Supplemental Material for “Classifying Video with Kernel Dynamic Textures”

Antoni B. Chan and Nuno Vasconcelos

SVCL-TR 2007/03

April 2007

Supplemental Material for “Classifying Video with Kernel Dynamic Textures”

Antoni B. Chan and Nuno Vasconcelos

Statistical Visual Computing Lab
Department of Electrical and Computer Engineering
University of California, San Diego

April 2007

Abstract

This is the supplemental material for “Classifying Video with Dynamic Textures” [1]. It contains information about the attached videos, a brief review of kernel PCA, the “centered” versions of the algorithms discussed in the paper, and the derivation of the inner-product between the feature-transformations of two Gaussian kernels.

Author email: abchan@ucsd.edu

1 Introduction

This is the supplemental material for “Classifying Video with Dynamic Textures” [1]. The supplemental is organized as follows. Section 2 contains information about the attached video. Section 3 briefly reviews kernel PCA, and Section 4 presents the “centered” versions of the algorithms discussed in [1]. Finally, the derivation of the inner-product between the feature-transformations of two Gaussian kernels appears in Section 5.

2 Example Video

The following videos are attached to this supplemental, and are described here. All video is in Quicktime (h.264) format and should be viewable with the most recent version of the Quicktime player (http://www.quicktime.com).

- `montage_ucla.mov` – examples from the UCLA video texture database.
- `montage_ucla_pan.mov` – examples from the UCLA-pan database (see Figure 3), containing video textures with panning camera motion.
- `misclass_water.mov` – a montage of chaotic water video clips (see Figure 5), which are poorly modeled by the dynamic texture.

3 Kernel PCA

Kernel PCA [2] is the kernelized version of standard PCA [3]. With standard PCA, the data is projected onto the linear subspace (linear principal components) that best captures the variability of the data. In contrast, kernel PCA (KPCA) projects the data onto non-linear functions in the input-space. These non-linear principal components are defined by the kernel function, but are never explicitly computed. An alternative interpretation is that kernel PCA first applies a non-linear feature transformation to the data, and then performs standard PCA in the feature-space.

3.1 Learning KPCA coefficients

Given a training data set of N points $Y = [y_1, \ldots, y_N]$ with $y_i \in \mathbb{R}^m$ and a kernel function $k(y_1, y_2)$ with associated feature transformation $\phi(y)$, i.e. $k(y_1, y_2) = \langle \phi(y_1), \phi(y_2) \rangle$, the kernel principal components are the eigenvectors of the covariance matrix of the transformed data $\phi(y_i)$. Assuming that the transformed data is centered (i.e. has zero mean in the feature space), the c-th principal component in the feature-space has the form [2]:

$$v_c = \sum_{i=1}^{N} \alpha_{i,c} \phi(y_i) \tag{1}$$

The KPCA weight vector $\alpha_c = [\alpha_{1,c}, \ldots, \alpha_{N,c}]^T$ is computed as

$$\alpha_c = \frac{1}{\sqrt{\lambda_c}} v_c \tag{2}$$

where λ_c and v_c are the c-th largest eigenvalue and corresponding eigenvector of the kernel matrix K, which has entries $[K]_{i,j} = k(y_i, y_j)$. The scaling of v_c ensures that the principal component in the feature-space is unit length.

Given the training data point y_j, the c-th KPCA coefficient $x_{c,j}$ is computed as the projection of $\phi(y_j)$ onto the principal component v_c, i.e.

$$x_{c,j} = \langle \phi(y_j), v_c \rangle = \sum_{i=1}^{N} \alpha_{i,c} k(y_i, y_j) \tag{3}$$

and hence, the KPCA coefficients $X = [x_1, \cdots, x_N]$ of the training set Y can be computed as $X = \alpha^T K$, where $\alpha = [\alpha_1, \cdots, \alpha_n]$ is the KPCA weight matrix, and n is the number of principal components.

3.2 Reconstruction from KPCA coefficients

KPCA directly models the mapping from the input-space to the KPCA coefficient-space. However since the principal components are never actually computed explicitly (only the projections on to them), the reverse mapping from the coefficient-space to the input-space is not as straightforward to compute. Given the KPCA coefficients, $x_t = [x_{1,t}, \ldots, x_{n,t}]^T$, the KPCA reconstruction problem is to find the pre-image y_t in the input-space that generated these coefficients. In general, this is an ill-posed problem since no y_t could exist for some KPCA coefficients [4].

The minimum-norm reconstruction method [4, 5] aims to find the pre-image y_t that minimizes the norm of the error in the feature-space,

$$y_t^* = \operatorname*{argmin}_{y_t} \left\| \phi(y_t) - \sum_{c=1}^{n} x_{c,t} v_c \right\|^2 \tag{4}$$

$$= \operatorname*{argmin}_{y_t} k(y_t, y_t) - 2 \sum_{i=1}^{N} \gamma_i k(y_t, y_i) \tag{5}$$

where $\gamma_i = \sum_{c=1}^{n} x_{c,t} \alpha_{i,c}$. When the kernel function is the Gaussian kernel, $k_g(y_1, y_2) = \exp(-\frac{1}{2\sigma^2} \|y_1 - y_2\|^2)$, a solution can be found using an iterative fixed-point procedure [4]. Given some initial guess $y_t^{(0)}$, refinements are computed using

$$y_t^{(j+1)} = \frac{\sum_{i=1}^{N} \gamma_i k_g(y_t^{(j)}, y_i) y_i}{\sum_{i=1}^{N} \gamma_i k_g(y_t^{(j)}, y_i)} \tag{6}$$

where $y_t^{(j)}$ is the estimate of y_t^* at iteration j. In practice, the initial guess $y_t^{(0)}$ can be initialized using nearest neighbors in the coefficient space, i.e. choosing $y_t^{(0)} = y_{i^*}$

such that $i^* = \text{argmin}_i \|x_t - x_i\|^2$. Minimum-norm reconstruction has been shown to be useful in image de-noising applications [4].

Alternatively, reconstruction can also be achieved using other methods, e.g. those based on distance constraints [6], or by explicitly modeling the function between the KPCA coefficients and the input-space, e.g. using kernelized ridge regression [7].

4 Kernel Centering

In this section, we derive the "centered kernel" versions of KPCA, minimum-norm reconstruction, and the inner-product between KPCA components.

4.1 Centering for KPCA

So far we have assumed that the transformed data is centered in the feature-space. In the general case, the transformed data must be centered explicitly by subtracting the empirical mean, resulting in the centered feature transformation

$$\tilde{\phi}(y) = \phi(y) - \frac{1}{N}\sum_{n=1}^{N}\phi(y_n) \tag{7}$$

where y_j are the training data. The centered kernel matrix between two points y and y' is then

$$\begin{aligned}
\tilde{k}(y,y') &= \left\langle \tilde{\phi}(y), \tilde{\phi}(y') \right\rangle && (8)\\
&= \left\langle \phi(y) - \frac{1}{N}\sum_{n=1}^{N}\phi(y_n), \phi(y') - \frac{1}{N}\sum_{n=1}^{N}\phi(y_n) \right\rangle && (9)\\
&= k(y,y') - \frac{1}{N}\sum_{n=1}^{N}k(y',y_n) - \frac{1}{N}\sum_{n=1}^{N}k(y_n,y) && (10)\\
&\quad + \frac{1}{N^2}\sum_{n,m}k(y_n,y_m)
\end{aligned}$$

Hence, for the training kernel, the centering is obtained from the non-centered kernel as [7]

$$\begin{aligned}
\tilde{K} &= K - \frac{1}{N}\mathbf{e}\mathbf{e}^T K - \frac{1}{N}K\mathbf{e}\mathbf{e}^T + \frac{1}{N^2}\mathbf{e}\mathbf{e}^T K\mathbf{e}\mathbf{e}^T && (11)\\
&= (I - \frac{1}{N}\mathbf{e}\mathbf{e}^T)K(I - \frac{1}{N}\mathbf{e}\mathbf{e}^T) && (12)
\end{aligned}$$

where $\mathbf{e}$ is the vector of N ones. Given a test point y_t, the centered kernel between the test point and the training points is obtained from the non-centered kernel $K_t = [k(y_t,y_1),\cdots,k(y_t,y_N)]$ as

$$\tilde{K_t} = K_t - \frac{1}{N}\mathbf{e}^T K - \frac{1}{N}K_t\mathbf{e}\mathbf{e}^T + \frac{1}{N^2}\mathbf{e}^T K\mathbf{e}\mathbf{e}^T \tag{13}$$

$$= K_t(I - \frac{1}{N}\mathbf{e}\mathbf{e}^T) - \frac{1}{N}\mathbf{e}^T K(I - \frac{1}{N}\mathbf{e}\mathbf{e}^T) \quad (14)$$

$$= (K_t - \frac{1}{N}\mathbf{e}^T K)(I - \frac{1}{N}\mathbf{e}\mathbf{e}^T) \quad (15)$$

4.2 Centering for minimum-norm reconstruction

Minimum-norm reconstruction using the centered kernel is given by

$$\begin{aligned}
y_t^* &= \operatorname*{argmin}_{y_t} \left\| \tilde{\phi}(y_t) - \sum_c x_{c,t} v_c \right\|^2 & (16)\\
&= \operatorname*{argmin}_{y_t} \tilde{k}(y_t, y_t) - 2\sum_c x_{c,t} \left\langle v_c, \tilde{\phi}(y_t) \right\rangle + \sum_{c,c'} x_{c,t} x_{c',t} \left\langle v_c, v_{c'} \right\rangle & (17)\\
&= \operatorname*{argmin}_{y_t} \tilde{k}(y_t, y_t) - 2\sum_c x_{c,t} \sum_n \alpha_{n,c} \tilde{k}(y_t, y_n) + \sum_c (x_{c,t})^2 & (18)\\
&= \operatorname*{argmin}_{y_t} \tilde{k}(y_t, y_t) - 2\sum_n \gamma_n \tilde{k}(y_t, y_n) & (19)\\
&= \operatorname*{argmin}_{y_t} \left[k(y_t, y_t) - \frac{2}{N}\sum_n k(y_t, y_n) + \frac{1}{N^2}\sum_{n,m} k(y_n, y_m) \right] & (20)\\
&\quad - 2\sum_n \gamma_n \left[k(y_t, y_n) - \frac{1}{N}\sum_j k(y_j, y_n) - \frac{1}{N}\sum_j k(y_j, y_t) \right. \\
&\qquad \left. + \frac{1}{N^2}\sum_{i,j} k(y_i, y_j) \right] \\
&= \operatorname*{argmin}_{y_t} k(y_t, y_t) - \frac{2}{N}\sum_n k(y_t, y_n) - 2\sum_n \gamma_n k(y_t, y_n) & (21)\\
&\quad + 2\sum_n \gamma_n \frac{1}{N}\sum_j k(y_t, y_j) \\
&= \operatorname*{argmin}_{y_t} k(y_t, y_t) - 2\sum_n \left(\gamma_n - \frac{1}{N}\sum_i \gamma_i + \frac{1}{N} \right) k(y_t, y_n) & (22)
\end{aligned}$$

Hence, the reconstruction problem using the centered kernel reduces to the standard reconstruction problem with modified weights,

$$\tilde{\gamma}_n = \gamma_n - \frac{1}{N}\sum_i \gamma_i + \frac{1}{N} \quad (23)$$

4.3 Centering for the inner-product between KPCA components

Consider two data sets $\{y_i^a\}_{i=1}^{N_a}$ and $\{y_i^b\}_{i=1}^{N_b}$, and two centered kernel functions $\tilde{k}_a$ and $\tilde{k}_b$ with centered feature-transformations $\tilde{\phi}(y)$ and $\tilde{\psi}(y)$, i.e.

$$\tilde{k}_a(y_1, y_2) = \left\langle \tilde{\phi}(y_1), \tilde{\phi}(y_2) \right\rangle \tag{24}$$

$$\tilde{k}_b(y_1, y_2) = \left\langle \tilde{\psi}(y_1), \tilde{\psi}(y_2) \right\rangle \tag{25}$$

which share the same inner-product and feature-spaces. Running KPCA on each of the data-sets with their centered kernels yields the KPCA weight matrices α and β, respectively. The c-th and d-th KPCA components in each of the feature-spaces are given by,

$$u_c = \sum_i \alpha_{i,c} \tilde{\phi}(y_i^a) \tag{26}$$

$$u_d = \sum_i \beta_{i,d} \tilde{\psi}(y_i^b) \tag{27}$$

Hence, their inner product is given by

$$\begin{aligned}
\langle u_c, u_d \rangle &= \left\langle \sum_i \alpha_{i,c} \tilde{\phi}(y_i^a), \sum_j \beta_{j,d} \tilde{\psi}(y_j^b) \right\rangle && (28)\\
&= \sum_{i,j} \alpha_{i,c} \beta_{j,d} \left\langle \tilde{\phi}(y_i^a), \tilde{\psi}(y_j^b) \right\rangle && (29)\\
&= \sum_{i,j} \alpha_{i,c} \beta_{j,d} \left\langle \phi(y_i^a) - \frac{1}{N_a} \sum_k \phi(y_k^a), \psi(y_i^b) - \frac{1}{N_b} \sum_k \psi(y_k^b) \right\rangle && (30)\\
&= \sum_{i,j} \alpha_{i,c} \beta_{j,d} \Bigg(g(y_i^a, y_j^b) - \frac{1}{N_a} \sum_k g(y_k^a, y_j^b) && (31)\\
&\qquad - \frac{1}{N_b} \sum_k g(y_i^a, y_k^b) + \frac{1}{N_a N_b} \sum_{k,k'} g(y_k^a, y_{k'}^b) \Bigg) \\
&= \alpha_c^T G \beta_d - \frac{(\mathbf{e}^T \alpha_c)}{N_a} \mathbf{e}^T G \beta_d - \frac{(\mathbf{e}^T \beta_d)}{N_b} \alpha_c^T G \mathbf{e} && (32)\\
&\quad + \frac{(\mathbf{e}^T \alpha_c)(\mathbf{e}^T \beta_d)}{N_a N_b} \mathbf{e}^T G \mathbf{e} \\
&= \left(\alpha_c - \frac{(\mathbf{e}^T \alpha_c)}{N_a} \mathbf{e} \right)^T G \left(\beta_d - \frac{(\mathbf{e}^T \beta_d)}{N_b} \mathbf{e} \right) && (33)
\end{aligned}$$

where $g(y_1, y_2) = \langle \phi(y_1), \psi(y_2) \rangle$, and G is the matrix with entries $[G]_{i,j} = g(y_i^a, y_j^b)$. Hence, the computation for the centered kernel is equivalent to using the non-centered kernel, but with α_c and β_d normalized by subtracting their respective means.

$$\tilde{\alpha}_c = \alpha_c - \frac{(\mathbf{e}^T \alpha_c)}{N_a} \mathbf{e} \tag{34}$$

$$\tilde{\beta}_d = \beta_d - \frac{(\mathbf{e}^T\beta_d)}{N_b}\mathbf{e} \qquad (35)$$

4.4 Summary

A summary of the modifications for centering the kernel for the various algorithms is given in the following table:

Algorithm	Modification for centering
KPCA training kernel	$\tilde{K} = (I - \frac{1}{N}\mathbf{e}\mathbf{e}^T)K(I - \frac{1}{N}\mathbf{e}\mathbf{e}^T)$
KPCA testing kernel	$\tilde{K}_t = (K_t - \frac{1}{N}\mathbf{e}^T K)(I - \frac{1}{N}\mathbf{e}\mathbf{e}^T)$
KPCA min-norm reconstruction	$\tilde{\gamma}_i = \gamma_i - \frac{1}{N}\sum_{j=1}\gamma_j + \frac{1}{N}$
inner-product btwn. KPCA comp.	$\tilde{\alpha}_c = \alpha_c - \frac{(\mathbf{e}^T\alpha_c)}{N_a}\mathbf{e}$, $\tilde{\beta}_d = \beta_d - \frac{(\mathbf{e}^T\beta_d)}{N_b}\mathbf{e}$

5 Inner-product between Gaussian feature-spaces

In this section, we derive the inner-product between the feature-transformations of two Gaussian kernels with different bandwidth parameters. Let $\Phi(y)$ be the feature transformation induced by the Gaussian kernel with unit variance, i.e.

$$k(y_1, y_2) = e^{-\frac{1}{2}\|y_1 - y_2\|^2} = \langle \Phi(y_1), \Phi(y_2) \rangle \qquad (36)$$

Now consider the Gaussian kernel parameterized by σ^2,

$$k_\sigma(y_1, y_2) = e^{-\frac{1}{2\sigma^2}\|y_1 - y_2\|^2} \qquad (37)$$

$$= e^{-\frac{1}{2}\left\|\frac{1}{\sigma}y_1 - \frac{1}{\sigma}y_2\right\|^2} \qquad (38)$$

$$= \left\langle \Phi\left(\frac{1}{\sigma}y_1\right), \Phi\left(\frac{1}{\sigma}y_2\right) \right\rangle \qquad (39)$$

Hence, the feature transformation of the Gaussian with variance σ^2 is related to the Gaussian kernel with unit variance via $\Phi_\sigma(y) = \Phi(\frac{1}{\sigma}y)$. Finally, for two Gaussian kernels parameterized by σ_a^2 and σ_b^2, the inner product between their feature-transformations is

$$g(y_1, y_2) = \langle \Phi_a(y_1), \Phi_b(y_2) \rangle \qquad (40)$$

$$= \left\langle \Phi\left(\frac{1}{\sigma_a}y_1\right), \Phi\left(\frac{1}{\sigma_b}y_2\right) \right\rangle \qquad (41)$$

$$= e^{-\frac{1}{2}\left\|\frac{1}{\sigma_a}y_1 - \frac{1}{\sigma_b}y_2\right\|^2} \qquad (42)$$

References

[1] A. B. Chan and N. Vasconcelos, "Classifying video with kernel dynamic textures," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[2] B. Schölkopf, A. Smola, and K. R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[3] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. John Wiley and Sons, 2001.

[4] S. Mika, B. Schölkopf, A. Smola, K. R. Müller, M. Scholz, and G. Rätsch, “Kernel PCA and de-noising in feature spaces,” in *Neural Information Processing Systems*, vol. 11, 1999, pp. 536–52.

[5] B. Schölkopf, S. Mika, A. Smola, G. Rätsch, and K. R. Müller, “Kernel pca pattern reconstruction via approximate pre-images,” in *ICANN, Perspectives in Neural Computing*, 1998, pp. 147–52.

[6] J.-Y. Kwok and I.-H. Tsang, “The pre-image problem in kernel methods,” *IEEE Trans. Neural Networks*, vol. 15, no. 6, 2004.

[7] T. De Bie, N. Cristianini, and R. Rosipal, *Handbook of Computational Geometry for Pattern Recognition, Computer Vision, Neurocomputing and Robotics*. Springer-Verlag, 2004, ch. Eigenproblems in Pattern Recognition.

SVCL-TR 2007/03

April 2007

Supplemental Material for "Classifying Video with Kernel Dynamic Textures"

Antoni B. Chan