SU

Strojno učenje

Kombiniranje strojno naučenih algoritama

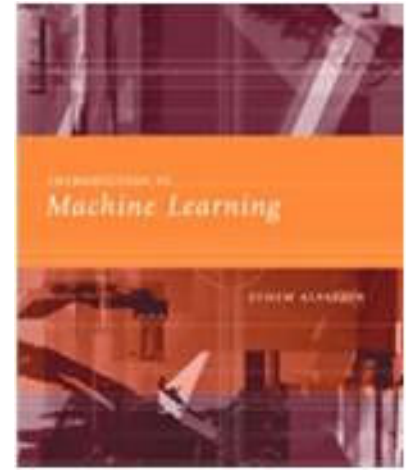
Siniša Šegvić

Zavod za elektroniku, mikroelektroniku,
računalne i inteligentne sustave
Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu

Plan

Kombiniranje strojno naučenih algoritama

1. **Pregled značajnijih pristupa**:
   - nezavisno naučeni eksperti (varijante glasanja)
   - višerazinsko učenje (npr: boostanje, kaskadiranje)
   - [Alpaydin04], poglavlje 15



2. Postupak Viole i Jonesa:
   - o detekciji kaskadom ojačanih (boostanih) Haarovih klasifikatora
   - P. Viola and M. Jones, Robust Real-Time Face Detection, IJCV04.

Uvod

Kod većine aplikacija strojnog učenja **više** pristupa vodi do rješenja

Ideja: kombiniranjem naučenih osnovnih algoritama postići **sinergiju**.

Potrebno smisliti sljedeće:

1. kako doći do komplementarne vojske osnovnih algoritama?
   - ovisi o domeni, ali neke opće upute na sljedećoj stranici
2. kako ih kombinirati?
   - dosta se može postići **linearnom kombinacijom**!

Uvod: varijacija

Elementi varijacije pri oblikovanju osnovnih algoritama:

- različite grupe algoritama (npr, parametarski + neparametarski)
- različiti parametri istog algoritma
  (k kod kNN, broj skrivenih neurona kod MLP, ...)
- fokus na različite modalitete ulaza
  (prst+dlan+šarenica u biometriji, različito pretprocesiranje)
  - srodno fuziji senzora u području obrade signala
- različiti odabiri skupova za učenje
  (slučajno, ili prema nedostacima, strukturi, potproblemima, ...)

Fokus osnovnih algoritama na različitim aspektima problema!

- pretp. algoritam A_i ima uspješnost od 80%
- za A_{i+1} ključno kako se ponaša na ostalih 20%!

Uvod: kombiniranje

Dva su temeljna koncepta za kombiniranje osnovnih algoritama:

- nezavisni eksperti (multiexpert combination):
  algoritmi međusobno **potpuno odvojeni**, kombinacijski modul se brine za fuziju rezultata
  - predstavnici: glasanje (s varijantama)
- višerazinska kombinacija (multistage combination):
  - višerazinsko **učenje**: sljedbenici uče na greškama prethodnika
    - predstavnici: jačanje (boosting), kaskadiranje (cascading)
  - višerazinska **primjena**: sljedbenici se primjenjuju kad prethodnici ne donesu odluku
    - predstavnik: kaskadiranje

UVOD: NOTACIJA

L ... broj osnovnih algoritama

K ... broj razreda kod klasifikacije

Ako algoritmi imaju po jedan izlaz:

- $d_j(\mathbf{x})$ ... rezultat j-tog algoritma M_j za ulaz $\mathbf{x}$
- $y = f(d_1, d_2, ..., d_L|\Phi)$ ... konačni rezultat uz parametre Φ

Ako algoritmi imaju po K izlaza (klasifikacija):

- $d_{ji}(\mathbf{x})$ ... i-ti rezultat j-tog algoritma za ulaz $\mathbf{x}$
- $y_i = f(d_{1i}, d_{2i}, ..., d_{Li}|\Phi_i)$ ... konačni i-ti rezultat uz parametre Φ_i
- npr, u klasifikaciji, možemo odabrati razred $i = \arg\max_i(y_i)$

GLASANJE: UVOD

Kod glasanja **nezavisni** eksperti **usporedno** evaluiraju ulaz.

Konačni rezultat linearna kombinacija pojedinačnih "osnovnih" rezultata.

Fuzija linearnom kombinacijom:

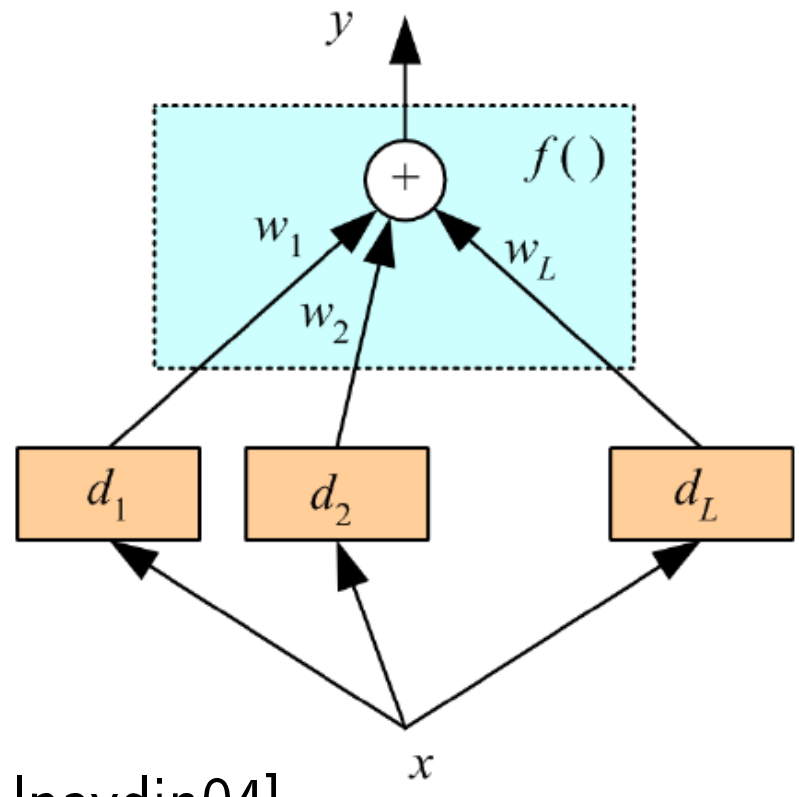
$$y = \sum_{j=1}^{L} w_j d_j, \;\; w_j \geq 0 \;\; \forall j, \;\; \sum_{j=1}^{L} w_j = 1$$

Klasifikacija:

- $y_i = \sum_{j=1}^{L} w_j d_{ji}, \; i \in [1, K]$
- odluka: $i = \arg\max_i (y_i)$

Jednostavno glasanje: $w_j = \frac{1}{L}$

Većinsko glasanje: $w_j = \frac{1}{L}, K = 2$



[Alpaydin04]

GLASANJE: PRIMJER

Binarna klasifikacija, n nezavisnih glasača, pogađaju s vjerojatnošću p.

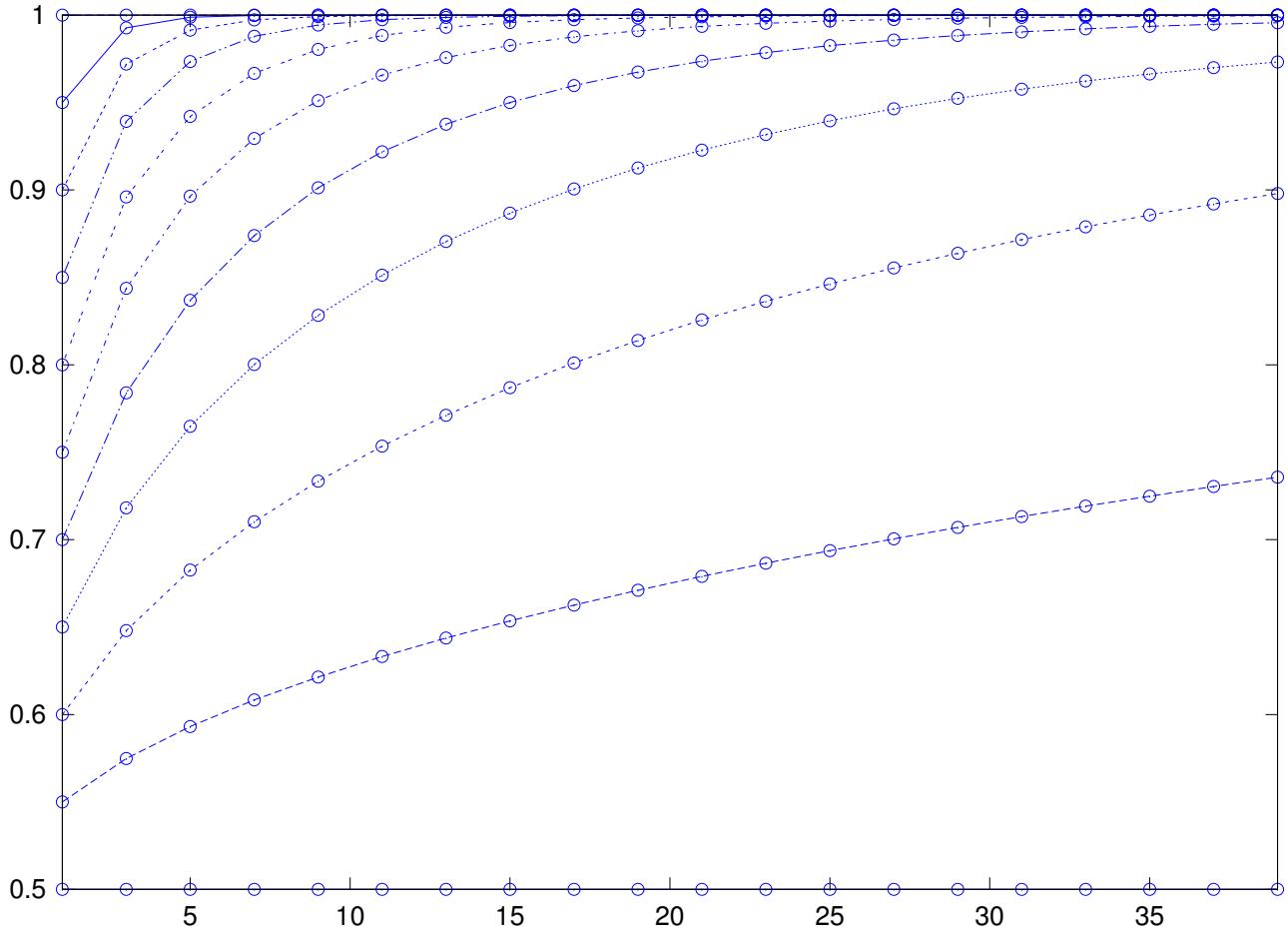
Kolika je vjerojatnost ispravne klasifikacije?

- X ... slučajna varijabla, broj glasača koji su glasali ispravno
- $X \sim \mathcal{B}(n, p)$, $P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$
- $P(X >= \lceil n/2 \rceil) = \sum_{i=\lceil n/2 \rceil}^{n} \binom{n}{i} p^i (1-p)^{n-i}$

Rezultat ovisi o n i p, kako izgledaju krivulje?

GLASANJE: VEĆINSKO, GRAF

Ako je p samo manji od $0{,}5$, vjerojatnost uspjeha monotono raste s n



Ima nade za demokraciju :-)

GLASANJE: VARIJANCA

Sad ćemo evaluirati efekte glasanja analizom varijance rezultata y

Pretpostavimo:

- d_j nezavisni, imaju istu distribuciju
- $w_j = 1/L$
- $y = \frac{1}{L} \sum_j d_j$

Lako se vidi da rezultat ima isto očekivanje te manju varijancu:

- $E[y] = \frac{1}{L} \sum_j E[d_j] = E[d_j]$
- $\mathrm{Var}[y] = \frac{1}{L^2} \sum_j \mathrm{Var}[d_j] = \frac{1}{L^2} \cdot L \cdot \mathrm{Var}[d_j] = \frac{1}{L} \mathrm{Var}[d_j]$

Glasanje: Bayes

Pretp. da rezultat j-tog algoritma odgovara posteriornoj vjerojatnosti ispravne klasifikacije:

$$d_{ji} = P(C_i|\mathbf{x}, M_j)$$

Tada glasanje možemo predstaviti Bayesovom kombinacijom modela!

$$P(C_i|\mathbf{x}) = \sum_j P(C_i|\mathbf{x}, M_j)P(M_j)$$

Ideja: odabrati skup "smislenih" modela za koje je $P(M_j)$ relativno visok

Apriorna vjerojatnost $P(M_j) = w_j$ može favorizirati jednostavne modele

GLASANJE: PRIMJERI

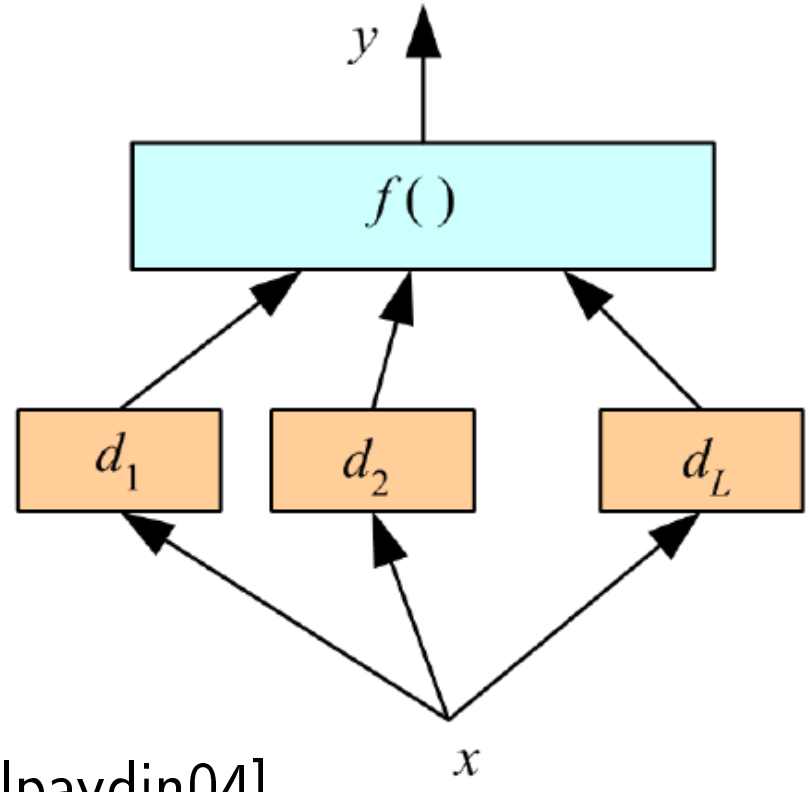
Sad ćemo ukratko spomenuti tri konkretna primjera primjene koncepta usporedne evaluacije nezavisnim ekspertima (glasanja):

- Stacking:
  - parametre kombinacijske funkcije strojno naučiti zasebnim algoritmom
- Error correcting output codes (izlazni kôd s ispravljanjem pogreške):
  - dobiti optimalni višerazredni klasifikator iz skupa binarnih klasifikatora
- Bagging
  - osnovne algoritme konstruirati variranjem skupa za učenje

GLASANJE: STACKING

Generalizacija glasanja:

- parametre kombinacije Φ **učimo** odvojenim postupkom:
  $y = f(d_1, d_2, ..., d_j|\Phi)$
- f ne mora biti linearna!
- učenje Φ potrebno provesti na zasebnom skupu za učenje (npr, k-struka cross-validacija)



[Alpaydin04]

Npr, predviđanje sekundarne proteinske strukture [Zhang92]:

- osnovni klasifikatori: parametarski postupak + kNN + MLP
- kombinacijski postupak: MLP

GLASANJE: ECOC

Error correcting output codes (izlazni kôd s ispravljanjem pogreške):

- ideja: **binarnu** klasifikaciju primijeniti na slučaj s **više** razreda
- generalizacija rješenja s K klasifikatora "jedan protiv svih" (JPS)
- strukturu rješenja prikazujemo matricom $\mathbf{W}_{K \times L}$

Jedan JPS po klasi 4×4:

$$\mathbf{W} = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}$$

$$L = K$$

Svi parovi "jedan na jednog" 4×6:

$$\mathbf{W} = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ \mathbf{-1} & \mathbf{0} & \mathbf{0} & \mathbf{+1} & \mathbf{+1} & \mathbf{0} \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

$$L = \frac{K \cdot (K-1)}{2}$$

Drugi **redak** desno: razred 2 korišten u algoritmima 1 (-), 4 (+) i 5 (+)

Drugi stupac desno: algoritam 2 raspoznaje razred 1 od razreda 3

GLASANJE: ECOC, SHEMA

ECOC se može promatrati kao glasačka shema:

- binarni klasifikatori određuju osnovne rezultate $d_j(\mathbf{x})$
- elementi $\mathbf{W}$ odgovaraju težinama osnovnih algoritama w_{ij}

$$y_i = \sum_{j=1}^{L} w_{ij} d_j$$

Ovakva formulacija omogućava i rad s osnovnim klasifikatorima koji na izlazu ne daju binarne rezultate!

Razlike u odnosu na općenitu glasačku shemu:

- težine ovise o razredu (w_{ij} vs w_j)
- $w_{ij} \in \{-1, 0, 1\}$

GLASANJE: ECOC, HAMMING

Broj svih mogućih binarnih algoritama jako velik:

- binarni algoritam definiran ternarnom sekvencom od K znamenki
- npr za K=4: (1, -1, 0, 1), (-1, -1, 0, 1), (0,-1, 0, 1), ...
- broj svih algoritama ipak manji od 3^K jer nas ne interesiraju:
  - komplementi (1,0,-1,0) = (-1,0,1,0)
  - kombinacije sa samo jednim polom: (1,0,1,0)
- zato, za dani K, odabiremo **podesan** L u skladu sa zahtjevima

Kad imamo K i L, biramo $\mathbf{W}$:

- "bliski" redci impliciraju osjetljivost konačnog rezultata na pojedinačne klasifikacijske greške
- "udaljeni" stupci bolje diskriminiraju pojedinačne rezultate d_j
- $\Rightarrow$ retci i stupci $\mathbf{W}$ trebaju imati što veću Hammingovu udaljenost!

GLASANJE: ECOC, ZAKLJUČAK

Varijanta glasanja prikladna za binarne osnovne klasifikatore

Ideja: povećati toleranciju na greške osnovnih algoritama redundancijom

Detalji:

- posebno ima smisla koristiti kad su osnovni algoritmi **slabo korelirani** (ne griješe zajedno)
- nije jasno kako pri izgradnji $\mathbf{W}$ koristiti znanje o strukturi problema (potproblemi iz $\mathbf{W}$ dobiveni optimizacijom mogu biti teži od JPS)
- nije jasno da li i kako puniti 0 u $\mathbf{W}$

GLASANJE: BAGGING, DEFINICIJA

Bagging: varijanta **glasanja** gdje osnovne algoritme dobivamo variranjem **skupa za učenje**

Neka je dan skup za učenje $\mathbf{X}$, $N = |\mathbf{X}|$:

- generirati L skupova za učenje $\mathbf{X}_j$ uzorkovanjem iz $\mathbf{X}$ s ponavljanjem
  - $|\mathbf{X}_j| = N$, osim za velike N kad je bolje uzeti $|\mathbf{X}_j| < N$
- naučiti L osnovnih algoritama $\mathbf{M}_j$ istim postupkom učenja
- kombinirati pojedinačne rezultate $d_j = \mathbf{M}_j(\mathbf{x})$ uz $w_j = 1/L$

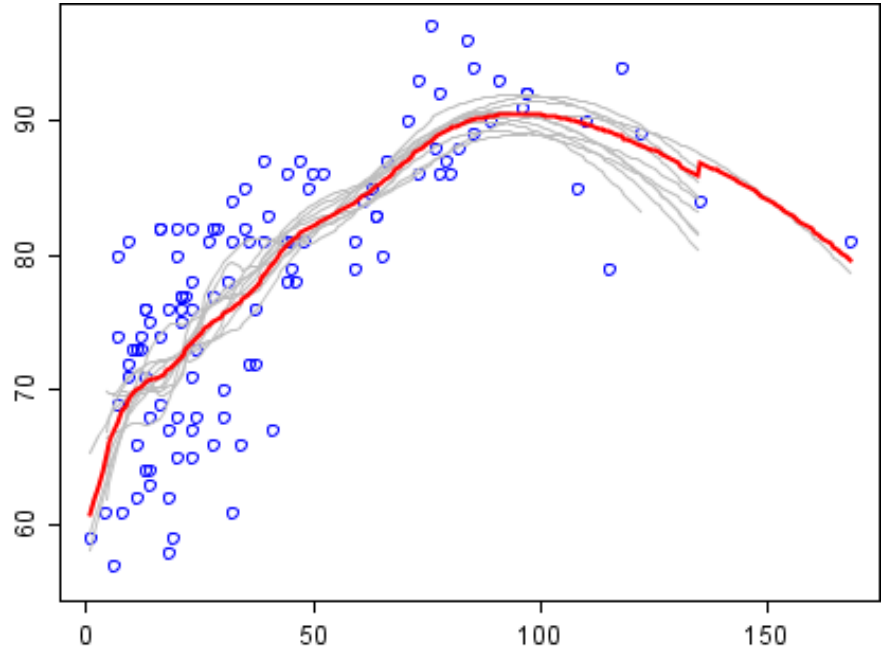
GLASANJE: BAGGING, SVOJSTVA

Pokazuje se da se ovakvim postupkom postiže:

1. **manja nestabilnost** nego kod originalnog algoritma
   - nestabilnost: male promjene ulaza $\Rightarrow$ velike promjene na izlazu
   - višerazinski perceptroni i stabla odluke nestabilni
2. **manja varijanca**
3. **otpornost na zasićenje**

Na slici desno [Wikipedia]:

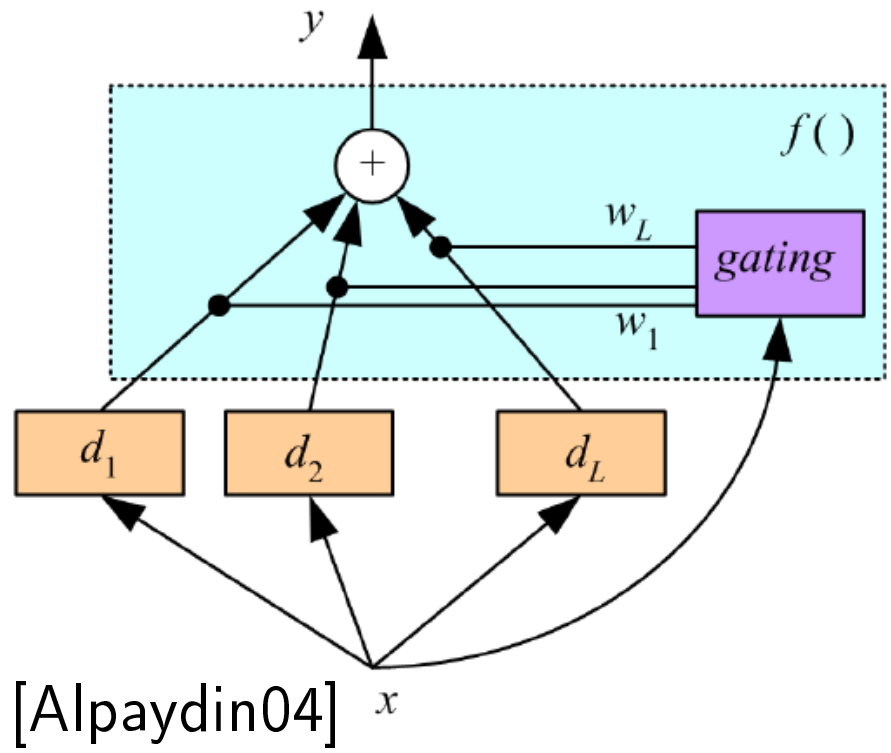
- sivo: 10 pojedinačnih rezultata
- crveno: regresija baggingom (100)
- crvena linija je glađa



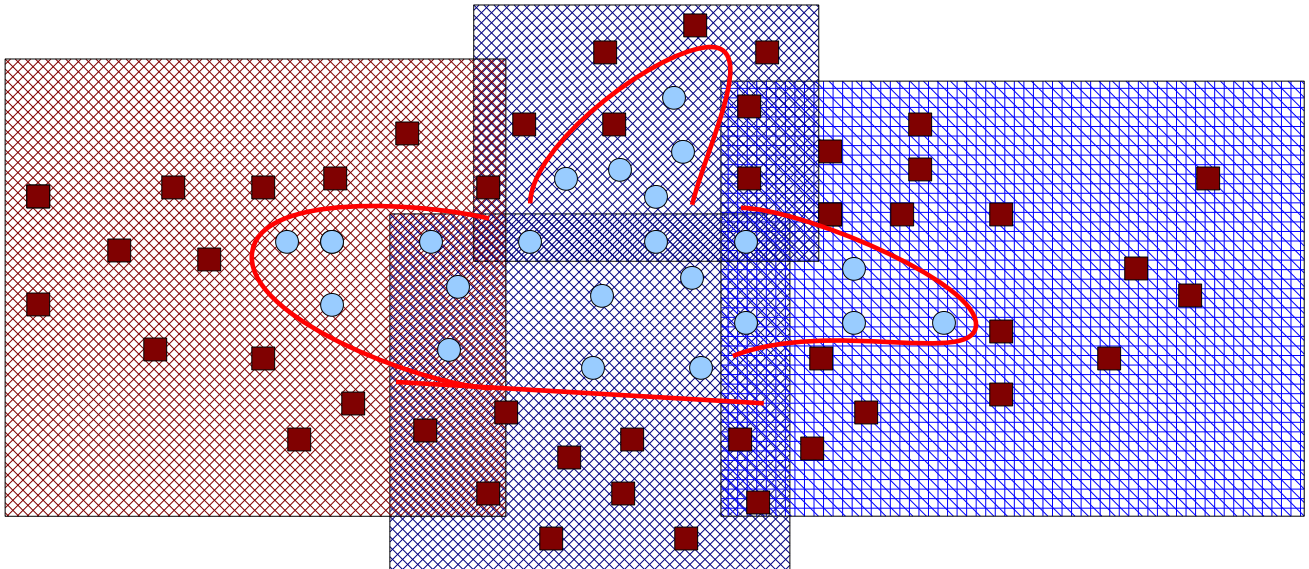
GLASANJE: MOE

Mixture of experts: nezavisni eksperti specijaliziraju područje ekspertize:

- osnovni algoritmi $\mathbf{M}_j$ dobivaju se kompetitivnim učenjem
- **rezultat**: $\mathbf{M}_j$ eksperti za pojedine regije ukupne domene
- težina eksperata ovisi o $\mathbf{x}$:
  $y = \sum_{j=1}^{L} w_j(x) d_j$



[Alpaydin04]



Boostanje: uvod

Kao kod bagginga, i kod boostanja osnovne algoritme konstruiramo variranjem **skupa za učenje**

Međutim kod bagginga je **varijacija** prepuštena slučaju i nestabilnosti osnovnog postupka učenja

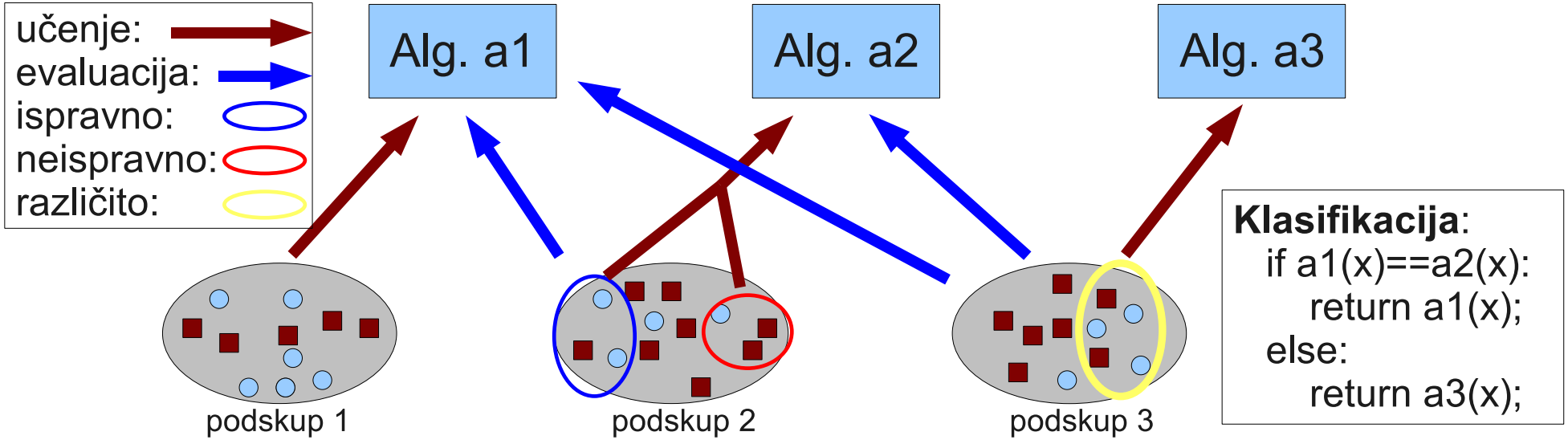
Kod boostanja skup osnovnih algoritama konstruiramo tako da sljedbenici **uče** na greškama prethodnika

Kao posljedica toga, boostanje može proizvesti jaki algoritam kombiniranjem prikladnog skupa slabih osnovnih algoritama

- slabi algoritam: vjerojatnost uspjeha samo malo veća od 50%
- jaki algoritam: vjerojatnost uspjeha **proizvoljno velika**

BOOSTANJE: ORIGINALNI POSTUPAK (1990)

- particionirati skup za učenje X na podskupove X_1, X_2, X_3
- naučiti a1 s X_1, te ga evaluirati nad X_2
- iz X_2 odabrati sve krivo klasificirane plus jednako toliko dobro klasificiranih uzoraka, te dobivenim skupom naučiti a2
- evaluirati a1, a2 nad X_3; različito klasificiranim uzorcima naučiti a3



- klasifikaciju vršimo s a1 i a2: ako je odgovor različit, a3 presuđuje.
- ohrabrujući **rezultati**, ali zahtjeva se veliki skup za učenje.

BOOSTANJE: ADABOOST.M1 (1996)

Modifikacija originalnog algoritma boostanja:

- koristi se isti skup za učenje za sve osnovne algoritme
- broj osnovnih algoritama može biti proizvoljan

Posuđuje se ideja iz bagginga (slučajno uzorkovanje s ponavljanjem):

- skup za učenje svakog osnovnog algoritma uzorkuje se (s ponavljanjem) iz zajedničkog skupa
- međutim, vjerojatnost izvlačenja pojedinih uzoraka nije jednaka!
- **ideja**: modulirati distribuciju vjerojatnosti biranja uzoraka u ovisnosti o uspješnosti klasifikacije prethodnim algoritmom!
- **preciznije**: vjerojatnost uzorkovanja ispravno klasificiranih uzoraka u sljedećem koraku bit će **smanjena**

BOOSTANJE: ADABOOST.M1 (1996) UČENJE

```
# X = {x_t, r_t}_{t=1}^N ... ukupni skup za učenje
# p = {p_jt} ... distribucija za odabir X_j
# X_j ... skup za učenje j-tog algoritma d_j
# δ_jt ∈ {0,1} ... da li je t-ti rezultat d_j pogrešan?
# ε_j ∈ [0,1] ... ponderirana ukupna greška algoritma d_j
# β_j ∈ [0,1⟩ ... faktor promjene p_jt za ispravno klasificirane x_t
def AdaBoostLearn(X):
  p_1t=1/N, ∀t
  for j=1,...,L:
    # nauči klasifikator d_j na ponderiranim uzorcima
    X_j=uzorkuj(X,p)
    d_j=nauči(X_j)

    # odredi ponderiranu pogrešku ε_j
    δ_jt=(d_j(x_t)≠r_t), ∀t
    ε_j=∑_t p_jt δ_jt
    if ε_j>1/2:
      L=j-1; break;

    # odredi faktor promjene težina uzoraka β_j
    β_j=ε_j/(1-ε_j)

    # ažuriraj i normaliziraj težine uzoraka
    p_{j+1,t}=β_j^{1-δ_jt} p_jt, ∀t
    p_{j+1,t}=p_{j+1,t}/∑_t p_{j+1,t}
  return {d_j, β_j}
```

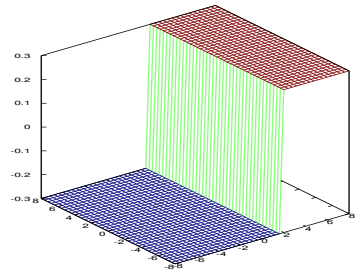
BOOSTANJE: ADABOOST.M1 (1996) KLASIFIKACIJA

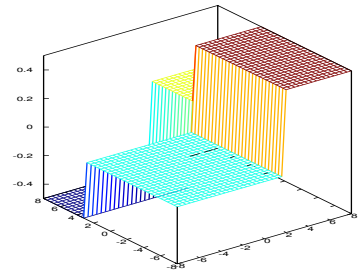
```
# x ... ulazni vektor
# {d_j, β_j} ... naučeni boostani klasifikator
# y ... rezultat klasifikacije
def AdaBoostClassify(x, {d_j, β_j}):
  y_i=∑_j log(1/β_j) d_ji(x), ∀i
  return sgn(y)
```

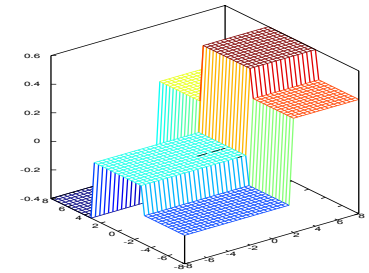
Za algoritam je ključan odabir faktora β_j promjene težine ispravno klasificiranih uzoraka:

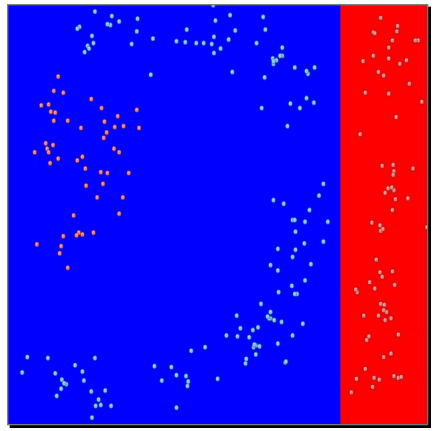
- zašto β_j treba biti baš ϵ_j/(1-ϵ_j)?
- zašto je veza između β_j i faktora w_j algoritma d_j baš $w_j = \log(1/\beta_j)$?
- na ta pitanja ćemo se vratiti kasnije :-)

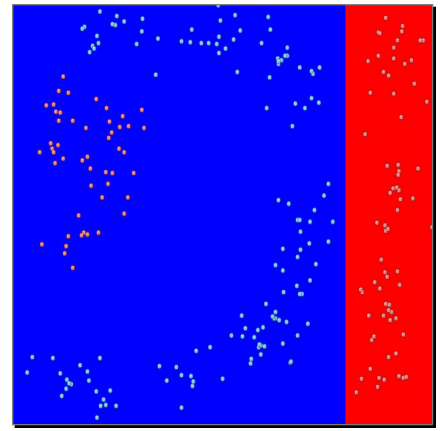
Boostanje: AdaBoost ilustracija (1)

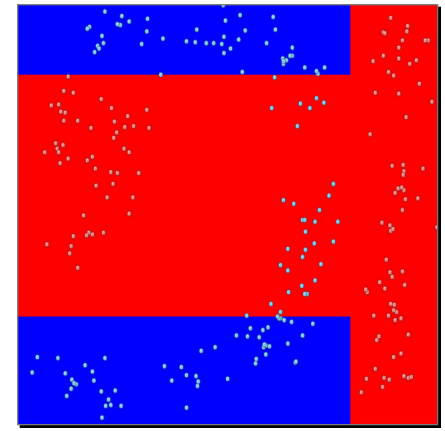


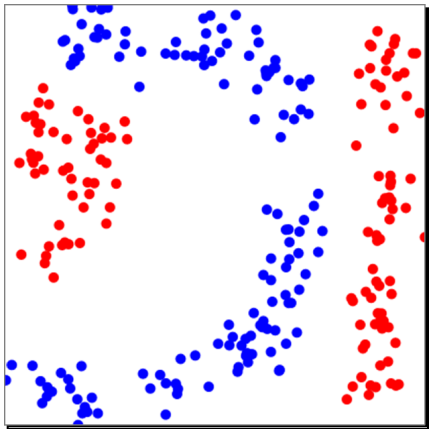


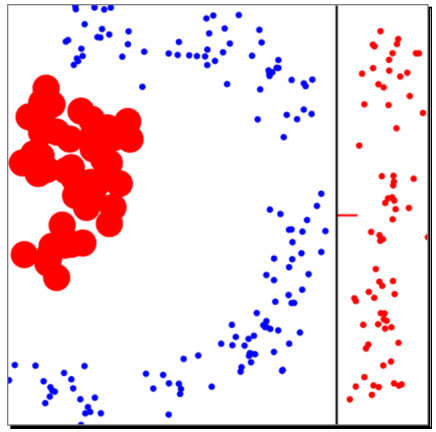


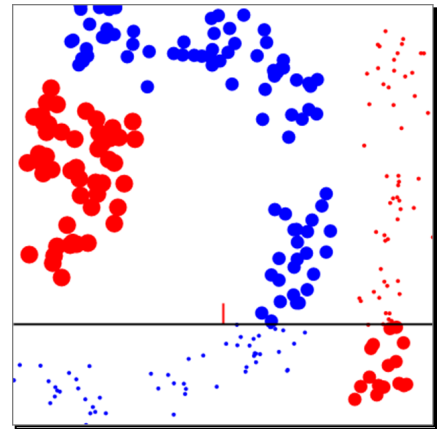


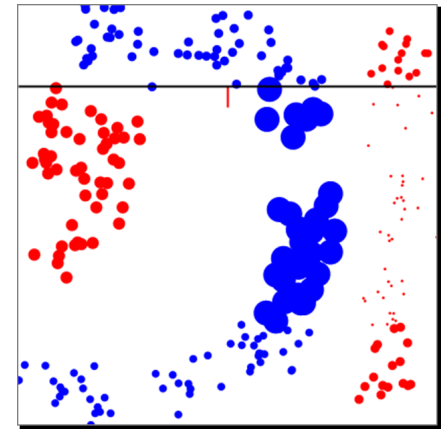






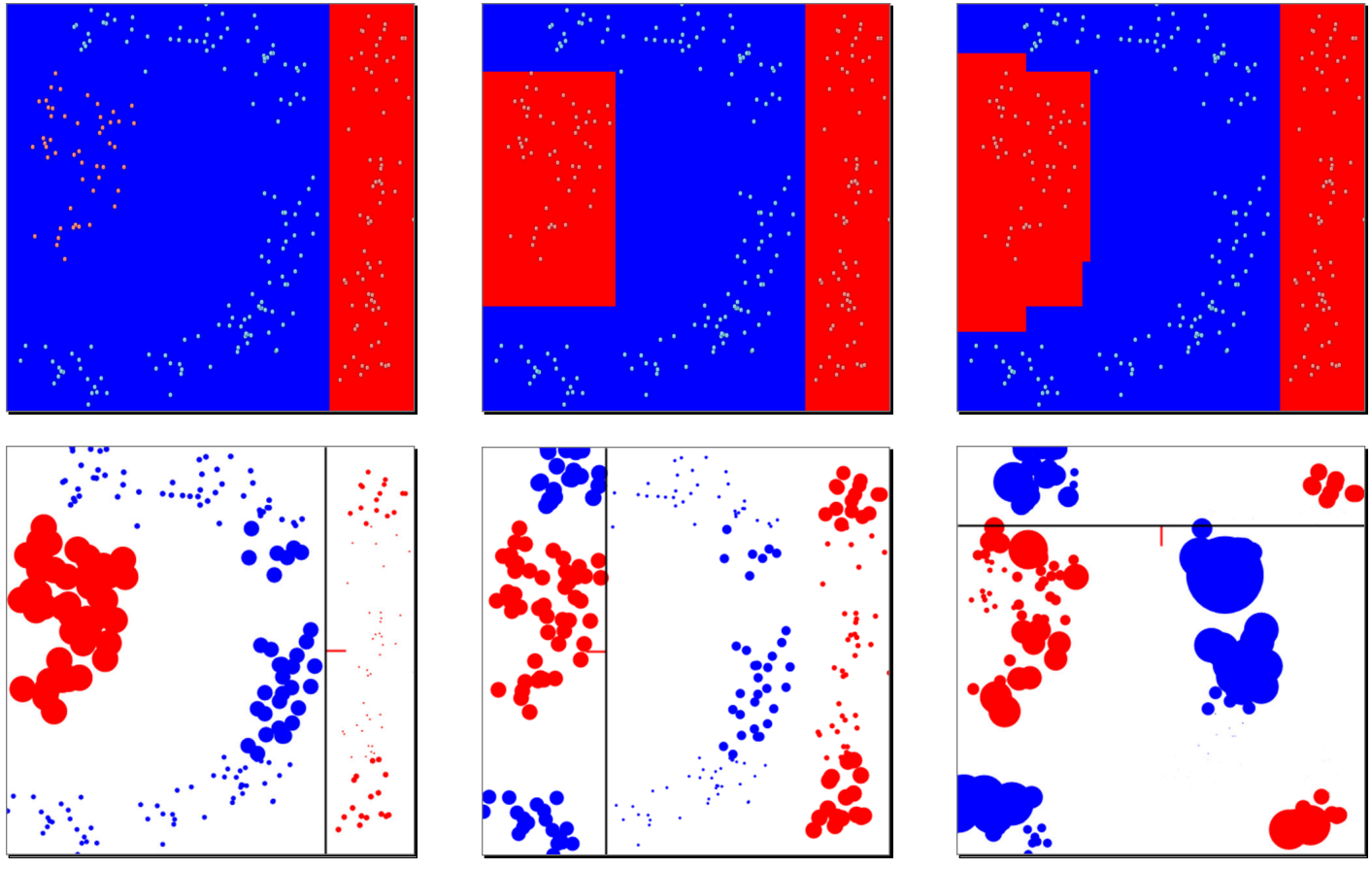






k=0 k=1 k=2 k=3

Boostanje: AdaBoost ilustracija (2)



k=4 k=5 k=50

[Kim,Shotton,Stenger;ICCV09 tutorial]

BOOSTANJE: ADABOOST, SVOJSTVA

- boostanje **optimira** marginu između razreda (kao i SVM)
- u praksi, boostanje se pokazuje relativno otpornim na pretreniranje
- osnovni algoritmi moraju biti **podesivi** i **jednostavni**:
  - inače, sljedbenici će raditi na uzorcima u kojima se ponavljaju sumnjivi podatci (šum ili krivo označavanje)
  - obično boostamo plitka stabla odluke (decision stumps) ili slične postupke s velikom varijancom
  - linearne diskriminante nema smisla boostati
- nakon učenja, boostanje se koristi kao metoda glasanja:
  $y_i = \sum_j w_j d_{ji}(\mathbf{x})$, $w_j = \log(1/\beta_j)$

BOOSTANJE: ADABOOST, PRIMJENE

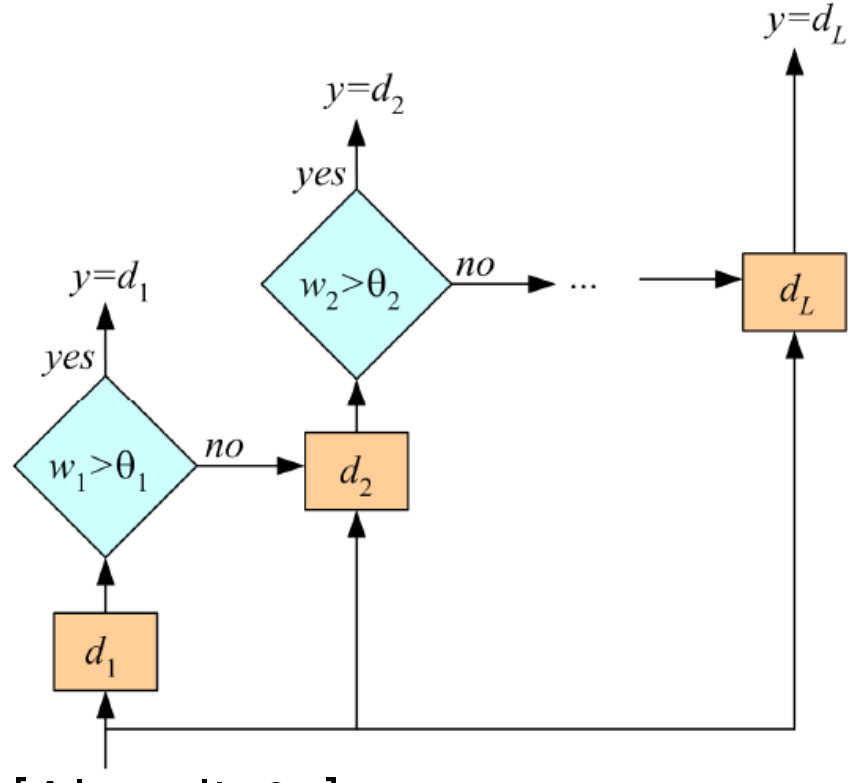
- rezultati iz originalnog članka upućuju na široku primjenljivost metode, pod uvjetima:
  - osnovni algoritam **treba** biti slab, podesiv, s velikom varijancom
  - skup za učenje treba biti velik
  - nema puno šuma i krivo označenih podataka za učenje
- Boostanje je popularno za klasifikaciju u računalnom vidu (glavni problem velika varijabilnost objekata i pozadina):
  - mnogo postupaka zadovoljava kriterije osnovnog algoritma
  - Boostanje je posebno zanimljivo kad je vrijeme za obradu ograničeno (detekcija objekata binarnom klasifikacijom)
  - najbolji rezultati prepoznavanja ipak se dobivaju jakim klasifikatorima (LDA, SVM)

KASKADIRANJE: UVOD

Kao kod boostanja, i ovdje skup osnovnih algoritama konstruiramo tako da sljedbenici **uče** na greškama prethodnika

Međutim, konačni rezultat ne dobivamo linearnom kombinacijom:

- ideja je osnovne algoritme sortirati prema složenosti
- sljedbenik se primijenjuje samo ako je prethodnik nesiguran!
- algoritam d_j pored rezultata d_{ji} producira i pouzdanost w_{ji}
- definiraju se pragovi pouzdanosti θ_j: $1/K < \theta_j < \theta_{j+1} < 1$



[Alpaydin04]

Konačno, vrijedi: $y_i = d_{ji}$, i to samo ako $w_j > \theta_j \wedge w_k < \theta_k \quad \forall k < j$

KASKADIRANJE: DETALJI

Tijek učenja:

- u j-tom koraku učimo klasifikator d_j nad skupom $\mathcal{X}_j$.
- formira se skup $\mathcal{X}_{j+1}$ s uzorcima za koje vrijedi $w_{ji} >= \theta_j$
- $\mathcal{X}_{j+1}$ može sadržavati i pravilno i nepravilno klasificirane uzorke

Kaskadiranje je jedinstvena metoda po tome što se pri klasifikaciji može dogoditi da se svi osnovni algoritmi ne primijene

Ova metoda se primijenjuje kad je vrijeme odziva vrlo važno

KASKADIRANJE: PRIMJENE

Osnovni algoritmi mogu biti perceptroni s rastućim **složenostima evaluacije** (npr, brojevi neurona)

U računalnom vidu, popularno je kaskadiranje boostanih Haarovih klasifikatora pri detekciji objekata u **stvarnom vremenu**

Kaskadiranje se može promatrati kao srednji put između parametarske i neparametarske klasifikacije:

- parametarska klasifikacija pokušava naći jezgrovito pravilo koje objašnjava sve podatke
- neparametarska klasifikacija (npr, kNN) sprema gomilu uzoraka bez generiranja pravila koje bi ih objasnilo
- kaskadiranje objašnjava uzorke glavnim jednostavnim pravilom, te skupom složenijih pravila koja opisuju iznimke

KASKADIRANJE: KRAJ