Data Subset Selection for Efficient SVM Training

Sara Mourad
Department of Electrical and Computer Engineering
The University of Texas at Austin

Ahmed Tewfik
Department of Electrical and Computer Engineering
The University of Texas at Austin

Haris Vikalo
Department of Electrical and Computer Engineering
The University of Texas at Austin

***Abstract*—Training a support vector machine (SVM) on large data sets is a computationally intensive task. In this paper, we study the problem of selecting a subset of data for training the SVM classifier under requirement that the loss of performance due to training data reduction is low. A function quantifying suitability of a selected subset is proposed, and a greedy algorithm for solving the subset selection problem is introduced. The algorithm is evaluated on hand digit recognition and other binary classification tasks, and its performance is compared to stratified sampling methods.**

I. INTRODUCTION

The Support Vector Machine (SVM) is a classification method that has gained a lot of attention 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 incorporating a kernel in the objective function, SVM can solve the linear separation problem in higher dimensions. However, SVM suffers from high running time and memory requirements. This problem becomes particularly pronounced as the dimensions of datasets become very large.

Motivated by the proliferation of big data applications, a number of methods that attempt to address high computational complexity of training SVM on large datasets has been proposed in literature. These include the working set methods, or decomposition methods [10], which iteratively break the problem into subproblems that can then be solved analytically; a popular algorithm in this class is the sequential minimization algorithm (SMO) [24]. Other approaches include low-rank approximation methods [29], [12]. However, all of the aforementioned techniques become computationally intensive as the problem size scales, thus limiting their practical feasibility.

Another approach to reducing the computational complexity of SVM training is to search for representative subsets of data while minimizing the loss of performance incurred by making such selection. The problem of selecting a subset of data to train a classifier while minimizing the performance loss due to training set reduction is often referred to as the supervised data selection problem [30]. This task is also known as the instance selection [15] or prototype selection [23] problem. Among the instance selection methods for SVM, the nearest neighbor SVM (NNSVM) [18] 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. In sampled SVM (SSVM) [11] and reduced SVM (RSVM) [17], random sampling methods are used to reduce the size of the training set. The authors of [9] propose an algorithm that performs k-means clustering and retains clusters of points from the same class. In [5], [32], [21], the data geometry is exploited to make use of the centroids of classes to reduce the training set size.

On a related note, submodular maximization has recently been used in various problems that require subset selection. In [31], to select a subset of acoustic data for automatic speech recognition based on a variety of phonetic or prosodic features, a submodular objective function is maximized over the choice of the features. In [30], submodular maximization is used for data subset selection for two classification algorithms, Naive Bayes and Nearest Neighbors, by maximizing the likelihood of the entire data set under those two models trained on the subset. In [16], the informative vector machine algorithm is proposed to efficiently train sparse Gaussian process models. This is made possible by greedily selecting a subset of the training set such that the conditional entropy of the approximated posterior is minimized; since the entropy measure is shown to be submodular, the performance of greedy selection comes with guarantees [26].

In this paper, we are interested in selecting a subset of data to train the support vector machine (SVM) classifier. We distinguish this problem from the data summarization of large datasets [3], [20], which is concerned with reducing the size of the data irrespective of the target, and from the task of selecting a subset of features, which is encountered in dimensionality reduction problems [30], [6], [27], [25]. In contrast to the existing work, we propose an algorithm that can return any desired size of the subset and that takes into account both the proximity of elements to the boundary (using accelerated methods for nearest neighbor search) as well as providing a diverse subset. Our scheme first performs an approximate nearest neighbor search and then employs a greedy algorithm to select the subset for SVM training. Due to the submodular property of the judiciously chosen objective function, the greedy optimization has guaranteed performance. We show that the proposed algorithm has a relatively low complexity (particularly, sub-quadratic in the number of samples) and demonstrate that it outperforms random sampling methods.

ISBN 978-0-9928626-7-1

II. DATA SUBSET SELECTION

A. Problem definition

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. We are interested in selecting a subset $\mathcal{S}$ of size t from the total set of samples $\mathcal{V}$ of size n such that training the SVM algorithm using $\mathcal{S}$ rather than $\mathcal{V}$ incurs minimal loss of performance. To this end, we focus on the dual of the SVM objective that takes the form of the following quadratic optimization problem,

$$\begin{aligned} \max_{\alpha} \; & W(\alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \\ & \text{s.t. } \alpha_i \geq 0 \; \forall i \\ & \sum_{i=1}^{n} y_i \alpha_i = 0 \end{aligned} \tag{1}$$

where x_i, $i = 1, ..., n$, is the input vector, $y_i \in \{-1, 1\}$ is the target variable, α_i is the Lagrange variable and $k(x_i, x_j)$ denotes the kernel function measuring the similarity between x_i and x_j. If the training data is not separable in the transformed space, then a slack variable ζ_i is introduced for each data point in the primal problem and the dual problem (1) is modified by upper bounding each α_i by C, where C is the penalty parameter associated with the sum of the slack variables in the objective function of the primal problem.

A combinatorial search over all possible subsets of size t would reveal the subset that provides the lowest test error but such search is clearly impractical due to having exponential complexity. To facilitate a computationally efficient subset selection, we seek an appropriate function that assigns a meaningful value to the sets and then optimize this objective function over all subsets. Note that the optimal separating hyperplane w^* in the primal SVM problem that uses a linear kernel is readily expressed as the combination of x_i's of the form $w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$, where α_i^* are the optimal Lagrange multipliers. Moreover, the classifier of an input x is expressed as $f(x) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i k(x, x_i))$. In other words, samples corresponding to large values of α_i contribute more to w^* and to the decision function $f(x)$ than samples corresponding to low values of α_i. In fact, α_i's that are strictly greater than zero correspond to the support vectors.

B. Proposed solution

From the dual problem formulation (1), we see that whenever two samples have targets y_i and y_j of opposite sign, then high similarity between x_i and x_j (i.e. high value of $k(x_i, x_j)$) promotes large corresponding α_i and α_j (to maximize the objective). Also, when y_i and y_j are of the same sign, then high similarity between x_i and x_j is undesired and hence the corresponding α_i and α_j are encouraged to be small. This motivates us to define the value of a subset as being large whenever its elements from the same class have low similarity whereas elements from the opposite classes have high similarity. We thus propose solving the subset selection problem by performing the following optimization

$$\underset{\mathcal{S} \subseteq [N]: |\mathcal{S}| = t}{\text{argmax}} \; H(\mathcal{S}) = \sum_{i \in \mathcal{S}} f(i) - \gamma \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}, j \neq i} g(i, j), \tag{2}$$

where $f(\cdot)$ is a function that computes the value of a sample and $g(\cdot, \cdot)$ is a function that computes a notion of similarity between two samples. We denote by F and G the set functions such that $F(\mathcal{S}) = \sum_{i \in \mathcal{S}} f(i)$ and $G(\mathcal{S}) = \gamma \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}, j \neq i} g(i, j)$. $F(\mathcal{S})$ will hence represent the relevance of the subset $\mathcal{S}$, and $-G(\mathcal{S})$ the diversity (or coverage) of the subset. The notion of jointly optimizing the relevance and the coverage of a subset has been proposed in the context of data summarization (e.g., see [19]).

The function $f(\cdot)$ indicates whether a sample is close to the samples of the opposite class. More specifically, it takes on binary values: if a sample i is close to the opposite class samples, $f(i) = 1$, otherwise it is 0 (Sample i is close to/similar to sample j if $k(x_i, x_j)$ is large). Finally, the value of function $g(i, j)$ reflects whether two samples i and j are relatively close to each other. We note that whenever two samples are from opposite classes, then $g(i, j)$ is set to zero[1].

C. Locality sensitive hashing

To characterize similarities between points in a training data set, one could in principle attempt to compute the similarity between each pair of samples. Alternatively, to find the values of the functions $f(\cdot)$ and $g(\cdot, \cdot)$ for a given sample, we may restrict our attention to finding only the closest samples to it and set the similarity with the remaining samples to zero. However, the problem of finding the closest sample to a given sample, typically referred to as the nearest neighbor problem, is $O(nd)$ if approached via linear search, where n denotes the number of samples and d is the dimensionality; this implies that finding the nearest neighbors for all samples is quadratic in the number of samples. A remedy to such a costly search is to find an approximate set of nearest neighbors instead of the exact ones. Many approximate nearest neighbor search algorithms have been proposed in literature, the most popular being the kd-trees [4] and the locality sensitive hashing (LSH) [14]. While kd-trees are effective and work well for low to medium problem dimensions, their queries become near linear at high dimensions. LSH has been shown to work well even in high dimensional settings and is thus our method of choice for approximate nearest neighbor search. In particular, we use LSH to approximately find the k-nearest neighbors to each sample. Given a distance metric (e.g. Euclidean distance [7], inner product [28]), LSH groups similar data points with high probability. Several methods [8], [1] are capable of performing LSH with sublinear query time and sub-quadratic space requirements, and the approximate nearest neighbors set that they find is often as good as the nearest one [8].

Details of our proposed scheme are given below.

Computation of the function $f(\cdot)$. In order to compute $f(\cdot)$, we construct 2 LSH tables; in table L_{01} we evaluate the

[1]Functions f and g will be further discussed in Section II-C)

ISBN 978-0-9928626-7-1 © EURASIP 2017

k-nearest neighbors, from class 0, to each sample from class 1, and in table L_{10} we compute the k-nearest neighbors, from class 1, to each sample from class 0. We define function $f(\cdot)$ (which assigns a value to each sample) to be binary, i.e., $f(\cdot)$ assigns 1 to a sample from class y_i if the sample is one of the k-nearest neighbors to any sample from class $1-y_i$. In other words, to compute the value of a sample from class y_i, we use table $L_{y_i(1-y_i)}$ to check if the sample has been identified as one of the k-nearest neighbors of any sample from class $1-y_i$.

Computation of the function $g(\cdot,\cdot)$. In order to compute the values of $g(\cdot,\cdot)$, we construct two other LSH tables; in table L_{00} we compute the k-nearest neighbors, from class 0, to each sample from class 0, while in table L_{11} we compute the k-nearest neighbors, from class 1, to each sample from class 1. Let $k_{y_jy_j}$ be the number of nearest neighbors used when constructing $L_{y_jy_j}$. Given 2 samples i and j from the same class, $g(i,j) = 1/kk_{y_jy_j}$ if j and i are k-nearest neighbors. In all other cases $g(i,j) = 0$. The denominator in $g(i,j)$ normalizes the contribution of each additional sample j, i.e. the quantity $\sum_{j\in\mathcal{S},j\neq i} g(i,j)$ in Eq.2.

Choice of the parameters. The number k of nearest neighbors for each LSH table is a hyperparameter that can be adjusted. Let $m_{y_i}(\mathcal{V})$ denote the number of samples from class y_i in the set of samples $\mathcal{V}$. For table $L_{y_i(1-y_i)}$, the number of nearest neighbors k increase with $m_{y_i}(\mathcal{V})$; in other words, the number of nearest neighbors from class y_i to the samples in class $1-y_i$ depends on the number of samples in class y_i. As for the table $L_{y_iy_i}$, the number of nearest neighbors k increases with $m_{y_i}(\mathcal{V})$. The hyperparameter γ in equation (2) depends on the size t of the subset. The larger the value of t, the larger the value of γ, since the contribution of function $-G(\mathcal{S})$ (which ensures the diversity of the subset) increases with more data to select. γ also depends on the data; in particular, data with lots of outliers requires larger values of γ. This is because in the case of very noisy data, the metric $F(\mathcal{S})$ in equation (2) is uncertain and hence the metric $G(\mathcal{S})$ increases robustness of the algorithm to outliers. Finally, the hyperparameter r represents the proportion of class 0 samples in the selected subset. We propose to set $r = 0.5$ so that the selected subset $\mathcal{S}$ is balanced even in the case where $\mathcal{V}$ is not. Setting $r = 0.5$ yields better performance of our algorithm than if $r = m_0(\mathcal{V})/(m_0(\mathcal{V}) + m_1(\mathcal{V}))$. Preserving the proportions of classes can lead to a better performance than that achieved by setting $r = 0.5$ for the random algorithm, as we show in Section III.

D. Algorithm and complexity analysis

Definition (Submodularity)**.** Let $\mathcal{S}$ be a finite set and 2^S denote the corresponding power set. A discrete set function $f : 2^{\mathcal{S}} \to R$ is said to be submodular [13] iff

$$\forall \mathcal{A},\mathcal{B} \subseteq \mathcal{S}, f(\mathcal{A}\cup\mathcal{B}) + f(\mathcal{A}\cap\mathcal{B}) \leq f(\mathcal{A}) + f(\mathcal{B}). \quad (3)$$

For finite set $\mathcal{S}$ this is equivalent to $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{S}, \forall j \in \mathcal{S}\backslash\mathcal{B}$,

$$f(\mathcal{A}+j) - f(\mathcal{A}) \geq f(\mathcal{B}+j) - f(\mathcal{B}), \quad (4)$$

i.e., function f satisfies the diminishing return property. If this is satisfied everywhere with equality, then the function is called modular. Moreover, a set function f is monotone non-decreasing if $f(\mathcal{A}) \leq f(\mathcal{B}), \forall \mathcal{A} \subseteq \mathcal{B}$. We say that f is normalized if $f(\emptyset) = 0$.

Let $f(S)$ be a monotone submodular function; then optimization $\max f(\mathcal{S})$ s.t. $|\mathcal{S}| \leq t$ corresponds to maximizing a submodular function under cardinality constraint. Even though NP-hard, it was shown in [22] that this problem can approximately be solved by a greedy algorithm with the solution being within $(1-1/e)$ of the optimal value.

Lemma 1. *$H(\mathcal{S})$ in problem* (2) *is a monotone submodular function, and hence the solution to problem (2) via a greedy algorithm is $(1-1/e)$ optimal.*

Proof. The objective function in (2) is formed as a difference of two set functions. The first, which we denoted as the value function $F(\cdot)$, is a modular function. The second, denoted by $G(\cdot)$, is a supermodular function. Therefore, $-G(\cdot)$ is submodular and so is $F(\cdot)-G(\cdot)$. Note that $G(\cdot)$ is supermodular since $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{S}, \forall j \in \mathcal{S}\backslash\mathcal{B}$, $G(\mathcal{A}+j) - G(\mathcal{A}) = \sum_{i\in\mathcal{A}} g(j,i)$, $G(\mathcal{B}+j)-G(\mathcal{B}) = \sum_{i\in\mathcal{B}} g(j,i)$, and hence $G(\mathcal{B}+j)-G(\mathcal{B})-(G(\mathcal{A}+j)-G(\mathcal{A})) = \sum_{i\in\mathcal{B}\backslash\mathcal{A}} g(j,i) \geq 0$ since g is a positive function. $F(\cdot)$ is modular since $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{S}, \forall j \in \mathcal{S}\backslash\mathcal{B}$, $F(\mathcal{A}+j) - F(\mathcal{A}) = f(j) = F(\mathcal{B}+j) - F(\mathcal{B})$. □

The greedy subset selection algorithm is summarized as Algorithm 1. To ensure the selected subset $\mathcal{S}$ is balanced, i.e. that $r = 0.5$, line 4 of Algorithm 1 would be modified to $j \in \mathcal{V}_0$ ($j \in \mathcal{V}_1$) whenever the number of samples from class 1 (class 0) reaches $t/2$, where $\mathcal{V}_0(\mathcal{V}_1)$ is the set of samples in $\mathcal{V}$ from class 0 (class 1) respectively.

Algorithm 1 Greedy subset selection algorithm for SVM

Input: The dataset $\mathcal{V} = \{(x_i, y_i)_{i=1}^n\}$, subset size t, $\mathcal{S} = \emptyset$.
Construct L_{00}, L_{11}, L_{01}, L_{10}.
while $|S| < t$ **do**
  for $j \in \mathcal{V}$ **do**
    Compute $\delta_j = H(\mathcal{S}\cup\{j\}) - H(\mathcal{S})$ according to equation (2)
  $i = \text{argmax}_{j\in\mathcal{V}}\delta_j$
  $\mathcal{S} \leftarrow \mathcal{S}\cup\{i\}$
  $\mathcal{V} \leftarrow \mathcal{V}\backslash\{i\}$

Analysis of complexity of the algorithm. At each step, we choose the sample j that maximizes $H(\mathcal{S}\cup\{j\}) - H(\mathcal{S}) = f(j) + \gamma\sum_{i\in\mathcal{S}} g(i,j)$, where $\mathcal{S}$ is the subset obtained up to this particular step. Computing $f(j)$ is $\mathcal{O}(1)$, and computing $\sum_{i\in\mathcal{S}} g(i,j)$ is $\mathcal{O}(\min(|\mathcal{S}|, k_{y_jy_j}) = \mathcal{O}(\min(t, k_{y_jy_j})$, where $k_{y_jy_j}$ is the number of nearest neighbors used when constructing $L_{y_jy_j}$, where y_j denotes the target of sample j. This holds because $g(i,j)$ is either 1 or 0, and so $\sum_{i\in\mathcal{S}} g(i,j)$ is the cardinality of the intersection of the set of nearest neighbors of j and S. Let us introduce $k = \max(k_{00}, k_{11})$. The complexity of each step of the greedy algorithm is $\mathcal{O}(n\min(t,k))$, and

ISBN 978-0-9928626-7-1 © EURASIP 2017

hence the total complexity of the algorithm is $\mathcal{O}(nt\min(t,k))$. Note that the time to construct the LSH tables is sub-quadratic since each query time is sublinear.

III. EXPERIMENTS

A. MNIST dataset

We test our subset selection algorithm on the MNIST handwritten digits database, where we consider two types of datasets: imbalanced datasets where one class is dominant, and balanced datasets where both classes have approximately the same number of samples. For testing the imbalanced case, we consider the binary classification problem of the form 'c vs rest' where digit c is mapped to class 1 and all others are mapped to class 0. The training set is of size 50000 and the test set size is 10000. In Figure 1, we show the results of using our algorithm to select a subset to classify '1 vs rest', and compare its performance to random selection schemes. We distinguish balanced random selection where the classes are present at the same proportion from random selection that chooses an equal number of samples from both class; we refer to the latter as the equal random sampling. The random selection algorithm results are averaged over 10 runs. For our algorithm, we set $k_{y_i y_i} = 0.0025 m_{y_i}(V)$, and for table $L_{y_i(1-y_i)}$ we use $k_{y_i(1-y_i)} = 0.005 m_{y_i}(V)$. We employ LSH with the Euclidean distance from the Falconn library [2] to match the RBF kernel used for the SVM algorithm. The factor γ is adjusted depending on the subset size – in particular, small γ is used for a small subset and large γ for a large subset. The intuition is that the larger the subset, the more relevant is the coverage metric. After the subset is selected, we run the SVM algorithm on the subset using the RBF kernel. The penalty parameter used for training SVM is equal to 3. The metric we use for evaluating unbalanced datasets is the F1 score on the test data, the harmonic mean of precision and recall. Figure 1 shows that our algorithm is able to consistently outperform both random sampling procedures and to achieve near-optimal F1 score using only 2% of data. For the balanced dataset, we consider the binary classification task where the goal is to distinguish between two digits. The training set is hence of size 10000 and the test set is of size 2000. We consider the binary classification task of choosing between digits 3 and 7. The evaluation metric is the test error rate since both classes have approximately the same number of samples. The results in Figure 2 show that the subset selection algorithm consistently outperforms stratified random sampling. Note that balanced random sampling and equal random sampling essentially coincide since classes have the same number of samples.

B. Adult dataset

We further evaluate our algorithm 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 70% of training data (22744 samples) and 30% of test data (9762 samples). The SVM is trained on the subsets with

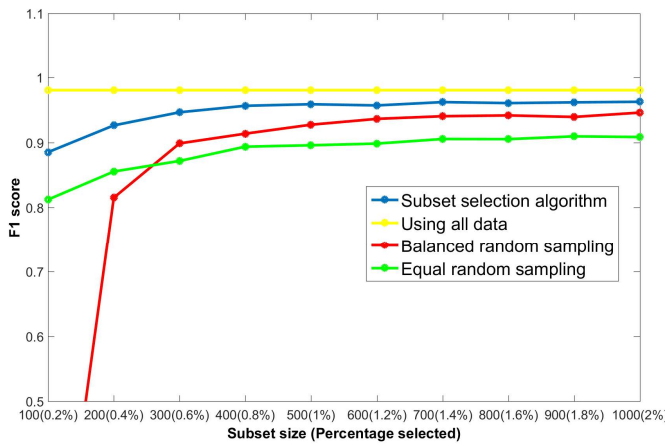

Fig. 1. Performance of SVM (F1 score) in function of subset size for 1 vs rest classification

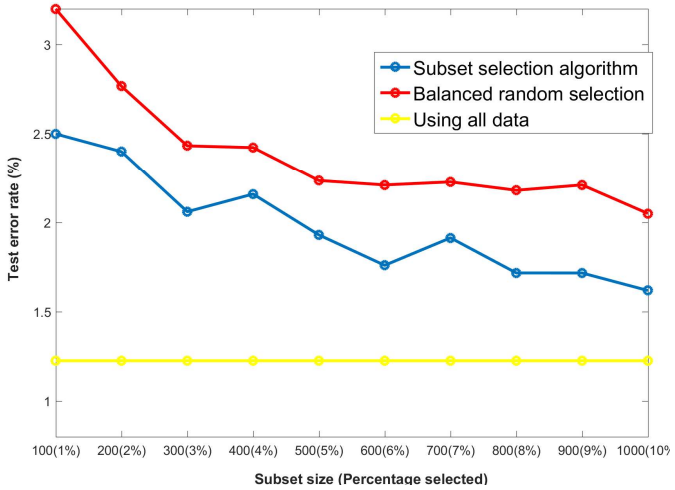

Fig. 2. Performance of SVM (Test error rate) in function of subset size for 3 vs 7 classification

penalty parameter $C = 3$ while using the RBF kernel. We evaluate our algorithm in Figure 3 on the test data using the F1 score; note that the dataset is imbalanced. The random selection algorithm results are averaged over 10 runs. Our algorithm outperforms both the balanced random sampling as well as the equal random sampling. Note that with only 4% of the training data, the proposed algorithm achieves an F1 score very close to the one obtained by the SVM trained using all the data. Note that in order to optimize the F1 score of the imbalanced dataset when training the SVM using all the data, weights inversely proportional to the class frequencies are assigned to the samples. For imbalanced datasets, the proposed subset selection algorithm allows us to judiciously balance the selected subset, leading to an F1 score very close to the one obtained when using all the data.

ISBN 978-0-9928626-7-1 © EURASIP 2017

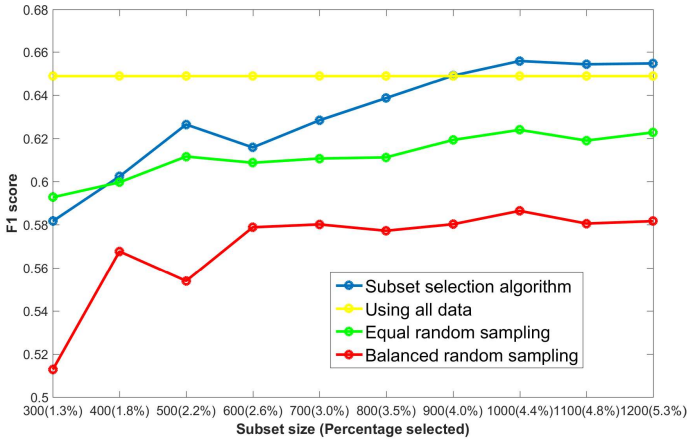

Fig. 3. Performance of SVM (F1 score) in function of subset size for adult dataset classification

IV. CONCLUSION

In conclusion, we proposed in this paper a computationally efficient subset selection algorithm for fast SVM training on large scale data. We verified the performance of model fitting on the subset obtained by our algorithm, and compared it to the optimal training based on using all the data as well as to stratified random sampling. We showed that our algorithm outperforms stratified random sampling methods, and achieves in some cases near-optimal performance.

REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.

[2] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233, 2015.

[3] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680. ACM, 2014.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] S. Cao, X. Liu, and Z. 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.

[6] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*, 2011.

[7] M. Datar, N. Immorlica, P. Indyk, and V. 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.

[8] M. Datar, N. Immorlica, P. Indyk, and V. 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.

[9] M. B. De Almeida, A. de Pádua Braga, and J. P. 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] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines. *Journal of machine learning research*, 6(Dec):1889–1918, 2005.

[11] E. M. Ferragut and J. 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.

[12] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec):243–264, 2001.

[13] S. Fujishige. Submodular systems and related topics. In *Mathematical Programming at Oberwolfach II*, pages 113–131. Springer, 1984.

[14] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[15] N. Jankowski and M. Grochowski. Comparison of instances seletion algorithms i. algorithms survey. In *International conference on artificial intelligence and soft computing*, pages 598–603. Springer, 2004.

[16] N. Lawrence, M. Seeger, R. Herbrich, et al. Fast sparse gaussian process methods: The informative vector machine. *Advances in neural information processing systems*, pages 625–632, 2003.

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

[18] H.-L. Li, C. Wang, and B. Yuan. An improved svm: Nn-svm. *CHINESE JOURNAL OF COMPUTERS-CHINESE EDITION-*, 26(8):1015–1020, 2003.

[19] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

[20] E. Lindgren, S. Wu, and A. G. Dimakis. Leveraging sparsity for efficient submodular data summarization. In *Advances in Neural Information Processing Systems*, pages 3414–3422, 2016.

[21] C. Liu, W. Wang, M. Wang, F. Lv, and M. Konan. An efficient instance selection algorithm to reconstruct training set for support vector machine. *Knowledge-Based Systems*, 116:58–73, 2017.

[22] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[23] E. Pkekalska, R. P. Duin, and P. Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, 2006.

[24] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[25] A. Prasad, S. Jegelka, and D. Batra. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *Advances in Neural Information Processing Systems*, pages 2645–2653, 2014.

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

[27] M. Shamaiah, S. Banerjee, and H. Vikalo. Greedy sensor selection: Leveraging submodularity. In *Proceedings of the 49th IEEE Conference on Decision and Control, CDC 2010, December 15-17, 2010, Atlanta, Georgia, USA*, pages 2572–2577. IEEE, 2010.

[28] A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014.

[29] A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. 2000.

[30] K. Wei, R. K. Iyer, and J. A. Bilmes. Submodularity in data subset selection and active learning. In *ICML*, pages 1954–1963, 2015.

[31] K. Wei, Y. Liu, K. Kirchhoff, C. Bartels, and J. Bilmes. Submodular subset selection for large-scale speech training data. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3311–3315. IEEE, 2014.

[32] L. Q. J. L. C. ZHOU and W. Da. Pre-extracting support vector for support vector machine based on vector projection [j]. *Chinese Journal of Computers*, 2:000, 2005.

ISBN 978-0-9928626-7-1 © EURASIP 2017