FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

ONLINE FEATURE SELECTION WITH ANNEALING AND ITS APPLICATIONS

By

LIZHE SUN

A Dissertation submitted to the
Department of Statistics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2019

Lizhe Sun defended this dissertation on April 29th, 2019.
The members of the supervisory committee were:

Adrian Barbu

Professor Directing Dissertation

Piyush Kumar

University Representative

Yiyuan She

Committee Member

Antonio Linero

Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

To my family and my friends

ACKNOWLEDGMENTS

First, I would like to give special thanks to my supervisor Professor Adrian Barbu, who always, on one hand, encouraged me to learn more and explore new algorithms, and on the other hand, provided valuable and insightful advice during my PhD career. Without his assistance, I would not be able to complete this dissertation.

Also, I am thankful to my dissertation committee members Professor Yiyuan She, Professor Antonio Linero, and Professor Piyush Kumar, who have kindly spent their time and efforts on reading this work.

My experience at the Florida State University has been enjoyable, with many friends and professors inside and outside the department of Statistics. I am very grateful for their support for these years. Especially, many thanks to Professor Fred Huffer and Professor Yiyuan She. Professor Huffer took me into the area of theoretical Statistics. Without his help, I could not have had such a strong theoretical background in Statistics. Professor Yiyuan She showed me the big picture for the machine learning. His passion for research motivated me to pursue higher goals. I will never give up my dream to be a better researcher in the area of machine learning.

Finally, I would like to express the most gratitude to my family who have always been supporting and encouraging me unconditionally. They gave me great power to pursue this PhD degree and challenge the hard research problems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

The following short list of symbols are used throughout the document. The symbols represent quantities that I tried to use consistently.

n	the number of observations
p	the number of variables
k	the number of true features
$\mathbf{x}_i$	the observation for the data matrix $\mathbf{X}$, $i = 1, 2, \cdots, n$
$\boldsymbol{\beta}$	the coefficient vector for the linear model
μ_{x_j}	the population mean of random variable X_j, $j = 1, 2, \cdots$
$\bar{x}_j$	the sample mean of random variable X_j, $j = 1, 2, \cdots$
σ_j	the estimated standard deviation of variable X_j
$\boldsymbol{\mu}_{\mathbf{x}}$	the running averages $\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$
μ_y	the running averages $\frac{1}{n}\sum_{i=1}^{n} y_i$
$\mathbf{S}_{xx}$	the running averages $\frac{1}{n}\mathbf{X}^T\mathbf{X}$
$\mathbf{S}_{xy}$	the running averages $\frac{1}{n}\mathbf{X}^T\mathbf{y}$
$\mathbf{S}_{yy}$	the running averages $\frac{1}{n}\mathbf{y}^T\mathbf{y}$
$\|\boldsymbol{\beta}\|_0$	the ℓ_0 penalty for the coefficient vector $\boldsymbol{\beta}$
$\|\boldsymbol{\beta}\|_1$	the ℓ_1 penalty for the coefficient vector $\boldsymbol{\beta}$
$\|\boldsymbol{\beta}\|_2$	the ℓ_2 penalty for the coefficient vector $\boldsymbol{\beta}$
∇f	the derivative for the function $f(\cdot)$

ABSTRACT

Feature selection is an important technique for high dimensional statistics and machine learning. It has many applications in computer vision, natural language processing, bioinformatics, etc. However, most of the feature selection methods in the literature are proposed for offline learning while the existing online feature selection methods have limitations in true feature recovery. In this dissertation, we propose some novel online feature selection methods and a framework. One is called Stochastic Feature Selection with Annealing, and the other one is the framework of running averages. Based on the methods and the framework we developed, we can recover the support of the true features with higher accuracy. We provide a theoretical analysis, and through simulations and experiments on real sparse datasets, we show that our proposed methods compare favorably with some state-of-the-art online methods in the literature.

CHAPTER 1

INTRODUCTION

Feature selection is a very important topic in high dimensional statistics and machine learning, and has been studied for many years. In high dimensional statistics, various feature selection methods were proposed in the literature, such as [7, 38, 52, 48]. By removing the irrelevant and redundant features, feature selection methods can improve the prediction accuracy, enhance the model interpretability, and reduce the computational complexity. Also, feature selection methods have many applications, for instance in computer vision, nature language processing and bioinformatics.

However, the existing feature selection methods have some limitations. First, most of existing feature selection methods are restricted to the offline learning setup, which assumes that all the features and observations are given in advance. For instance, the classical regularized methods are designed for the offline learning case, such as the ℓ_1 based method [38], $\ell_1+\ell_2$ based method [52], and non-convex penalized methods such as [7, 48]. However, in the real world applications, the offline assumption may not hold when we have to handle the large streaming data, where observations arrive sequentially with the time period t. Besides, considering the problem of data storage, a very large dataset cannot fit in the computer memory. Here, we have some examples for the applications of the large streaming data. Consider the 120-day URL data for malicious website detection [21], the training instances are collected day by day, and there are about two million observations in total and the features are more than three millions. In this case, it is natural to consider using regularized-based approach to solve the problem. However, on such a massive dataset, the classical penalty based algorithms will be computational expensive and memory demanding. In the click-through rate (CTR) prediction problem, a very large number of users click the advertisement everyday, so it is not surprising that the advertisement companies collect millions of observations everyday. For example, in the Avazu click-through data [14], there are more than 4 million instances collected in one day, and the number of features is one million. The last example is the credit card fraud detection. Because there are a very large number of people that use credit cards everyday, the credit card center will gather many transaction records and the reports of fraud everyday. Actually, it is

a typical example for the online imbalanced data classification, since the fraud records are a very small part of all transaction records. In these cases, we cannot use and deploy the regular batch learning feature selection techniques timely and sometimes it is impossible to fit such large datasets in the computer memory.

Moreover, online learning approaches converge slower than offline learning methods. In general, we use online gradient descent to update the coefficients in the online models. However, online gradient descent is a sequential method, using one observation or a mini-batch of observations for acceleration [4] in each time period, so we cannot access the full gradient at each iteration. As a consequence, online algorithms suffer a lower convergence rate than the traditional offline learning algorithms, $\mathcal{O}(1/\sqrt{n})$ for general convexity [50] and $\mathcal{O}(\log(n)/n)$ for strongly convex functions [30]. In comparison, offline gradient descent enjoys the convergence rate of $\mathcal{O}(1/n)$, which is much faster than the online gradient descent.

Finally, the existing online feature selection methods have limitations in true features recovery. There are some online feature selection methods in the literature, such as online proximal gradient (OPG) [5, 6], regularized dual averaging (RDA) [44], and their variants OPG-ADMM and RDA-ADMM [26, 37]. The above methods use convex-relaxation approaches by replacing the ℓ_0-norm with the ℓ_1-norm as a sparsity inducing penalty. Some other feature selection methods were proposed based on greedy-optimization techniques, such as the truncated gradient [18], diffusion approximation [8], and the second order methods in [40, 43]. Although these methods will induce sparse solutions and improve the model interpretability, they cannot recover the support of true features even under mild assumptions on the data matrix $\mathbf{X}$. Also, it is inefficient to control the sparsity level through the ℓ_1 penalty. In this dissertation, we performed simulations to support these statements.

Motivated by these concerns, we propose in this dissertation novel online feature selection methods and a framework. In the paper [40], the authors describe the goal of online feature selection for supervised learning: developing online methods that involve only a small and fixed number of features for regression and classification. Based on this goal, we would like to solve one further important problem: designing methods to recover the support of all true features for the linear sparse model. In general, most of the online feature selection methods can recover the support of the true features for datasets in which the features are independent with each other.

But, for the data with strongly correlated features, these existing methods cannot perform very well in the true support recovery. Thus, in this dissertation, we will propose new ideas to solve this issue. Our contribution can be split into two parts, one is the Stochastic Feature Selection with Annealing (SFSA) method, which can handle data no matter whether the features are orthogonal or strongly correlated. And the other one is the running averages framework for online supervised learning.

The major contributions in this dissertation are: (1) we propose a novel algorithm and a new framework for online feature selection; (2) compared to the other online feature selection methods, our methods can simultaneously recover the support of the true features for the data with strongly correlated features and build a regression or classification model on the selected features; (3) we provide a theoretical analysis for our methods; (4) we verify the empirical performance of our methods by conducting numerical experiments on simulated and real data.

1.1 Related Work

In this dissertation, we will present literature review for two important parts related to the our work for online feature selection. The first part is online learning and online feature selection. In this part, we will start our introduction from the traditional feature selection methods, then we will extend them into the online case. After that, we will introduce the variants of some classic online feature selection. Finally, we talk about the performance of the above methods. It worth to mention that, in the paper [40], the authors think that the existing feature selection methods can be grouped into three categories: supervised, unsupervised, and semi-supervised. In this dissertation, we just focus on supervised learning.

The second part is about greedy algorithms and the feature selection with annealing algorithm (FSA). Feature selection with annealing, proposed in [1], is the main idea we use to select features. Also, in this dissertation, we provide theoretical guarantees for FSA and its variants. Thus, we would like to review the literature related to greedy algorithms and feature selection with annealing.

1.1.1 Online Feature Selection

Nowadays, high dimensional data analysis is very common in the machine learning community. To handle high dimensional data (e.g. $p > n$), various feature selection techniques have been

developed to exploit the sparse structure of the coefficients in the linear model. The most famous among these feature selection methods is the regularized method based on the ℓ_1 penalty, which we call Lasso [38] for linear regression. Then, various penalized methods were proposed by researchers. For instance, as an extension of Lasso, the Elastic net was proposed in [52]. Considering the limitations of the true feature recovery for Lasso, Zou also proposed the adaptive Lasso in [51], which can select all true features in the low dimensional case ($n > p$). The Lasso estimator is derived from a convex optimization problem, which is convenient for computation. However, since it is based on the ℓ_1 penalty, the Lasso estimator is biased. To solve this problem, Fan and Li proposed the Smoothly Clipped Absolute Deviation (SCAD) penalty in [7] to replace the ℓ_1 penalty. Later on, Zhang proposed the Minimax Concave Penalty (MCP) in [48]. Both of these regularized methods will induce a non-convex optimization problem, but they can provide an unbiased estimator. Besides the ℓ_1 penalty and the non-convex penalty, many methods were proposed in the literature [1, 12, 20, 49] to solve the ℓ_0 penalty problem.

In the online learning case, there are two frameworks for online feature selection based on convex optimization. One is the Forward-Backward-Splitting method [5], building an online feature selection framework by online proximal gradient (OPG) [6]. Duchi and Singer [5] proposed this algorithmic framework for the regularized convex optimization problem, which is to minimize the following sum of two functions:

$$\min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{\beta}; \mathbf{z}_i) + \mathbf{P}(\boldsymbol{\beta}; \lambda), \tag{1.1}$$

in which the first part is the empirical loss, and the second part is the penalty function. The hyper-parameter λ is a tuning parameter. Although it is very easy to extend the penalty to non-convex functions, the authors just considered the convex penalty in this paper. The OPG method can be split into two step: one step uses gradient descent to update the coefficients, the second step is to solve a convex optimization problem.

The other online feature selection framework is Xiao's Regularized Dual Averaging method (RDA) [44], which extended the primal-dual sub-gradient method [23] to the online case. The RDA algorithmic framework is also used to solve the optimization problem from eq. (1.1). However, unlike the OPG framework which can be applied for both of online and offline cases, the RDA method is designed based on online streaming data. In general, we need to solve the following

optimization problem in the RDA framework:

$$\boldsymbol{\beta}^{(t+1)} = \arg\min_{\boldsymbol{\beta}} \left\{ \langle \bar{\mathbf{g}}_t, \boldsymbol{\beta} \rangle + \mathbf{P}(\boldsymbol{\beta}; \lambda) + \frac{\gamma_t}{t} h(\boldsymbol{\beta}) \right\},$$

in which $\mathbf{P}(\cdot;\cdot)$ is a penalty function, and $h(\boldsymbol{\beta})$ is an auxiliary strongly convex function, and $\{\gamma_t\}_{t\geq 1}$ is a nonnegative and nondecreasing input sequence, which determines the convergence rate of the algorithm. Besides, we can compute the average of the gradient by using

$$\bar{\mathbf{g}}_t = \frac{t-1}{t}\bar{\mathbf{g}}_{t-1} + \frac{1}{t}\mathbf{g}_t.$$

This is why this method is called the dual averaging method.

Within the both of frameworks, some variants have been developed. For instance, the stochastic ADMM algorithm can be designed in both frameworks as OPG-ADMM and RDA-ADMM [37]. Independently, Ouyang also proposed the stochastic ADMM algorithm in [26], the same algorithm as OPG-ADMM.

Another line of research are the greedy-based online feature selection methods such as diffusion approximation [8], truncated gradient [18], FOFS/SOFS [40, 43], as the first order/second order online feature selection method.

There is a third line of research called online streaming feature selection [42, 45]. In this scenario, features arrive one at a time while all the training examples are available before the learning process starts, and the goal is to select a subset of features and build an appropriate model at each time. Unlike the traditional online learning, the disadvantage of this new online scenario is that we cannot select all true features and train a model for prediction until all features are disclosed. In this dissertation, we assume that observations arrive sequentially in time, so we will not consider algorithms such as [42] or [45] for comparison.

Finally, the online Newton method [11] uses a similar idea with our running averages idea to update the inverse of the Hessian matrix. This method enjoys the computational complexity $\mathcal{O}(p^2)$, but they did not address the issues of variable standardization and feature selection.

1.1.2 Feature Selection with Annealing

The Feature Selection with Annealing (FSA) algorithm is designed by Barbu, She, and their co-workers in the paper [1]. FSA is used for the constrained optimization problem

$$\min_{\|\boldsymbol{\beta}\|_0 \leq k} \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{\beta}; \mathbf{z}_i),$$

in which $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ and $\ell(\boldsymbol{\beta}; \mathbf{z}_i)$ is the loss function. The FSA algorithm will be applied in all the dissertation. Also, we will introduce the main idea of the FSA algorithm in Chapter 2 and Chapter 3. In this part, we will review the literature related to the FSA algorithm.

The FSA algorithm belongs to the class of greedy algorithms. Usually, greedy algorithms are used to solve the ℓ_0 constrained optimization problem. In the paper [49], Zhang proposed a forward backward greedy algorithm to study the feature selection problem in the linear regression model. Then, in the paper [20], the authors extended this method for the general convex optimization problem. Besides, Iterative Hard Thresholding (IHT) [12] and Hard Thresholding Pursuit (HTP) [9] are also classical algorithms for ℓ_0 constrained problems.

In this dissertation, the main idea for the theoretical analysis of FSA comes from the HTP algorithm. In the literature, there are some papers deal with the problem of support recovery by HTP algorithm. In the paper [46], the authors proved that, when the restricted isometry property (RIP) condition holds, the HTP algorithm can recover the support of the true features. However, Shen and Li [32] provided the proof of the same conclusion without using the RIP condition. Here, our proof use the similar technique with the paper [46]. We prove that, with the RIP condition, our FSA algorithm can recovery the support of true features with a high probability.

1.2 Setup and Notation

In this dissertation, we consider the sparse learning problem for the regularized linear regression and classification model. Let $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ be a sequence of i.i.d training instances, arriving one at a time. Our goal is to minimize the constrained optimization problem

$$\min_{\|\boldsymbol{\beta}\|_0 \leq k} \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{\beta}; \mathbf{z}_i) + \mathbf{P}(\boldsymbol{\beta}; \lambda), \tag{1.2}$$

in which $\ell(\cdot; \mathbf{z}_i) : \mathbb{R}^p \to \mathbb{R}$ is a per-example loss function, $\mathbf{P}(\cdot; \cdot)$ is the penalty function, and λ is a tuning parameter for the linear model. The coefficient vector $\boldsymbol{\beta}$ is estimated sequentially, one example at a time, from $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \cdots, \boldsymbol{\beta}_{i-1}$ we obtain the coefficient vector $\boldsymbol{\beta}_i$. For instance, in Chapter 2, we will optimize the loss function of the linear regression model:

$$L(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2 + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2, \tag{1.3}$$

and the loss of the logistic regression model for classification

$$L(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\log(1+e^{-y_i\mathbf{x}_i^T\boldsymbol{\beta}}) + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2, \tag{1.4}$$

with $y_i \in \{-1, +1\}$. In the theoretical analysis of online learning, it is of interest to obtain an upper bound of the regret,

$$R_n = \frac{1}{n}\sum_{i=1}^{n}[\ell(\boldsymbol{\beta}_i; \mathbf{z}_i) + \mathbf{P}(\boldsymbol{\beta}_i; \lambda)] - \min_{\boldsymbol{\beta}}\left[\frac{1}{n}\sum_{i=1}^{n}\ell(\boldsymbol{\beta}; \mathbf{z}_i) + \mathbf{P}(\boldsymbol{\beta}; \lambda)\right] \tag{1.5}$$

which measures what is lost in terms of loss compared to their offline versions, and in a way measuring the speed of convergence of the online algorithms.

Here, we establish the notation formally. Vectors are lower case bold letters, such as $\mathbf{x} \in \mathbb{R}^d$, and scalars are lower case letters, e.g. $x \in \mathbb{R}$. A sequence of vectors is denoted by subscripts, i.e. $\mathbf{w}_1, \mathbf{w}_2, \cdots$, and the entries in a vector are denoted by non-bold subscripts, like w_j. Matrices are upper case bold letters, such as $\mathbf{M} \in \mathbb{R}^{d\times d}$, and random variables are upper case letters, such as Z. Given a vector $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \cdots, \gamma_n)^T \in \mathbb{R}^n$, we define vector norms: $\|\boldsymbol{\gamma}\|_1 = \sum_{i=1}^n |\gamma_i|$, $\|\boldsymbol{\gamma}\| = \sqrt{\sum_{i=1}^n \gamma_i^2}$ and $\|\boldsymbol{\gamma}\|_0 = \#\{j : \gamma_j \neq 0\}$. Finally, $\nabla_S f(\mathbf{x})$ is the gradient vector of f restricted to the variables indexed by S.

CHAPTER 2

STOCHASTIC FEATURE SELECTION WITH ANNEALING

2.1 Stochastic Feature Selection with Annealing

The new online feature selection method we proposed here is motivated by the feature selection with annealing (FSA) algorithm [1]. Similar to the FSA algorithm for offline learning, the SFSA algorithm can simultaneously solve the feature selection problem and train a model for prediction.

According to the description in [1], the key ideas in this feature selection algorithm design are: (1) use an annealing procedure to lessen the greediness in reducing the dimensionality from p to k, (2) gradually remove the most irrelevant variables to facilitate computation. The algorithm starts with an initialized parameter vector $\boldsymbol{\beta}$, generally $\boldsymbol{\beta} = 0$, and then alternates two basic steps: one step updating the parameters to minimize the per-example loss $\ell(\boldsymbol{\beta}; \mathbf{z}_i)$ by gradient descent

$$\boldsymbol{\beta} = \boldsymbol{\beta} - \eta \frac{\partial f_i}{\partial \boldsymbol{\beta}},$$

in which $f_i(\boldsymbol{\beta}) = \ell(\boldsymbol{\beta}; \mathbf{z}_i) + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2$, and the other step is a feature selection step that removes some variables based on the ranking of $|\beta_j|$, $j = 1, 2, \cdots, p$.

The difference between our stochastic feature selection with annealing and the offline counterpart is that one approximates the gradient by using a mini-batch of observations, and the other one uses all data to construct the gradient. It also differs on how it handles non-normalized data, since in the offline FSA one can directly normalize the data, while in the proposed algorithm one uses estimated variances for the variables, as described in Remark 1. The algorithm is summarized in Algorithm 1.

In general, we will input a mini-batch of observations rather than one at a time, but the one at a time situation can be always recovered by setting the mini-batch size to 1. Also, we design an annealing schedule M_t, the number of features to keep at time t:

$$M_t = k + (p - k)\max\{0, \frac{T - t}{t\mu + T}\}, t = 1, 2, \cdots, T,$$

in which k is the desired sparsity level, μ is the annealing parameter in this model and T is the maturity time for this schedule, when exactly k features have been selected.

Algorithm 1 Stochastic Feature Selection with Annealing

Input: Training data $\mathbf{z}_t = (\mathbf{x}_t, y_t)$ arriving one at a time, learning rate η, sparsity level k, annealing parameter μ, maturity time T.

Output: Trained regression parameter vector $\boldsymbol{\beta}$ with $\|\boldsymbol{\beta}\|_0 \leq k$.

Initialize $\boldsymbol{\beta} = 0$.

for $t = 1$ to ∞ **do**

Receive an observation $\mathbf{x}_t$.

Predict $\hat{\mathbf{y}}_t$.

Receive the true $\mathbf{y}_t$.

Update $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta \frac{\partial f(\boldsymbol{\beta}, \mathbf{z}_t)}{\partial \boldsymbol{\beta}}$

Keep only the M_t variables with highest $|\beta_j|$ and renumber them $1, ..., M_t$.

end for

A longer maturity time uses more observations, therefore it has better feature selection capabilities. In practice, we use multiple maturity times as illustrated in Figure 2.1 so that we always have a "current" set of selected features of a desired sparsity while building a better one as more data is available.

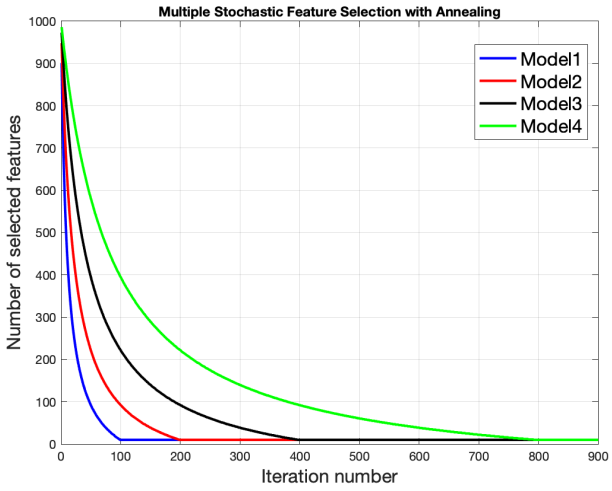

Figure 2.1: Multiple maturity times T for stochastic FSA.

Finally, we would like to emphasize that the Algorithm 1 is just based on simple stochastic gradient descent. However, we also can combine SFSA algorithm with momentum, Nesterov accel-

erated gradient [35] and Adam [17]. Some of these techniques will be evaluated in the experimental section.

2.2 Truncated Stochastic Gradient Descent

From the stochastic feature selection with annealing algorithm, we can see that using online gradient descent we can select features by keeping the largest k coefficient magnitudes. Now we consider a special case for online feature selection where we don't use an annealing procedure with online gradient descent, but select the largest k coefficient magnitudes at time T. The prototype algorithm is described in Algorithm 2.

In the literature, somewhat similar truncated stochastic gradient descent algorithms were also proposed in [8], [18] and [40], based on different loss functions. However, their methods will truncate the coefficient magnitudes at each time t. This will mislead the algorithms to select irrelevant features, especially when the features have strong correlation. Our Algorithm 2 will select the subset of features according to the largest values of the coefficient magnitudes after T iterations, which will improve the accuracy for feature selection.

Algorithm 2 Truncated Stochastic Gradient Descent

Input: Training data $\mathbf{z}_t = (\mathbf{x}_t, y_t)$ arriving one at a time, learning rate η, sparsity level k, maturity time T.

Output: Trained regression parameter vector $\boldsymbol{\beta}$ with $\|\boldsymbol{\beta}\|_0 \leq k$.

Initialize $\boldsymbol{\beta} = 0$.

for $t = 1$ to ∞ **do**

  Receive an observations $\mathbf{x}_t$.

  Predict $\hat{\mathbf{y}}_t$.

  Receive the true $\mathbf{y}_t$.

  Update $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta \frac{\partial f(\boldsymbol{\beta}, \mathbf{z}_t)}{\partial \boldsymbol{\beta}}$

end for

Keep only the k variables with highest $|\beta_j|$.

2.3 Theoretical Analysis

In this section, we will prove that the SFSA algorithm will converge to the true $\boldsymbol{\beta}^*$ when the loss function is strongly convex. Our proof follows the OPG framework [6] and is inspired by [47].

The SFSA Algorithm 1 consists of the following two steps at each iteration:

$$\begin{aligned}\tilde{\boldsymbol{\beta}}_{t+1} &= \boldsymbol{\beta}_t - \eta\frac{\partial f(\boldsymbol{\beta}_t, \mathbf{z}_t)}{\partial \boldsymbol{\beta}},\\ \boldsymbol{\beta}_{t+1} &= \Theta_{M_t}(\tilde{\boldsymbol{\beta}}_{t+1}),\end{aligned}$$

in which $\Theta_s(\cdot)$ is the hard thresholding operator that keeps s features. In this paper, we define $\boldsymbol{\beta}^*$ as the true coefficient vector, and assume $\|\boldsymbol{\beta}^*\|_0 = k^*$. Before we start to prove our main theorem, we give a definition of the Restricted Strong Convexity/Smoothness(RSC/RSS) property.

Definition 1. (Restricted Strong Convexity/Smoothness) *For any integer $s > 0$, we say that a differentiable function $f(x)$ is restricted strongly convex (RSC) with parameter m_s and restricted strongly smooth (RSS) with parameter M_s if there exist $m_s, M_s > 0$ such that*

$$\frac{m_s}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}'\|^2 \leq f(\boldsymbol{\beta}) - f(\boldsymbol{\beta}') - \langle \nabla f(\boldsymbol{\beta}'), \boldsymbol{\beta} - \boldsymbol{\beta}'\rangle \leq \frac{M_s}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}'\|^2$$

for $\forall \|\boldsymbol{\beta} - \boldsymbol{\beta}'\|_0 \leq s$.

In this dissertation, we consider the regularized linear model, using the ℓ_2 penalized loss functions (1.3) and (1.4). Thus the loss function for each observation is strongly convex and the Hessian matrix $\nabla^2 f(\boldsymbol{\beta})$ must be positive definite. As a result, there exist $M, m > 0$ satisfying

$$0 < m < \lambda_{\min}(\nabla^2 f(\boldsymbol{\beta})) < \lambda_{\max}(\nabla^2 f(\boldsymbol{\beta})) < M.$$

where $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ are the smallest and largest eigenvalues. It is also easy to verify that there exist $M_s, m_s > 0$ satisfying the RSC/RSS conditions for the per-example loss function at any sparsity level ($0 < s \leq p$).

Now, we introduce the two basic Lemmas for our theoretical analysis. Lemma 1 is a classical result of gradient descent method, and Lemma 2 shows the upper bound of hard thresholding function. We can find the proof of Lemma 1 in [24], and Lemma 2 is a main theorem in [33].

Lemma 1. *Assume that a differentiable function f is restricted strongly convex/smooth with parameter m_s/M_s for some $s \leq p$. For any $\boldsymbol{\beta}$, $\boldsymbol{\beta}'$ with $|S_{\boldsymbol{\beta}} \cup S_{\boldsymbol{\beta}'}| \leq s$, if the learning rate $\eta \in (0, 2m_s/M_s^2)$, then*

$$\|\boldsymbol{\beta} - \boldsymbol{\beta}' - \eta\nabla f(\boldsymbol{\beta}) + \eta\nabla f(\boldsymbol{\beta}')\| \leq \sqrt{1 - 2\eta m_s + \eta^2 M_s^2}\|\boldsymbol{\beta} - \boldsymbol{\beta}'\|$$

in which $\sqrt{1 - 2\eta m_s + \eta^2 M_s^2} < 1$.

Lemma 2. *Let $\mathbf{y} \in \mathbb{R}^d$ be an arbitrary d-dimensional vector, and $\mathbf{x} \in \mathbb{R}^d$ be any k^*-sparse vector, thus $\|\mathbf{x}\|_0 = k^* \leq k$. Then, we have the following bound*

$$\|\Theta_k(\mathbf{y}) - \mathbf{x}\| \leq \sqrt{\nu}\|\mathbf{y} - \mathbf{x}\|, \ \nu = 1 + \frac{\gamma + \sqrt{(4+\gamma)\gamma}}{2},$$

in which $\gamma = \dfrac{\min\{k^*, d-k\}}{k - k^* + \min\{k^*, d-k\}}$ and $\Theta_k(\mathbf{y})$ is the hard thresholding operator that keeps the k largest magnitudes $|y_i|$. Because $\gamma \leq 1$, it is easy to verify that $\sqrt{\nu_{\min}} \leq 1.62$.

We start to build our main theorem now. The proofs of the following proposition and theorem are inspired from the proof in [47]. In Proposition 1, we build the relationship between $\boldsymbol{\beta}^{(t+1)}$ and $\boldsymbol{\beta}^{(t)}$ and disclose how the $\boldsymbol{\beta}^{(t)}$ converge to the true $\boldsymbol{\beta}^*$. And in the Theorem 1, we prove that, under some conditions, and after t iterations, the SFSA algorithm will converge to the true coefficients $\boldsymbol{\beta}^*$.

Proposition 1. *Let $\boldsymbol{\beta}^*$ be an arbitrary k^*-sparse vector, so $\|\boldsymbol{\beta}^*\|_0 = k^*$, $\boldsymbol{\beta}^{(t)}$ be the SFSA coefficient vector at iteration t, $S_{\boldsymbol{\beta}^{(t)}}$ be its support, and $k = |S_{\boldsymbol{\beta}^{(t)}}| \geq k^*$. Let $s = |S_{\boldsymbol{\beta}^{(t)}} \cup S_{\boldsymbol{\beta}^*}| \leq k + k^*$, and if f is a differentiable function which is m_s-convex and M_s-smooth, then for any learning rate $0 < \eta < 2m_s/M_s^2$, we have*

$$\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^*\| \leq 1.62\rho\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| + 1.62\eta\sqrt{s}\|\nabla f_t(\boldsymbol{\beta}^*)\|_\infty, \tag{2.1}$$

where $\rho = \sqrt{1 - 2\eta m_s + \eta^2 M_s^2} < 1$.

Proof. Let S be the index set $S = S_{\boldsymbol{\beta}^{(t)}} \cup S_{\boldsymbol{\beta}^*}$, thus we must have $|S| \leq k + k^*$. Consider the following vector

$$\tilde{\boldsymbol{\beta}}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \eta\nabla_S f_t(\boldsymbol{\beta}^{(t)}),$$

where ∇_S has been defined in Section 1.2. By using the triangle inequality, we have

$$\begin{aligned}
\|\tilde{\boldsymbol{\beta}}^{(t+1)} - \boldsymbol{\beta}^*\| &= \|\boldsymbol{\beta}^{(t)} - \eta\nabla_S f_t(\boldsymbol{\beta}^{(t)}) - \boldsymbol{\beta}^*\| \\
&\leq \|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^* - \eta\nabla_S f_t(\boldsymbol{\beta}^{(t)}) - \eta\nabla_S f_t(\boldsymbol{\beta}^*)\| + \eta\|\nabla_S f_t(\boldsymbol{\beta}^*)\| \\
&\leq \rho\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| + \eta\sqrt{s}\|\nabla f_t(\boldsymbol{\beta}^*)\|_\infty,
\end{aligned}$$

where the last inequality follows from Lemma 1 and the fact that $\|\nabla_S f_t(\boldsymbol{\beta}^*)\| \leq \sqrt{s}\|\nabla_S f_t(\boldsymbol{\beta}^*)\|_\infty$. Then because we have $\boldsymbol{\beta}^{(t+1)} = \Theta_k(\tilde{\boldsymbol{\beta}}^{(t+1)})$, by using Lemma 2, we have that (2.1) follows. □

Theorem 1. *Let $\boldsymbol{\beta}^{(0)} = 0$ and $S_{\boldsymbol{\beta}^{(0)}} = \{1, 2, \cdots, p\}$. With the same notations and settings as Proposition 1, assume that $M_s/m_s < 1.26$ and there exists a constant $G > 0$, s.t. $\|\nabla f_t(\boldsymbol{\beta})\|_\infty < G, \forall \boldsymbol{\beta} \in \mathbb{R}^p$. Then the SFSA coefficient vector $\boldsymbol{\beta}^{(t)}$, satisfies*

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^t \|\boldsymbol{\beta}^*\| + \frac{1.62\eta\sqrt{p}}{1 - 1.62\rho} G.$$

Proof. Based on our conditions and settings, for any $k^* \leq k \leq p$, we have $s = k + k^*$, $\eta \in (0, 2m_s/M_s^2)$, and $M_s/m_s < 1.26$. Thus, $\rho < 0.62$ and $1.62\rho < 1$. By using Proposition 1 recursively with decreasing k from p to k^*, for instance, at the time period t, we have

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq 1.62\rho \|\boldsymbol{\beta}^{(t-1)} - \boldsymbol{\beta}^*\| + 1.62\eta\sqrt{s}\|\nabla f_t(\boldsymbol{\beta})\|_\infty,$$

and at the time period $t-1$, we have

$$\|\boldsymbol{\beta}^{(t-1)} - \boldsymbol{\beta}^*\| \leq 1.62\rho \|\boldsymbol{\beta}^{(t-2)} - \boldsymbol{\beta}^*\| + 1.62\eta\sqrt{s'}\|\nabla f_t(\boldsymbol{\beta})\|_\infty,$$

in which s and s' are the number of selected features at time period t and $t-1$, respective. Because $\|\nabla f_t(\boldsymbol{\beta})\|_\infty < G$, and $s = k + k^* \leq s' \leq p$, then we have

$$\begin{aligned} \boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| &\leq (1.62\rho)^2 \|\boldsymbol{\beta}^{(t-2)} - \boldsymbol{\beta}^*\| + 1.62^2\rho\eta\sqrt{s'}\|\nabla f_t(\boldsymbol{\beta})\|_\infty + 1.62\eta\sqrt{s}\|\nabla f_t(\boldsymbol{\beta})\|_\infty \\ &\leq (1.62\rho)^2 \|\boldsymbol{\beta}^{(t-2)} - \boldsymbol{\beta}^*\| + 1.62^2\rho\eta\sqrt{p}G + 1.62\eta\sqrt{p}G, \end{aligned}$$

By applying the above inequality repeatedly all the way to $t = 0$ we can get

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^t \|\boldsymbol{\beta}^{(0)} - \boldsymbol{\beta}^*\| + [(1.62\rho)^{t-1} + ... + 1]1.62\eta\sqrt{p}G.$$

which implies the conclusion since $\boldsymbol{\beta}^{(0)} = 0$. □

Remark 1. *The above proposition and theorem assume that the data is normalized. If that is not the case, we can maintain the running averages $\hat{\boldsymbol{\mu}}_{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i \in \mathbb{R}^p$ and $\mathbf{S}_{\mathbf{x}} = \text{diag}\{\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T\} \in \mathbb{R}^p$ by updating them one example at a time. Then we can use $\hat{\boldsymbol{\sigma}}_{\mathbf{x}} = \mathbf{S}_{\mathbf{x}} - \hat{\boldsymbol{\mu}}_{\mathbf{x}}^2 \in \mathbb{R}^p$ to perform SFSA on non-normalized data using the thresholding $\Theta_k(\hat{\boldsymbol{\sigma}}_{\mathbf{x}}\boldsymbol{\beta})$ instead of $\Theta_k(\boldsymbol{\beta})$.*

In Theorem 1, the non-vanishing term in the error bound shows that the estimation error in SFSA is controlled by the upper bound of $\|\nabla f_t(\boldsymbol{\beta})\|_\infty$, $\forall t > 0$. Here we defined it as a constant $G > 0$. Thus, by using Theorem 1, we can provide the following theoretical guarantee that the true support recovery is possible for SFSA when $\beta^*_{\min}$ is significant larger than the upper bound of $\sqrt{p}\|\nabla f_t(\boldsymbol{\beta})\|_\infty$, $\forall t > 0$.

Corollary 1. *Assume that the conditions in Proposition 1 and Theorem 1 hold, and also assume that* $\|\hat{\boldsymbol{\beta}}\|_0 = k^*$. *With the condition*

$$\beta^*_{\min} := \min_{j \in S_{\boldsymbol{\beta}^*}} |\beta^*_j| > \frac{2\eta\sqrt{p}}{1 - 1.62\rho} G,$$

after $t = [\frac{1}{1.62\rho} \log(\frac{2\|\boldsymbol{\beta}^*\|}{\beta_{\min}})] + 1$ *iterations, the SFSA algorithm will output a* $\boldsymbol{\beta}^{(t)}$ *satisfying* $S_{\boldsymbol{\beta}^{(t)}} = S_{\boldsymbol{\beta}^*}$.

Proof. When $t > [\frac{1}{1.62\rho} \log(\frac{2\|\boldsymbol{\beta}^*\|}{\beta_{\min}})] + 1$, we have

$$(1.62\rho)^t \|\boldsymbol{\beta}^*\| < \frac{1}{2}\beta_{\min}.$$

Then, we can imply that

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| < \beta_{\min}.$$

Thus, $S_{\boldsymbol{\beta}^*} = S_{\boldsymbol{\beta}^{(t)}}$ must hold. □

CHAPTER 3

RUNNING AVERAGES FOR ONLINE SUPERVISED LEARNING

In Chapter 2, we proposed the stochastic feature selection with annealing (SFSA) method for online feature selection, and provide theoretical guarantees for it. However, SFSA method is still in the traditional framework, the online proximal gradient (OPG). In this chapter, we will develop a new framework to solve online feature selection problem. The new framework we proposed here for online learning is based on the statistical query model [3, 15], and we call our framework the running averages framework.

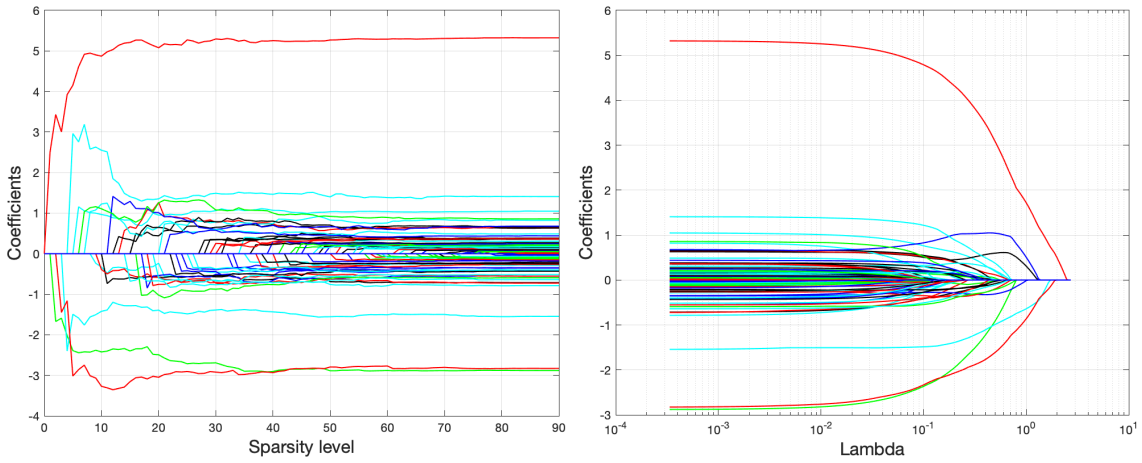

Figure 3.1: The solution path for online OLS-th (Left) and online Lasso (Right) for the Year Prediction MSD dataset.

Many of the methods proposed in our framework enjoy a fast convergence rate and can recover the support of the true signal. Moreover, the proposed methods can address the issue of model selection, which is to obtain models with different sparsity levels and decide on the best model, e.g. using an AIC/BIC criterion. For example in Figure 3.1 are shown the solution paths obtained by the proposed online least squares with thresholding method, as well as the proposed online

Table 3.1: Comparison between different online methods

Algorithm	Memory	Computation		Convergence		Feature Selection	True Feature Recovery
		Running Avgs.	Algorithms	Coefficients	Regret		
SGD	$\mathcal{O}(p)$	-	$\mathcal{O}(np)$	$O(n^{-1/2})$	Slow	No	No
SADMM[26]	$\mathcal{O}(p)$	-	$\mathcal{O}(np)$	$O(n^{-1/2})$	Slow	Yes	No
SIHT[8]	$\mathcal{O}(p)$	-	$\mathcal{O}(np)$	$O(\log(n)/n)$	Slow	Yes	No
OFSA	$\mathcal{O}(p^2)$	$\mathcal{O}(np^2)$	$\mathcal{O}(p^2)$	$O(n^{-1})$	Fast	Yes	Yes
OLS-th	$\mathcal{O}(p^2)$	$\mathcal{O}(np^2)$	$\mathcal{O}(p^3)$	$O(n^{-1})$	$O(n^{\alpha-1})$	Yes	Yes
OMCP	$\mathcal{O}(p^2)$	$\mathcal{O}(np^2)$	$\mathcal{O}(p^2)$	$O(n^{-1})$	Fast	Yes	Yes
OElnet	$\mathcal{O}(p^2)$	$\mathcal{O}(np^2)$	$\mathcal{O}(p^2)$	$O(n^{-1})$	Fast	Yes	No

Lasso method. A brief summary of the convergence rates and computational complexity of various methods including the proposed methods are shown in Table 3.1.

Here, we summarize the advantages and disadvantages of the proposed running averages algorithms: although the proposed online methods based on running averages sacrifice computational complexity and memory compared with classical online methods, they enjoy a fast convergence rate and high estimation accuracy. More importantly, the proposed methods can select features and recover the support of true features with high accuracy and they can obtain models with any desired sparsity level for model selection at any time.

3.1 Running Averages

The idea of running averages comes from the statistical query model and the issues of standard online methods. In mathematical statistics, given a distribution with unknown parameters θ and the i.i.d random variables $X_1, X_2, \cdots, X_n$, a sufficient statistic $T(X_1, X_2, \cdots, X_n)$ contains all the information necessary for estimating the model parameters.

In big data learning, the large datasets cannot fit in memory, and the online methods in the literature cannot recover the support of the true features. Motivated by these concerns, we propose the running averages framework, which contains two modules, a running averages module that is updated online as new data is available, and a model extraction module that can build different types of models with any desired sparsity from the running averages. A diagram of the framework is shown in Figure 3.2.

Let $(\mathbf{x}_i, y_i)$, $i = \overline{1,n}$ be observations with $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, and we denote data matrix $\mathbf{X} = (\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_n^T)^T$, $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$. The running averages are the

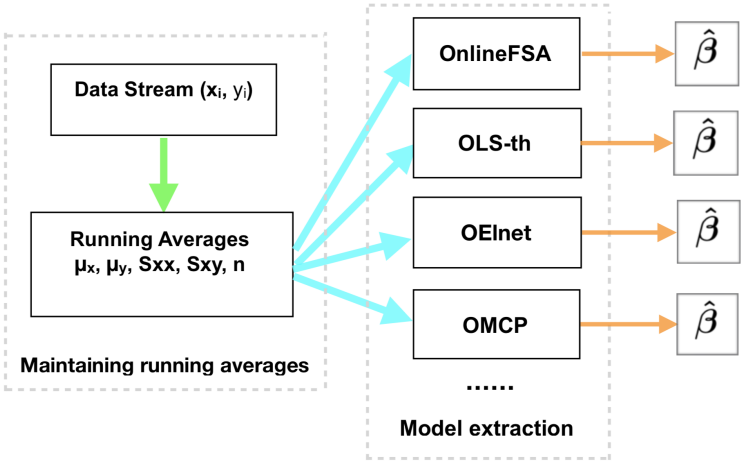

Figure 3.2: Diagram of the running averages based methods. The running averages are updated as the data is received. The model is extracted from the running averages only when desired.

cumulative averages over the observations. They are

$$\begin{aligned} \boldsymbol{\mu}_x &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i,\ \mu_y = \frac{1}{n}\sum_{i=1}^{n} y_i, \\ \mathbf{S}_{xx} &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T,\ \mathbf{S}_{xy} = \frac{1}{n}\sum_{i=1}^{n} y_i\mathbf{x}_i,\ S_{yy} = \frac{1}{n}\sum_{i=1}^{n} y_i^2 \end{aligned}$$

and the sample size n. The running averages can be updated in an incremental manner, for example

$$\boldsymbol{\mu}_x^{(n+1)} = \frac{n}{n+1}\boldsymbol{\mu}_x^{(n)} + \frac{1}{n+1}\mathbf{x}_{n+1}, \tag{3.1}$$

similar to the procedure from Chapter 2.5 in [36].

The running averages have the following advantages: a) they cover all necessary sample information for model estimation, b) the dimension of the running averages will not increase with sample size n, c) they can be used in the online learning setting because they can be updated one example at one time.

3.1.1 Data Standardization

Data standardization is an important procedure in real data analysis, especially for feature selection, because a feature could have an arbitrary scale (unit of measure) and the scale should not influence its importance in the model. For this purpose, the data matrix $\mathbf{X}$ and the response vector $\mathbf{y}$ are usually standardized by removing the mean, and $\mathbf{X}$ is further standardized by making

all columns on the same scale. However, because we discard the data and only use the running averages, we will need to standardize the running averages.

Denote $\mathbf{1}_n = [1, 1, \cdots, 1]^T \in \mathbb{R}^n$, and by σ_{x_j} the sample standard deviation for the random variable X_j. By running averages, we can estimate the standard deviation of variable j as:

$$\sigma_{x_j} = \sqrt{(\mathbf{S}_{xx})_j - (\boldsymbol{\mu}_x)_j^2},$$

in which $(\mathbf{S}_{xx})_j$ is the j-th diagonal entry for $p \times p$ matrix $\mathbf{S}_{xx}$. Then, denote by $\boldsymbol{\Pi} = \text{diag}(\sigma_{x_1}, ..., \sigma_{x_p})^{-1}$ the $p \times p$ diagonal matrix containing the inverse of standard deviations σ_{x_j} on the diagonal. Denoting by $\tilde{\mathbf{X}}$ the standardized data matrix $\mathbf{X}$, and $\tilde{\mathbf{y}}$ as the centralized $\mathbf{y}$, we can standardize data as

$$\tilde{\mathbf{X}} = (\mathbf{X} - \mathbf{1}_n \boldsymbol{\mu}_x^T)\boldsymbol{\Pi}, \;\; \tilde{\mathbf{y}} = (\mathbf{y} - \mu_y \mathbf{1}_n)$$

From these equations we obtain the running averages of the standardized dataset:

$$\mathbf{S}_{\tilde{x}\tilde{y}} = \frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{y}} = \frac{1}{n}\boldsymbol{\Pi}\mathbf{X}^T\mathbf{y} - \mu_y\boldsymbol{\Pi}\boldsymbol{\mu}_x = \boldsymbol{\Pi}\mathbf{S}_{xy} - \mu_y\boldsymbol{\Pi}\boldsymbol{\mu}_x \tag{3.2}$$

$$\mathbf{S}_{\tilde{x}\tilde{x}} = \frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} = \boldsymbol{\Pi}(\frac{\mathbf{X}^T\mathbf{X}}{n} - \boldsymbol{\mu}_x\boldsymbol{\mu}_x^T)\boldsymbol{\Pi} = \boldsymbol{\Pi}(\mathbf{S}_{xx} - \boldsymbol{\mu}_x\boldsymbol{\mu}_x^T)\boldsymbol{\Pi} \tag{3.3}$$

For convenience, hereinafter, we will still use $\mathbf{S}_{xx}$ and $\mathbf{S}_{xy}$ to represent the running averages after standardization.

3.2 Algorithms

In this section, we propose several running averages-based online algorithms. First, we design online least squares based on running averages, which can be used for feature selection by thresholding. We also propose the online feature selection with annealing (OFSA) to solve the constrained least squares problem. Then we consider some regularization models, such as Lasso, Elastic Net, and Minimax Concave Penalty. To simplify notation, we denote OLS to represent online least squares, OLSth for online least squares with thresholding, OLasso for online Lasso, OElnet for online elastic net, and OMCP for online minimax concave penalty.

3.2.1 Preliminaries

Before we start introducing the running averages-based algorithms, we prove that these online algorithms are equivalent to their offline counterparts. Actually, in our running averages framework,

we share the same objective loss function with offline learning, which is the key point to prove their equivalence.

Proposition 2. *Consider the following penalized regression problem:*

$$\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2 + \mathbf{P}(\boldsymbol{\beta};\lambda), \tag{3.4}$$

in which $\boldsymbol{\beta}$ *is the coefficient vector and* $\mathbf{P}(\boldsymbol{\beta};\lambda) = \sum_{j=1}^{p}\mathbf{P}(\beta_j;\lambda)$ *is a penalty function. It is equivalent to the online optimization problem based on running averages.*

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\boldsymbol{\beta}^T\mathbf{S}_{xx}\boldsymbol{\beta} - \boldsymbol{\beta}^T\mathbf{S}_{xy} + \mathbf{P}(\boldsymbol{\beta};\lambda), \tag{3.5}$$

Proof. The loss function (3.4) can be rewritten as

$$\begin{aligned}\frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2 + \mathbf{P}(\boldsymbol{\beta};\lambda) &= \frac{1}{2n}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \mathbf{P}(\boldsymbol{\beta};\lambda)\\ &= \frac{\mathbf{y}^T\mathbf{y}}{2n} - \frac{\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y}}{n} + \boldsymbol{\beta}^T\frac{\mathbf{X}^T\mathbf{X}}{2n}\boldsymbol{\beta} + \sum_{j=1}^{p}\mathbf{P}(\beta_j;\lambda),\end{aligned}$$

in which $S_{yy} = \mathbf{y}^T\mathbf{y}/n$, $\mathbf{S}_{xy} = \mathbf{X}^T\mathbf{y}/n$, and $\mathbf{S}_{xx} = \mathbf{X}^T\mathbf{X}/n$ are running averages. Thus, the offline learning problem is equivalent to our running averages online learning problem. □

3.2.2 Online Least Squares

In OLS, we need to find the solution for the equations $\frac{\mathbf{X}^T\mathbf{X}}{n}\boldsymbol{\beta} = \frac{\mathbf{X}^T\mathbf{y}}{n}$. Since $\frac{\mathbf{X}^T\mathbf{X}}{n}$ and $\frac{\mathbf{X}^T\mathbf{y}}{n}$ can be computed by using running averages, we obtain:

$$\mathbf{S}_{xx}\boldsymbol{\beta} = \mathbf{S}_{xy}. \tag{3.6}$$

Thus, online least squares is equivalent to offline least squares.

3.2.3 Online Least Squares with Thresholding

The OLSth is aimed at solving the following constrained minimization problem:

$$\min_{\|\boldsymbol{\beta}\|_0 \le k} \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2. \tag{3.7}$$

It is a non-convex and NP-hard problem because of the sparsity constraint. Here, we propose a three step procedure to solve it: first, we use the online least squares to estimate $\hat{\boldsymbol{\beta}}$, then we remove unimportant variables according to the coefficient magnitudes $|\beta_j|$, $j = 1, 2, \cdots, p$. Finally, we use least squares to refit the model on the subset of selected features. The prototype algorithm is described in Algorithm 3. In the high dimensional case ($p > n$), we can use the ridge regression estimator in the first step.

Algorithm 3 OLS with Thresholding

Input: Training running averages $\mathbf{S}_{xx}, \mathbf{S}_{xy}$ and sample size n, sparsity level k.

Output: Trained regression parameter vector $\boldsymbol{\beta}$ with $\|\boldsymbol{\beta}\|_0 \leq k$.

1: Find $\hat{\boldsymbol{\beta}}$ by OLS.
2: Keep only the k variables with largest $|\hat{\beta}_j|$.
3: Fit the model on the selected features by OLS.

3.2.4 Online Feature Selection with Annealing

Unlike OLSth, OFSA is an iterative thresholding algorithm. The OFSA algorithm can simultaneously solve the coefficient estimation problem and the feature selection problem. The main ideas in OFSA are: 1) uses an annealing plan to lessen the greediness in reducing the dimensionality from p to k, 2) removes irrelevant variables to facilitate computation. The algorithm starts with an initialized parameter $\boldsymbol{\beta}$, generally $\boldsymbol{\beta} = 0$, and then alternates two basic steps: one is updating the parameters to minimize the loss $L(\boldsymbol{\beta})$ by gradient descent

$$\boldsymbol{\beta} = \boldsymbol{\beta} - \eta \frac{\partial L}{\partial \boldsymbol{\beta}},$$

and the other one is a feature selection step that removes some variables based on the ranking of $|\beta_j|$, $j = 1, 2, \cdots, p$. In the second step, we design an annealing schedule to decide the number of features M_t we keep in each time period t,

$$M_t = k + (p - k) \max\{0, \frac{T - t}{t\mu + T}\}, t = 1, 2, \cdots, T.$$

More details are shown in [1] about the offline FSA algorithm, such as applications and theoretical analysis. For the square loss, the computation of

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = -\frac{\mathbf{X}^T \mathbf{y}}{n} + \frac{\mathbf{X}^T \mathbf{X} \boldsymbol{\beta}}{n} = \mathbf{S}_{xx} \boldsymbol{\beta} - \mathbf{S}_{xy}, \tag{3.8}$$

falls into our running averages framework. Thus, we derive the OFSA which is equivalent to the offline FSA in [1]. The algorithm is summarized in Algorithm 4.

3.2.5 Online Regularization Methods

Penalized methods can also be used to select features, and we can map them into our running averages framework. A popular one is the Lasso estimator [38], it solves the convex optimization

Algorithm 4 Online FSA

Input: Training running averages $\mathbf{S}_{xx}, \mathbf{S}_{xy}$, sample size n, learning rate η, hyper-parameter μ, and sparsity level k.

Output: Trained regression parameter vector $\boldsymbol{\beta}$ with $\|\boldsymbol{\beta}\|_0 \leq k$.

Initialize $\boldsymbol{\beta} = 0$.

for $t = 1$ to n^{iter} **do**

Update $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta(\mathbf{S}_{xx}\boldsymbol{\beta} - \mathbf{S}_{xy})$

Keep only the M_t variables with highest $|\beta_j|$ and renumber them $1, ..., M_t$.

end for

Fit the model on the selected features by OLS.

problem

$$\arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{3.9}$$

in which $\lambda > 0$ is a tuning parameter.

Besides Lasso, the SCAD [7], the Elastic Net [52] and the MCP [48] were proposed to deal with the variable selection and estimation problem. Here, we use the gradient-based method with a thresholding operator $\boldsymbol{\Theta}(t; \lambda)$ to solve the regularized loss minimization problems [31]. For instance, in Lasso and Elastic net, $\boldsymbol{\Theta}$ is the soft thresholding operator, and in MCP,

$$\boldsymbol{\Theta}(t; \lambda) = \begin{cases} 0 & \text{if } 0 \leq |t| \leq \lambda, \\ \frac{t - \lambda \text{ sign}(t)}{1 - 1/b} & \text{if } \lambda < |t| \leq b\lambda, \\ t & \text{if } |t| > b\lambda, \end{cases} \tag{3.10}$$

in which b is a constant. The general algorithm is given in Algorithm 5.

Algorithm 5 Online Regularized Methods by GD

Input: Training running averages $\mathbf{S}_{xx}, \mathbf{S}_{xy}$, sample size n, learning rate η, penalty parameter λ.

Output: Trained sparse regression parameter vector $\boldsymbol{\beta}$.

Initialize $\boldsymbol{\beta} = 0$.

for $t = 1$ to n^{iter} **do**

Update $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta(\mathbf{S}_{xx}\boldsymbol{\beta} - \mathbf{S}_{xy})$

Update $\boldsymbol{\beta} \leftarrow \boldsymbol{\Theta}(\boldsymbol{\beta}; \eta\lambda)$

end for

Fit the model on the selected features by OLS.

3.2.6 Online Classification Methods

The aforementioned algorithms not only can select features for regression, but can also be used for classification, even though these algorithms are based on the ℓ_2 loss. In fact, for the two class problem with labels $+1$ and -1, the coefficient vector for classification from linear least squares is proportional to the coefficient vector by linear discriminant analysis without intercept [10]. Besides, one can use the Lasso method to select variable for classification under some assumptions [25]. We will give the theoretical guarantees in Section 3.3.

3.2.7 Memory and Computational Complexity

In general, the memory complexity for the running averages is $\mathcal{O}(p^2)$ because $\mathbf{S}_{xx}$ is a $p \times p$ matrix. The computational complexity of maintaining the running averages is $\mathcal{O}(np^2)$. Except for OLSth, the computational complexity for obtaining the model using the running average-based algorithms is $\mathcal{O}(p^2)$ based on the limited number of iterations, each taking $\mathcal{O}(p^2)$ time. As for OLSth, it is $\mathcal{O}(p^3)$ if done by Gaussian elimination or $O(p^2)$ if done using an iterative method such as conjugate gradient that takes much fewer iterations than p. We can conclude that the running averages storage does not depend on the sample size n, and the computation is linear in n. Hence, when $n >> p$, compared to the batch learning algorithms, the running averages based methods need less memory and have less computational complexity. And they can achieve the same convergence rate as the batch learning algorithms.

3.2.8 Model Adaptation

Detecting changes in the underlying model and rapidly adapting to the changes are common problems in online learning, and some applications are based on varying-coefficient models [13]. Our running averages online methods can adapt to coefficients change for large scale data streams. For that the update equation (3.1) can be regarded in a more general form as

$$\boldsymbol{\mu}_x^{(n+1)} = (1 - \alpha_n)\boldsymbol{\mu}_x^{(n)} + \alpha_n \mathbf{x}_{n+1} \tag{3.11}$$

where we only show one of the running averages for illustration but the same type of updates are used for all of them.

The original running averages use $\alpha_n = 1/(n+1)$, which gives all observations equal weight in the running average. For the coefficients-varying models, we use a larger value of α_n that gives more

weight to the recent observations. However, too much adaptation is also not good because then the model will not be able to recover weak coefficients that can only be recovered given sufficiently many observations. More details about simulation and application will be covered in Chapter 5.

3.3 Theoretical Analysis

In this section we will give the theoretical analysis for our methods. First, because of Proposition 2, we have the equivalence of the online penalized models including Lasso, Elastic Net, SCAD and MCP with their offline counterparts, and thus all their theoretical guarantees of consistency, oracle inequalities, etc., carry over to their online counterparts.

Here, we start by showing that the OLSth and OFSA method can recover the support of the true features with high probability. The main idea of our proof comes from [47]. After that, we will prove an upper bound for the regret of the OLSth method. In the end of theoretical part, we will give a theoretical justification for the support recovery of our method in classification.

We start building our theory from two lemmas. The proof of Lemma 3 and 4 can be found in the textbook [39].

Lemma 3. *If X is a Gaussian random variable $X \sim N(0, \sigma^2)$, then for all $t > 0$,*

$$\mathbb{P}(|X| \geq t) \leq 2\exp\{-t^2/2\sigma^2\}$$

Lemma 4. *Let $\mathbf{X} \in \mathbb{R}^{n\times p}$ be a data matrix drawn from $\mathcal{N}(0, \mathbf{\Sigma})$. Then when $n > p$, for all $t > 0$, the minimum singular value satisfies the lower deviation inequality*

$$P\Big(\frac{\lambda_{\min}(\mathbf{X})}{\sqrt{n}} \leq \lambda_{\min}(\sqrt{\mathbf{\Sigma}})(1-t) - \sqrt{\frac{tr(\mathbf{\Sigma})}{n}}\Big) \leq e^{-nt^2/2},$$

in which $\lambda_{\min}(\mathbf{X})$ is the smallest singular value of matrix $\mathbf{X}$.

Here, we start to present our first result, when the sample size n is large enough, OLSth can recover the support of true features with a high probability. Also, we will cover the data normalization in our theoretical analysis. Although the intercept β_0 is necessary in the real data applications, we will not cover it here.

Proposition 3. *Suppose we have the linear model*

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim N(0, \sigma^2\mathbf{I}),$$

where $\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_n^T]^T$ *is the data matrix, in which* $\mathbf{x}_i \in \mathbb{R}^p$, $i = 1, 2, \cdots$, *are independently drawn from* $\mathcal{N}(0, \mathbf{\Sigma})$. *Let* $\alpha \in (0, 1]$ *and* $S_{\boldsymbol{\beta}^*} = \{j, \beta_j^* \neq 0\}$, $|S_{\boldsymbol{\beta}^*}| = k^*$ *and*

$$\min_{j \in S_{\boldsymbol{\beta}^*}} |\beta_j^*| > \frac{2\sigma}{\lambda}\sqrt{\frac{\log(p)}{n^\alpha}}, \text{ for some } \lambda \text{ satisfying } 0 < \lambda \leq \lambda_{\min}(\frac{1}{n}\mathbf{X}^T\mathbf{X}). \tag{3.12}$$

Then with probability $1 - 2p^{1-C_1 n^{1-\alpha}}$, *where* $C_1 > 1$ *is a constant, the index set of top* k^* *values of* $|\hat{\boldsymbol{\beta}}_j|$ *is exactly* $S_{\boldsymbol{\beta}^*}$.

Proof. According to the assumptions, we have $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}^* + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim N(0, \sigma^2\mathbf{I})$. It is equivalent to that

$$|\hat{\boldsymbol{\beta}}| = |\boldsymbol{\beta}^* + \left(\frac{\mathbf{X}^T\mathbf{X}}{n}\right)^{-1}\frac{\mathbf{X}^T\boldsymbol{\epsilon}}{n}|.$$

Then by the Lemma 3, for $\forall j = 1, 2, \cdots, p$ and $\forall t > 0$ we have

$$\mathbb{P}\left(|\frac{\mathbf{x}_j^T\boldsymbol{\epsilon}}{n}| \geq t\right) \leq 2\exp\{-\frac{n^2t^2}{2\sigma^2\|\mathbf{x}_j\|_2^2}\},$$

in which $\frac{\mathbf{x}_j^T\boldsymbol{\epsilon}}{n}$ has the normal distribution $\mathcal{N}(0, \frac{\sigma^2}{n} \cdot \frac{\|\mathbf{x}_j\|_2^2}{n})$ for the given $\mathbf{x}_j$. In general case, we can assume that $\|\mathbf{x}_j\|_2^2/n$, $\forall j = 1, 2, \cdots, p$, is bounded by a constant $C > 0$. Here we can assume that $0 < C < 2$, and it is easy to verify that $C = 1$ when we standardize the data matrix $\mathbf{X}$. Then, let $t = 2\sigma\sqrt{\frac{\log(p)}{n^\alpha}}$, so we have

$$\mathbb{P}\left(|\frac{\mathbf{x}_j^T\boldsymbol{\epsilon}}{n}| \geq 2\sigma\sqrt{\frac{\log(p)}{n^\alpha}}\right) \leq 2\exp\{-\frac{2n^{2-\alpha}\log(p)}{\|\mathbf{x}_j\|_2^2}\} \leq 2\exp\{-\frac{2n^{1-\alpha}\log(p)}{C}\}, \text{for } \forall j = 1, 2, \cdots, p.$$

Let $C_1 = \frac{2}{C}$, so $C_1 > 1$, and we use the union bound of the above inequality,

$$\mathbb{P}\left(\|\frac{\mathbf{X}^T\boldsymbol{\epsilon}}{n}\|_\infty \leq 2\sigma\sqrt{\frac{\log(p)}{n^\alpha}}\right) \geq 1 - 2p\exp\{-C_1 n^{1-\alpha}\log(p)\} = 1 - 2p^{1-C_1 n^{1-\alpha}}.$$

Therefore, with probability $1 - 2p^{1-C_1 n^{1-\alpha}}$, for $\forall j \notin S_{\boldsymbol{\beta}^*}$ we have

$$|\hat{\beta}_j| \leq \frac{1}{\lambda_{\min}(\frac{1}{n}\mathbf{X}^T\mathbf{X})}\|\frac{\mathbf{X}^T\boldsymbol{\epsilon}}{n}\|_\infty \leq \frac{2\sigma}{\lambda_{\min}(\frac{1}{n}\mathbf{X}^T\mathbf{X})}\sqrt{\frac{\log(p)}{n^\alpha}} \leq \frac{2\sigma}{\lambda}\sqrt{\frac{\log(p)}{n^\alpha}}.$$

where we used the fact $\frac{1}{\lambda} \geq \frac{1}{\lambda_{\min}(\frac{1}{n}\mathbf{X}^T\mathbf{X})}$. And by the smallest $\boldsymbol{\beta}^*$ condition, we get our conclusion.

□

Corollary 2. *With the same conditions as in the Proposition 3, let $\alpha = 1$, so we have the condition for the minimum $\boldsymbol{\beta}^*$:*

$$\min_{j \in S_{\boldsymbol{\beta}^*}} |\beta_j^*| > \frac{2\sigma}{\lambda}\sqrt{\frac{\log(p)}{n}}, \text{ for some } \lambda \text{ satisfying } 0 < \lambda \leq \lambda_{\min}(\frac{1}{n}\mathbf{X}^T\mathbf{X}). \tag{3.13}$$

Then with probability $1 - 2p^{1-C_1}$, where $C_1 > 1$ is a constant, the index set of top k^* values of $|\hat{\boldsymbol{\beta}}_j|$ is exactly $S_{\boldsymbol{\beta}^*}$.

Proof. Let $\alpha = 1$ in Proposition 3 and then we can get the conclusion. □

The Proposition 3 shows the theoretical guarantee of true feature recovery for OLSth. We can observe that the probability of true feature recovery does not depend on the true sparsity k^*. We will verify it by numerical experiments in Chapter 5. Here, we also provide the theoretical guarantees for the standardization case.

Remark 2. *Denote $\Pi = \text{diag}\{\sigma_{\mathbf{x}_1}, \sigma_{\mathbf{x}_2}, \cdots, \sigma_{\mathbf{x}_p}\}^{-1}$, $\hat{\Pi} = \text{diag}\{\hat{\sigma}_{\mathbf{x}_1}, \hat{\sigma}_{\mathbf{x}_2}, \cdots, \hat{\sigma}_{\mathbf{x}_p}\}^{-1}$. Given the conditions $\min_{j \in S_{\boldsymbol{\beta}^*}} |\sigma_{\mathbf{x}_j}\beta_j^*| > \frac{2\sigma}{\lambda}\sqrt{\frac{\log(p)}{n}}$, for some λ satisfying $0 < \lambda \leq \lambda_{\min}(\frac{1}{n}\Pi\mathbf{X}^T\mathbf{X}\Pi)$, then with high probability the index set of top k^* values of $|\sigma_{\mathbf{x}_j}\hat{\boldsymbol{\beta}}_j|$ is exactly $S_{\boldsymbol{\beta}^*}$.*

In the following theorem, we will consider the relationship of the smallest eigenvalue between the true covariance matrix $\boldsymbol{\Sigma}$ and the sample covariance matrix $\frac{1}{n}\mathbf{X}^T\mathbf{X}$.

Theorem 2. (True feature recovery for OLS-th) *With the same notations as Proposition 3, if*

$$\min_{j \in S_{\boldsymbol{\beta}^*}} |\beta_j^*| > \frac{2\sigma}{\lambda}\sqrt{\frac{\log(p)}{n^\alpha}}, \text{ for some } \lambda \text{ s.t. } \sqrt{\lambda} \leq 0.8\lambda_{\min}(\sqrt{\boldsymbol{\Sigma}}) - \rho(\boldsymbol{\Sigma})\sqrt{\frac{p}{n}}, \tag{3.14}$$

where $\rho(\boldsymbol{\Sigma})$ is the largest diagonal value in Σ, then with probability $1 - 2p^{1-C_1 n^{1-\alpha}} - e^{-n/50}$ the index set of top k^* values of $|\hat{\boldsymbol{\beta}}_j|$ is exactly $S_{\boldsymbol{\beta}^*}$.

Proof. According to Lemma 3, we have

$$\mathbb{P}\left(\frac{\lambda_{\min}(\mathbf{X})}{\sqrt{n}} \geq \lambda_{\min}(\sqrt{\boldsymbol{\Sigma}})(1-\delta) - \sqrt{\frac{\text{tr}(\boldsymbol{\Sigma})}{n}}\right) > 1 - \exp\{-n\delta^2/2\},$$

From here since $\text{tr}(\boldsymbol{\Sigma}) \leq p\rho^2(\boldsymbol{\Sigma})$ we obtain

$$\mathbb{P}(\frac{\lambda_{\min}(\mathbf{X})}{\sqrt{n}} \geq \lambda_{\min}(\sqrt{\boldsymbol{\Sigma}})(1-\delta) - \rho(\boldsymbol{\Sigma})\sqrt{\frac{p}{n}}) > 1 - \exp\{-n\delta^2/2\}.$$

Taking $\delta = 1/5$ and since $0 < \sqrt{\lambda} \leq \lambda_{\min}(\sqrt{\mathbf{\Sigma}})(1-\delta) - \rho(\mathbf{\Sigma})\sqrt{\frac{p}{n}}$, we have

$$\mathbb{P}(\frac{\lambda_{\min}(\mathbf{X})}{\sqrt{n}} \geq \sqrt{\lambda}) > 1 - \exp\{-n/50\}$$

Because $(\lambda_{\min}(\mathbf{X})/\sqrt{n})^2 = \lambda_{\min}(\mathbf{X}^T\mathbf{X})/n$, thus we have

$$\mathbb{P}(\frac{\lambda_{\min}(\mathbf{X}^T\mathbf{X})}{n} \geq \lambda) > 1 - \exp\{-n/50\}.$$

Thus, combining with Proposition 3 and Lemma 3, we complete the proof. □

Then we consider the theoretical guarantees of true feature recovery for OFSA algorithms. Here, we need to use the definition of restricted strong convexity/smoothness (RSC/RSS), which we used in Chapter 2. In the linear regression case, the RSC/RSS conditions are equivalent to the restricted isometric property (RIP):

$$m_s\|\boldsymbol{\beta} - \boldsymbol{\beta}'\|^2 \leq \frac{1}{n}\|\mathbf{X}(\boldsymbol{\beta} - \boldsymbol{\beta}')\|^2 \leq M_s\|\boldsymbol{\beta} - \boldsymbol{\beta}'\|^2, \ \forall \|\boldsymbol{\beta} - \boldsymbol{\beta}'\|_0 \leq s. \tag{3.15}$$

And when $n > p$, the RIP condition will degenerate to

$$0 < m < \lambda_{\min}(\frac{\mathbf{X}^T\mathbf{X}}{n}) < \lambda_{\max}(\frac{\mathbf{X}^T\mathbf{X}}{n}) < M.$$

Proposition 4. *With the same conditions as Proposition 3, let $\boldsymbol{\beta}^*$ be an arbitrary k^*-sparse vector, so $\|\boldsymbol{\beta}^*\|_0 = k^*$. Let $\boldsymbol{\beta}^{(t)}$ be the OFSA coefficient vector at iteration t, $S_{\boldsymbol{\beta}^{(t)}}$ be its support, $k = |S_{\boldsymbol{\beta}^{(t)}}| \geq k^*$ and $s = k + k^*$. If f is a differentiable function which is m_s-convex and M_s-smooth, then for any learning rate $0 < \eta < 2m_s/M_s^2$, we have*

$$\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^*\| \leq 1.62\rho\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| + 1.62\eta\sqrt{s}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty,$$

where $\rho = \sqrt{1 - 2\eta m_s + \eta^2 M_s^2} < 1$.

Proof. Let $S = S_{\boldsymbol{\beta}^{(t)}} \cup S_{\boldsymbol{\beta}^*}$. Consider the following vector

$$\tilde{\boldsymbol{\beta}}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \eta\nabla_S f(\boldsymbol{\beta}^{(t)}),$$

By using the triangle inequality, we have

$$\begin{aligned}
\|\tilde{\boldsymbol{\beta}}^{(t+1)} - \boldsymbol{\beta}^*\| &= \|\boldsymbol{\beta}^{(t)} - \eta\nabla_S f(\boldsymbol{\beta}^{(t)}) - \boldsymbol{\beta}^*\| \\
&\leq \|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^* - \eta\nabla_S f(\boldsymbol{\beta}^{(t)}) - \eta\nabla_S f(\boldsymbol{\beta}^*)\| + \eta\|\nabla_S f(\boldsymbol{\beta}^*)\| \\
&\leq \rho\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| + \eta\sqrt{s}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty,
\end{aligned}$$

where the last inequality follows from lemma 1 and the fact $\|\nabla_S f(\boldsymbol{\beta}^*)\| \leq \sqrt{s}\|\nabla_S f(\boldsymbol{\beta}^*)\|_\infty$. Then we also have $\boldsymbol{\beta}^{(t+1)} = \Theta_k(\tilde{\boldsymbol{\beta}}^{(t+1)})$, thus by following Lemma 2, we can get

$$\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^*\| \leq 1.62\rho\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| + 1.62\eta\sqrt{s}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty.$$

□

Theorem 3. (Convergence of OFSA) *With the same assumptions as Proposition 4, let* $\boldsymbol{\beta}^{(0)} = 0$ *and* $S_{\boldsymbol{\beta}^{(0)}} = \{1, 2, \cdots, p\}$*.Assume we have* $M_s/m_s < 1.26$ *for any* $k^* \leq s \leq p$*. Then, with the probability* $1 - 2p^{1-C_1}$*, where* $C_1 > 1$ *is a constant, the OFSA coefficient vector* $\boldsymbol{\beta}^{(t)}$ *satisfies*

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^t\|\boldsymbol{\beta}^*\| + \frac{3.24\sigma\eta}{1 - 1.62\rho}\sqrt{\frac{p\log(p)}{n}}.$$

Proof. Because for any $k^* \leq s \leq p$ and $\eta \in (0, 2m_s/M_s^2)$, we have $M_s/m_s < 1.26$. Thus, we can get that $\rho < 0.62$ and $1.62\rho < 1$. Then, by using Proposition 4 recursively, we can get the upper bound of the $\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\|$ when the dimension of $\boldsymbol{\beta}^{(t)}$ decrease from p to k^*. At the time period t, we have

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq 1.62\rho\|\boldsymbol{\beta}^{(t-1)} - \boldsymbol{\beta}^*\| + 1.62\eta\sqrt{s}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty,$$

and at the time period $t-1$, we also have

$$\|\boldsymbol{\beta}^{(t-1)} - \boldsymbol{\beta}^*\| \leq 1.62\rho\|\boldsymbol{\beta}^{(t-2)} - \boldsymbol{\beta}^*\| + 1.62\eta\sqrt{s'}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty,$$

in which s and s' are the number of selected features at time period t and $t-1$, respective. Thus, we have

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^2\|\boldsymbol{\beta}^{(t-2)} - \boldsymbol{\beta}^*\| + 1.62^2\rho\eta\sqrt{s'}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty + 1.62\eta\sqrt{s}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty.$$

Because we have $p \geq s' \geq s$, and $1.62\rho < 1$, we get

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^2\|\boldsymbol{\beta}^{(t-2)} - \boldsymbol{\beta}^*\| + (1.62\rho + 1)1.62\eta\sqrt{p}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty.$$

Applying the same idea repeatedly all the way to $t = 0$ we get

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^t\|\boldsymbol{\beta}^{(0)} - \boldsymbol{\beta}^*\| + [(1.62\rho)^{t-1} + ... + 1]1.62\eta\sqrt{p}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty.$$

And because $\nabla f(\boldsymbol{\beta}^*) = \frac{1}{n}\mathbf{X}^T(\mathbf{y} - \mathbf{X}^T\boldsymbol{\beta}^*) = \frac{\mathbf{X}^T\boldsymbol{\epsilon}}{n}$, then we can use the inequality proved in the proof of Proposition 2

$$\mathbb{P}(\|\frac{\mathbf{X}^T\boldsymbol{\epsilon}}{n}\|_\infty \leq 2\sigma\sqrt{\frac{\log(p)}{n}}) \geq 1 - 2p^{1-C_1}.$$

So, with the probability $1 - 2p^{1-C_1}$, we have

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^t\|\boldsymbol{\beta}^*\| + \frac{3.24\sigma\eta}{1-1.62\rho}\sqrt{\frac{p\log(p)}{n}}.$$

□

Please note that the dimension of the vector $\boldsymbol{\beta}^{(t)}$ will reduce from p to k^* with the iteration t, thus we follow the Proposition 4 recursively with varying $k \geq k^*$. Here, we assume that $\|\boldsymbol{\beta}^{(t)}\|_0 = k^*$, and the OFSA algorithm stops at the time period t. Then we will show that the OFSA algorithm can recover the support of true features with high probability.

Corollary 3. (True feature recovery for OFSA) *With the same conditions and assumptions as Theorem 3, and define the smallest true $\boldsymbol{\beta}^*$ as*

$$\beta_{\min} := \min_{j \in S_{\boldsymbol{\beta}^*}} |\beta_j^*| > \frac{4\sigma\eta}{1-1.62\rho}\sqrt{\frac{p\log(p)}{n}}.$$

Then after $t = [\frac{1}{1.62\rho}\log(\frac{10\|\boldsymbol{\beta}^*\|}{\beta_{\min}})] + 1$ iterations, the OFSA algorithm will output $\boldsymbol{\beta}^{(t)}$ satisfying $S_{\boldsymbol{\beta}^*} = S_{\boldsymbol{\beta}^{(t)}}$ with probability $1 - 2p^{1-C_1}$, where $C_1 > 1$ is a constant.

Proof. From Theorem 3 we have the following result

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^t\|\boldsymbol{\beta}^*\| + \frac{1.62\eta\sqrt{p}}{1-1.62\rho}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty.$$

And we know that $\nabla f(\boldsymbol{\beta}^*) = \frac{1}{n}\mathbf{X}^T(\mathbf{y} - \mathbf{X}^T\boldsymbol{\beta}^*) = \frac{\mathbf{X}^T\boldsymbol{\epsilon}}{n}$. Then by using the inequality proved in the proof of Proposition 2

$$\mathbb{P}(\|\frac{\mathbf{X}^T\boldsymbol{\epsilon}}{n}\|_\infty \leq 2\sigma\sqrt{\frac{\log(p)}{n}}) \geq 1 - 2p^{1-C_1},$$

and then with probability $1 - 2p^{1-C_1}$, we have

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^t\|\boldsymbol{\beta}^*\| + \frac{3.24\sigma\eta\sqrt{p}}{1-1.62\rho}\sqrt{\frac{\log(p)}{n}}.$$

After $t = [\frac{1}{1.62\rho}\log(\frac{10\|\boldsymbol{\beta}^*\|}{\beta_{\min}})] + 1$ iterations, we can show that $(1.62\rho)^t\|\boldsymbol{\beta}^*\| < \frac{1}{10}\beta_{\min}$. Thus, with probability $1 - 2p^{1-C_1}$, we have

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| < \beta_{\min}.$$

And the conclusion that $S_{\boldsymbol{\beta}^*} = S_{\boldsymbol{\beta}^{(t)}}$ must hold. □

Finally, we consider the upper bound of the regret for the OLS and OLSth algorithms. In fact, all the feature selection algorithms we mentioned will degenerate to OLS if the true features are selected. First, we define the regret for a sparse model with sparsity levels $\|\boldsymbol{\beta}\|_0 \leq k^*$:

$$R_n = \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{\beta}_i; \mathbf{z}_i) - \min_{\boldsymbol{\beta}, \|\boldsymbol{\beta}\|_0 \leq k^*} \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{\beta}; \mathbf{z}_i), \tag{3.16}$$

in which $\boldsymbol{\beta}_i$ is the sparse coefficient vector at step i and $\mathbf{z}_i = (\mathbf{x}_i, y_i)$.

Observe that for $\forall i > 0$, the loss functions f from (3.16) are twice continuously differentiable. We denote $\boldsymbol{\beta}_{n+1} = \arg\min_{\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{\beta})$ and $(\mathbf{X}^T\mathbf{X})_n = \sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i^T$. Then, we will need the following assumptions:

Assumption 1. *Given $n > p$, then we have $0 < m < M$ satisfy*

$$0 < m < \lambda_{\min}(\frac{1}{n}(\mathbf{X}^T\mathbf{X})_n) < \lambda_{\max}(\frac{1}{n}(\mathbf{X}^T\mathbf{X})_n) < M.$$

Assumption 2. *Given $n > p$, there exist constants D and G such that $\|\boldsymbol{\beta}_i - \boldsymbol{\beta}_j\| < D, \forall i, j > n$ and $\|\nabla f(\boldsymbol{\beta}_i)\| \leq G, \forall i \geq n$.*

Proposition 5. *Given $n > p$, under Assumptions 1, 2, the regret of OLS satisfies:*

$$R_n = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_i)^2 - \min_{\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\boldsymbol{\beta})^2 \leq \mathcal{O}(\frac{\log(n)}{n}).$$

Proof. According to our OLS algorithm, we have the following equations:

$$\sum_{i=1}^{n-1} \mathbf{x}_i\mathbf{x}_i^T\boldsymbol{\beta}_n = \sum_{i=1}^{n-1} y_i\mathbf{x}_i, \tag{3.17}$$

$$\sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i^T\boldsymbol{\beta}_{n+1} = \sum_{i=1}^{n} y_i\mathbf{x}_i, \tag{3.18}$$

Here we have $\boldsymbol{\beta}_1 = 0$.

Add $\mathbf{x}_n\mathbf{x}_n^T\boldsymbol{\beta}_n$ to both sides of (3.17), obtaining

$$(\mathbf{X}^T\mathbf{X})_n\boldsymbol{\beta}_n = \sum_{i=1}^{n-1} y_i\mathbf{x}_i + \mathbf{x}_n\mathbf{x}_n^T\boldsymbol{\beta}_n. \tag{3.19}$$

where again we denoted by $(\mathbf{X}^T\mathbf{X})_n = \sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i^T$. Subtracting (3.19) from (3.18) we obtain:

$$(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_{n+1} - \boldsymbol{\beta}_n) = y_n\mathbf{x}_n - \mathbf{x}_n\mathbf{x}_n^T\boldsymbol{\beta}_n = -\nabla f_n(\boldsymbol{\beta}_n),$$

in which we denote $f_n(\boldsymbol{\beta}_n) = \frac{1}{2}(y_n - \mathbf{x}_n^T\boldsymbol{\beta}_n)^2$. Hence we have the iterative formula in n-th ($n \geq n_0$) iteration:

$$\boldsymbol{\beta}_{n+1} = \boldsymbol{\beta}_n - (\mathbf{X}^T\mathbf{X})_n^{-1}\nabla f_n(\boldsymbol{\beta}_n). \tag{3.20}$$

For $\forall \bar{\boldsymbol{\beta}} \in \mathbb{R}^p$, we have:

$$\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}} = \boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}} - (\mathbf{X}^T\mathbf{X})_n^{-1}\nabla f_n(\boldsymbol{\beta}_n),$$

thus we have the following equation:

$$(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}}) = (\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}}) - \nabla f_n(\boldsymbol{\beta}_n) \tag{3.21}$$

Multiplying by the transpose of $\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}}$ on both sides of (3.21) we get

$$(\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}}) = (\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}}) - (\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}})^T\nabla f_n(\boldsymbol{\beta}_n) \tag{3.22}$$

We plug (3.20) into (3.22):

$$\begin{aligned}
&(\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}}) \\
&= (\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}} - (\mathbf{X}^T\mathbf{X})_n^{-1}\nabla f_n(\boldsymbol{\beta}_n))^T(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}}) - (\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}} - (\mathbf{X}^T\mathbf{X})_n^{-1}\nabla f_n(\boldsymbol{\beta}_n))^T\nabla f_n(\boldsymbol{\beta}_n) \\
&= (\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}}) - 2\nabla f_n(\boldsymbol{\beta}_n)^T(\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}}) + \nabla f_n(\boldsymbol{\beta}_n)^T(\mathbf{X}^T\mathbf{X})_n^{-1}\nabla f_n(\boldsymbol{\beta}_n)
\end{aligned}$$

Rearranging terms, we have for $\forall n \geq n_0$:

$$\begin{aligned}
2\nabla f_n(\boldsymbol{\beta}_n)^T(\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}}) = (\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_n - \bar{\boldsymbol{\beta}}) - \\
(\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}}) + \nabla f_n(\boldsymbol{\beta}_n)^T(\mathbf{X}^T\mathbf{X})_n^{-1}\nabla f_n(\boldsymbol{\beta}_n)
\end{aligned} \tag{3.23}$$

For $\forall n > n_0$, we sum equation (3.23) from $(n_0 + 1)$ to n on both sides,

$$\begin{aligned}
2\sum_{i=n_0+1}^{n} \nabla f_i(\boldsymbol{\beta}_i)^T(\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}}) = &\sum_{i=n_0+1}^{n} (\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_i(\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}}) - \sum_{i=n_0+1}^{n} (\boldsymbol{\beta}_{i+1} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_i(\boldsymbol{\beta}_{i+1} - \bar{\boldsymbol{\beta}}) \\
&+ \sum_{i=n_0+1}^{n} \nabla f_i(\boldsymbol{\beta}_i)^T(\mathbf{X}^T\mathbf{X})_i^{-1}\nabla f_i(\boldsymbol{\beta}_i)
\end{aligned}$$

After rearranging the formula, we get

$$
\begin{aligned}
& 2 \sum_{i=n_0+1}^{n} \nabla f_i(\boldsymbol{\beta}_i)^T(\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}}) \\
= & \sum_{i=n_0+1}^{n} \nabla f_i(\boldsymbol{\beta}_i)^T(\mathbf{X}^T\mathbf{X})_i^{-1}\nabla f_i(\boldsymbol{\beta}_i) + (\boldsymbol{\beta}_{n_0+1} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_{n_0+1}(\boldsymbol{\beta}_{n_0+1} - \bar{\boldsymbol{\beta}}) \\
+ & \sum_{i=n_0+2}^{n} (\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}})^T((\mathbf{X}^T\mathbf{X})_i - (\mathbf{X}^T\mathbf{X})_{i-1})(\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}}) - (\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}}) \\
\leq & \sum_{i=n_0+1}^{n} \nabla f_i(\boldsymbol{\beta}_i)^T(\mathbf{X}^T\mathbf{X})_i^{-1}\nabla f_i(\boldsymbol{\beta}_i) + (\boldsymbol{\beta}_{n_0+1} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_{n_0+1}(\boldsymbol{\beta}_{n_0+1} - \bar{\boldsymbol{\beta}}) \\
+ & \sum_{i=n_0+2}^{n} (\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}})^T((\mathbf{X}^T\mathbf{X})_i - (\mathbf{X}^T\mathbf{X})_{i-1})(\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}}) \\
= & \sum_{i=n_0+1}^{n} \nabla f_i(\boldsymbol{\beta}_i)^T(\mathbf{X}^T\mathbf{X})_i^{-1}\nabla f_i(\boldsymbol{\beta}_i) + (\boldsymbol{\beta}_{n_0+1} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_{n_0+1}(\boldsymbol{\beta}_{n_0+1} - \bar{\boldsymbol{\beta}}) \\
+ & \sum_{i=n_0+2}^{n} (\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}})^T(\mathbf{x}_i\mathbf{x}_i^T)(\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}})
\end{aligned}
$$

The inequality holds because $(\mathbf{X}^T\mathbf{X})_n$ is positive definite, hence we have:

$$(\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_n(\boldsymbol{\beta}_{n+1} - \bar{\boldsymbol{\beta}}) \geq 0.$$

Then we denote

$$Q_i = \nabla f_i(\boldsymbol{\beta}_i)^T(\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}}) - \frac{1}{2}(\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}})^T(\mathbf{x}_i\mathbf{x}_i^T)(\boldsymbol{\beta}_i - \bar{\boldsymbol{\beta}})$$

By rearranging the formula and taking $\bar{\boldsymbol{\beta}} = \boldsymbol{\beta}_{n+1}$, we get

$$
\begin{aligned}
\sum_{i=n_0+1}^{n} Q_i &\leq \frac{1}{2}\sum_{i=n_0+1}^{n} \nabla f_i(\boldsymbol{\beta}_i)^T(\mathbf{X}^T\mathbf{X})_i^{-1}\nabla f_i(\boldsymbol{\beta}_i) + \frac{1}{2}(\boldsymbol{\beta}_{n_0} - \bar{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})_{n_0}(\boldsymbol{\beta}_{n_0} - \bar{\boldsymbol{\beta}}) \\
&\leq \frac{1}{2}\sum_{i=n_0+1}^{n} \nabla f_i(\boldsymbol{\beta}_i)^T(\mathbf{X}^T\mathbf{X})_i^{-1}\nabla f_i(\boldsymbol{\beta}_i) + \frac{1}{2}\lambda_{\max}(\mathbf{X}^T\mathbf{X})_{n_0}\|\boldsymbol{\beta}_{n_0} - \bar{\boldsymbol{\beta}}\|_2^2 \\
&\leq \frac{1}{2}\sum_{i=n_0+1}^{n} \nabla f_i(\boldsymbol{\beta}_i)^T(\mathbf{X}^T\mathbf{X})_i^{-1}\nabla f_i(\boldsymbol{\beta}_i) + \frac{1}{2}Mn_0D^2
\end{aligned}
$$

where in the last inequality we used Assumptions 1 and 2. Because $f_i(\boldsymbol{\beta}) = \frac{1}{2}(y_i - \mathbf{x}_i^T\boldsymbol{\beta})^2$ is second order differentiable, according to its Taylor expression, we have

$$
\begin{aligned}
f_i(\bar{\boldsymbol{\beta}}) &= f_i(\boldsymbol{\beta}_i) + \nabla f_i(\boldsymbol{\beta}_i)^T(\bar{\boldsymbol{\beta}} - \boldsymbol{\beta}_i) + \frac{1}{2}(\bar{\boldsymbol{\beta}} - \boldsymbol{\beta}_i)^T\nabla^2 f_i(\zeta)(\bar{\boldsymbol{\beta}} - \boldsymbol{\beta}_i) \\
&= f_i(\boldsymbol{\beta}_i) + \nabla f_i(\boldsymbol{\beta}_i)^T(\bar{\boldsymbol{\beta}} - \boldsymbol{\beta}_i) + \frac{1}{2}(\bar{\boldsymbol{\beta}} - \boldsymbol{\beta}_i)^T\mathbf{x}_i\mathbf{x}_i^T(\bar{\boldsymbol{\beta}} - \boldsymbol{\beta}_i)
\end{aligned}
$$

Thus we have $\sum_{i=n_0+1}^{n}(f_i(\boldsymbol{\beta}_i) - f_i(\bar{\boldsymbol{\beta}})) = \sum_{i=n_0+1}^{n} Q_i$ and we get

$$\sum_{i=n_0+1}^{n}(f_i(\boldsymbol{\beta}_i) - f_i(\bar{\boldsymbol{\beta}})) \leq \frac{1}{2}\sum_{i=n_0+1}^{n}\nabla f_i(\boldsymbol{\beta}_i)^T(\mathbf{X}^T\mathbf{X})_i^{-1}\nabla f_i(\boldsymbol{\beta}_i) + Mn_0D^2$$

Based on Assumptions 1 and 2, we have $0 < m < \lambda_{\min}(\frac{1}{i}(\mathbf{X}^T\mathbf{X})_i)$ and $\|\nabla f_i(\boldsymbol{\beta}_i)\|_2 \leq G$, so we have

$$\frac{1}{2}\sum_{i=n_0+1}^{n}\nabla f_i(\boldsymbol{\beta}_i)^T((\mathbf{X}^T\mathbf{X})_i)^{-1}\nabla f_i(\boldsymbol{\beta}_i) \leq \sum_{i=n_0+1}^{n}\frac{1}{2mi}G^2 = \frac{G^2}{2m}\sum_{i=n_0+1}^{n}\frac{1}{i} \leq \frac{G^2}{2m}\log(n)$$

So we get

$$\frac{1}{n}\sum_{i=n_0+1}^{n}(f_i(\boldsymbol{\beta}_i) - f_i(\bar{\boldsymbol{\beta}})) \leq \frac{G^2}{2m}\frac{\log(n)}{n} + \frac{Mn_0D^2}{2n}$$

Then, we consider the regret bound for the first n_0 observations.

For $i = 1$ to n_0, we use online ridge regression to replace online least squares, considering the equations for any $\lambda > 0$:

$$\sum_{i=1}^{n-1}\mathbf{x}_i\mathbf{x}_i^T\boldsymbol{\beta}_n + \lambda\boldsymbol{\beta}_n = \sum_{i=1}^{n-1}y_i\mathbf{x}_i, \tag{3.24}$$

$$\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T\boldsymbol{\beta}_{n+1} + \lambda\boldsymbol{\beta}_{n+1} = \sum_{i=1}^{n}y_i\mathbf{x}_i, \tag{3.25}$$

Add $\mathbf{x}_n\mathbf{x}_n^T\boldsymbol{\beta}_n$ to both sides of (3.24), we obtain

$$\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T\boldsymbol{\beta}_n + \lambda\boldsymbol{\beta}_n = \sum_{i=1}^{n-1}y_i\mathbf{x}_i + \mathbf{x}_n\mathbf{x}_n^T\boldsymbol{\beta}_n. \tag{3.26}$$

Then we use (3.25) substract (3.26), and we have:

$$\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T(\boldsymbol{\beta}_{n+1} - \boldsymbol{\beta}_n) + \lambda(\boldsymbol{\beta}_{n+1} - \boldsymbol{\beta}_n) = y_n\mathbf{x}_n - \mathbf{x}_n\mathbf{x}_n^T\boldsymbol{\beta}_n \tag{3.27}$$

Thus, we can get

$$\boldsymbol{\beta}_{n+1} = \boldsymbol{\beta}_n - (\sum_{i=1}^{n}\mathbf{x}_n\mathbf{x}_n^T + \lambda\mathbf{I})^{-1}\nabla f_n(\boldsymbol{\beta}_n)$$

Because it is easy to find real values m_0 and M_0 satisfy

$$0 < m_0 < \lambda_{\min}(\sum_{i=1}^{n}\mathbf{x}_n\mathbf{x}_n^T + \lambda\mathbf{I}) < \lambda_{\max}(\sum_{i=1}^{n}\mathbf{x}_n\mathbf{x}_n^T + \lambda\mathbf{I}) < M_0 < \infty,$$

using the similar technique by replacing $\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T$ to $\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T + \lambda\mathbf{I}$ and considering that n_0 is a given scalar, we obtain that there is a constant $C > 0$ satisfying

$$\sum_{i=1}^{n_0}(f_i(\boldsymbol{\beta}_i) - f_i(\bar{\boldsymbol{\beta}})) \leq Cn_0$$

Consequently,

$$\frac{1}{n}\sum_{i=1}^{n}(f_i(\boldsymbol{\beta}_i) - f_i(\bar{\boldsymbol{\beta}})) \leq \frac{Cn_0}{n} + \frac{G^2}{2m}\frac{\log(n)}{n} + \frac{MD^2n_0}{2n} = \mathcal{O}(\frac{\log(n)}{n})$$

□

Theorem 4. (Regret of OLS-th) *With the Assumptions 1, 2 holding for* $\mathbf{X}_{S_{\beta^*}}$*, there exists a constant* $C_1 > 1$ *such that if the true* $\boldsymbol{\beta}^*$ *satisfies*

$$\min_{j\in S_{\boldsymbol{\beta}^*}} |\beta_j^*| > \frac{2\sigma}{\lambda}\sqrt{\frac{\log(p)}{\sqrt{n}}}, \text{ for some } \lambda > 0 \text{ satisfying } \sqrt{\lambda} < 0.8\lambda_{\min}(\sqrt{\boldsymbol{\Sigma}}) - \sqrt{\frac{p}{n_0}}. \tag{3.28}$$

where $n_0 = \max(p+1, 100\log(n), \frac{1}{C_1^2}\left(\frac{2\log(n)}{\log(p)}+1\right)^2) > p$*, then with probability at least* $1-3/n$ *the regret of OLSth satisfies:*

$$R_n = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_i)^2 - \min_{\|\boldsymbol{\beta}\|_0\leq k}\frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\boldsymbol{\beta})^2 \leq \mathcal{O}(\frac{\log^2(n)}{n}).$$

Proof. According to Theorem 2 with $\alpha = 1/2$, the probability for true feature selection for sample size $i \geq n_0$ is greater than

$$1 - 2p^{1-C_1\sqrt{i}} - e^{-i/50} \geq 1 - 2p^{1-C_1\sqrt{n_0}} - e^{-n_0/50}.$$

Using the union bound we get that the true features are selected for all sample sizes i, with $n_0 \leq i \leq n$ with probability at least

$$\begin{aligned} 1 - 2np^{1-C_1\sqrt{n_0}} - ne^{-n_0/50} &\leq 1 - 2npe^{-C_1\sqrt{n_0}\log(p)} - ne^{-2\log(n)} \\ &\leq 1 - 2/n - n^{-1} = 1 - 3/n \end{aligned}$$

where we used that $(\sqrt{n_0}C_1 - 1)\log(p) > 2\log(n)$ and $n_0 \geq 100\log(n)$.

Thus with this probability the true features are selected for all sample sizes i with $n \geq i \geq n_0$, and the OLSth algorithm degenerates to the OLS algorithm on the features from S_{β^*}. Assumption 1 is

$$0 < m < \lambda_{\min}(\frac{1}{n}(\mathbf{X}^T\mathbf{X})_{S_{\boldsymbol{\beta}^*}}) < \lambda_{\max}(\frac{1}{n}(\mathbf{X}^T\mathbf{X})_{S_{\boldsymbol{\beta}^*}}) < M.$$

Following the proof of Proposition 5, we obtain

$$\frac{1}{n}\sum_{i=1}^{n}(f_i(\boldsymbol{\beta}_i) - f_i(\bar{\boldsymbol{\beta}})) \leq \frac{Cn_0}{n} + \frac{G^2}{2m}\frac{\log(n)}{n} + \frac{Mn_0D^2}{2n}. \tag{3.29}$$

From (3.29) since $n_0 = \mathcal{O}(\log^2(n))$ we have

$$\frac{1}{n}\sum_{i=1}^{n}(f_i(\boldsymbol{\beta}_i) - f_i(\bar{\boldsymbol{\beta}})) \leq \mathcal{O}(\frac{\log^2(n)}{n})$$

Thus we get the conclusion. □

Theoretical guarantees for feature selection in classification. Proposition 2.3.6 and Remark 2.3.7 from [25] show that the least squares Lasso algorithm (therefore the Online Lasso) can recover the support of true variables for the discrete Y under some assumptions.

Theorem 5. (True support recovery) *Consider the special case of a single index model,* $\mathbf{y} = G\{h(\mathbf{X}\boldsymbol{\beta}^*) + \boldsymbol{\epsilon}\}$*, in which* $\mathbf{X} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ *and* $\boldsymbol{\Sigma}$ *satisfies the irrepresentable condition. If* G*,* h *are known strictly increasing continuous functions and under the assumptions from [25], the least squares Lasso algorithm can recover the support of true features correctly for discrete response* $\mathbf{y}$*.*

The proof and more mathematical details can be found in [25]. Based on Theorem 5, we have theoretical guarantees for support recovery for some of our running averages-based online classification methods.

CHAPTER 4

EXPERIMENTS WITH STOCHASTIC FEATURE SELECTION WITH ANNEALING

In this chapter, we conduct numerical experiments and real data analysis to evaluate the performance of the proposed stochastic feature selection with annealing algorithm from Chapter 2. First, we present experiments on large sparse simulated datasets to compare the performance of SFSA with other state-of-the-art methods in the literature for the linear regression and classification. Then we conduct experiments on large sparse real datasets to compare the performance of various online feature selection algorithms. All experiments are replicated 20 times on a desktop computer with Core i5 - 4460S CPU and 16Gb memory. And we present the averages of the experiments times.

4.1 Experiments for Simulated Data

In this experiment we use uniformly correlated data generated as follows: given a scalar α, we generate $z_i \sim \mathcal{N}(0,1)$, then we generate an observation $\mathbf{x}_i$:

$$\mathbf{x}_i = \alpha z_i \mathbf{1}_{p\times 1} + \mathbf{u}_i, \text{ with } \mathbf{u}_i \sim \mathcal{N}(0, \mathbf{I}_p).$$

We generate i.i.d. observations this way to obtain the data matrix $\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_n^T]^T$. It is easy to verify that the correlation between any pair of predictors is $\alpha^2/(1+\alpha^2)$. We set $\alpha = 1$ in our simulation, thus the correlation between any two features is 0.5. Then, the dependent response $\mathbf{y}$ is generated from the following linear regression model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\eta}, \text{ with } \boldsymbol{\eta} \sim \mathcal{N}(0, \mathbf{I}_n), \tag{4.1}$$

and for classification:

$$\mathbf{y} = \text{sign}(\mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\eta}), \text{ with } \boldsymbol{\eta} \sim \mathcal{N}(0, \mathbf{I}_n), \tag{4.2}$$

where $\boldsymbol{\beta}^*$ is a p-dimensional sparse parameter vector. The true coefficients are $\beta_j^* = 0$ except $\beta_{10j^*}^* \neq 0$, $j^* = 1, 2, \cdots, k$. Please note that the classification data cannot be perfectly separated by a linear model. The simulation is based on the following data parameter setting: $p = 10,000$

Table 4.1: Simulation experiments for online regression, averaged 20 runs.

	Variable Detection Rate DR (%)					RMSE					Time(s)				
	SFSA	SFSA-AG	TSGD	OPG	RDA	SFSA	SFSA-AG	TSGD	OPG	RDA	SFSA	SFSA-AG	TSGD	OPG	RDA
$p = 10000, k = 100$, strong signal $\beta = 1$, mini-batch = 25															
5×10^3	56.75	43.80	**98.05**	1.30	0.80	**60.07**	62.61	97.14	100.0	100.1	0.065	0.066	0.034	250.4	464.1
10^4	84.30	73.10	**100**	1.20	1.25	**48.74**	50.90	96.05	100.9	100.9	0.124	0.132	0.067	500.7	917.2
2×10^4	**100**	**100**	**100**	1.30	1.10	**39.93**	41.54	91.80	100.3	100.4	0.219	0.238	0.132	751.6	1822
$p = 10000, k = 100$, weak signal β increase from 0.05 to 1, mini-batch = 25															
10^4	67.95	59.80	**84.55**	1.05	0.95	**25.68**	26.81	50.44	53.06	53.07	0.124	0.124	0.069	504.8	603.5
3×10^4	91.35	90.60	**93.40**	1.05	1.40	**16.37**	16.95	46.54	52.99	53.00	0.343	0.339	0.198	1260	1774
10^5	**98.30**	98.20	98.25	0.95	1.30	**7.89**	8.098	34.84	52.44	52.46	1.002	1.002	0.663	3026	5758
3×10^5	**100**	**100**	**100**	0.85	1.35	**2.06**	2.602	2.475	52.83	52.86	3.002	2.970	1.759	9085	17057

and $k^* = 100$. We consider the signal strength $\beta^*_{10j^*} = 1$ for strong signal and $\beta^*_{10j^*} \in [0.05, 1]$ increase linearly from 0.05 to 1 for weak signal. The sample size n varies from 5×10^3 to 3×10^5.

For both of regression and classification experiments, we cover two classical online feature selection methods for comparison: the OPG [5] and RDA [44] methods. In both frameworks, we consider ℓ_1 regularization for regression and $\ell_1+\ell_2$ regularization for classification. For classification, apart from the OPG and RDA, our simulation include the first order online feature selection (FOFS) and the second order online feature selection (SOFS) methods [40, 43].

In the simulation, the sparsity controlling parameters are tuned to obtain $k = k^*$ variables. This can be done directly for SFSA, FOFS and SOFS methods, and indirectly through the regularization parameter for the OPG and RDA methods. In OPG and RDA, we use 200 values of λ on an exponential grid and choose the λ that induces the $\hat{k}$ non-zero features, where $\hat{k}$ is the largest number of non-zeros features smaller than or equal to k^*, the number of true features. The following criteria are used in the numerical experiments: the true variable detection rate (DR), the root mean square error (RMSE) on the test data for regression, the area under ROC curve (AUC) on the test data for classification, and the running time (Time) of the algorithms. The variable detection rate DR is defined as the average number of true variables that are correctly detected by an algorithm is divided by the number of true variables. So when $S_{\boldsymbol{\beta}}$ is the set of selected variables and $S_{\boldsymbol{\beta}^*}$ is the set of the true variables, then we have

$$DR = \frac{E(|S_{\boldsymbol{\beta}} \cap S_{\boldsymbol{\beta}^*}|)}{|S_{\boldsymbol{\beta}^*}|}.$$

Table 4.2: Comparison between SFSA, SFSA-AG(AG), SFSA-Adam(Adam), TSGD and other online algorithms for classification, averaged 20 runs.

	Variable Detection Rate DR(%)								AUC							
	SFSA	AG	Adam	TSGD	FOFS	SOFS	OPG	RDA	SFSA	AG	Adam	TSGD	FOFS	SOFS	OPG	RDA
$p = 10000, k = 100$, strong signal $\beta = 1$, mini-batch = 25																
10^4	40.20	40.15	40.45	**49.25**	0.7	1.0	1.05	1.50	0.996	0.996	**0.997**	**0.997**	0.994	0.994	0.986	0.991
3×10^4	91.85	**93.40**	92.30	92.40	0.7	1.2	0.75	1.45	0.999	**1.0**	**1.0**	**1.0**	0.994	0.993	0.986	0.990
10^5	**100**	**100**	**100**	**100**	0.7	1.0	0.75	1.80	**1.0**	**1.0**	**1.0**	**1.0**	0.993	0.993	0.954	0.989
$p = 10000, k = 100$, weak signal β increase from 0.05 to 1, mini-batch = 25																
10^4	39.30	38.95	40.50	**46.25**	0.8	1.0	1.40	1.45	**0.997**	0.996	**0.997**	**0.997**	0.993	0.993	0.981	0.990
3×10^4	67.80	67.85	**68.50**	68.45	0.8	1.2	0.55	1.60	**0.999**	**0.999**	**0.999**	**0.999**	0.992	0.992	0.985	0.989
10^5	**86.10**	85.94	84.35	84.25	0.8	1.0	0.85	1.70	**1.0**	**1.0**	**1.0**	**1.0**	0.992	0.992	0.925	0.988
3×10^5	93.55	**93.60**	89.10	90.95	0.8	0.9	1.10	3.65	**1.0**	**1.0**	**1.0**	**1.0**	0.993	0.993	0.981	0.986

4.1.1 Experimental Results for Regression

In this subsection, we introduce the empirical performance of SFSA for the regression task. The performance of various algorithms is shown in Table 4.1. In terms of detection rate (DR), SFSA, SFSA-AG and TSGD algorithms are much better than RDA and OPG. When the sample size n increases, our proposed methods can select all true features. Also, our algorithms have a less computation time because they can directly control the desired sparsity level. In contrast, the OPG and RDA methods cannot recover the support of the true features, and because of the need to vary the regularization parameter in order to control the sparsity level, these algorithms are computationally expensive. Considering the test RMSE, we can observe that if the sample size n is large, the RMSE will converge for SFSA, SFSA-AG and TSGD methods.

4.1.2 Experimental Results for Classification

Similarly, we evaluate the empirical performance of SFSA and the variants for the classification task in this subsection. The experimental results are shown in Table 4.2 and Table 4.3.

First, we analyze the variable detection rate. In the binary classification, it is more challenging to select the true features than in regression. In order to recover all true features, we need more training instances in classification than regression. Then, for weaker signal strength, given the largest sample size $n = 3 \times 10^5$, our SFSA algorithms and the variants cannot select all true features, even though their performance is much better than the regularized based methods and standard online feature selection methods, FOFS and SOFS. Similar to the regression problem, the regularized based methods OPG and RDA cannot recover the support of true features for the

Table 4.3: The running time for SFSA, SFSA-AG(AG), SFSA-Adam(Adam), TSGD and other online methods.

	Time(s)							
	SFSA	AG	Adam	TSGD	FOFS	SOFS	OPG	RDA
$p = 10000, k = 100$, strong signal $\beta = 1$								
10^4	0.163	0.171	0.201	0.075	0.396	0.029	505.8	1123
3×10^4	0.441	0.427	0.525	0.218	1.131	0.085	1521	3333
10^5	1.251	1.368	1.706	0.730	3.685	0.284	5063	10994
$p = 10000, k = 100$, weak signal β increase from 0.05 to 1								
10^4	0.162	0.171	0.196	0.075	0.396	0.028	504.9	1121
3×10^4	0.439	0.470	0.527	0.222	1.142	0.085	1514	3344
10^5	1.261	1.377	1.706	0.747	3.735	0.284	5051	11012
3×10^5	3.773	4.104	5.107	2.210	11.129	0.851	15065	32699

dataset with strong correlated features. As for the state-of-the-art algorithms FOFS and SFOS [40] and [43] from the literature, they cannot detect the true features either.

Then we consider the test AUC (Area under the ROC curve). Because our SFSA algorithms and the variants cover most of true features in the processing of modeling, it is easy to see that the results of AUC for SFSA and the variants are better. However, while the variable detection rate results show a significant difference between the family of SFSA and the others, the results of AUC for the other methods are very close to the SFSA variants. It looks like that building a good prediction model does not depend on true feature recovery.

The analysis of time complexity is the same as in regression. Because of the parameter tuning problem, regularized based methods are computationally expensive. By contrast, greedy-based methods have huge advantage in computational time.

4.1.3 Experiments on Large Sparse Datasets

In order to verify that we implemented the FOFS, SOFS, OPG and RDA algorithms correctly, we also performed similar numerical experiments on large sparse data to the ones described in [43].

First, we introduce the data generation. We generate three large sparse datasets $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$, with each observation being a sparse vector with 200, 400, and respectively 500 nonzero entries at random locations. Each nonzero entry is generated from the i.i.d Gaussian distribution $\mathcal{N}(0, 1)$. The label $\mathbf{y} \in \{-1, +1\}$ is generated from the following noiseless linear model:

$$\mathbf{y} = \text{sign}(\mathbf{X}\boldsymbol{\beta}^*),$$

Table 4.4: Comparison among SFSA, SFSA-AG, TSGD, FOFS, and SOFS for the simulated sparse dataset

					Variable Detection Rate DR (%)							AUC						
Dataset	n	p	k	noise	SFSA	SFSA-AG	TSGD	FOFS	SOFS	OPG	RDA	SFSA	SFSA-AG	TSGD	FOFS	SOFS	OPG	RDA
$\mathbf{X}_1$	10^5	10^4	100	200	**100**	**100**	**100**	99.95	97	70.45	83.30	**0.923**	**0.923**	0.918	0.917	0.768	0.712	0.814
$\mathbf{X}_2$	10^5	2×10^4	200	400	**100**	**100**	**100**	99.70	97.15	67.75	86.88	**0.919**	**0.919**	0.913	0.910	0.758	0.671	0.818
$\mathbf{X}_3$	10^5	10^5	500	500	**100**	**100**	**100**	99.87	98.47	71.74	-	**0.916**	**0.916**	0.902	0.899	0.749	0.642	-

where the true parameter vector $\boldsymbol{\beta}^*$ is sparse with 100, 200, and 500 nonzero entries, respectively. The nonzero entries are sampled from the uniform distribution $\mathcal{U}(0, 1)$.

In this large sparse data simulation, we just evaluate the algorithms in terms of the true variable detection rate (DR) and the area under ROC curve (AUC) on test data. Based on the previous experiment, we remove the SFSA-Adam algorithm from the large sparse data simulation because it has a similar performance to SFSA-AG. The results of the simulation are shown in Table 4.4.

These experimental results verify that all the tested algorithms can detect the true features quite well if the features are independent. However, compared to the greedy algorithms, regularized based methods, OPG and RDA, still suffer lower variable detection rate and lower prediction accuracy. As to the greedy-based methods, our SFSA based methods and TSGD outperform the other methods in true feature recovery and parameter estimation. The experimental results for the large sparse datasets prove our proposed SFSA methods can handle various type of datasets and that our implementation of the competing algorithms has similar performance to that reported in [43].

4.2 Large Sparse Real Data Analysis

In this section, we apply our SFSA algorithms and the variants to three large real datasets.

The first dataset is the URL dataset [21], also analyzed using SOFS in [43]. The URL dataset is a large scale and high dimensional dataset about predicting whether a website is malicious or not based on a number of features. It has more than 3 million features and 2 million observations. However, because the URL instances are obtained day by day from from a large web server, it makes sense to apply online learning models to this dataset. Following [21], we used the data from day 0 to day 99 as the training data, and the data in day 100 to evaluate the performance of the models we trained. Hence we report the prediction accuracy for the data in day 100. We select about 0.5% features in this dataset.

Table 4.5: Comparison between SFSA, SFSA-AG, TSGD, FOFS, and SOFS for the real datasets.

			Prediction Accuracy (%)				
Dataset	n	p	SFSA	SFSAAG	TSGD	FOFS	SOFS
URL	2,396,130	3,231,961	**98.21**	98.13	97.38	97.27	95.39
Webspam	350,000	16,609,143	93.66	93.69	90.44	90.41	**96.63**
News	19,996	1,355,191	**69.04**	66.42	54.71	54.45	68.47

The second one is the web spam dataset [41], with 350,000 observations and 16,609,143 features. We use the first 200,000 observations for training, and the following 50,000 as the validation data. We train the online models by using this training data and predict the accuracy for validation data. The SFSA and SFSA-AG methods are trained using the hinge loss and TSGD is trained by the logistic loss. The FOFS method is also based on the hinge loss. We selected about 0.1% of features in the dataset.

The last one is the news dataset [16], with 19,996 observations and 1,355,191 features. Here, we random select 18,000 observations as the training observations, and the others are the validation data. We train the online models by using the training data and predict the accuracy for the validation data. And we select about 1% features in this dataset.

The results are shown in Table 4.5. On the URL dataset, we can observe that SFSA method based on hinge loss performs better than the other methods. Especially, based on the same loss function, we can find the SFSA is better than FOFS on all three of datasets. Besides, the SFSA-AG also performs very well in the real data analysis. On the web spam dataset, although the SOFS provide the best AUC result, SFSA and the variant still perform very well. On the news dataset, the SFSA algorithm shows the best result among the all of the methods.

CHAPTER 5

EXPERIMENTS WITH RUNNING AVERAGES ALGORITHMS

In this chapter, we evaluate the performance of our proposed online running averages algorithms introduced in Chapter 3, and compare them with offline learning methods and some standard stochastic algorithms. First, we present the results of numerical experiments on synthetic data, comparing the performance on feature selection and prediction. We also provide regret plots for the running averages based algorithms and compare them with classical online algorithms. Finally, we present an evaluation on real data. All simulation experiments are run on a desktop computer with Core i5 - 4460S CPU and 16Gb memory.

5.1 Experiments for Simulated Data

Here, we use the same method to generate the data as Section 4.1. Our simulation is based on the following data parameter setting: $p = 1000$ and $k = 100$. We consider the signal strength $\beta \in \{0.01, 1\}$ (weak and strong signals). The sample size n varies from 1000 to 10^6 for both regression and classification settings. For regression, we compare with our algorithms with SADMM [26] and the offline Lasso [38]. We also implement the following truncated stochastic gradient descent (TSGD) [8, 43]:

$$\tilde{\boldsymbol{\beta}}^{(n)} = \text{Truncate}\big(\boldsymbol{\beta}^{(n-1)} + \eta(y_n - \mathbf{x}_n^T\boldsymbol{\beta}^{(n-1)})\mathbf{x}_n, k\big),$$

where the operator "Truncate" keeps the k largest $|\tilde{\boldsymbol{\beta}}_j^{(n)}|$.

For classification, we cover four methods for comparison: the OPG [5] and RDA [44] frameworks for elastic net, the first order online feature selection (FOFS) method [43] and the second order online feature selection (SOFS) method [43].

For each method, the sparsity controlling parameters are tuned to obtain k variables. This can be done directly for OFSA and OLSth, and indirectly through the penalty parameter for the other methods. In RDA, OPG and SADMM, we use 200 values of λ on an exponential grid and chose the

Table 5.1: Comparison between running averages method and the other online and offline methods for regression, averaged 100 runs.

	Variable Detection Rate (%)							RMSE						
n	Lasso	TSGD	SADMM	OLSth	OFSA	OMCP	OElnet	Lasso	TSGD	SADMM	OLSth	OFSA	OMCP	OElnet
$p = 1000, k = 100$, strong signal $\beta = 1$														
10^3	32.14	11.22	18.10	77.40	**99.81**	73.71	32.12	11.63	23.15	95.05	5.592	**1.136**	6.282	11.61
$3 \cdot 10^3$	46.05	11.22	41.23	**100**	**100**	98.02	45.19	9.464	13.45	93.50	**1.017**	**1.017**	1.745	9.557
10^4	72.40	11.22	65.78	**100**	**100**	**100**	72.42	6.07	13.34	94.92	**1.003**	**1.003**	**1.003**	6.042
$p = 1000, k = 100$, weak signal $\beta = 0.01$														
10^3	14.09	10.89	13.53	10.11	12.40	**15.55**	14.08	1.128	**1.027**	1.363	1.069	1.169	1.049	1.124
10^4	31.58	10.89	19.80	22.48	**32.47**	32.32	31.54	1.009	1.007	1.370	1.025	1.006	**1.005**	1.006
10^5	81.93	10.89	11.30	80.55	**85.14**	84.86	81.80	**1.001**	1.010	1.382	1.003	1.003	1.003	1.003
$3 \cdot 10^5$	98.66	10.89	10.80	98.94	**99.27**	99.26	98.71	0.999	1.008	1.383	**0.998**	**0.998**	**0.998**	**0.998**
10^6	-	10.89	-	**100**	**100**	**100**	**100**	-	1.005	-	**0.996**	**0.996**	**0.996**	**0.996**

Table 5.2: Running time (s) for the different methods, averaged 100 runs.

$p = 1000, k = 100$, strong signal $\beta = 1$								
n	Lasso	TSGD	SADMM	OLSth	OFSA	OMCP	OElnet	RAVE
10^3	4.332	0.007	5.326	0.052	0.289	15.49	9.648	0.026
$3 \cdot 10^3$	26.91	0.019	15.73	0.051	0.288	13.86	7.113	0.076
10^4	47.32	0.065	51.80	0.051	0.288	6.508	5.885	0.246
$p = 1000, k = 100$, weak signal $\beta = 0.01$								
10^3	5.353	0.006	6.703	0.052	0.288	13.20	9.741	0.026
10^4	48.13	0.067	67.82	0.051	0.287	14.98	4.961	0.249
10^5	452.2	0.672	679.7	0.051	0.287	15.93	5.120	2.458
$3 \cdot 10^5$	1172	2.001	2044	0.051	0.287	13.96	3.749	7.326
10^6	-	6.651	-	0.051	0.288	7.352	1.726	24.36

λ that induces the $\hat{k}$ non-zero features, where $\hat{k}$ is the largest number of non-zeros features smaller than or equal to k, the number of true features.

The following criteria are used in the numerical experiments: the true variable detection rate (DR), the root of mean square error (RMSE) on the test data for regression, the area under ROC curve (AUC) on the test data in classification setting, and running time (Time) of the algorithms. The details of these criteria have been introduced in the Chapter 4.

The results are presented in Tables 5.1 and 5.3. We replicate the experiments 100 times and present the average results. Compared to the batch learning method Lasso, in regression, the running averages online methods enjoy lower memory complexity. Also, the larger datasets cannot fit in memory, hence we cannot obtain the experimental results for Lasso for the large datasets. In

our methods, we input the running averages rather than data matrix. The memory complexity for running averages is $\mathcal{O}(p^2)$, which is better than $\mathcal{O}(np)$ for batch learning in the setting of $n > p$.

Table 5.3: Comparison between running averages methods and the other online methods for classification, averaged 100 runs.

	Variable Detection Rate (%)								AUC							
	FOFS	SOFS	OPG	RDA	OFSA	OLSth	OLasso	OMCP	FOFS	SOFS	OPG	RDA	OFSA	OLSth	OLasso	OMCP
$p = 1000, k = 100$, strong signal $\beta = 1$																
10^4	10.64	10.19	10.46	10.97	38.89	30.30	34.70	**41.54**	0.995	0.992	0.992	0.990	0.995	0.990	**0.996**	**0.996**
3×10^4	10.64	9.95	10.42	10.34	**67.67**	59.32	56.18	67.52	0.994	0.992	0.992	0.989	**0.998**	0.996	0.997	**0.998**
10^5	10.64	9.95	10.43	11.08	**94.95**	93.21	86.90	94.77	0.994	0.992	0.992	0.990	**1.000**	**1.000**	0.999	**1.000**
$p = 1000, k = 100$, weak signal $\beta = 0.01$																
10^4	13.40	10.19	10.00	10.37	19.41	15.93	22.55	**23.81**	0.827	0.829	0.828	0.828	0.824	0.815	0.829	**0.830**
3×10^4	15.86	9.95	10.23	10.34	34.46	27.35	35.14	**37.70**	0.827	0.829	0.829	0.829	0.831	0.827	**0.832**	**0.832**
10^5	17.36	9.95	10.32	10.91	64.84	56.42	61.07	**64.95**	0.830	0.831	0.831	0.830	**0.834**	0.833	**0.834**	**0.834**
3×10^5	17.13	9.23	10.32	10.37	91.55	88.91	88.69	**91.58**	0.826	0.828	0.828	0.827	**0.833**	**0.833**	**0.833**	**0.833**
10^6	17.72	9.91	-	-	**99.97**	99.94	99.88	**99.97**	0.828	0.829	-	-	**0.834**	**0.834**	**0.834**	**0.834**

Table 5.4: Running time (s) for different methods, averaged 100 runs.

n	FOFS	SOFS	OPG	RDA	OFSA	OLSth	OLasso	OMCP	RAVE
$p = 1000, k = 100$, strong signal $\beta = 1$									
10^4	0.001	0.001	0.490	0.848	0.005	0.001	0.080	0.160	0.247
3×10^4	0.003	0.004	1.471	2.210	0.005	0.001	0.083	0.158	0.742
10^5	0.010	0.015	4.900	6.118	0.005	0.001	0.079	0.159	2.478
$p = 1000, k = 100$, strong signal $\beta = 0.01$									
10^4	0.001	0.001	0.494	0.815	0.005	0.001	0.073	0.148	0.249
3×10^4	0.003	0.004	1.481	2.093	0.005	0.001	0.074	0.152	0.743
10^5	0.010	0.015	4.935	5.827	0.005	0.001	0.078	0.161	2.472
3×10^5	0.030	0.044	14.81	17.31	0.005	0.001	0.073	0.164	7.446
10^6	0.100	0.146	-	-	0.005	0.001	0.039	0.110	24.85

From the numerical experiments, we can draw the conclusion that none of the online methods we tested (RDA, OPG, SADMM, FOFS and SOFS) performs well in true feature recovery. Only the offline Lasso and the proposed running averages based online methods can recover the true signal with high probability. When the signal is weak ($\beta = 0.01$), although the running averages methods need a large sample size n to recover the weak true signal, they outperform the batch learning methods and the other online methods in our experiment.

In prediction, most methods do well except in regression the existing methods (Lasso, TSGD and SADMM) do not work well when the signal is strong. In contrast, the proposed running

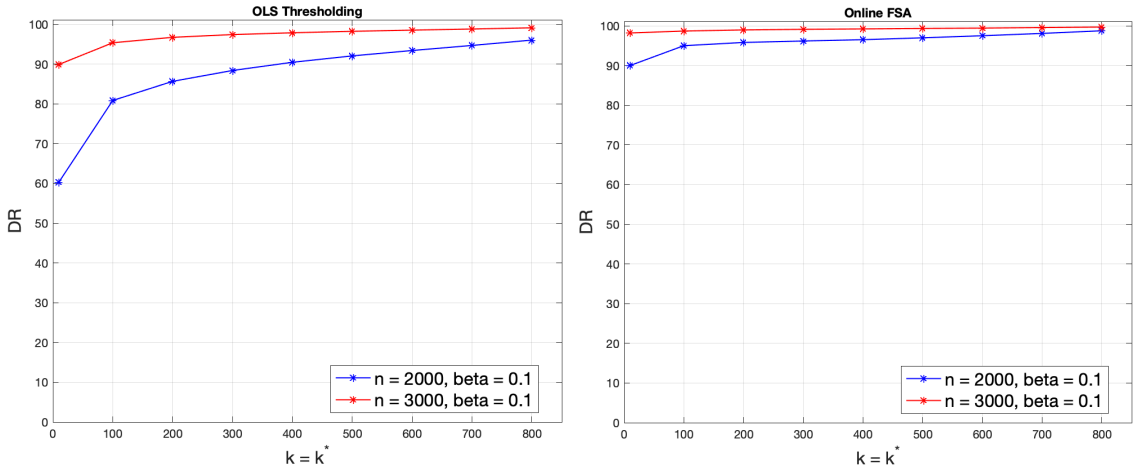

Figure 5.1: Variable detection rate vs the number of true features k^*. Left: OLSth, Right: OFSA

averages perform very well in prediction regardless whether the signal is weak or strong, in both regression and classification.

In Figure 5.1, we can observe that the true feature detection rate (DR) is irrelevant with the number of true features k^*, when the sample size n is fixed. The results in the figures follow our theoretical analysis in Proposition 3.

Finally, we know that the computational complexity for obtaining the model from the running averages does not depend on the sample size n, but the time to update the running averages, shown as RAVE in Tables 5.1 and 5.3, does increase linearly with n. Indeed, we observe in Tables 5.2 and 5.4 that the running time of OFSA and OLSth does not have significant changes. However, because of the need to tune the penalty parameters in OLasso, OElnet, and OMCP, it takes more time to run these algorithms. The computational complexity for traditional online algorithms will increase with sample size n. This is especially true for OPG, RDA, and SADMM, which take a large amount of time to tune the parameters to select k features. When the sample size n is very large, running these algorithms takes more than a day.

5.2 Theoretical Upper Bound for OLS-th

In this section, we compare the theoretical upper bound of OLS-th method from Eq. (3.12) and (3.14). In Eq. (3.12), we derived our upper bound based on the design data matrix $\mathbf{X}$. However, in Eq. (3.14), we assumed the observations $\mathbf{x}_i$, $i = 1, 2, \cdots$, are drawn from the multivariate normal

distribution with covariance matrix $\mathbf{\Sigma}$. Thus, we can introduce theoretical upper bound by using the minimum eigenvalue of the covariance $\mathbf{\Sigma}$.

We also compute the experimental bound as the baseline. The experimental bound is based on numerical experiments and shows the actual true feature recovery capability of the OLS-th method. We find it as the smallest $\beta_{\min}$ that achieves a 100% variable detection rate. To do that, we search on an exponential grid $\beta_i = \beta_0 \cdot 0.9^k, k = 0, 1, 2, ...,$ where β_0 is the theoretical bound from Eq. (3.14), and find the last β_i that still has 100% variable detection rate out of 100 runs.

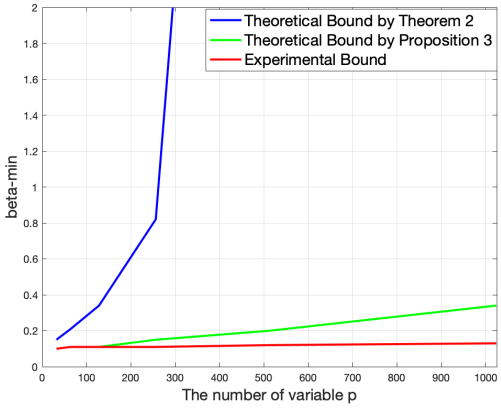

Figure 5.2: Theoretical and experimental bounds for the OLS-th method, $\beta_{\min}$ vs. number of variables p.

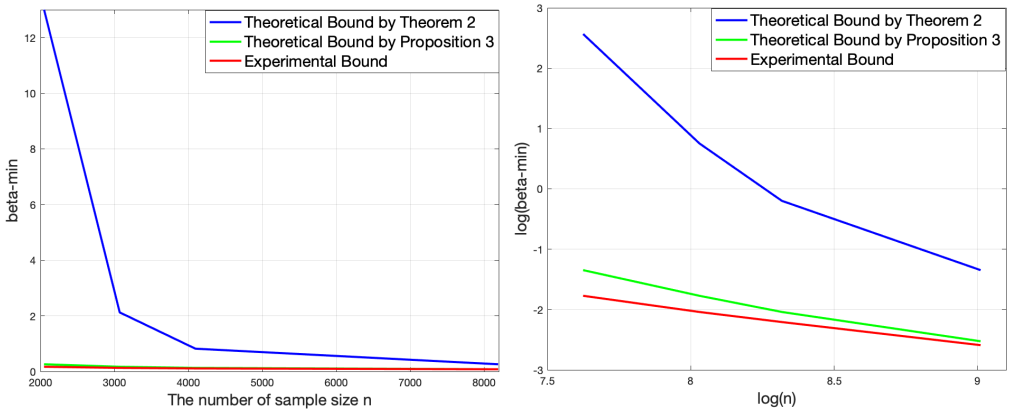

Figure 5.3: Theoretical and experimental bounds for the OLS-th method. Left: $\beta_{\min}$ vs sample size n. Right: $\log(\beta_{\min})$ vs. $\log(n)$.

The two theoretical bounds and the experimental bound are shown in Figures 5.2 and 5.3. In Figure 5.2, we can see that our theoretical upper bound based on Theorem 2 increases rapidly with

the number of variables p. Comparing with the upper bound based on Theorem 2, the upper bound based on the Proposition 3 is closer to the experimental one. And in Figure 5.3, we can see that theoretical upper bound based on Theorem 2 decreases drastically when the sample size increases, and it enjoys the highest decrease rate among the three bounds. And the upper bound based on the Proposition 3 is very close to the experimental one.

5.3 Regret Analysis

In this section, we present results about the regret of the different online methods in regression settings. In traditional online learning, the theoretical analysis of upper bound for the regret was studied in [11] and [50]. Here, we focus on comparing the regret of the running averages-based online algorithms with the state of the art online algorithms.

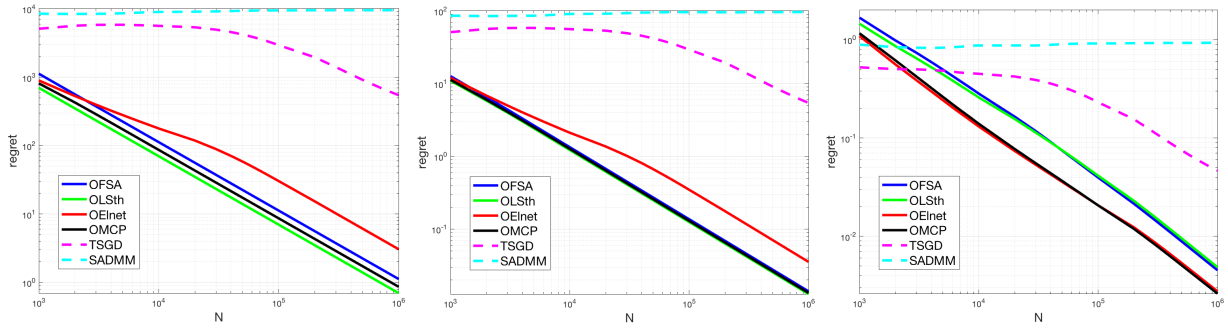

Figure 5.4: log(Regret) vs log(n) for TSGD, SADMM and running averages based online algorithms, averaged over 20 runs. Left: strong signal ($\boldsymbol{\beta} = 1$), middle: medium signal($\boldsymbol{\beta} = 0.1$), right: weak signal($\boldsymbol{\beta} = 0.01$).

Figure 5.4 shows the curve of the regret for $\boldsymbol{\beta} = 1$(left), $\boldsymbol{\beta} = 0.1$(middle), $\boldsymbol{\beta} = 0.01$(right). The sample size n varies from 1000 to 10^6. The regret of the stochastic ADMM method does not converge when we control the number of selected features to be at most k. We compare slopes to see the difference in convergence rates. The convergence rate for the running averages methods is close to $O(n^{-1})$. TSGD seems to also have the same convergence rate but starts off with a plateau where the regret does not converge. The SADMM does not converge at all in our experiments.

5.4 Model Adaptation

In this section we present two simulations for linear regression models where the coefficients drift in time. In the first one, we follow the data generation method in the simulation part to generate data and we used the parameter setting: $p = 100$ and $k = 10$. But here we assume each nonzero β_j is varying with t:

$$\beta_{tj} = a \sin\{2\pi \frac{(t - 100j)}{T}\} + b, \ j = 1, 2, \cdots, k, \ t = 1, 2, \cdots, T, \tag{5.1}$$

in which T is an unknown period. In our simulation, we have $a = 0.4, b = 0.6$ and $T = 1000$. In each time period, we generate 1000 observations. We use model adaptation based on equation (3.11) with the model adaptation rate $\alpha_n = 0.01$. According to Figure 5.5, our model adaptation

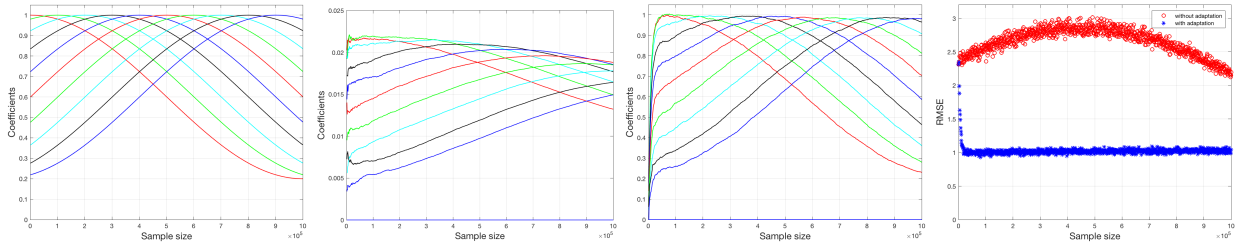

Figure 5.5: Model adaptation experiment. From left to right: true signal, parameters without adaptation, parameter with adaption, RMSE for prediction.

method can track the varying coefficients and perform better in prediction than without model adaptation. In Table 5.5 are shown the RMSE for the last few hundred time steps, averaged over 20 independent runs. One can see that the RMSE with model adaptation is close to the best RMSE possible (1.0) and without model adaptation the prediction is quite poor.

Table 5.5: RMSE for adapted model and non-adapted model, averaged over 20 independent runs.

	With model adaptation	Without adaptation
RMSE	1.028	2.280

In the second numerical experiment, we simulate a high dimensional dynamic pricing and demand problem [27]. Here we assume the demand D_t follows a linear combination of price and other covariates. Hence we consider a simple model as

$$D_t = \beta_0 + \gamma p_t + \mathbf{x}_t \boldsymbol{\beta}_t + \epsilon_t, \ t = 1, 2, \cdots, \tag{5.2}$$

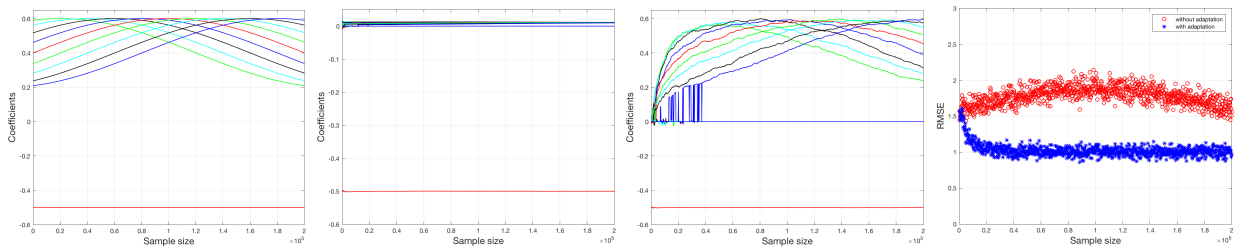

Figure 5.6: Model adaptation for dynamic pricing with feature selection. From left to right: true signal, parameters without adaptation, parameter with adaption, RMSE for prediction.

in which $\gamma \in \mathbb{R}$ is coefficient for price at time period t, $\boldsymbol{\beta}_t \in \mathbb{R}^{p-1}$ is parameter vector for the other covariates, and we have $\gamma < 0$ in the model. The parameter γ, β_0, $\boldsymbol{\beta}$ are unknown to the seller and need to be estimated. Here we assume $\boldsymbol{\beta}_t$ is sparse and varying with time. The above model is commonly used in the economic community to disclose the relationship between the demand and the price. More details about the model of demand and price are given in [27].

For the true price parameter we have $\gamma = -0.5$, and $p_t \sim \mathcal{U}[10, 20]$. For the other covariates $(\beta_{jt}, j = 2, 3, \cdots, k)$, we still use the equation (5.1), with $a = 0.2, b = 0.4, T = 2000$. For each time period t, we generate 200 observations and the model adaptation rate was also $\alpha_n = 0.01$. Our simulation discloses the relationship between demand and price in a varying marketplace. Here we assume that the marketplace varies slowly in a very long period. Our simulation setting is better than [27] because we consider continuous varying-coefficients and true feature recovery in our setting, which is more complex. However, we will not discuss the theoretical analysis here, which is left for a future study.

The results of our dynamic pricing simulation are shown in Figure 5.6. One can see again that the model adaptation works quite well in following the drifting coefficients, and the RMSE is much smaller than without adaptation. Here we also see the model selection power of the running averages in practice. In the plot of the coefficients with adaptation, the smallest non-zero coefficient (blue line) oscillates between being in the model and being zero until sufficient data is available, then it is permanently added to the model.

5.5 Real Data Analysis

In real data analysis, we apply the running averages based methods to some real world datasets. The first dataset is about age estimation from a single image. Age estimation is a regression

problem, as the age has a continuous range of values. The dataset is called Wikiface [28, 29], containing 53,040 face images of actors from Wikipedia and their age. The faces are cropped and resized to 224×224 pixels. From each face image a 4096 dimensional feature vector is extracted using the pre-trained VGG-16 [34] convolutional neural network (CNN). A linear regression model is used to estimate the age from the 4096 dimensional feature vector.

The second dataset is the Year Prediction MSD dataset, from the UCI Machine Learning Repository [19]. This dataset, with 90 features and 463,715 observations, is about the prediction of the release year of a song from audio features. In this dataset, we show how to extend the linear model to a polynomial model by using running averages: we generate new features as products of all pairs of the 90 features, obtaining a 4185 dimensional feature vector. Then we compute the running averages and input them into OLSth or OFSA. Here, we will compare the R^2 of the linear model with the nonlinear model.

The results are shown as the average of 20 random splits of 87.5% training and 12.5% test data for the first dataset, 80% training and 20% test data for the second dataset. For each method, multiple models are trained using various values of the tuning parameters and sparsity levels k. Then the parameter combination with the largest average test R^2 over 20 random splits is reported in Table 5.6.

Table 5.6: Regression results on real data. The average R^2 for regression obtained over 20 random splits.

Dataset	n	p	OLSth	OFSA	Lasso	TSGD	SADMM
Regression data							
WIKIFace	53040	4096	**0.547**	0.545	0.503	0.400	0.487
Year Prediction MSD (nonlinear)	463715	4185	**0.303**	0.298	-	0	0
Year Prediction MSD	463715	90	**0.237**	**0.237**	**0.237**	0.157	0.183

In the real data analysis, we can see that offline Lasso cannot handle the large size of the Year Prediction MSD data with pairwise interactions, and some online methods obtain an R^2 of 0. In contrast, our running averages based methods not only can be used to build the non-linear model, but also they have better performance than the linear model.

CHAPTER 6

FUTURE STUDY

In this dissertation, we proposed a new online feature selection method, stochastic feature selection with annealing, and a new framework of online learning, running averages framework. Compared with the existing online feature selection methods, the SFSA method and its variants can recover the support of the true features for datasets that have strong correlation between features. Besides, they can select features and estimate the parameters simultaneously. In the theoretical analysis, we provided an analysis of the convergence and consistency to the true coefficients. Then, we evaluated the empirical performance of our proposed method and compared it with other state-of-the-art methods. From our experiments, we can conclude that our methods have excellent performance on different types of datasets, both real and simulated.

However, further study is needed for the proposed stochastic feature selection with annealing method. First, we do not consider the model drift problem of SFSA, a common issue in the online learning. Second, it is worth mentioning that the stochastic feature selection with annealing can also be used in offline learning. One can subsample the data and approximate the gradient using a mini-batch, then select the M_t features by the higher $|\beta_j|$ at each epoch t. Finally, the proposed stochastic feature selection method can be used for feature selection in deep learning. In general, the neural networks are usually trained by stochastic gradient descent (SGD) and its variants. Thus it would be an interesting topic to study if the SGD was replaced by SFSA for training the model, especially for non-vision data.

In the running averages framework, we defined the running averages to replace the data matrix, and we showed how to normalize the data in the running averages and designed a series of feature selection algorithms based on them. In contrast to the standard online methods, the proposed framework can be used for model selection, in the sense that different models with different sparsity levels can be built at the same time, without seeing the data again. This is especially useful when the number of observations increases and more complex models can be extracted from the data.

The running averages based methods enjoy good convergence rate and a low computation complexity. More importantly, they can recover the support of the true signal with high probability. We give theoretical guarantees for OLSth and OFSA that they can recover the support of the true signal in the setting of $n >> p$.

In numerical experiments, we have demonstrated that the running averages based methods outperform traditional stochastic learning algorithms and batch learning methods in prediction and feature selection. Moreover, the regret of the running averages methods diminishes faster than the traditional online algorithms.

The running averages based methods could have a wide variety of applications. For instance, in the problem of multi-armed bandit with high-dimensional covariates [2], we can put the multi-armed bandit with linear regression and Lasso into our running averages framework. Also, we could consider to use our online FSA and OLS-th for feature selection in the online decision-making problem.

However, we also need to pay attention to the weaknesses of the running averages-based methods, as they cannot address ultra-high dimensional datasets, the case of $p >> n$, or $p \to \infty$ as $n \to \infty$. The memory complexity and computational complexity for the running averages methods both are $\mathcal{O}(p^2)$. A very large p will be an issue since the running averages would not fit in the computer memory in this case.

APPENDIX A

ALL EXPERIMENTAL RESULTS FOR REGRESSION

In this part, we present the all numerical experiments of the regression for running averages based methods. The simulation is based on two settings: (1): $p = 1000$ and $k = 100$; (2): $p = 10000$ and $k = 1000$. In each data parameter setting, we consider the signal strength $\beta \in \{0.01, 0.1, 1\}$ (weak, medium and strong signals). The sample size n varies from 10^3 to 10^6 for both parameter settings. Stochastic ADMM [26] and TSGD [8] are used to compare with our algorithms. Besides, we cover Lasso [38] as a offline learning method for the comparison.

Table A.1: Comparison between different online and offline algorithms for regression setting, averaged 20-100 runs.

	Variable Detection Rate (%)							test RMSE							Time (s)							
n	Lasso	TSGD	SADMM	OLSth	OFSA	OMCP	OElnet	Lasso	TSGD	SADMM	OLSth	OFSA	OMCP	OElnet	Lasso	TSGD	SADMM	OLSth	OFSA	OMCP	OElnet	RAve
$p = 1000, k = 100$, strong signal $\beta = 1$																						
10^3	32.14	11.22	18.10	77.40	**99.81**	73.71	32.12	11.63	23.15	95.05	5.592	**1.136**	6.282	11.61	4.332	0.007	5.326	0.052	0.289	15.49	9.648	0.026
$3 \cdot 10^3$	46.05	11.22	41.23	**100**	**100**	98.02	45.19	9.464	13.45	93.50	**1.017**	**1.017**	1.745	9.557	26.91	0.019	15.73	0.051	0.288	13.86	7.113	0.076
10^4	72.40	11.22	65.78	**100**	**100**	**100**	72.42	6.07	13.34	94.92	**1.003**	**1.003**	**1.003**	6.042	47.32	0.065	51.80	0.051	0.288	6.508	5.885	0.246
$p = 1000, k = 100$, weak signal $\beta = 0.1$																						
10^3	31.33	10.89	17.53	11.92	**76.92**	60.78	31.33	1.557	2.522	9.560	1.728	**1.201**	1.311	1.555	3.989	0.006	5.387	0.051	0.288	15.32	7.706	0.027
$3 \cdot 10^3$	44.85	10.89	40.11	95.57	**98.43**	89.96	44.11	1.389	1.674	9.392	1.044	**1.026**	1.075	1.403	27.82	0.018	15.98	0.052	0.288	15.84	6.332	0.076
10^4	70.53	10.89	62.48	**100**	**100**	99.98	71.10	1.183	1.663	9.541	**1.003**	**1.003**	**1.003**	1.176	54.50	0.066	53.01	0.051	0.288	10.05	5.814	0.251
$p = 1000, k = 100$, weak signal $\beta = 0.01$																						
10^3	14.09	10.89	13.53	10.11	12.40	**15.55**	14.08	1.128	**1.027**	1.363	1.069	1.169	1.049	1.124	5.353	0.006	6.703	0.052	0.288	13.20	9.741	0.026
10^4	31.58	10.89	19.80	22.48	**32.47**	32.32	31.54	1.009	1.007	1.370	1.025	1.006	**1.005**	1.006	48.13	0.067	67.82	0.051	0.287	14.98	4.961	0.249
10^5	81.93	10.89	11.30	80.55	**85.14**	84.86	81.80	**1.001**	1.010	1.382	1.003	1.003	1.003	1.003	452.2	0.672	679.7	0.051	0.287	15.93	5.120	2.458
$3 \cdot 10^5$	98.66	10.89	10.80	98.94	**99.27**	99.26	98.71	0.999	1.008	1.383	**0.998**	**0.998**	**0.998**	**0.998**	1172	2.001	2044	0.051	0.287	13.96	3.749	7.326
10^6	-	10.89	-	**100**	**100**	**100**	**100**	-	1.005	-	**0.996**	**0.996**	**0.996**	**0.996**	-	6.651	-	0.051	0.288	7.352	1.726	24.36
$p = 10000, k = 1000$, strong signal $\beta = 1$																						
10^4	22.79	10.20	24.01	98.09	**99.09**	24.84	22.76	40.38	42.21	916.7	4.606	**3.341**	40.09	40.72	759.8	0.773	563.5	18.88	28.40	468.4	1451	12.54
$3 \cdot 10^4$	26.64	10.20	10.22	**100**	**100**	39.73	26.48	37.01	42.01	921.7	**1.017**	**1.017**	32.19	36.99	2049	2.319	1687	18.81	28.59	462.9	1092	37.62
10^5	-	10.20	8.89	**100**	**100**	68.76	34.65	-	41.75	865.4	**1.006**	**1.006**	20.44	33.35	-	7.739	5633	19.00	28.49	456.7	983.9	124.8
$p = 10000, k = 1000$, weak signal $\beta = 0.1$																						
10^4	22.69	10.22	21.03	14.51	**97.65**	24.96	22.91	4.188	4.326	92.82	4.351	**1.178**	4.133	4.194	788.1	0.770	564.3	18.89	28.34	429.9	1241	12.48
$3 \cdot 10^4$	26.69	10.22	8.76	**100**	**100**	39.78	26.46	3.833	4.321	93.23	**1.017**	**1.017**	3.373	3.838	1887	2.320	1689	18.92	28.64	435.4	859.1	37.41
10^5	-	10.22	8.87	**100**	**100**	68.81	34.60	-	4.291	86.55	**1.006**	**1.006**	2.272	3.485	-	7.747	5632	18.91	28.64	411.6	884.1	124.5
$p = 10000, k = 1000$, weak signal $\beta = 0.01$																						
10^4	21.89	10.21	17.03	10.07	**31.16**	21.32	21.83	1.104	1.089	9.140	1.144	**1.076**	1.109	1.105	827.4	0.773	564.6	18.91	28.54	442.1	965.3	12.49
$3 \cdot 10^4$	25.87	10.21	9.30	35.02	**52.44**	33.22	26.12	1.070	1.086	9.199	1.108	**1.046**	1.069	1.079	1973	2.327	1693	18.89	28.58	439.6	759.9	37.32
10^5	-	10.21	10.19	77.32	**83.73**	57.38	33.37	-	1.083	8.423	1.025	**1.016**	1.035	1.061	-	7.742	5662	18.81	28.55	449.9	681.9	124.8
$3 \cdot 10^5$	-	10.21	9.92	98.53	**98.95**	85.95	45.66	-	1.082	7.479	1.002	**1.001**	1.009	1.043	-	23.21	16989	18.98	28.54	440.6	741.6	373.0
10^6	-	10.21	-	**100**	**100**	99.55	72.54	-	1.079	-	**1.000**	**1.000**	**1.000**	1.017	-	77.40	-	19.02	28.48	341.3	686.2.	1242

APPENDIX B

THE RUNNING AVERAGES FRAMEWORK IN THE GENERAL ONLINE LEARNING CASE

In general case of online learning, the observations are not arriving as i.i.d. Actually, the online algorithms do not need any distributional assumption and sometimes we assume an adversarial scenario in the online learning for theoretical analysis [22]. In this section, we will prove that our online running averages framework can work in the non-i.i.d case of online learning. In detail, we will prove that OLS-th algorithm and OFSA algorithm can work under this assumption.

B.1 Model Assumption

Here, we still consider the linear regression model from Proposition 3:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon},$$

in where $\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_n^T]^T$ are designed data matrix, and $\mathbf{x}_t \in \mathbb{R}^p$ is a random vector drawn from an unknown distribution at the time period t. $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \cdots, \epsilon_n]^T$ is a random error vector with i.i.d assumption. In this linear model, $\mathbf{x}_t$ and ϵ_t are assumed independent. $\boldsymbol{\beta}^*$ is the true coefficients in the linear regression model.

However, in this case, we do not assume the observations $\mathbf{x}_t, t = 1, 2, \cdots$ are independent with each other. Instead, we assume that the $\mathbf{x}_t$ depend on the past predictor and response $\{\mathbf{x}_i, y_i\}_{i=1}^{t-1}$. Also, we assume that for all $t = 1, 2, \cdots$, we have $\|\mathbf{x}_t\|_\infty < M$, where M is a constant upper bound of the observations $\mathbf{x}_t$.

According to the model assumption and the paper [2], for each feature $\mathbf{X}_j$, $j = 1, 2, \cdots, p$, we define $D_{tj} = \epsilon_t X_{tj}$, in which X_{tj} represent the random variable for the t-th observation and j-th feature. As a consequence, we have the sequence $D_{1j}, D_{2j}, \cdots, D_{tj}$ as a martingale difference. Here, we use the background of measure theory to describe this fact.

Let the σ-algebra $\mathcal{F}_t$ generated by the random variables $X_1, X_2, \cdots, X_{t-1}$ and $Y_1, Y_2, \cdots, Y_{t-1}$, and we have the filtration

$$\mathcal{F}_1 \subset \mathcal{F}_2 \cdots \subset \mathcal{F}_t.$$

Because of the fact $\mathbb{E}[\epsilon_t X_{tj}|\mathcal{F}_t] = 0$, and based on the definition of martingale difference, we can conclude that the sequence $D_{1j}, D_{2j}, \cdots, D_{tj}$ is a martingale difference sequence adapted to the σ-field $\mathcal{F}_1 \subset \mathcal{F}_2 \cdots \subset \mathcal{F}_t$.

B.2 Theoretical Analysis

We start to prove the probabilistic upper bound for our OLS-th algorithm and OFSA algorithm. In the beginning, we present the Azuma inequality for the martingale difference as a Lemma. The proof of this Lemma can be found in [39].

Lemma 5. *Let the $\{(D_i, \mathcal{F}_i)\}_{i=1}^{\infty}$ be the martingale difference sequence, and we assume that D_i is σ_i-subgaussian for every $i > 0$, i.e., for all $\lambda \in \mathbb{R}$, $\mathbb{E}(\exp\{\lambda D_i\}|\mathcal{F}_{i-1}) \leq \exp\{\lambda^2\sigma_i^2/2\}$ almost surely. Then, for every $t > 0$, we have*

$$\mathbb{P}\big(|\sum_{i=1}^{n} D_i| \geq t\big) \leq 2\exp\big(-\frac{t^2}{2\sum_{i=1}^{n}\sigma_i^2}\big).$$

Now we will extend our conclusion of Proposition 3 to the non-i.i.d case. Here, let $\alpha = 1$, which appears in Proposition 3 and so we can have a simple proof and a beautiful conclusion.

Proposition 6. *Suppose we have the linear model*

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon},$$

where $\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_n^T]^T$ is the design matrix, in which $\mathbf{x}_i \in \mathbb{R}^p$, $i = 1, 2, \cdots$, are drawn from an unknown distribution. We also assume there exists a constant $M > 0$ such that $\|\mathbf{x}_i\|_\infty < \sqrt{M}$, for every $i > 0$. Also assume that the random variable $X_{tj} \cdot \epsilon_t$ is a $(\sqrt{M}\sigma)$-subgaussian random variable. Let $S_{\boldsymbol{\beta}^*} = \{j, \beta_j^* \neq 0\}$, $|S_{\boldsymbol{\beta}^*}| = k^*$ and assume

$$\min_{j \in S_{\boldsymbol{\beta}^*}} |\beta_j^*| > \frac{2\sigma}{\lambda}\sqrt{\frac{\log(p)}{n}}, \text{ for some } \lambda \text{ satisfying } 0 < \lambda \leq \lambda_{\min}(\frac{1}{n}\mathbf{X}^T\mathbf{X}). \tag{B.1}$$

Then with probability $1 - 2p^{1-\frac{2}{M}}$, the index set of top k^* values of $|\hat{\boldsymbol{\beta}}_j|$ is exactly $S_{\boldsymbol{\beta}^*}$.

Proof. According to the assumptions, we have $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}^* + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}$. It is equivalent to that

$$|\hat{\boldsymbol{\beta}}| = |\boldsymbol{\beta}^* + \left(\frac{\mathbf{X}^T\mathbf{X}}{n}\right)^{-1}\frac{\mathbf{X}^T\boldsymbol{\epsilon}}{n}|.$$

Because $X_{ij} \cdot \epsilon_i$ is a sequence of martingale difference for every $i > 0$, and each $X_{ij} \cdot \epsilon_i$ is sub-gaussian random variable bounded by $M\sigma^2$, then by the Lemma 5, for $\forall j = 1, 2, \cdots, p$ and for every $t > 0$ we have

$$\mathbb{P}\left(|\frac{\mathbf{X}_{\cdot j}^T \boldsymbol{\epsilon}}{n}| \geq t \right) \leq 2\exp\{-\frac{nt^2}{2M\sigma^2}\}.$$

Then, let $t = 2\sigma\sqrt{\frac{\log(p)}{n}}$, so we have

$$\mathbb{P}\left(|\frac{\mathbf{X}_{\cdot j}^T \boldsymbol{\epsilon}}{n}| \geq 2\sigma\sqrt{\frac{\log(p)}{n}} \right) \leq 2\exp\{-\frac{2\log(p)}{M}\} \leq 2\exp\{-\frac{2\log(p)}{M}\}, \text{for } \forall j = 1, 2, \cdots, p.$$

Then we use the union bound of the above inequality,

$$\mathbb{P}\left(\|\frac{\mathbf{X}^T \boldsymbol{\epsilon}}{n}\|_\infty \leq 2\sigma\sqrt{\frac{\log(p)}{n}} \right) \geq 1 - 2p\exp\{-\frac{2\log(p)}{M}\} = 1 - 2p^{1-\frac{2}{M}}.$$

Therefore, with probability $1 - 2p^{1-\frac{2}{M}}$, for $\forall j \notin S_{\boldsymbol{\beta}^*}$ we have

$$|\hat{\beta}_j| \leq \frac{1}{\lambda_{\min}(\frac{1}{n}\mathbf{X}^T\mathbf{X})} \|\frac{\mathbf{X}^T \boldsymbol{\epsilon}}{n}\|_\infty \leq \frac{2\sigma}{\lambda_{\min}(\frac{1}{n}\mathbf{X}^T\mathbf{X})}\sqrt{\frac{\log(p)}{n}} \leq \frac{2\sigma}{\lambda}\sqrt{\frac{\log(p)}{n}}.$$

where we used the fact $\frac{1}{\lambda} \geq \frac{1}{\lambda_{\min}(\frac{1}{n}\mathbf{X}^T\mathbf{X})}$. And by the smallest $\boldsymbol{\beta}^*$ condition, we get our conclusion.

□

We have finished the proof of true feature recovery property of OLS-th algorithm in the non-i.i.d case. With the mild conditions, we still can recover the true support of $\boldsymbol{\beta}^*$ even though the observations are non-i.i.d. Now, we will extend the property of true support recovery for the OFSA algorithm to the non-i.i.d case.

Proposition 7. (True feature recovery for OFSA) *With the same conditions and assumptions as Proposition 6, and define the smallest true $\boldsymbol{\beta}^*$ as*

$$\beta_{\min} := \min_{j \in S_{\boldsymbol{\beta}^*}} |\beta_j| > \frac{4\sigma\eta}{1 - 1.62\rho}\sqrt{\frac{p\log(p)}{n}}.$$

Then after $t = [\frac{1}{1.62\rho}\log(\frac{10\|\boldsymbol{\beta}^*\|}{\beta_{\min}})] + 1$ iterations, the OFSA algorithm will output $\boldsymbol{\beta}^{(t)}$ satisfying $S_{\boldsymbol{\beta}^*} = S_{\boldsymbol{\beta}^{(t)}}$ with probability $1 - 2p^{1-\frac{2}{M}}$, where $M > 0$ satisfy $\|\mathbf{x}_i\|_\infty < \sqrt{M}, i = 1, 2, \cdots, n$.

Proof. In Theorem 3 we have the following result

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^t \|\boldsymbol{\beta}^*\| + \frac{1.62\eta\sqrt{p}}{1 - 1.62\rho}\|\nabla f(\boldsymbol{\beta}^*)\|_\infty.$$

And we know that $\nabla f(\boldsymbol{\beta}^*) = \frac{1}{n}\mathbf{X}^T(\mathbf{y} - \mathbf{X}^T\boldsymbol{\beta}^*) = \frac{\mathbf{X}^T\boldsymbol{\epsilon}}{n}$. Then by using the inequality proved in the proof of Proposition 6

$$\mathbb{P}(\|\frac{\mathbf{X}^T\boldsymbol{\epsilon}}{n}\|_\infty \leq 2\sigma\sqrt{\frac{\log(p)}{n}}) \geq 1 - 2p^{1-\frac{2}{M}},$$

and then with probability $1 - 2p^{1-\frac{2}{M}}$, we have

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| \leq (1.62\rho)^t\|\boldsymbol{\beta}^*\| + \frac{3.24\sigma\eta\sqrt{p}}{1 - 1.62\rho}\sqrt{\frac{\log(p)}{n}}.$$

After $t = [\frac{1}{1.62\rho}\log(\frac{10\|\boldsymbol{\beta}^*\|}{\beta_{\min}})] + 1$ iterations, we can show that $(1.62\rho)^t\|\boldsymbol{\beta}^*\| < \frac{1}{10}\beta_{\min}$. Thus, with probability $1 - 2p^{1-\frac{2}{M}}$, we have

$$\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\| < \beta_{\min}.$$

And the conclusion that $S_{\boldsymbol{\beta}^*} = S_{\boldsymbol{\beta}^{(t)}}$ must hold. □

REFERENCES

[1] Adrian Barbu, Yiyuan She, Liangjing Ding, and Gary Gramajo. Feature selection with annealing for computer vision and big data learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2):272–286, 2017.

[2] Hamsa Bastani and Mohsen Bayati. Online decision-making with high-dimensional covariates. 2015.

[3] Cheng-Tao Chu, Sang K Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Kunle Olukotun, and Andrew Y Ng. Map-reduce for machine learning on multicore. In *NIPS*, pages 281–288, 2007.

[4] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *NIPS*, pages 1647–1655, 2011.

[5] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10(Dec):2899–2934, 2009.

[6] John C Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In *COLT*, pages 14–26, 2010.

[7] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

[8] Jianqing Fan, Wenyan Gong, Chris Junchi Li, and Qiang Sun. Statistical sparse online regression: A diffusion approximation perspective. In *AISTATS*, pages 1017–1026, 2018.

[9] Simon Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, 2011.

[10] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[11] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

[12] Prateek Jain, Ambuj Tewari, and Purushottam Kar. On iterative hard thresholding methods for high-dimensional m-estimation. In *Advances in Neural Information Processing Systems*, pages 685–693, 2014.

[13] Adel Javanmard. Perishability of data: dynamic pricing under varying-coefficient models. *The Journal of Machine Learning Research*, 18(1):1714–1744, 2017.

[14] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50. ACM, 2016.

[15] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

[16] S Sathiya Keerthi and Dennis DeCoste. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6(Mar):341–361, 2005.

[17] D Kinga and J Ba Adam. A method for stochastic optimization. In *ICLR*, volume 5, 2015.

[18] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(Mar):777–801, 2009.

[19] M. Lichman. UCI machine learning repository, 2013.

[20] Ji Liu, Jieping Ye, and Ryohei Fujimaki. Forward-backward greedy algorithms for general convex smooth functions over a cardinality constraint. In *International Conference on Machine Learning*, pages 503–511, 2014.

[21] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Identifying suspicious urls: an application of large-scale online learning. In *ICML*, pages 681–688, 2009.

[22] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. 2018.

[23] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[24] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[25] Matey Neykov, Jun S Liu, and Tianxi Cai. L1-regularized least squares for support recovery of high dimensional single index models with gaussian designs. *Journal of Machine Learning Research*, 17(87):1–37, 2016.

[26] Hua Ouyang, Niao He, Long Tran, and Alexander Gray. Stochastic alternating direction method of multipliers. In *ICML*, pages 80–88, 2013.

[27] Sheng Qiang and Mohsen Bayati. Dynamic pricing with demand covariates. 2016.

[28] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *ICCV Workshops*, pages 10–15, 2015.

[29] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, pages 1–14, 2016.

[30] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[31] Yiyuan She et al. Thresholding-based iterative selection procedures for model selection and shrinkage. *Electronic Journal of statistics*, 3:384–415, 2009.

[32] Jie Shen and Ping Li. On the iteration complexity of support recovery via hard thresholding pursuit. In *ICML*, pages 3115–3124, 2017.

[33] Jie Shen and Ping Li. A tight bound of hard thresholding. *The Journal of Machine Learning Research*, 18(1):7650–7691, 2017.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[35] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, pages 1139–1147, 2013.

[36] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[37] Taiji Suzuki. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *ICML*, pages 392–400, 2013.

[38] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[39] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

[40] Jialei Wang, Peilin Zhao, Steven CH Hoi, and Rong Jin. Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):698–710, 2014.

[41] Steve Webb, James Caverlee, and Calton Pu. Introducing the webb spam corpus: Using email spam to identify web spam automatically. In *CEAS*, 2006.

[42] Xindong Wu, Kui Yu, Hao Wang, and Wei Ding. Online streaming feature selection. In *ICML*, pages 1159–1166, 2010.

[43] Yue Wu, Steven CH Hoi, Tao Mei, and Nenghai Yu. Large-scale online feature selection for ultra-high dimensional sparse data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(4):48, 2017.

[44] Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010.

[45] Haichuan Yang, Ryohei Fujimaki, Yukitaka Kusumura, and Ji Liu. Online feature selection: A limited-memory substitution algorithm and its asynchronous parallel variation. In *SIGKDD*, pages 1945–1954. ACM, 2016.

[46] Xiao-Tong Yuan, Ping Li, and Tong Zhang. Gradient hard thresholding pursuit. 2018.

[47] Xiaotong Yuan, Ping Li, and Tong Zhang. Gradient hard thresholding pursuit for sparsity-constrained optimization. In *International Conference on Machine Learning*, pages 127–135, 2014.

[48] Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*, pages 894–942, 2010.

[49] Tong Zhang. Adaptive forward-backward greedy algorithm for learning sparse representations. *IEEE transactions on information theory*, 57(7):4689–4708, 2011.

[50] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.

[51] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

[52] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

BIOGRAPHICAL SKETCH

Lizhe Sun received his Bachelor of Science degree in Applied Mathematics from Wuhan University, Hubei, China in 2013. In 2014 Lizhe started his PhD degree in Statistics in Florida State University. Due to his interests in machine learning, Lizhe worked on his PhD thesis under the supervision of Prof. Adrian Barbu. Lizhe's research topic is feature selection in online learning. He has finished a number of projects on this topic, such as the online running averages framework and the stochastic feature selection with annealing. Besides these topics, he is interested in reinforcement learning and robust regression.