UČENJE IZ PODATKOVNIH TOKOV

Aljaž Osojnik

aljaz.osojnik@ijs.si

Odsek za tehnologije znanja,

Inštitut Jožef Štefan

POVZETEK

- Kaj je podatkovni tok?
- Lastnosti podatkovnih tokov & omejitve pri učenju
- Ilustrativen primer
- Formalizacija metod za učenje na podatkovnih tokovih
- Računanje statistik
- Vrednotenje na podatkovnih tokovih
- Vrste sprememb
- Učenje dreves na podatkovnih tokovih in FIMT-DD

KAJ JE PODATKOVNI TOK?

Podatkovni tok (an. *data stream*) je podatkovni vir, kjer vsi podatki niso na voljo od začetka, ampak postanejo na voljo sčasoma, običajno po en primer na enkrat.

PODATKOVNI TOK

		Opisni prostor			Ciljni prostor
…		…			…
Primer n+5	1	TRUE	0.49	0.69	0.45
Primer n	1	TRUE	0.49	0.69	0.45
Primer n+1	4	FALSE	0.08	0.07	0.12
Primer n+2	6	FALSE	0.08	0.07	1.54
Primer n+3	8	TRUE	0.00	1.00	3.12
…		…			…

LASTNOSTI PODATKOVNIH TOKOV

- Podatki prihajajo z visoko frekvenco
- Odziv na podatke je pričakovan v kratkem času (an. *real-time*)
- Podatkov je lahko poljubno veliko
- Postopek, ki ustvarja podatke, se lahko spremeni

Učenje iz podatkovnih tokov pogosto uvrščamo med "Big Data" probleme

OMEJITVE UČENJA IZ PODATKOVNIH TOKOV

- Ker je podatkov veliko, vseh ne moremo shraniti – praviloma vsak podatkovni primer obravnavamo enkrat (*an. one-pass*) nato pa ga zavržemo ali arhiviramo
- Pri učenju iz podatkovnih tokov nimamo nadzora nad vrstnim redom prihoda primerov
- Zaradi visokih hitrosti prihoda primerov se želimo učiti hitro

OMEJITVE UČENJA IZ PODATKOVNIH TOKOV (2)

- Na podatkovnih tokovih se pogosto učimo na napravah z malo spomina, zato želimo nizko porabo spomina
- Podatki se lahko spremenijo (*an. concept drift*), za to moramo spremembe zaznati ter se nanje prilagoditi

OMEJITVE UČENJA IZ PODATKOVNIH TOKOV (POVZETEK)

- Vsak primer obravnavamo le enkrat
- Nimamo nadzora nad vrstnim redom primerov
- Učenje mora potekati hitro
- Učenje mora porabiti čim manj spomina
- Spremembe v podatkih moramo zaznati in se nanje prilagoditi

ILUSTRATIVEN PRIMER

- Imamo podatkovni tok, katerega primeri so prodajni artikli
- Na voljo imamo le par KB spomina
- Primerov je toliko, da jih ne moremo shraniti v spomin (npr. nekaj TB)
- Zanima nas najpogostejših k artiklov

PRIMER: POGOSTI ARTIKLI

- Brez dodatnih predpostavk bomo težko shajali
- Predpostavimo: pogostost artiklov je neenakomerna
  - Nekaj artiklov je izjemno pogostih
  - Preostali artikli so zelo redki

PRIMER: POGOSTI ARTIKLI – PRISTOP

- Vzdrževali bomo m ($m > k$) števcev
  - Toliko kot jih lahko shranimo v spomin
  - Vsakemu števcu lahko tudi pripišemo enega izmed artiklov
- Na začetku imajo vsi števci vrednost 0 in nimajo pripisanih artiklov

PRIMER: POGOSTI ARTIKLI – PRISTOP (2)

- Ko dobimo nov primer (artikel):
  - Če temu artiklu pripada eden izmed števcev, ga povečamo za 1
  - Sicer: če obstaja števec, ki ima vrednost 0, to vrednost ponastavimo na 1 in števcu pripišemo nov artikel
  - Sicer: (vsi števci so zasedeni) vse števce zmanjšamo za 1
- Deluje kadar:
  - $m > k$
  - $m \ll N$, kjer je N število različnih artiklov

PRIMER: POGOSTI ARTIKLI – ZAKLJUČKI

- Definirali smo **Misra-Gries povzetek** [Misra & Gries, 1982]
- Model lahko v vsakem trenutku uporabimo za izračun k najpogostejših artiklov
- Model se spreminja
- Model ni popolnoma natančen, računa približke

OBIČAJNO UČENJE

- Pri običajnem učenju se metode učijo iz (končnih) podatkovnih naborov
- Vsaka metoda se nauči modelov določene oblike
  - Primer: CART/ID3 naučita odločitvena drevesa
- Metoda torej "sprejme" podatkovni nabor in "vrne" model

OBIČAJNO UČENJE (FORMALNO)

- Naj bo $\mathbb{D}$ prostor podatkovnih naborov
- Naj bo $\mathbb{H}_M$ prostor hipotez metode M
  - Primer: hipoteze CART/ID3 so odločitvena drevesa
  - $\mathbb{H}_{CART}$ je torej množica vseh odločitvenih dreves
- Metoda M je tedaj preslikava

$$M : \mathbb{D} \rightarrow \mathbb{H}_M$$

SPROTNO UČENJE (FORMALNO)

- Pri sprotnem učenju to ni dovolj, saj je (vmesnih) modelov več
  - Ko pride nov primer, se model spremeni
- Formalno metodo za sprotno učenje M podamo kot par (h_0, u), kjer:
  - h_0 je **začetna hipoteza** (model)
  - $u : \mathbb{H}_M \times X \rightarrow \mathbb{H}_M$ ali $u : \mathbb{H}_M \times \mathbb{B}(X) \rightarrow \mathbb{H}_M$ je **posodobitveni operator**
- X je prostor primerov, $\mathbb{B}(X)$ je prostor "vreč" primerov (množic s ponovitvami)

ZAČETNA HIPOTEZA h_0

- Začetna hipoteza $h_0 \in H_M$ je "nenaučen" model
- **Misra-Gries:** začetna hipoteza – m števcev z vrednostmi 0 in brez pripisanih artiklov
- Pri odločitvenih drevesih je začetna hipoteza praviloma prazen list
- Pri nevronskih mrežah je začetna hipoteza naključno inicializirana mreža

POSODOBITVENI OPERATOR u

- Posodobitveni operator sprejme trenutni model in nov(e) primer(e) in vrne nov model
- Posodobitveni operator lahko sprejme:
  - **En primer** (model se posodobi z vsakim prihajajočim primerom – an. instance-incremental)
  - **Več primerov** (model se posodobi, ko se nabere več primerov – an. batch-incremental)
- Model po enem primeru: $h_1 = u(h_0, x_1)$, po dveh: $h_2 = u(h_1, x_2)$, itd.
- **Misra-Gries:** posodobitveni operator popravlja števce

POSODABLJANJE MODELA S PRIMEROM

- V zelo grobem posodobitveni operator deluje na dva načina:
  - **Inkrementalno:** primer ali vrednosti njihovih atributov se shranijo
  - **Dekrementalno:** primer ali vrednosti so izbrisane, “pozabljanje”
- Pri Misra-Gries: kadar števcem prištevamo vrednosti se učimo inkrementalno, kadar odštevamo pozabljamo

NAČINI POSODOBITEV MODELOV Z NOVIMI PRIMERI

- Nove primere lahko metoda uporabi na več načinov
- Nevronske mreže lahko vsak primer direktno uporabijo za posodobitev uteži s pomočjo vzvratnega razširjanja napake (an. *backpropagation*)
- Drevesa čakajo, da se nabere podpore za delitev lista
- Modelna drevesa primere uporabljajo na oba načina:
  - za delitev listov čakajo na statistične dokaze
  - modeli v listih se posodabljajo z vsakim primerom

PRIBLIŽNO UČENJE

- **$\boldsymbol{\varepsilon}$-približno učenje:** izračunane vrednosti so znotraj ε-okolice prave vrednosti

$$1 - \varepsilon < \frac{izračunana}{prava} < 1 + \varepsilon$$

- **$(\boldsymbol{\varepsilon}, \boldsymbol{\delta})$-približno učenje:** izračunane vrednosti so z verjetnostjo $1 - \delta$ v ε-okolici prave vrednosti

$$\Pr(|prava - izračunana| \geq \varepsilon) < \delta$$

Takem učenju pravimo tudi verjetno približno pravilno učenje (*an. probably approximately correct*)

RAČUNANJE STATISTIK NA PODATKOVNIH TOKOVIH

RAČUNANJE STATISTIK

- Običajno učenje: za računanje statistik so na voljo vsi primeri
- Primer: za računanje povprečja si ogledamo vrednosti vseh primerov, jih seštejemo in delimo s številom primerov
- Ko se učimo iz podatkovne tokove, vseh primerov nimamo na voljo hkrati
- Kako računamo statistike koračno?

KORAČNO RAČUNANJE POVPREČJA

- Denimo, da poznamo povprečje do n-tega primera $\overline{x}(n)$
- Prispe nov primer $x(n+1)$
- Kako posodobimo povprečje z novim primerom?

1. možnost:

$$\overline{x}(n+1) = \frac{\overline{x}(n) * n + x(n+1)}{n+1}$$

2. možnost:

$$\overline{x}(n) = \frac{1}{n}\sum_{i=1}^{n} x(n) = \frac{\Sigma_x(n)}{n}$$

Hranimo $\boldsymbol{\Sigma_x(n)}$ in ga posodobimo z vsakim novim primerom.

$$\overline{x}(n+1) = \frac{\Sigma_x(n+1)}{n+1}$$

KORAČNO RAČUNANJE VARIANCE

Kot prej, poznamo varianco $Var_x(n)$ in $x(n+1)$, kako izračunamo $Var_x(n+1)$?

$$Var_x(n) = \frac{1}{n}\sum_{i=1}^{n}\left(x(i) - \overline{x}(n)\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x(i) - \frac{\Sigma_x(n)}{n}\right)^2 =$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(x(i)^2 \ - 2x(i)\frac{\Sigma_x(n)}{n} + \left(\frac{\Sigma_x(n)}{n}\right)^2\right) =$$

$$= \frac{1}{n}\left(\Sigma_{x^2}(n) - \frac{1}{n}2\Sigma_x(n)^2 + \frac{1}{n^2}\sum_{i=1}^{n}\Sigma_x(n)^2\right)$$

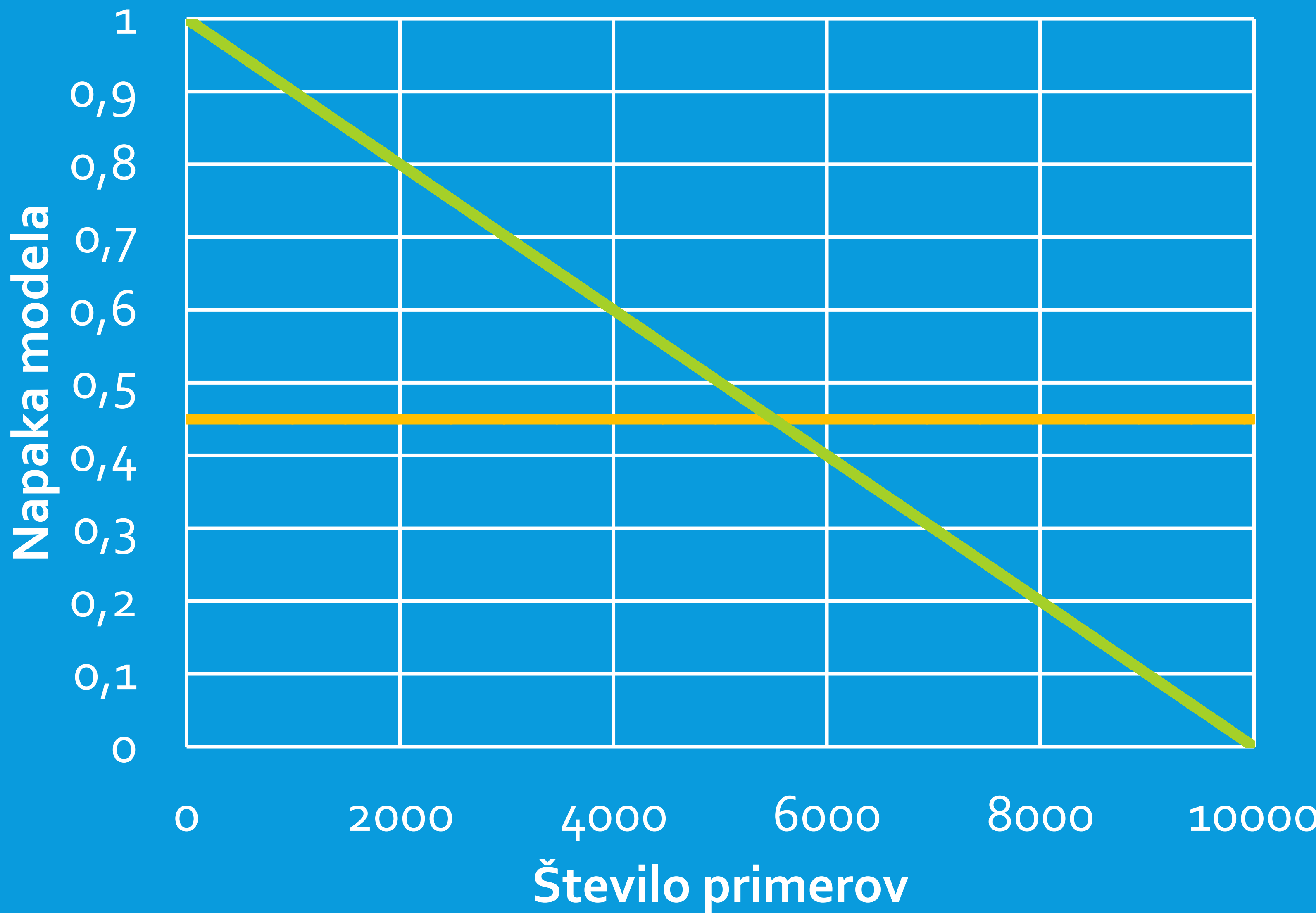
$$= \frac{1}{n}\left(\Sigma_{x^2}(n) - \frac{\Sigma_x(n)^2}{n}\right)$$

Hranimo in posodabljamo $\boldsymbol{\Sigma_x(n)}$ **in** $\boldsymbol{\Sigma_{x^2}(n)}$.

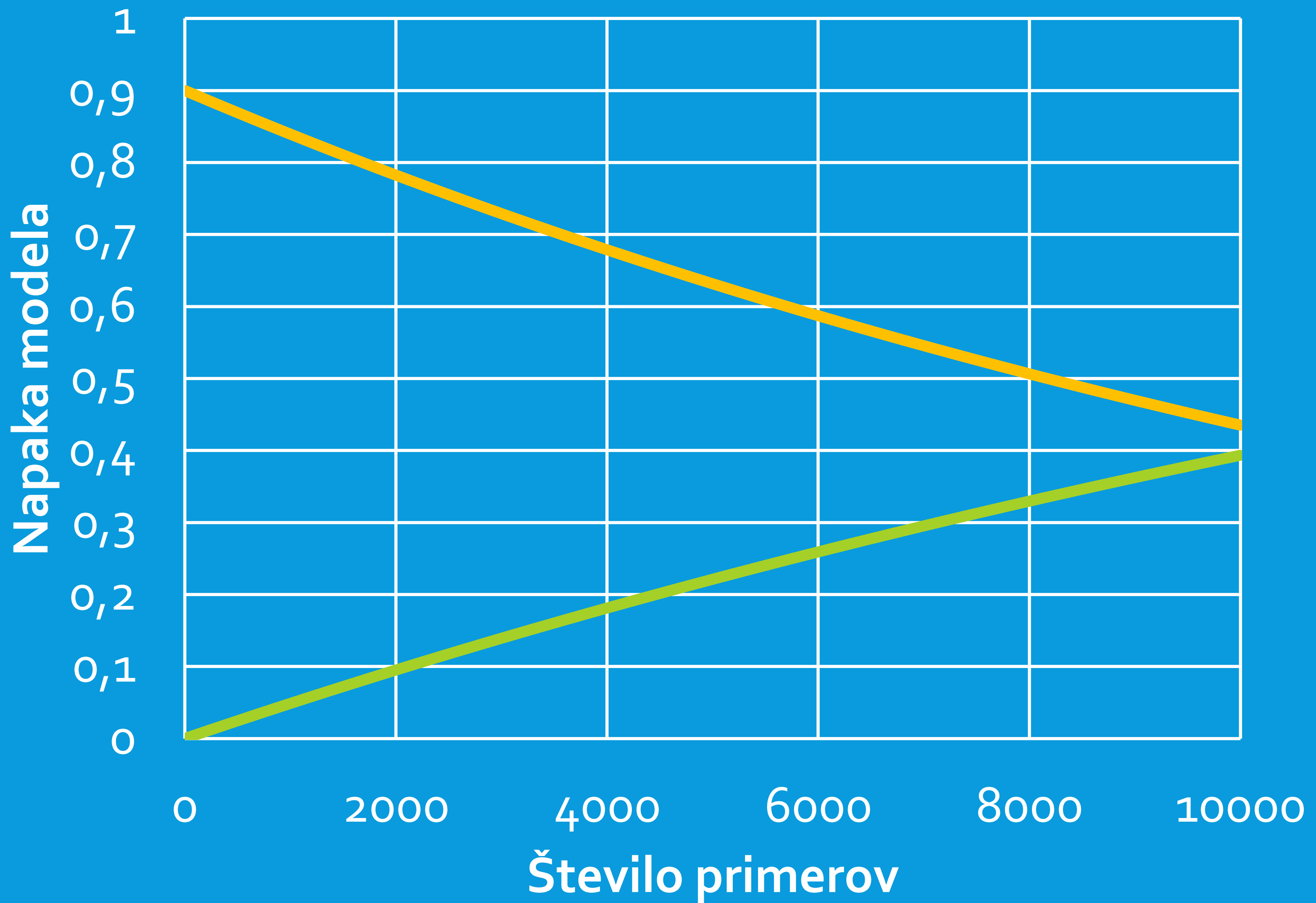
KORAČNO RAČUNANJE STATISTIK

- **Povprečje** – hranimo $\Sigma_x(n)$:
$$\overline{x}(n) = \frac{\Sigma_x(n)}{n}$$
- **Varianca** – hranimo še $\Sigma_{x^2}(n)$:
$$Var_x(n) = \frac{1}{n}\left(\Sigma_{x^2}(n) - \frac{\Sigma_x(n)^2}{n}\right)$$
- **Korelacijski koeficient** – hranimo $\Sigma_x(n)$, $\Sigma_{x^2}(n)$, $\Sigma_y(n)$, $\Sigma_{y^2}(n)$, $\Sigma_{xy}(n)$:
$$Corr_{x,y}(n) = \frac{\Sigma_{xy}(n) - \frac{1}{n}\Sigma_x(n)\Sigma_y(n)}{n\ \Sigma_{x^2}(n)^{\frac{1}{2}}\ \Sigma_{y^2}(n)^{\frac{1}{2}}}$$

VREDNOTENJE MODELOV NA PODATKOVNIH TOKOVIH

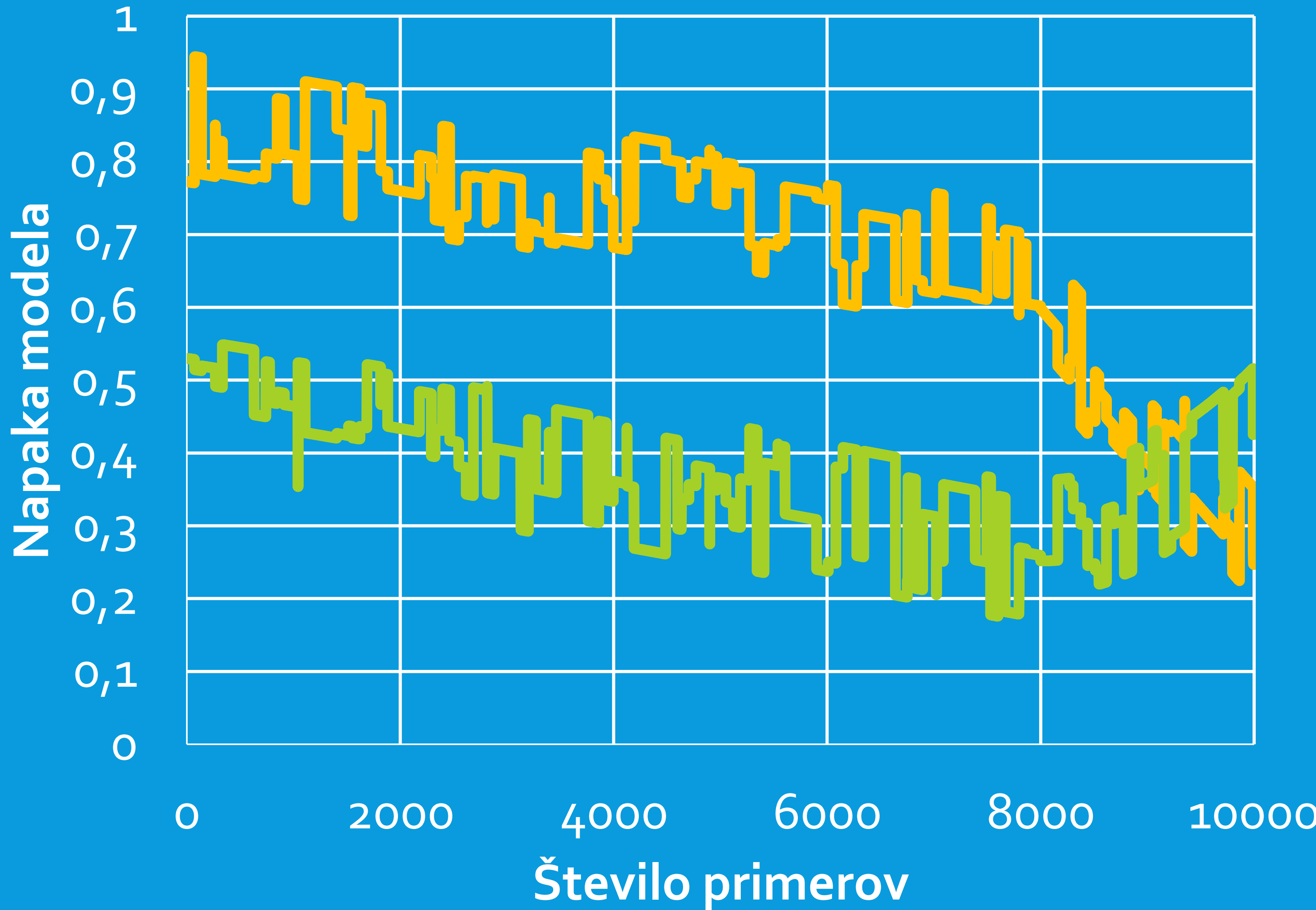


$\overline{\mathrm{err}}\,(\ ) = 0.45$ $\overline{\mathrm{err}}\,(\ ) = 0.5$



$\overline{\mathrm{err}}\,(\blacksquare) = 0.64$

$\overline{\mathrm{err}}\,(\blacksquare) = 0.21$



$\overline{\mathrm{err}}$ (■) = 0.68

$\overline{\mathrm{err}}$ (■) = 0.38

BLEDEČE NAPAKE

- Modelom se na podatkovnih tokovih metrike vrednotenja stalno spreminjajo
- Ko je model naučen le na nekaj primerih, je njegova napovedna vrednost slaba
- Ko se model nauči iz dovolj primerov, se njegova napovedna vrednost izboljša
- Če napako na celotnem podatkovnem toku utežimo na enak način, dobimo pesimistično oceno napovedne vrednosti modela

BLEDEČE NAPAKE (2)

- Na podatkovnih tokovih zato pogosto uporabljamo bledeče napake (an. *faded error*)
- Če lahko izbrano napako zapišemo v obliki funkcije izgube $L(y, \hat{y})$, jo preprosto pretvorimo v bledečo napako
- Če je običajna napaka definirana kot

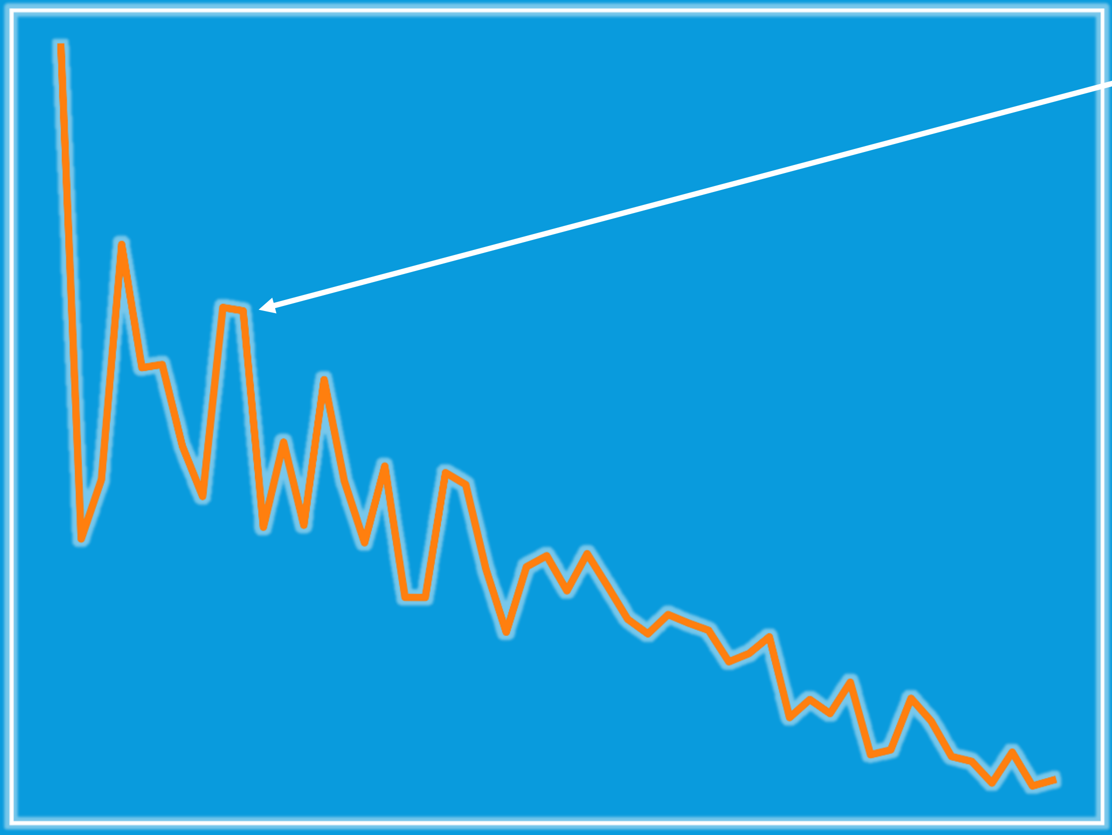
$$E = \frac{\sum_{i=1}^{n} L(y_i, \hat{y}_i)}{n} = \frac{\sum_{i=1}^{n} L(y_i, \hat{y}_i)}{\sum_{i=1}^{n} 1},$$

je ustrezna bledeča napaka definirana kot

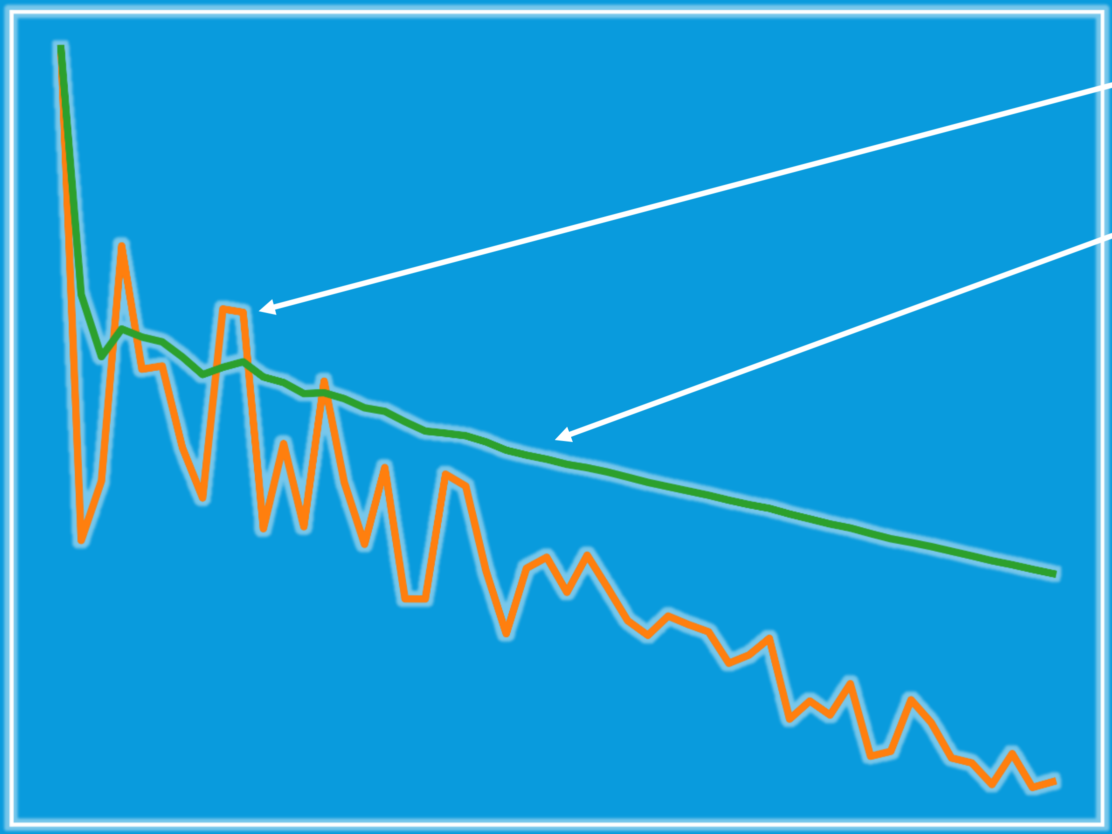
$$E_f = \frac{\sum_{i=1}^{n} \alpha^{n-i} L(y_i, \hat{y}_i)}{\sum_{i=1}^{n} \alpha^{n-i}}$$

- α imenujemo faktor bledenja.

PRIMER BLEDEČE NAPAKE



PRIMER BLEDEČE NAPAKE

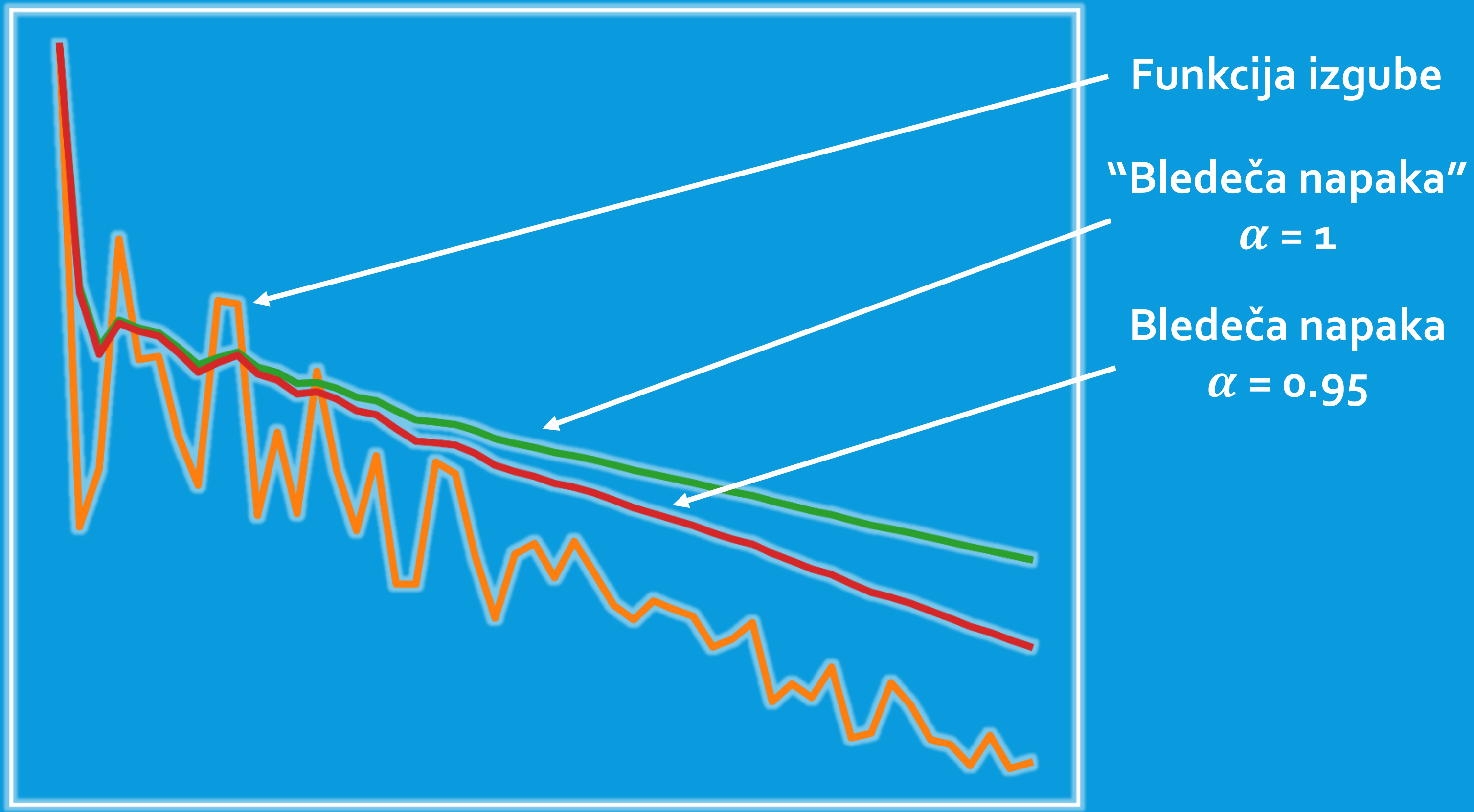


Funkcija izgube

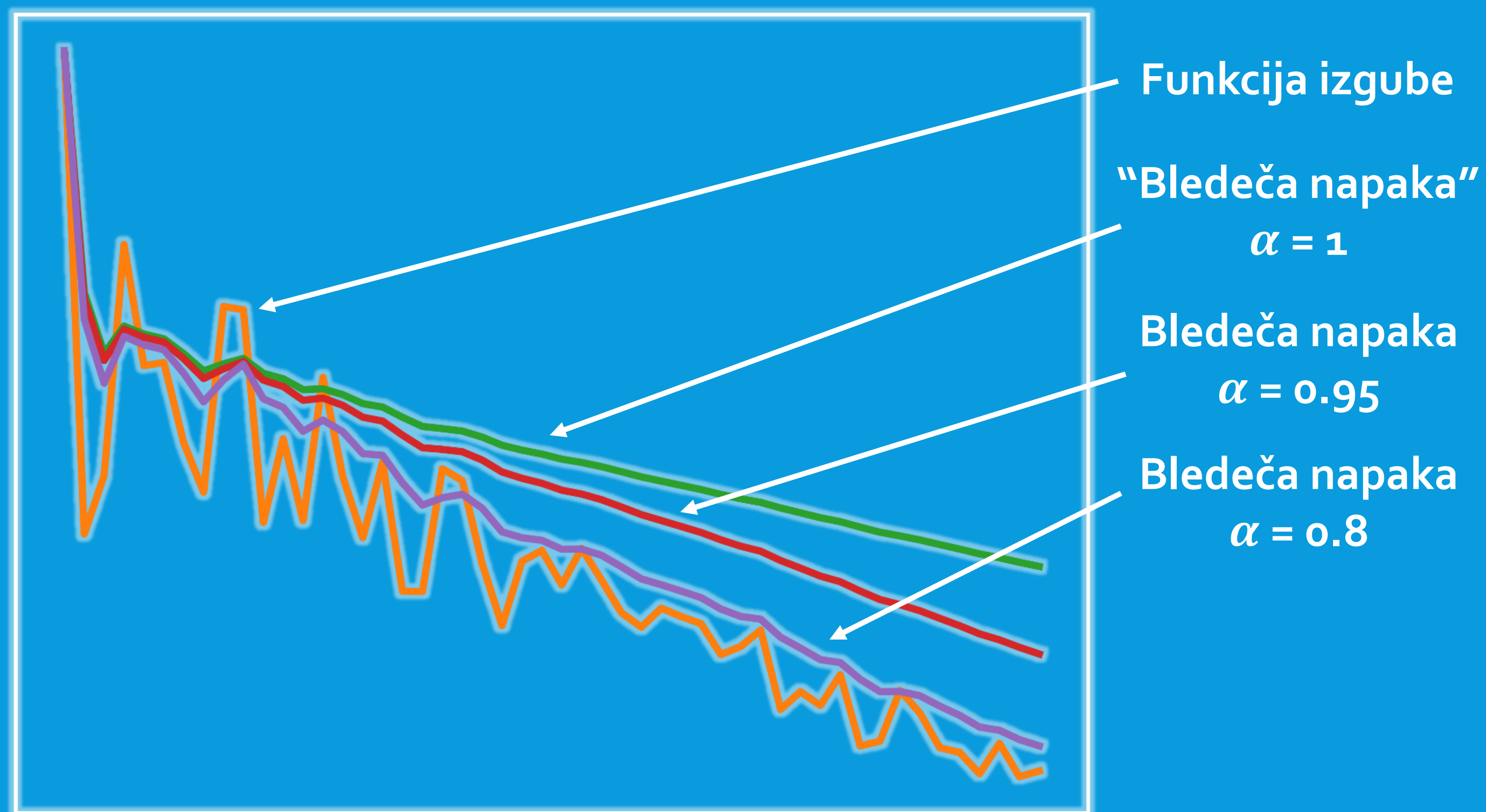
"Bledeča napaka"

$\alpha = 1$

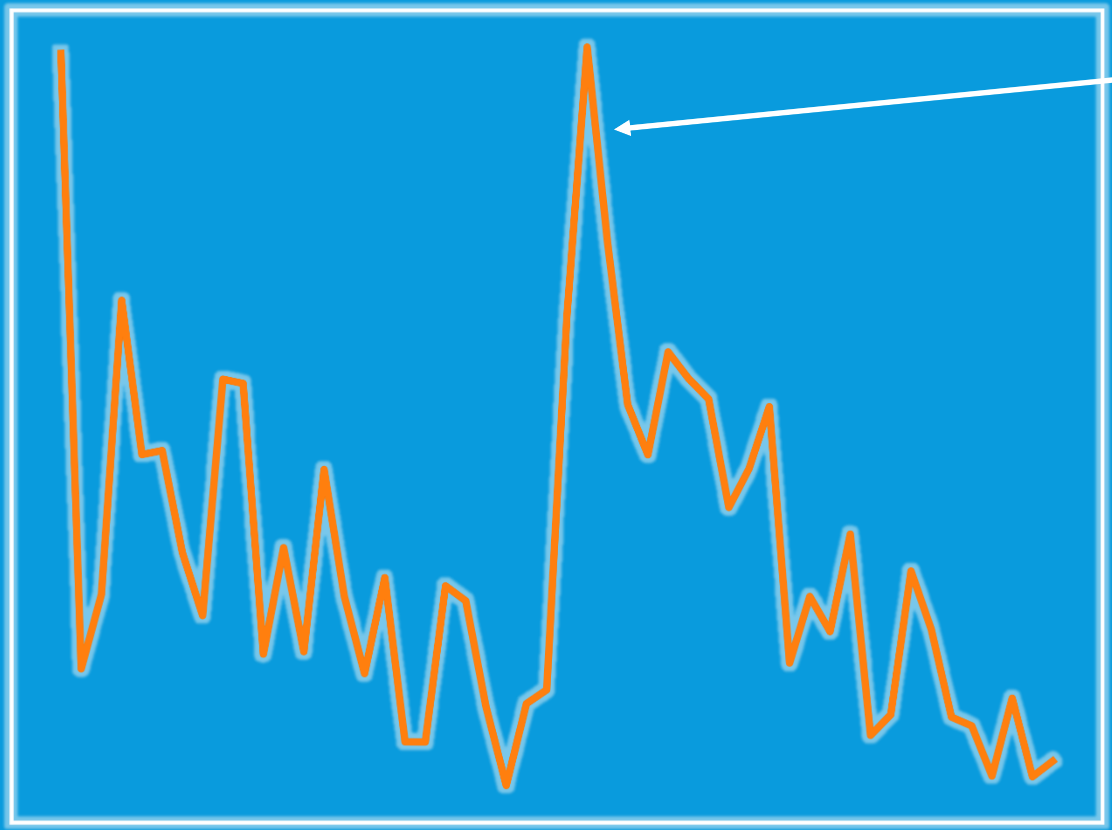
PRIMER BLEDEČE NAPAKE



PRIMER BLEDEČE NAPAKE

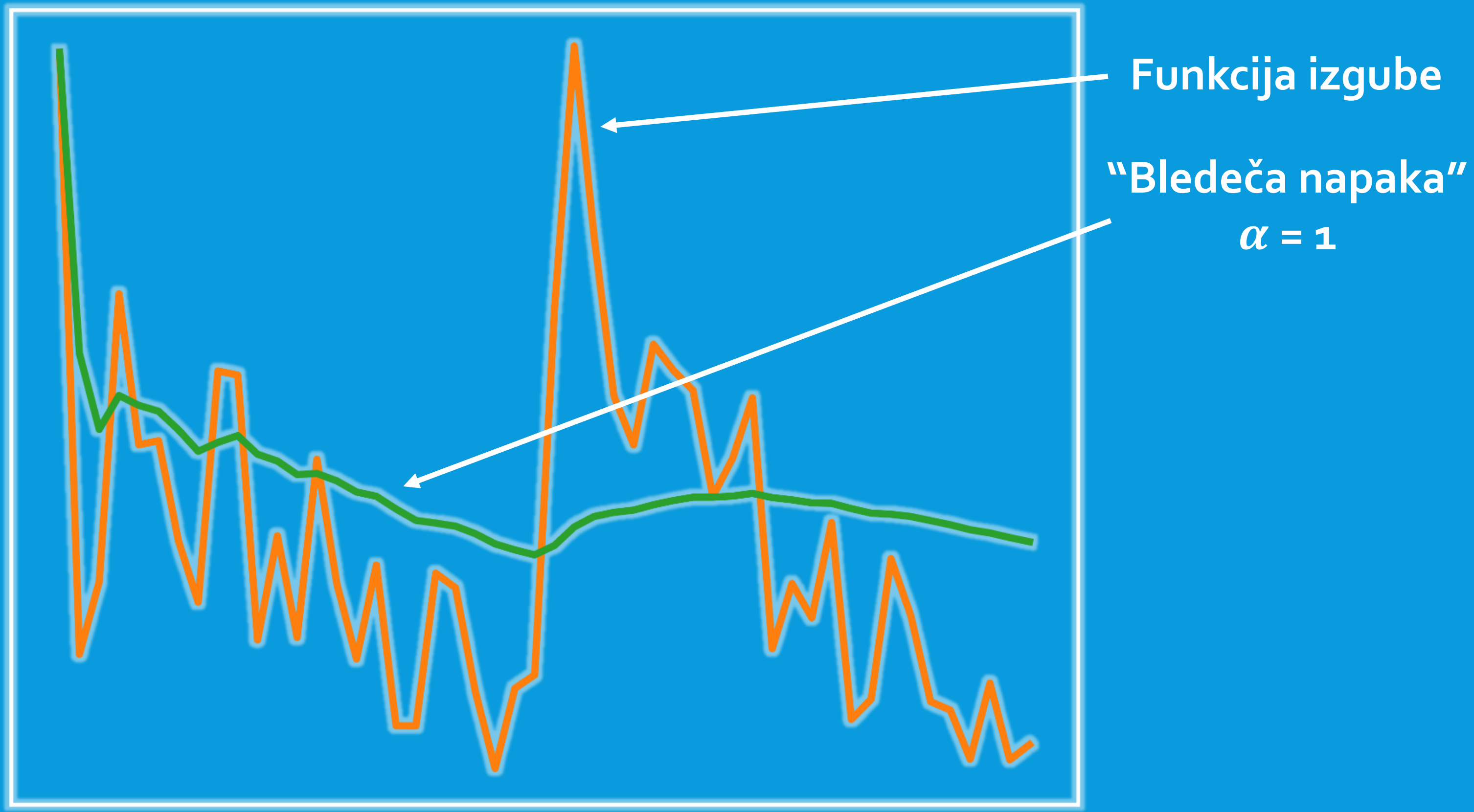


PRIMER BLEDEČE NAPAKE

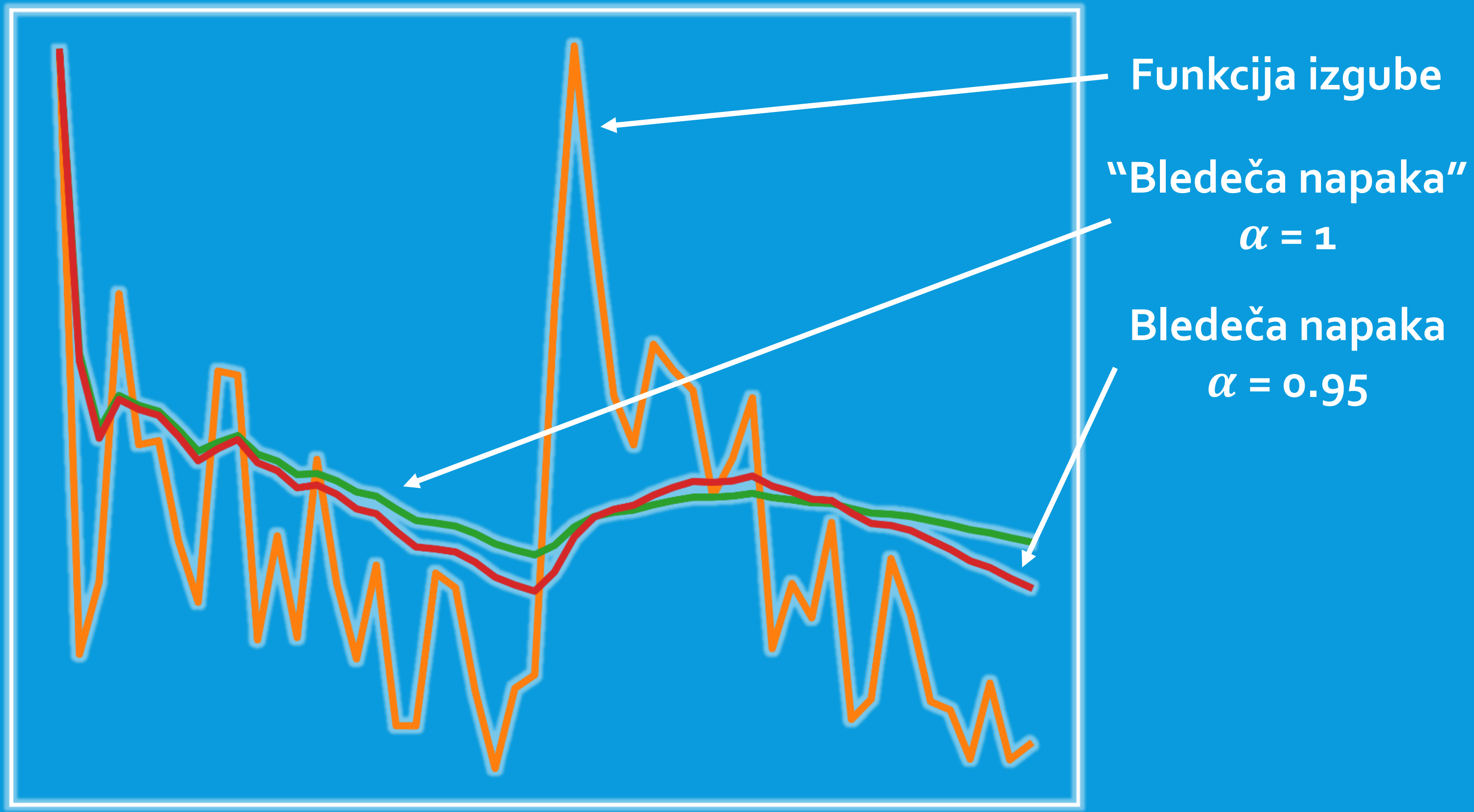


Funkcija izgube

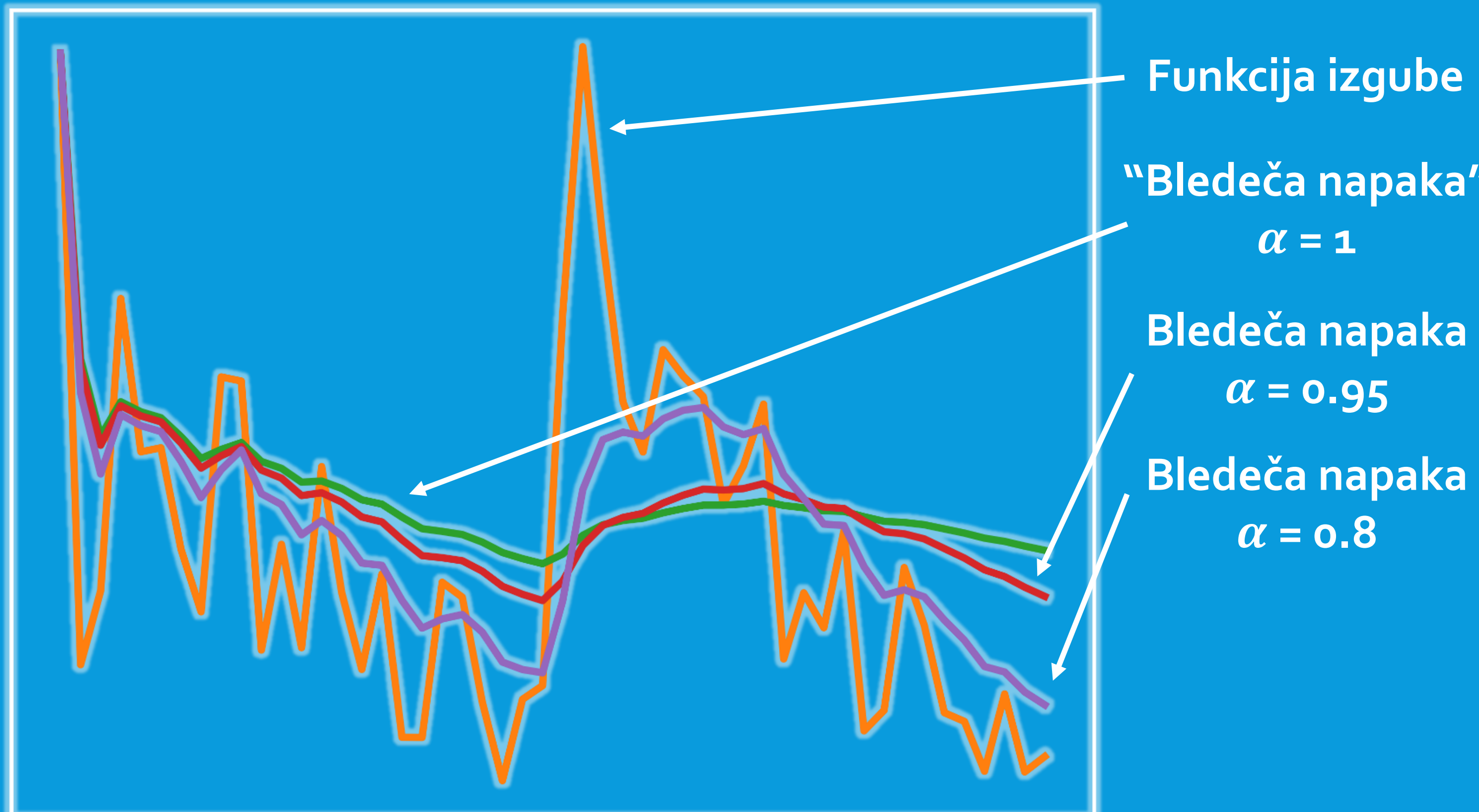
PRIMER BLEDEČE NAPAKE



PRIMER BLEDEČE NAPAKE



PRIMER BLEDEČE NAPAKE



PRISTOPI K VREDNOTENJU NA PODATKOVNIH TOKOVIH

- **Zadržano** (an. *holdout*) vrednotenje:
  - prihajajoče primere zbiramo v okno (an. *window*)
  - ko se okno napolni, na njem izračunamo metrike vrednotenja
  - model naučimo na zbranih primerih v oknu in okno spraznimo
- **Napovedno zaporedno** (an. *predictive sequential, prequential*) vrednotenje:
  - vsak primer takoj uporabimo za izračun napovedi (metrik vrednotenja)
  - primer takoj podamo modelu za učenje

ZADRŽANO VREDNOTENJE

- Imamo okno W, napovedi P, trenutni model h, posodobitveni operator u
- Incializacija: $W := []$
- Za vsak nov primer (x, y):
  - Izračunamo napoved $h(x) = \hat{y}$ in jo dodamo v P
  - (x, y) dodamo v W
  - Če je W poln:
    - Za vsak $(x, y) \in W$: Posodobimo model $h := u(h, (x, y))$
    - $W := []$

NAPOVEDNO ZAPOREDNO VREDNOTENJE

- Imamo napovedi P, trenutni model h in posodobitveni operator u
- Za vsak nov primer (x, y):
  - Izračunamo napoved $h(x) \ = \hat{y}$ in jo dodamo v P
  - Posodobimo model $h \ := \ u(h, (x, y))$

VRSTE SPREMEMB

SPREMEMBE

- Neformalno: odvisnost ciljnih spremenljivk od vhodnih se s časom spremeni
- Formalno:

$$\exists t_0, t_1 : \Pr_{t_0}(X, Y) \neq \Pr_{t_1}(X, Y)$$

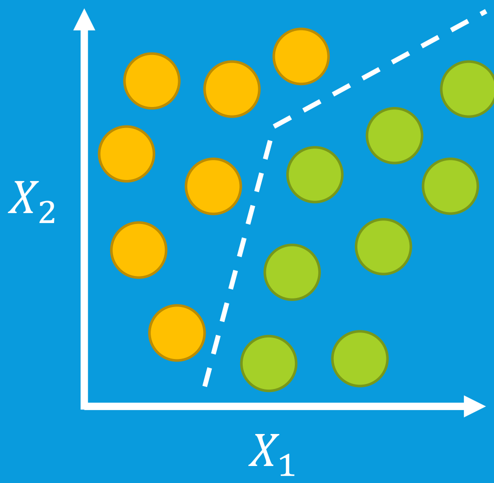
- Spremembe klasificiramo glede na 2 kriterija:
  - **Tip sprememb** (zaradi česa se je spremenila odvisnost)
  - **Časovni profil** sprememb

IZVOR SPREMEMB

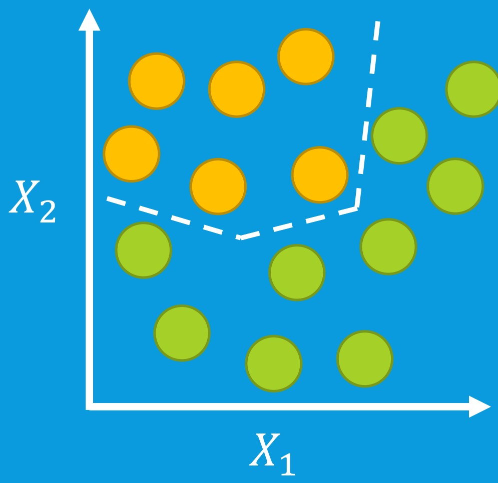
- Spremembe v $\Pr(X, Y)$ lahko povzroči:
- $\Pr(Y)$ – spremeni se porazdelitev v ciljnem prostoru
- $\Pr(Y \mid X)$ – spremeni se odvisnost ciljnih vrednosti od vhodnih
- $\Pr(X)$ – spremeni se porazdelitev v vhodnem prostoru

PRAVE SPREMEMBE

- **Prave spremembe** se nanašajo na spremembe v $\Pr(Y \mid X)$
- Spremenjena odvisnosti zahteva učenje novih modelov (premik odločitvene meje)

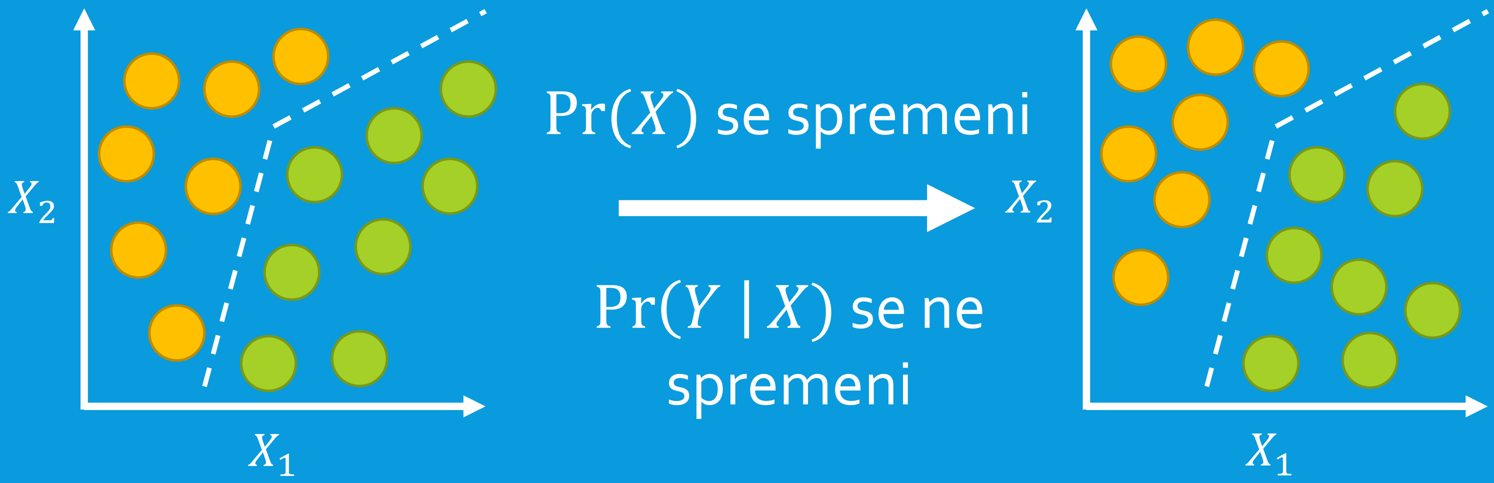


$\Pr(Y \mid X)$ se spremeni

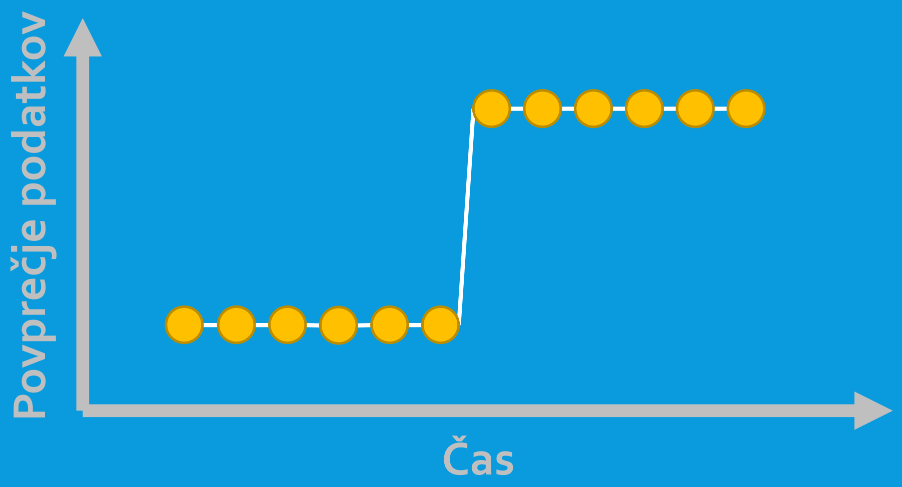


NAVIDEZNE SPREMEMBE

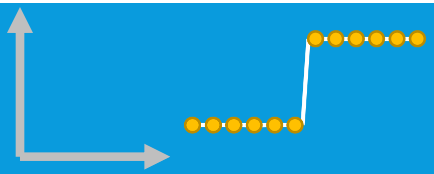
- **Navidezne spremembe** se nanašajo na spremembi v $\Pr(X)$ ali $\Pr(Y)$
- Naučeni modeli modelirajo odvisnost med X in Y, zato jih ni treba posodobiti



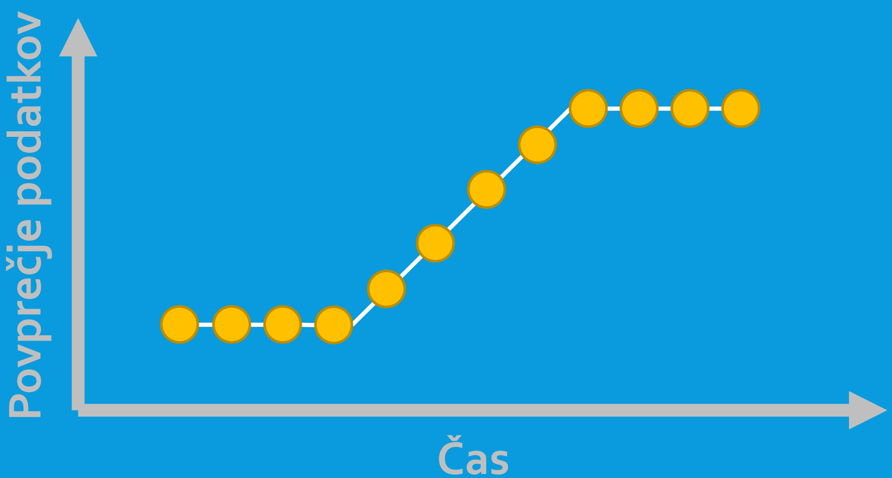
ČASOVNI PROFILI SPREMEMB



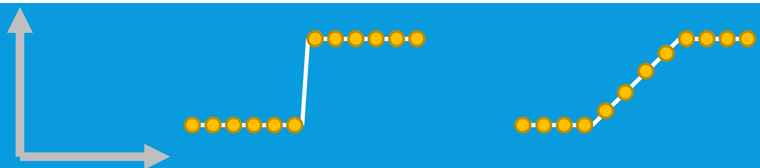
ČASOVNI PROFILI SPREMEMB



Nenadna sprememba

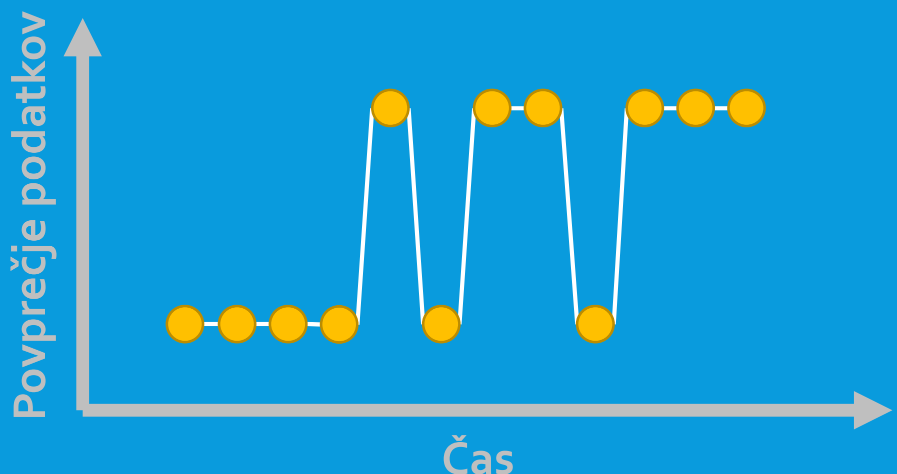


ČASOVNI PROFILI SPREMEMB

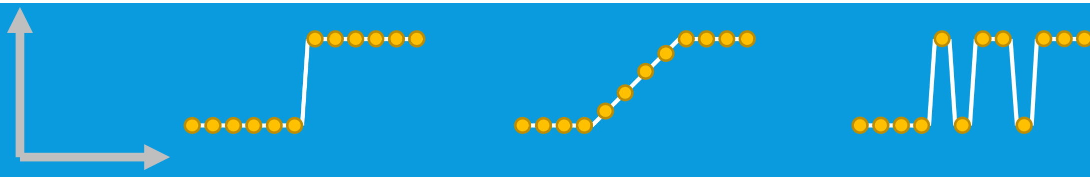


Nenadna sprememba

Koračna sprememba



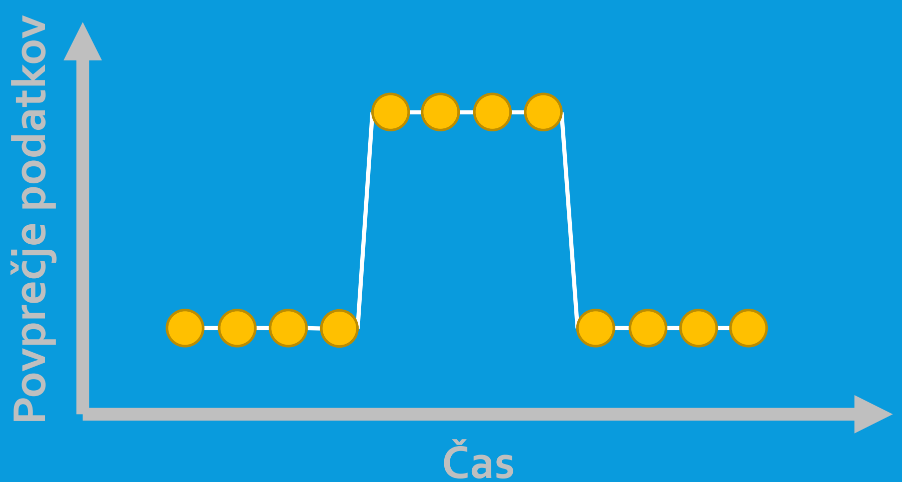
ČASOVNI PROFILI SPREMEMB



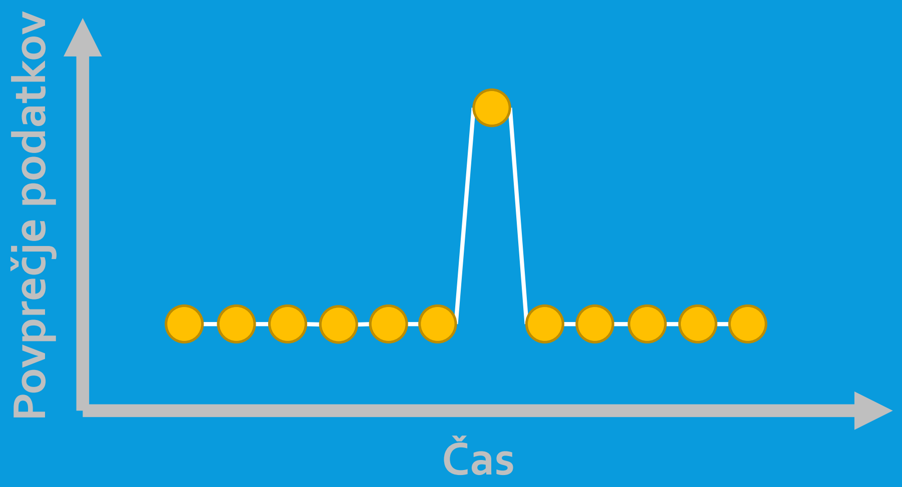
Nenadna sprememba

Koračna sprememba

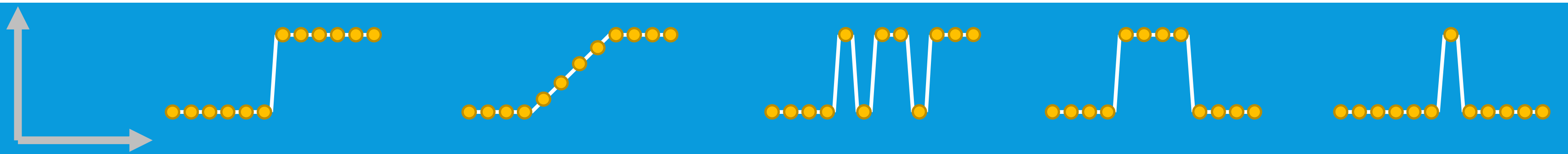
Postopna sprememba



ČASOVNI PROFILI SPREMEMB



ČASOVNI PROFILI SPREMEMB



Nenadna sprememba

Koračna sprememba

Postopna sprememba

Ponavljajoči koncepti

Osamelec (ni sprememba)

MEHANIZMI ZA ZAZNAVANJE SPREMEMB

- Opazovanje mere ustreznosti (napovedna vrednost, hevristika)
  - Page-Hinkleyev test
  - Preverjanje ali pogoji za izgradnjo drevesa še držijo (Hoeffdingova neenakost)
- Primerjava distribucij na dveh časovnih oknih (referenčno okno in okno nedavnih primerov)
  - ADWIN (naslednjič)

PAGE-HINKLEYEV TEST

- Opazujemo signal $x(t)$, najpogosteje napako
- Definiramo $m(t)$ in $M(t)$ kot:

$$m(t) = \sum_{i=1}^{t} (x(t) - \overline{x}(t) - \alpha)$$

$$M(t) = \min_{i=1,\ldots,t} m(t)$$

- Kadar $m(t) - M(t)$ večja od predpisane λ, zaznamo spremembo

PAGE-HINKLEYEV TEST (2)

$$m(t) = \sum_{i=1}^{t} (x(t) - \overline{x}(t) - \alpha)$$

$$M(t) = \min_{i=1,\ldots,t} m(t)$$

$$m(t) - M(t) > \lambda$$

Parameter α uravnava najmanjšo spremembo, ki jo želimo zaznati. Parameter λ uravnava, koliko lažnih preplahov dovolimo.

DREVESA NA PODATKOVNIH TOKOVIH

DREVESA NA PODATKOVNIH TOKOVIH

- Pri običajnem strojnem učenju za učenje dreves najpogosteje uporabljamo TDIDT (an. *top-down induction of decision trees*)
- TDIDT ne deluje na podatkovnih tokovih (nimamo dostopa do vseh podatkov hkrati za izračun hevristike)
- Za gradnjo dreves na podatkovnih tokovih uporabljamo prilagojen pristop

UČENJE DREVES NA PODATKOVNIH TOKOVIH

- Pri običajnem učenju dreves delitve izbiramo s pomočjo vseh podatkov
- Na podatkovnih tokovih to ni možno
- **Ideja:** za izbor prave delitve potrebujemo le majhno podmnožico vseh primerov
  - Delitev v korenu bo določilo prvih nekaj primerov
  - Nadaljnje delitve v listih bodo podmnožice primerov, ki so v liste razvrščene

UČENJE DREVESA

- Začnemo s praznim listom (začetni model h_0)
- Ko dobimo nov primer, na dosedanjih primerih* ovrednotimo delitve
- Oglejmo si najbolje ovrednoteno delitev
  - Ta delitev je lahko najboljša po naključju (še posebej, če imamo malo primerov)
  - Kako vemo kdaj imamo dovolj primerov, da je najboljša delitev res najboljša?
  - Pomagamo si s Hoeffdingovo neenakostjo

HOEFFDINGOVA NEENAKOST

- **Izrek:** Naj bodo $X_1, \ldots, X_n$ neodvisne omejene naključne spremenljivke, tj.,

$$\Pr(X_i \in [a_i, b_i]) = 1, \qquad i = 1, \ldots, n.$$

Naj bo $\overline{X} = \frac{1}{n}(X_1 + \cdots + X_n)$. Tedaj velja

$$\Pr\left(\overline{X} - \mathbb{E}\left[\overline{X}\right] \geq \varepsilon\right) \leq \exp\left(-\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

HOEFFDINGOVA NEENAKOST (2)

Posledica: Predpostavimo kot prej. Tedaj velja

$$\Pr\left(\left|\overline{X} - \mathbb{E}\left[\overline{X}\right]\right| \geq \varepsilon\right) \leq 2\exp\left(-\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right) =: \delta.$$

ε izrazimo z δ:

$$\varepsilon \leq \sqrt{\frac{\sum_{i=1}^{n}(b_i - a_i)^2}{2n^2}\ln\left(\frac{2}{\delta}\right)}.$$

HOEFFDINGOVA NEENAKOST(3)

Če smo izmerili vrednosti spremenljivk $X_1, \dots, X_n$ lahko izračunamo njihovo povprečno vrednost $\overline{X}$, ki jo označimo z X_{vzorec}.

Ker so X_i naključne spremenljivke, X_{vzorec} verjetno odstopa od $\mathbb{E}\left[\overline{X}\right] = \mathbb{E}[X_{vzorec}]$, ki jo označimo z X_{prava}.

Hoeffdingova neenakost nam pove, da X_{prava} leži v ε-okolici X_{vzorec} z verjetnostjo $1 - \delta$.

UPORABA HOEFFDINGOVE NEENAKOSTI

- Imamo hevristiko H, ki jo želimo maksimirati
  - Kot primer bomo uporabili redukcijo variance
- Za vsak atribut poiščemo najboljšo delitev
  - Če je atribut nominalen, je to praviloma delitev po vseh vrednostih
  - Če je atribut numeričen, preverimo vse smiselne točke delitve
- Ali lahko izberemo atribut z maksimalno vrednostjo hevristike?
- **NE.**
- **Izmerjeno se je lahko zgodilo po naključju. Za utemeljitev uporabimo Hoeffdingovo neenakost.**

REDUKCIJA VARIANCE

- Redukcija variance – za regresijo:

$$VarRed = Var(S) - \frac{|S_L|}{|S|} Var(S_L) - \frac{|S_D|}{|S|} Var(S_D)$$

S – množica primerov zbranih v listu
S_L in S_D – množici primerov po delitvi

- Pri klasifikaciji lahko uporabimo informacijski prispevek ali Gini indeks

UPORABA HOEFFDINGOVE NEENAKOSTI

- Poznamo redukcije varianc za vse atribute
- $VarRed_1$ pripada najboljši delitvi
- $VarRed_2$ pripada drugi najboljši delitvi
- Opazujemo

$$r(n) = \frac{VarRed_1(n)}{VarRed_2(n)}$$

kot slučajno spremenljivko

UPORABA HOEFFDINGOVE NEENAKOSTI (2)

- Ker je $r(n)$ razmerje med redukcijo variance najboljše ter druge najboljše delitve, gotovo velja $r(n) \in [0,1]$, tj., $a_i = 0, b_i = 1, \forall i$.
- Tedaj

$$\bar{r} = \frac{r(1) + \cdots + r(n)}{n}$$

ustreza pogojem Hoeffdingove neenakosti*.

UPORABA HOEFFDINGOVE NEENAKOSTI (3)

- $\overline{r}_{prava}$ z verjetnostjo $1 - \delta$ leži v ε-okolici $\overline{r}_{vzorec}$, kjer je

$$\varepsilon = \sqrt{\frac{\ln(1/\delta)}{2n}}$$

oziroma:

$$\overline{r}_{prava} \in [\overline{r}_{vzorec} - \varepsilon, \overline{r}_{vzorec} + \varepsilon,] = [r^-, r^+].$$

- Če $r^+ < 1$, je z verjetnostjo $1 - \delta$ tudi $\overline{r}_{prava} < 1$
- Najboljša delitev je "res" boljša od ostalih delitev

DELITEV LISTA

- Preverjanje delitev je potratno
- Za poljuben primer je verjetnost, da bomo list razdelili ravno za njim, majhna
- Delitve ovrednotimo le, ko se nabere dovolj primerov
  - Ovrednotimo po 200, 400, 600, … nabranih primerih
  - Interval preverjanja delitev je parameter učenja dreves na podatkovnih tokovih

FIMT-DD

- Hitra indukcija modelnih dreves (an. *Fast Induction of Model Trees with Drift Detection*) [Ikonomovska, Džeroski 2010]
- **Zaznavanje sprememb:** Page-Hinkleyev test
- **Delitev listov:** redukcija variance in Hoeffdingova neenakost
- V vsakem listu hrani linearen model (perceptron)
  - za napovedovanje ciljne spremenljivke uporablja vrednosti vhodnih atributov

FIMT-DD – ZAZNAVANJE IN PRILAGAJANJE SPREMEMBAM

- V vsakem vozlišču drevesa opazujemo Page-Hinkleyev test
- Ko vozlišče sporoči spremembo, poiščemo najvišjega prednika, ki jo je še zaznal
- V najdenem vozlišču začnemo izgradnjo novega poddrevesa
- Ko je novo drevo natančnejše, staro zavržemo


1. V drevesu je zaznana sprememba
A > 1.0
B > 0.7
C > 0.1
42
15
83
7
2. Sproži se gradnja novega poddrevesa
A > 1.0
B > 0.7
C > 0.1
5
42
15
83
7
3. Novo poddrevo raste
A > 1.0
B > 0.7
C > 0.1
C > 1.7
42
15
83
7
66
13
4. Novo poddrevo nadomesti starega
A > 1.0
B > 0.7
C > 1.7
42
15
66
13

FIMT-DD – h_0 IN u

- Začetni model h_0: prazen list
- Posodobitveni operator u:
  - Primer razvrsti v list (tudi v pomožna poddrevesa)
  - **Posodobitev statistik:**
    - V listu posodobi statistike za vrednotenje delitev
  - **Zaznavanje sprememb:**
    - Na podlagi napovedi posodobi Page-Hinkleyev test v listu in njegovih prednikih
  - **Prilagajanje spremembam:**
    - Če je sprožena sprememba, v ustreznem vozlišču ustvari pomožno poddrevo
    - Če je pomožno poddrevo že obstaja in je boljše, zamenja izvirno poddrevo
  - **Delitev:**
    - Če je zbranih dovolj primerov, ovrednoti delitve in poizkusi razdeliti list

FIMT-DD – PARAMETRI

- Parametri Page-Hinkleyevega testa
  - α – uravnava minimalno spremembo
  - λ – uravnava dovoljeno pojavnost lažnih signalov
- Parametri Hoeffdingove neenakosti
  - δ – določa verjetnost za (ε, δ)-aproksimacijo
- Interval preverjanja delitev n
  - n – posredno določi ε za (ε, δ)-aproksimacijo (Hoeffdingova neenakost)
- Številni drugi parametri