CSE 475: Statistical Methods in AI **Monsoon 2018**

Lec 16: More on Dimensionality Reduction: LDA and KPCA

Lecturer: C. V. Jawahar *Date: Oct. 4, 2018*

16.52 PCA: Summary

- Objective of PCA is to project the data to direction which best *preserves its covariance structure.*
- Let us assume that the data is centered $\sum_i \mathbf{x_i} = 0$. Also $\mathbf{x_i} \in R^N$, $i = 1, \ldots, M$. Centering is done by subtracting the mean from all the samples.
- Covariance matrix $C = \frac{1}{M}\sum_{i=1}^{M} \mathbf{x_i}\mathbf{x_i^T}$ is first calculated. C is a $N \times N$ matrix. We avoid the use of Σ for the covariance matrix here. Let us not get it confused with the $\sum$ we use for Summation later extensively here.
- Assume we have a set of M centered observations: $\mathbf{x_k}, k = 1 \ldots M, \mathbf{x}_k \in \mathbf{R}^N, \sum_{k=1}^{M} \mathbf{x}_k = 0$

  For linear PCA, we solve the equation

$$C\mathbf{v} = \lambda\mathbf{v}$$

- This leads to eigen vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots$. There are a total of $\min(M, N)$ nonzero eigen values.
- And data is projected to P eigen vectors corresponding to top $P < N$ eigen values of the covariance matrix C.

We see two obvious scopes for improvement:

- PCA does not focus on discriminative nature. It aims at compression/compaction. There is no explicit objective that helps separation.
- PCA is linear. A nonlinear dimensionality reduction could have been more useful.

16.53 LDA

Let us now consider a "discriminative dimensionality reduction". Popularly this is called Linear Discriminant Analysis or Fisher Discriminant Analysis. This is supervised in nature (i.e., we use the class labels y_i also.).

Let us assume that we have two classes A and B. We are interested in finding a new feature $z = \mathbf{u}^T\mathbf{x}$ by projecting on to a new vector $\mathbf{u}$. What are our requirements so that this is good for the classification?

- After the projection, classes should be compact (i.e., samples from A should all come closer and samples from B should all come closer.). All the samples come closer to their own means.
- After the projection classes A and B should be well separated (i.e., they are easily separable.) Means of the classes become farther.

Let us introduce two new notations i.e., within scatter and between scatter of the classes. Both are captured as:

$$S_B = [\mu_A - \mu_B][\mu_A - \mu_B]^T$$

$$S_W = S_A + S_B$$

$$= \sum_{\mathbf{x_i} \in A} [\mathbf{x_i} - \mu_A][\mathbf{x_i} - \mu_A]^T + \sum_{\mathbf{x_i} \in B} [\mathbf{x_i} - \mu_B][\mathbf{x_i} - \mu_B]^T$$

Assume we project onto a new (unknown) direction $\mathbf{u}$. After the projection S_B becomes $\mathbf{u}^T\mathbf{S_B}\mathbf{u}$ and S_w becomes $\mathbf{u}^T\mathbf{S_W}\mathbf{u}$

We convert this into a new objective function

$$J(\mathbf{u}) = \frac{\mathbf{u}^T\mathbf{S_B}\mathbf{u}}{\mathbf{u}^T\mathbf{S_W}\mathbf{u}}$$

We would like to maximize this quantity. Since the optima is invariant scaling, let us consider the problem as

$$\max_{\mathbf{u}} \frac{1}{2}\mathbf{u}^T\mathbf{S_B}\mathbf{u}$$

such that $\mathbf{u}^T\mathbf{S_W}\mathbf{u} = 1$. The maximization problem now (cf: Lagrangian)

$$\max_{\mathbf{u}} \frac{1}{2}\mathbf{u}^T\mathbf{S_B}\mathbf{u} - \frac{\lambda}{2}(\mathbf{u}^T\mathbf{S_W}\mathbf{u} - 1)$$

Differentiating with respect to $\mathbf{u}$ and equating to zero.

$$\mathbf{S_B}\mathbf{u} = \lambda\mathbf{S_W}\mathbf{u} \tag{16.29}$$

Such problems are called generalized eigen value problems. $\mathbf{S_B}\mathbf{u}$ is a vector along $[\mu_A - \mu_B]$. Therefore if $\mathbf{S_W}$ can be inverted,

$$\mathbf{u} = \alpha\mathbf{S_W}^{-1}[\mu_A - \mu_B]$$

Many questions are left to you to figure out.

- How do we extend this to multi class (beyond two classes)? How does the definition of S_w and S_B change?
- How do we get multiple direction/features/$\mathbf{u}$ than one? What is the second best direction, given that the first one is identified?
- What is the generalized eigen value problem mentioned here? Isn't it LDA also an eigen vector solution?

16.53.1 Tricks

There are a number of tricks that we could do to make our life easy. For example, $\mathbf{S_W}$ could be redefined as:

$$\mathbf{S_W} = \mathbf{S_W} + \rho\mathbf{I} \tag{16.30}$$

where ρ is a small real quantity. How does this way of regularizing $\mathbf{S_W}$ help?

Alternatively, one could define $\mathbf{S_W}$ as (see "Multiple-Exemplar Discriminant Analysis for Face Recognition" for details). Here the scatter is computed not with respect to mean but with respect to each samples and added.

$$\mathbf{S_W} = \sum_{i=1}^{C} \frac{1}{N_i^2} \sum_{j=1}^{N_i} \sum_{k=1}^{N_i} [\mathbf{x_j^i} - \mathbf{x_k}^i][\mathbf{x_j^i} - \mathbf{x_k}^i]^T \tag{16.31}$$

The new definitions of $\mathbf{S_W}$ is some what different from that of the original one. However, both these help in making the $\mathbf{S_W}$ full rank.

Another trick is to first apply PCA, and then LDA. This also helps in making $\mathbf{S_W}$ full rank.

Q: *Explain how all the three above methods help in practice?*

16.54 Kernel PCA

16.54.1 Preliminaries

We start with the assumption that the data is centered. Then we know that the covariance matrix is:

$$C = \frac{1}{M} \sum_{i=1}^{M} \mathbf{x_i} \mathbf{x_i^T} \text{ and } C\mathbf{v} = \lambda\mathbf{v}$$

In Kernel PCA, PCA is applied in the feature space. The data is mapped into another space, i.e.,

$$\mathbf{x} \to \phi(\mathbf{x}).$$

Then assuming that data is centered, covariance matrix can be computed as:

$$C = \frac{1}{M} \sum_{j=1}^{M} \phi(\mathbf{x_j})\phi(\mathbf{x_j})^T \tag{16.32}$$

For KPCA, we need to solve:

$$C\mathbf{V} = \lambda\mathbf{V} \tag{16.33}$$

- We do not want to work with $\phi(\cdot)$ explicitly. We can use the kernel $\kappa(\cdot,\cdot)$ and curcumvent this requriement.
- If $\phi(\cdot)$ maps to an infinite dimension, we will have to work with $\infty \times \infty$ matrices. That is impractical.
- Note that even in such cases, we will have ONLY $\leq M$ eigen vectors for C.

We can rewrite it as

$$\mathbf{V} = \frac{1}{\lambda} C\mathbf{V} = \frac{1}{\lambda M} \sum_{i=1}^{M} \phi(\mathbf{x_i})\phi(\mathbf{x_i})^{\mathbf{T}}\mathbf{V}$$

$$= \frac{1}{M} \sum_{i=1}^{M} (\phi(\mathbf{x_i})^{\mathbf{T}}\mathbf{V})\phi(\mathbf{x_i}) = \sum_i \alpha_i \phi(\mathbf{x_i})$$

Or

$$\mathbf{V} = \sum_{i=1}^{M} \alpha_i \phi(\mathbf{x_i})$$

16.54.2 Computing

Taking a dot product with $\phi(\mathbf{x_k})$ on both sides of Equation 16.33: $(C\mathbf{V} = \lambda\mathbf{V})$

$$\phi(\mathbf{x_k}) \cdot C\mathbf{V} = \lambda\phi(\mathbf{x_k}) \cdot \mathbf{V} \ \ \forall k \tag{16.34}$$

$$\sum_{j=1}^{M} (\phi(\mathbf{x_k}) \cdot \phi(\mathbf{x_j})) \sum_{i=1}^{M} \alpha_i (\phi(\mathbf{x_j}) \cdot \phi(\mathbf{x_i})) \tag{16.35}$$

$$= \lambda M \sum_{i=1}^{M} \alpha_i (\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_k})) \forall k \tag{16.36}$$

Defining an MxM matrix K by

$$K_{ij} = (\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_i})) = \kappa(\mathbf{x_i}, \mathbf{x_j})$$

we have

$$M\lambda K\alpha = K^2\alpha$$
$$M\lambda\alpha = K\alpha$$

Thus the vectors α are the eigen vectors of K. α is a vector of dimension M. Let α^k be the eigen vector corresponding to λ_k. The resulting set of eigenvectors $\mathbf{V}^k$ can now be computed as:

$$\mathbf{V^k} = \sum_{i=1}^{M} \alpha_i^k \phi(\mathbf{x_i})$$

16.54.3 Projecting a New Sample

And the projection of a sample $\phi(\mathbf{x})$ onto this principal component can be evaluated as:

$$\begin{aligned}(\mathbf{V^k} \cdot \phi(\mathbf{x})) &= \sum_{i=1}^{M} \alpha_i^k (\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x})) \\ &= \sum_{i=1}^{M} \alpha_i^k \kappa(\mathbf{x_i}, \mathbf{x})\end{aligned}$$

16.54.4 Centering

We started with the assumption that the data is centered. In practice it is not. How do we center the data then?

Let $\hat{\phi}(\mathbf{x_i})$ be the centered version of $\phi(\mathbf{x_i})$. The (i, j) th element of the Kernel matrix (corresponding to centered data) is:

$$\hat{K}_{ij} = \hat{\phi}(\mathbf{x_i})^T \hat{\phi}(\mathbf{x_j}) \tag{16.37}$$

$$\hat{K}_{ij} = (\phi(\mathbf{x_i}) - \mu)^T (\phi(\mathbf{x_j}) - \mu)$$

Where $\mu = \frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x_n})$

$$\begin{aligned}\hat{K}_{ij} &= (\phi(\mathbf{x_i}) - \mu)^T (\phi(\mathbf{x_j}) - \mu) \\ &= (\phi(\mathbf{x_i}) - \frac{1}{N} \sum_{m=1}^{N} \phi(\mathbf{x_m}))^T (\phi(\mathbf{x_j}) - \frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x_n})) \\ &= K(\mathbf{x_i}, \mathbf{x_j}) - \frac{1}{N} \sum_{n=1}^{N} K(\mathbf{x_n}, \mathbf{x_i}) - \frac{1}{N} \sum_{m=1}^{N} K(\mathbf{x_m}, \mathbf{x_j}) \\ &+ \frac{1}{N^2} \sum_{m,n=1} K(\mathbf{x_m}, \mathbf{x_n})\end{aligned}$$

$$\hat{K}_{ij} = K_{ij} - \frac{1}{N} \sum_{n=1}^{N} K_{ni} - \frac{1}{N} \sum_{m=1}^{N} K_{mj} + \frac{1}{N^2} \sum_{mn} K_{mn}$$

16.54.5 Basic Procedure

1. First compute the kernel matrix K.
2. Then compute the kernel matrix $\hat{K}$ corresponding to the centered data.
3. Compute the eigen values and eigen vectors (α) of $\hat{K}$
4. Use α and the kernel function, evaluate the projection.

16.54.5.1 Kernel LDA and Beyond (*)

The kernel trick is not limited to PCA. You can also kernelaize algorithms like LDA. Please see:

- B. Scholkof, A Smola and K Muller, Nonlinear Component Analysis as a Kernel Eigenvalue Problem for KPCA
- S. Mika et al, Fisher Discriminant Analysis with Kernels for KLDA

16.55 Application: Face

We had discussed PCA being a useful cue for representing faces in the form of eigen faces. The dimensionality reduction techniques like LDA (or Fisher discriminant) and KPCA are also used in very similar manner.

Q: *Write pseudo codes for fisher and KPCA faces.*

"Face recognition using kernel Methods" (NIPS 2001) is provided a nice comparison of these methods on two benchmarks. (see more details)

Method	Reduced Space	Error Rate (%)
ICA	40	6.25 (25/400)
Eigenface	30	2.75 (11/400)
Fisherface	14	1.50 (6/400)
Kernel Eigenface, d=2	50	2.50 (10/400)
Kernel Eigenface, d=3	50	2.00 (8/400)
Kernel Fisherface (P)	14	1.25 (5/400)
Kernel Fisherface (G)	14	1.25 (5/400)

Method	Reduced Space	Error Rate (%)
ICA	30	29.09 (48/165)
Eigenface	30	28.48 (47/165)
Fisherface	14	8.48 (14/165)
Kernel Eigenface, d=2	80	27.27 (45/165)
Kernel Eigenface, d=3	60	24.24 (40/165)
Kernel Fisherface (P)	14	6.67 (11/165)
Kernel Fisherface (G)	14	6.06 (10/165)