Praktikum zo strojového učenia a umelej inteligencie na vizuálnych dátach

Andrej Lúčny

Katedra aplikovanej informatiky FMFI UK

lucny@fmph.uniba.sk

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

www.agentspace.org/praktikum

Všeobecná schéma klasifikácie

obraz

feature vector

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

Metóda 1 → Metóda 2 → kategória (klasifikátor)

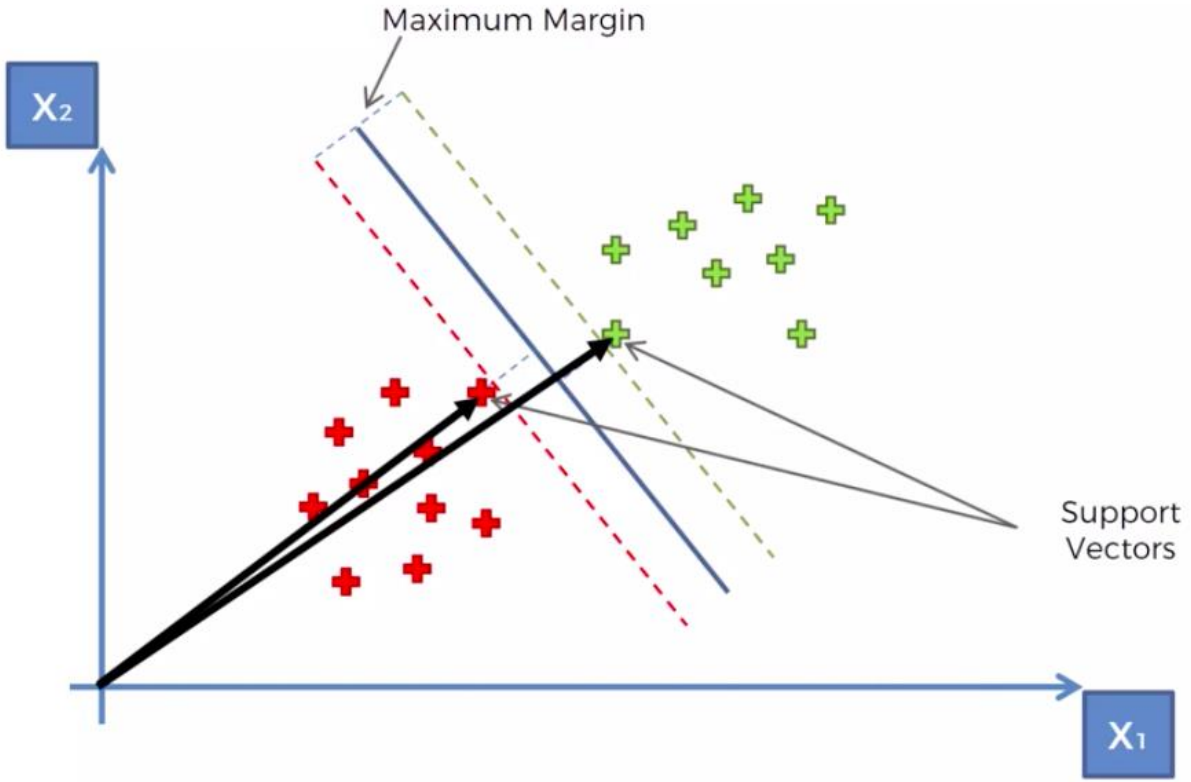
SVM

Haar Decision tree (Viola-Jones)

→ hodnota (regresor)

BP Decision tree (Kazemi)

Support Vector Machines (SVM)

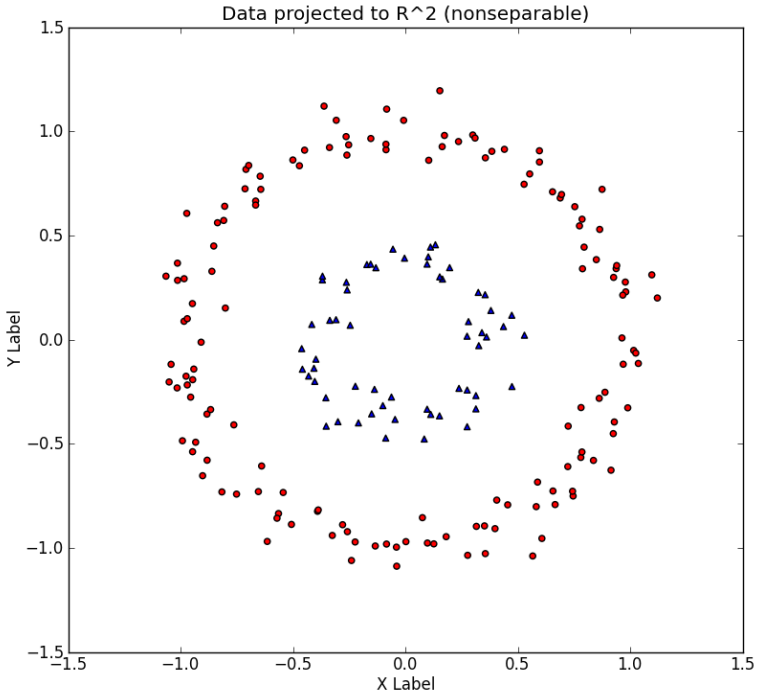


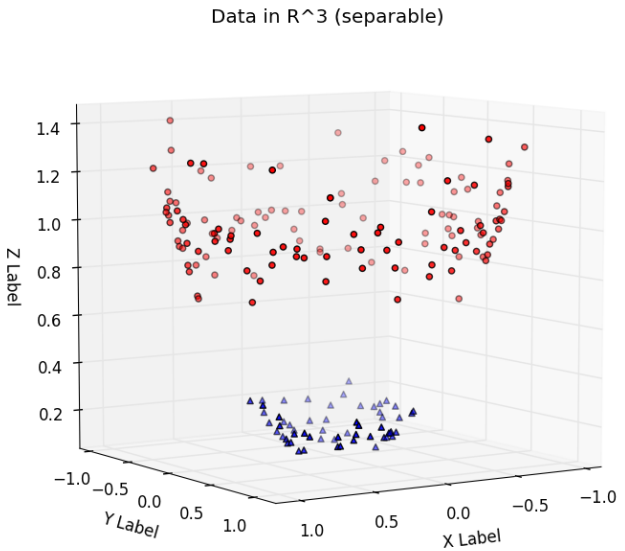
Klasifikátor, ktorý položí takú nadrovinu medzi dáta, ktorá ich čo najlepšie oddeľuje.

SVM s kernelom

Nie vždy je možné oddeliť kategórie rovinou

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





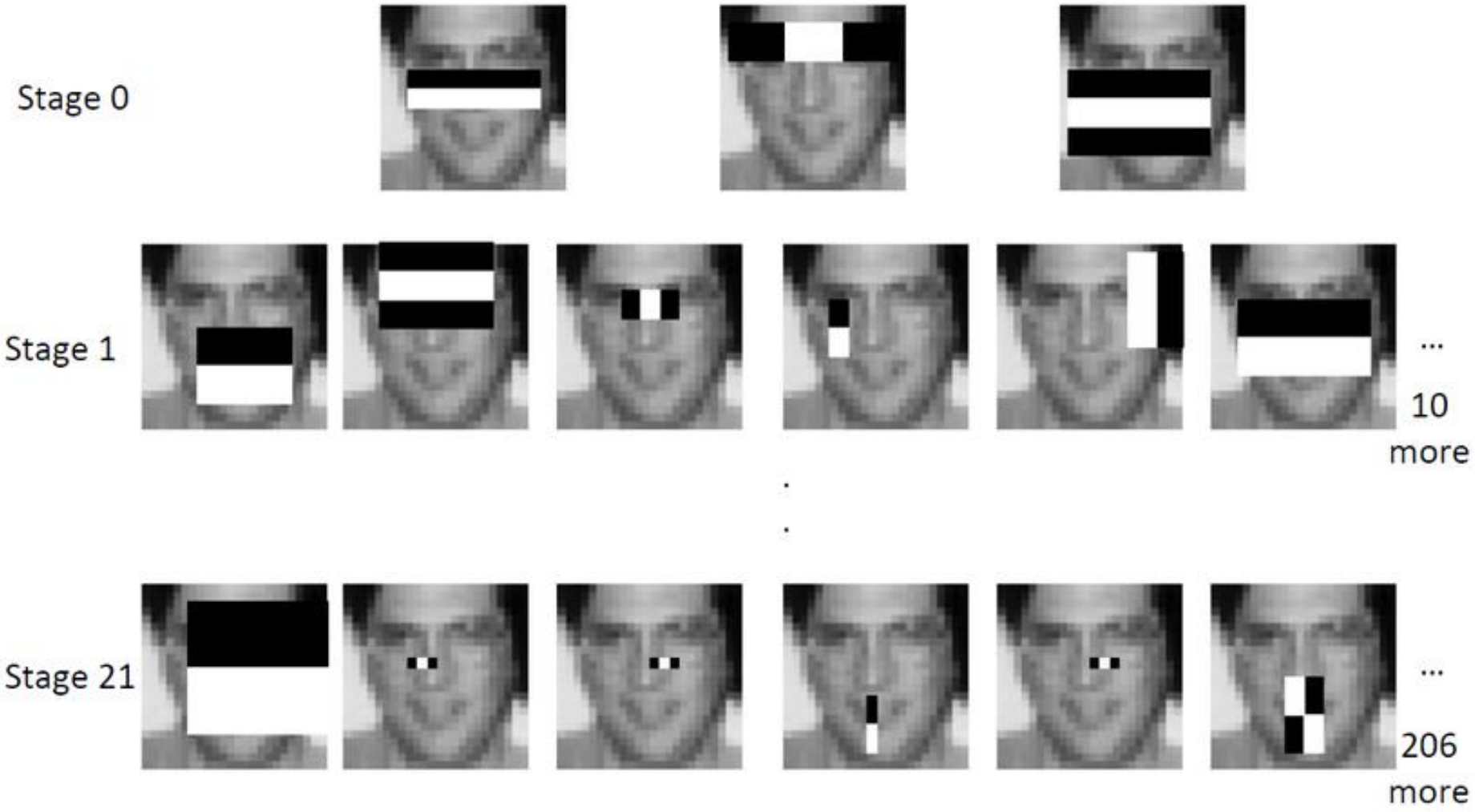
Kernel trick

Minimalizačný algoritmus SVM často potrebuje spočítať vzdialenosť medzi dvomi vzorkami

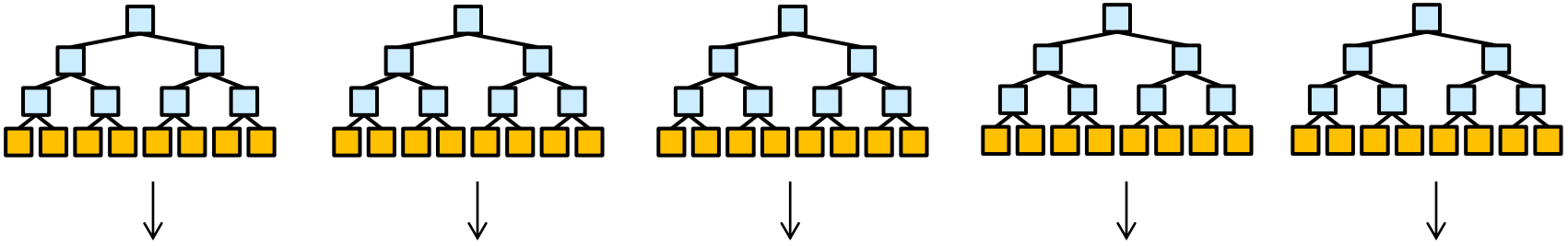
Na to by musel pomocou kernelu vypočítať všetky obrazy vzoriek a to je časovo náročné

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

Haar Features



Cascade classifier



NO or MAY-BE

If MAYBE, then NO or MAY-BE

If MAYBE & MAYBE, then NO or MAY-BE

…

NO or YES

Viola-Jones algoritmus: Haar features + cascade classifier

Sliding Window Algorithm

Kaskádny klasifikátor od Viola-Jones nám umožní rozhodnúť, či nejaký obrázok je tvár, alebo nie.

Ako však nájsť tvár na obrázku?

Každý klasifikátor sa dá premeniť hrubou silou na detektor pomocou algoritmu používajúceho plávajúce okno:

- otestujeme všetky možné výskyty
- zosumarizujeme výskyty cez Non-Maximum Suppression

Binary Pattern

- ak je jeden pixel svetlejší od druhého na jednom obraze, bude tento vzťah zachovaný aj obraze so zmenenou jasnosťou, kontrastom či ostrosťou.

Všeobecná schéma regresie

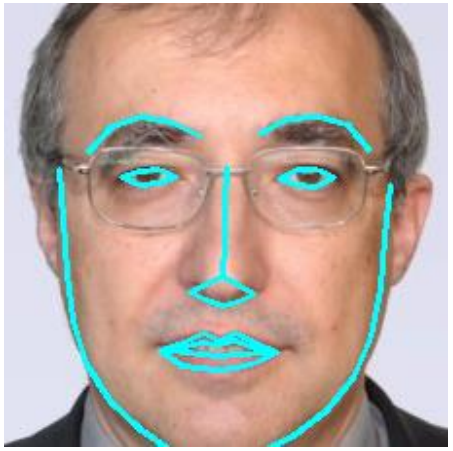
obraz

Metóda 1

feature vector

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

Metóda 2



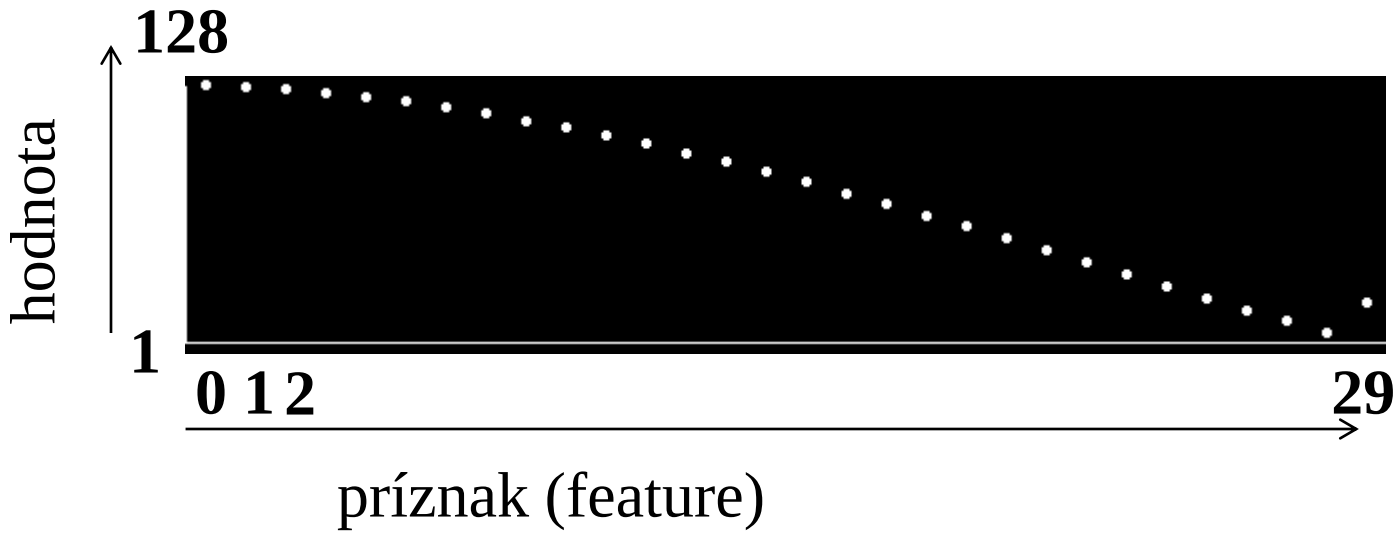
<

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



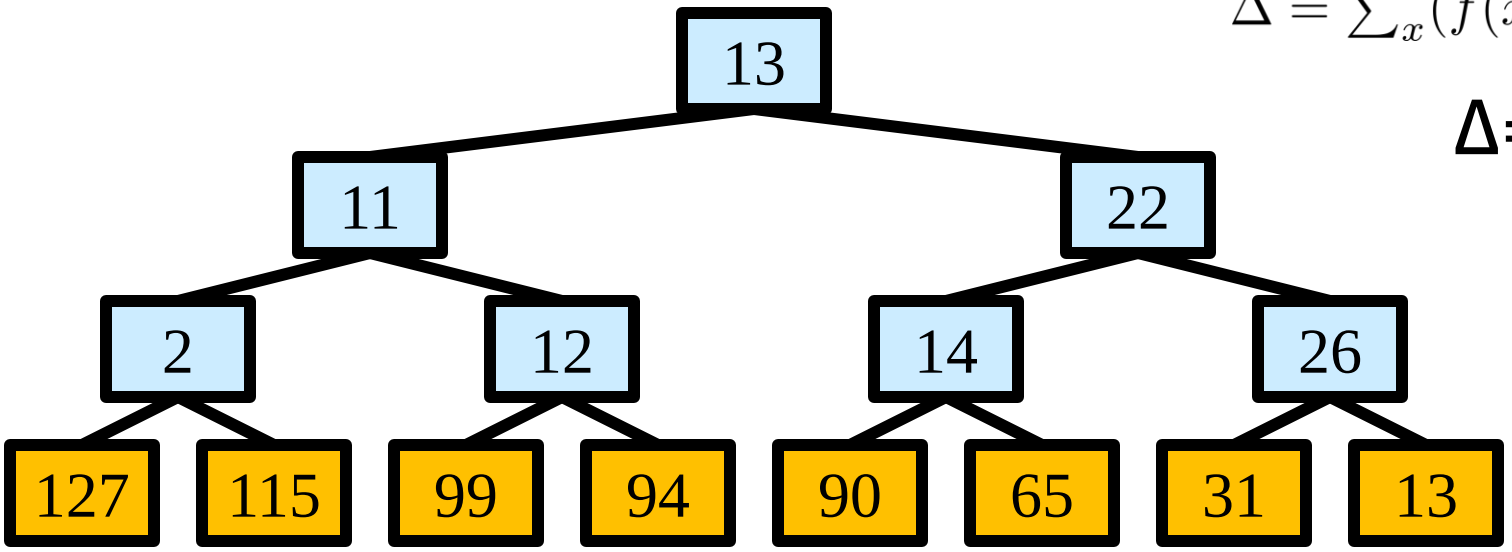
- Učíme stroj nejakú funkciu ku ktorej máme sadu vzoriek (príznak, hodnota)

Slabý regresor (Regresný strom)

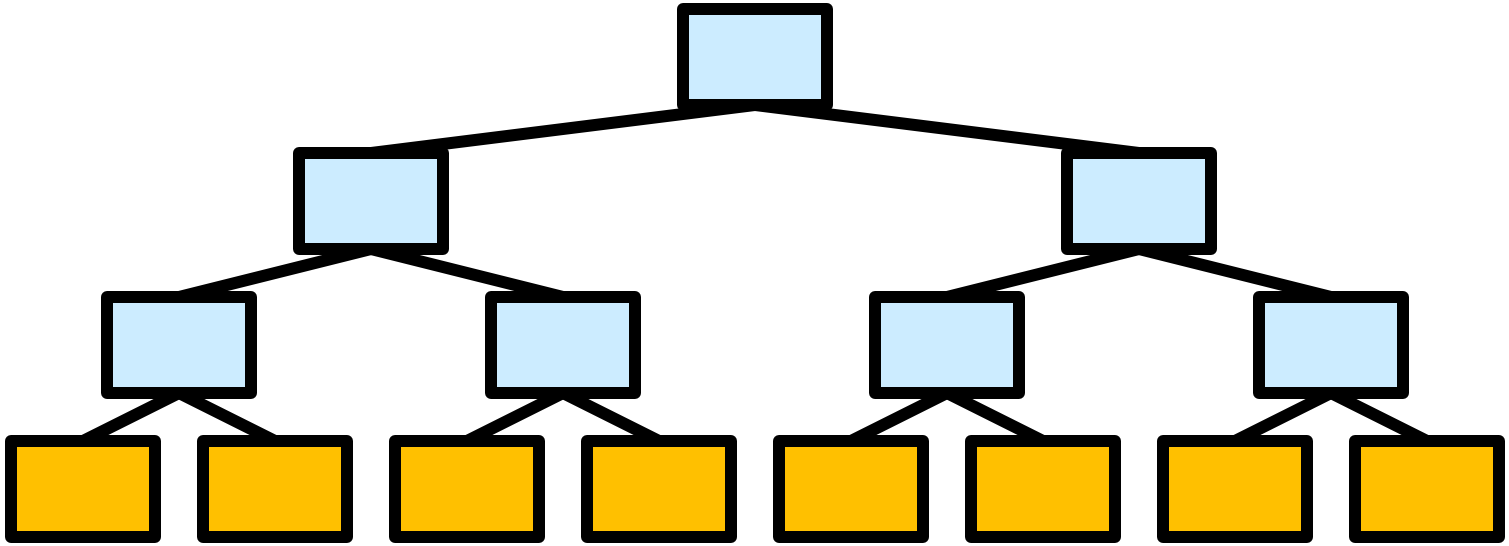
x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
f	128	127	126	124	122	120	117	114	110	107	103	99	94	90	85	80	74	69	63	58	52	46	40	34	28	22	16	11	5	20
g	127	127	115	115	115	115	115	115	115	115	115	99	94	90	65	65	65	65	65	65	65	65	31	31	31	31	13	13	13	13

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

Δ=2126



Ako získame regresný strom?

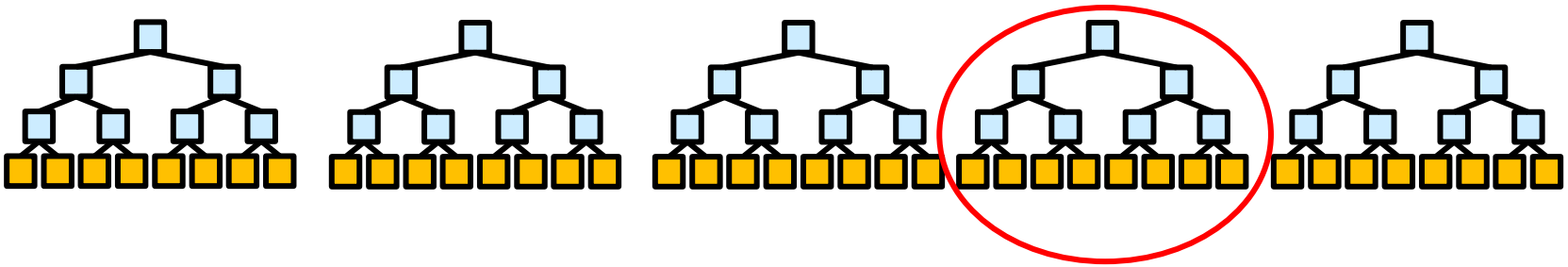


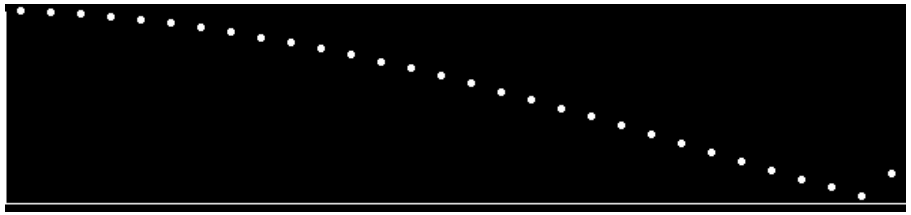
- Povieme si nech je hĺbky 3
- Vo vnútornom vrchole dáme porovnanie na náhodne zvolený (deliaci) príznak (menší vľavo, väčší vpravo)
- V liste dáme priemer z hodnôt, ktoré zodpovedajú príznakom s ktorými sa do listu dostaneme

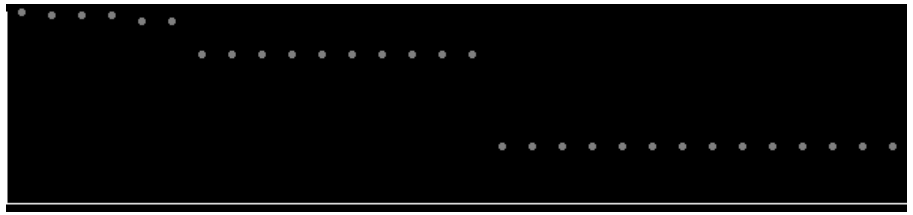
Ako nájsť strom s malou chybou?

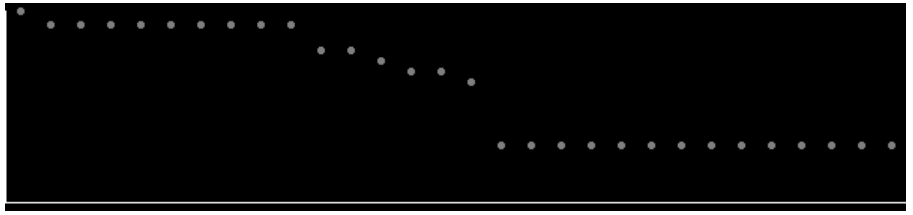
- Takto vygenerovaný strom nemusí byť práve najvhodnejší, čo s tým?

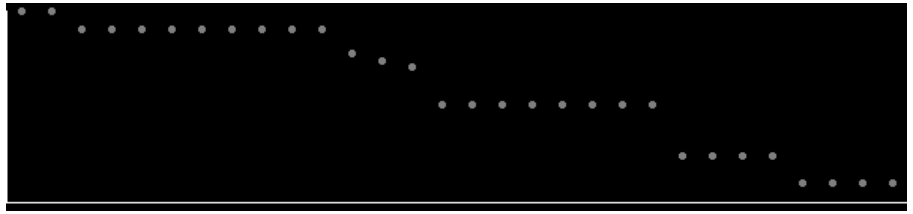
1. Pri voľbe deliaceho príznaku môžeme uvažovať viac možností a vybrať si lepšiu (takú, ktorá aproximuje dáta s menšou chybou, t.j. súčet druhých mocnín rozdielov vzoriek v ľavom podstrome od ich priemeru plus analogický súčet pre vzorky v pravom podstrome je najmenší)
2. Môžeme vygenerovať viac stromov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

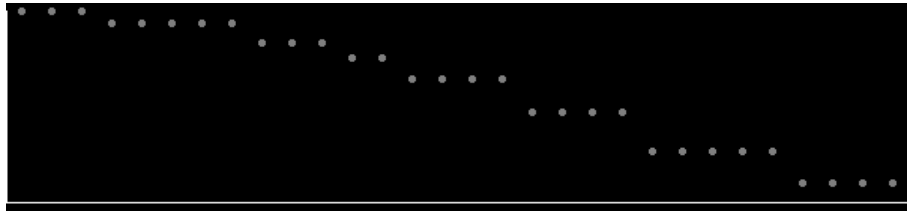


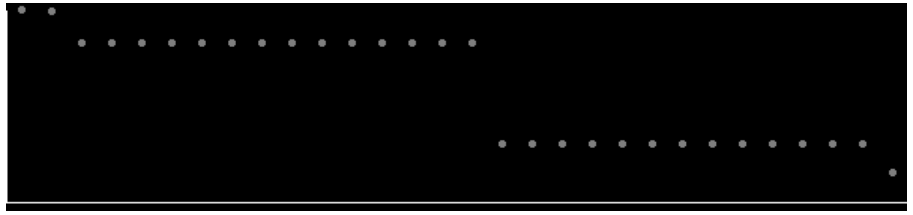


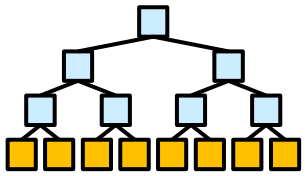


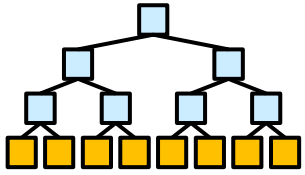


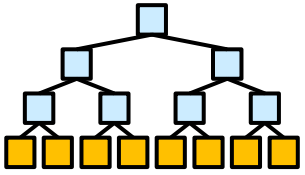


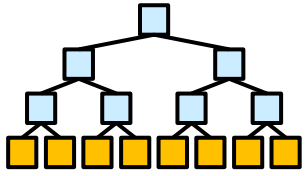


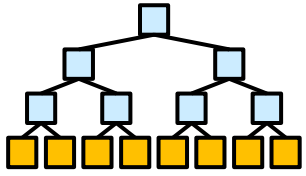












RegressionTree

Δ=2000

Δ=1200

Δ=820

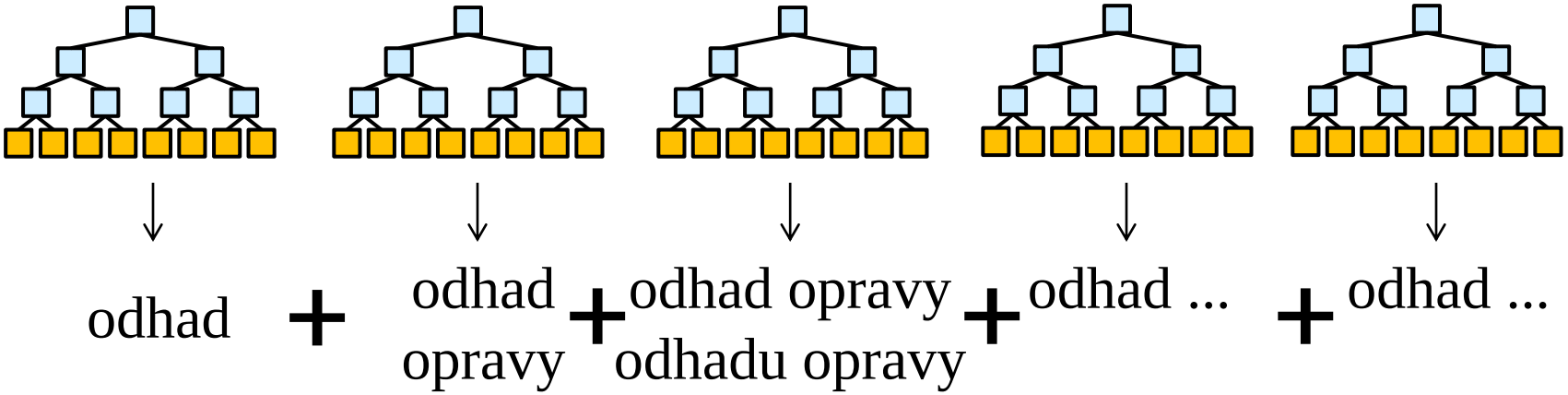
Δ=436

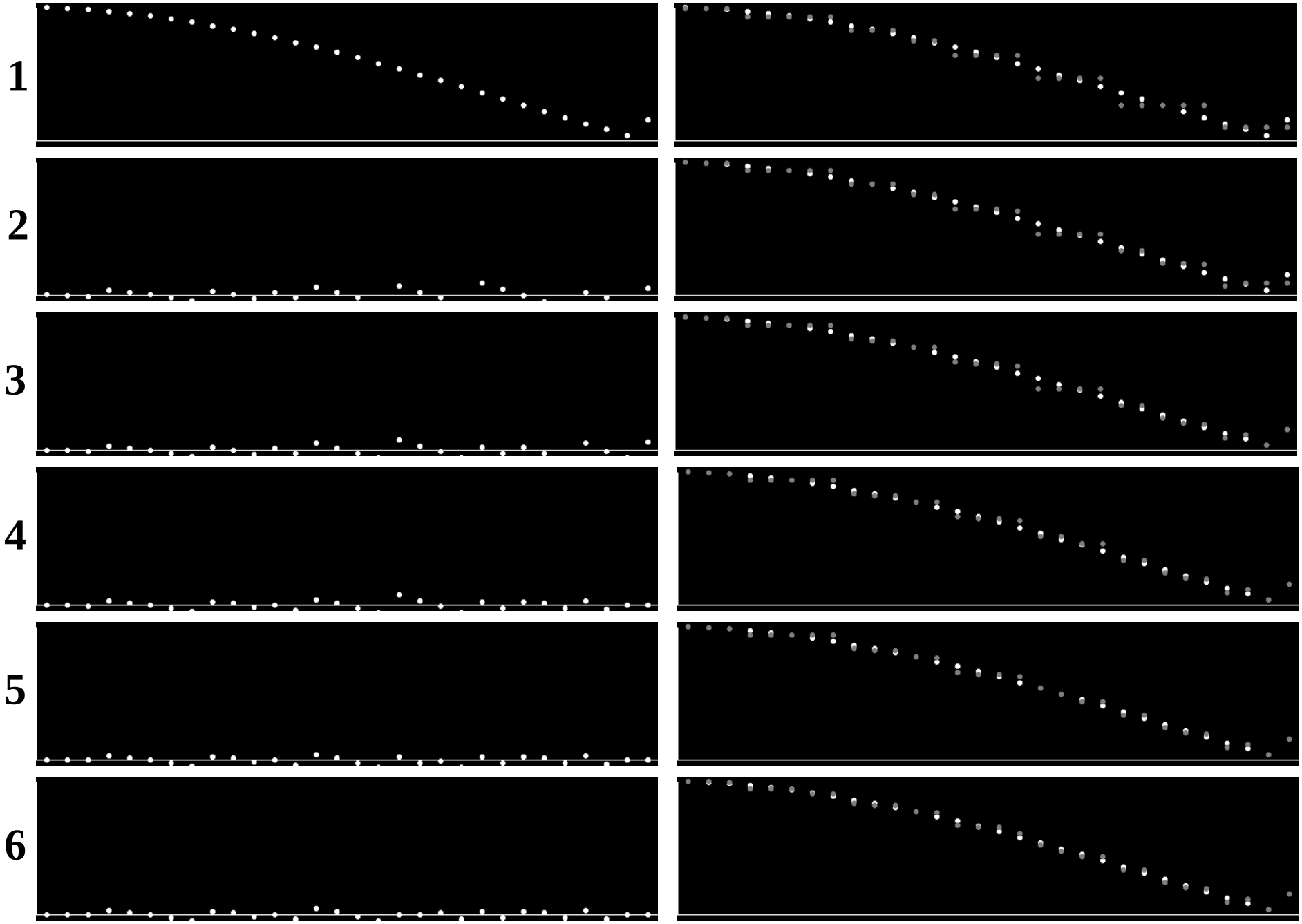
Δ=4000

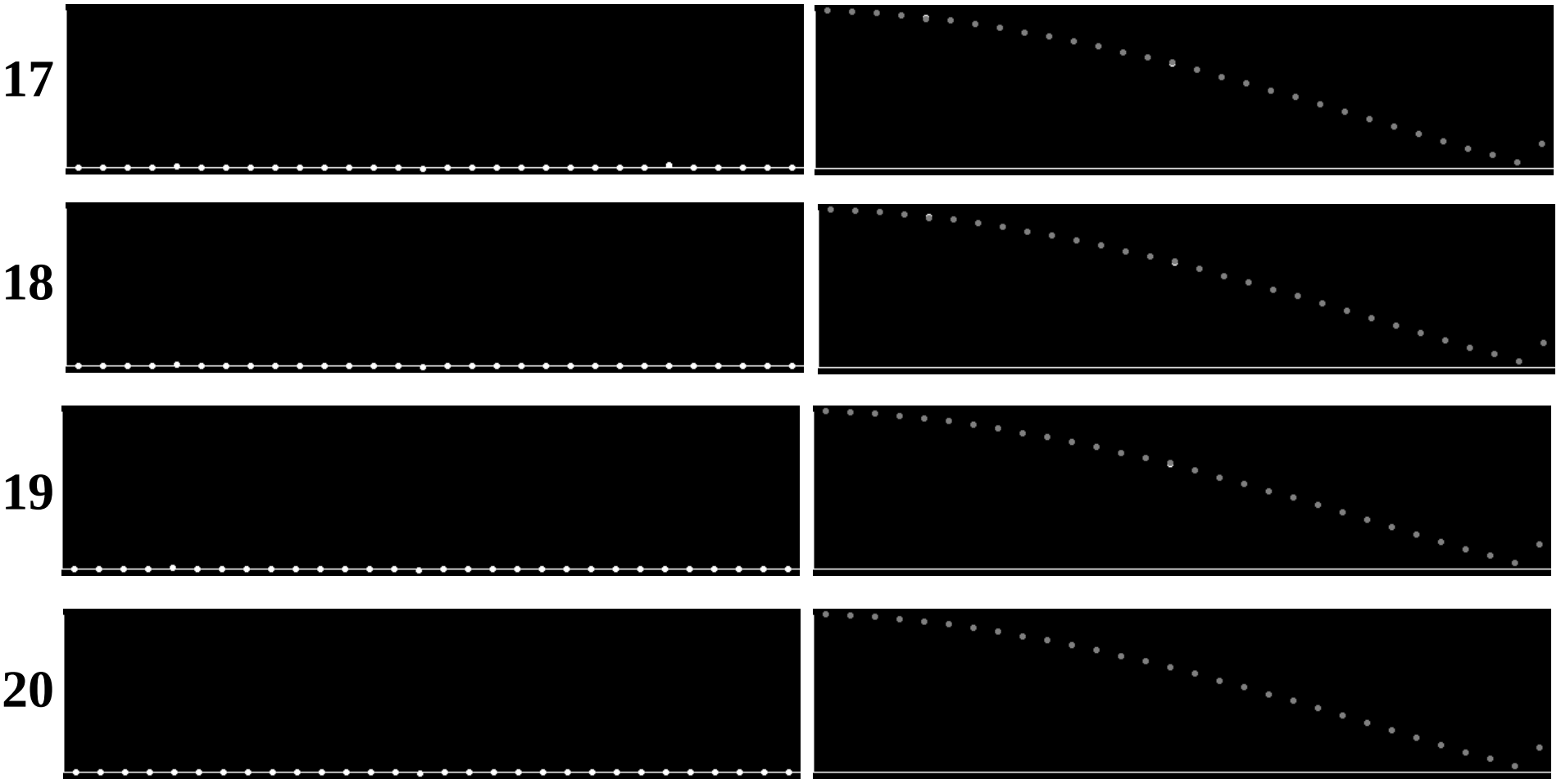
Gradient Boosting

- 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



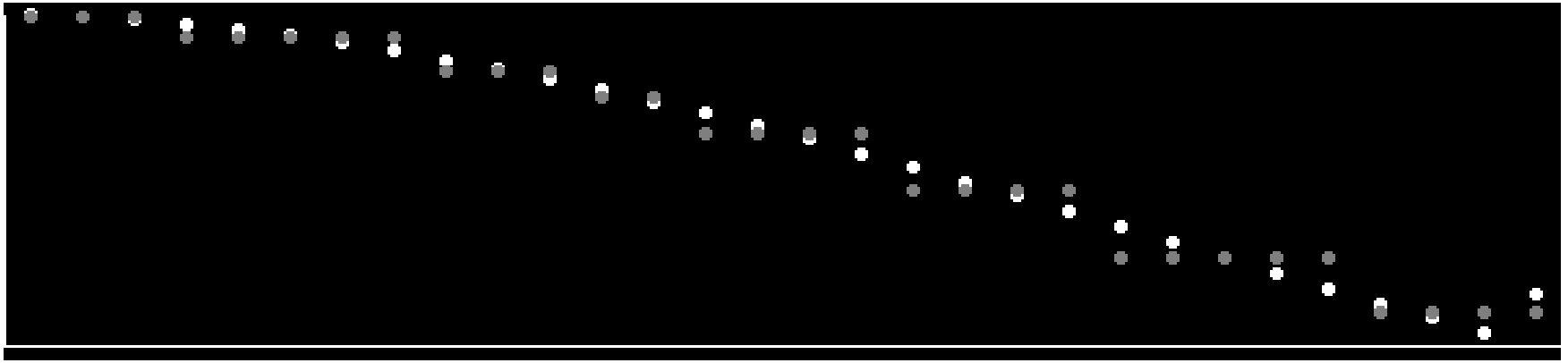

1
2
3
4
5
6



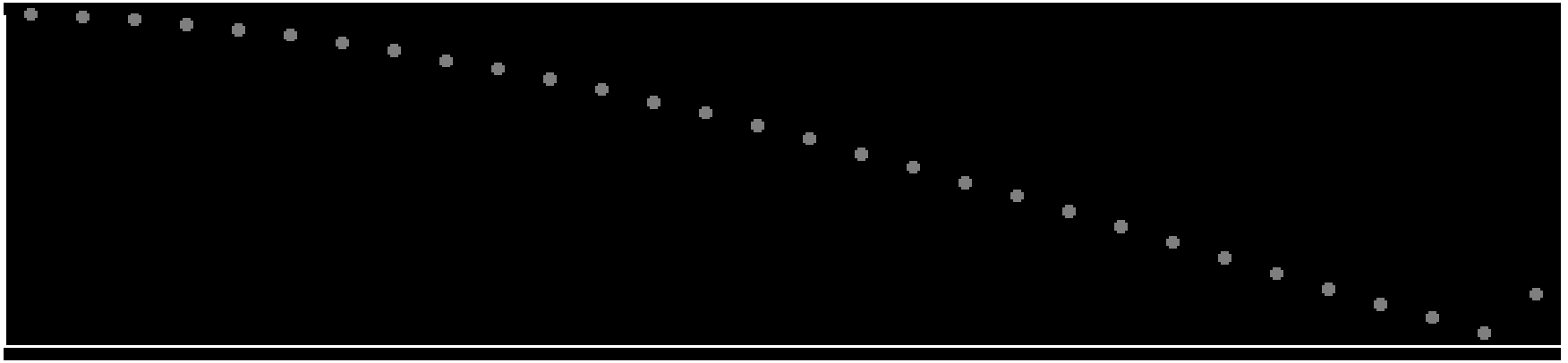
Δ=0

Gradient Boosting

1

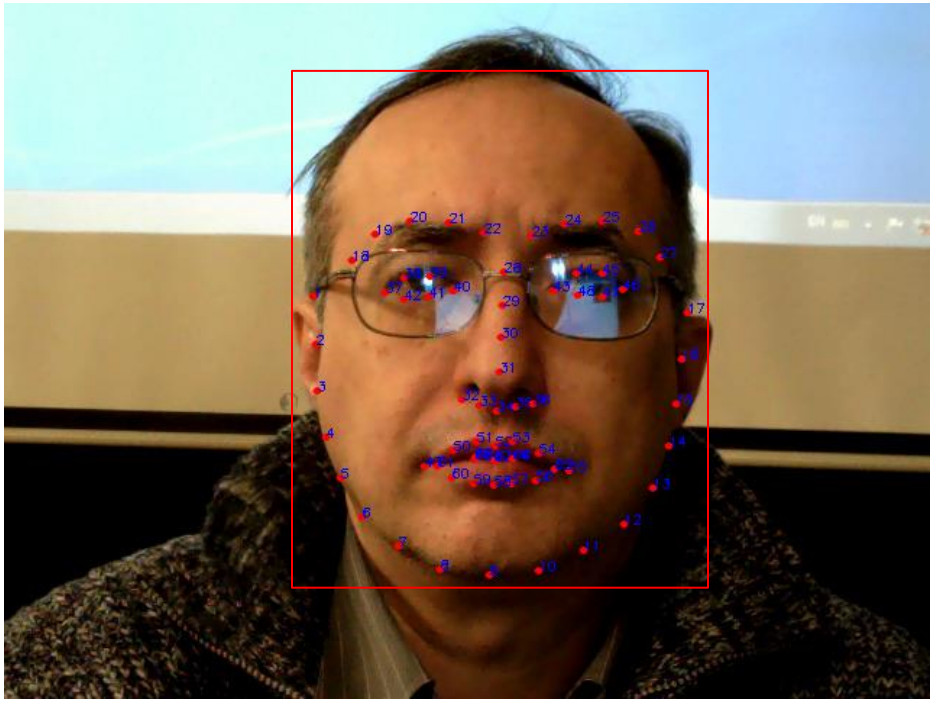


20



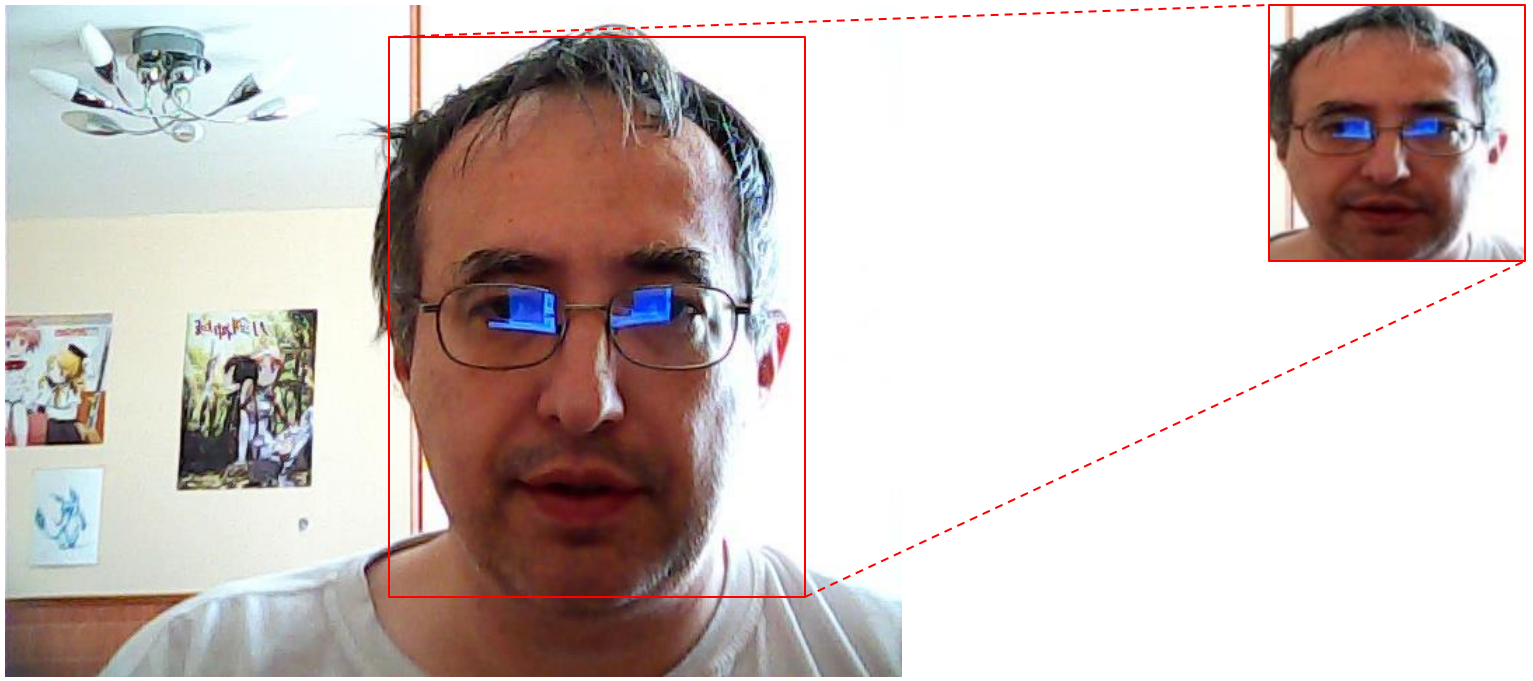
Kazemiho prediktor

- 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.



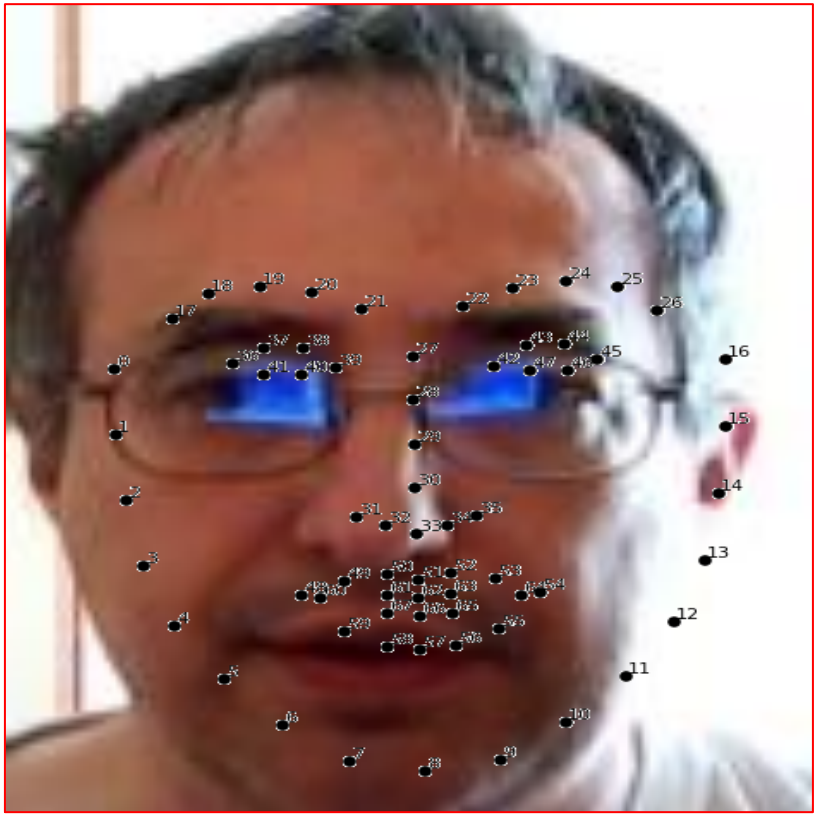
Predspracovanie

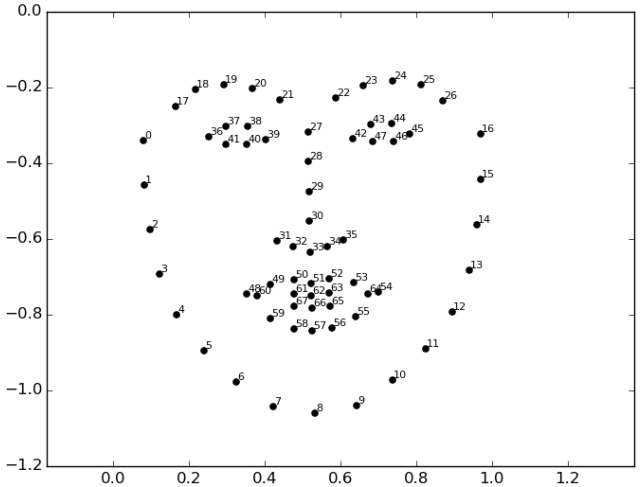
- Remapujeme na štandardnú veľkosť



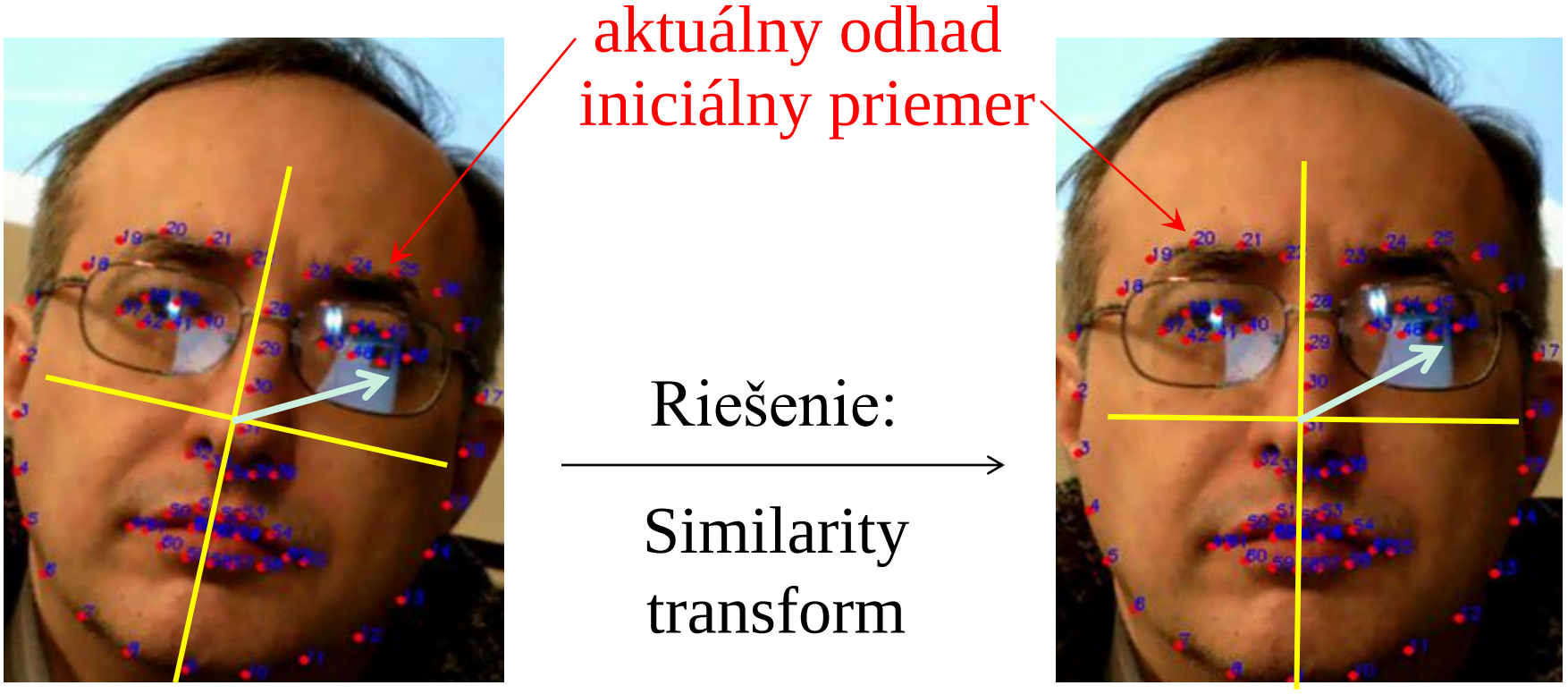
- Riešime tým zväčšenie či zmenšenie spracúvanej tváre

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



Similarity transform

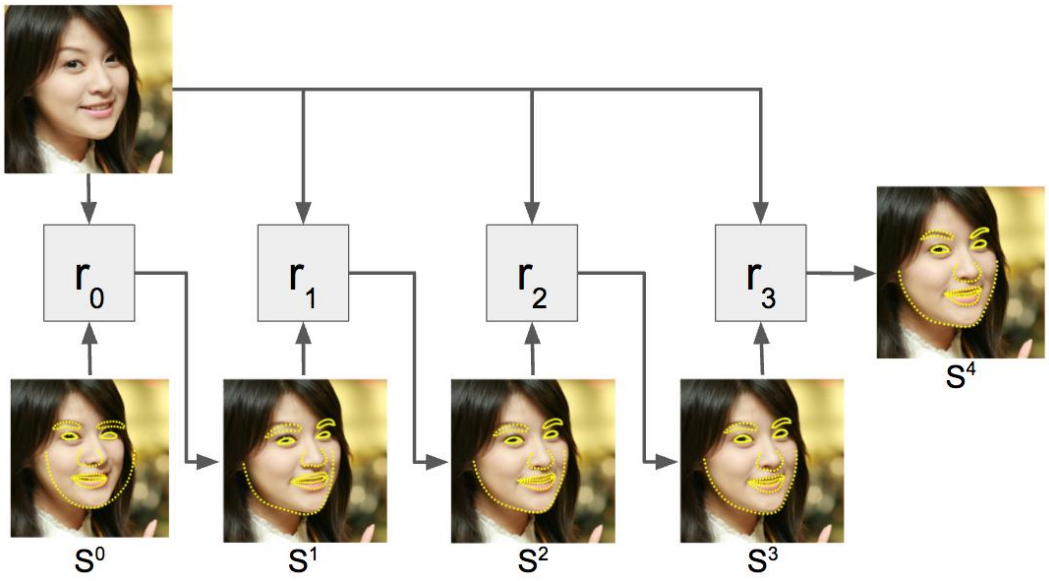
- Je afínne zobrazenie, ktoré jednu sadu bodov čo najpresnejšie umiestňuje na inú sadu bodov
- V našom prípade ide o momentálny odhad polohy landmarkov a priemerné rozloženie landmarkov (teoreticky by rovnako poslúžilo akékoľvek pevne zvolené)
- Keď sa potom 68 landmarkov pozerá na dva pixely umiestnené 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 prediktor

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