Online Feature Selection: A Limited-Memory Substitution Algorithm and Its Asynchronous Parallel Variation

Haichuan Yang
Department of Computer Science
University of Rochester
Rochester, NY 14627
h.yang@rochester.edu

Ryohei Fujimaki
NEC
Cupertino, CA 95014
rfujimaki@nec-labs.com

Yukitaka Kusumura
NEC
Cupertino, CA 95014
ykusumura@nec-labs.com

Ji Liu
Department of Computer Science
University of Rochester
Rochester, NY 14627
ji.liu.uwsic@gmail.com

ABSTRACT

This paper considers the feature selection scenario where only a few features are accessible at any time point. For example, features are generated sequentially and visible one by one. Therefore, one has to make an online decision to identify key features after all features are only scanned once or twice. The optimization based approach is a powerful tool for the online feature selection.

However, most existing optimization based algorithms explicitly or implicitly adopt L_1 norm regularization to identify important features, and suffer two main disadvantages: 1) the penalty term for L_1 norm term is hard to choose; and 2) the memory usage is hard to control or predict. To overcome these two drawbacks, this paper proposes a limited-memory and model parameter free online feature selection algorithm, namely online substitution (OS) algorithm. To improve the selection efficiency, an asynchronous parallel extension for OS (Asy-OS) is proposed. Convergence guarantees are provided for both algorithms. Empirical study suggests that the performance of OS and Asy-OS is comparable to the benchmark algorithm Grafting, but requires much less memory cost and can be easily extended to the parallel implementation.

Keywords

Feature Selection; Online Learning; Asynchronous Parallel Optimization

1. INTRODUCTION

Feature selection plays a key role in many learning and mining tasks [12]. Substantial research efforts have focused on the batch selection, where all features are assumed to be accessible at any time point and can be accessed for arbitrarily many times if needed e.g. works in [19, 3, 25, 6]. However, the batch selection may meet the bottleneck in terms of computation and memory for the big data application, while the ultimately wanted features only occupy a tiny space. Some works [18, 28] can handle large number of features, but all the features are assumed to be accessible in the selection process.

Online feature selection [14, 32] relaxes the requirement in the batch selection and makes a series of decisions to identify key features, to fit important applications where features cannot be accessed at one time, for example,

- features are too many to store locally;
- features can only be queried remotely and one needs to identify important features after scanning all the features one time immediately.

There are usually two types of online feature selection algorithms: statistical algorithms [32, 23, 33, 22] and optimization based algorithms (or embedded methods) [14, 34], which have different application scenarios. The statistical algorithms usually do not have a given objective and features are selected based on certain statistical quantity, e.g. mutual information or correlation. Thus, selected features can be used for many different tasks, but are usually sub-optimal for any specific task (e.g. a given objective function).

This paper mainly focuses on the optimization based approaches, which are target oriented with a clear objective (e.g. loss function) in feature selection, for example, minimizing square loss or classification error. The key difference to statistical methods is that optimization based methods give different selection result for different objective.

Most existing optimization based algorithms such as Grafting [14] or its variation [34] essentially solve an L_1 norm minimization problem similar to batch methods, for example, LASSO [19] and L_1 logistic regression [12]. There are two main drawbacks in practice: 1) As pointed out in [21], it is hard to choose the model parameter (that is, L_1 norm penalty) for online methods. The hyper-parameter in batch methods is usually decided by cross validation (CV). How-

KDD '16, August 13-17, 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-4232-2/16/08...$15.00
DOI: http://dx.doi.org/10.1145/2939672.2939881

ever, the CV strategy is unavailable for the online feature selection scenario, since we can only see a few features. 2) The L_1 norm based online method cannot strictly control the sparsity of the solution path. In the worst case, the required space is comparable to the batch methods.

To overcome these two practical disadvantages, this paper proposes an online feature selection algorithm, namely online substitution (OS) algorithm. OS essentially aims at solving an L_0 norm constraint problem:

$$\min_{\mathbf{w}} \quad L(X\mathbf{w}, \mathbf{y}) \qquad \text{s.t.} \quad \|\mathbf{w}\|_0 \leq s. \tag{1}$$

where $X \in \mathbb{R}^{n\times p}$ is a feature matrix of n samples with p features, $\mathbf{y} \in \mathbb{R}^n$ is the label vector, and s is the total number of features we want to select. Comparing to Grafting types approaches, it has the following advantages:

- (Limited-memory) The memory usage is strictly controlled up to $O(ns)$. The memory restriction has a very special meaning for big data applications. Imagine the following scenario: Given $p = 2M$ features and $n = 0.5M$ samples, one wants to select top $s = 100$ important features. Using the batch method, one needs terabytes of space to save it and solve an optimization to identify top 100 key features using the batch selection. Typically, this work needs to involve parallel computation to solve the batch problem (e.g., L_1 norm minimization). If one can reduce the memory cost to $s \times n \leq$ 1GB to obtain top 100 features, the whole work can be done on a single PC.
- (Model parameter free) One only needs to specify the total number of desired features s (that is much easier to decide the L_1 norm penalty), thus it avoids the dilemma of setting hyper parameters (depending all features).

To improve the computational efficiency in big data application, we propose an asynchronous parallel variation for OS (Asy-OS). Asynchronous parallelism is proven to be more efficient than synchronous parallelism in solving many large scale problems in deep learning, sparse learning, etc. It is mainly because that asynchronous parallelism substantially reduces the communication and coordination cost which cannot be avoided in synchronous parallelism. This paper applies this efficient scheme to parallelize the proposed OS algorithm. Besides of improving the computational efficiency, the memory cost can be shared by multiple machines in Asy-OS, which is able to tackle even larger scale applications.

The contribution of our work can be summarized as follows:

- We propose a limited-memory and model parameter free online feature selection algorithm, which avoids disadvantages of some benchmark algorithms and can be easily extended to the parallel implementation;
- We propose a novel *asynchronous* parallel algorithm to improve the efficiency of the proposed OS algorithm, while only slightly sacrificing the accuracy. To the best of our knowledge, this is the first parallel algorithm for online feature selection.
- Theoretical convergence guarantees are provided for both algorithms.

Notations and Definitions

Throughout this paper, we use the following notations and definitions.

- n is the number of samples.
- p is the number of features.
- $\mathbf{y}^{(i)}$ is the label for the i-th sample.
- $\mathbf{x}^{(i)} \in \mathbb{R}^p$ is a column vector representing i-th sample.
- $\mathbf{x}_j \in \mathbb{R}^n$ is a column vector representing the j-th feature.
- $X \in \mathbb{R}^{n\times p}$ is the data matrix: $X := [\mathbf{x}_1, \cdots, \mathbf{x}_p]$ or $[\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}]^\top$.
- $S \subset \{1, \cdots, p\}$ is the index set of features, and $\overline{S}$ denotes its complementary set.
- X_S is the matrix containing features in set S.
- $\mathbf{w}$ is the p dimensional vector containing coefficients of all the features.
- $\mathbf{w}_j \in \mathbb{R}^p$ is the coefficient of the j-th feature.
- $\mathbf{w}_S$ is the coefficients of the features in set S.
- $\mathbf{w}^k$ is the coefficient vector in the k-th iteration.

2. RELATED WORK

Our work relates to topics including batch feature selection, streaming feature selection, and asynchronous parallel optimization. In this section, we will introduce methods on all the areas and emphasize the ones which are closely related to ours.

2.1 Batch Feature Selection

Batch feature selection is the prototype used in most traditional feature selection methods. The entire set of features is given before doing selection. Most batch feature selection methods can be classified as statistical methods or optimization methods. Statistical methods select features according to certain criteria which are usually built based on statistical rules. For example, method based on mutual information [13], selecting features based on relevance and redundancy [24], and method based on dependency [16].

Optimization based feature selection methods are also called embedded methods in some literature. Its basic idea is doing feature selection and learning the model concurrently. These methods usually have a linear model $\mathbf{w}$, and require it to be sparse, i.e. $\|\mathbf{w}\|_0 \leq s$. Different feature selection methods handle the sparsity requirement differently. For instance, L_1 regularization based methods [19, 12, 3] relax the hard constraint to L_1 penalty, and transform the problem to be convex. Besides using an overall sparse regularization, there are many works [25, 2, 6] proposed for structured sparsity. Unsupervised objective is also used in some works [8, 9]. However, we cannot directly control the number of selected features by using the L_1 regularized methods. Different with the original L_0 constraint, they use a regularization parameter λ to control the sparsity, but there is no explicit relation between s and λ, which means we need to tune λ if we want to get a reasonable result, i.e. will not select too much or too little features.

An alternative way to handle the sparsity requirement is optimizing the problem just with hard L_0 constraint [5, 26]. Although this leads to an NP-hard problem and we may never get the global optimal solution, there is still theoretical guarantee for the error bound [26]. The basic approach of these methods is projected gradient descent. Thanks to the simplicity of projection onto L_0 ball, their efficiency is somehow satisfactory.

2.2 Streaming Feature Selection

The basic problem setting of streaming feature selection is we only have access to a small part of features, e.g., the input feature. Previously input features are allowed to be retained, but we also should consider the limit on memory. This obstacle is an issue when feature selection method considers the relation between different features. Many streaming feature selection methods can be seen as extension of some batch feature selection approach, so they also belong to statistical methods or optimization methods. For example, Zhou et.al. [32] proposed the alpha-investing criterion to select features. OSFS [22] dynamically selects the features based on the online analysis and the online redundancy analysis. Both of their criteria are based on statistical quantities.

In this paper, we focus on optimization based streaming feature selection methods. Grafting method [14] is built based on the L_1 regularized formulation, and it can be used in any problem with L_1 regularization theoretically. It mainly select features at two time points. When the feature just come, grafting test it with a simple criterion and reject it if it fails. When the feature pass the test, grafting will include it into the other selected features and solve a small scale batch sparse learning problem, finally only the features with nonzero coefficient can be retained. Since grafting is actually the online version of L_1 regularization method, it has all the problems of L_1 based method. What's worse, tuning λ seems harder since we even cannot access all the features, and we do not know how much space we need in the process of feature selection. Besides that, it lacks theoretical analysis about convergence rate and error bound with respect to the number of iteration. It also has another practical problem that it takes too much time if the feature pass the test, since a full optimization need to be conduct. This defect has been pointed out and improved by grafting-light [34].

2.3 Asynchronous Parallel Optimization

Asynchronous parallelism is a new parallel mechanism to speedup the optimization efficiency and received broad attention recently. It avoids the coordination and synchronization cost comparing the traditional synchronous parallelism and received remarkable successes in optimization and machine learning for solving deep learning [7, 30, 31], SVM [11], linear programming [17], linear equations [1], LASSO [10], matrix completion [15], and many others [29].

3. ALGORITHM

The online feature selection problem considered in this paper can be formally defined in the following: while features come one by one, given a memory budget and a target (e.g., sample labels or measurements on samples), we need to decide what features to retain.

The loss function L in (1) can be in many different forms. For linear regression, the most common L is in the squared loss form

$$L_r := \frac{1}{2n}\|X\mathbf{w} - \mathbf{y}\|^2. \tag{2}$$

For classification task, it could be the squared hinge loss

$$L_c := \frac{1}{2n}\sum_{i=1}^{n}\max(0, 1 - \mathbf{y}^{(i)}\mathbf{w}^\top\mathbf{x}^{(i)})^2 \tag{3}$$

where $\mathbf{y}^{(i)} \in \{-1, 1\}$. L could also be logistic regression loss, SVM regression loss, etc. In this section, we do not restrict the form of L.

There are several methods [5, 26] that work on analyzing and solving problem (1). The main idea of their algorithm is projected gradient descent (Proj-GD), which makes a normal gradient descent step and projects the current point onto the L_0 ball. Yuan et.al. [26] derive the convergence rate and error bound for general convex and smooth loss function. In a word, Proj-GD works very well on the L_0 constrained problem.

However, in order to use Proj-GD to do feature selection, we must have the access to all the features. For streaming feature selection, grafting [14] and grafting-light [34] uses the stage wise strategy to update the model $\mathbf{w}$. Since they are all based on the L_1 regularized formulation, it is easy to guarantee descent on objective value, and the global optimum can be attained if we can scan all the features again and again. But L_0 constraint based formulation is another story, even batch method Proj-GD can only get the approximate solution [26]. Furthermore, how can we guarantee descent and convergence is not clear.

3.1 Online Substitution Algorithm

The basic reason that we cannot do Proj-GD for streaming feature selection is that we can nether get the gradient, nor update the coefficients for the unseen feature. We are only allowed to access couple of features at the same time, which are actually the new coming feature, and some features temporarily retained. Since new features keep coming, and we cannot retain all of them, this means we need to make decision on rejecting or accepting features in this process.

We propose a method which is motivated by Proj-GD. We maintain a set S containing features temporarily retained. The maximum size of S is s. When S is not full, new features are always accepted. If the set S is full, we have to reject an existing feature in S if we decide to accept a new feature, that is, the new feature will *substitute* the old feature. We call this process "online substitution". The criteria for substitution is comparing the potential of the new feature with the worst features in S. At iteration k, if the coming feature has index j, then the procedure is:

1. Update coefficients of the retained features with step length $\frac{\eta}{m}$: $\mathbf{w}_S = \mathbf{w}_S - \frac{\eta}{m}\nabla_S L$, and update coefficient of the new feature with step length η: $\mathbf{w}_j = -\eta\nabla_j L$;
2. Project $\mathbf{w}$ onto $\Omega(s)$: $\mathbf{w} = P_{\Omega(s)}(\mathbf{w})$.

where $m \geq 1$ is a parameter, $P_{\Omega(s)}(\mathbf{w})$ means projecting $\mathbf{w}$ onto the set $\Omega(s) := \{\mathbf{w} | \|\mathbf{w}\|_0 \leq s\}$, that is, retain top k largest (in the sense of magnitude) elements in $\mathbf{w}$ and set the rest to zero. In section 4.2, We will see that the parameter m is useful to the convergence guarantee of our asynchronous parallel method. Since $\mathbf{w}$ in step 2 has at most $s+1$ nonzero elements, so the projection step is just setting the minimum (in magnitude) nonzero element as zero.

The pseudo code is shown in Algorithm 1. From the algorithm we can see that the partial gradient $\nabla_S L$ is written as $X_S^\top \frac{\partial L}{\partial \mathbf{u}}$, where $\mathbf{u} := X\mathbf{w}$. Actually $\frac{\partial L}{\partial \mathbf{u}}$ can be formed as a function of $\mathbf{u}$, for least square loss (2), it is:

$$\frac{\partial L}{\partial \mathbf{u}} = \frac{1}{n}(\mathbf{u} - \mathbf{y}) \tag{4}$$

and for the squared hinge loss (3), it is

$$\frac{\partial L}{\partial \mathbf{u}} = -\frac{1}{n}\max(0, 1 - \mathbf{u} \circ \mathbf{y}) \circ \mathbf{y} \tag{5}$$

where $\circ$ means element-wise multiplication.

3.2 Asynchronous Online Substitution Algorithm

Because we only make one partial gradient step (W.R.T. $S \cup \{j\}$), our method OS is much efficient for handling each

Algorithm 1: Online Substitution (OS) algorithm.

```
Data: Label $\mathbf{y}$.
Result: Set $S$ consisting of selected features.
1  set $S = \emptyset$;
2  repeat
3      Receive a feature $\mathbf{x}_j$ from the pool $\overline{S}$ with index $j$;
4      $\mathbf{u} := X_S \mathbf{w}_S$;
5      $\mathbf{w}_S = \mathbf{w}_S - (\eta/m) X_S^\top \frac{\partial L}{\partial \mathbf{u}}$;
6      $\mathbf{w}_j = -\eta \mathbf{x}_j^\top \frac{\partial L}{\partial \mathbf{u}}$;
7      update $S = S \cup \{j\}$;
8      if $|S| > s$ then
9          $\mathbf{w}_{k^*} = 0$, $S = S \setminus \{k^*\}$, where
           $k^* = \arg\min_{k \in S} |\mathbf{w}_k|$
10     end
11 until Reach convergence;
```

coming feature when compare with grafting. However, if the new features are input too frequently, the online substitution speed may not catch up its speed. One promising way to improve the efficiency is using parallel optimization. We extend our method OS to an asynchronous method.

The asynchronous parallelism implementation has a similar procedure with Algorithm 1, but here we have multiple workers selecting features. Each worker t uses set $S(t)$ with size s/q, where q is the number of workers. In addition, computing the gradient needs information from other workers, we can use a central machine to collect all updates in different workers as in Algorithm 2. The procedure for the central machine is very simple: it just receives all the $\Delta\mathbf{u}(t)$ sent by each worker t, and add it to the central state variable $\mathbf{u}_C$, which actually represents $X\mathbf{w}$.

The procedure for workers is shown in Algorithm 3. Firstly, pull the central state variable $\mathbf{u}_C$ from the central machine, then save the current local state variable $\mathbf{u}(t)$. Secondly, use $\mathbf{u}_C$ to compute the partial gradient W.R.T. $\mathbf{w}_{S(t)}$ and $\mathbf{w}_j$, and perform update according to the gradient and projection. At last, calculate the difference variable $\Delta\mathbf{u}(t)$ and send it to the central machine. In the whole procedure of central machine and workers, there is no synchronization process.

Algorithm 2: Asy-OS: procedure of the central machine.

```
1  set $\mathbf{u}_C = \mathbf{0}$;
2  repeat
3      if Receive $\Delta\mathbf{u}(t)$ from a certain worker $t$. then
4          $\mathbf{u}_C = \mathbf{u}_C + \Delta\mathbf{u}(t)$;
5      end
6  until Workers stop pushing $\Delta\mathbf{u}(t)$;
```

4. THEORETICAL ANALYSIS

In this section, we show the main result of our theoretical analysis, including convergence of OS and Asy-OS algorithm, and the property of their local minimum. Proofs are provided in the Appendix A.

Firstly, we make some certain Lipschitzian assumptions to prepare the following theoretical analysis. Lipschitzian assumptions are commonly used in analyzing optimization algorithms. Define function $f(\mathbf{w})$ as $f(\mathbf{w}) := L(X\mathbf{w}, \mathbf{y})$ and

Algorithm 3: Asy-OS: procedure of worker t.

```
Data: Label $\mathbf{y}$.
Result: Set $S(t)$ consisting of selected features.
1  set $S(t) = \emptyset$;
2  repeat
3      Receive a feature $\mathbf{x}_j$ from the pool $\overline{S}$ with index $j$;
4      Read $\mathbf{u}_C$ from the central machine;
5      $\mathbf{u}(t) = X_{S(t)} \mathbf{w}_{S(t)}$;
6      $\mathbf{w}_{S(t)} = \mathbf{w}_{S(t)} - (\eta/m) X_{S(t)}^\top \frac{\partial L}{\partial \mathbf{u}_C}$;
7      $\mathbf{w}_j = -\eta \mathbf{x}_j^\top \frac{\partial L}{\partial \mathbf{u}_C}$;
8      update $S(t) = S(t) \cup \{j\}$;
9      if $|S(t)| > s/q$ then
10         $\mathbf{w}_{k^*} = 0$, $S(t) = S(t) \setminus \{k^*\}$, where
           $k^* = \arg\min_{k \in S(t)} |\mathbf{w}_k|$
11     end
12     Push $\Delta\mathbf{u}(t) = X_{S(t)} \mathbf{w}_{S(t)} - \mathbf{u}(t)$ to the central
       machine;
13 until Reach convergence;
```

construct function $F_{\mathbf{w}}(\mathbf{t}, r)$ by:

$$F_{\mathbf{w}^k}(\mathbf{t}, r) := f(\mathbf{w}^k + \frac{1}{m} \sum_{i \in S} \mathbf{t}_i \mathbf{e}_i + r\mathbf{e}_j)$$

where $\mathbf{t} \in \mathbb{R}^{|S|}$, $r \in \mathbb{R}$. Assume we have:

Assumption 1. **(Lipschitz Gradient.)** *There exists constant $L_F < +\infty$ which satisfies: $\forall \mathbf{t} \in \mathbb{R}^{|S|}$ and $r \in \mathbb{R}$*

$$\begin{aligned} &F_{\mathbf{w}^k}(\mathbf{t}, r) - F_{\mathbf{w}^k}(\mathbf{0}, 0) \\ \leq &\langle \nabla_S F_{\mathbf{w}^k}(\mathbf{0}, 0), \mathbf{t} \rangle + r\nabla_j F_{\mathbf{w}^k}(\mathbf{0}, 0) + \frac{L_F}{2}(\|\mathbf{t}\|^2/m^2 + r^2). \end{aligned} \tag{6}$$

There exists a constant $L_f < +\infty$ satisfying: $\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^p$

$$\|\nabla f(\mathbf{u}) - \nabla f(\mathbf{v})\| \leq L_f \|\mathbf{u} - \mathbf{v}\|. \tag{7}$$

4.1 Convergence of the OS Algorithm

Then we are ready to present the convergence rate for the proposed OS algorithm (Algorithm 1) with the following Theorem:

Theorem 1. *For K iterations, the average distance of $\mathbf{w}$ between two successive iterations in the OS algorithm satisfies*

$$\frac{1}{K\eta^2} \sum_{k=1}^{K} \|\mathbf{w}^{k+1} - \mathbf{w}^k\|^2 \leq \frac{2\left(f(\mathbf{w}^1) - f(\mathbf{w}^{K+1})\right)}{K(1 - \eta L_F)\eta} \tag{8}$$

when $\eta < 1/L_F$ establishes.

The left hand side of (8) is nothing but the $\|\nabla f(\mathbf{w}^k)\|^2$ if using gradient descent to solve an unconstrained optimization. Theorem 1 basically suggests that the sequence $\|\mathbf{w}^{k+1} - \mathbf{w}^k\|^2$ converges and its average rate is $O(1/k)$. It is also worth to point out that the optimal choice for η is $0.5/L_F$.

4.2 Convergence of Asy-OS Algorithm

Next we establish the convergence of the proposed Asy-OS algorithm. For the asynchronous version in Algorithm 2 and 3, the convergence is guaranteed under some conditions. We assume that we have bounded staleness, i.e.

Assumption 2. **(Bounded staleness.)** *In one update iteration of any worker, the central state variable $\mathbf{u}_C$ will not update more than $\tau_{\max}$ times. Thus $\mathbf{u}_C$ updates at most $\tau_{\max}$ times between the each pair of line 4 and line 12 in Algorithm 3.*

Formally, we have the following theorem:

Theorem 2. *The average distance of two successive iterations the Asy-OS algorithm with q workers converges in the following sense:*

$$\frac{1}{K\eta^2}\sum_{k=1}^{K}\mathbb{E}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2 \le \frac{f(\mathbf{w}^1)-\mathbb{E}f(\mathbf{w}^{K+1})}{K\eta^2\Delta}$$

if the step length η is appropriately chosen such that

$$\Delta := \left(\frac{1}{2\eta} - L_F - \left(\frac{1}{2m^2q}+\frac{1}{2(p-s)}\right)\frac{L_f^2}{L_F}\tau_{\max}^2\right) > 0.$$

Note that $\mathbf{w}$ consists of all disjoint pieces from q workers and the iteration counts the change happening to $\mathbf{w}$ in any single worker. Form the above theorem we can see the relation between the step length η and the staleness bound $\tau_{\max}$. Larger $\tau_{\max}$ requires that η to be smaller.

4.3 Local Optimality

Assume that we have the following **Restricted Strong Convexity** assumption:

Assumption 3. *There exists $\rho_-(s) > 0$ which satisfies:*

$$f(\bar{\mathbf{w}}) - f(\mathbf{w}) \ge \langle \nabla f(\mathbf{w}), \bar{\mathbf{w}} - \mathbf{w}\rangle + \frac{\rho_-(s)}{2}\|\bar{\mathbf{w}}-\mathbf{w}\|^2$$
$$\forall\ \mathbf{w}\in\Omega(s), \bar{\mathbf{w}}\in\Omega(s)$$

Then our local optimum has the property

Theorem 3. *If $\mathbf{w}^\infty$ is the local optimal point that we get by our method. Then*

$$f(\mathbf{w}^\infty) - f(\mathbf{w}^*) \le \frac{\alpha}{1+\alpha}\left(f(\mathbf{0}) - f(\mathbf{w}^*)\right)$$
$$\textit{where } \alpha = \left(\frac{1}{(\rho_-(s))\eta}\right)^2$$

where $\mathbf{w}^*$ is the global optimal point. Theorem 3 shows that our local minimum is better with smaller value of α, which also has a relationship with step length η. Larger step length η leads to better local optimum. However, as shown in Theorem 1 and 2, there is an upper bound for η to guarantee convergence.

5. EXPERIMENT

In this section, we demonstrate the empirical performance and the effects of some parameters. The experiments are conducted on two different models: linear regression and hinge loss classification.

Linear regression. Feature selection in linear regression aims at recovering the sparse vector $\mathbf{w}^* \in \mathbb{R}^p$, given the training data $X \in \mathbb{R}^{n\times p}$ and the corresponding observations $\mathbf{y} = X\mathbf{w}^* + \epsilon$, where $\epsilon \in \mathbb{R}^n$ is the noise. In our experiment, the training data X are generated from standard i.i.d. Gaussian distribution. We force the ground truth vector $\mathbf{w}^*$ to be sparse, i.e. $\|\mathbf{w}^*\|_0 \le s$. The nonzero positions of $\mathbf{w}^*$ are randomly selected and their values follow the standard i.i.d. Gaussian distribution. Noise ϵ is generated from i.i.d. Gaussian distribution with mean 0 and variance 0.1^2. We adopt the cost function (2) for linear regression.

Hinge loss classification. Besides regression, we also try our method for classification model. The data matrix X and the sparse ground truth vector $\mathbf{w}^*$ are generated exactly the same way as in linear regression settings. Class labels $\mathbf{y} \in \{-1, +1\}^n$ are generated by taking the sign of $X\mathbf{w}^*$. Another data matrix $X_{test} \in \mathbb{R}^{n\times p}$ and vector $\mathbf{y}_{test}$ are generated exactly the same way as X and $\mathbf{y}$. We use them as the test data. We adopt the squared hinge loss function (3).

In addition, we also test our method on the privacy image classification problem [27]. Its target is to distinguish images which could contain some privacy information from the public images. In the experiment, we use a collected data set in [20] with roughly 3400 images in both classes i.e. privacy and public. We combine them as a dataset with 6914 images, and randomly select one percent data as training set, leave the rest as testing set. Each image is represented by 7488 features generated by several image feature extraction methods including color histogram, linear binary pattern, histogram of oriented gradients, etc.

Baseline Methods. To demonstrate our online feature selection method, we compare it with L_0 constrained batch method (Proj-GD), L_1 regularized batch method i.e. LASSO and L_1 hinge loss classification, grafting method, and two naive online implementations which are based on coordinate descent method and projected or proximal operation. We adopt proximal gradient descent to solve LASSO, and use the implementation of LIBLINEAR [4] to solve L_1 hinge loss classification.

Performance Measure. To compare the performance, we use feature selection recall, estimation error and classification error. Feature selection recall is the ratio that correctly selected features (at most s) over the total amount of used features in the true model. For linear regression, we show the recovery error which is the normalized distance between the recovered model and the true model. For classification problem, we evaluate the classification error. For all the methods, we use the selected features to fit the training data again.

To demonstrate the efficiency improvement of our parallel method Asy-OS, we show the running time and training error (i.e. fitting error for linear regression and classification error for hinge loss classification). We also show the speedup compared with the sequential version.

Implementation Details. In our experiment, we focus on methods based on two basic sparsity formulations: L_0 constraint and L_1 regularization. For L_0 methods including our method OS, the objective function is just equation (1). The L_1 regularized formulation is $\min_{\mathbf{w}} L + \lambda\|\mathbf{w}\|_1$. In this formulation, L is the specific loss function, i.e. L_r in (2) or L_c in (3). L_1 regularization parameter λ is set to guarantee the batch methods selecting at least s features.

In our synthetic data experiments, the total number of features p is set to $2000 \sim 6000$ for linear regression and $1000 \sim 5000$ for classification. The number of nonzero elements is set $s = 100$ both. The number of samples n is set to $1.2s\log_2 p$. For all the online feature selection method, we make the limitation that we can only scan all the features twice. The results are averaged on 30 repeated experiments with different random data.

5.1 Performance Comparison

In the first column of Figures 1 and 2, we show the performance comparison on linear regression and classification problems respectively. We can see that for both problems, L_0 based methods i.e. Proj-GD and our method OS, have higher feature selection recall. For linear regression, our method OS has almost the same performance with the L_0 batch counterpart Proj-GD. But we can find that the estimation error and the feature selection recall are not very consistent. The L_1 methods also have low recovery error, although they do not have the highest feature selection recall. In the first column of Figure 4, we can find that the L_1 batch method has higher error when s is small, and other methods are not influenced too much.

5.2 Sensitivity about Scanning Features

One important issue in practice is how many times of scanning features is sufficient. To demonstrate this, we test our method by scanning features for different number of times, and compare the performance in the second column of Figures 1, 2 and 4, which suggests that 2 or 3 times would be enough since the error does not clearly improve after that.

5.3 Asynchronously Parallel Algorithm

In this section, we study the performance of Asy-OS. We first simulate Asy-OS on a single machine, such that the staleness value $\tau_{\max}$ can be easily controlled. We set $\tau_{\max}$ to be the number of workers, that is, each worker uses the information from other workers "#Workers" iterations iterates ago. From the right column of Figures 1 (for linear regression), 2 (for classification) and 4, we can observe that the recall almost monotonically degrade while the number of workers increases. One interesting finding is that the classification problem seems to be more robust to the staleness (or #Workers).

Next we implement Asy-OS on a computer network consisting of 6 machines. We use Message Passing Interface (MPI) to implement the parallel mechanism. The central node (i.e. Algorithm 2) runs on a single machine, and every two worker-processes run on a specific machine. All the machines have the same hardware (Intel Xeon E5-2430 CPU with 24 cores, 64 GB RAM). The data are generated with the similar way but here we set $n = 1000, p = 8192, s = 64$ for linear regression and $s = 32$ for classification. To compute the speedup, we set an error threshold and record the time used for attaining such error. We repeat the experiment 10 times and report the average result.

In Figure 3, we show the running time of Asy-OS. In the first two sub-figures we can observe that the error decreases rapidly when more workers are involved. When the number of workers is larger than 8, there is almost no improvement of efficiency. We think this is caused by two reasons. First, since every worker will communicate with the central machine, the communication cost over the network will increase when the number of workers becomes larger. Second, the staleness upper-bound $\tau_{\max}$ will be larger when using more workers. In the speedup result, we can find we have more than linear speedup for both applications. The reason is that we separate both the total iterations and the workload for each iteration. For each iteration, each worker only needs to compute its local $X_{S(t)}\mathbf{w}_{S(t)}$ and retrieves $\mathbf{u}_C$ from the central machine (shown in Algorithm 3).

6. CONCLUSIONS

This paper proposes a limited-memory and model parameter free online feature selection algorithm, namely OS, which overcomes the disadvantages of existing optimization based online feature selection algorithm such as Grafting and its variation. To improve the computational efficiency and solve problems with huge scale problem, we propose an asynchronous parallel OS algorithm. Theoretical convergence analysis is provided for both algorithms. Empirical study suggests that the performance of OS and Asy-OS is comparable to the benchmark algorithm Grafting, but requires much less memory cost and is much easier to set sparsity parameter.

7. ACKNOWLEDGMENT

We thank the reviewers for constructive comments. This project is supported by the NEC fellowship and the NSF grant CNS-1548078. The implementation of Asy-OS significantly benefits from the first author's course project on Professor Sandhya Dwarkadas's Parallel and Distributed Systems course at University of Rochester.

8. REFERENCES

[1] H. Avron, A. Druinsky, and A. Gupta. Revisiting asynchronous linear solvers: Provable convergence rate through randomization. *Journal of the ACM*, 62(6):51, 2015.

[2] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, et al. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 2012.

[3] E. Candes and T. Tao. The dantzig selector: statistical estimation when p is much larger than n. *The Annals of Statistics*, pages 2313–2351, 2007.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

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

[6] D. Kong, R. Fujimaki, J. Liu, F. Nie, and C. Ding. Exclusive feature learning on arbitrary structures via $\ell_{1,2}$-norm. In *NIPS*, pages 1655–1663, 2014.

[7] M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola. Parameter server for distributed machine learning. *ArXiv*, 2013.

[8] Z. Li, J. Liu, Y. Yang, X. Zhou, and H. Lu. Clustering-guided sparse structural learning for unsupervised feature selection. *TKDE*, 26(9):2138–2150, 2014.

[9] Z. Li and J. Tang. Unsupervised feature selection via nonnegative spectral analysis and redundancy control. *IEEE Transactions on Image Processing*, 24(12):5343–5355, 2015.

[10] J. Liu and S. J. Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.

[11] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *JMLR*, 16(1):285–322, 2015.

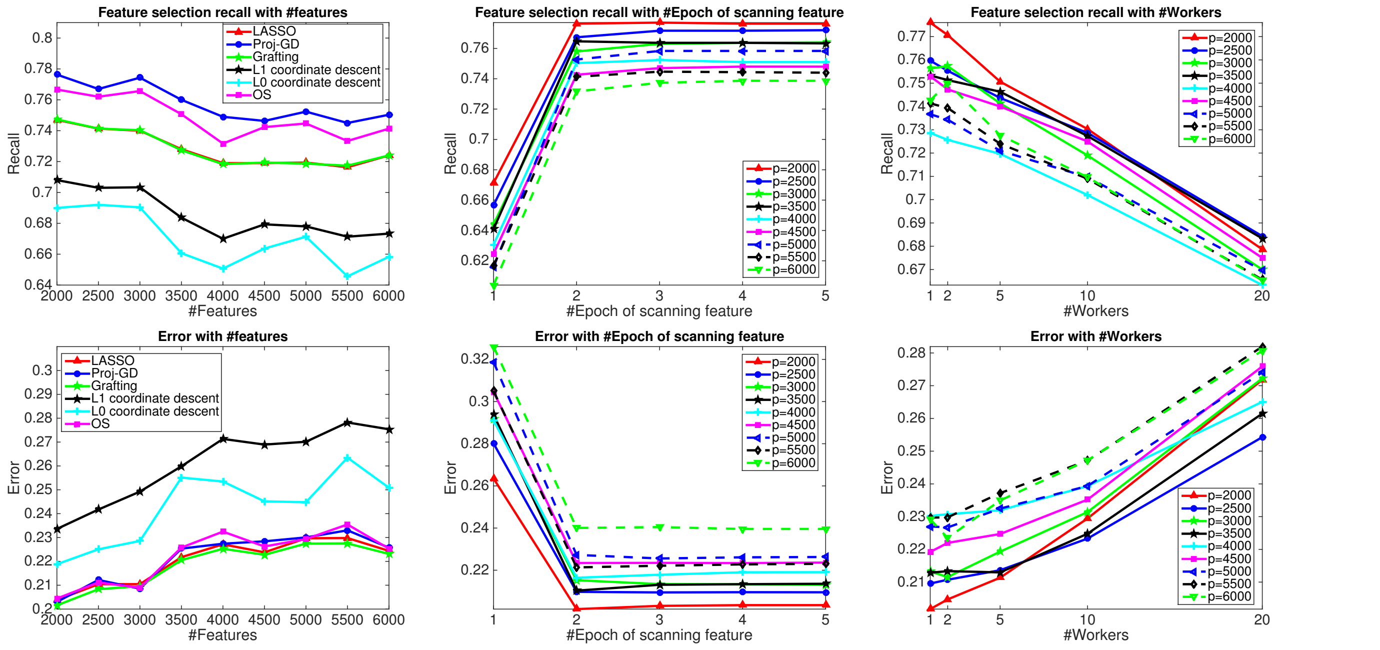

Figure 1: Recovery error and feature selection recall of linear regression on synthetic data sets. The left column shows the comparison of feature selection methods with different number of features; the middle column shows the result with different times of scanning features; and the right column shows the result with different number of workers for Asy-OS.

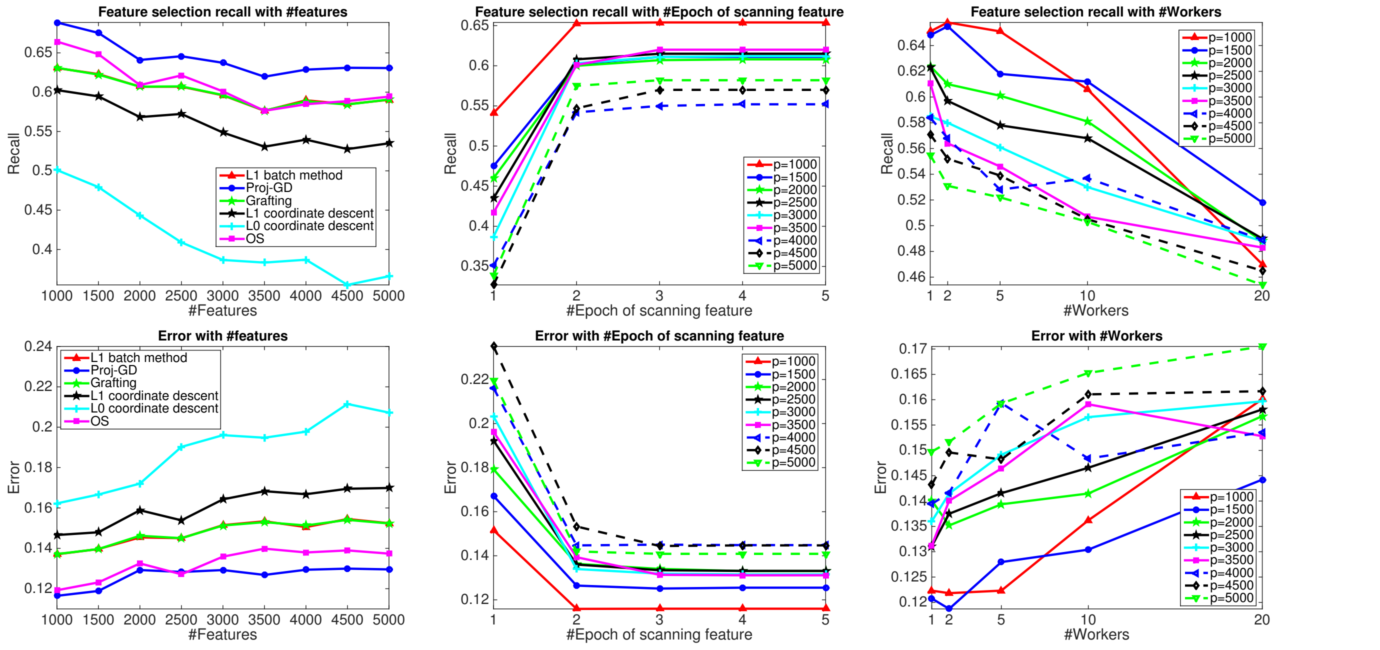

Figure 2: Classification error and feature selection recall of hinge loss classification on synthetic data sets. The left column shows the comparison of feature selection methods with different number of features; the middle column shows the result with different times of scanning features; and the right column shows the result with different number of workers for Asy-OS.

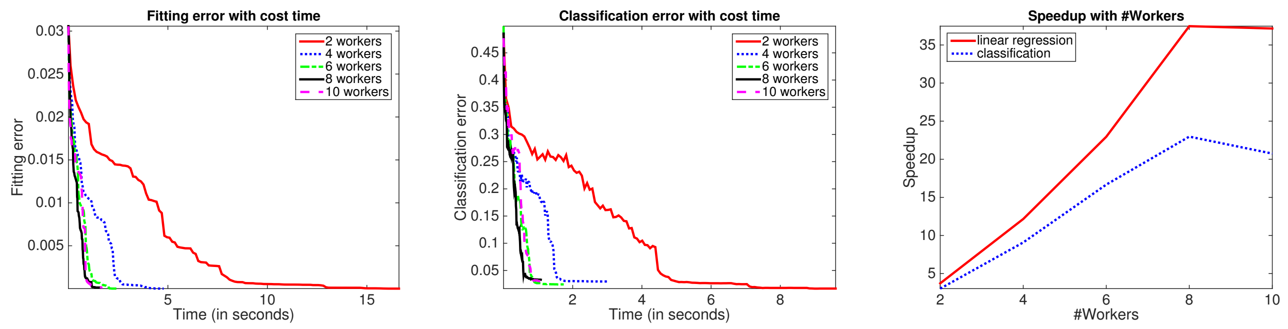

Figure 3: Evaluation on running time for Asy-OS. The left and middle columns show the training error with elapsed time, the right column shows the speedup compared with the sequential method.

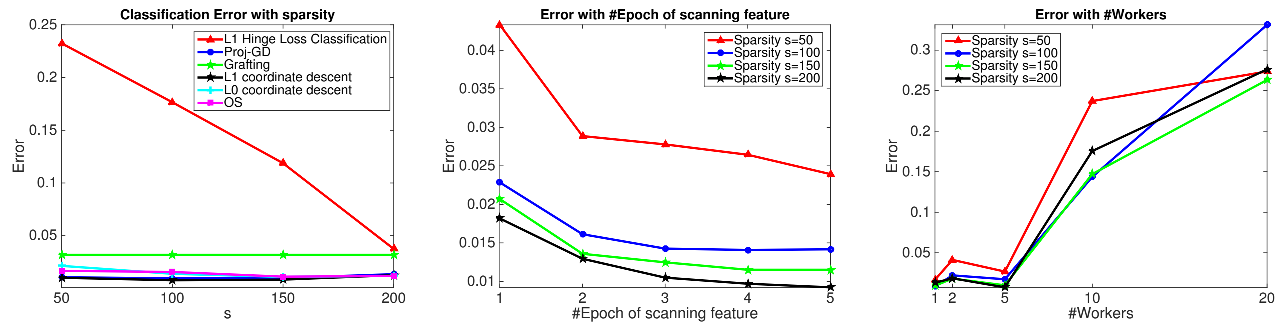

Figure 4: Classification error of privacy image classification. The left column shows the comparison of feature selection methods with different value of s; the middle column shows the result with different times of scanning features; and the right column shows the result with different number of workers for Asy-OS.

[12] A. Y. Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *ICML*, page 78, 2004.

[13] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *TPAMI*, 27(8):1226–1238, 2005.

[14] S. Perkins and J. Theiler. Online feature selection using grafting. In *ICML*, pages 592–599, 2003.

[15] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[16] L. Song, A. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo. Supervised feature selection via dependence estimation. In *ICML*, pages 823–830, 2007.

[17] S. Sridhar, S. Wright, C. Re, J. Liu, V. Bittorf, and C. Zhang. An approximate, efficient lp solver for lp rounding. In *NIPS*, pages 2895–2903, 2013.

[18] M. Tan, I. W. Tsang, and L. Wang. Towards ultrahigh dimensional feature selection for big data. *JMLR*, 15(1):1371–1429, 2014.

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

[20] L. Tran, D. Kong, H. Jin, and J. Liu. Privacy-cnh: A framework to detect photo privacy with convolutional neural network using hierarchical features. In *AAAI*, 2016.

[21] J. Wang, M. Wang, P. Li, L. Liu, Z. Zhao, X. Hu, and X. Wu. Online feature selection with group structure analysis. *TKDE*, 27(11):3029–3041, 2015.

[22] X. Wu, K. Yu, H. Wang, and W. Ding. Online streaming feature selection. In *ICML*, pages 1159–1166, 2010.

[23] K. Yu, X. Wu, W. Ding, and J. Pei. Towards scalable and accurate online feature selection for big data. In *ICDM*, pages 660–669, 2014.

[24] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *JMLR*, 5:1205–1224, 2004.

[25] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[26] X.-T. Yuan, P. Li, and T. Zhang. Gradient hard thresholding pursuit for sparsity-constrained optimization. *ICML*, 2014.

[27] S. Zerr, S. Siersdorfer, J. Hare, and E. Demidova. Privacy-aware image classification and search. In *SIGIR*, pages 35–44, 2012.

[28] Y. Zhai, M. Tan, Y. S. Ong, and I. W. Tsang. Discovering support and affiliated features from very high dimensions. In *ICML*, pages 1455–1462, 2012.

[29] R. Zhang and J. Kwok. Asynchronous distributed admm for consensus optimization. In *ICML*, pages 1701–1709, 2014.

[30] S. Zhang, A. E. Choromanska, and Y. LeCun. Deep learning with elastic averaging sgd. In *NIPS*, pages 685–693, 2015.

[31] W. Zhang, S. Gupta, X. Lian, and J. Liu. Staleness-aware async-sgd for distributed deep learning. *IJCAI*, 2016.

[32] J. Zhou, D. Foster, R. Stine, and L. Ungar. Streaming

feature selection using alpha-investing. In *SIGKDD*, pages 384–393. ACM, 2005.

[33] J. Zhou, D. P. Foster, R. A. Stine, and L. H. Ungar. Streamwise feature selection. *JMLR*, 7:1861–1885, 2006.

[34] J. Zhu, N. Lao, and E. P. Xing. Grafting-light: fast, incremental feature selection and structure learning of markov random fields. In *SIGKDD*, pages 303–312, 2010.

APPENDIX

A. PROOFS

A.1 Proof of Theorem 1

Define S^k as the set of selected features and j_k as the new input feature in the $(k+1)$-th iteration. The OS algorithm following the update rule that:

$$\mathbf{w}^{k+1} = P_{\Omega(s)}(\mathbf{w}^k - \frac{\eta}{m}\sum_{i\in S^k}\nabla_i f(\mathbf{w}^k) - \eta\nabla_{j_k} f(\mathbf{w}^k)).$$

Suppose that $(\mathbf{w}^{k+1}-\mathbf{w}^k)_{S^k} = \mathbf{t}$, $(\mathbf{w}^{k+1}-\mathbf{w}^k)_{j_k} = r$, and we already know that $\mathbf{w}^{k+1}$ and $\mathbf{w}^k$ are the same at other positions. So we have:

$$f(\mathbf{w}^{k+1}) = F_{\mathbf{w}^k}(\mathbf{t}, r).$$

According to inequality (6) from the Assumption 1, we have:

$$\begin{aligned}
&f(\mathbf{w}^{k+1}) - f(\mathbf{w}^k)\\
=&F_{\mathbf{w}^k}(\mathbf{t}, r) - F_{\mathbf{w}^k}(\mathbf{0},0)\\
\le&\langle\nabla_{S^k}F_{\mathbf{w}^k}(\mathbf{0},0),\mathbf{t}\rangle + r\nabla_{j_k}F_{\mathbf{w}^k}(\mathbf{0},0) + \frac{L_F}{2}(\|\mathbf{t}\|^2/m^2 + r^2)\\
=&\frac{1}{m}\langle\nabla_{S^k}f(\mathbf{w}^k), \mathbf{w}^{k+1}-\mathbf{w}^k\rangle + (\mathbf{w}^{k+1}-\mathbf{w}^k)_{j_k}\nabla_{j_k}f(\mathbf{w}^k)\\
&+\frac{L_F}{2}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2\\
\le&\left(\frac{L_F}{2} - \frac{1}{2\eta}\right)\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2 \qquad (9)
\end{aligned}$$

where the last inequality comes from the fact that

$$\begin{aligned}
&\|\mathbf{w}^{k+1} - (\mathbf{w}^k - \eta\frac{1}{m}\sum_{i\in S^k}\nabla_i f(\mathbf{w}^k) - \eta\nabla_{j_k}f(\mathbf{w}^k))\|^2\\
\le&\|\mathbf{w}^{k} - (\mathbf{w}^k - \eta\frac{1}{m}\sum_{i\in S^k}\nabla_i f(\mathbf{w}^k) - \eta\nabla_{j_k}f(\mathbf{w}^k))\|^2\\
=&\|\eta\frac{1}{m}\sum_{i\in S^k}\nabla_i f(\mathbf{w}^k) + \eta\nabla_{j_k}f(\mathbf{w}^k)\|^2.
\end{aligned}$$

Summing up inequality (9) from $k=1$ to K, we get

$$f(\mathbf{w}^1) - f(\mathbf{w}^{K+1}) \ge \left(\frac{1}{2\eta} - \frac{L_F}{2}\right)\sum_{k=1}^{K}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2.$$

Since we know that function f is bounded below, i.e., $f(\mathbf{w}) > -\infty, \forall \mathbf{w}\in\Omega(s)$. If $\left(\frac{1}{2\eta} - \frac{L_F}{2}\right) > 0$, we get

$$\frac{1}{K}\sum_{k=1}^{K}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2 \le \frac{2\eta}{K(1-\eta L_F)}\left(f(\mathbf{w}^1) - f(\mathbf{w}^{K+1})\right).$$

It completes the proof.

A.2 Proof of Theorem 2

To analyze the proposed asynchronous algorithm, we monitor the values of $\mathbf{w}$ concatenating all disjoint pieces from q workers. The central node actually records the value of $\mathbf{u} = X\mathbf{w}$. To be convenient, we define a vector function

$$g_k(\mathbf{w}) = \frac{1}{m}\sum_{i\in S^k(t_k)}\nabla_i f(\mathbf{w}) + \nabla_j f(\mathbf{w}) = \frac{1}{m}\nabla_{S^k(t_k)}f(\mathbf{w}) + \nabla_{j_k}f(\mathbf{w}).$$

where t_k is the worker index which makes the update at the $(k+1)$-th iteration, and $S^k(t_k)$ denotes the set of selected features at worker t_k.

Then the $(k+1)$-th update happens in the central node follows:

$$\begin{aligned}
\mathbf{w}^{k+1}_{S^k(t_k)\cup\{j_k\}} &= P_{\Omega(s/q)}[(\mathbf{w}^k - \eta g_k(\hat{\mathbf{w}}^k))_{S^k(t_k)\cup\{j_k\}}]\\
\mathbf{w}^{k+1}_{\overline{S^k(t_k)\cup\{j_k\}}} &= \mathbf{w}^{k}_{\overline{S^k(t_k)\cup\{j_k\}}}.
\end{aligned}$$

where $\overline{S^k(t_k)\cup\{j_k\}}$ denotes the complementary set of $S^k(t_k)\cup\{j_k\}$.

In the asynchronous parallelism, we can not guarantee $\hat{\mathbf{w}}^k = \mathbf{w}^k$ in general, but we have the following equation when the staleness is limited:

$$\hat{\mathbf{w}}^k = \mathbf{w}^k - \sum_{l\in\tau(k)}(\mathbf{w}^{l+1} - \mathbf{w}^l).$$

where $\tau(k)\subset\{k-1, k-2, ..., k-\tau_{\max}\}$. Since $\mathbf{w}^{k+1}$ and $\mathbf{w}^k$ only differ in $S^k(t_k)\cup\{j_k\}$, we have the inequality:

$$\|\mathbf{w}^{k+1} - (\mathbf{w}^k - \eta g_k(\hat{\mathbf{w}}^k))\|^2 \le \|\mathbf{w}^{k} - (\mathbf{w}^k - \eta g_k(\hat{\mathbf{w}}^k))\|^2,$$

so we get:

$$\langle g_k(\hat{\mathbf{w}}^k), \mathbf{w}^{k+1}-\mathbf{w}^k\rangle \le -\frac{1}{2\eta}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2. \qquad (10)$$

With similar steps of getting inequality (9), we have

$$\begin{aligned}
&f(\mathbf{w}^{k+1}) - f(\mathbf{w}^k)\\
\le&\langle g_k(\mathbf{w}^k), \mathbf{w}^{k+1}-\mathbf{w}^k\rangle + \frac{L_F}{2}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2\\
=&\langle g_k(\hat{\mathbf{w}}^k), \mathbf{w}^{k+1}-\mathbf{w}^k\rangle + \frac{L_F}{2}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2\\
&+\langle g_k(\mathbf{w}^k) - g_k(\hat{\mathbf{w}}^k), \mathbf{w}^{k+1}-\mathbf{w}^k\rangle\\
\le&\left(\frac{L_F}{2} - \frac{1}{2\eta}\right)\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2 + \underbrace{\langle g_k(\mathbf{w}^k) - g_k(\hat{\mathbf{w}}^k), \mathbf{w}^{k+1}-\mathbf{w}^k\rangle}_{=:T_1}.
\end{aligned} \qquad (11)$$

where the last inequality uses (10). Introduce $\alpha > 0, \beta > 0$, we obtain:

$$\begin{aligned}
&T_1\\
\le&\frac{1}{2\alpha}\|(\mathbf{w}^{k+1}-\mathbf{w}^k)_{S^k(t_k)}\|^2 + \frac{1}{2\beta}\|(\mathbf{w}^{k+1}-\mathbf{w}^k)_{j_k}\|^2 +\\
&\underbrace{\frac{\alpha}{2}\|\frac{1}{m}\nabla_{S^k(t_k)}f(\mathbf{w}^k) - \frac{1}{m}\nabla_{S^k(t_k)}f(\hat{\mathbf{w}}^k)\|^2 + \frac{\beta}{2}\|\nabla_{j_k}f(\mathbf{w}^k) - \nabla_{j_k}f(\hat{\mathbf{w}}^k)\|^2}_{=:T_2}
\end{aligned}$$

We use U_k to denote the union of $S^k(t_k)$: $U_k = \cup_{t=1}^{q}S^k(t_k)$. Then we take the expectation of T_2:

$$\begin{aligned}
&\mathbb{E}(T_2)\\
\le&\frac{\alpha}{2m^2}\mathbb{E}\|\nabla_{S^k(t_k)}f(\mathbf{w}^k) - \nabla_{S^k(t_k)}f(\hat{\mathbf{w}}^k)\|^2
\end{aligned}$$

$$
\begin{aligned}
&+\frac{\beta}{2(p-s)}\mathbb{E}\|\nabla_{\overline{S^k(t_k)}}f(\mathbf{w}^k)-\nabla_{\overline{S^k(t_k)}}f(\hat{\mathbf{w}}^k)\|^2\\
\le&\frac{\alpha}{2m^2q}\mathbb{E}\|\nabla_{U_k}f(\mathbf{w}^k)-\nabla_{U_k}f(\hat{\mathbf{w}}^k)\|^2\\
&+\frac{\beta}{2(p-s)}\mathbb{E}\|\nabla_{\overline{U_k}}f(\mathbf{w}^k)-\nabla_{\overline{U_k}}f(\hat{\mathbf{w}}^k)\|^2\\
\le&\frac{\alpha}{2m^2q}L_f^2\mathbb{E}\|\mathbf{w}^k-\hat{\mathbf{w}}^k\|^2+\frac{\beta}{2(p-s)}L_f^2\mathbb{E}\|\mathbf{w}^k-\hat{\mathbf{w}}^k\|^2\\
\le&\frac{\alpha}{2m^2q}L_f^2\mathbb{E}\|\mathbf{w}^k-\hat{\mathbf{w}}^k\|^2+\frac{\beta}{2(p-s)}L_f^2\mathbb{E}\|\mathbf{w}^k-\hat{\mathbf{w}}^k\|^2\\
=&\left(\frac{\alpha L_f^2}{2m^2q}+\frac{\beta L_f^2}{2(p-s)}\right)\mathbb{E}\|\sum_{l\in\tau(k)}(\mathbf{w}^{l+1}-\mathbf{w}^l)\|^2\\
\le&\left(\frac{\alpha L_f^2}{2m^2q}+\frac{\beta L_f^2}{2(p-s)}\right)\tau_{\max}\mathbb{E}\sum_{l\in\tau(k)}\|\mathbf{w}^{l+1}-\mathbf{w}^l\|^2\\
\le&\left(\frac{\alpha L_f^2}{2m^2q}+\frac{\beta L_f^2}{2(p-s)}\right)\tau_{\max}\mathbb{E}\sum_{l=k-\tau_{\max}}^{k-1}\|\mathbf{w}^{l+1}-\mathbf{w}^l\|^2
\end{aligned}
$$

where the second inequality comes from (7) in Assumption 1. Combine with equation (11) and choose $\alpha=\beta$ to be $1/L_F$, then

$$
\begin{aligned}
&f(\mathbf{w}^{k+1})-f(\mathbf{w}^k)\\
\le&-\left(\frac{1}{2\eta}-L_F\right)\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2+T_2.
\end{aligned}
$$

So we have

$$
\begin{aligned}
&\mathbb{E}\left(f(\mathbf{w}^{k+1})-f(\mathbf{w}^k)\right)\\
\le&-\left(\frac{1}{2\eta}-L_F\right)\mathbb{E}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2\\
&+\left(\frac{L_f^2}{2m^2q}+\frac{L_f^2}{2(p-s)}\right)\frac{\tau_{\max}}{L_F}\mathbb{E}\sum_{l=k-\tau_{\max}}^{k-1}\|\mathbf{w}^{l+1}-\mathbf{w}^l\|^2.
\end{aligned}
\tag{12}
$$

Summing (12) from $k=1$ to K, we get

$$
\begin{aligned}
&\mathbb{E}\left(f(\mathbf{w}^{K+1})-f(\mathbf{w}^1)\right)\\
\le&-\left(\frac{1}{2\eta}-L_F\right)\sum_{k=1}^{K}\mathbb{E}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2\\
&+\left(\frac{L_f^2}{2m^2q}+\frac{L_f^2}{2(p-s)}\right)\frac{\tau_{\max}}{L_F}\sum_{k=1}^{K}\sum_{l=k-\tau_{\max}}^{k-1}\mathbb{E}\|\mathbf{w}^{l+1}-\mathbf{w}^l\|^2\\
\le&-\left(\frac{1}{2\eta}-L_F\right)\sum_{k=1}^{K}\mathbb{E}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2\\
&+\left(\frac{L_f^2}{2m^2q}+\frac{L_f^2}{2(p-s)}\right)\frac{\tau_{\max}^2}{L_F}\sum_{k=1}^{K}\mathbb{E}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2\\
=&-\left(\frac{1}{2\eta}-L_F-\left(\frac{1}{2m^2q}+\frac{1}{2(p-s)}\right)\frac{L_f^2}{L_F}\tau_{\max}^2\right)\\
&\sum_{k=1}^{K}\mathbb{E}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2.
\end{aligned}
$$

If the condition

$$
\left(\frac{1}{2\eta}-L_F-\left(\frac{1}{2m^2q}+\frac{1}{2(p-s)}\right)\frac{L_f^2}{L_F}\tau_{\max}^2\right)>0
$$

is satisfied, that is,

$$
\eta<\frac{1}{2\left(L_F+\left(\frac{1}{2m^2q}+\frac{1}{2(p-s)}\right)\frac{L_f^2}{L_F}\tau_{\max}^2\right)}.
$$

Finally, we get

$$
\begin{aligned}
&\frac{1}{\eta^2K}\sum_{k=1}^{K}\mathbb{E}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|^2\\
\le&\frac{f(\mathbf{w}^1)-\mathbb{E}f(\mathbf{w}^{K+1})}{\left(\frac{1}{2\eta}-L_F-\left(\frac{1}{2m^2q}+\frac{1}{2(p-s)}\right)\frac{L_f^2}{L_F}\tau_{\max}^2\right)K\eta^2}.
\end{aligned}
$$

It completes the proof.

A.3 Proof of Theorem 3

Let us prove a lemma first.

LEMMA 4. *We have*

$$
f(\bar{\mathbf{w}})\ge f(\mathbf{w}^k)-\frac{s}{2\rho_-(s)}\|\nabla f(\mathbf{w}^k)\|_\infty^2,\forall\bar{\mathbf{w}},\mathbf{w}^k\in\Omega(s)
$$

PROOF. From the restricted strong convexity, i.e. Assumption 3, we have:

$$
\begin{aligned}
f(\bar{\mathbf{w}})-f(\mathbf{w}^k)\ge&\langle\nabla f(\mathbf{w}^k),\bar{\mathbf{w}}-\mathbf{w}^k\rangle+\frac{\rho_-(s)}{2}\|\bar{\mathbf{w}}-\mathbf{w}^k\|^2\\
\ge&\min_{\mathrm{supp}(\mathbf{w})\subseteq\bar{\Omega}}\langle\nabla f(\mathbf{w}^k),\mathbf{w}-\mathbf{w}^k\rangle+\frac{\rho_-(s)}{2}\|\mathbf{w}-\mathbf{w}^k\|^2\\
=&-\frac{1}{2\rho_-(s)}\|[\nabla f(\mathbf{w}^k)]_{\bar{\Omega}}\|^2\\
\ge&-\frac{|\bar{\Omega}|}{2\rho_-(s)}\|[\nabla f(\mathbf{w}^k)]_{\bar{\Omega}}\|_\infty^2\ge-\frac{s}{2\rho_-(s)}\|\nabla f(\mathbf{w}^k)\|_\infty^2
\end{aligned}
$$

where $\bar{\Omega}=\mathrm{supp}(\bar{\mathbf{w}})$ It completes the proof. □

Then we are ready to prove Theorem 3.

PROOF. From Lemma 4, we have:

$$
f(\mathbf{w}^\infty)-f(\mathbf{w}^*)\le\frac{s}{2\rho_-(s)}\|\nabla f(\mathbf{w}^\infty)\|_\infty^2\le\frac{s}{2\rho_-(s)}(\frac{1}{\eta}\mathbf{w}_{min}^\infty)^2
$$

where $|\mathbf{w}_{min}^\infty|=\min_i|\mathbf{w}_i^\infty|$.

From Assumption 3, we also have:

$$
\begin{aligned}
&\frac{\rho_-(s)}{2}\|\mathbf{w}^\infty\|^2\le f(\mathbf{0})-f(\mathbf{w}^\infty)-\langle\nabla f(\mathbf{w}^\infty)(\mathbf{0}-\mathbf{w}^\infty)\rangle\\
&\|\mathbf{w}^\infty\|^2\le\frac{2\left(f(\mathbf{0})-f(\mathbf{w}^\infty)\right)}{\rho_-(s)}\Rightarrow(\mathbf{w}_{min}^\infty)^2\le\frac{2\left(f(\mathbf{0})-f(\mathbf{w}^\infty)\right)}{s\rho_-(s)}
\end{aligned}
$$

then we have

$$
\begin{aligned}
f(\mathbf{w}^\infty)-f(\mathbf{w}^*)\le&\frac{s}{2\rho_-(s)}\frac{1}{\eta^2}\frac{2}{s\rho_-(s)}\left(f(\mathbf{0})-f(\mathbf{w}^\infty)\right)\\
\le&\alpha\left(f(\mathbf{0})-f(\mathbf{w}^\infty)\right)\\
\le&\alpha\left(f(\mathbf{0})-f(\mathbf{w}^*)+f(\mathbf{w}^*)-f(\mathbf{w}^\infty)\right)\\
(1+\alpha)\left(f(\mathbf{w}^\infty)-f(\mathbf{w}^*)\right)\le&\alpha\left(f(\mathbf{0})-f(\mathbf{w}^*)\right)\\
f(\mathbf{w}^\infty)-f(\mathbf{w}^*)\le&\frac{\alpha}{1+\alpha}\left(f(\mathbf{0})-f(\mathbf{w}^*)\right).
\end{aligned}
$$

It completes the proof. □