Fast SVM Trained by Divide-and-Conquer Anchors

Meng Liu[†], Chang Xu[‡], Chao Xu[†], Dacheng Tao[‡]
[†]Key Laboratory of Machine Perception (MOE), Cooperative Medianet Innovation Center,
School of Electronics Engineering and Computer Science, Peking University, China
[‡]UBTech Sydney AI Institute, The School of Information Technologies, The University of Sydney
mengliu@pku.edu.cn, c.xu@sydney.edu.au,
xuchao@cis.pku.edu.cn, dacheng.tao@sydney.edu.au

Abstract

Supporting vector machine (SVM) is the most frequently used classifier for machine learning tasks. However, its training time could become cumbersome when the size of training data is very large. Thus, many kinds of representative subsets are chosen from the original dataset to reduce the training complexity. In this paper, we propose to choose the representative points which are noted as anchors obtained from non-negative matrix factorization (NMF) in a divide-and-conquer framework, and then use the anchors to train an approximate SVM. Our theoretical analysis shows that the solving the DCA-SVM can yield an approximate solution close to the primal SVM. Experimental results on multiple datasets demonstrate that our DCA-SVM is faster than the state-of-the-art algorithms without notably decreasing the accuracy of classification results.

1 Introduction

Supporting vector machine (SVM) [Cortes and Vapnik, 1995] can be considered as the most popular classifier in machine learning tasks. Due to its importance, optimization methods for SVM have been widely studied [Li *et al.*, 2015; Tsang *et al.*, 2005; Liu and Tao, 2016; Gu *et al.*, 2015; Li and Guo, 2013; Xu *et al.*, 2015; Luo *et al.*, 2016], and efficient libraries such as LIBSVM [Chang and Lin, 2011] and SVM$^{\text{light}}$ [Joachims, 1999] are well developed. However, its application on real-world datasets is limited due to the training time which will increase tremendously as the size of training set becomes large. For example, training time complexity for SVMs with non-linear kernels is typically quadratic in the size of the training dataset [Shalev-Shwartz and Srebro, 2008].

A great number of works have been made to accelerate the training procedure in this literature [Fan *et al.*, 2008; Hsieh *et al.*, 2014; Shalev-Shwartz *et al.*, 2011]. The SVM primal problem is a convex optimization problem with strong duality, thus its solution can be arrived at by solving its dual formulation [Boyd and Vandenberghe, 2004].

Training set selection methods attempt to reduce the SVM training time by optimizing over a selected subset of the training set. Several distinct approaches have been used to select the subset. A core set is defined as the subset of X and its solution of an optimization problem has a solution similar to that for the entire data set [Clarkson, 2010]. In [Tsang *et al.*, 2005], core vector machine (CVM) is proposed which can approximately solve the L2-SVM formulation using core sets, and proved that L2-SVM is a reformulation of the minimum enclosing ball problem for some kernels. Ball vector machine (BVM) further improves CVM by focusing on the enclosing ball [Tsang *et al.*, 2007].

Another type of approximate SVM algorithms is based on the geometric property of data distributions. [Bennett and Bredensteiner, 2000] developed an intuitive geometric interpretation of the standard support vector machine classification of both linearly separable and inseparable data, and proved that finding the maximum margin between the two sets is equivalent to finding the closest points in the smallest convex hulls that contain each class for the separable case. However, Early work [Chazelle, 1993] proved that the calculation complexity of obtaining an exact convex hull is unacceptable in real applications. [Zhou *et al.*, 2013] developed a divide-and-conquer algorithm to obtain the approximate convex hull.

Inspired by recent developments on obtaining representative points, we propose a fast SVM algorithm based on the anchors of approximate convex hull obtained by NMF, and prove that our algorithm can yield an approximate solution close to the primal SVM. We conduct the experiments both on synthetic and multiple real datasets. The results show that our DCA-SVM outperforms the state-of-the-art algorithms, and validate the efficiency and significance of our method.

2 Related Work and Preliminaries

Given a binary-class dataset X with n vectors $x_i \in \mathbb{R}^m$, its corresponding labels $Y = \{y_i : y \in \{-1, 1\}, i = 1, \cdots, n\}$. The primal SVM can be represented as follows:

$$\min_{w,b} J_1(w, b) = \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \ell(w, b, \phi(x_i)) \qquad (1)$$

where $\ell(w, b, \phi(x_i))$ is the hinge loss of x_i. The penalty parameter C is divided by n, which has been frequently used. The optimization of the objective function (1) requires n samples. The training time of traditional SVM can be decreased by reducing the size of the training set.

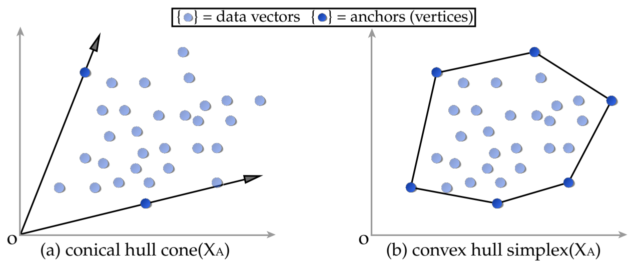

{●} = data vectors {●} = anchors (vertices)
o
(a) conical hull cone(X_A)
o
(b) convex hull simplex(X_A)

Figure 1: Illustration of the conical hull and the convex hull generated by NMF.

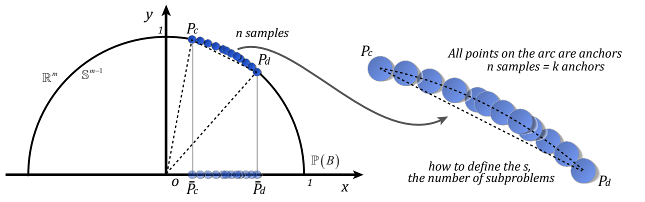


Figure 2: An extreme case for divide-and-conquer anchoring. If all points lie on the surface of the Gaussian ball, they will all be considered as anchors by exact convex hull calculating algorithms. However, its computation cost will be too expensive.

Among a large number of works focusing on obtaining representative subsets, obtaining the geometry convex hull of X is one of the most popular methods. First, let's make a brief revisit on the geometric properties of a set of data. Given a set of points $R = \{r_i\}_{i=1}^{k}$, its cone $cone(R)$ is defined as the conical combinations of k points.

$$cone(R) = \{\sum_{i=1}^{k} h_i r_i | r_i \in R, h_i \in \mathbb{R}_+\} \tag{2}$$

Similarly, a simplex is a non-empty convex set that is closed with respect to convex combinations of its elements. Given a set of points $V = \{v_i\}_{i=1}^{k}$, a simplex $\Delta(V)$ can be defined as follows:

$$\Delta(V) = \{\sum_{i=1}^{k} h_i r_i | r_i \in V, h_i \in \mathbb{R}_+, \sum_{i=1}^{k} h_i = 1\} \tag{3}$$

For a given dataset X, let $\Delta(X)$ denote its convex hull and X_A be anchors (vertices) of the convex hull $\Delta(X)$. Therefore, all points of X can be represented by following convex combination:

$$x_i = \sum_{x_t \in \Delta(X)} h_{i,t} x_t, \tag{4}$$

where $0 \leq h_{i,t} \leq 1$, $\sum_{x_t \in \Delta(X)} h_{i,t} = 1$ and $h_{i,t}$ indicates the convex combination coefficient of anchor x_t for point x_i. Figure 1 shows examples of conical hull and convex hull.

Although the anchors on the convex hull can fully represent the property of all points, the computational complexity of exact convex hull for high-dimensional datasets can be extremely cumbersome. It was proved in [Chazelle, 1993] that the calculation complexity of obtaining an exact convex hull of n vectors of m features is $\mathcal{O}(n^{\lceil m/2 \rceil} + n \log n)$. One example shown in Figure 2 indicates one extreme situation where all points are in the convex hull. Therefore, the approximate yet representative subset of points is needed.

2.1 Approximate Convex Hull of NMF

Non-negative matrix factorization (NMF) decomposes a matrix $X \in \mathbb{R}_+^{n \times m}$ which contains n non-negative m-dimensional vectors $\{x_i\}_{i=1}^{n}$ into the form of $X = HW$, where $H \in \mathbb{R}_+^{n \times k}$, $W \in \mathbb{R}_+^{k \times m}$ and $k \ll \min\{n, m\}$. The rows of W are composed of k non-negative basis vectors representing all the samples, while the n rows of H are non-negative weight vectors [Zheng *et al.*, 2015].

Many additional assumptions are further imposed on the H and W to transform the original NP-hard NMF problem into tractable [Vavasis, 2009]. For example, an early work in [Donoho and Stodden, 2003] gives a separability assumption and prove that a uniqueness of NMF solution can be achieved under this additional assumption. Therefore, the geometric concepts of cone, conical hull, simplex and convex hull can be defined both geometrically and algebraically under the separability assumption. We can also know that a separable matrix is one that admits a non-negative factorization where $X = HX(:, \mathcal{K})$, i.e., W just consists of a subset of the columns of X. The index set $\mathcal{K}$ of columns are called extreme columns. Namely, in separable NMF, $X = HX(:, \mathcal{K})$ implies that all columns of X lie in the cone generated by the columns indexed by $\mathcal{K}$. For any $k \in \mathcal{K}$, $\{\alpha X(:, k) | \alpha \in \mathbb{R}_+\}$ is an extreme ray of this cone. Computing $\mathcal{K}$ is reduced to finding the extreme rays of a cone.

Besides, a near-separable matrix is one where $X = HX(:, \mathcal{K}) + N$, where N is the noise matrix. Determining $\mathcal{K}$ is reduced to finding the extreme points of a convex hull.

Separability assumption selects a few data points to represent the other data points in the whole dataset. This constraint is more than merely an artificial trick: it is favored and justified by various practical applications. For example, in big data challenges, it is more natural, interpretable and efficient to represent high-dimensional data by a few actual data points selected from a huge dataset rather than artificial basis vectors. The separability assumption allows the anchors such *data expresses itself* assumption has become a popular trend in the recent study of other related matrix factorization [Zhou *et al.*, 2013].

Although traditional methods such as linear programming (LP) and greedy pursuit methods can pick out the anchors from noisy data and results in a near-separable NMF, their efficiency could be seriously weakened in high dimensions. Recent work [Zhou *et al.*, 2013] presents a quite efficient divide-and-conquer anchoring (DCA) framework to address near-separable NMF problem by solving several independent sub-problems in low-dimensional spaces, and then obtain an approximate convex hull from the large-scale data in high-dimensional space. Specifically, DCA is a divide-and-conquer framework [Liu *et al.*, 2011] for near-separable NMF and with two steps: the divide step equals applying near-separable NMF to data random projections in multiple subspaces, whilst the conquer step is a fast hypothesis testing based on statistics of the low-dimensional anchors achieved in the divide step.

In each sub-problem of the divide step, DCA projects all the row vectors in X to a randomly generated d-dimensional hyperplane $\mathbb{P} = \mathbb{P}(B)$, where B denotes a subspace $B = [\eta_1; \cdots; \eta_d] \in \mathbb{R}^{d \times m}$ spanned by d random vectors $\{\eta_i\}_{i=1}^d$ uniformly sampled from the unit hypershere $\mathbb{S}^{m-1}$ in $\mathbb{R}^m$. The projections of X on $\mathbb{P}$ is $Y = XB^T$. Since the geometry of conical hull cone(X_A) is partially preserved in Y, the output of separable NMF in i-th subproblem is the low-dimensional anchors's indexes can be represented as follows:

$$\bar{A}^i = \text{SNMF}(X(B^i)^T) \tag{5}$$

The conquer step of DCA is composed of a hypothesis testing that accepts or rejects each data point associated with a detected low-dimensional anchor (from the s subproblems) as an anchor. Since the anchors are usually detected in sub-problems with higher probability than non-anchors, the hypothesis testing can then be reduced to picking out the k data points whose random projections are most frequently selected as anchors in all the sub-problems. The anchor number k can be predetermined or determined automatically. Let $I(i \in \bar{A}^j) : i \to \{0, 1\}$ be an indicator function for the event that data index i is within the $\bar{A}^j$ of the jth sub-problem. For predetermined k, DCA selects the top k largest $\sum_{j=1}^s I(i \in \bar{A}^j)$. In some applications, the rank k is unknown and needs to be determined automatically. When the noise is not overwhelming, a large gap can be observed between anchor and non-anchor on their statistics $\sum_{j=1}^s I(i \in \bar{A}^j)$.

Hence, a tolerance μ can be pre-defined to detect such gap in the sorted $\sum_{j=1}^s I(i \in \bar{A}^j)$ of all data points and automatically identify k. Let p be the new index set after sorting $\sum_{j=1}^s I(i \in \bar{A}^j)$ of all $i \in [n]$ in descending order. By defining $g(p_l) \sum_{j=1}^s I(p_l \in \bar{A}^j)$, anchor set A can be estimated without knowing k by

$$A := p_{[l^*]}, l^* = \min l : g(p_l) - g(p_{l+1}) \leq s\mu \tag{6}$$

DCA can be further accelerated by projecting vectors onto extremely low-dimensional space, such as 1D or 2D space. By this means, DCA gets a promising approximate convex hull. Based on this development, we propose to use the approximate convex hull to be further applied in the SVM training to reduce the training time.

3 Fast SVM Trained on Anchors

In this section, we will introduce the proposed fast SVM trained on anchors, named DCA-SVM. We use the anchors of approximate convex hulls obtained by the divide-and-conquer NMF framework to train the approximate SVM. Figure 3 shows the illustration of our method. The objective function of our method can be written as follows:

$$\min_{w,b} J_2(w, b) = \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{t=1}^{k} \beta_t \ell(w, b, \phi(x_t)) \tag{7}$$

where $\beta_t = \sum_{i=1}^n h_{i,t}$ is the sum of weights for vector x_i, and $\frac{C}{n}$ is the same penalty parameters in problem (1).

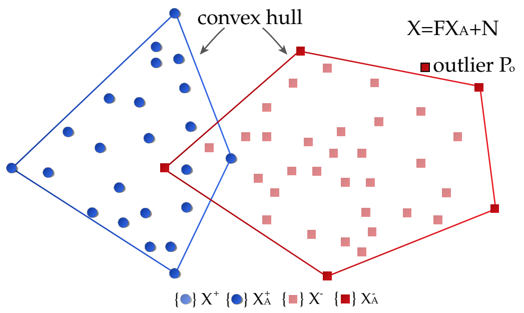


Figure 3: Illustration of the proposed approximate SVM trained on anchors by NMF. X^+ and X^- denote the two different classes, and X_A^+ and X_A^- stand for sets of anchors (vertices) of the convex hull for each class. The outlier point P_o, as well as the inner points, can be represented by linear combination of anchors and corresponding noise. The proposed approximate SVM will be trained on X_A^+ and X_A^-.

3.1 Getting Anchors

To obtain the anchors of the approximate convex hull, it is required to first rearrange X according to their labels as $X = \{X^+, X^-\}$. The divide-and-conquer methods [Xu *et al.*, 2016] which pursue the anchor points are conducted on X^+ and X^- separately. For simplicity, we use the form of the explicit representation of transformed data vectors in the kernel space:

$$Z = \{z_i : z_i = \phi(x_i), \forall x_i \in X\} \tag{8}$$

3.2 Defining Convex Combination Coefficients

After getting the anchors X_A of the original dataset X, we need to determine the coefficients of the anchors corresponding to other points by following equations,

$$\begin{aligned} &\min_{H^+} \sum_{i=1}^{n} \|X - HX_A^+\|_F^2, \\ &\text{s.t. } 0 \leq h_{i,t} \leq 1, \text{and} \sum_{x_t \in X_A} h_{i,t} = 1 \end{aligned} \tag{9}$$

Where H is the coefficient matrix. Since most of the points are inner points, their convex combination coefficients can be quickly obtained. More specifically, for each point x_i, it required to be determined whether it can be fully represented by the anchors,

$$\begin{aligned} &f(x_i, X_A) = \min_{\mathbf{h}_i} \|\phi(x_i) - \sum_{x_t \in X_A} h_{i,t}\phi(x_t)\|^2, \\ &\text{s.t. } \forall i, 0 \leq h_{i,t} \leq 1, \text{and} \sum_{x_t \in X_A} h_{i,t} = 1 \end{aligned} \tag{10}$$

where $\|\phi(x_i) - \sum_{x_t \in X_A} h_{i,t}\phi(x_t)\|^2 = K(x_t, x_t) + \sum_{t=1}\sum_{r=1} h_{i,t}h_{i,r}K(x_t, x_s) - 2\sum_{t=1} K(x_i, x_t)$. In order to solve this quadratic optimization problem, we set a threshold $\xi > 0$, if $f(x_i, X_A) \leq \xi$, x_i is considered as an inner point of the convex hull X_A, otherwise, x_i will be considered as an outer point of the convex hull. In this way, the weight coefficients for each x_i can be calculated separately, which can be accelerated by coordinate descent algorithms.

Algorithm 1 Approximate SVM trained on divide-and-conquer anchors, where the anchor number k is determined automatically.

```
Input: training data $X$, sub-problem number $s$, random vector number $d$.
Output: Parameters of DCA-SVM for classification
  Split training set into $X^+$ and $X^-$ according to their labels;
  (1) Divide and Conquer Step:
  for $i = 1$ to $s$ do
      generate random projection matrix $B$;
      obtain anchors $A_i$ of $X(B^i)^T$ by SNMF in Eq. (5).
  end for
  combing anchors to get $X_A^+$ and $X_A^-$ by Eq. (6);
  (2) Coefficients Learning Step:
  Determining the weight matrices $F^\pm$ and noise matrices $N^\pm$ of $X^\pm = FX_A^\pm + N^\pm$.
  (3) Training procedure:
  train SVM using anchors $X_A^+$ and $X_A^-$ according to Eq. (7).
```

After getting the coefficient parameters, all points can be presented as follows:

$$z_i = \sum_{z_t \in Z^*} h_{i,t} z_t + \tau_i \tag{11}$$

where τ_i is a vector indicates the representation error between $f(x_i, X_A)$,

After getting the coefficients $h_{i,t}(1 \leq t \leq k)$ for point x_i, for an anchor $x_t(1 \leq t \leq k)$, we then obtain its compound coefficient $\beta_t = \sum_{i=1}^{n} h_{i,t}$, which will be used in the objective function of DCA-SVM in Eq. (7). Further, the proposed DCA-SVM can be solved by standard SVM solver such SMO algorithm.

3.3 Computational Complexity

The computation complexity of Algorithm 1 mainly consists of two parts: finding out the anchors for two classes and training SVM on the representative subsets. In the practical calculation, the complexity of getting anchor by DCA-1D or DCA-2D is $\mathcal{O}(nk \log k)$; For the proposed approximate SVM algorithm, its input number of vectors is reduced from n to ρk, where ρ is a constant value. The computational complexity of our approximate SVM algorithm is equal to the primal SVM with same number of reduced training samples.

Let (w_1^*, b_1^*) and (w_2^*, b_2^*) be the optimal solution of $J_1(w,b)$ and $J_2(w,b)$, respectively. The following theorem proved that our SVM can yield an approximate solution close to the primal SVM by thresholding the value of noise.

Theorem 1. *Let $J_1(w,b)$ and $J_2(w,b)$ be the objective functions of primal SVM and DCA-SVM, Then,*

$$J_1(w,b) - \frac{C}{N}\sum_{i=1}^{N} \max\{0, -y_i w^T \tau_i\} \leq J_2(w,b) \tag{12}$$

where τ_i is the noise for vector x_i. Due to limited space, the proof of Theorem 1 is not presented here. In a nutshell, the proof process is straightforward by taking the weighting coefficients of anchors into Eq. (7).

4 Experiments

In this section, we will present the experimental results on synthetic datasets and popular real datasets. We perform all compared algorithms on three real-world datasets: KDD99Lite, UCI Forest[1] and IJCNN1[2]. KDD99Lite is a simplified version of KDD99[3] by removing the repeated data vectors as described in [Tavallaee *et al.*, 2009]. KDD99Lite consists of a training set with 1,074,974 vectors and a test set with 77,216 vectors of 41 features. UCI Forest dataset has 581,012 vectors with 54 features, and it is used to classify the areas of forest cover into one of seven types. We follow the settings of [Tavallaee *et al.*, 2009] to obtain as a classification the 2nd forest cover type and the other types. For IJCNN1, its training set and testing set have 49,990 and 91,701 vectors, respectively. All vectors of IJCNN1 have 22 features. Table 1 summarizes the information of three datasets.

Table 1: Summarization of three datasets on their numbers of training sets, test sets and features.

Datasets	KDD99Lite	UCI Forest	IJCNN1
Training set	1,074,974	283,301	49,990
Test set	77,216	297,711	91,701
Features	41	54	22

For the sake of accuracy of the experiment, we partitioned the data randomly for five-fold cross-validation. The parameter C varies in the range $\{2^{-6}, 2^{-5}, \ldots, 2^5, 2^6\}$.

Our proposed DCA-SVM will be compared with AESVM, CVM, BVM, SVM$^{\text{perf}}$ and LIBSVM. These algorithms can be summarized as follows:

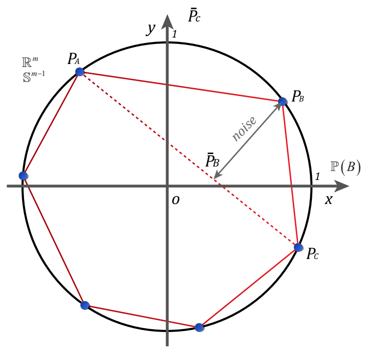
- AESVM: reduces the excessive training time by selecting the approximate *extreme points* according to Euclidean distance between each point within a divide-and-conquer framework. We set the parameter $\epsilon = 10^{-2}$ when using AESVM [Nandan *et al.*, 2014].
- CVM: core vector machine, approximately solves the L2-SVM formulation using *core sets*, which is a subset of the original entire dataset [Clarkson, 2010].
- BVM: ball vector machine, a simplified version of CVM, only utilizes the points lying on the enclosing ball [Tsang *et al.*, 2007].
- SVM$^{\text{perf}}$: an implementation of the SVM formulation for optimizing multivariate performance measures [Joachims, 2005]. We set the given number of support vectors as 1000 in our experiments.
- LIBSVM: a widely used implementation of SVM based SMO algorithm [Chang and Lin, 2011].

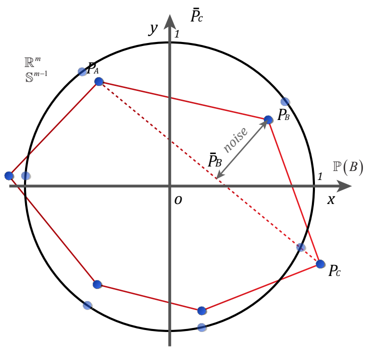
In addition to classification accuracy, we use other measures to evaluate the performances of these methods, which are expected training time speedup T_{te}, overall training time speedup T_{to}, expected classification time speedup T_{ce} and

[1]https://archive.ics.uci.edu/ml/datasets/Covertype

[2]http://www.csie.ntu.edu.tw/$\sim$cjlin/libsvmtools/datasets/binary.html\#ijcnn1

[3]http://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data


(a) all anchors on the hypersurface


(b) small noise won't change the hull

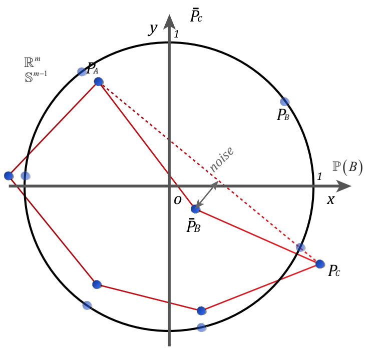
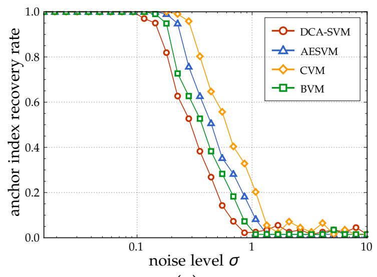
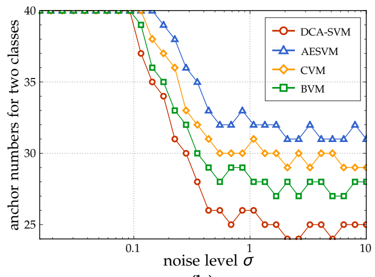

(c) large noise makes P_B an inner point

Figure 4: Influence of noise on the original anchor points. (a) Points sampled on the unit hypersphere $\mathbb{S}^{d-1}$ can be considered as the anchors. (b) Small noise will slightly change the shape of the convex hull while its vertices remain to be anchors. (c) As the noise get larger, the original anchors become inner points while new anchors emerge.


(a)


(b)

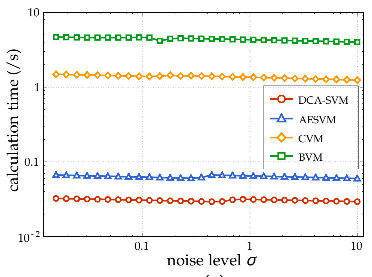

(c)

Figure 5: Comparison of DCA-SVM, AESVM, CVM and BVM on the time they need to get the number of anchors and number of vectors from 20 anchors for each class. (a) Anchors may become inner points after adding noise. (b) The total number of representative subsets for all methods. (c) Training time for getting the subsets for all methods.

Table 2: Classification results of DCA-SVM, AESVM, CVM, BVM, SVM^{perf} and LIBSVM on three real datasets in terms of four time measures, and the maximum ($\times 10^2$), mean ($\times 10^2$) and standard deviation ($\times 10^2$) of accuracy.

Algorithms	T_{te}	T_{to}	T_{ce}	T_{co}	acc(max)	acc(mean± std)
			KDD99Lite			
DCA-SVM	**1712.2**	**173.1**	**6.1**	**3.9**	94.1	92.4±0.6
AESVM	1211.0	156.2	5.9	3.2	94.2	92.3±0.7
CVM	9.1	6.3	1.5	2.2	94.2	92.5±0.9
BVM	26.2	21.7	2	1.9	94.0	92.6±1.7
SVM^{perf}	3.1	1.1	2.6	2.6	**94.3**	92.6±1.2
LIBSVM	1.0	1.0	1.0	1.0	94.1	**92.7**±0.7
			UCI Forest			
DCA-SVM	**1402.4**	**51.8**	28.4	71.8	67.5	60.2±2.2
AESVM	966.1	32.8	22.9	68.4	67.2	59.8±2.8
CVM	7.9	5.8	10.5	25.7	63.8	59.1±4.1
BVM	6.1	4.9	11.3	8.2	64.2	60.2±2.4
SVM^{perf}	3.2	1.2	**183.5**	**261.2**	67.2	61.1±2.9
LIBSVM	1.0	1.0	1.0	1.0	**68.3**	**61.3**±3.4
			IJCNN1			
DCA-SVM	**40.1**	**6.2**	3.2	1.9	98.7	96.3±2.6
AESVM	21.8	4.3	3.1	1.5	98.6	95.9±2.2
CVM	0.3	0.2	0.7	0.6	98.7	96.6±3.1
BVM	0.5	0.4	1.1	1.0	99.0	96.1±2.9
SVM^{perf}	0.3	0.2	**5.1**	**4.2**	**99.1**	96.3±2.5
LIBSVM	1.0	1.0	1.0	1.0	**99.1**	**96.7**±1.7

classification time speedup for optimal hyper-parameters T_{co} as described in [Nandan *et al.*, 2014]. Denoting $\mathcal{F}$ as any concrete SVM algorithm such as DCA-SVM, AESVM and CVM. Four time-related measures are represented as follows.

Expected training time speedup T_{te} stands for the expected speedup time in training procedure:

$$T_{te} = \frac{1}{RS}\sum_{r=1}^{R}\sum_{s=1}^{S}\frac{T_{L_s}^{r}}{T_{\mathcal{F}_s}^{r}} \tag{13}$$

where $T_{L_s}^{r}$ and $T_{\mathcal{F}_s}^{r}$ stand for the training times of LIBSVM and given algorithm $\mathcal{F}$ in the sth cross-validation fold with the r_{th} set of hyper-parameters of grid search.

Overall training time speedup T_{to} represents the overall training time including the time spent on calculating the representative subset such as in DCA-SVM and AESVM.

$$T_{to} = \frac{\sum_{r=1}^{R}\sum_{s=1}^{S} T_{L_s}^{r}}{\sum_{r=1}^{R}\sum_{s=1}^{S} T_{\mathcal{F}_s}^{r} + T_{X^*}} \tag{14}$$

where T_{X^*} notes the time used to obtain the subset.

Expected classification time speedup T_{ce} is indicated as follows:

$$T_{ce} = \frac{1}{RS}\sum_{r=1}^{R}\sum_{s=1}^{S}\frac{N_{L_s}^{r}}{N_{\mathcal{F}_s}^{r}} \tag{15}$$

where $N_{L_s}^{r}$ and $N_{\mathcal{F}_s}^{r}$ represent the numbers of support vectors in the solution of LIBSVM and $\mathcal{F}$, respectively.

Classification time speedup for optimal hyper-parameters T_co chooses the corresponding optimal classification accuracy results of LIBSVM and given $\mathcal{F}$ in grid search:

$$T_{co} = \frac{\max_r \sum_{s=1}^{S} N_{L_s}^{r}}{\max_r \sum_{s=1}^{S} N_{\mathcal{F}_s}^{r}} \tag{16}$$

4.1 Experimental Study on Synthetic Data

For illustrative purpose, we conduct our first on the synthetic dataset. Two sets of vectors X^+ and X^- are generated according to $X^{\pm} = FX_A^{\pm} + N^{\pm}$. The noise matrices N^+ and N^- both are generated by i.i.d Gaussian distribution $\mathcal{N}(0, \sigma^2)$ where σ represents the noise level. The number of anchors for each class is fixed as 20. After setting noises of different levels, we get two sets of points. The number of points for each set is 1000 and the feature number is 10000. As elaborated in Figure 4, different noise levels will bring different changes to the anchors of convex hull.

Figure 5 shows the results of the proposed DCA-SVM compared with three popular approximate SVM algorithms: AESVM, CVM and BVM. Their results are evaluated in terms of three measures: anchor index recovery rate representing the ratio of the observed number of original anchors to the total number, overall training time consisting of the time of determining representative subsets, and overall training time. It can be observed that most algorithms are able to find out the original anchors when the noise level is close to 1 where the original convex hull remains its shape. Moreover, the numbers of points in the representative subsets obtained by four algorithms have large difference. DCA-SVM aims to find the approximate convex hull, thus its point numbers are less than the other three algorithms. It is worth noting that DCA-SVM uses the least training time. Their property on classification accuracy will be further studied on real datasets in the rest of this section.

4.2 Comparision on Real Datasets

We evaluate the classification performance of DCA-SVM, AESVM, CVM, BVM and LIBSVM. We follow the same experimental settings in [Nandan *et al.*, 2014]. There are two notable kinds of parameters, classification accuracy and training time. The classification accuracy is defined as the ratio of the number of correct classifications to the total number of samples involved in training procedure, while the training time consists of four time measures mentioned above.

Table 2 shows the classification results of DCA-SVM, AESVM, CVM, BVM, SVM$^{\text{perf}}$ and LIBSVM on KDD99Lite, UCI Forest and IJCNN1. These results are evaluated in terms of four time measures and three accuracy-related measures. We can observe that: (1) Most approximate SVM algorithms achieve faster overall training time T_{to} than LIBSVM on KDD99Lite and UCI Forest datasets, while these approximate SVM algorithms run much slower on IJCNN1 except DCA-SVM and AESVM. This is because the sizes of training sets of KDD99Lite and UCI Forest are much larger than IJCNN1, which makes the training time on the whole original training sets become tremendous. (2) The proposed DCA-SVM outperforms other algorithms notably on expected training time speedup T_{te} and overall training time speedup T_{to}. Specifically, the T_{te} of DCA-SVM is 1712.2 times faster than LIBSVM on KDD99Lite dataset and nearly twice as fast as the competitive AESVM on IJCNN1 dataset. (3) All algorithms produce similar accuracy performances on three datasets. The proposed DCA-SVM achieves decent accuracy which is only 0.2% less than the largest accuracy on KDD99Lite dataset. In short, our method outperforms most of the compared methods on speed and produce fairly good classification accuracy results.

5 Conclusions

In this paper, we propose to train an approximate SVM by using the anchors obtained from non-negative matrix factorization (NMF) in a divide-and-conquer framework. To be specific, the weighting coefficients of the anchors corresponding to other points are used in the training procedure of the approximate SVM. Our theoretical analysis shows that the solving the DCA-SVM can yield an approximate solution close to the primal SVM. Experimental results on the synthetic datasets and multiple real-world datasets show that the proposed DCA-SVM is faster than other state-of-the-art algorithms, and does not lead to a notable decrease in the accuracy of classification results, which validate the efficiency and significance of our method.

Acknowledgements

This research is partially supported by grants from NSFC 61375026 and 2015BAF15B00, and Australian Research Council Projects FT-130101457, DP-140102164, LP-150100671.

References

[Bennett and Bredensteiner, 2000] Kristin P Bennett and Erin J Bredensteiner. Duality and geometry in svm classifiers. In *ICML*, pages 57–64, 2000.

[Boyd and Vandenberghe, 2004] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Chang and Lin, 2011] ChihChung Chang and ChihJen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[Chazelle, 1993] Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10(4):377–409, 1993.

[Clarkson, 2010] Kenneth L Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms*, 6(4):63, 2010.

[Cortes and Vapnik, 1995] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[Donoho and Stodden, 2003] David Donoho and Victoria Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in neural information processing systems*, page None, 2003.

[Fan *et al.*, 2008] 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.

[Gu *et al.*, 2015] Bin Gu, Victor S Sheng, and Shuo Li. Biparameter space partition for cost-sensitive svm. In *IJCAI*, pages 3532–3539, 2015.

[Hsieh *et al.*, 2014] Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon. A divide-and-conquer solver for kernel support vector machines. In *ICML*, pages 566–574, 2014.

[Joachims, 1999] Thorsten Joachims. Svmlight: Support vector machine. *SVM-Light Support Vector Machine*, 19(4), 1999.

[Joachims, 2005] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384, 2005.

[Li and Guo, 2013] Xin Li and Yuhong Guo. Active learning with multi-label svm classification. In *IJCAI*, 2013.

[Li *et al.*, 2015] Xiang Li, Huaimin Wang, Bin Gu, and Charles X Ling. Data sparseness in linear svm. In *IJCAI*, pages 3628–3634, 2015.

[Liu and Tao, 2016] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2016.

[Liu *et al.*, 2011] Qi Liu, Yong Ge, Zhongmou Li, and Enhong Chen. Personalized travel package recommendation. pages 407–416, 2011.

[Luo *et al.*, 2016] Yong Luo, Yonggang Wen, Dacheng Tao, Jie Gui, and Chao Xu. Large margin multi-modal multitask feature extraction for image classification. *IEEE Transactions on Image Processing*, 25(1):414–427, 2016.

[Nandan *et al.*, 2014] Manu Nandan, Pramod P Khargonekar, and Sachin S Talathi. Fast SVM training using approximate extreme points. *Journal of Machine Learning Research*, 15(1):59–98, 2014.

[Shalev-Shwartz and Srebro, 2008] Shai Shalev-Shwartz and Nathan Srebro. Svm optimization: inverse dependence on training set size. In *Proceedings of the 25th international conference on Machine learning*, pages 928–935. ACM, 2008.

[Shalev-Shwartz *et al.*, 2011] 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.

[Tavallaee *et al.*, 2009] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali-A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications*, 2009.

[Tsang *et al.*, 2005] 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.

[Tsang *et al.*, 2007] Ivor W Tsang, Andras Kocsor, and James T Kwok. Simpler core vector machines with enclosing balls. In *ICML*, pages 911–918, 2007.

[Vavasis, 2009] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

[Xu *et al.*, 2015] Chang Xu, Dacheng Tao, and Chao Xu. Large-margin multi-label causal feature learning. In *AAAI*, pages 1924–1930, 2015.

[Xu *et al.*, 2016] Chang Xu, Dacheng Tao, and Chao Xu. Robust extreme multi-label learning. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA August*, pages 13–17, 2016.

[Zheng *et al.*, 2015] Xiaodong Zheng, Shanfeng Zhu, Junning Gao, and Hiroshi Mamitsuka. Instance-wise weighted nonnegative matrix factorization for aggregating partitions with locally reliable clusters. In *IJCAI*, pages 4091–4097, 2015.

[Zhou *et al.*, 2013] Tianyi Zhou, Wei Bian, and Dacheng Tao. Divide-and-conquer anchoring for near-separable nonnegative matrix factorization and completion in high dimensions. In *International Conference on Data Mining*, pages 917–926, 2013.