Greedy approximation of kernel PCA by minimizing the mapping error

Peng Cheng  
*University of Wollongong, pc175@uow.edu.au*

Wanqing Li  
*University of Wollongong, wanqing@uow.edu.au*

Philip Ogunbona  
*University of Wollongong, philipo@uow.edu.au*

Publication Details

Abstract
In this paper we propose a new kernel PCA (KPCA) speed-up algorithm that aims to find a reduced KPCA to approximate the kernel mapping. The algorithm works by greedily choosing a subset of the training samples that minimizes the mean square error of the kernel mapping between the original KPCA and the reduced KPCA. Experimental results have shown that the proposed algorithm is more efficient in computation and effective with lower mapping errors than previous algorithms.

Keywords
minimizing, pca, kernel, error, approximation, mapping, greedy

Disciplines
Physical Sciences and Mathematics

Publication Details

This conference paper is available at Research Online: http://ro.uow.edu.au/infopapers/2130
Greedy approximation of kernel PCA by minimizing the mapping error

Peng Cheng #1, Wanqing Li #2, Philip Ogunbona #3
# School of Computer Science and Software Engineering
University of Wollongong
Wollongong, Australia
1pc175@uow.edu 2wanqing@uow.edu 3philipo@uow.edu

Abstract—In this paper we propose a new kernel PCA (KPCA) speed-up algorithm that aims to find a reduced KPCA to approximate the kernel mapping. The algorithm works by greedily choosing a subset of the training samples that minimizes the mean square error of the kernel mapping between the original KPCA and the reduced KPCA. Experimental results have shown that the proposed algorithm is more efficient in computation and effective with lower mapping errors than previous algorithms.

Keywords—Kernel PCA, Greedy approximation

I. INTRODUCTION

Kernel methods are a class of widely used machine learning techniques that extend the ability of linear learning methods (e.g. regression [1], support vector machine (SVM) [2], principal component analysis (PCA) [3] and linear discriminative analysis (LDA) [4]) so as to deal with non-linear patterns. Recently the kernel methods are gaining increasing popularity in many applications due to their simple and flexible nature. For example, kernel PCA was employed in computer vision as a manifold learning tool and a one-class classifier [5] [6] [7]. A major problem in applying kernel methods is that the computational cost is proportional to the number of training samples that define the kernel models and is often not affordable in practice when the number is large. A common solution to the problem is the reduced set approximation [8] by replacing the original kernel model with a similar one defined by a reduced number of examples with an acceptable or little loss of performance. This paper is concerned with the approximation of kernel PCA model. The proposed algorithm can also be extended to other kernel methods [9] [10].

In general, the approximation problem can be regarded as an optimization problem, in which the difference between the reduced model and the original model is minimized. A solution to linear and polynomial kernels was first introduced in the context of kernel SVM [11] and subsequently extended to kernel PCA [3]. A similar solution was also developed for the radial basis function (RBF) kernel cases [8] [12]. These works represent the state-of-the-art of the kernel approximation and are widely adopted in applications involving large dataset. However, in [8], the greedy selection of the examples is not optimal since the examples are incrementally selected to expand the linear span of the kernel space without considering the correlation between a newly selected example and the previous examples in the kernel feature space. Although this problem is addressed in [12], it uses a different objective function that pursues training speed at a cost of significant drop in the accuracy of approximation. Both methods do not guarantee that the mapping error is minimized. Additionally, both methods define their objective functions in the implicitly-defined kernel feature space which increase the computational cost.

This paper propose a new solution to the approximation problem that keeps a balanced trade-off between accuracy and efficiency. The objective function is designed to best reflect the purpose of the KPCA and the proposed algorithm performs the minimization in the mapping space instead of the kernel feature space so as to ensure the reduced KPCA is an optimal approximation of the original KPCA. Experimental results have shown that the proposed algorithm can achieve much lower mapping errors with comparable computational cost compared to the algorithm proposed in [8] [12].

The rest of this paper is organized as follows: Section 2 presents a brief overview of kernel PCA. Section 3 formulates the reduced set approximation problem, describes the proposed algorithm and highlights its differences to the previous algorithms. Experimental results are given in Section 4. Section 5 concludes the paper.

II. KERNEL PCA

Kernel PCA (KPCA) is a non-linear dimensionality reduction technique that can be regarded as a kernel expansion of the conventional linear PCA [13]. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of $n$ samples, each sample $x_i, i = 1, 2, \ldots, n$ being a $D$ dimensional feature vector. KPCA maps the samples into a kernel feature space using the mapping $\phi(x)$ and then performs PCA in the feature space. It is based on the fact that a vector mapping $\phi(x)$ can always be found if its inner-product kernel $k(x, y) = \langle \phi(x), \phi(y) \rangle$ is defined and positive semi-definitive. Given the $n$ samples, A kernel PCA model can be trained in two steps [3]:

1) Define the kernel $k(x, y)$ and calculate the kernel matrix

$$
K = [K_{ij}], K_{ij} = k(x_i, x_j)
$$
and centered kernel matrix

$$\tilde{K} = HKH$$

where $H = I_n - 1_n$, $I_n$ is an $n \times n$ identity matrix and $1_n$ denotes a $n \times n$ matrix in which each element takes the value 1/n.

2) Compute $d$ largest eigenvalues $\lambda_1...\lambda_d$ and the corresponding eigenvectors

$$A = [a_1, \cdots, a_d]$$

of $\tilde{K}$. where $a_i, i = 1, 2, \cdots, d$ are all $n$ dimensional column vectors and scaled such that $|a_i| = \frac{1}{\sqrt{n}}$.

Given a new sample $z$, its mapping onto the PCA subspace can be calculated by

$$y(z) = P^T \phi(z) = P^T(\phi(z) - \Phi_1 1_n)$$
$$P = \Phi H_A$$

where $P$ is the column matrix of the first $d$ principal components, $\Phi_X$ is the column matrix of $\phi(x_i), 1_n, 1$ is a column vector with each elements being 1/n and $\phi(z) = \phi(z) - \Phi_1 1_n$ is the centered $\phi(z)$.

Eq. 2 can be rewritten as

$$y(z) = w(z) - b$$
$$w(z) = (H_A)^T(k(x_1, z), \cdots, k(x_n, z))^T$$
$$b = (H_A)^TK1_n$$

where $H_A = HA$ and both $w(z)$ and $b$ are $d$-dimensional vectors representing the projection of $\phi(z)$ onto the principal components and the mean of $\phi(z)$ over the $n$ samples.

Notice that in cases where $d$ is close or equal to $n$, $\tilde{K}$ is often not a full rank matrix, thus some $\lambda_i$ could be very small or equal to zero. Consequently, scaling of the $a_i$ can lead to infinite $|a_i|$ and numerical instability would occur in the training. To avoid this, we first reconstruct the feature vectors in an $n$-dimensional interim space by employing multidimensional scaling (MDS) technique, in which the relative distances and inner-products between vectors are preserved. Then, PCA is performed in the interim space. Details of the algorithm are given below:

1) Obtain the kernel matrix $K$ of the dataset $X = x_1, x_2, \cdots, x_n$.

2) Find the $n$-dimensional representation $U_X = (u(x_1)u(x_2)...u(x_n))^T$ such that $< u(x_i), u(x_j)> = k(x_i, x_j)$, $U_X = D^{\frac{1}{2}}B^{T}$

where $D$ and $B$ are the diagonal matrix of the eigenvalues and column matrix of eigenvectors of $K$ respectively. Notice that $K$ is not the centered $\tilde{K}$.

3) Perform PCA on $U_X$, and obtain the projections of $U_X$ on all the principal components, the result is $W_X$.

4) Calculate $H_A$ by solving

$$w(z) = (H_A)^T\Phi_X^T\phi(z) = (H_A)^TW_X^Tw(z)$$

The result is $H_A = W_X^+$, where + denotes Moore-Penrose pseudoinverse.

5) calculate $b$ by $b = (H_A)^TK1_n$.

Although the algorithm requires twice eigen-decomposition, it avoids numerical instability when $d$ is high.

III. GREEDY APPROXIMATION

A major drawback in applying KPCA is the requirement to keep both $H_A$ and all of the $n$ training samples in $X$, and the time required to calculate $k(X, z)$ for any given new sample $z$. When $n$ is very large, KPCA often becomes impractical to use. One way to solve the problem is to find a reduced KPCA model, given by $X$ and $H_A$, such that the mapping of a new observation, $z$ in the reduced KPCA model is sufficiently close to the mapping of the sample in the original KPCA space, given by $X$ and $H_A$, that is, to minimize the the mean squared error of the mapping:

$$\varepsilon = E(||\tilde{y}(z) - y(z)||^2)$$

where $\tilde{y}(z)$ is the mapping function of the reduced KPCA model:

$$\tilde{y}(z) = \tilde{P}^T\hat{\phi}(z)$$

$P$ follows the definition of Eq.2 and $\tilde{P} = \Phi_X \tilde{H}_A$, where $\tilde{H}_A$ and $X$ are a $m \times d$ matrix and a column matrix of $m$ examples that are used to approximate $H_A$, $X$ respectively and $m << n$

Eq.6 can be written as:

$$\varepsilon = E(||(\tilde{P} - P)^T\hat{\phi}(z)||^2)$$

where $tr$ denotes the trace of matrix. If the factor of $\hat{\phi}(z)$ is ignored, then minimizing Eq.(6) is equivalent to minimizing $||P - \tilde{P}||$ [8]. Notice that $E(\hat{\phi}(z)\hat{\phi}(z)^T)$ is the covariance of the sample population in the kernel feature space which can be estimated from the training samples.

$$E(\hat{\phi}(z)\hat{\phi}(z)^T) \approx \text{cov}(\hat{\phi}(x))$$

where $\text{cov}(\hat{\phi}(x))$ is the covariance matrix of $\hat{\phi}(x)$ in the feature space, also considering:

$$P_n^T\text{cov}(\hat{\phi}(x))P_n = D_n$$

where $D_n$ is an $n \times n$ diagonal matrix of all $n$ eigenvalues $\lambda_1, \cdots, \lambda_n$ of $K$, and $P_n$ is the column matrix of all $n$
principal components, Eq.8 can be more effectively written as:

\[
\varepsilon = \text{tr}((\tilde{P} - P)^T P_n D_n P_n^T (\tilde{P} - P)) = \text{tr}((\tilde{P}^T P_n - I_{d,n}) D_n (\tilde{P}^T P_n - I_{d,n})^T) = ||(\tilde{P}^T P_n D_n^2 - D_n^2) ||^2 \\
= ||(\tilde{H}_A^T Q^T P_n D_n^2 - D_n^2) ||^2 \\
= ||(||D_n^2 \tilde{W}_X\tilde{H}_A - D_n^2) ||^2
\]

where \(I_{d,n}\) is the first \(d\) row of \(I_{n,n}\), \(D_n^4\) is the \(n \times n\) diagonal matrix of \(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n}\) and \(D_n^2\) is the first \(d\) row of \(D_n^4\). Unlike [12] and [8] which define the objective function in kernel feature space, our objective function Eq. 11 is defined in the KPCA mapping space which provides significant computational advantages.

Finding all vectors in \(\tilde{H}_A\) and \(\tilde{X}\) simultaneously according to Eq.11 is not trivial. We propose a greedy algorithm that selects \(\tilde{x}\) one-by-one by iteratively executing the following two steps [8] [12]:

1) select one sample \(\tilde{x}\) from \(X\) such that \(\varepsilon(\tilde{X} \cup \tilde{x})\) is minimized
2) add the sample \(x\) into \(\tilde{X}\), \(\tilde{X} = \tilde{X} \cup \tilde{x}\)

To simplify the first step, assuming \(\tilde{D}_n^2 \tilde{W}_X\) can be QR decomposed into a column matrix of orthogonalized vector \(Q\) and a right triangular square matrix \(R\): \(\tilde{D}_n^2 \tilde{W}_X = QR\), \(\varepsilon\) then becomes:

\[
\varepsilon = ||(QR\tilde{H}_A - (\tilde{D}_n^2))^T)||^2
\]

This is a typical quadratic programming problem, the solution of \(\tilde{H}_A\) for a given \(Q\) is:

\[
\tilde{H}_A = (QR)^+ (\tilde{D}_n^2)^T = R^{-1}Q^T (\tilde{D}_n^2)^T
\]

Hence Eq.12 becomes:

\[
\varepsilon = ||((QQ^T - I)(\tilde{D}_n^2)^T)||^2 = \sum_{i=1}^{d} ||QQ^T v_i - v_i||^2 = \sum_{i=1}^{d} (||v_i||^2 + ||QQ^T v_i||^2 - 2v_i^T QQ^T v_i) = \sum_{i=1}^{d} (\lambda_i - ||Q^T v_i||^2)
\]

where \(v_i\) is the \(i^{th}\) column of \((\tilde{D}_n^2)^T\). So minimizing \(\varepsilon\) is equivalent to maximizing:

\[
\zeta = ||(Q^T (\tilde{D}_n^2)^T)||^2
\]

This optimization term is easy to calculate, and most importantly, when a new \(\tilde{x}\) is added to \(\tilde{X}\), matrix \(Q\) would change little because the new \(Q\) can be easily obtained by \(Q = [q_1, \ldots, q_\tilde{x}]\), being Gram-Schmidt orthogonalized with the rest of the column vectors of \(Q\). Therefore, the increase of Eq.(15) for each new \(\tilde{x}\) being added to \(\tilde{X}\) can be expressed as:

\[
\Delta \zeta = ||(Q^T (\tilde{D}_n^2)^T)||^2 - ||Q^T (\tilde{D}_n^2)^T||^2 = ||q^T (\tilde{D}_n^2)^T||^2
\]

Iterative maximization of this term forms the main idea of the proposed greedy approximation algorithm. Although it is possible to find the new \(\tilde{x}\) in the entire input space by gradient descent optimization, such optimization will be slow and unstable because Eq.(16) will have many local maxima with respect to \(\tilde{x}\). Nevertheless, finding \(X\) from \(\tilde{X}\) can be considered as a problem of sampling samples from the clusters defined by \(X\) [8]. Therefore, minimizing Eq.(12) becomes a finite-state searching problem and we propose the following algorithm:

1) Given a dataset \(X\) in the input space and a kernel function \(k(.,.)\), obtain the image of \(X\) in the KPCA mapping space, denoted by \(\tilde{W}_X\). Initialize the orthogonalized candidate set \(V = \tilde{D}_n^2 \tilde{W}_X\).(Complexity: \(O(n^2)\))
2) Calculate the inner-product matrix \(N = V^T \tilde{D}_n^2\), find one of its rows with the largest norm. (Complexity: \(O(2nd)\))
3) Add the \(i^{th}\) column vector of \(V\) to the reduced set \(\tilde{X} = \tilde{X} \cup v_i\), \(v_i\) can be removed from \(V\).
4) Update \(V\) by orthogonize its column vectors \(V = V - v_i v_i^T V\). (Complexity: \(O(n^2)\))
5) Repeat step 3 to 5 until \(m\) examples are chosen or the mapping error is lower than a threshold. (Complexity: \(O(n^2) + O(2nd) + O(n^2)\))
6) Obtain \(\tilde{H}_A\) by \(\tilde{H}_A = (\tilde{D}_n^2 \tilde{W}_X)^+\).

The complexity of the proposed algorithm is in the order of \(O(m(2n^2 + 2nd))\). This is slightly faster than the algorithm proposed in [12], whose complexity is \(O(mp^2)\) (\(p\) is the search depth that usually equals to a quarter of \(n\)).

### IV. Experimental Results

Six real datasets from the intelligent data analysis (IDA) benchmark repository [14] were used to evaluate the performance of the proposed algorithm. These datasets are banana, breast cancer, diabetes, flare, german and heart. Samples in each dataset have been manually labeled into two classes.
Since each dataset contains large number of samples, we randomly chose a subset of the samples in each experiment due to the limitation of the available computing resources. The numbers of samples chosen are same as the ones used in [12]. Table I lists the numbers of samples used in the experiments and the dimensions of the six datasets.

The KPCA is primarily used as an effective tool for manifold learning or data compression. In manifold learning, the KPCA aims to compute the projection of an unknown sample into the kernel mapping space so its distance to the manifold can be determined. In data compression, the unknown sample is reconstructed from a compact set of vectors defining the kernel mapping space. Due to these reasons, the following two sets of experiments were designed to evaluate the proposed algorithm by measuring the mapping and reconstruction errors between the reduced KPCA and original KPCA. The reconstruction error is calculated between the original example and the reconstructed one from the mapping of the example in the kernel feature space. The results were compared to the ones obtained by the Franc’s method [12].

Figures 1 and 2 show the mean squared mapping and reconstruction errors versus the number of samples in the reduced set. The errors were averaged over 20 runs, in each run, the number of samples specified in Table I were randomly selected from the original datasets. Since in Franc’s method the maximum number of principal components are reduced to \( m \), the mapping and reconstruction errors were calculated using the first \( m \) principal components. In other words, when \( m = 1 \), the errors were calculated using the first principal component. The mapping errors of the reduced KPCA obtained by the proposed method produced substantially lower mapping errors in all cases than Franc’s method. This indicates the proposed method is particularly useful for the cases when KPCA is used as a manifold learning tool.

In terms of the reconstruction error, the proposed method performed comparable to Franc’s method. Since the kernel reconstruction is a reverse mapping from the kernel feature space to the input space and all principal components are used, the reconstruction result is solely depending on the configuration of the kernel feature space but not relevant to the direction of the principal components, therefore we can conclude that the proposed method algorithm has comparable capability of approximating the kernel feature space to Franc’s method. However, the directions of the principle components of the reduced set obtained by Franc’s method are not necessarily align with the direction of the original KPCA.

V. CONCLUSION

This paper propose a new algorithm to obtain a reduced KPCA model that can be used to accelerate the speed of kernel mapping and reconstruction. It aims to select a reduced example set to approximate the original kernel feature space by greedily minimizing the mean squared error of the mapping function. This algorithm balances the trade-off between effectiveness and efficiency. Experimental results have shown that the proposed algorithm is more efficient in computation and effective with lower mapping errors than previous algorithms.

REFERENCES


Figure 1. Mapping errors (y axis) w.r.t. m (x axis), the results obtained by Franc’s algorithm and the proposed algorithm are represented by red and blue curves respectively.

Figure 2. Reconstruction errors (y axis) w.r.t. m (x axis), the results obtained by Franc’s algorithm and the proposed algorithm are represented by red and blue curves respectively.