

Strojno učenje

Ansambli modela

Tomislav Šmuc

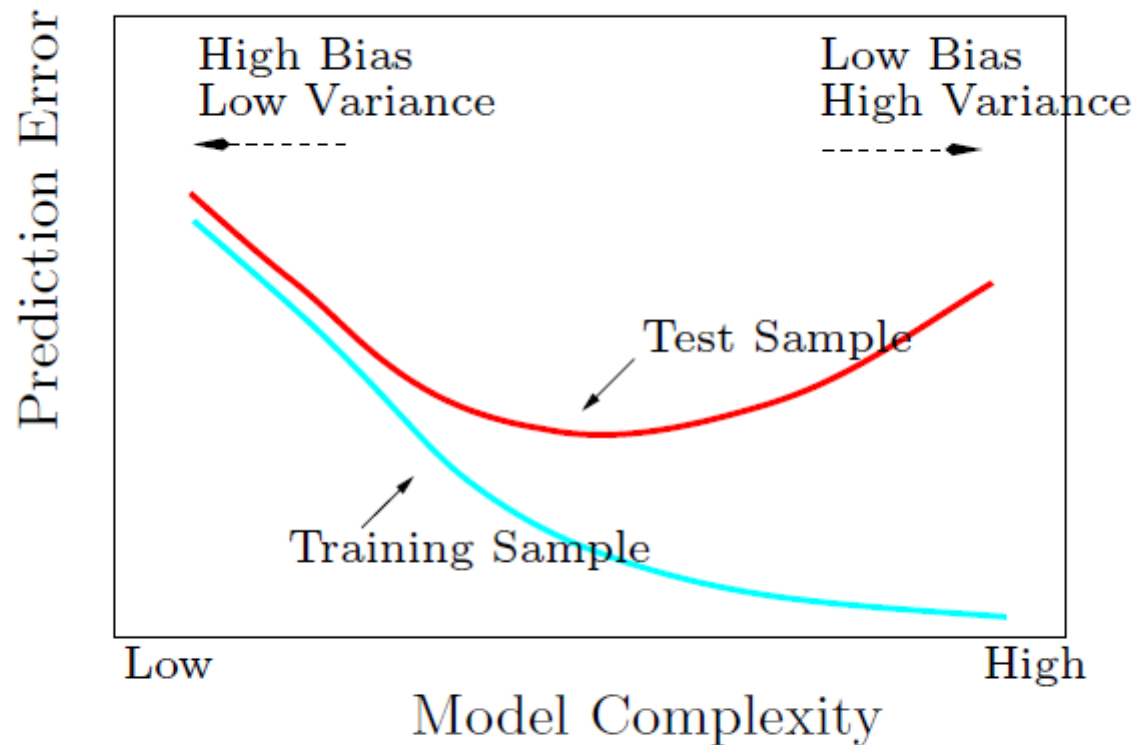


FIGURE 2.11. *Test and training error as a function of model complexity.*

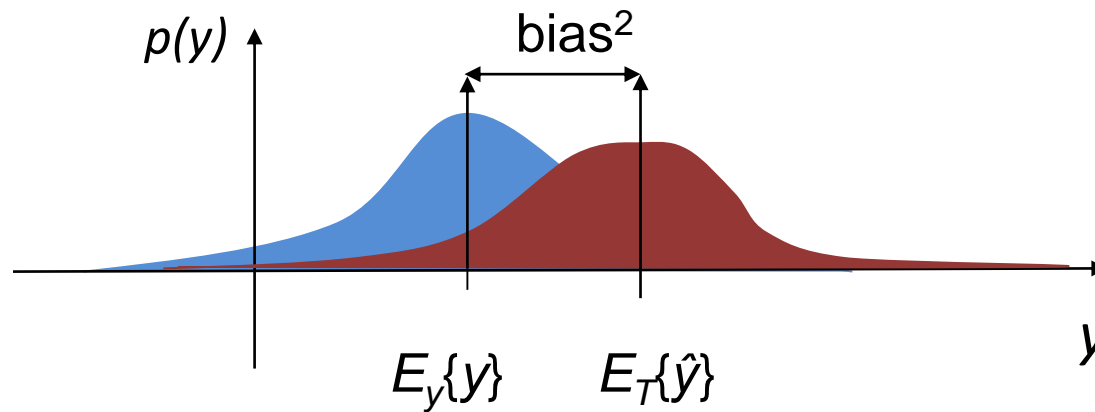
Očekivane vrijednosti

$$E(e) = \text{bias}^2 + \text{varijanca} + \text{šum}$$

$$E_T[(y - \hat{y})^2] = \underbrace{(E_T[\hat{y}] - y)^2}_a + \underbrace{E_T[(\hat{y} - E_T[\hat{y}])^2]}_b + \underbrace{E[\varepsilon | x]}_c$$

- **Pristranost/Bias:** sistematska greška na točki x - prosjek preko “svih” skupova za učenje T veličine N
- **Varijanca:** Varijacija greške oko prosječne vrijednosti prediktora (\sim razl. Podaci za učenje)
- **Šum:** Greška u određivanju stvarnih vrijednosti $y(x)$

Dekompozicija prediktivne pogreške: Pristranost i varijanca modela

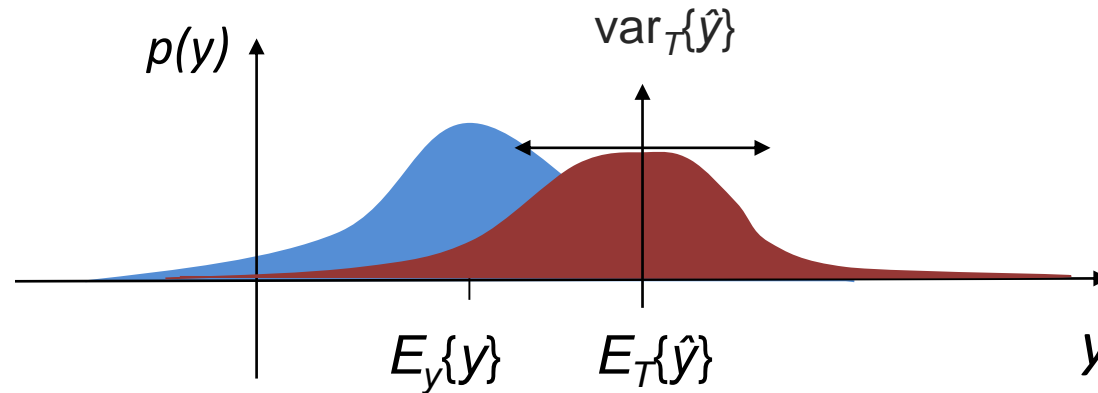


$$(E_y\{y\} - E_T\{\hat{y}\})^2$$

$E_T\{\hat{y}\}$ = prosječni rezultat modela (preko svih T)

bias^2 = greška između stvarne vrijednosti i prosječnog estimacijskog modela

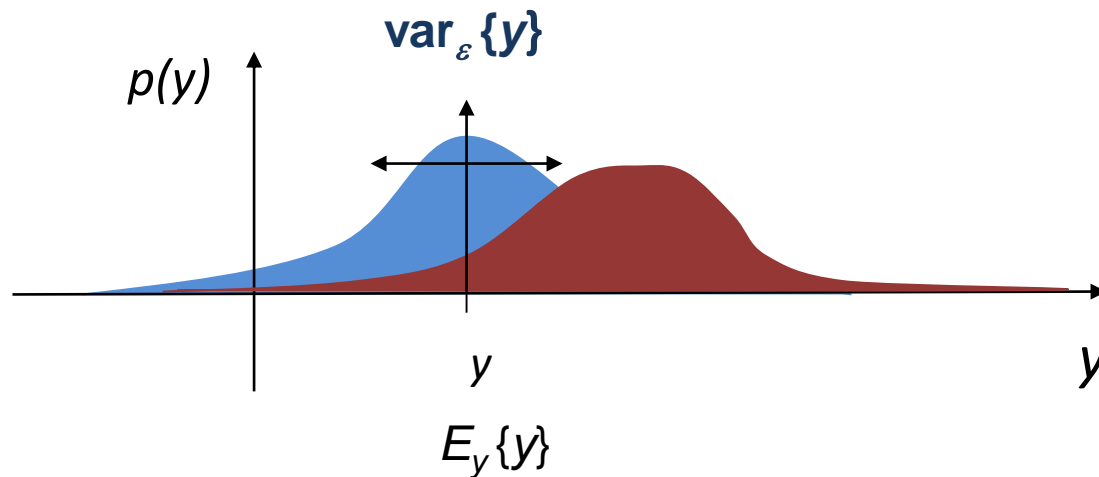
Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



$$\text{var}_T\{y\} = E_T\{(\hat{y} - E_T\{\hat{y}\})^2\}$$

$\text{var}_T\{\hat{y}\}$ = estimacijska varijanca = zbog over-fitinga

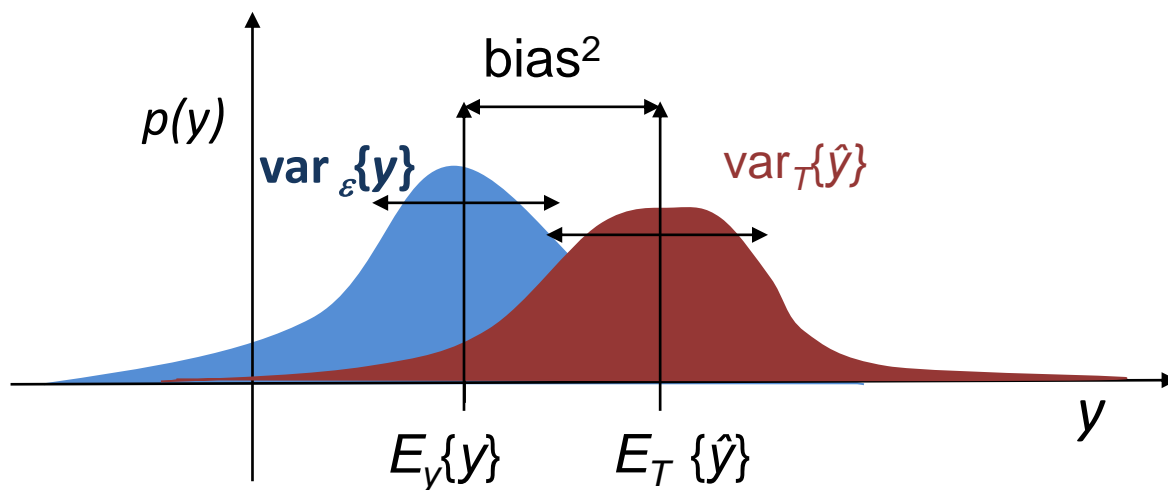
Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



$$\text{var}_\epsilon\{y\} = E_y\{(y - E_y\{y\})^2\}$$

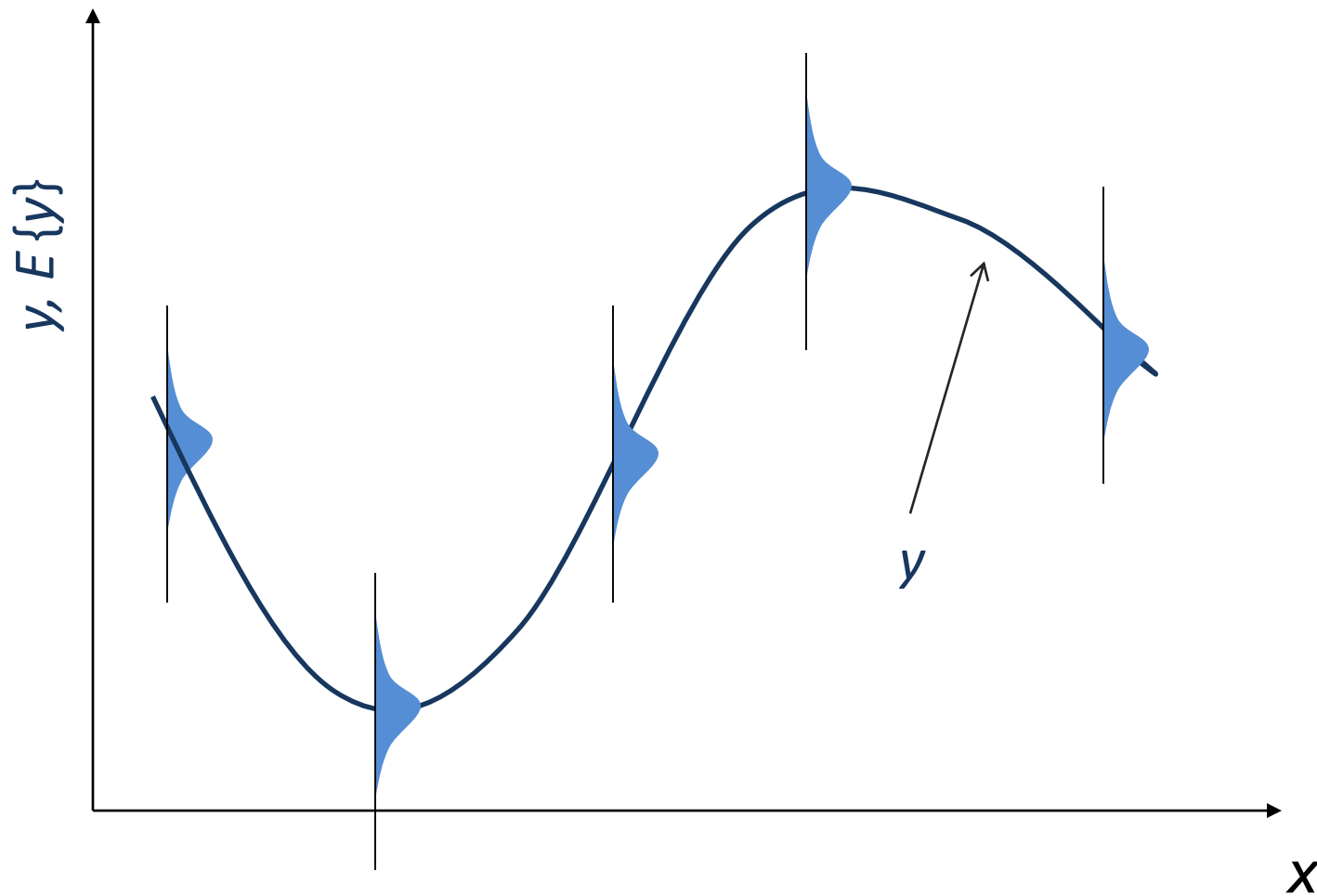
rezidualna greška = minimalna greška koju možemo dostići

Dekompozicija prediktivne pogreške: Pristranost i varijanca modela

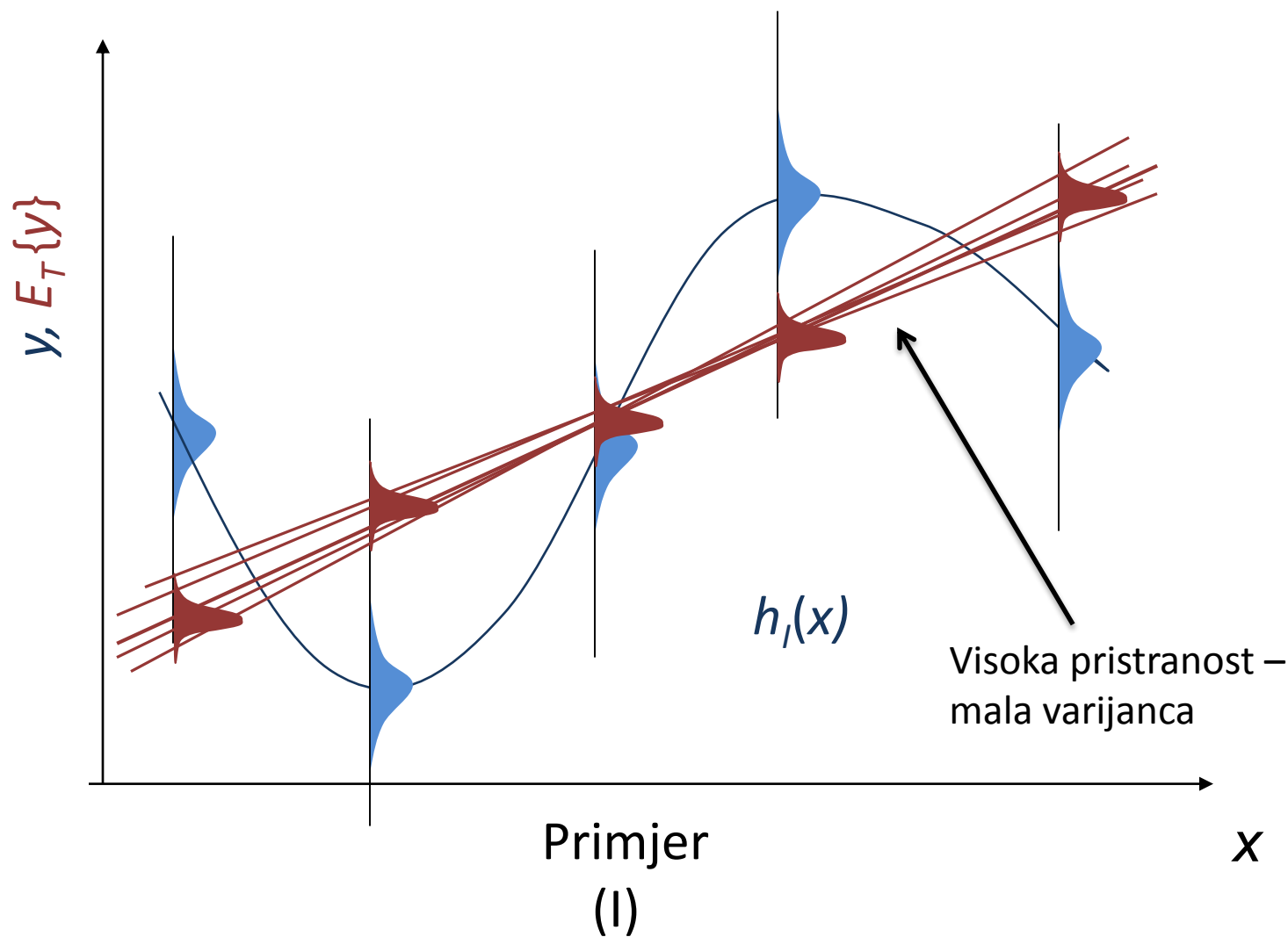


$$E = \text{var}_\varepsilon\{y\} + \text{bias}^2 + \text{var}_T\{\hat{y}\}$$

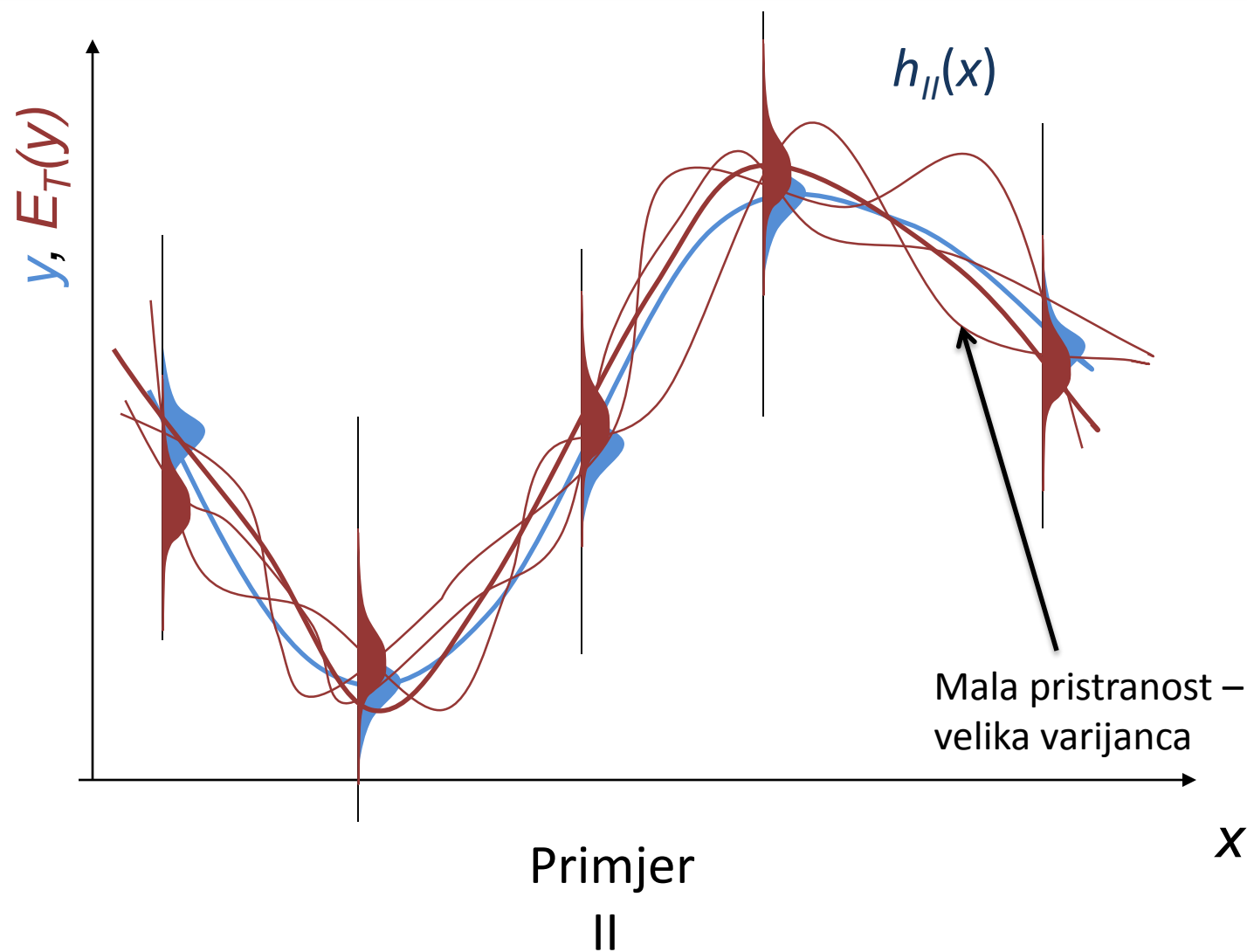
Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



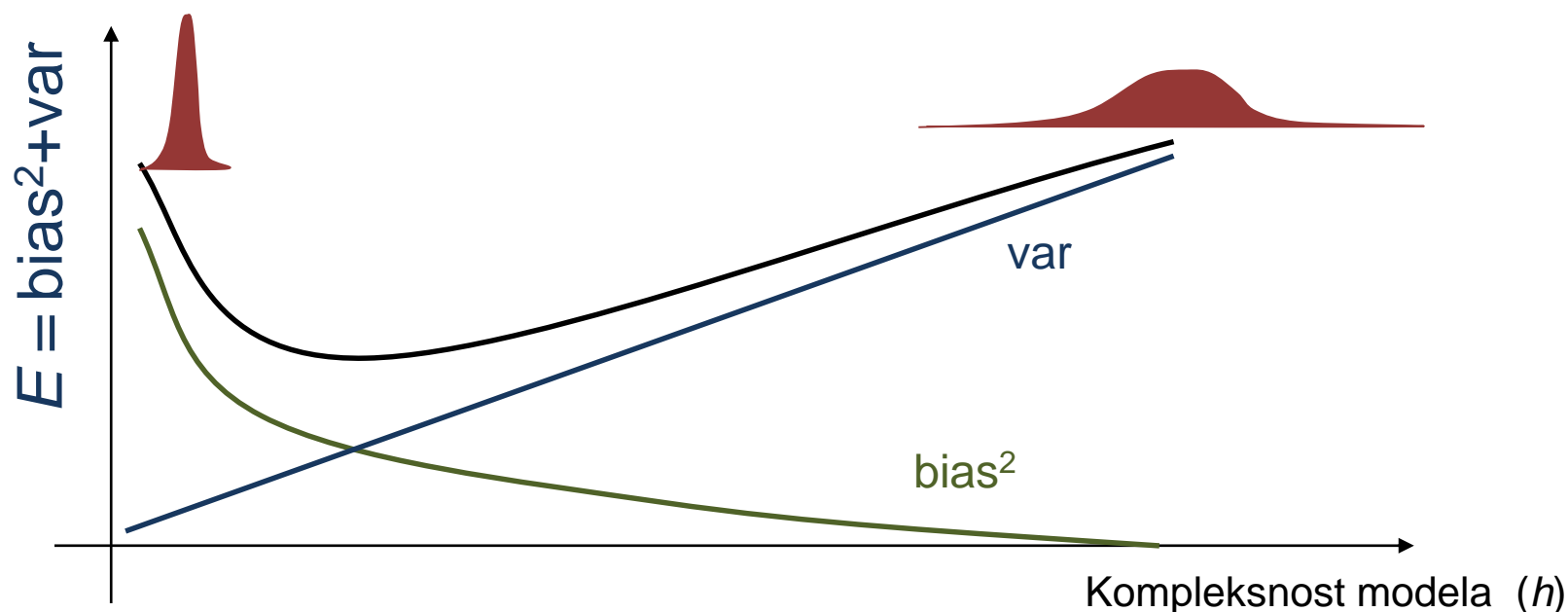
Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



Pristranost (bias) obično pada s povećanjem kompleksnosti modela, dok se varijanca povećava s kompleksnosti modela

Ansambli

IDEJA:

- Umjesto jednog modela/klasifikatora uči se skup/ansambl modela/klasifikatora
- Kombiniraju se predikcije više klasifikatora

MOTIVACIJA:

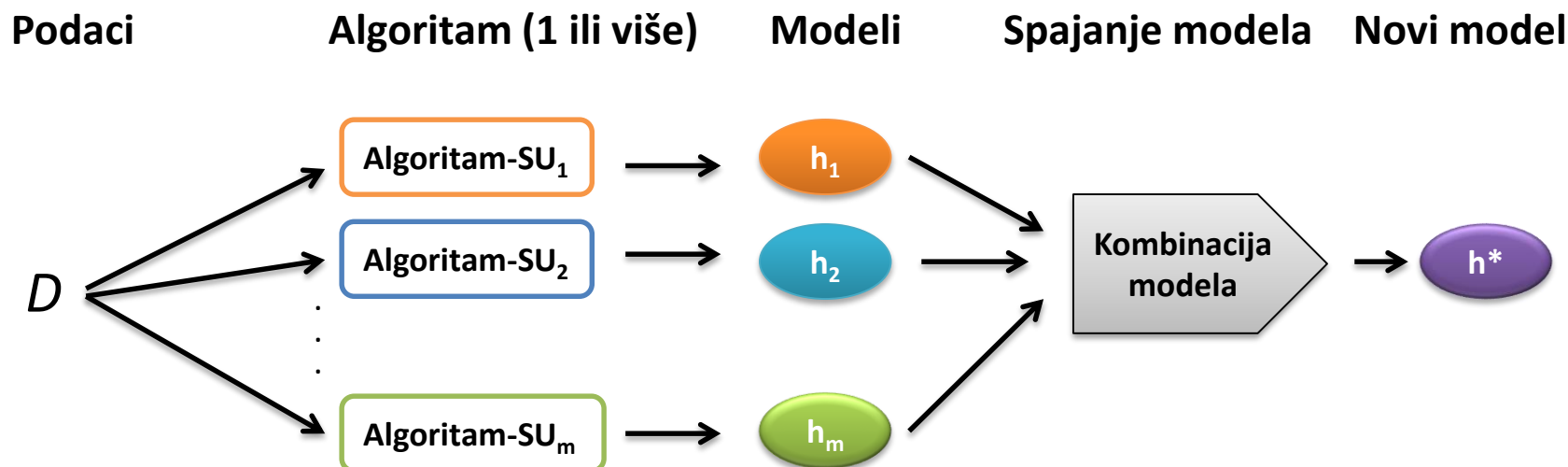
- Redukcija varijance: rezultati združenog modela manje ovise o specifičnostima jednog skupa za učenje
- Redukcija pristranosti: kombinacija više modela/klasifikatora može učiti/reprezentirati složenije koncepte nego jedan model/klasifikator

KLJUČNI KORAK:

- Kako formirati ansambl klasifikatora – na osnovu samo jednog skupa za treniranje

Ansambli (en Ensembles)

= Kombiniranje predikcija više modela koji su napravljeni s istim/različitim algoritmom na istim/različitim podacima - s ciljem poboljšavanja predikcije u odnosu na jedan model



Ansambli (en Ensembles)

= Kombiniranje modela

Zašto ansambli ?

Pretpostavimo da imamo L nezavisnih modela (klasifikatora npr.), čija je individualna točnost p , tada se može pokazati da vrijedi:

$$P(\hat{y} = y) = \sum_{k=0}^{\lfloor L/2 \rfloor} p^{L-k} (1-p)^k$$

Gdje je:

$P(\hat{y} = y)$ - vjerojatnost točne klasifikacije ansambla

$\lfloor L/2 \rfloor \Rightarrow$ najveći cijeli broj $\leq L/2$
(većina pobjeđuje)

... točnost ansambla-klasifikatora dobivenog glasanjem L nezavisnih klasifikatora točnosti p (većina pobjeđuje)

| | $L=3$ | $L=5$ | $L=7$ |
|---------|-------------------------|-------------------------|-------------------------|
| $P=0.6$ | 0.648 | 0.683 | 0.733 |
| $P=0.7$ | 0.784 | 0.837 | 0.901 |
| $P=0.8$ | 0.896 | 0.942 | 0.980 |

Osnovni problemi

1. Kako učiti/generirati bazne modele (klasifikatore) h_1, h_2, \dots, h_m
 - Različiti algoritmi ili različiti podaci
2. Kako ih kombinirati u procesu odlučivanja

$$h^* = F(h_1(x), h_2(x), \dots, h_m(x))$$

$F(\mathbf{h})$ – prosječna vrijednost, težinski usrednjena
prosječna vrijednost, većinsko glasanje

Tipovi ansambl metoda

1. Bazirani na učenju nad različitim dijelovima/distribucijama iz skupa podataka za učenje
 - Bagging; Boosting
2. Manipuliranje izlaznim varijablama
 - ECOC (Error Correcting Output Coding)
 - Stacking (stacked generalization)
3. Zašto ansambli (dobro) funkcioniraju ?

Ansambli

Pogled na bagging i boosting

– Tehnike usrednjavanja:

- “Paralelno/nezavisno” generirani modeli - usrednjena predikcija
- Bagging, random forests
- Ovim pristupima smanjuje se primarno varijanca greške

– Tehnike “boosting” tipa (en. boost - pojačati)

- “Sekvencijalno” generirani modeli
- Primjeri: Adaboost, Gradient-Boosting (XGBoost)
- Ovim pristupom smanjuje se primarno pristranost (bias)

(kasnija istraživanja pokazala su da boosting smanjuje i varijancu modela)

Bagging (Bootstrap AGGregatING)

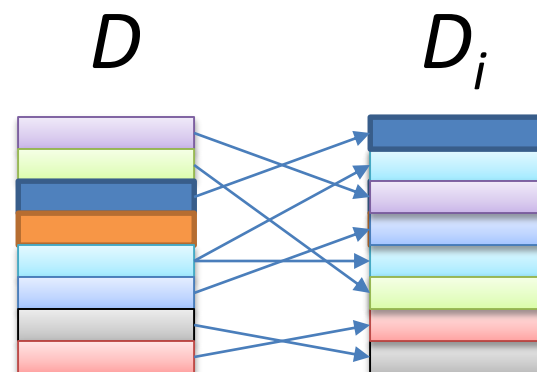
- Napraviti (velik) broj modela koristeći bootstrap replike podataka
- Spojiti u jedan zajednički (bagged) model – odnosno predikciju
- Svi modeli “glasaju”:
 - U slučaju klasifikacije – pravilo većine
 - U slučaju regresije – prosjek svih predikcija

Bootstrapping

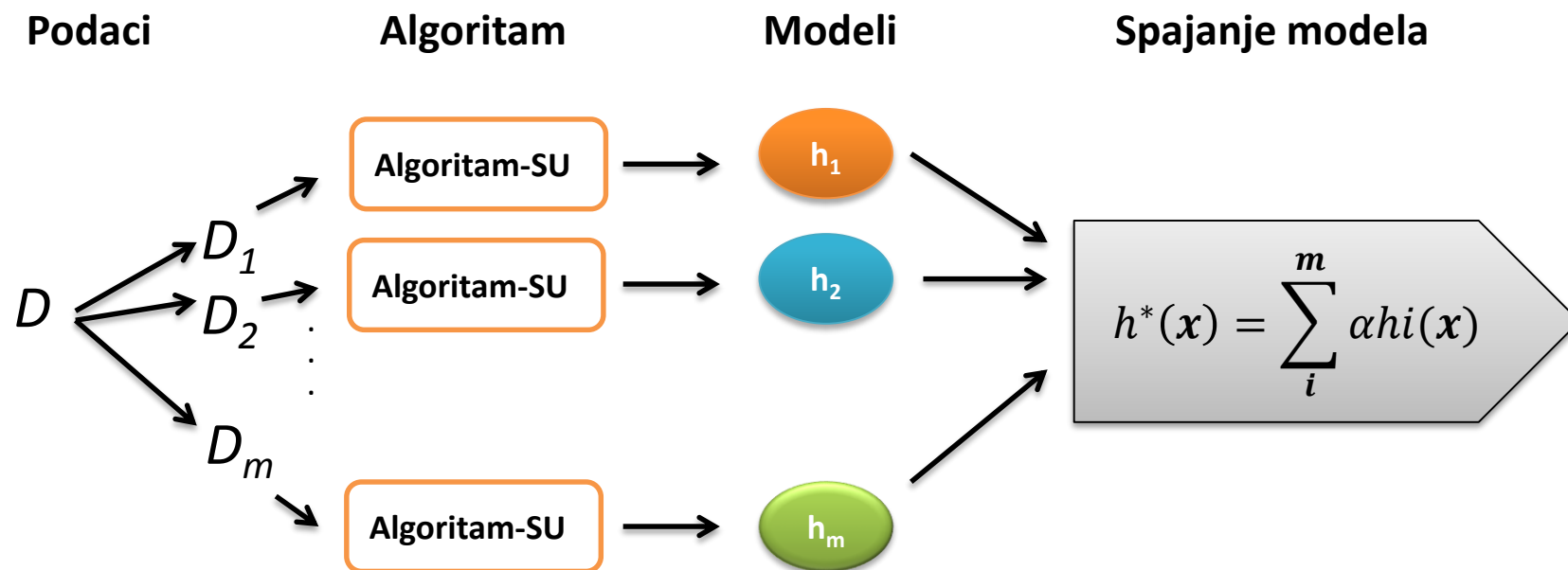
- metoda ponavljanog uzorkovanja iz skupa podataka (en resampling statistical method)
- Korišćenje skupa podataka za učenje da se naprave slučajni skupovi – replike originalnog skupa podataka, radi dobivanja informacija o nekim statistikama skupa podataka (bias, variance)

Bootstrap D_i replika skupa podataka D

- Dobiva se slučajnim uzorkovanjem (primjer po primjer) $|D|$ primjera iz D sa ponavljanjem (to znači da u D_i nalazimo i kopije istog primjera) (en. with replacement!)



Bagging



D_i - bootstrap replika podataka

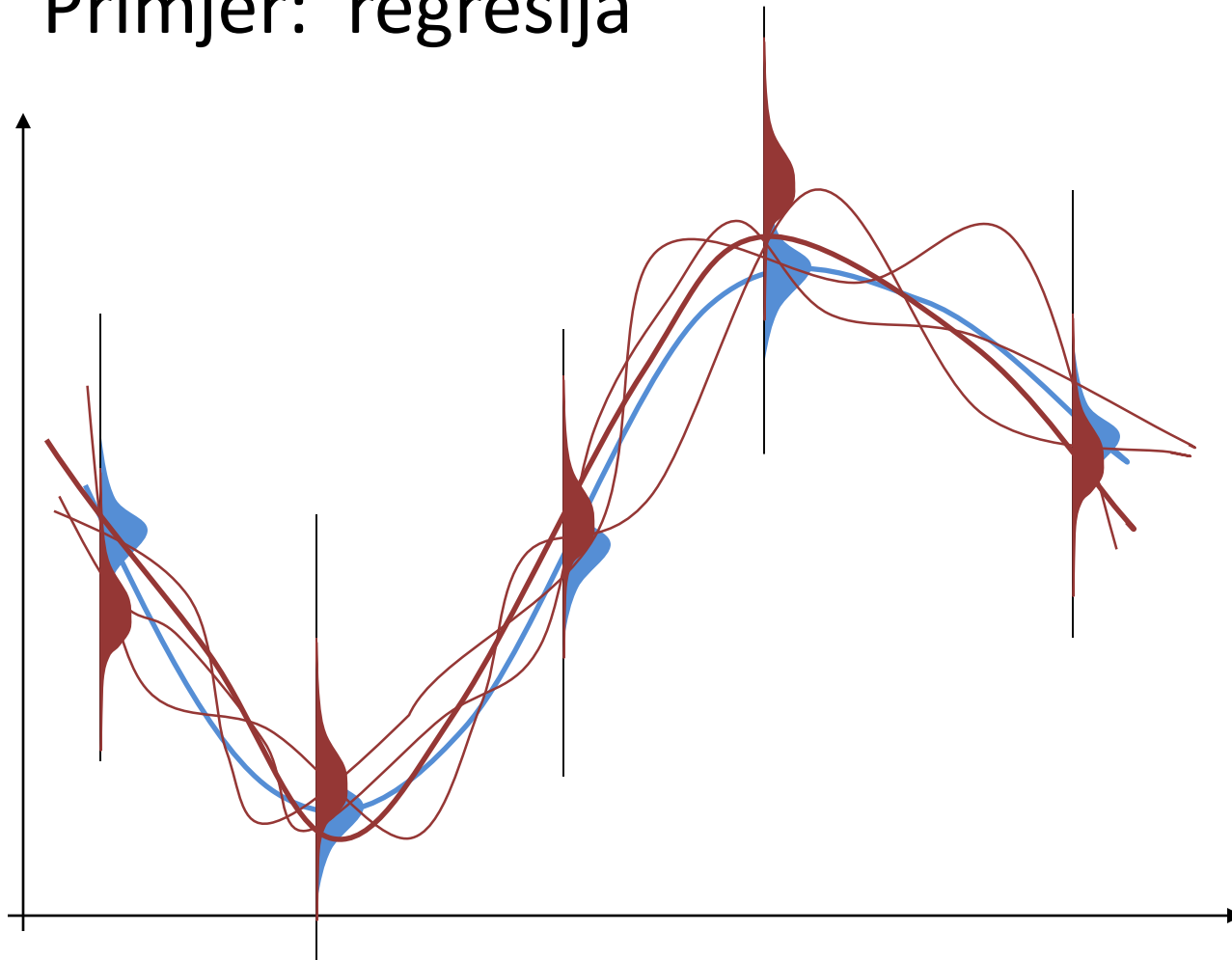
h_i - klasifikator baziran na D_i

$\alpha_i = 1/m$

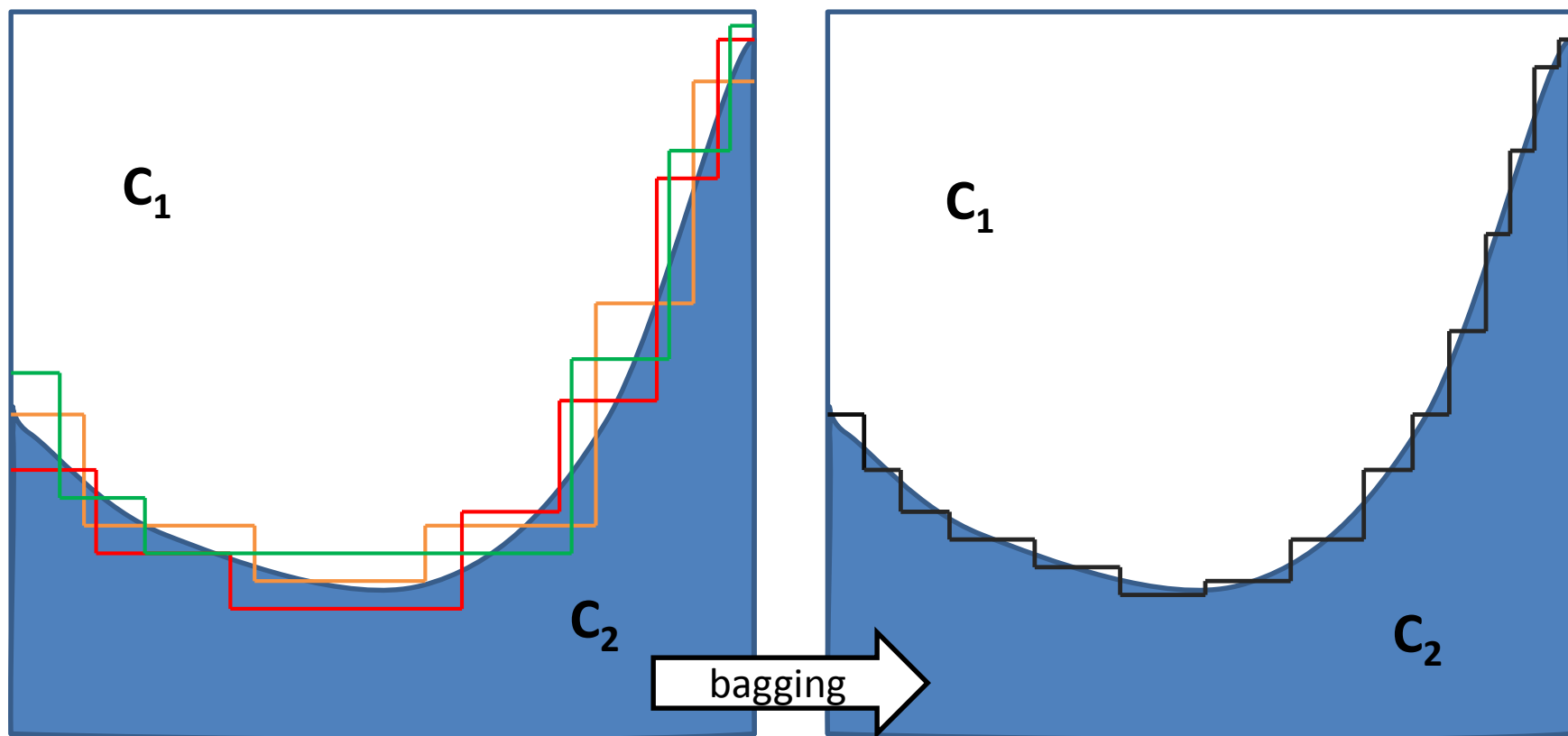
Bagging

- Koji modeli (tipovi algoritama) bi najviše dobili baggingom ?
- Baggingom eliminiramo varijancu => dakle kompleksni modeli
- To uključuje stabla odlučivanja – tipično “nestabilni” model, ali i Neuralne mreže ...

Primjer: regresija



klasifikacijski primjer: stabla odlučivanja



Ansambli: bagging

Očekivana kv. greška *jednog* modela (regresija)

$$h_i(\mathbf{x}) = y(\mathbf{x}) + \varepsilon_i(\mathbf{x})$$

$$E[(h_i(\mathbf{x}) - y(\mathbf{x}))^2] = E[(\varepsilon_i(\mathbf{x}))^2]$$

Prosječna kv. greška m modela:

$$\bar{E}_m = \frac{1}{m} \sum_i E[(\varepsilon_i(\mathbf{x}))^2]$$

Kolika je prosječna kv. greška h^* modela ? $h^*(\mathbf{x}) = \frac{1}{m} \sum_i h_i(\mathbf{x})$

.... uz pretpostavku da su greške modela $\varepsilon_i(\mathbf{x})$ takve da vrijedi

$$E[\varepsilon_i(\mathbf{x})] = 0 \quad \Rightarrow \text{srednja vrijednost} \sim 0$$

$$E[\varepsilon_j(\mathbf{x})\varepsilon_i(\mathbf{x})] = 0 \quad \Rightarrow \text{greške su medjusobno nekorelirane}$$

$$\bar{E}_{h^*} = E \left[\left(y(\mathbf{x}) - \frac{1}{m} \sum_i h_i(\mathbf{x}) \right)^2 \right] = E \left[\left(\frac{1}{m} \sum_i \varepsilon_i(\mathbf{x}) \right)^2 \right] = \frac{1}{m} \bar{E}_m$$

$$\bar{E}_{h^*} \leq \bar{E}_m !!$$

Bagging - Pristranost i varijanca

Pristranost(Bias): $E[h^*(\mathbf{x}) - y]$

Varijanca: $\sum_i (h^*(\mathbf{x}) - h_i(\mathbf{x}))^2 / (m - 1) \Rightarrow 0$, za $m \gg$

Dakle:

- Bagging reducira varijancu
- To je povoljno svojstvo za algoritme/modele koji imaju visoku varijancu, a mali bias („sklonost overfitanju”)
 - stabla odlučivanja, 1-nn...

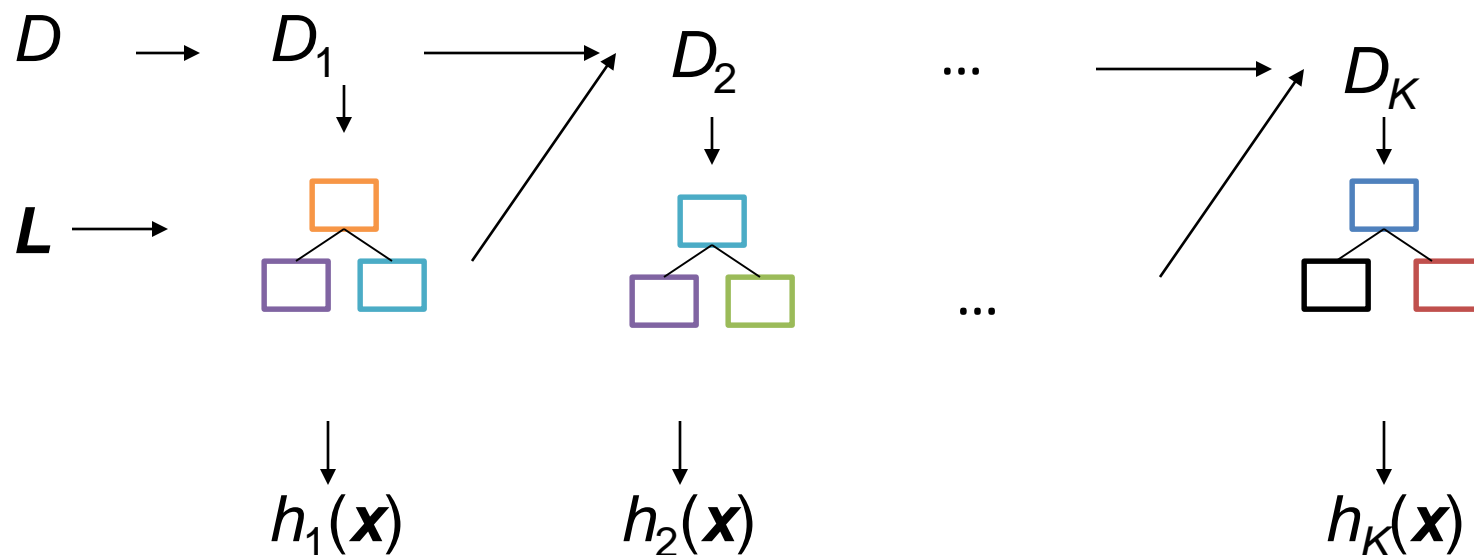
Random forests algoritam

- Kombinira bagging sa slučajnim odabirom podskupa varijabli/atributa (perturbacija modela)
 - Gradi stabla odlučivanja iz bootstrap uzorka skupa za učenje
 - Umjesto izabiranja najboljeg atributa za *split* – između svih atributa – izabire $k(split)$ slučajno odabranih atributa
(= bagging , uz n atributa u skupu podataka:
=> tipično za RF broj splitting atributa $k(split)=\sqrt{n}$)
- Balans bias/varijanca korištenjem k :
 - Što je manji k – veća je redukcija varijance, ali je i veća pristranost (bias) stabala

Boosting metode – “jačanje” slabih modela

- Motivacija:
 - kombiniranje outputa “slabih” modela da bi se napravio točniji ansambl modela.
- “Slabi” modeli:
 - modeli s visokom pristranosti (bias) (klasifikacija - malo bolji od slučajne predikcije)
- U odnosu na bagging:
 - Modeli se rade “sekvencijalno” **na modificiranim verzijama podataka**
 - Krajnja predikcija je kombinacija predikcija pojedinačnih modela uz korištenje težinskih faktora

Ansampli: boosting



Klasifikacija: $h(\mathbf{x}) = \text{većina od } \{h_1(\mathbf{x}), \dots, h_K(\mathbf{x})\}$
uz težine $\{\beta_1, \beta_2, \dots, \beta_K\}$

Regresija: $h(\mathbf{x}) = \beta_1 h_1(\mathbf{x}) + \beta_2 h_2(\mathbf{x}) + \dots + \beta_K h_K(\mathbf{x})$

AdaBoost (Adaptive Boosting) algoritam

- (Freund & Schapire, 1997)
- Generira modele tako da sukcesivno mijenja težine primjera u skupu za učenje
- Adaboost povećava težine primjera za koje su prethodni modeli imali loše predikcije – dakle fokusira učenje na “teške” slučajeve
- Na kraju: glasanje s težinskom-većinom; točniji modeli imaju veći utjecaj u glasanju

AdaBoost algoritam

Ulaz: **D** – podaci za učenje, $Y=[1,-1]$ – binarni klasifikacijski problem

Algoritam **L** (algoritam s visokim high biasom)

T – broj iteracija, N – broj primjera za učenje

Izlaz: “Ojačani” (boosted) klasifikator **F** (model s niskim biasom)

Inicijaliziraj težine primjera: $w_i^1 = 1/N$, za $i = 1, \dots, N$

Za $t \leftarrow 1, 2, \dots, T$ **radi**

1. $\mathbf{p}^t = \mathbf{w}^t / \sum_{i=1}^N w_i^t$
2. Pozovi algoritam **SU L** (\mathbf{p}^t, \mathbf{X}) \Rightarrow rezultat je “slabi” model ($h_t: \mathbf{X} \rightarrow Y$)
3. Odredi grešku $h_t: \varepsilon_t = \sum_{i=1}^N p_i^t \frac{1}{2} |h_t(x_i) - y_i|$
4. Odredi $\alpha^t = \log\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$
5. Odredi nove težine primjera $w_i^{t+1} = w_i^t \exp(\alpha^t \frac{1}{2} |h_t(x_i) - y_i|)$, za $i = 1, 2, \dots, N$

Vrati klasifikator **F**:

$$F(x) = \begin{cases} 1, & \text{ako je } \sum_{i=1}^T \alpha_i h_i(x) \geq 0, \\ -1, & \text{inace} \end{cases}$$

AdaBoost algoritam

Izraz kojim mijenjamo težine primjera u iteraciji $t+1$:

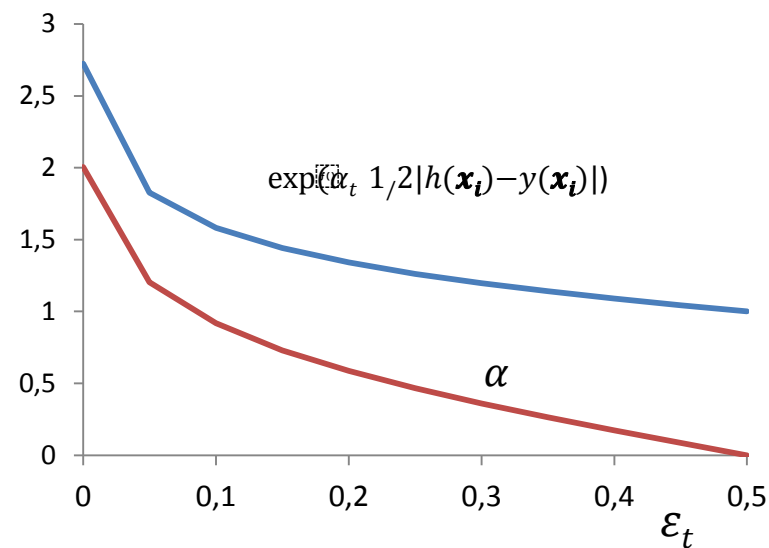
$$w_i^{t+1} = w_i^t \exp(\alpha_t \frac{1}{2} |h_t(x_i) - y_i|), \text{ za } i = 1, 2, \dots, N$$

Gdje je α_t – težinski faktor modela u iteraciji t :

$$\alpha_t = \log\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

a ε_t prosječna greška u iteraciji t :

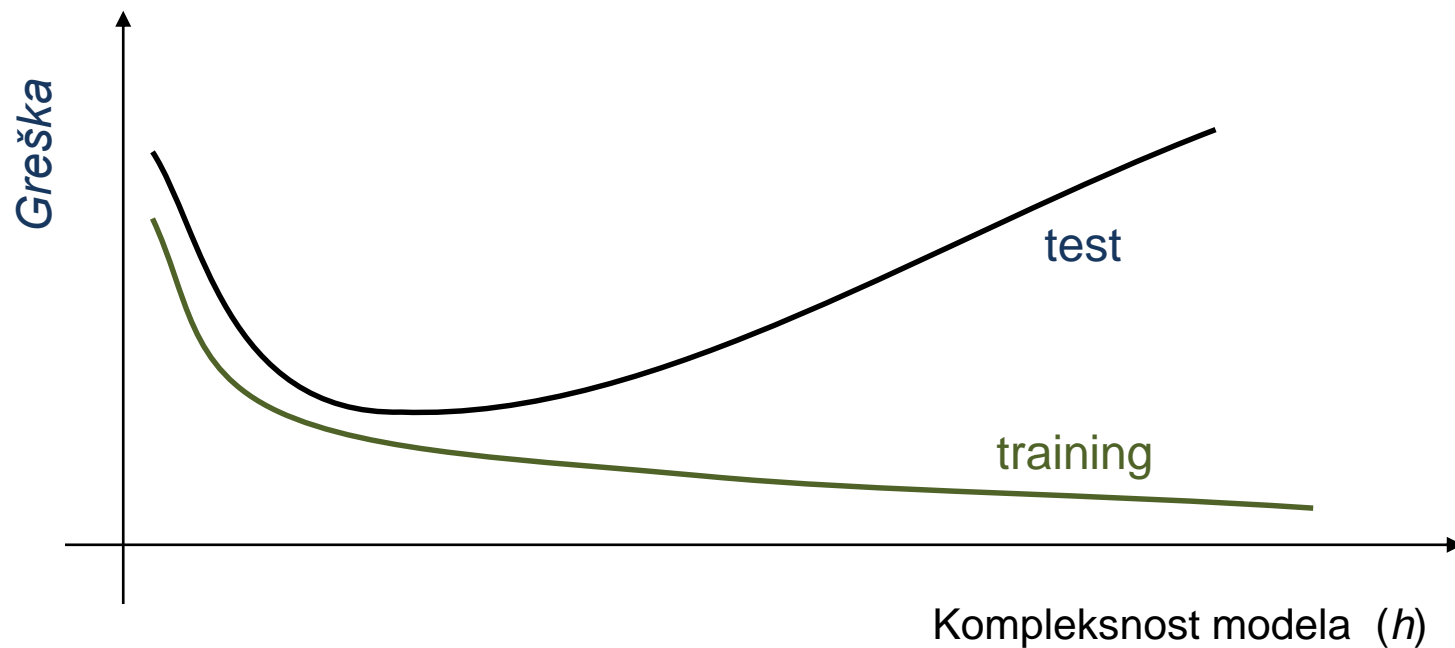
$$\varepsilon_t = \sum_{i=1}^t p_i \frac{1}{2} |h_t(x_i) - y_i|$$



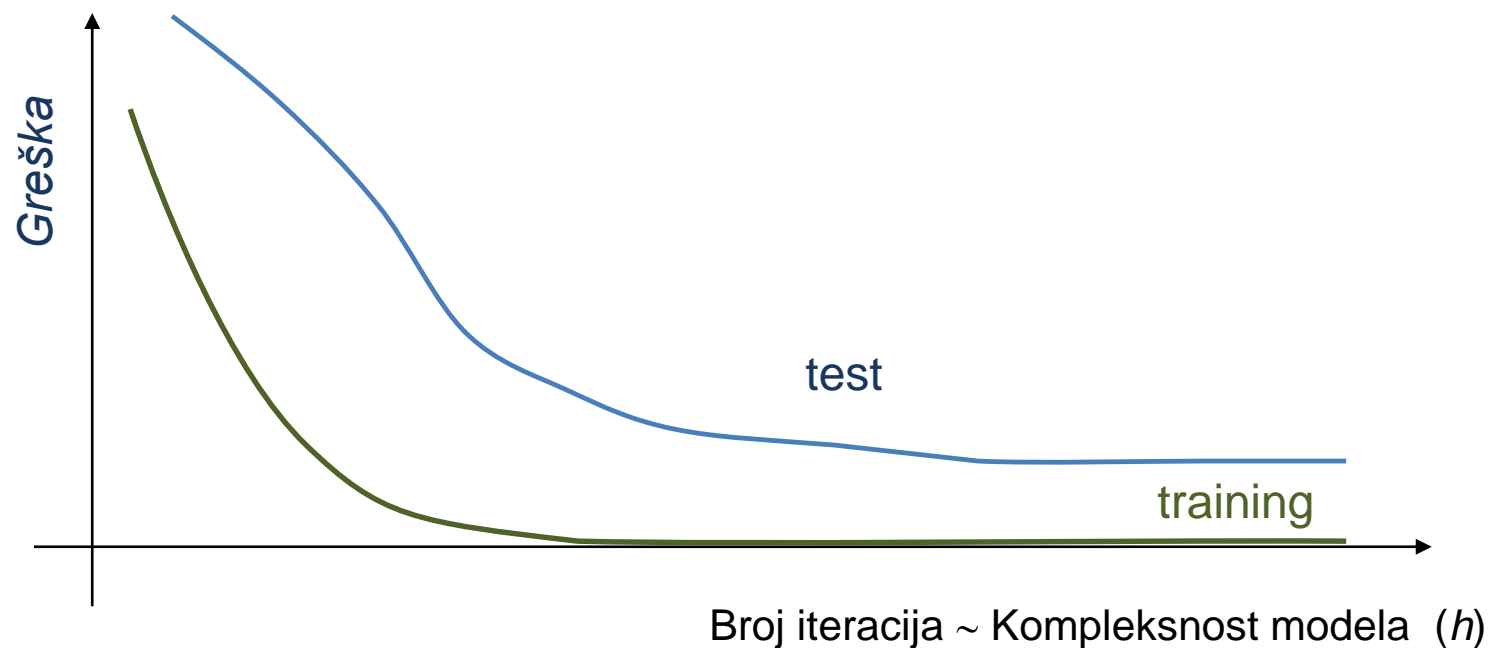
Zašto boosting radi dobro ?

- Kombinira modele koji imaju visoki bias (jednostavni), tako da se ukupno dobije kompleksniji/ekspresivniji klasifikator
- Boosting => redukcija pristranosti (bias-a), svakom iteracijom model postaje kompleksniji ?
- Što se dešava ako imamo velik broj iteracija (K) ? Dobit ćemo vrlo složeni model...a greška na novim primjerima – problem overfittinga ?

Algoritmi strojnog učenja - tipični slučaj



Boosting – tipični slučaj !?



Boosting – Objašnjenje (I)

- Modeli koji su generirani boosting-om ($Y=[-1,1]$):
 - Klasifikacija primjera je korektna ako je $h^*(\mathbf{x}) = y(\mathbf{x})$, odnosno pogrešna ako $h^*(\mathbf{x}) \neq y(\mathbf{x})$
- No – u boosting algoritmu **možemo mjeriti pouzdanost** klasifikacije !
 - $h^*(\mathbf{x})$ - je težinski zbroj glasova pojedinih (slabih) klasifikatora !
- **Mjera pouzdanosti \sim Margina (sjetite se SVM!) primjera:**
pouzdanost glasanja =
= (težinski zbroj korektnih glasanja)-(težinski zbroj krivih glasanja)

Boosting – Objašnjenje (II)

- Što je margina veća na skupu za učenje – za očekivati je i manju grešku i na testnom skupu (generalizacijska greška je manja)

Dakle:

- Iako je konačni klasifikator $h^*(\mathbf{x})$ naizgled složeniji, margina između primjera različitih klasa se povećava, dodavanjem novih modela !?
- dakle $h^*(\mathbf{x})$ na neki način postaje robustniji a istovremeno se smanjuje njegova greška na novim primjerima !
- Boosting algoritmi rade na povećavanju margine => smanjenju varijance modela
- Boosting algoritmi smanjuju pristranost, ali i varijancu konačnog modela !

Boosting – Problemi

Osjetljivost na outliere (podatke sa povećanim šumom)

- Primjeri koji mogu predstavljati greške – dobijaju sve veću težinu pri izgradnji $h^*(x)$!
- Novije varijante boosting algoritama robustnije su na outliere
 - gradient boosting (XGboost) => gradient descent + boosting

Usporedba Bagging vs Boosting

- Bagging

- Tolerira šum u podacima
- Dobro statistički utemeljena
- bazirana na slučajnom uzorkovanju (primjeri/bootstrap/)
- Daje bolju procjenu vjerojatnosti pripadanja nekoj klasi pri testiranju novih primjera

- Boosting

- Osjetljiva na šum u podacima
- bazirana na teoriji strojnog učenja
- Obično je moguće dobiti nešto točnije modele nego bagging metodom
- Lošija procjena vjerojatnosti pripadanja nekoj klasi pri testiranju novih primjera

Ansambli bazirani na manipuliranju ciljnom varijablom

- Algoritmi poput stabla odlučivanja, naivnog Bayes algoritma, neuralnih mreža u principu uče jedan model/klasifikator za višeklasne ($C > 2$) probleme.
- Moguće je konstruirati i meta-algoritme, koji odluku o pripadnosti klasi baziraju na binarnim klasifikatorima
- ECOC – meta-algoritam koji kombinira više (puno) binarnih klasifikatora da bi riješio problem razlikovanja više klasa.

Ansambli bazirani na manipuliranju ciljnom varijablom

Problem **sa više klasa (>2)** primjera: $y=1,2,\dots,K$ – klasa

Mogući pristupi:

1. Učenje K binarnih klasifikatora

- $y=1 \leftrightarrow (y=2,3,4,\dots,K)$
- $y=2 \leftrightarrow (y=1,3,4,\dots,K)$
-

Odluka: klasa s najvećom $P(C = C_i \mid \mathbf{X})$

2. Učenje $\log_2(K)$ binarnih klasifikatora (bitovi - indeksiranje K klasa)

- $h_0(\mathbf{x}) = 1$ ako $y=2,4,6,8$ inače 0
- $h_1(\mathbf{x}) = 1$ ako $y=3,4,7,8$ inače 0
- ...

Odluka: klasa s najvećim brojem glasova

→ ECOC – Error Corecting Output Coding

Error Correcting Output Coding

- 3 bitni kod je dovoljan za reprezentiranje 8 klasa ($\log_2 8 = 3$).

| | Binarni klasif. Problem / klasifikator | | |
|-------|--|----|----|
| | h1 | h2 | h3 |
| Klasa | | | |
| 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 |
| 5 | 1 | 0 | 0 |
| 6 | 1 | 0 | 1 |
| 7 | 1 | 1 | 0 |
| 8 | 1 | 1 | 1 |

- No, tipično – koriste se kodovi koji imaju više bitova nego što je potrebno !
- Korištenjem redundantnih bitova (\Rightarrow ime ECOC !!), omogućava robustniji klasifikator, te korigiranje grešaka (...koje su posljedica konačnog trening skupa, „loših” atributa/varijabli, odabira algoritma...)
- ECOC metoda se „oporavlja” od individualnih grešaka !

ECOC

- Ako je minimalna Hamming-ova udaljenost između 2 para kodnih riječi = d , tada će ECOC korigirati minimalno $(d-1)/2$ bitova:
 - Najbliža kodna riječ onoj koja označava pravu klasu (tj njenu kodnu riječ) je u tom slučaju još uvijek korektna:
 - ECOC model dakle može ispraviti najmanje $(d-1)/2$ bitova !

| | Binarni klasif. Problem / klasifikator | | | | | | | | | | | | | | |
|-------|--|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|
| | h1 | h2 | h3 | h4 | h5 | h6 | h7 | h8 | h9 | h10 | h11 | h12 | h13 | h14 | h15 |
| Klasa | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

Gornja tablica: 15 kodni zapis – 5-klasnog klasifikacijskog problema.

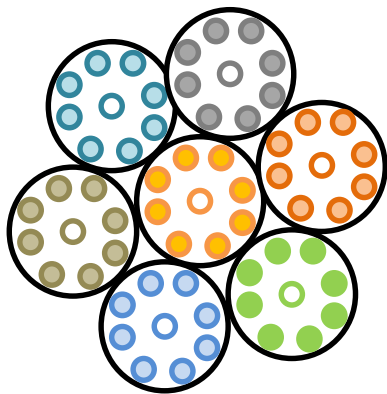
ECOC: Error Correcting Output Coding

Mapiraj klase koje možeš kodirati sa $\log_2(K)$ bita, u $M > \log_2(K)$
(redundancija => robustnost na šum)

Broj kodova >> broj poruka (klasa)

Kod dekodiranja klasa se određuje prema najbližoj kodnoj-riječi

- Svaka $\log_2(K)$ kodna-riječ (klasa) „okružena je” sa buffer-zonom sličnih M-bitnih kodnih-riječi – nijedna druga kodna-riječ(klasa) ne može biti mapirana u buffer-zonu



Klasa – kodna riječ

(Y=-1) (Y=1)

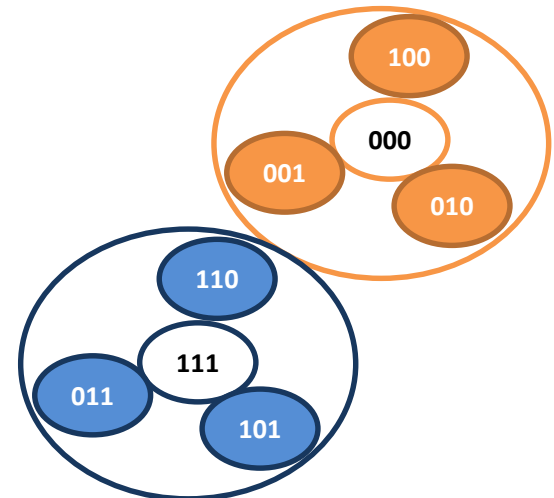
000 111

„Buffer” zona

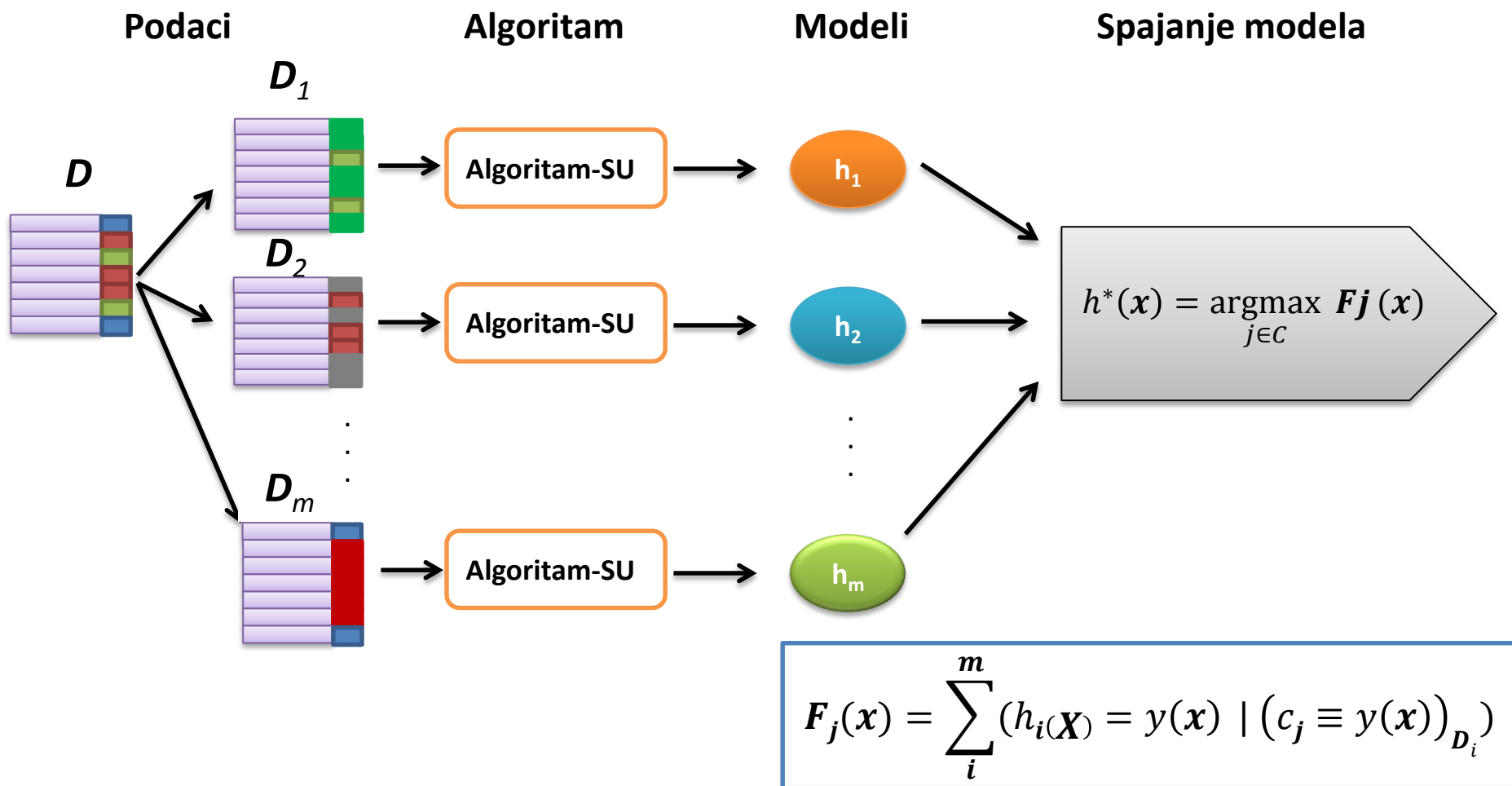
001 011

010 110

100 101



ECOC: Error Correcting Output Coding



ECOC: Error Correcting Output Coding

(Dietterich/Bakiri, 1995)

Ideja: Učenje klasifikatora $[h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_m(\mathbf{x})]$ kao kodnih-riječi

$M \gg \log_2(K)$ - redundancija ?!

Učenje

Za $i=1, M$

- a) (Slučajno) particioniranje K klasa u dva različita podskupa $\{A, B\}_i$
- b) Re-labeliranje primjera u dvije nove klase $\{A, B\}_i$
- c) Učenje $h_i(\mathbf{x})$ za klasifikaciju primjera $\{A, B\}_i$
- d) Ponavljanje

Klasifikacija novog primera

- a) Ako je $h_i(\mathbf{x})=A_i$, tada sve originalne klase u A_i dobiju 1 glas; odnosno ukoliko je $h_i(\mathbf{x})=B_i$ sve originalne klase u B_i dobiju 1 glas
- b) Konačno, klasa s najviše dobivenih glasova je predikcija ECOC ansambla

Kombiniranje predikcija u ansamblu

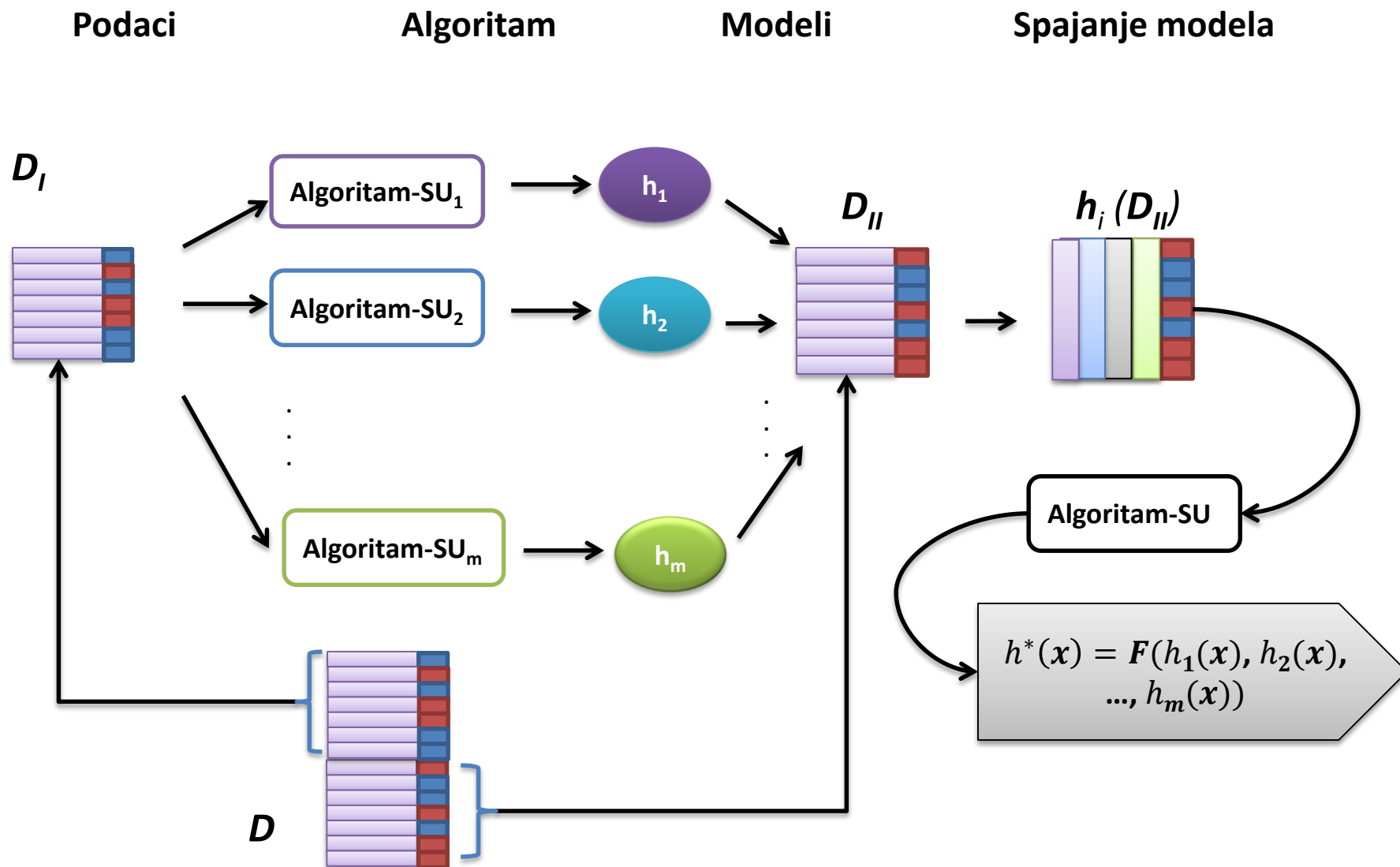
Glasanje

- Svaki član ansambla daje glas za jednu klasu (za ECOC je malo kompliciranije)
- Predviđanje klase – ona koja ima najveći broj glasova

Težinsko glasanje

- Težinska suma glasova članova ansambla
- Težine koje se daju članovima zavise o:
 - Pouzdanost koju pojedini klasifikator daje uz svoju predikciju (vjeroj. Pripadanja predviđenoj klasi)
 - O procjeni prosječne točnosti klasifikatora (npr. boosting)
- **Stacking**
 - Zašto ne učiti iz predviđanja članova ansambla ?

Stacking - Stacked generalization



Stacking; Stacked generalization (Wolpert -1992)

(? Stog modela; generalizacija preko stoga modela?)

= Učenje meta-modela nad predikcijama baznih modela

Učenje se odvija u dva nivoa:

- 1 Razdvoji skup za učenje D na dva dijela D_I i D_{II} (*slučajno* stratificirano uzorkovanje)
Na prvom dijelu T_I se “uči” nekoliko baznih algoritama L_i – za koje je poželjno da stvaraju što različitiije modele h_i
- 2 Nakon što su naučeni modeli baznih algoritama na D_I , ti se modeli iskoriste za predikcije na D_{II} => i stvara se novi skup primjera na bazi tih predikcija D'_{II}
- 3 Novi algoritam uči kombinirati predikcije modela (**meta-model**) na D'_{II}

Klasifikacija novog primjera

- 1 Bazni modeli prvo daju svoje predikcije
- 2 Meta-model koristi ove predikcije da bi napravio konačnu predikciju ansambla

Zašto kombiniranje modela dobro funkcionira (I)

Statistički pogled

- Uz konačni skup primjera za učenje – mnoge hipoteze tipično funkcioniraju približno jednako dobro
- Podsjetnik : Klasifikator prema Optimalnom Bayesovom principu

$$c(x_i) = \arg \max_{c_j \in C} \sum_{h_k \in H} P[c_j | h_k] \cdot P[h_k | D]$$

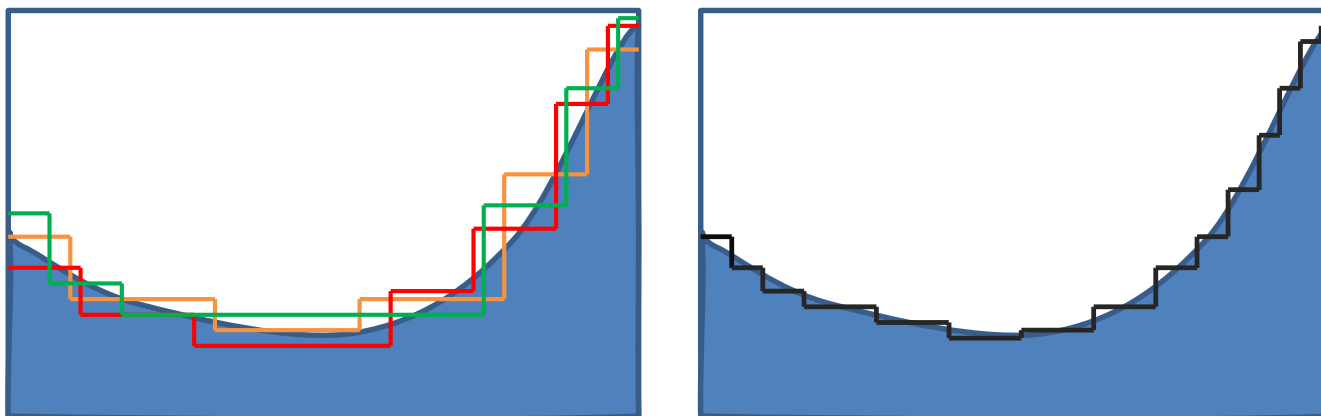
- Težinski usrednjeno glasanje svih hipoteza
- Težine – aposteriorne vjerojatnosti hipoteza
- Teorijski pokazano – najbolji mogući klasifikator!

Ansambli predstavljaju aproksimaciju Optimalnog Bayesovog klasifikatora !

Zašto kombiniranje modela dobro funkcionira (II)

Problem reprezentacije modela

- Optimalna funkcija cilja ne mora biti ni jedan individualni klasifikator – no može se bolje aproksimirati usrednjavanjem većeg broja individualnih modela
- Primjer – Stabla odlučivanja \Rightarrow slučajna šuma



Zašto kombiniranje modela dobro funkcionira (III)

Problem optimizacije

- Svi algoritmi pretražuju prostor hipoteza tražeći dovoljno dobru hipotezu
- Kako takvih može biti beskonačno mnogo – heuristika pretraživanja postaje ključna
- Algoritam pretraživanja može zapeti u lokalnom minimumu
- Jedna strategija za izbjegavanje lokalnih minimuma – ponavljanje pretraživanja uz randomizaciju (početne točke)
 - To vodi na => bagging !

Ansambli: sažetak

- Metode bazirane na kombiniranju više modela u jednu predikciju
- Poboljšavaju točnost u odnosu na individualne modele, jer reduciraju ili varijancu ili bias (ili oboje !)
- Bagging – redukcija “varijance”; efikasna za nestabilne, kompleksnije modele/hipoteze
 - “paralelno” stvaranje modela
 - Osnova su: repetitivno (bootstrap) uzorkovanje i usrednjavanje predikcija (regresija), odnosno većinsko glasanje (klasifikacija)
- Boosting – redukcija pristranosti (bias-a), ali i povećanje margine
 - “sekvencijalno” stvaranje modela
 - fokus na teže dijelove/primjere; daje težinu pojedinim modelima prema njihovoj točnosti
 - osjetljivost na „outliere”

Ansambli: sažetak

- S obzirom da zahtijevaju učenje većeg broja modela
 - vremenski su i memorijski zahtjevnije metode
- Gotovo na svim realnim problemima, kod kojih je važna prediktivna točnost - najbolje rezultate postižu ansambli (Kaggle => XGBoost) !

Literatura - Ansambli

- The Elements of Statistical Learning
Hastie, Tibshirani, Friedman (ch. 15)
- AI – Modern approach
Russel & Norvig (ch 18.4)
- T. Dieterich: Ensemble Methods in Machine Learning
Lecture Notes in Computer Science, Vol. 1857 (2000), pp. 1-15
- Bagging (L. Breiman)
Random forests: <http://stat-www.berkeley.edu/users/breiman/rf.html>
Bolje: R -package (randomForest); PARF => IRB
- Boosting (www.boosting.org)
Y. Freund, Robert E. Schapire: Experiments with a new boosting algorithm.
In: Thirteenth International Conference on Machine Learning, San Francisco, 148-156, 1996