribution

$$p(\boldsymbol{\theta}|\mathbf{X}) = p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})/Z, \tag{4.1}$$

where Z is the normalization constant, $Z = \int p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$. To simplify the notation, we denote

$$\pi(\boldsymbol{\theta}) \triangleq p(\boldsymbol{\theta}|\mathbf{X}) = \tilde{\pi}(\boldsymbol{\theta})/Z. \tag{4.2}$$

The fundamental problem we wish to address in Bayesian inference involves find the expectation of some function $g(\boldsymbol{\theta})$ with respect to the posterior distribution $\pi(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathbf{X})$,

$$\mathbb{E}\left[g(\boldsymbol{\theta})\right] = \int g(\boldsymbol{\theta})\pi(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}, \tag{4.3}$$

Generally, the expectation in Eq (4.3) is too complex to be evaluated exactly using analytical techniques. The Monte Carlo sampling methods solve this issue by obtaining a set of samples $\boldsymbol{\theta}_t$ (where $t = 1, \ldots, T$) drawn independently from $\pi(\boldsymbol{\theta})$ and approximately evaluating the expectation by a finite sum

$$\mathbb{E}\left[g(\boldsymbol{\theta})\right] \approx \frac{1}{T}\sum_{i=1}^{T} g(\boldsymbol{\theta}_t). \tag{4.4}$$

4.2 MCMC Methods

A general and powerful Monte Carlo sampling framework, called Markov Chain Monte Carlo (MCMC), allows sampling from a large class of distributions and scales well with dimensionality of the sample space. MCMC methods have their origins in physics (Metropolis and Ulam, 1949), and it was only towards the end of the 1980s that they started to have a significant impact in the field of statistics.

The MCMC methods involve continuously sampling from certain proposal distribution $q(\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1})$ depending on its current state $\boldsymbol{\theta}_{t-1}$, and so the sequence of samples

Algorithm 4.1 Metropolis-Hasting algorithm (MH)

1: Choose a starting state $\boldsymbol{\theta}^{(0)}$.

2: **for** $t = 0, 1, 2, \ldots, T-1$ **do**

3: Sample $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1})$;

4: Computing the accepting ratio

$$\rho_t = \min\left(1, \frac{\tilde{\pi}(\boldsymbol{\theta}^*)q(\boldsymbol{\theta}_t|\boldsymbol{\theta}^*)}{\tilde{\pi}(\boldsymbol{\theta}_t)q(\boldsymbol{\theta}^*|\boldsymbol{\theta}_t)}\right) \tag{4.5}$$

5: Generate $u_t \sim \mathcal{U}(0,1)$, and accept or reject according to the following,

$$\boldsymbol{\theta}_{t+1} = \begin{cases} \boldsymbol{\theta}^* & \text{if } u_n \leq \rho_t, \\ \boldsymbol{\theta}_t & \text{otherwise.} \end{cases} \tag{4.6}$$

6: **end for**

$\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots$ forms a Markov chain. It is assumed that $\tilde{\pi}(\boldsymbol{\theta})$ can readily be evaluated for any given value of $\boldsymbol{\theta}$ although Z may be unknown. The proposal distribution itself is chosen to be sufficiently simple that it is straightforward to draw samples from it directly. At each iteration of the algorithm, we generate a candidate sample $\boldsymbol{\theta}^*$ from the proposal distribution and then accept the sample according to an appropriate criterion. For more comprehensive details of MCMC methods, see the review books and articles (Brooks et al., 2011; Robert and Casella, 2013; Neal, 1993). The generic Metropolis-Hasting (MH) algorithm is summarized in Algorithm 4.1.

In the Metropolis-Hasting, it is flexible to choose the proposal distribution $q(\cdot)$. However the choice has a dramatic effect on the algorithm performance. Traditional MCMC methods often use some random walk proposal (e.g., Gaussian distribution), which lead to highly correlated samples. High acceptance rates can be obtained by proposing smaller transitions; however, it will require more time to make full traversals of parameter space. In high dimensions, when D is large, the random walk becomes inefficient, resulting in low rates of acceptance, poor mixing of the chain and highly correlated samples. A consequence of this is a small effective sample size (ESS) from the chain; see Robert and Casella (2013).

In the following, we introduce several dynamics-based sampling methods to alleviate this issue with better proposal distributions. These methods use information from the gradient of the log density to reduce the random walk effect, and the Metropolis

step to be a correction of the discretization error introduced by numerical integration of the corresponding dynamical systems.

4.3 Dynamical MCMC

In this section, we will review several dynamics-based Monte Carlo methods, which are faster than traditional MH approaches, largely because the dynamical methods avoid the random walk behaviour inherent in simple forms of the MH methods.

The dynamical sampling methods originally derive from the "molecular dynamics" approach (Alder and Wainwright, 1959), which was developed concurrently with the MH algorithm as a means of simulating physical systems. These methods are widely applicable to problems with continuous state variables, provided the gradient of the log density can be calculated.

Dynamical sampling methods are based on a physical analogy. To facilitate this analogy, we often write the distribution $\pi(\boldsymbol{\theta})$ into a canonical distribution form,

$$\pi(\theta) = (1/Z)\exp(-\beta U(\boldsymbol{\theta})), \tag{4.7}$$

where $\beta = 1/(k_B T)$ is a positive constant, k_B is the Boltzmann constant, and T is the system temperature. In the context of Bayesian inference, we often let β be unity, i.e, $\beta = 1$. Thus,

$$U(\boldsymbol{\theta}) = -\log p(\mathbf{X}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) \tag{4.8}$$

is referred as potential energy function.

The gradient of the potential energy for a physical system with respect to its configuration coordinates defines the "force",

$$\mathbf{f}(\boldsymbol{\theta}) = -\nabla U(\boldsymbol{\theta}) \tag{4.9}$$

that acts to change this configuration, via its effect on the system's momentum. When a physical system is in contact with a heat reservoir , it also experiences influences that can be modelled as being random. These dynamical and stochastic effects together result in the system visiting states with a frequency given by its canonical distribution. Therefore, simulating the dynamics of such a physical system provides a way of sampling from the canonical distribution. Dynamical simulation also allows one to observe in detail how the system behaves as it visits states with this distribution.

4.3.1 Metropolis Adjusted Langevin Algorithm (MALA)

MALA is based on a Langevin diffusion process, with $\pi(\boldsymbol{\theta})$ as its stationary and limiting distribution, defined by the stochastic differential equation (SDE)

$$\mathrm{d}\boldsymbol{\theta}(t) = \mathbf{f}(\boldsymbol{\theta})\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{b}(t), \tag{4.10}$$

where $\mathbf{f}(\boldsymbol{\theta})$ is often called as the drift term, $\mathbf{b}$ denotes a D-dimensional Brownian motion, and $\mathrm{d}\mathbf{b}(t)$ colloquially represents a vector of infinitesimal Wiener increments, which is often informally written as $\mathcal{N}(\mathbf{0}, \mathrm{d}t\mathbf{I})$. Simulating the dynamics using a first-order Euler discretization of the SDE above gives the proposal mechanism

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_t + \mathbf{f}(\boldsymbol{\theta})h + \sqrt{2h}\mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4.11}$$

where h is the integration step size. Convergence to the invariant distribution $\pi(\boldsymbol{\theta})$ is no longer guaranteed for finite step size h due to the introduced first-order integration error. A Metropolis acceptance probability after each integration step is utilized to correct the integration error and thus guarantee the convergence to the invariant distribution. Observing that discrete form of the SDE defines a proposal density

$$q(\boldsymbol{\theta}^*|\boldsymbol{\theta}_t) = \mathcal{N}(\boldsymbol{\theta}^*|\boldsymbol{\theta}_t + \mathbf{f}(\boldsymbol{\theta})h, 2h\mathbf{I}), \tag{4.12}$$

then MALA accepts each sample with the probability $\min\left(1, \frac{\tilde{\pi}(\boldsymbol{\theta}^*)q(\boldsymbol{\theta}_t|\boldsymbol{\theta}^*)}{\tilde{\pi}(\boldsymbol{\theta}_t)q(\boldsymbol{\theta}^*|\boldsymbol{\theta}_t)}\right)$.

We can observe that the drift term in the Brownian dynamics determines the direction for the proposal based on the gradient information. However, when the dimensions in $\boldsymbol{\theta}$ are highly correlated with widely different variances, the isotropic diffusion might be inefficient to accommodate the variate with smallest variance. This issue can be circumvented by employing a preconditioning matrix that applies Riemann Manifold metric $\mathbf{M}$ (Girolami and Calderhead, 2011)

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_t + \mathbf{M}f(\boldsymbol{\theta})h + \sqrt{2\mathbf{M}h}\mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4.13}$$

for more details of Riemann Manifold MALA, see Girolami and Calderhead (2011).

4.3.2 Hamiltonian Monte Carlo (HMC)

This subsection briefly describes the HMC methods (also known as Hybrid Monte Carlo); for a detailed introduction and extensive review see Neal (2011). HMC introduces an independent auxiliary variable $\mathbf{p} \in \mathbb{R}^D$ with density $\pi(\mathbf{p}) = \mathcal{N}(\mathbf{p}|\mathbf{0}, \mathbf{M})$. The joint density follows in a factorized form as

$$\pi(\boldsymbol{\theta}, \mathbf{p}) = \pi(\boldsymbol{\theta})\pi(\mathbf{p}) = \pi(\boldsymbol{\theta})\mathcal{N}(\mathbf{p}|\mathbf{0}, \mathbf{M}). \tag{4.14}$$

The negative joint log probability is

$$H(\boldsymbol{\theta}, \mathbf{p}) = U(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} + \text{const.} \tag{4.15}$$

This negative joint log-probability can be interpreted as a Hamiltonian (Duane et al., 1987; Horowitz, 1991; Leimkuhler and Reich, 2004) in physical analogy, which describes the sum of a potential energy function $U(\boldsymbol{\theta})$ at the position $\boldsymbol{\theta}$, and a kinetic energy $\frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}$. The auxiliary variable $\mathbf{p}$ can be interpreted as a momentum variable and the covariance matrix $\mathbf{M}$ denotes a mass matrix.

The Hamiltonian has an nice property that the derivatives of H with respect to $\boldsymbol{\theta}$ and $\mathbf{p}$ are equivalent to the derivatives of $\boldsymbol{\theta}$ and $\mathbf{p}$ with respect to a fictitious time t, respectively. This can be described by the Hamilton's equations

$$\frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t} = \frac{\mathrm{d}H}{\mathrm{d}\mathbf{p}} = \mathbf{M}^{-1}\mathbf{p} \tag{4.16}$$

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\frac{\mathrm{d}H}{\mathrm{d}\boldsymbol{\theta}} = -\nabla U(\boldsymbol{\theta}). \tag{4.17}$$

The solutions to the differential equations at every time t have three markable characteristics:

- energy preservation, i.e. $H(\boldsymbol{\theta}(t), \mathbf{p}(t)) = H(\boldsymbol{\theta}(0), \mathbf{p}(0))$, and hence the joint density $\pi(\boldsymbol{\theta}(t), \mathbf{p}(t)) = \pi(\boldsymbol{\theta}(0), \mathbf{p}(0))$;
- volume preservation $\mathrm{d}\boldsymbol{\theta}(t)\mathrm{d}\mathbf{p}(t) = \mathrm{d}\boldsymbol{\theta}(0)\mathrm{d}\mathbf{p}(0)$, which implies that as the region within the space of variables $(\boldsymbol{\theta}, \mathbf{p})$ evolves under the Hamiltonian dynamcis, its shape might change but its volume will not;
- time reversibility (Leimkuhler and Reich, 2004).

Using the first two characteristics of H, it follows that the Hamiltonian dynamics will leave $\pi(\boldsymbol{\theta}, \mathbf{p})$ invariant. Therefore, by integrating the dynamics over a finite time duration it is feasible to make large moves to the position $\boldsymbol{\theta}$ in a systematic way that avoids random walk behaviour.

However, in practical applications, we cannot solve the differential equations (4.16) and (4.17 analytically, where certain numerical methods are employed. Leimkuhler and Reich (2004) review a number of numerical integrators for Hamiltonian systems which fully satisfy volume preservation and time reversibility, and approximately satisfy total energy preservation with a given order of integration error. One popular

integrator is the leapfrog Duane et al. (1987), alternatively updating discrete-time approximations of $\boldsymbol{\theta}$ and $\mathbf{p}$,

$$\mathbf{p}(t+h/2) = \mathbf{p}(t) - h\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}(t))/2, \tag{4.18}$$

$$\boldsymbol{\theta}(t+h) = \boldsymbol{\theta}(t) + h\mathbf{M}^{-1}\mathbf{p}(t+h/2), \tag{4.19}$$

$$\mathbf{p}(t+h) = \mathbf{p}(t+h/2) - h\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}(t+h))/2. \tag{4.20}$$

We can investigate the properties of the leapfrog integrator as follows. Since the Hamiltonian is separable, it is easy to observe that each complete leapfrog step (Eq (4.18), (4.19) and (4.20)) is reversible by the negation of the step size h. Likewise as the Jacobians of the transformations

$$(\boldsymbol{\theta}, \mathbf{p}) \mapsto (\boldsymbol{\theta}, \mathbf{p} - h\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})/2) \tag{4.21}$$

$$(\boldsymbol{\theta}, \mathbf{p}) \mapsto (\boldsymbol{\theta} + h\mathbf{M}^{-1}\mathbf{p}, \mathbf{p}) \tag{4.22}$$

have unit determinant then the volume is preserved. Using this integrator, the total energy is only approximately preserved and then certain bias will be introduced into the joint distribution, where a Metropolis accept-reject step is employed to correct the bias. For a deterministic mapping $(\boldsymbol{\theta}, \mathbf{p}) \mapsto (\boldsymbol{\theta}^*, \mathbf{p}^*)$ obtained from a number of leapfrog integration steps, the acceptance probability is

$$\rho = \min\left(1, H(\boldsymbol{\theta}, \mathbf{p}) - H(\boldsymbol{\theta}^*, \mathbf{p}^*)\right) \tag{4.23}$$

and owing to the reversibility of the dynamics the joint density and hence the marginals $\pi(\boldsymbol{\theta})$ and $\pi(\mathbf{p})$ are left invariant.

The overall HMC sampling from the invariant posterior density $\pi(\boldsymbol{\theta}|\mathbf{X})$ can be considered as a Gibbs sampler where the momentum $\mathbf{p}$ acts simply as an auxiliary variable drawn from a symmetric density

$$\mathbf{p}_{t+1}|\boldsymbol{\theta}_t \sim \pi(\mathbf{p}_{t+1}|\boldsymbol{\theta}_t) = \pi(\mathbf{p}_{t+1}) = \mathcal{N}(\mathbf{p}_{t+1}|\mathbf{0}, \mathbf{M}), \tag{4.24}$$

$$\boldsymbol{\theta}_{t+1}|\mathbf{p}_{t+1} \sim \pi(\boldsymbol{\theta}_{t+1}|\mathbf{p}_{t+1}), \tag{4.25}$$

where samples of $\boldsymbol{\theta}_{t+1}$ from $\pi(\boldsymbol{\theta}_{t+1}|\mathbf{p}_{t+1})$ are obtained by running leapfrog integrator from initial values $\mathbf{p}_{t+1}$ and $\boldsymbol{\theta}_{t+1}$ for a number of steps to give proposed moves $\boldsymbol{\theta}^*$ and $\mathbf{p}^*$ and accepting or rejecting with probability $\min\left(1, \exp\left(H(\boldsymbol{\theta}_t, \mathbf{p}_{t+1}) - H(\boldsymbol{\theta}^*, \mathbf{p}^*)\right)\right)$. This Gibbs sampling scheme produces an ergodic, time reversible Markov chain satisfying detailed balance whose stationary marginal density is $\pi(\boldsymbol{\theta})$.

In HMC, the choice of step size h and the number of leapfrog steps can be manually tuned to make it satisfy certain accepting rate, or can be automatically tuned by "No-U-Turn" technique (Homan and Gelman, 2014). Similar to Riemann Manifold MALA, the mass matrix $\mathbf{M}$ can also be set based on Hamiltonian on a Riemann Manifold, see Girolami and Calderhead (2011) for more details.

Interim Summary Both of the two dynamical sampling methods, MALA and HMC, ultilize the gradient information of log density to explore the state space efficiently, and apply Metropolis step to be a correction of the discretization error introduced by numerical integration. However, in each iteration for generating one sample, we have to evaluate the full log likelihood and its gradient over the entire data set. When faced with large-scale data, the involved computation becomes dramatically expensive and even intractable in certain cases. To handle this issue, inspired by stochastic gradient descent (SGD) in optimization community, recent works explore several possibilities for Bayesian posterior sampling with large-scale data sets, which will be reviewed and discussed in the following section.

4.4 Stochastic Gradient Dynamical Sampling Methods

As mentioned before, though dynamical MCMC methods alleviate random walk behaviour, they are incapable of handling large-scale data due to intractable computation induced by the evaluation of full likelihood and its gradient over entire data set. Stochastic gradient dynamical methods overcome this issue from two aspects:

- instead of evaluating the log likelihood in each iteration over entire data set, a new random subset of $\mathbf{X}$ in each iteration is used to approximate it, i.e.,

$$\log p(\mathbf{X}|\boldsymbol{\theta}) \approx \frac{N}{n} \log p(\mathbf{X}_r) = \frac{N}{n} \sum_{i=1}^{n} \log p(\mathbf{x}_{r_i}|\boldsymbol{\theta}), \tag{4.26}$$

  where $\mathbf{X}_r = \{\mathbf{x}_{r_i}\}_{i=1}^{n}$ represents a random subset of $\mathbf{X}$. Thus, the noisy potential energy can be written as

$$\tilde{U}(\boldsymbol{\theta}) = -\frac{N}{n} \sum_{i=1}^{n} \log p(\mathbf{x}_{r_i}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}), \tag{4.27}$$

  Then, the stochastic force can be defined as

$$\tilde{\mathbf{f}}(\boldsymbol{\theta}) = -\nabla \tilde{U}(\boldsymbol{\theta}). \tag{4.28}$$

We introduce a crucial assumption on the distribution of the noisy force, which will be used for developing efficient sampling methods discussed later. Given the observed data $\{\mathbf{x}_i\}_{i=1}^N$ and the size of the random subset n is large enough for the central limit theorem to hold, we can make following assumption that the gradient noise follows a normal distribution with zero mean and covariance $\boldsymbol{\Sigma}(\boldsymbol{\theta})$,

$$\tilde{\mathbf{f}}(\boldsymbol{\theta}) = -\nabla U(\boldsymbol{\theta}) + \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta})). \tag{4.29}$$

It is easily observed that the covariance of the stochastic gradient noise depends on the size of subset and the current parameters. As the size of n increases, this Gaussian approximation becomes more accurate.

- Another way of overcoming computational issue is to omit the Metropolis correction step since the calculation of accepting probability requires evaluating the true potential energy, which cancels the benefits of using stochastic gradients.

Therefore, dynamical sampling methods based on stochastic gradient are a class of approximate sampling methods, which trade accuracy with computation. They still bring computational benefits and practical advantages, which will be shown later. Now we review several stochastic gradient dynamical sampling methods.

In the following description for various stochastic gradient dynamical sampling methods, we always assume the mass matrix (metric matrix) $\mathbf{M} = \mathbf{I}$ for simplicity. We leave how to select the metric matrix as future work.

4.4.1 Stochastic Gradient Langevin Dynamics (SGLD)

SGLD proposed by Welling and Teh (2011) is the first attempt of Bayesian sampling with large-scale data based on stochastic gradient. SGLD generates samples by simulating Brownian dynamics with stochastic gradient and annealed step sizes,

$$\mathrm{d}\boldsymbol{\theta} = \tilde{\mathbf{f}}(\boldsymbol{\theta})\mathrm{d}t + \mathcal{N}(\mathbf{0}, 2\mathrm{d}t\mathbf{I}), \tag{4.30}$$

where the stochastic force $\tilde{\mathbf{f}}(\boldsymbol{\theta})$ is defined from Eq. (4.27) and (4.28) based on a random subset of the entire data $\mathbf{X}$. After discretization, the update has the form

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + h_t\tilde{\mathbf{f}}(\boldsymbol{\theta}) + \mathcal{N}(\mathbf{0}, 2h_t\mathbf{I}), \tag{4.31}$$

where the step sizes $\{h_t\}_{t=0}^{T-1}$ have to be annealed to guarantee the convergence to the desired invariant distribution, i.e.,

$$\sum_{t=0}^{\infty} h_t = \infty, \quad \sum_{t=0}^{\infty} h_t^2 < \infty. \tag{4.32}$$

Intuitively, the first constraint ensures that parameters will reach the high probability regions no matter how far away it was initialized to, while the second ensures that the parameters will sample around the mode due to introduced variance by stochastic gradient. Typically, step sizes $h_t = a(b+t)^{-\gamma}$ are decayed polynomially with $\gamma \in (0.5, 1]$ and certain user-specified parameter a and b.

SGLD is a valid marriage between Brownian dynamics and stochastic gradient descent in optimization community. However, since the step sizes are reduces to zero, the mixing rate is reduced as well, and a large number of iterations are required to obtain a good coverage of the state space.

4.4.2 Stochastic Gradient Hamiltonian Monte Carlo (SGHMC)

SGHMC (Chen et al., 2014) proposed to apply second-order Langevin dynamics with a friction term that counteracts the effects of the noisy gradient approximation, maintaining the desired target distribution as the invariant distribution. Another difference of SGHMC from SGLD is that it allows fixed step size to produce faster mixing rate. We now introduce this method.

Recalling that we assume the noisy gradient follows an approximate normal distribution with certain variance as shown in Eq. (4.29), larger size of the subsets will induce more smaller variances. However, we want the subsets (mini batches of the data) to be small to obtain sought-after computational gains. In a wide range of practical settings, simply considering a a mini batch size on the order of hundreds of data points is sufficient for the central limit theorem to hold (Ahn et al., 2012).

For the Gaussian approximation of the gradient in a typical setting of numerical integration with associated step size h, one has

$$h\nabla\tilde{U}(\boldsymbol{\theta}) = h\nabla U(\boldsymbol{\theta}) + \sqrt{h}\sqrt{h\boldsymbol{\Sigma}(\boldsymbol{\theta})}\mathcal{N}(\mathbf{0},\mathbf{I}). \tag{4.33}$$

If we know the noise model $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ of the stochastic gradient, SGHMC simulates the following SDE (modified from standard HMC dynamics expressed in Eq. (4.16) and (4.17)) with associated h-discretization,

$$\begin{aligned} \mathrm{d}\boldsymbol{\theta} &= \mathbf{p}\mathrm{d}t, \\ \mathrm{d}\mathbf{p} &= -\nabla U(\boldsymbol{\theta})\mathrm{d}t - \frac{1}{2}h\boldsymbol{\Sigma}(\boldsymbol{\theta})\mathbf{p}\mathrm{d}t + \mathcal{N}(\mathbf{0}, h\boldsymbol{\Sigma}(\boldsymbol{\theta})\mathrm{d}t). \end{aligned} \tag{4.34}$$

where the noise term $\mathcal{N}(\mathbf{0}, h\boldsymbol{\Sigma}(\boldsymbol{\theta})\mathrm{d}t)$ is introduced by the stochastic gradient noise, and the "friction" term $-\frac{1}{2}h\boldsymbol{\Sigma}(\boldsymbol{\theta})\mathbf{p}\mathrm{d}t$ prevents the states to run far away and helps decrease

the energy $H(\boldsymbol{\theta}, \mathbf{p})$, and thus reducing the influence of the noise. This type of dynamical system is commonly referred as second-order Langevin dynamics in physics (Wang and Uhlenbeck, 1945). Note that the Brownian dynamics used in SGLD are first-order, which are a limiting case of the second-order Langevin dynamics when the friction term is large.

Chen et al. (2014) showed that $\pi(\boldsymbol{\theta}, \mathbf{p}) \propto \exp(-H(\boldsymbol{\theta}, \mathbf{p}))$ is the unique stationary distribution of the dynamics in Eq. (4.34). However, in practice, we do not know the true noise model $\boldsymbol{\Sigma}(\boldsymbol{\theta})$, instead, we have to rely on its estimate $\hat{\boldsymbol{\Sigma}}(\boldsymbol{\theta})$. SGHMC introduces a user-specified friction term A such that $A\mathbf{I} \succeq \frac{1}{2}h\hat{\boldsymbol{\Sigma}}(\boldsymbol{\theta})$ and consider the following dynamics,

$$\begin{aligned} \mathrm{d}\boldsymbol{\theta} &= \mathbf{p}\mathrm{d}t, \\ \mathrm{d}\mathbf{p} &= -\nabla U(\boldsymbol{\theta})\mathrm{d}t + \mathcal{N}(\mathbf{0}, h\boldsymbol{\Sigma}(\boldsymbol{\theta})\mathrm{d}t) - A\mathbf{p}\mathrm{d}t + \mathcal{N}(\mathbf{0}, 2(A\mathbf{I} - \frac{1}{2}h\hat{\boldsymbol{\Sigma}}(\boldsymbol{\theta}))\mathrm{d}t), \end{aligned} \tag{4.35}$$

resulting in the Algorithm 4.2.

In Chen et al. (2014), the authors showed when the estimate of $\boldsymbol{\Sigma}$ is exact, $\hat{\boldsymbol{\Sigma}}(\boldsymbol{\theta}) = \boldsymbol{\Sigma}(\boldsymbol{\theta})$, the dynamics of Eq. (4.35) yield the stationary distribution $\pi(\boldsymbol{\theta}, \mathbf{p}) \propto \exp(-H(\boldsymbol{\theta}, \mathbf{p}))$. Thus, to guarantee the performance of SGHMC empirically, one has to consider two aspects: one is to estimate the noise model $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ as accurate as possible though difficult in practice; the other is to set the hyperparameter A to guarantee the positive semidefiniteness of the matrix $A\mathbf{I} - \frac{1}{2}h\hat{\boldsymbol{\Sigma}}(\boldsymbol{\theta})$. One may attempt to use a large value of the friction term A and/or a small step size h. However, too-large a friction would essentially reduce SGHMC to SGLD, which is not desirable, as pointed out in Chen et al. (2014), while extremely small step size would significantly impact the computational efficiency.

4.4.3 Stochastic Gradient Nosé-Hoover Thermostat (SGNHT)

Recall that SGHMC suffers from inaccurate estimate of stochastic gradient noise model $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ and selection of friction of parameter A, SGNHT (Ding et al., 2014) attempted to adaptively fit to the noise without explicit estimation, an idea originally coming from the practice of sampling a canonical ensemble in statistical physics.

In statistical physics, a canonical ensemble represents the possible states of a system in thermal equilibrium with a heat bath at fixed temperature T_e. The probability of the states in a canonical ensemble follows the canonical distribution $\pi(\boldsymbol{\theta}, \mathbf{p}) \propto (-\beta H(\boldsymbol{\theta}, \mathbf{p}))$, where $\beta = 1/(k_B T_e)$, k_B is the Boltzmann constant. A critical characteristic of the canonical ensemble is that the system temperature, defined as the mean

Algorithm 4.2 Stochastic Gradient HMC (SGHMC)

Input: h, A, estimate $\hat{\mathbf{\Sigma}}(\boldsymbol{\theta})$.
Initialize $\boldsymbol{\theta}_0$, $\mathbf{p}_0$.
for $t = 1, 2, \ldots$ **do**
  Optionally, resample the momentum $\mathbf{p}$ as $\mathbf{p}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $(\boldsymbol{\theta}^{(0)}, \mathbf{p}^{(0)}) = (\boldsymbol{\theta}_t, \mathbf{p}_t)$;
  Simulate the dynamics in Eq. (4.35) as follows:
  for $\tau = 1, 2, \ldots, m$ **do**
    $\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} + h\mathbf{p}^{(\tau-1)}$
    $\mathbf{p}^{(\tau)} = \mathbf{p}^{(\tau-1)} - h\nabla\tilde{U}(\boldsymbol{\theta}^{(\tau)}) - hA\mathbf{p}^{(\tau-1)} + \mathcal{N}(\mathbf{0}, 2h(A\mathbf{I} - \frac{1}{2}h\hat{\mathbf{\Sigma}}(\boldsymbol{\theta})))$
  end for
  $(\boldsymbol{\theta}_{t+1}, \mathbf{p}_{t+1}) = (\boldsymbol{\theta}^{(m)}, \mathbf{p}^{(m)})$
end for

kinetic energy, satisfies the following thermal equilibrium condition,

$$k_B T_e = \mathbb{E}[\mathbf{p}^T \mathbf{p}]/D, \tag{4.36}$$

We can easily observe that all dynamics-based sampling methods approximate the canonical ensemble to generate samples. In Bayesian statistics, D is the dimension of the parameter of interest $\boldsymbol{\theta}$, and $k_B T_e = 1$ so that $\pi(\boldsymbol{\theta}, \mathbf{p}) \propto (-H(\boldsymbol{\theta}, \mathbf{p}))$ which leaves its marginal distribution as $\pi(\boldsymbol{\theta}) \propto \exp(-U(\boldsymbol{\theta}))$. As emphasized in Ding et al. (2014), the dynamics that correctly simulate the canonical ensemble must maintain the thermal equilibrium condition in Eq. (4.36).

We can easily verify that both of standard HMC and MALA using true gradient maintain the condition in Eq. (4.36). However, when using stochastic gradient $\nabla\tilde{U}(\boldsymbol{\theta})$, the dynamics of SGHMC might drift away from thermal equilibrium if $\mathbf{\Sigma}(\boldsymbol{\theta})$ is poorly estimated. To adaptively control the mean kinetic energy, SGNHT adopts the "thermostat" idea, which is widely used in molecular dynamics (Frenkel and Smit, 2001; Leimkuhler and Matthews, 2015). Concretely, SGNHT simulates the following second-order Langevin dynamics with extended variable (with $\beta = 1$),

$$\begin{aligned} \mathrm{d}\boldsymbol{\theta} &= \mathbf{p}\mathrm{d}t\,, \\ \mathrm{d}\mathbf{p} &= -\nabla\tilde{U}(\boldsymbol{\theta})\mathrm{d}t - \xi\mathbf{p}\mathrm{d}t + \mathcal{N}(\mathbf{0}, 2A\mathrm{d}t) \\ \mathrm{d}\xi &= \mu^{-1}\left(\mathbf{p}^T\mathbf{p} - D\right)\mathrm{d}t\,, \end{aligned} \tag{4.37}$$

where the auxiliary variable $\xi \in \mathbb{R}$ is governed by a Nosé-Hoover device (Hoover, 1991; Nosé, 1984) via a negative feedback mechanism, i.e. when the instantaneous

Algorithm 4.3 Stochastic Gradient Nośe-Hoover Thermostat (SGNHT)

Input: h, A.
Initialize $\boldsymbol{\theta}_0$, $\mathbf{p}_0 \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, and $\xi_0 = A$.
for $t = 1,2,\ldots$ **do**
  $\mathbf{p}_t = \mathbf{p}_{t-1} - \nabla\tilde{U}(\boldsymbol{\theta}_{t-1})h - \xi_{t-1}\mathbf{p}_{t-1}h + \sqrt{2Ah}\mathcal{N}(\mathbf{0},\mathbf{I})$;
  $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{p}_t h$;
  $\xi_t = \xi_{t-1} + (\mathbf{p}^T\mathbf{p}/D - 1)h$;
end for

system temperature is below the target temperature, the "dynamical friction" ξ would decrease allowing an increase of temperature, while ξ would increase when the temperature is above the target. The parameter μ is a scaling factor, in practice, we set it as $\mu = D$ to ensure good practical performance (Ding et al., 2014). We summarize SGNHT in Algorithm 4.3.

A key assumption made by SGNHT is that the noise model of stochastic gradient approximation in Eq. (4.29) has a *constant* covariance matrix, i.e., $\Sigma(\boldsymbol{\theta}) = \sigma^2\mathbf{I}$. Therefore, with associated h-discretization, the dynamics in Eq. (4.37) could also be written as

$$\begin{aligned} \mathrm{d}\boldsymbol{\theta} &= \mathbf{p}\mathrm{d}t\,, \\ \mathrm{d}\mathbf{p} &= -\nabla U(\boldsymbol{\theta})\mathrm{d}t + \mathcal{N}(\mathbf{0}, h\sigma^2\mathrm{d}t) - \xi\mathbf{p}\mathrm{d}t + \mathcal{N}(\mathbf{0}, 2A\mathrm{d}t) \\ \mathrm{d}\xi &= \mu^{-1}\left(\mathbf{p}^T\mathbf{p} - D\right)\mathrm{d}t\,. \end{aligned} \tag{4.38}$$

Then the following proposition showed that the dynamics above has the stationary invariant distribution as our desired target distribution.

Proposition 3. *(See Jones and Leimkuhler (2011)) The SGNHT method* (4.38) *preserves the modified canonical (stationary) distribution*

$$\pi(\boldsymbol{\theta},\mathbf{p},\xi) = \frac{1}{Z}\exp\left(-H(\boldsymbol{\theta},\mathbf{p})\right)\exp\left(-\frac{\mu}{2}(\xi-\bar{\xi})^2\right), \tag{4.39}$$

where Z is the normalizing constant, $H(\boldsymbol{\theta},\mathbf{p}) = \mathbf{p}^T\mathbf{p}/2 + U(\boldsymbol{\theta})$ is the Hamiltonian, and

$$\bar{\xi} = A + \frac{h\sigma^2}{2}\,. \tag{4.40}$$

Proposition 3 tells us that the SGNHT method can adaptively dissipate excess noise pumped into the system while maintaining the correct distribution. The variance of the gradient noise, σ^2, does not need to be known a priori. As long as σ^2 is constant,

the auxiliary variable ξ will be able to automatically find its mean value (4.36) on the fly. However, with a parameter-dependent $\mathbf{\Sigma}(\boldsymbol{\theta})$, the SGNHT method (4.38) would not produce the required target distribution (4.39).

Ding et al. (2014) claimed that it is reasonable to assume the covariance matrix $\mathbf{\Sigma}(\boldsymbol{\theta})$ is constant when the size of the dataset, N, is large, in which case the variance of the posterior of $\boldsymbol{\theta}$ is small. The magnitude of the posterior variance does not actually relate to the constancy of the $\mathbf{\Sigma}$, however, in general $\mathbf{\Sigma}$ is not constant. Simply assuming the non-constancy of $\mathbf{\Sigma}$ can have a significant impact on the performance of the method (most notably the stability measured by the largest usable step size). Therefore, it is essential to have an approach that can handle parameter-dependent noise. In the following section we propose a covariance-controlled thermostat that can effectively dissipate parameter-dependent noise while maintaining the target stationary distribution.

4.5 Covariance-Controlled Adaptive Langevin Thermostat

As mentioned in the previous section, the SGNHT method can only dissipate noise with a constant covariance matrix. When the covariance matrix becomes parameter-dependent, in general a parameter-dependent covariance matrix does not imply the required "thermal equilibrium", i.e. the system cannot be expected to converge to the desired invariant distribution (4.39), typically resulting in poor estimation of functions of parameters of interest. In fact, in that case it is not clear whether or not there exists an invariant distribution at all.

In order to construct a stochastic-dynamical system that preserves the canonical distribution, we suggest adding a suitable damping (viscous) term to effectively dissipate the parameter-dependent gradient noise. To this end, we propose the following covariance-controlled adaptive Langevin (CCAdL) thermostat (with $\beta = 1$)

$$
\begin{aligned}
\mathrm{d}\boldsymbol{\theta} &= \mathbf{p}\mathrm{d}t\,,\\
\mathrm{d}\mathbf{p} &= -\nabla U(\boldsymbol{\theta})\mathrm{d}t + \mathcal{N}(\mathbf{0}, h\mathbf{\Sigma}(\boldsymbol{\theta})\mathrm{d}t) - (h/2)\mathbf{\Sigma}(\boldsymbol{\theta})\mathbf{p}\mathrm{d}t - \xi\mathbf{p}\mathrm{d}t + \mathcal{N}(\mathbf{0}, 2A\mathrm{d}t)\,,\\
\mathrm{d}\xi &= \mu^{-1}\left(\mathbf{p}^T\mathbf{p} - D\right)\mathrm{d}t\,.
\end{aligned}
\tag{4.41}
$$

Theorem 3. *The CCAdL thermostat* (4.41) *preserves the modified Gibbs (stationary) distribution*

$$
\rho(\boldsymbol{\theta}, \mathbf{p}, \xi) = \frac{1}{Z}\exp\left(-H(\boldsymbol{\theta}, \mathbf{p})\right)\exp\left(-\frac{\mu}{2}(\xi - A)^2\right)\,. \tag{4.42}
$$

Proof. The Fokker-Planck equation corresponding to (4.41) is

$$\begin{aligned}
\partial_t \rho(\boldsymbol{\theta}, \mathbf{p}, \xi) =& -\mathbf{p} \cdot \nabla_{\boldsymbol{\theta}} \rho + \nabla U(\boldsymbol{\theta}) \cdot \nabla_{\mathbf{p}} \rho \\
&+ \frac{h}{2} \nabla_{\mathbf{p}} \cdot (\Sigma(\boldsymbol{\theta}) \mathbf{p} \rho) + \xi \nabla_{\mathbf{p}} \cdot (\mathbf{p} \rho) - \mu^{-1} \left(\mathbf{p}^T \mathbf{p} - D\right) \nabla_{\xi} \rho \\
&+ \frac{h}{2} \nabla_{\mathbf{p}} \cdot (\Sigma(\boldsymbol{\theta}) \nabla_{\mathbf{p}} \rho) + A \nabla_{\mathbf{p}} \cdot \nabla_{\mathbf{p}} \rho \\
=& \mathbf{p} \cdot \nabla U(\boldsymbol{\theta}) \rho - \nabla U(\boldsymbol{\theta}) \cdot (\rho \mathbf{p}) \\
&+ \frac{h}{2} \nabla_{\mathbf{p}} \cdot (\Sigma(\boldsymbol{\theta}) \mathbf{p} \rho) + \xi D \rho - \xi \mathbf{p} \cdot \mathbf{p} + \left(\mathbf{p}^T \mathbf{p} - D\right) (\xi - A) \rho \\
&- \frac{h}{2} \nabla_{\mathbf{p}} \cdot (\Sigma(\boldsymbol{\theta}) \mathbf{p} \rho) - AD\rho + A\mathbf{p} \cdot (-\mathbf{p} \rho) \\
=& 0.
\end{aligned}$$

Thus, the vanishing of the Fokker-Planck equation guarantees the proposed dynamics has the desired stationary distribution. □

The above proof needs some preliminary knowledge about Fokker-Planck equation for SDEs, which we provide in Appendix A.

The incorporation of the parameter-dependent covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ in (4.41) is intended to offset the covariance matrix coming from the gradient approximation. However, in practice, one does not know $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ a priori. Thus instead one must estimate $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ during the simulation, a task which will be addressed in Section 4.5.1. This procedure is related to the method used in the SGHMC in Eq. (4.35).

Although both CCAdL (4.41) and SGHMC (4.35) preserve their respective invariant distributions, let us note several advantages of the former over the latter in practice:

(i) CCAdL and SGHMC both require estimation of the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ during simulation, which can be costly in high dimension. In numerical experiments, we have found that simply using the diagonal of the matrix, at significantly reduced computational cost, works quite well in CCAdL. By contrast, it is difficult to find a suitable value of the parameter A in SGHMC since one has to make sure the matrix $A\mathbf{I} - h\boldsymbol{\Sigma}(\boldsymbol{\theta})/2$ is positive semi-definite. One may attempt to use a large value of the "effective friction" A and/or a small stepsize h. However, too-large a friction would essentially reduce SGHMC to SGLD, which is not desirable, as pointed out in Chen et al. (2014), while extremely small stepsize would significantly impact the computational efficiency.

(ii) Estimation of the covariance matrix $\mathbf{\Sigma}(\boldsymbol{\theta})$ unavoidably introduces additional noise in both CCAdL and SGHMC. Nonetheless, CCAdL can still effectively control the system temperature (i.e. maintaining the correct distribution of the momenta) due to the use of the stabilizing Nosé-Hoover control, while in SGHMC poor estimation of the covariance matrix may lead to significant deviations of the system temperature (as well as the distribution of the momentum), resulting in poor sampling of the parameters of interest.

4.5.1 Covariance Estimation of Noisy Gradient

Under the assumption that the noise of the stochastic gradient follows a normal distribution, we apply a similar method to that of Ahn et al. (2012) to estimate the covariance matrix associated with the noisy gradient. If we let $g(\boldsymbol{\theta};\mathbf{x}) = \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{x}|\boldsymbol{\theta})$ and assume that the size of subset n is large enough for the central limit theorem to hold, we have

$$\frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{\theta}_t;\mathbf{x}_{r_i}) \sim \mathcal{N}\left(\mathbb{E}_{\mathbf{x}}[g(\boldsymbol{\theta}_t;\mathbf{x})], \frac{1}{n}\mathbf{\Gamma}_t\right), \tag{4.43}$$

where $\mathbf{\Gamma}_t = \mathrm{Cov}[g(\boldsymbol{\theta}_t;\mathbf{x})]$ is the covariance of the gradient at $\boldsymbol{\theta}_t$. Given that the noisy (stochastic) gradient based on current subset $\nabla\tilde{U}(\boldsymbol{\theta}_t) = -\frac{N}{n}\sum_{i=1}^{n} g(\boldsymbol{\theta}_t;\mathbf{x}_{r_i}) - \nabla \log \pi(\boldsymbol{\theta}_t)$, and the clean (full) gradient $\nabla U(\boldsymbol{\theta}_t) = -\sum_{i=1}^{N} g(\boldsymbol{\theta}_t;\mathbf{x}_i) - \nabla \log \pi(\boldsymbol{\theta}_t)$, we have $\mathbb{E}_{\mathbf{x}}[\nabla\tilde{U}(\boldsymbol{\theta}_t)] = \mathbb{E}_{\mathbf{x}}[\nabla U(\boldsymbol{\theta}_t)]$, and thus

$$\nabla\tilde{U}(\boldsymbol{\theta}_t) = \nabla U(\boldsymbol{\theta}_t) + \mathcal{N}\left(\mathbf{0}, \frac{N^2}{n}\mathbf{\Gamma}_t\right), \tag{4.44}$$

i.e. $\mathbf{\Sigma}(\boldsymbol{\theta}_t) = N^2\mathbf{\Gamma}_t/n$. Assuming $\boldsymbol{\theta}_t$ does not change dramatically over time, we use the moving average update to estimate $\mathbf{\Gamma}_t$,

$$\hat{\mathbf{\Gamma}}_t = (1-\kappa_t)\hat{\mathbf{\Gamma}}_{t-1} + \kappa_t \mathbf{V}(\boldsymbol{\theta}_t), \tag{4.45}$$

where $\kappa_t = 1/t$, and

$$\mathbf{V}(\boldsymbol{\theta}_t) = \frac{1}{n-1}\sum_{i=1}^{n} \left(g(\boldsymbol{\theta}_t;\mathbf{x}_{r_i}) - \bar{g}(\boldsymbol{\theta}_t)\right)\left(g(\boldsymbol{\theta}_t;\mathbf{x}_{r_i}) - \bar{g}(\boldsymbol{\theta}_t)\right)^T \tag{4.46}$$

is the empirical covariance of gradient. $\bar{g}(\boldsymbol{\theta}_t)$ represents the mean gradient of the log likelihood computed from a subset. As proved in Ahn et al. (2012), this estimator has a convergence order of $O(1/N)$. The selection of κ_t (i.e., a fixed value or changing value w.r.t time) depends on different applications, which needs certain cross-validating procedure. Here we use $\kappa_t = 1/t$ suggested by Ahn et al. (2012) and it works well in our

Algorithm 4.4 Covariance-Controlled Adaptive Langevin (CCAdL)

Input: h, A, $\{\kappa_t\}_{t=1}^{\infty}$.
Initialize $\boldsymbol{\theta}_0$, $\mathbf{p}_0$, $\boldsymbol{\Gamma}_0$, and $\xi_0 = A$.
for $t = 1, 2, \ldots,$ **do**
  $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{p}_{t-1} h$;
  Estimate $\hat{\boldsymbol{\Gamma}}_t$ using Eq. ((4.45));
  $\mathbf{p}_t = \mathbf{p}_{t-1} - \nabla \tilde{U}(\boldsymbol{\theta}_t) h - \frac{h}{2} \frac{N^2}{n} \hat{\boldsymbol{\Gamma}}_t \mathbf{p}_{t-1} h - \xi_{t-1} \mathbf{p}_{t-1} h + \sqrt{2Ah}\mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $\xi_t = \xi_{t-1} + \left(\mathbf{p}_t^T \mathbf{p}_t / D - 1\right) h$;
end for

experiments. We can observe that in a long run of the estimation, it might be asymptotically converging to a fixed value. We leave the issue of how to estimate an optimal parameter-dependent covariance as future work.

As already mentioned, estimating the full covariance matrix is computationally intractable in high dimension. However, we have found that employing a diagonal approximation of the covariance matrix (i.e. only estimating the variance along each dimension of the noisy gradient), works quite well in practice, as demonstrated in Section 4.6.

The procedure of the CCAdL method is summarized in Algorithm 4.4, where we simply used $\beta = 1$, and $\mu = D$ in order to be consistent with the original implementation of SGNHT Ding et al. (2014).

Note that this is a simple, first-order (in terms of the stepsize) algorithm. A recent article Leimkuhler and Shang (2015) has introduced higher order of accuracy schemes which can improve accuracy, but our interest here is in the direct comparison of the underlying machinery of SGHMC, SGNHT and CCAdL, so we avoid further modifications and enhancements related to timestepping at this stage.

In the following section, we compare the newly-established CCAdL method with SGHMC and SGNHT on various machine learning tasks to demonstrate the benefits of CCAdL in Bayesian sampling with a noisy gradient.

4.6 Numerical Experiments

4.6.1 Bayesian Inference for a Gaussian Distribution

We first compare the performance of the newly-established CCAdL method with SGHMC and SGNHT for a simple task using synthetic data, i.e. Bayesian inference of both the mean and variance of a one-dimensional normal distribution. We apply the same experimental setting as in Ding et al. (2014). We generated $N = 100$ samples from the standard normal distribution $\mathcal{N}(0,1)$. We used the likelihood function of $\mathcal{N}(\mathbf{x}_i|\mu,\gamma^{-1})$ and assigned Normal-Gamma distribution as their prior distribution, i.e.

$$\mu,\gamma \sim \mathcal{N}(\mu|0,\gamma)\mathrm{Gam}(\gamma|1,1). \tag{4.47}$$

Then the corresponding posterior distribution is another Normal-Gamma distribution,

$$(\mu,\gamma)|\mathbf{X} \sim \mathcal{N}(\mu|\mu_N,(\kappa_N\gamma)^{-1})\mathrm{Gam}(\gamma|\alpha_N,\beta_N), \tag{4.48}$$

with

$$\mu_N = \frac{N\bar{\mathbf{x}}}{N+1}, \qquad \kappa_N = 1+N, \qquad \alpha_N = 1+\frac{N}{2}, \qquad \beta_N = 1+\sum_{i=1}^{N}\frac{(\mathbf{x}_i-\bar{\mathbf{x}})^2}{2}+\frac{N\bar{\mathbf{x}}^2}{2(1+N)},$$

where $\bar{\mathbf{x}} = \sum_{i=1}^{N}\mathbf{x}_i/N$. The posterior marginals are

$$\mu|\mathbf{X} \sim \mathrm{T}_2\left(\mu|\mu_N,\beta_N/(\alpha_N\kappa_N)\right) \tag{4.49}$$

$$\gamma|\mathbf{X} \sim \mathrm{Gam}(\gamma|\alpha_N,\beta_N), \tag{4.50}$$

where $\mathrm{T}_\nu(\cdot)$ represents the Student t distribution with ν as its degree of freedom. For more details of derivation of conjugate Bayesian analysis of the Gaussian distribution, see Murphy (2007).

A random subset of size $n = 10$ was selected at each timestep to approximate the full gradient, resulting in the following stochastic gradients,

$$\nabla_\mu\tilde{U} = (N+1)\mu\gamma - \frac{\gamma N}{n}\sum_{i=1}^{n}\mathbf{x}_{r_i}, \qquad \nabla_\gamma\tilde{U} = 1-\frac{N+1}{2\gamma}+\frac{\mu^2}{2}+\frac{N}{2n}\sum_{i=1}^{n}(\mathbf{x}_{r_i}-\mu)^2.$$

It can be seen that the variance of the stochastic gradient noise is no longer constant and actually depends on the size of the subset, n, and the values of μ and γ in each iteration. This directly violates the constant noise variance assumption of SGNHT Ding et al. (2014), while CCAdL adjusts to the varying noise variance.

The marginal distributions of μ and γ obtained from various methods with different combinations of h and A were compared and plotted in Figure 4.1, with Table 4.1

consisting of the corresponding root mean square error (RMSE) of the distribution and autocorrelation time from 10^6 samples. The autocorrelation time is defined as $1+2\sum_{s=1}^{\infty}\omega(s)$ with $\omega(s)$ the autocorrelation at lag s (Neal, 1993).

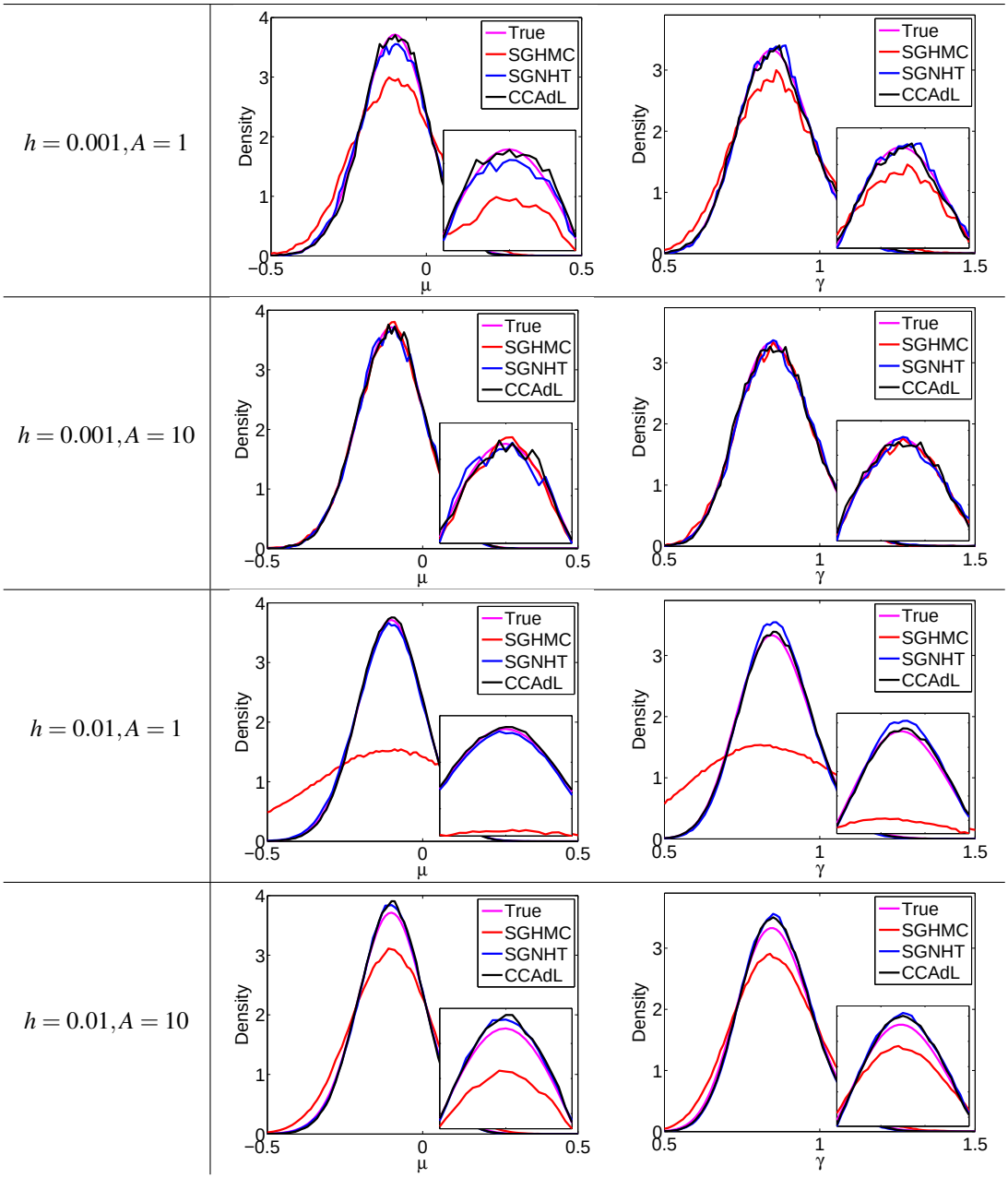


Figure 4.1: Comparisons of marginal distribution (density) of μ (top row) and γ (bottom row) with various values of h and A indicated in each column. The peak region is highlighted in the inset.

In most of the cases, both SGNHT and CCAdL easily outperform the SGHMC method possibly due to the presence of the Nosé-Hoover device, with SGHMC only showing superiority with small values of h and large value of A, neither of which is

Table 4.1: Comparisons of (RMSE, Autocorrelation time) of (μ, γ) of various methods for Bayesian inference of Gaussian mean and variance.

Methods	$h = 0.001, A = 1$	$h = 0.001, A = 10$	$h = 0.01, A = 1$	$h = 0.01, A = 10$
SGHMC	$(0.0148, 236.12)$	$(\mathbf{0.0029}, \mathbf{333.04})$	$(0.0531, 29.78)$	$(0.0132, 39.33)$
SGNHT	$(0.0037, 238.32)$	$(0.0035, 406.71)$	$(0.0044, 26.71)$	$(0.0043, 55.00)$
CCAdL	$(\mathbf{0.0034}, \mathbf{238.06})$	$(0.0031, 402.45)$	$(\mathbf{0.0021}, \mathbf{26.71})$	$(\mathbf{0.0035}, \mathbf{54.43})$

desirable in practice as discussed in Section 4.5. Between SGNHT and the newly-proposed CCAdL method, the latter achieves better performance in each of the cases investigated, highlighting the importance of the covariance control with parameter-dependent noise.

4.6.2 Large-scale Bayesian Logistic Regression

We then consider a Bayesian logistic regression model trained on the benchmark MNIST dataset for binary classification of digits 7 and 9 using $12,214$ training data points, with a test set of size 2037. A 100-dimensional random projection of the original features was used. We used the likelihood function

$$p\left(\mathbf{w}|\{\mathbf{x}_i, y_i\}_{i=1}^N\right) \propto \prod_{i=1}^N 1/\left(1+\exp(-y_i \mathbf{w}^T \mathbf{x}_i)\right), \tag{4.51}$$

and the prior distribution

$$p(\mathbf{w}) \propto \exp(-\mathbf{w}^T \mathbf{w}/2), \tag{4.52}$$

respectively. A subset of size $n = 500$ was used at each timestep. Since the dimensionality of this problem is not that high, a full covariance estimation was used for CCAdL.

We investigate the convergence speed of each method through measuring test log likelihood using posterior mean against the number of passes over the entire dataset, see Figure 4.2 (first column). CCAdL displays significant improvements over SGHMC and SGNHT with different values of h and A: (1) CCAdL converges much faster than the other two, which also indicates its faster mixing speed and shorter burn-in period; (2) CCAdL shows robustness in different values of the "effective friction" A, with SGHMC and SGNHT relying on a relative large value of A (especially the SGHMC method) which is intended to dominate the gradient noise.

To compare the sample quality obtained from each method, Figure 4.2 (second column) plots the two-dimensional marginal posterior distribution in randomly-selected

dimensions of 2 and 5 based on 10^6 samples from each method after the burn-in period (i.e. we start to collect samples when the test log likelihood stabilizes). The true (reference) distribution was obtained by a sufficiently long run of standard HMC. The step size was automatically tuned by No-U-Turn technique by Homan and Gelman (2014). We implemented 10 runs of standard HMC and found there was no variation between these runs, which guarantees its qualification as the true (reference) distribution. Again, CCAdL shows much better performance than SGHMC and SGNHT. Note that the SGHMC does not even fit in the region of the plot, and in fact it shows significant deviation even in the estimation of the mean.

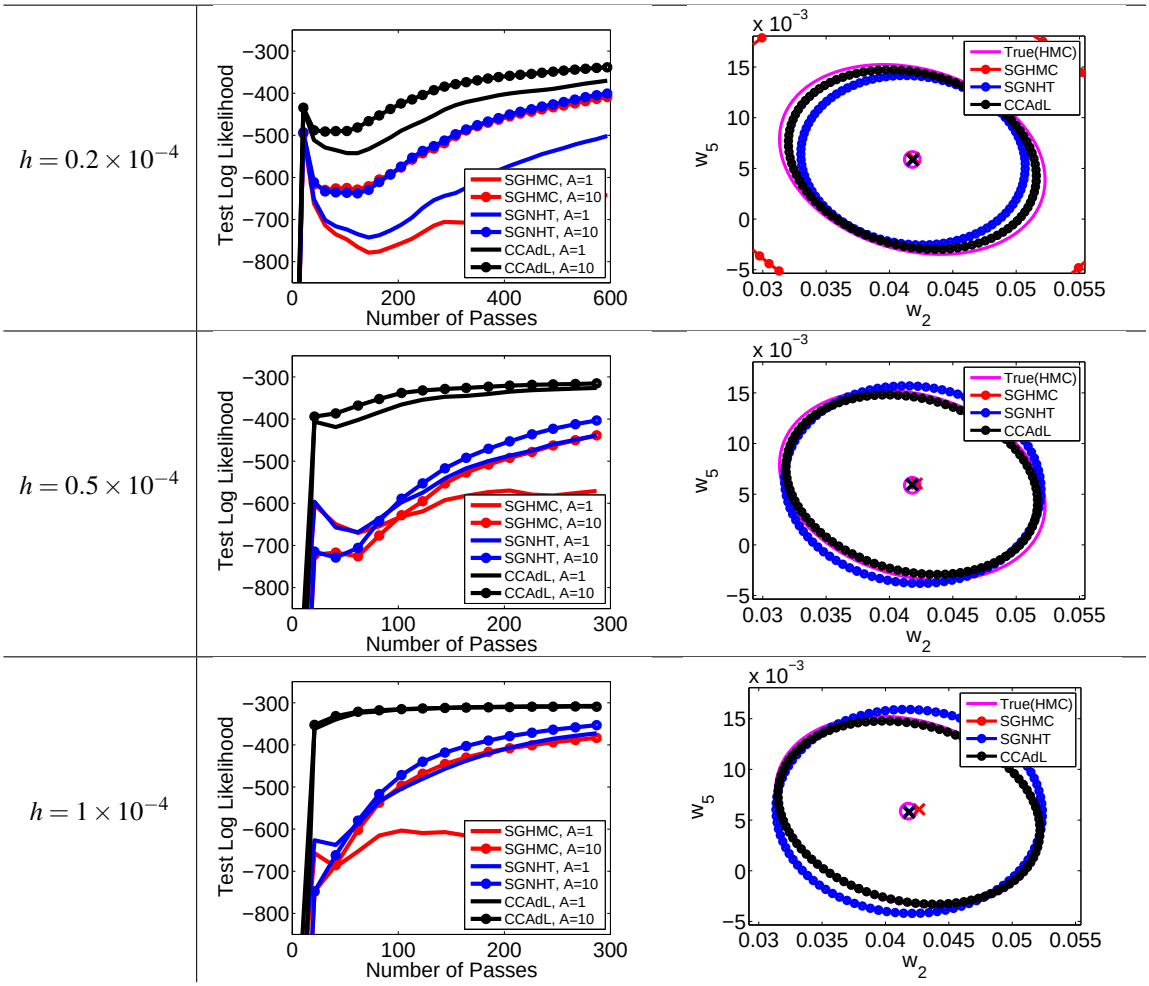


Figure 4.2: Comparisons of Bayesian Logistic Regression of various methods on the MNIST dataset of digits 7 and 9 with various values of h and A: (first column) test log likelihood using posterior mean against number of passes over the entire dataset; (second column) two-dimensional marginal posterior distribution in (randomly selected) dimensions 2 and 5 with $A = 10$ fixed, based on 10^6 samples from each method after the burn-in period (i.e. we start to collect samples when the test log likelihood stabilizes). Magenta circle is the true (reference) posterior mean obtained from standard HMC, and crosses represent the sample mean computed from various methods. Ellipses represent iso-probability contours covering 95% probability mass. Note that the contour of SGHMC is well beyond the scale of figure and thus we do not include it here.

4.6.3 Discriminative Restricted Boltzmann Machine (DRBM)

DRBM (Larochelle and Bengio, 2008) is a self-contained non-linear classifier, and the gradient of its discriminative objective can be explicitly computed. Different from traditional RBMs (Smolensky, 1986), DRBM adds an extra visible layer showing the class label of the observed covariates, as shown in Figure 4.3.

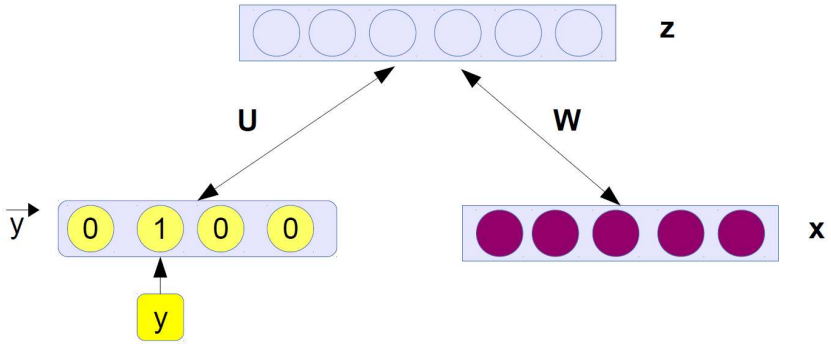


Figure 4.3: Model illustration of Discriminative Restricted Boltzmann Machine (DRBM). $\{\mathbf{x}, y\}$ is the pair of input data and its class label, $\overrightarrow{y}$ is the one-hot coding for the class label y, and $\mathbf{z}$ represents the hidden variables. $\mathbf{W}$ is the weight matrix connecting the input and hidden layer, while $\mathbf{U}$ connects the hidden and output layer. In this illustration, we ignore the bias vector for each layer, see text for complete models details.

A DRBM with N_h hidden units is a parametric model to describe the joint distribution between the hidden layer $\mathbf{z} = (z_1, z_2, \ldots, z_{N_h})$, the observed variables $\mathbf{x} \in \mathbb{R}^D$, and the class label y, which has the form,

$$p(y, \mathbf{x}, \mathbf{z}) \sim \exp\left(-E(y, \mathbf{x}, \mathbf{z})\right), \tag{4.53}$$

where the energy function

$$E(y, \mathbf{x}, \mathbf{z}) = -\mathbf{z}^T \mathbf{W}\mathbf{x} - \mathbf{b}^T \mathbf{x} - \mathbf{c}^T \mathbf{z} - \mathbf{d}^T \overrightarrow{y} - \mathbf{z}^T \mathbf{U} \overrightarrow{y} \tag{4.54}$$

with parameters $\mathbf{\Theta} = (\mathbf{W}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{U})$ and the vector $\overrightarrow{y}$ is one-hot coding for class label, $\overrightarrow{y} = (1_{y=i})_{i=1}^{C}$ for C classes.

DRBM directly optimizes $p(y|\mathbf{x})$ instead of the joint distribution $p(y, \mathbf{x})$, since one is ultimately only interested in correct classification,

$$\mathcal{L}(\mathbf{X}; \mathbf{\Theta}) = -\sum_{i=1}^{N} \log p(y_i | \mathbf{x}_i; \mathbf{\Theta}), \tag{4.55}$$

with the conditional distribution

$$p(y|\mathbf{x};\boldsymbol{\Theta}) = \frac{\exp(d_y)\prod_{j=1}^{N}\left(1+\exp(c_j+U_{jy}+\sum_i W_{ji}x_i)\right)}{\sum_{y'}\exp(d_{y'})\prod_{j=1}^{N}\left(1+\exp(c_j+U_{jy'}+\sum_i W_{ji}x_i)\right)}. \tag{4.56}$$

Since this conditional distribution can be computed exactly, we can also evaluate its gradient tractably,

$$\frac{\partial \log p(y_i|\mathbf{x}_i)}{\partial\theta} = \sum_j \mathrm{sigm}\left(o_{yj}(\mathbf{x}_i)\right)\frac{\partial o_{yj}(\mathbf{x}_i)}{\partial\theta} - \sum_{j,y'}\mathrm{sigm}\left(o_{y'j}(\mathbf{x}_i)\right)p(y'|\mathbf{x}_i)\frac{\partial o_{y'j}(\mathbf{x}_i)}{\partial\theta}, \tag{4.57}$$

where the sigmoid function $\mathrm{sigm}(a) = 1/(1+\exp(-a))$, and $o_{ji}(\mathbf{x}) = c_j + \sum_k W_{jk}x_k + U_j y$. For more details of DRBM, we refer the readers to Larochelle and Bengio (2008) for more details.

We assume the prior distribution over parameters $\boldsymbol{\Theta}$ are flat, and thus we can sample $\boldsymbol{\Theta}$ directly from the conditional distribution $\prod_{i=1}^{N} p(y|\mathbf{x};\boldsymbol{\Theta})$. We trained a DRBM on different large-scale multi-class datasets from LIBSVM[1] dataset collection, including *connect-4*, *letter*, and *SensIT Vehicle acoustic*. The detailed information of these datasets are presented in Table 4.2.

We selected the number of hidden units using cross-validation to achieve their best results. Since the dimension of parameters, N_{d}, is relatively high, we only used diagonal covariance matrix estimation for CCAdL to significantly reduce the computational cost, i.e. only estimating the variance along each dimension. The size of the subset was chosen as 500–1000 to obtain a reasonable variance estimation. For each dataset, we chose the first 20% of the total number of passes over the entire dataset as the burn-in period, and collected the remaining samples for prediction.

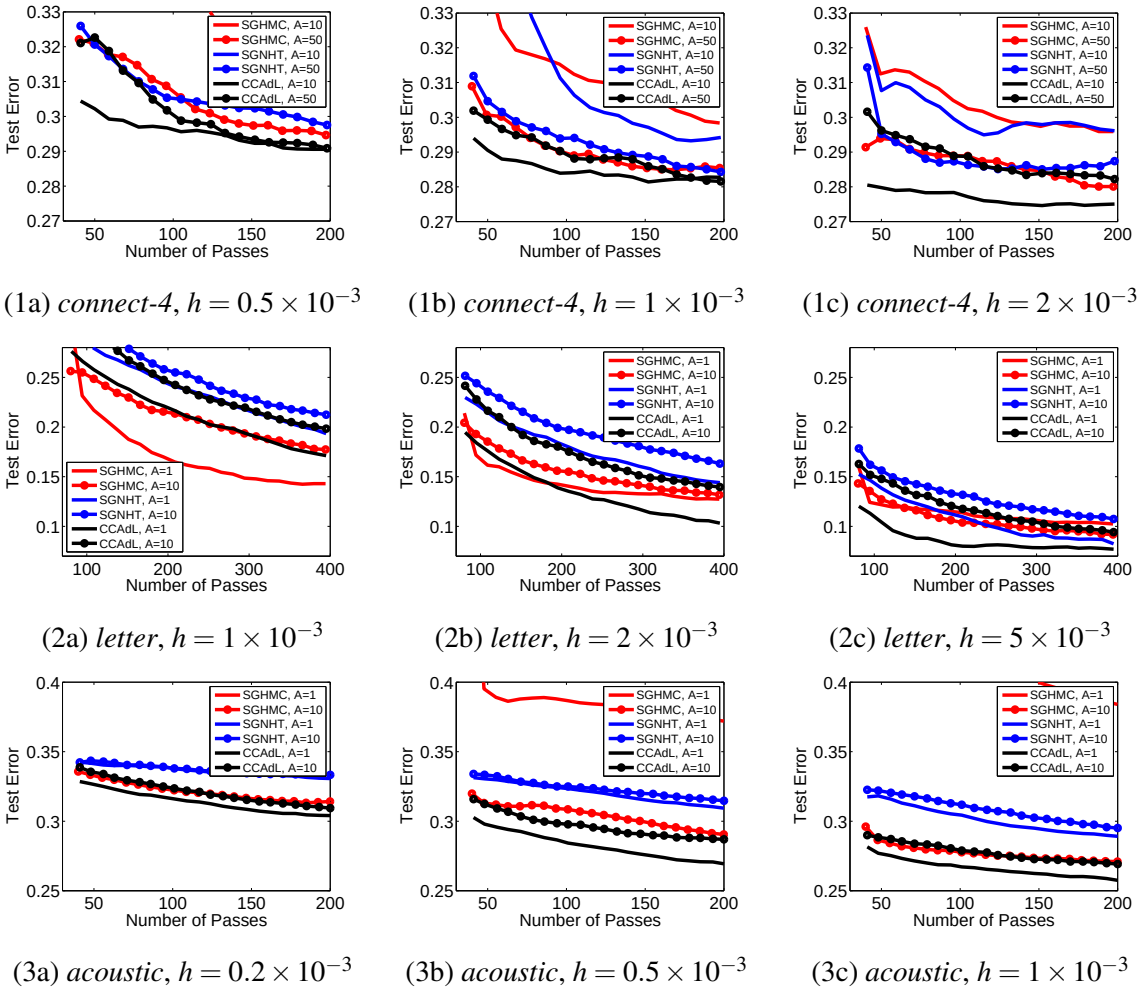
Table 4.2: Datasets used in DRBM with corresponding parameter configurations.

Datasets	training/test set	classes	features	hidden units	number of parameters D
connect-4	54,046/13,511	3	126	20	2603
letter	10,500/5,000	26	16	100	4326
acoustic	78,823/19,705	3	50	20	1083

The error rate computed by various methods on the test set using posterior mean against number of passes over entire dataset was plotted in Figure 4.4. It can be observed that SGHMC and SGNHT only work well with a large value of the effective friction A, which corresponds to a strong random walk effect and thus slows down the

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html

convergence. On the contrary, CCAdL works reliably (much better than the other two) in a wide range of A, and more importantly in the large stepsize regime, which speeds up the convergence rate in relation to the computational work performed. It can be easily seen that the performance of SGHMC heavily relies on using a small value of h and large value of A, which significantly limits its usefulness in practice.



(1a) *connect-4*, $h = 0.5 \times 10^{-3}$ (1b) *connect-4*, $h = 1 \times 10^{-3}$ (1c) *connect-4*, $h = 2 \times 10^{-3}$

(2a) *letter*, $h = 1 \times 10^{-3}$ (2b) *letter*, $h = 2 \times 10^{-3}$ (2c) *letter*, $h = 5 \times 10^{-3}$

(3a) *acoustic*, $h = 0.2 \times 10^{-3}$ (3b) *acoustic*, $h = 0.5 \times 10^{-3}$ (3c) *acoustic*, $h = 1 \times 10^{-3}$

Figure 4.4: Comparisons of DRBM on datasets *connect-4* (top row), *letter* (middle row), and *acoustic* (bottom row) with various values of h and A indicated: test error rate of various methods using posterior mean against number of passes over the entire dataset.

4.7 Conclusions and Future Work

In this chapter, we have provided a systematic analysis on dynamical sampling methods for Bayesian posterior inference. We pointed out the key challenge faced by Bayesian inference in "Big Data" era, i.e., *how to handle the computational issues*

within large-scale data. Several recent developed stochastic gradient methods are reviewed.

The fundamental issue arising in these stochastic gradient sampling methods is how to deal with the noise introduced by stochastic gradient approximation. Particularly, we proposed a novel Covariance-Controlled Adaptive Langevin (CCAdL) formulation that can effectively dissipate parameter-dependent noise while maintaining a desirable invariant distribution. CCAdL combines ideas of SGHMC and SGNHT from the literature, but achieves significant improvements over each of these methods in practice. Our findings have been verified in large-scale machine learning applications. In particular, we have consistently observed that SGHMC relies on a small stepsize h and large friction A, which significantly reduces its usefulness in practice as discussed. The techniques presented in this article could be of use in the more general setting of large-scale Bayesian sampling and optimization, which we leave for future work.

A naive first-order discretization method for SDEs has been applied for CCAdL for fair comparison in this thesis. However, we point out that optimal design of splitting methods in ergodic SDE systems has been explored recently in the mathematics community (Abdulle et al., 2015; Leimkuhler and Matthews, 2015; Leimkuhler et al., 2015). Moreover, it has been shown Leimkuhler and Shang (2015) that a certain type of symmetric splitting method for the Ad-Langevin/SGNHT method with a clean (full) gradient inherits the superconvergence property (i.e. fourth order convergence to the invariant distribution for configurational quantities) recently demonstrated in the setting of Langevin dynamics (Leimkuhler and Matthews, 2013; Leimkuhler et al., 2015). We leave further exploration of this direction in the context of noisy gradients for future work.

Chapter 5

Conclusions

Employing the philosophy of integrating local information, we have developed several novel approaches to solve three important tasks in machine learning: aggregating probabilistic predictions, large-scale optimization and Bayesian posterior sampling. Now, we summarize our contributions, and point out potential research directions.

5.1 Contributions

The key contributions, summarized by chapter, are as follows.

Chapter 2

- We introduced a class of Rényi divergence aggregators which interpolate between linear opinion pools and log opinion pools, and show that they are the maximum entropy estimators for aggregation of beliefs potentially subject to bias. We also demonstrate this relationship practically via simulated and real problems. Particularly, we designed a real-world Kaggle-in-Class machine learning competition, ran the competition, and used the obtained competition results to verify our theoretical findings.

- We discovered the theoretical connection between Rényi divergence aggregators and machine learning markets, i.e., Rényi divergence aggregators can be directly implemented by machine learning markets with different isoelastic utilities. And we showed that the risk averseness of the isoelastic utility relates to the Rényi divergence parameter that is used to control the assumed individual

bias. This important connection unifies two streams of the research in aggregating beliefs, which also provides the possibility of implementing a general recipe of aggregating probabilistic predictions in machine learning context through an collaborative and incentivized market-based environment.

Chapter 3

- We introduced the Sep-CCSP problem, a general form covering a wide range of machine learning models, including ERM and separable linear constrained optimization. We analyzed the problem structure with large-scale data, and investigated the possibilities of developing efficient algorithms for solving this type of problems.

- We developed efficient stochastic block coordinate descent methods for large-scale Sep-CCSP problems, where adaptive stepsizes are used to accelerate the convergence and parallelization of separable block coordinates is possible. Particularly, we proposed scalable methods for large-scale Sep-CCSP with both strongly convex and non-strongly convex functions.

  – AdaSPDC proposed in Section 3.3 focuses on Sep-CCSP with strongly convex functions. It extends the previous method SPDC in a non-trivial way, where adaptive stepsizes are introduced to dramatically improve its linear convergence rate. Various types of ERM problems have been tested through extensive experiments to demonstrated its superior empirical performance compared with other state-of-the-art methods.

  – SP-BCD proposed in Section 3.4 handles Sep-CCSP with (general) non-strongly convex functions, which extends its applicability to more general problems, such as separable function minimization with linear constraints. We developed novel stepsize rules in SP-BCD to accommodate non-strongly convex Sep-CCSP. SP-BCD owns a sublinear convergence rate $O(1/T)$, which is the best rate achieved for general convex functions. We compared SP-BCD with other methods in different types of scenarios, such as matrix decomposition, Lasso and group Lasso, etc.

Chapter 4

- We provided a comprehensive review on the existing scalable dynamics-based sampling methods relying on stochastic approximation techniques. We compared these methods in a systematic way in terms of stepsizes, the variance of noisy stochastic gradients, and preservation of canonical distribution. In particular, we pointed out the existing drawbacks in these sampling schemes, either the ignorance of the variance introduced by stochastic gradient, or the improper handling of the variance in practice.

- We proposed a covariance-controlled adaptive Langevin thermostat that can effectively dissipate parameter-dependent noise variance while maintaining a desired target distribution. This novel proposal contains two important components, correction term for noise variance, and Nosé-Hoover thermostat, both of which contribute to maintain the system temperature, and thus allow larger stepsizes and faster convergence compared with existing methods. This proposal also brings a fresh message to the field of large-scale Bayesian posterior sampling, i.e., it is beneficial of incorporating the introduced noise variance in dynamics-based systems for efficient sampling.

5.2 Future Directions

Here, we discuss some preliminary investigations and potential future research directions aimed at extending some of our methods and ideas.

The Collaborative Mechanism for Crowdsourcing Prediction

In Chapter 2, we showed that Rényi divergence aggregators are successful in aggregating probabilistic predictions even with individual biases, In particular, these findings are verified by Kaggle machine learning competition environments. Though there is some power in providing aggregated prediction mechanisms as part of competition environments, there is the additional question of the competition mechanism itself. The current competition mechanism has several weakness pointed out by Abernethy and Frongillo (2011).

- Anti-collaborativeness. Due to the strong incentive to win, competitors rarely share their models or techniques used. This is contrary to crowdsourcing projects,

such as Wikipedia, where participants must build upon the work of others. Although it is possible to implement aggregation by collecting all the competitors' prediction after the competition (i.e., what we have done in the thesis), there is ignorance of possibilities of collaboration between the competitors during the competition.

- Misaligned incentives. The winner-take-all prize structure easily leads to a situation that only a few teams are actually competing, and potential new teams never form since it is too difficult to catch up. This obvious discourages other teams' continuous contributions during the competitions.

- Precluding the use of proprietary methods. Most the machine learning competitions require the final winners to reveal their methods, and then the potential competitors trying to use non-open software or proprietary techniques will be unwilling to compete. By participating in the competition, a user has to effectively give away his intellectual property.

To overcome some of the issues above, Abernethy and Frongillo (2011) proposed a general framework of collaborative learning mechanism for crowdsourcing predictions. It is interesting to investigate the relationship between divergence-based (or other distance-based) aggregators with this collaborative learning mechanism, which potentially helps to simplify this mechanism and make it implementable in practice. Also, in various machine learning scenarios, such as regression, classification and clustering, further developments and implementations under this general framework are also worthy of considering to provide practical benefits.

A Unified Optimization Framework for Sep-CCSP

In Chapter 3, we developed efficient stochastic block coordinate descent methods for large-scale Sep-CCSP with strongly convex and general non-strongly convex functions, AdaSPDC and SP-BCD, respectively. Though derived from the same primal-dual framework, there exists some differences in the updates to accommodate the different properties of the functions involved. Fortunately, we found that in both of the two methods, a particular matrix (see Eq. (3.41) and (3.54)) has to be positive definite to guarantee algorithm convergence. It is hopeful that, using the positive definiteness constraint of a more unified matrix, we can construct a unified optimization framework such that it can be flexibly applied for both strongly convex and general non-strongly

convex functions. Additionally, the stepsizes configuration in both AdaSPDC and SP-BCD is not unique, it is crucial to investigate possible configurations to compare their theoretical and empirical performance in different applications.

Another potential direction is to investigate the exact benefits of parallelism in these methods. Further, if implemented in a distributed manner, the communication cost will become a non-negligible issue, where further exploration are need for communication efficient optimization schemes.

Further Explorations for Dynamics-based Sampling

Dynamical systems with stochastic gradient provides a general framework for scalable Bayesian sampling, where the smaller mini batch of data can be used for stochastic approximation. Here we discuss some of its potential extensions.

- The variance introduced by noisy gradient plays a crucial role in the performance of stochastic gradient methods. In our work, we tried to incorporate the estimated variance of the noisy gradient into the Langevin dynamics. However, a bad estimation of this variance will make the dynamics unstable, where smaller stepsizes have to be used. Therefore, an accurate estimation of the variance is always of need, which will not only be beneficial for large-scale Bayesian sampling, but also for stochastic optimization. Another way of handling the noisy gradients is to reduce the variance as much as possible, where the ideas from optimization community are worthy of considering, such as control variate technique (Wang et al., 2013) and semi-stochastic gradients (Johnson and Zhang, 2013; Konečnỳ and Richtárik, 2013).

- A naive nonsymmetric splitting method has been applied for simulating the Langevin dynamics for fair comparison with other methods. However, we note that optimal design of splitting methods in ergodic SDE systems has been explored recently in the mathematics community (Abdulle et al., 2015; Leimkuhler and Matthews, 2015; Leimkuhler et al., 2015). Moreover, it has been shown (Leimkuhler and Shang, 2015) that a certain type of symmetric splitting method for the Ad-Langevin/SGNHT method with a clean (full) gradient inherits the superconvergence property (i.e. fourth order convergence to the invariant distribution for configurational quantities) recently demonstrated in the setting of Langevin dynamics. It might be beneficial to apply other splitting methods for dynamics-

based sampling methods, and investigate their convergence performance (Chen and Carin, 2015; LeCun and Hinton, 2015).

- Langevin dynamics also provides a theoretical foundation for some stochastic optimization methods, heavily used for deep learning models (Bengio, 2009; LeCun and Hinton, 2015), such as SGD with momentum (Sutskever et al., 2013) and Adam (Kingma and Ba, 2015). It is promising to extend CCAdL as a stochastic optimizer to leverage the power of thermostat and variance estimation for fast and stable optimization.

5.3 Concluding Remarks

Generally, when models are large or data is intensive to the extent that we cannot handle, a natural way is to split them into parts and integrate the local information to achieve the big goal. The key point is to properly handle the *inconsistency* or *approximation error* introduced by the individual or local information. The three pieces of works done in this thesis are devoted to solving this issue and providing some benefits to distributed model aggregation, large-scale optimization and inference in several machine learning scenarios. We hope that these new frameworks and schemes will be valuable to future research in these fields.

Appendix A

Background on Fokker-Planck Equation for SDEs

In the literature of statistical mechanics and molecular dynamics, the Fokker-Planck equation is a partial differential equation that describes the time evolution of the probability density function of a particle $\mathbf{z}$ under the influence of a drag force $\mathbf{u}(\mathbf{z})$ and random diffusion, as in Brownian motion and second-order Langevin dynamics. For instance, consider the following SDE:

$$\mathrm{d}\mathbf{z} = \mathbf{u}(\mathbf{z})\mathrm{d}t + \sqrt{2\mathbf{Q}(\mathbf{z})}\mathrm{d}\mathbf{b}(t), \tag{A.1}$$

where $\mathbf{z} \in \mathbb{R}^M$, the drift term $\mathbf{u}(\mathbf{z}) \in \mathbb{R}^M$, $\mathbf{Q} \in \mathbb{R}^{M\times M}$, $\mathbf{b}$ denotes a D-dimensional Brownian motion, and $\mathrm{d}\mathbf{b}(t)$ colloquially represents a vector of infinitesimal Wiener increments. The probability distribution of $\mathbf{z}$ governed by Eq. (A.1) (we denote it as $\rho_t(\mathbf{z})$), evolves as time under the following equation, referred as Fokker-Planck equation,

$$\partial_t \rho_t(\mathbf{z}) = -\sum_{i=1}^{M} \partial_{z_i}\left[u_i(\mathbf{z})\rho_t(\mathbf{z})\right] + \sum_{i=1}^{M}\sum_{j=1}^{M} \partial_{z_i}\partial_{z_j}\left[Q_{ij}(\mathbf{z})\rho_t(\mathbf{z})\right]. \tag{A.2}$$

When the condition $\partial_t \rho_t(\mathbf{z}) = 0$ is satisfied, it indicates that the dynamics has stationary distribution as $\rho(\mathbf{z})$.

In Theorem 3 of Chapter 4, the proposed dynamics by CCAdL can be reformulated

as the form in standard SDE as Eq. (A.1),

$$\mathbf{z} = \begin{bmatrix} \boldsymbol{\theta} \\ \mathbf{p} \\ \xi \end{bmatrix}, \quad \mathbf{u}(\mathbf{z}) = \begin{bmatrix} \mathbf{M}^{-1}\mathbf{p} \\ -\nabla U(\boldsymbol{\theta}) - (h/2)\boldsymbol{\Sigma}(\boldsymbol{\theta})\mathbf{p} - \xi\mathbf{p} \\ \mu^{-1}\left(\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} - D\right) \end{bmatrix}$$
$$\mathbf{Q}(\mathbf{z}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A\mathbf{M} + (h/2)\Sigma(\boldsymbol{\theta})\mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{A.3}$$

Then we can derive the Fokker-Planck equation for CCAdL as shown in Theorem 3 of Chapter 4, we can verify that with the distribution $\partial_t \rho(\boldsymbol{\theta}, \mathbf{p}, \xi) = 0$, which indicates that it is indeed the stationary distribution of the dynamics.

Appendix B

Convergence Proofs for AdaSPDC

We restate the AdaSPDC algorithm procedure in Chapter 3.3 in the following.

AdaSPDC for Sep-CCSP problem (3.32)

Input: number of blocks picked in each iteration m and number of iterations T.

Initialize: $\mathbf{x}^0$, $\mathbf{y}^0$, $\overline{\mathbf{x}}^0 = \mathbf{x}^0$, $\mathbf{r}^0 = \frac{1}{N}\sum_{i=1}^{N} \mathbf{A}_i \mathbf{y}_i^0$

for $t = 0, 1, \ldots, T-1$ **do**

Randomly pick a subset with size m from all the N coordinate blocks, denoted as S_t.

According to the selected subset S_t, compute the adaptive parameter configuration of σ_i, τ^t and θ^t as following,

$$\sigma_i = \frac{1}{2R_i}\sqrt{\frac{N\lambda}{m\gamma}}, \tag{B.1}$$

$$\tau^t = \frac{1}{2R_{\max}^t}\sqrt{\frac{m\gamma}{N\lambda}}, \tag{B.2}$$

$$\theta^t = 1 - \frac{1}{n/m + R_{\max}^t\sqrt{(N/m)/(\lambda\gamma)}}, \tag{B.3}$$

where $R_i = \sqrt{\mu_{\max}\left(\mathbf{A}_i^T \mathbf{A}_i\right)}$, and $R_{\max}^t = \max\{R_i | i \in S_t\}$.

for each selected block in parallel **do**

Update the dual variable block,

$$\mathbf{y}_i^{t+1} = \operatorname{argmin}_{\mathbf{y}_i} \phi_i(\mathbf{y}_i) - \left\langle \overline{\mathbf{x}}^t, \mathbf{A}_i \mathbf{y}_i \right\rangle + \frac{1}{2\sigma_i}\|\mathbf{y}_i - \mathbf{y}_i^t\|_2^2, \text{ if } i \in S_t. \tag{B.4}$$

end for

Update primal variable,

$$\mathbf{x}^{t+1} = \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}) + \left\langle \mathbf{x}, \mathbf{r}^t + \frac{1}{m}\sum_{j \in S_t} \mathbf{A}_j(\mathbf{y}_j^{t+1} - \mathbf{y}_j^t) \right\rangle + \frac{1}{2\tau^t}\|\mathbf{x} - \mathbf{x}^t\|_2^2. \tag{B.5}$$

Extrapolate primal variable block,

$$\overline{\mathbf{x}}^{t+1} = \mathbf{x}^{t+1} + \theta^t(\mathbf{x}^{t+1} - \mathbf{x}^t), \tag{B.6}$$

Update the auxiliary variable $\mathbf{r}$,

$$\mathbf{r}^{t+1} = \mathbf{r}^t + \frac{1}{N}\sum_{j \in S_t} \mathbf{A}_j\left(\mathbf{y}_j^{t+1} - \mathbf{y}_j^t\right). \tag{B.7}$$

end for

Before presenting the proof of Theorem 1, we firstly provide the following lemma and its proof, which characterizes positive definiteness of an important matrix used in the proof of Theorem 1.

Lemma 1. *Given any matrix $\mathbf{K} \in \mathbb{R}^{d\times m}$, we partition the matrix $\mathbf{K}$ into J column blocks, $\mathbf{K}_j \in \mathbb{R}^{d\times m_j}$, $j = 1,\ldots,J$, and then $\sum_{j=1}^{J} m_j = m$. We then define two diagonal matrices, $\mathbf{U} = u\mathbf{I} \in \mathbb{R}^{d\times d}$, and $\mathbf{V} = diag\,(v_1\mathbf{I}_{m_1}, v_2\mathbf{I}_{m_2},\ldots,v_J\mathbf{I}_{m_J}) \in \mathbb{R}^{m\times m}$ and let $\mathbf{V}_j = v_j\mathbf{I}_{m_j}$. And denote $R_j = \|\mathbf{K}_j\|_2 = \sqrt{\lambda_{max}\left(\mathbf{K}_j^T\mathbf{K}_j\right)}$, where $\|\cdot\|_2$ is the spectral norm and $\lambda_{\max}(\cdot)$ is the maximum singular value of a matrix. And let $R_{\max} = \max\left\{R_j | j = 1,\ldots,J\right\}$. Now we consider the following parameter configuration, for any positive constant $c > 0$,*

$$v_j = \frac{c}{R_j}, \quad j = 1,\ldots,J \tag{B.8}$$

$$u = \frac{1}{cJR_{\max}}. \tag{B.9}$$

Under the above parameter configuration, the following matrix is positive semi-definite,

$$\mathbf{P} = \begin{bmatrix} \mathbf{U}^{-1} & -\mathbf{K} \\ -\mathbf{K}^T & \mathbf{V}^{-1} \end{bmatrix} \succeq 0. \tag{B.10}$$

Proof. Firstly consider each separable column block $\mathbf{K}_j$, then

$$\|\mathbf{U}^{\frac{1}{2}}\mathbf{K}_j\mathbf{V}_j^{\frac{1}{2}}\|_2^2 \leq \left(\|\mathbf{U}^{\frac{1}{2}}\|_2\|\mathbf{K}_j\|_2\|\mathbf{V}_j^{\frac{1}{2}}\|_2\right)^2 = \left(\frac{1}{\sqrt{cJR_{\max}}}R_j\sqrt{\frac{c}{R_j}}\right)^2 \leq \frac{1}{J}. \tag{B.11}$$

For any $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{y}_j \in \mathbb{R}^{m_j}$, we consider

$$-2\left\langle \mathbf{x}, \mathbf{K}_j\mathbf{y}_j\right\rangle = -2\left\langle \mathbf{U}^{-\frac{1}{2}}\mathbf{x}, \mathbf{U}^{\frac{1}{2}}\mathbf{K}_j\mathbf{V}_j^{\frac{1}{2}}\mathbf{V}_j^{-\frac{1}{2}}\mathbf{y}_j\right\rangle. \tag{B.12}$$

Applying the Cauchy-Schwarz inequality and the fact that $2ab \leq ha^2 + b^2/h$ for any a, b and $h > 0$, we obtain,

$$-2\left\langle \mathbf{x}, \mathbf{K}_j\mathbf{y}_j\right\rangle \geq -2\|\mathbf{U}^{-\frac{1}{2}}\mathbf{x}\|_2\|\mathbf{U}^{\frac{1}{2}}\mathbf{K}_j\mathbf{V}_j^{\frac{1}{2}}\mathbf{V}_j^{-\frac{1}{2}}\mathbf{y}_j\|_2 \tag{B.13}$$

$$\geq -\left(\frac{1}{h}\left\langle \mathbf{x}, \mathbf{U}^{-1}\mathbf{x}\right\rangle + h\|\mathbf{U}^{\frac{1}{2}}\mathbf{K}_j\mathbf{V}_j^{\frac{1}{2}}\|_2^2\left\langle \mathbf{y}_j, \mathbf{V}_j^{-1}\mathbf{y}_j\right\rangle\right) \tag{B.14}$$

In view of the inequality (B.11), it is obvious that there exists certain $\varepsilon > 0$ such that the following equality holds,

$$(J+\varepsilon)(1+\varepsilon)\|\mathbf{U}^{\frac{1}{2}}\mathbf{K}_j\mathbf{V}_j^{\frac{1}{2}}\|_2^2 = 1. \tag{B.15}$$

Thanks to this equality, now we set $h = J + \varepsilon$, and the inequality (B.14) can be further simplified,

$$-2\left\langle \mathbf{x}, \mathbf{K}_j \mathbf{y}_j \right\rangle \geq -\left(\frac{1}{J+\varepsilon} \left\langle \mathbf{x}, \mathbf{U}^{-1}\mathbf{x} \right\rangle + \frac{(J+\varepsilon)\|\mathbf{U}^{\frac{1}{2}}\mathbf{K}_j\mathbf{V}_j^{\frac{1}{2}}\|_2^2}{(J+\varepsilon)(1+\varepsilon)\|\mathbf{U}^{\frac{1}{2}}\mathbf{K}_j\mathbf{V}_j^{\frac{1}{2}}\|_2^2} \left\langle \mathbf{y}_j, \mathbf{V}_j^{-1}\mathbf{y}_j \right\rangle \right) \tag{B.16}$$

$$= -\left(\frac{1}{J+\varepsilon} \left\langle \mathbf{x}, \mathbf{U}^{-1}\mathbf{x} \right\rangle + \frac{1}{1+\varepsilon} \left\langle \mathbf{y}_j, \mathbf{V}_j^{-1}\mathbf{y}_j \right\rangle \right) \tag{B.17}$$

Let $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_J) \in \mathbb{R}^m$, and now we consider for any non-zero $(\mathbf{x}^T, \mathbf{y}^T)^T \in \mathbb{R}^{d+m}$, the following inner product can be expanded,

$$(\mathbf{x}^T, \mathbf{y}^T)\mathbf{P}(\mathbf{x}^T, \mathbf{y}^T)^T = \left\langle \mathbf{x}, \mathbf{U}^{-1}\mathbf{x} \right\rangle + \sum_{j=1}^{J} \left\langle \mathbf{y}_j, \mathbf{V}_j^{-1}\mathbf{y}_j \right\rangle - 2\sum_{j=1}^{J} \left\langle \mathbf{x}, \mathbf{K}_j\mathbf{y}_j \right\rangle. \tag{B.18}$$

Inserting the inequality (B.17) into the above equation, we obtain,

$$(\mathbf{x}^T, \mathbf{y}^T)\mathbf{P}(\mathbf{x}^T, \mathbf{y}^T)^T \geq \left\langle \mathbf{x}, \mathbf{U}^{-1}\mathbf{x} \right\rangle + \sum_{j=1}^{J} \left\langle \mathbf{y}_j, \mathbf{V}_j^{-1}\mathbf{y}_j \right\rangle \tag{B.19}$$

$$- \sum_{j=1}^{J} \left(\frac{1}{J+\varepsilon} \left\langle \mathbf{x}, \mathbf{U}^{-1}\mathbf{x} \right\rangle + \frac{1}{1+\varepsilon} \left\langle \mathbf{y}_j, \mathbf{V}_j^{-1}\mathbf{y}_j \right\rangle \right) \tag{B.20}$$

$$= \frac{\varepsilon}{J+\varepsilon} \left\langle \mathbf{x}, \mathbf{U}^{-1}\mathbf{x} \right\rangle + \frac{\varepsilon}{1+\varepsilon} \left\langle \mathbf{y}_j, \mathbf{V}_j^{-1}\mathbf{y}_j \right\rangle \geq 0, \tag{B.21}$$

which guarantees the positive semi-definiteness of the matrix $\mathbf{P}$. □

Now we are ready to prove the following theorem, which appears as Theorem 1 in Chapter 3.3:

Theorem 1 in Chapter 3.3: *Assume that each $\phi_i^*(\cdot)$ is γ-strongly convex, and $g(\cdot)$ is λ-strongly convex, and given the parameter configuration in Eq. (B.1), (B.2) and (B.3), then after T iterations in Algorithm 3.1, the algorithm achieves the following convergence performance*

$$\left(\frac{1}{2\tau^T} + \lambda \right) \mathbb{E}\left[\|\mathbf{x}^T - \mathbf{x}^\star\|_2^2 \right] + \mathbb{E}\left[\|\mathbf{y}^T - \mathbf{y}^\star\|_{\mathbf{v}}^2 \right]$$

$$\leq \left(\prod_{t=1}^{T} \theta^t \right) \left(\left(\frac{1}{2\tau^0} + \lambda \right) \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 + \|\mathbf{y}^0 - \mathbf{y}^\star\|_{\mathbf{v}'}^2 \right), \tag{B.22}$$

where $(\mathbf{x}^\star, \mathbf{y}^\star)$ is the optimal saddle point, $\nu_i = \frac{1/(4\sigma_i)+\gamma}{m}$, $\nu_i' = \frac{1/(2\sigma_i)+\gamma}{m}$, and $\|\mathbf{y}^T - \mathbf{y}^\star\|_{\mathbf{v}}^2 = \sum_{i=1}^{N} \nu_i \|\mathbf{y}_i^T - \mathbf{y}_i^\star\|_2^2$.

Proof. First, we analyze the value of the dual variable $\mathbf{y}$ after t-th update in Algorithm 1. For any $i \in \{1,2,\ldots,N\}$, let $\tilde{\mathbf{y}}_i$ be the value of $\mathbf{y}_i^{t+1}$ if $i \in S_t$, i.e.,

$$\tilde{\mathbf{y}}_i = \text{argmin}_{\mathbf{y}_i} \phi_i^*(\mathbf{y}_i) - \left\langle \overline{\mathbf{x}}^t, \mathbf{A}_i \mathbf{y}_i \right\rangle + \frac{1}{2\sigma_i} \|\mathbf{y}_i - \mathbf{y}_i^t\|_2^2 \tag{B.23}$$

Since $\phi^*(\cdot)$ is γ-strongly convex, thus the function to be minimized above is $(1/\sigma_i + \gamma)$-strongly convex. Then we have,

$$\begin{aligned}\phi_i^*(\mathbf{y}_i^\star) - \left\langle \overline{\mathbf{x}}^t, \mathbf{A}_i \mathbf{y}_i^\star \right\rangle + \frac{1}{2\sigma_i} \|\mathbf{y}_i^\star - \mathbf{y}_i^t\|_2^2 \geq \phi_i^*(\tilde{\mathbf{y}}_i) - \left\langle \overline{\mathbf{x}}^t, \mathbf{A}_i \tilde{\mathbf{y}}_i \right\rangle + \frac{1}{2\sigma_i} \|\tilde{\mathbf{y}}_i - \mathbf{y}_i^t\|_2^2 \\ + \left(\frac{1}{\sigma_i} + \gamma\right) \frac{\|\tilde{\mathbf{y}}_i - \mathbf{y}_i^\star\|_2^2}{2}\end{aligned} \tag{B.24}$$

Since the $(\mathbf{x}^\star, \mathbf{y}^\star)$ is the saddle point, we can obtain following inequality,

$$\phi_i^*(\tilde{\mathbf{y}}_i) - \langle \mathbf{x}^\star, \mathbf{A}_i \tilde{\mathbf{y}}_i \rangle \geq \phi_i^*(\mathbf{y}_i^\star) - \langle \mathbf{x}^\star, \mathbf{A}_i \mathbf{y}_i^\star \rangle \tag{B.25}$$

Adding the two inequalities together, we have

$$\frac{\|\mathbf{y}_i^t - \mathbf{y}_i^\star\|_2^2}{2\sigma_i} \geq \left(\frac{1}{2\sigma_i} + \frac{\gamma}{2}\right) \|\tilde{\mathbf{y}}_i - \mathbf{y}_i^\star\|_2^2 + \frac{1}{2\sigma_i} \|\tilde{\mathbf{y}}_i - \mathbf{y}_i^t\|_2^2 + \left\langle \mathbf{x}^\star - \overline{\mathbf{x}}^t, \mathbf{A}_i (\tilde{\mathbf{y}}_i - \mathbf{y}_i^\star) \right\rangle \tag{B.26}$$

In our algorithm, an index set S_t is randomly chosen. For every specific index i, the event $i \in S_t$ happens with probability m/N. If $i \in S_t$, then $\mathbf{y}_i^{t+1}$ is updated to the value $\tilde{\mathbf{y}}_i^t$. Otherwise, $\mathbf{y}_i^{t+1}$ is kept to be its old value $\mathbf{y}_i^t$. Let ξ_t be the random event that contains the set of all random variable before round t,

$$\xi_t = \{S_1, S_2, \ldots, S_t\}, \tag{B.27}$$

and then we have

$$\begin{aligned}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}_i^{t+1} - \mathbf{y}_i^\star\|_2^2\right] &= \frac{m}{N} \|\tilde{\mathbf{y}}_i - \mathbf{y}_i^\star\|_2^2 + \frac{N-m}{N} \|\mathbf{y}_i^t - \mathbf{y}_i^\star\|_2^2 \\ \mathbb{E}_{\xi_t}\left[\|\mathbf{y}_i^{t+1} - \mathbf{y}_i^t\|_2^2\right] &= \frac{m}{N} \|\tilde{\mathbf{y}}_i - \mathbf{y}_i^t\|_2^2 \\ \mathbb{E}_{\xi_t}\left[\mathbf{y}_i^{t+1}\right] &= \frac{m}{N} \tilde{\mathbf{y}}_i + \frac{N-m}{N} \mathbf{y}_i^t,\end{aligned}$$

where $\mathbb{E}_{\xi_t}[\cdot]$ denotes the conditional expectation $\mathbb{E}[\cdot|\xi_t]$ for simplicity.

As a consequence, we can insert the representations of $\|\tilde{\mathbf{y}}_i - \mathbf{y}_i^\star\|_2^2$, $\|\tilde{\mathbf{y}}_i - \mathbf{y}_i^t\|_2^2$ and $\tilde{\mathbf{y}}_i$ in terms of the above expectations into the inequality (B.26),

$$\begin{aligned}\left(\frac{N}{2m\sigma_i} + \frac{N-m}{2m}\gamma\right) \|\mathbf{y}_i^t - \mathbf{y}_i^\star\|_2^2 \geq \left(\frac{N}{2m\sigma_i} + \frac{N}{2m}\gamma\right) \mathbb{E}_{\xi_t}\left[\|\mathbf{y}_i^{t+1} - \mathbf{y}_i^\star\|_2^2\right] \\ + \frac{N}{2m\sigma_i} \mathbb{E}_{\xi_t}\left[\|\mathbf{y}_i^{t+1} - \mathbf{y}_i^t\|_2^2\right] \\ + \left\langle \mathbf{x}^\star - \overline{\mathbf{x}}^t, \mathbf{A}_i \left(\mathbf{y}_i^t - \mathbf{y}_i^\star + \frac{N}{m} \mathbb{E}_{\xi_t}\left[\mathbf{y}_i^{t+1} - \mathbf{y}_i^t\right]\right) \right\rangle\end{aligned} \tag{B.28}$$

Then we add the above inequality from $i = 1, 2, \ldots, N$, and divide both sides by N, and obtain

$$\|\mathbf{y}^t - \mathbf{y}^\star\|_{\boldsymbol{\mu}}^2 \geq \mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1} - \mathbf{y}^\star\|_{\boldsymbol{\mu}'}^2\right] + \frac{1}{2m}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1} - \mathbf{y}^t\|_{\boldsymbol{\sigma}}^2\right] + \mathbb{E}_{\xi_t}\left[\left\langle \mathbf{x}^\star - \overline{\mathbf{x}}^t, \mathbf{u}^t - \mathbf{u}^\star + \frac{1}{m}\sum_{j\in S_t}\mathbf{A}_j\left(\mathbf{y}_j^{t+1} - \mathbf{y}_j^t\right)\right\rangle\right], \tag{B.29}$$

where $\mu_i = \frac{1}{2m\sigma_i} + \frac{N-m}{2mN}\gamma$, $\mu_i' = \frac{1}{2m\sigma_i} + \frac{\gamma}{2m}$, $\mathbf{u}^\star = \frac{1}{N}\sum_{i=1}^N \mathbf{A}_i\mathbf{y}_i^\star$, and $\mathbf{u}^t = \frac{1}{N}\sum_{i=1}^N \mathbf{A}_i\mathbf{y}_i^t$. In the crossing term between primal and dual variable, we use the fact that $\sum_{i=1}^N \mathbf{A}_i(\mathbf{y}_i^{t+1} - \mathbf{y}_i^t) = \sum_{j\in S_t}\mathbf{A}_j(\mathbf{y}_j^{t+1} - \mathbf{y}_j^t)$ since only the blocks in index set S_t are chosen and updated in t-th update.

Now we characterize the t-th update of primal variable $\mathbf{x}$. Following the same derivation for dual variable and using the assumption that $g(\cdot)$ is λ-strongly convex, we can easily obtain

$$\frac{1}{2\tau^t}\|\mathbf{x}^t - \mathbf{x}^\star\|_2^2 \geq \left(\frac{1}{2\tau^t} + \lambda\right)\|\mathbf{x}^{t+1} - \mathbf{x}^\star\|_2^2 + \frac{1}{2\tau^t}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|_2^2 + \left\langle \mathbf{x}^{t+1} - \mathbf{x}^t, \mathbf{u}^t - \mathbf{u}^\star + \frac{1}{m}\sum_{j\in S_t}\mathbf{A}_j\left(\mathbf{y}_j^{t+1} - \mathbf{y}_j^t\right)\right\rangle. \tag{B.30}$$

Taking expectation over both sides of the above inequality and adding it to the the inequality (B.29), then we have

$$\begin{aligned}\frac{1}{2\tau^t}\|\mathbf{x}^t - \mathbf{x}^\star\|_2^2 + \|\mathbf{y}^t - \mathbf{y}^\star\|_{\boldsymbol{\mu}}^2 \geq{}& \left(\frac{1}{2\tau^t} + \lambda\right)\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1} - \mathbf{x}^\star\|_2^2\right] + \mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1} - \mathbf{y}^\star\|_{\boldsymbol{\mu}'}^2\right] \\ &+ \frac{1}{2\tau^t}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1} - \mathbf{x}^t\|_2^2\right] + \frac{1}{2m}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1} - \mathbf{y}^t\|_{\boldsymbol{\sigma}}^2\right] \\ &+ \mathbb{E}_{\xi_t}\left[\left\langle \mathbf{x}^{t+1} - \mathbf{x}^t - \theta^t\left(\mathbf{x}^t - \mathbf{x}^{t-1}\right), \mathbf{A}\left(\frac{1}{N}(\mathbf{y}^t - \mathbf{y}^\star) + \frac{1}{m}(\mathbf{y}^{t+1} - \mathbf{y}^t)\right)\right\rangle\right],\end{aligned} \tag{B.31}$$

where the matrix $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n]$.

Now we focus on the most crucial part of the proof: bounding the last term of R.H.S. of the above inequality (B.31). Firstly we rearrange this crossing term as follows,

$$\begin{aligned}&\left\langle \mathbf{x}^{t+1} - \mathbf{x}^t - \theta^t\left(\mathbf{x}^t - \mathbf{x}^{t-1}\right), \mathbf{A}\left(\frac{1}{N}(\mathbf{y}^t - \mathbf{y}^\star) + \frac{1}{m}(\mathbf{y}^{t+1} - \mathbf{y}^t)\right)\right\rangle \\ &= \frac{1}{N}\left\langle \mathbf{x}^{t+1} - \mathbf{x}^t, \mathbf{A}\left(\mathbf{y}^{t+1} - \mathbf{y}^t\right)\right\rangle - \frac{\theta^t}{N}\left\langle \mathbf{x}^t - \mathbf{x}^{t-1}, \mathbf{A}\left(\mathbf{y}^t - \mathbf{y}^\star\right)\right\rangle \\ &+ \frac{N-m}{mN}\left\langle \mathbf{x}^{t+1} - \mathbf{x}^t, \mathbf{A}\left(\mathbf{y}^{t+1} - \mathbf{y}^t\right)\right\rangle - \frac{\theta^t}{m}\left\langle \mathbf{x}^t - \mathbf{x}^{t-1}, \mathbf{A}\left(\mathbf{y}^{t+1} - \mathbf{y}^t\right)\right\rangle.\end{aligned} \tag{B.32}$$

Given the parameter configuration in Eq (B.2) and (B.1), we consider the following symmetric matrix,

$$\mathbf{P} = \begin{bmatrix} \frac{m}{2\tau^t}\mathbf{I} & -\mathbf{A}_{S_t} \\ -\mathbf{A}_{S_t}^T & \frac{1}{2\text{diag}(\boldsymbol{\sigma}_{S_t})} \end{bmatrix} \tag{B.33}$$

Applying the Lemma 1, we can guarantee the positive semi-definiteness of the matrix $\mathbf{P}$, which naturally leads the following inequality,

$$\frac{m}{4\tau^t}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|_2^2 + \sum_{i\in S_t}\frac{1}{4\sigma_i}\|\mathbf{y}_i^{t+1}-\mathbf{y}_i^t\|_2^2 \geq \left\langle \mathbf{x}^{t+1}-\mathbf{x}^t, \sum_{i\in S_t}\mathbf{A}_i\left(\mathbf{y}_i^{t+1}-\mathbf{y}_i^t\right)\right\rangle \tag{B.34}$$

Similarly, we can also obtain

$$\frac{m}{4\tau^t}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|_2^2 + \sum_{i\in S_t}\frac{1}{4\sigma_i}\|\mathbf{y}_i^{t+1}-\mathbf{y}_i^t\|_2^2 \geq -\left\langle \mathbf{x}^{t+1}-\mathbf{x}^t, \sum_{i\in S_t}\mathbf{A}_i\left(\mathbf{y}_i^{t+1}-\mathbf{y}_i^t\right)\right\rangle \tag{B.35}$$

Taking the expectation for both sides of the above two equalities and using the facts that

$$\begin{aligned}
&\mathbb{E}_{\xi_t}\left[\frac{m}{4\tau^t}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|_2^2 + \|\mathbf{y}^{t+1}-\mathbf{y}^t\|^2_{\frac{1}{4\text{diag}(\boldsymbol{\sigma})}}\right] \\
&= \mathbb{E}_{\xi_t}\left[\frac{m}{4\tau^t}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|_2^2 + \sum_{i\in S_t}\frac{1}{4\sigma_i}\|\mathbf{y}_i^{t+1}-\mathbf{y}_i^t\|_2^2\right], \\
&\mathbb{E}_{\xi_t}\left[\left|\left\langle \mathbf{x}^{t+1}-\mathbf{x}^t, \mathbf{A}\left(\mathbf{y}^{t+1}-\mathbf{y}^t\right)\right\rangle\right|\right] \\
&= \mathbb{E}_{\xi_t}\left[\left|\left\langle \mathbf{x}^{t+1}-\mathbf{x}^t, \sum_{i\in S_t}\mathbf{A}_i\left(\mathbf{y}_i^{t+1}-\mathbf{y}_i^t\right)\right\rangle\right|\right],
\end{aligned}$$

we have that

$$\mathbb{E}_{\xi_t}\left[\left|\left\langle \mathbf{x}^{t+1}-\mathbf{x}^t, \mathbf{A}\left(\mathbf{y}^{t+1}-\mathbf{y}^t\right)\right\rangle\right|\right] \leq \mathbb{E}_{\xi_t}\left[\frac{m}{4\tau^t}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|_2^2 + \|\mathbf{y}^{t+1}-\mathbf{y}^t\|^2_{\frac{1}{4\text{diag}(\boldsymbol{\sigma})}}\right] \tag{B.36}$$

Similarly, we can obtain

$$\mathbb{E}_{\xi_t}\left[\left|\left\langle \mathbf{x}^{t}-\mathbf{x}^{t-1}, \mathbf{A}\left(\mathbf{y}^{t+1}-\mathbf{y}^t\right)\right\rangle\right|\right] \leq \mathbb{E}_{\xi_t}\left[\frac{m}{4\tau^t}\|\mathbf{x}^{t}-\mathbf{x}^{t-1}\|_2^2 + \|\mathbf{y}^{t+1}-\mathbf{y}^t\|^2_{\frac{1}{4\text{diag}(\boldsymbol{\sigma})}}\right] \tag{B.37}$$

Therefore,

$$\begin{aligned}
\mathbb{E}_{\xi_t}\left[\left\langle \mathbf{x}^{t+1}-\mathbf{x}^t, \mathbf{A}\left(\mathbf{y}^{t+1}-\mathbf{y}^t\right)\right\rangle\right] \geq &-\mathbb{E}_{\xi_t}\left[\frac{m}{4\tau^t}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|_2^2\right] \\
&-\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}^t\|^2_{\frac{1}{4\text{diag}(\boldsymbol{\sigma})}}\right] \qquad \text{(B.38)} \\
\mathbb{E}_{\xi_t}\left[\left\langle \mathbf{x}^{t}-\mathbf{x}^{t-1}, \mathbf{A}\left(\mathbf{y}^{t+1}-\mathbf{y}^t\right)\right\rangle\right] \geq &-\mathbb{E}_{\xi_t}\left[\frac{m}{4\tau^t}\|\mathbf{x}^{t}-\mathbf{x}^{t-1}\|_2^2\right] \\
&-\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}^t\|^2_{\frac{1}{4\text{diag}(\boldsymbol{\sigma})}}\right] \qquad \text{(B.39)}
\end{aligned}$$

Now we insert the Eq. (B.32) into the inequality (B.31), and then apply the two bounds (B.38) and (B.39), we have

$$\begin{aligned}\frac{1}{2\tau^t}\|\mathbf{x}^t-\mathbf{x}^\star\|_2^2+\|\mathbf{y}^t-\mathbf{y}^\star\|_{\boldsymbol{\mu}}^2 \geq \left(\frac{1}{2\tau^t}+\lambda\right)\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1}-\mathbf{x}^\star\|_2^2\right]+\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}^\star\|_{\boldsymbol{\mu}'}^2\right]\\ +\frac{1}{4\tau^t}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1}-\mathbf{x}^t\|_2^2\right]+\frac{1}{N}\mathbb{E}_{\xi_t}\left[\left\langle\mathbf{x}^{t+1}-\mathbf{x}^t,\mathbf{A}\left(\mathbf{x}^{t+1}-\mathbf{x}^t\right)\right\rangle\right]\\ -\frac{\theta^t}{4\tau^t}\|\mathbf{x}^t-\mathbf{x}^{t-1}\|_2^2-\frac{\theta^t}{N}\left\langle\mathbf{x}^t-\mathbf{x}^{t-1},\mathbf{A}\left(\mathbf{y}^t-\mathbf{y}^\star\right)\right\rangle+\frac{1-\theta^t+m/N}{4m}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}^t\|_{\boldsymbol{\sigma}}^2\right].\end{aligned}$$

Recall the configuration for θ^t in Eq. (B.3), the last term of R.H.S. of the above inequality is non-negative, and can be bounded away. Then we have the following,

$$\begin{aligned}\frac{1}{2\tau^t}\|\mathbf{x}^t-\mathbf{x}^\star\|_2^2+\|\mathbf{y}^t-\mathbf{y}^\star\|_{\boldsymbol{\mu}}^2+\frac{\theta^t}{4\tau^t}\|\mathbf{x}^t+\mathbf{x}^{t-1}\|_2^2+\frac{\theta^t}{N}\left\langle\mathbf{x}^t-\mathbf{x}^{t-1},\mathbf{A}\left(\mathbf{y}^t-\mathbf{y}^\star\right)\right\rangle\geq\\ \left(\frac{1}{2\tau^t}+\lambda\right)\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1}-\mathbf{x}^\star\|_2^2\right]+\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}^\star\|_{\boldsymbol{\mu}'}^2\right]+\frac{1}{4\tau^t}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1}-\mathbf{x}^t\|_2^2\right]\\ +\frac{1}{N}\mathbb{E}_{\xi_t}\left[\left\langle\mathbf{x}^{t+1}-\mathbf{x}^t,\mathbf{A}\left(\mathbf{x}^{t+1}-\mathbf{x}^t\right)\right\rangle\right]\triangleq\mathbb{E}_{\xi_t}\left[\Delta^{t+1}\right]\end{aligned}\tag{B.40}$$

According to the defined sequence Δ^{t+1}, we have

$$\begin{aligned}\theta^t\Delta^t=&\theta^t\left(\frac{1}{2\tau^t}+\lambda\right)\|\mathbf{x}^t-\mathbf{x}^\star\|_2^2+\theta^t\|\mathbf{y}^t-\mathbf{y}^\star\|_{\boldsymbol{\mu}'}^2\\ &+\frac{\theta^t}{4\tau^t}\|\mathbf{x}^t-\mathbf{x}^{t-1}\|_2^2+\frac{\theta^t}{N}\left\langle\mathbf{x}^t-\mathbf{x}^{t-1},\mathbf{A}\left(\mathbf{y}^t-\mathbf{y}^\star\right)\right\rangle\end{aligned}\tag{B.41}$$

According to the parameter configuration for τ^t, σ_i and θ^t, we can easily verify that

$$\begin{aligned}\theta^t\left(\frac{1}{2\tau^t}+\lambda\right)&\geq\frac{1}{2\tau^t}\\ \theta^t\mu_i'&\geq\mu_i\end{aligned}$$

Combining these two inequalities with the inequality (B.40) and Eq. (B.41), we have

$$\mathbb{E}_{\xi_t}\left[\Delta^{t+1}\right]\leq\theta^t\Delta^t.\tag{B.42}$$

Apply this relation recursively and taking expectation with respect to all random variables up to time T, we have

$$\begin{aligned}\mathbb{E}\left[\left(\frac{1}{2\tau^T}+\lambda\right)\|\mathbf{x}^T-\mathbf{x}^\star\|_2^2\right]+\mathbb{E}\left[\|\mathbf{y}^T-\mathbf{y}^\star\|_{\boldsymbol{\mu}'}^2\right]+\mathbb{E}\left[\frac{1}{4\tau^T}\|\mathbf{x}^T-\mathbf{x}^{T-1}\|_2^2\right]\\ +\frac{1}{N}\mathbb{E}\left[\left\langle\mathbf{x}^T-\mathbf{x}^{T-1},\mathbf{A}\left(\mathbf{y}^T-\mathbf{y}^\star\right)\right\rangle\right]\leq\left(\prod_{t=1}^T\mathbb{E}\left[\theta^t\right]\right)\Delta^0,\end{aligned}\tag{B.43}$$

where

$$\Delta^0 = \left(\frac{1}{2\tau^0} + \lambda\right) \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 + \|\mathbf{y}^0 - \mathbf{y}^\star\|_{\boldsymbol{\mu}'}^2. \tag{B.44}$$

Consider the following matrix

$$\mathbf{Q} = \begin{bmatrix} \frac{N}{2\tau^T}\mathbf{I} & \pm\mathbf{A}_{S_T} \\ \pm\mathbf{A}_{S_T}^T & \frac{n}{2m\text{diag}(\boldsymbol{\sigma}_{S_t})} \end{bmatrix} \tag{B.45}$$

Applying the Lemma 1 again, we can guarantee the positive definiteness of the matrix, which implies that

$$\pm\frac{1}{N}\left\langle \mathbf{x}^T - \mathbf{x}^{T-1}, \sum_{i\in S_T} \mathbf{A}_i\left(\mathbf{y}_i^T - \mathbf{y}_i^\star\right)\right\rangle \leq \frac{1}{4\tau^T}\|\mathbf{x}^T - \mathbf{x}^{T-1}\|_2^2 + \frac{1}{4m}\sum_{i\in S_T}\frac{1}{\sigma_i}\|\mathbf{y}_i^T - \mathbf{y}_i^\star\|_2^2 \tag{B.46}$$

Taking expectation,

$$\begin{aligned}\frac{1}{N}\mathbb{E}\left[\left|\left\langle \mathbf{x}^T - \mathbf{x}^{T-1}, \mathbf{A}\left(\mathbf{y}^T - \mathbf{y}^\star\right)\right\rangle\right|\right] \leq & \mathbb{E}\left[\frac{1}{4\tau^T}\|\mathbf{x}^T - \mathbf{x}^{T-1}\|_2^2\right] \\ & + \frac{1}{4m}\mathbb{E}\left[\|\mathbf{y}^T - \mathbf{y}^\star\|_{1/\text{diag}(\boldsymbol{\sigma})}\right]\end{aligned} \tag{B.47}$$

Thus,

$$\begin{aligned}\frac{1}{N}\mathbb{E}\left[\left\langle \mathbf{x}^T - \mathbf{x}^{T-1}, \mathbf{A}\left(\mathbf{y}^T - \mathbf{y}^\star\right)\right\rangle\right] \geq & -\mathbb{E}\left[\frac{1}{4\tau^T}\|\mathbf{x}^T - \mathbf{x}^{T-1}\|_2^2\right] \\ & - \frac{1}{4m}\mathbb{E}\left[\|\mathbf{y}^T - \mathbf{y}^\star\|_{1/\text{diag}(\boldsymbol{\sigma})}\right]\end{aligned} \tag{B.48}$$

Then combining the above inequality with inequality (B.43), we have

$$\begin{aligned}&\mathbb{E}\left[\left(\frac{1}{2\tau^T} + \lambda\right)\|\mathbf{x}^T - \mathbf{x}^\star\|_2^2\right] + \mathbb{E}\left[\|\mathbf{y}^T - \mathbf{y}^\star\|_{\mathbf{v}}^2\right] \\ &\leq \left(\prod_{t=1}^T \mathbb{E}\left[\theta^t\right]\right)\left(\left(\frac{1}{2\tau^0} + \lambda\right)\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 + \|\mathbf{y}^0 - \mathbf{y}^\star\|_{\mathbf{v}'}^2\right),\end{aligned} \tag{B.49}$$

where $\nu_i = \frac{1}{(4m\sigma_i)} + \frac{\gamma}{2m}$, $\nu_i' = \mu_i' = \frac{1}{(2m\sigma_i)} + \frac{\gamma}{2m}$, and $\|\mathbf{y}^T - \mathbf{y}^\star\|_{\mathbf{v}}^2 = \sum_{i=1}^N \nu_i\|\mathbf{y}_i^T - \mathbf{y}_i^\star\|_2^2$, which completes the proof. □

Appendix C

Convergence Proofs for SP-BCD

Now restate the SP-BCD in Algorithm 3.2 in the following for convenience.

SP-BCD for Sep-CCSP problem (3.45)

Input: number of blocks picked in each iteration m.

Initialize: $\mathbf{x}^0$, $\mathbf{y}^0$, $\overline{\mathbf{x}}^0 = \mathbf{x}^0$, $\overline{\mathbf{r}}^0 = \sum_{j=1}^{q} \mathbf{A}_j \overline{\mathbf{x}}_j^0$

for $t = 1, 2, \ldots, T$ **do**

Randomly pick a subset with size m from all the q coordinate blocks, denoted as S_t.

Compute the configuration of parameter θ, $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}^t$:

$$\theta = \frac{m}{q} \tag{C.1}$$

$$\tau_d = \frac{1}{\sum_{j=1}^{D_{\mathbf{y}}} |A_{jd}|}, \quad d = 1, 2, \ldots, D_{\mathbf{x}} \tag{C.2}$$

$$\sigma_k^t = \frac{m}{q} \cdot \frac{1}{\sum_{j \in S_t} |A_{kj}|}, \quad k = 1, 2, \ldots, D_{\mathbf{y}}. \tag{C.3}$$

for each block in parallel **do**

Update each selected primal variable block using

$$\mathbf{x}_j^{t+1} = \text{argmin}_{\mathbf{x}_j} f_j(\mathbf{x}_j) + \left\langle \mathbf{y}^t, \mathbf{A}_j \mathbf{x}_j \right\rangle + \frac{1}{2} \|\mathbf{x}_j - \mathbf{x}_j^t\|_{1/\boldsymbol{\tau}_j}^2, \tag{C.4}$$

Extrapolate primal variable block using

$$\overline{\mathbf{x}}_j^{t+1} = \begin{cases} \mathbf{x}_j^{t+1} + \theta \left(\mathbf{x}_j^{t+1} - \mathbf{x}_j^t \right) & \text{if } j \in S_t \\ \overline{\mathbf{x}}_j^t & \text{otherwise,} \end{cases} \tag{C.5}$$

end for

Update dual variable using

$$\mathbf{y}^{t+1} = \text{argmin}_{\mathbf{y}} g^*(\mathbf{y}) - \left\langle \mathbf{y}, \overline{\mathbf{r}}^t + \frac{q}{m} \sum_{j \in S_t} \mathbf{A}_j (\overline{\mathbf{x}}_j^{t+1} - \overline{\mathbf{x}}_j^t) \right\rangle + \frac{1}{2} \|\mathbf{y} - \mathbf{y}^t\|_{1/\boldsymbol{\sigma}^t}^2, \tag{C.6}$$

Update $\overline{\mathbf{r}}^{t+1}$ using

$$\overline{\mathbf{r}}^{t+1} = \overline{\mathbf{r}}^t + \sum_{j \in S_t} \mathbf{A}_j \left(\overline{\mathbf{x}}_j^{t+1} - \overline{\mathbf{x}}_j^t \right). \tag{C.7}$$

end for

Recall Theorem 2 in Section 3.4 as follows.

Theorem 2 in Section 3.4: *Given that all $f_j(\mathbf{x}_j)$ and $g^*(\mathbf{y})$ are convex functions, and we set $\theta = m/q$, proximal parameters for primal and dual update as Eq.(C.2) and (C.3), respectively. Then for any saddle point $(\mathbf{x},\mathbf{y})$, the expected gap decays as the following rate:*

$$\mathbb{E}\left[L\left(\sum_{t=1}^{T}\mathbf{x}^t/T,\mathbf{y}\right)-L\left(\mathbf{x},\sum_{t=1}^{T}\mathbf{y}^t/T\right)\right]\leq\frac{1}{T}M(0),$$

where

$$\begin{aligned}M(0)=\frac{q}{m}\cdot\frac{1}{2}\|\mathbf{x}^0-\mathbf{x}\|^2_{1/\boldsymbol{\tau}}+\frac{1}{2}\|\mathbf{y}^0-\mathbf{y}\|^2_{1/\boldsymbol{\sigma}^0}-\left\langle\mathbf{y}^0-\mathbf{y},\mathbf{A}\left(\mathbf{x}^0-\mathbf{x}\right)\right\rangle\\+\frac{q-m}{m}\left(f(\mathbf{x}^0)+\left\langle\mathbf{y},\mathbf{A}\mathbf{x}^0\right\rangle-\left(f(\mathbf{x})+\langle\mathbf{y},\mathbf{A}\mathbf{x}\rangle\right)\right).\end{aligned}$$

Proof. First, we analyze the primal and dual variables $\mathbf{x}$ and $\mathbf{y}$ after t-th update in the Algorithm 3.2. We introduce a temporary variable $\tilde{\mathbf{x}}_j$ to be the value of $\mathbf{x}_j^{t+1}$ if $j\in S_t$, for any $j\in\{1,2,\ldots,q\}$, i.e. (crossref Eq.(C.4)),

$$\mathbf{x}_j^{t+1}=\mathrm{argmin}_{\mathbf{x}_j}f_j(\mathbf{x}_j)+\left\langle\mathbf{y}^t,\mathbf{A}_j\mathbf{x}_j\right\rangle+\frac{1}{2}\|\mathbf{x}_j-\mathbf{x}_j^t\|^2_{1/\boldsymbol{\tau}_j},$$

Due to the strong convexity of the added proximal term, the function minimized above is $1/\boldsymbol{\tau}_j$-strongly convex, and then for any $\mathbf{x}_j$ we have

$$f_j(\mathbf{x}_j)+\left\langle\mathbf{y}^t,\mathbf{A}_j\mathbf{x}_j\right\rangle+\frac{1}{2}\|\mathbf{x}_j-\mathbf{x}_j^t\|^2_{1/\boldsymbol{\tau}_j}\geq f_j(\tilde{\mathbf{x}}_j)+\left\langle\mathbf{y}^t,\mathbf{A}_j\tilde{\mathbf{x}}_j\right\rangle+\frac{1}{2}\|\tilde{\mathbf{x}}_j-\mathbf{x}_j^t\|^2_{1/\boldsymbol{\tau}_j}+\frac{1}{2}\|\tilde{\mathbf{x}}_j-\mathbf{x}_j\|^2_{1/\boldsymbol{\tau}_j}.\tag{C.8}$$

In our algorithm, an index set S_t is randomly chosen. For every specific index j, the event $j\in S_t$ happens with probability m/q. If $j\in S_t$, then $\mathbf{x}_j^{t+1}$ is updated to the value $\tilde{\mathbf{x}}_j^t$. Otherwise, $\mathbf{x}_j^{t+1}$ is kept to be its old value $\mathbf{x}_j^t$. Let ξ_t be the random event that contains the set of all random variable before round t,

$$\xi_t=\{S_1,S_2,\ldots,S_t\},\tag{C.9}$$

and then we have

$$\begin{aligned}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}_j^{t+1}-\mathbf{x}_j\|^2_{1/\boldsymbol{\tau}_j}\right]&=\frac{m}{q}\|\tilde{\mathbf{x}}_j-\mathbf{x}_j\|^2_{1/\boldsymbol{\tau}_j}+\frac{q-m}{q}\|\mathbf{x}_j^t-\mathbf{x}_j\|^2_{1/\boldsymbol{\tau}_j}\\\mathbb{E}_{\xi_t}\left[\|\mathbf{x}_j^{t+1}-\mathbf{x}_j^t\|^2_{1/\boldsymbol{\tau}_j}\right]&=\frac{m}{q}\|\tilde{\mathbf{x}}_j-\mathbf{x}_j^t\|^2_{1/\boldsymbol{\tau}_j}\\\mathbb{E}_{\xi_t}\left[\mathbf{x}_j^{t+1}\right]&=\frac{m}{q}\tilde{\mathbf{x}}_j+\frac{q-m}{q}\mathbf{x}_j^t\\\mathbb{E}_{\xi_t}\left[f_j(\mathbf{x}_j^{t+1})\right]&=\frac{m}{q}f_j(\tilde{\mathbf{x}}_j)+\frac{q-m}{q}f_j(\mathbf{x}_j^t)\end{aligned}$$

where $\mathbb{E}_{\xi_t}[\cdot]$ denotes the conditional expectation $\mathbb{E}[\cdot|\xi_t]$ for simplicity.

With these equality relationships, we can substitute $f_j\left(\tilde{\mathbf{x}}_j\right)$, $\tilde{\mathbf{x}}_j$, $\|\tilde{\mathbf{x}}_j - \mathbf{x}_j^t\|^2_{1/\boldsymbol{\tau}_j}$ and $\|\tilde{\mathbf{x}}_j - \mathbf{x}_j\|^2_{1/\boldsymbol{\tau}_j}$ into the inequality (C.8),

$$\begin{aligned}
&\mathbb{E}_{\xi_t}\left[f_j(\mathbf{x}_j^{t+1})\right] - f_j(\mathbf{x}_j) \\
&\leq \left(\frac{q}{m}\cdot\frac{1}{2}\|\mathbf{x}_j^t - \mathbf{x}_j\|^2_{1/\boldsymbol{\tau}_j} + \frac{q-m}{m}f_j(\mathbf{x}_j^t)\right) - \left(\frac{q}{m}\cdot\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}_j^{t+1} - \mathbf{x}_j\|^2_{1/\boldsymbol{\tau}_j}\right] + \frac{q-m}{m}f_j(\mathbf{x}_j^{t+1})\right) \\
&- \frac{q}{m}\cdot\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}_j^{t+1} - \mathbf{x}_j^t\|^2_{1/\boldsymbol{\tau}_j}\right] - \mathbb{E}_{\xi_t}\left[\left\langle \mathbf{y}^t, \mathbf{A}_j\left(\frac{q}{m}\mathbf{x}_j^{t+1} - \frac{q-m}{m}\mathbf{x}_j^t - \frac{m}{q}\mathbf{x}_j\right)\right\rangle\right].
\end{aligned}$$

Then summing the above inequality with all the indices $i = 1, \ldots, q$, we can obtain

$$\begin{aligned}
&\mathbb{E}_{\xi_t}\left[f(\mathbf{x}^{t+1})\right] - f(\mathbf{x}) \\
&\leq \left(\frac{q}{m}\cdot\frac{1}{2}\|\mathbf{x}^t - \mathbf{x}\|^2_{1/\boldsymbol{\tau}} + \frac{q-m}{m}f(\mathbf{x}^t)\right) - \left(\frac{q}{m}\cdot\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1} - \mathbf{x}\|^2_{1/\boldsymbol{\tau}}\right] + \frac{q-m}{m}f(\mathbf{x}^{t+1})\right) \\
&- \frac{q}{m}\cdot\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2_{1/\boldsymbol{\tau}}\right] - \mathbb{E}_{\xi_t}\left[\left\langle \mathbf{y}^t, \mathbf{A}\left(\frac{q}{m}\mathbf{x}^{t+1} - \frac{q-m}{m}\mathbf{x}^t - \frac{m}{q}\mathbf{x}\right)\right\rangle\right]. \qquad \text{(C.10)}
\end{aligned}$$

Now, we consider dual update in Eq. (C.6),

$$\begin{aligned}
&g^*(\mathbf{y}) - \left\langle \mathbf{y}, \overline{\mathbf{r}}^t + \frac{q}{m}\sum_{j\in S_t}\mathbf{A}_j\left(\overline{\mathbf{x}}_j^{t+1} - \overline{\mathbf{x}}_j^t\right)\right\rangle + \frac{1}{2}\|\mathbf{y} - \mathbf{y}^t\|^2_{\boldsymbol{\sigma}^t} \\
&\geq g^*(\mathbf{y}^{t+1}) - \left\langle \mathbf{y}^{t+1}, \overline{\mathbf{r}}^t + \frac{q}{m}\sum_{j\in S_t}\mathbf{A}_j\left(\overline{\mathbf{x}}_j^{t+1} - \overline{\mathbf{x}}_j^t\right)\right\rangle \quad + \frac{1}{2}\|\mathbf{y}^{t+1} - \mathbf{y}^t\|^2_{\boldsymbol{\sigma}^t} + \frac{1}{2}\|\mathbf{y}^{t+1} - \mathbf{y}\|^2_{\boldsymbol{\sigma}^t}
\end{aligned} \qquad \text{(C.11)}$$

Since in each iteration, we always keep $\overline{\mathbf{r}}^t = \sum_{i=1}^q \mathbf{A}_j\overline{\mathbf{x}}_j^t$, thus we have

$$\begin{aligned}
\mathbb{E}_{\xi_t}\left[\overline{\mathbf{r}}^t + \frac{q}{m}\sum_{j\in S_t}\mathbf{A}_j\left(\overline{\mathbf{x}}_j^{t+1} - \overline{\mathbf{x}}_j^t\right)\right] &= \mathbb{E}_{\xi_t}\left[\overline{\mathbf{r}}^t + \frac{q}{m}\sum_{i=1}^{q}\mathbf{A}_j\left(\overline{\mathbf{x}}_j^{t+1} - \overline{\mathbf{x}}_j^t\right)\right] \\
&= \mathbb{E}_{\xi_t}\left[\mathbf{A}\left(\frac{q}{m}\overline{\mathbf{x}}^{t+1} - \frac{q-m}{m}\overline{\mathbf{x}}^t\right)\right] \qquad \text{(C.12)}
\end{aligned}$$

Considering the intermediate variable $\overline{\mathbf{x}}_j^{t+1}$ in Eq.(C.5), we have

$$\begin{aligned}
\mathbb{E}_{\xi_t}\left[\overline{\mathbf{x}}_j^{t+1}\right] &= \frac{m}{q}\left(\tilde{\mathbf{x}}_j^{t+1} + \theta\left(\tilde{\mathbf{x}}_j^{t+1} - \mathbf{x}_j^t\right)\right) + \frac{q-m}{q}\overline{\mathbf{x}}_j^t \\
&= \frac{m}{q}\left(\frac{q}{m}\mathbb{E}_{\xi_t}\left[\mathbf{x}_j^{t+1}\right] - \frac{q-m}{m}\mathbf{x}_j^t + \theta\left(\frac{q}{m}\mathbb{E}_{\xi_t}\left[\mathbf{x}_j^{t+1}\right] - \frac{q-m}{m}\mathbf{x}_j^t - \mathbf{x}_j^t\right)\right) + \frac{q-m}{q}\overline{\mathbf{x}}_j^t \\
&= \mathbb{E}_{\xi_t}\left[\mathbf{x}_j^{t+1}\right] + \theta\left(\mathbb{E}_{\xi_t}\left[\mathbf{x}_j^{t+1}\right] - \mathbf{x}_j^t\right) + \frac{q-m}{q}\left(\overline{\mathbf{x}}_j^t - \mathbf{x}_j^t\right)
\end{aligned}$$

Given the parameter $\theta = \frac{m}{q}$, then

$$\mathbb{E}_{\xi_t}\left[\overline{\mathbf{x}}^{t+1}\right] = \left(1 + \frac{m}{q}\right)\mathbb{E}_{\xi_t}\left[\mathbf{x}^{t+1}\right] + \left(1 - \frac{2m}{q}\right)\mathbf{x}^t + \left(1 - \frac{m}{q}\right)\overline{\mathbf{x}}^t$$

Plugging the above equality into the equality (C.12),

$$\mathbb{E}_{\xi_t}\left[\overline{\mathbf{r}}^t+\frac{q}{m}\sum_{j\in S_t}\mathbf{A}_j\left(\overline{\mathbf{x}}_j^{t+1}-\overline{\mathbf{x}}_j^t\right)\right]=\mathbb{E}_{\xi_t}\left[\mathbf{A}\left(\frac{q+m}{m}\mathbf{x}^{t+1}-\frac{q}{m}\mathbf{x}^t\right)\right] \tag{C.13}$$

We assign expectation to both sides of the inequality (C.11) and plug in Eq. (C.13), and after some manipulations,

$$\begin{aligned}\mathbb{E}_{\xi_t}\left[g^*(\mathbf{y}^{t+1})\right]-g^*(\mathbf{y})\leq&\frac{1}{2}\|\mathbf{y}^t-\mathbf{y}\|_{\boldsymbol{\sigma}}^2-\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}\|_{1/\boldsymbol{\sigma}^t}^2\right]-\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}^t\|_{1/\boldsymbol{\sigma}^t}^2\right]\\&+\mathbb{E}_{\xi_t}\left[\left\langle\mathbf{y}^t-\mathbf{y},\mathbf{A}\left(\frac{q+m}{m}\mathbf{x}^{t+1}-\frac{q}{m}\mathbf{x}^t\right)\right\rangle\right].\end{aligned} \tag{C.14}$$

Now we are ready to use the two key inequalities (C.10) and (C.14) to construct the following gap

$$\begin{aligned}&\mathbb{E}_{\xi_t}\left[L(\mathbf{x}^{t+1},\mathbf{y})-L(\mathbf{x},\mathbf{y}^{t+1})\right]\\&=\mathbb{E}_{\xi_t}\left[f(\mathbf{x}^{t+1})+\left\langle\mathbf{y},\mathbf{A}\mathbf{x}^{t+1}\right\rangle\right]-g^*(\mathbf{y})-\left(f(\mathbf{x})+\mathbb{E}_{\xi_t}\left[\left\langle\mathbf{y}^{t+1},\mathbf{A}\mathbf{x}\right\rangle-g^*(\mathbf{y}^{t+1})\right]\right)\\&=\mathbb{E}_{\xi_t}\left[f(\mathbf{x}^{t+1})\right]-f(\mathbf{x})+\mathbb{E}_{\xi_t}\left[g^*(\mathbf{y}^{t+1})\right]-g^*(\mathbf{y})+\mathbb{E}_{\xi_t}\left[\left\langle\mathbf{y},\mathbf{A}\mathbf{x}^{t+1}\right\rangle-\left\langle\mathbf{y}^{t+1},\mathbf{A}\mathbf{x}\right\rangle\right]\end{aligned} \tag{C.15}$$

$$\begin{aligned}&\leq\left(\frac{q}{m}\cdot\frac{1}{2}\|\mathbf{x}^t-\mathbf{x}\|_{1/\boldsymbol{\tau}}^2+\frac{1}{2}\|\mathbf{y}^t-\mathbf{y}\|_{1/\boldsymbol{\sigma}^t}^2+\frac{q-m}{m}f(\mathbf{x}^t)\right)\\&-\left(\frac{q}{m}\cdot\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1}-\mathbf{x}\|_{1/\boldsymbol{\tau}}^2\right]+\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}\|_{1/\boldsymbol{\sigma}^t}^2\right]+\frac{q-m}{m}f(\mathbf{x}^{t+1})\right)\\&-\left(\frac{q}{m}\cdot\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1}-\mathbf{x}^t\|_{\mathbf{h}}^2\right]+\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}^t\|_{1/\boldsymbol{\sigma}^t}^2\right]\right)\\&-\mathbb{E}_{\xi_t}\left[\left\langle\mathbf{y}^t,\mathbf{A}\left(\frac{q}{m}\mathbf{x}^{t+1}-\frac{q-m}{m}\mathbf{x}^t-\frac{m}{q}\mathbf{x}\right)\right\rangle\right]\\&+\mathbb{E}_{\xi_t}\left[\left\langle\mathbf{y}^t-\mathbf{y},\mathbf{A}\left(\frac{q+m}{m}\mathbf{x}^{t+1}-\frac{q}{m}\mathbf{x}^t\right)\right\rangle\right]+\mathbb{E}_{\xi_t}\left[\left\langle\mathbf{y},\mathbf{A}\mathbf{x}^{t+1}\right\rangle-\left\langle\mathbf{y}^{t+1},\mathbf{A}\mathbf{x}\right\rangle\right].\end{aligned} \tag{C.16}$$

After some sophisticated manipulations and rearrangements of the R.H.S of the above inequality, we can obtain

$$\mathbb{E}_{\xi_t}\left[L(\mathbf{x}^{t+1},\mathbf{y})-L(\mathbf{x},\mathbf{y}^{t+1})\right]\leq M(t)-\mathbb{E}_{\xi_t}\left[M(t+1)\right]-\mathbb{E}_{\xi_t}\left[C(t,t+1)\right], \tag{C.17}$$

where

$$\begin{aligned}M(t)=\frac{q}{m}\cdot\frac{1}{2}\|\mathbf{x}^t-\mathbf{x}\|_{1/\boldsymbol{\tau}}^2+\frac{1}{2}\|\mathbf{y}^t-\mathbf{y}\|_{1/\boldsymbol{\sigma}^t}^2-\left\langle\mathbf{y}^t-\mathbf{y},\mathbf{A}\left(\mathbf{x}^t-\mathbf{x}\right)\right\rangle\\+\frac{q-m}{m}\left(f(\mathbf{x}^t)+\left\langle\mathbf{y},\mathbf{A}\mathbf{x}^t\right\rangle-\left(f(\mathbf{x})+\left\langle\mathbf{y},\mathbf{A}\mathbf{x}\right\rangle\right)\right),\end{aligned} \tag{C.18}$$

and

$$\mathbb{E}_{\xi_t}\left[M(t+1)\right] = \frac{q}{m}\cdot\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1}-\mathbf{x}\|^2_{1/\boldsymbol{\tau}}\right] + \frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}\|^2_{1/\boldsymbol{\sigma}^t}\right]$$
$$-\mathbb{E}_{\xi_t}\left[\left\langle \mathbf{y}^{t+1}-\mathbf{y}, \mathbf{A}\left(\mathbf{x}^{t+1}-\mathbf{x}\right)\right\rangle\right] + \frac{q-m}{m}\mathbb{E}_{\xi_t}\left[\left(f(\mathbf{x}^{t+1}) + \left\langle \mathbf{y}, \mathbf{A}\mathbf{x}^{t+1}\right\rangle - (f(\mathbf{x}) + \langle \mathbf{y}, \mathbf{A}\mathbf{x}\rangle)\right)\right], \tag{C.19}$$

and

$$\begin{aligned}\mathbb{E}_{\xi_t}\left[C(t,t+1)\right] &= \frac{q}{m}\cdot\frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2_{1/\boldsymbol{\tau}}\right] + \frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}^t\|^2_{1/\boldsymbol{\sigma}^t}\right] \\ &\quad - \frac{q}{m}\mathbb{E}_{\xi_t}\left[\left\langle \mathbf{y}^{t+1}-\mathbf{y}^t, \mathbf{A}\left(\mathbf{x}^{t+1}-\mathbf{x}^t\right)\right\rangle\right] \qquad \text{(C.20)} \\ &= \frac{q}{m}\cdot\frac{1}{2}\mathbb{E}_{\xi_t}\left[\sum_{j\in S_t}\|\mathbf{x}_j^{t+1}-\mathbf{x}_j^t\|^2_{1/\boldsymbol{\tau}}\right] + \frac{1}{2}\mathbb{E}_{\xi_t}\left[\|\mathbf{y}^{t+1}-\mathbf{y}^t\|^2_{1/\boldsymbol{\sigma}^t}\right] \\ &\quad - \frac{q}{m}\mathbb{E}_{\xi_t}\left[\left\langle \mathbf{y}^{t+1}-\mathbf{y}^t, \sum_{j\in S_t}\mathbf{A}_j\left(\mathbf{x}_j^{t+1}-\mathbf{x}_j^t\right)\right\rangle\right]. \qquad \text{(C.21)}\end{aligned}$$

Now we bound the term $C(t,t+1)$ given the following parameter configuration,

$$\tau_d = \frac{1}{\sum_{j=1}^{D_\mathbf{y}}|A_{jd}|}, \quad d = 1,2,\ldots,D_\mathbf{x}$$
$$\sigma_k^t = \frac{m}{q}\cdot\frac{1}{\sum_{j\in S_t}|A_{kj}|}, \quad k = 1,2,\ldots,D_\mathbf{y}.$$

We can easily observe that the above parameter configuration makes the following symmetric matrix $\mathbf{P}$ diagonally dominant, which guarantees its positive semi-definiteness:

$$\mathbf{P} = \begin{bmatrix}\mathrm{diag}(1/\boldsymbol{\tau}_{S_t}) & -\mathbf{A}_{S_t}^T \\ -\mathbf{A}_{S_t} & \frac{m}{q}\mathrm{diag}(1/\boldsymbol{\sigma}^t)\end{bmatrix} \succeq 0. \tag{C.22}$$

Therefore, this directly leads $C(t,t+1)\geq 0$, and we can further simplify the inequality (C.17),

$$\mathbb{E}_{\xi_t}\left[L(\mathbf{x}^{t+1},\mathbf{y}) - L(\mathbf{x},\mathbf{y}^{t+1})\right] \leq M(t) - \mathbb{E}_{\xi_t}\left[M(t+1)\right]. \tag{C.23}$$

We now apply above inequality recursively, take expectation with respect to all random variables up to time T, and then sum together and we find

$$\mathbb{E}\left[\sum_{t=1}^{T} L(\mathbf{x}^t,\mathbf{y}) - L(\mathbf{x},\mathbf{y}^t)\right] \leq M(0) - \mathbb{E}\left[M(T)\right] \tag{C.24}$$

Since $(\mathbf{x},\mathbf{y})$ is a saddle point, for any t we have

$$f(\mathbf{x}^{t+1}) + \left\langle \mathbf{y}, \mathbf{A}\mathbf{x}^{t+1}\right\rangle - (f(\mathbf{x}) + \langle \mathbf{y}, \mathbf{A}\mathbf{x}\rangle) \geq 0 \tag{C.25}$$

Thanks again to the positive semi-definiteness of the matrix $\mathbf{P}$ and the inequality (C.25) when $t = T - 1$, we have

$$M(T) \geq 0,$$

which further simplifies inequality (C.24)

$$\mathbb{E}\left[\sum_{t=1}^{T} L(\mathbf{x}^t, \mathbf{y}) - L(\mathbf{x}, \mathbf{y}^t)\right] \leq M(0)$$

Finally, applying the convexity of the function $(\mathbf{x}', \mathbf{y}') \mapsto L(\mathbf{x}', \mathbf{y}) - L(\mathbf{x}, \mathbf{y}')$, we have

$$\mathbb{E}\left[L\left(\sum_{t=1}^{T} \mathbf{x}^t/T, \mathbf{y}\right) - L\left(\mathbf{x}, \sum_{t=1}^{T} \mathbf{y}^t/T\right)\right] \leq \frac{1}{T} M(0), \tag{C.26}$$

which completes the proof. □

Bibliography

Abdulle, A., Vilmart, G., and Zygalakis, K. C. (2015). Long time accuracy of Lie-Trotter splitting methods for Langevin dynamics. *SIAM Journal on Numerical Analysis*, 53(1):1–16.

Abernethy, J. and Frongillo, R. (2011). A collaborative mechanism for crowdsourcing prediction problems. In *Advances in Neural Information Processing Systems 24 (NIPS2011)*.

Ahn, S., Korattikara, A., and Welling, M. (2012). Bayesian posterior sampling via stochastic gradient Fisher scoring. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1591–1598.

Alder, B. J. and Wainwright, T. (1959). Studies in molecular dynamics. i. general method. *The Journal of Chemical Physics*, 31(2):459–466.

Amari, S. (2007). Integration of stochastic models by minimizing α-divergence. *Neural Computation*, 19(10):2780–2796.

Bache, K. and Lichman, M. (2013). UCI machine learning repository.

Barbu, A. and Lay, N. (2011). An introduction to artificial prediction markets for classification. arXiv:1102.1465v3.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Bertsekas, D. P. and Tsitsiklis, J. N. (1989). *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. CRC Press.

Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11.

Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145.

Chambolle, A. and Pock, T. (2014). On the ergodic convergence rates of a first-order primal-dual algorithm. *Optimization-online preprint*.

Chandrasekaran, V.and Recht, B., Parrilo, P. A., and Willsky, A. S. (2012a). The convex geometry of linear inverse problems. *Foundations of Computational Mathematics*, 12(6):805–849.

Chandrasekaran, V., Parrilo, P. A., and Willsky, A. S. (2012b). Latent variable graphical model selection via convex optimization. *The Annals of Statistics*, 40(4):1935–1967.

Chen, C., D. N. and Carin, L. (2015). On the convergence of stochastic gradient mcmc algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*.

Chen, S., Donoho, D., and Saunders, M. A. (2001). Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159.

Chen, T., Fox, E. B., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1683–1691.

Chen, Y. and Wortman Vaughan, J. (2010). A new understanding of prediction markets via no-regret learning. In *Proceedings of the 11th ACM conference on Electronic commerce*.

Cole, R. and Fleischer, L. (2007). Fast-converging tatonnement algorithms for the market problem. Technical report, Dept. Computer Science. Dartmouth College.

Dani, V., Madani, O., Pennock, D., Sanghai, S., and Galebach, B. (2006). An empirical comparison of algorithms for aggregating expert predictions. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014). Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654.

Dietrich, F. (2010). Bayesian group belief. *Social choice and welfare*, 35(4):595–626.

Dietrich, F. and List, C. (2014). Probabilistic opinion pooling. *Oxford Handbook of Probability and Philosophy*.

Dietterich, T. (2000). Ensemble methods in machine learning. In *Lecture Notes in Computer Science*, volume 1857, pages 1–5. Springer Verlag.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., and Neven, H. (2014). Bayesian sampling using stochastic gradient thermostats. In *Advances in Neural Information Processing Systems 27*, pages 3203–3211.

Domingos, P. (1997). Why does bagging work? a Bayesian account and its implications. In *Proceedings KDD*.

Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222.

Duchi, J. and Singer, Y. (2009). Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934.

Esser, E., Zhang, X., and Chan, T. (2010). A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM Journal on Imaging Sciences*, 3(4):1015–1046.

Everingham, M. et al. (2006). The 2005 PASCAL visual object classes challenge. In *Selected Proceedings of the first PASCAL Challenges Workshop LNAI*, number 3944, pages 117–176.

Frenkel, D. and Smit, B. (2001). *Understanding Molecular Simulation: From Algorithms to Applications, Second Edition*. Academic Press.

Garg, A., Jayram, T., Vaithyanathan, S., and Zhu, H. (2004). Generalized opinion pooling. *AMAI*.

Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.

Goldbloom, A. (2010). Data prediction competitions – far more than just a bit of fun. In *IEEE International Conference on Data Mining Workshops*.

Green, K. (2006). The $1 million Netflix challenge. *Technology Review*.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning*, volume 2. Springer.

Hateren, J. H. v. and Schaaf, A. v. d. (1998). Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings: Biological Sciences*, 265(1394):359–366.

He, B. and Yuan, X. (2012). Convergence analysis of primal-dual algorithms for a saddle-point problem: from contraction perspective. *SIAM Journal on Imaging Sciences*, 5(1):119–149.

He, Y. and Monteiro, R. D. (2014). An accelerated HPE-type algorithm for a class of composite convex-concave saddle-point problems. *Optimization-online preprint*.

Heskes, T. (1998). Selecting weighting factors in logarithmic opinion pools. In *Advances in Neural Information Processing Systems 10*.

Hiriart-Urruty, J. and Lemaréchal, C. (2001). *Fundamentals of convex analysis*. Springer Science & Business Media.

Homan, M. D. and Gelman, A. (2014). The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *The Journal of Machine Learning Research*, 15(1):1593–1623.

Hong, M. and Luo, Z. (2012). On the linear convergence of the alternating direction method of multipliers. *arXiv preprint arXiv:1208.3922*.

Hoover, W. (1991). *Computational Statistical Mechanics, Studies in Modern Thermodynamics*. Elsevier Science.

Horowitz, A. M. (1991). A generalized guided Monte Carlo algorithm. *Physics Letters B*, 268(2):247–252.

Jacob, L., Obozinski, G., and Vert, J.-P. (2009). Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440. ACM.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323.

Jones, A. and Leimkuhler, B. (2011). Adaptive stochastic methods for sampling driven molecular systems. *The Journal of Chemical Physics*, 135:084125.

Kahn, J. M. (2004). A generative Bayesian model for aggregating experts' probabilities. In *Proceedings of the 20th conference on Uncertainty in Artificial Intelligence*, pages 301–308. AUAI Press.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Konečnỳ, J. and Richtárik, P. (2013). Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*.

Larochelle, H. and Bengio, Y. (2008). Classification using discriminative restricted Boltzmann machines. In *Proceedings of the 25th International Conference on Machine Learning*, pages 536–543.

Lay, N. and Barbu, A. (2010). Supervised aggregation of classifiers using artificial prediction markets. In *Proceedings of ICML*.

LeCun, Y., B. Y. and Hinton, G. (2015). Deep learning. *Nature*, 521:436–444.

Leimkuhler, B. and Matthews, C. (2013). Rational construction of stochastic numerical methods for molecular sampling. *Applied Mathematics Research eXpress*, 2013(1):34–56.

Leimkuhler, B. and Matthews, C. (2015). *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Springer.

Leimkuhler, B., Matthews, C., and Stoltz, G. (2015). The computation of averages from equilibrium and nonequilibrium Langevin molecular dynamics. *IMA Journal of Numerical Analysis*.

Leimkuhler, B. and Reich, S. (2004). *Simulating Hamiltonian dynamics*, volume 14. Cambridge University Press.

Leimkuhler, B. and Shang, X. (2015). Adaptive thermostats for noisy gradient systems. *arXiv preprint arXiv:1505.06889*.

Ma, S., Xue, L., and Zou, H. (2013). Alternating direction methods for latent variable gaussian graphical model selection. *Neural computation*, 25(8):2172–2198.

Maynard-Reid, P. and Chajewska, U. (2001). Aggregating learned probabilistic beliefs. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 354–361. Morgan Kaufmann Publishers Inc.

Meier, L., Van De Geer, S., and Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71.

Metropolis, N. and Ulam, S. (1949). The Monte Carlo method. *Journal of the American statistical association*, 44(247):335–341.

Minka, T. (2002). Bayesian model averaging is not model combination. Technical report, MIT Media Lab Note.

Murphy, K. P. (2007). Conjugate Bayesian analysis of the Gaussian distribution. *def*, 1(2σ2):16.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto. 144pp.

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2.

Nesterov, Y. (2004). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer.

Nesterov, Y. (2012a). Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362.

Nesterov, Y. (2012b). Subgradient methods for huge-scale optimization problems. *Mathematical Programming*, pages 1–23.

Neumaier, A. (2014). Osga: A fast subgradient algorithm with optimal complexity. *arXiv preprint arXiv:1402.1125*.

Nosé, S. (1984). A unified formulation of the constant temperature molecular dynamics methods. *The Journal of Chemical Physics*, 81:511.

Ottaviani, M. and Sørensen, P. (2007). Aggregation of information and beliefs in prediction markets. FRU Working Papers.

Parikh, N. and Boyd, S. (2013). Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231.

Parikh, N. and Boyd, S. (2014). Block splitting for distributed optimization. *Mathematical Programming Computation*, 6(1):77–102.

Pennock, D. and Wellman, M. (1997). Representing aggregate belief through the competitive equilibrium of a securities market. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 392–400.

Pock, T. and Chambolle, A. (2011). Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 1762–1769. IEEE.

Richtárik, P. and Takáč, M. (2012). Parallel coordinate descent methods for big data optimization. *arXiv preprint arXiv:1212.0873*.

Richtárik, P. and Takáč, M. (2014). Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407.

Robert, C. and Casella, G. (2013). *Monte Carlo Statistical Methods*. Springer.

Robert M. Bell, Y. K. and Volinsky, C. (2010). All together now: A perspective on the NETFLIX PRIZE. *Chance*, 24.

Roberts, G. and Rosenthal, J. (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, pages 255–268.

Roth, V. and Fischer, B. (2008). The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *Proceedings of the 25th international conference on Machine learning*, pages 848–855. ACM.

Rubinstein, M. (1974). An aggregation theorem for securities markets. *Journal of Financial Economics*, 1(3):225–244.

Rubinstein, M. (1975). Securities market efficiency in an Arrow-Debreu economy. *American Economic Review*, 65(5):812–824.

Rubinstein, M. (1976). The strong case for the generalised logarithmic utility model as the premier model of financial markets. *Journal of Finance*, 31(2):551–571.

Schmidt, M., Roux, N. L., and Bach, F. (2013). Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*.

Shalev-Shwartz, S. and Zhang, T. (2013). Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory.

Storkey, A. (2009). When training and test sets are different: Characterising learning transfer. In Lawrence, C. S. S., editor, *Dataset Shift in Machine Learning*, chapter 1, pages 3–28. MIT Press.

Storkey, A. (2011). Machine learning markets. In *Proceedings of Artificial Intelligence and Statistics*, volume 15. Journal of Machine Learning Research W&CP.

Storkey, A., Millin, J., and Geras, K. (2012). Isoelastic agents and wealth updates in machine learning markets. In *Proceedings of ICML 2012*.

Su, W., Boyd, S., and Candes, E. (2014). A differential equation for modeling nesterovs accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147.

Tseng, P. (2008). On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*.

Wainwright, M. J. (2014). Structured regularizers for high-dimensional problems: Statistical and computational issues. *Annual Review of Statistics and Its Application*, 1:233–253.

Wang, C., Chen, X., Smola, A. J., and Xing, E. (2013). Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, pages 181–189.

Wang, H., Banerjee, A., and Luo, Z. (2014). Parallel direction method of multipliers. In *Advances in Neural Information Processing Systems 27*, pages 181–189.

Wang, M. C. and Uhlenbeck, G. E. (1945). On the theory of the Brownian motion II. *Reviews of Modern Physics*, 17(2-3):323.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, pages 681–688.

West, M. (1984). Bayesian aggregation. *Journal of the Royal Statistical Society*, 147:600–607.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241 – 259.

Wright, J., Ganesh, A., Rao, S., Peng, Y., and Ma, Y. (2009). Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, pages 2080–2088.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34.

Wu, D. (2009). Parameter estimation for α-GMM based on maximum likelihood criterion. *Neural computation*, 21(6):1776–1795.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zhang, Y. and Xiao, L. (2015). Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML2015)*.

Zhao, P., Rocha, G., and Yu, B. (2009). The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pages 3468–3497.

Zhu, M. and Chan, T. (2008). An efficient primal-dual hybrid gradient algorithm for total variation image restoration. *UCLA CAM Report*, pages 08–34.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.