L28: kernel-based feature extraction

Kernel PCA

Kernel LDA

Principal Components Analysis

As we saw in L9, PCA can only extract a linear projection of the data

- To do so, we first compute the covariance matrix

$$C = \frac{1}{M}\sum_{j=1}^{M} x_j x_j^T$$

- Then, we find the eigenvectors and eigenvalues

$$Cv = \lambda v$$

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

$$y = [v_1 v_2 \dots v_D]x$$

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

Kernel PCA

To derive kernel-PCA

- We would first project the data into the high-dim feature space F

$$\Phi: R^N \rightarrow F; x \rightarrow X$$

- Then we would compute the covariance matrix

$$C_F = \frac{1}{M} \sum_{j=1}^{M} \varphi(x_j) \varphi(x_j)^T$$

  - where we have assumed that the data in F is centered $E[\varphi(x)] = 0$ (more on this later)

- Then we would compute the principal components by solving the eigenvalue problem

$$C_F v = \lambda v$$

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

Schölkopf et al., (Neural Computation, 1998)

Solution

– As we saw in the snapshot PCA lecture, the eigenvectors can be expressed as linear combinations of the training data

$$C_F V = \left(\frac{1}{M}\sum_{i=1}^{M}\varphi(x_i)\varphi(x_i)^T\right)V = \lambda V \Rightarrow$$

$$V = \left(\frac{1}{M\lambda}\sum_{i=1}^{M}\varphi(x_i)\varphi(x_i)^T\right)V = \sum_{i=1}^{M}\frac{\left(\varphi(x_i)^T V\right)}{M\lambda}\varphi(x_i) = \sum_{i=1}^{M}\alpha_i\varphi(x_i)$$

– We then multiply by $\varphi(x_k)$ both sides of $\lambda V = C_F V$

$$\lambda[\varphi(x_k)V] = [\varphi(x_k)C_F V]$$

– which, combining with the previous expression

$$\lambda\left[\varphi(x_k)\sum_{i=1}^{M}\alpha_i\varphi(x_i)\right] = \varphi(x_k)\left[\frac{1}{M}\sum_{j=1}^{M}\varphi(x_j)\varphi(x_j)^T\right]\left[\sum_{i=1}^{M}\alpha_i\varphi(x_i)\right]$$

– and regrouping terms, yields

$$\lambda\sum_{i=1}^{M}\alpha_i\varphi(x_k)\varphi(x_i) = \frac{1}{M}\sum_{i=1}^{M}\alpha_i\left(\varphi(x_k)\sum_{j=1}^{M}\varphi(x_j)\right)\left(\varphi(x_j)\varphi(x_i)\right)$$

– Defining an $M \times M$ matrix K as

$$K_{ij} := \left(\varphi(x_i) \cdot \varphi(x_j)\right)$$

– the previous expression becomes

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

– which can be solved through the eigenvalue problem

$$M\lambda\alpha = K\alpha$$

Normalization

– To ensure that eigenvectors V are orthonormal, we then scale eigenvectors α

$$\left(V^k \cdot V^k\right) = 1 \Rightarrow \left(\sum_{i=1}^{M} \alpha_i^k \varphi(x_i)\right)\left(\sum_{j=1}^{M} \alpha_j^k \varphi(x_j)\right) = 1$$

$$\sum_{i,j=1}^{M} \alpha_i^k \alpha_j^k \varphi(x_i)\varphi(x_j) = 1 \Rightarrow \sum_{i,j=1}^{M} \alpha_i^k \alpha_j^k K_{ij} = 1 \Rightarrow \left(\alpha^k K \alpha^k\right) = 1$$

– which, since α are the eigenvectors of K, yields

$$\lambda_k\left(\alpha^k \alpha^k\right) = 1$$

To find the k-th principal component of a new sample x

$$\left(V^k \cdot \varphi(x)\right) = \left(\sum_{i=1}^{M} \alpha_i^k \varphi(x_i)\right) \cdot \varphi(x) = \sum_{i=1}^{M} \alpha_i^k K(x_i, x)$$

- Note that, when the kernel function is the dot-product, the kernel PCA solution reduces to the snapshot PCA solution
- However, unlike in snapshot PCA, here will be unable to find the eigenvectors since they reside in the high dimensional space F

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

- This implies that kernel PCA can be used for feature extraction but CANNOT be used (at least directly) for reconstruction purposes

Centering in the high-dimensional space

Earlier we assumed that the data was centered in F

$$\tilde{\varphi}(x_i) \coloneqq \varphi(x_i) - \frac{1}{M}\sum_{i=1}^{M}\varphi(x_i)$$

- So the covariance matrix in this centered space is

$$\tilde{K}_{ij} = \left(\tilde{\varphi}(x_i) \cdot \tilde{\varphi}(x_j)\right)$$

- And the eigenvalue problem that we need to solve is

$$\tilde{\lambda}\tilde{\alpha} = \tilde{K}\tilde{\alpha}$$

- Merging the first expression into the second one

$$\tilde{K}_{ij} = \left[\left(\varphi(x_i) - \frac{1}{M}\sum_{m=1}^{M}\varphi(x_m)\right)\left(\varphi(x_j) - \frac{1}{M}\sum_{n=1}^{M}\varphi(x_n)\right)\right] =$$

$$K_{ij} - \frac{1}{M}\sum_{m=1}^{M}1_{im}K_{mj} - \frac{1}{M}\sum_{n=1}^{M}1_{in}K_{nj} + \frac{1}{M^2}\sum_{m=1}^{M}1_{im}K_{mn}1_{nj} =$$

$$[K - 1_M K - K1_M + 1_M K 1_M]_{ij}$$

  - where $1_{ij} = 1$ (for all i,j), $(1_M)_{ij} := 1/M$

- So the centered kernel matrix can be computed from the uncentered one

To project new test data $t_1, t_2, \ldots, t_L$

- First, we define two matrices

$$K_{ij}^{test} = \left(\varphi(t_i) \cdot \varphi(x_j)\right)$$

$$\tilde{K}_{ij}^{test} = \left(\left(\varphi(t_i) - \frac{1}{M}\sum_{m=1}^{M} \varphi(x_m)\right) \cdot \left(\varphi(x_j) - \frac{1}{M}\sum_{n=1}^{M} \varphi(x_n)\right)\right)$$

- Then, we express $\tilde{K}^{test}$ in terms of K^{test}

$$\tilde{K}^{test} = K^{test} - 1'_M K - K^{test} 1_M + 1'_M K 1_M$$

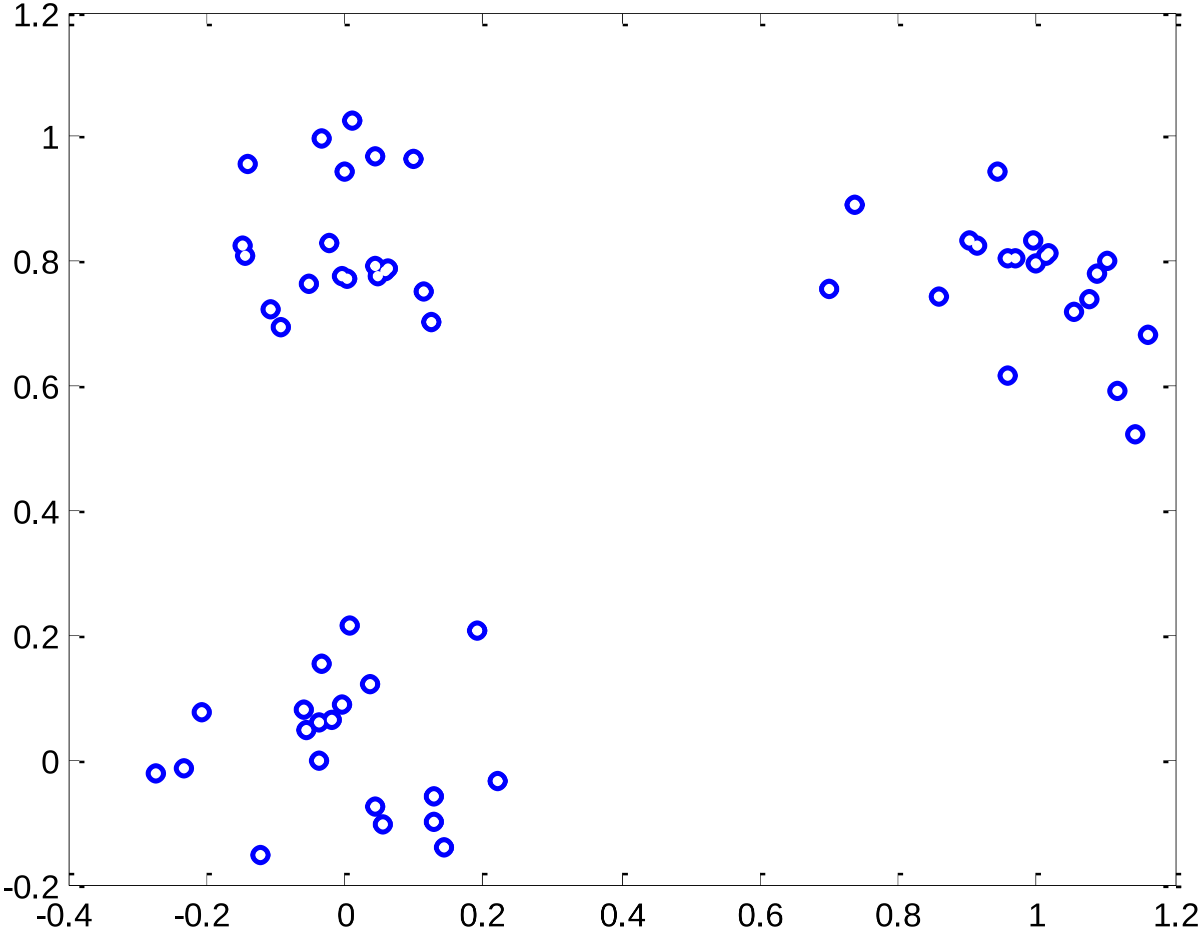
  - where $1'_M$ is an L×M matrix with all entries equal to 1/M

- From here, we can then find the principal components of test data as

$$\left(\tilde{V}^k \tilde{\varphi}(t)\right) = \left(\sum_{i=1}^{M} \tilde{\alpha}_i^k \tilde{\varphi}(x_i)\right) \tilde{\varphi}(t) = \sum_{i=1}^{M} \tilde{\alpha}_i^k \tilde{K}(x_i, t)$$

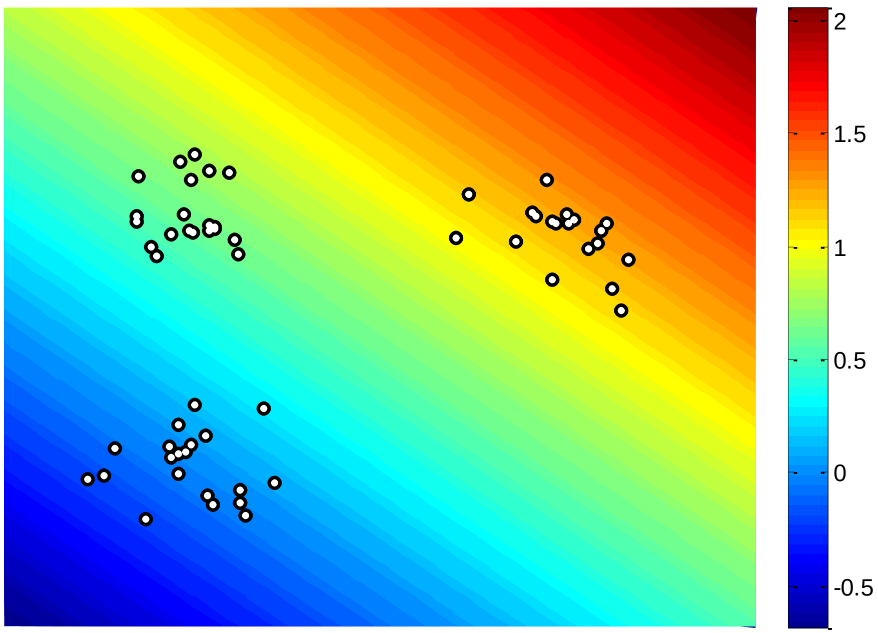
Kernel PCA example

Simple dataset with three modes, 20 samples per mode

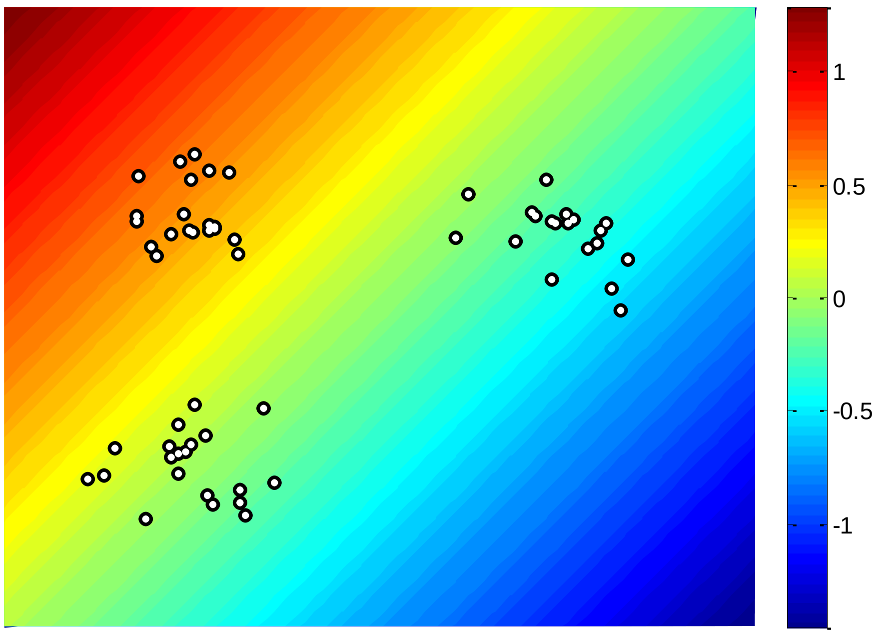


The (linear) PCA solution

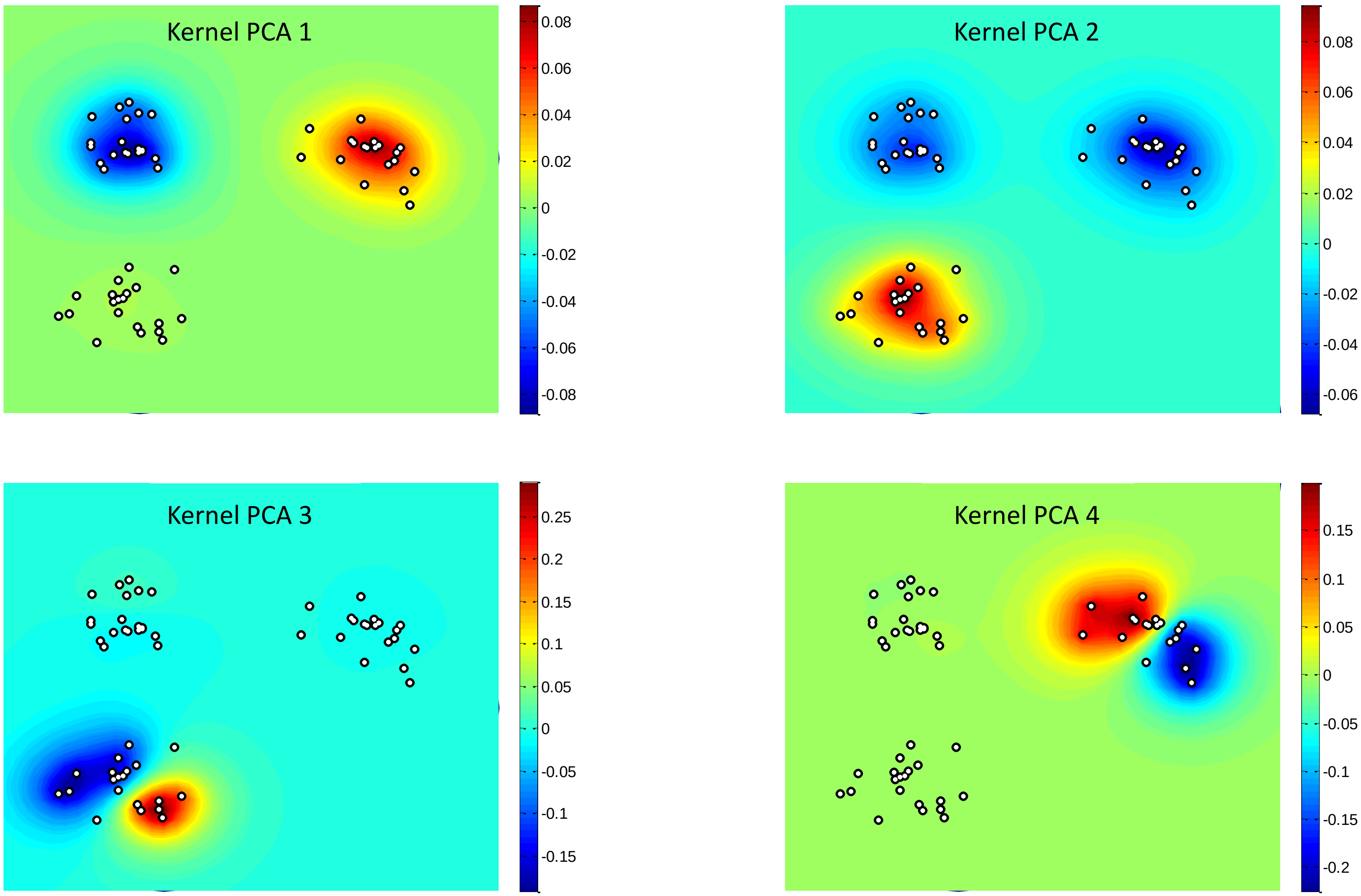
PCA 1



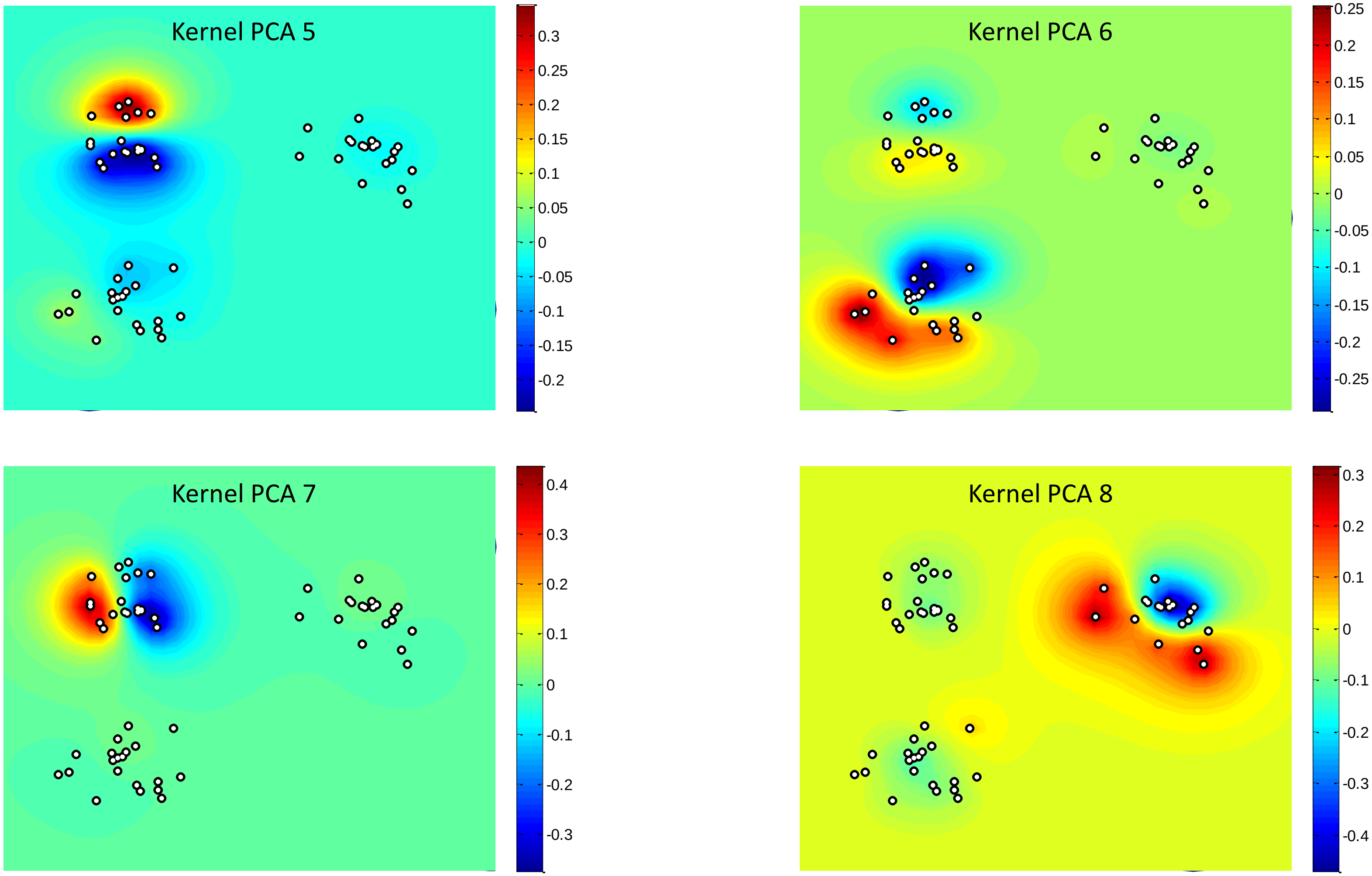
PCA 2



The kernel PCA solution (Gaussian Kernel)



More kernel PCA projections (out of 60)



Kernel LDA

Assume a two-class discrimination problem, with N_1 and N_2 examples from classes ω_1 and ω_2, respectively

- From L10, and under the homoscedatic Gaussian assumption, the optimum projection v is obtained by maximizing the Rayleigh quotient

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

- where

$$S_W = \sum_{i=1}^{2} \sum_{x \in \omega_i} (x - m_i)(x - m_i)^T$$

$$S_B = (m_2 - m_1)(m_2 - m_1)^T$$

$$m_i = \frac{1}{N_i} \sum_{j=1}^{N_i} x_j^i$$

[Mika et al., 1999]

Can we solve this problem (implicitly) in a high-D kernel space F to yield a non-linear version of the Fisher's LDA?

- To do so, we would define between-class and within-class covariance matrices in kernel space F to obtain the following quotient

$$J(v) = \frac{v^T S_B^\Phi v}{v^T S_W^\Phi v}$$

- where now $V \in F$, and mean and covariance are defined in F as

$$S_W^\Phi = \sum_{i=1}^{2} \sum_{x \in \omega_i} \left(\varphi(x) - m_i^\Phi\right)\left(\varphi(x) - m_i^\Phi\right)^T$$

$$S_B^\Phi = \left(m_2^\Phi - m_1^\Phi\right)\left(m_2^\Phi - m_1^\Phi\right)^T$$

$$m_i^\Phi = \frac{1}{N_i} \sum_{j=1}^{N_i} \varphi(x_j^i)$$

– As earlier, we make use of the fact that the eigenvector V can be expressed as linear combinations of the training data

$$V = \sum_{j=1}^{N} \alpha_j \varphi(x_j)$$

– which, when multiplied by m_i^Φ, yields

$$V^T m_i^\Phi = \left(\sum_{j=1}^{N} \alpha_j \varphi(x_j)\right)^T \left(\frac{1}{N_i} \sum_{k=1}^{N_i} \varphi(x_k^i)\right) =$$

$$\frac{1}{N_i} \sum_{j=1}^{N} \sum_{k=1}^{N_i} \alpha_j K(x_j, x_k^i) = \alpha^T M_i$$

- where we have defined

$$(M_i)_j := \frac{1}{N_i} \sum_{k=1}^{N_i} K(x_j, x_k^i)$$

– Merging this result with the definition of S_B^Φ yields the following expression for the numerator

$$V^T S_B^\Phi V = \alpha^T M \alpha$$

- where

$$M = (M_1 - M_2)(M_1 - M_2)^T$$

– Likewise, merging with the definition of S_W^Φ yields

$$V^T S_W^\Phi V = \alpha^T N \alpha$$

– where

$$N := \sum_{j=1}^{2} K_j \left(1 - 1_{N_j}\right) K_j^T$$

– where I is a $N_j \times N_j$ identity matrix, 1_{N_j} is a $N_j \times N_j$ matrix with all entries equal to $1/N_j$, and K_j is a $N \times N_j$ matrix such that

$$\left(K_j\right)_{nm} := K\left(x_n, x_m^j\right)$$

– Combining these results, we obtain a new expression for the Rayleigh quotient

$$J(\alpha) = \frac{\alpha^T M \alpha}{\alpha^T N \alpha}$$

- which can be solved by finding the leading eigenvector of $N^{-1}M$

– And the projection of a new pattern t is given by

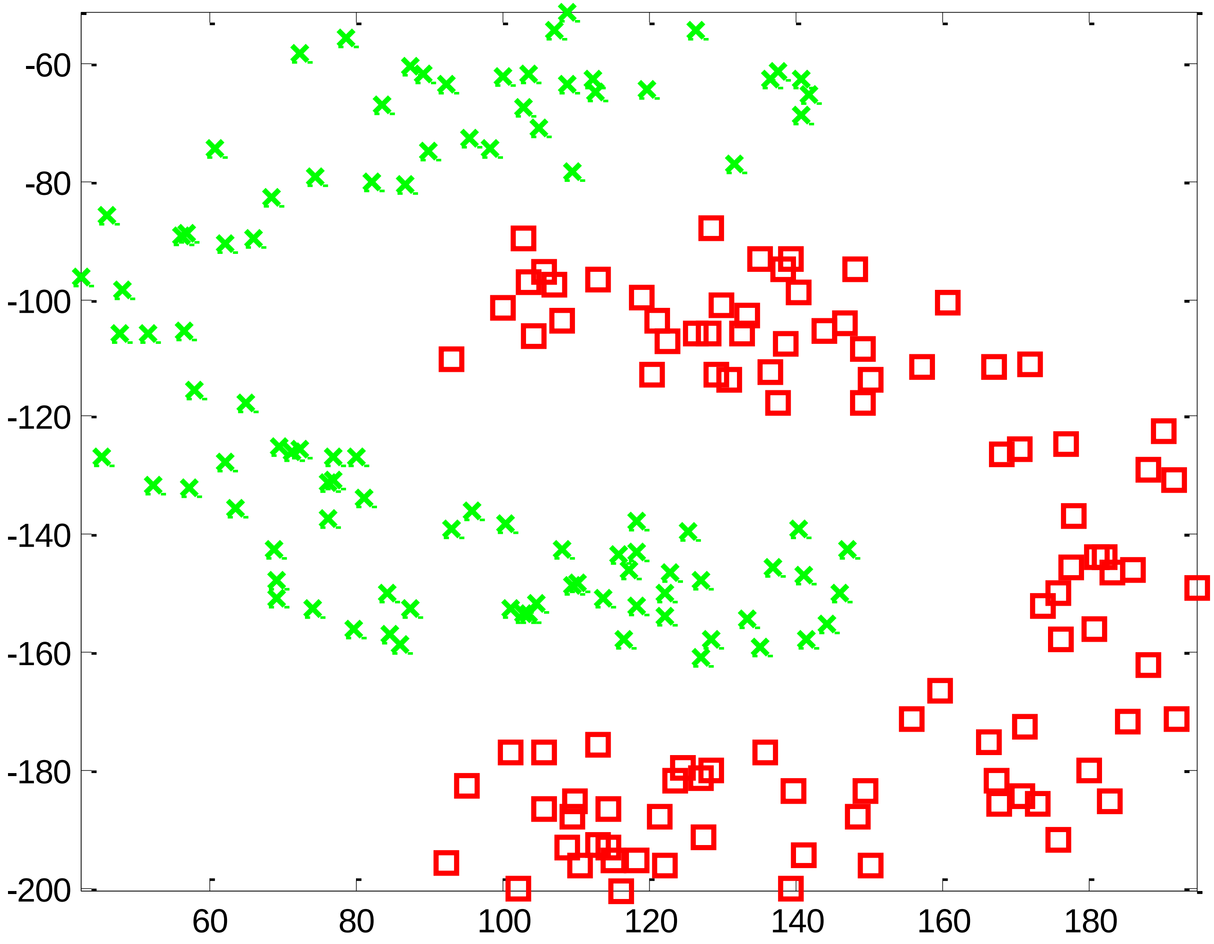
$$\left(V \cdot \varphi(t)\right) = \sum_{i=1}^{N} \alpha_i K(x_i, t)$$

Regularization

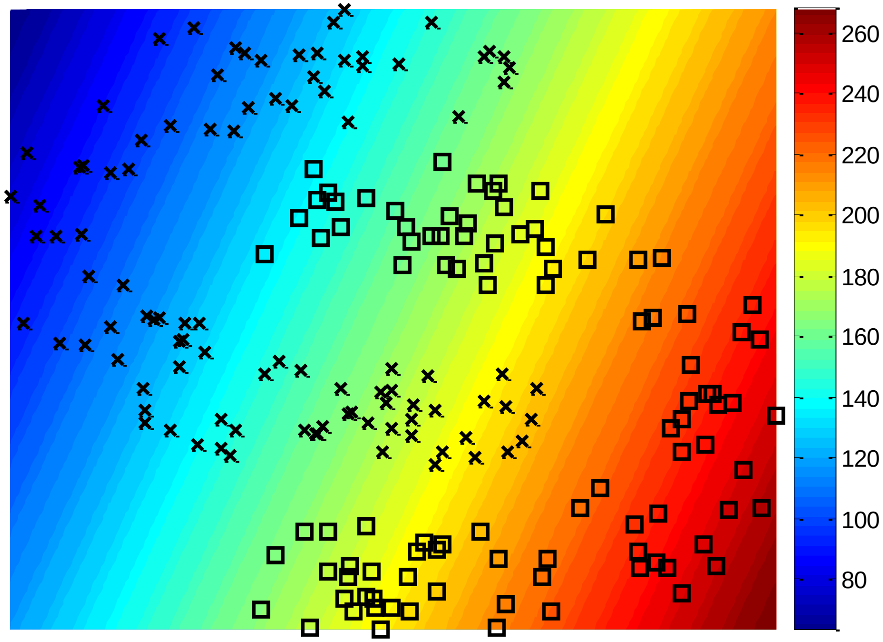
– To avoid numerical ill-conditioning, one may regularize matrix N by adding a multiple of the identity matrix

$$N = N + \mu \mathrm{I}$$

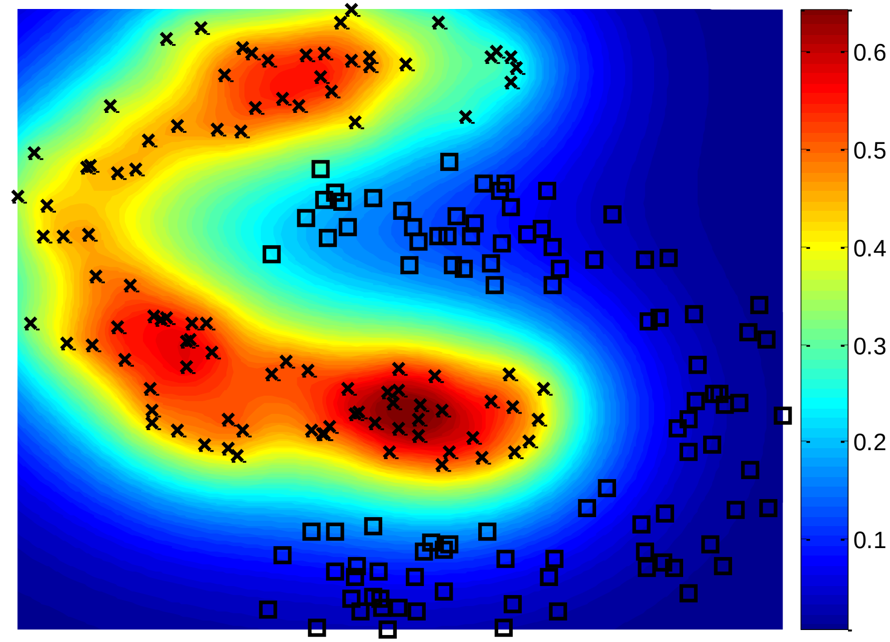
Kernel LDA examples

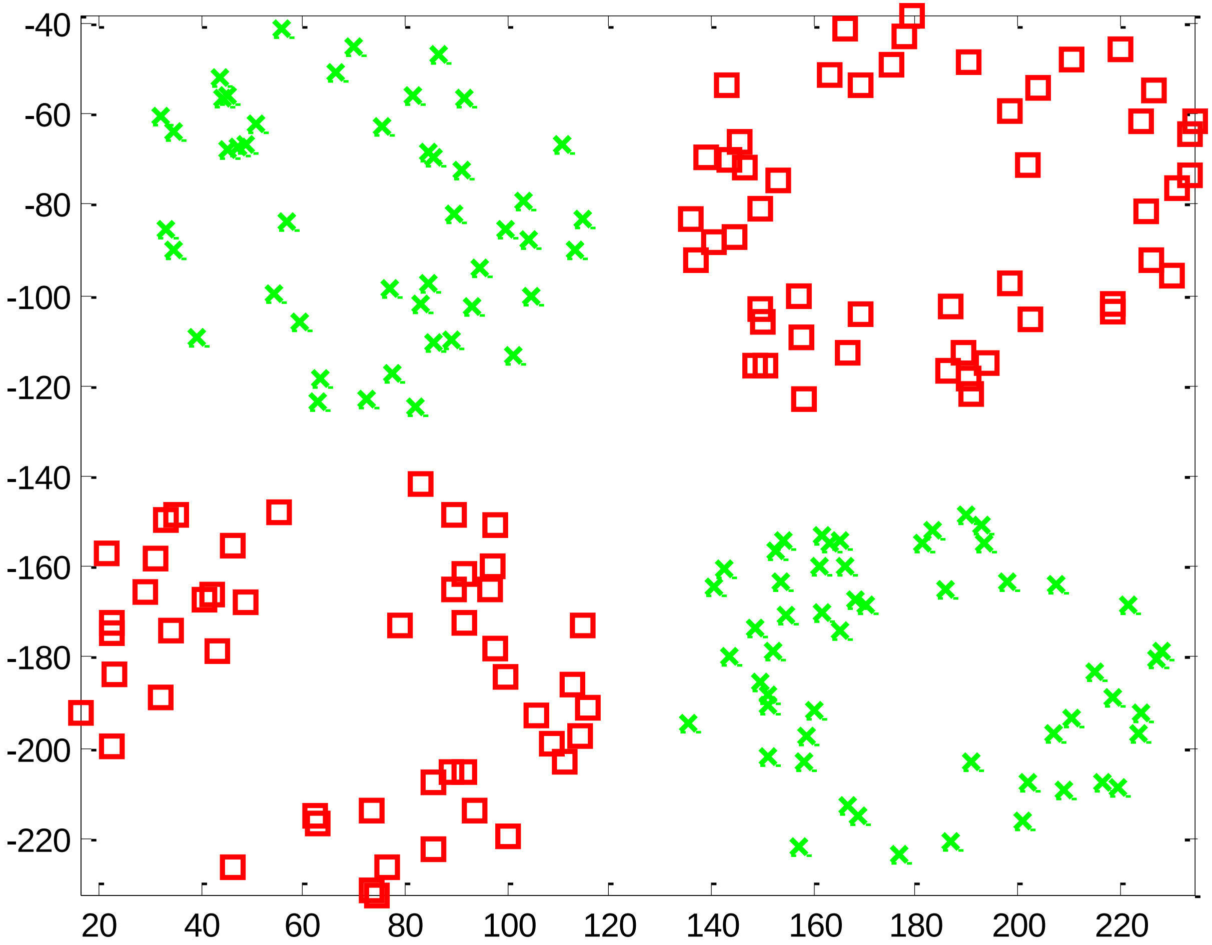


LDA

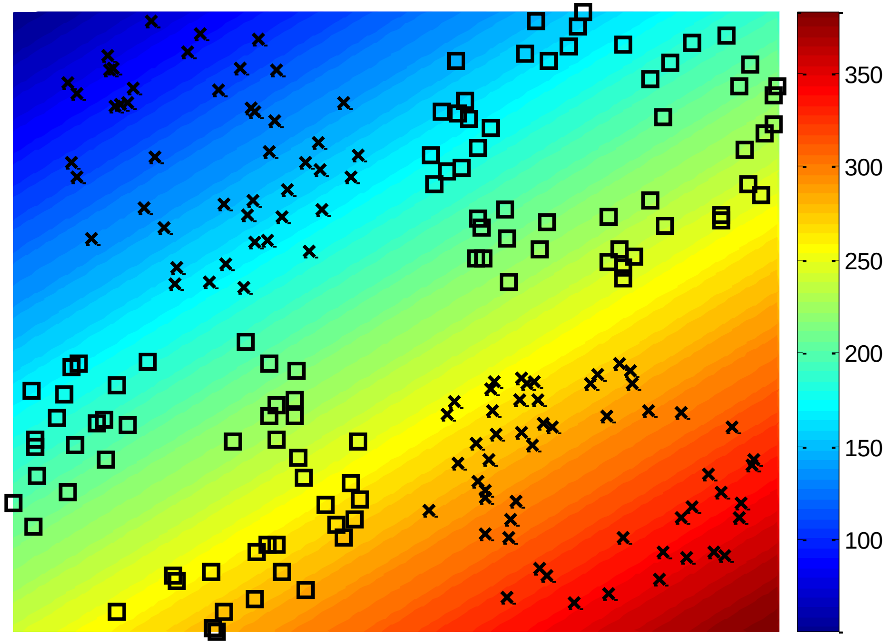


Kernel LDA




-40
-60
-80
-100
-120
-140
-160
-180
-200
-220
20
40
60
80
100
120
140
160
180
200
220

LDA



Kernel LDA