A New Support Vector Compression Method Based on Singular Value Decomposition

Sang-Hun Yoon, Chun-Gi Lyuh, Ik-Jae Chun, Jung-Hee Suk, and Tae Moon Roh

In this letter, we propose a new compression method for a high dimensional support vector machine (SVM). We used singular value decomposition (SVD) to compress the norm part of a radial basis function SVM. By deleting the least significant vectors that are extracted from the decomposition, we can compress each vector with minimized energy loss. We select the compressed vector dimension according to the predefined threshold which can limit the energy loss to design criteria. We verified the proposed vector compressed SVM (VCSVM) for conventional datasets. Experimental results show that VCSVM can reduce computational complexity and memory by more than 40% without reduction in accuracy when classifying a 20,958 dimension dataset.

Keywords: RBF SVM, SVD, vector compression.

I. Introduction

1. Support Vector Machine

Support vector machines (SVMs) [1] are popular techniques for machine learning classification. While SVMs have good accuracy and generalization properties, they can be slow to classify new examples relative to other machine learning methods such as neural networks. An SVM must compute the dot product of each query example with each of the support vectors (SVs), which can number in the hundreds or thousands. Previous research has focused on methods to speed up SVM evaluation by means of a reduction in the number of SVs [2]-[5].

Manuscript received Sept. 15, 2010; revised Nov. 26, 2010; accepted Dec. 9, 2010.

This work was supported by the IT R&D program of MKE/KEIT [KI002162, Multi-Camera Based High Speed Image Recognition SoC Platform].

Sang-Hun Yoon (phone: +82 42 860 1352, email: shyoon11@etri.re.kr), Chun-Gi Lyuh (email: cglyuh@etri.re.kr), Ik-Jae Chun (email: ijchun@etri.re.kr), Jung-Hee Suk (email: jhsuk@etri.re.kr), and Tae Moon Roh (email: tmroh@etri.re.kr) are with the Convergence Components & Materials Research Laboratory, ETRI, Daejeon, Rep. of Korea.

doi:10.4218/etrij.11.0210.0349

2. Problem Definition

SV set $\mathbf{X}_s$ is a two dimensional matrix whose row vector is made up of an SV by

$$\mathbf{X}_s = \left[\mathbf{x}_{s,1}{}^T, \mathbf{x}_{s,2}{}^T, \cdots, \mathbf{x}_{s,M}{}^T \right]^T, \quad (1)$$

where $\mathbf{x}_{s,m}$ is the m-th support vector. Also, $\mathbf{x}$ is the input vector to be classified. In this letter, we consider only a binary radial basis function (RBF) kernel SVM since others can be extended easily. The decision rule for a binary RBF SVM is

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{M} \alpha_i y_i e^{-\gamma \|\mathbf{x}_{s,i} - \mathbf{x}\|^2} + b \right). \quad (2)$$

An SVM is defined by M+1 parameters: a weight α_i, associated with each training example, and a bias term b.

Since the norm value in the exponential part of $f(\mathbf{x})$ in (2) must be calculated for each support vector $\| \mathbf{x}_{s,i} - \mathbf{x} \|$, the computations are concentrated on the difference and square operations in order to obtain this norm value, as shown in Table 1. In Table 1, the third column shows the required computations using a conventional SVM. The computations required to obtain norm values are almost N times that of other methods where N is a dimension of vector $\mathbf{x}$.

When we classify the image sources, vector dimension N usually reaches up to tens of thousands. If we assume that N and M are 2,000 and 1,500, respectively, the total number of required register words and squarers are both 3 million. This is not easy to implement on embedded systems for real-time operations. In this letter, we focus on reducing the computational complexity to make it possible to classify high dimensional feature vectors in real-time on embedded systems.

Table 1. Computational complexity for classification.

	Computation	orgSVM	VCSVM
1	Vector dimension	N	N
1-1	Compressed dimension		P
2	Support vectors	M	M
3	Register for SVs	$N\times M$	$(N+M)\times P$
3-1	Multiply **X** by $\mathbf{V}_s$		$N\times P$
3-2	Accumulate for $\mathbf{XV}_s$		$(N-1)\times P$
4	Squares for norm	$N\times M$	$P\times M$
5	Differences for norm	$N\times M$	$P\times M$
6	Multiply – γ	M	M
7	Exponential func.	M	M
8	Multiply a_i	M	M
9	Accumulate	$M-1$	$M-1$
10	Add b	1	1

II. Previous Work

Since classification time scales with the number of support vectors used, one approach is to construct a reduced-set SVM that approximates a given SVM using far fewer support vectors [3]. Reduced-set vectors can also be used to selectively spend greater effort on examples that likely belong to a positive class, such as image regions likely to contain a face for face detection applications [6]. The ProgSVM method considers examples from all classes, and it can be applied to multiclass problems. The nearest support vectors method [2] is similar to ProgSVM in that it proposes an incremental classification process.

However, all of these methods consider only the number of support vectors. For image classification applications, the vector dimension of each support vector is larger than the number of support vectors. The reduced vector dimension in addition to the reduced number of SVs can be a solution for a real time image classifier.

III. Proposed Vector Compression Method

1. Vector Decomposition

The norm operation part can be decomposed using singular value decomposition (SVD) [7] as in

$$\left\|\mathbf{X}_s - \mathbf{X}\right\| = \left\|\mathbf{U}_s\mathbf{D}_s\mathbf{V}_s^T - \mathbf{X}\mathbf{V}_s\mathbf{V}_s^T\right\|, \quad (3)$$

where

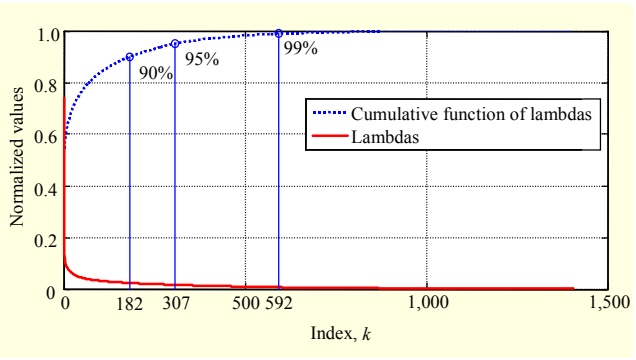

Fig. 1. Cumulative function of lambda.

$$\mathbf{X}_s = \left[\mathbf{x}_{s,1}{}^T, \mathbf{x}_{s,2}{}^T, \cdots, \mathbf{x}_{s,M}{}^T\right]^T = \mathbf{U}_s\mathbf{D}_s\mathbf{V}_s^T,$$

$\mathbf{U}_s, \mathbf{V}_s$: orthonormal and unitary matrix,

$$\mathbf{D}_s = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \lambda_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & \lambda_n \end{bmatrix}, \quad \lambda_i \geq \lambda_{i+1}.$$

The diagonal components λ_i indicate the eigen values for the eigen vector $\mathbf{V}_s$, and the total vector energy can be represented as $E(n)=\Sigma_{i=1}^{n}\lambda_i^2$. Thus, we can obtain the approximated matrix $\mathbf{X}_s$ simply by replacing λ_k through λ_n with zeros in $\mathbf{D}_s$ whose total vector energy is reduced by the amount of $E_r(k,n)=\Sigma_{i=k}^{n}\lambda_i^2$. Figure 1 shows an example of λ_s and their cumulative energies from the SVs. This example was extracted from the pedestrian images featured by the histogram of oriented gradient (HOG [9]) method. In Fig. 1, the solid line represents λ_k, and the dashed line means $E(k)$. As one can see in Fig. 1, almost all energy (90%, 95%, and 99%) is concentrated on the first P (182, 307, and 592) elements, respectively.

2. Vector Compression Method

Based on the vector decomposition given previously, we can obtain (4) with minimum signal energy loss with only P dimension vectors:

$$\begin{aligned} \left\|\mathbf{X}_s - \mathbf{X}\right\| &= \left\|\mathbf{U}_s\mathbf{D}_s\mathbf{V}_s^T - \mathbf{X}\mathbf{V}_s\mathbf{V}_s^T\right\| \\ &= \left\|\left(\mathbf{U}_s\mathbf{D}_s - \mathbf{X}\mathbf{V}_s\right)\mathbf{V}_s^T\right\| \\ &= \left\|\left(\mathbf{U}_s\mathbf{D}_s - \mathbf{X}\mathbf{V}_s\right)\right\| \\ &\approx \left\|\left(\mathbf{U}_s\mathbf{D}_s(:,1:P) - \mathbf{X}\mathbf{V}_s(:,1:P)\right)\right\|. \end{aligned} \quad (4)$$

The norm for each support vector can also be compressed with minimum signal energy loss:

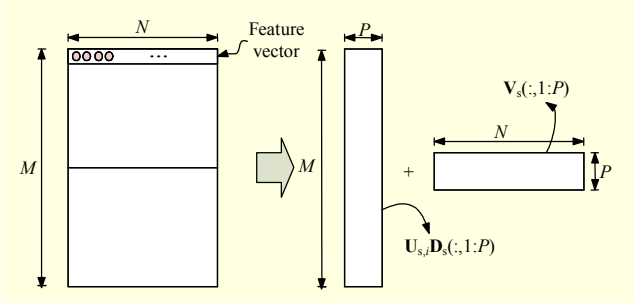

Fig. 2. Required registers for support vectors.

$$\left\|\mathbf{X}_{s,i}-\mathbf{X}\right\| \approx \left\|\left(\mathbf{U}_{s,i}\mathbf{D}_s\left(:,1:P\right)-\mathbf{X}\mathbf{V}_s\left(:,1:P\right)\right)\right\|. \quad (5)$$

In (5), $\mathbf{U}_{s,i}\mathbf{D}_s(:,1:P)$ can be calculated before classification, and $\mathbf{X}\mathbf{V}_s(:,1:P)$ needs to be calculated only once for all support vectors. The required registers for support vector storage are then reduced from $M\times N$ to $(M+N)\times P$ as can be seen in Fig. 2. In Fig. 2, the required register block whose size is $M\times P$ is used for compressed support vector storage and the $N\times P$ sized one is used to store eigen vectors for vector compression.

The computational complexities for the SVM can be calculated as in Table 1. We call the proposed SVM vector compressed SVM (VCSVM) since we compressed the SVs in terms of their vector dimension. As you can see in Table 1, the items from 3-1 and 3-2 are overheads of VCSVM. So, we choose P as less than $MN/(M+N)$.

3. VCSVM Training/Classification

Figure 3 shows a flow chart of VCSVM. In Fig. 3, there exists a step named 'Reduce SVs'. It reduces support vector sets by using the method described in [3] as commented in section I. It can reduce computational complexity at the expense of reduced accuracy. In Fig. 3, $\mathbf{x}_t$ is a training vector, S_{SV} is the original support vector, T is the number of training vectors, N is vector dimension of each training vector, M is the number of support vectors, b is bias, S is reduced-set support vector, $\mathbf{x}$ is input vector, $f(\mathbf{x})$ is classified result of $\mathbf{x}$, and P is compressed support vector dimension. Tilde (~) means 'approximated'.

IV. Experimental Results

1. Experimental Environment

We tested the proposed VCSVM with the datasets in [9]. Since the proposed algorithm was designed for high dimensional vector classification, we choose relatively high dimensional datasets in [9]. We used LIBSVM [10] for vector training and adopted SPRToolbox [11] to reduce support vector sets. Details of experimental examples are shown in Table 2.

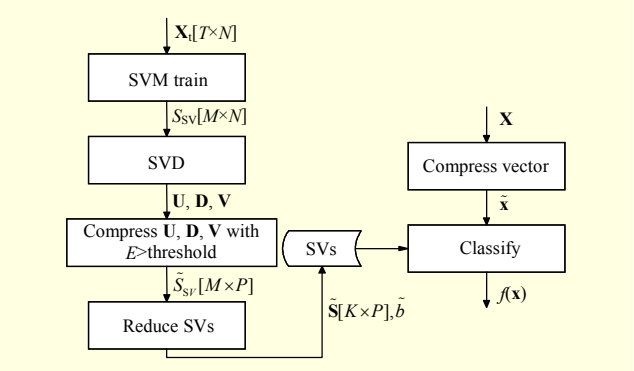

Fig. 3. Flow chart of VCSVM train/classify.

Table 2. Parameters used in experiments.

Dataset	Vector dimension	Training vectors	Test vectors
Splice	60	1,000	2,175
Mushrooms	112	1,000	7,124
A1a	123	1,605	30,956
Leukemia	7,129	38	34
Real-sim	20,958	1,000	71,309

Table 3. Computational complexity for Splice dataset classification.

	Splice (total SVs: 362, P: 51) set [9], threshold=99%			
	Original SVM	RSSVM [3]	VCSVM wo RS	VCSVM w RS
Req. reg. word (relative)	21,720 (100%)	4,380 (20%)	21,522 (99%)	4,947 (23%)
Req. mult. times (relative)	22,444 (100%)	5,104 (23%)	22,246 (99%)	5,671 (25%)
Accuracy (relative)	88.46% (100%)	88.55% (100.1%)	83.45% (94.34%)	82.99% (93.82%)

2. Results

Tables 3 to 6 show the computational complexities and accuracies of the original SVM and VCSVM according to Table 1 when we classify the datasets in Table 2. We compared the experimental results of the original SVM using LIBSVM [10], the reduced-set SVM (RSSVM) using SPRToolbox [11], and the proposed VCSVM with/without the 'Reduce SVs' step in Fig. 3.

For fair comparison, the parameters of RSSVM are set to have almost same accuracy with 'VCSVM with RS'. In Tables 3 through 6, the proposed VCSVM shows superior performance

Table 4. Computational complexity for Mushroom data set classification.

	Mushroom (total SVs: 224, *P*: 46) set [9], threshold=99%			
	Original SVM	RSSVM [3]	VCSVM wo RS	VCSVM w RS
Req. reg. word (relative)	25,088 (100%)	3,584 (14%)	15,456 (62%)	5,612 (22%)
Req. mult. times (relative)	25,536 (100%)	4,032 (16%)	15,904 (62%)	6,060 (24%)
Accuracy (relative)	99.72% (100%)	99.70% (99.98%)	99.50% (99.78%)	99.05% (99.33%)

Table 5. Computational complexity for A1a data set classification.

	A1a (total SVs: 563, *P*:76) set [9], threshold=99%			
	Original SVM	RSSVM [3]	VCSVM wo RS	VCSVM w RS
Req. reg. word (relative)	69,249 (100%)	28,782 (42%)	52,136 (75%)	27,132 (39%)
Req. mult. times (relative)	70,375 (100%)	29,908 (42%)	53,262 (76%)	28,258 (40%)
Accuracy (relative)	82.59% (100%)	82.51% (99.90%)	82.6% (100.0%)	82.57% (99.98%)

Table 6. Computational complexity for Leukemia data set classification.

	Leukemia (*N*:7,129, total SVs: 19, *P*: 5) set [11], threshold=60%			
	Original SVM	RSSVM [3]	VCSVM wo RS	VCSVM w RS
Req. reg. word (relative)	135,451 (100%)	106,935 (79%)	35,740 (26%)	35,655 (26%)
Req. mult. times (relative)	135,489 (100%)	106,973 (79%)	35,778 (26%)	35,693 (26%)
Accuracy (relative)	61.76% (100%)	61.67% (100%)	67.65% (109.5%)	67.65% (109.5%)

to the conventional RSSVM when the vector dimension N is large although it shows different performance when N is small. Since the severe complexity problem is mainly caused by the high input vector dimension seen in Tables 3 through 6, the proposed algorithm, which has good performance in the case of a large N, can be a good solution for real-time high-dimensional classification systems.

V. Conclusion

In this letter, we proposed a new vector compression algorithm to reduce the size of the support vector dimension. The proposed algorithm is very efficient when classifying high-dimensional input vectors which need much more complex computations. We expect that if we apply the proposed vector compression method to the training step, the complexity and the training time will be reduced significantly.

References

[1] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, 1995, pp. 273-297.

[2] D. DeCoste and D. Mazzoni, "Fast Query-Optimized Kernel Machine Classification via Incremental Approximate Nearest Support Vectors," *Proc. 20th ICML*, 2003, pp. 115-122.

[3] K.L. Wagstaff et al., "Progressive Refinement for Support Vector Machines," *Data Mining and Knowledge Discovery*, vol. 20, no. 1, 2010, pp. 53-69.

[4] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, 1998, pp. 121-167.

[5] D. Kim et al., "Use of Support Vector Regression in Stable Trajectory Generation for Walking Humanoid Robots," *ETRI J.*, vol. 31, no. 5, Oct. 2009, pp. 565-575.

[6] S. Romdhani et al., "Computationally Efficient Face Detection," *Proc. ICCV*, 2001, pp. 695-700.

[7] G. Strang, *Introduction to Linear Algebra*, 3rd ed., Wellesley-Cambridge Press, 1998.

[8] P. Viola and M.J. Jones, "Robust Real-Time Face Detection," *IJCV*, 2004, pp. 137-154.

[9] LIBSVM Data. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[10] LIBSVM Tools. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

[11] SPRToolbox. http://cmp.felk.cvut.cz/cmp/software/stprtool/