Adaptive Kernel Approximation for Large-Scale Non-Linear SVM Prediction

Michele Cossalter MICHELE.COSSALTER@SV.CMU.EDU
Carnegie Mellon University

Rong Yan RONGYAN@FACEBOOK.COM
Facebook

Lu Zheng LU.ZHENG@SV.CMU.EDU
Carnegie Mellon University

Abstract

The applicability of non-linear support vector machines (SVMs) has been limited in large-scale data collections because of their linear prediction complexity to the size of support vectors. We propose an efficient prediction algorithm with performance guarantee for non-linear SVMs, termed *AdaptSVM*. It can selectively collapse the kernel function computation to a reduced set of support vectors, compensated by an additional correction term that can be easily computed on-line. It also allows adaptive fall-back to original kernel computation based on its estimated variance and maximum error tolerance. In addition to theoretical analysis, we empirically evaluate on multiple large-scale datasets to show that the proposed algorithm can speed up the prediction process up to 10^4 times with only $< 0.5\%$ accuracy loss.

1. Introduction

The solid theoretical foundation and proved effectiveness of support vector machines (SVMs) have contributed to their success in diverse applications (Burges, 1998). One of the most appealing properties for SVMs is their ability to discriminate non-linearly separable data by mapping original feature space into a higher dimensional space via kernel functions. However, the applicability of non-linear SVMs is hindered by their prediction complexity which is linearly growing to the number of support vectors (SVs). The SVM prediction process may become computationally demanding (Joachims & Yu, 2009; Rai et al., 2009; Catanzaro et al., 2008) especially for large-scale data collections with complicated decision boundaries, such as image and audio data, where the size of SVs is not much smaller than the training data. Unlike learning which can be performed off-line, prediction must take place on-line and thereby its efficiency plays a much more crucial role for the applicability in real world datasets (Liu et al., 2003; Ukkonen, 2010).

Most existing approaches address this issue by optimizing the SVM learning stage to produce a smaller set of support vectors. For instance, ClusterSVM (Boley, 2004) partitions the training data into pair-wise disjoint clusters and replaces the cluster containing only non-support vectors by a point. Tsang et al. (2005) proposed Core Vector Machine (CVM), which formulates many kernel methods as equivalent minimum enclosing ball problems, with a time complexity linear in the training set size. Joachims & Yu (2009) proposed Cutting-Plane Subspace Pursuit (CPSP), which trains on a reduced number of arbitrary support vectors and iteratively constructs the set of vectors from a cutting-plane model. The number of SVs can be reduced by two orders of magnitude without a significant loss of accuracy. While being effective in reducing SVM prediction complexity, these approaches usually require a significant upgrade of learning algorithms, and might not exploit full potential of prediction speedup due to their focus on training optimization.

A second type of approaches completely focus on accelerating the prediction process. Starting from a set of support vectors learned from existing methods, they aim to reduce the size of support vectors by ei-

Appearing in *Proceedings of the 28^{th} International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

ther data selection or aggregation methods. Burges & Scholkopf (1996) and Schölkopf & Smola (2001) suggested searching for a reduced set of support vectors such that the Euclidean distance between the original and approximated solution is minimized. They can achieve a significant prediction speedup, but suffer from the drawback of expensive computation. Liu et al. (2003) presented a ball tree algorithm that can be exploited to achieve faster k-NN and SVM classification. Ukkonen (2010) proposed an SV tree idea which maintain a flexible subset of basis vectors depending on the classifier input. When having a comparable accuracy to non-linear SVMs, its number of basis functions is dramatically reduced.

Following the second school of thought, we propose an efficient prediction algorithm for non-linear SVMs called *AdaptSVM*, which achieves orders of magnitude speedup without significant loss on the prediction performance on large-scale datasets. By assuming SVs are randomly sampled from centain distribution, we provide a key theorem to approximate the mean and variance of kernel functions under two general assumptions. This allows us to selectively compress non-linear kernel computation into functions of a much smaller set of aggregated SVs, compensated by an additional correction term, where both can be computed efficiently on-line. In contrast to the prior approaches, we also develop an adaptive fall-back strategy to switch back to original kernel computation when the estimated variance is larger than the maximum error tolerance, so as to guarantee the prediction performance is not degraded. Since the proposed method is only related to prediction parameters, it can work together with any training optimization approaches to achieve even further speedup. The proposed idea is related to adaptive multi-pole expansion proposed by Greengard & Rokhlin (1987) and its further development in Improved Fast Gauss Transform proposed by Yang et al. (2004). However, since the approximation in these works does not account for training distribution, the derived error bound has no probabilistic semantics and is looser than our variance estimation, thus being unsuitable for dynamic fallback.

The remainder of the paper is structured as follows. The proposed algorithm and theoretical analysis are described in Section 2. Experimental evaluation results on three large-scale datasets are provided in Section 3 to demonstrate its effectiveness. Section 4 concludes the paper with discussions on future directions.

2. AdaptSVM: adaptive prediction for large-scale non-linear SVMs

Given the labeled set of M training samples $\{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i \in \mathcal{R}^K$ with K features, and $y_i \in \{-1, 1\}$ is the corresponding label, an SVM binary classifier finds the optimal hyperplane to correctly separate the data points while maximizing the margins using constrained optimization. The SVM prediction problem can then be written as the following kernel form (Burges, 1998):

$$\sum_{i=1}^{M} y_i \alpha_i K(\mathbf{x}_i, \mathbf{z}) = \sum_{y_i=1} \alpha_i K(\mathbf{x}_i, \mathbf{z}) - \sum_{y_i=-1} \alpha_i K(\mathbf{x}_i, \mathbf{z})$$

where $\mathbf{z}$ is the data to be classified and α_i is the learned weight of $\mathbf{x}_i$ with $\alpha_i \geq 0$. Without loss of generality, we omit the bias term in the prediction function and only keep the support vectors $\mathbf{x}_i$ with $\alpha_i > 0$. $K(\cdot, \cdot)$ is a kernel function which can project the original input space to a high dimensional feature space without being computationally intractable. To classify non-linearly separable data, the commonly used kernel functions include radial basis function (RBF) and polynomials. Despite their effectiveness, SVMs suffer from the drawback that the prediction complexity is growing linearly to the number of support vectors, which can become an issue for large-scale datasets.

To this end, we propose using an adaptive SVM prediction process, which compresses the representation of the decision boundary to be determined by only a small number of support vectors, while dynamically falling back to the original kernel computation if the compression quality is insufficient. First, it is legitimate to consider the support vectors $\mathbf{x}_i$ as i.i.d. realizations of a random variable $\mathbf{x}$, and α_i as a function of $\mathbf{x}$, e.g. $\alpha_{\mathbf{x}}$. Given that the SVM prediction function only depends on the sum of kernel functions, we can estimate the kernel summation using the expected value of the kernel function $\mathbb{E}[\alpha_{\mathbf{x}} K(\mathbf{x}, \cdot)]$ as surrogate, leading to a much faster on-line prediction process if the expected value can be computed from off-line statistics. This also provides an elegant way to measure the approximation quality based on the variance of the prediction function $\mathrm{Var}[\alpha_{\mathbf{x}} K(\mathbf{x}, \cdot)]$, thus we can adjust the prediction formulation based on error tolerance.

2.1. Main theorem

In this section, we present a main theorem to derive both $\mathbb{E}[\alpha_{\mathbf{x}} K(\mathbf{x}, \cdot)]$ and $\mathrm{Var}[\alpha_{\mathbf{x}} K(\mathbf{x}, \cdot)]$ when $\forall \alpha_{\mathbf{x}} > 0$. Let us start by introducing our notation. We denote $f(\mathbf{x}) = K(\mathbf{x}, \mathbf{z})$, and transform $\mathbb{E}[\alpha_{\mathbf{x}} K(\mathbf{x}, \cdot)]$ to a weighted expectation form:

$$\mathbb{E}[\alpha_{\mathbf{x}} f(\mathbf{x})] = \alpha_0 \mathbb{E}[f(\mathbf{x}) \alpha_{\mathbf{x}} / \alpha_0] = \alpha_0 \mathbb{E}_{\alpha}[f]$$

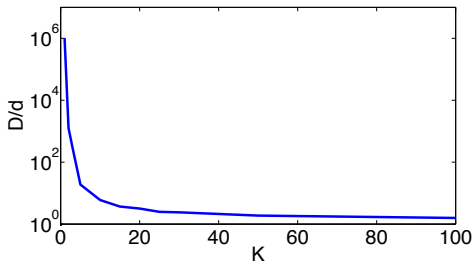

Figure 1. Within-SV distance $r_{\mathbf{x}}$ are close to each other for a sufficiently large feature dimension.

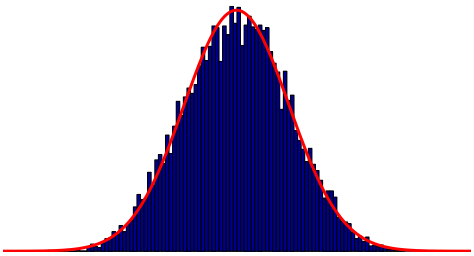

Figure 2. The distribution of $\cos\theta_{\mathbf{x}}$ closely fits a normal distribution $\mathcal{N}(0, \lambda/K)$.

where $\alpha_0 = \mathbb{E}[\alpha_{\mathbf{x}}]$. For simplicity, we will drop the index of α in the following discussions, and use $\mathbb{E}$ to represent $\mathbb{E}_\alpha$. $\mathrm{Var}[f]$ is re-defined in a similar way.

Given the weighted mean of $\mathbf{x}$ as $\mu = \mathbb{E}[\mathbf{x}]$, we rewrite $f(x)$ using its first order Taylor expansion w.r.t. μ:

$$f(\mathbf{x}) = f(\mu) + \nabla f(\mu)^T R(\mathbf{x}) + \Delta_{\mathbf{x}} \tag{1}$$

where $R(\mathbf{x}) = \mathbf{x} - \mu$, and $\Delta_{\mathbf{x}}$ is the correction residual term with the following integral form:

$$\Delta_{\mathbf{x}} = \int_0^1 (1-\beta) R^T(\mathbf{x}) H(\mu + \beta R(\mathbf{x})) R(\mathbf{x}) d\beta \tag{2}$$

with $H(\mathbf{x})$ being the Hessian matrix of $f(\mathbf{x})$.

Because $\mathbb{E}[R(\mathbf{x})] = 0$, taking expectation on both sides yields the following:

$$\mathbb{E}[f(\mathbf{x})] = f(\mu) + \mathbb{E}[\Delta_{\mathbf{x}}]$$

Similarly, the variance term can be computed by the following Taylor expansion:

$$\mathrm{Var}[f(\mathbf{x})] = \mathrm{Var}[f(\mu) + \nabla_{\mathbf{x}}] = \mathrm{Var}[\nabla_{\mathbf{x}}]$$

where $\nabla_{\mathbf{x}}$ is the residual term:

$$\nabla_{\mathbf{x}} = \int_0^1 \nabla f(\mu + \beta R(\mathbf{x})) R(\mathbf{x}) d\beta \tag{3}$$

This suggests to estimate the expected prediction function as its value on weighted mean $f(\mu)$ plus a correction residual term $\mathbb{E}[\Delta_{\mathbf{x}}]$, as well as the variance in an integral form. Since μ can be trivially precomputed, our problem boils down to computing $E[\Delta_{\mathbf{x}}]$ in an efficient way with only simple off-line statistics.

Because most popular kernels are functions of inner product between two vectors, the expectation of kernels can be transformed into a function of their norms and the angle between them. Therefore, let us define the centroid distance $d = \|\mu - \mathbf{z}\|$, the within-SV distance $r_{\mathbf{x}} = \|\mathbf{x} - \mu\|$ with mean r, and $\theta_{\mathbf{x}}$ as the angle between the vectors $\mu - \mathbf{z}$ and $\mathbf{x} - \mu$. In the following, we focus our analysis on the RBF kernel[1] $f(\mathbf{x}) = K(\mathbf{x}, \mathbf{z}) = e^{-\rho\|\mathbf{z}-\mathbf{x}\|^2}$, which can then be rewritten as $f(\mathbf{x}) = \exp(-\rho(d^2 + r_{\mathbf{x}}^2 + 2dr_{\mathbf{x}}\cos\theta_{\mathbf{x}}))$.

Given that $r_{\mathbf{x}}$ and $\theta_{\mathbf{x}}$ are the sufficient statistics for $\mathbf{x}$ in the kernel function, we make the following two assumptions before establishing the main theorem:

Assumption 1. *The distance between r_x and r becomes indiscernible when the feature dimension $K \to \infty$, i.e., $\lim_{K\to\infty}(r_x - r) = 0$.*

Related to the curse of dimensionality, this assumption states that r_x are close to each other for a sufficiently large feature dimension. A similar statement is proved by Beyer et al. (Beyer et al., 1999), showing that $(D-d)/d$ converges to 0 where $D = \max\{r_x\}$ is the maximal distance, and $d = \min\{r_i\}$ is the minimal distance. In practice, d can be bounded by feature normalization, and thus our assumption can be satisfied. To validate, we generated a synthetic dataset by randomly drawing 10^5 random samples $x_i \in \mathbb{R}^K$ from a uniform distribution, with K varying between 1 and 100, and computed the distance r_i of each point. Figure 1 shows that our results are aligned with (Beyer et al., 1999). The ratio between maximal and minimal distance drops down quite fast as dimensionality increases, e.g. $D/d \approx 3$ for $K = 20$ and $D/d < 2$ for $K = 100$. This experiment is repeated on the empirical datasets in Section 3, obtaining $D/d < 2$ for all of them (Table 1).

Assumption 2. *$\cos\theta_{\mathbf{x}}$ follows an independent normal distribution $\cos\theta_{\mathbf{x}} \sim N(0, \sigma^2)$, where $\sigma^2 = \lambda/K$ where λ is a constant dependent on training data.*

This assumption is based on (Hammersley, 1950) which shows that the distance d between two points uniformly distributed on a sphere follows the normal distribution of $\mathcal{N}(a\sqrt{2}, a^2/2K)$, where a is the radius.

[1] Our method can be straightforwardly extended to other kinds of kernels, e.g. polynomial, in a similar fashion, although we skip the discussion due to space limit.

Denoting the angle between these vectors as $\theta_{\mathbf{x}}$, we can derive $\cos\theta_{\mathbf{x}}$ to approximately follow a normal distribution $N(0, 1/K)$, given that $d = 2a\cos(\theta_{\mathbf{x}}/2)$.

The validity of this assumption can be demonstrated by Figure 2, where we computed $\cos\theta_{\mathbf{x}}$ after randomly drawing 100 points $\mathbf{x}_i$ and other 100 points $\mathbf{z}_j$ from a uniform distribution, with feature size K varying between 10 and 100. We test the null hypothesis that the resulting samples are distributed as $\mathcal{N}(0, 1/K)$ using a Kolmogorov-Smirnov test at 5% significance level. After 100 repetitions of the experiment, the null hypothesis was accepted 99% of the times for $K \geq 20$. In practice, we found that $\cos\theta_{\mathbf{x}}$ mostly follow a normal distribution, but the variance is not necessarily $1/K$ due to non-uniform distributions. Therefore, we introduced λ as a variance adjustment factor that can be estimated off-line based on the training data (Table 1). We can now establish our main theorem based on the above assumptions:

Theorem 1. *If Assumption 1 and 2 hold, the mean and variance of the kernel function can be derived as:*

$$\begin{aligned} \lim_{r_{\mathbf{x}}\to r} \mathbb{E}[f(\mathbf{x})] &= e^{-\rho d^2} G(d, r) + O(1/K^7) \\ \lim_{r_{\mathbf{x}}\to r} Var[f(\mathbf{x})] &\leq d^2 e^{-2\rho d^2} V(r) \end{aligned} \tag{4}$$

where $\beta_n = n/N$, $r_n = r\beta_n$, *and*

$$\begin{aligned} G(d, r) &= 1 - \frac{1}{N}\sum_{i=0}^{3} \sigma^{2i} d^{2i} \left[\sum_{n=0}^{N-1} \gamma_n(r) A_{ni}(r)\right] \\ V(r) &= \sigma^2 \sum_{n=0}^{N-1} \left[2\rho e^{-\rho r_n^2}(1 - 2\rho r_n^2)\right]^2 \\ A_{n0}(r) &= (1 - 2\rho r_n^2) \\ A_{n1}(r) &= (-4\rho^2 r_n^4 + 10\rho r_n^2 - 2) \\ A_{n2}(r) &= (-4\rho^3 r_n^6 + 18\rho^2 r_n^4 - 12\rho r_n^2) \\ A_{n3}(r) &= (-4/3\rho^4 r_n^8 + 52/3\rho^3 r_n^6 - 20\rho^2 r_n^4) \\ \gamma_n(r) &= 2\rho(1 - \beta_n) r e^{-\rho r_n^2} \end{aligned}$$

Proof. See Appendix. □

A key observation for this theorem is that except the distance d, all the other variables do not depend on the testing data $\mathbf{z}$. Therefore we can precompute and store these coefficients after the training process without wasting on-line computation resources. The only required on-line computation comes from d^{2i}, $e^{-\rho d^2}$ and their weighted sum, which has a much lower time complexity than the original kernel evaluation that combines a large number of $e^{-\rho\|\mathbf{z}-\mathbf{x}\|^2}$.

Algorithm 1 AdaptSVM: adaptive prediction for non-linear SVMs

Input: SVs $\{\mathbf{x}_i\}$ with weights $\{\alpha_i\}$, target $\mathbf{z}$, kernel function K, group size J, tolerance τ
Output: SVM prediction $f(\cdot)$ for $\mathbf{z}$

1. Offline aggregation stage
Cluster pos and neg SVs into $J/2$ groups each
for $j = 1 \dots J$ **do**
  Compute $\alpha^j = \sum_{i=1}^{M} \alpha_i$, $\mu_j = \sum_{i=1}^{M_j} \alpha_i x_i$
  Compute all the terms irrelevant to distance d in Theorem 1, e.g. $V(r)$, $A(r)$ and $\gamma(r)$
end for
2. Online prediction stage
for $j = 1 \dots J$ **do**
  Estimate $\mathbb{E}[\mathbf{f}]$ and $\mathrm{Var}[\mathbf{f}]$ using Theorem 1
  if $\mathrm{Var}[\mathbf{f}] < \tau\mathbb{E}^2[\mathbf{f}]$ **then**
    $f^j = y^j \alpha^j \mathbb{E}[\mathbf{f}]$ (approximation)
  else
    $f^j = y^j \sum_{i=1}^{M_j} \alpha_i f(x_i)$ (original)
  end if
end for
return $f = \sum_{j=1}^{J} f^j$

2.2. Overall algorithm

The main theorem provides an efficient way to estimate the mean and variance of kernel functions by using the aggregated statistics under general conditions. It also offers a solid foundation to develop the adaptive SVM prediction algorithm described in this section. We can now approximate the original SVM prediction using its expected value by substituting the expected value in Eq. (4) into the SVM decision function:

$$\sum_{i=1}^{M} \alpha_i K(\mathbf{x}_i, \mathbf{z}) = \alpha\mathbb{E}[f(\mathbf{x})]$$

where $\alpha = \sum_{i=1}^{M} \alpha_i$. On the other hand, the variance of $\mathbf{f}$ can be used to measure the quality of such kernel approximation. We control the approximation error by imposing the constraint on the variance:

$$\mathrm{Var}[\mathbf{f}] \leq (\sum_{i=1}^{M} |\alpha_i|)^2 \mathrm{Var}[\mathbf{f}(\mathbf{x})] < \tau\mathbb{E}^2[\mathbf{f}]$$

where τ is a predefined threshold indicating the level of error tolerance. If this constraint is not satisfied, we will fall back to the original kernel prediction without using the approximation. In the worst case when fallback always happens, the proposed algorithm may suffer from additional cost for checking the constraints. However, this rarely happens in practice, and Theo-

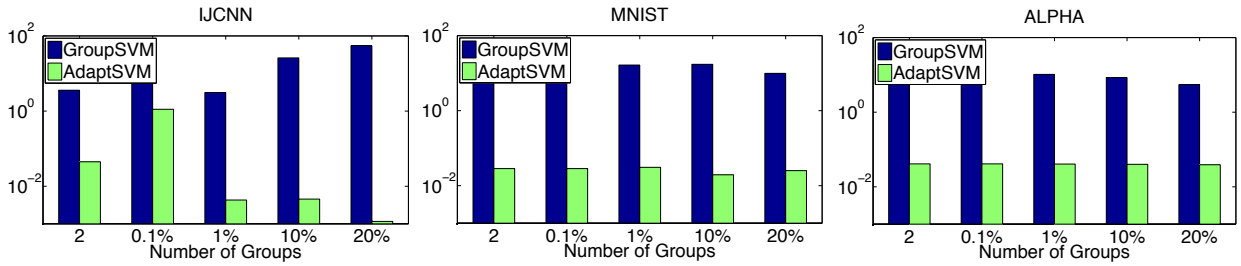

Figure 3. Comparison of mean squared error (MSE) against the number of SV groups. The y-axis is in log-scale.

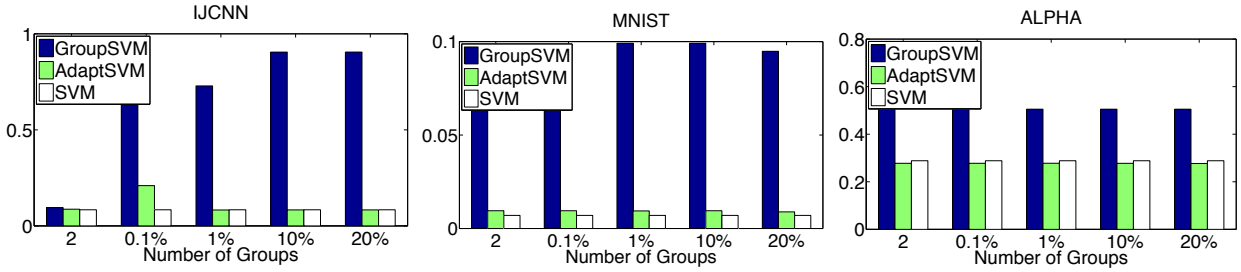

Figure 4. Comparison of classification error (Error) against the number of SV groups.

rem 1 indicates $\text{Var}[\mathbf{f}]/\mathbb{E}^2[\mathbf{f}]$ has only polynomial terms involved and thus can be computed efficiently.

Theoretically, as long as our assumptions are satisfied, the proposed algorithm can aggregate all the support vectors into only two points, one for positive SVs and the other for negative SVs. However, since real data are typically not uniformly distributed and the feature dimension is limited, our assumptions are not guaranteed to hold empirically. To mitigate this issue, we cluster the support vectors into a small number of groups, where the distances $r_{\mathbf{x}}$ are sufficiently close to each other. Though any clustering method can be applied, such as k-means or even random grouping, we choose hierarchical clustering in our experiments due to its simplicity. Formally, we equally cluster the set of positive and negative SVs into $J/2$ groups, each of which has its own mean μ_j and contains M_j SVs denoted as $\mathbf{x}_i^j$. We should limit the size of groups as much as possible in order to achieve the maximum speedup. The proposed algorithm is described in Algorithm 1.

3. Experimental results

This section presents our experimental results on three large-scale datasets, i.e., IJCNN[2], MNIST[3] and ALPHA[4], to demonstrate the effectiveness of our algorithm (**AdaptSVM**). Each data collection was split into two disjoint sets, one for training and another for testing. We considered a binary classification task between digit 0 and all other digits for MNIST, while using the standard settings for the other two collections. Table 1 summarizes the main characteristics of the datasets, including their distance ratios D/d and scaling factor λ for $\cos\theta_{\mathbf{x}}$ distribution, which show that our key assumptions are reasonable for these datasets.

The support vectors were learned from the training sets using *LibSVM* (Chang & Lin, 2001) with RBF kernel, then treated as input for all the prediction algorithms. Unless stated otherwise, we adopted the default RBF kernel parameters for training, i.e., $\rho = 1/K$ where K is feature number. We compared AdaptSVM with two baseline approaches, i.e., the standard SVM prediction by summing up kernel functions of every support vector (**SVM**), and the group-based prediction (**GroupSVM**), which uses the function value of cluster centroids as outputs without correction terms and dynamic fallback. For both GroupSVM and AdaptSVM, we applied single-linked hierarchical clustering to construct SV groups, with the number of groups varying based on experimental settings.

3.1. Empirical evaluation

To measure the quality of our adaptive approximation method, Figure 3 shows the mean squared error (MSE) between AdaptSVM and the original SVM pre-

[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools`
[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools`
[4] `ftp://largescale.ml.tu-berlin.de/largescale/`

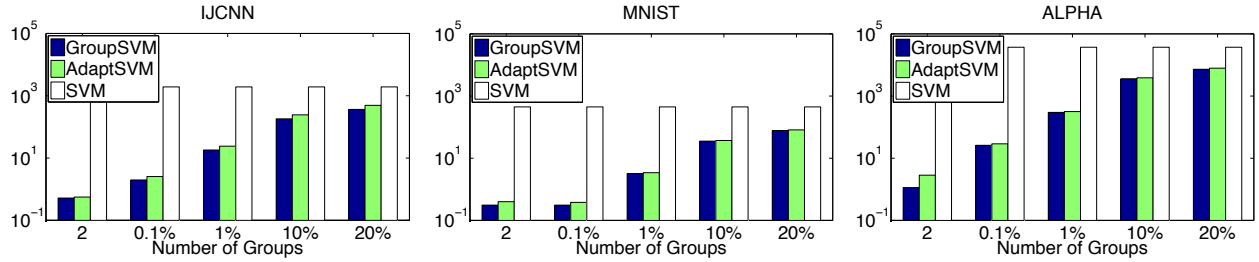

Figure 5. Comparison of prediction time against the number of SV groups. The y-axis is in log-scale.

Table 1. Details of the considered datasets: number of features, training samples, test samples, and original support vectors. Distance ratio and λ are also shown.

	K	Train	Test	SVs	D/d	λ
IJCNN	22	49990	91701	9597	1.27	0.25
MNIST	780	50000	10000	2233	1.95	16
ALPHA	500	50000	50000	47007	1.16	100

Table 2. Comparison of prediction speedup among ball tree, SV tree and AdaptSVM. The table shows the maximum achievable speedup with $< 0.5\%$ loss of accuracy. The median, average and best performance are reported.

	Ball tree	SV tree	**AdaptSVM**
Datasets	10	11	**3**
Median	3	18	**3400**
Average	60	20	**5833**
Best	500	50	**13000**

diction value, compared with the baseline GroupSVM method. The group size is growing from 0.1% to 20% of the original SV size. Figure 4 compares the classification error (Error) of AdaptSVM against both SVM and GroupSVM in similar settings. For both figures, we also include an extreme case where we only use two groups by aggregating all positive SVs into one group and all negative SVs into the other one. In this experiment, we set the error tolerance threshold $\tau = 1000$.

As can be observed from Figures 3 and 4, AdaptSVM, benefited from its accurate mean estimation, can significantly outperform GroupSVM in terms of both error metrics. In contrast, GroupSVM prediction is not as reliable and may severely degrade the prediction performance. In terms of classification errors, AdaptSVM almost always provide comparable performance to SVM with no considerable loss of accuracy. Its performance is in fact better than SVM for Alpha dataset. Surprisingly, the case with only two SV groups can produce a reasonable performance on par with baseline SVM in two out of three datasets. Since the original SV number of these datasets is in the order of $10^3 \sim 10^4$, we can obtain an up-to-10^4 speedup without a noticeable increase on errors.

For both types of errors, AdaptSVM tend to decrease with a growing number of SV groups, which can be explained by the fact that our assumptions can be better justified as the group size increases, leading to a more precise approximate prediction. The trend of such error drop is not as significant as expected, possibly because of the imperfect clustering output and a decreasing number of SV number in each group. The mere exception of error dropping trend happens in IJCNN with 0.1% of original SV size. Our investigation shows that the reason for this error peak stems from an unexpected bad clustering distribution, where many clusters contain one data point. In fact, this behavior appears to go away when the SVs were clustered with other more robust clustering methods.

Figure 5 compares the prediction time of SVM, GroupSVM and AdapSVM. As expected, a linear behavior is observed with a speedup close to the ratio between the original number of SVs and the number of groups. The figure also confirms that the overhead introduced by computing correction term and estimated variance is negligible with respect to prediction time.

When the estimated variance of the correction term is below the threshold τ for all the groups, we can achieve the maximum speedup by always using approximated prediction. However, this is not always desirable given the accuracy requirement. Therefore, we vary the threshold τ to obtain different degrees of approximation, and Figure 6 shows the accuracy change patterns when the number of groups is fixed to 1% of the original number of SVs. As τ increases from 0.5 to 500, the number of groups using the approximated prediction increases from 0 up to 1%. At the same time, the speedup increases from 1 up to 100, while the accuracy fluctuates within a small range, i.e., -0.3% to 0.5%, which shows the robustness of our algorithm.

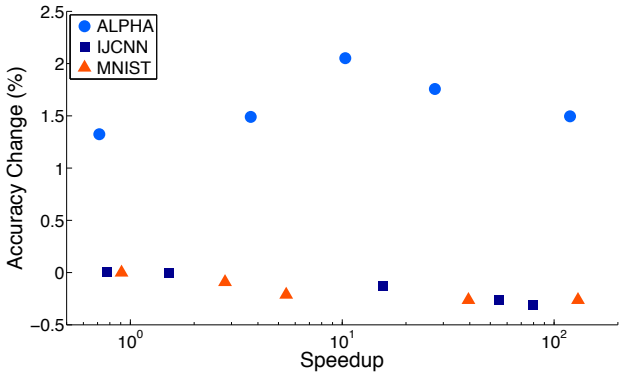

Figure 6. Accuracy change against speedup for different values of the threshold.

The proposed algorithm was also evaluated on two additional collections: ADULT[5] (K=123, 20073 SVs), and a second version of MNIST (K=580, 18711 SVs). Both datasets show AdapSVM can achieve significantly better approximation (10x smaller w.r.t MSE) than GroupSVM while preserving accuracy. The detailed analysis is skipped due to space constraints.

3.2. Comparison with related methods

We now compare the speedup of the proposed AdaptSVM algorithm with related SVM acceleration methods reported in the literature. Adopting the most common settings, we collected the maximum prediction speedup without a significant loss of accuracy ($< 0.5\%$) on each dataset. Table 2 compares the performance of AdaptSVM with the ball tree (Liu et al., 2003) and SV tree (Ukkonen, 2010) methods, which have a complete focus on improving the prediction process as well. The table shows the median, average, and largest speedup achievable with each method, where the number of ball tree and SV tree are directly referred from the related publications. We also reported the number of tested collections for each algorithm. We observe that AdaptSVM can on average achieve an additional two orders of magnitude speedup on top of the other two methods, the improvement of which is arguably significant. We believe this provides another data point to support the benefits AdaptSVM against other related methods.

4. Conclusion

In this paper, we propose an efficient prediction algorithm with performance guarantee for non-linear SVMs, called *AdaptSVM*. Powered by our theoretical analysis on approximating the mean and variance of kernel functions, it can selectively collapse the kernel function computation to be on a reduced set of support vectors, compensated by an additional correction term that can be easily computed on-line. To guarantee the prediction performance is not degraded, we also develop an adaptive fall-back strategy to switch back to original kernel computation if the estimated variance is larger than the maximum error tolerance. Our experimental results on three large-scale datasets show that AdaptSVM can speed up the prediction process up to 10^4 times with only $< 0.5\%$ accuracy loss.

Our future work includes extending the empirical analysis to more non-linear kernels, e.g. polynomial kernels, evaluating different clustering strategies for SV grouping, and employing a hierarchical grouping structure to dynamically choose various size of clusters.

[5] `http://archive.ics.uci.edu/ml/datasets/Adult`

Acknowledgments

This material is based, in part, upon work by Michele Cossalter and Lu Zheng supported by NSF grants CCF-0937044 and ECCS-0931978.

References

Beyer, Kevin, Goldstein, Jonathan, Ramakrishnan, Raghu, and Shaft, Uri. When Is "Nearest Neighbor" Meaningful? In *Database Theory - ICDT'99*, volume 1540 of *Lecture Notes in Computer Science*, pp. 217–235. Springer Berlin / Heidelberg, 1999.

Boley, Daniel. Training Support Vector Machine using Adaptive Clustering. In *Proc. of SDM'04*, pp. 126–137, Lake Buena Vista, FL, USA, Apr 2004.

Burges, Christopher J. C. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

Burges, Christopher J. C. and Scholkopf, Bernhard. Improving the Accuracy and Speed of Support Vector Machines. In *Proc. of NIPS'96*, pp. 375–381, Denver, CO, USA, Dec 1996.

Catanzaro, Bryan, Sundaram, Narayanan, and Keutzer, Kurt. Fast Support Vector Machine Training and Classification on Graphics Processors. In *Proc. of ICML'08*, pp. 104–111, Helsinki, Finland, Jul 2008.

Chang, Chih-Chung and Lin, Chih-Jen. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Greengard, L. and Rokhlin, V. A Fast Algorithm for Particle Simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

Hammersley, John M. The Distribution of Distance in a Hypersphere. In *The Annals of Mathematical Statistics*, volume 21, pp. 447–452. Institute of Mathematical Statistics, 1950.

Joachims, Thorsten and Yu, Chun-Nam. Sparse Kernel SVMs via Cutting-Plane Training. *Machine Learning*, 76(2):179–193, 2009.

Liu, Ting, Moore, Andrew W., and Gray, Alexander. Efficient Exact k-NN and Nonparametric Classification in High Dimensions. In *Proc. of NIPS'03*, Whistler, BC, Canada, Dec 2003.

Rai, Piyush, Daume, Hal, and Venkatasubramamian, Suresh. Streamed Learning: One-Pass SVMs. In *Proc. of IJCAI'09*, pp. 1211–1216, Pasadena, CA, USA, Jul 2009.

Schölkopf, Bernhard and Smola, Alexander J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001. ISBN 0262194759.

Tsang, Ivor .W., Kwok, James T., and Cheung, Pak-Ming. Very Large SVM training using Core Vector Machine. In *Proc. of AISTATS'05*, Barbados, Jan 2005.

Ukkonen, Antti. The Support Vector Tree. In *Algorithms and Applications*, volume 6060 of *Lecture Notes in Computer Science*, pp. 244–259. Springer Berlin / Heidelberg, 2010.

Yang, Changjiang, Duraiswami, Ramani, and Davis, Larry. Efficient Kernel Machines Using the Improved Fast Gauss Transform. In *Proc. of NIPS'04*, pp. 1561–1568, Vancouver, BC, Canada, Dec 2004.

Appendix: proof sketch of theorem 1

As the first step to derive $\mathbb{E}[f(\mathbf{x})]$, let us first derive the Hessian matrix of RBF function $f(\mathbf{x})$:

$$H(\mathbf{x}) = -2\rho[I - 2\rho D(\mathbf{x})D^T(\mathbf{x})]e^{-\rho\|D(\mathbf{x})\|^2}$$

where $D(x) = \mathbf{x} - \mathbf{z}$. Based on Assumption 1 and $D^T(x)R(x) = dr_{\mathbf{x}}\cos\theta_{\mathbf{x}}$, we can compute the limit of the integral form of the residual in Eq. (2) as follows:

$$\Delta = \lim_{r_{\mathbf{x}}\to r}\Delta_{\mathbf{x}} = \int_0^1 g(\beta,\mathbf{x})d\beta$$

where

$$g(\beta,\mathbf{x}) = -2\rho(1-\beta)r^2e^{-\rho(d^2+\beta^2r^2)}g_1(\beta,r,\theta_{\mathbf{x}})$$

$$g_1(\beta,r,\theta_{\mathbf{x}}) = e^{-2\rho\beta rd\cos\theta_{\mathbf{x}}}[1-2\rho(d\cos\theta_{\mathbf{x}}+\beta r)^2]$$

We apply the composite rectangular rule to integrate g numerically, i.e., we take N sampling points $\beta_n = n/N, n = 0\ldots N-1$ and replace the residual expectation into the following sum:

$$\mathbb{E}[\Delta] = \frac{1}{N}\left(\sum_{n=0}^{N-1}\mathbb{E}[g(\beta_n,\mathbf{x})]\right)$$

To compute each $\mathbb{E}[g(\beta_n,\mathbf{x})]$, we only need to compute the value of $\mathbb{E}[g(\beta_n,r,\theta_{\mathbf{x}})]$. We approximate the exponential function using a 6-order Taylor expansion. By further simplifying $r_n = r\beta_n$, we can have:

$$e^{-2\rho r_n d\cos\theta_{\mathbf{x}}} = \sum_{i=0}^{6}\frac{1}{i!}(-2\rho r_n)^i d^i\cos^i\theta_{\mathbf{x}} + O(\cos^6\theta_{\mathbf{x}})$$

We can then simplify the expectation of g_1 by ignoring the $O(\cos^6\theta_{\mathbf{x}})$ term:

$$\mathbb{E}\left[\sum_{i=0}^{6}\frac{1}{i!}(-2\rho r_n)^i d^i\cos^i\theta_{\mathbf{x}}[1-2\rho(d\cos\theta_{\mathbf{x}}+r_n)^2]\right]$$

Because Assumption 2 suggests the distribution of $\cos\theta_{\mathbf{x}}$ can be approximated as $\mathcal{N}(0,\sigma^2)$, we can compute expectation of g_1 based on the Gaussian integral:

$$\begin{aligned}\mathbb{E}[g_1] = {} & (1-2\rho r_n^2) + (-4\rho^2r_n^4 + 10\rho r_n^2 - 2)\rho\sigma^2 \\ & + (-4\rho^3r_n^6 + 18\rho^2r_n^4 - 12\rho r_n^2)\rho^2\sigma^4 \\ & + (-4/3\rho^4r_n^8 + 52/3\rho^3r_n^6 - 20\rho^2r_n^4)\rho^3\sigma^6\end{aligned}$$

where $\sigma^2 = \lambda/K$ is the variance of $\cos\theta_{\mathbf{x}}$. On the other hand, the expectation of residual term $\mathbb{E}[O(\cos^6\theta_{\mathbf{x}})] = O(1/K^6)$. Replacing both formulas back to $g(\beta,\mathbf{x})$, we can obtain $\mathbb{E}(f)$ in Theorem 1.

We can estimate the residual variance using similar techniques. The limit of gradient can be rewritten as

$$\lim_{r_{\mathbf{x}}\to r}\nabla_{\mathbf{x}} = \int_0^1 -2\rho e^{-\rho(d^2+\beta r^2)}h(d_\theta)d\beta$$

where $d_\theta = d\cos\theta_{\mathbf{x}}$, and $h(x) = (rx+\beta r^2)e^{-2\rho\beta rx}$. Based on Delta method, the variance of $h(d_\theta)$ converges to following value when $\cos\theta_{\mathbf{x}}$ approaches 0:

$$\begin{aligned}\mathrm{Var}[h(d_\theta)] &= [h'(d\cos\theta_{\mathbf{x}})|_{\cos\theta_{\mathbf{x}}=0}]^2d^2\mathrm{Var}[\cos\theta_{\mathbf{x}}] \\ &= (1-2\rho\beta^2r^2)^2r^2d^2\sigma^2\end{aligned}$$

Replacing $\mathrm{Var}[h(d_\theta)]$ we can obtain the variance of $\nabla_{\mathbf{x}}$:

$$\lim_{r_{\mathbf{x}}\to r}\mathrm{Var}[\nabla_{\mathbf{x}}] \le \int_0^1[-2\rho e^{-\rho(d^2+\beta r^2)}]^2\mathrm{Var}[h(d_\theta)]d\beta$$

which can then be approximated using numerical integration and the definition of $\gamma_n(r)$ in Theorem 1

$$\lim_{r_{\mathbf{x}}\to r}\mathrm{Var}[\nabla_{\mathbf{x}}] \le d^2e^{-2\rho d^2}\sigma^2\sum_{n=0}^{N-1}\left[2\rho e^{-\rho r_n^2}(1-2\rho r_n^2)\right]^2$$