Pattern Recognition Letters 77 (2016) 50–57

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Kernel matrix decomposition via empirical kernel map☆

Sanparith Marukatat*

IMG Lab, NECTEC, 112 Thailand Science Park, Pathumthani, Thailand

ARTICLE INFO

Article history:
Received 5 August 2015
Available online 9 April 2016

Keywords:
Kernel PCA
Eigen-decomposition
Empirical kernel map
Random projection

ABSTRACT

Kernel principal component analysis (KPCA) is a popular extension of the classical PCA that allows non-linear subspace projection. It is based on eigen-decomposition of the kernel matrix. The main crux of KPCA lies in the computational cost of the eigen-decomposition step. In this paper, we show that this decomposition can also be done by analyzing the covariance matrix obtained from the *empirical kernel map*. We can further reduce the computational cost by combining the empirical kernel map with random projection. Experimental results show that the proposed method accurately approximates the eigenvalues/eigenvectors of the original kernel matrix.

1. Introduction

Kernel principal component analysis (KPCA) is a popular extension of the classical PCA that allows non-linear subspace projection. The main crux of KPCA lies in the computational cost of the eigen-decomposition step of the kernel matrix. For a given set of n points, the size of the kernel matrix is $\mathcal{O}(n^2)$ and the complexity of its eigen-decomposition is $\mathcal{O}(n^3)$. As the size of the dataset is becoming larger and larger nowadays, direct application of KPCA is becoming impractical.

To reduce the complexity, Kim et al. [10] proposed a kernelized version of the classical generalized Hebbian algorithm [16,20]. This adaptive algorithm allows computing kernel principal subspace without eigen-decomposing the kernel matrix. However, to use this method, one has to choose the dimension of the kernel subspace beforehand which could be impractical in some cases.

Smola and Schökopf [23] proposed a simpler method that constructs the kernel subspace by incrementally selecting new basis vector from the dataset. The dimension of the kernel subspace is then determined as function of the kernel matrix approximation using the selected basis vectors. Similar approach was considered by Franc and Hlavac [7] but with selection criterion that measures the reconstruction error of vectors in kernel space instead. The main computational cost of this approach lies in the evaluation of the new basis vector.

To further reduce the computational cost, some authors, e.g. [12,14], proposed to group the data first then use the *pre-image* of the mean of each cluster to construct the kernel matrix. The pre-image problem refers to finding the point in the input space given its image by the kernel mapping function. As the kernel mapping function is generally unknown, this is a non-trivial problem. In fact, in some kernel space the pre-image may not even exist [15]. Hence, these methods contain two sources of imprecision namely the use of cluster's representation and the pre-image reconstruction. Chin and Suter [4] also employed the pre-image reconstruction in their incremental KPCA in order to maintain the complexity of the update procedure.

Instead of synthesizing data using pre-image reconstruction, Williams and Seeger [27] showed how to extend the eigenstructure (e.g. eigenvalues/eigenvectors) computed from a small subset of examples to cover the whole dataset using the Nyström approximation formula. This approach has received large attention recently. Indeed, this method is simple yet provides good results in practice [6,18]. Several authors have extended this method by investigating different subset selection strategies, e.g. [5,13,25].

Another interesting approach is proposed by Achlioptas et al. [2]. In this work, the given kernel κ is replaced by a "randomized kernel" which behaves like κ in expectation. The authors showed that one can recover the eigenvectors of the original kernel matrix using the decomposition of the randomized kernel matrix. Achlioptas et al. constructed the randomized kernel by randomly *omitting* entries of the original kernel matrix leading to a sparser matrix that is easier to be eigen-decomposed. We can also construct randomized kernel using other methods such as the *tensor sketch* for polynomial kernel [17] or *random Fourier features* for RBF kernel [19]. This approach is, nonetheless, not generic; it depends on the type of the given kernel.

This work proposes a generic algorithm to approximate the eigen-decomposition of any kernel matrix. The proposed method

☆ This paper has been recommended for acceptance by S. Sarkar.

* Tel.: +66 2 564 6900x2249; fax: +66 2 564 6873.
E-mail address: sanparith.marukatat@nectec.or.th, peune.i3@gmail.com

http://dx.doi.org/10.1016/j.patrec.2016.03.031
0167-8655/

is based on two ideas. Firstly, the eigen-decomposition of a large covariance matrix can be performed in a lower dimensional space if we can preserve the dot product between vectors in this dataset. This dot product preservation can be done, for example using the random projection method. Secondly, the eigen-structure of the kernel matrix is related to that of the *covariance matrix* constructed from the *empirical kernel map (EKM)* [21]. The combination of these two ideas gives a simple yet accurate method for approximate the eigen-structure of the kernel matrix.

In the following, Section 2 first discusses eigen-structure approximation using dot product preserving function. Section 3 presents the link between the empirical kernel map and the eigen-structure of the kernel matrix. Then Section 4 and 5 present the experiments and the conclusion, respectively.

2. Dot product and eigen-structure approximation

The structure of the principal subspace depends on the dot product between vectors in the dataset. If we can transform data into a lower dimensional subspace while preserving the dot product, then we may work in this lower dimensional subspace instead of the original one. This can largely reduce the computational cost of the eigen-decomposition procedure. Next section presents formal algorithm and justification of this method. Then Section 2.2 discusses the choice of dot preserving function.

2.1. Basic idea

Consider a set of multi-dimensional vectors $x_1, \ldots, x_n$, $x_t \in \mathbb{R}^d$. We shall assume that this dataset has zero mean. Let $X = [x_1; \ldots; x_n]$ be the data matrix of size $d \times n$. The *covariance matrix* is given by XX^T and the *dot matrix* is X^TX. It is straightforward proving that if v is an eigenvector of XX^T, then X^Tv is an eigenvector of X^TX. In an analogous manner, if u is an eigenvector of X^TX, then Xu is an eigenvector of XX^T. Noted that in both cases, the eigenvalue remain unchanged.

Suppose that each point x_t can be transformed into another vector z_t using a dot product preserving function f such that, $z_i = f(x_i)$ and $\forall i, j, 1 \le i, j \le n$

$$x_i^T x_j = z_i^T z_j + \varepsilon_{ij}, \tag{1}$$

where ε_{ij} is the error in preserving the dot product. Therefore, we have

$$X^TX = Z^TZ + E \tag{2}$$

with Z the data matrix of the transformed vectors and E the error matrix. From the perturbation theory, we know that the difference between the eigenvalue of X^TX and Z^TZ can be upper-bounded by the size of E. Let $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n$ be eigenvalues of X^TX and $\tilde{\lambda}_1 \ge \cdots \ge \tilde{\lambda}_n$ be eigenvalues of Z^TZ, then we have

$$|\lambda_i - \tilde{\lambda}_i| \le \|E\|_2 \le n \max_{i,j} \varepsilon_{ij}. \tag{3}$$

Thus, if the dot product error is small enough, the approximate eigenvalues should be close to the correct ones.

Eigenvectors analysis is more complicated. Indeed, if λ is a repeated eigenvalue of X^TX, then there are infinite number of possible eigenvector bases for the associated invariant subspace [8]. Hence, to correctly identify an eigenvector, the corresponding eigenvalue must be sufficiently distinct from other eigenvalues. In this work, we consider a softer condition that large and small eigenvalues are separable:

$$\min_{i=1,\ldots,m,\, j=m+1,\ldots,n} |\lambda_i - \lambda_j| = \Delta > 0. \tag{4}$$

Then, a result from [24] says that if $\|E\|_2 \le \Delta/5$, then the difference between the m-dimensional principal subspaces obtained from X^TX and from Z^TZ can be upper bounded as follows:

$$\text{dist}(\text{PCA}(X^TX), \text{PCA}(Z^TZ)) \le \frac{4}{\Delta}\|F_{21}\|_2, \tag{5}$$

where F_{21} is an $(n-m) \times m$ matrix given by

$$V^TEV = \begin{bmatrix} F_{11} & F_{21}^T \\ F_{21} & F_{22} \end{bmatrix} \tag{6}$$

with $V = [v_1, \ldots, v_n]$ the matrix of all eigenvectors of X^TX and the "dist" function in Eq. (5) is the norm of the difference between then eigenvectors of the two subspaces. In summary, if the eigenvalues separation (Δ) is large and if we have small dot product preserving error ($\max_{i,j}\varepsilon_{ij} \le \Delta/5n$), then the obtained principal space should be closed to the real one.

From the above idea, an eigenvector of X^TX can be approximated by Z^Tv where v is an eigenvector of Z^TZ. Then we can convert this eigenvector into that of XX^T by left-multiplying it with X. This idea is summarized in Algorithm 1.

Algorithm 1 Approximate eigen-decomposition of the covariance matrix of the input dataset with a dot product preserving function f such as f_{RP} or $f_{CS}^{(h,s)}$.

1: **procedure** APPROX–EIG–COV($x_1, \ldots, x_n$, f)
2: Let $z_t = f(x_t), t = 1, \ldots, n$ be the image of $x_1, \ldots, x_n$ by f and $Z = [z_1, \ldots, z_n]$ be the data matrix having z_t as tth column.
3: Compute the covariance matrix ZZ^T
4: Let (λ_i, v_i), $i = 1, \ldots, m$ be m largest eigenvalues and the corresponding eigenvectors of ZZ^T.
5: Compute $(\hat{\lambda}_i, \hat{v}_i)$ approximate eigenvalues/eigenvectors of X^TX with $\hat{\lambda}_i = \lambda_i$ and $\hat{v}_i = Z^Tv_i$.
6: Compute $\hat{u}_i$ approximate eigenvectors of XX^T with $\hat{u}_i = X\hat{v}_i$. Noted that the corresponding eigenvalue of $\hat{u}_i$ is $\hat{\lambda}_i$.
7: **return** $(\hat{\lambda}_i, \hat{u}_i)$, $i = 1, \ldots, m$
8: **end procedure**

2.2. Dot product preserving function

Algorithm 1 relies on a dot product preserving function f. This function should be easy to construct. In this work, we consider two particular functions that involve *randomization* but allow preserving dot product *with high probability*. These two functions are the *random projection (RP)* (Section 2.2.1) and the *count sketch (CS)* (Section 2.2.2).

2.2.1. Random projection

Random projection owes its popularity to the classical result called the Johnson–Lindenstrauss lemma [9]. This lemma asserts that we can embed d-dimensional data into a lower subspace of dimension q, that is independent of d and that preserves the Euclidean distance up to a multiplicative factor between $1-\varepsilon$ and $1+\varepsilon$. Known constructions of such embedding involve random projections [1], e.g. the Gaussian random projection defined as follows:

$$f_{RP}(x) = \frac{1}{\sqrt{q}}Ax \tag{7}$$

where A is a $q \times d$ matrix whose elements are sampled from Gaussian distribution with zero mean and unit variance.

It can be proved that with probability at least $1-\delta$ if $q = \mathcal{O}(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ then[1]

$$\Pr\left\{|x^Ty - f_{RP}(x)^T f_{RP}(y)| \ge \varepsilon\right\} \le 4\exp\left(-\frac{(\varepsilon^2 - \varepsilon^3)q}{4}\right) \tag{8}$$

[1] In general, we consider $\delta = \frac{1}{n}$ for dataset of n points.