Metric tree partitioning and Taylor approximation for fast support vector classification

Thang V. Pham Arnold W. M. Smeulders

ISLA, Faculty of Sciences, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, the Netherlands
{vietp, smeulders}@science.uva.nl

Abstract

This paper presents a method to speed up support vector classification, especially important when data is high-dimensional. Unlike previous approaches which focus on less support vectors, we partition the data space into local regions, and perform approximation by linear functions. The experimental results on 31 datasets show that the performance degrades marginally, while the speedup is significant, up to three orders of magnitude.

1. Introduction

The support vector method [10] requires the evaluation of a function $f(\boldsymbol{x})$, $\boldsymbol{x} \in R^d$, in the classification phase

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{s}_i) + b \tag{1}$$

where $\boldsymbol{s}_i \in R^d$, $\alpha_i \neq 0$, n is the number of support vectors, $b \in R$ is the bias, and $k(\boldsymbol{x}, \boldsymbol{s}_i)$ is a kernel function. When $k(\boldsymbol{x}, \boldsymbol{s}_i)$ is a linear kernel, $k(\boldsymbol{x}, \boldsymbol{s}_i) = \boldsymbol{s}_i^T \boldsymbol{x}$, (1) can be collapsed into a linear function. In other cases, n kernel evaluations have to be carried out where the complexity for each evaluation is typically $O(d)$. Hence, for non-linear SVM the computation of (1) is significant when n or d are large. In this paper, we aim to derive a method to compute a nearly identical approximation of (1) fast.

Without loss of generality, we consider the popular Gaussian kernel

$$k(\boldsymbol{x}, \boldsymbol{s}_i) = e^{-\frac{||\boldsymbol{x}-\boldsymbol{s}_i||^2}{\sigma^2}} \tag{2}$$

where σ denotes the scale. For this kernel, the fast Gauss transform [4] can be used to speed up the evaluation of (1). However, it is limited to low-dimensional cases, $d < 4$. A recent improvement works for $d < 30$ only [11]. Here we aim to tackle problems in high dimensions.

The method in [6] employs a data structure called ball-tree that organized *test data* in a way that bounds on (1) can be computed for a large number of test points, and hence SVM classification can be determined without full evaluation of (1). Nevertheless, the speedup is moderate, about two to five folds in most cases. Furthermore, this method does not allow instant, online classification as computational gain is achieved by processing a large set of data all at once.

The reduced set method [1, 8] represents a class of techniques that approximate (1) by another function $g(\boldsymbol{x})$ with $m < n$ support vectors

$$g(\boldsymbol{x}) = \sum_{i=1}^{m} \beta_i k(\boldsymbol{x}, \boldsymbol{t}_i) + c \tag{3}$$

where β_i, $\boldsymbol{t}_i$ and c are output of the approximation algorithms [5].

We take another approach to approximate (1) by multiple functions. The main issue is to split the data space into local regions so that the region containing a new test point can be located efficiently in run-time. For this purpose, we use a data structure called metric tree [7] which has been shown to work well for the nearest neighbor search problem.

In the next section, we will describe the approximation by the combination of metric tree partitioning and local Taylor approximation. In section 3 we summarize the classification using the approximated model. Then in section 4, the experiments are presented. Finally, the discussion in section 5 concludes the paper.

The 18th International Conference on Pattern Recognition (ICPR'06)
0-7695-2521-0/06 $20.00 © 2006 IEEE

2. Approximation by space partitioning

2.1. Metric tree partitioning

A set of points Q with size $|Q|$ is used to partition the data space. In the experiment we will use the training set as Q. Nevertheless, unlabeled data can be used.

Metric-tree is a binary tree that organizes data points in a hierarchical manner. The root node represents every points in Q. Starting from Q a metric tree is constructed recursively by performing node splitting until the node contains only one data point.

Let $\bar{Q} \subseteq Q$ denote the point set covered by a particular node of the tree being constructed, $|\bar{Q}| > 1$. To split $\bar{Q}$ into two disjoint subsets $\bar{Q}.l$ and $\bar{Q}.r$, we find two points $\boldsymbol{u}^*$ and $\boldsymbol{v}^*$ having the largest distance within $\bar{Q}$

$$(\boldsymbol{u}^*, \boldsymbol{v}^*) = \arg \max_{\boldsymbol{u},\boldsymbol{v} \in \bar{Q}} ||\boldsymbol{u} - \boldsymbol{v}|| \tag{4}$$

and subsequently the hyperplane $h_{\bar{Q}}(\boldsymbol{x})$ orthogonal to $(\boldsymbol{u}^* - \boldsymbol{v}^*)$ passing through the midpoint $(\boldsymbol{u}^* + \boldsymbol{v}^*)/2$ is used as the decision boundary (see Figure 1). Simple algebra gives us the formulae for $h_{\bar{Q}}(\boldsymbol{x})$

$$h_{\bar{Q}}(\boldsymbol{x}) = (\boldsymbol{u}^* - \boldsymbol{v}^*)^T \boldsymbol{x} + \frac{1}{2}(\boldsymbol{v}^{*T}\boldsymbol{v}^* - \boldsymbol{u}^{*T}\boldsymbol{u}^*) \tag{5}$$

and thus,

$$\bar{Q}.l = \{\boldsymbol{x} | \boldsymbol{x} \in \bar{Q}, h_{\bar{Q}}(\boldsymbol{x}) < 0\} \tag{6}$$

$$\bar{Q}.r = \{\boldsymbol{x} | \boldsymbol{x} \in \bar{Q}, h_{\bar{Q}}(\boldsymbol{x}) \geq 0\} \tag{7}$$

An alternative approach is to project all points in $\bar{Q}$ onto $(\boldsymbol{u}^* - \boldsymbol{v}^*)$ and then use the median point instead of the midpoint ensure a tree of depth $O(\log |Q|)$. Nevertheless, the former approach is more efficient, and in practice, also produces trees of depth $O(\log |Q|)$.

The number of distance computations is $O(|Q|^2)$. When $|Q|$ is large, one can resort to a linear-time heuristic as in [7].

2.2. Local Taylor approximation

At each leaf node in the tree $\bar{Q} = \{\boldsymbol{x}_0\}$, we perform a first order Taylor approximation of (1), resulting in a linear classifier. Specifically,

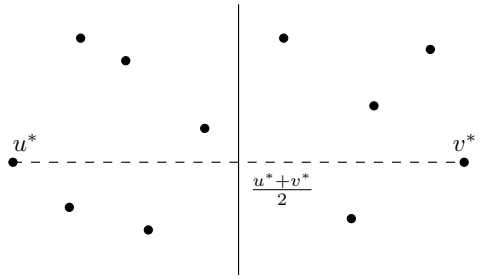
$$h_{\{\boldsymbol{x}_0\}}(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + (\boldsymbol{x} - \boldsymbol{x}_0)^T f'(\boldsymbol{x}_0) \tag{8}$$

$$= f'(\boldsymbol{x}_0)^T \boldsymbol{x} + \left\{ f(\boldsymbol{x}_0) - \boldsymbol{x}_0^T f'(\boldsymbol{x}_0) \right\} \tag{9}$$

where for the Gaussian kernel (2) we have

$$f'(\boldsymbol{x}_0) = -\frac{2}{\sigma^2} \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_0, \boldsymbol{x}_i)(\boldsymbol{x}_0 - \boldsymbol{x}_i) \tag{10}$$

Thus, at each left node there is a linear classifier as (5) in inner nodes.



$$h_{\bar{Q}}(x) = (u^* - v^*)^T x + \tfrac{1}{2}(v^{*T}v^* - u^{*T}u^*)$$

Figure 1. During metric tree construction, the pair of points with maximal distance is selected to guide the splitting of the current node.

3. Classification by the approximated model

The classification is straightforward. Given a new test point $\boldsymbol{x}$, we traverse the metric tree to reach a leaf node, and then apply the final classification.

Procedure to approximate (1):

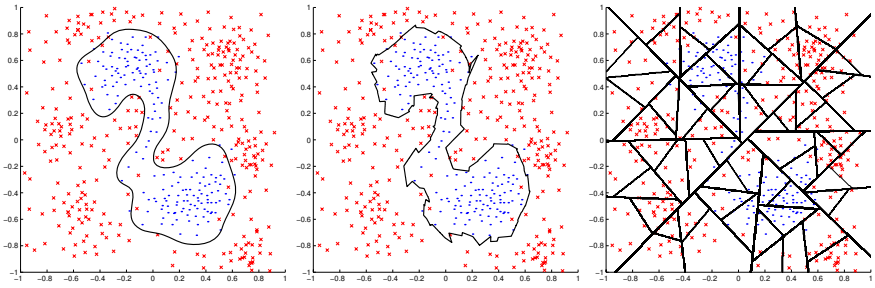
```
// starting from the root
Q̄ = Q
// traversing the tree
while |Q̄| > 1 do
    if h_Q̄(x) < 0
        Q̄ = Q̄.l
    else
        Q̄ = Q̄.r
    end if
end while
// now we are at a leaf node
return h_Q̄(x)
```

In the worse case, the number of dot products equals the maximum depth of the tree.

4. Experiments

4.1. A 2D example

First we create a 2D dataset to demonstrate the working of the algorithm. There are 512 data points in $[-1, 1]^2$ in total. In this case we use least squares SVM (LS-SVM) [9] to produce a decision boundary as depicted in Figure 2(a). In this case, all data points are support vectors, that is $n = 512$.



(a) (b) (c)

Figure 2. (a) A 2D artificial dataset and result by LS-SVM, effectively using all 512 data points as support vectors. (b) Result of the proposed method using a tree with maximum depth 14. (c) Partitions of the data space with the metric tree at depth 6.

Figure 2(b) shows the decision boundary produced by the proposed algorithm, and Figure 2(c) shows the data space as partitioned by the metric tree at level 6. It can be seen that the decision boundary of LS-SVM is well approximated by multiple linear boundaries. The advantage is that it requires, in the worse case, 14 dot products to classify a new example with the approximated boundary as opposed to 512 kernel evaluations required by LS-SVM (14 is the maximal tree depth).

4.2. On real-life data

We performed experiment with 30 two-class datasets (scaled) from [2] and a pedestrian dataset from [3]. The latter serves as an example for problem in high dimension where $d = 3780$. For all datasets, we train SVM models by LIBSVM 2.8 [2] using the default parameter settings.

Table 1 summarizes the datasets, and compares standard SVM learning and the proposed algorithm. The difference in accuracy is marginal in most cases, while the speedup is from one to three orders of magnitude. (Timing is done on a Pentium 4 CPU 2.40GHz).

4.3. Comparison with reduced set methods

Here we compare the new algorithm with the result in [5]. Specifically, the dataset number 8 (*ijcnn1*) and 14 (*a1a*) were used in [5] to evaluate four different reduced set methods. We refer the reader to [5] for details of these methods.

We repeat the experiment using the software provided by the authors. Table 2 contrasts the result of the four reduced set methods and ours using training parameters in [5]. (Note that the results differ from the ones in Table 1 as we used the default setting there). Again, one observes that our method approximates the standard SVM well. It is more accurate than all four reduced set methods for the *ijccn1* dataset and equal for the *a1a* dataset. Furthermore, the new algorithm produces classifiers which are 70 and 80 times faster than the reduced set methods on the two datasets.

5. Discussion

We have presented a method to speed up the support vector classification by data space partitioning and local approximation. The experimental results in table 1 on over 30 well-known datasets indicate that the proposed method produces classifiers that perform classification from 40 to over 3000 times faster than standard SVM at the expense of a marginal loss in accuracy. This improvement is a gain of a factor 70 and 80 over alternative approaches on two large datasets in speedup, while offers more accurate SVM approximation.

Acknowledgments

This research is supported by MultimediaN and the European Network of Excellence MUSCLE.

References

[1] C. Burges. Simplified support vector decision rules. In 13^{th} *ICML*, pages 71–78, 1996.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.

[4] L. Greengard and J. Strain. The fast Gauss transform. *SIAM J. Sci. Stat. Comput.*, 12(1):79–94, 1991.

No.	Dataset name		d	Data size Train	Test	Standard SVM default LibSVM 2.8 #SVs	acc.	test time	The proposed method avg. depth	acc.	test time	**abs. acc. diff.***	**speed gain factor**
1	fourclass		2	431	431	192	76.8	0.05	9.73	77.0	≈ 0	**0.2**	**110**
2	svmguide1		4	3089	4000	630	96.2	3.14	14.04	96.1	0.01	**0.0**	**290**
3	diabetes		8	384	384	237	75.3	0.16	11.33	75.3	≈ 0	**0.0**	**230**
4	breast-c		10	341	342	46	97.4	0.02	12.41	97.1	≈ 0	**0.3**	**40**
5	australian		12	345	345	137	87.3	0.09	10.48	87.5	≈ 0	**0.3**	**120**
6	heart		13	135	135	72	82.2	0.01	8.86	80.0	≈ 0	**2.2**	**40**
7	svmguide3		21	1243	41	605	12.2	0.06	12.46	12.2	≈ 0	**0.0**	**260**
8	ijcnn1		22	49990	91701	8971	92.8	1452.74	23.19	92.8	0.97	**0.0**	**1500**
9	german		24	500	500	317	75.4	0.38	10.70	74.2	≈ 0	**1.2**	**290**
10	ionosphere		34	175	176	88	92.1	0.05	11.25	85.8	≈ 0	**6.3**	**70**
11	covtype		54	20000	561012	12410	75.6	30776.00	18.25	75.6	9.18	**0.0**	**3350**
12	sonar		60	104	104	88	77.9	0.04	8.74	71.2	≈ 0	**6.7**	**60**
13	mushrooms		112	4062	4062	487	99.8	7.00	15.30	99.7	0.05	**0.0**	**140**
14	adult UCI	a1a	123	1605	30956	754	83.6	55.49	12.72	83.6	0.34	**0.0**	**160**
15	-	a2a	123	2265	30296	1067	84.0	72.43	13.46	83.9	0.30	**0.1**	**240**
16	-	a3a	123	3185	29376	1388	83.8	94.01	14.23	83.9	0.37	**0.1**	**250**
17	-	a4a	123	4781	27780	2000	84.0	127.12	15.02	84.0	0.30	**0.1**	**420**
18	-	a5a	123	6414	26147	2606	84.2	159.37	15.40	84.2	0.34	**0.0**	**470**
19	-	a6a	123	11220	21341	4341	84.2	227.88	16.31	84.2	0.37	**0.0**	**620**
20	-	a7a	123	16100	16461	6158	84.6	235.40	17.02	84.5	0.18	**0.1**	**1310**
21	-	a8a	123	22696	9865	8545	85.0	196.56	17.78	84.9	0.14	**0.1**	**1400**
22	-	a9a	123	32561	16281	11954	84.8	452.07	18.59	84.8	0.27	**0.0**	**1670**
23	w1a		300	2477	47272	208	97.0	23.38	22.53	97.0	0.45	**0.0**	**50**
24	w2a		300	3470	46279	275	97.0	30.02	21.72	97.0	0.48	**0.0**	**60**
25	w3a		300	4912	44837	354	97.0	35.17	23.72	97.0	0.33	**0.0**	**110**
26	w4a		300	7366	42383	499	97.0	49.06	25.82	97.0	0.52	**0.0**	**90**
27	w5a		300	9888	39861	625	97.1	52.86	26.20	97.1	0.48	**0.0**	**110**
28	w6a		300	17188	32561	1131	97.2	79.62	26.37	97.2	0.52	**0.0**	**150**
29	w7a		300	24692	25057	1564	97.4	93.21	27.65	97.3	0.59	**0.0**	**160**
30	w8a		300	49749	14951	2958	97.4	105.50	30.66	97.4	0.42	**0.0**	**250**
31	pedestrian [3]		3780	38868	19788	5026	94.5	1421.03	19.23	94.5	6.69	**0.0**	**210**

Table 1. Comparison on 31 different datasets. The average tree depth, accuracy (%) and computation time (in seconds) are for the whole test set. (*absolute accuracy difference)

Dataset	Standard SVM para. in [5] #SVs	acc.	Reduced SVM #SVs	accuracy M1	M2	M3	M4	test time	The proposed method avg. depth	acc.	test time	**accuracy contrast**	**speed gain factor**
a1a	642	84.2	160	84.1	84.2	83.7	84.2	12.21	12.72	84.2	0.15	***comparable***	80
ijcnn1	4555	99.0	200	92.2	97.1	93.3	96.5	32.60	23.19	98.1	0.50	***better***	70

Table 2. Comparison between the reduced set methods in [5] and our method.

[5] K.-M. Lin and C.-J. Lin. A study on reduced support vector machines. *IEEE Trans. on Neural Net.*, 14(6):1449 –1459, 2003.

[6] T. Liu, A. Moore, and A. Gray. Efficient exact k-NN and nonparametric classification in high dimensions. In *NIPS 16*, 2003.

[7] T. Liu, A. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *NIPS 17*, 2004.

[8] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Trans. on Neural Net.*, 10(5):1000–1017, 1999.

[9] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Process. Lett.*, 9(3):293–300, 1999.

[10] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc, 1998.

[11] C. Yang, R. Duraiswami, and L. Davis. Efficient kernel machines using the improved fast Gauss transform. In *NIPS 17*, 2004.