Rozpoznávanie obrazcov
šk.r. 2017-18

PCA a LDA

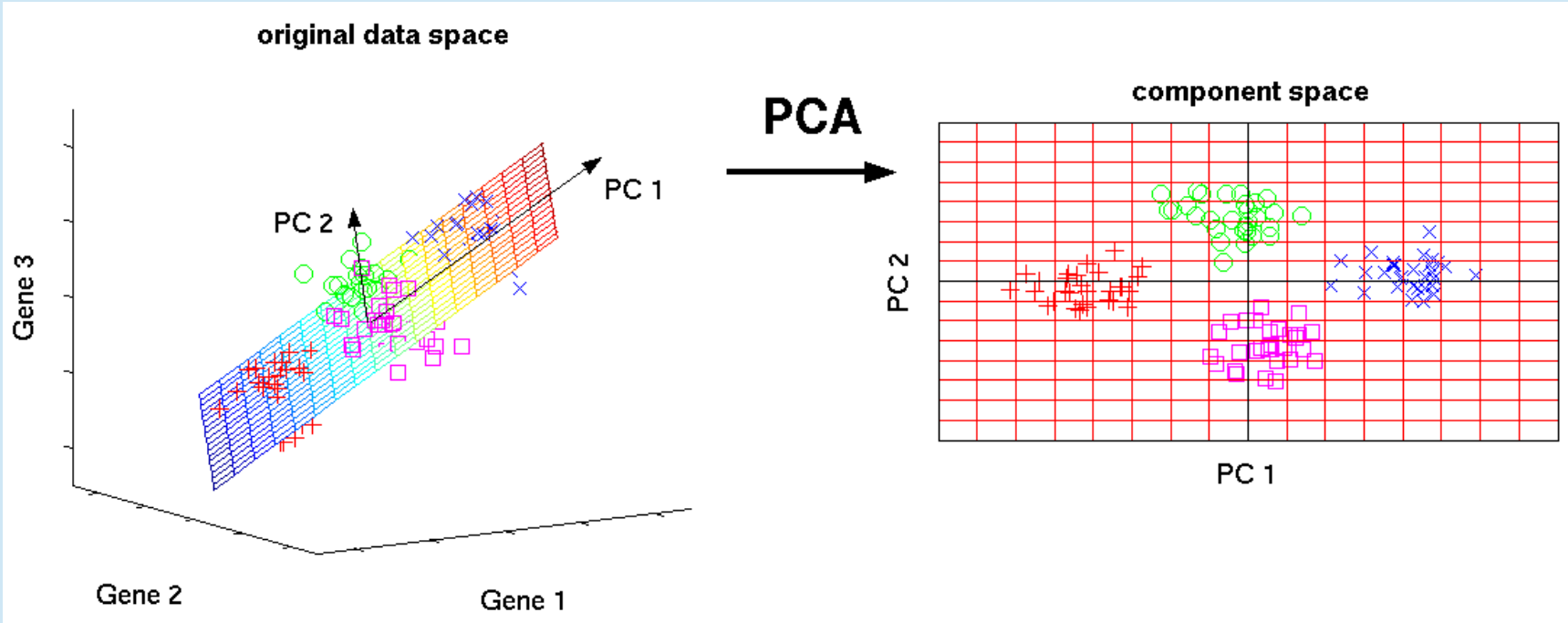
Zuzana Berger Haladová

Redukcia počtu príznakov

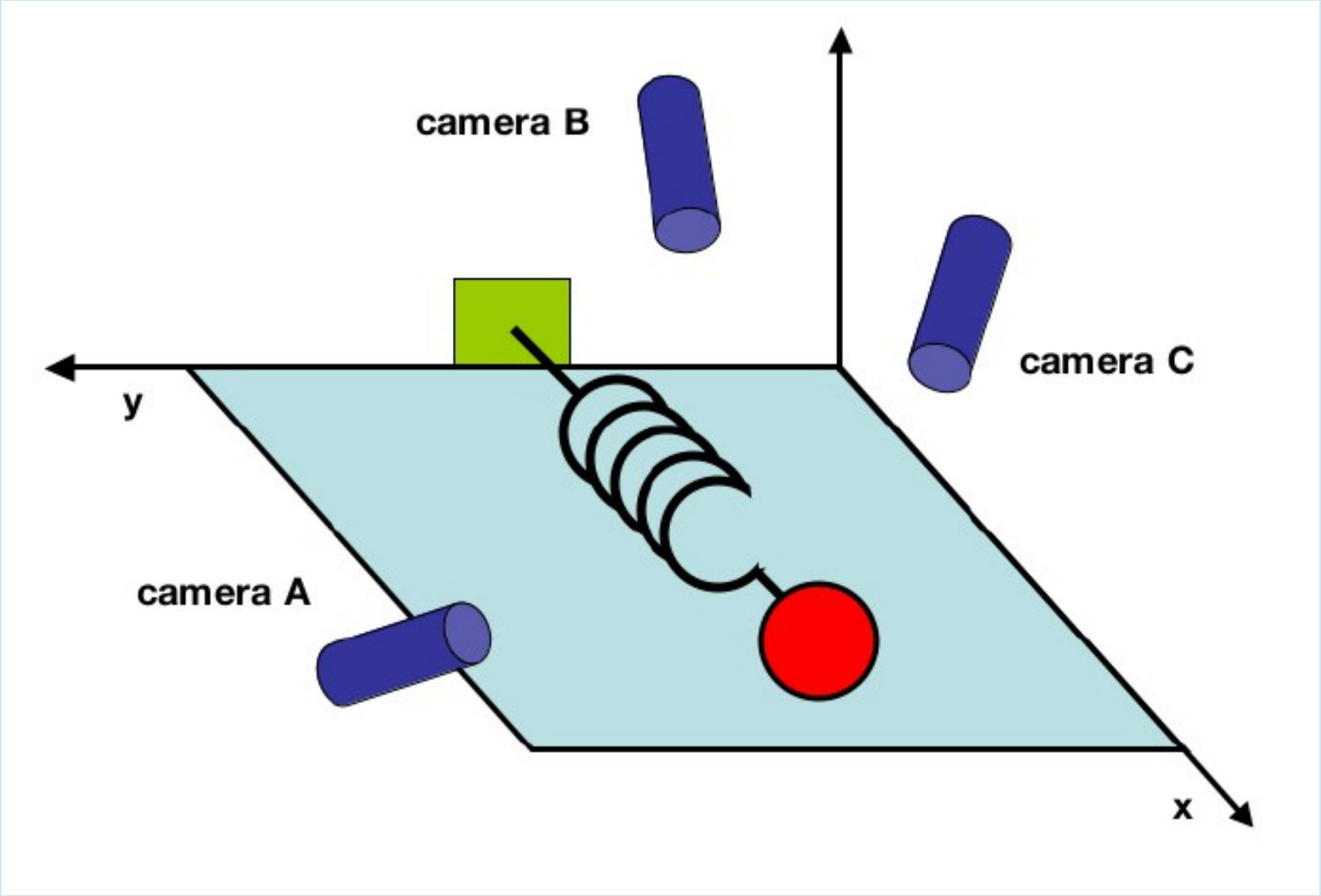
- Principal Component Analysis – PCA
- Linear Discriminant Analysis – LDA

PCA

- otočí súradnicovú sústavu tak, aby prvá os bola v smere najväčšej variability a ďalšie boli na ňu kolmé v smeroch najväčšej zvyšnej variability



PCA – motivačný príklad



PCA – motivačný príklad II

- Pohyb pružiny sa dá opísať ako rovnica s premennou *x*
- Troma kamerami budeme zaznamenávať dvojrozmernú polohu červenej gule 2 min
- Pohľady kamier nie sú na seba kolmé
- Ako môžeme z týchto dát získať rovnicu závislú iba od *x*?

PCA – motivačný príklad III

- Máme príliš veľa dimenzií problému, teda zbytočne viacrozmerné dáta
- A dáta sú ešte zaťažené šumom, lebo kamery ani pohyb pružiny nie sú dokonalé
- PCA má pomôcť inak vyjadriť zašumené a skomolené dáta a nájsť ten dôležitý smer dynamiky javu, ktorým je os *x*

Dáta v motivačnom príklade

- 12 tisíc 6 rozmerných vektorov (za každú kameru dve súradnice)
- Vo všeobecnosti máme *n* záznamov *m*-rozmerných vektorov
- Hľadáme lineárnu transformáciu *P*, ktorá zobrazuje pôvodné dáta *X* na lepšie vyjadrené dáta *Y*, tak že platí $PX = Y$

PCA cez Kovariančnú maticu

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{(n-1)}$$

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

$$C^{n\times n} = (c_{i,j},\ c_{i,j} = cov(Dim_i, Dim_j)),$$

Príklad:

1. Vypočítajte Kovariančnú maticu pre dáta (2 vzorky, 3 príznaky)X=[1,2] Y=[2,1] Z=[1,3]

2. Vypočítajte vlastné vektory a čísla pre maticu [2 -4; -1 -1]

Vlastné čísla a vektory

$$\mathbf{A} = \begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix}$$

$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{x} = 0$$

$$p(\lambda) = \begin{vmatrix} 2-\lambda & -4 \\ -1 & -1-\lambda \end{vmatrix} =$$
$$= (2-\lambda)(-1-\lambda) - (-4)(-1) =$$
$$= \lambda^2 - \lambda - 6 =$$
$$= (\lambda - 3)(\lambda + 2)$$

$$\begin{bmatrix} 2-3 & -4 \\ -1 & -1-3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 2-(-2) & -4 \\ -1 & -1-(-2) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$-x_1 - 4x_2 = 0$$
$$-x_1 - 4x_2 = 0$$

$$4x_1 - 4x_2 = 0$$
$$-x_1 + x_2 = 0$$

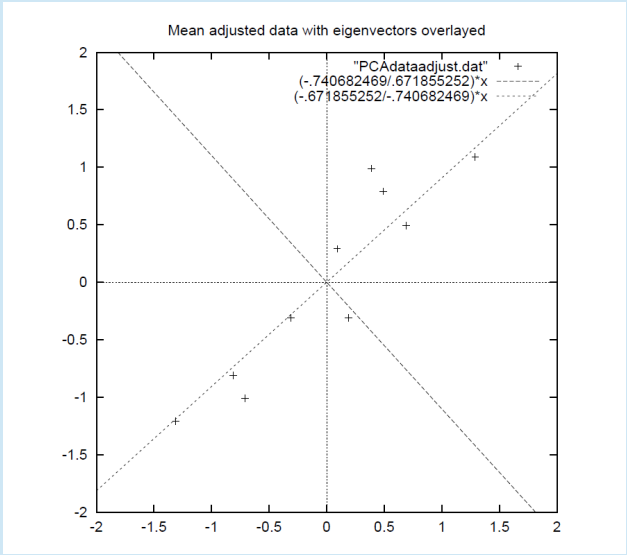
$$x_1 = -4x_2$$

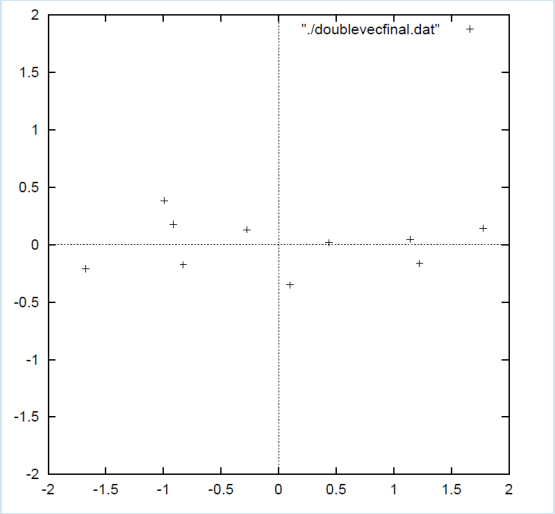
$$x_1 = x_2$$

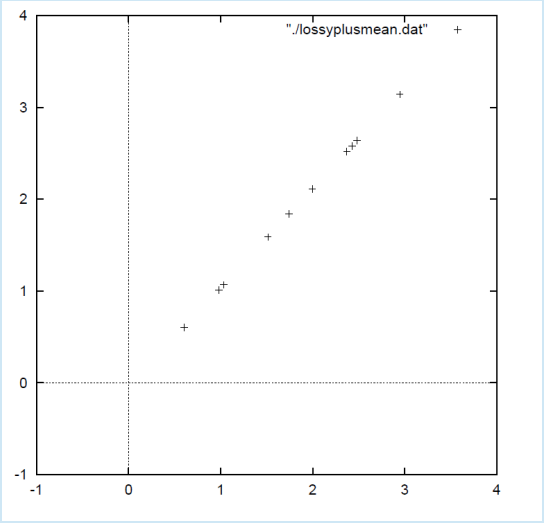
$$\mathbf{x_1} = \begin{bmatrix} 4 \\ -1 \end{bmatrix}, \quad \mathbf{x_2} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Sumarizácia predpokladov

- Hlavné komponenty sú na seba kolmé
- Ako sa dospeje k redukcii príznakov?







PCA MATLAB – príklad

```
X1=[1,0.5;2,1;3,1;4,2;5,3];

X2=[1,2;2.5,3;3,5;2.5,4;4,5];

X=[X1;X2];


data=X-M;   %M = Mean

[W, EvalueMatrix] = eig(cov(data));

 pc = data*W';
```

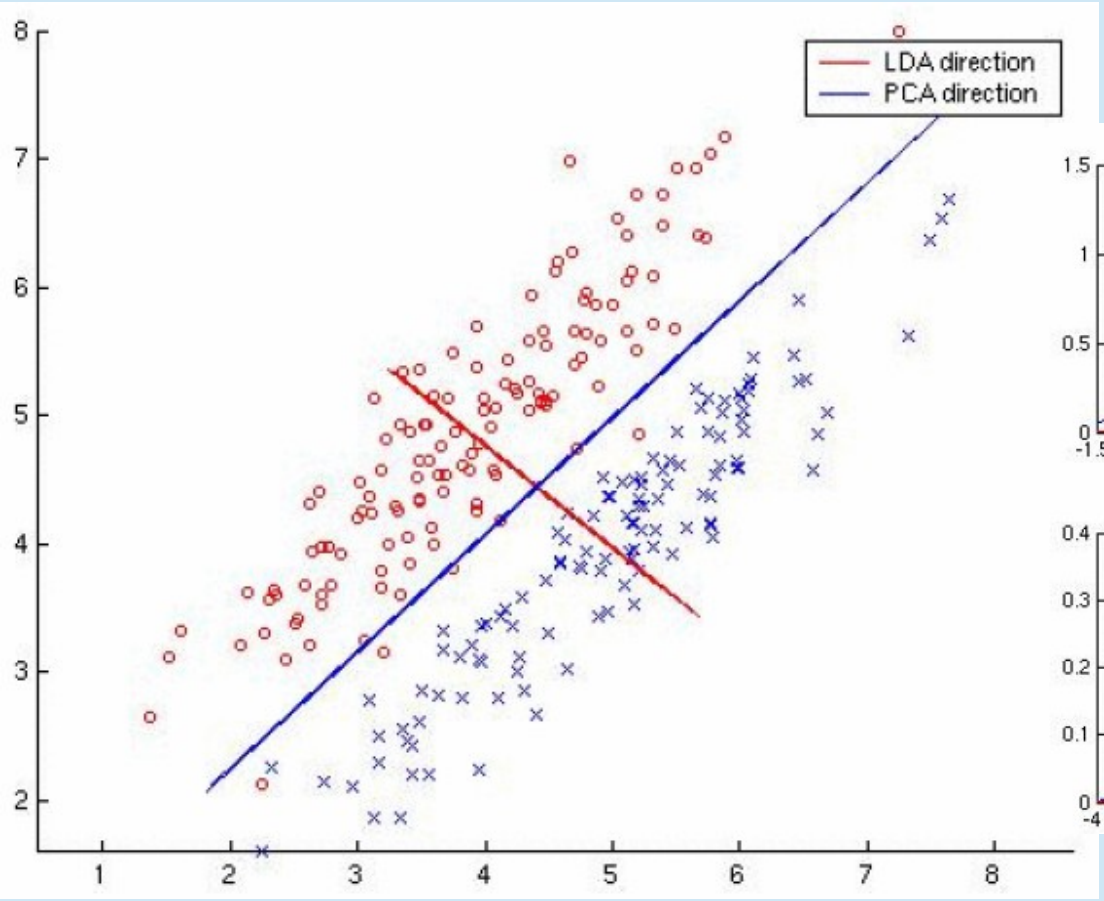
PCA MATLAB

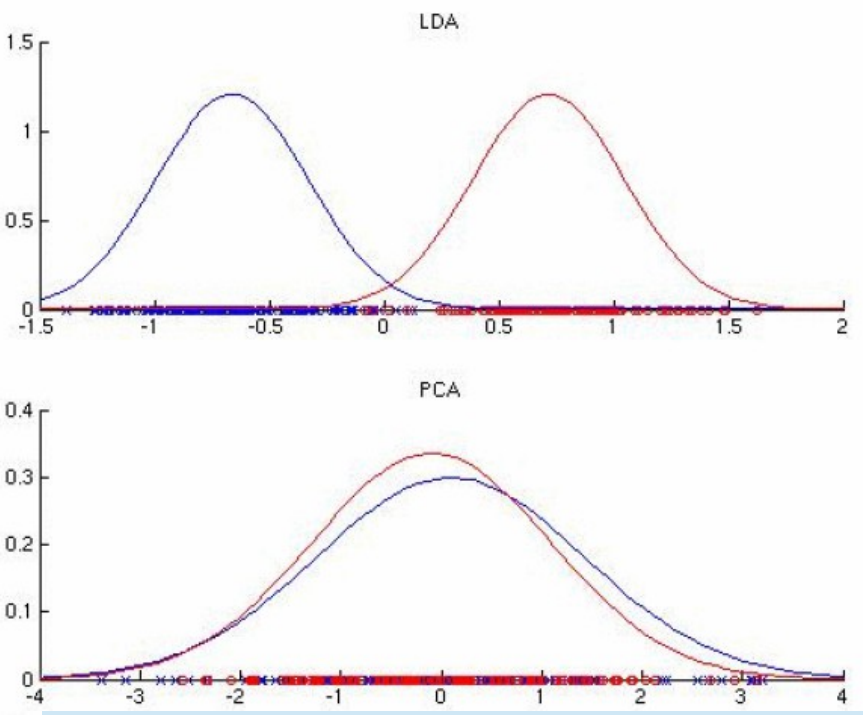
- [COEFF,SCORE] = princomp(X)
- [COEFF,SCORE,latent] = princomp(X), kde COEFF je usporiadaná matica hlavných komponentov, SCORE je vyjadrenie pôvodných dát v novej báze a *latent* obsahuje vlastné hodnoty kovariančnej matice ako ich príspevok ku variancii dát

PCA MATLAB – príklad II

```
figure, plot(X(:,1), X(:,2),'.');
[COEFF,SCORE,latent] = princomp(X)
figure, plot(SCORE(:,1), SCORE(:,2),'r.');
```

PCA vs. LDA

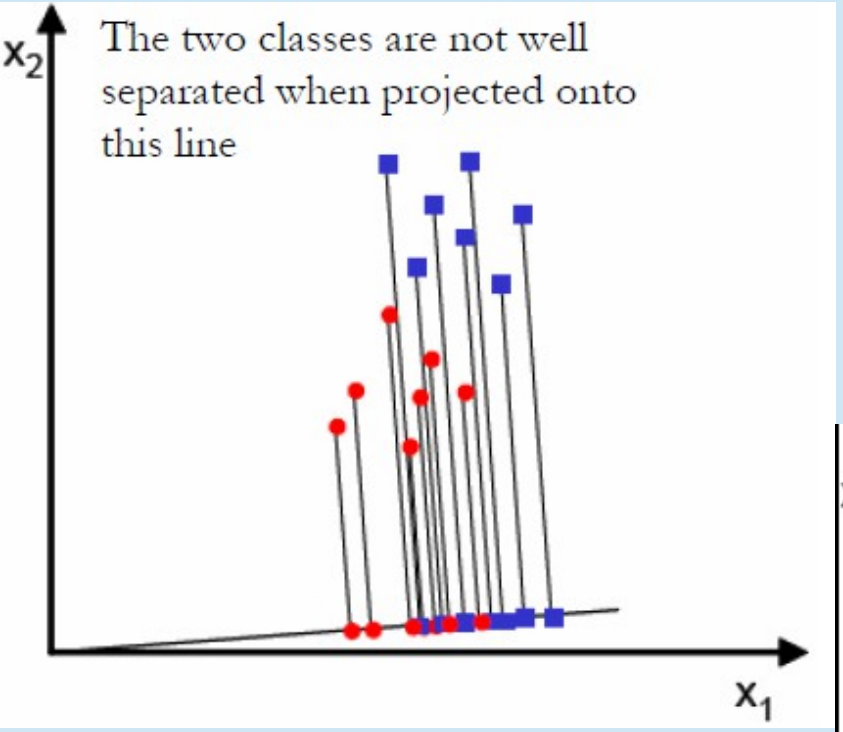


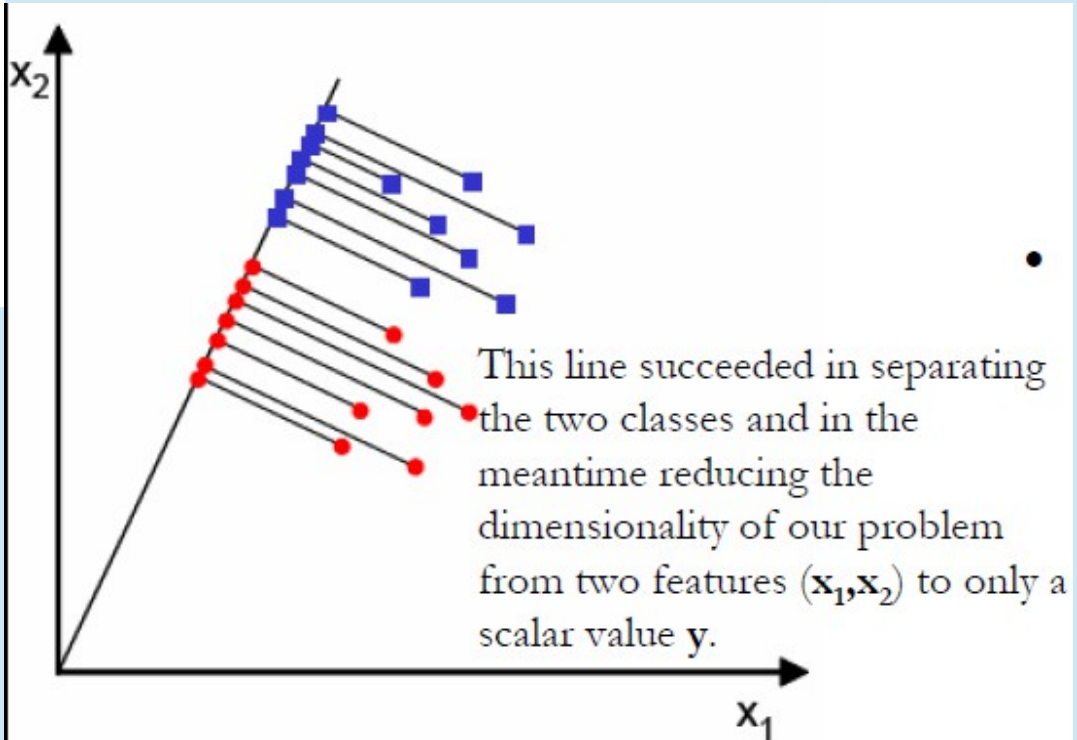


PCA vs. LDA II

- PCA nepotrebuje informáciu o zaradení jednotlivých meraní do tried
- LDA ju potrebuje, pretože maximalizuje medzitriednu vzdialenosť a minimalizuje vnútrotriednu vzdialenosť
- Ak je *C* tried, tak LDA transformuje *m*-rozmerný priestor na *C-1* -rozmerný

LDA pre dve triedy

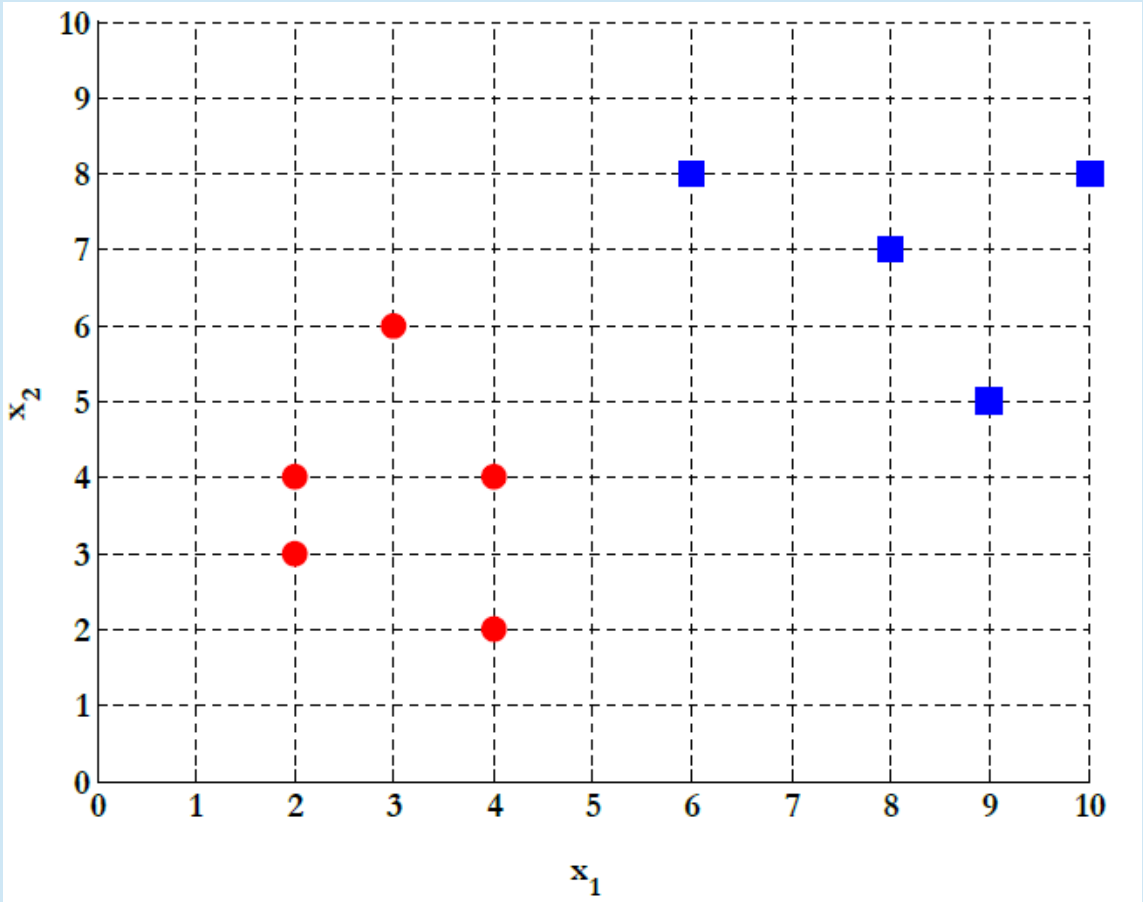




LDA pre dve triedy - príklad

Samples for class ω_1 : $\mathbf{X_1}=(x_1,x_2)=\{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class ω_2 : $\mathbf{X_2}=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$



LDA pre dve triedy – príklad II

```
% samples for class 1
X1 = [4,2;
      2,4;
      2,3;
      3,6;
      4,4];


% samples for class 2
X2 = [9,10;
      6,8;
      9,5;
      8,7;
      10,8];
```

$$\mu_3 = p_1 \times \mu_1 + p_2 \times \mu_2$$

$$S_w = \sum_j p_j \times (cov_j)$$

$$S_b = \sum_j (\mu_j - \mu_3) \times (\mu_j - \mu_3)^T$$

$$criterion = inv(S_w) \times S_b$$

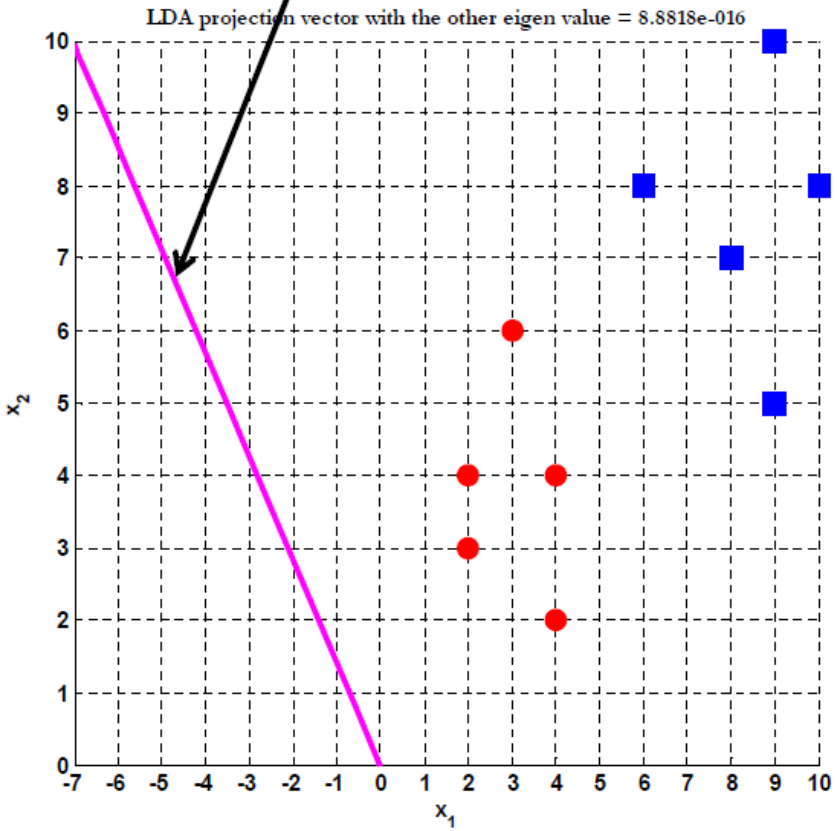
LDA pre dve triedy – príklad III

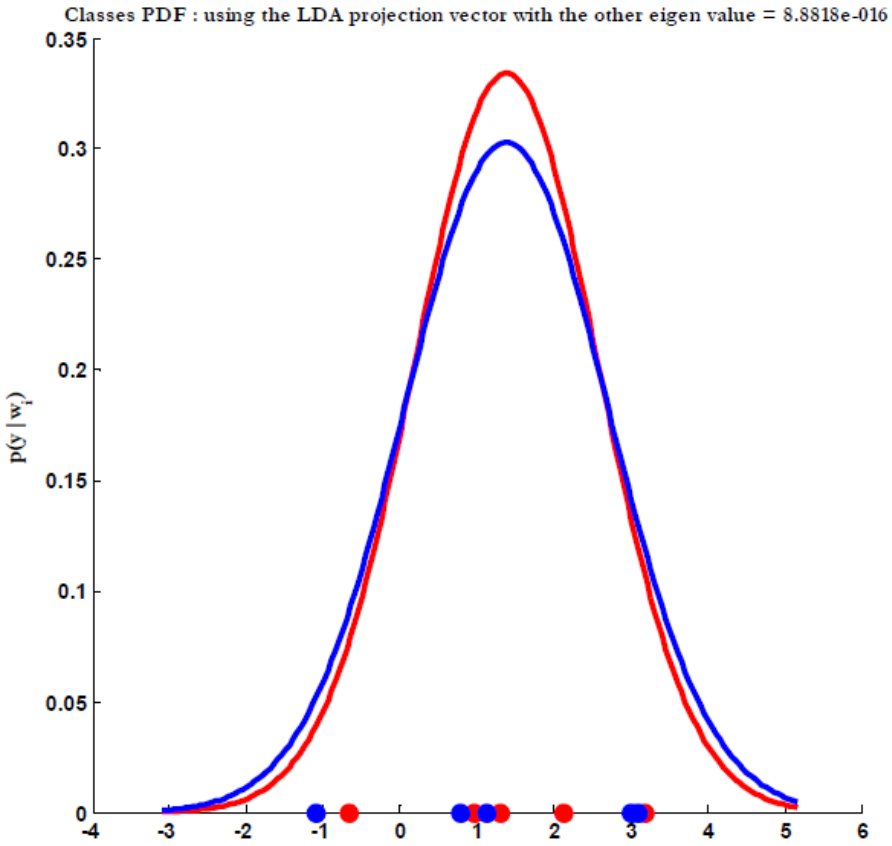
```
% computing the LDA projection
invSw = inv(Sw);

invSw_by_SB = invSw * SB;

% getting the projection vector
[V,D] = eig(invSw_by_SB)

% the projection vector
W = V(:,1);
```

$$w_1 = \begin{pmatrix} -0.5755 \\ 0.8178 \end{pmatrix} \quad and \quad w_2 = \begin{pmatrix} 0.9088 \\ 0.4173 \end{pmatrix} = w^*$$

LDA pre dve triedy – príklad IV

The projection vector corresponding to the **smallest** eigen value

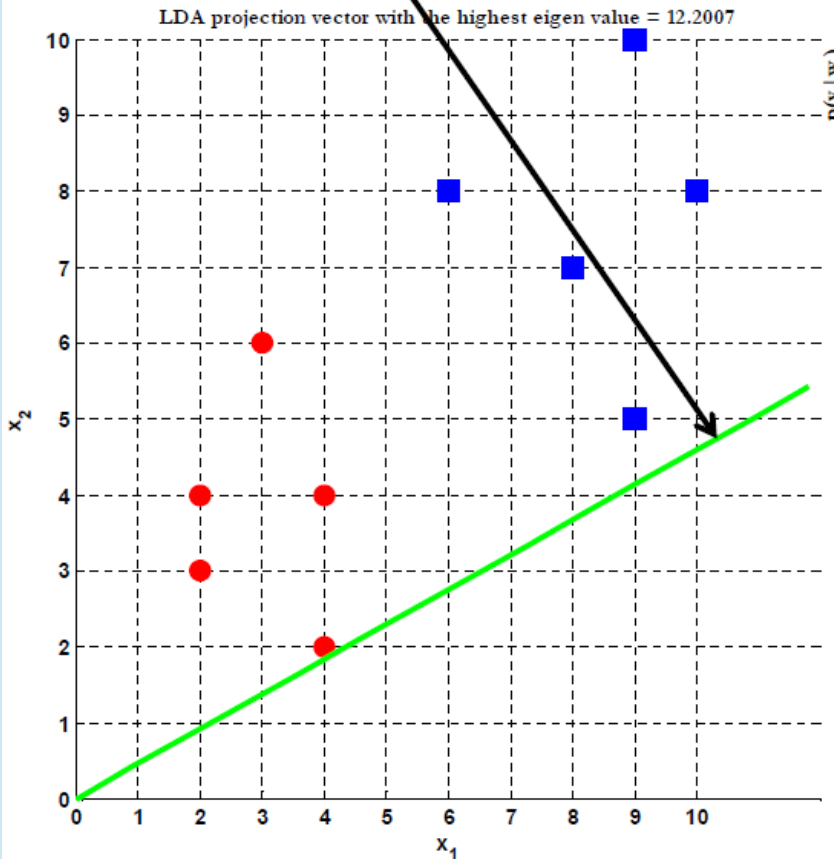


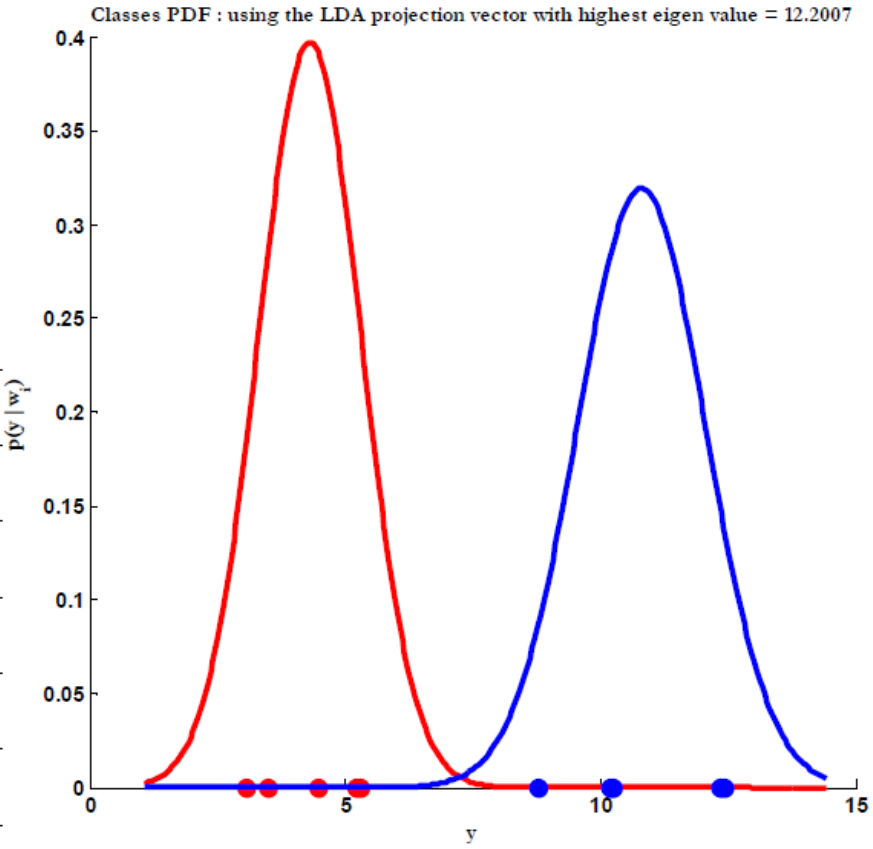


Using this vector leads to **bad separability** between the two classes

LDA pre dve triedy – príklad V

The projection vector corresponding to the **highest** eigen value





Using this vector leads to **good separability** between the two classes

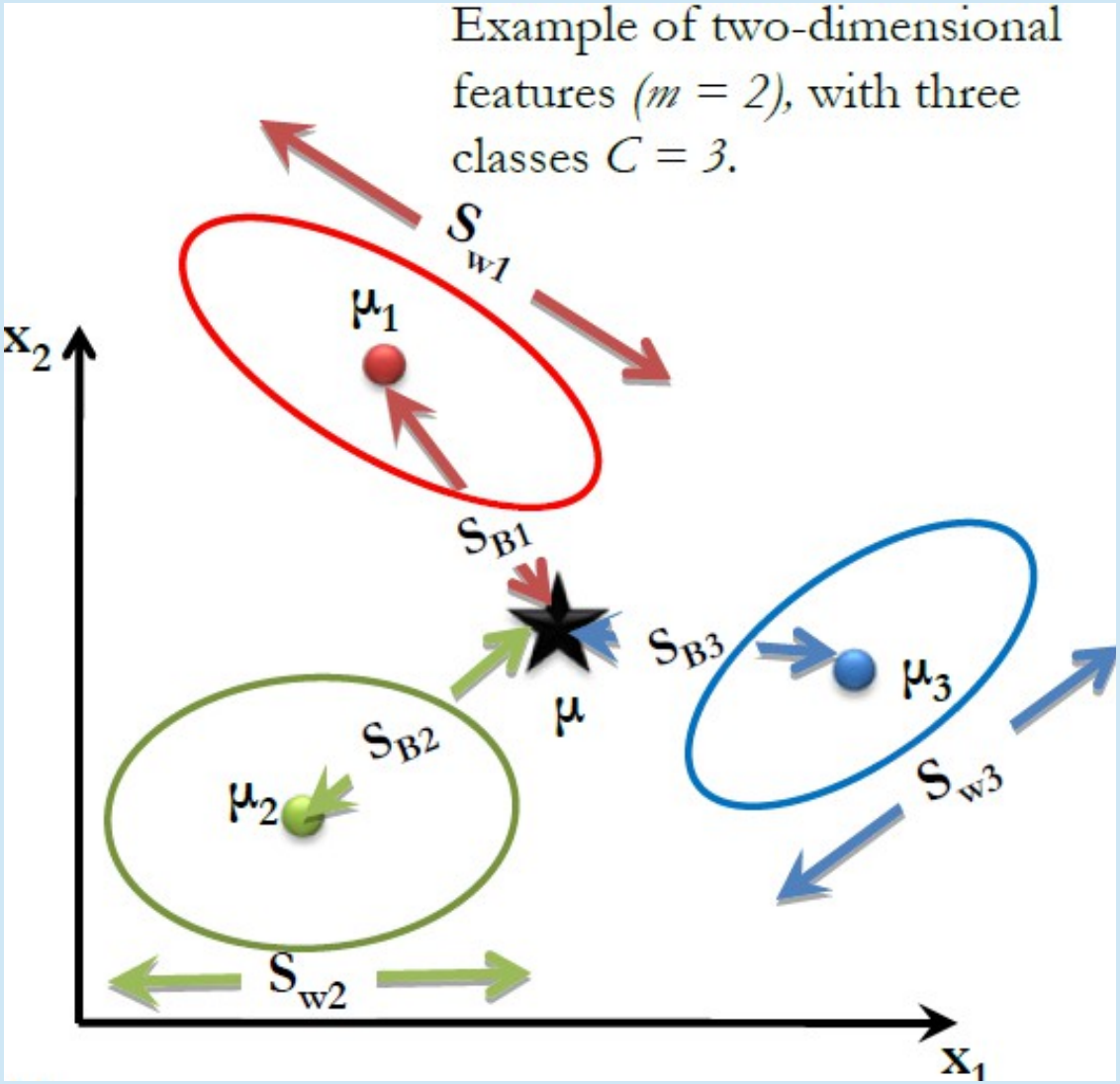
LDA pre dve triedy – príklad V

X1=[1,0.5;2,1;3,1;4,2;5,3];

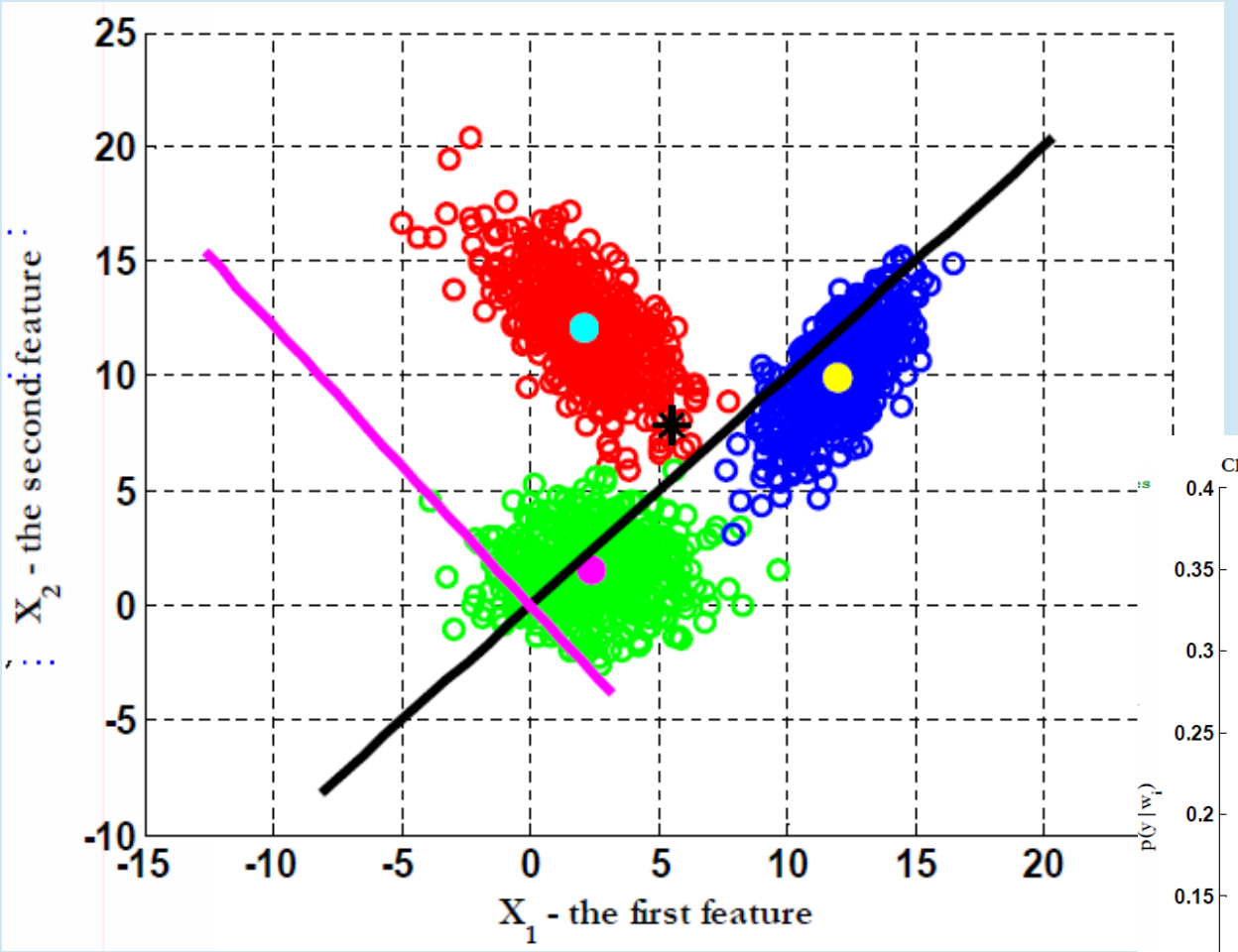
X2=[1,2;2.5,3;3,5;2.5,4;4,5];

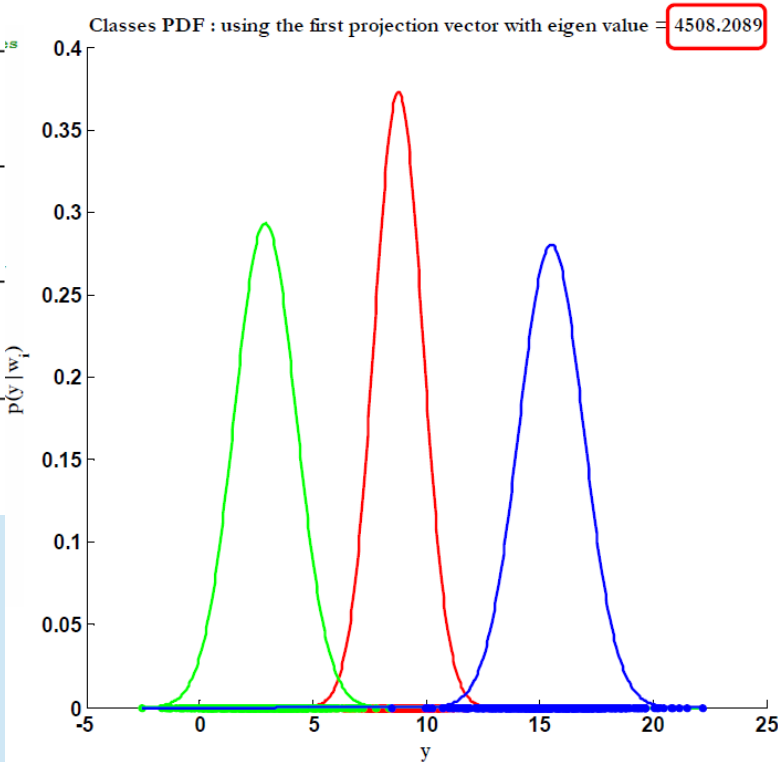
Porovnajte PCA a LDA

LDA pre C tried



LDA pre C tried II





Toolboxy

- Matlab Toolbox for Dimensionality Reduction
  - http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html