OpenCV

Andrej Lúčny

Katedra aplikovanej informatiky FMFI UK

lucny@fmph.uniba.sk

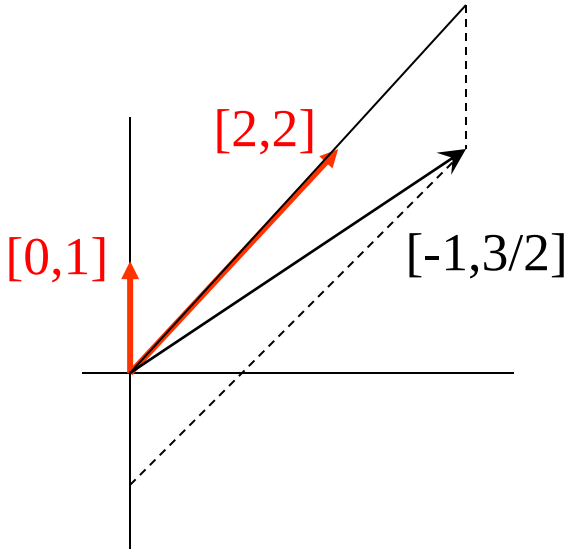
http://dai.fmph.uniba.sk/w/Andrej_Lucny

www.agentspace.org/opencv

Zmena bázy

- matica prechodu:
  obrazy jednotkových vektorov
  (zobrazenie na vhodné lineárne kombinácie)

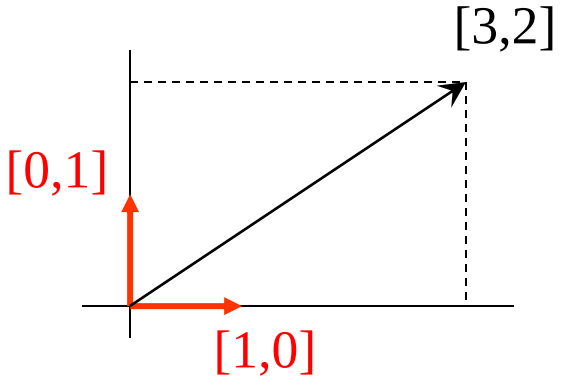
$$[x_1, x_2] \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} = [x'_1, x'_2]$$



$$[-1, 3/2] \begin{pmatrix} 0 & 1 \\ 2 & 2 \end{pmatrix} = [3, 2]$$

znamená, že

$$x'_1 = a_{1,1}x_1 + a_{2,1}x_2$$
$$x'_2 = a_{1,2}x_1 + a_{2,2}x_2$$



[x_1, x_2] v báze [a_{11},a_{12}] [a_{21},a_{22}] je to isté ako

[x'_1, x'_2] v báze [1,0] [0,1]

Dataset

- Množina n vzoriek dimenzie m

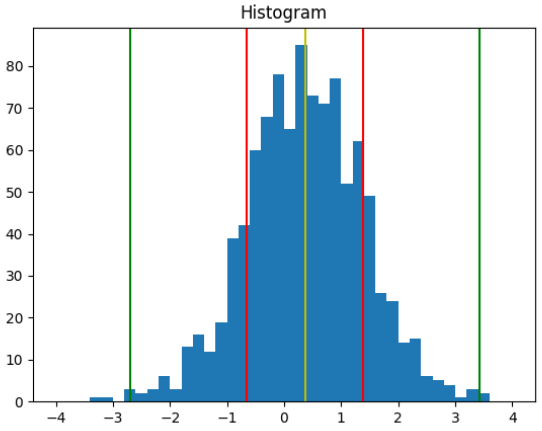
$$[a_1, \ldots, x_1, y_1 \ldots]$$
$$[a_2, \ldots, x_2, y_2 \ldots]$$
$$\ldots$$
$$[a_n, \ldots, x_n, y_n \ldots]$$

$$[x_1, x_2, \ldots, x_n]$$
$$[y_1, y_2, \ldots, y_n]$$

Ako zistiť či pozícia x súvisí s y ?

Priemer, smerodajná odchýľka

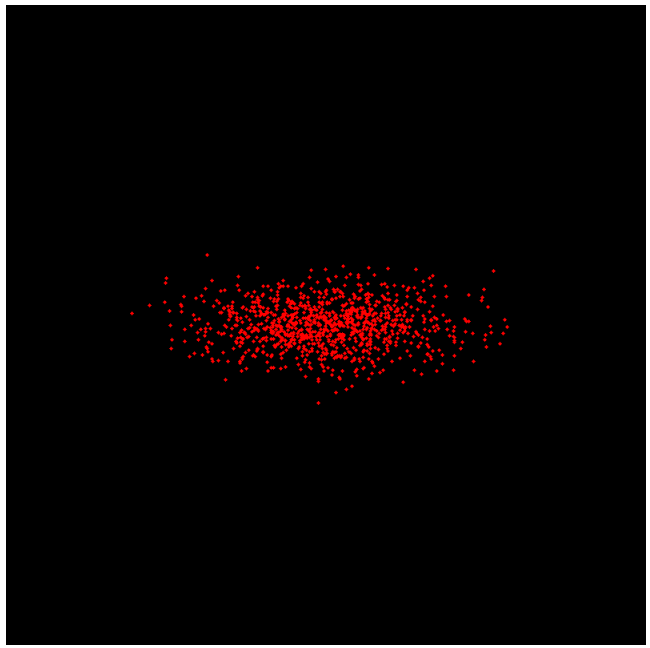
$$\overline{x} = \sum_{i=0}^{n} x_i \qquad \sigma_x = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_i - \overline{x})^2}$$



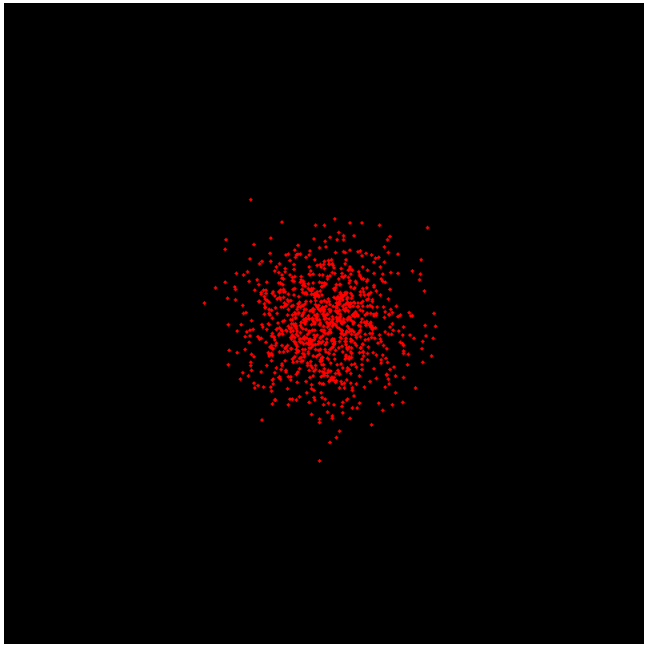
$[x_i, y_i]$

$\left[\frac{x_i - \overline{x}}{\sigma_x}, \frac{y_i - \overline{y}}{\sigma_y}\right]$

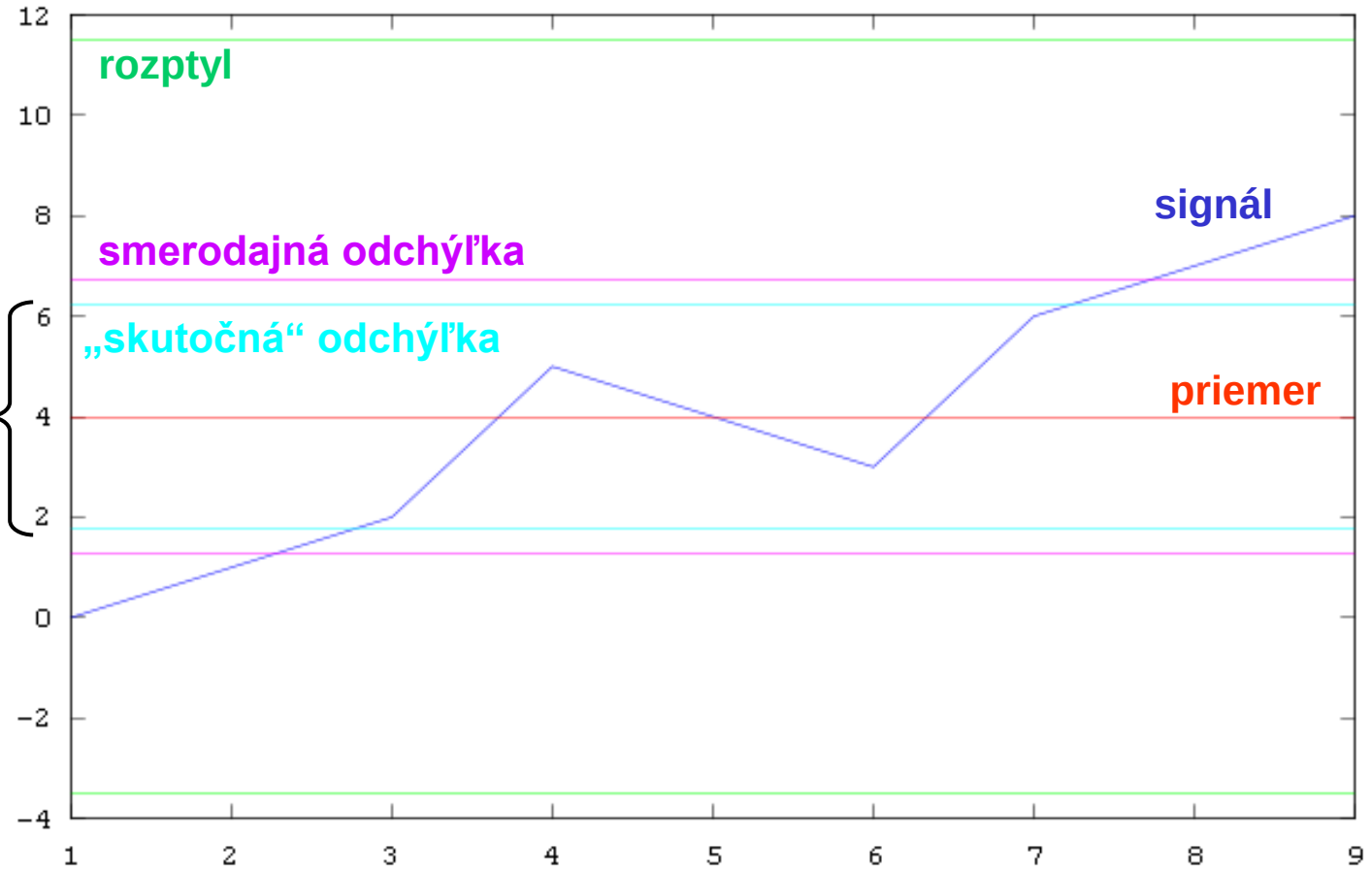
m=2



normalizácia



Odchýľka a smerodajná odchýľka



68% { 50% { $\langle \overline{x} - \sigma_x, \overline{x} + \sigma_x \rangle$

99% $\langle \overline{x} - 3\sigma_x, \overline{x} + 3\sigma_x \rangle$ pravidlo 3σ

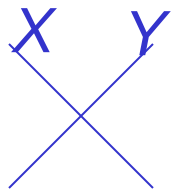
Kovariancia

$$\overline{x} = \sum_{i=0}^{n} x_i$$

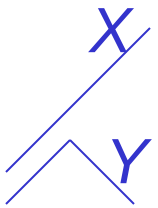
$$\overline{y} = \sum_{i=0}^{n} y_i$$

Kovariancia dvoch signálov X a Y:

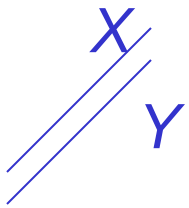
$$cov(x,y) = \frac{1}{n}\sum_{i=0}^{n}(x_i - \overline{x})(y_i - \overline{y})$$



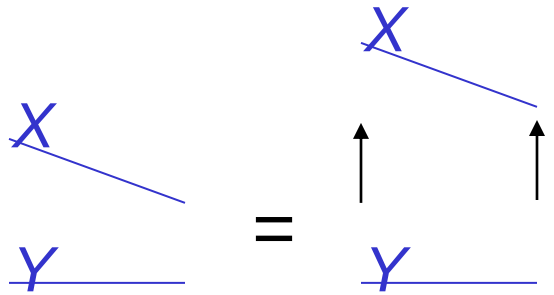
záporná

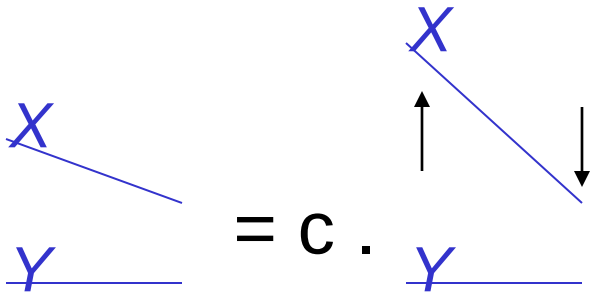


nulová



kladná





$$cov(x,y) = \frac{1}{n}\sum_{i=0}^{n}(x_i - \overline{x})(y_i - \overline{y}) =$$

$$= \frac{1}{n}\frac{\sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_i - \overline{x})^2}\sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - \overline{y})^2}}{\sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_i - \overline{x})^2}\sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - \overline{y})^2}}\sum_{i=0}^{n}(x_i - \overline{x})(y_i - \overline{y}) =$$

$$= \sigma_x\sigma_y\frac{\sum_{i=0}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=0}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=0}^{n}(y_i - \overline{y})^2}} =$$

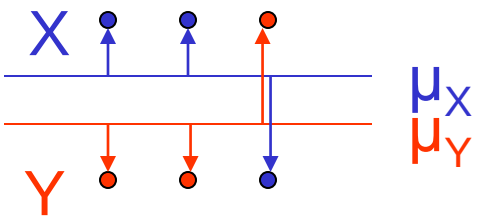
$$= \sigma_x\sigma_y\frac{(x - \overline{x})\cdot(y - \overline{y})}{|x - \overline{x}||y - \overline{y}|} =$$

$$= \sigma_x\sigma_y\cos(\phi) = \sigma_x\sigma_y cor(x,y)$$

$$cov(a,a) = \sigma_a\sigma_a\cos(0) = \sigma_a^2 = \rho_a \qquad cov(a,b) = cov(b,a)$$

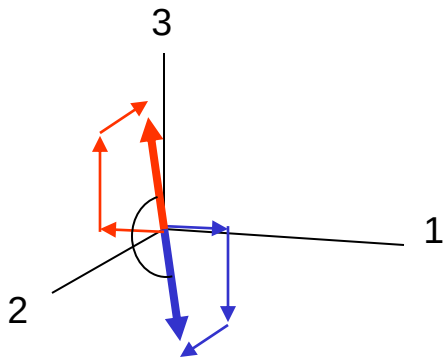
Korelácia

n=3



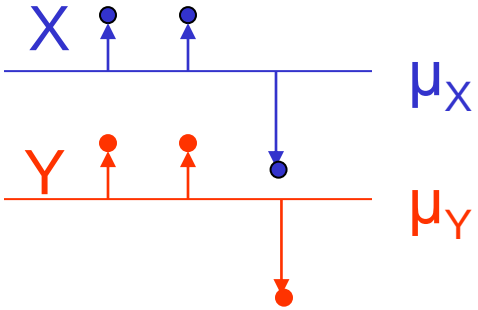
μ_X

μ_Y



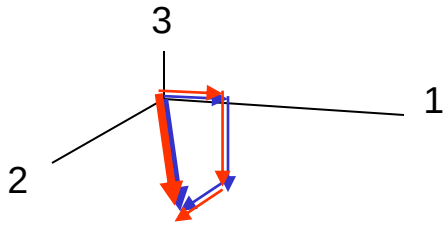
$\varphi = \pi$

$cor(X,Y) = -1$



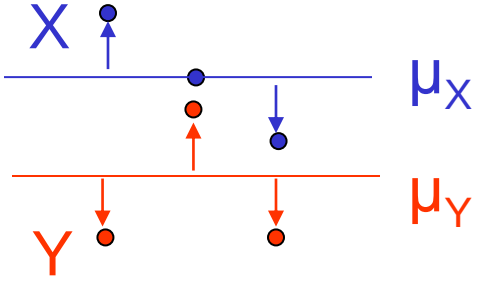
μ_X

μ_Y



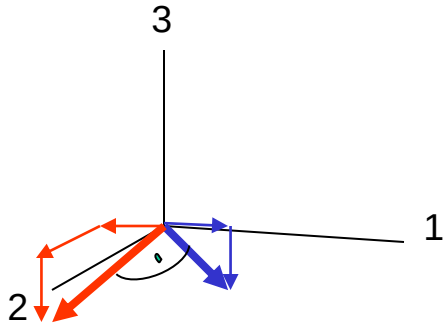
$\varphi = 0$

$cor(X,Y) = 1$



μ_X

μ_Y



$\varphi = \pi/2$

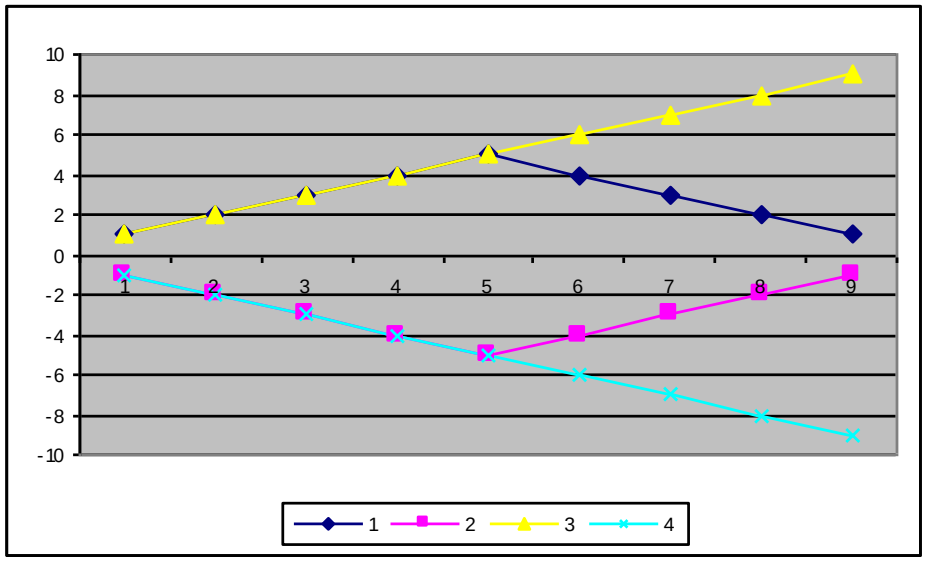
$cor(X,Y) = 0$

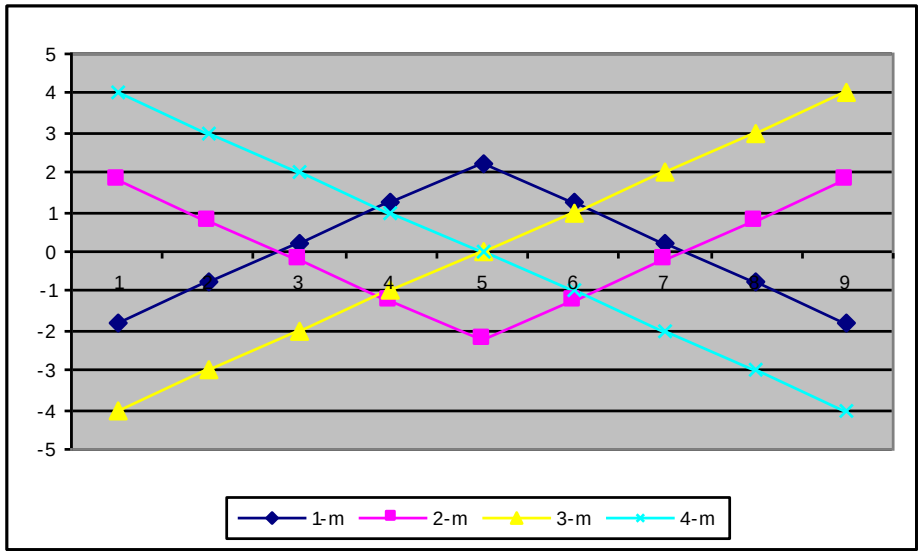
Kovariančná matica

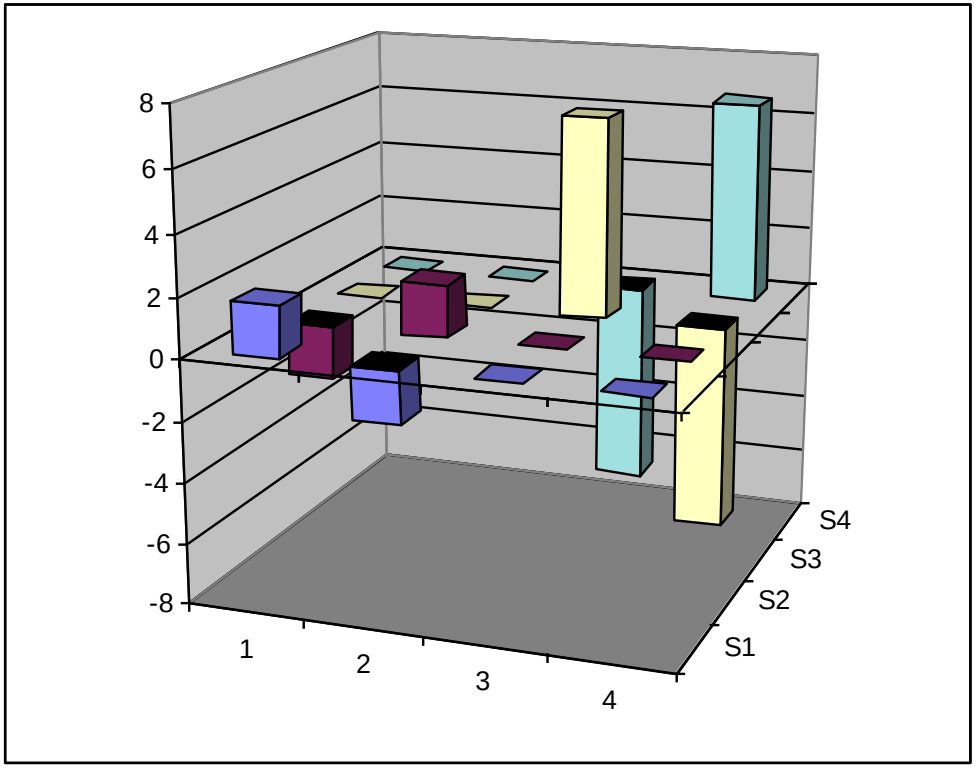
Matica rozmerov m x m udávajúca vzájomné kovariancie dimenzií datasetu

$$\begin{pmatrix} cov(a,a) & \dots & cov(a,x) & cov(a,y) & \dots \\ \dots & & & & \\ cov(x,a) & \dots & cov(x,x) & cov(x,y) & \dots \\ cov(y,a) & \dots & cov(y,x) & cov(y,y) & \dots \\ \dots & & & & \end{pmatrix}$$

Kovariančná matica







Idea PCA

- Vedeli by sme nájsť takú zmenu bázy, že kovariančná matica vyjde diagonálna?
- To je dobré na to, že jednotlivé dimenzie sú od seba nezávislé

- Odpoveď: vedeli, takú zmenu bázy zabezpečí matica prechodu po stĺpcoch zložená z normalizovaných vlastných vektorov. Hodnoty na diagonále potom budú vlastnými hodnotami kovariančnej matice a zodpovedajú rozptylu

Vlastné vektory a hodnoty

- Transformujme tieto body pomocou lineárnej transformácie

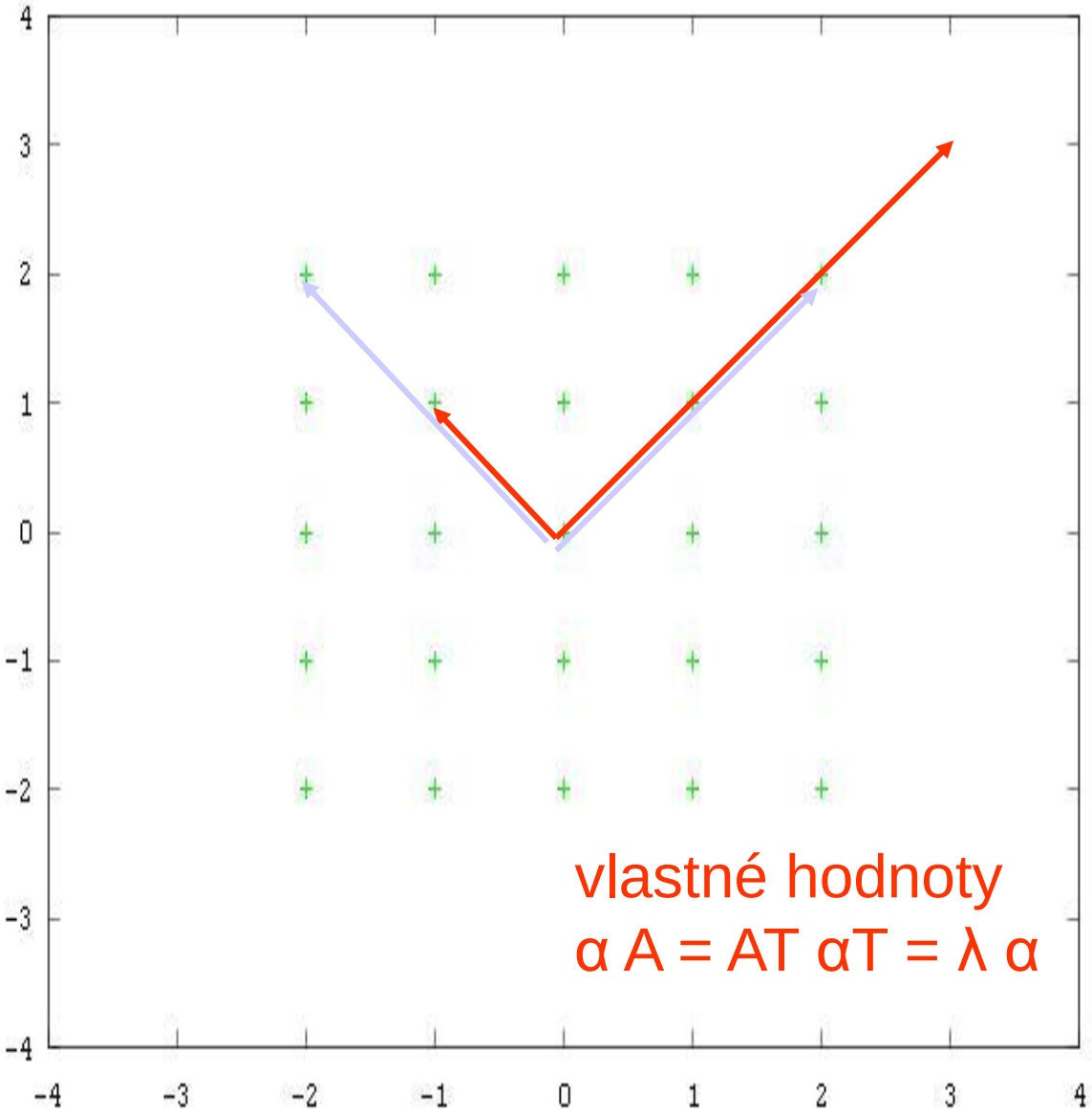
Vlastné vektory a hodnoty

- Transformované body
  pomocou transformácie A

Vlastné vektory a hodnoty

vlastné vektory udávajú v akých smeroch sa vzor deformuje na obraz

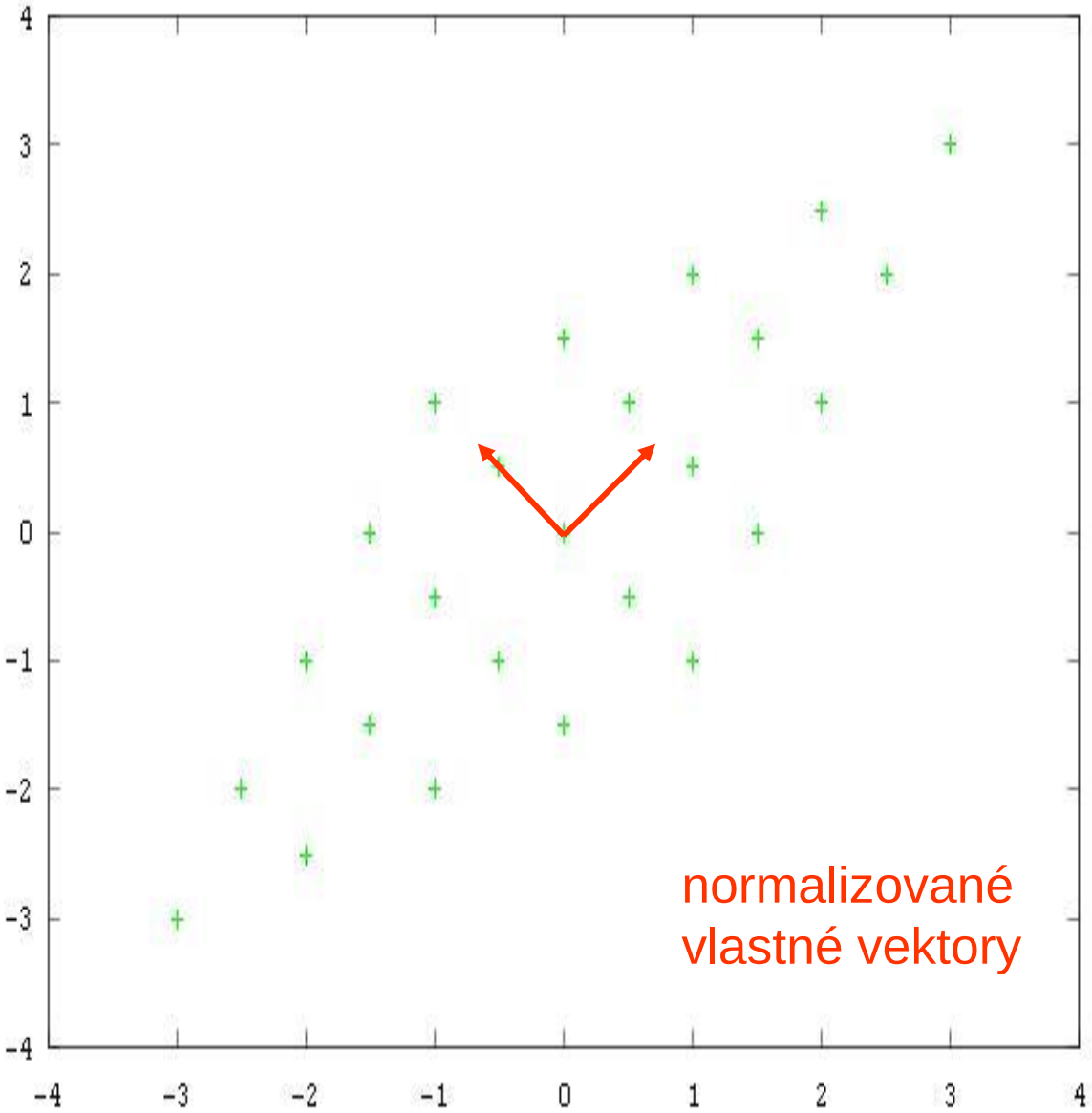
vlastné hodnoty udávajú zväčšovanie či zmenšovanie v deformačných smeroch



$\frac{|\leftarrow|}{|\leftarrow|} = 1/2$

$\frac{|\leftarrow|}{|\leftarrow|} = 3/2$

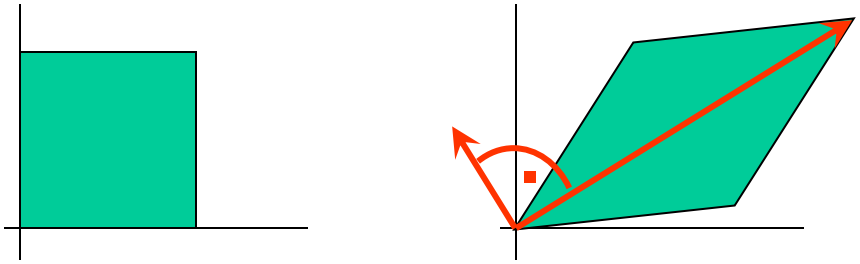
každý násobok vlastného vektoru je rovnako vhodný ale iba jeden má dĺžku 1



| ← | = 1

Symetrická matica

- $A = A^T$
- zmena bázy má vždy charakter kosoštvorcovej deformácie s otočením a s prevrátením alebo bez:



- vlastné hodnoty sú reálne čísla (A je Hermitovská)

  a vlastné vektory sú na seba kolmé (uhlopriečky kosoštvorca sú na seba kolmé) a preto pre $\alpha = V^T$ $\alpha\alpha^T = \text{diag}$ takže ak vlastné vektory znormalizujeme, tak $\alpha\alpha^T = I$ a teda $\alpha^T = \alpha^{-1}$

PCA (Principal component analysis)

- Nech X je centrovaný dataset
- Hľadáme teda zmenu bázy, t.j. regulárnu maticu P rozmerov m x m takú, že

$$XP = Y \text{ a } C_Y \text{ je diagonálna}$$

- $E(X) = \overline{X}$
- centrovanie dát sa zmenou bázy nepokazí, ak bolo centrované X, bude centrované aj Y, lebo $E(Y) = E(XP) = E(X)P = 0\,P = 0$

PCA (Principal component analysis)

pre ľubovoľné P platí:

$$C_Y = E(Y^T Y) = E((XP)^T XP) = E(P^T X^T XP) =$$
$$= P^T E(X^T X) P = P^T C_X P$$

Pre C_X sa dajú nájsť jej vlastné hodnoty a vektory (je to symetrická matica):

$$\alpha_i C_X = \lambda_i \alpha_i \quad C_X = \alpha^{-1} \Lambda \alpha \quad \alpha = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} \quad \Lambda = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \lambda_n \end{pmatrix}$$

α_i sú ortogonálne, dajú sa z normalizovať a potom: $\alpha^{-1} = \alpha^T$

Takže keď položíme $P = \alpha^T$ dostávame:

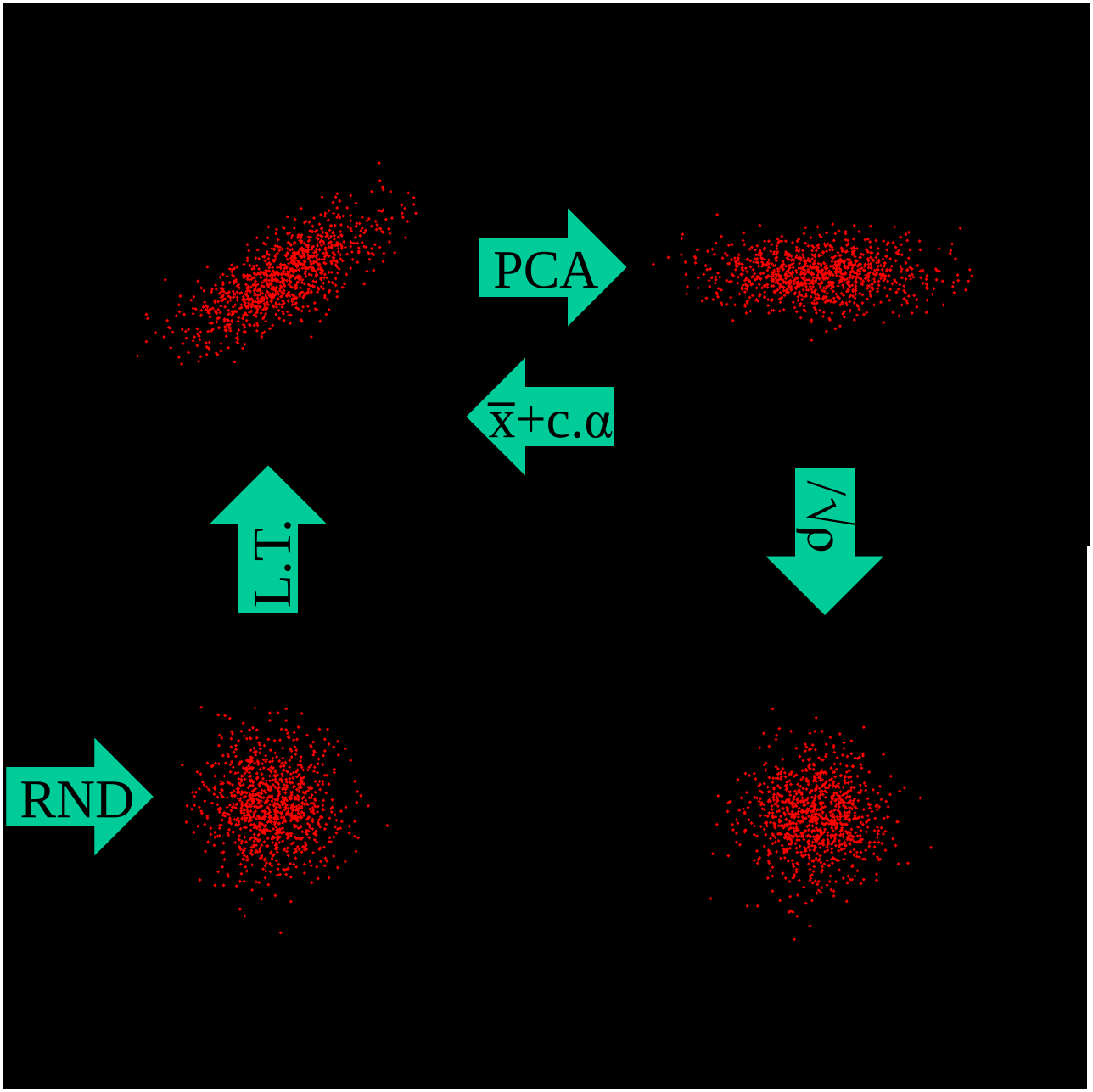
$$C_Y = P^T C_X P = P^T \alpha^{-1} \Lambda \alpha P = \alpha \alpha^{-1} \Lambda \alpha \alpha^T = \Lambda$$

Hľadanou transformáciou je matica, kde sú po stĺpcoch normalizované vlastné vektory C_X

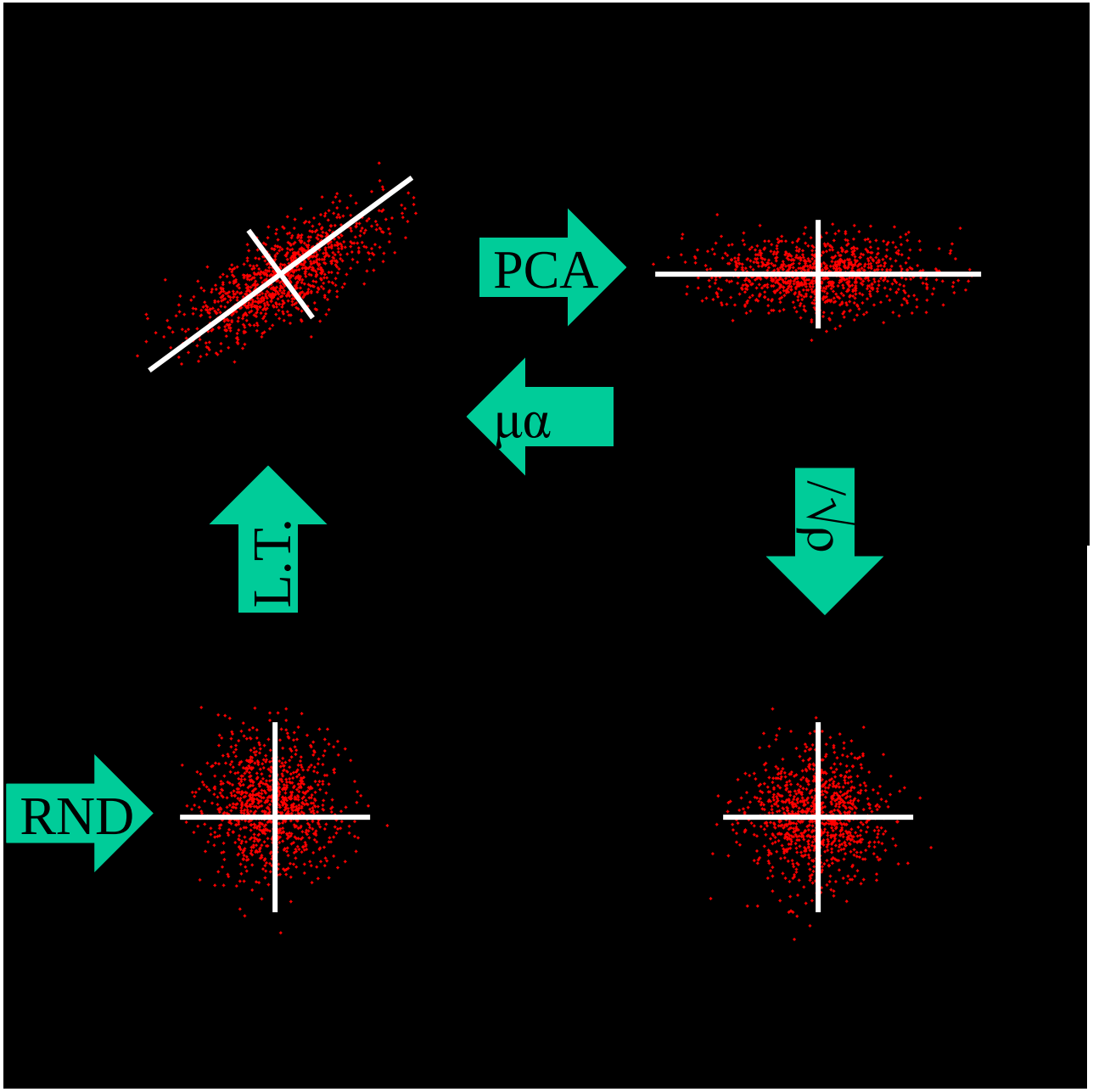
PCA (Principal component analysis)

- Matica P, kde sú po stĺpcoch normalizované vlastné vektory C_X transformuje dáta to priestoru, kde sú ich jednotlivé zložky nekorelujúce
- absolútna hodnota vlastných hodnôt C_X dáva navyše informáciu aký vplyv má daný component (rovná sa jeho rozptylu), čo súvisí s jeho významnosťou

Principal Component Analysis (PCA)



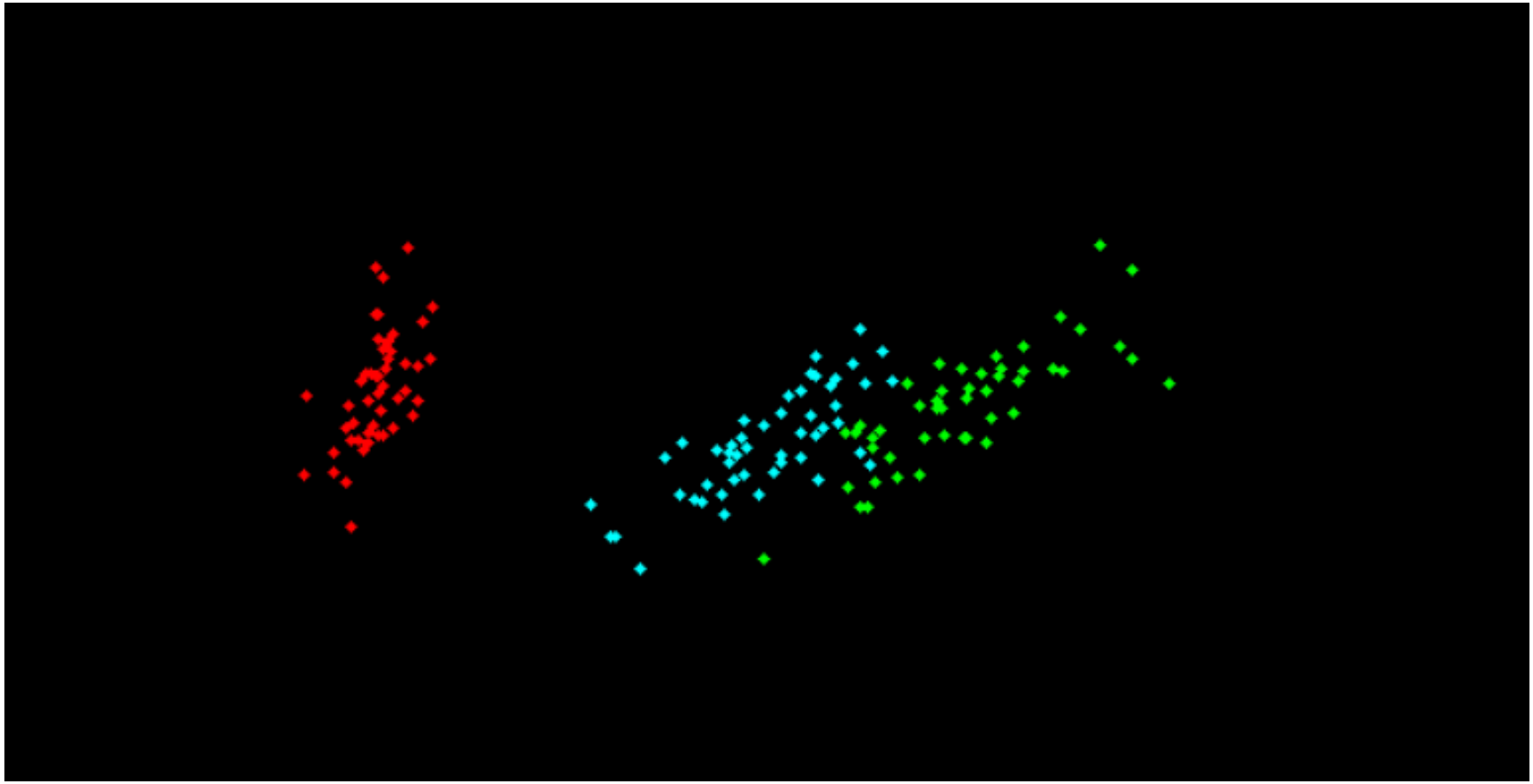
Principal Component Analysis (PCA)



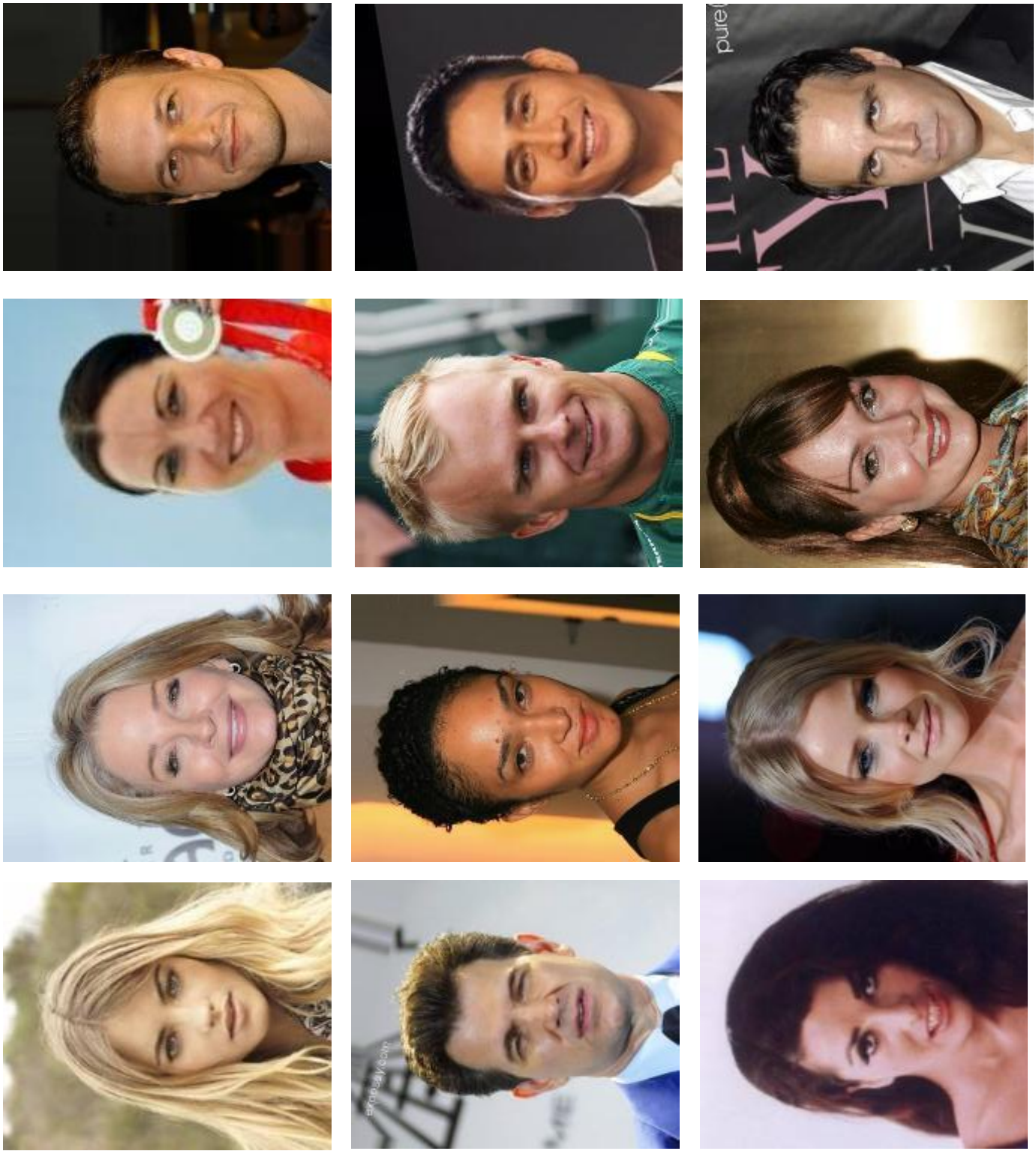
PCA (Principal component analysis)

- PCA vyjadruje pôvodné dáta ako priemer + lineárnu kombináciu vlastných vektorov (= normalizovaných vlastných vektorov vynásobených vlastnými hodnotami)
- Dimenzie s malými vlastnými hodnotami len málo vplývajú na výslednú hodnotu, takže ich môžeme zanedbať a tak PCA sa dá použiť na redukciu dimenzie

Zobrazenie mnohorozmerných dát pomocou PCA



Dataset tvárí



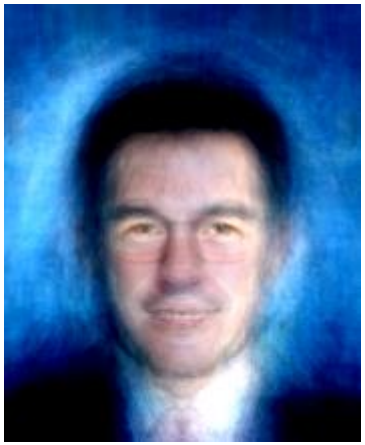
Eigenimages

čo sa dá urobiť s bodkami, dá sa urobiť aj s obrázkami...

Dataset tvárí rozložíme na priemer + násobky vlastných vektorov

← rôzne násobky →

Eigenimages



Obmedzením sa na významné komponenty poskytuje nám PCA transformáciu obrazu na **feature vector**, ktorý má oveľa menšiu dimenziu

$$\longrightarrow \quad c = [c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8]$$

Matica prechodu zložená z vlastných vektorov dokáže takto transformovať akýkoľvek obrázok daných rozmerov

Ako to využiť na rozpoznanie či je niečo fotka z pasu alebo nie? Z PCA vieme aj priemer μ a vlastné hodnoty a ich odmocniny sú smerodajné odchýlky: $\sigma = [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_8,]$

Takže ak $\mu - 3\sigma < c < \mu + 3\sigma$ povieme, že ide o fotku pasu a inak, že nie. (Nefunguje to moc dobre, ale trochu áno)

Všeobecná schéma klasifikácie

obraz

feature vector

$[c_1, c_2, \ldots, c_n]$

Metóda 1 → Metóda 2 → kategória

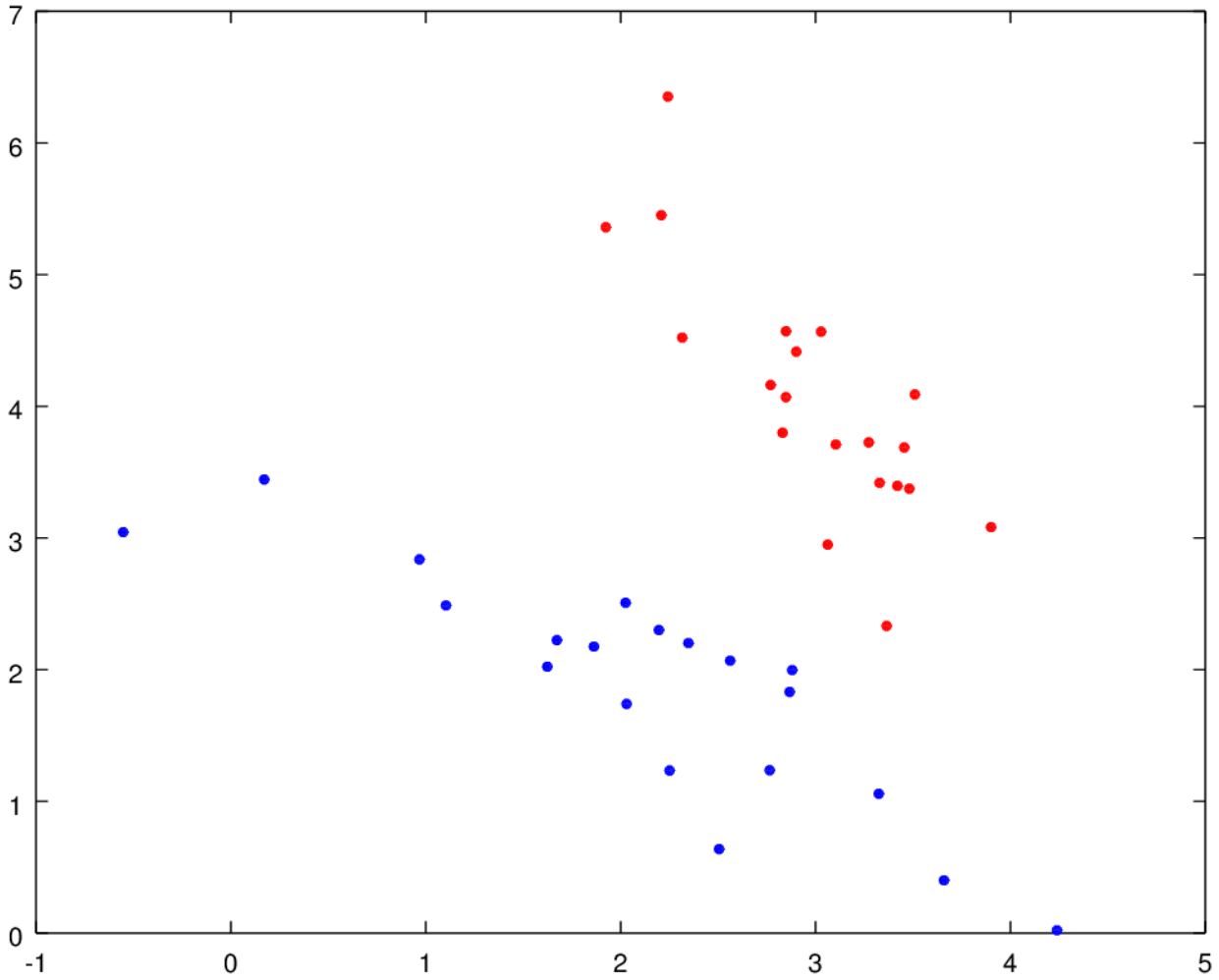
eigenimages	3σ	
eigenimages	LDA	
LBPH	LDA	(Fisher faces)
LBPH	SVM	
HOG	SVM	
Haar	Decision tree	(Viola-Jones)

Linear Discriminant Analysis (LDA)

- metóda strojového učenia
- LDA klasifikuje dáta v n-rozmernom priestore do viacerých kategórii, avšak spôsob jej fungovania si priblížime najprv na dvojrozmernom priestore a dvoch kategóriách
- Podstatou LDA je nájsť takú lineárnu transformáciu priestoru, ktorá najlepšie rozlišuje (diskriminuje) dané vzorky pri redukcii dimenzie priestoru.
- Je založená nájdení hlavných vektorov a hodnôt určitej matice, ktorá sa skonštruje zo vzoriek.

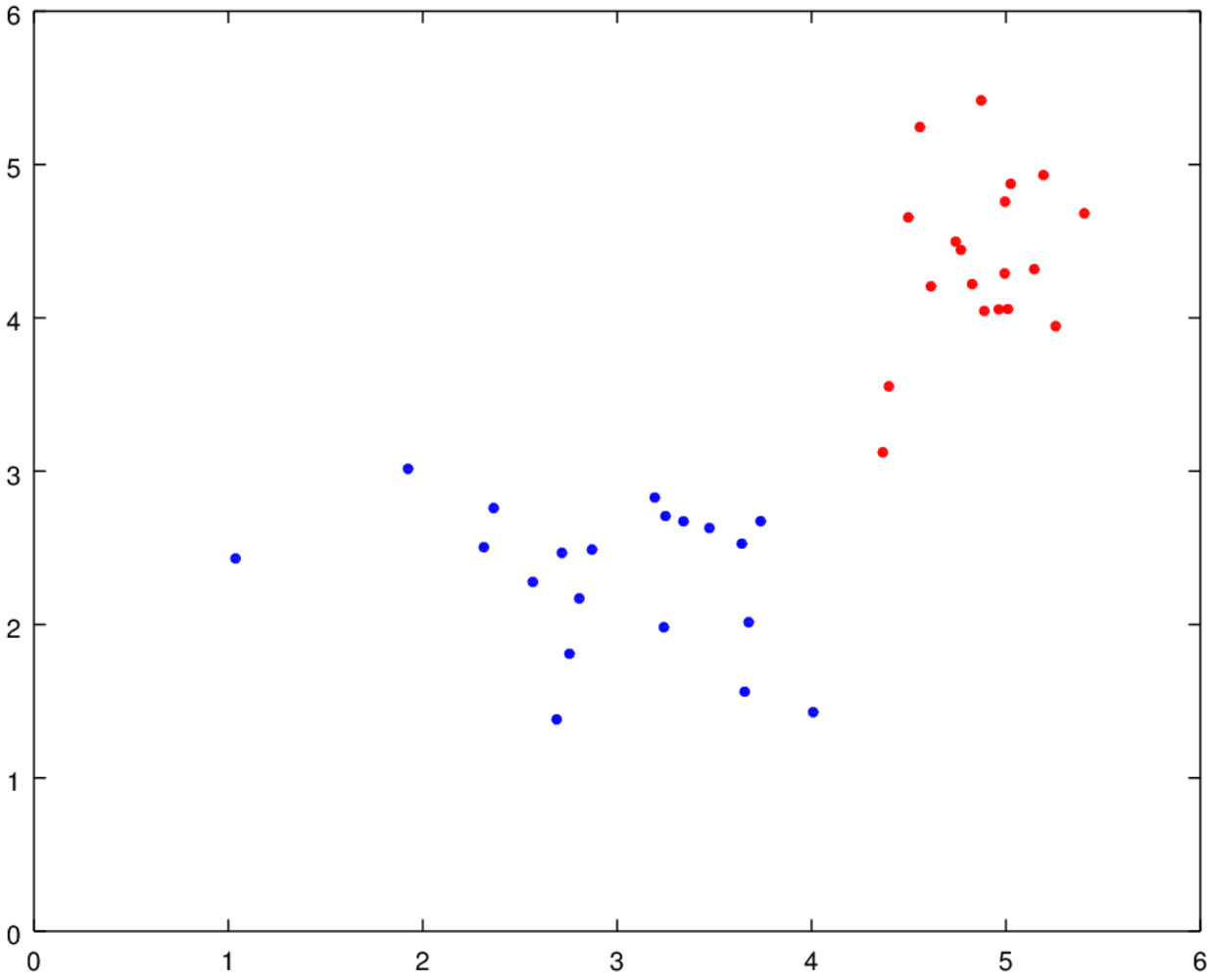
Linear Discriminant Analysis (LDA)

Uvažujme tieto vzorky dvoch kategórii



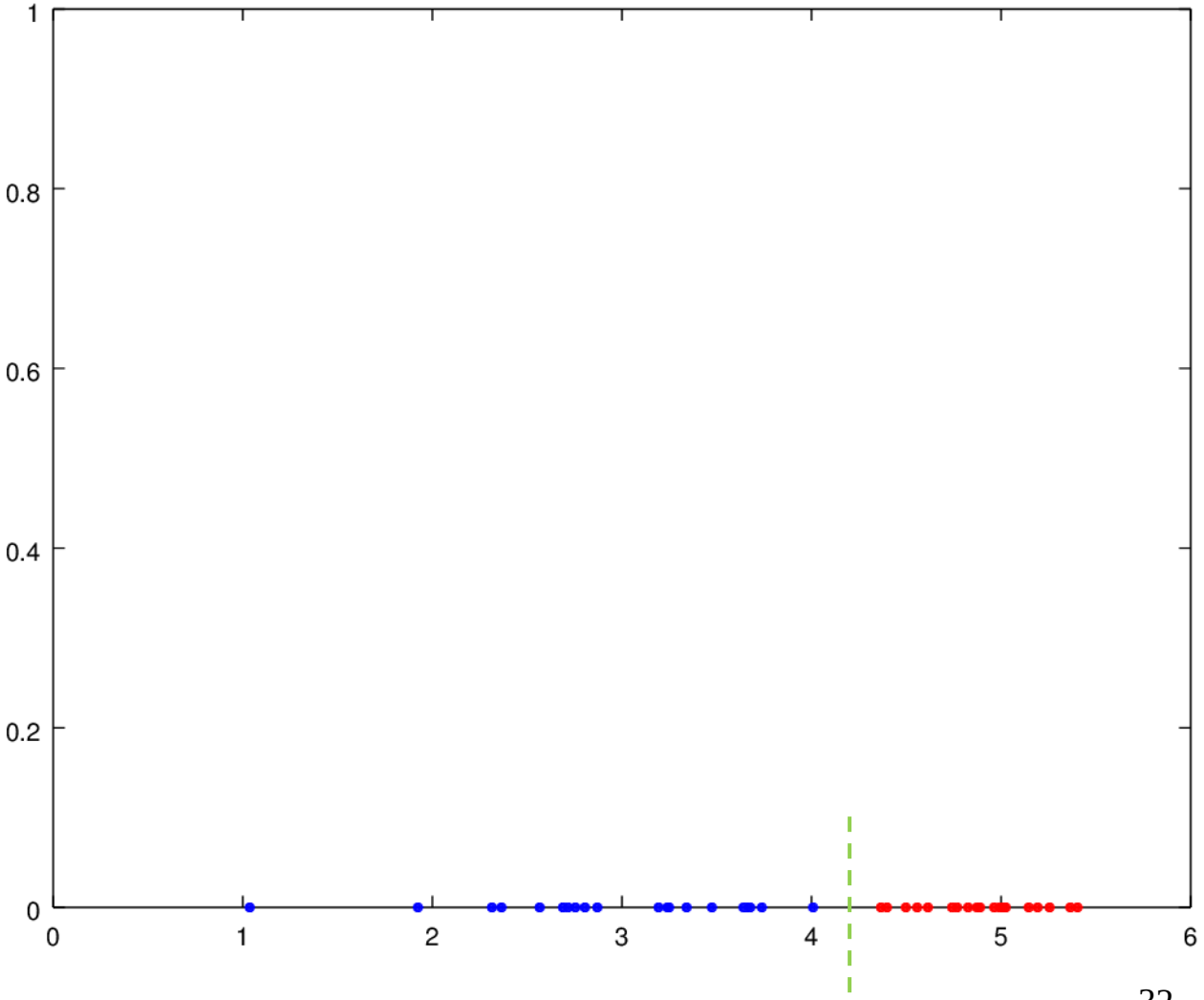
Linear Discriminant Analysis (LDA)

LDA ich dokáže transformovať ich súradnice tak ...

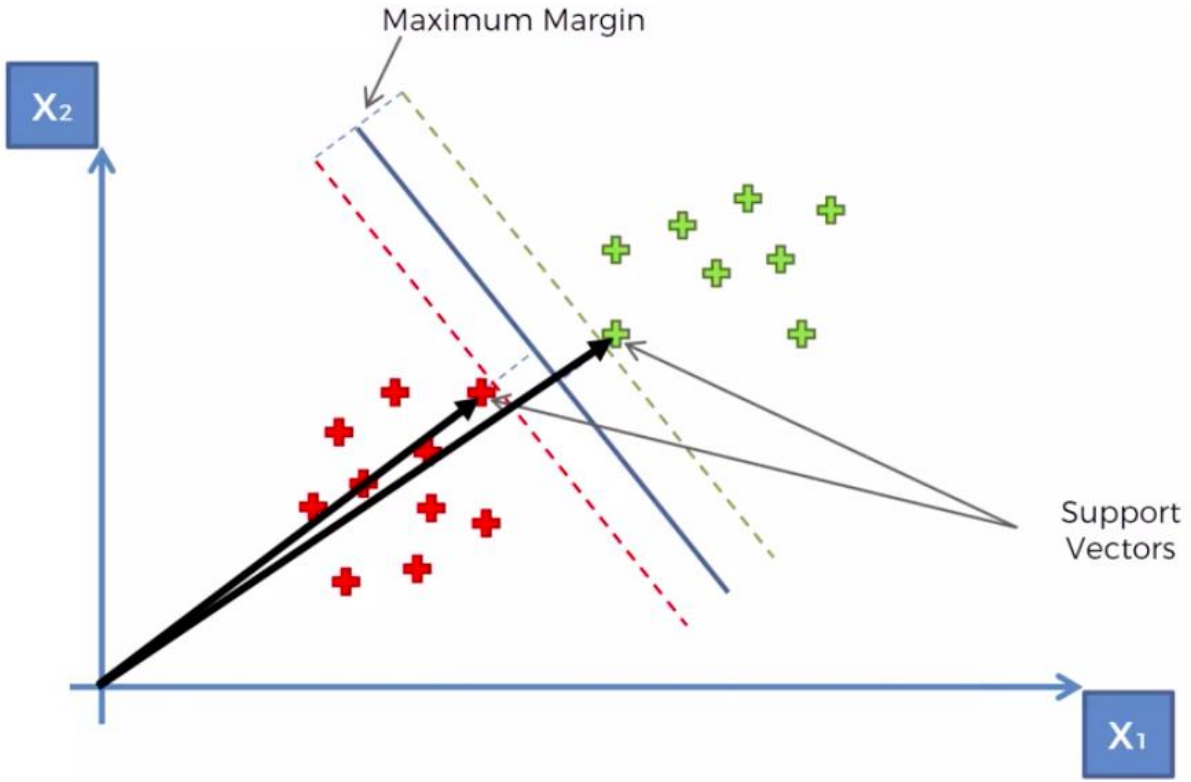


Linear Discriminant Analysis (LDA)

... že po redukcií dimenzie možno kategórie ľahko rozlíšiť



Support Vector Machines (SVM)

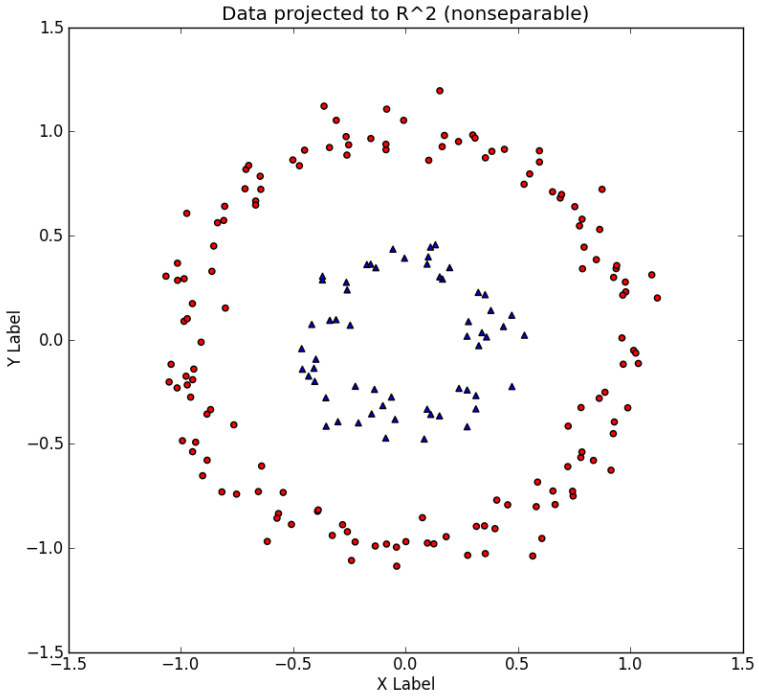


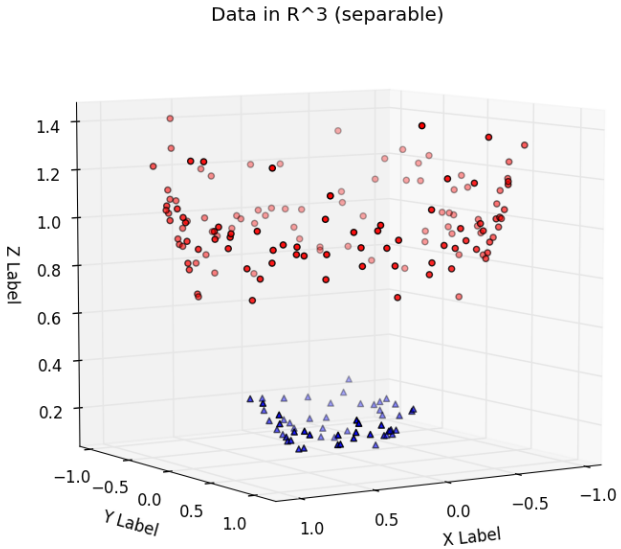
Klasifikátor, ktorý položí takú nadrovinu medzi dáta, ktorá ich čo najlepšie oddeľuje.

SVM s kernelom

Nie vždy je možné oddeliť kategórie rovinou

Hoci SVM pripúšťa aj chyby klasifikácie, je lepšie sa im vyhnúť





Kernel trick

Minimalizačný algoritmus SVM často potrebuje spočítať vzdialenosť medzi dvomi vzorkami

Na to by musel pomocou kernelu vypočítať všetky obrazy vzoriek a to je časovo náročné

Preto vždy volíme len také kernely, ktorých vzdialenosti obrazov sa dajú spočítať priamo zo vzdialenosti vzorov. A tým pádom nemusíme obrazy vôbec počítať.

Všeobecná schéma regresie

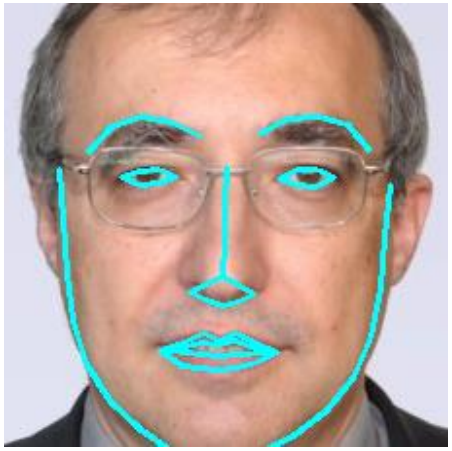
obraz

Metóda 1

feature vector

$[c_1, c_2, \ldots, c_n]$

Metóda 2



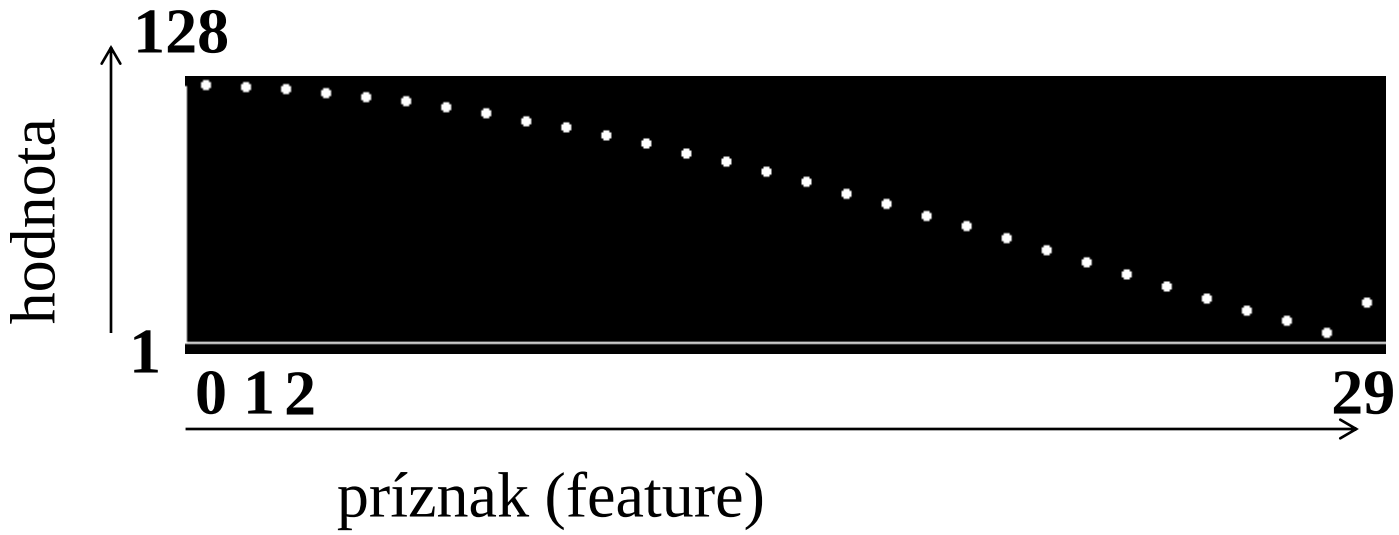
<

porovnanie vopred vybraných pixelov

Decision tree (Kazemi)

Regresia

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
f	128	127	126	124	122	120	117	114	110	107	103	99	94	90	85	80	74	69	63	58	52	46	40	34	28	22	16	11	5	20



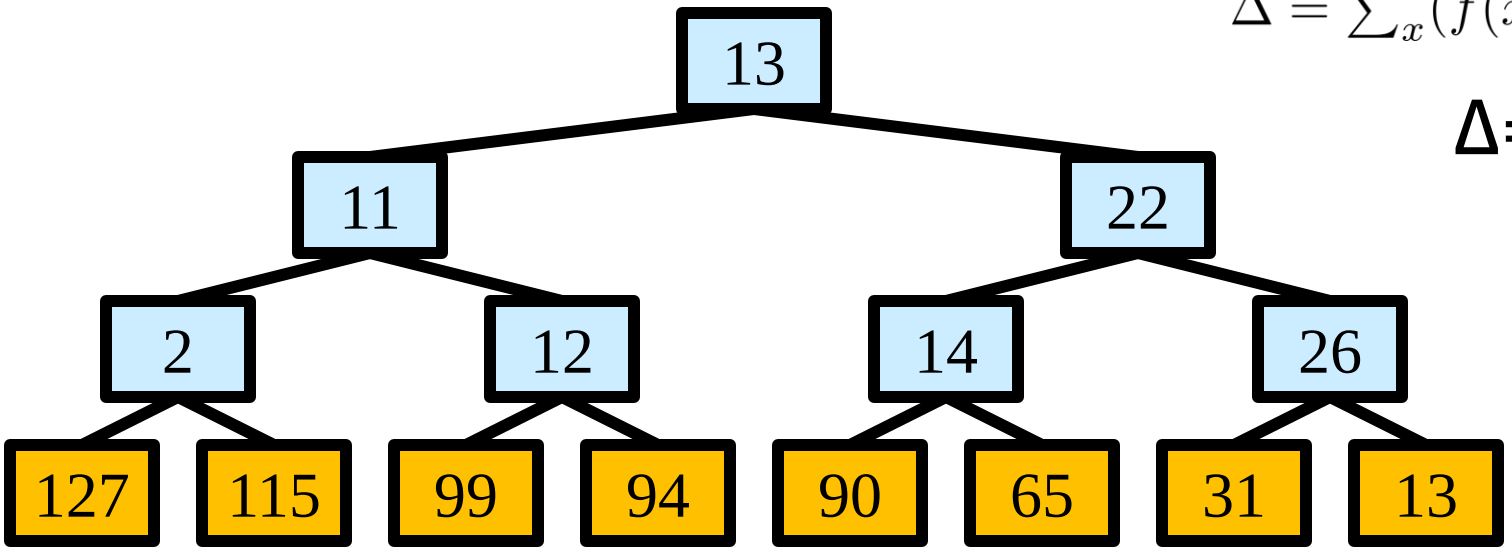
- Učíme stroj nejakú funkciu ku ktorej máme sadu vzoriek (príznak, hodnota)

Slabý regresor (Regresný strom)

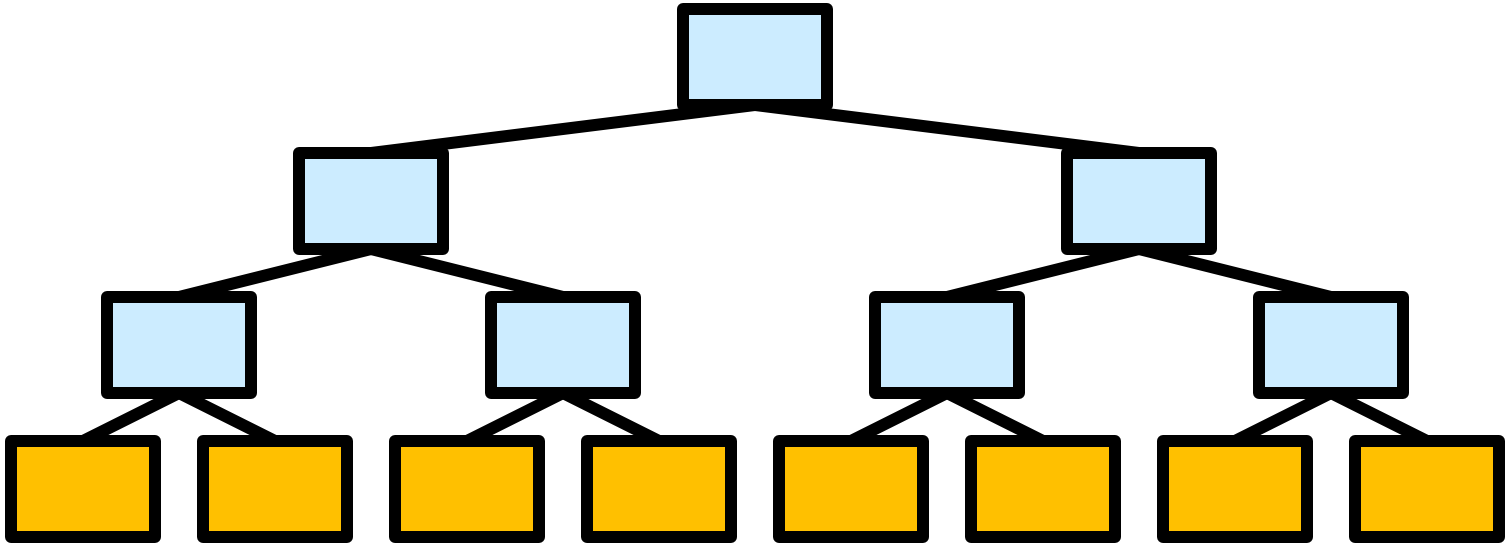
x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
f	128	127	126	124	122	120	117	114	110	107	103	99	94	90	85	80	74	69	63	58	52	46	40	34	28	22	16	11	5	20
g	127	127	115	115	115	115	115	115	115	115	115	99	94	90	65	65	65	65	65	65	65	65	31	31	31	31	13	13	13	13

$$\Delta = \sum_x (f(x) - g(x))^2$$

Δ=2126



Ako získame regresný strom?

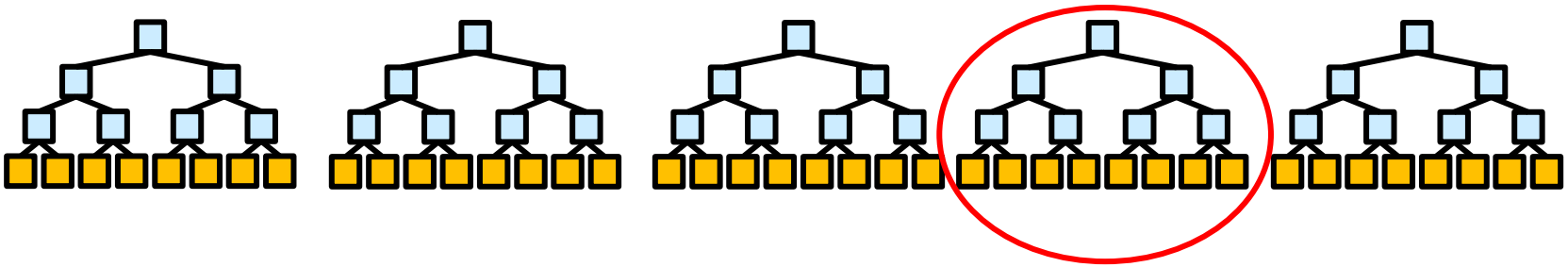


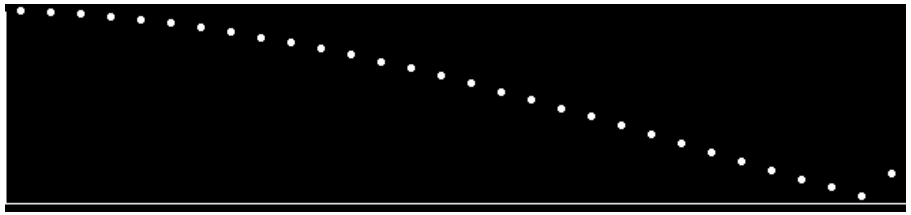
- Povieme si nech je hĺbky 3
- Vo vnútornom vrchole dáme porovnanie na náhodne zvolený (deliaci) príznak (menší vľavo, väčší vpravo)
- V liste dáme priemer z hodnôt, ktoré zodpovedajú príznakom s ktorými sa do listu dostaneme

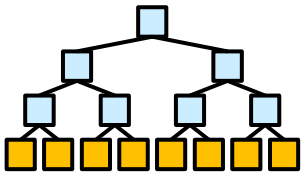
Ako nájsť strom s malou chybou?

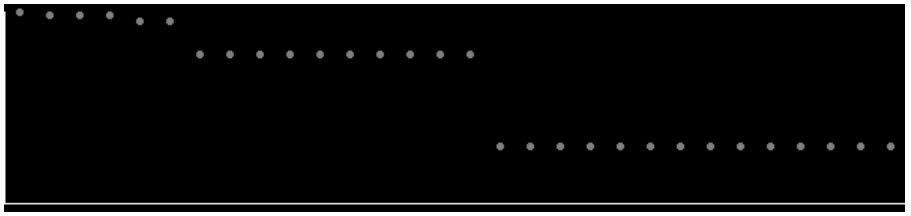
- Takto vygenerovaný strom nemusí byť práve najvhodnejší, čo s tým?

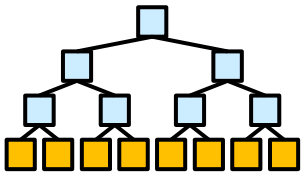
1. Pri voľbe deliaceho príznaku môžeme uvažovať viac možností a vybrať si lepšiu (takú, ktorá aproximuje dáta s menšou chybou, t.j. súčet druhých mocnín rozdielov vzoriek v ľavom podstrome od ich priemeru plus analogický súčet pre vzorky v pravom podstrome je najmenší)
2. Môžeme vygenerovať viac stromov a vybrať ten s najmenšou chybou

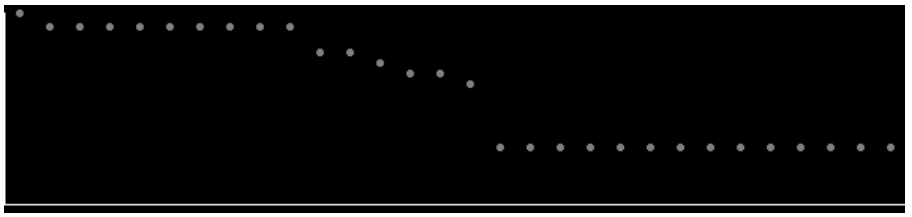


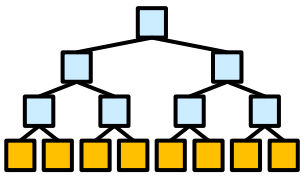


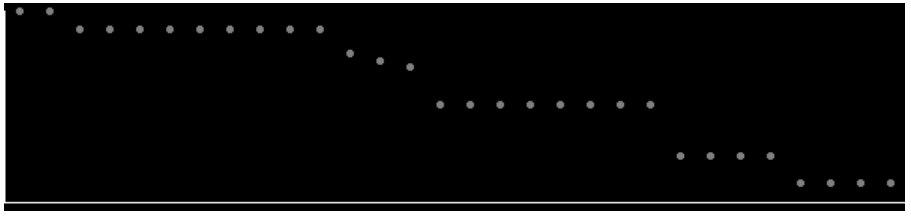


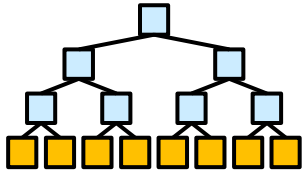


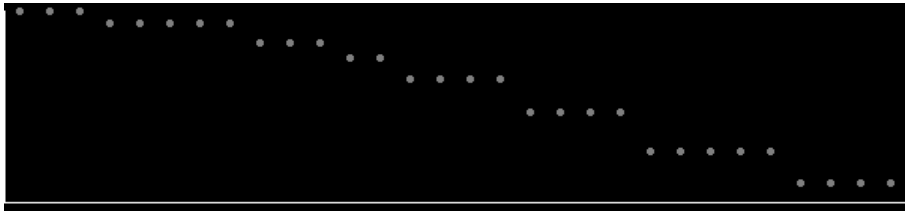












RegressionTree

Δ=2000

Δ=1200

Δ=820

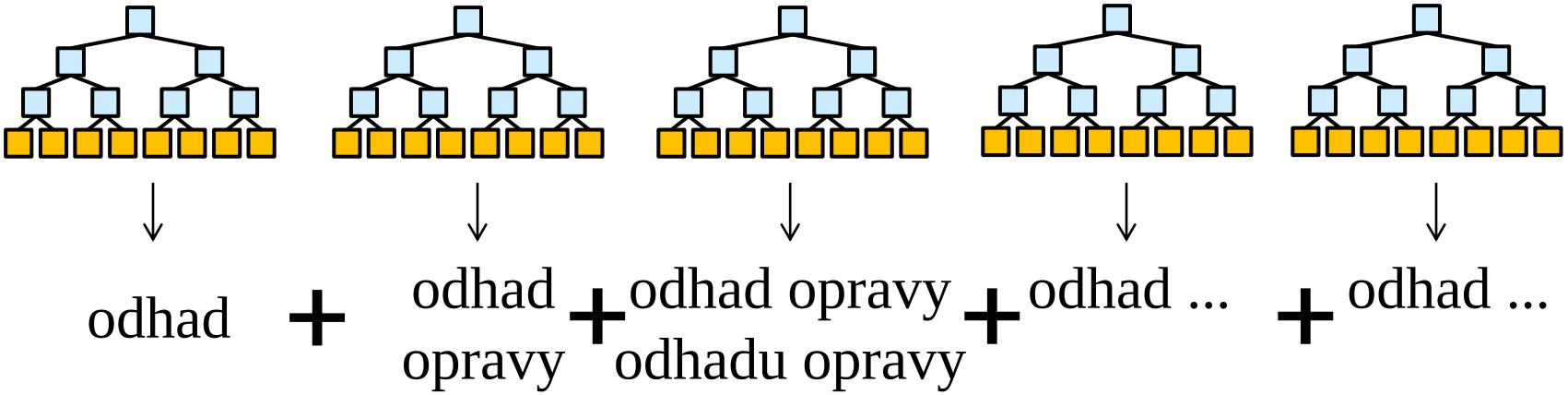
Δ=436

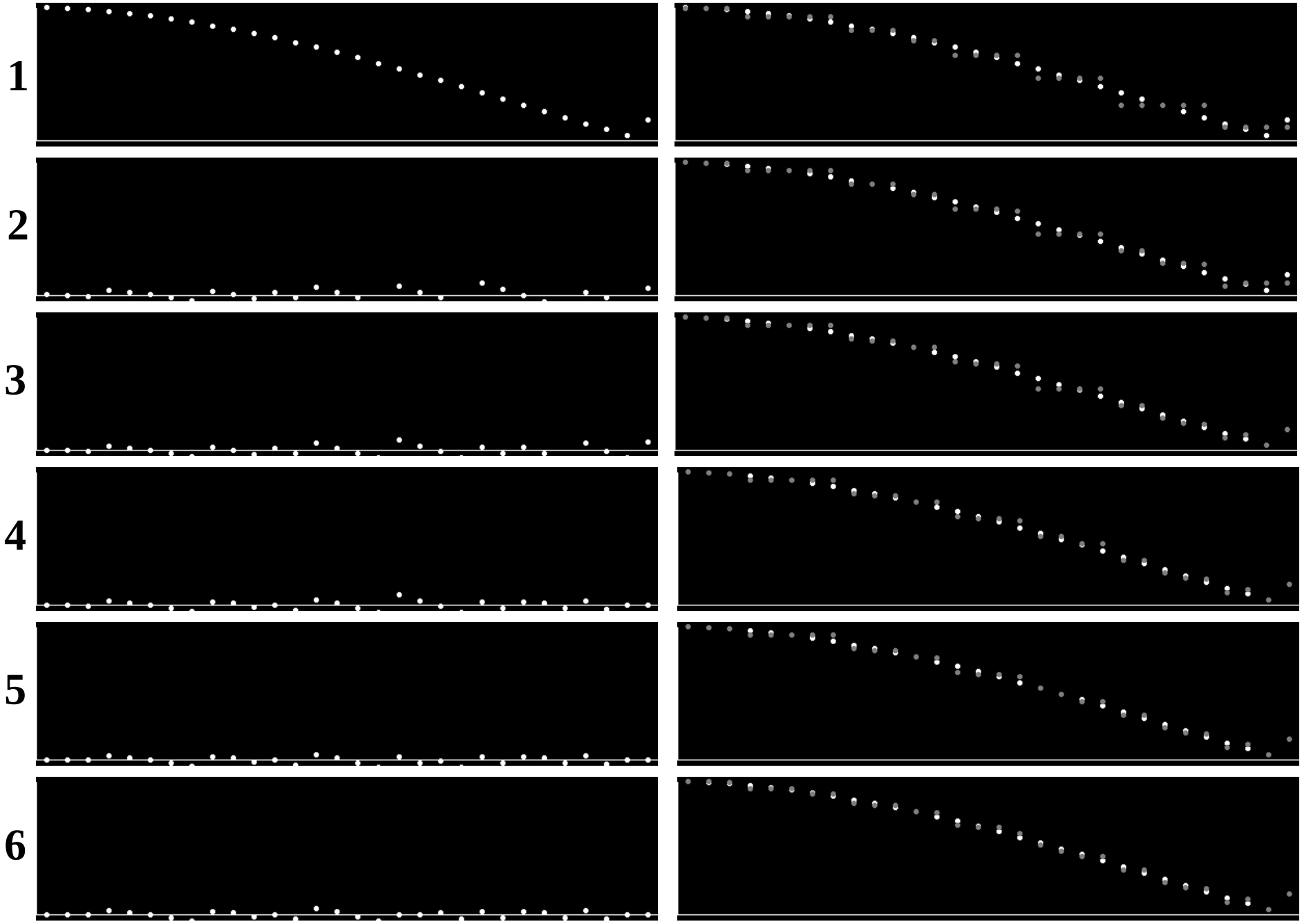
Δ=4000

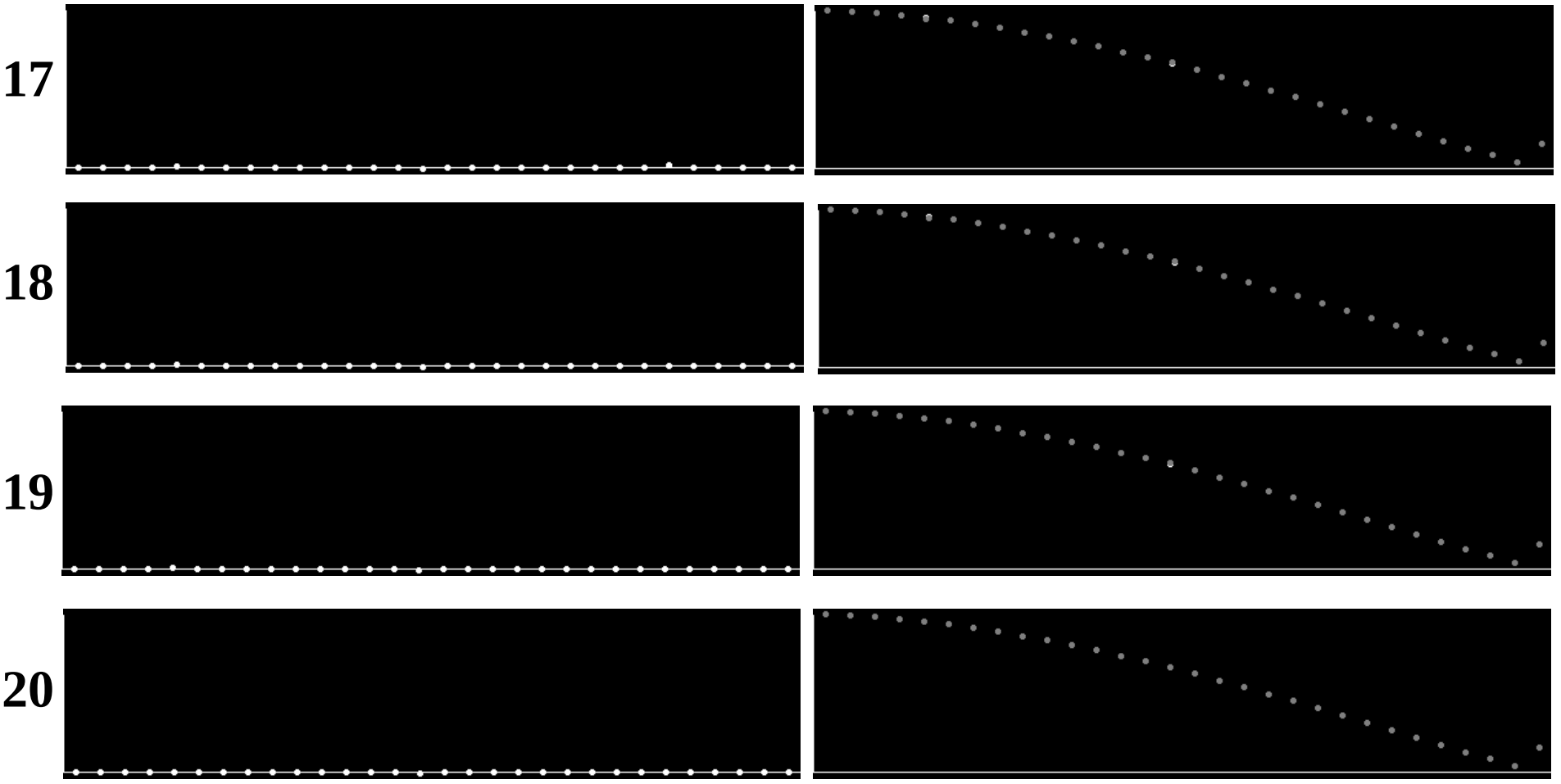
Gradient Boosting

- Ako to vylepšiť ?

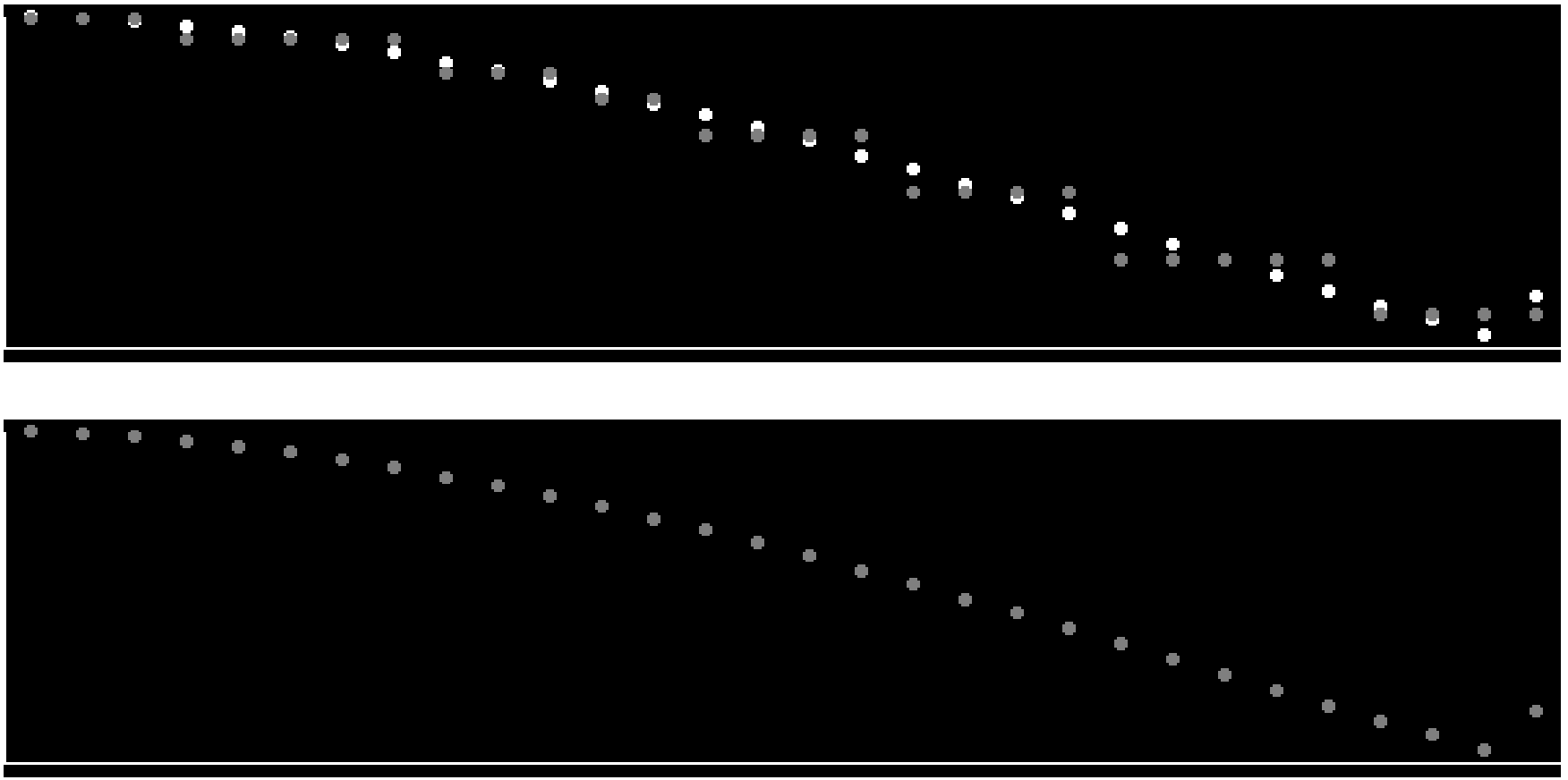
1. Vyrátame rozdiel medzi tým, čo sa strom naučil a čo sme ho chceli naučiť
2. Tento rozdiel učíme ďalší slabý regressor
3. A tak ďalej až po určitý zvolený počet regresorov
4. Odozva takejto sústavy regresorov bude súčet ich odozviev na príznak




1
2
3
4
5
6

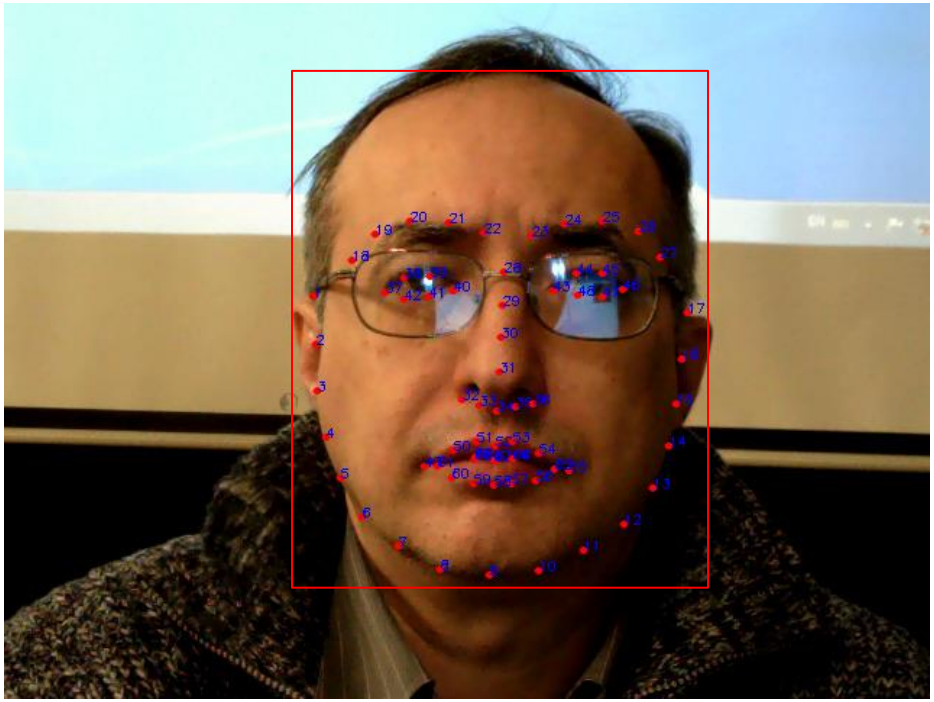

17
18
19
20
Δ=0

Gradient Boosting



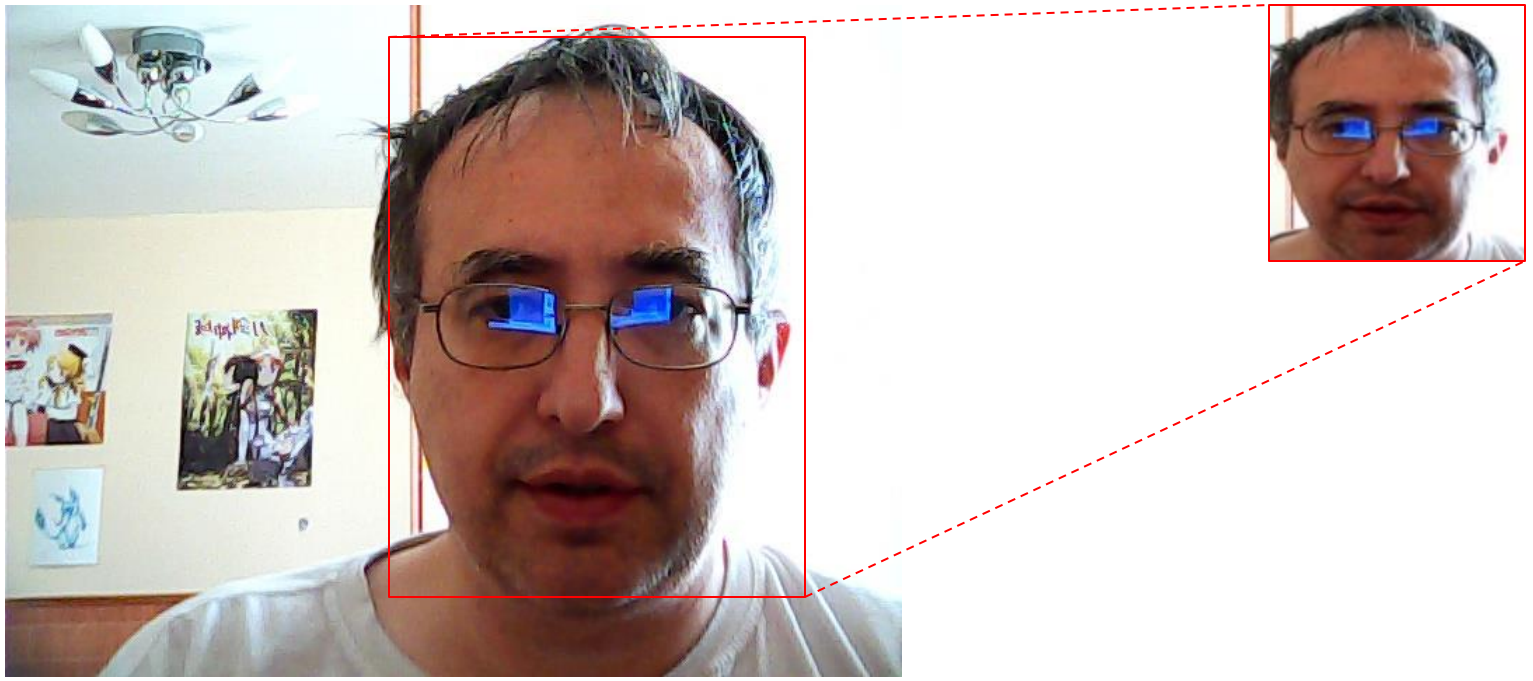
Kazemiho regressor

- Urobíme mnoho fotiek tvárí a označíme na ne potrebné body, čím ich bude viac, tým lepšie to bude fungovať.
- Popíšeme pritom tiež kde na obraze sa tvár nachádza.



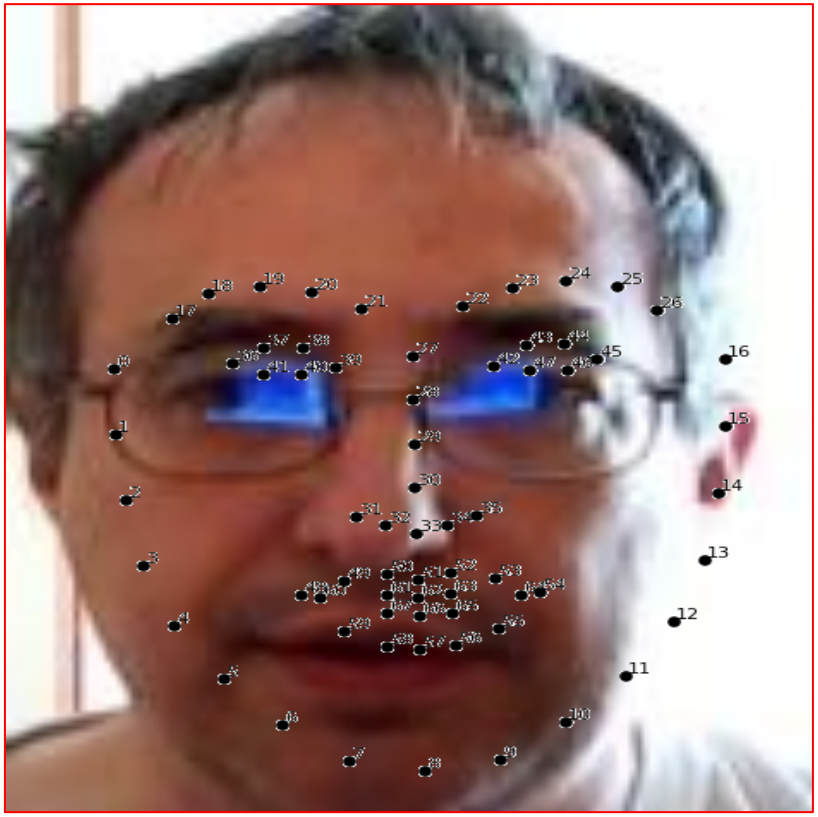
Predspracovanie

- Remapujeme na štandardnú veľkosť

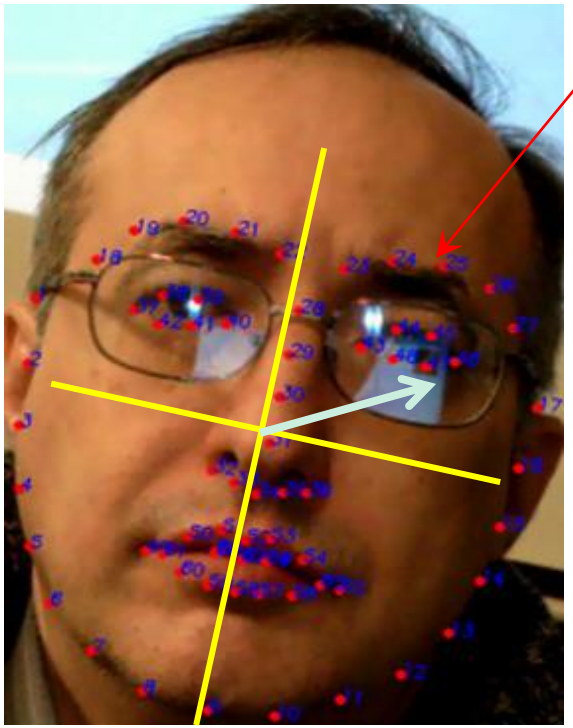


- Riešime tým zväčšenie či zmenšenie spracúvanej tváre

Iniciálny stav – priemer vzoriek



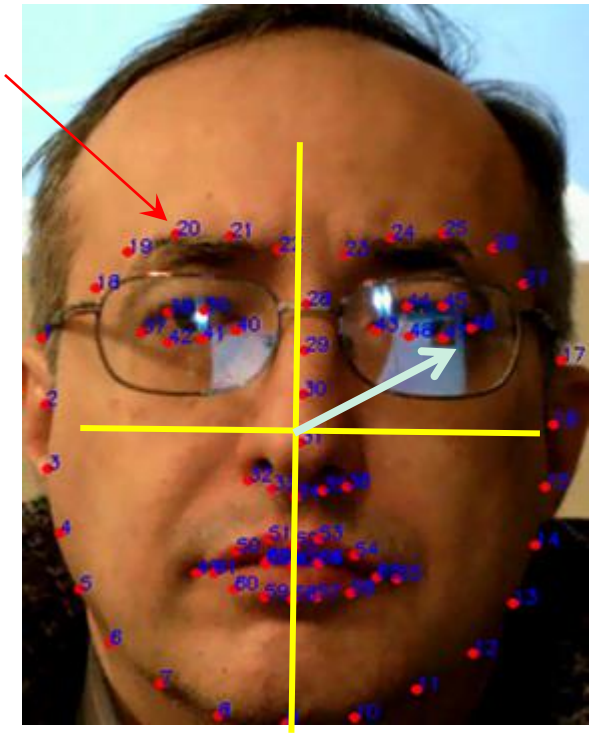
- Ako relativizovať súradnice pixelov použitých pri deliacich porovnávaniach voči aktuálnemu odhadu landmarkov?
- Tvár na spracúvanom obraze môže byť posunutá i mierne pootočená



aktuálny odhad
iniciálny priemer

Riešenie:

Similarity
transform



Similarity transform

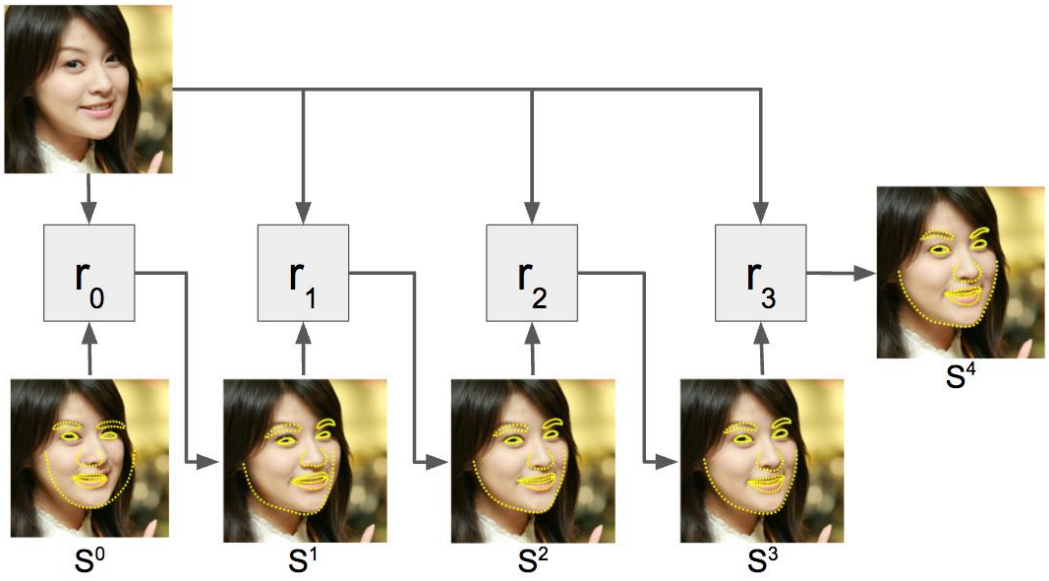
- Je afínne zobrazenie, ktoré jednu sadu bodov čo najpresnejšie umiestňuje na inú sadu bodov
- V našom prípade ide o momentálny odhad polohy landmarkov a priemerné rozloženie landmarkov (teoreticky by rovnako poslúžilo akékoľvek pevne zvolené)
- Keď sa potom 68 landmarkov pozerá na dva pixely umiestnené k nim podľa určitého vektora, tak vlastne sa pozerá vždy na rovnaké súradnice voči priemernému rozloženiu landmarkov na obraze získanému cez similarity transform (nemusíme samozrejme transformovať celý obraz, len dva pixle)

Features

- Pracuje sa s obrázkami 128x128, čo je stále veľmi veľa možností pre dvojice prixelov, ktoré budeme porovnávať
- Preto náhodne a spravodlivo vyberieme len určitý počet pixelov a len tie používame na porovnávanie
- (pritom je to skôr počet súradníc pixelov)

Kazemiho regressor

je kaskádny regresor



- Trénovanie trvá dlho, ale 10 x 500 x 4 porovnaní pixelov pri použití je nič. Preto je veľmi rýchly, < 1ms

- 10 kaskád
- v každej regresor
- v ňom 500 slabých regresných stromov hĺbky 4
- 400 pixelov náhodne vybraných zo 128x128
- 20 pokusov pri voľbe deliaceho príznaku