

## A NEW FAST TRAINING ALGORITHM FOR SVM

ZHI-JIE HE, LIAN-WEN JIN

School of Electronics and Information Engineering, South China University of Technology  
E-MAIL: lianwen.jin@gmail.com

### Abstract:

A fast SVM training algorithm is proposed in this paper. By integrating kernel caching, shrinking and using second order information, a fast Quadric Programming(QP) trainer is achieved. For traditional two-class SVM, the generalized error bound derived from Statistical Learning Theory(SLT) is computed and minimized for the selection of parameters, with the Zoutendijk(ZQP) idea and parallel method to speed up the process. For one-class SVM, a compression criterion is proposed to search the best kernel width automatically. Experiments demonstrate that the proposed method is significantly faster than LibSVM and requires less support vectors to achieve good classification accuracy.

### Keywords:

Support vector machine; statistical learning theory; Gaussian kernel

### 1. Introduction

Support Vector Machine(SVM) has become one of the most popular classification method in Machine Learning area in recent years. The basic concepts and theory of SVM has been described in many previous papers(e.g. [1][2][3][4]). Three major problems involved in practical usage of SVM training are speed, generalized error and free parameter optimization. Due to the endeavor of researchers in recent years, several methods are developed to speed up the SVM training in every aspect. Such as Sequential Minimal Optimization(SMO)[5], Kernel Caching[6], Shrinking[7] and using second order information[8]. In this paper, Multi-Operation Mixing is proposed as an effective integration of all of these technologies to design a fast Quadric Programming(QP) trainer for SVM.

Free parameter optimization for SVM is less investigated in previous works[9], and most of the existed method are based on a search to minimize the error estimated by cross-validation or leave-one-out method. There are two disadvantages of such methods. On one hand, time consumption is great due to the repeating training. On the other hand, there is uncertainty with the estimated error

which is relied on, and the chances, that the free parameter which is found with a minimum estimated error has large actual error, will be increasing when searching in free parameter space.

A deterministic criterion which can be calculated directly is needed to overcome these disadvantages. In two-class situation, the upper bound of generalized error derived from SLT(Statistical Learning Theory) [2] is the obvious choice. In one-class situation, which is less investigated, a compression criterion is proposed in this paper, which are deterministic and fast to be calculated.

The search procedure is also important due to the criterion function always has some local minimal. An exhaustive grid search is proposed for the sake of integrating the Zoutendijk Quadric Programming(ZQP) idea[10] and parallel method to mitigate the time consumption and guaranteeing the optimum. Also, for the precision required in free parameter optimization is always low, large search step is set to make the searching fast.

This paper is organized as follows: In Section 2, the proposed fast QP trainer for SVM is described. In Section 3, the suggested criterion is introduced and the method to determine training parameter is described. In Section 4, the proposed search algorithm for free parameter optimization is presented. In Section 5, experiments are carried out to show the effectiveness of the proposed method. Conclusion and discussion are made in Section 6.

### 2. A new quadric programming trainer

The idea of Support Vector Machine is to separate the training samples by a hyperplane with maximal margin. Actually, finding such a hyperplane is a Quadric Programming(QP) problem, so it is called QP trainer. The basic formulation of this quadric problem has been introduced in previous papers(e.g. [11]) and will be omitted in the paper. We just adopt C-SVM form in two-class case and OC-SVM[11][12] form in one-class case to introduce our algorithm.

Decomposition[13] is an important idea in QP, which divides the whole problem into a sequence of subproblems, and SMO[5] is the typical and most efficient one. The success of decomposition may due to the fact that the subproblem can be solved efficiently. And, kernel calculation is the main consumption in the whole training time. Caching the kernel is an effective way to speed up the algorithm.

In this paper, a two-step decomposition framework [7] is used to facilitate the maximal re-usage of kernel caching in the first step and the embedding of SMO in the second step. A new method, Multi-Operation Mixing(MOM), is proposed to make a smooth and fast integration of Kernel Caching, Shrinking and SMO. The framework of MOM training procedure is illustrated in figure 1.

The popular Least Recent Use(LRU) caching policy seems the most frequently used method for caching. Unfortunately, there is deficiency of LRU in SVM[6]. The Matrix Kernel Cache(MKC)[7] technology is employed to solve the deficiency of LRU. By the way of MKC, the first decomposition step is put on the whole problem, and the kernel of each sample pair in the subproblem is cached. It is expected that more support-vector samples are in the matrix kernel cache so that the cache can be reused effectively. Obviously, larger cache means more kernel re-usage.

We think that the samples in cache represent the scale of subproblem, and the subproblem should be gradually changed, or, gradually selected to smooth the algorithm in believing that smoothness brings fast convergence.

Before introducing the operation of gradually changing the subproblem, we first describe the data structure of caching. The samples in cache are organized as a bidirectional list with a head node and a tail node, for convenience of delisting. Also, there are two queues with FIFO(First In First Out) property to deal with samples outside the cache. One is called *unknown-queue*, which contains the samples never cached and is initially set as the all the sample with arbitrary order. The other is called *active-queue*, which is used to contain samples when cache reaches its memory limit. When we say that caching a sample, it means that first the sample dequeues, and then it is added to the tail of cache list, and finally the kernel cache is updated. If the cache is full, it will delist the head of cache list and push it to *active-queue* to get a place. So, we can cache one sample either from *unknown-queue* or from *active-queue* to change the subproblem gradually.

In the second decomposition step, the subproblem in the kernel cache is solved by SMO, which further decompose the subproblem into the two-sample problem called working pair. The center problem of SMO is how to select the working pair. In our algorithm, an efficient

approach is adopted, which uses second order information for working pair selection. Detail can be found in [8].

Also, shrinking technique, which eliminate the non-support-vector sample in advance to reduce the problem size and accelerate the convergence, is implemented by a *shrink-queue*, and a shrinking operation is inserted into every  $K$  SMO iterations(in our algorithm, we set  $K=100$ ). In shrinking operation, we check the shrinking condition of every sample in kernel cache, delist the samples satisfied it, and push them into *shrink-queue*. Detail of shrinking condition can be found in [11].

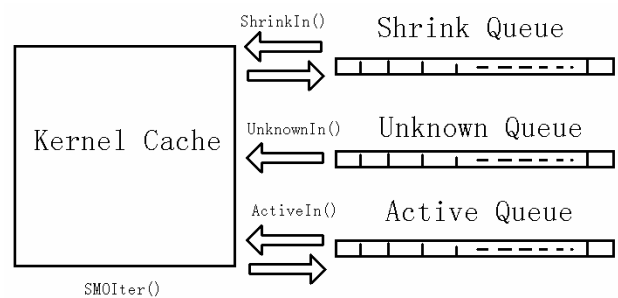


Figure 1. The framework of the proposed QP trainer

Now, integration of QP training algorithm is made by the process of Multi-Operation Mixing(MOM) of four types of operation, which are called in mix order with particular proportion. As seen in Figure.1, the first operation is *UnknownIn()*, which is defined as caching one sample from *unknown-queue*. The second and third is *ActiveIn()* and *ShrinkIn()* respectively. The last one is *SMOIter()*, which is defined as one SMO iteration in kernel cache. The first three ones are considered as the subproblem selection in the first step decomposition, and the last one can be regard as the further decomposition and quadric optimization.

In the process of MOM, the more times one type of operation is called the more proportion it has. The proportion control of these operations is a heuristic technique, which is got by the following consideration. First, it's appropriate to make more *UnknownIn()* in early stage, so that the algorithm will quickly get the general information of enough sample to form a rough result. Second, when the *active-queue* became large, which means the SMO takes too much load, then the *UnknownIn()* should be slowed down. Third, we give *ShrinkIn()* a certain proportion to see whether the shrunked-sample should be back to cache. Forth, *active-queue* is an extension of cache, so we should spend equal CPU time on them, and their only difference is *ActiveIn()* is  $fastK$  times slower than *SMOIter()* as for kernel calculation. The algorithm is given in figure 2.

```

First calculate the following variables in order
    unknowrate = unknowsize / samplesize;
    alistrate = (cachelistsize + activequeuesize)
                / (samplesize - unknowsize);
    unknowpart = unknowrate * (1 - alistrate);
    knownpart = 1 - unknowpart;
Then, the proportion of four operations is given by:
    SMOIter() : ActiveIn() : ShrinkIn() :
    UnknownIn()
    = (fastK* knownpart*5) : (knownpart*5)
      : knownpart : unknowpart
    
```

Figure 2. Proportion updating algorithm

In Figure2, *unknowsize* denotes the length of *unknown-queue*, and *samplesize* denotes the total size of sample, and *cachelistsize* denotes the length of the cache list. *fastK* denotes the times by which *SMOIter()* is faster than other operations, since it can get the kernel value in cache without kernel calculation. Therefore, it is appropriate to let *SMOIter()* have larger proportion. Also, the proportion will be changed after each operation, so it need to be updated. As this method is a heuristic technique, it is updated every *T* operations to weaken its time consumption (we set *T=100* in our experiments).

In summary, the proposed fast SVM training algorithm is given in Figure 3.

```

Step 1: Compute the proportion by algorithm shown in
fig.2.
Step 2: Repeat MOM T times (T=100 in our experiments)
according to the proportion of four types of operations,
and call them in random order.
Step 3: repeat 1 and 2 until stopping condition is met.
    
```

Figure 3. The new fast SVM training algorithm

### 3. Criterion calculation for automatically parameter determination

Although in section 2 we have showed an efficient QP training algorithm, there are still free parameters needed to be tuned. In this paper, Gaussian kernel is considered, in which the width is the main free parameter. A criterion is needed to determine the goodness of a classifier with specific parameter. Search the parameters and minimize the error estimated by Cross Validation is frequently used. But that is slow for repeated training, and the error will rely on

the selection of validation set in probability, that may cause a general over-fitting while searching the parameter space.

The upper bound of generalized error derived from SLT (Statistical Learning Theory) is adopted as the criterion to determine the free parameters for C-SVM in our method. As a deterministic criterion, it won't be affected by probability distribution. Therefore it should get a stable result with better generalization performance. There is just a tiny time consumption needed immediate after a single train. For one-class case, another criterion is proposed in section B.

#### 3.1. C-SVM

In two-class case, the upper bound of generalized error is adopted as the criterion. Recall from STL, we can choose some  $\eta$  ( $0 \leq \eta \leq 1$ ) such that with probability  $1-\eta$ , the following bound holds:

$$P_{error} \leq P_{emp} + \frac{\epsilon}{2} \left( 1 + \sqrt{1 + \frac{4P_{emp}}{\epsilon}} \right) \quad (1)$$

$$\text{where, } \epsilon = 4 \frac{h(\ln \frac{2l}{h} + 1) - \ln \frac{\eta}{4}}{l}$$

$P_{error}$  is the actual generalized error.  $P_{emp}$  is the empirical error obtained in training samples.  $l$  is the size of training samples.  $h$  is the VC-dimension of the SVM classifier, which has the following bound.

$$h \leq \frac{R^2}{\Delta^2} + 1 \quad (2)$$

$R$  is the minimal radius of the hypersphere, that contains all the support-vector samples.  $\Delta$  is the margin of the hyperplane. The hypersphere and hyperplane are both defined in feature space generated by Gaussian kernel, and can be calculated by geometry. Actually, computing  $R$  is another QP, but this can be simplified by calculating the maximal distance  $D$  among the support-vector samples, because the following holds.

$$R \leq D \Rightarrow h \leq \frac{D^2}{\Delta^2} + 1 \quad (3)$$

To further speed up the process, a heuristic algorithm shown in Figure 4 is proposed to approximate the maximal distance  $D$ .

Initially, we defined the distance function  $D(x_1, x_2)$  in the feature space by kernel function.

$$D(x_1, x_2) = \sqrt{2 - 2\text{Kernel}(x_1, x_2)}$$

And let  $k = 1$ .

Step1: for each sample, find  $j$  that maximize  $D(x_j, x_k)$

Step2:  $k \leftarrow j$

Step3: Repeat Step1 and Step2  $N$  times

Output: the final value of  $D(x_j, x_k)$

Figure 4. An algorithm computing the maximal distance  $D$

There is also a tiny problem in choosing  $\eta$ . We just use (4) to simplify the formulation and make  $\eta$  adaptively be smaller when lower  $h/l$  rate.

$$\eta = 2h/l \quad \text{or} \quad \ln \frac{\eta}{4} = -\ln \frac{2l}{h} \quad (4)$$

### 3.2. OC-SVM

In one-class case, there are still two free parameters [12]. The idea of tuning them by a criterion is the same as two-class case. Before introducing the criterion proposed, we must recall the theory of OC-SVM. OC-SVM uses a hyperplane to separate training samples from origin with maximal margin. In another sense, the hyperplane generates a region that covers most of the training samples, while the region is of minimal volume. To do so, we can estimate the support of a high-dimensional distribution. It is expected to find a region that is small in volume and high in coverage rate.

The first free parameter is used to control the fraction of outlier samples, which is not covered by the region. The selection of this is based on the consideration of the tradeoff between coverage rate and the volume of the region. This tradeoff should be decided by the algorithm user, so this parameter is set as an input value.

The other free parameter is used to select different kernel. Gaussian kernel is adopted here, and this free parameter should become the width of the Gaussian kernel. The smaller the kernel width is, the stronger the region representation ability will be, but the more chances of over-fitting. So, the criterion proposed to solve the selection of the kernel width is the following.

$$\min[\text{Entropy}(P_{\text{mod}}) + N \times P_{\text{mod}} \times \text{KLDistance}(r, \text{SVrate})]$$

where

$$\text{KLDistance}(x, y) = x \ln \frac{x}{y} + (1-x) \ln \frac{1-x}{1-y} \quad (5)$$

$$\text{Entropy}(x) = x \ln \frac{1}{x} + (1-x) \ln \frac{1}{1-x}$$

Where  $N$  is the training sample size.  $P_{\text{mod}}$  is the model pri-probability, and  $\text{Entropy}(P_{\text{mod}})$  is large when the model is complex.  $r$  is just the first parameter used to control the fraction of outlier as an user input.  $\text{SVrate}$  is the support-vector rate got from dividing the number of support-vector samples by the total size of samples, which can be look on as an empirical fraction of outlier that we actually got. So,  $\text{KLDistance}(r, \text{SVrate})$  is the K-L distance of these two fractions. The nearer these two values are, the better the result we got, and the smaller the K-L distance will be, and vice versa.

This criterion gets idea from information compression by Minimum Description Length(MDL) principle. Considering a new test sample set of size  $N$ , we need to store which are the outliers most efficiently. First, we must store the information that whether this set is under the distribution has been modeled, which causes  $\text{Entropy}(P_{\text{mod}})$  bits on average. When it is true with probability  $P_{\text{mod}}$ , we supposed to have a fraction  $r$  of outliers but actually the fraction is  $\text{SVrate}$ , so extra bits are needed to represent this difference.

The estimation of  $P_{\text{mod}}$  is a tough problem. In our method, the uniformed bias of the hyperplane, which is bias divided by the sum of alpha, is used as the  $P_{\text{mod}}$  heuristically. It is worth to notice that a complex model is always with a small bias.

$$P_{\text{mod}} = \frac{\text{bias}}{\sum \alpha} \quad (6)$$

### 4. Free parameter optimization

As we have seen that there are two free parameters in C-SVM formulation. One is the width  $\sigma$  of the Gaussian Kernel. The other is the coefficient  $C$  of data penalty term. Is there any explicit relation among these two factors and the criterion function described in the above section?

No, the criterion function is lack of characteristic in the two parameters space with many local minimal. So, a grid search is needed to achieve a stable and appropriate optimal solution.

To make the grid search fast, a parallel method and ZQP[10] idea is used. We use parallel QP trainer to solve QP problems with different width  $\sigma$  in separate memory. For a specific width, the ZQP idea indicates that the support vectors in problem with larger  $C$  mainly come from support vectors in problem with smaller  $C$ . So, we keep the result of the small- $C$  problem, then put all non-support vector into *shrink-queue* (recalled that *shrink-queue* is supposed to store non-support-vector), then go on to train the problem with

larger C. By doing this, the large-C problem contain much fewer samples and can be solved much faster.

In OC-SVM case, there is just one free parameter. And we just use parallel method to search different width  $\sigma$  of the Gaussian Kernel.

We search  $\sigma$  from large to small, but how small the searching should stop is an important issue to be considered. We determine the lower bound of  $\sigma$ -searching by the following consideration. Because small  $\sigma$  makes a complex function, which cause over-fitting, we use a threshold of VC-dimension(got by method in Section 3) to prevent too small  $\sigma$ . If VC-dimension is larger than the threshold, the searching terminated. Vapnik suggest the VC-dimension should not be over sample size multiply 0.1, otherwise it's easy to over-fitting.

In our implement, the initial value of  $\sigma$  is set to twice the standard variance of the samples, and the initial C is set to 1. Each step of C is amplified by factor 2, and  $\sigma$  is multiplied by factor 0.7.

## 5. Experiment

We implement our method using c++ language and we call it MySVMLIB. By performing some experiments on UCI database [16], we compare our method with the LIBSVM v2.84[11] in terms of the training speed and classify accuracy. To extend SVM to multi-class problem, we use one-against-one method with logit committee voting strategy.

The comparisons between the proposed method and LIBSVM are carried out in the following two aspects.

- a. MySVMLIB has automatic free parameters selection based on VC-dimension. LIBSVM just output a cross-validation error, and doesn't specify the free parameters algorithm.
- b. Both have the same SMO using second order information. But MySVMLIB uses Matrix Kernel cache with MOM. LIBSVM uses the LRU cache technique with Shrinking policy

To make LIBSVM comparable, a  $11 \times 5$  grid search is used to select the free parameters by minimize the cross-validation error provided by LIBSVM. The initial value and the search step of the free parameters are the same as MySVMLIB.

In experiment, the recognition rate is got by a 10-fold cross validation. The CPU Time is the total training time including the time spent on cross validation, the one against one combination and the free parameters selection. The SV Rate is the average support vector rate, which is the number

of support vector divided by the number of all the training samples.

The intension of the experiment is to compare the proposed parameter selection algorithm with the commonly used grid search algorithm. It is carried on several small scale(less than 1000 samples) UCI dataset[16]. So, there are always enough memory to cache all sample, so both the caching policy(MOM and LRU) won't take effect, and both program have the same SMO with shrinking. The results are shown in Table 1.

From Table 1, it can be seen that, the proposed method is far better than the LIBSVM(version 2.84) especially in speed performance. It can be also seen that our algorithm required much less support vectors. The recognition accuracy of our algorithm outperform that of LIBSVM for most dataset except "australian" and "vehicle".

Table 1. Experiment on small sets

Database	Recognition Rate		CPU Time (s)		SV Rate	
	MySVMLIB	LIBSVM	MySVMLIB	LIBSVM	MySVMLIB	LIBSVM
australian	63.6%	68.7%	15.4	298	44.7%	65.4%
balance	89.9%	89.6%	9.14	159	33.1%	100%
cancer	96.8%	86.7%	5.91	101	12.8%	66.7%
diabetes	74.9%	66.1%	14.3	209	59.3%	82.0%
german	72.2%	69.4%	33.4	511	68.4%	97.6%
glass	67.8%	60.3%	2.23	32.8	37.1%	95.4%
heart	67.0%	58.9%	12.3	44.3	65.2%	99.5%
liver	69.9%	65.5%	8.45	49.6	68.6%	97.3%
vehicle	63.9%	67.1%	14.7	408	47.4%	85.8%
votes	94.9%	77.0%	2.17	66.1	27.5%	79.6%
vowel	90.9%	68.4%	5.39	204	34.9%	99.3%
wine	76.4%	75.8%	0.827	14.6	28.0%	68.4%

Generally speaking, experiments show that the proposed SVM training method is much faster, and requires less support vectors to achieve a good enough classification accuracy.

## 6. Conclusion and Discussion

In this paper, a fast and parameter-free SVM training algorithm is presented. Multi-Operation Mixing is proposed to effectively integrate the advancing SVM training techniques together, including SMO with using second order information, matrix kernel caching and shrinking. Experiment shows that it is faster than the SMO with LRU cache method. Further, the upper bound of generalized error derived from SLT is found to be better than estimated error by cross-validation as a minimization criterion for free

parameters selection. At last, by using ZQP idea and parallel method, the free parameters searching process can be speed up dramatically.

Though SVM training is intensive investigated and we have achieved a "fast" one, it is still too slow to tolerance in some cases, such as huge classification task that contains millions of training samples(e.g. handwritten Chinese character recognition problem, or face recognition). How to further speed up the proposed algorithm is a research topic worth for study.

### Acknowledgments

This work is supported in part by the research fundings of NSFC (no. U0735004, 60772216) and GDNSF (no. 07118074).

### References

- [1] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, Springer Netherlands. Vol 2, Number 2. June, 1998
- [2] Vapnik, V. *Statistical learning theory*. John Wiley and Sons, Inc., New York, 1998.
- [3] Pavel Laskov. Feasible direction decomposition algorithms for training support vector machines. *Machine Learning*, 46(1), 2002:315-349
- [4] Norikazu Takahashi, Tetsuo Nishi. Rigorous proof of termination of smo algorithm for support vector machines. *IEEE Transactions on Neural Networks*, Vol. 16, No. 3, May 2005.
- [5] John C.Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods- Support Vector Learning*, Cambridge, MA, MIT Press, 1998
- [6] G. W. Flake and S. Lawrence. Efficient svm regression training with smo. *Machine Learning*, 46(1-3):271-290, March 2002.
- [7] Jian-xiong Dong, Adam Krzyzak, Ching Y.Suen. A fast svm training algorithm. *First International Workshop on SVM*, Springer-Verlag Berlin Heidelberg 2002:53-67.
- [8] Rong-En Fan, Pai-Hsuen Chen, Chih-Jen Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research* 6,2005:1889-1918
- [9] Matthew Boardman and Thomas Trappenberg. A heuristic for free parameter optimization with support vector machines. *IEEE International Joint Conference on Neural Networks*, Canada, July, 2006:1337-1344
- [10] Rodolfo E.Ibarra Orozco, Neil Hernandez-Gress, Juan Frausto-Solis, Jaime Mora Vargas. Increasing the training speed of svm the zoutendijk algorithm Case. F.F.Ramos et al.(Eds.): *ISSADS 2005*, LNCS 3563, 2005:312-320
- [11] Chih-Chung Chang, Chih-Jen Lin, LIBSVM: a library for support vector machines. 2007. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>
- [12] Bernhard Schölkopf, John C.Platt, John Shawe-Taylor, Alex J. Smola, Robert C.Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation* 13, 2001:1443-1471
- [13] Pai-Hsuen Chen, Rong-En Fan, Chih-Jen Lin. A study on smo-type decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*,17,July 2006:893-908
- [14] C.-L. Liu, K. Nakashima, H. Sako, H. Fujisawa, Handwritten digit recognition: investigation of normalization and feature extraction techniques, *Pattern Recognition*, 37(2): 265-279, 2004
- [15] Tong Luo, Lawrence O.Hall, Dmitry B.Goldgof, Andrew Remsen. Bit reduction support vector machine. *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)*,2005
- [16] C. J. Merz and P. M. Murphy et al., UCI repository of machine learning databases, Univ. California at Irvine. URL <http://www.ics.uci.edu/~mllearn/MLRepository.html>