Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Approximate Conditional Gradient Descent on Multi-Class Classification

Zhuanghua Liu, Ivor Tsang
Centre for Artifical Intelligence
University of Technology Sydney
liuzhuanghua1991@gmail.com Ivor.Tsang@uts.edu.au

Abstract

Conditional gradient descent, aka the Frank-Wolfe algorithm, regains popularity in recent years. The key advantage of Frank-Wolfe is that at each step the expensive projection is replaced with a much more efficient linear optimization step. Similar to gradient descent, the loss function of Frank-Wolfe scales with the data size. Training on big data poses a challenge for researchers. Recently, stochastic Frank-Wolfe methods have been proposed to solve the problem, but they do not perform well in practice. In this work, we study the problem of approximating the Frank-Wolfe algorithm on the large-scale multi-class classification problem which is a typical application of the Frank-Wolfe algorithm. We present a simple but effective method employing internal structure of data to approximate Frank-Wolfe on the large-scale multi-class classification problem. Empirical results verify that our method outperforms the state-of-the-art stochastic projection-free methods.

Introduction

The Frank-Wolfe (FW) optimization algorithm (Frank and Wolfe 1956)(Jaggi 2013) is a popular first-order method to solve the optimization problem in the form of:

$$\min_{w \in \mathcal{M}} F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) \tag{1}$$

where F is a continuously differentiable function over the domain $\mathcal{M}$ which is convex and compact. Frank-Wolfe resurges in recent years thanks to the cheap linear optimization step when applied on the problem with complex structure(Lacoste-Julien et al. 2013)(Osokin et al. 2016). Multi-class classification with bounded matrix trace-norm is one typical application which Frank-Wolfe can be applied on, detailed discussion can be viewed from (Jaggi 2013).

When faced with large-scale optimization problem, similar to batch gradient descent, gradient calculation and linear optimization step of Frank-Wolfe are prohibitively expensive. Recently, stochastic Frank-Wolfe optimization methods(Hazan and Kale 2012)(Lan and Zhou 2016)(Hazan and Luo 2016) have been proposed to conquer the big data challenge. Extensive experiments show that stochastic Frank-Wolfe optimization methods do not perform as desired.

In this work, we propose a novel method of approximating Frank-Wolfe on multi-class classification problem on large-scale datasets. The method employs both the advantage of stochastic Frank-Wolfe and full Frank-Wolfe. We give the definition of multi-class classification problem below:

Given a set of n training data (x_i, y_i), here $x_i \in \mathcal{R}^d$ is a vector of feature and $y_i \in \{1, \ldots, h\}$ is the label. Previous work (Dudík, Harchaoui, and Malick 2012)(Zhang, Yu, and Schuurmans 2012) found that the multi-class classification problem can be cast as minimizing the objective (1) with f_i defined as:

$$f_i(w) = \log(1 + \sum_{l \neq y_i} \exp(w_l^T x_i - w_{y_i}^T x_i)) \tag{2}$$

on the constrained convex set $\mathcal{M} = \{w \in \mathcal{R}^{h \times m} : \|w\|_* \leq \tau\}$ where $\| \cdot \|_*$ is the nuclear norm.

When optimizing a multi-class classification problem on large-scale datasets, the cost of computation at each step is proportional to the size of dataset. Thus our method uses a subset of data points as the surrogates of the whole dataset at each iteration. Unlike the stochastic optimization methods using random data points at each step, we pick up surrogate data points by taking the internal structure of data into consideration. Specifically, inspired by the Locality-Sensitive Hashing (Indyk and Motwani 1998)(Datar et al. 2004)(Kulis and Grauman 2012) that close data points collide in the same bin, we consider using low cost hash method to map training data from each class into bins, and select one surrogate data point from each bin as the approximation of data points collide in the same bin. We gradually divide more bins to select more surrogates to obtain a better approximation as iteration proceeds.

The convergence rate of our algorithm is proved to be sublinear on multi-class classification objective, the same as the full Frank-Wolfe. Experiments verify that our method converge fast as stochastic Frank-Wolfe methods while better and more stable objective values are attained.

The rest of the paper is organized as follows: Section 2 introduces the background of Frank-Wolfe and its extensions, we present our algorithm and analysis in detail in Section 3, followed by the experiments in Section 4. We conclude our work in Section 5.

Preliminaries and Background

Frank Wolfe Algorithm

At iteration t, same as batch gradient descent, Frank-Wolfe algorithm computes the gradient of F at current w_t as $\nabla F(w_t)$. Before it updates the weight, Frank-Wolfe solve a linear minimization subproblem, i.e., $u_t = \min_{u \in \mathcal{M}} \nabla F(w)^T u$. Then the weight is updated as:

$$w_{t+1} = (1 - \gamma_t) w_t + \gamma_t u_t \tag{3}$$

Without the need of projection onto the constraint set $\mathcal{M}$, γ_t is set to be $\frac{2}{t+1}$. In the case the linear minimization subproblem is cheaper than projection, Frank-Wolfe is much faster than projected gradient descent. For multi-class classification objective, the complexity of linear optimization step is linear to the number of non-zeros in the weight matrix while projection takes the complexity of $\mathcal{O}(hd \min(h, d))$.

Complexity of training with Frank-Wolfe is proportional to the data size n. When n is large, each training iteration is expensive. To conquer the challenge of big data, researchers employ stochastic optimization methods used in optimizing batch gradient descent on the Frank-Wolfe algorithm. Oracle complexity of stochastic gradients of latest work on smooth objective are summarized in the following table:

Measures	SCGS	SFW	SVRF	STORC
Complexity	$\mathcal{O}(\frac{1}{\epsilon^2})$	$\mathcal{O}(\frac{1}{\epsilon^3})$	$\mathcal{O}(\frac{1}{\epsilon^2})$	$\mathcal{O}(\frac{1}{\epsilon^{1.5}})$

Table 1: Oracle Complexity Comparison

For SFW, SVRF and STORC, total complexity of optimizing the objective function consists of three parts: the complexity of exact gradient, stochastic gradient and linear optimization. Fortunately, all the three alternatives share the same complexity of linear optimization. SVRF and STORC have the same complexity of exact gradient while no exact gradient is needed to evaluate in SFW. But, as pointed out in the experiment section of (Hazan and Luo 2016), STORC performs worst compared with SFW and SVRF while STORC enjoys the best rate of convergence of $\mathcal{O}(\frac{1}{\epsilon^{1.5}})$. Note that in the work of (Lan and Zhou 2016), although SCGS needs $\mathcal{O}(\frac{1}{\epsilon^2})$ for stochatic gradient evaluations and no full gradient evaluation at all, in practice it performs inferior to SVRF and STORC.

Locality-Sensitive Hashing

Locality-Sensitive Hashing(LSH) is a popular method for nearest neighbour search(Indyk and Motwani 1998)(Charikar 2002). It admits a provable sub-linear query time and sub-quadratic space complexity, and enjoys good performance on a wide range of applications(Andoni et al. 2015). Specifically, LSH supports fast retrieval of near neighbours with low time and space cost.

Assumptions in Our Analysis

For objective $f_i(w)$, we assume it is convex and L-smooth, so f_i satisfied the following two assumptions. for any $w, w_0 \in \mathcal{M}$, one has:

$$f_i(w) - f_i(w_0) \leq \nabla f_i(w)^T (w - w_0) \tag{4}$$

$$f_i(w) - f_i(w_0) - \nabla f(w_0)^T (w - w_0) \leq \frac{L}{2} \|w - w_0\|^2 \tag{5}$$

L is the Lipstchiz constant of $\nabla f_i(w)$. Multi-class classification objective (2) satisfies the above two properties.

We assume the convex compact domain $\mathcal{M}$ with diameter of D, i.e., for any $w, w_0 \in \mathcal{M}$, $\|w - w_0\| \leq D$. In the setting of multi-class classification, $D = 2\tau$.

Approximate Frank-Wolfe

Limitation of Stochastic Frank-Wolfe Methods

Despite the promising theoretical results of stochastic Frank-Wolfe methods, these stochastic methods pose two major limitations when evaluated in practice.

First, the empirical performance does not conform to their theoretical claim. As pointed out in the previous section, STORC performs worse than SVRF and SFW although STORC has the best rate of convergence.

Another drawback is variance cannot be completely reduced for SVRF in practice. Extensive experiments show that on some datasets, SVRF has large variance when the objective is close to the optimum. We defer detailed comparison in the experiment part.

One possible explanation for such limitations is that stochastic Frank-Wolfe algorithms not only suffer from the randomness, the weight update step (3) also leads to large fluctuation in practice. When objective is near the optimum, the combination of these two factors make stochastic methods not converge well.

Motivation and Explanation of AFW

When the objective loss starts to converge, the loss function should be as close to Equation (1) as possible so that the weight update step (3) will generate similar result as the loss function of the full Frank-Wolfe. Sample a random subset of $f_i(w)$ can not have any guarantee on the quality of approximation although the loss converges fast at the beginning of the training phase. On the other hand, training full Frank-Wolfe on the whole dataset is extremely slow on large-scale datasets but it obtain better and more stable optimum.

Can we combine both the advantage of fast convergence of stochastic methods at the first a few iterations and that of the stable/low-variance convergence of the full Frank-Wolfe when near the optimum? It motivates the development of our approximation method.

We consider the internal structure of data, and use a subset of data points to represent the whole dataset. Specifically, we consider that using Locality-Sensitive Hashing techniques to hash data to different bins. We select one data point from each bin to represent the other points in the same bin. We use coarse hash partition of the dataset to select few surrogates and we can have fast convergence at the beginning of training. When the objective starts to converge, more finer partition is used to guarantee our method will converge to the optimum achieved by full Frank-Wolfe.

Algorithm 1 Approximate Frank-Wolfe

```
1: Inputs: $\{x_i, y_i\}_{i=1}^n, \nu, \delta, k_0$
2: Generate $\rho \in \mathcal{R}^d$ whose element is drawn from Gaussian Distribution $\sim \mathcal{N}(0,1)$
3: Set hash value $h_i \leftarrow \rho^T x_i$
4: Group the data points by $y_i$
5: For each group, sort $h_i$ and partition $x_i$ into $k_0$ bins according to $h_i$
6: for $t = 0, 1, 2, \dots,$ do
7:     if Bin partition has been changed then
8:         Sample one data point from each bin, compute the corresponding $\nabla f_i(w^{(t)})$ of the sampled data
9:     end if
10:    Compute $\nabla \tilde{f}(w^{(t)})$ as the weighted sum of $\nabla f_i(w^{(t)})$
11:    Set $u_t = \min_{v \in \mathcal{M}} \nabla \tilde{f}(w^{(t)})^T v$
12:    Update $w^{(t+1)} \leftarrow (1-\gamma_t) w^{(t)} + \gamma_t u_t$
13:    if $t$ mod $\nu = 0$ then
14:        For each bin containing more than $\delta$ points, divide bin into two equal-width bins.
15:    end if
16: end for
```

We present our algorithm as follows:

Our algorithm is stated in Algorithm 1. The key idea of Algorithm 1 is to obtain $\tilde{f}(w)$, which is the weighted sum of $f_i(w)$ from each hash bin from every class and we use this objective as the approximation of the objective of full Frank-Wolfe. When bin width is large, some data points in the same bin have long distance. So the surrogate cannot represent all the data in the same bin effectively. When objective starts to converge, we narrow the bin width to refine the hash partition and take more data points to get a more reliable approximation. But having such finer partition at the beginning of the training phase is undesirable, as the number of selected surrogates is large, training on this objective is expensive. At the initial phase of training, we only need a coarse partition of $f_i(w)$.

Initialization: Line 2-3 calculates hash value for each data point. The generation of random Gaussian variables are inspired by p-stable distribution, which we will analysis in depth later. For now, we can imagine $\langle \rho, x_i \rangle$ for each data x_i as a series of parallel hyperplane passing each data point from the training set.

Partition: In line 4-5, In every class group in which all data points share the same y_i, we divide the group evenly to bins according to their hash values. It is like that all the hyperplanes $\langle \rho, x_i \rangle$ in the same group can be divided by a set of parallel equi-distance hyperplanes.

Weight update: Line 7-8 samples data from each bin in each class group. Each weighted sum term is obtained from the product of number of data points collided in the hash bin multiplied by the surrogate data point. Line 10-12 employs Frank-Wolfe algorithm to do the weight update. γ_t is set as $\frac{2}{t+1}$, same as the learning rate used in full Frank-Wolfe algorithm.

Partition refinement: The "coarser to finer" bin partition scheme is achieved from line 13 to 16. After the algorithm runs for every ν iterations, it is closer to the optimum. Then we change the partition of the bin by divide each bin into two equi-length bins. It can be imagined as a new perpendicular bisector hyperplane is added between the two neighbour hyperplanes.

Our work is related to some prior works (Allen Zhu, Yuan, and Sridharan 2016)(Zhao and Zhang 2014). These works consider exploiting internal structure on SGD. Our method is different from these works in mainly two aspects: (1) we use hashing instead of clustering as it takes less preprocessing time before training. (2) once hash partition is done, our method is no longer stochastic while their methods are based on stochastic after clustering.

Analysis of the Approximation Error

The hash partition scheme is inspired by p-stable distribution which is widely used in locality-sensitive hashing. We give the definition of p-stable distribution as follows:

Definition 1. (Datar et al. 2004) A distribution $\mathcal{D}$ over $\mathcal{R}$ is called *p-stable*, if there exists $p \geq 0$ such that for any n real numbers $v_1 \dots v_n$ and i.i.d. variables $X_1 \dots X_n$ with distribution $\mathcal{D}$, the random variable $\sum_i v_i X_i$ has the same distribution as the variable $(\sum_i |v_i|^p)^{(1/p)} X$, where X is a random variable with distribution $\mathcal{D}$. In particular, a *Gaussian distribution* $\mathcal{D}_G$, defined by the density function $g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$, is 2-stable.

For any p-stable distribution, the probability of any two data points collide in the same bin enjoys the following property:

Lemma 2. For any two data points $\{x_i, y_i\}, \{x_j, y_j\}$, let $c = \|x_i - x_j\|$, $f_p(t)$ be the probability distribution function of the *absolute value* of the p-stable distribution, and bin width $r \in \mathcal{R}$, then the probability of two data points under the mapping of the p-stable distribution collide in the same bin is

$$Pr[h(i) = h(j)] = \int_0^r \frac{1}{c} f_p(\frac{t}{c})(1 - \frac{t}{r}) dt \qquad (6)$$

$h(\cdot)$ is the hash value for the instance. A direct result for Gaussian distribution following the previous lemma is expressed as:

Lemma 3. The probability of any two data points under the mapping of the Gaussian distribution collide in the same bin is bounded by the Gaussian integral $\frac{2}{\sqrt{\pi}} \int_0^{\frac{r}{\sqrt{2}c}} e^{-x^2} dx$

Proof. Plug $f_p(x) = 2 * \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ into (6):

$$
\begin{aligned}
Pr[h(i)=h(j)] &= 2*\int_0^r \frac{1}{c}\frac{1}{\sqrt{2\pi}}e^{-(\frac{t}{c})^2/2}(1-\frac{t}{r})dt\\
&= \frac{2}{\sqrt{\pi}}(\int_0^r e^{-\frac{t^2}{2c^2}}\,d\frac{t}{\sqrt{2}c}-\frac{1}{\sqrt{2}cr}\int_0^r e^{-\frac{t^2}{2c^2}}t\,dt)\\
&\le \frac{2}{\sqrt{\pi}}\int_0^r e^{-\frac{t^2}{2c^2}}\,d\frac{t}{\sqrt{2}c}\\
&= \frac{2}{\sqrt{\pi}}\int_0^{\frac{r}{\sqrt{2}c}} e^{-x^2}dx
\end{aligned}
$$

□

Remark. When $c \to 0$, the probability of collision is almost 1, which means when two data points are close to each other, they will collide in the same bin with probability of 1. On the other hand, when $c \to +\infty$, $\frac{r}{\sqrt{2}c} \to 0$, the integral approaches 0. That means, those two data points with distance c are impossible to collide in the same bin. The Gaussian integral bound gives us the intuition the farther two points, they will be hashed into the same with less probability. Furthermore, r is one of the key factor to decide to whether two data points collide in the same bin. if $c \gg r$, the two data points are unlikely to be hashed into same bin.

Lemma 4. (Bouchard 2007) (Quadratic upper bound for softmax function) For any $x \in \mathcal{R}^d$ and for any $y \in \mathcal{R}^d$, one has

$$
\begin{aligned}
\log\sum_{i=1}^d e^{x_i} \le \sum_{i=1}^d (x_i-y_i)^2 - \frac{1}{d}(\sum_{i=1}^d (x_i-y_i))^2 + \\
\sum_{i=1}^d \frac{(x_i-y_i)e^{y_i}}{\sum_{i'=1}^d e^{y_{i'}}} + \log\sum_{i=1}^d e^{y_i}
\end{aligned}
\tag{7}
$$

Given two data points with distance d, the difference of their multi-class loss objective can be bounded as:

Lemma 5. Given two data points within the same class group with distance q, i.e., $y_i = y_j$ and $\|x_i - x_j\| = q$, then $f_i(w) - f_j(w) \le \sum_{l\neq y_i}\|w_l - w_{y_i}\|^2 q^2 + \sum_{l\neq y_i}\frac{q\|w_l-w_{y_i}\|\exp(w_l^T x_j - w_{y_i}^T x_j)}{\sum_{l\neq y_i} exp(w_l^T x_j - w_{y_i}^T x_j)}$

Proof. Thanks to the upper bound of Lemma 4, one has

$$
\begin{aligned}
f_i(w)-f_j(w) &= log(1+\sum_{l\neq y_i}\exp(w_l^T x_i - w_{y_i}^T x_i)) -\\
&\quad log(1+\sum_{l\neq y_j}\exp(w_l^T x_j - w_{y_j}^T x_j))\\
&\le \sum_{l\neq y_i}((w_l - w_{y_i})^T(x_i-x_j))^2 +\\
&\quad \sum_{l\neq y_i}\frac{(w_l-w_{y_i})^T(x_i-x_j)exp(w_l^T x_j - w_{y_i}^T x_j)}{\sum_{l\neq y_i} exp(w_l^T x_j - w_{y_i}^T x_j)}\\
&\le \sum_{l\neq y_i}\|w_l - w_{y_i}\|^2\|x_i-x_j\|^2 +\\
&\quad \sum_{l\neq y_i}\frac{\|w_l-w_{y_i}\|\|x_i-x_j\|exp(w_l^T x_j - w_{y_i}^T x_j)}{\sum_{l\neq y_i} exp(w_l^T x_j - w_{y_i}^T x_j)}
\end{aligned}
$$

Replace $\|x_i - x_j\|$ with q, we get the desired upper bound. □

Previous lemma shows that right hand side of the upper bound is proportional to q, the distance of data points. So when two data point within the same bin is close to each other, the objective loss of one data point can be approximated by that of another data point. Put previous lemma and Lemma 3 together, if the bin width r is small enough, the loss objective of one data point of the hash bin can approximate another point in the same bin quite well.

The $\mathcal{O}(1/t)$ convergence rate of Frank-Wolfe algorithm on L-smooth objective is given by the following lemma:

Lemma 6. (Bubeck 2015) Let f be a convex and L-smooth function, and $\gamma_t = \frac{2}{t+1}$ for $t \ge 1$. Then for any $t \ge 2$, one has

$$
f(w_t) - f(w^*) \le \frac{2LD^2}{t+1} \tag{8}
$$

For $\tilde{f}(\cdot)$ as a weighted sum of f_i, Lemma 6 can be applied on $\tilde{f}$ because it is convex and L-smooth. Similar to w^* is the optimum of the $f(w)$, we assume $\tilde{w}^*$ as the optimum of function $\tilde{f}(w)$. For any $w \in \mathcal{M}$, one has $\tilde{f}(\tilde{w}^*) \le \tilde{f}(w)$.

Given any $\tilde{f}(\cdot)$, when $\tilde{f}(w)$ runs for t iterations, can we give the bound between $\tilde{f}(w)$ and optimum objective loss of the full Frank-Wolfe $F(w^*)$? Our next theorem answers this question:

Theorem 7. *Assume $\tilde{f}(w)$ is a weighted sum of f_i from each hash bins. For any hash bin in current partition of data points, the distance of each surrogate with the other data points in the same hash bin does not exceed B. Furthermore, set $\mathcal{M}$ satisfy condition that for any weight matrix w defined on set $\mathcal{M}$, for any $i, j \in \{1, \dots, h\}$, $\|w_i - w_j\| \le D'$. Then the following holds true:*

$$
\begin{aligned}
\tilde{f}(w_t) - f(w^*) \le \frac{2LD^2}{t+1} + n(h-1)D'^2B^2\\
+ n(h-1)D'B
\end{aligned}
\tag{9}
$$

Proof. Combine the results of Lemma 5 and Lemma 6, one has

$$
\begin{aligned}
\tilde{f}(w_t) - f(w^*) &= \tilde{f}(w_t) - \tilde{f}(\tilde{w}^*) + \tilde{f}(\tilde{w}^*) - f(w^*)\\
&\le \tilde{f}(w_t) - \tilde{f}(\tilde{w}^*) + \tilde{f}(w^*) - f(w^*)\\
&\le \underbrace{\frac{2LD^2}{t+1}}_{\epsilon_{fw}} + \underbrace{n(h-1)D'^2B^2 + n(h-1)D'B}_{\epsilon_h}
\end{aligned}
$$

□

The bound for the theorem consists of two parts: ϵ_{fw} and ϵ_h. The ϵ_{fw} conforms to the convergence rate of the Frank-Wolfe algorithm, while ϵ_h is generated by the approximation in the partitions of the hash bins. ϵ_h is proportional to B. When the bin width r is small, the data points with close distance are likely to be hashed in the same bin, so B is also small. $\tilde{f}(w_t)$ approximates the objective loss of full Frank-Wolfe f.

Rate of Convergence

Finally, we turn to the analysis of the rate of convergence our algorithm. Our algorithm optimize a series of $\tilde{f}(w)$, each $\tilde{f}(w)$ is produced by the partition refine during the training process. Each $\tilde{f}(w)$ is the weighted sum of training surrogates.

Theorem 8. *Approximate Frank-Wolfe takes $\mathcal{O}(1/\epsilon)$ iterations to achieve the ϵ-approximation of the optimum of full Frank-Wolfe.*

Proof. For any $\tilde{f}(w)$, the rate of convergence is given by Lemma 6. The oracle complexity for $\tilde{f}(w)$ is thus $\mathcal{O}(1/\epsilon)$. Because during the whole training process, there are at most n weighted sums $\tilde{f}(w)$. (At the first iteration, we may use only one surrogate f_i to represent the f. More f_i is added into the set of surrogates loss as iteration proceeds. The number of surrogates functions is monotonically increasing.) So the total oracle complexity is no more than $\mathcal{O}(n/\epsilon)$. n is a constant throughout the training progress, so we arrived at the desired convergence rate of our algorithm. $\square$

Experiment

Here we evaluate the convergence of the proposed algorithm on the multi-class classification task. We compare the objective and time of the proposed algorithm with other stochastic Frank-Wolfe methods. In addition, we compare the convergence of AFW with full Frank-Wolfe.

Experiment Setup

Our algorithm and baseline methods are implemented in Matlab. Our experiments run on a server with 3.1 GHZ CPU and 132 GB memory. We conducted our experiment on several large-scale datasets from the libsvm website[1]. The datasets are summarized in the following table:

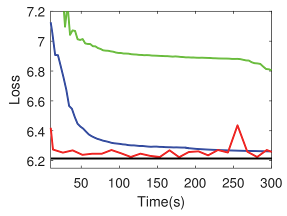
Dataset	aloi	covtype	mnist	poker	sensIT
data	108,000	581,012	60,000	1,000,000	78,823
features	128	54	780	10	50
classes	1,000	7	10	10	3

Table 2: Summary of the datasets

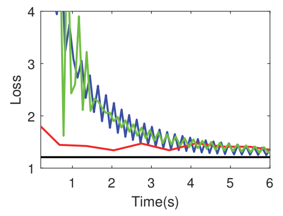
To evaluate the quality of the solution in optimization, we use the multi-class classification objective as the measure. We compare our method with the state-of-the-art stochastic Frank-Wolfe methods, i.e., SVRF, SFW. We also include the optimum achieved by full Frank-Wolfe in the comparison. For fair comparison, we use the default parameter for SFW and SVRF from (Hazan and Luo 2016), i.e., The size of mini-batches at round t is t^2, t for SFW and SVRF, respectively. In the evaluation of approximate Frank Wolfe, we set $\nu = 50$, $\delta = 10$ and $k_0 = 100$.

Objective of Approximate Frank-Wolfe

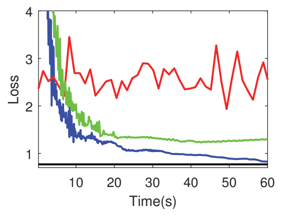
Figure 1 depicts the evaluation of AFW and other baselines with multi-class classification loss residual on the datasets



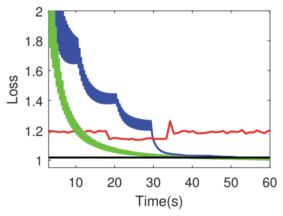
(a) aloi



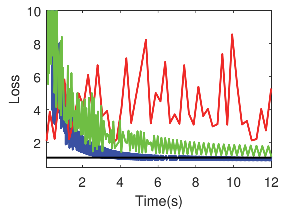
(b) covtype



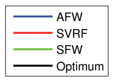
(c) mnist



(d) poker



(e) sens



(f) legend

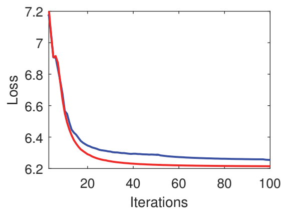
Figure 1: Comparison of AFW and baseline methods

previously mentioned. Note that our evaluation of AFW counts in the time of hash partition during the training process. We do not include full Frank-Wolfe as our baseline since full Frank-Wolfe, though it has the best rate of convergence $\mathcal{O}(1/t)$, it takes too much time to converge because of the expensive evaluation of gradient using the whole dataset at each iteration. Thus we only depicts the optimum obtained by the full Frank-Wolfe in the figure (The optimum is obtained by running full Frank-Wolfe for a very long time).

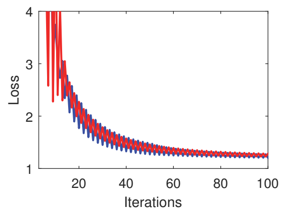
Observe that AFW converges on every dataset, and AFW is the fastest method to reach the optimum on almost every dataset. Since AFW can be divided into multiple epochs. At every epoch, AFW is trained on $\tilde{f}(w)$ which is a fixed weighted sum of $f_i(w)$. So AFW behaves just the same as the full Frank-Wolfe during each epoch since the objective is fixed at that epoch. The objective loss of AFW is more smooth compared with the stochastic Frank-Wolfe methods.

SVRF converges very fast on every dataset at early stage of training. However, when SVRF converges to some points close to the optimum, SVRF sometimes becomes divergence. Unlike their counterpart SVRG, variance reduction on SVRF works not as good as SVRG which has no variance in later iterations. We believe such oscillation is introduced by the weight update step (3). Even full Frank-Wolfe is not a monotonically decreasing function, so random samples of the dataset inevitably introduce variance. The biggest

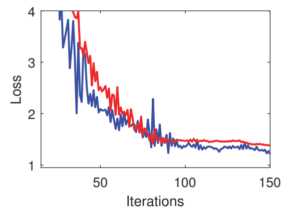
[1]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/



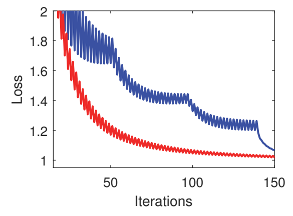
(a) aloi



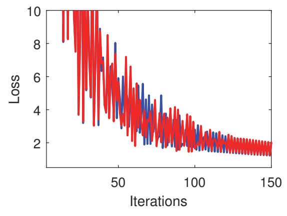
(b) covtype



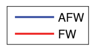
(c) mnist



(d) poker



(e) sens



(f) legend

Figure 2: Comparison of AFW and FW on Convergence

drawback of SVRF is that it can not achieve the global minimum on some datasets, see Figure 1c, 1d and 1e. It oscillates wildly at some point far from the global optimum.

For SFW, although it takes the longest time to converge the optimum, it does not oscillate like SVRF. Thanks in part to the mini-batch size of t^2 at iteration t, its variance is thus reduced by a factor of t^2. So we can see SFW is quite smooth when it is near the optimum.

Convergence of Approximate Frank-Wolfe

We compare the convergence of Approximate Frank-Wolfe and full Frank-Wolfe on the same set of datasets. Except Figure 2d, approximate Frank-Wolfe almost share the same convergence rate as the full Frank-Wolfe in Figure2b, 2c and 2e. Full Frank-Wolfe converges marginally faster than approximate Frank-Wolfe in Figure 2a. It validates our claim that AFW has the same rate of convergence as full FW in the previous sections.

Conclusion

We propose the approximate Frank-Wolfe algorithm on multi-class classification problems. Our method enjoys the advantages from stochastic Frank-Wolfe methods and full Frank-Wolfe. It converges fast at the beginning of training, while it has the same rate of convergence as full Frank-Wolfe method. It outperforms existing stochastic Frank-Wolfe methods on most datasets in the experiment. In the future work, we will extend our approach on more general tasks and demonstrate its effectiveness on a wider range of applications.

Acknowledgments

Ivor W. Tsang is supported by the ARC Future Fellowship FT130100746 and ARC grant LP150100671

References

Allen Zhu, Z.; Yuan, Y.; and Sridharan, K. 2016. Exploiting the structure: Stochastic gradient methods using raw clusters. *CoRR* abs/1602.02151.

Andoni, A.; Indyk, P.; Laarhoven, T.; Razenshteyn, I. P.; and Schmidt, L. 2015. Practical and optimal LSH for angular distance. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 1225–1233.

Bouchard, G. 2007. Efficient bounds for the softmax function and applications to approximate inference in hybrid models. In *Workshop for Approximate Bayesian Inference in Continuous/Hybrid Systems*. Citeseer.

Bubeck, S. 2015. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning* 8(3-4):231–357.

Charikar, M. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, 380–388.

Datar, M.; Immorlica, N.; Indyk, P.; and Mirrokni, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, 253–262. ACM.

Dudík, M.; Harchaoui, Z.; and Malick, J. 2012. Lifted coordinate descent for learning with trace-norm regularization. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, April 21-23, 2012*, 327–336.

Frank, M., and Wolfe, P. 1956. An algorithm for quadratic programming. *Naval research logistics quarterly* 3(1-2):95–110.

Hazan, E., and Kale, S. 2012. Projection-free online learning. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

Hazan, E., and Luo, H. 2016. Variance-reduced and projection-free stochastic optimization. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 1263–1271.

Indyk, P., and Motwani, R. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, 604–613.

Jaggi, M. 2013. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, 427–435.

Kulis, B., and Grauman, K. 2012. Kernelized locality-sensitive hashing. *IEEE Trans. Pattern Anal. Mach. Intell.* 34(6):1092–1104.

Lacoste-Julien, S.; Jaggi, M.; Schmidt, M. W.; and Pletscher, P. 2013. Block-coordinate frank-wolfe optimization for structural svms. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, 53–61.

Lan, G., and Zhou, Y. 2016. Conditional gradient sliding for convex optimization. *SIAM Journal on Optimization* 26(2):1379–1409.

Osokin, A.; Alayrac, J.; Lukasewitz, I.; Dokania, P. K.; and Lacoste-Julien, S. 2016. Minding the gaps for block frank-wolfe optimization of structured svms. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 593–602.

Zhang, X.; Yu, Y.; and Schuurmans, D. 2012. Accelerated training for matrix-norm regularization: A boosting approach. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, 2915–2923.

Zhao, P., and Zhang, T. 2014. Accelerating minibatch stochastic gradient descent using stratified sampling. *CoRR* abs/1405.3080.