KMI/ZZD – Získávání znalostí z dat

Redukce dimenze dat

Jan Konečný

5. května 2015

Extrakce rysů a redukce dimenzionality

Předzpracování dat může zahrnovat projekci původních objektů do transformovaného prostoru. Toto je často provázeno *redukcí dimenzionality*, kdy jsou extrahovány jen ty nejinformativnější rysy.

Redukce dimenzionality původních dat je transfomace původních n-dimenzionálních vektorů na jiný n-dimenzionální vektor ($m \leq n$).
Proč:

- Odstranění redundance v datech,
- Komprese datasetu,
- Získání transformovaných a redukovaných vzorů obsahujících pouze relevantní množinu rysů, která pomůže navrhnout klasifikátory s lepší schopností zobecňování.
- Objevení podstatných proměnných, které pomůžou navrhnout model dat, a zlepšení porozumění základním faktorům, které generují objekty.
- Můžeme visuálně objevit shluky a jiné zajímavé vztahy v datech.

Metody extrakce rysů, které probereme:

- Principal component analysis (PCA) *unsupervised*
- Fishers's linear discriminant analysis *supervised*

Analýza hlavních komponent

(Principal component analysis, PCA)

Pravděpodobně nejpopulárnější statistická metoda lineární transformace vzorů a extrakce rysů.

Založena na statistické charakteristice daného datasetu representované

- *kovarianční maticí*,
- jejími *vlastními hodnotami* a *vlastními vektory*.

Vlastní vektory jsou pak v klesajícím pořadí (podle velikosti odpovídajících vlastních hodnot) usp. jako řádky transformační matice $\mathbf{W}$.

PCA určuje optimální lineární transformaci $\vec{y} = \mathbf{W}\vec{x}$,
kde $\vec{x} \in \mathbb{R}^n$, $\vec{y} \in \mathbb{R}^m$.

$m \times n$ transformační matice $W \in \mathbb{R}^{m \times n}$ je optimální v tom smyslu, že dostáváme maximální uchování informace (vzhledem k LSE)

PCA uvažujeme jako učení bez učitele (unsupervised learning).

Statistické charakteristiky dat vyžadované PCA

Uvažujme objekty charakterizované sloupcovými vektory $\vec{x} \in \mathbb{R}^n$. Předpokládejme, že máme N vektorů $\vec{x}_i$ v trénovacím datasetu

$$T_{\text{tra}} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$$

Celý trénovací dataset bude reprezentovaný jako $N \times n$ matice.

$$\mathbf{X} = \begin{bmatrix} (\vec{x}_1)^T \\ (\vec{x}_2)^T \\ \vdots \\ (\vec{x}_N)^T \end{bmatrix}$$

Data mohou být charakterizována n-dimenzionálním *středním vektorem* (mean vector)

$$\vec{\mu} = E[\vec{x}] = [E[x_1], E[x_2], \ldots, E[x_n]]^T$$

a čtvercová $n \times n$-dimenzionální *kovarianční matice*

$$\mathbf{R}_{xx} = E[(\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T],$$

kde $E[.]$ označuje operátor očekávání.

Čtvercová, positivně semidefinitní, symetrická, reálná kovarianční matice $\mathbf{R}_{xx}$ popisuje korelace mezi prvky vektorů $\vec{x}_i$.

Opravdové hodnoty $\vec{\mu}$ a $\mathbf{R}_{xx}$ nejsou v praxi k dispozici.

Proto bereme odhady:

$$\hat{\vec{\mu}} = \frac{1}{N}\sum_{i=1}^{N}\vec{x}_i$$

a

$$\hat{\mathbf{R}}_{xx} = \frac{1}{N-1}\sum_{i=1}^{N}(\vec{x}_i - \vec{\mu})(\vec{x}_i - \vec{\mu})^T$$

založené na $\mathbf{X}$.

Pro data s nulovým průměrem, odhad kovariance bude

$$\hat{\mathbf{R}}_{xx} = \frac{1}{N-1}\sum_{i=1}^{N}\vec{x}_i(\vec{x}_i)^T = \hat{\mathbf{R}}_{xx} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}$$

Další podstatná charakteristika dat $\mathbf{X}$ je množina n vlastních hodnot λ_i a odpovídající vlastní vektory $\vec{e}_i$ kovarianční matice $\mathbf{R}_{xx}$:

$$\mathbf{R}_{xx}\vec{e}_i = \lambda_i \vec{e}_i, \quad i = 1, 2, \ldots, n$$

Uvažujeme ortogonální vektory (protože $\mathbf{R}_{xx}$ je symetrická a reálná). Tzn.

$$(\vec{e}_i)^T \vec{e}_j = \begin{cases} 0 & i \neq j, \\ 1 & i = j. \end{cases}$$

V PCA je důležité mít vlastní hodnoty $\mathbf{R}_{xx}$ seřazeny v nerostoucím pořadí

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0.$$

Odpovídající ortonormální vlastní vektory $\vec{e}_i$ budou poskládány do $n \times n$ matice

$$\mathbf{E} = [\vec{e}_i, \vec{e}_2, \ldots, \vec{e}_n],$$

kde i-tý sloupec reprezentuje vlastní vektor $\vec{e}_i$ odpovídající vlastní hodnotě λ_i.

Rovnice problému vlastních hodnot může být zapsána maticově jako

$$\mathbf{R}_{xx}\mathbf{E} = \mathbf{E}\mathbf{\Lambda},$$

kde $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

Protože pro ortogonální matici $\mathbf{E}$ máme

$$\mathbf{E}^T\mathbf{E} = \mathbf{I} \quad \text{a} \quad \mathbf{E}^{-1} = \mathbf{E}^T.$$

Z předchozího můžeme vytvořit tzv. *ortogonální podobnostní transformaci*

$$\mathbf{E}^{-1}\mathbf{R}_{xx}\mathbf{E} = \mathbf{E}^T\mathbf{R}_{xx}\mathbf{E} = \mathbf{\Lambda}$$

a následně *spektrální faktorizaci* kovarianční matice $\mathbf{R}_{xx}$

$$\mathbf{R}_{xx} = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T$$

Optimalizační kritérium PCA

Cílem PCA je najít lineární transformaci

$$\vec{y} = \mathbf{W}\vec{x}$$

původního n-dimenzionálního datapointu $\vec{x}$ do m-dimenzionálního vektoru $\vec{y}$ s možností $m < n$.

Formálněji: PCA je statický optimalizační problém, se specificky definovaným optimalizačním kritériem, které zaručí, že transformovaný vektor bude mít vyžadované charakteristiky:

- optimální transformace je ortogonální (s ortonormální bází),
- prvky transformovaného vektoru $\vec{y}$ jsou nekorelované,
- ortonormální báze lineárních projekcí ukazuje (v klesajícím pořadí), ortogonální směry, ve kterých jsou změny v datech (variance) maximální,
- chyba v rekonstrukci bude minimální ve smyslu nejmenších čtverců.

Pro lineární transformaci $\vec{y} = \mathbf{W}\vec{x}$, kde $\mathbf{W}$ je $m \times n$-dimenzionální ortonormální transformační matice, odhad rekonstruovaného datapointu je $\hat{\vec{x}} = \mathbf{W}\vec{y}$.

Pro ortonormální matice máme $\mathbf{W}^{-1} = \mathbf{W}^T$, tedy

$$\hat{\vec{x}} = \mathbf{W}^{-1}\vec{y} = \mathbf{W}^T\vec{y} = \mathbf{W}^T\mathbf{W}\vec{x}$$

Rekonstrukční chyba:

$$J_{\text{lse}}(\mathbf{W}) = E[\|\vec{x} - \hat{\vec{x}}\|^2],$$

kde $\|\vec{x} - \hat{\vec{x}}\|^2 = \|\vec{x} - \hat{\vec{x}}\|_2^2$ je mocnina Euklidovské vzdálenosti.

Pro praktické výpočty můžeme využít kritérium

$$J_{\text{lse}}(\mathbf{W}) = \frac{1}{2}\sum_{i}^{N} \|\vec{x}_i - \hat{\vec{x}}_i\|^2 = \frac{1}{2}\sum_{i}^{N}\sum_{j}^{n}(x_{ij} - \hat{x}_{ij})^2$$

PCA hledá optimální transformační matici $\mathbf{W}$, která garantuje minimalizaci rekonstrukční chyby pro daný dataset.

Pro lepší pochopení cíle PCA se podívejme blíže na interpretaci PCA kritéria:

$$\begin{aligned} J_{\text{lse}}(\mathbf{W}) &= E[\|\vec{x} - \hat{\vec{x}}\|^2] \\ &= E\{\text{trace}[(\vec{x} - \hat{\vec{x}})(\vec{x} - \hat{\vec{x}})^T]\} \\ &= \text{trace}(\mathbf{R}_{xx}) - \text{trace}(\mathbf{W}\mathbf{R}_{xx}\mathbf{W}^T) \end{aligned}$$

Vidíme, že druhý term v PCA kritériu

$$J_{\text{lse}}(\mathbf{W}) = \text{trace}(\mathbf{R}_{xx}) - \text{trace}(\mathbf{W}\mathbf{R}_{xx}\mathbf{W}^T)$$

je

$$\text{trace}(\mathbf{W}\mathbf{R}_{xx}\mathbf{W}^T) = J_{\text{variance}}(\mathbf{W})$$

a je roven varianci transformovaného vektoru $\vec{y}$
a následně varianci rekonstruovaného datapointu $\hat{\vec{x}}$:

$$J_{\text{variance}}(\mathbf{W}) = \text{trace}(\mathbf{W}\mathbf{R}_{xx}\mathbf{W}^T) = E[\text{trace}(\vec{y}\vec{y}^T)] = \sum_{i=1}^{m} y_i^2$$

Závěr této analýzy: minimalizace $J_{\text{lse}}(\mathbf{W})$ je maximalizací $J_{\text{variance}}(\mathbf{W})$.

PCA možno interpretovat PCA jako

- minimalizaci rekonstrukční chyby (ve smyslu nejmenčích čtverců)
- (ekvivalentně) maximalizaci variance transformovaných vektorů

Optimální lineární transformace založené na PCA povede k projekci původního datapointu na vektor $\vec{y}$ s prvky v prostoru rysů ve směrech s maximální variancí.

Věta PCA: Nechť máme

- (trénovací) dataset $T_{\text{tra}} = \{\vec{x}^1, \vec{x}^2, \ldots, \vec{x}^N\}$ obsahující N n-dimenzionálních datapointů $\vec{x} \in \mathbb{R}^n$ s nulovým průměrem,
- symetrickou, reálnou matici $\mathbf{R}_{xx} \in \mathbb{R}^{n \times n}$,
- vlastní hodnoty matice $\mathbf{R}_{xx}$ jsou usp. v nerostoucím pořadí

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0,$$

- odpovídající ortonormální vlastní vektory $\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n$ jsou usp. v ortonormální matici $\mathbf{E} = [\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n]$.

Pak optimální lineární transformace $\hat{\vec{y}} = \hat{\mathbf{W}}\vec{x}$ transformuje původní n-dimenzionální datapointy $\vec{x}$ do m-dimenzionálních rysových vektorů s minimální rekonstrukční chybou $J_{\text{lse}}(\mathbf{W})$ a poskytuje optimální transformační matici $\hat{\mathbf{W}}$ ve tvaru

$$\hat{\mathbf{W}} = \begin{bmatrix} (\vec{e}_1)^T \\ (\vec{e}_2)^T \\ \vdots \\ (\vec{e}_m)^T \end{bmatrix}$$

Tato transformace se nazývá **Karhunen-Loéve**ova transformace (KLT).

Vlastnosti KLT

KLT transformace zaručuje minimální rekonstrukční chybu ve smyslu nejmenších čtverců s minimální hodnotou

$$\min J_{\text{lse}}(\mathbf{W}) = \sum_{i=m+1}^{n} \lambda_i$$

Současně transformace zaručuje maximální varianci

$$\max J_{\text{variance}}(\mathbf{W}) = \sum_{i=1}^{m} \lambda_i$$

hlavní vlastní vektory (principal eigenvectors)

- se nazývají vl. vektory $\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n$ (tedy řádky $\hat{\mathbf{W}}$),
- ukazují ortogonální směry, ve kterých se datapointy liší s největší variancí,
- m z nich usp. jako řádky matice dávají matici $\hat{\mathbf{W}}$.

hlavní komponenty (principal components)

- jsou prvky $y_1, y_2, \ldots, y_m$ vektoru $\vec{y} = \hat{\mathbf{W}}\vec{x}$.
- jsou statisticky nekorelované s kovariancemi

$$E[y_i y_j] = \vec{e}_i^T \mathbf{R}_{xx} \vec{e}_j = 0$$

a s variancemi rovými odpovídajícím vl. hodnotám

Optimální KLT transformace původních datapointů

Jakmile máme optimální KLT matici $\hat{\mathbf{W}}$, inverzní transformaci získáme jako

$$\hat{\vec{x}} = \hat{\mathbf{W}}^{-1}\vec{y} = \hat{\mathbf{W}}^{T}\vec{y} = \sum_{i=1}^{m} y_i \vec{e_i}.$$

Optimalní transformace celého původního datasetu $\mathbf{X}$ je dána formulí

$$\mathbf{Y} = (\hat{\mathbf{W}}\mathbf{X}^{T})^{T} = \mathbf{X}\hat{\mathbf{W}}^{T}$$

$m \times m$ kovarianční matice $\mathbf{R}_{yy} = E[\vec{y}\vec{y}^{T}]$ transformovaných vektorů může být odhadnuta jako

$$\mathbf{R}_{yy} = \frac{1}{N-1}\mathbf{Y}^{T}\mathbf{Y} = \hat{\mathbf{W}}\mathbf{R}_{xx}\hat{\mathbf{W}}^{T} = \mathrm{diag}[\lambda_i]$$

Celý zrekonstruovaný dataset je dán

$$\hat{\mathbf{X}} = \mathbf{Y}\hat{\mathbf{W}}$$

Redukce dimensionality

PCA může být efektivně použita pro extrakci rysů a redukci dimenzionality.
Namísto velého n-dimenzionálního datapointu $\vec{x}$ můžeme vytvořit m-dimenzionální ($m \leq n$) vektor $\vec{y} = [y_1, y_2, \ldots, y_m]^T$ obsahující hlavní komponenty $\vec{x}$.

PCA je nejlepší technika pro lineární extrakci rysů z původního datasetu, ve smyslu minimalizace rekonstrukční chyby.

Projekce datapointu z vysoko-dimenzionálního prostoru na redukovaný prostor hlavních komponent (dimenze $m \ll n$) povede ke kvadrátu aproximační chyby

$$\vec{e}^2 = \sum_{i=m+1}^{n} \lambda_i$$

m hlavních komponent y_i jsou nejvýznamější rysy datasetu.

Vektor aproximačních chyb $\vec{e}$ je ortogonální k rekonstruovanému datapointu $\hat{\vec{x}}$.
Chyba nejmenších čverců mezi $\vec{x}$ a $\hat{\vec{x}}$ je

$$\|\vec{e}\| = \|\vec{x} - \hat{\vec{x}}\| = \left[\sum_{i=1}^{n}(x_i - \hat{x}_i)^2\right]^{1/2}$$

Chyba nejmenších průměrných čverců je

$$E[\|\vec{x} - \hat{\vec{x}}\|^2] = \sum_{i=m+1}^{n} \lambda_i$$

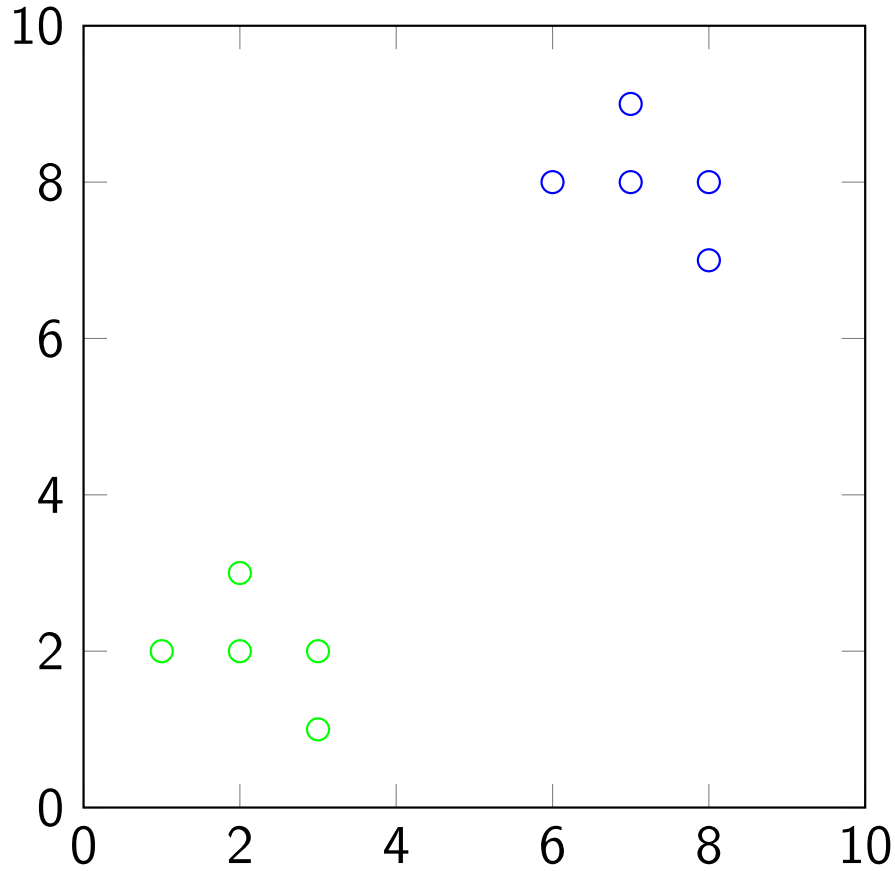
Výběr dimenze m

- vzít tolik hlavních komponent, aby rekonstrukční chyba byla menší než heuristicky nastavená hranice ϵ.
- (praktičtější) vybrat m nejdůležitějších komponent, pro které je procento V součtu nepoužitých vlastních hodnot a součtu všech hodnot

$$V = \frac{\sum_{i=m+1}^{n} \lambda_i}{\sum_{i=1}^{n} \lambda_i} 100\%$$

menší, než určená hranice ϵ.

1	2	1
2	2	1
2	3	1
3	1	1
3	2	1
6	8	2
7	8	2
8	7	2
8	8	2
7	9	2



$$\mathbf{X}_{\text{orig}} = \begin{bmatrix} 1 & 2 \\ 2 & 2 \\ 2 & 3 \\ 3 & 1 \\ 3 & 2 \\ 6 & 8 \\ 7 & 8 \\ 8 & 7 \\ 8 & 8 \\ 7 & 9 \end{bmatrix} \qquad \vec{\mu} = [4.7, 5.0]^T \qquad \mathbf{X} = \begin{bmatrix} -3.7 & -3.0 \\ -2.7 & -3.0 \\ -2.7 & -2.0 \\ -1.7 & -4.0 \\ -2.7 & -3.0 \\ 1.3 & 3.0 \\ 2.3 & 3.0 \\ 3.3 & 2.0 \\ 3.3 & 3.0 \\ 2.3 & 4.0 \end{bmatrix}$$

Kovarianční matice je

$$\mathbf{R}_{xx} = \begin{bmatrix} 7.5667 & 8.1111 \\ 8.1111 & 10.4444 \end{bmatrix}$$

Vlastní hodnoty $\mathbf{R}_{xx}$ jsou

$$\lambda_1 = 17.2433, \quad \lambda_2 = 0.7678$$

Vlastní hodnoty $\mathbf{R}_{xx}$ jsou

$$\lambda_1 = 17.2433, \quad \lambda_2 = 0.7678$$

a odpovídající vlastní vektory jsou

$$\vec{e_1} = [0.64240.7664]^T, \quad \vec{e_2} = [-0.76640.6424]^T$$

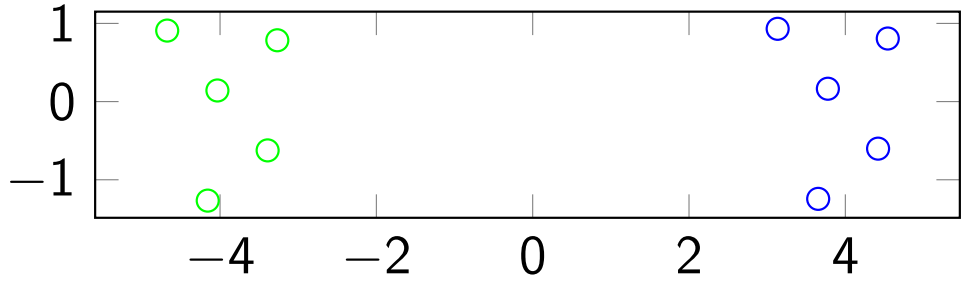
Matice $\mathbf{E}$ složená z vlastních vektorů $\vec{e_i}$

$$\mathbf{E} = \begin{bmatrix} 0.6424 & -0.7664 \\ 0.7664 & 06424 \end{bmatrix}$$

KLT transformační matice $\hat{\mathbf{W}}$ s řádky odpovídajícími vlastním hodnotám kovarianční matice v klesajícím pořadí.

$$\hat{\mathbf{W}} = \begin{bmatrix} 0.6424 & 0.7664 \\ -0.7664 & 06424 \end{bmatrix}$$

$$\mathbf{Y} = \mathbf{X}\mathbf{W}^T = \begin{bmatrix} -4.6760 & 0.9084 \\ -4.0336 & 0.1421 \\ -3.2672 & 0.7844 \\ -4.1576 & -1.2667 \\ -3.3912 & -0.6243 \\ 3.1342 & 0.9309 \\ 3.7766 & 0.1645 \\ 3.6526 & -1.2443 \\ 4.4190 & -0.6019 \\ 4.5430 & 0.8069 \end{bmatrix}$$



$$\hat{\mathbf{X}} = \mathbf{Y}\hat{\mathbf{W}} = \begin{bmatrix} -4.6760 & 0.9084 \\ -4.0336 & 0.1421 \\ -3.2672 & 0.7844 \\ -4.1576 & -1.2667 \\ -3.3912 & -0.6243 \\ 3.1342 & 0.9309 \\ 3.7766 & 0.1645 \\ 3.6526 & -1.2443 \\ 4.4190 & -0.6019 \\ 4.5430 & 0.8069 \end{bmatrix} \cdot \begin{bmatrix} 0.6424 & 0.7664 \\ -0.7664 & 06424 \end{bmatrix} = \begin{bmatrix} -3.7 & -3.0 \\ -2.7 & -3.0 \\ -2.7 & -2.0 \\ -1.7 & -4.0 \\ -2.7 & -3.0 \\ 1.3 & 3.0 \\ 2.3 & 3.0 \\ 3.3 & 2.0 \\ 3.3 & 3.0 \\ 2.3 & 4.0 \end{bmatrix}$$

Po přičtení $\vec{\mu}$ k řádkům dostaneme původní matici $\mathbf{X}_{\text{orig}}$

Redukce dimenzionality

$$\hat{\mathbf{W}} = [\vec{e}_1] = \begin{bmatrix} 0.6424 & 0.7664 \end{bmatrix}$$

$$\mathbf{Y} = \mathbf{X}\hat{\mathbf{W}}^T = \begin{bmatrix} -4.6760 \\ -4.0336 \\ -3.2672 \\ -4.1576 \\ -3.3912 \\ 3.1342 \\ 3.7766 \\ 3.6526 \\ 4.4190 \\ 4.5430 \end{bmatrix}$$

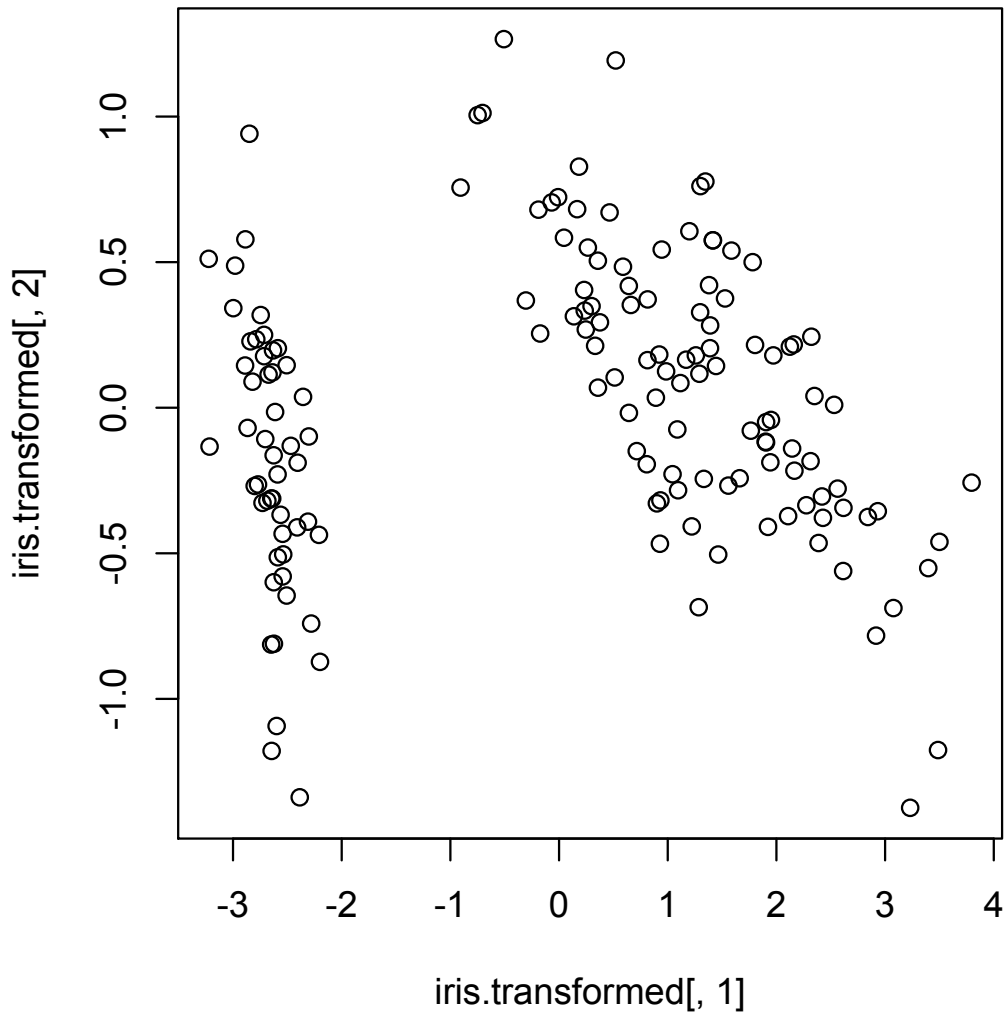
PCA v R

```
> iris.stripped  <- iris[,1:4]

> head(iris.stripped)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4

> iris.mean  <- mean(iris.stripped)

> iris.mean
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     3.758000     1.199333
```

```
> iris.centered  <- sweep(iris.stripped ,MARGIN=2,
+                          iris.mean ,FUN="-")
> head(iris.centered)
   Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
1    -0.7433333   0.44266667        -2.358   -0.9993333
2    -0.9433333  -0.05733333        -2.358   -0.9993333
3    -1.1433333   0.14266667        -2.458   -0.9993333
4    -1.2433333   0.04266667        -2.258   -0.9993333
5    -0.8433333   0.54266667        -2.358   -0.9993333
6    -0.4433333   0.84266667        -2.058   -0.7993333

> iris.cov  <- cov(iris.centered)
> iris.cov
     Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
S.L   0.68569351  -0.04243400     1.2743154   0.5162707
S.W  -0.04243400   0.18997942    -0.3296564  -0.1216394
P.L   1.27431544  -0.32965638     3.1162779   1.2956094
P.W   0.51627069  -0.12163937     1.2956094   0.5810063
```

```
> eigen(iris.cov)
$values
[1] 4.22824171 0.24267075 0.07820950 0.02383509

$vectors
            [,1]        [,2]        [,3]       [,4]
[1,]  0.36138659 -0.65658877 -0.58202985  0.3154872
[2,] -0.08452251 -0.73016143  0.59791083 -0.3197231
[3,]  0.85667061  0.17337266  0.07623608 -0.4798390
[4,]  0.35828920  0.07548102  0.54583143  0.7536574



> iris.wt <- eigen(iris.cov)$vectors
> iris.transformed  <- as.matrix(iris.centered)
+                                %*% iris.wt
> head(iris.transformed,5)
          [,1]       [,2]        [,3]         [,4]
[1,] -2.684126 -0.3193972 -0.02791483  0.002262437
[2,] -2.714142  0.1770012 -0.21046427  0.099026550
[3,] -2.888991  0.1449494  0.01790026  0.019968390
[4,] -2.745343  0.3182990  0.03155937 -0.075575817
[5,] -2.728717 -0.3267545  0.09007924 -0.061258593
```

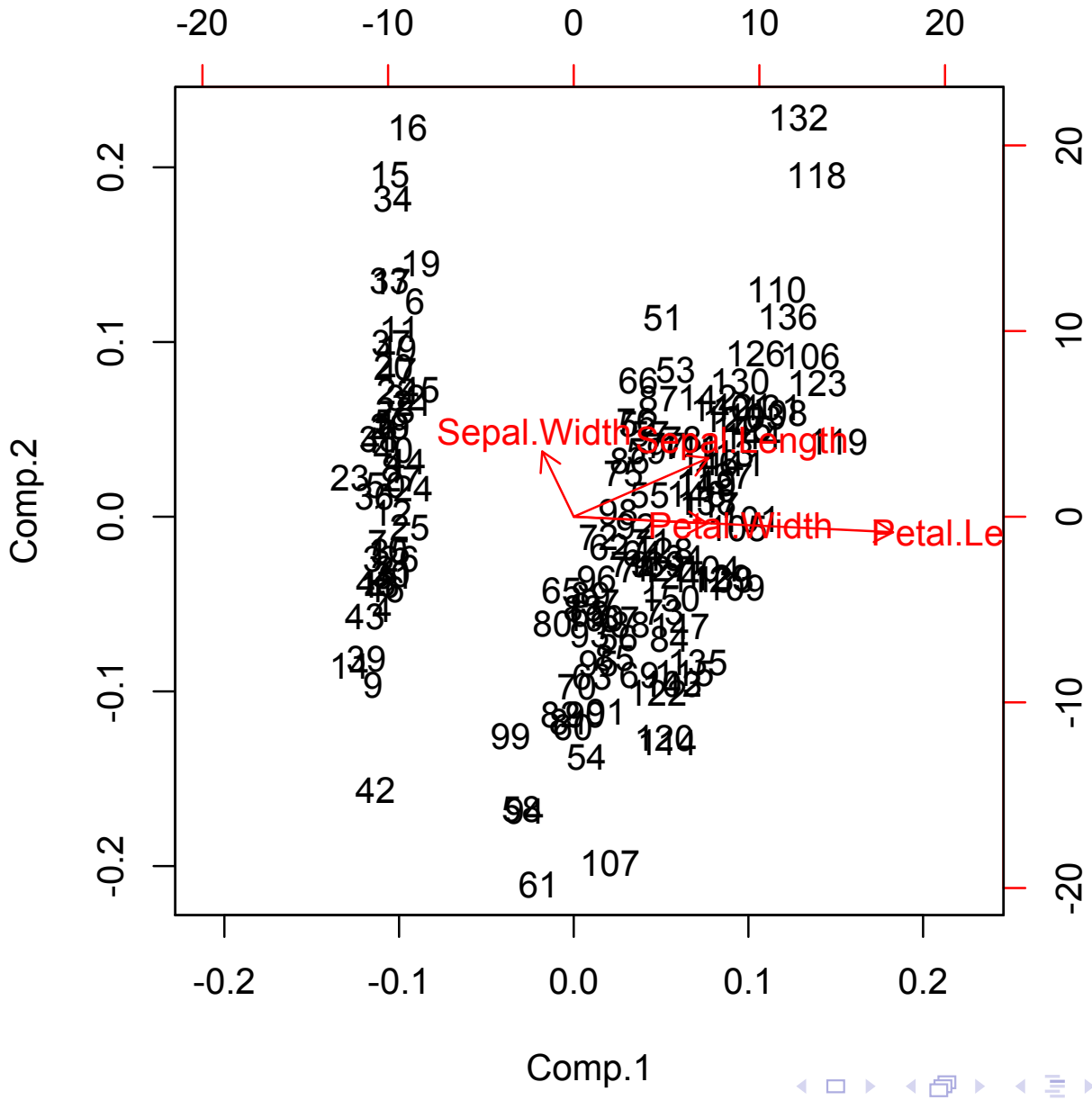
```
plot(iris.transformed[,1], iris.transformed[,2])
```



```
> iris.reconstruct <- as.matrix(iris.transformed)
+                         %*% t(iris.wt)

> head(iris.reconstruct)
           [,1]         [,2]   [,3]         [,4]
[1,] -0.7433333  0.44266667 -2.358 -0.9993333
[2,] -0.9433333 -0.05733333 -2.358 -0.9993333
[3,] -1.1433333  0.14266667 -2.458 -0.9993333
[4,] -1.2433333  0.04266667 -2.258 -0.9993333
[5,] -0.8433333  0.54266667 -2.358 -0.9993333
[6,] -0.4433333  0.84266667 -2.058 -0.7993333


> head(iris.centered)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1   -0.7433333  0.44266667        -2.358  -0.9993333
2   -0.9433333 -0.05733333        -2.358  -0.9993333
3   -1.1433333  0.14266667        -2.458  -0.9993333
4   -1.2433333  0.04266667        -2.258  -0.9993333
5   -0.8433333  0.54266667        -2.358  -0.9993333
6   -0.4433333  0.84266667        -2.058  -0.7993333
```

```
> iris.pca <- princomp(iris.centered)
> iris.pca
Call:
princomp(x = iris.centered)

Standard deviations:
   Comp.1    Comp.2    Comp.3    Comp.4
2.0494032 0.4909714 0.2787259 0.1538707

 4  variables and  150 observations.

> head(iris.pca$scores, 3)
        Comp.1     Comp.2      Comp.3       Comp.4
[1,] -2.684126  0.3193972  0.02791483  0.002262437
[2,] -2.714142 -0.1770012  0.21046427  0.099026550
[3,] -2.888991 -0.1449494 -0.01790026  0.019968390

> head(iris.transformed,3)
          [,1]       [,2]        [,3]         [,4]
[1,] -2.684126 -0.3193972 -0.02791483  0.002262437
[2,] -2.714142  0.1770012 -0.21046427  0.099026550
[3,] -2.888991  0.1449494  0.01790026  0.019968390
```

```
biplot(iris.pca)
```



Fisherova lineární diskriminantová analýza (Fisher's Linear Discriminant Analysis)

Fisherova lineární diskriminantová analýza

- je klasická metoda extrakce rysů a redukce dimenze.
- je metoda učení s učitelem (supervised learning) je konstruována na základě datasetu T_{tra} o N olabelovaných datapointech.
- je lineární transformace založená statistických charakteristikách daného datasetu reprezentované **maticí rozptylu** (scatter matrix).

Uvažujeme, že T_{tra} obsahuje

- N dvojic $(\vec{x}^i, c^i_{\text{target}}), \quad (i = 1, 2, \ldots, N), \vec{x}^i \in \mathbb{R}^n$.
- N_i dvojic z třídy c_i, $(N = \sum_i N_i)$

FLDA určuje optimální lineární trasformaci transformaci

$$\vec{y} = \mathbf{W}\vec{x}$$

Transformační matice $\mathbf{W}$ je navržena optimálně tak, aby zajistila co největší separabilitu mezi třídami.
To umožňuje najít kompatní reprezentaci dat, která umožňuje navrhnout lepší klasifikátory.

Navrhnout Fisherovu lineární transformaci je statický optimalizační problém.
Optimalizační kritérium J_{fisher} je zvoleno tak, aby byla nalezena maximální separabilita mezi třídami.

Vyhodnocení separability mezi třídami je založeno na matici rozptylu odhadnuté na základě daných dat.
FLT je konstruována tak, aby

- transformovala datapointy do prostoru s nižší dimenzí,
- zajišťovala co nejmenší rozptyl uvnitř tříd,
- zajišťovala co největší rozptyl mezi třídami.

Data s 2ma třídami a Fisherova projekce na osu

Nejdříve jednodušší verze

Uvažujeme dataset s dvěma třídami a analyzujeme lineární transformaci n-dimenzionálních vektorů $\vec{x} \in \mathbb{R}^n$ na 1-dimenzionální prostoru – osu, s objekty $y \in \mathbb{R}$.

Tato transformace má podobu

$$y = \vec{w}^T \vec{x},$$

kde $\vec{w} = [w_1, w_2, \ldots, w_n]^T$ je n-dimenzionální tranformační vektor.

y je extrahovaný rys.

Budeme uvažovat FLT konstruovanou na základě datasetu T_{tra}.

T_{tra} obshauje N labelovaných vektorů

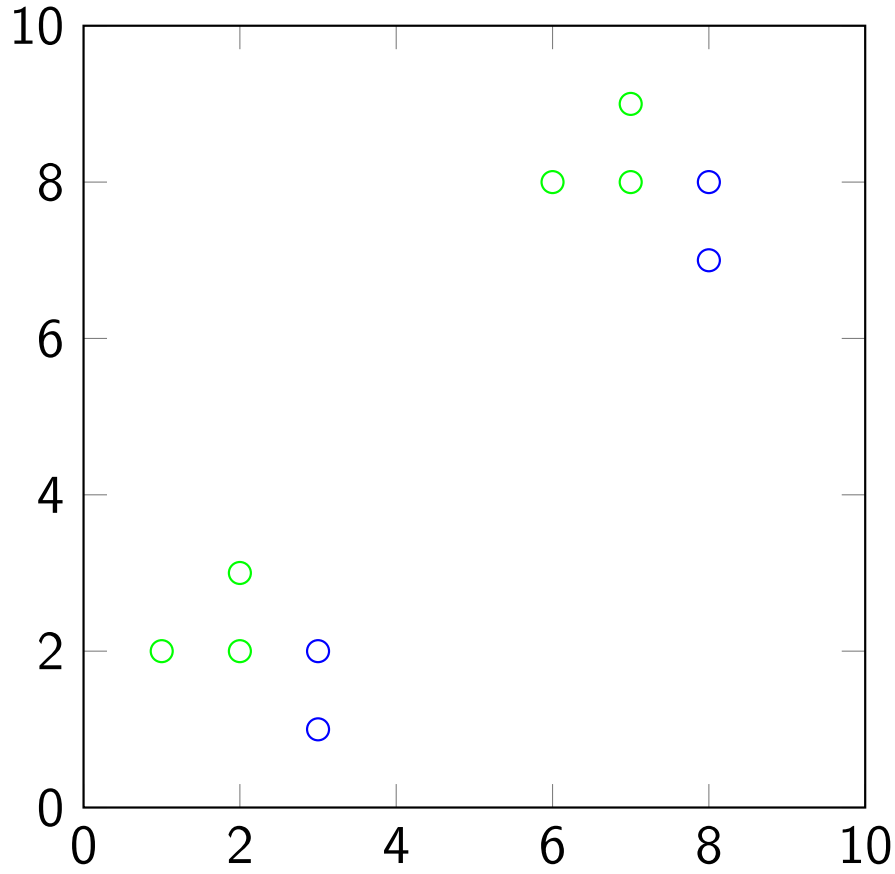
$$(\vec{x}_i, c^i_{\text{target}}) \quad i = 1, 2, \ldots, N$$

kde

- $\vec{x}_i$ je n-dimenzionální vektor,
- $c^i_{\text{target}} \in \{c_1, c_2\}$,
- T_{tra} obs. N_1 vektorů s labelem c_1 a N_1 vektorů s labelem c_2.

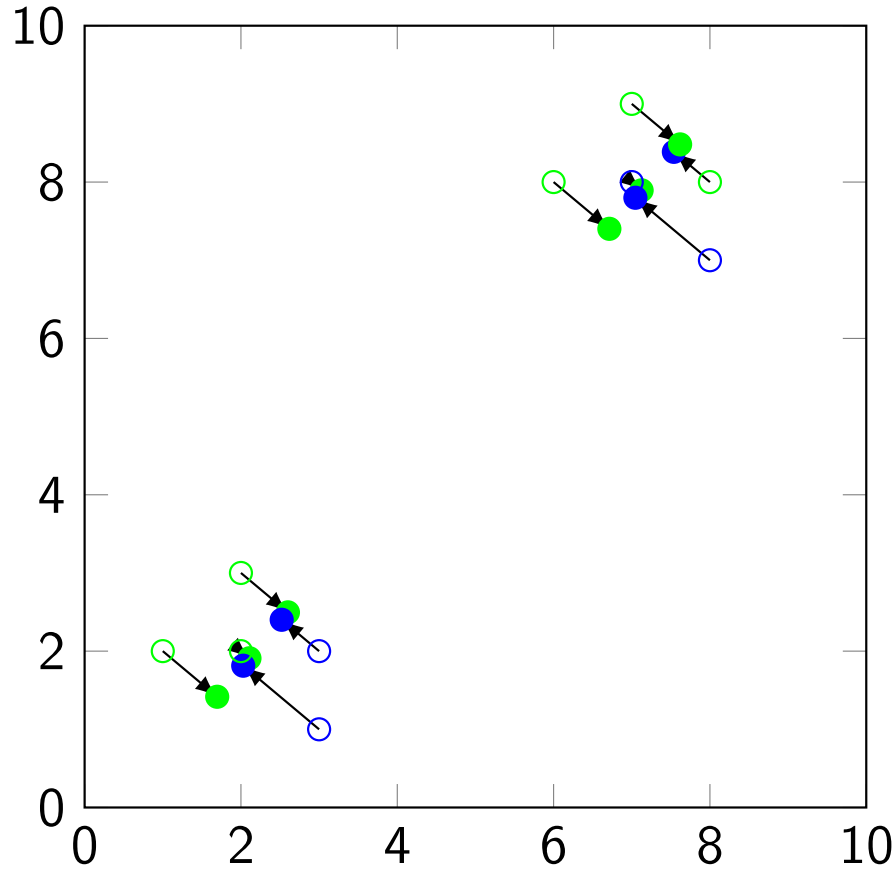
Stejné, zadání jako předtím, jen třídy jsou teď jinak

1	2	1
2	2	1
2	3	1
3	1	2
3	2	2
6	8	1
7	8	1
8	7	2
8	8	2
7	9	1



Po redukci dimenzionality přes PCA

$$\hat{\mathbf{X}} = \begin{bmatrix} -3.0038 & -3.5836 \\ -2.5911 & -3.0913 \\ -2.0988 & -2.5039 \\ -2.6708 & -3.1863 \\ -2.1785 & -2.5989 \\ 2.0134 & 2.4020 \\ 2.4261 & 2.8943 \\ 2.3464 & 2.7993 \\ 2.8387 & 3.3866 \\ 2.9184 & 3.4817 \end{bmatrix}$$



Statistické charakteristiky datapointů z původního nD datasetu T_{tra}

Průměr $\vec{\mu}_i$ pro každou třídu c_i může být odhadnut jako

$$\vec{\mu}_i = \frac{1}{N_i} \sum_{\vec{x} \in c_i} \vec{x}, \quad (i = 1, 2)$$

Matice rozptylu v třídě c_i

$$\mathbf{S}_i = \sum_{\vec{x} \in c_i} (\vec{x} - \vec{\mu}_i)(\vec{x} - \vec{\mu}_i)^T, \quad (i = 1, 2)$$

Matice rozptylu v třídách

$$\mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2$$

Matice celkového rozptylu

$$\mathbf{S}_t = \sum_{j=1}^{N} (\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T$$

Celkový průměr $\vec{\mu}$ může být odhadnut jako

$$\vec{\mu} = \frac{1}{N} \sum_{j=1}^{N} \vec{x}^j = \frac{1}{N}(N_1 \vec{\mu}_1 + N_2 \vec{\mu}_2)$$

Matici celkového rozptylu můžeme rozložit jako $\mathbf{S}_t = \mathbf{S}_w + \mathbf{S}_b$, kde $\mathbf{S}_b$ je matice rozptylu mezi třídami:

$$\mathbf{S}_b = N_1(\vec{\mu}_1 - \vec{\mu})(\vec{\mu}_1 - \vec{\mu})^T + N_2(\vec{\mu}_2 - \vec{\mu})(\vec{\mu}_2 - \vec{\mu})^T$$

Statistické charakteristiky transformovaných datapointů do prostoru o jednom rysu

Podobné statistické charakteristiky mohou být poskytnuty pro vzory z původního datasetu transformovaného do prostoru o jednom rysu (lineární transformací $y = \vec{w}^T \vec{x}$).

Všechny transformované datapointy z T_{tra} tvoří dataset $T_{\text{tra,proj}}$ s N dvojicemi $(y^i, c^i_{\text{target}})$.

Průměr transformovaných datapointů v $T_{\text{tra,proj}}$ pro každou z tříd c_1 a c_2 může být odhadnut z

$$\mu_{i,p} = \frac{1}{N_i} \sum_{y \in c_i} y = \frac{1}{N_i} \sum_{y \in c_i} \vec{w}^T \vec{x} = \vec{w}^T \vec{\mu}_i \quad (i = 1, 2)$$

a rozptyl jednoho rysu y pro každou třídu je

$$s^2_{i,p} = \frac{1}{N_i} \sum_{y \in c_i} (y - \mu_{i,p})^2 \quad (i = 1, 2)$$

Odhad variance transform. datapointů pro každou třídu je $(1/N_i)s^2_{i,p}$.
Celkový rozptyl y uvnitř tříd v celém $T_{\text{tra,proj}}$ je

$$s^2_{t,p} = s^2_{1,p} + s^2_{2,p}.$$

Vidíme, že vzdálenost mezi transformovanými průměry může být nalezena jako

$$|\mu_{1,p} - \mu_{2,p}| = |\vec{w}^T(\vec{\mu}_1 - \vec{\mu}_1)|$$

Kritérium

$$J_{\text{Fisher}}(\vec{w}) = \frac{|(\mu_{1,p} - \mu_{2,p})^2|}{s^2_{1,p} + s^2_{2,p}}$$

bude mít velkou hodnotu:

- pro velký rozptyl mezi třídami,
- pro malý rozptyl uvnitř tříd.

Toto kritérium se nazývá **F-ratio**.

Načtneme řešení pro optimální lineární transformaci:
Pokusíme se najít transformační vektor $\hat{\vec{w}}$, který maximalizuje $J_{\text{Fisher}}(\vec{w})$.

Prostým odvozením se ukáže, že

$$(\mu_{1,p} - \mu_{2,p})^2 = \vec{w}^T \mathbf{S}_b \vec{w},$$

kde $\mathbf{S}_b = (\vec{\mu}_1 - \vec{\mu}_2)(\vec{\mu}_1 - \vec{\mu}_2)^T$ a že

$$s_{t,p}^2 = s_{1,p}^2 + s_{2,p}^2 = \vec{w}^T \mathbf{S}_w \vec{w}.$$

Tedy máme $$J_{\text{Fisher}}(\vec{w}) = \frac{\vec{w}^T \mathbf{S}_b \vec{w}}{\vec{w}^T \mathbf{S}_w \vec{w}}$$

Je známo, že optimální transformační vektor $\vec{w}$, který maximalizuje $J_{\text{Fisher}}(\vec{w})$ musí být řešení zobecněného problému vlastních hodnot

$$\mathbf{S}_b \vec{w} = \lambda \mathbf{S}_w \vec{w}$$

neboli, v jiném tvaru

$$\mathbf{S}_w^{-1} \mathbf{S}_b \vec{w} = \lambda \vec{w}.$$

Pro nesingulární $\mathbf{S}_w$ máme konečné řešení pro optimální vektor $\vec{w}$:

$$\hat{\vec{w}} = \mathbf{S}_w^{-1}(\vec{\mu}_1 - \vec{\mu}_2)$$

Fisherův lineární diskriminant – klasifikace

Hlavní cíl FLT je optimálně konvertovat datapointy do prostoru s nižší dimenzionalitou, tak aby byly vhodnější ke klasifikaci.

My ovšem můžeme navrhnout klasifikátor založený na FLT, použitím vzorce na optimální projekci, který jedná jako diskriminant.

$$y = d(\vec{x}) = \hat{\vec{w}}^T \vec{x}$$

Pokud zvolíme prahovou hodnotu y_{treshold} pro transformované vektory y nový vektor by mohl být klasifikován následovně

$$\vec{x} \begin{cases} \in c_1 & \text{pokud } y = \mathrm{d}(\vec{x}) = \hat{\vec{w}}^T \vec{x} > y_{\text{treshold}} \\ \in c_2 & \text{jinak.} \end{cases}$$

Teď uvažujme dataset T_{tra} o l třídách s n-dimenzionálními labelovými vektory ($l < n$). Proberem, jak transformovat n-dimenzionální vektorů z T_{tra} na m-dimenzionální vektory, abychom zajistili maximální separabilitu mezi třídami.
Nechť máme m lineárních transformací (diskriminantů)

$$y_i = \vec{w}_i^T \vec{x} \quad (i = 1, 2, \ldots, m)$$

Všech m rysů y_i tvoří tranformovaný vektor

$$\vec{y} = [y_1, y_2, \ldots, y_m]^T \in \mathbb{R}^m$$

Lineární transformace má tvar

$$\vec{y} = \mathbf{W}\vec{x},$$

kde $\mathbf{W} \in \mathbb{R}^{m \times n}$ je transformační matice, kde každý řádek je $\vec{w}_i$ transformační vektor odpovídajícího rysu y_i.

Můžeme zobecnit definice statistických charakteristik pro oba datasety T_{tra} a $T_{\text{tra, proj}}$ potřebné pro nalezení optimální transformační matice $\hat{\mathbf{W}}$:

Matice rozptylu v třídě c_i

$$\mathbf{S}_i = \sum_{\vec{x} \in c_i} (\vec{x} - \vec{\mu}_i)(\vec{x} - \vec{\mu}_i)^T, \quad (i = 1, 2, \ldots, l),$$

kde $\vec{\mu}_i$ je průměr pro třídu c_i

$$\vec{\mu}_i = \frac{1}{N_i} \sum_{\vec{x} \in c_i} \vec{x}, \quad (i = 1, 2, \ldots, l)$$

Matice rozptylu v třídách

$$\mathbf{S}_w = \sum_{i=1}^{l} \mathbf{S}_i$$

matice celkového rozptylu

$$\mathbf{S}_t = \sum_{j=1}^{N} (\vec{x}_j - \vec{\mu})(\vec{x}_j - \vec{\mu})^T$$

kde $\vec{x}$ je odhad celkového průměru

$$\vec{\mu} = \frac{1}{N} \sum_{j=1}^{N} \vec{x}_j = \frac{1}{N} \sum_{i=1}^{I} N_i \vec{\mu}_i$$

$$\mathbf{S}_t = \mathbf{S}_w + \mathbf{S}_b,$$

kde $\mathbf{S}_w$ je matice rozptylu uvnitř tříd a $\mathbf{S}_b$ je matice rozptylu mezi třídami

$$\mathbf{S}_b = \sum_{i=1}^{I} N_i (\vec{\mu}_i - \vec{\mu})(\vec{\mu}_i - \vec{\mu}_i)^T$$

Projekce n-dimensionálního prostoru do $(l-1)$-dimenzionálního prostoru diskriminantů je dán $(l-1)$ funkcemi

$$y_i = \vec{w}_i^T \vec{x}, \quad i = 1, \ldots, l-1$$

Můžeme také psát maticovou verzi; předp., že $\vec{y} \in \mathbb{R}^{(l-1)}$ a $\mathbf{W}$ je $n \times (l-1)$ matice obsahující, jako řádky, transformační vektory w_i:

$$\vec{y} = \mathbf{W}\vec{x}$$

Statistické charakteristiky pro dataset $T_{\text{tra,proj}}$

$$\vec{\mu}_{i,p} = \frac{1}{N_i} \sum_{\vec{y} \in c_i} \vec{y} \qquad (i = 1, 2, \ldots, l)$$

$$\vec{\mu}_p == \frac{1}{N} \sum_{j=1}^{N} \vec{y}_j = \frac{1}{N} \sum_{\vec{y} \in c_i} N_i \vec{y}$$

$$S_{i,p} = \sum_{\vec{y} \in c_i} (\vec{y} - \vec{\mu}_{i,p})(\vec{y} - \vec{\mu}_{i,p})^T \qquad (i = 1, 2, \ldots, l)$$

$$S_{w,p} = \sum_{i=1}^{l} S_{i,p} = \sum_{j=1}^{N} (\vec{y}_j - \vec{\mu}_p)(\vec{y}_j - \vec{\mu}_p)^T$$

Můžeme vidět, že

$$\mathbf{S}_{w,p} = \mathbf{W}^T \mathbf{S}_w \mathbf{W} \quad \text{a} \quad \mathbf{S}_{b,p} = \mathbf{W}^T \mathbf{S}_b \mathbf{W}$$

Pro více tříd ma $m \geq 2$ může být definován následující kritérium separability mezi třídami takto:

$$J(\mathbf{W}) = \frac{|\mathbf{S}_{b,p}|}{|\mathbf{S}_{w,p}|} = \frac{|\mathbf{W}^T \mathbf{S}_b \mathbf{W}|}{|\mathbf{W}^T \mathbf{S}_w \mathbf{W}|}$$

Optimální transformace $\hat{\mathbf{W}}$ může být nalezena jako řešení obecného problému vlastních hodnot

$$\mathbf{S}_b \vec{w}_i = \lambda_i \mathbf{S}_w \vec{w}_i^T \qquad (i = 1, 2, \ldots, n)$$

Násobením obou stran $\mathbf{S}_w^{-1}$ dostaneme

$$\mathbf{S}_w^{-1} \mathbf{S}_b \vec{w}_i = \lambda_i \vec{w}_i^T \qquad (i = 1, 2, \ldots, n)$$

Řešení tohoto problému je tedy výpočet (a uspořádání do nerostoucího pořadí), vlastních hodnot a odpovídajících vektorů matice $\mathbf{S}_w^{-1} \mathbf{S}_b$.

Předp. že vlastní hodnoty matice $\mathbf{S}_w^{-1}\mathbf{S}_b$ jsou uspořádány v nerostoucím pořadí

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$$

Odpovídající vlastní vektory $\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n$ dávají $n \times n$ matici

$$\mathbf{E} = [\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n] .$$

Optimální transformaci $\hat{\mathbf{W}}$ dostaneme použitím prvních m sloupců matice $\mathbf{E}$ jako řádků:

$$\hat{\mathbf{W}} = \begin{bmatrix} \vec{e}_1^T \\ \vec{e}_2^T \\ \vdots \\ \vec{e}_m^T \end{bmatrix}$$

Sekvence PCA a Fisherovy LDA

- PCA (a výsledná KLT) poskytuje extrakci rysů, která je optimální vzhledem k minimalizaci rekonstrukční chyby, ale negarantuje, že výběr hlavních komponent bude adekvátní pro klasifikaci.
- FTL je navržena tak, aby zajišťovala co největší separabilitu tříd.
- U FTL se ale může stát, že $\mathbf{S}_w$ zdegeneruje (není invertibilní). To se může stát, pokud $N < n$.

Možné řešení je nejdříve přes KLT zredukovat dimenzi na $m < N$ a pak až aplikovat FLDA (k redukci na dimenzi c) Pro vztahy mezi dimenzemi by mělo platit:

$$c < l \leq m \leq N - l,$$

Algoritmus pro Karhunen-Loéve-Fisherovu transformaci:
Vstup: Dataset T obsahující olabelované vektory $(\vec{x}_i, c^i_{\text{tartget}})$

1. Extrahuj z T jen vektory, vytvoř z nich $n \times N$ vytvoř $\mathbf{X}$ ($\vec{x}_i$ jsou řádky)
2. Vyber dimenzi m (počet hlavních komponent), která splňuje $l \leq m \leq N - l$.
3. Vypočítej $m \times n$ KLT matici $\mathbf{W}_{KL}$
4. Transformuj $\mathbf{X}$ touto tranformací, tj. vypočítej $\mathbf{Y} = \mathbf{X}\mathbf{W}_{KL}^T$.
5. Vyber dimenzi c (počet hlavních komponent), která splňuje $c + 1 \leq l$.
6. Vypočítej $c \times m$ FLT matici $\mathbf{W}_F$
7. Transformuj $\mathbf{Y}$ touto tranformací, tj. vypočítej $\mathbf{Z} = \mathbf{Y}\mathbf{W}_F^T$.

Výstup: $\mathbf{W}_{KL}, \mathbf{W}_F, \mathbf{Z}$.

LDA v R

```
> iris1 <- as.matrix(iris[1:50,1:4])
> iris2 <- as.matrix(iris[51:100,1:4])
> iris3 <- as.matrix(iris[101:150,1:4])

> iris1.mean <- mean(as.data.frame(iris1))
> iris2.mean <- mean(as.data.frame(iris2))
> iris3.mean <- mean(as.data.frame(iris3))

> iris1.centered  <- sweep(iris1,2,
+                          iris1.mean,FUN='-')
> iris2.centered  <- sweep(iris1,2,
+                          iris2.mean,FUN='-')
> iris3.centered  <- sweep(iris3,2,
+                          iris3.mean,FUN='-')

> iris.mean  <- mean(iris[,1:4])
>
```

```
> iris.centered  <- as.matrix(
+     sweep(iris[,1:4],2,iris.mean,FUN='-'))

> S1  <- t(iris1.centered) %*% iris1.centered
> S2  <- t(iris2.centered) %*% iris2.centered
> S3  <- t(iris3.centered) %*% iris3.centered

> Sw  <- S1 + S2 + S3
> St  <- t(iris.centered) %*% iris.centered
> Sb  <- St - Sw

> Swi <- ginv(Sw)

> W  <- eigen(Swi %*% Sb)$vectors
trans  <- iris.centered %*% W
```

```
plot(trans[,1], col=iris[,5])
```