Instructor: Sael Lee

CS549 Spring – Computational Biology

LECTURE 17: KERNEL PCA

KERNEL-BASED FEATURE EXTRACTION

- PCA can only extract a linear projection of the data
  - To do so, we first compute the covariance matrix

$$S = \frac{1}{N}\sum_{n=1}^{N} \mathrm{x}_n \mathrm{x}_n^T$$

  - Then, we find the eigenvectors and eigenvalues

$$Su_i = \lambda_i u_i \text{ and } u_i^T u_i = 1$$
$$SU = \lambda U$$

  - And, finally, we project onto the eigenvectors with largest eigenvalues

$$y = \mathrm{Ux}$$

- Can the kernel trick be used to perform this operation implicitly in a higher-dimensional space?
  - If so, this would be equivalent to performing non-linear PCA in the feature space

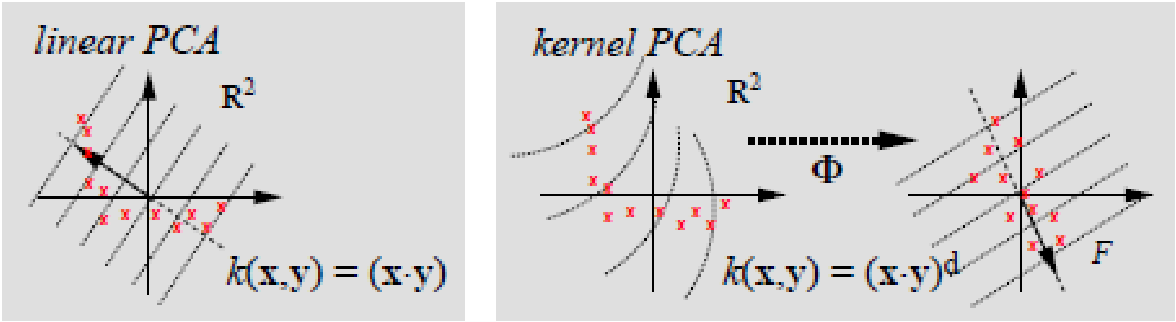


Fig. 1. Basic idea of kernel PCA: by using a nonlinear kernel function k instead of the standard dot product, we implicitly perform PCA in a possibly high–dimensional space F which is nonlinearly related to input space. The dotted lines are contour lines of constant feature value.

Scholkopf, B., Smola, A., Muller, K. R., & Kybernetik, M. (n.d.). Kernel Principal Component Analysis, 2–7.

DERIVING KERNEL-PCA

* Assume zero mean data (centralized data points)

1. Project the data into the high-dim feature space M

$$\phi: R^D \rightarrow R^M; \mathrm{x} \rightarrow \phi(\mathrm{x})$$

2. Compute the covariance matrix

    * Assume that projected data has zero mean (we will deal with it later)

$$C = \frac{1}{N}\sum_{n=1}^{N} \phi(\mathrm{x}_n)\phi(\mathrm{x}_n)^T$$

3. Compute the principal components by solving the eigenvalue problem

$$Cv_i = \lambda_i v_i \qquad \text{where } i = 1 \ldots M$$

$$\text{or } Cv = \lambda v$$

- **The challenge is… how do we do this implicitly?**

Schölkopf et al., (Neural Computation, 1998)

EXPRESSION INTO KERNEL FUNCTION

$$C = \frac{1}{N}\sum_{n=1}^{N}\phi(\mathrm{x}_n)\phi(\mathrm{x}_n)^T$$

$$Cv_i = \lambda_i v_i \qquad \text{where } i = 1 \dots M$$

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathrm{x}_n)\{\phi(\mathrm{x}_n)^T v_i\} = \lambda_i v_i$$

Dual variable notation

$$v_i = \sum_{n=1}^{N} a_{in}\phi(\mathrm{x}_n)$$

Observer that provided $\lambda_i > 0, v_i$ is given by linear combination of $\phi(\mathrm{x}_n)$

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathrm{x}_n)\{\phi(\mathrm{x}_n)^T v_i\} = \lambda_i v_i$$

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathrm{x}_n)\{\phi(\mathrm{x}_n)^T \sum_{m=1}^{N} a_{im}\phi(\mathrm{x}_m)\} = \lambda_i \sum_{n=1}^{N} a_{in}\phi(\mathrm{x}_n)$$

Multiply by $\phi(\mathrm{x}_l)^T$

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathrm{x}_l)^T\phi(\mathrm{x}_n)\{\sum_{m=1}^{N} a_{im}\phi(\mathrm{x}_n)^T\phi(\mathrm{x}_m)\} = \lambda_i \sum_{n=1}^{N} a_{in}\phi(\mathrm{x}_l)^T\phi(\mathrm{x}_n)$$

$$\boxed{\frac{1}{N}\sum_{n=1}^{N}\mathrm{k}(\mathrm{x}_l,\mathrm{x}_n)\,\{\sum_{m=1}^{N} a_{im}\mathrm{k}(\mathrm{x}_n,\mathrm{x}_m)\} = \lambda_i \sum_{n=1}^{N} a_{in}\,\mathrm{k}(\mathrm{x}_l,\mathrm{x}_n)}$$

$$\frac{1}{N}\sum_{n=1}^{N} \mathrm{k}(\mathrm{x}_l, \mathrm{x}_n)\left\{\sum_{m=1}^{N} a_{im}\mathrm{k}(\mathrm{x}_n, \mathrm{x}_m)\right\} = \lambda_i \sum_{n=1}^{N} a_{in}\,\mathrm{k}(\mathrm{x}_l, \mathrm{x}_n)$$

$$\mathbf{K}^2\boldsymbol{a}_i = \lambda_i N\mathbf{K}\boldsymbol{a}_i \xrightarrow{\text{Remove K from each side}} \mathbf{K}\boldsymbol{a}_i = \lambda_i N\boldsymbol{a}_i$$

* Right differs only by eigenvector of K having 0 eigenvalues and do not effect other PC projections

Normalization condition for the coefficients $\boldsymbol{a}_i$ is obtained by requiring the eigenvector in feature space be normalized.

$$\left.\begin{aligned} v_i &= \sum_{n=1}^{N} a_{in}\phi(\mathrm{x}_n) \\ \mathbf{K}\boldsymbol{a}_i &= \lambda_i N\boldsymbol{a}_i \end{aligned}\right\}$$

$$1 = v_i^T v_i = \sum_{n=1}^{N}\sum_{m=1}^{N} a_{in}a_{im}\phi(\mathrm{x}_n)^T\,\phi(\mathrm{x}_m)$$

$$= \boldsymbol{a}_i^{\mathbf{T}}\mathbf{K}\boldsymbol{a}_i = \lambda_i N\boldsymbol{a}_i^{\mathbf{T}}\boldsymbol{a}_i$$

PROJECTION USING KERNEL FUNCTION

Having solved the eigenvector problem, the resulting principal component projections can then also e cast in terms of the kernel function

$$\boldsymbol{a}_i^{\mathbf{T}}\mathbf{K}\boldsymbol{a}_i = \lambda_i N \boldsymbol{a}_i^{\mathbf{T}}\boldsymbol{a}_i \qquad v_i = \sum_{n=1}^{N} a_{in}\phi(\mathrm{x}_n)$$

$$y_i(\mathbf{x}) = \phi(\mathbf{x})^T \boldsymbol{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x})^T\phi(\mathbf{x}_{\boldsymbol{n}}) = \boxed{\sum_{n=1}^{N} a_{in}\mathrm{k}(\mathbf{x}, \mathbf{x}_{\boldsymbol{n}})}$$

ZERO MEAN PROJECTION DATA REVISITED

We assumed that $\sum_{n=1}^{N} \phi(\mathrm{x}_n) = 0$ which is in most cannot be controlled

Need to adjust for the zero mean assumption as follows:

zero mean projection data

$$\tilde{\phi}(\mathbf{x}_n) = \phi(\mathrm{x}_n) - \frac{1}{\mathrm{N}} \sum_{l=1}^{N} \phi(\mathrm{x}_l)$$

1. Evaluate $\widetilde{\mathrm{K}}$ to using kernel function K and

$$\widetilde{K}_{nm} = \tilde{\phi}(\mathbf{x}_n)^T \tilde{\phi}(\mathbf{x}_m)$$

$$\widetilde{\mathrm{K}} = \mathrm{K} - 1_{\mathrm{N}}\mathrm{K} - \mathrm{K}1_{\mathrm{N}} + 1_{\mathrm{N}}\mathrm{K}1_{\mathrm{N}}$$

Where 1_{N} is NxN matrix where every elements is 1/N

2. use $\widetilde{\mathrm{K}}$ to find the eigenvalues and eigenvector.

$$\boldsymbol{a}_i^{\mathbf{T}} \widetilde{\mathbf{K}} \boldsymbol{a}_i = \lambda_i N \boldsymbol{a}_i^{\mathbf{T}} \boldsymbol{a}_i$$

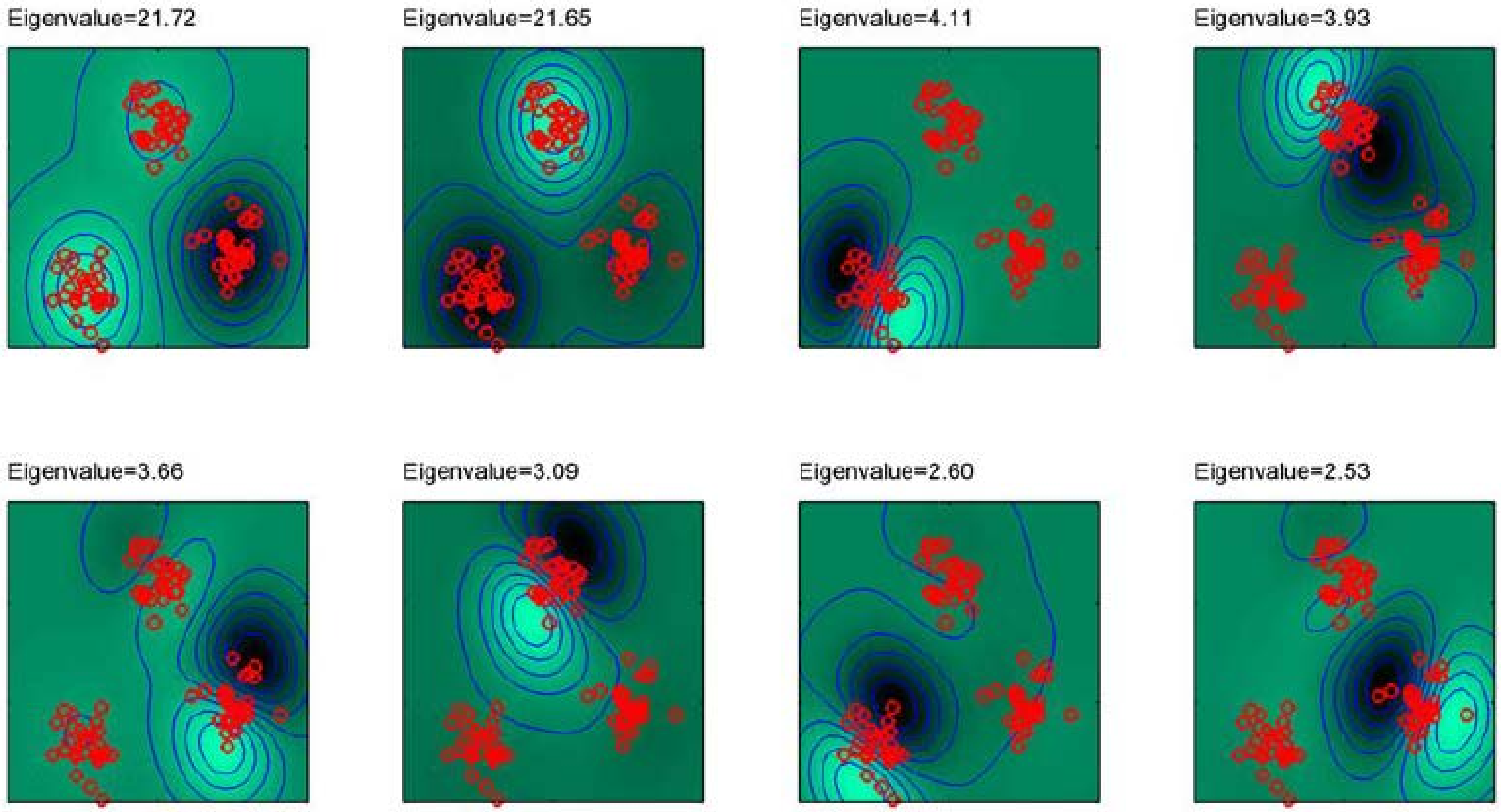
$$y_i(\mathbf{x}) = \sum_{n=1}^{N} a_{in} \tilde{\mathrm{k}}\,(\mathbf{x}, \mathbf{x}_{\boldsymbol{n}})$$

EXAMPLE : GAUSSIAN KERNELS

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 0.1)$$

Lines: projection onto the corresponding principal component,

$$\phi(\mathbf{x})^{\mathrm{T}} \mathbf{v}_i = \sum_{n=1}^{N} a_{in} k(\mathbf{x}, \mathbf{x}_n)$$



DIFFERENCES AND SHORTCOMES OF KERNEL PCA

- Kernel PCA involves finding the eigenvectors of the $N \times N$ matrix $\widetilde{\boldsymbol{K}}$ rather than the $D \times D$ matrix $\mathbf{S}$ of conventional linear PCA, and so in practice for large data sets approximations are often used
- In standard linear PCA, we often retain some reduced number $L < D$ of eigenvectors and then approximate a data vector $\boldsymbol{x}_n$ by its projection $\hat{\boldsymbol{x}}_n$ onto the L-dimensional principal subspace

$$\widehat{\mathbf{x}}_n = \sum_{i=1}^{L} \left(\mathbf{x}_n^{\mathrm{T}} \mathbf{u}_i\right) \mathbf{u}_i .$$

kernel PCA, this will in general not be possible