Lecturer: Prof. Homrighausen Scribe: Soo-Young Kim

1 Nonlinear Embeddings

1.1 Lower dimesional (metric) embeddings

Spectral connectivity analysis (SCA) is a general process for finding lower dimensional structure in the data. It can be...:

- Linear or nonlinear
- Used for dimension reduction or feature creation
- PCA, Fisher discriminant analysis, Locally linear embeddings, Hessian eigenmaps, Laplacian eigenmaps, kernel PCA
- Useful as an input to classification, clustering, and regression approaches

Let's take one last look at PCA before proceding

1.2 PCA examples

PCA can do effective dimension reduction (that is, explain most of the data with $m < p$ components) as long as the data can be efficiently represented as 'lines' (or planes, or hyperplanes). In two dimensions, the example can be found in Figure 1. Figure 1 shows other data structures when PCA doesn't works well.

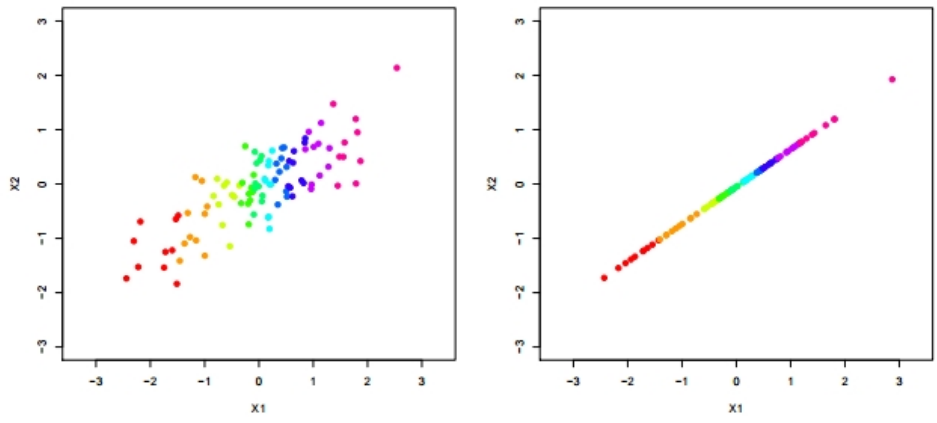

Figure 1: The example when PCA can do effective dimension reduction

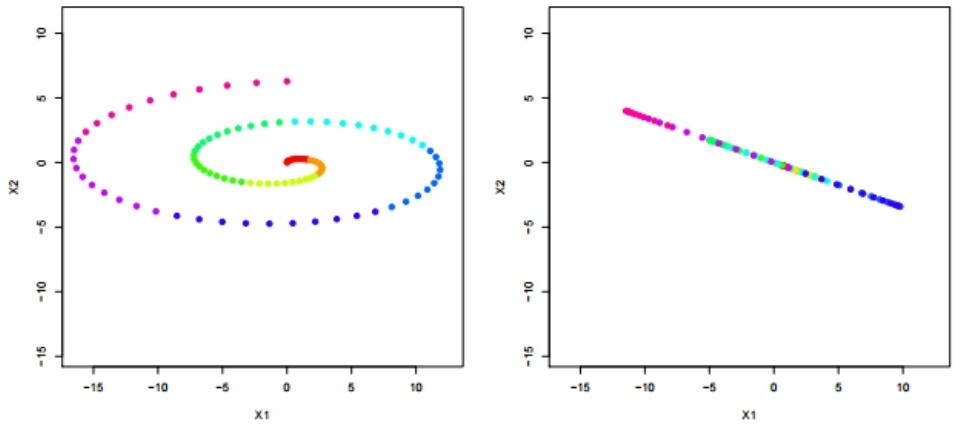

Figure 2: The example when PCA doesn't reduce dimension effectively

- PCA wants to minimize distances (equivalently maximize variance). This means it slices through the data at the meatiest point, and then the next one, and so on. If the data are 'curved' this is going to induce artifacts.
- PCA also looks at things as being close if they are near each other in a Euclidean sense [this is essentially all covariance is].
- On the spiral, our intuition says that things are 'close' only if the distance is constrained to go along the curve. In other words, purple and blue are close, blue and red are not.

1.3 PCA and covariance

PCA: Find the directions of greatest variance. This doesn't on its face seem like it maintains correlations, but observe:

$$var([a,b]^\top X) = var(ax_1 + bx_2) = a^2 Var(x_1) + b^2 Var(x_2) + 2abCov(x_1, x_2)$$

If we standardize the matrix, then this reduces to

$$var(ax_1 + bx_2) = a^2 + b^2 + 2abCov(x_1, x_2)$$

This gets maximized over $a^2 + b^2 = 1$.

- If $Cov(x_1, x_2) \approx 0$, then this gets maximized by any $a^2 + b^2 = 1$ (it doesn't matter)
- If $Cov(x_1, x_2) \approx 1$, then this gets maximized by setting $a = b = 1/\sqrt{2}$

So, in either case, we are really maintaining correlations

Correlation is fundamentally a linear phenomenon

1.4 Graphical example of the phenomenon

```
library(mvtnorm)
```

```
sigma   = matrix(c(1,sig,sig,1),nrow=2)
nsweep  = 1000
outcome = matrix(0,nrow=nsweep,ncol=2)
for(sweep in 1:nsweep){
  x               = rmvnorm(200,c(0,0),sigma)
  out.pca         = prcomp(x,center=T,scale=F)
  outcome[sweep,] = out.pca$rotation[,1]
}
plot(outcome,xlab='PC1',ylab='PC2')
```

2 Kernel PCA (KPCA)

Classical PCA comes from $\tilde{\mathbb{X}} = \mathbb{X} - M\mathbb{X} = UDV^\top$, where $M = \mathbf{1}\mathbf{1}^\top/n$ and $\mathbf{1} = (1, 1, \ldots, 1)^\top$.

However, we can just as easily get it from the outer product

$$\mathbb{K} = \tilde{\mathbb{X}}\tilde{\mathbb{X}}^\top = (I - M)\mathbb{X}\mathbb{X}^\top(I - M) = UD^2U^\top$$

and $\hat{\Sigma} \propto \tilde{\mathbb{X}}^\top\tilde{\mathbb{X}} = VD^2V^T(I - M)\mathbb{X} \in \mathbb{R}^{p\times p}$

The intuition behind KPCA is that $\mathbb{K}$ is an expansion into a kernel space, where

$$\mathbb{K}_{i,i'} = k(\tilde{X}_i, \tilde{X}_{i'}) = \langle \tilde{X}_i, \tilde{X}_{i'} \rangle$$

Reminder: Anytime we see an inner product, we can kernelize it

Following this intuition, the approach is simple:

1. Specify a (symmetric) kernel k
   e.g. $k(X, X') = \exp\{-\gamma^{-1}||X - X'||_2^2\}$
2. Form $K_{i,i'} = k(X_i, X_{i'})$
3. Standardize and get eigenvector decomposition

$$\mathbb{K} = (I - M)K(I - M) = UD^2U^\top$$

This implicitly finds the inner product:

$$k(X_i, X_{i'}) = \langle \phi(X_i), \phi(X_{i'}) \rangle$$

However, we need only specify the kernel

The scores are still $Z = UD$

The q^{th} KPCA score is (up to centering)

$$Z_{iq} = \sum_{i'=1}^{n} \beta_{i'q} k(X_i, X_{i'})$$

where $\beta_{i',q} = u_{i'q}/d_q$

Note: As we don't explicitly generate the feature map, there are no loadings

2.1 Reproducing kernel Hilbert space

Reminder: Mercer's theorem assures us that

$$k(X, X') = \sum_{j=1}^{\infty} \theta_j \phi_j(X) \phi_j(X')$$

Here, the system $(\phi_j)_{j=1}^{\infty}$ spans a space $\mathcal{H}_k$

The function space $\mathcal{H}_k$ is known as a reproducing kernel Hilbert space (RKHS)

It can also be thought of as roughly

$$\mathcal{H}_k = \{f : f(X) = \sum_{i=1}^{n} \beta_i k(X, X_i)\}$$

Which has a special inner product

$$\langle f, f \rangle_{\mathcal{H}_k} =$$

2.2 Reproducing kernel Hilbert space

Writing

$$f(X) = \sum_{i=1}^{n} \beta_i k(X, X_i)$$

The terms $k(X, X_i)$ are the representers, as

$$\langle k(\cdot, X), f \rangle_{\mathcal{H}_k} = f(X)$$

and $\mathcal{H}_k$ is called a reproducing kernel Hilbert space (RKHS) as

$$\langle k(\cdot, X), k(\cdot, X') \rangle_{\mathcal{H}_k} = k(X, X')$$

Note: For kernel methods, we are generalizing the finite dimensional Euclidean inner product

$$\langle X, X' \rangle = X^\top X'$$

2.3 Kernel methods via regularization

After specifying a kernel function k, we can define an estimator via

$$\min_{f \in \mathcal{H}_k} \hat{\mathbb{P}} \ell_f + \lambda$$

($\hat{\mathbb{P}} \ell_f = (1/n) \sum (Y_i - \delta(X_i))^2$ for the squared error) This is a (potentially) infinite dimensional optimization problem

hard, especially with a computer

It can be shown that the solution has the form

$$\hat{f}(X) = \sum_{i=1}^{n} \beta_i k(X, X_i)$$

This is known as the representer theorem

2.4 Kernel PCA

Reminder: To get the first PC in classical PCA, we want to solve

$$\max_{\alpha} \mathbb{V}\alpha^\top X \quad \text{subject to}$$

Translate this into the kernel setting, and we are trying to solve

$$\max_{g \in \mathcal{H}_k} \mathbb{V}g(X) \text{ subject to}$$

The representer theorem states that a solution to this problem is

$$\hat{g}(X) = \sum_{i=1}^{n} \beta_i k(X, X_i)$$

Compare

$$Z_{iq} = \sum_{i'=1}^{n} \beta_{i'q} k(X_i, X_{i'})$$

where $\beta_{i',q} = u_{i'q}/d_q$

2.5 KPCA: some results

Figure 3 shows the results with PCA and KPCA methods.

2.6 Semisupervised learning in practice

Looking at:

$$Z_{iq} = \sum_{i'=1}^{n} \beta_{i'q} k(X_i, X_{i'})$$

this is only defined at our observed features

Write

- $\mathcal{D}_{train} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$
- $\mathcal{D}_{test} = \{(X_1^*, Y_1^*), \ldots, (X_{n^*}^*, Y_{n^*}^*)\}$

Two common scenarios are

1. We are given $\mathcal{D}_{train}$ and $X_1^*, \ldots, X_{n^*}^*$ to build $\hat{f}$
2. We are given only $\mathcal{D}_{train}$ to build $\hat{f}$

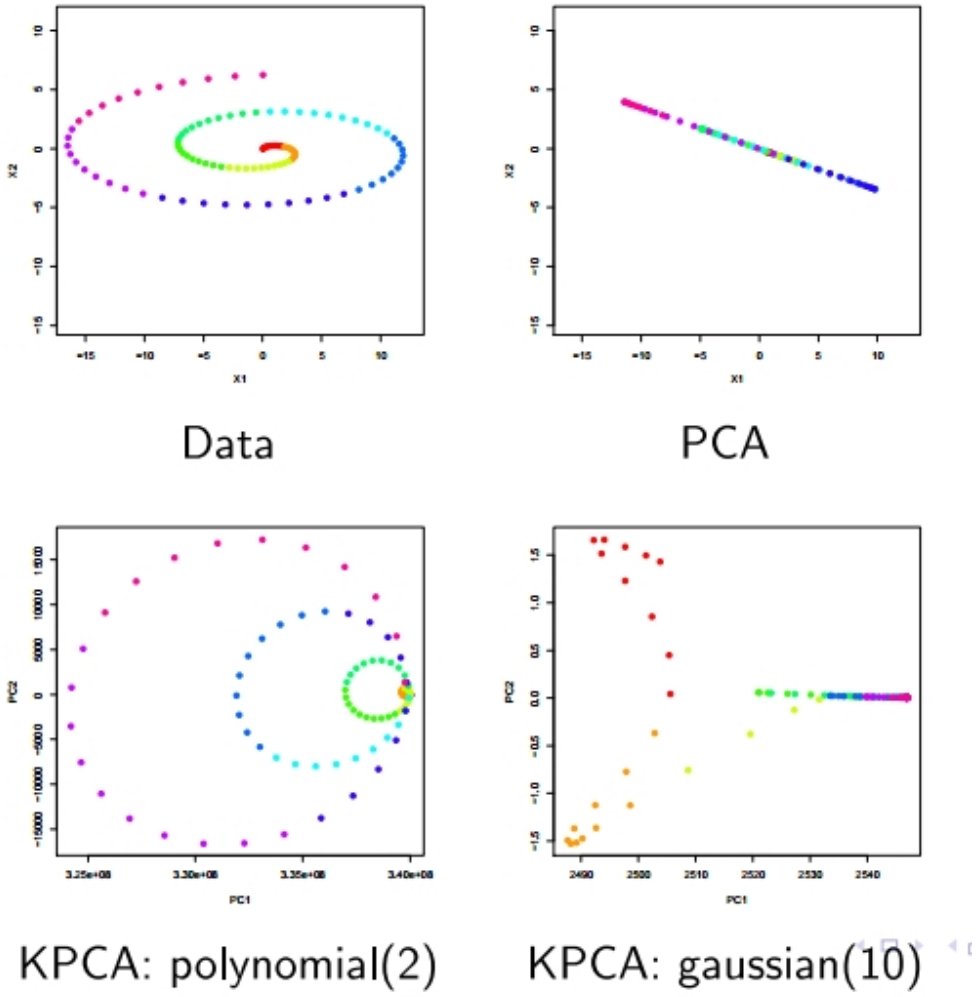

Figure 3: Results from PCA and KPCA method

2.7 Case 1

We are given $\mathcal{D}_{train}$ and $X_1^*, \ldots, X_{n^*}^*$ to build $\hat{f}$

Then we can just use straight forward KPCA

Or any unsupervised learning step

1. Form $\mathbb{K}$ on $\mathcal{D}_{train}$ and $X_1^*, \ldots, X_{n^*}^*$
2. Get UD
3. Pass $Z_q = UD[, 1:q]$ to train $\hat{f}$
4. Get $\hat{Y} = \hat{f}(Z_q) \in \mathbb{R}^{n+n^*}$

2.8 Case 2

We are given only $\mathcal{D}_{train}$ to build $\hat{f}$

Now, we don't know the coordinates of $X_1^*, \ldots, X_{n^*}^*$ in the representation space

To get a new observation X^* embedded into this representation:

$$Z_0 = D^{-1}U^\top (I - M)[k^* - K\mathbf{1}/n]$$

where $k^* = [k(X^*, X_1), \ldots, k(X^*, X_n)]^\top$

Then we compute:

1. Form $\mathbb{K}$ on $\mathcal{D}_{train}$
2. Get UD
3. Pass $Z_q = UD[, 1:q]$ to train $\hat{f}$
4. Form Z_q^* for all $X_1^*, \ldots, X_{n^*}^*$
5. Get $\hat{Y}_{test} = \hat{f}(Z_q^*) \in \mathbb{R}^{n^*}$

(Note:$\mathbb{K} = UD^2U^T = UD^2U^T$)

2.9 KPCA: summary

Kernel PCA seeks to generalize the notion of similarity using a kernel map

This can be interpreted as finding smooth, orthogonal directions in a RKHS

This can allow us to start picking up nonlinear (in the original feature space) aspects of our data

This new representation can be passed to a supervised method to form a semisupervised learner

Note: From the paper introduced by Scholkopf et al. [1], they mentioned the advantages of nonlinear KCPA compare to linear PCA: the performance for nonlinear components can be further improved by using more components than possible in the linear case. Also, the computational complexity of kernel PCA does not grow with the dimensionality of the feature space that we are implicitely working in.

References

[1] Scholkopf, B., Smola, A., and Mller, K.-R. (1999). Kernel principal component analysis. In *ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING*, pages 327–352. MIT Press.