Strojno učenje

Ansambli modela

Tomislav Šmuc

Literatura

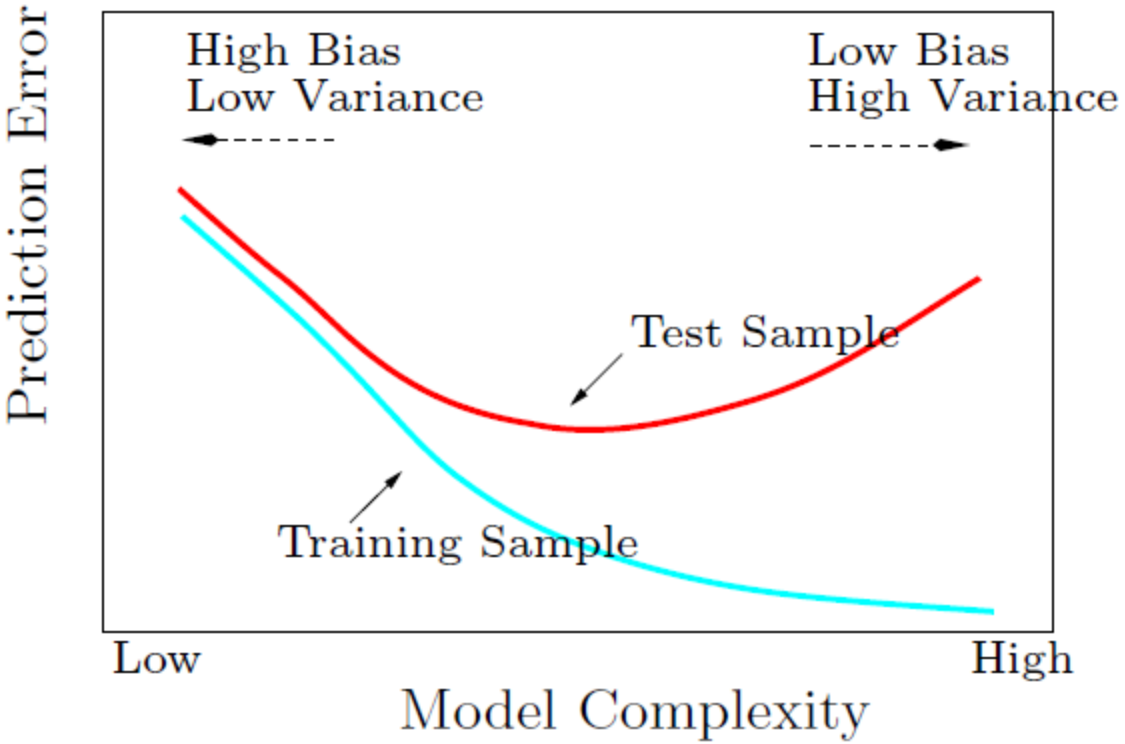
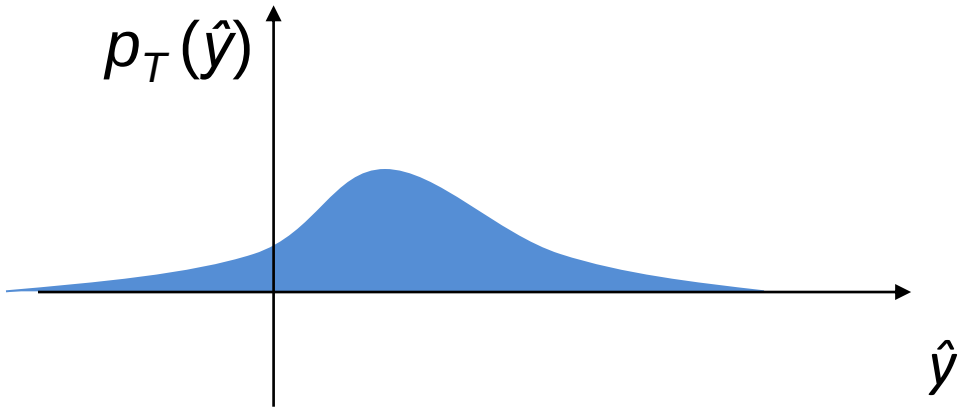


FIGURE 2.11. *Test and training error as a function of model complexity.*

Dekompozicija prediktivne pogreške: Pristranost i varijanca modela

Skup za učenje je T slučajno uzorkovan

=> predikcija $\hat{y}$ slučajna varijabla



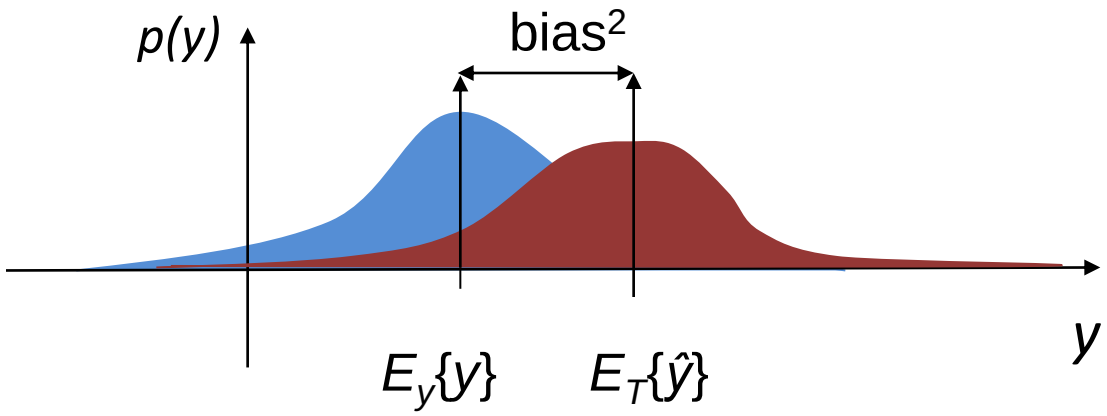
Očekivane vrijednosti

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

$$E_T[(y - \hat{y})^2] = \underbrace{(E_T[\hat{y}] - y)^2}_{a} + \underbrace{E_T[(\hat{y} - E_T[\hat{y}])^2]}_{b} + \underbrace{E[\varepsilon \mid x]}_{c}$$

- Pristranost/Bias: sistematska greška na točki
  x - prosjek preko "svih" skupova za učenje T veličine N
- Varijanca: Varijacija greške oko prosječne vrijednosti
- Šum: Greška u određivanju stvarnih vrijednosti y(x)

Dekompozicija prediktivne pogreške: Pristranost i varijanca modela

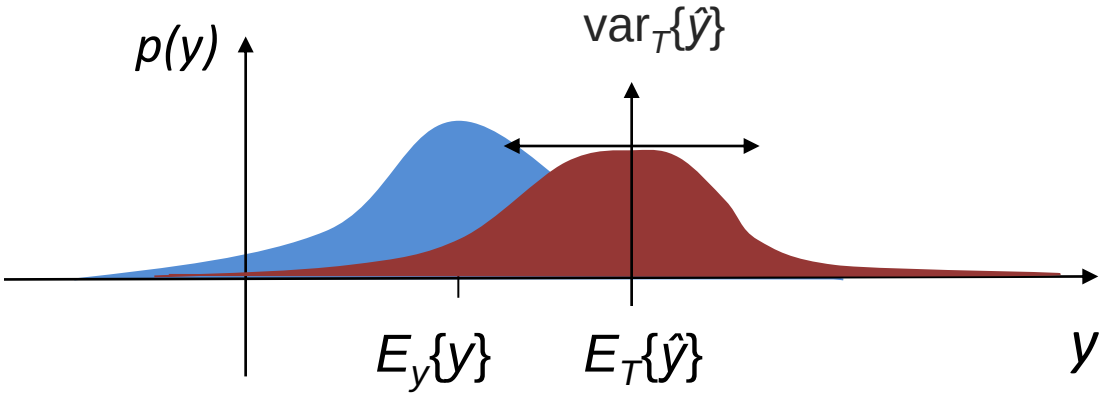


$$(E_y\{y\}-E_T\{\hat{y}\})^2$$

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

bias² = greška između stvarne vrijednosti i prosječnog estimacijskog modela

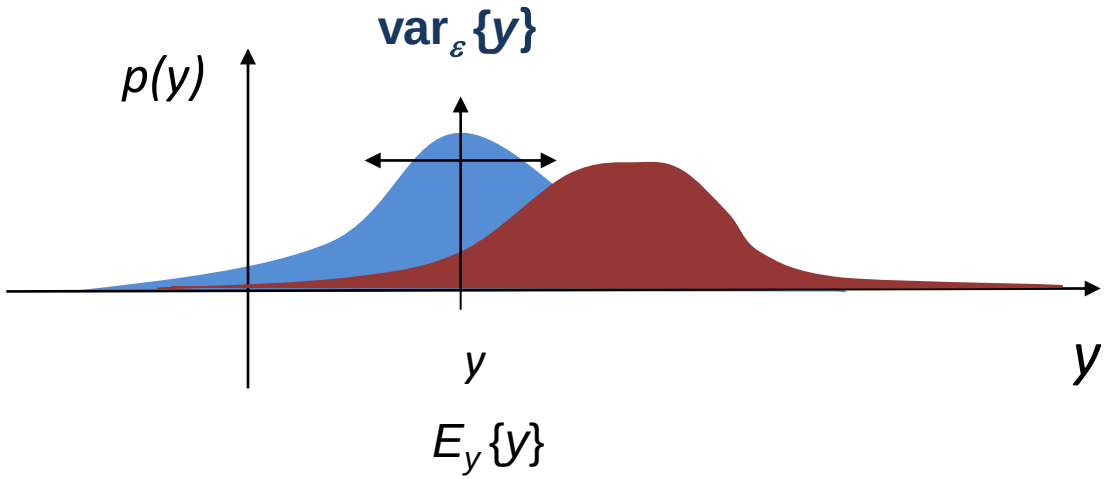
Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



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

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

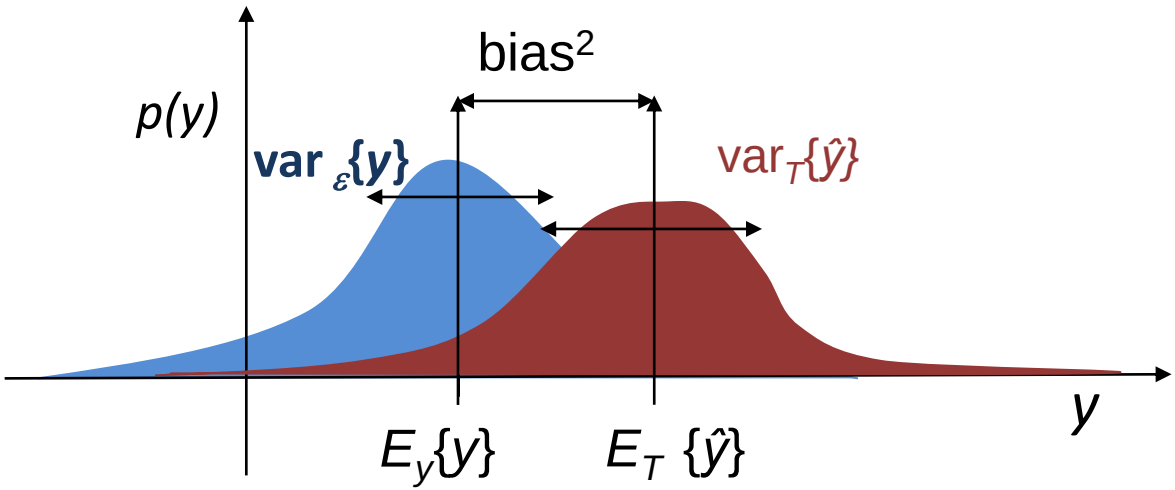
Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



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

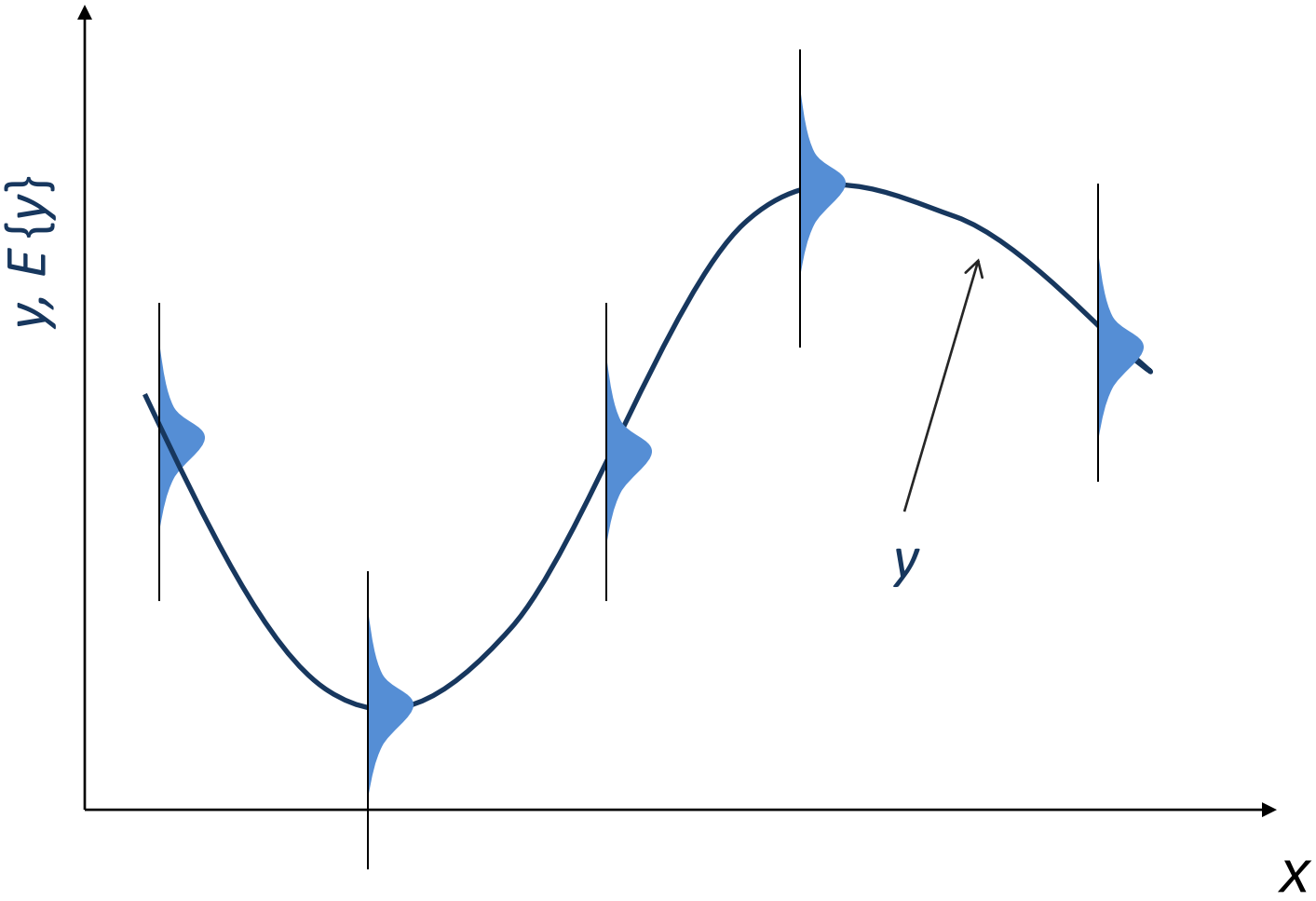
rezidualna greška = minimalna greška koju možemo dostići

Dekompozicija prediktivne pogreške: Pristranost i varijanca modela

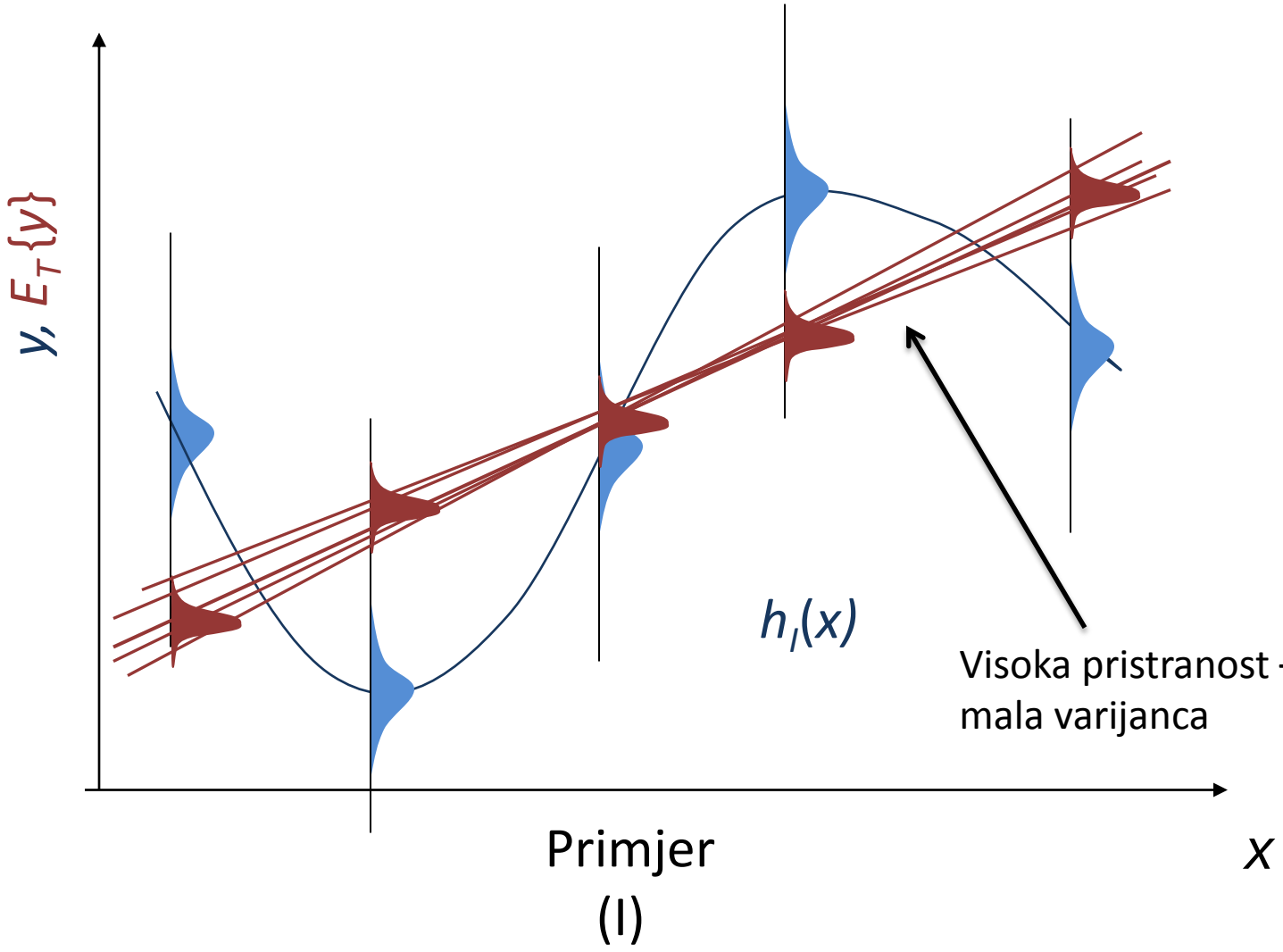


$$E = \text{var}_{\varepsilon}\{y\} + \text{bias}^2 + \text{var}_{T}\{y\}$$

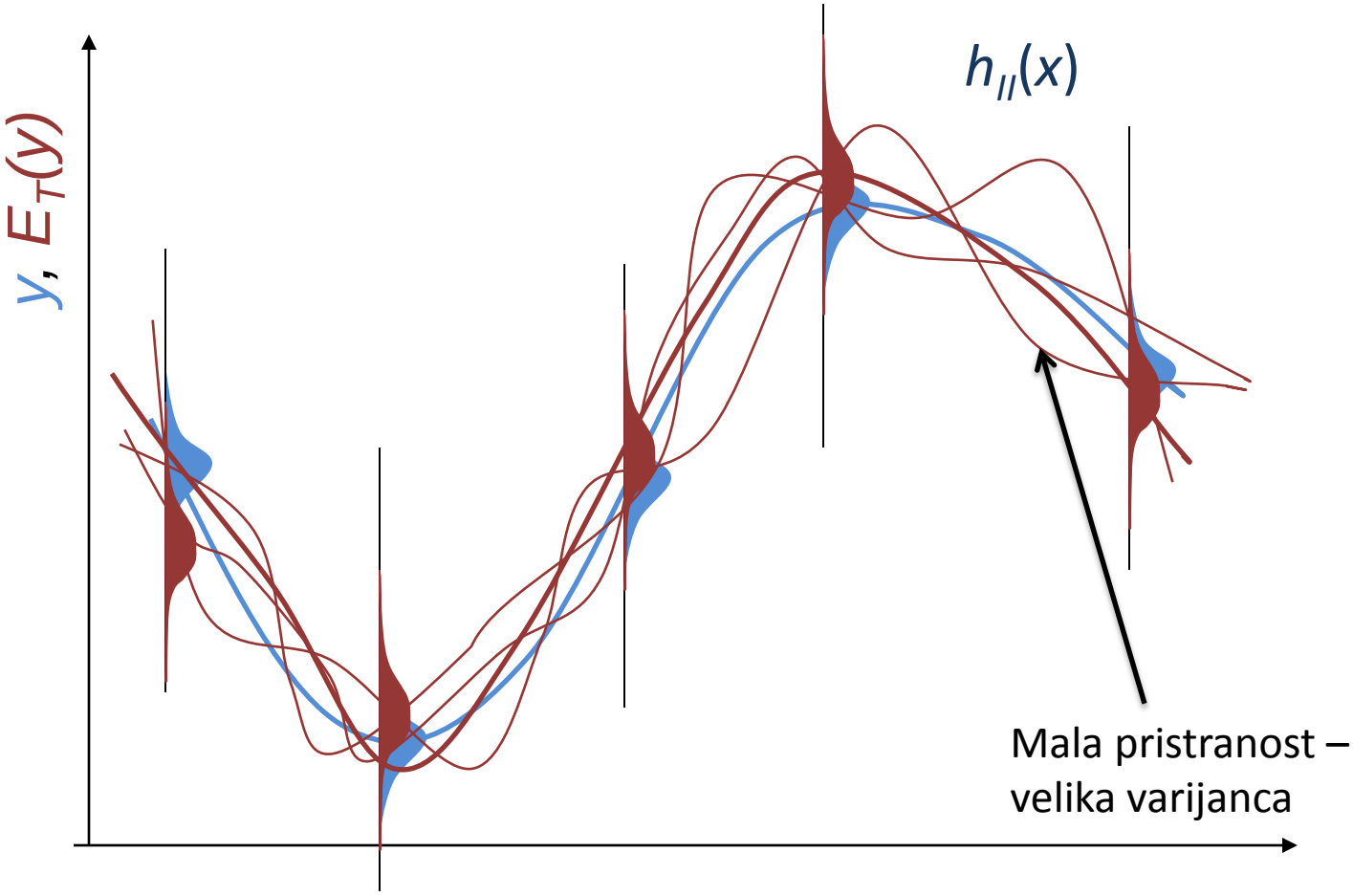
Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



Dekompozicija prediktivne pogreške: Pristranost i varijanca modela

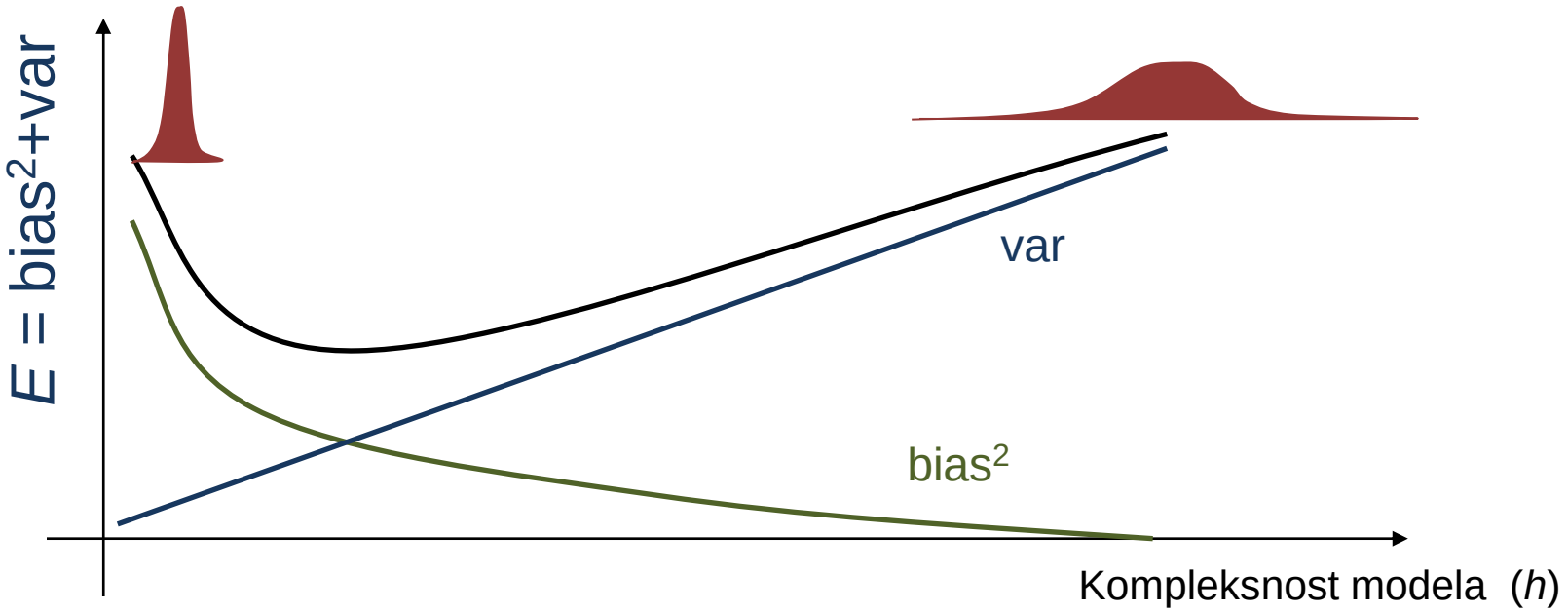


Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



Primjer
II

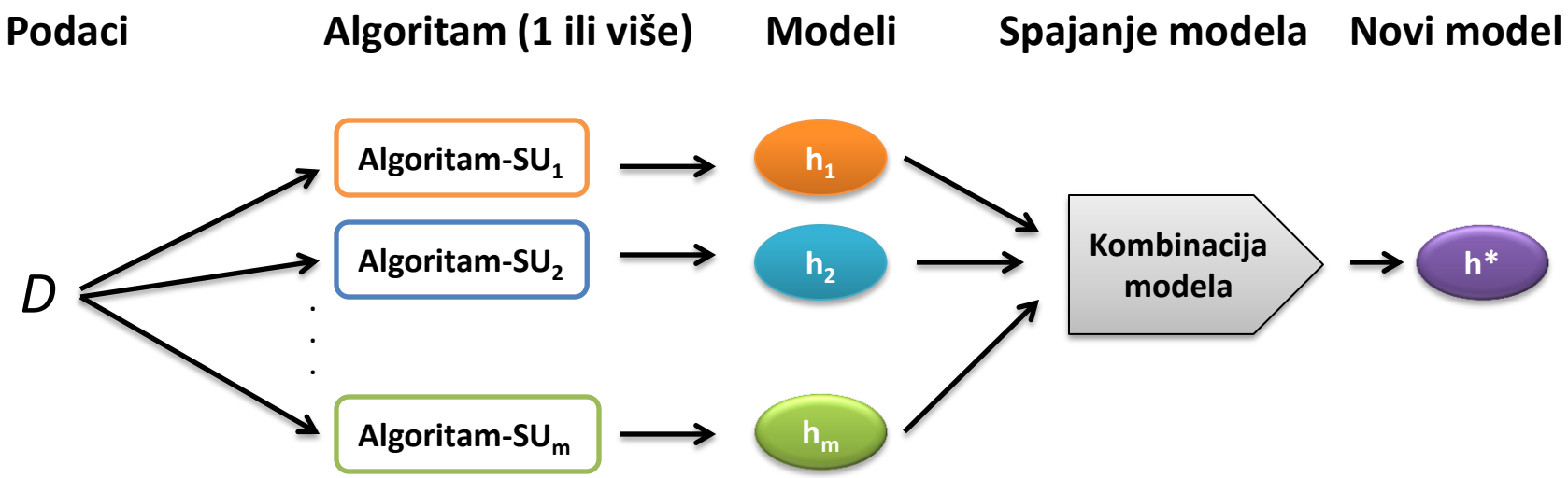
Dekompozicija prediktivne pogreške: Pristranost i varijanca modela



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

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 točnost *p*, tada se može pokazati da vrijedi:

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

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

Gdje je:

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

$\lfloor L/2 \rfloor$ => najveći cijeli broj $\leq L/2$

	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,…h_m
   - Različiti algoritmi ili različiti podaci

2. Kako ih kombinirati u procesu odlučivanja

$$h^* = F(h_{1(x)}, h_{2(x)}, \dots, h_{m(x)})$$

$F(\boldsymbol{h}) -$ 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, MART
  - Ovim pristupom smanjuje se primarno pristranost (bias), no kasnija istraživanja pokazala su da boosting smanjuje i varijancu modela

Bagging (**B**ootsrap **AGG**regat**ING**)

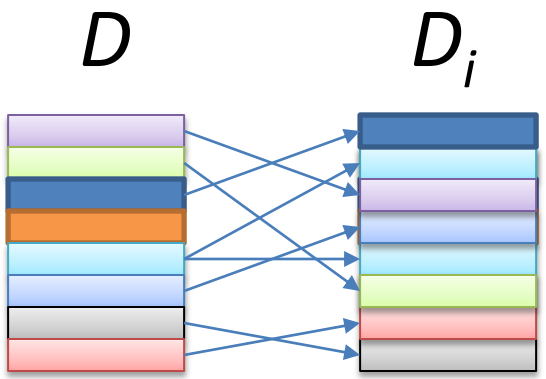
- Napraviti (velik) broj modela koristeći bootstrap replike podataka
- Spojiti u jedan zajednički (bagged) model – odnosno predikciju
- Svi modeli “glasaju”:
  - U slučaju klasifikacije – pravilo većine
  - U slučaju regresije – prosjek svih predikcija

Bootstraping

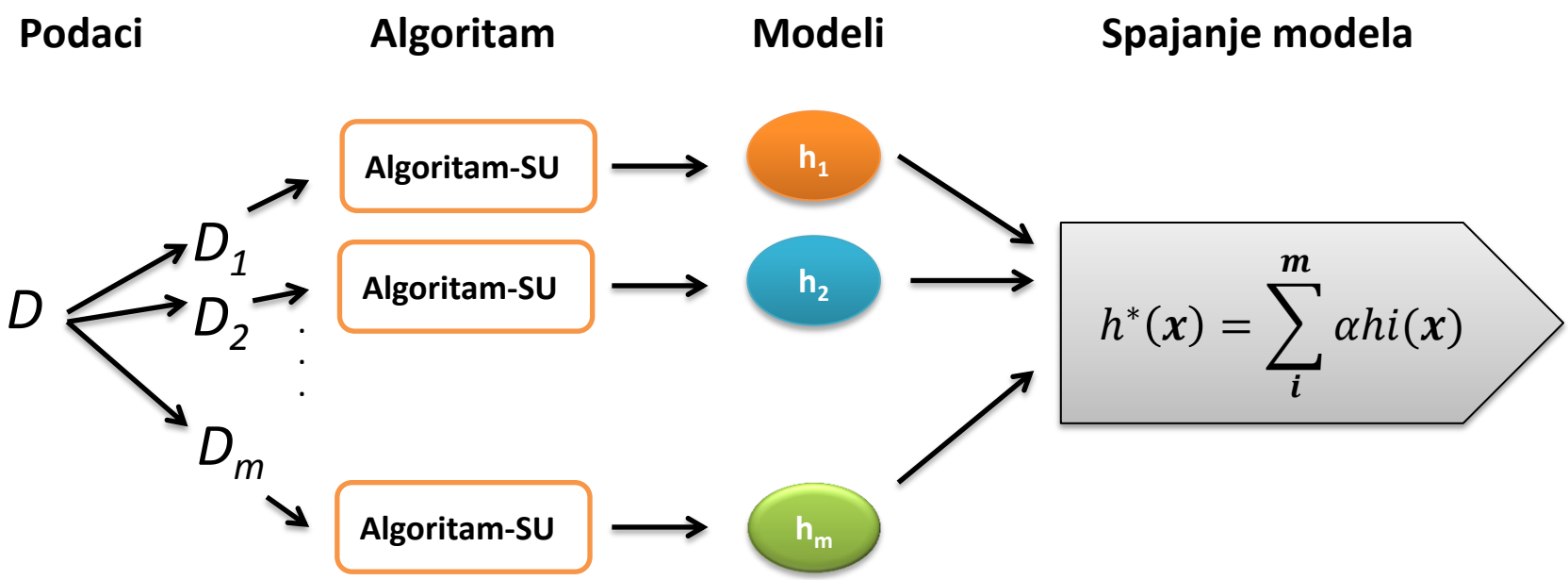
- metoda ponavljanog uzorkovanja iz skupa podataka (en resampling statistical method)
- Korištenje skupa podataka za učenje da se naprave slučajni skupovi – replike originalnog skupa podataka, radi dobivanja informacija o nekim statistikama skupa podataka (bias, variance)

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)

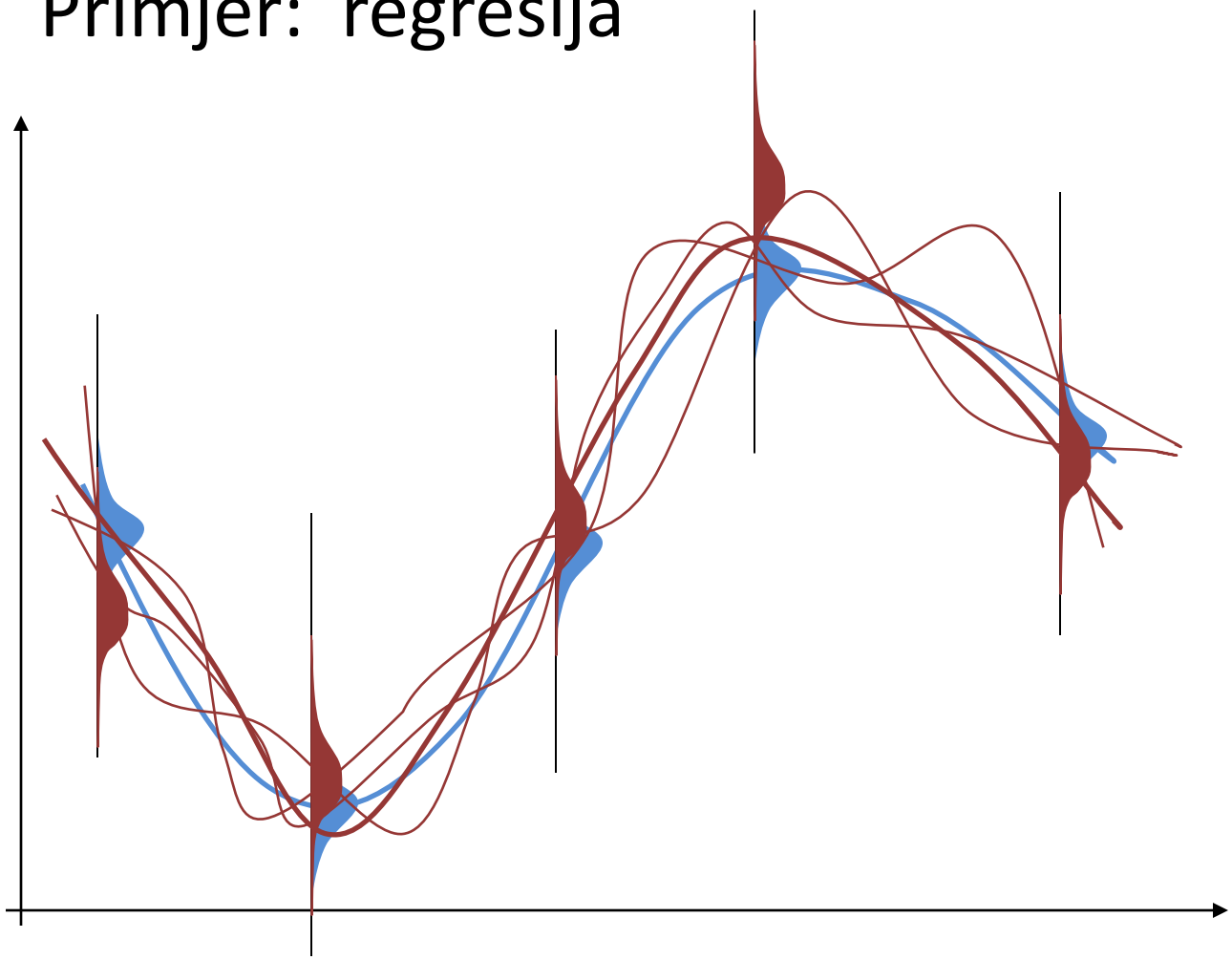


Bagging

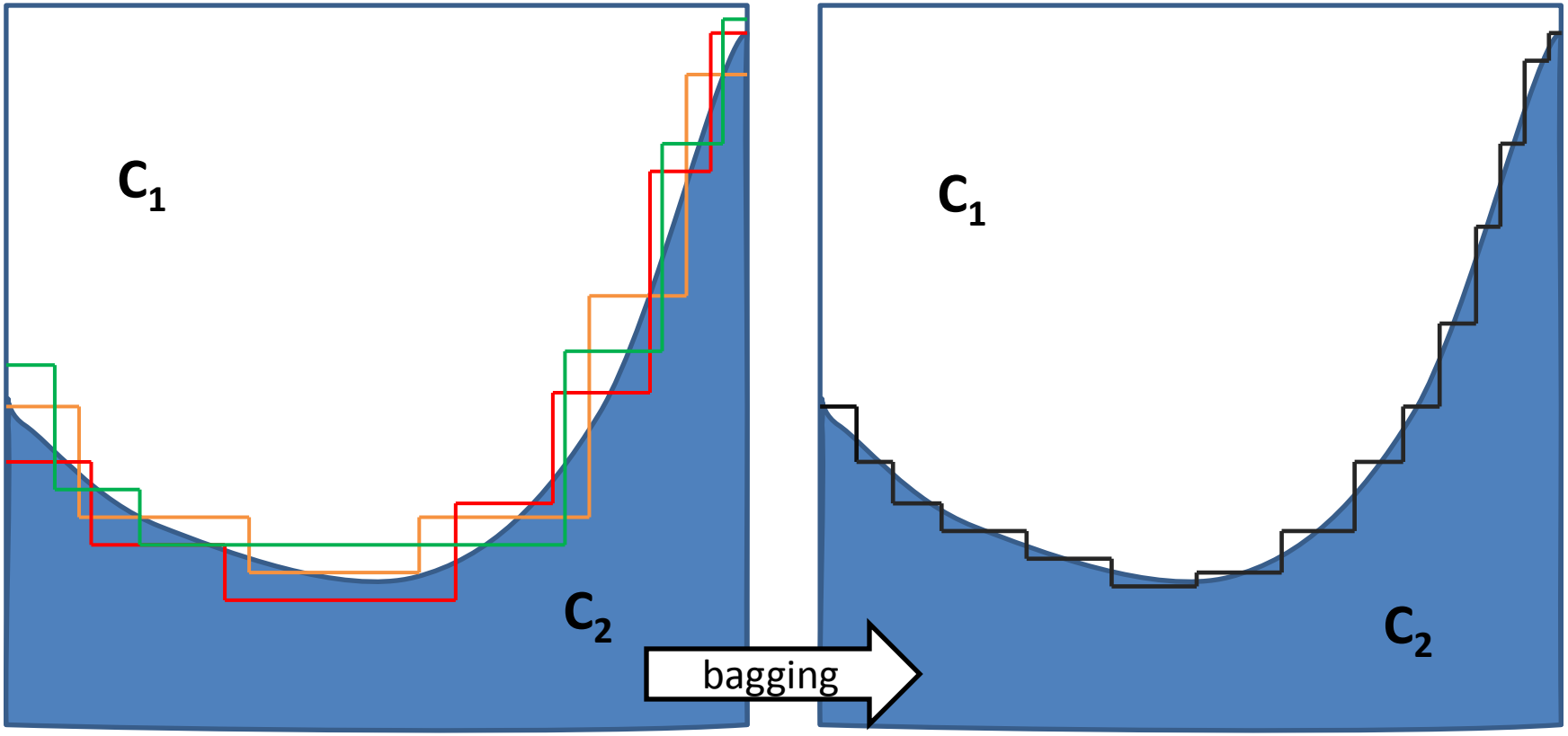


D_i - bootstrap replika podataka
h_i - klasifikator baziran na D_i
α_i = 1/m

Primjer: regresija



klasifikacijski primjer: stabla odlučivanja



Ansambli: bagging

Očekivana kv. greška *jednog* modela (regresija)

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

$$E[\big(h_i(\boldsymbol{x}) - y(\boldsymbol{x})\big)^2] = E[(\varepsilon_{i(\boldsymbol{x})})^2]$$

Prosječna kv. greška *m* modela:

$$\bar{E}_m = \frac{1}{m}\sum_i E\,[(\varepsilon_{i(\boldsymbol{x})})^2]$$

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

…. uz pretpostavku da su greške modela $\varepsilon_{i(\mathbf{x})}$ takve da vrijedi
$E[\varepsilon_{i(\boldsymbol{x})}] = 0$ -- srednja vrijednost ~ 0
$E[\varepsilon_{i(\boldsymbol{x})}\varepsilon_{j(\boldsymbol{x})}] = 0$ – medjusobno su nekorelirane

$$\bar{E}_{h^*} = E\left[\left(y(\boldsymbol{x}) - \tfrac{1}{m}\textstyle\sum_i h_i(\boldsymbol{x})\right)^2\right] = E\left[\left(\tfrac{1}{m}\textstyle\sum_i \varepsilon_{i(\boldsymbol{x})}\right)^2\right] \;=\; \tfrac{1}{m}\bar{E}_m$$

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

Bagging - Pristranost i varijanca

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

Varijanca: $\sum_i (h^*(\boldsymbol{x}) - h_i(\boldsymbol{x}))^2/(m-1)$ => 0, za m>>

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

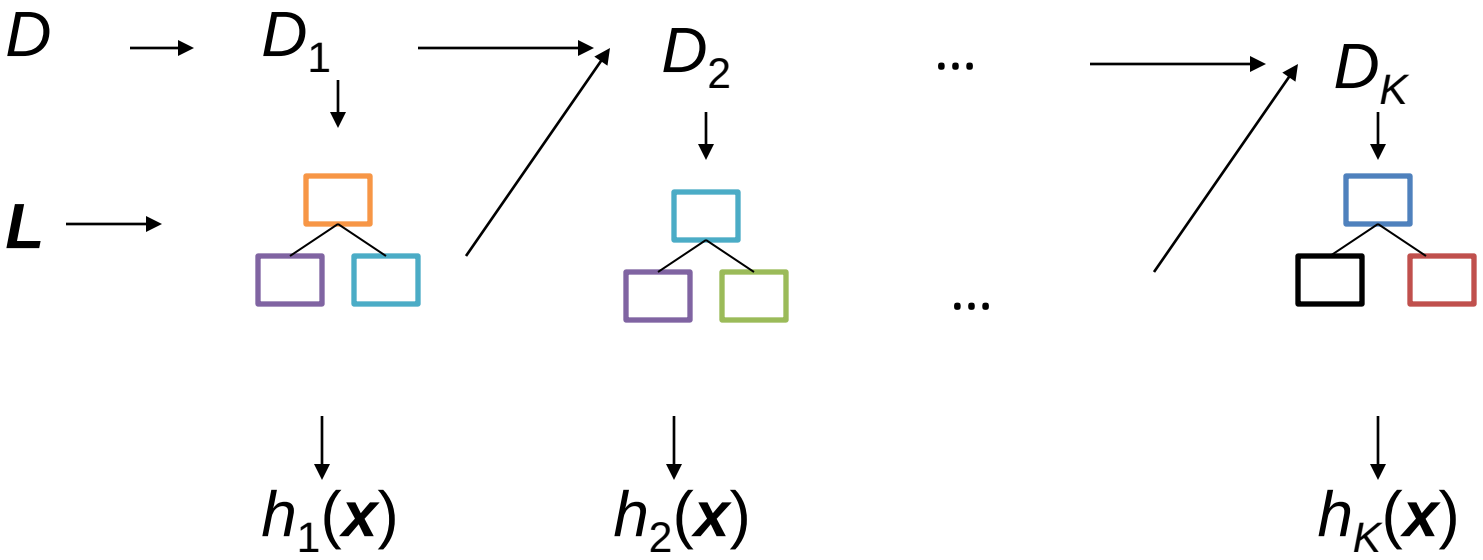
- Kombinira bagging sa slučajnim odabirom podskupa varijabli/atributa (perturbacija modela)
  - Gradi stabla odlučivanja iz bootstrap uzorka skupa za učenje
  - Umjesto izabiranja najboljeg atributa za split – između svih atributa – izabire između *k* slučajno odabranih atributa

  (= bagging , kad je *k* jednak broju atributa – tipično za RF k=$\sqrt{n}$)
- Balans bias/varijanca korištenjem k:
  - Što je manji k – veća je redukcija varijance, ali je i veći bias

Boosting metode – "jačanje" slabih modela

- Motivacija:
  - kombiniranje outputa "slabih" modela da bi se napravio točniji ansambl modela.
- "Slabi" modeli:
  - modeli s visokim bias-om (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

Ansambli: boosting



Klasifikacija: $h(\boldsymbol{x})$ = večina od $\{h_1(\boldsymbol{x}),\ldots,h_K(\boldsymbol{x})\}$
uz težine $\{\beta_1,\beta_2,\ldots,\beta_K\}$

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

AdaBoost (Adaptive Boosting) algoritam

- (Freund & Schapire) 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

"Slabi" algoritam SU ***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: $w_i^1 = 1/N, \text{ za } i = 1,\ldots,N$

Za *t <- 1,2,....,T* **radi**

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

Vrati klasifikator ***F:***

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

AdaBoost algoritam

Izraz kojim mijenjamo težine primjera u iteraciji t+1:

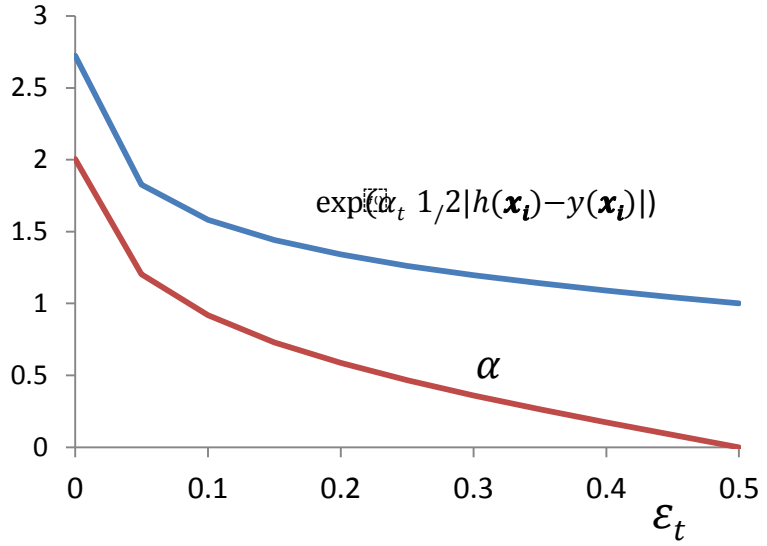
$$w_i^{t+1} = w_i^t \exp(\alpha_t \frac{1}{2}|h_t(x_i) - y_i|), \text{ za } i = 1,2,\ldots,N$$

Gdje je α_t – težinski faktor modela u iteraciji *t:*

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

a ε_t prosječna greška u iteraciji *t:*

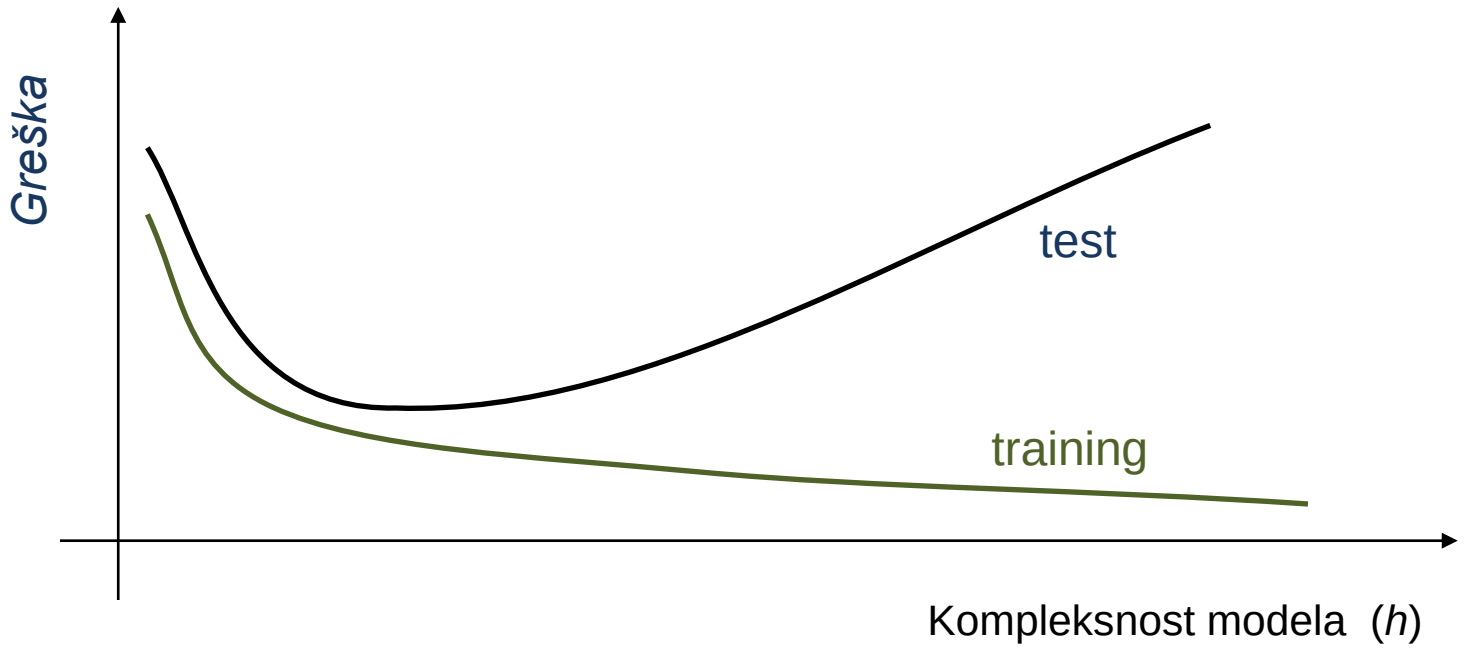
$$\varepsilon_t = \sum_{i=1}^{t} p_i^t \frac{1}{2}|h_t(x_i) - y_i|$$



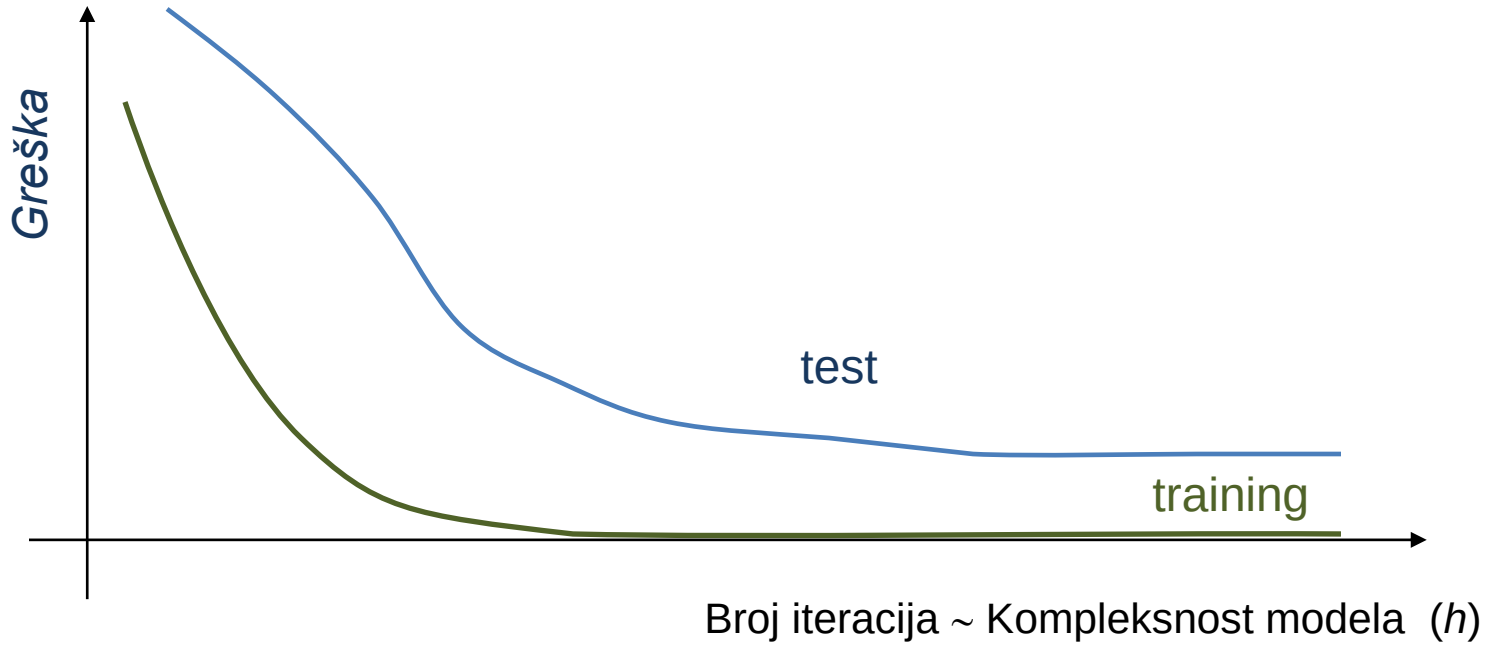
Zašto boosting radi dobro ?

- Kombinira modele koji imaju visoki bias (jednostavni), tako da se 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]):
  - $h^*(\boldsymbol{x}) = y(\boldsymbol{x})$
  - Klasifikacija primjera je korektna ako je$h^*(\boldsymbol{x}) = y(\boldsymbol{x})$, odnosno pogrešna ako $h^*(\boldsymbol{x}) \neq y(\boldsymbol{x})$

  No – u boosting algoritmu **možemo mjeriti pouzdanost** klasifikacije !
  - $h^*(\boldsymbol{x})$ - je težinski zbroj glasova pojedinih (slabih) klasifikatora !
  - **Mjera pouzdanosti ~ 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 $h^*(\boldsymbol{x})$ naizgled složeniiji, margine primjera se povećavaju, dakle $h^*(\boldsymbol{x})$ na neki način postaje robustniji-jednostavniji(!) i smanjuje se njegova greška na novim (testnim) primjerima !
- Boosting algoritmi rade na povećavanju margine
- Intuitivno – povećanje margine vodi 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 $h^*(\boldsymbol{x})$!

Ansambli bazirani na manipuliranju ciljnom varijablom

Problem sa y=1,2,..,K – klasa

Mogući pristupi:

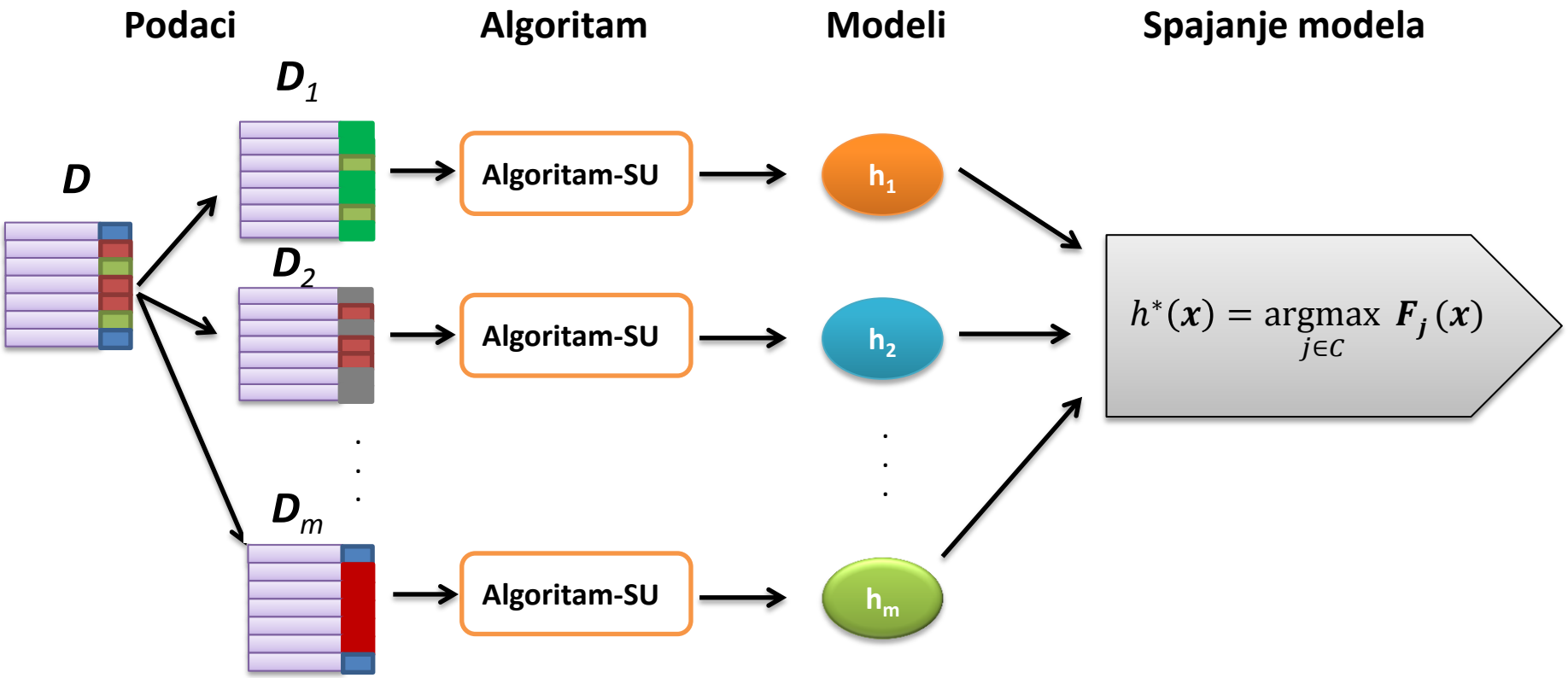
1. Učenje K binarnih klasifikatora
   - y=1 <-> (y=2,3,4,…,K)
   - y=2 <-> (y=1,3,4,..K)
   - ….

   Odluka – većinskim glasanjem

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
   - …

➔ ECOC – Error Corecting Output Coding

ECOC: Error Correcting Output Coding



$$F_j(x) = \sum_{i}^{m} (h_{i(X)} = y(x) \mid (c_j \equiv y(x))_{D_i})$$

ECOC: Error Correcting Output Coding

(Dietterich/Bakiri, 1995)

Ideja:

Učenje klasifikatora $[h_1(\boldsymbol{x}), h_2(\boldsymbol{x}),\ldots, h_m(\boldsymbol{x})]$ kao kodnih-riječi

M > $\log_2$(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) Ponavljaj

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

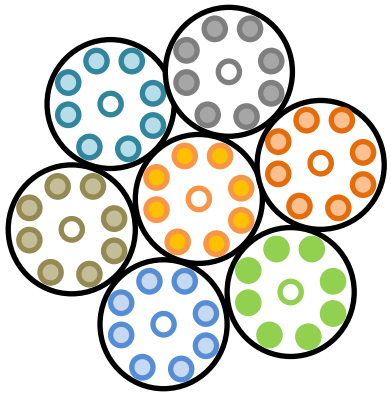
ECOC: Error Correcting Output Coding

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

Broj kodova >> broj poruka (klasa)

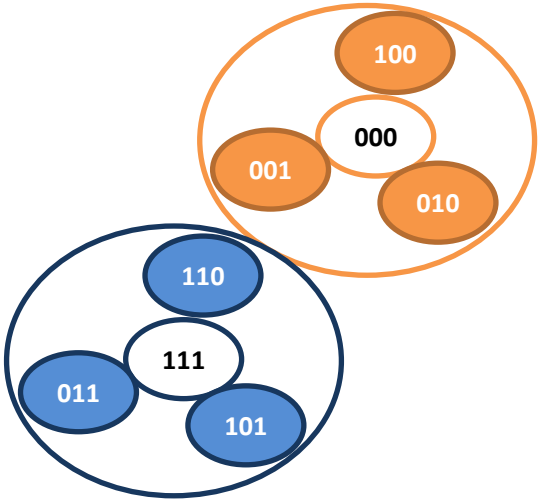
Kod dekodiranja klasa se odredjuje prema najbližoj kodnoj-riječi

- 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

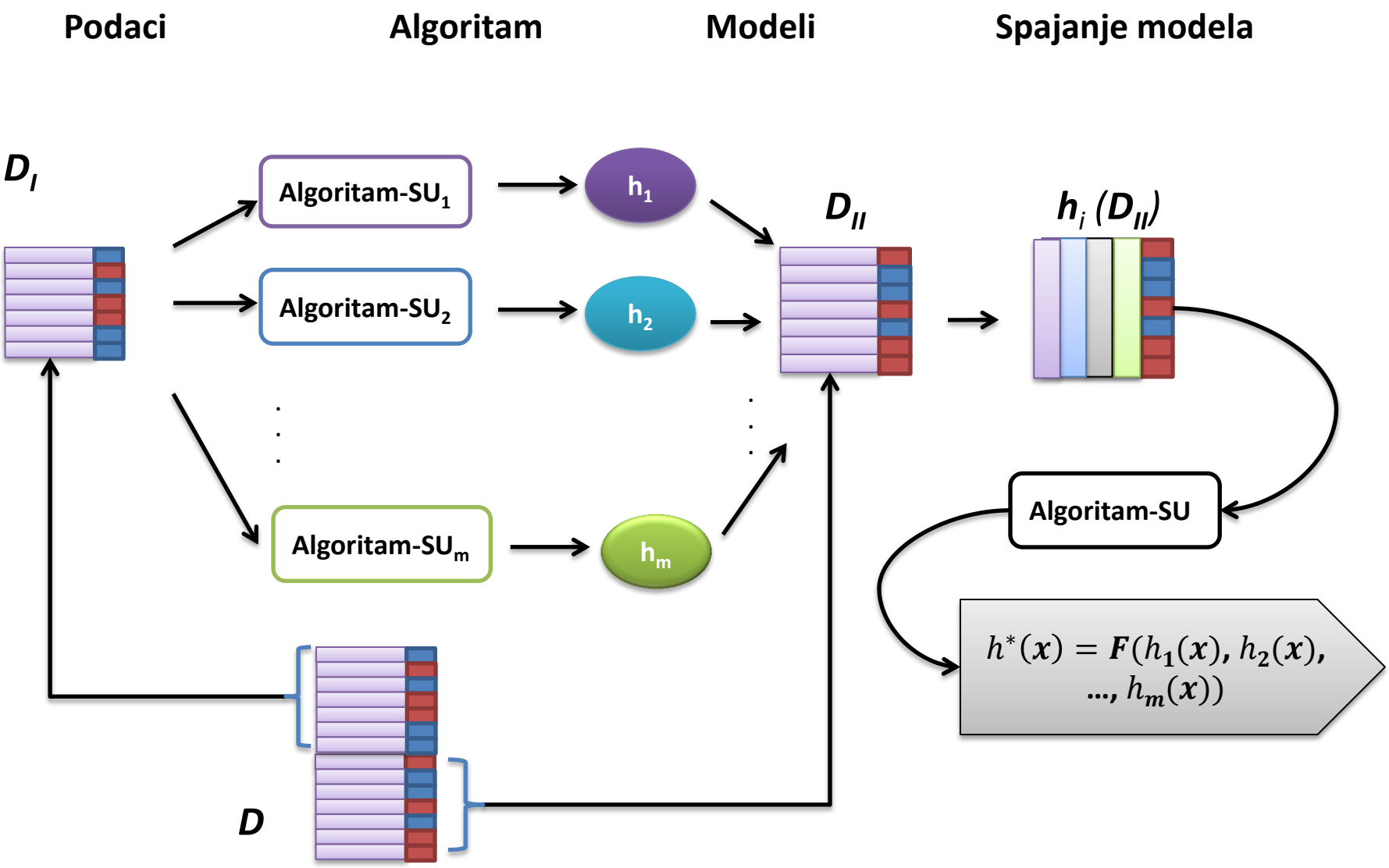


Klasa – kodna riječ

Buffer zona



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 različitije 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!

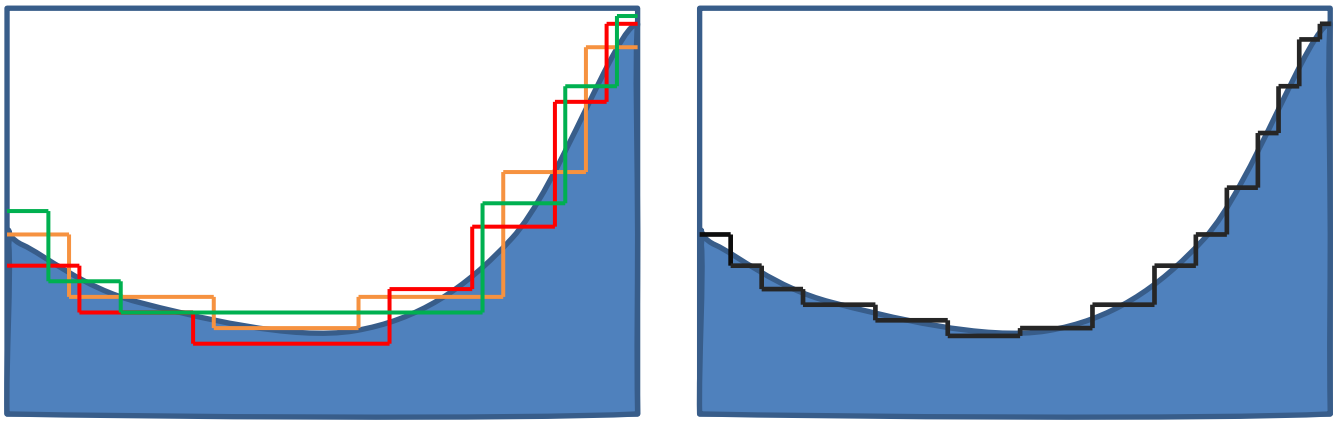
Ansambli predstavljaju aproksimaciju Optimalnog Bayesovog klasifikatora !

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 –=> slučajna šuma



Zašto kombiniranje modela dobro funkcionira (III)

Problem optimizacije

- Svi algoritmi pretražuju ustvari 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/sliči 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

Ansambli: sažetak

- S obzirom da zahtijevaju učenje većeg broja modela
  - vremenski su i memorijski zahtjevne metode

- Gotovo na svim realnim problemima, kod kojih je važna prediktivna točnost - najbolje rezultate postižu ansambli !

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