Kernel Map Compression using Generalized Radial Basis Functions

Omar Arif and Patricio Antonio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332
omararif@gatech.edu, pvela@ece.gatech.edu

Abstract

The use of Mercer kernel methods in statistical learning theory provides for strong learning capabilities, as seen in kernel principal component analysis and support vector machines. Unfortunately the computational complexity of the resulting method is of the order of the training set, which is quite large for many applications. This paper proposes a two step procedure for arriving at a compact and computationally efficient learning procedure. After learning, the second step takes advantage of the universal approximation capabilities of generalized radial basis function neural networks to efficiently approximate the empirical kernel maps. Sample applications demonstrate significant compression of the kernel representation with graceful performance loss.

1. Introduction

This paper presents a method for improving the computational and representational efficiency of algorithms based on Mercer kernel methods. Mercer kernel methods generate powerful techniques for studying non-linear data [16]. Many useful linear algorithms involving dot products can be made nonlinear by employing Mercer kernels. Two such algorithms are kernel principal component analysis (KPCA) [17] and support vector (SV) machines [18]. They have been applied in numerous image processing and computer vision applications, see for example [1, 10, 16, 22]. However, the superior performance of Mercer kernel-based algorithms comes at a price of increased storage and computational requirements [14].

Accurately learning the desired functional relationship in the training phase requires a large number of training samples. The large training set presents a difficulty, since it requires that the whole input space associated to the training set be stored and processed at once. Methods have been proposed for both the KPCA and the SV machines to learn the space incrementally [5, 8, 21]. However, once the space has been learned, either incrementally or in batch, all subsequent computations in the high-dimensional space are performed through the kernel in terms of linear combinations of all training vectors. For massive datasets this still requires high storage requirements and computational complexity, making kernel methods unfavorable for on line computer vision applications.

Schölkopf et al [15] discuss two solutions to reduce the execution space. The first, called *reduce set selection* (RSS), finds a reduced set of expansion vectors from the original space that approximates well the training set. RSS has been shown to work well (see refs. in [9, 15]). The second, called *reduce set construction* (RSC), identifies new elements of the input space that approximate well the training set. RSC has improved compression versus RSS [15], but has a more expensive upfront cost to find the reduced set. Related methods applicable only to SVMs include [7], which is a basis selection method, and [2], which extends [15] to incorporate shared support vectors.

A related form of learning is neural networks [6, 12]. Neural networks have the property of universal approximation [11, 13]. Further, they are more efficient than kernel methods [15]. Thus, in principal, neural networks should be capable of providing efficient representations of the functions or mappings arising from Mercer kernel methods, however previous attempts have not been so fruitful [15].

Contribution. This work exploits the relation between kernel methods, regularization theory and radial basis functions to propose a novel method to reduce the computational complexity associated with kernel methods. The work differs from previous efforts in that we approximate the learned function in a manner more compatible with the universal approximation capabilities of neural networks. In particular, once the function or mapping is learned using kernel methods, we identify the underlying algorithmic step involving the empirical kernel map, which is the part most suited to neural network approximation. The final algorithm is a two-step process beginning with the learning procedure, followed by the compression procedure. Efficient representations are achieved with minimal loss of performance.

2009 IEEE 12th International Conference on Computer Vision (ICCV)
978-1-4244-4419-9/09/$25.00 ©2009 IEEE

2. Mercer Kernels and Existing Efforts.

The basic idea behind Mercer kernels is to map the input space $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ into a high dimensional feature space $\mathcal{F}$ using a nonlinear mapping $\phi : \mathbb{R}^d \rightarrow \mathcal{F}$. The feature space $\mathcal{F}$ is a Hilbert space whose dot product is computed using Mercer kernels. A Mercer kernel is a continuous, symmetric and positive definite function $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, such that

$$\boldsymbol{k}(x_i, x_j) = (\phi(x_i) \cdot \phi(x_j)) . \tag{1}$$

Mercer kernels allow the computation of dot products in the feature space, without computing the mapping ϕ. Examples of the Mercer kernels are Gaussian kernel $\boldsymbol{k}(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$ and the polynomial kernel $\boldsymbol{k}(x_i, x_j) = (x_i \cdot x_j)^p$. Next, we summarize two popular kernel-based algorithms followed by existing attempts to reduce the space.

Kernel principal component analysis (KPCA) carries out PCA in the high-dimensional feature space by diagonalizing the kernel matrix K given by $K_{ij} = \boldsymbol{k}(x_i, x_j)$, for $i, j = 1, \ldots, n$. Let $\boldsymbol{\alpha}^k = [\frac{\alpha_1^k}{\sqrt{\lambda^k}}, \ldots, \frac{\alpha_n^k}{\sqrt{\lambda^k}}]^T$ be the k^{th} eigenvector of the kernel matrix K, with eigenvalue λ^k, then the principal components in the feature space are represented by $V_k = \sum_{i=1}^n \alpha_i^k \phi(x_i)$. A test point x is represented in the KPCA space by projecting it onto the eigenvectors. The projection of the k^{th} eigenvector is

$$f_k(x) = \sum_{i=1}^{n} \alpha_i^k \boldsymbol{k}(x_i, x). \tag{2}$$

Support Vector Machines (SV) perform classification by constructing a high-dimensional hyperplane that separates the data into two categories. The nonlinear decision boundary is given by

$$f(x) = sgn\left(\sum_{i=1}^{n} \alpha_i \boldsymbol{k}(x_i, x) + b\right). \tag{3}$$

The contributions with $\alpha_i \neq 0$ are called *support vectors* and are given by $V = \sum_{i=1}^n \alpha_i \phi(x_i)$. For each class to identify, there is a support vector, thus leading to a collection of support vectors $\{V_1, ..., V_{n_c}\}$ and coefficient vectors $\left\{\{\alpha_i^1\}, \ldots, \{\alpha_i^{n_c}\}\right\}$, where n_c is the number of classes.

Reduced Set Methods. To carry out either the projections (Equation (2)) or classification (Equation (3)), the computation depends on the entire input space $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ (or support space for SV), leading to the high storage and computational requirements. Schölkopf et al [15] propose two solutions to reduce the space. The first, called *reduce set selection* (RSS), deals with the problem of how to select a reduced set of expansion vectors from the original space so that the Equations (2) and (3) are represented in terms of the reduced expansion vectors. The second, called *reduce set construction* (RSC), constructs new vectors to achieve high reduction rates. RSC gives better compression results than RSS [15], but with a higher computation cost.

Both reduced set methods, RSC and RSS, obtain a reduced kernel mapping through the approximation of the vectors V_k by

$$\acute{V_k} = \sum_{j=1}^{l} \beta_j \phi(z_j), \tag{4}$$

where $l < n$ (or in the SV case $l < n_c$), $z_j \in \mathbb{R}^d$ are the new input vectors, and the $\beta_j \in \mathbb{R}$ are their coefficients [14]. The coefficients β_j and new input vectors z_j are found by minimizing

$$\left|\left|V_k - \acute{V_k}\right|\right|^2. \tag{5}$$

The procedure starts with $l = 1$ and, instead of minimizing Equation (5), maximizes the following equivalent function:

$$\frac{(\tilde{V_j} \cdot \phi(z_j))^2}{(\phi(z_j) \cdot \phi(z_j))}, \tag{6}$$

where $\tilde{V}_1 = V_k$, and the $\tilde{V}_j$ for $1 < j \leq l$ will be defined shortly.

To find the pre-image, z_1, fixed point or gradient based optimization procedures are used [4, 14]. Once the optimal pre-image z_1 is found, the coefficient β_1 is determined by setting $\beta_1 = \frac{(\tilde{V}_1 \cdot \phi(z_1))}{(\phi(z_1) \cdot \phi(z_1))}$. To find the next pre-image vector z_{j+1}, define $\tilde{V}_{j+1} := \tilde{V}_j - \beta_j \phi(z_j)$, where $\tilde{V}_j = \sum_{i=1}^n \alpha_i \phi(x_i) - \sum_{a=1}^{j-1} \beta_i \phi(c_i)$, and z_1 by z_{j+1}. In [4], once all the z_j and β_j are found, the procedure is followed by simultaneous optimization over all z_j and β_j. The procedure is performed to approximate all V_k, each with its own set of pre-image vectors $\{z_j^k\}$ and coefficients $\{\beta_j^k\}$, where the superscript of k specifies the connection to V_k.

Other methods that reduce the space include [20], where a maximum-likelihood approach is used to obtain sparse representation. However, the method is not general and is used for KPCA only.

3. Kernel Map Compression

The previous section described the reduced set method as an iterative procedure that identifies the input space pre-image vectors $\{z_j^k\}$ and associated coefficients $\{\beta_j^k\}$, $j = 1 \ldots l$, that approximates well the original Hilbert space vector(s), V_k. We propose a novel method to cull the input sample space by exploiting the relationship between (neural network) regularization theory and kernel methods [6, 19]. Rather than learn the eigenvectors in the Hilbert space (4) as in the reduced set methods, we propose to learn the empirical kernel maps, $V_k : \mathbb{R}^d \rightarrow \mathbb{R}$, using generalized radial basis functions (GRBFs).

In the finite-dimensional feature sub-space of $\mathcal{F}$, we have that the k-th coordinate of the image of x_i is

$$y_i^k = V_k(x_i) = \sum_{i=1}^{n} \alpha_i^k (x_i, x). \tag{7}$$

This input-output equation is found in both KPCA (2) and SV (3), and is determined during the KPCA and SV learning procedure. Further, the input-output relationship between x_i and y_i is much simpler than the x_i and V_k. In what follows, we describe how the collection of all $\{x_1, \ldots, x_n\}$ and associated $\left\{y_1^k, \ldots, y_n^k\right\}$ are learned for each k. In particular, we point out how the function in Equation (7) is of the class of functions that are efficiently approximated by generalized radial basis functions [13]. *Without loss of generality, and for purposes of exposition, we will ignore the superscript $\cdot^k$ in what follows, then introduce it again later.*

3.1. Setup.

In regularization theory [12], the approximation problem is: *given n different points $\{x_i \in \mathbb{R}^d, i = 1, \ldots, n\}$ and n real numbers, $\{y_i \in \mathbb{R}, i = 1, \ldots, n\}$, find a function $f : \mathbb{R}^d \to \mathbb{R}$ that minimizes*

$$H[f] = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \left|\left|Pf\right|\right|^2, \tag{8}$$

where P is a constraint operator (usually a differential operator), $||\cdot||$ is a norm on the function space, and λ is the regularization parameter. If $\left|\left|Pf\right|\right|^2$ is rotationally and translationally invariant, the regularized solution is given by the expansion in radial basis functions,

$$f(x) = \sum_{i=1}^{n} w_i r(|x_i - x|). \tag{9}$$

where $|\cdot|$ is a norm on the input space and r is the radial basis function (RBF). The RBF r is a real valued function depending only on the distance of the input from the origin. Commonly used types of RBFs include Gaussian, multiquadric, thin plate spline and polyharmonic spline. The function f is given by the sum of n RBFs, placed at points $\{x_i\}_{i=1}^n$ and weighted by coefficients $\{w_i\}_{i=1}^n$. The coefficients are found using the interpolation conditions $f(x_i) = y_i, i = 1, \ldots, n$. In many practical applications P is rotationally and translationally invariant.

In the RBF approach, the function f is expanded on a set of radial functions r centered at data points. The function is then a point in a multidimensional space, whose dimension is equal to the number of data points. This indicates that the function f can be approximated by an expansion in a basis with a smaller number of dimensions [12],

$$f^*(x) = \sum_{j=1}^{l} \gamma_j r(|c_j - x|), \tag{10}$$

where c_j and γ_j are $l < n$ center points and their coefficients. Now the problem consists of finding the optimal c_j and γ_j. Movable centers RBFs are called generalized radial basis functions (GRBFs).

3.2. Procedure

In what follows, we describe the procedure for generating the GRBF approximation to the input-output set $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$.

Finding optimal coefficients γ_j: Starting from an initial set of center locations c_j, the coefficients γ_j, can be found by imposing the constraints $f^*(x_i) = y_i$, leading to the following system of over constrained linear equations:

$$y_i = \sum_{j=1}^{l} \gamma_j r(|c_j - x_i|)\ , i = 1, \ldots, n. \tag{11}$$

The least square solution determines an approximate solution to the overdetermined system. The least squares formula is written as

$$\boldsymbol{\gamma} = (R^T R)^{-1} R^T \boldsymbol{y}, \tag{12}$$

where $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_l]^T$ is an l-dimensional coefficient vector, R is the $n \times l$ matrix with entries $R_{ij} = r(|x_i - c_j|)$, and $\boldsymbol{y} = [y_1, \ldots, y_n]^T$. The $l \times l$ matrix $R^T R$ is nonsingular [12] and therefore invertible.

Finding optimal center location c_j: To find the optimal center locations c_j, minimize the Equation (8), with f replaced by f^*. Different optimization algorithms can be used for this purpose. We show here calculations using gradient descent and setting $\lambda = 0$. We start with the initial estimation of l center points and use the following equation to find the update:

$$c_j(t+1) = c_j(t) + \delta t \frac{\partial H[f^*]}{\partial c_j}\ j = 1, \ldots, l, \tag{13}$$

where δt is the time step and $\frac{\partial H[f^*]}{\partial c_j}$ is given by

$$\frac{\partial H[f^*]}{\partial c_j} = -4\gamma_j \sum_{i=1}^{n} (y_i - \hat{y}_i) r(|c_j - x_i|)(c_j - x_i), \tag{14}$$

where $\hat{y}_i = f^*(x_i)$.

Since the method described above is based on gradient descent, the starting locations of c_j need to be estimated. We can start from equally spaced center locations spread over the original space, or use the center locations found as a result of RSS [15] as the starting center locations. Here, k-means applied to the original space to find l cluster locations will identify the starting locations.

Algorithm: In summary, the following steps are taken to find the reduced space:

1: use k-means to find the l initial centers c_j.
2: use Equation (12) to find the coefficients γ_j.
3: iterate Eqs. (12) and (13) until the error given by Eq. (8) goes below some threshold value.

Applied to Equation (7). The procedure applies to each of the functions V_k and their associated input-output sets $\{x_1, \ldots, x_n\}$ and $\{y_1^k, \ldots, y_n^k\}$. Note that all center locations and coefficients are optimized simultaneously. This contrasts with Schölkopf [14], where optimization is performed over one point at a time. In Burges method [4], optimization is performed over one point at a time, followed by a second phase with all parameters optimized jointly.

3.3. Approximating Several Functions at Once

Typical applications require approximating more than one function in the feature space. For example, we would like to efficiently compute all the functions $V_k(x)$, corresponding to $f_k(x)$ in Equation (2), and the collection of binary classifiers $V_k(x)$ arising from Equation (3). Approximating each function separately using the method described in Section 3 may not result in adequate compression. The procedure will return a pair of center locations and their coefficients for each function V_k, $\{c_j^k, \gamma_j^k\}_{j=1}^l$, where $k = 1, \ldots, m$ and m is the total number of functions to be approximated (varies based on KPCA vs SV). A more efficient approach is to share center locations c_j for all the functions V_k with different coefficients γ_j^k. The functions V_k are then approximated by

$$V_k^*(x) = \sum_{j=1}^{l} \gamma_j^k r(|c_j - x|). \tag{15}$$

The combined optimization gives a different update method for the coefficients and center locations. To find the coefficients γ_j^k, Equation (12) is used except that the $\boldsymbol{\gamma}$ and $\boldsymbol{y}$ are $l \times m$ matrices, where l is the total number of centers and m is the number of functions to be approximated.

To find the optimal center locations c_j, the functional to be minimized is of the form

$$H[f^*] = \sum_{k=1}^{m} \sum_{i=1}^{n} \left(y_i - f_k^*(x_i)\right)^2. \tag{16}$$

3.4. Achieving Further Compression

The method achieves compression by only optimizing over the location and the weights of the basis functions, with all other parameters of the basis functions equal. For Gaussian RBFs, additional modifiable parameters include the positive symmetric operator defining the norm $|\cdot|$ for each RBF, which affects the orientation and anisotropy associated to the RBF. Optimizing over these parameters may lead to further reductions in the number of RBFs needed.

3.5. Computational Cost

The proposed method iterates between finding the updated coefficients γ_i and the centers c_i, until the error given by Equation (8) goes below some threshold. The equations to iterate are Equations (12) and (13). Equation (12) requires taking the inverse of one $l \times l$ matrix, with the computational cost of $\mathcal{O}(l^3)$, where l is the number of reduced center points c_i. The computation cost updating the center positions for each c_i (Equation 13) is $\mathcal{O}(l \times n)$. This is the same cost as the RSC methods, since Equation (4) and the discussion following Equation (6) indicate that the principal computations of the RSC method are also of the order of $\mathcal{O}(l \times n)$ and $\mathcal{O}(l^3)$ for optimizing the center locations and coefficients. However when RSC method is followed by phase 2, in which the optimization is performed over all parameters, the computational complexity of RSC method increases to about two orders of magnitude [15].

4. Application

4.1. Synthetic Dataset

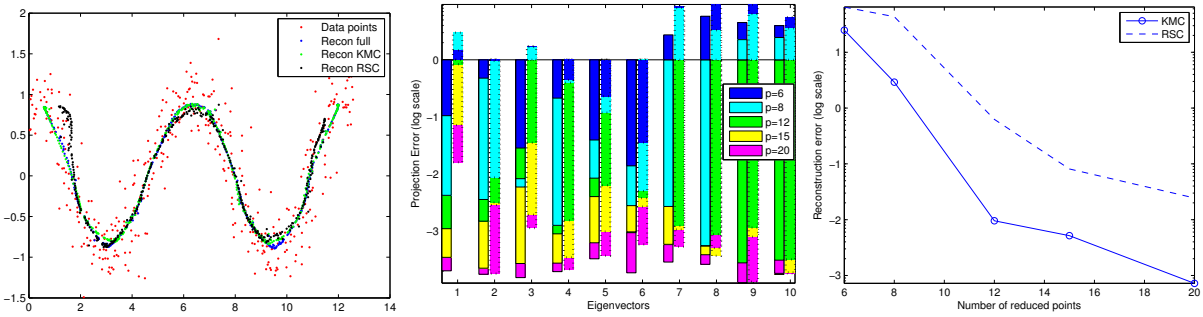
We first apply the kernel map compression to a simple 2D example. The first three eigenvectors obtained as a result of applying KPCA to a synthetic dataset of 400 points are approximated using $[6, 8, 12, 15, 20]$ points. For KPCA the Gaussian kernel is used with $\sigma = 1$. The same kernel is used for the kernel map compression (KMC) method proposed in this paper. The results are shown in Figure 1. In Figure 1(a), 400 data points shown in red, while the reconstruction with 10 eigenvectors using full space and using only 8 points is shown in blue, green (KMC) and black (RSC). Figure 1(b) shows the errors in projecting the space onto each of the 10 eigenvectors. Similarly, Figure 1(c) shows the error in reconstructing the data set using the reduced space. The figures show that the KMC method is better able to approximate the eigenvectors.

4.2. Speeding up SV Decision Rules

We applied the algorithm to the USPS database of handwritten digits [15]. The same database was used to gauge the performance of RSC methods in [4, 15]. The data consists of 9298 handwritten digits of dimensions 16×16, which was randomly shuffled and divided into training and test sets consisting of 4649 cases each. Ten binary classifiers were trained, one for each digit.

For classification, Gaussian kernel with $\sigma = 10$, was used. The classifiers were approximated using the proposed method (KMC) and RSC method [4]. The RSC method utilized the Matlab code publicly available for download at www.kyb.tuebingen.mpg.de/bs/people/spider/. The proposed method (KMC) was also implemented using Matlab.

The results for USPS hand written database are shown in



(a) Data set and reconstructed data set using 10 eigenvectors, each one approximated using 8 points.

(b) Projection errors for 10 eigenvectors, each one approximated using p points. Solid line: KMC, dashed line: RSC

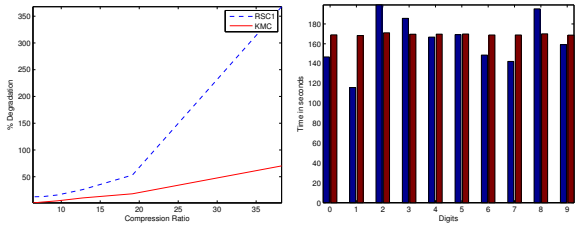
(c) Reconstruction error vs the number of points the space is reduced. Solid line: KMC, dashed line: RSC

Figure 1. Performance comparison for the synthetic 2D example.

Table 1. USPS handwritten digit. Top: number of SVs for the original SV and the number of test errors for each classifier. Bottom: number of test errors for each reduced classifier. KMC-p and RSC-p mean that, for each classifier, the space was reduced to p points. Third last column shows error rate across all classifiers and the second last column shows the compression ratio which is the ratio of the full space to the reduced space. The last column shows the degradation in performance.

digit	0	1	2	3	4	5	6	7	8	9	Error	C.ratio	Degradation
#SV	160	80	262	220	243	252	145	144	247	160	ave	ave	ave
SV	29	9	37	36	38	36	25	19	29	25	6.08%	1	-
RSC-5	33	17	148	162	156	123	58	47	273	305	28.44%	38.26	367.76 %
KMC-5	29	11	65	71	77	76	33	25	54	40	10.35%	38.26	70.23%
RSC-10	29	8	48	54	74	66	33	31	38	52	9.31%	19.13	53.13%
KMC-10	29	8	41	40	52	44	32	20	43	25	7.18%	19.13	18.09%
RSC-15	27	9	46	42	49	57	29	27	38	32	7.66%	12.75	25.99 %
KMC-15	31	9	41	41	43	42	24	19	37	25	6.71%	12.75	10.36 %
RSC-20	29	9	46	40	47	46	27	25	31	28	7.05%	9.56	15.95%
KMC-20	30	10	36	38	37	39	27	21	35	24	6.38%	9.56	4.99%
RSC-25	31	9	40	38	46	46	28	22	31	28	6.86%	7.65	12.83%
KMC-25	32	8	39	38	41	36	26	19	31	28	6.25%	7.65	2.80%
RSC-30	32	8	43	38	46	41	28	22	33	27	6.84%	6.37	12.50%
KMC-30	29	9	41	38	37	39	26	21	30	27	6.15%	6.37	1.15%

Table 1. The first row shows the letter being classified, with the second row giving the original number of, and the third row indicating the number of misclassified digits. KMC-p and RSC-p indicate that each binary classifier was reduced to p points using the corresponding method. As can be seen from the table, KMC approximates well the original SVM with graceful degradation for high compression levels (at compression ratio of 9.56, KMC-20 degrades less than 5%). This is seen more easily in Figure 2(a), where the degradation versus compression curve for the proposed method is lower than the RSC method. The runtime for compressing the classifiers for the case p = 20 is shown in Figure 2(b).



(a) % degradation vs. compression curves.

(b) Runtimes for each classifier in the case of p = 20.

Figure 2. Performance comparison for the USPS data set.

4.3. Efficient Sign Language Recognition w/KPCA

To test KPCA compression, we used the sign language database [3] for sign language recognition. The database consisted of 2040 images of a hand performing the different static signs used in the international sign language alphabet. The images were cropped and down-sampled to dimensions 20×15. The images were randomly divided into training and testing images, with the training set containing about $2/3$ of the total images. KPCA was performed on the train-

Table 2. Sign language recognition: [1] success rate of identifying all the 2040 images. [2] success rate for test cases. RSC-p and KMC-p mean that for each eigenvector the space was reduced to p points using the corresponding method. The last column shows the compression ratio.

Algorithm	S. Rate[1]	S. Rate[2]	C.Ratio	Deg
KPCA	99.12%	98.55%	1	-
RSC-2	76.62%	74.14%	155	23.73%
KMC-2	98.92%	97.93%	155	.41%
RSC-5	91.37%	90.73%	61	7.88%
KMC-5	99.5%	97.98%	61	0%
RSC-8	98.63%	97.93%	40	.56%
KMC-8	99.25%	98.23%	40	0%

ing images. For classification, the training and test images were projected on the first 5 eigenvectors and $k-$nearest neighbour rule with $k = 5$ was evaluated. The space was reduced using KMC and the results are shown in Table 2. At a compression ratio of 61, the performance is equal to using the full space. At the same compression ratio, the degradation for the case of RSC method is more than 7%.

5. Conclusion

This paper proposed a technique for achieving computational reductions in Mercer kernel methods, such as kernel principle component analysis (KPCA) and support vector (SV) machines. The technique takes advantage of the universal approximation characteristics of generalized radial basis function neural networks to approximate the empirical kernel map associated to KPCA or SV machines. Computational savings of an order of magnitude or more were achieved with minimal to no loss of performance.

References

[1] S. Avidan. Support vector tracking. In *IEEE CVPR*, volume 1, pages 184–191, 2001.

[2] T. Benyang and D. Mazzoni. Multiclass reduced-set support vector machines. In *Proc. of Int. Conf. on Machine Learning*, pages 921–928, 2006.

[3] H. Birk, T. B. Moeslund, and C. B. Madsen. Real-time recognition of hand alphabet gestures using principal component analysis. In *Conference on Image Analysis*, pages 261–268, 1997.

[4] C. J. C. Burges. Simplified support vector decision rules. In *Proc. of Int. Conf. on Machine Learning*, pages 71–77. Morgan Kaufmann, 1996.

[5] T.-J. Chin and D. Suter. Incremental kernel principal component analysis. *IEEE Transactions on Image Processing*, 16:1662–1674, 2007.

[6] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In *Advances in Computational Mathematics*, pages 1–50. MIT Press, 2000.

[7] S. S. Keerthi, O. Chapelle, and D. Decoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7:1493 – 1515, 2006.

[8] P. Laskov, C. Gehl, , S. Krüger, and K.-R. Müller. Incremental support vector learning: Analysis, implementation and applications. *Journal of Machince Learning Research*, 7:1909–1936, 2006.

[9] Q. Li, L. Jiao, and Y. Hao. Adaptive simplification of solution for support vector machine. *Pattern Recognition*, 40(3):972–980, 2007.

[10] J. Meltzer, S. Soatto, M.-H. Yang, and R. Gupta. Multiple view feature descriptors from image sequences via kernel principal component analysis. In *ECCV*, pages 215–227, 2004.

[11] J. Park and I. Sandberg. Approximation and radial-basis-function networks. *Neural Computation*, 5(2):305–316, 1993.

[12] T. Poggio and F. Girosi. A theory of networks for approximation and learning. Technical Report 1140, Massachusetts Institute of Technology, 1989.

[13] I. Sandberg. Gaussian radial-basis functions and inner-product spaces. In *International Conference on Artificial Neural Networks*, pages 177–182, 2001.

[14] B. Schölkopf, P. Knirsch, A. Smola, and C. Burges. Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces. In *Annual Symposium of the German Association for Pattern Recognition (DAGM)*, 1998.

[15] B. Scholkopf, S. Mika, C. Burges, P. Knirsch, K. Muller, G. Ratsch, and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, pages 1000–1017, 1999.

[16] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001.

[17] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. 10:1299–1319, 1998.

[18] J. Shawe-Taylor and N. Cristianini. *Support Vector Machines*. Cambridge University Press, 2000.

[19] A. J. Smola, R. Chaussee, and B. Schölkopf. From regularization operators to support vector kernels. In *In Advances in Neural information processings systems*, pages 343–349. MIT Press, 1998.

[20] M. Tipping. Sparse kernel principal component analysis. In *Neural Information Processing Systems*, pages 633–639, 2000.

[21] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688. MIT Press, 2001.

[22] L.-H. Zhao, X.-L. Zhang, and X.-H. Xu. Face recognition based on kpca with polynomial kernels. In *International Conference on Wavelet Analysis and Pattern Recognition*, pages 1213–1216, 2007.