Principal Component Analysis and Kernal PCA

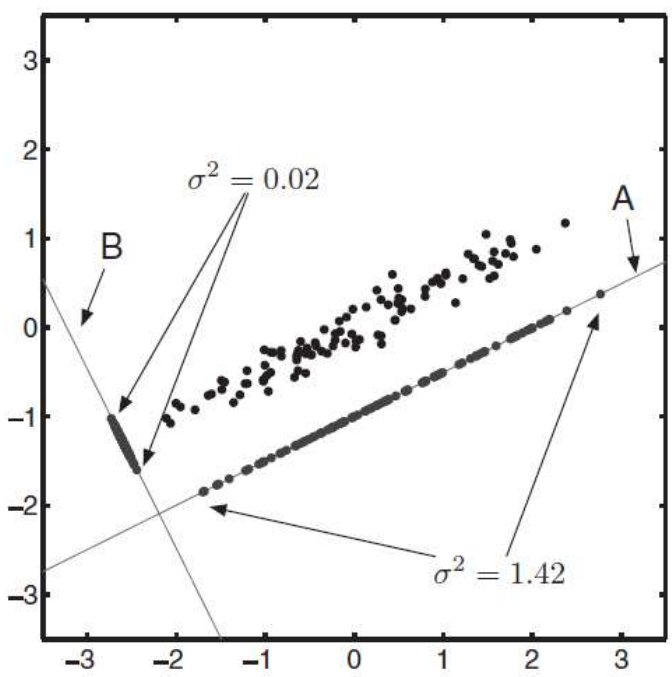
生醫光電所 吳育德

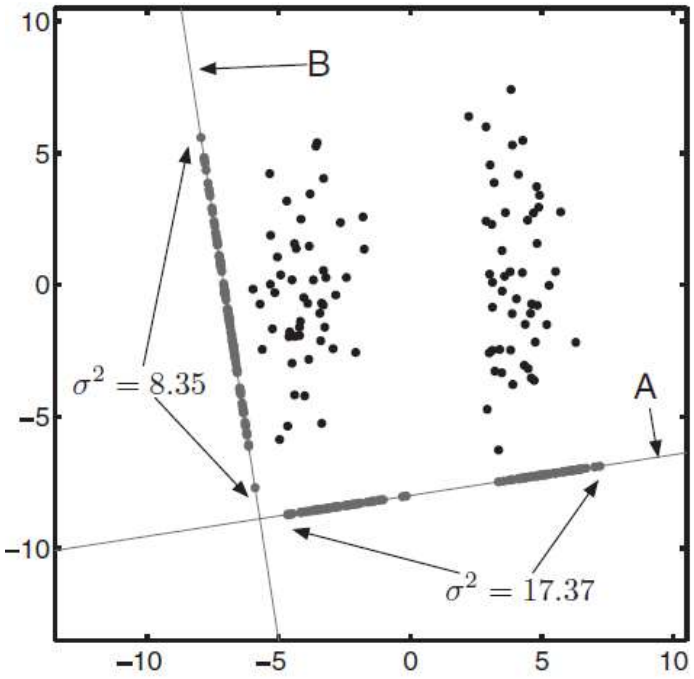
Chap7. A First Course in Machine Learning, 2ed, Simon Rogers and Mark Girolami, 2017
CSC421/2516 Lecture 12, Roger Grosse and Jimmy Ba, Univ. of Toronto

The General Problem

- We are projecting *M*-dimensional data into *D*-dimensions in a manner that will hopefully preserve the properties of interest.

- If cluster structure is present in the data, using the projection with the highest variance is likely to preserve this structure.





Remove the Mean from Data

- Given a set of N samples, $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N$, each is M-dimensional.

$$\underbrace{\mathbf{Y}}_{N\times M} = \begin{bmatrix} \boldsymbol{y}_1^T \\ \vdots \\ \boldsymbol{y}_N^T \end{bmatrix} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,M} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N,1} & y_{N,2} & \cdots & y_{N,M} \end{bmatrix},$$

- We first transform $\boldsymbol{y}_i, i = 1, \cdots, N$ to have zero mean by $\boldsymbol{y}_i - \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{y}_n$,

$$\text{i.e., } \mathbf{Y} \equiv \begin{bmatrix} \tilde{\boldsymbol{y}}_1^T \\ \vdots \\ \tilde{\boldsymbol{y}}_N^T \end{bmatrix} = \begin{bmatrix} (\boldsymbol{y}_1 - \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{y}_n)^T \\ \vdots \\ (\boldsymbol{y}_N - \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{y}_n)^T \end{bmatrix} = \begin{bmatrix} \tilde{y}_{1,1} & \tilde{y}_{1,2} & \cdots & \tilde{y}_{1,M} \\ \tilde{y}_{2,1} & \tilde{y}_{2,2} & \cdots & \tilde{y}_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{y}_{N,1} & \tilde{y}_{N,2} & \cdots & \tilde{y}_{N,M} \end{bmatrix}$$

i.e., The expectation of the random variable $\boldsymbol{y}$ is

$$E[\boldsymbol{y}] = \frac{1}{N}\sum_{i=1}^{N} \tilde{\boldsymbol{y}}_i = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{y}_i - \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{y}_n = 0$$

Find the Best Projection Vector w

- Let's start to find a unit $\underbrace{\boldsymbol{w}}_{M\times 1}$ vector and project $\underbrace{\boldsymbol{y}_n}_{M\times 1}$ into 1-dim $x_n = \boldsymbol{w}^T\boldsymbol{y}_n$.
- Note $\bar{\boldsymbol{y}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{y}_n = \boldsymbol{0} \Rightarrow \bar{x} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{w}^T\boldsymbol{y}_n = \boldsymbol{w}^T\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{y}_n = 0$
- The variance

$$\sigma_x^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})^2 = \frac{1}{N}\sum_{n=1}^{N}x_n^2 = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{w}^T\boldsymbol{y}_n\boldsymbol{y}_n^T\boldsymbol{w}$$

$$= \underbrace{\boldsymbol{w}^T}_{1\times M}\left(\frac{1}{N}\sum_{n=1}^{N}\underbrace{\boldsymbol{y}_n}_{M\times 1}\underbrace{\boldsymbol{y}_n^T}_{1\times M}\right)\underbrace{\boldsymbol{w}}_{M\times 1} = \boldsymbol{w}^T\underbrace{\mathbf{C}}_{M\times M}\boldsymbol{w},$$

$$\text{where } \mathbf{C} \equiv \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{y}_n - \bar{\boldsymbol{y}})(\boldsymbol{y}_n - \bar{\boldsymbol{y}})^T = \frac{1}{N}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$$

Find the Best Projection Vector w

- Let $\boldsymbol{w}$ denote a unit vector also of dimension M, onto which the vector $\boldsymbol{y}$ is to be projected, $\boldsymbol{w}^T\boldsymbol{w} = 1$.
- This projection is defined by the inner product of the vectors $\boldsymbol{y}$ and $\boldsymbol{w}$

$$x = \boldsymbol{y}^T\boldsymbol{w} = \boldsymbol{w}^T\boldsymbol{y}$$

- Note $E[x] = E[\boldsymbol{w}^T\boldsymbol{y}] = \boldsymbol{w}^T E[\boldsymbol{y}] = 0$
- The variance of x is

$$E[(x - E[x])^2] = E[x \cdot x] = E[\boldsymbol{w}^T\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{w}] = \boldsymbol{w}^T E[\boldsymbol{y}\boldsymbol{y}^T]\boldsymbol{w}^T$$

- We need to compute the covariance matrix $E[\boldsymbol{y}\boldsymbol{y}^T]$ using the sample vectors

$$E[\boldsymbol{y}\boldsymbol{y}^T] \equiv \underbrace{\mathbf{C}}_{M\times M} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{y}_n\boldsymbol{y}_n^T$$

$$\text{or} \quad \mathbf{C} = \frac{1}{N}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$$

Find the Best Projection Vector w

- Maximize $\sigma_x^2 = \boldsymbol{w}^T \mathbf{C} \boldsymbol{w}$ with constraint $\boldsymbol{w}^T \boldsymbol{w} = 1$
- Using the Lagrangian multiplier λ to define

$$L = \boldsymbol{w}^T \mathbf{C} \boldsymbol{w} - \lambda(\boldsymbol{w}^T \boldsymbol{w} - 1)$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = 2\mathbf{C}\boldsymbol{w} - 2\lambda \boldsymbol{w} = 0 \quad \Rightarrow \mathbf{C}\boldsymbol{w} = \lambda \boldsymbol{w}$$

$$\frac{\partial L}{\partial \lambda} = \boldsymbol{w}^T \boldsymbol{w} - 1 = 0 \qquad \Rightarrow \boldsymbol{w}^T \boldsymbol{w} = 1$$

- $\mathbf{C}\boldsymbol{w} = \lambda \boldsymbol{w}$ is the eigenvector/eigenvalue equation

$$\mathbf{C}\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i, \qquad i = 1, 2, \cdots, M$$

$$\lambda_1 > \lambda_2 > \cdots > \lambda_M$$

Orthogonal matrices and eigendecomposition

- $\mathbf{C}[\boldsymbol{w}_1\ \boldsymbol{w}_2 \cdots \boldsymbol{w}_M] = [\boldsymbol{w}_1\ \boldsymbol{w}_2 \cdots \boldsymbol{w}_M]\begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_M \end{bmatrix}$ or $\mathbf{CW} = \mathbf{W\Lambda}$
- $\mathbf{W}$ is orthogonal $\Rightarrow \mathbf{W}\mathbf{W}^T = \mathbf{W}^T\mathbf{W} = \mathbf{I} \Rightarrow \mathbf{W}^T\mathbf{CW} = \mathbf{W}^T\mathbf{W\Lambda} = \mathbf{\Lambda}$

$$\Rightarrow \boldsymbol{w}_j^T\mathbf{C}\boldsymbol{w}_k = \boldsymbol{w}_j^T(\lambda_k\boldsymbol{w}_k) = \begin{cases} \lambda_k\ , \text{if } j = k \\ 0\ \ , \text{if } j \neq k \end{cases}$$

$$\Rightarrow \max \sigma_x^2 = \boldsymbol{w}_1^T\mathbf{C}\boldsymbol{w}_1 = \lambda_1$$

$\Rightarrow$ If we find the M eigenvector/eigenvalue pairs of $\mathbf{C}$, the pair with the λ_1 corresponds to the projection with maximal variance, $\boldsymbol{w}_1$.

Principal Components Analysis

- Problem of PCA is to find the projections, $\underbrace{\boldsymbol{w}_1}_{M\times 1}, \cdots, \underbrace{\boldsymbol{w}_D}_{M\times 1}$
- PCA uses variance in the projected space as the criterion to choose $\boldsymbol{w}_d$.
  - $\boldsymbol{w}_1$ is the projection that makes the largest variance in the x_{n1}
  - $\boldsymbol{w}_2$ is also chosen to maximize the variance but $\boldsymbol{w}_2 \perp \boldsymbol{w}_1, \boldsymbol{w}_1^T \boldsymbol{w}_2 = 0$
  - $\boldsymbol{w}_3$ must maximize the variance and $\boldsymbol{w}_3 \perp \boldsymbol{w}_1$ and $\boldsymbol{w}_3 \perp \boldsymbol{w}_2$, etc.
  - In general, $\boldsymbol{w}_i \perp \boldsymbol{w}_j, \forall\, i \neq j$
  - If $D = M$, performing PCA is to rotate the original data without any loss of information.
  - PCA imposes the constraint $\boldsymbol{w}_i^T \boldsymbol{w}_i = 1$.
  - Assume $\overline{\boldsymbol{y}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{y}_n = \boldsymbol{0}$

Principal Components Analysis (PCA)

- PCA is the most widely used statistical technique for projecting data into a lower-dimensional space.

- PCA defines a linear projection: each of the projected dimensions is a linear combination of the original dimensions.

- If we are projecting $\underbrace{\boldsymbol{y}_n}_{M\times 1}$ to $\underbrace{\boldsymbol{x}_n}_{D\times 1} = [x_{n1}, \cdots, x_{nD}]^T$, $D \leq M$, PCA will define D vectors, $\underbrace{\boldsymbol{w}_1}_{M\times 1}, \cdots, \underbrace{\boldsymbol{w}_D}_{M\times 1}$, and the dth element of the projection,
  $x_{nd} = \boldsymbol{w}_d^T \boldsymbol{y}_n = w_{d1}y_{n1} + w_{d2}y_{n2} + \cdots + w_{dM}y_{nM}$

- The learning task is therefore to choose how many dimensions we want to project into (D) and then pick a projection vector, $\boldsymbol{w}_d$, for each.

PCA Algorithm

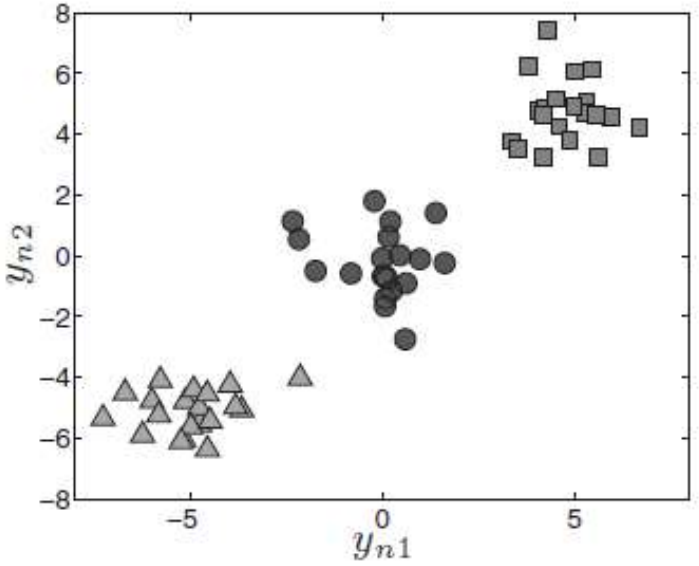
- Given a set of N samples, $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N$, each is M-dimensional.

$$\underbrace{\mathbf{Y}}_{N\times M} = \begin{bmatrix} \boldsymbol{y}_1^T \\ \vdots \\ \boldsymbol{y}_N^T \end{bmatrix} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,M} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N,1} & y_{N,2} & \cdots & y_{N,M} \end{bmatrix}, \text{ define } \underbrace{\mathbf{W}}_{M\times D} = [\boldsymbol{w}_1\, \boldsymbol{w}_2 \cdots \boldsymbol{w}_D]$$

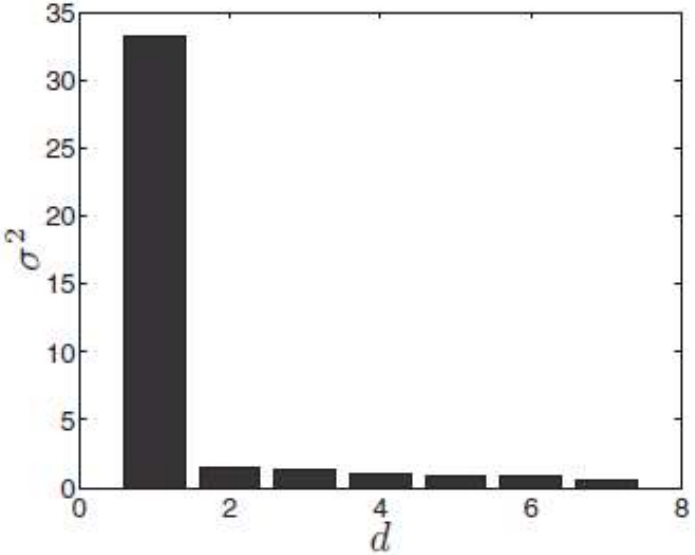
1. Transform $\boldsymbol{y}_i, i = 1, \cdots, N$ to have zero mean by $\boldsymbol{y}_i - \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{y}_n$
2. Compute the sample covariance matrix $\underbrace{\mathbf{C}}_{M\times M} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{y}_n \boldsymbol{y}_n^T$ or $\mathbf{C} = \frac{1}{N}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$
3. Find the M eigenvector/eigenvalue pairs of $\mathbf{C}$
4. Find eigenvectors $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_D$ corresponding to the D highest eigenvalues

5. Create the dth dimension for object n in the projection, $x_{nd} = \boldsymbol{w}_d^T \boldsymbol{y}_n$, or $\underbrace{\mathbf{X}}_{N\times D} = \underbrace{\mathbf{Y}}_{N\times M}\ \underbrace{\mathbf{W}}_{M\times D}$

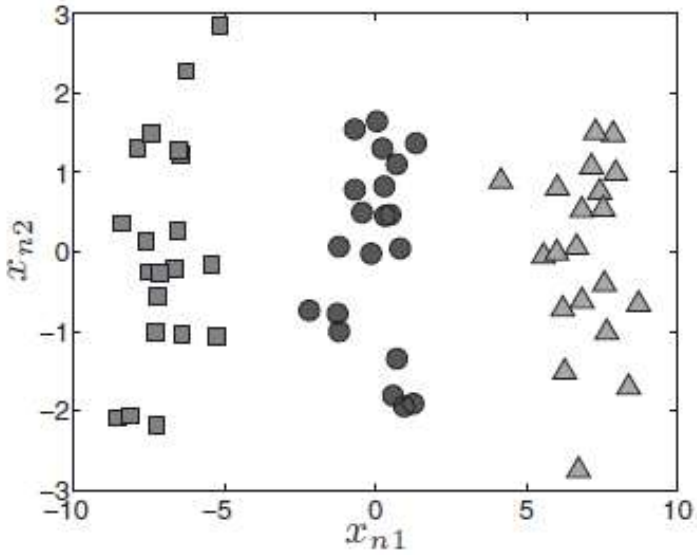
PCA Example 1



(a) First two dimensions of the data objects $\mathbf{y}_n$.



(b) The seven eigenvalues (variances of the projected dimensions).



(c) The data projected onto the first two principal components.

- We generated a dataset where each object is drawn from one of 3 clusters ((a)).
- We make the data more complex by adding additional 5 dimensions whose values are drawn from $N(0,1)$, i.e., $y_{nd} \sim N(0,1), d = 3, \cdots, 7, n = 1, \cdots, N$

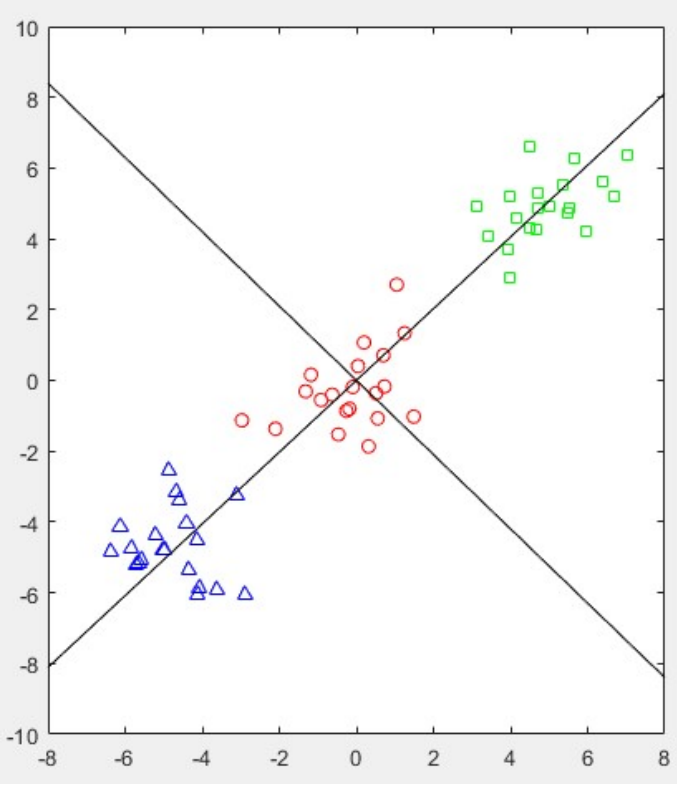
MATLAB script: pcaexample.m

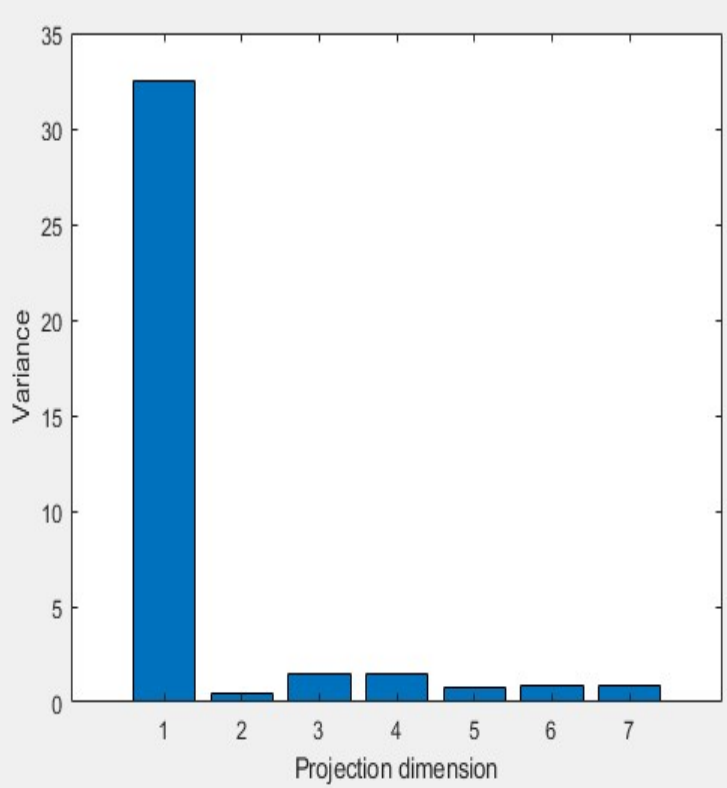
```
%% pcaexample.m % From A First Course in Machine Learning, Chapter 6.
% PCA example
clear all;close all; %% Generate the data
Y = [randn(20,2);randn(20,2)+5;randn(20,2)-5];
N = size(Y,1);
Y = [Y randn(N,5)]; % Add 5 random dimensions
% labels - just used for plotting
t = [repmat(1,20,1);repmat(2,20,1);repmat(3,20,1)];
%% Plot the original data
symbs = {'ro','gs','b^'}; figure(1);hold off
for k = 1:3
    pos = find(t==k);
    plot(Y(pos,1),Y(pos,2),symbs{k});
    hold on
end
%% Do the PCA
Y = Y - repmat(mean(Y,1),N,1); % Subtract the means
C = (1/N)*Y'*Y; % Compute covariance matrix
% Find the eigen-vectors/values
% columns of w correspond to the projection directions
[w,lam] = eig(C);
```

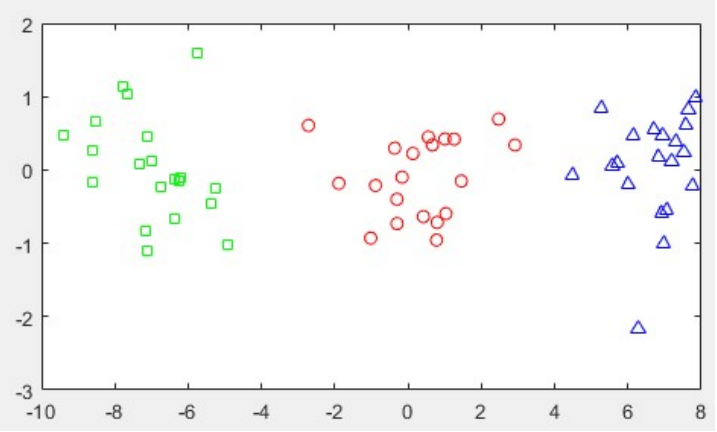
```
%% Plot the first two components on to the original data
figure(1);hold off
for k = 1:3
    pos = find(t==k);
    plot(Y(pos,1),Y(pos,2),symbs{k});
    hold on
end
xl = xlim;
for k = 1:2
    plot(xl,xl*w(1,k)/w(2,k),'k');
end
%% Bar plot of the eigenvalues
figure(2);hold off
bar(diag(lam));xlabel('Projection dimension');ylabel('Variance');

%% Plot the data projected into the first two dimensions
X = Y*w(:,1:2);
figure(3);hold off
for k = 1:3
    pos = find(t==k);
    plot(X(pos,1),X(pos,2),symbs{k});
    hold on
end
```

Results of MATLAB script: pcaexample.m

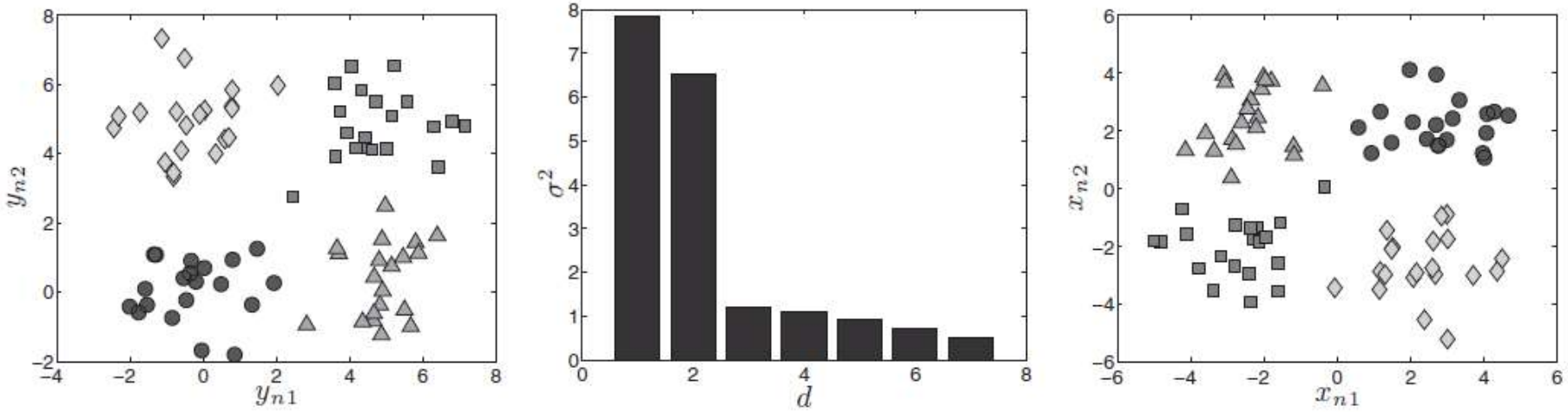






- The magnitudes of the eigenvalues (middle plot) gives us some indication of how many interesting features there are in our data. In particular, we are unlikely to gain much by using two projected dimensions rather than one

PCA Example 2



(a) First two dimensions of the data objects $\mathbf{y}_n$.

(b) The seven eigenvalues (variances of the projected dimensions).

(c) The data projected onto the first two principal components.

- We generated a dataset where each object is drawn from one of 4 clusters ((a)).
- We make the data more complex by adding additional 5 dimensions whose values are drawn from $N(0,1)$, i.e., $y_{nd} \sim N(0,1), d = 5, \cdots, 9, n = 1, \cdots, N$
- The first two eigenvalues are both much higher than the remainder.
- The data projected onto these first two components can be seen in (c), and it is clear that the cluster structure is preserved in the reduced space.

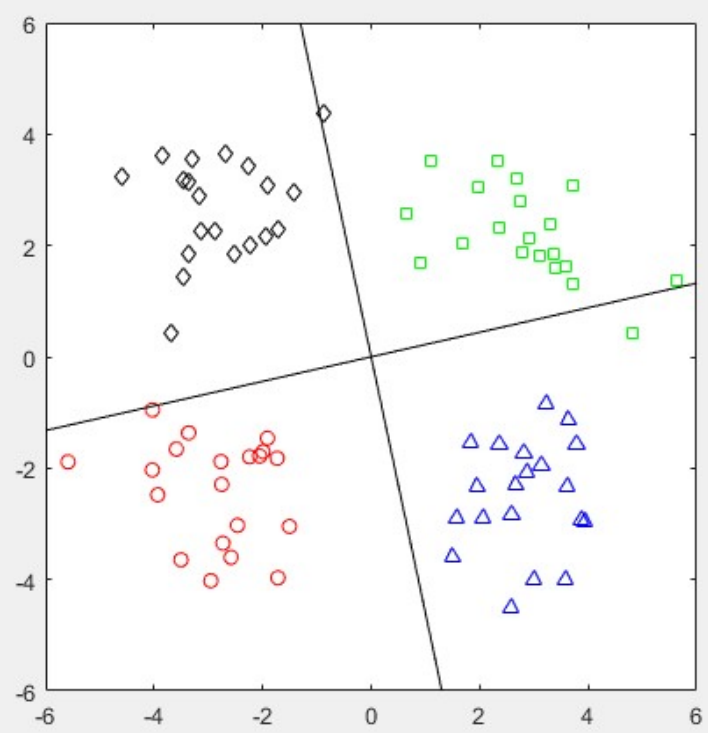
MATLAB script: pcaexample2.m

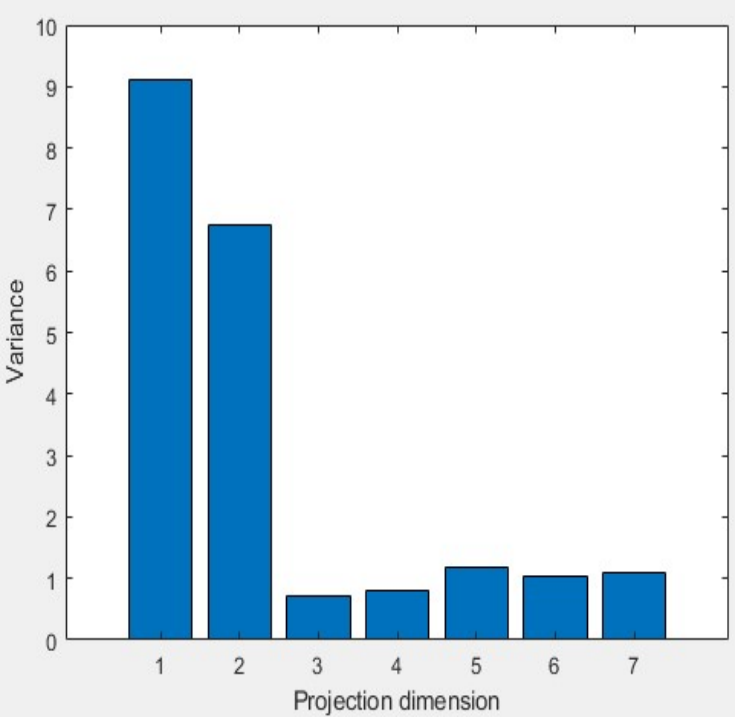
```
%% pcaexample2.m % From A First Course in Machine Learning, Chapter 6.
% Second PCA example
clear all;close all;
%% Generate the data
Y = [randn(20,2);randn(20,2)+5;randn(20,2)+repmat([5
0],20,1);randn(20,2)+repmat([0 5],20,1)];
N = size(Y,1);
Y = [Y randn(N,5)]; % Add 5 random dimensions
% labels - just used for plotting
t = [repmat(1,20,1);repmat(2,20,1);repmat(3,20,1);repmat(4,20,1)];
%% Plot the original data
symbs = {'ro','gs','b^','kd'}; figure(1);hold off
for k = 1:4
    pos = find(t==k);
    plot(Y(pos,1),Y(pos,2),symbs{k}); hold on
end
%% Do the PCA
Y = Y - repmat(mean(Y,1),N,1); % Subtract the means
C = (1/N)*Y'*Y; % Compute covariance matrix
% Find the eigen-vectors/values
% columns of w correspond to the projection directions
[w,lam] = eig(C);
```

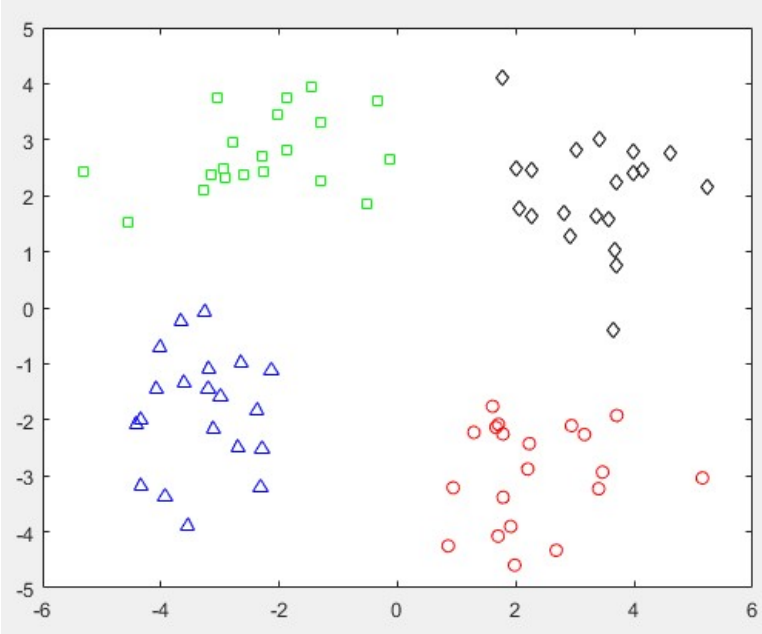
```
%% Plot the first two components on to the original data
figure(1);hold off
for k = 1:4
    pos = find(t==k);
    plot(Y(pos,1),Y(pos,2),symbs{k});hold on
end
xl = xlim;
for k = 1:2
    plot(xl,xl*w(1,k)/w(2,k),'k');
end
ylim(xl);
%% Bar plot of the eigenvalues
figure(2); hold off
bar(diag(lam)); xlabel('Projection dimension');ylabel('Variance');

%% Plot the data projected into the first two dimensions
X = Y*w(:,1:2);
figure(3);hold off
for k = 1:4
    pos = find(t==k);
    plot(X(pos,1),X(pos,2),symbs{k});
    hold on
end
```

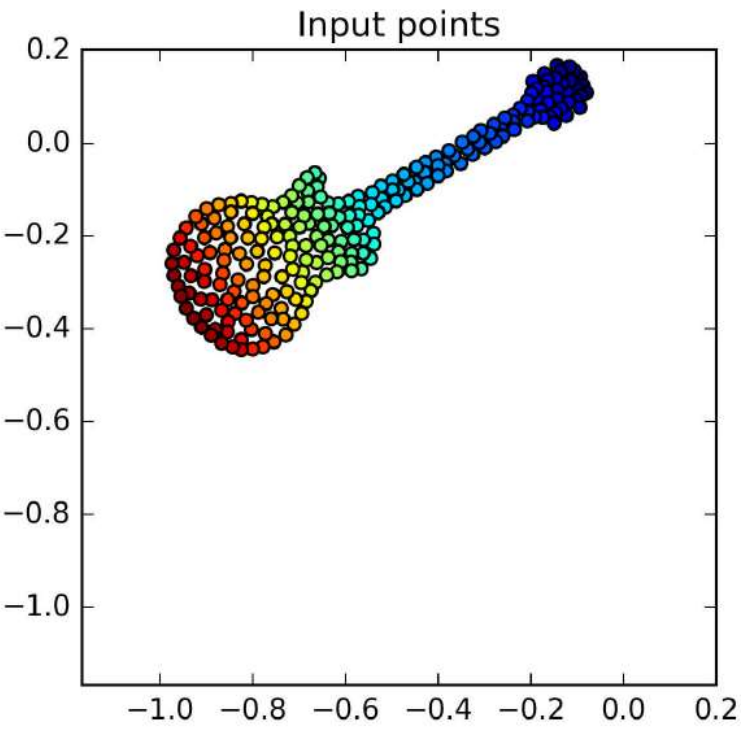
Results of MATLAB script: pcaexample2.m

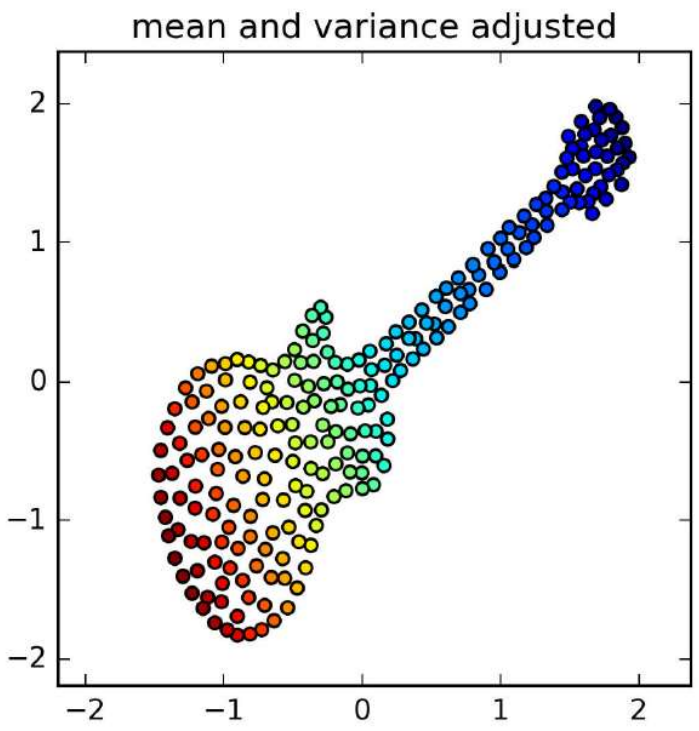


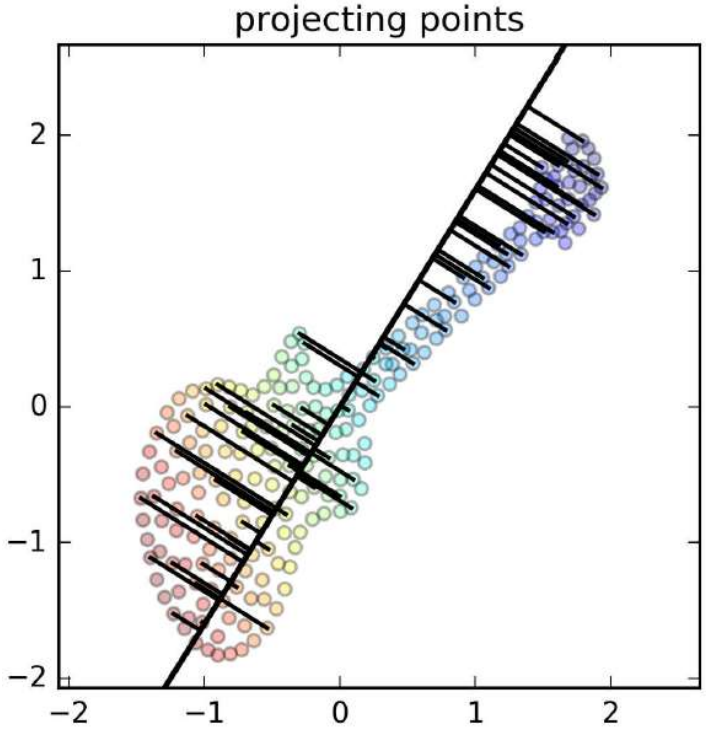




Principal Component Analysis (PCA)





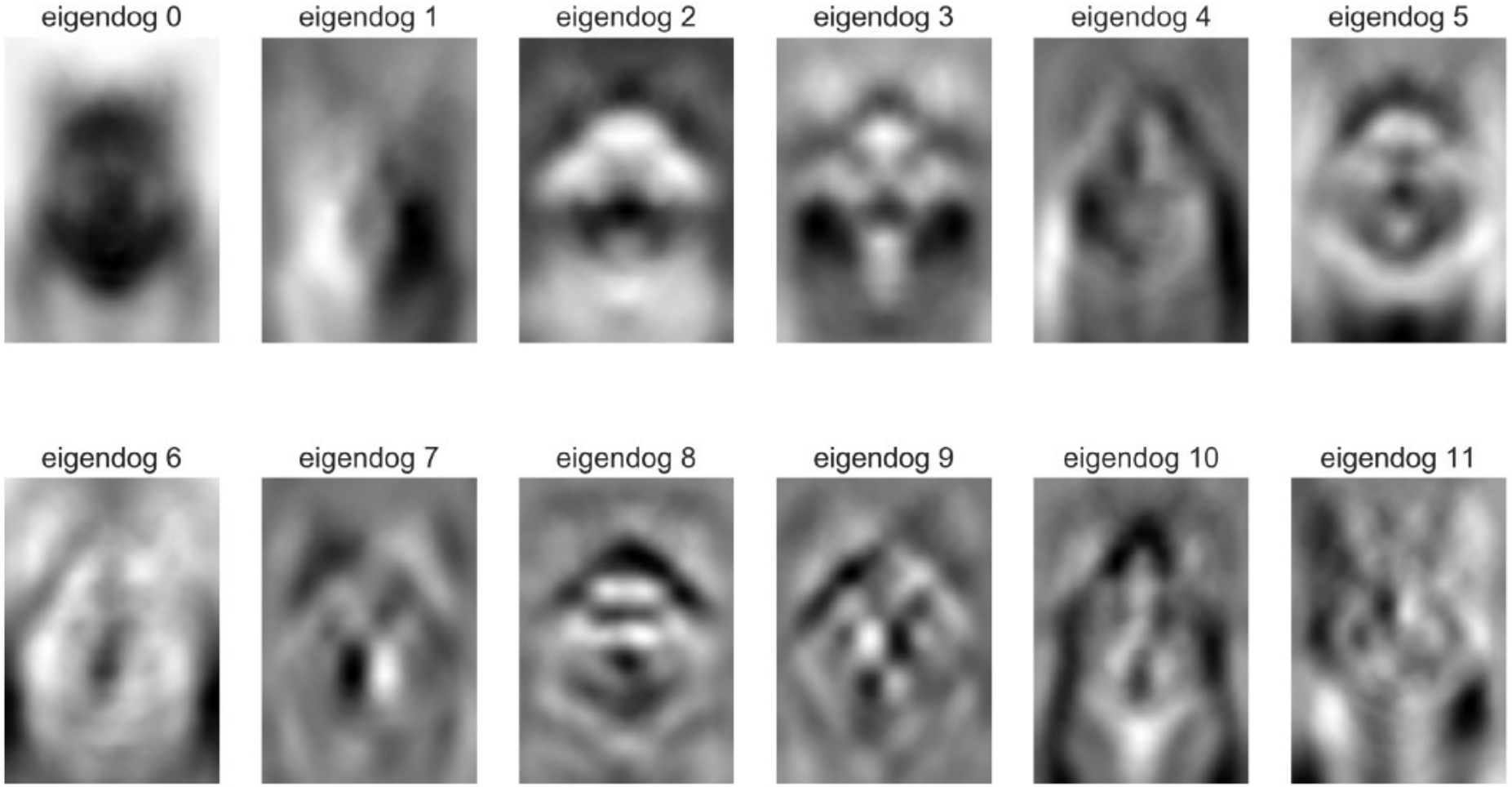


Chapter 10 in Deep Learning, Vol. 1_ From Basics to Practice Andrew Glassner, 2018

PCA for Real Images: Starting set of Huskies

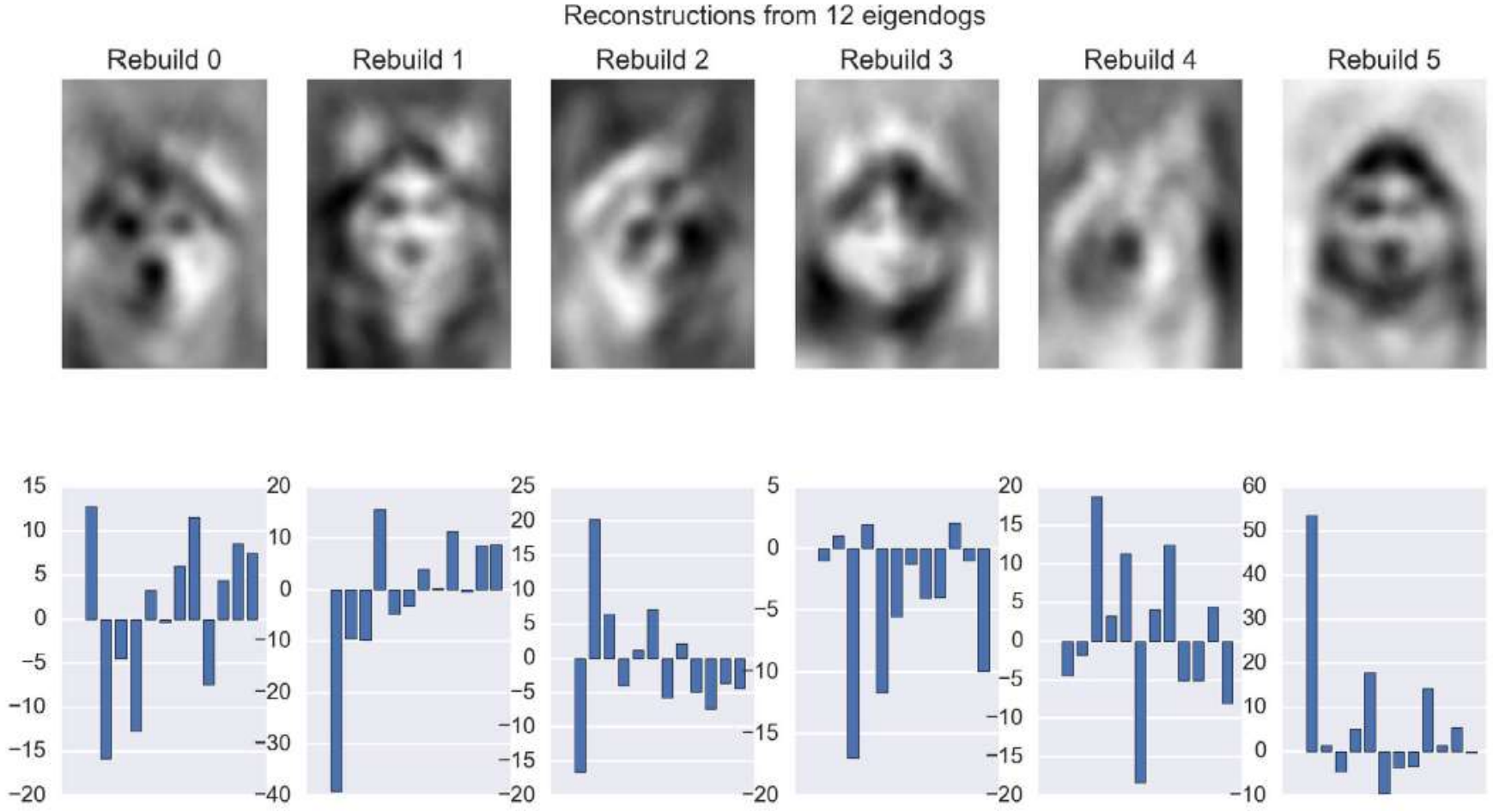
First six Huskies after standardization

The 12 eigendogs produced by PCA



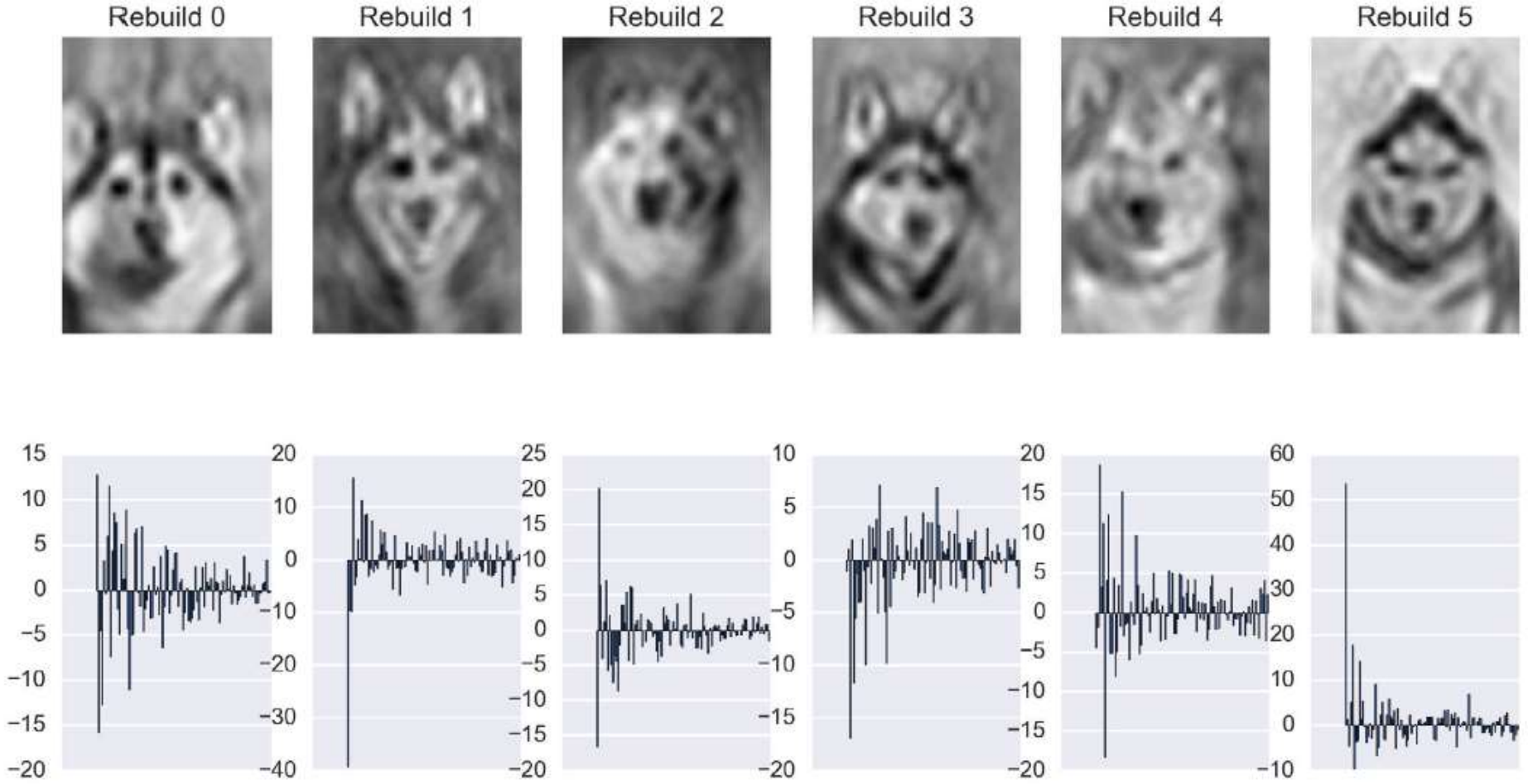
Reconstructing our original inputs from a set of 12 eigendogs

Top row: The reconstructed dog. Bottom row: The weights applied to the eigendogs of Figure in previous slide to build the image directly above.



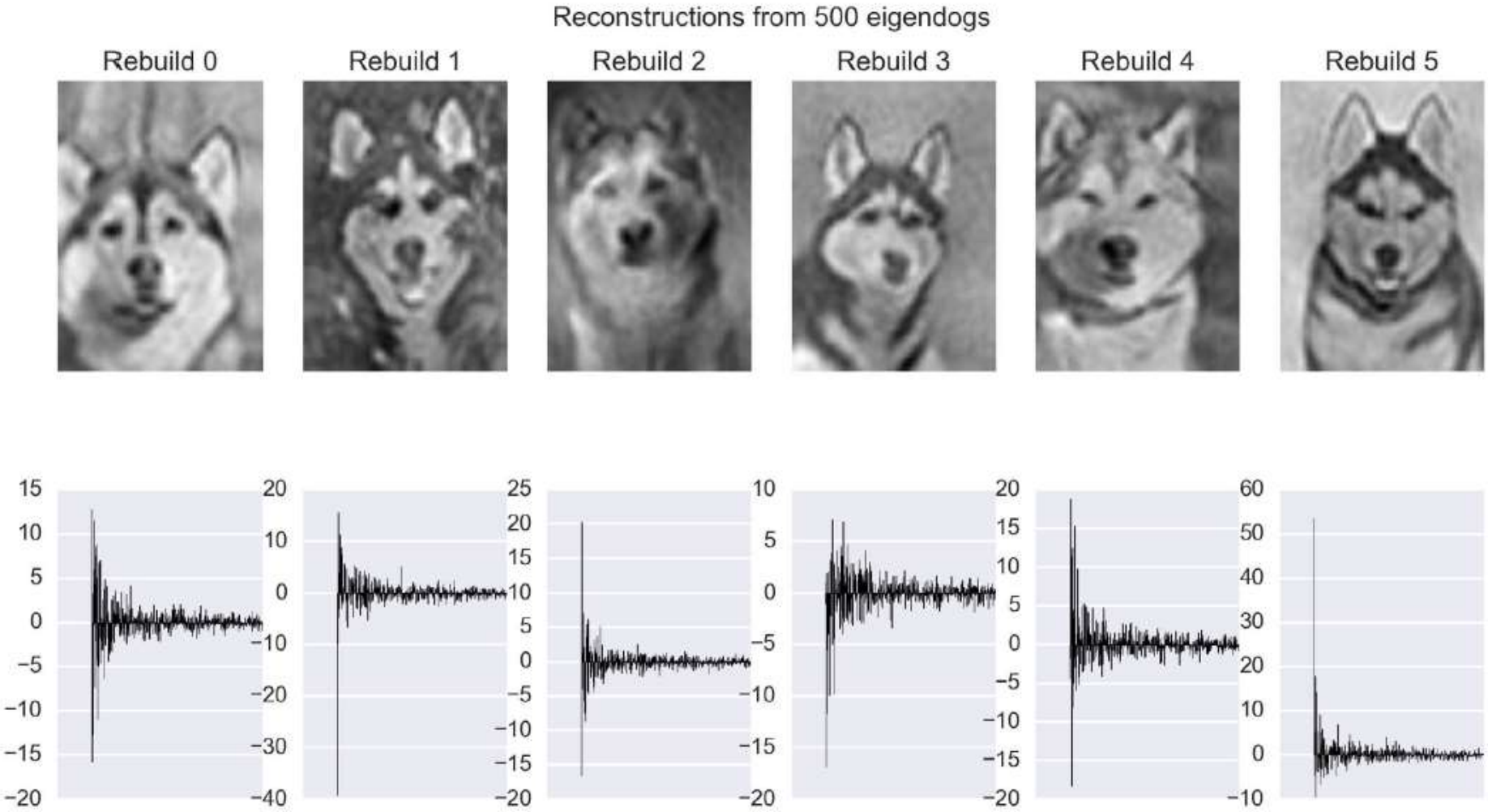
Reconstructing our original inputs from a set of 100 eigendogs

Top: The reconstructed dog. Bottom: The weights applied to the eigendogs.

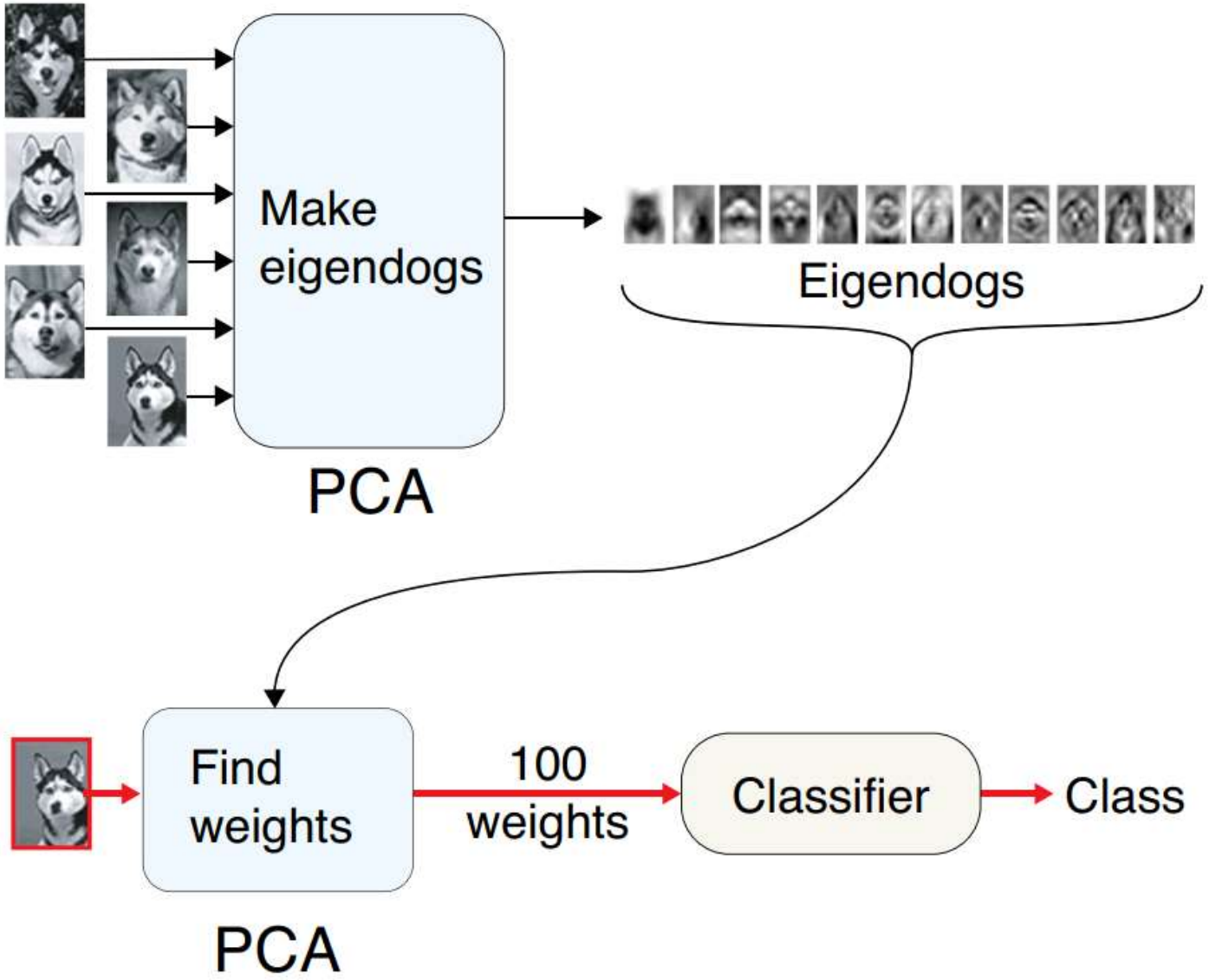


Reconstructing our original inputs from a set of 500 eigendogs

Top: The reconstructed dog. Bottom: The weights applied to the eigendogs.



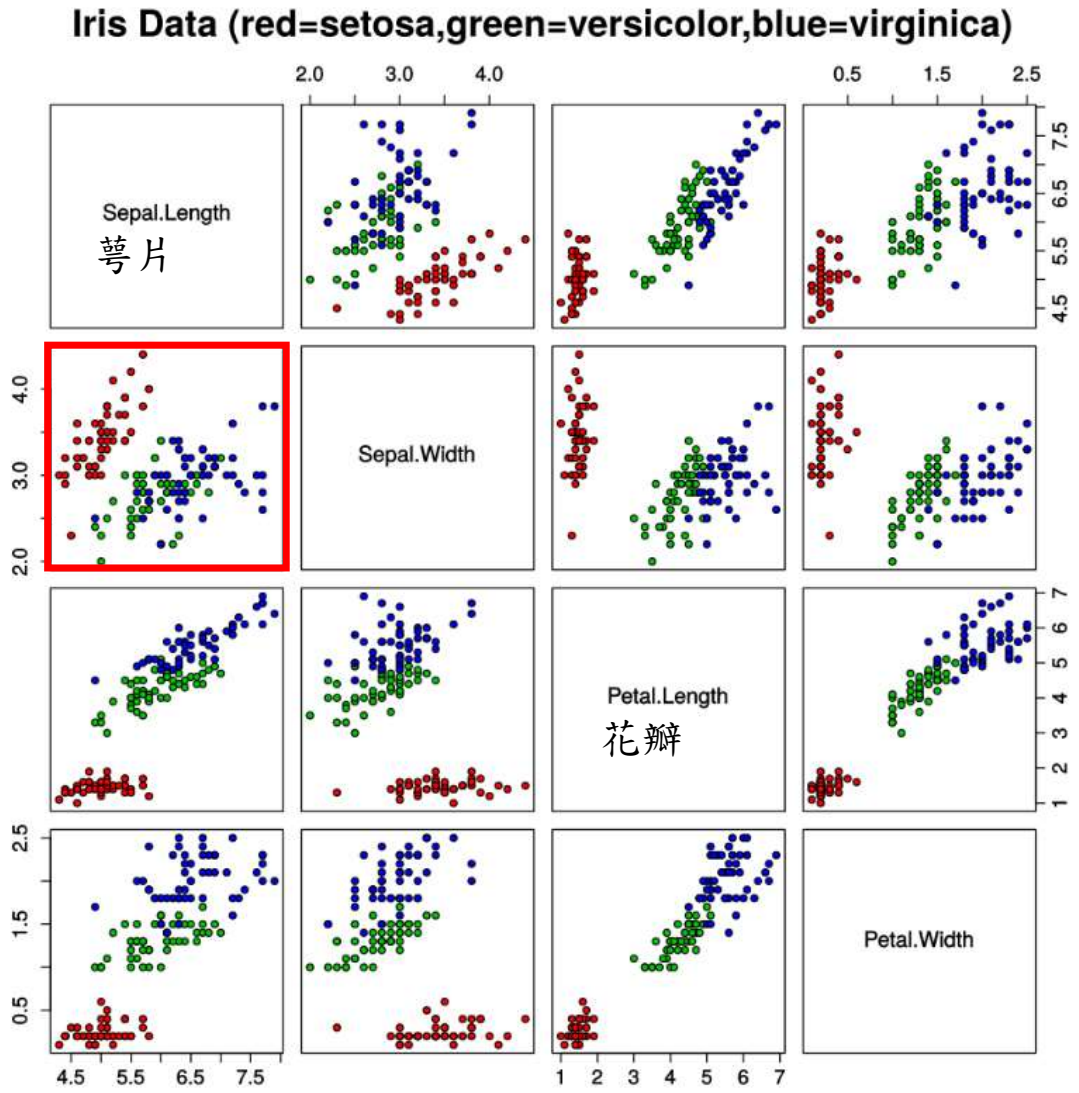
Only uses those weights to find the input's class

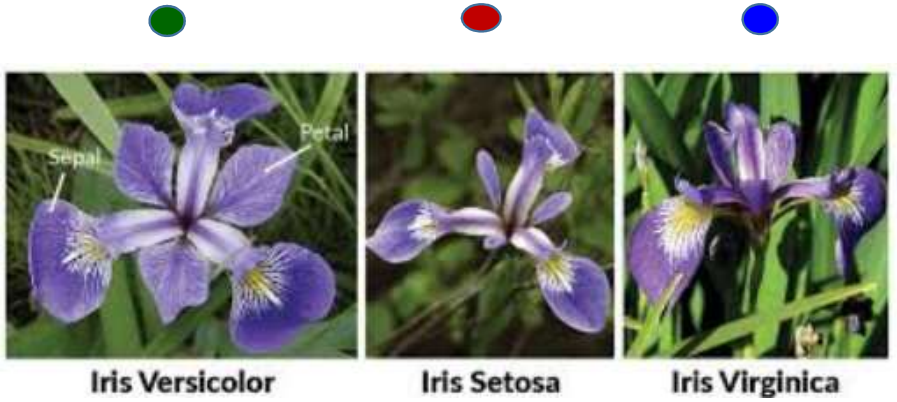


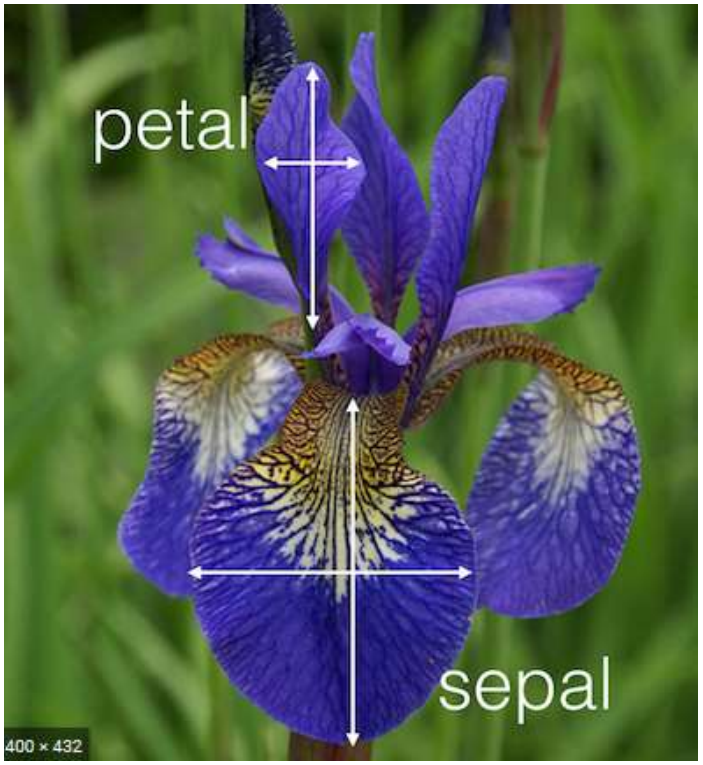
安德森鳶尾花卉數據集

- 安德森鳶尾花卉數據集（Anderson's Iris data set），也稱鳶尾花卉數據集（Iris flower data set）或費雪鳶尾花卉數據集（Fisher's Iris data set），是一類多重變量分析的數據集。
- 它最初是埃德加·安德森（Edgar Anderson）從加拿大加斯帕半島上的鳶尾屬花朵中提取的形態學變異數據，後由羅納德·費雪作為判別分析的一個例子，運用到統計學中。
- 其數據集包含了150個樣本，都屬於鳶尾屬下的三個亞屬，分別是山鳶尾、變色鳶尾和維吉尼亞鳶尾（Virginia Iris）。
- 四個特徵被用作樣本的定量分析，它們分別是花萼和花瓣的長度和寬度。

https://zh.wikipedia.org/wiki/%E5%AE%89%E5%BE%B7%E6%A3%AE%E9%B8%A2%E5%B0%BE%E8%8A%B1%E5%8D%89%E6%95%B0%E6%8D%AE%E9%9B%86


Iris Data (red=setosa,green=versicolor,blue=virginica)
Sepal.Length
萼片
Sepal.Width
Petal.Length
花瓣
Petal.Width

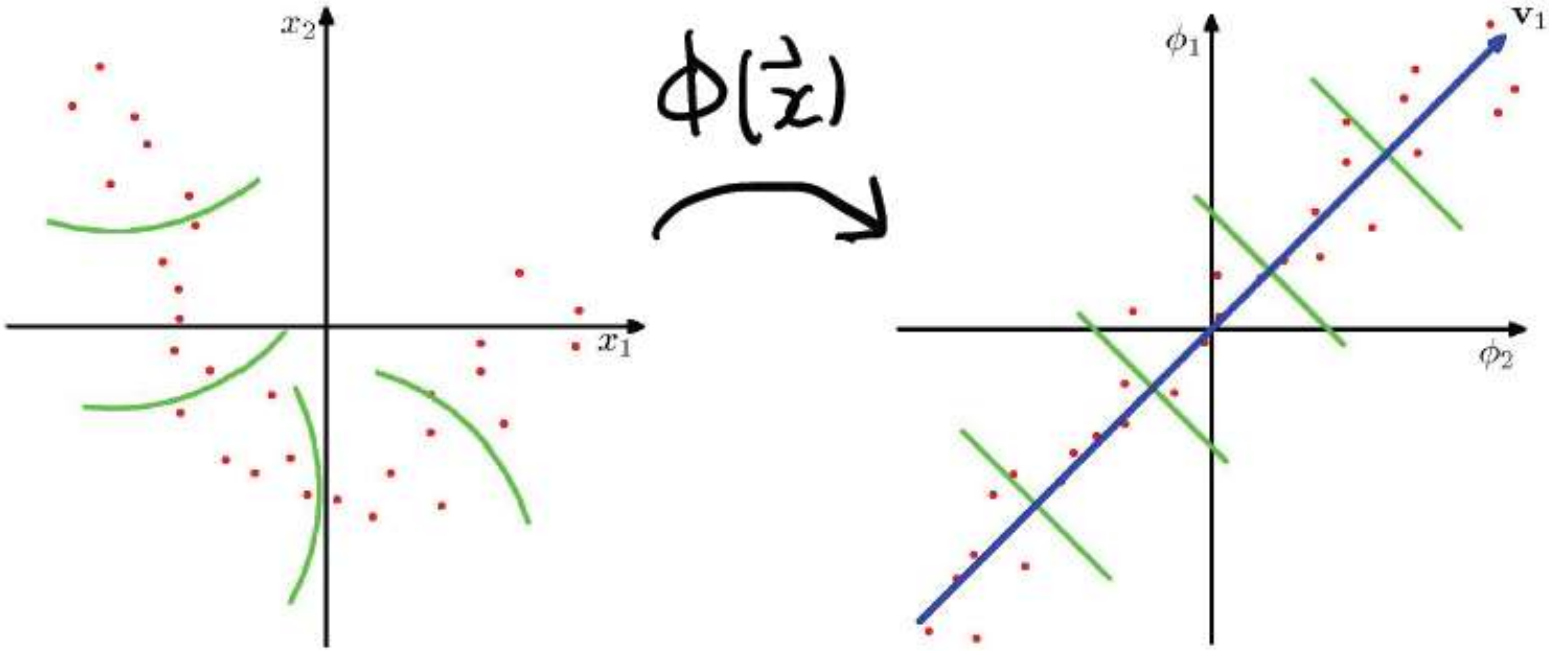

Sepal
Petal
Iris Versicolor
Iris Setosa
Iris Virginica


petal
sepal
400 × 432

Kernel PCA

Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2007

Kernel PCA



A data set in the original data space (left-hand plot) is projected by a nonlinear transformation $\boldsymbol{\varphi}(\mathbf{x})$ into a feature space (right-hand plot). By performing PCA in the feature space, we obtain the principal components, of which the first is shown in blue and is denoted by the vector $\mathbf{v}_1$. The green lines in feature space indicate the linear projections onto the first principal component, which correspond to nonlinear projections in the original data space. Note that in general it is not possible to represent the nonlinear principal component by a vector in $\mathbf{x}$ space.

Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2007

Kernel PCA

- Given observations $\vec{x}_1, \cdots, \vec{x}_N, \vec{x}_i \in \mathbb{R}^D$, assume zero mean $\sum_{i=1}^{N} \vec{x}_i = 0$
- In PCA, we solve $S\vec{u}_i = \lambda_i \vec{u}_i, \vec{u}_i{}^T \vec{u}_i = 1$, where $S = \frac{1}{N} \sum_{i=1}^{N} \underset{D\times 1}{\vec{x}_i} \underset{1\times D}{\vec{x}_i{}^T}$
- In kernel PCA, $\vec{x}_i \in \mathbb{R}^D \rightarrow \varphi(\vec{x}_i) \in \mathbb{R}^M$, $M \geq D$
- Assume transformed data has zero mean $\sum_{i=1}^{N} \varphi(\vec{x}_i) = 0$
- Compute the sample covariance matrix $C = \frac{1}{N} \sum_{i=1}^{N} \underset{M\times 1}{\varphi(\vec{x}_i)} \underset{1\times M}{\varphi(\vec{x}_i)^T}$ which is symmetric

Eigendecomposition problem

- $C\vec{v}_i = \lambda_i \vec{v}_i,\ i = 1, \cdots, M\ , \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M > 0$

$$\Rightarrow \frac{1}{N}\sum_{i=1}^{N} \underset{M\times 1}{\varphi(\vec{x}_i)} \left\{ \underset{1\times M}{\varphi(\vec{x}_i)^T}\ \underset{M\times 1}{\vec{v}_i} \right\} = \lambda_i \vec{v}_i$$

$$\Rightarrow \vec{v}_i = \sum_{n=1}^{N} a_{in}\varphi(\vec{x}_n)\ , a_{in} = \frac{1}{N\lambda_i} \underset{1\times M}{\varphi(\vec{x}_i)^T}\ \underset{M\times 1}{\vec{v}_i}$$

- Finding $\vec{v}_i$ is equivalent to finding a_{in},

$$C\vec{v}_i = \lambda_i \vec{v}_i \Rightarrow \frac{1}{N}\sum_{n=1}^{N} \varphi(\vec{x}_n)\varphi(\vec{x}_n)^T \sum_{m=1}^{N} a_{im}\varphi(\vec{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}\varphi(\vec{x}_n)$$

Multiply both sides by $\varphi(\vec{x}_l)^T$, $l = 1, \cdots, N$ to have kernel

$$\mathrm{K}(\vec{x}_n, \vec{x}_m) = \varphi(\vec{x}_n)^T \varphi(\vec{x}_m) ,$$

$$\Rightarrow \frac{1}{N} \sum_{n=1}^{N} \underbrace{\varphi(\vec{x}_l)^T \varphi(\vec{x}_n)}_{\mathrm{K}(\vec{x}_l, \vec{x}_n)} \sum_{m=1}^{N} a_{im} \underbrace{\varphi(\vec{x}_n)^T \varphi(\vec{x}_m)}_{\mathrm{K}(\vec{x}_n, \vec{x}_m)} = \lambda_i \sum_{n=1}^{N} a_{in} \underbrace{\varphi(\vec{x}_l)^T \varphi(\vec{x}_n)}_{\mathrm{K}(\vec{x}_l, \vec{x}_n)}$$

$$\Rightarrow \sum_{n=1}^{N} \sum_{m=1}^{N} \underbrace{\mathrm{K}(\vec{x}_l, \vec{x}_n)}_{\triangleq K_{ln}} \underbrace{\mathrm{K}(\vec{x}_n, \vec{x}_m)}_{\triangleq K_{nm}} a_{im} = \lambda_i N \sum_{n=1}^{N} \underbrace{\mathrm{K}(\vec{x}_l, \vec{x}_n)}_{\triangleq K_{ln}} a_{in}$$

Example N=2 problem

$l = 1$,

$$\sum_{n=1}^{2}\sum_{m=1}^{2} k_{1n}\,k_{nm}a_{im} = k_{11}k_{11}a_{i1} + k_{12}k_{21}a_{i1} + k_{11}k_{12}a_{i2} + k_{12}k_{22}a_{i2}$$

$$\lambda_i \times 2\sum_{n=1}^{2} k_{1n}a_{in} = 2\lambda_i(k_{11}a_{i1} + k_{12}a_{i2})$$

$$\Rightarrow \left[{k_{11}}^2 + k_{12}k_{21}\;\; k_{11}k_{12} + k_{12}k_{22}\right]\begin{bmatrix}a_{i1}\\ a_{i2}\end{bmatrix} = 2\lambda_i[k_{11}\;k_{12}]\begin{bmatrix}a_{i1}\\ a_{i2}\end{bmatrix} \qquad ①$$

Example *N*=2 problem

$l = 2$,

$$\sum_{n=1}^{2}\sum_{m=1}^{2} k_{2n}\, k_{nm} a_{im} = k_{21}k_{11}a_{i1} + k_{22}k_{21}a_{i1} + k_{21}k_{12}a_{i2} + k_{22}k_{22}a_{i2}$$

$$\lambda_i \times 2\sum_{n=1}^{2} k_{2n}a_{in} = 2\lambda_i(k_{21}a_{i1} + k_{22}a_{i2})$$

$$\Rightarrow \begin{bmatrix} k_{21}k_{11} + k_{22}k_{21} & k_{21}k_{12} + k_{22}{}^2 \end{bmatrix} \begin{bmatrix} a_{i1} \\ a_{i2} \end{bmatrix} = 2\lambda_i [k_{21}\ k_{22}] \begin{bmatrix} a_{i1} \\ a_{i2} \end{bmatrix} \quad ②$$

Example N=2 problem

① & ②

$$\Rightarrow \underbrace{\begin{bmatrix} {k_{11}}^2 + k_{12}k_{21} & k_{11}k_{12} + k_{12}k_{22} \\ k_{21}k_{11} + k_{22}k_{21} & k_{21}k_{12} + {k_{22}}^2 \end{bmatrix}}_{=\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}} \begin{bmatrix} a_{i1} \\ a_{i2} \end{bmatrix} = 2\lambda_i \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{bmatrix} a_{i1} \\ a_{i2} \end{bmatrix}$$

$$\Rightarrow K^2 \vec{a}_i = N\lambda_i K \vec{a}_i$$

$$\Rightarrow K\vec{a}_i = (N\lambda_i)\vec{a}_i, \quad \text{if } K \text{ is invertible}$$

eigenvalue problem (positive definite)

Normalization condition for $\vec{a}_i$

$$1 = \vec{v}_i{}^T \vec{v}_i = \left(\sum_{n=1}^{N} a_{in}\varphi(\vec{x}_n)\right)^T \sum_{m=1}^{N} a_{im}\varphi(\vec{x}_m)$$

$$= \sum_{n=1}^{N}\sum_{n=1}^{N} a_{in}a_{im}\underbrace{\varphi(\vec{x}_n)^T\varphi(\vec{x}_m)}_{K_{nm}}$$

$$= [a_{i1}\cdots a_{iN}]\begin{bmatrix} K_{11} & \cdots & K_{1N} \\ \vdots & \ddots & \vdots \\ K_{N1} & \cdots & K_{NN} \end{bmatrix}\begin{bmatrix} a_{i1} \\ \vdots \\ a_{iN} \end{bmatrix}$$

$$= \vec{a}_i{}^T K\vec{a}_i$$

$$= N\lambda_i \vec{a}_i{}^T \vec{a}_i$$

$$\Rightarrow \vec{a}_i{}^T \vec{a}_i = \|\vec{a}_i\|^2 = \frac{1}{N\lambda_i}$$

Check $N = 2$

$$[a_{i1}\ a_{i2}]\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}\begin{bmatrix} a_{i1} \\ a_{i2} \end{bmatrix}$$

$$= [a_{i1}K_{11} + a_{i2}K_{21}\ \ a_{i1}K_{12} + a_{i2}K_{22}]\begin{bmatrix} a_{i1} \\ a_{i2} \end{bmatrix}$$

$$= {a_{i1}}^2 K_{11} + a_{i2}a_{i1}K_{21} + a_{i1}a_{i2}K_{12} + {a_{i2}}^2 K_{22}$$

$$= \sum_{n=1}^{2} \sum_{n=1}^{2} a_{in}a_{im}\, K_{nm}$$

Project a new observation $\vec{x}$

- Once $\vec{v}_i$ is obtained, we can project a new observation $\vec{x}$ onto $\vec{v}_i$ by

$$\varphi(\vec{x})^T \vec{v}_i = \varphi(\vec{x})^T \sum_{n=1}^{N} a_{in} \varphi(\vec{x}_n) = \sum_{i=1}^{N} a_{in}\, \varphi(\vec{x})^T \varphi(\vec{x}_n)$$

$$= \sum_{i=1}^{N} a_{in}\, K(\vec{x}\,, \vec{x}_n)$$

- Note the above derivations assume zero mean

$$\frac{1}{N} \sum_{n=1}^{N} \varphi(\vec{x}_n) = 0$$

- We don't need to know $\varphi(\vec{x})$ exactly to compute the projection

If $\varphi(\vec{x}_n)$ are not zero mean

If $\frac{1}{N}\sum_{n=1}^{N}\varphi(\vec{x}_n) \neq 0$, define $\tilde{\varphi}(\vec{x}_n) = \varphi(\vec{x}_n) - \frac{1}{N}\sum_{l=1}^{N}\varphi(\vec{x}_l)$

The Gram matrix becomes

$$\begin{aligned}\tilde{K}_{nm} &= \tilde{\varphi}(\vec{x}_n)^T\tilde{\varphi}(\vec{x}_m) \\ &= \varphi(\vec{x}_n)^T\varphi(\vec{x}_m) - \frac{1}{N}\sum_{l=1}^{N}\varphi(\vec{x}_n)^T\varphi(\vec{x}_l) - \frac{1}{N}\sum_{l=1}^{N}\varphi(\vec{x}_l)^T\varphi(\vec{x}_m) + \\ &\quad \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}\varphi\left(\vec{x}_j\right)^T\varphi(\vec{x}_l) \\ &= K(\vec{x}_n,\vec{x}_m) - \frac{1}{N}\sum_{l=1}^{N}K(\vec{x}_n,\vec{x}_l) - \frac{1}{N}\sum_{l=1}^{N}K(\overrightarrow{x_l},\vec{x}_m) + \\ &\quad \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}K\left(\vec{x}_j,\vec{x}_l\right)\end{aligned}$$

$$\Rightarrow \tilde{K} = K - KI_N - I_NK + I_NKI_N \text{ where } I_N = \frac{1}{N}\underbrace{\begin{bmatrix}1 & \cdots & 1\\ \vdots & \ddots & \vdots\\ 1 & \cdots & 1\end{bmatrix}}_{N\times N}$$

Example

$K(\vec{x}, \vec{x}') = e^{\frac{-\|\vec{x}-\vec{x}'\|^2}{0.1}}$ is applied to a synthetic data.

The lines correspond to contours along which the projection onto the corresponding principal component

$$\varphi(\vec{x})^T \vec{v}_i = \sum_{n=1}^{N} a_{in} K(\vec{x}, \vec{x}_n)$$

is constant

Projection onto the Principal Component

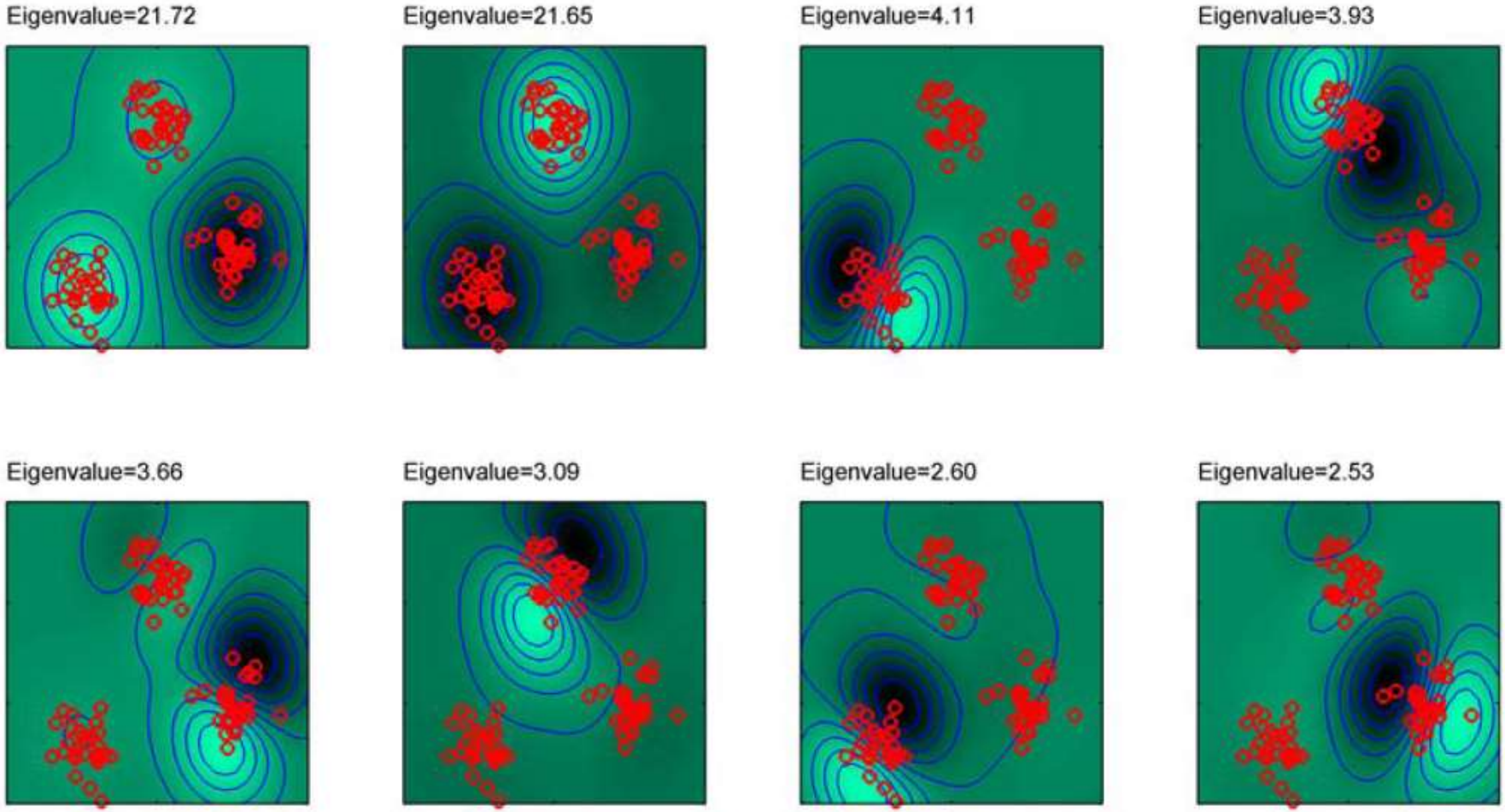
$$\varphi(\vec{x}_1)^T \vec{v}_i = a_{i1} K(\vec{x}_1, \vec{x}_1) + a_{i2} K(\vec{x}_1, \vec{x}_2) + \cdots + a_{iN} K(\vec{x}_1, \vec{x}_N) = [K_{11}\ K_{12} \cdots K_{1N}] \begin{bmatrix} a_{i1} \\ \vdots \\ a_{iN} \end{bmatrix}$$

$$\varphi(\vec{x}_2)^T \vec{v}_i = a_{i1} K(\vec{x}_2, \vec{x}_1) + a_{i2} K(\vec{x}_2, \vec{x}_2) + \cdots + a_{iN} K(\vec{x}_2, \vec{x}_N) = [K_{21}\ K_{22} \cdots K_{2N}] \begin{bmatrix} a_{i1} \\ \vdots \\ a_{iN} \end{bmatrix}$$

$$\vdots$$

$$\varphi(\vec{x}_N)^T \vec{v}_i = a_{i1} K(\vec{x}_N, \vec{x}_1) + a_{i2} K(\vec{x}_N, \vec{x}_2) + \cdots + a_{iN} K(\vec{x}_N, \vec{x}_N) = [K_{N1}\ K_{N2} \cdots K_{NN}] \begin{bmatrix} a_{i1} \\ \vdots \\ a_{iN} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \varphi(\vec{x}_1)^T \vec{v}_i \\ \vdots \\ \varphi(\vec{x}_N)^T \vec{v}_i \end{bmatrix} = \begin{bmatrix} K_{11} & \cdots & K_{1N} \\ \vdots & \ddots & \vdots \\ K_{N1} & \cdots & K_{NN} \end{bmatrix} \begin{bmatrix} a_{i1} \\ \vdots \\ a_{iN} \end{bmatrix}$$



Example of kernel PCA, with a Gaussian kernel applied to a synthetic data set in two dimensions, showing the first eight eigenfunctions along with their eigenvalues. The contours are lines along which the projection onto the corresponding principal component is constant. Note how the first two eigenvectors separate the three clusters, the next three eigenvectors split each of the cluster into halves, and the following three eigenvectors again split the clusters into halves along directions orthogonal to the previous splits.

Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2007

Homework # 2

Rewrite the code Lecture 02_3 Kernel_PCA_RBF.jpynb into the fashion of object-oriented from

- Hint: see Homework2 Kernel_PCA_RBF_oop_to_do.jpynb

Deadline of Homework #2: 2022/10/3 3:30pm