Seminár Robotika.SK

Ako naučiť robota vidieť ľudskú tvár pomocou knižnice Dlib

Andrej Lúčny

Katedra aplikovanej informatiky FMFI UK

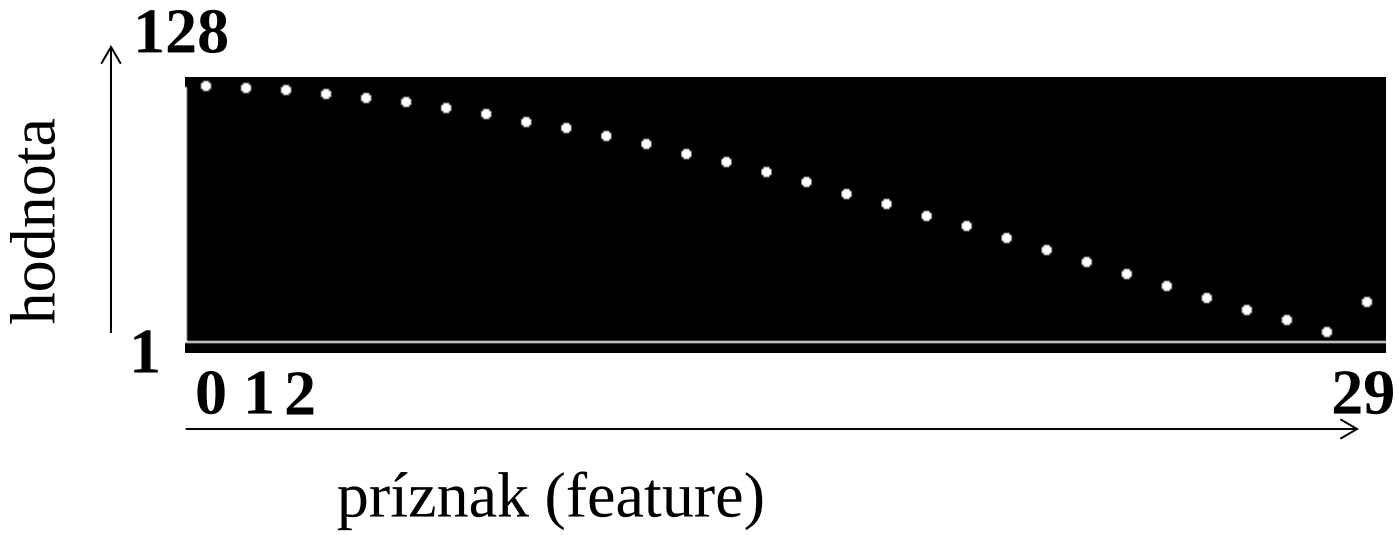
lucny@fmph.uniba.sk

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

`www.robotika.sk/cviko7-faces.zip`
`www.robotika.sk/cviko7.zip`

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



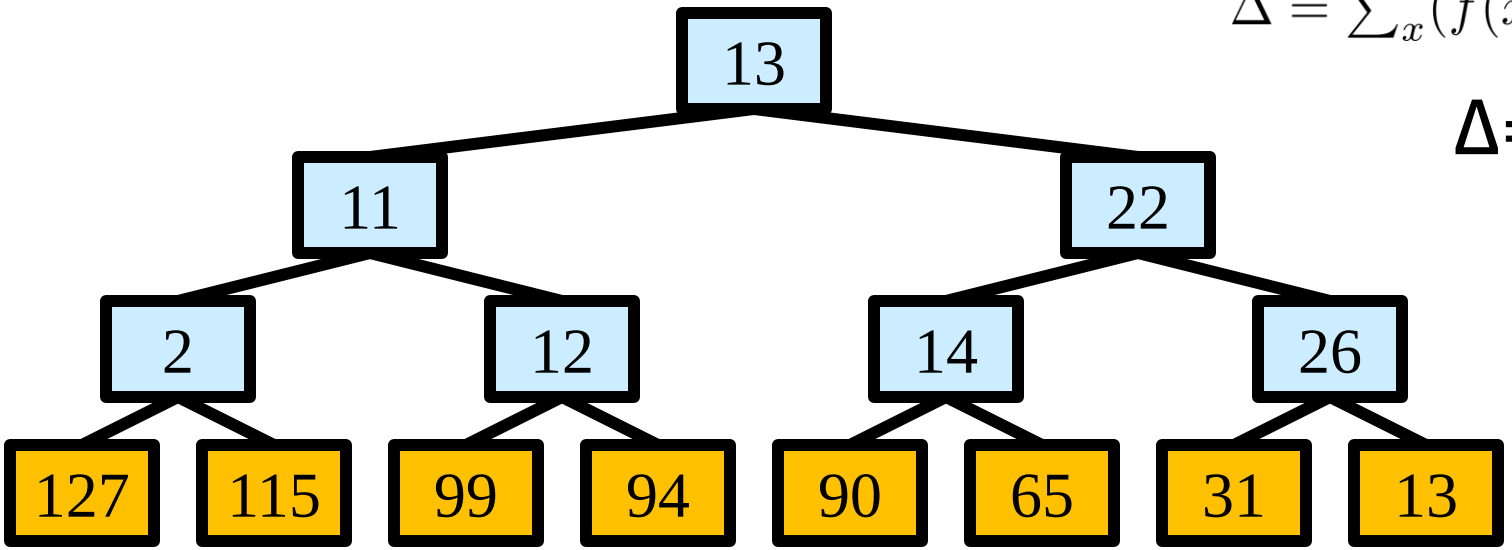
- Učíme stroj nejakú funkciu ku ktorej máme sadu vzoriek (príznak, hodnota)

Slabý regresor (Regresný strom)

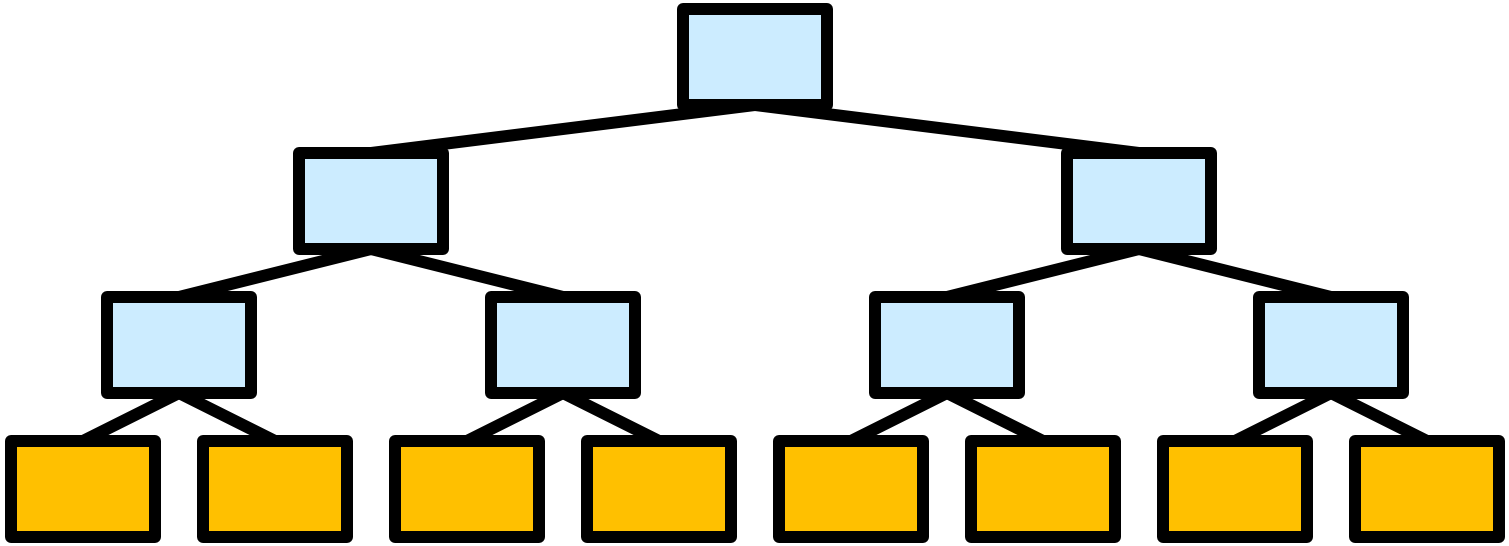
x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
f	128	127	126	124	122	120	117	114	110	107	103	99	94	90	85	80	74	69	63	58	52	46	40	34	28	22	16	11	5	20
g	127	127	115	115	115	115	115	115	115	115	115	99	94	90	65	65	65	65	65	65	65	65	31	31	31	31	13	13	13	13

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

Δ=2126



Ako získame regresný strom?

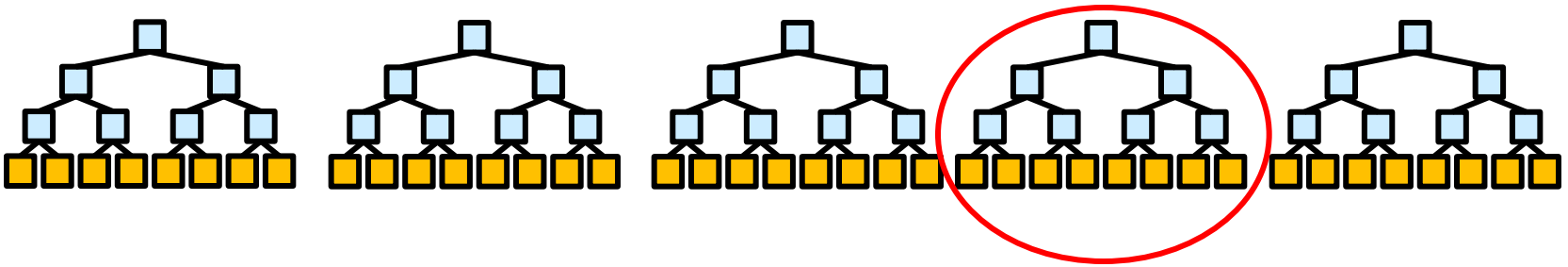


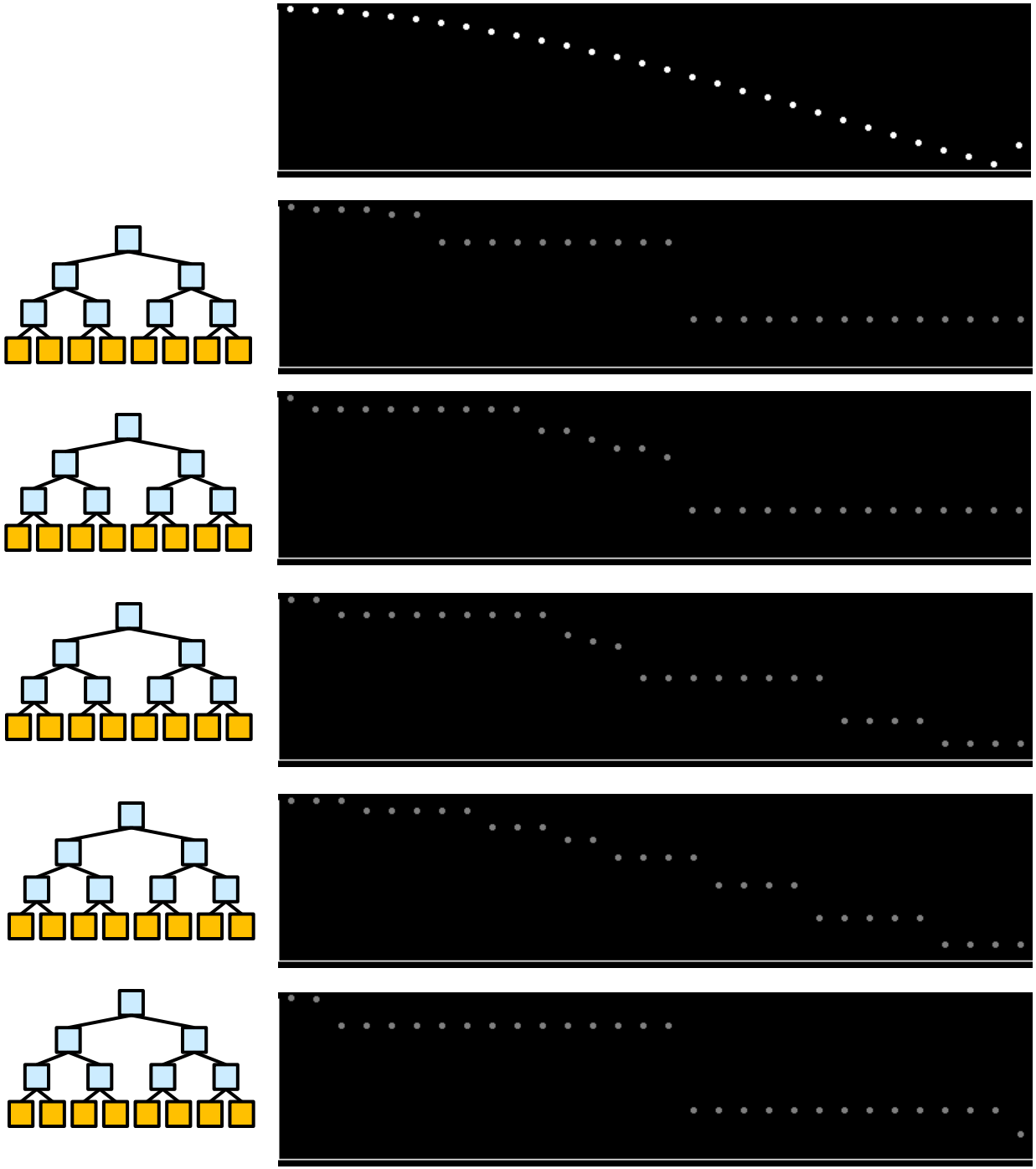
- Povieme si nech je hĺbky 3
- Vo vnútornom vrchole dáme porovnanie na náhodne zvolený (deliaci) príznak (menší vľavo, väčší vpravo)
- V liste dáme priemer z hodnôt, ktoré zodpovedajú príznakom s ktorými sa do listu dostaneme

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

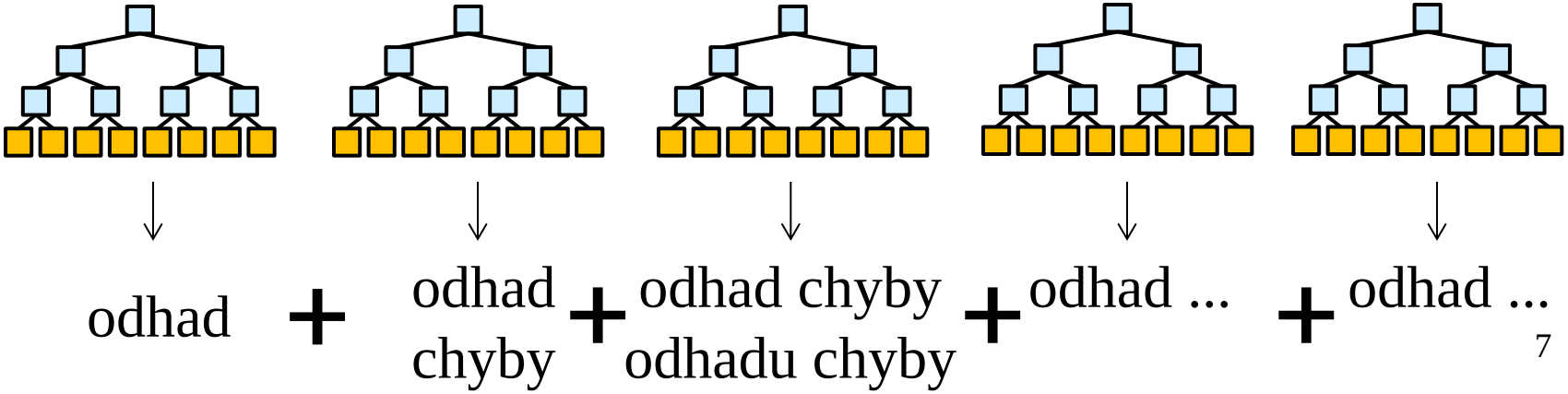
Δ=436

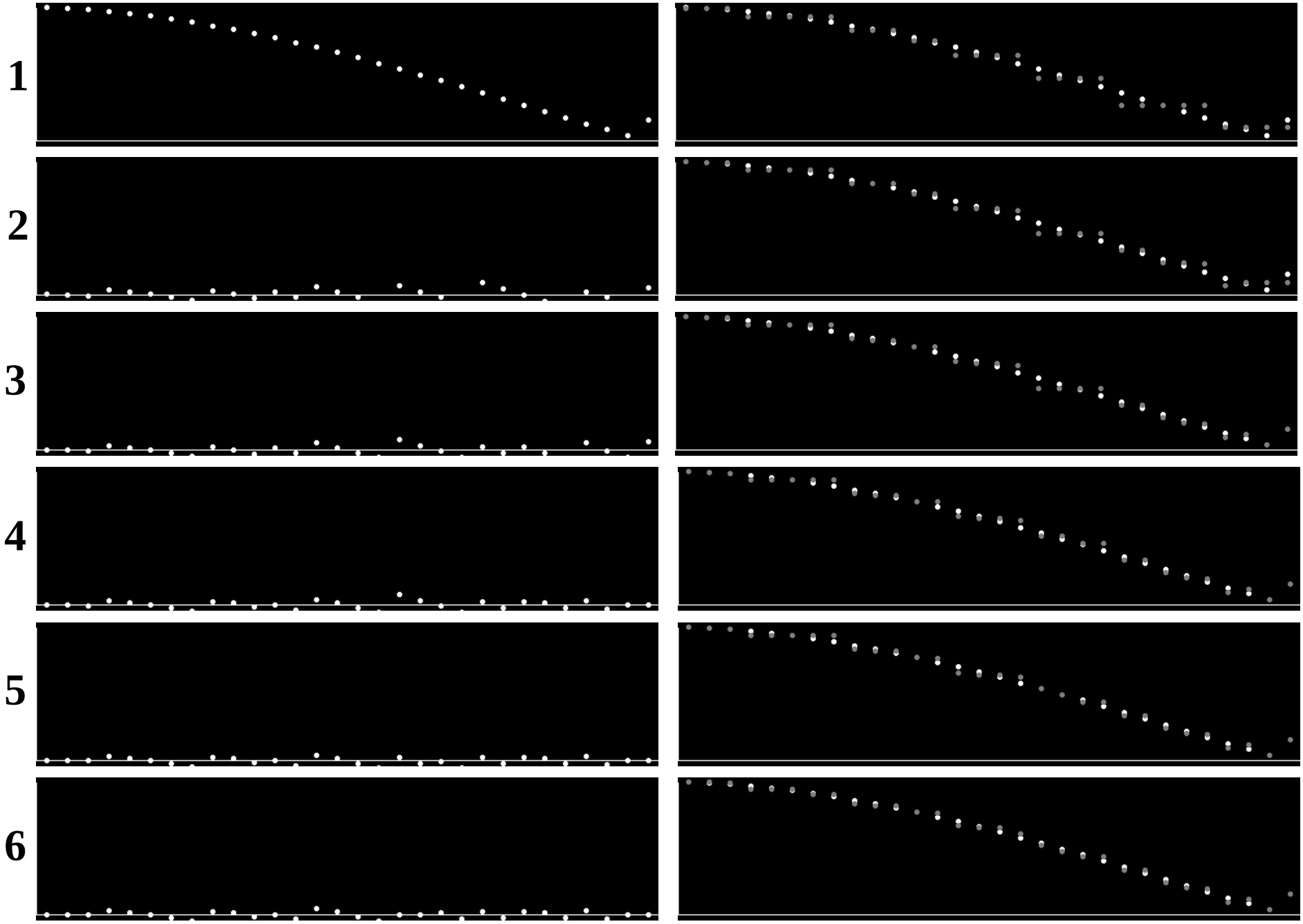
Δ=4000

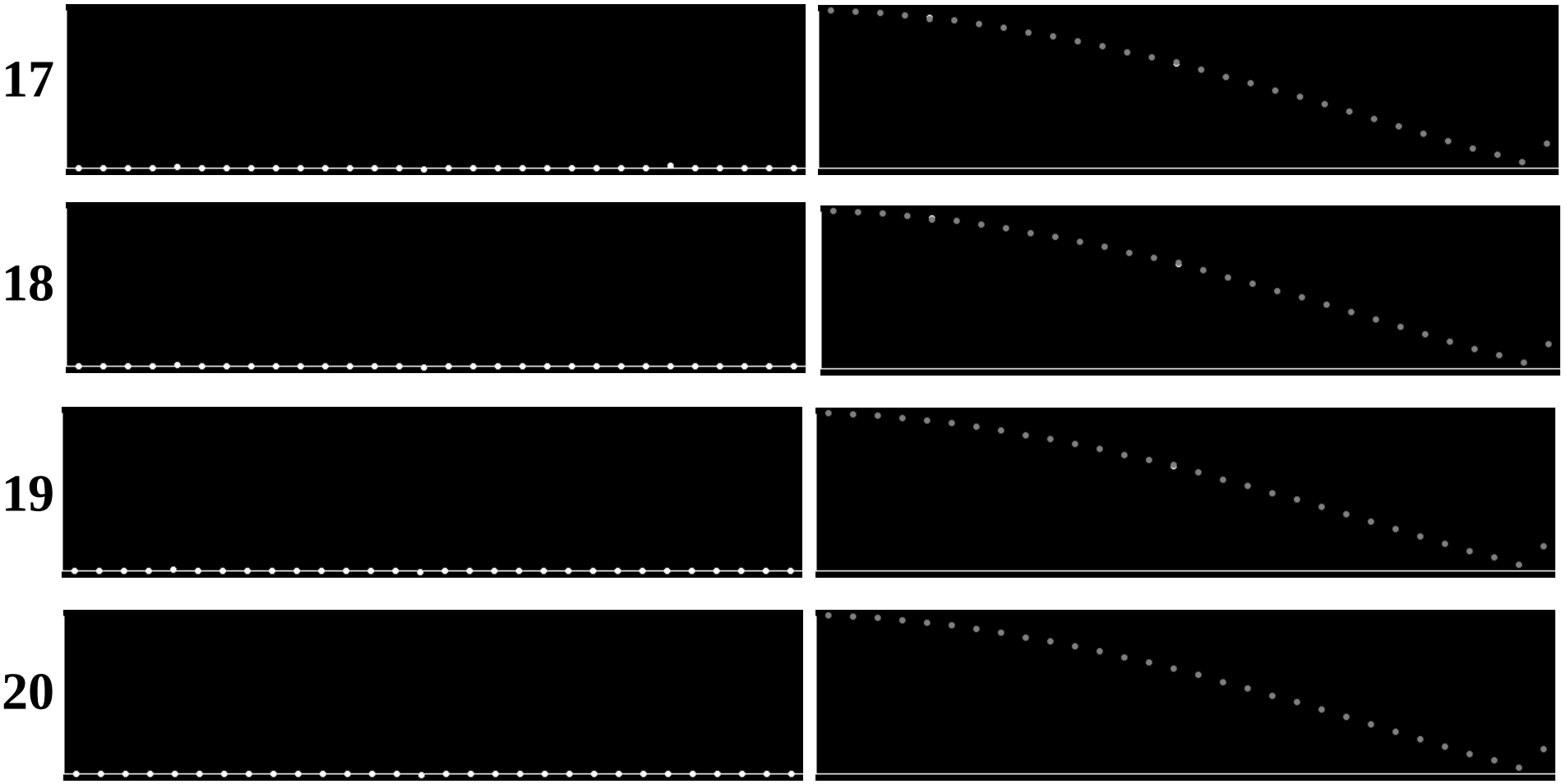
Gradient Boosting

- 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 odzviev na príznak



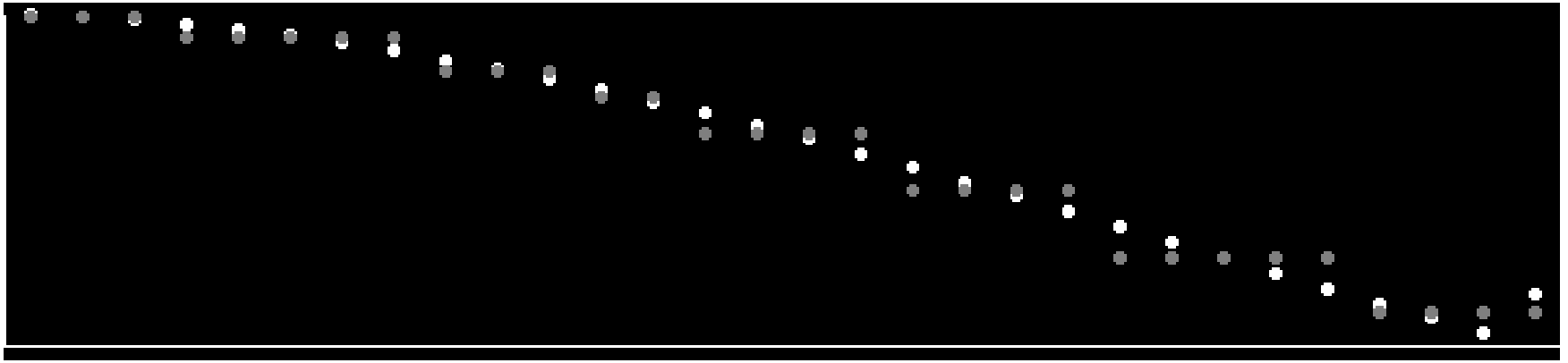

1
2
3
4
5
6



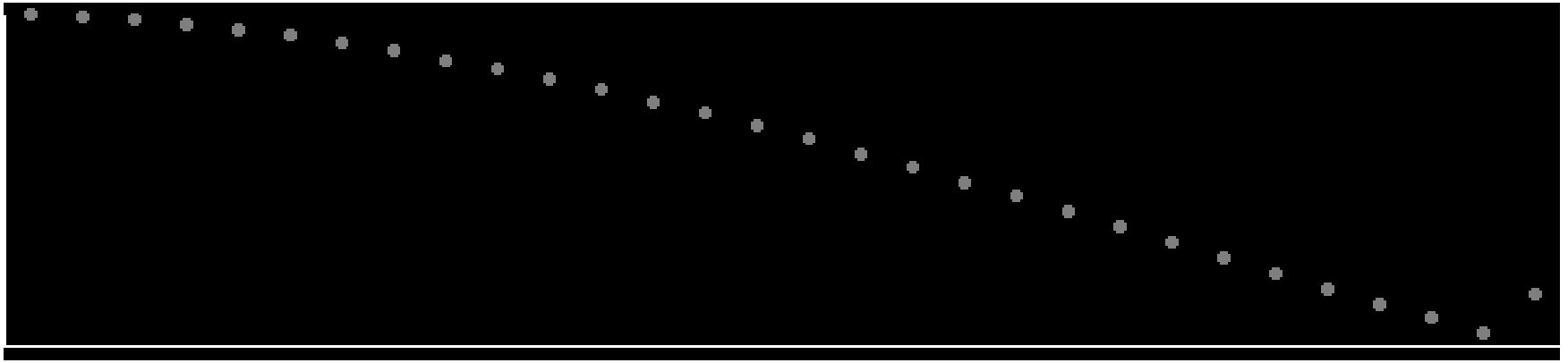
$\Delta=0$

Gradient Boosting

1

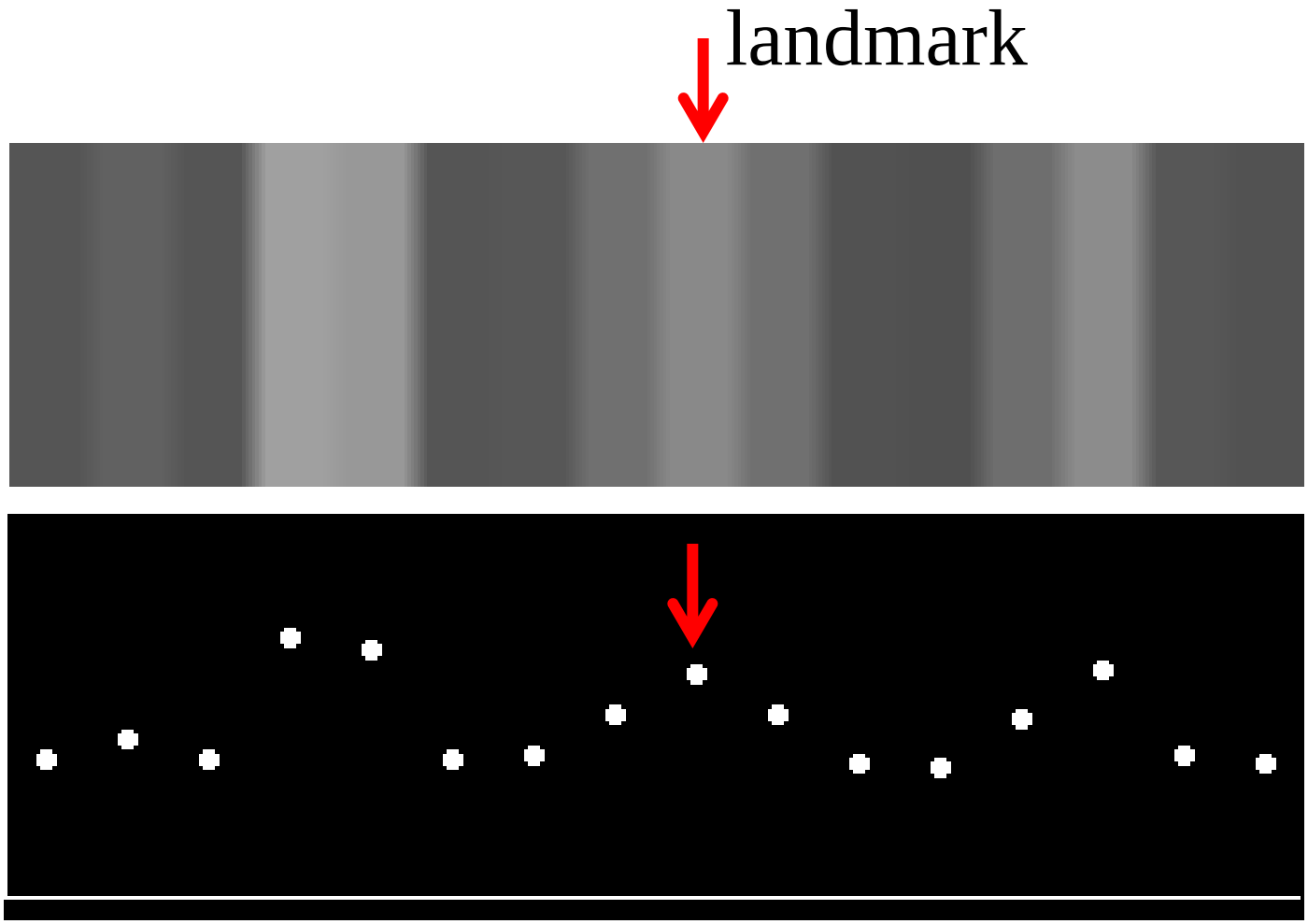


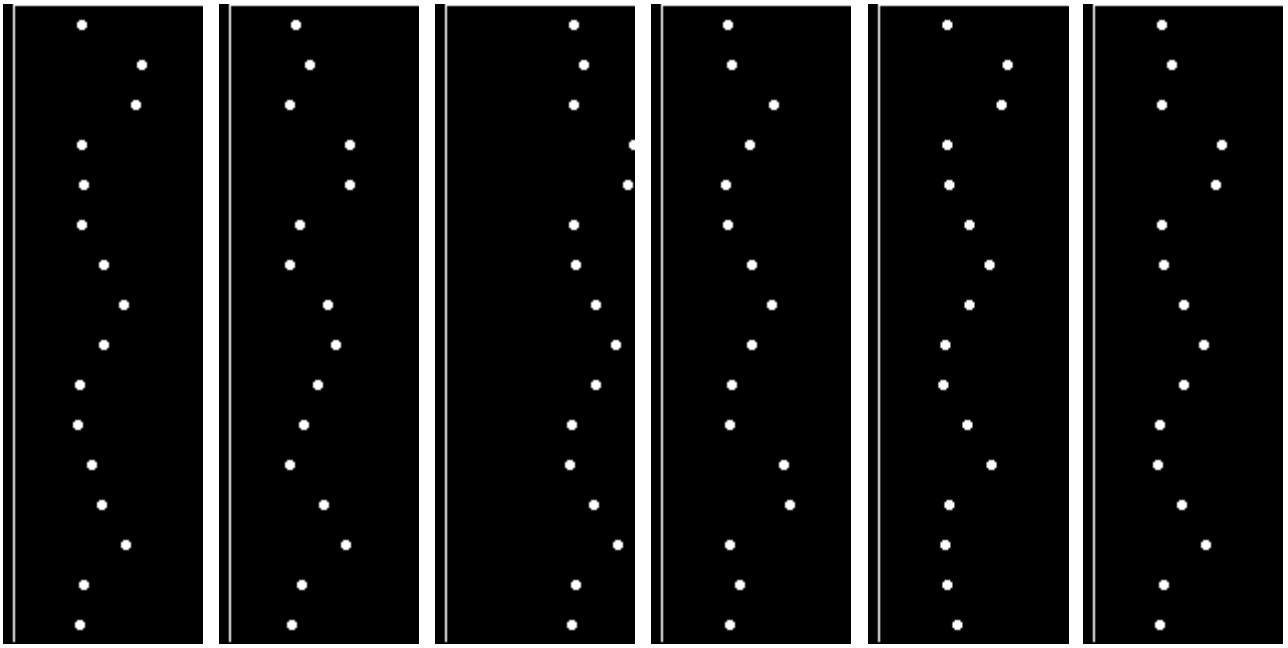
20

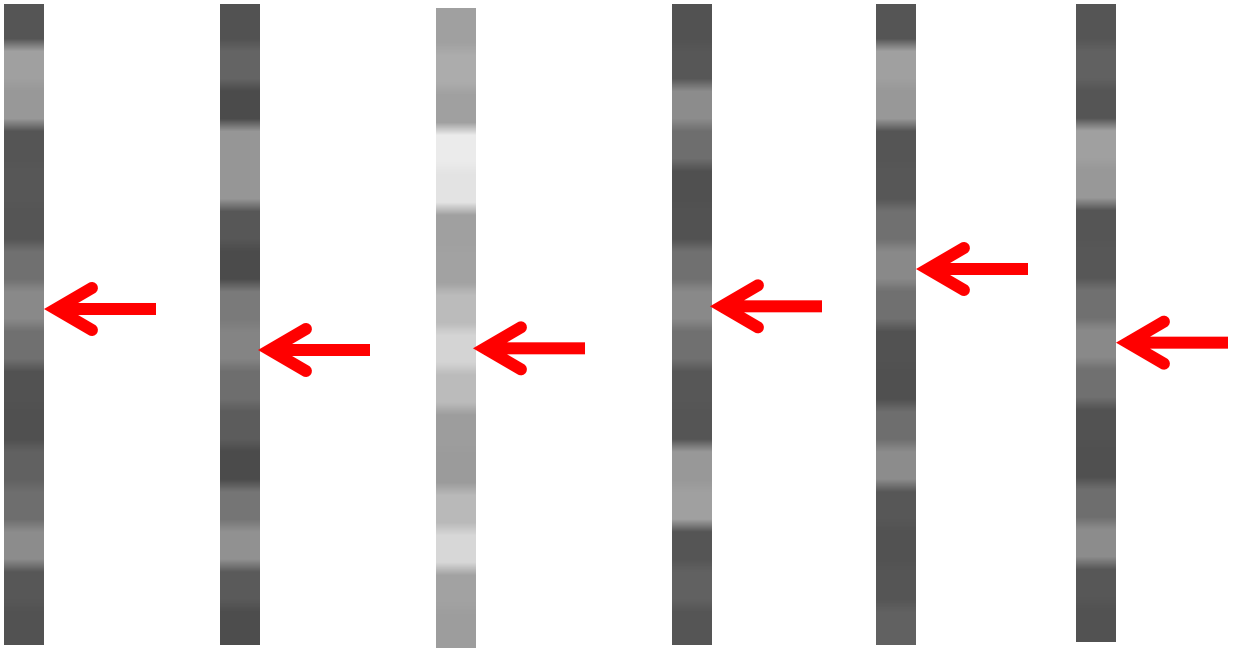


RegressionTree

Príznaky na obraze





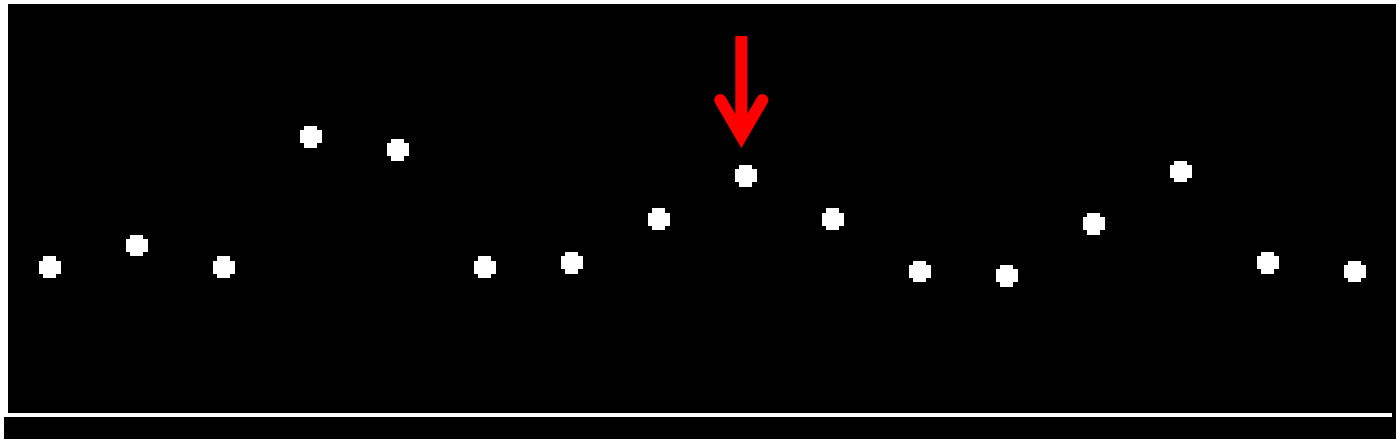


- Príznak by mal byť odolný voči zmene osvetlenia, posunutiu, preklopeniu, zašumeniu

Príznaky na obraze

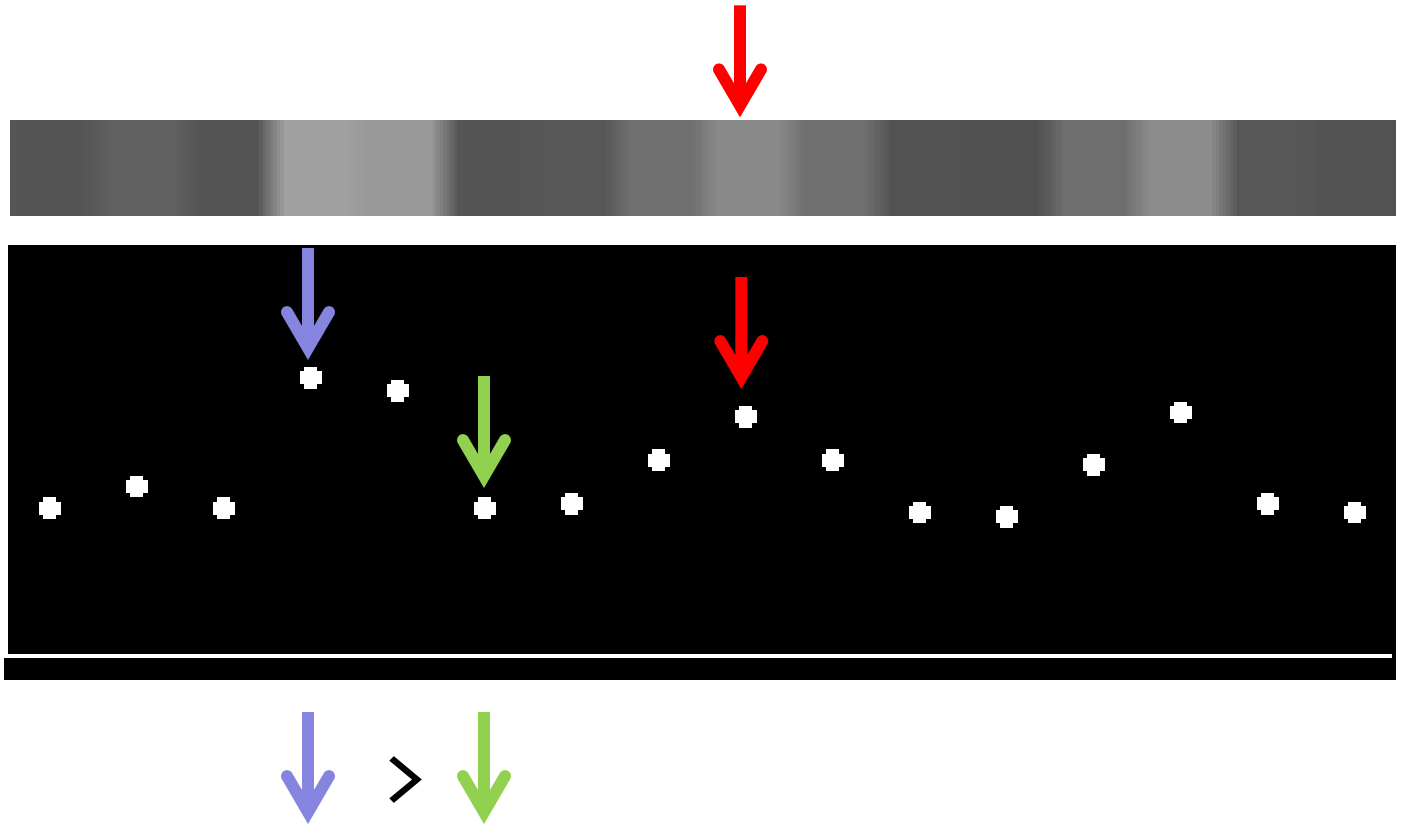
- Ako pozná táto bodka, že je tá, ktorú chceme?

1. Porovnáva svoju hodnotu s okolím.
2. Blízko nej sú hodnoty menšie
3. Ďalej od nej sú hodnoty väčšie



Príznaky na obraze

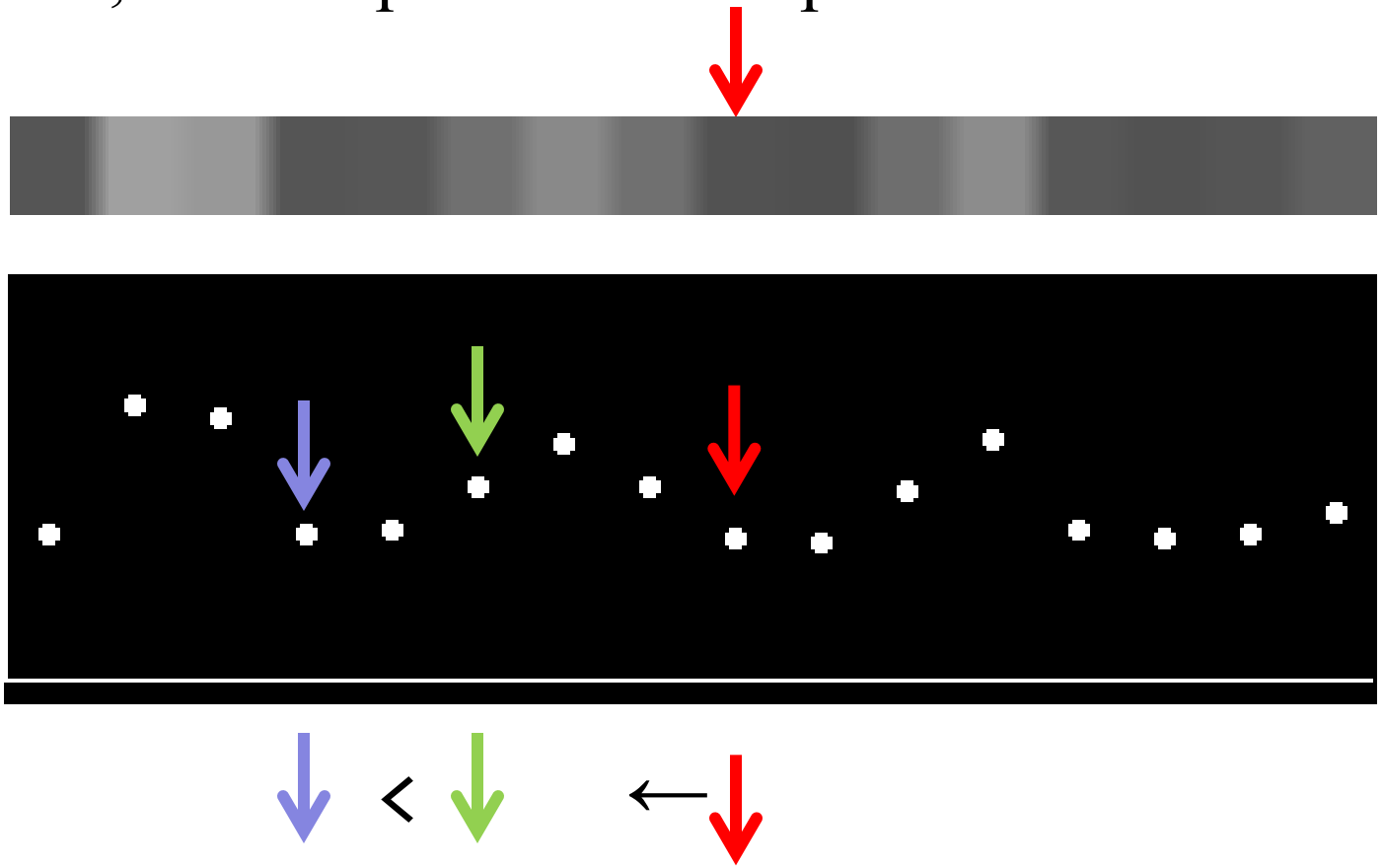
- Najjednoduchší vhodný príznak je teda porovnanie intenzít dvoch pixelov v okolí relatívnom ku pozícii z ktorej sa pýtame



Regresor landmarkov

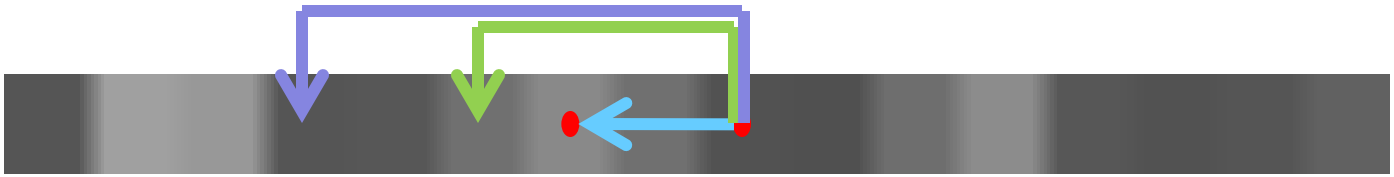
- Čo keď nie sme v správnom mieste, ale sme blízko?

- z toho, že porovanie pixelov nie je v poriadku, môže vyplynúť, že sa napríklad máme posunúť vľavo



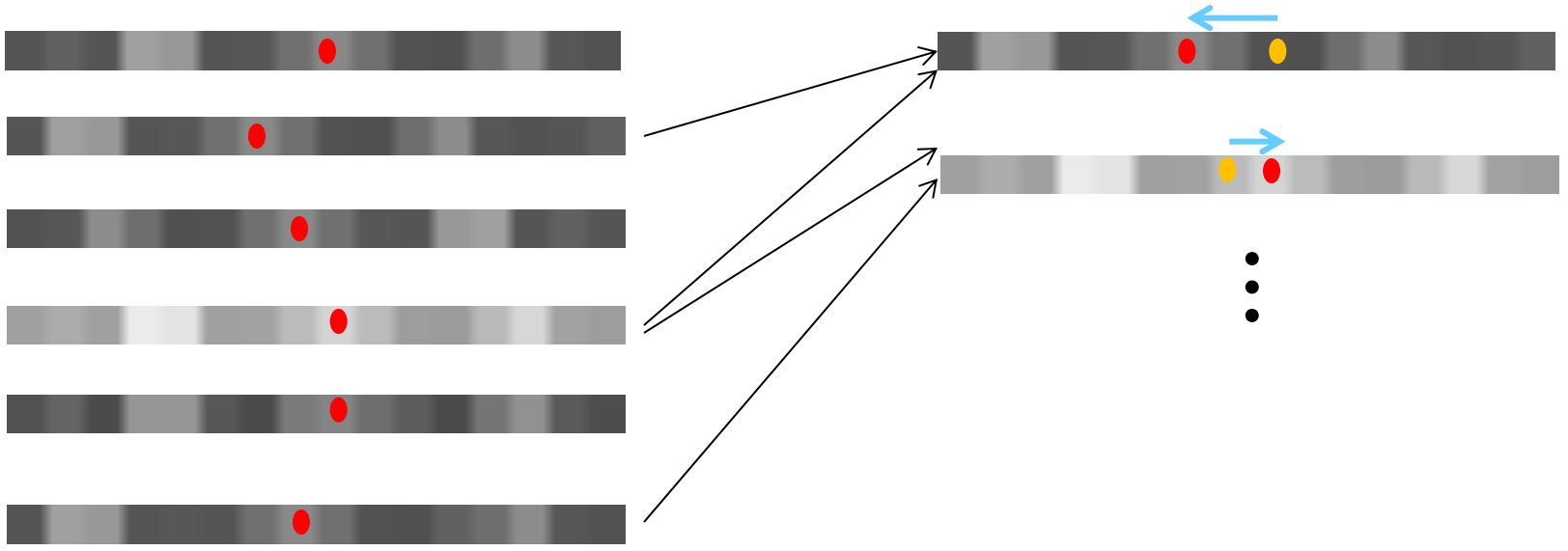
Regresor landmarkov

- Príznaky: porovnanie intenzít pixelov relatívne k miestu kde sme
- Vrátená hodnota: ako posunúť miesto



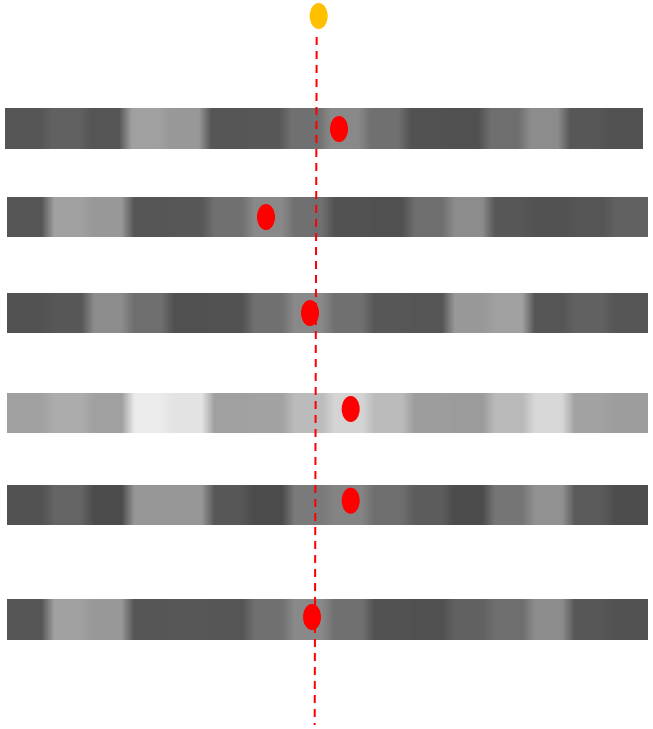
Kde na to zoberieme vzorky?

- Na vzorkách obrazu si označíme landmarky
- A ako vzorovú zlú pozíciu vezmeme landmark z inej vzorky



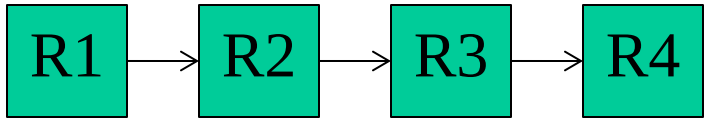
Kde začneme?

- Na obvyklej, t.j. priemernej pozícii



- Pokiaľ sa nám podarí naučiť regresor s nulovou chybou, tak na jedno jeho zavolanie sa presunieme na správne miesto.

Kaskádny regresor



- A čo keď sa to nepodarí?
- Tu už odpoveď poznáme: použijeme gradient boosting, akurát druhej úrovne
- Vzorkami učenia druhej kaskády budú výsledky posunutia priemernej polohy regresorom v prvej kaskáde
- Pritom je dobre vziať na posun nie celý výstup regresora, ale len 0.9 časť z neho, lepšie to konverguje

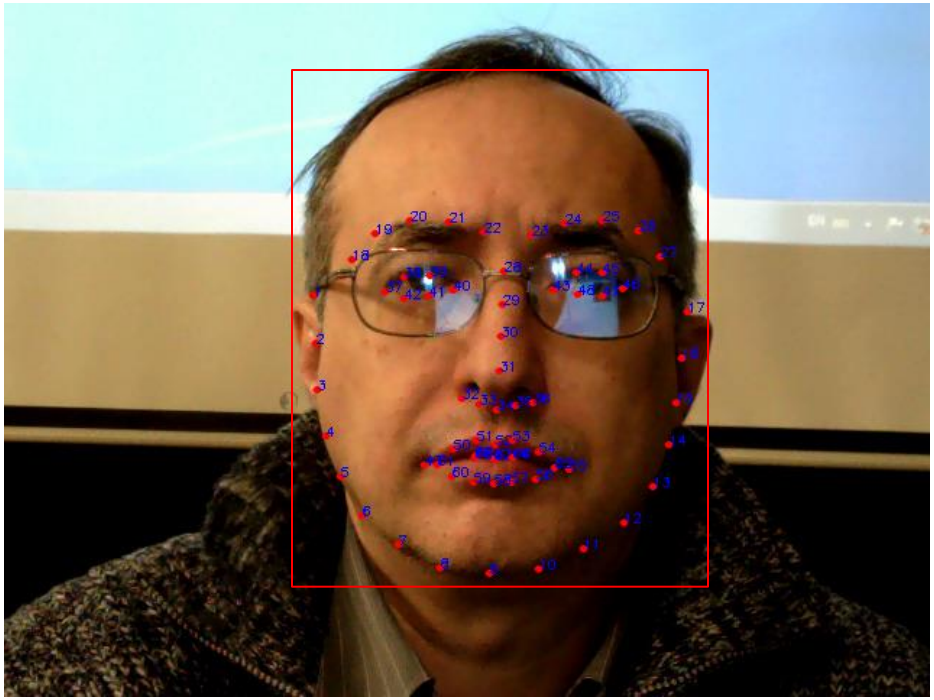
1D → 2D, 1 → 68

- Čo sa zmení, keď chceme hľadať landmarky v 2D?
- Čo sa zmení, keď nie jeden landmark, ale viacero?

1. Príznak ostáva (porovnanie dvoch pixelov), ale oveľa ťažie sa definuje čo to znamená, pozerať sa na ne zo svojho miesta (lebo landmarkov je veľa a brať ich po jednom nie je dobrý nápad, musia držať určitý tvar). Riešenie: similarity transform
2. Hodnota sa mení z vektora posunutia na viacero vektorov posunutia.
3. Zadefinovať inciálnu úvodnú polohu nie je problém

Príprava vzoriek

- 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.



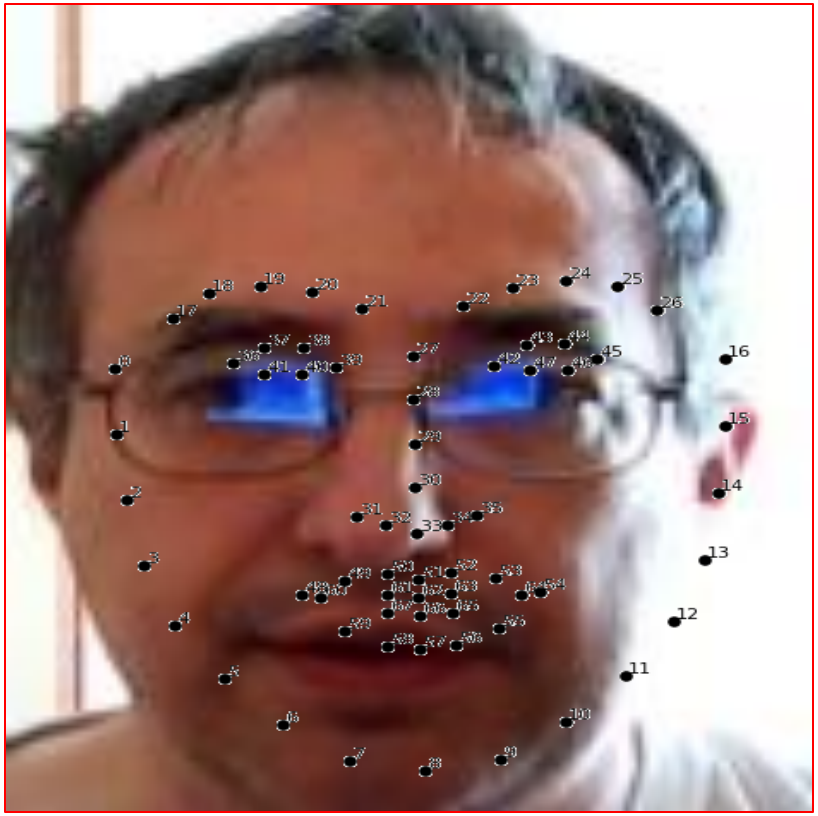
- Dlib používa 68 bodov
- Existujú voľne dostupné datasety, viď. `data/datasets`

Predspracovanie

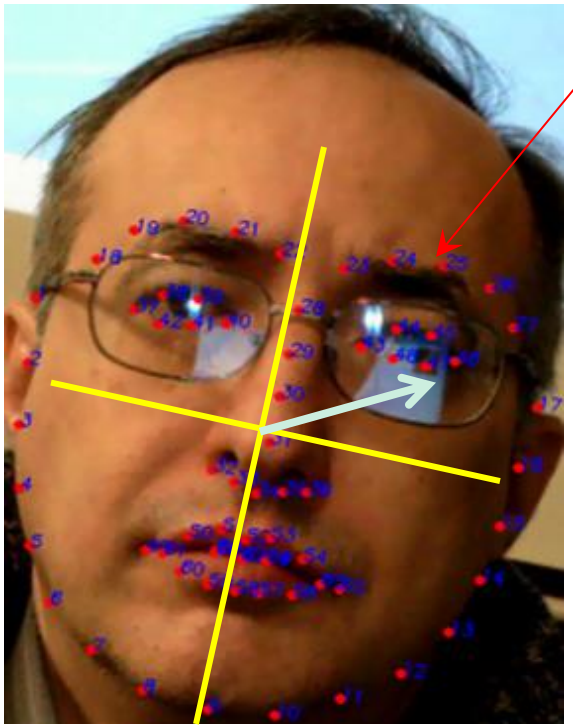
- Remapujeme na štandardnú veľkosť

- Riešime tým zväčšenie či zmenšenie spracúvanej tváre

Iniciálny stav – priemer vzoriek



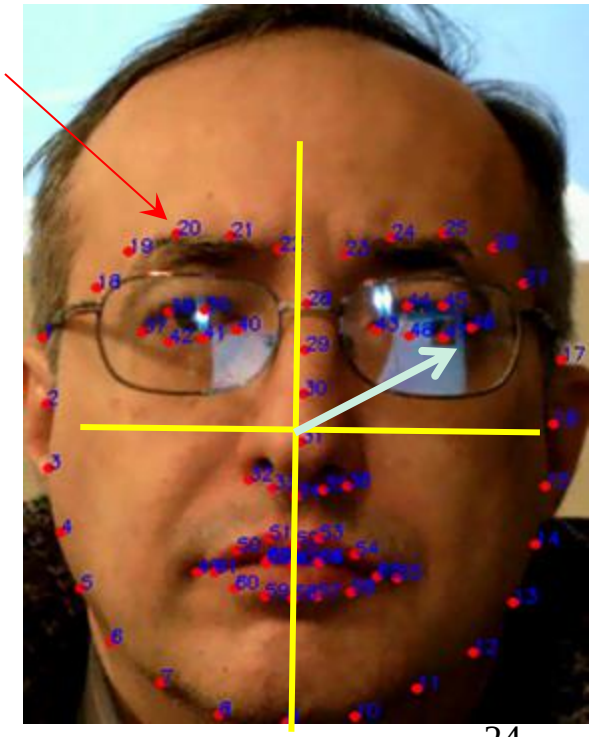
- Ako relativizovať súradnice pixelov použitých pri deliacich porovnávaniach voči aktuálnemu odhadu landmarkov?
- Tvár na spracúvanom obraze môže byť posunutá i mierne pootočená



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)

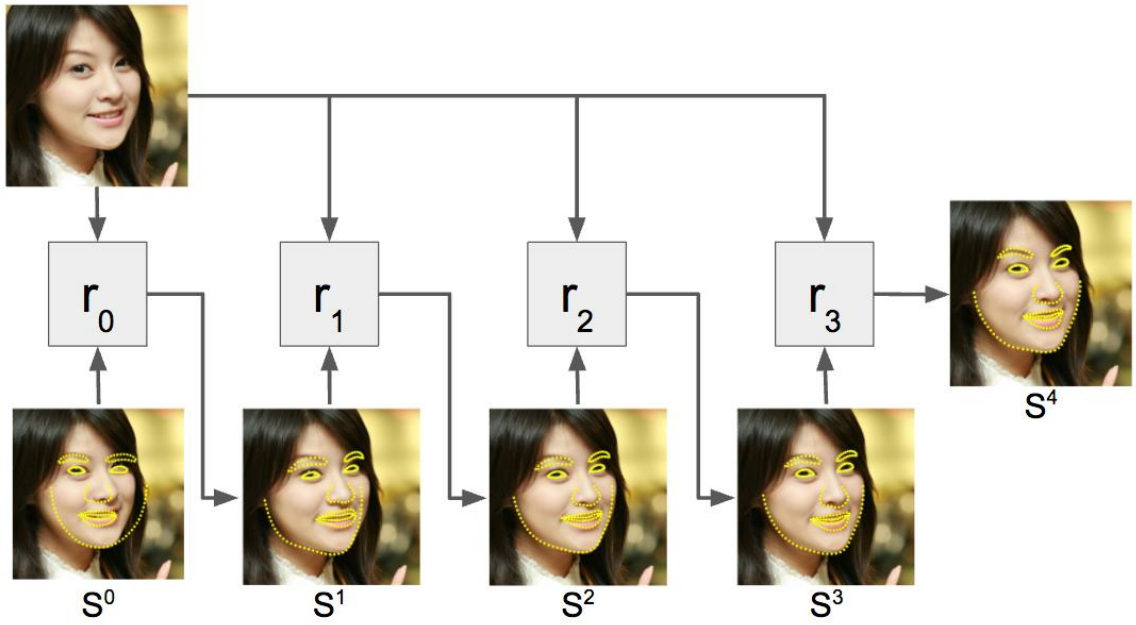
Features

- 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)

Trénovanie kaskádneho regresora

`TrainRegression`

- Po 5 minútach oznámi koľko to bude trvať, cca dokopy 40 minút na 8 jadier + CUDA
  - 10 kaskád
  - v každej regresor
  - v ňom 500 regresných stromov hĺbky 4
  - 400 pixelov vybraných zo 128x128
  - 20 pokusov pri voľbe deliaceho príznaku



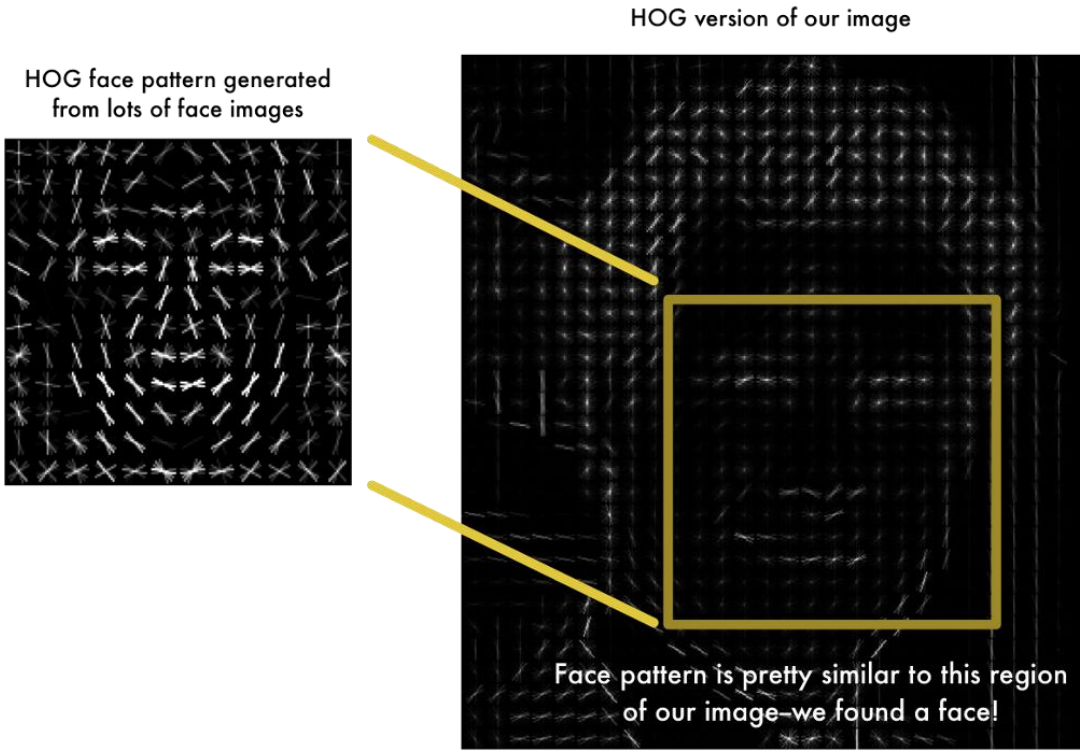
Použitie natrénovaného regresora

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

`FacialLandmarks`

Ale kde je tvár?

- Regresoru musíme povedať kde na obrázku je tvár
- A to musíme zistiť nejakou inou metódou
- Dlib používa na tento účel HOG detektor



Autori metódy

One Millisecond Face Alignment with an Ensemble of Regression Trees

Vahid Kazemi & Josephine Sullivan

KTH, Royal Institute of Technology
Computer Vision and Active Perception Lab
Teknikringen 14, Stockholm, Sweden

www.csc.kth.se/~vahidk/face/KazemiCVPR14.pdf

- Open source knižnica na strojové učenie a počítačové videnie
- Autor: Davis King
- Boost Software License
- Linux, Windows, Mac
- CUDA based
- Od roku 2002

Použitie

Pokiaľ vieme pozíciu facial landmarkov, dokážeme veľa vecí:

- Zvýšiť úspešnosť detekcie tvárí
- Sledovať ľudí, počítať ich
- Aplikovať na ich tvár rôzne filtre.

- Zistiť polohu hlavy, robot potom môže človeka sledovať, alebo napodobňovať

Veľa pekných zdrojákov sa dá získať za registráciu zdarma z článkov na www.learnopencv.com

Ďakujeme za pozornosť

Seminár Robotika.SK

Ako naučiť robota vidieť ľudskú tvár pomocou knižnice Dlib

Andrej Lúčny

Katedra aplikovanej informatiky FMFI UK

lucny@fmph.uniba.sk

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