Weighted Subset Selection for Fast SVM Training

Sara Mourad, Ahmed Tewfik and Haris Vikalo
Department of Electrical and Computer Engineering
The University of Texas at Austin
saramourad92@gmail.com, tewfik@austin.utexas.edu, hvikalo@ece.utexas.edu

***Abstract*—We propose a data reduction method that improves the speed of training the support vector machine (SVM) algorithm. In particular, we study the problem of finding a weighted subset of training data to efficiently train an SVM while providing performance guarantees. Relying on approximate nearest neighborhood properties, the proposed method selects relevant points and employs the concept of maximal independent set to achieve desired coverage of the training dataset. Performance guarantees are provided, demonstrating that the proposed approach enables faster SVM training with minimal effect on the accuracy. Empirical results demonstrate that the proposed method outperforms existing weighted subset selection techniques for SVM training.**

***Index Terms*—SVM, nearest neighbors, independent set**

I. Introduction

Support vector machine (SVM) classifiers have found wide use due to being theoretically well-motivated and having desirable performance in practice. The SVM learning algorithm essentially solves a quadratic program where the objective is to maximize the width of the linear separation between different data classes. When the objective function incorporates a kernel, SVM can perform nonlinear classification tasks by solving the linear separation problem in higher dimensions. However, SVM suffers from high running time and memory requirements. These limitations become particularly pronounced as the sizes of datasets become very large.

A number of algorithms that attempt to speed up the training of SVM classifiers have been proposed in literature. The Sequential Minimal Optimization (SMO) [1] algorithm focuses on the dual problem and employs an active set of constraints to select a subset of the dual variables. These algorithms converge slowly and have time complexity that is super linear in the size of the data. Approximate algorithms include LASVM [2], an online version of the SMO algorithm that iteratively constructs the set of support vectors over a number of epochs, and then applies the SMO algorithm on the constructed set of support vectors. The computational complexity of LASVM is between $O(n)$ and $O(n^3)$ depending on the size of the selected support vectors, where n is the size of the data. Another approximate method is Pegasos [3], a stochastic subgradient descent algorithm that solves the primal optimization problem. Its running time for non linear kernels is $O(n/\lambda\epsilon)$, where λ is the regularization parameter of SVM and ϵ is the accuracy of the solution. For linear kernels, its runtime is $O(d/\lambda\epsilon)$ where d is the bound on the number of non zero features in each example. Note that when the linear kernel is used, LIBlinear [4], an online algorithm which employs dual coordinate descent, has linear running time. Cutting plane approaches such as SVM-perf [5] find a solution with accuracy ϵ in time $O(nd/\lambda\epsilon^2)$; this was further improved to $O(nd/\lambda\epsilon)$.

Data reduction, also referred to as instance selection, takes a different approach to scaling machine learning algorithms to large scale data. Among the instance selection methods that have been developed for training SVMs, the nearest neighbor SVM (NNSVM) [6] algorithm attempts to select points that are close to the boundary of the classifier by searching for the nearest point to each point in the dataset, and removing the points whose nearest neighbor is from the opposite class. However, the NNSVM algorithm is impractical for large datasets. In sampled SVM (SSVM) [7] and reduced SVM (RSVM) [8], random sampling methods are employed to reduce the size of the training set. Furthermore, in an attempt to enhance generalization, the authors of [9] proposed an algorithm that performs k-means clustering and retains clusters of points from the same class. In [10]–[12], data geometry is exploited to make use of the centroids of classes to reduce the training set size. In [13], the informative vector machine algorithm is proposed to efficiently train sparse Gaussian process classification models. In particular, training is enabled by greedily selecting a subset of the training set such that the conditional entropy of the approximated posterior is minimized [14]. In [15], a coreset selection for SVM training using the core vector machine (CVM) algorithm is proposed. Coresets are small summaries of the dataset such that solutions on the summary are close to solutions on the whole dataset [16]. In [17], a proposed algorithm leverages submodularity to optimize both the relevance and the coverage of a selected subset, based on the approximate nearest neighborhood properties of the dataset.

Relying on the approximate nearest neighborhood properties, in this paper we maximize the training data coverage by constructing a maximal independent set on the k nearest neighbor graph. Furthermore, we provide performance guarantees for the proposed method. Empirical results obtained on two datasets demonstrate that the proposed method achieves competitive running time and near optimal accuracy, generally outperforming existing subset selection algorithms for SVM.

II. Problem Formulation

Let $\mathcal{V} = \{(x_i, y_i)_{i=1}^n\}$ denote a set of n training samples, where $x_i \in X^d$ is a d-dimensional feature vector and y_i is the corresponding binary target. To simplify handling of a bias term, the last dimension of every x_i is equal to 1. We

978-9-0827-9703-9/19/$31.00 ©2019 IEEE

are interested in selecting a small subset $\mathcal{S}$ from $\mathcal{V}$ such that training the SVM on $\mathcal{S}$ rather than $\mathcal{V}$ incurs minimal loss of performance. To this end, we consider minimization of an objective that consists of the average hinge loss evaluated on the training set $\mathcal{V}$ and a regularization term, i.e.,

$$\min_{w} \frac{1}{n} \sum_{i \in \mathcal{V}} \max(0, 1 - y_i(w^T x_i)) + \lambda \|w\|, \tag{1}$$

where x_i, $i = 1, ..., n$, is the input vector and $y_i \in \{-1, 1\}$ is the target variable. We define $Cost(\mathcal{V}, w) = 1/|\mathcal{V}| \sum_{i \in \mathcal{V}} \max(0, 1 - y_i(w^T x_i))$ to be the average hinge loss achieved by a classifier w and $Cost(\mathcal{V}, w^*_{\mathcal{V}}) = 1/|\mathcal{V}| \sum_{i \in \mathcal{V}} \max(0, 1 - y_i(w^{*T}_{\mathcal{V}} x_i))$ to be the average hinge loss achieved by the optimal classifier $w^*_{\mathcal{V}}$, i.e., $w^*_{\mathcal{V}}$ is the solution to optimization problem (1).

We are interested in finding a weighted subset $\mathcal{S}$ (we assign a weight z_i for each element x_i in $\mathcal{S}$) such that

$$\begin{aligned} &\min_{\mathcal{S}} \mid Cost(\mathcal{V}, w^*_{\mathcal{V}}) - Cost(\mathcal{V}, w^*_{\mathcal{S}})| \\ &\text{s.t. } |\mathcal{S}|/|\mathcal{V}| \leq \eta, \end{aligned} \tag{2}$$

where η denotes the fraction of training data retained. η is a real number between 0 and 1; values closer to 0 correspond to significant reductions in the size of the training dataset. $w^*_{\mathcal{S}}$ denotes the hyperplane obtained after training on the selected subset $\mathcal{S}$. Since $\mathcal{S}$ is a weighted subset, $w^*_{\mathcal{S}}$ minimizes $Cost(\mathcal{S}, w) = \frac{1}{\sum_{i \in \mathcal{S}} z_i} \sum_{i \in \mathcal{S}} z_i \max(0, 1 - y_i(w^T x_i))$, a redefined version of the cost for a weighted subset.

In case of non linear problems, the dual form of Eq. 1, which can be expressed in function of the dot products between all pair of samples, uses the dot products of some transformation $\phi(.)$ of the data. Dot products in the transformed space are facilitated by the kernel function, where the kernel function $k(x_i, x_j) = \phi^T(x_i)\phi(x_j)$. The cost function is now expressed in function of $\phi(x_i)$ instead of x_i. And so from now onwards in the paper, we will replace x_i by $\phi(x_i)$ is the cost expression such that for example $Cost(\mathcal{V}, w) = 1/|\mathcal{V}| \sum_{i \in \mathcal{V}} \max(0, 1 - y_i(w^T \phi(x_i)))$.

A. Nearest neighborhood properties

As a preprocessing step for our algorithm, we require to obtain the nearest neighborhood properties of points in the dataset. Constructing the exact k nearest neighborhood graph of the dataset is quadratic in the number of samples, and hence infeasible for large datasets. We compute instead approximate nearest neighbors using locality sensitive hashing (LSH) which uses similarity preserving hash functions. We use the LSH implementation in [18] with sublinear query time and subquadratic space requirements. As in [17], and for each point, we compute the corresponding k nearest neighbors belonging to the same class, as well as the k nearest neighbors belonging to the opposite class. Accordingly, the function $f(\cdot)$ is defined as follows. $f(x_i)$ is equal to 1 if x_i is the k nearest neighbor of any point from the opposite class. Points with $f(\cdot)$ equal to 1 are the points close to the boundary, i.e. the relevant points, which is useful for us as explained later in Sec. III-C.

III. Coverage based on the Maximal independent set

In order to optimize Eq. 2, we observe that the cost depends on the distance between a point and the separating hyperplane. More generally, in case we are using a non linear kernel for the SVM, then the cost depends on the distance between $\phi(x_i)$ and w, where $\phi(\cdot)$ is the nonlinear transformation facilitated by the kernel function (i.e., $k(x_i, x_j) = \phi^T(x_i)\phi(x_j)$). The intuition of our approach is that a weighted point can summarize the contribution of its k nearest neighbors belonging to the same class, where the distance metric used for nearest neighbors should be proportional to $\|\phi(x_i) - \phi(x_j)\|_2$. For both the RBF kernel as well as the linear kernel, the Euclidean distance $D(x_1, x_2) = \|x_1 - x_2\|_2$ is a suitable metric. In order to present the proposed algorithm, we first introduce some concepts.

Definition 1 (Independent set)**.** *An independent set is a set of vertices in a graph such that no pair of vertices are adjacent.*

Definition 2 (Maximal independent set)**.** *A maximal independent set (MIS) is an independent set that is not a subset of any other independent set.*

A maximal independent set is also dominating. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with set of vertices denoted as $\mathcal{V}$ and set of edges denote as $\mathcal{E}$, a dominating set for the graph $\mathcal{G}$ is a subset $\mathcal{D}$ of $\mathcal{V}$ such that every vertex not in $\mathcal{D}$ is adjacent to at least one member of $\mathcal{D}$.

To find a weighted subset $\mathcal{S}$ and its corresponding weight vector z, we search for the maximal independent set given the undirected version of the directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ defined as follows; the set of vertices of the graph corresponds to the set of points in $\mathcal{V}$, and the set of edges denoted by $\mathcal{E}$ is defined as follows; there exists an edge between vertex v_1 and vertex v_2 if the data point corresponding v_2 is a k nearest neighbor from the same class of the data point corresponding to v_1, or vice versa. The number of nearest neighbors k is a parameter that can be adjusted while constructing $\mathcal{G}$ and, consequently, used to control the size of the subset $\mathcal{S}$. As we show in Theorem 1, the lower the value of k, the tighter the performance guarantees; unfortunately, this comes at the cost of increase in the size of the MIS $\mathcal{S}$. In general, the number of nearest neighbors for each class is a hyperparameter that can be tuned.

A. Algorithm

Our sequential algorithm for finding a maximal independent set is formalized in Algorithm 1. The algorithm first constructs the undirected graph described in Sec III after computing the approximate nearest neighbors in each class. The running time of this preprocessing step is sub quadratic in the size of the data. Then we randomly pick a data point v from the total set $\mathcal{V}$, we add the point to the set $\mathcal{S}$, we assign to the point a weight z equal to the number of k nearest neighbors of v existing in $\mathcal{V}$ at the time we selected it, and then we remove the point v and its nearest neighbors in $\mathcal{V}$ from $\mathcal{V}$. We repeat until $\mathcal{V}$ is empty.

Algorithm 1 Greedy weighted subset selection based on MIS

Construct the undirected graph $\mathcal{G}$ from data.
Initialize $\mathcal{S}$ to an empty set.
while $\mathcal{V}$ is not empty: **do**
  Choose randomly a node $v \in \mathcal{V}$
  Add v to the set $\mathcal{S}$
  Assign to node v a weight z equal to the numbers of neighbors of v in $\mathcal{V}$
  Remove from $\mathcal{V}$ the node v and all its neighbors in $\mathcal{V}$.
end while
Return $\mathcal{S}, \mathbf{z}$

The selection algorithm runtime is $O(|\mathcal{E}|) = O(nk)$. We also note that even though our approach doesn't guarantee as specified in Eq. 2 that the selected subset size will be under some fraction of the original dataset size, increasing the number of nearest neighbors k decreases the size of the subset $\mathcal{S}$, at the expense of the tightness of the performance guarantee.

B. Performance guarantees

By constructing a weighted subset $\mathcal{S}$ as an maximal independent set of $G(\mathcal{V}, \mathcal{E})$, the performance guarantees of Theorem 1 are as follows:

Theorem 1. *Let $\mathcal{S}$ denote a maximal independent set constructed by selecting points from set $\mathcal{V}$ as described in Alg.1. Assuming that for any pair of k nearest neighbors x_i and x_j from the same class, there exists $\epsilon > 0$ such that $\|\phi(x_i) - \phi(x_j)\|_2 \leq \epsilon$, then for any hyperplane w, $\|w\|_2 = 1$, it holds that $\mid Cost(\mathcal{V}, w) - Cost(\mathcal{S}, w)\mid \leq \epsilon$.*

Proof.

$$Cost(\mathcal{V}, w) \tag{3}$$
$$= \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \max(0, 1 - y_i w^T \phi(x_i)) \tag{4}$$
$$= \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{S}} \sum_{j \in N'(x_i)} \max(0, 1 - y_j w^T \phi(x_j)) \tag{5}$$
$$= \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{S}} z_i \mathbb{E}[\max(0, 1 - y_{C_i} w^T \phi(x_{C_i}))] \tag{6}$$

where $w^T \phi(x_i)$ is the distance from the sample $\phi(x_i)$ to the separating hyperplane w, $N'(x_i)$ is the set of vertices adjacent to x_i at the time x_i is added to the set $\mathcal{S}$, i.e. $|N'(x_i)| = z_i$ and x_{C_i} is a random variable uniformly distributed on $N'(x_i)$. Now,

$$Cost(\mathcal{S}, w) \tag{7}$$
$$= \frac{1}{\sum_{i \in S} z_i} \sum_{i \in \mathcal{S}} z_i \max(0, 1 - y_i w^T \phi(x_i)) \tag{8}$$
$$= \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{S}} z_i \max(0, 1 - y_i w^T \phi(x_i)). \tag{9}$$

We note that $\sum_{i \in S} z_i = |\mathcal{V}|$ by the MIS construction. We define $D_i = \mathbb{E}[\max(0, 1 - y_{C_i} w^T \phi(x_{C_i}))] - \max(0, 1 - y_i w^T \phi(x_i))$. Therefore, deterioration of the objective due to selecting a subset of points can be bounded as

$$\mid Cost(\mathcal{V}, w) - Cost(\mathcal{S}, w)\mid \tag{10}$$
$$= \mid \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{S}} z_i \mathbb{E}[\max(0, 1 - y_{C_i} w^T \phi(x_{C_i}))] - \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{S}} z_i \max(0, 1 - y_i w^T \phi(x_i))\mid \tag{11}$$
$$= \mid \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{S}} z_i D_i \mid \tag{12}$$
$$\leq \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{S}} z_i \mid D_i \mid . \tag{13}$$

Moreover, the terms in the last summation can be bounded as

$$\mid D_i \mid = \mid \frac{1}{z_i} \sum_{j \in N'(x_i)} \max(0, 1 - y_j w^T \phi(x_j)) - \frac{1}{z_i} \sum_{j \in N'(x_i)} \max(0, 1 - y_i w^T \phi(x_i))\mid \tag{14}$$
$$\leq \frac{1}{z_i} \sum_{j \in N'(x_i)} \mid \max(0, 1 - y_j w^T \phi(x_j)) \tag{15}$$
$$- \max(0, 1 - y_i w^T \phi(x_i)) \mid \tag{16}$$
$$\leq \epsilon \tag{17}$$

where (14) follows since

$$\mathbb{E}[\max(0, 1 - y_{C_i} w^T \phi(x_{C_i}))] = \frac{1}{z_i} \sum_{j \in N'(x_i)} \max(0, 1 - y_j w^T \phi(x_j))$$

and

$$\max(0, 1 - y_i w^T \phi(x_i) = \frac{1}{z_i} \sum_{j \in N'(x_i)} \max(0, 1 - y_i w^T \phi(x_i)) \tag{18}$$

and where (17) follows since

$$\epsilon \geq \|\phi(x_i) - \phi(x_j)\| \tag{19}$$
$$\geq \mid y_i w^T \phi(x_i) - y_j w^T \phi(x_j)\mid \tag{20}$$
$$\geq \mid (1 - y_i w^T \phi(x_i)) - (1 - y_j w^T \phi(x_j))\mid$$
$$\geq \mid \max(0, 1 - y_i w^T \phi(x_i)) - \max(0, 1 - y_j w^T \phi(x_j))\mid \tag{21}$$

In the above, (19) follows from the assumption in Theorem 1, and (20) follows from the Cauchy-Schwarz inequality.

We can now write

$$\mid Cost(\mathcal{V}, w) - Cost(\mathcal{S}, w)\mid = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{S}} z_i \mid D_i \mid$$
$$\leq \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{S}} z_i \epsilon \leq \epsilon \frac{1}{\sum_{i \in S} z_i} \sum_{i \in \mathcal{S}} z_i = \epsilon \tag{22}$$

and hence $|\ Cost(\mathcal{V}, w) - Cost(\mathcal{S}, w)|\ \leq \epsilon$. □

Corollary 1. *Let $w^*_{\mathcal{S}}$ denote the optimal separating hyperplane formed using the weighted subset $\mathcal{S}$, and $w^*_{\mathcal{V}}$ denote the optimal separating hyperplane formed using the points in $\mathcal{V}$. Then $Cost(\mathcal{V}, w^*_{\mathcal{S}}) \leq Cost(\mathcal{V}, w^*_{\mathcal{V}}) + 2\epsilon$.*

Proof.

$$Cost(\mathcal{V}, w^*_{\mathcal{S}}) \leq Cost(\mathcal{S}, w^*_{\mathcal{S}}) + \epsilon \quad (23)$$
$$\leq Cost(\mathcal{S}, w^*_{\mathcal{V}}) + \epsilon \quad (24)$$
$$\leq Cost(\mathcal{V}, w^*_{\mathcal{V}}) + 2\epsilon, \quad (25)$$

where (23) follows from Theorem 1, (24) holds because $w^*_{\mathcal{S}}$ minimizes the SVM objective over $\mathcal{S}$ and thus $Cost(\mathcal{S}, w^*_{\mathcal{S}}) \leq Cost(\mathcal{S}, w^*_{\mathcal{V}})$, and (25) follows from Theorem 1. □

To conclude, in this section we have shown that applying MIS on $\mathcal{V}$ ensures construction of a subset $\mathcal{S}$ such that $Cost(\mathcal{V}, w^*_{\mathcal{S}}) \leq Cost(\mathcal{V}, w^*_{\mathcal{V}}) + 2\epsilon$.

C. Choosing points close to the boundary

The support vectors are the points within the margin or the misclassified points, i.e., the points x_i for which it holds that $\max(0, 1 - y_i w^T \phi(x_i)) > 0$. Training the SVM only using the support vectors yields the same solution as training the SVM on the whole dataset. This motivates us to only cover the support vectors rather than the whole dataset. In order to estimate the support vectors, we only select boundary points in $\mathcal{V}$ which are the points in $\mathcal{V}$ with $f(.)$ equal to 1. As described in Sec.II-A, $f(.)$ evaluated on a point is equal to 1 if the point is the nearest neighbor of any points from the opposite class. And hence we apply Alg.1 only on the points in $\mathcal{V}$ with $f(.)$ equal to 1.

IV. Results

A. Adult dataset

We first evaluate our algorithms on the adult dataset from the UCI machine learning repository. The target of the dataset is a binary value which corresponds to whether a person income is more or less than $\$50K$. The dataset is divided into training data (32562 samples) and test data (16282 samples). The SVM is trained on the subsets with penalty parameter $C = 50$ while using the RBF kernel. We evaluate our Algorithm 1 in Figure 1 on the test data by plotting the error rate versus the time to run the training algorithms. To benchmark the proposed method, we also evaluate the performance of the core vector machine algorithm [15] which aims to find a coreset for SVM training as well the algorithm based on submodularity from [17] that computes an unweighted subset. We also record the optimal performance of running LibSVM on the entire dataset. We further compare all the algorithms with the stratified random selection algorithm; its results are averaged over 10 runs. The size of the selected random subset is comparable to the size of the subset returned by MIS, which is around 3000. The figure demonstrates that the proposed algorithms are faster than the CVM algorithm while achieving almost optimal accuracy. All the algorithms outperform random sampling, which while being very fast offers relatively poor performance.

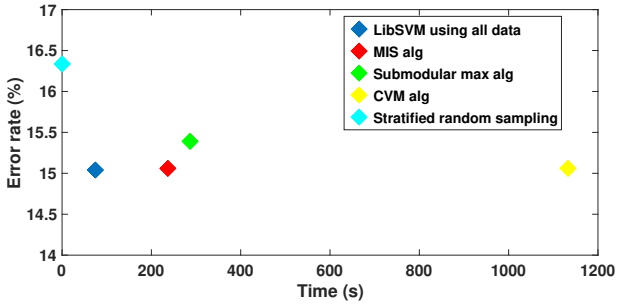

Fig. 1. *Performance of SVM (error rate %) in function of training time.*

B. Census-Income dataset (KDD)

The census dataset from the UCI machine learning repository has 199523 training samples and 99762 test samples. It is a binary dataset of dimension 40. The dataset, like the adult dataset, has a target variable that specifies whether the salary of a person is above or below $\$50$k.

In order to simulate the effect of the size of the training data on the running time of our proposed algorithm, we plot in Fig. 2 the runtime (including the preprocessing time for constructing the LSH tables) versus the size of the training data; the training datasets are formed by taking random subsets of varying sizes from the full dataset. We also evaluate the performance of the CVM algorithm, as well as run LibSVM on the entire dataset. The CVM algorithm is the fastest among the algorithms but its speed comes at the cost of relatively higher error rate. We note that we could obtain a lower error rate for the CVM algorithm by decreasing the stopping rate criterion ϵ but its running time would increase drastically. (For example, setting $\epsilon = 1e-6$ leads to the low error rate of 5.17 but the running time goes up to 12024 s). The figure shows that after 40000 samples of training data, using LibSVM becomes costly. We also plot in Fig. 3 the error rate of the algorithms versus the size of the training data. Our proposed algorithm provided error rates very close to the optimal ones using all the data.

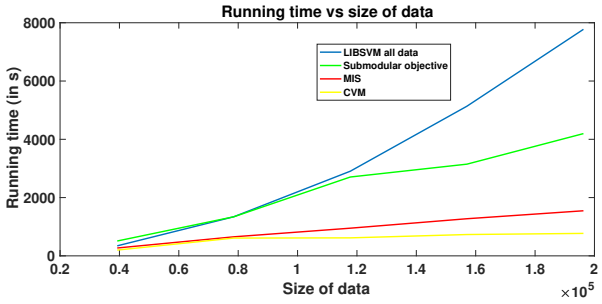

Fig. 2. *Running time in function of training data size.*

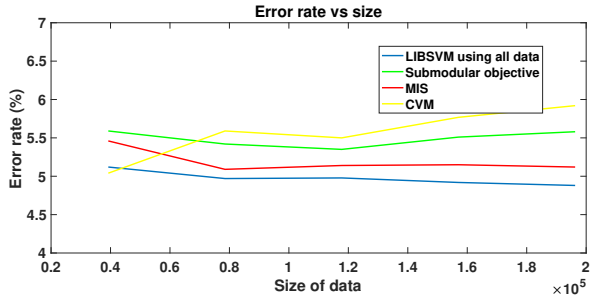

Fig. 3. *Error rate percentage in function of the size of training data.*

V. CONCLUSION

We presented a method for improving the speed of SVM training via weighted subset selection of training data. In particular, the method seeks the maximal independent set (MIS) of the approximate k nearest neighbor graph constructed during a preprocessing step, while assigning weights to the selected subset. Training is done on the selected weighed subset, and the overall performance of the obtained classifier comes with guarantees. Simulations demonstrate that the proposed algorithm enables fast SVM training with near optimal performance.

REFERENCES

[1] S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and Karuturi Radha Krishna Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural computation*, 13(3):637–649, 2001.

[2] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(Sep):1579–1619, 2005.

[3] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[5] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

[6] Hong-Lian Li, Chunhua Wang, and Baozong Yuan. An improved svm: Nn-svm. *CHINESE JOURNAL OF COMPUTERS-CHINESE EDITION-*, 26(8):1015–1020, 2003.

[7] Erik M Ferragut and Jason Laska. Randomized sampling for large data applications of svm. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 1, pages 350–355. IEEE, 2012.

[8] Yuh-Jye Lee and Olvi L Mangasarian. Rsvm: Reduced support vector machines. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–17. SIAM, 2001.

[9] M Barros De Almeida, Antônio de Pádua Braga, and João Pedro Braga. Svm-km: speeding svms learning with a priori cluster selection and k-means. In *Neural Networks, 2000. Proceedings. Sixth Brazilian Symposium on*, pages 162–167. IEEE, 2000.

[10] Shujuan Cao, Xiaomao Liu, and Zhenbing Liu. Fuzzy suppor t vector machine of dismissing margin based on the method of class-center. *Computer Engineering and Applications*, 42(22):146–149, 2006.

[11] LI Qing JIAO Li Cheng ZHOU and Wei Da. Pre-extracting support vector for support vector machine based on vector projection [j]. *Chinese Journal of Computers*, 2:000, 2005.

[12] Chuan Liu, Wenyong Wang, Meng Wang, Fengmao Lv, and Martin Konan. An efficient instance selection algorithm to reconstruct training set for support vector machine. *Knowledge-Based Systems*, 116:58–73, 2017.

[13] Neil Lawrence, Matthias Seeger, Ralf Herbrich, et al. Fast sparse gaussian process methods: The informative vector machine. *Advances in neural information processing systems*, pages 625–632, 2003.

[14] Matthias Seeger. Greedy forward selection in the informative vector machine. Technical report, Technical report, University of California at Berkeley, 2004.

[15] Ivor W Tsang, James T Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(Apr):363–392, 2005.

[16] Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

[17] Sara Mourad, Ahmed Tewfik, and Haris Vikalo. Data subset selection for efficient svm training. In *Signal Processing Conference (EUSIPCO), 2017 25th European*, pages 833–837. IEEE, 2017.

[18] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.