

# Strojno učenje

## Struktura metoda/algoritama strojnog učenja

Tomislav Šmuc

## Osnovni pojmovi

- Ulazni vektor varijabli (engl. *attributes, features*):  $\mathbf{x} = (x_1, x_2, \dots, x_d)$ 
  - Broj ulaznih varijabli:  $d$
- Izlazna ili ciljna varijabla (engl. *target variable*):  $y$
- Primjer za učenje (engl. *training example*):  $(\mathbf{x}, y)$
- Skup primjera za učenje (engl. *training examples*):  
 $D = \{(\mathbf{x}_i, y_i); i = 1 \dots N\}; \quad N - \text{broj primjera za učenje}$
- Nepoznata ciljna (idealna) funkcija (koncept):  $f: \mathcal{X} \rightarrow \mathcal{Y}, y = f(\mathbf{x})$

- **Cilj problema nadziranog učenja:**
  - za dani skup primjera za učenje
  - „naučiti” funkciju  $h: \mathcal{X} \rightarrow \mathcal{Y}$  t.d.  $h(\mathbf{x}) \approx f(\mathbf{x})$ 
    - da  $h(\mathbf{x})$  bude „dobar prediktor” za odgovarajuću vrijednost  $y$
  - **Regresija:** ako ciljna varijabla  $y$  prima kontinuirane vrijednosti
  - **Klasifikacija:** ako ciljna varijabla  $y$  kategorička odnosno može poprimiti samo mali broj diskretnih vrijednosti
- Skup, prostor hipoteza/modela (klasa hipoteze):  $\mathcal{H}$ 
  - npr. skup svih klasifikatora:
$$\mathcal{H} = \{h_{\mathbf{w}}: h_{\mathbf{w}}(\mathbf{x}) = 1\{\mathbf{w}^T \mathbf{x} \geq 0\}, \mathbf{w} \in \mathbb{R}^{d+1}\}$$
  - Skup, prostor konzistentnih hipoteza:  $VS_{\mathcal{H}, D}$ 
    - Podskup hipoteza iz  $\mathcal{H}$  konzistentnih s primjerima za učenje iz  $D$
    - $VS_{\mathcal{H}, D} = \{h \in \mathcal{H} | \text{konzistentna}(h|D)\}$

## Karakteristike skupa hipoteza

- Veličina skupa hipoteza:
  - fiksna
  - varijabilna
- Parametrizacija hipoteze – opis hipoteze:
  - skupom simboličkih (diskretnih) varijabli
  - skupom kontinuiranih parametara
- Evaluacija hipoteze/modela
  - s obzirom na **determinističku hipotezu** primjer za učenje je ili konzistentan ili ne konzistentan
  - s obzirom na **stohastičku hipotezu** primjer za učenje je više ili manje vjerojatan

## Regresija

x1	x2	y
0.5	3.4	2.3
1.0	2.3	1.5
0.8	0.2	3.3
5.1	4.1	4.0

$$y \in \mathbb{R}$$

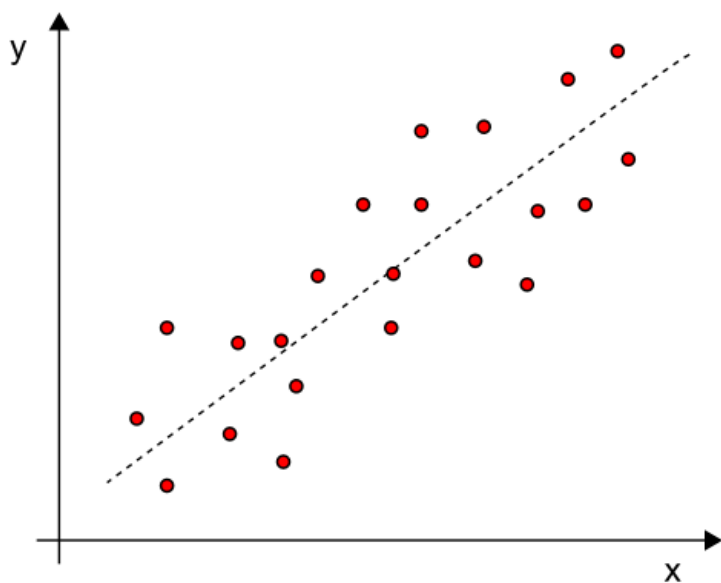
Razlika je u obliku  
ciljne varijable!

## Klasifikacija

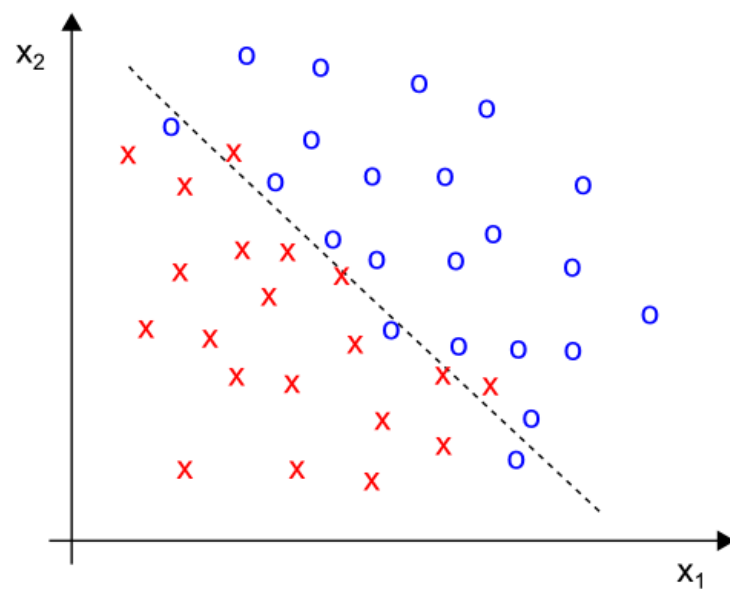
x1	x2	y
0.5	3.4	x
1.0	2.3	o
0.8	0.2	o
5.1	4.1	x

$$y \in \{x, o\}$$

Regresija

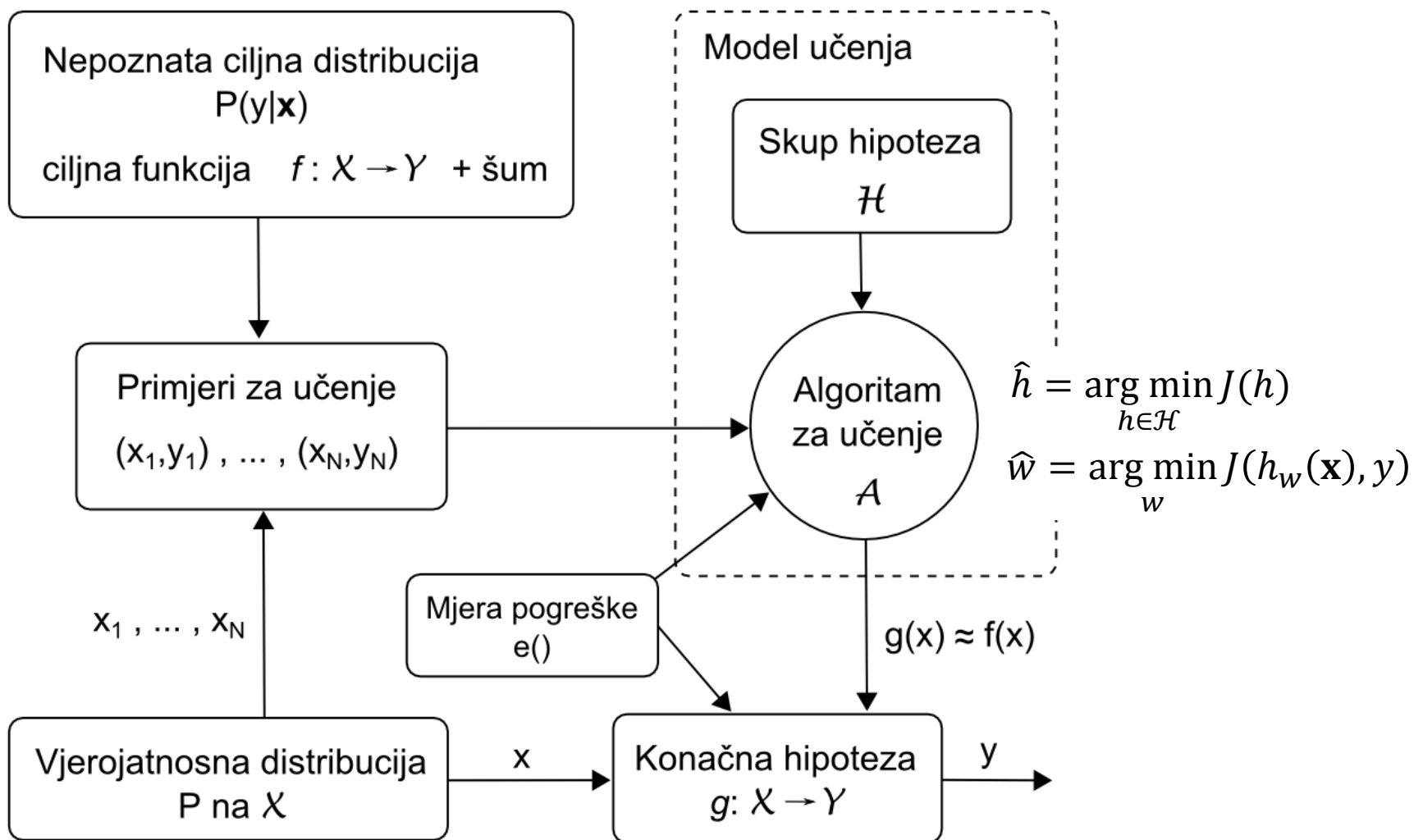


Klasifikacija



- Svaki algoritam za učenje uključuje:
  - Reprezentaciju
    - Linearna regresija, stabla odluke, skup pravila, neuronske mreže, strojevi potpornih vektora, ansambli
  - Evaluaciju  $\leq$  mjera greške
    - Točnost, preciznost i odaziv, kvadratna greška, ...
  - Optimizaciju / pretraživanje nad prostorom parametara
    - Pretraživanje prostora: (Hill-climbing, Greedy search, pattern search)
    - Kombinatorna optimizacija  $\leq$  diskretna reprezentacija
      - Iscrpna pretraga
    - Konveksna optimizacija
      - gradijentni spust (engl. *gradient decent*)
    - Optimizacija uz ograničenja
      - Linearno programiranje, Lagrangeova funkcija

# Dijagram učenja





# Reprezentacija modela za učenje: primjer

- Hipoteza:

$$h_{\mathbf{w}}(x) = w_0 + w_1 x_1$$

- Parametri:

$$\mathbf{w} = (w_0, w_1)$$

- Mjera greške ili funkcija troška:

$$J(\mathbf{w}) = \frac{1}{2N} \sum_{i=1}^N (h(\mathbf{x}_i) - y_i)^2$$

- Cilj:

$$\min_{\mathbf{w}} J(\mathbf{w})$$

optimizacijski problem  $\Rightarrow$  algoritmi optimizacije

# Reprezentacija modela: linearni modeli

regresija

Linearna regresija (engl. *linear regression*)

Hipoteza:  $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$     Funkcija troška:  $J(h) = \frac{1}{2N} \sum_{i=1}^N (h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$

Lokalno otežana linearna regresija (engl. *locally weighted linear regression*)

Hipoteza:  $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$     Funkcija troška:  $J(h) = \frac{1}{2N} \sum_{i=1}^N s_i (h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$

klasifikacija

Logistička regresija (engl. *logistic regression*)

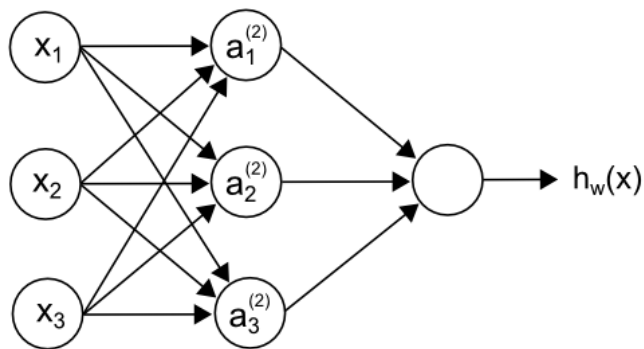
Želimo:  $h_{\mathbf{w}}(\mathbf{x}) \in [0,1]$     za **binarnu klasifikaciju**:  $y \in \{0,1\}$

Hipoteza:  $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$      $P(y = 1 | \mathbf{x}; \mathbf{w})$

Funkcija troška:  $J(\mathbf{w}) = -\frac{1}{N} \left[ \sum_{i=1}^N (y_i \log h_{\mathbf{w}}(\mathbf{x}_i) + (1 - y_i) \log(1 - h_{\mathbf{w}}(\mathbf{x}_i))) \right]$

# Reprezentacija modela: nelinearna klasifikacija

Neuronska mreža za klasifikaciju (engl. *neural networks*)



ulazni sloj    skriveni sloj    izlazni sloj

$$a_1^{(2)} = g(w_{10}^{(1)} x_0 + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(w_{20}^{(1)} x_0 + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(w_{30}^{(1)} x_0 + w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3)$$

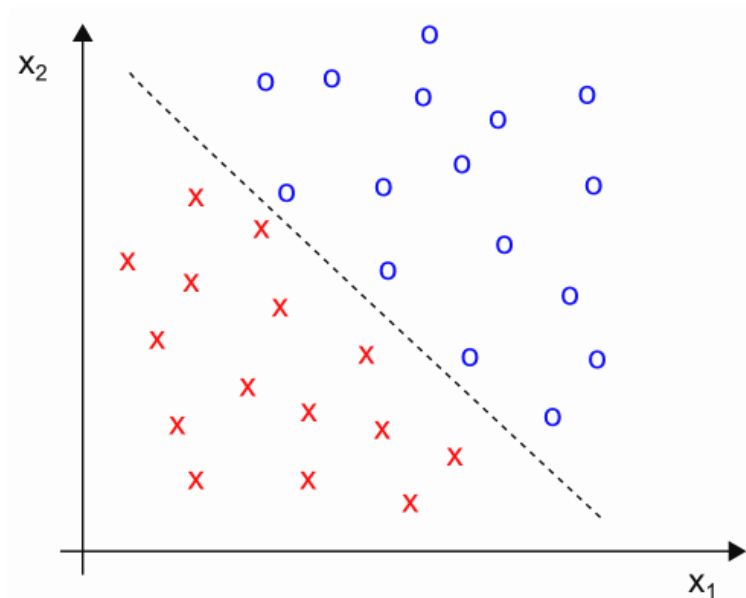
$$h_w(x) = a_1^{(3)} = g(w_{10}^{(2)} a_0^{(2)} + w_{11}^{(2)} a_1^{(2)} + w_{12}^{(2)} a_2^{(2)} + w_{13}^{(2)} a_3^{(2)})$$

Tipično  $g(x)$ :

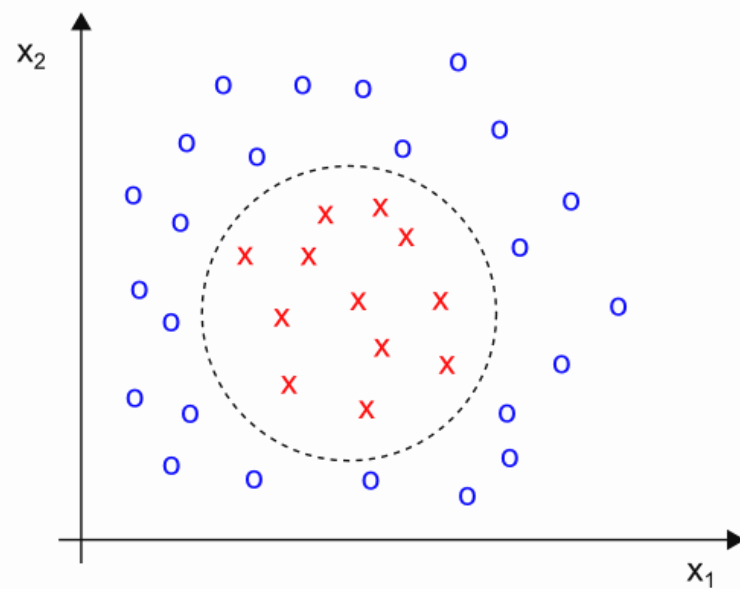
$$g(x) = 1/(1 - e^{-x})$$

# Oblici/složenost hipoteza: klasifikacija

Linearni model

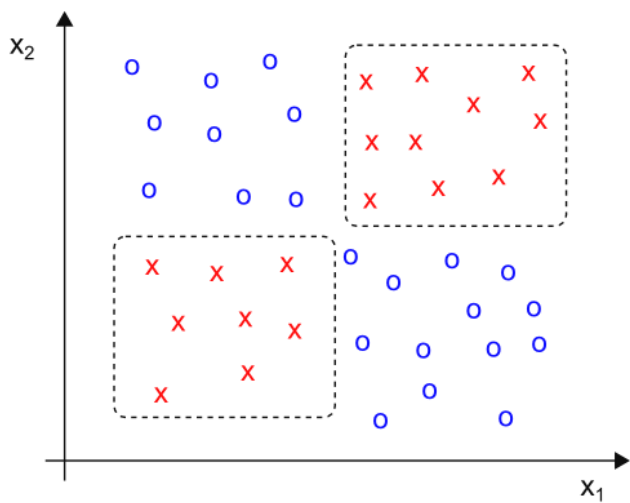


Nelinearni model



# Reprezentacija modela: klasifikacija

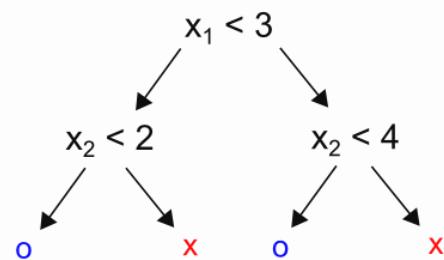
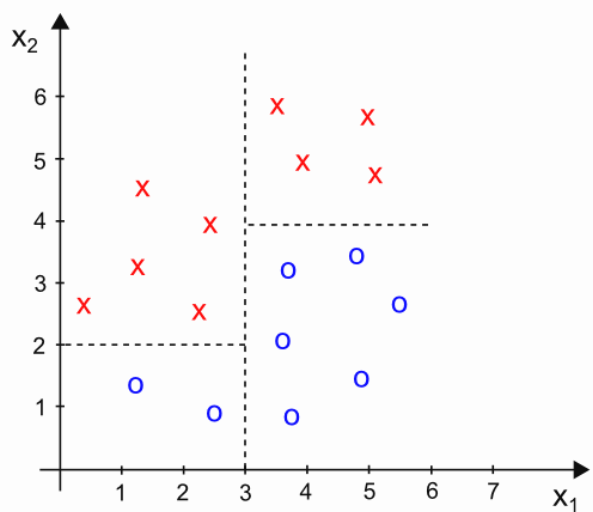
## Pravila (engl. *Rules*)



IF ( $x_1 > 0.5$ ) AND ( $x_1 < 4.5$ ) AND ( $x_2 > 0.5$ ) AND ( $x_2 < 4.0$ ) THEN  $y = \text{red}$

IF ( $x_1 > 5.0$ ) AND ( $x_1 < 7.0$ ) AND ( $x_2 > 4.5$ ) AND ( $x_2 < 6.5$ ) THEN  $y = \text{red}$

## Stablo odluke (engl. *Decision tree*)



## Parametarski-ne-parametarski algoritmi (Podjela koja nije sasvim jasno definirana)

### Parametarski algoritmi

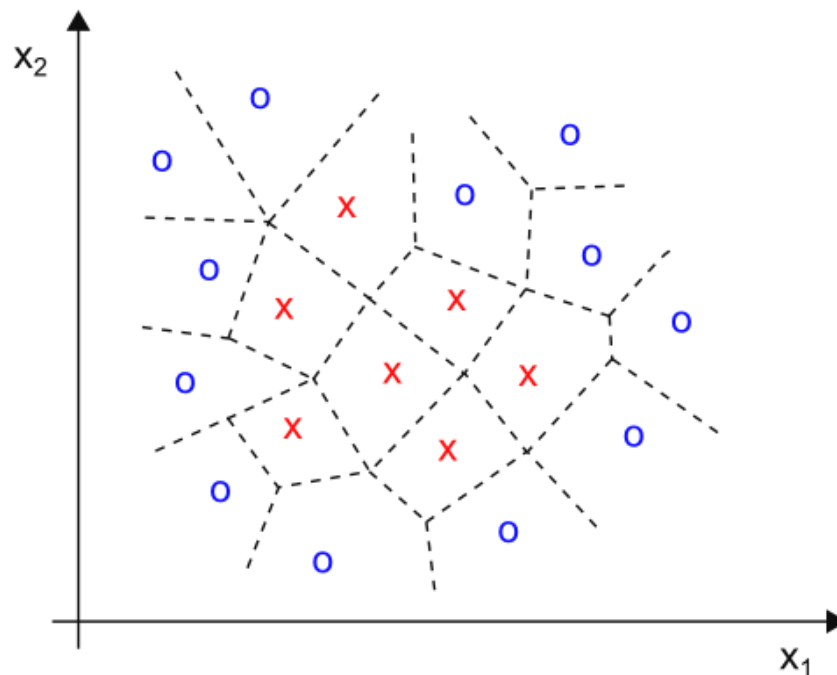
- Algoritmi kod kojih je broj parametara model fiksiran – t.j. ne zavisi od broja primjera u skupu za učenje (Linearna regresija)

### Ne-parametarski algoritmi

- Algoritmi kod kojih model zavisi (raste) s brojem primjera u skupu primjera za učenje – dakle ne može se komprimirati u fiksni skup parametara
- SVM – kombinacija NePar./Par. ?

# Reprezentacija modela: nelinearna klasifikacija

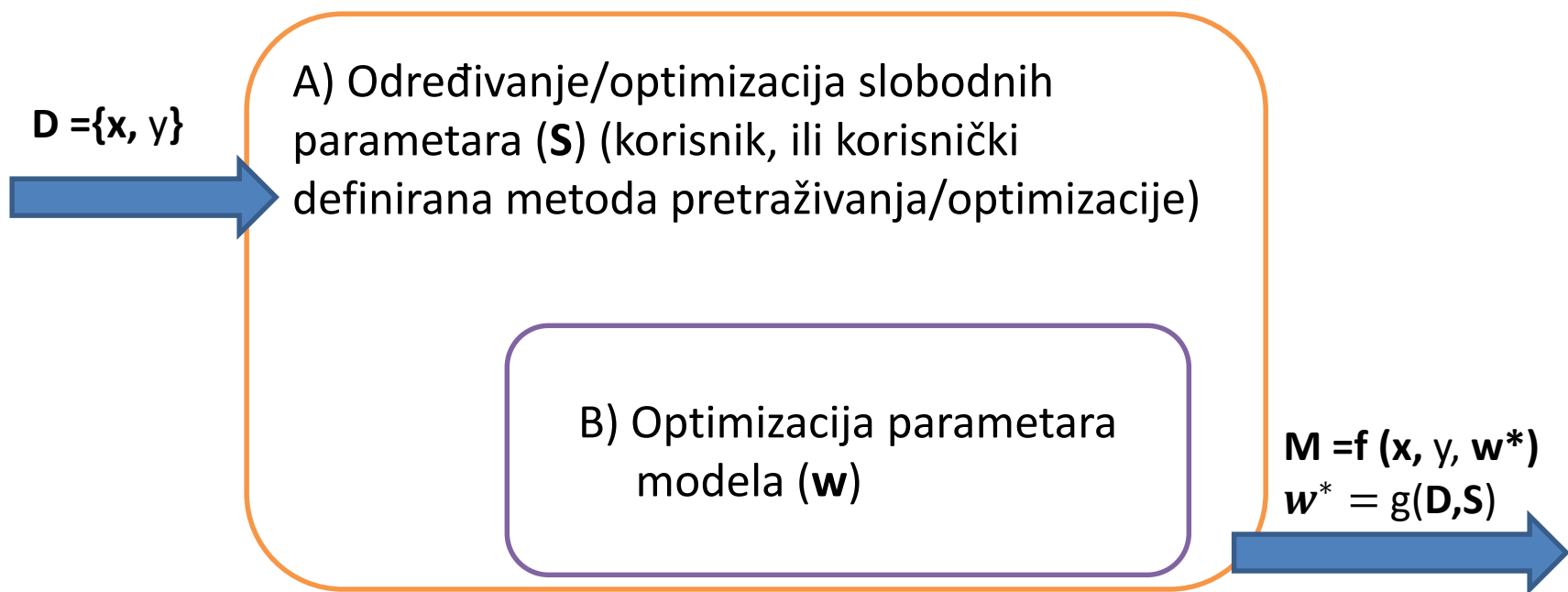
K najbližih susjeda (engl. *K nearest neighbors*, *k-NN*)



**ne-parametarski** algoritam učenja - broj parametara raste s obzirom na skup za učenje (skup za učenje je u stvari model!)

## Slobodni parametri algoritama

Većina algoritama SU ima slobodne parametre (korisnik ih zadaje/određuje prilikom učenja modela)




$w^* = g(D, S)$  – optimalni set parametara modela –  $w$  ovisi o podacima  $D$  i odabiru slobodnih parametara algoritma  $S$



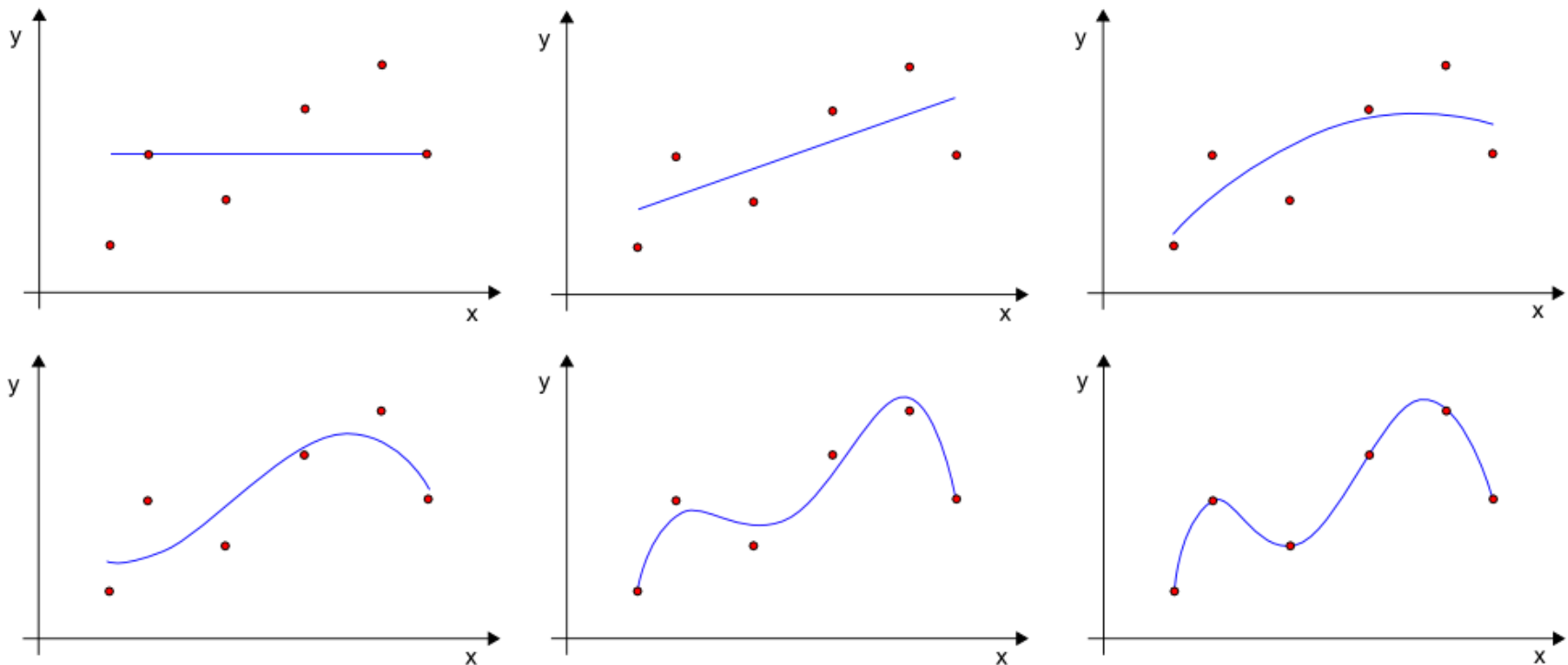
## Slobodni parametri tipično određuju kompleksnost modela/hipoteza

složenost modela


$$\begin{aligned}h_w(x) &= w_0 + w_1 x_1 \\h_w(x) &= w_0 + w_1 x_1 + w_2 x_1^2 \\&\vdots \\h_w(x) &= w_0 + \sum_{j=1}^n w_j x_1^j\end{aligned}$$

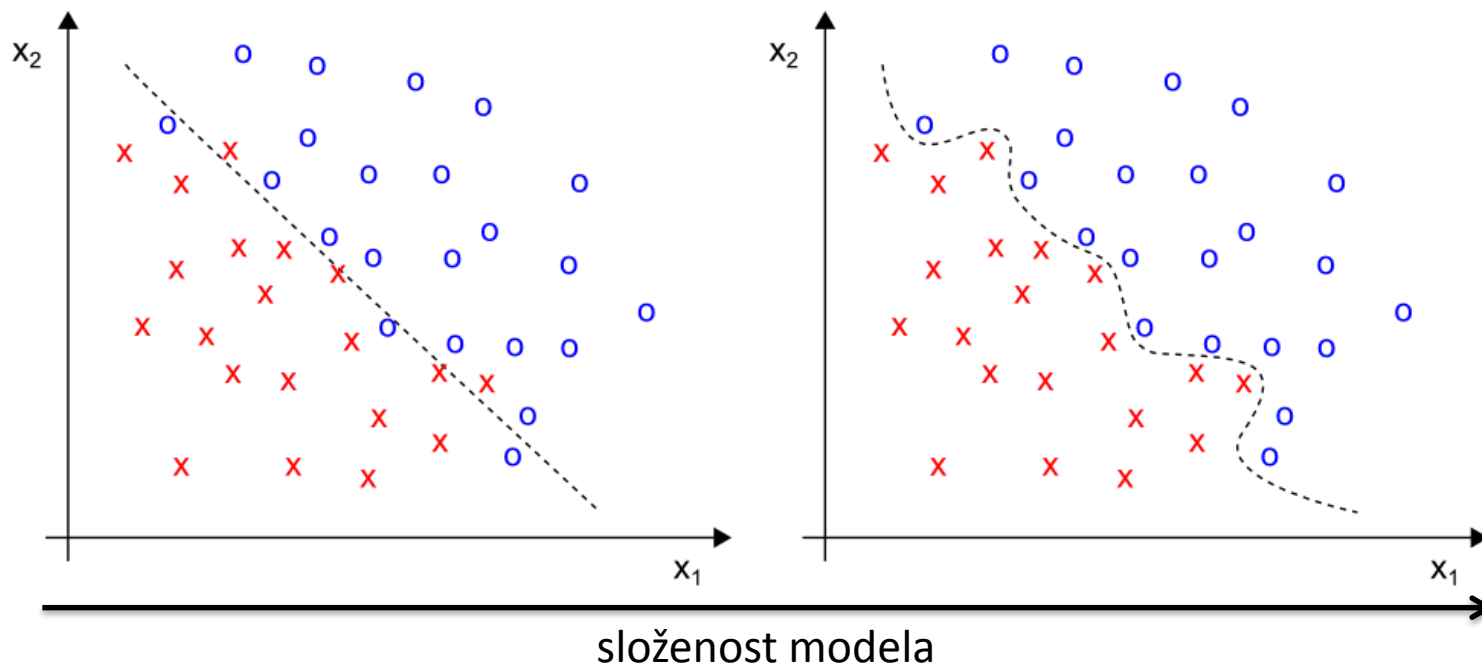
## Problem - Regresija

Slobodni parametri – određuju stupanj polinoma



## Problem - Klasifikacija

**Slobodni parametri – određuju složenost granične plohe  
(npr. broj čvorova u neuralnoj mreži)**



## Generativni i diskriminativni algoritmi

### Generativni algoritmi/modeli (Bayes modeli)

- Input i output varijable se modeliraju preko združene distribucije vjerojatnosti, koristeći uvjetne vjerojatnosti
- Modelira se združena vjerojatnost:

$$p(y, \mathbf{x}) = f(\mathbf{x}) ; \quad y \in \{C_k\}, k = 1, n - \text{oznaka klase}$$

- „generativni” – jer se modeli mogu koristiti za generiranje sintetičkih primjera/podataka – dodatni zadatak – generator = model podataka !
- Deskriptivni – „objašnjavaju” podatke
- Algoritmi: **Naivni Bayes, Mix Gauss, HMM, Bayesove mreže...**

## Taksonomija algoritama strojnog učenja

### Diskriminativni

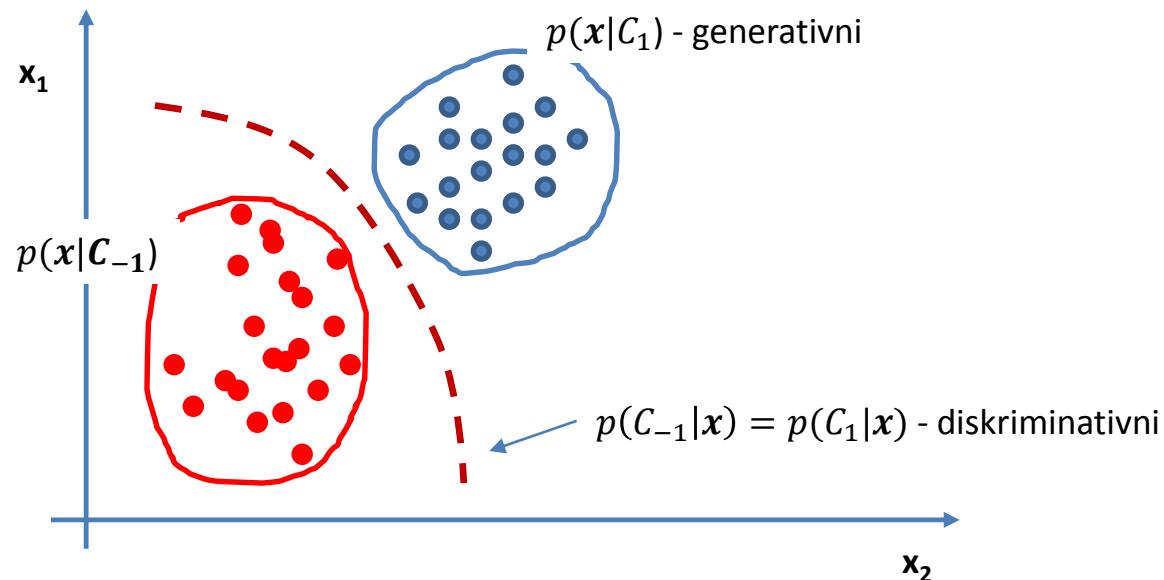
- Ne modeliraju vjerojatnosnu distribuciju varijabli, nego direktno određuju vrijednost ciljne variable, pomoću:
  - granične ploha koja razdvaja klase (klasifikacija)
  - funkcije koja aproksimira ponašanje ciljne varijable (regresija)
- Modelira se direktno aposteriori vjerojatnost:

$$p(C_k|\mathbf{x}) = f(\mathbf{x})$$

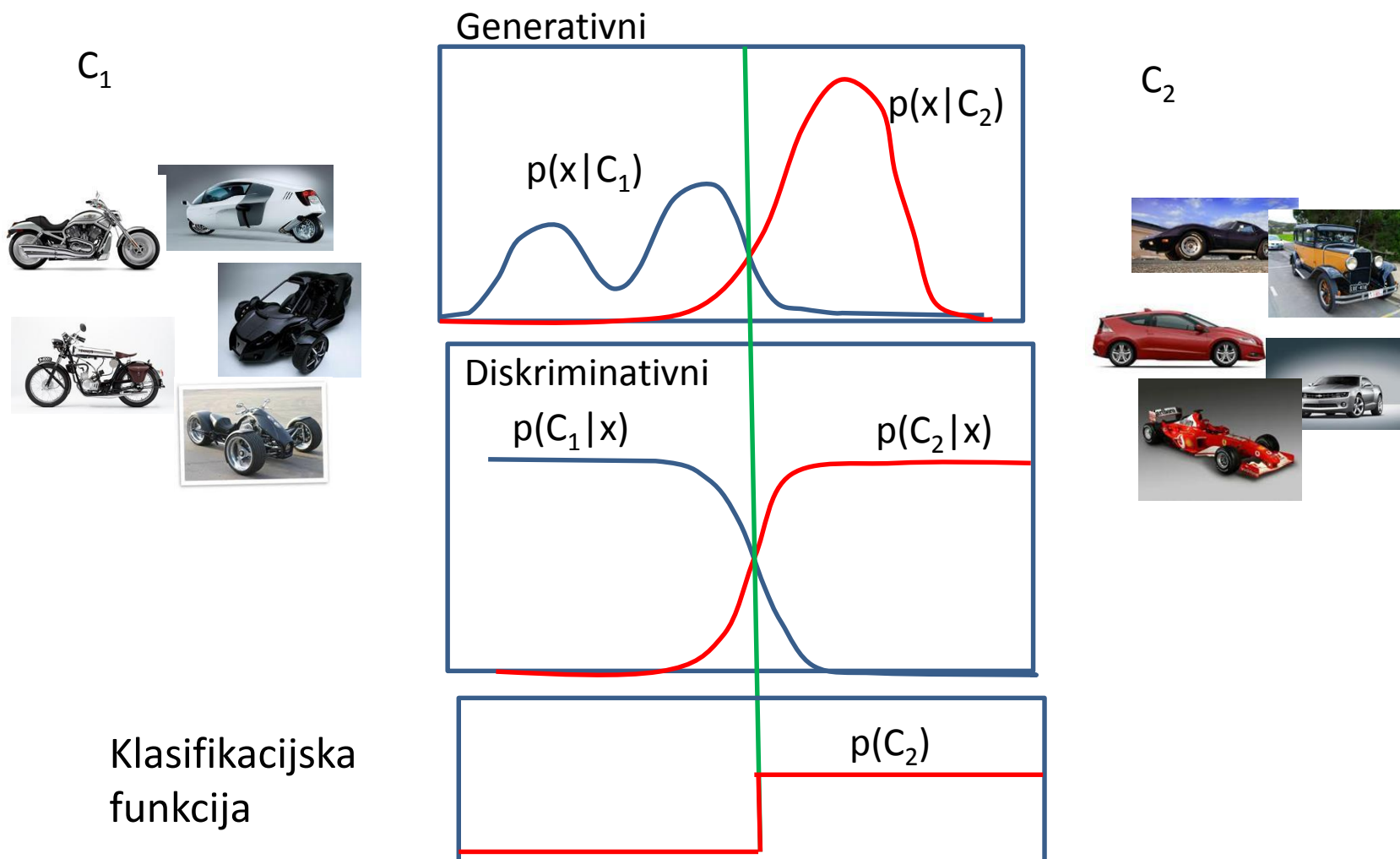
- Nema dodatnog zadatka (određivanje v. distribucije) !

**Algoritmi: NN-neuralne mreže, SVM-metoda potpornih vektora, Logistička regresija....**

## Generativni i diskriminativni modeli



## Generativni i diskriminativni modeli



## Generativni algoritmi/ modeli

Model pretpostavlja neku parametarsku formu distribucija vjerojatnosti, te algoritam određuje vrijednosti parametara na osnovu podataka

$$p(\mathbf{x}|C_k) = p(\mathbf{x}|Ck, \mathbf{w})$$

Parametre  $\mathbf{w}$  određujemo, pod pretpostavkom i.i.d. podataka maksimiziranjem (optimizacija!) vjerojatnosti  $p(\mathbf{x}|Ck, \mathbf{w})$  (**Maximum Likelihood - ML**), odnosno njenog logaritma

$$\log(p(\mathbf{x}|C_k, \mathbf{w})) \triangleq \sum_{i=1}^N \log p(\mathbf{x}_i | C_k, \mathbf{w})$$

ML procjena  $\mathbf{w}$  :

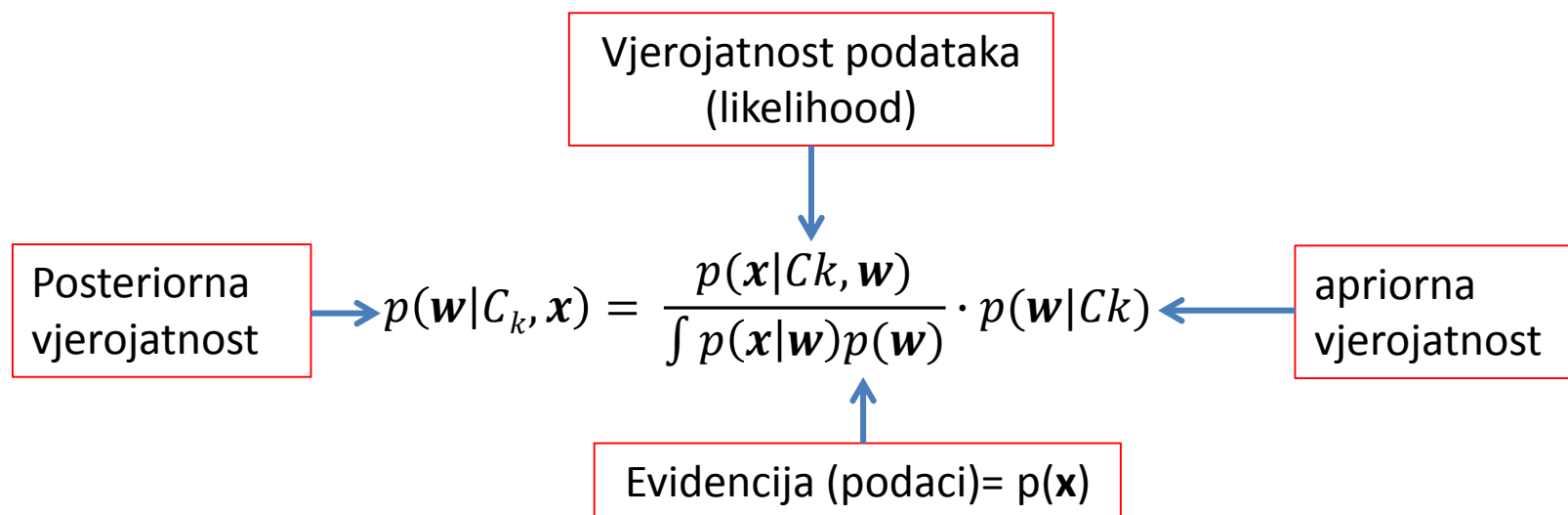
$$\hat{\mathbf{w}}_{ML} \triangleq \underset{\mathbf{w}}{\operatorname{argmax}} \log(p(\mathbf{x}|Ck, \mathbf{w}))$$

Podaci koje imamo  
- najvjerojatniji su uz  
parametre  $\hat{\mathbf{w}}_{ML}$



## Bayesovo pravilo za generativne algoritme

kolika je vjerojatnost parametara uz dane podatke ?



Posteriorna vjerojatnost je ono što nam treba:

- Kombinacija apriorne vjerojatnosti parametara za svaku klasu i vjerojatnosti podataka za tu klasu
- normalizirana sa podacima ( $\int$  poster. vj. podataka =1 !)

# **Metode optimizacije modela u algoritmima strojnog učenja**

Većina algoritama u SU predstavljaju određeni oblik optimizacije odnosno pretraživanja

- Učenje  $\sim$  Optimizacija
- optimiramo parametre modela koristeći *funkciju greške odnosno maksimalnu izvjesnost (max likelihood -MLE)*
- brža optimizacijska metoda => efikasnije učenje modela ...
- Vrlo često – funkcija greške uključuje regularizacijski dio

## A) Određivanje/optimizacija slobodnih parametara ( $\mathbf{S}$ )

- Korisnik zadaje  $\mathbf{S}$  (iskustveno)
- Pretraživanje (Grid search)
- Optimizacija (meta-heuristike: genetski algoritmi)

## B) Optimizacija/pretraživanje parametara modela ( $\mathbf{w}$ )

- Pretraživanje (greedy, hill-climbing, „pattern search”)
- Konveksni opt. problemi => Gradijentne metode
- Ograničenja => Lagrangeovi Multiplikatori
- Skaliranje => batch, online učenje

## Hill-climbing, greedy pretraživanje

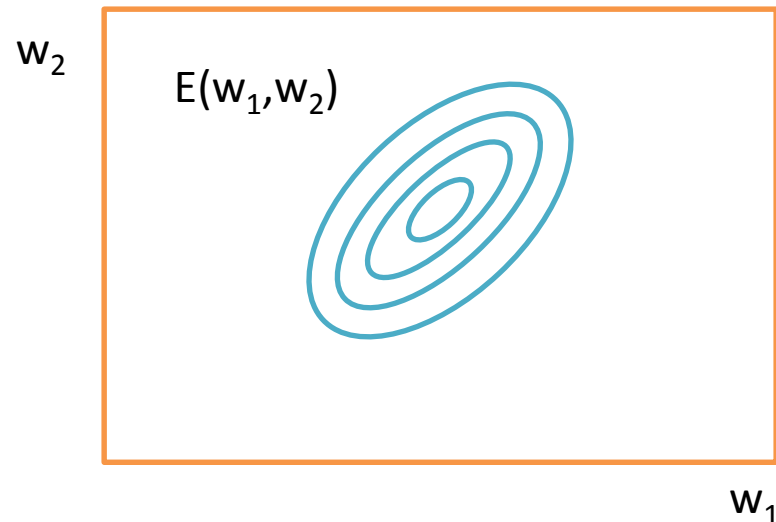
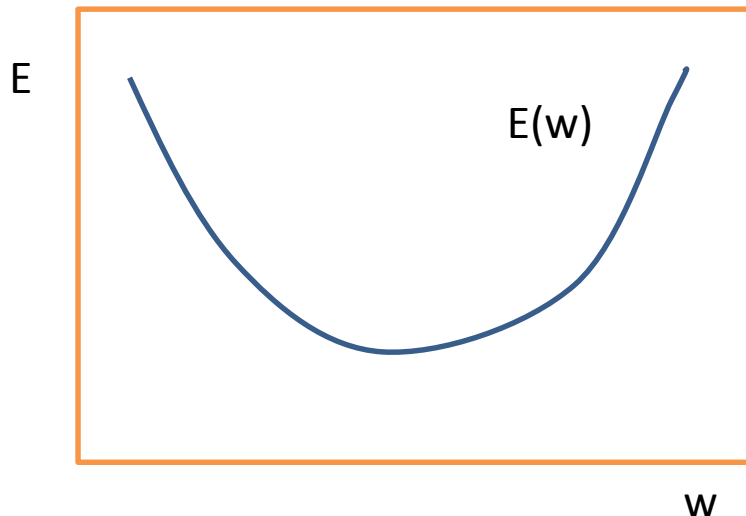
### Primjeri algoritama:

#### stabla odlučivanja (DT), učenje pravila (RL)

- Ovi algoritmi baziraju se na (rekurzivnom) razdvajanju ili „prekrivanju” primjera iz skupa za učenje
- U određenom trenutku (iteraciji) tražimo najbolji uvjet, ili pravilo koje:
  - a) Najbolje razdvaja dvije klase primjera (DT)
  - b) Najbolje pokriva određeni dio podataka neke klase (RL)
- To ne mora voditi prema najboljem modelu (uzimamo lokalno najbolje rješenje - pohlepno=**greedy**), no uvijek ide u smjeru smanjenja greške (do prvog vrha - **hill-climbing**)
- U principu konačni model/hipoteza s obzirom na grešku predstavlja lokalni optimum, ali je zato postupak efikasan.

## Funkcija greške – nad prostorom parametara

- Minimiziramo funkciju greške
- Ili - Maksimiziramo vjerojatnost podataka u generativnom modelu (Max Likelihood)
- Učenje je „kretanje po plohi” - funkcije greške –  $E_n$
- Ukupna funkcija greške modela je suma individualnih greški (za svaku točku trening skupa zasebno)



## Gradijent na površini funkcije greške

- Kako se micati po površini  $E(\mathbf{w})$  ? Ako mijenjamo  $w_k$  a sve ostalo držimo fiksnim da li se greška smanjuje ?
- $\frac{\partial E}{\partial w_k} \Rightarrow$  ako je  $E$  diferencijabilna – izračunati parcijalne derivacije za svaki  $w_k$ :
  - $\nabla E(\mathbf{w}) = \left( \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_d} \right)$
- Vektor parcijalnih derivacija = gradijent  $E(\mathbf{w})$ . **Negativni gradijent usmjeren je u pravcu najstrmijeg spusta (steepest gradient)** u prostoru  $\mathbf{w}$

Tri osnovna problema:

1. Kako efikasno izračunati gradijent
2. Kako minimizirati grešku – kad imamo gradijent
3. Gdje ćemo završiti na površini  $E(\mathbf{w})$

## Gradijent na površini funkcije greške – parcijalne derivacije

- Najstrmiji gradijent:

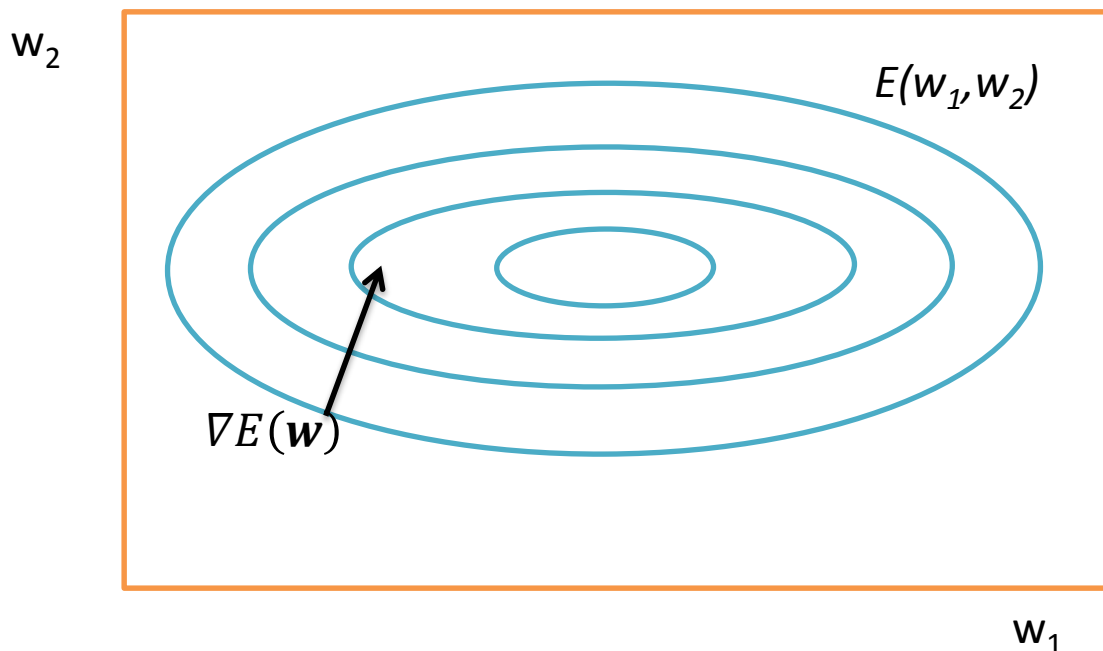
$$\mathbf{w}^{t+1} = \mathbf{w}^t - \epsilon \nabla E(\mathbf{w})$$

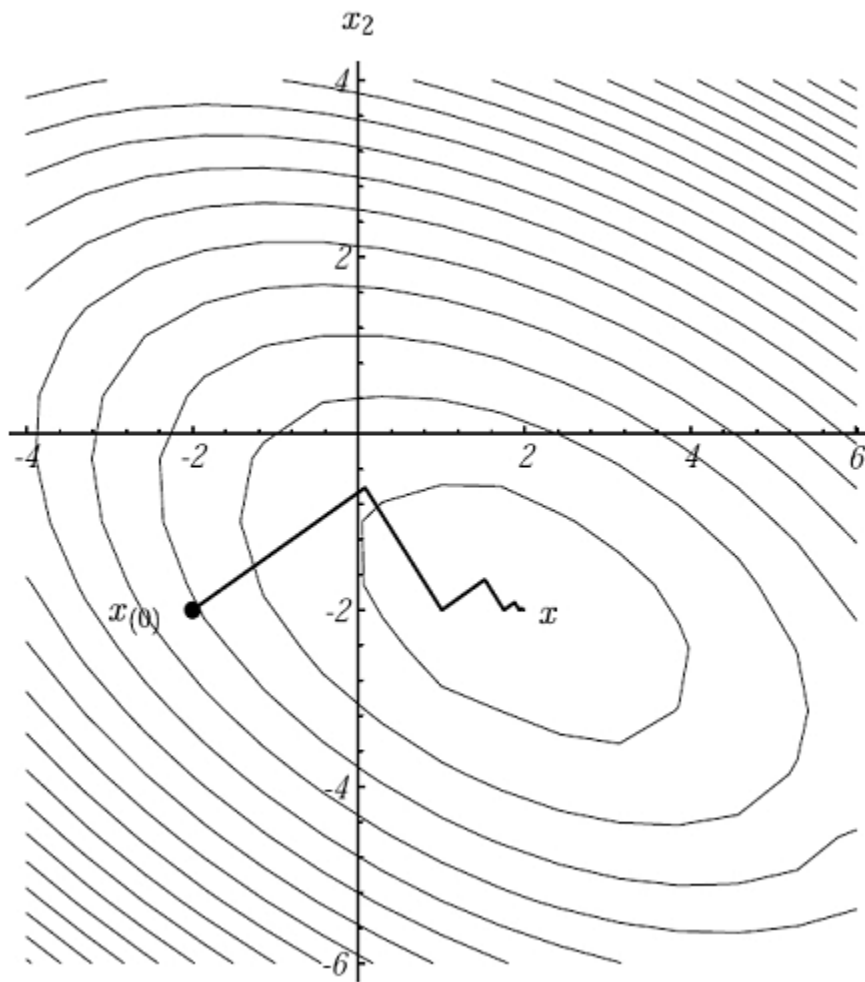
- Ako su koraci promjene  $\mathbf{w}$  dovoljno mali – postupak će konvergirati u minimum (barem lokalni)
- No, ako nas brine brzina konvergencije onda:
  - Veličina koraka  $\epsilon$  (slobodni parametar?) se mora pažljivo odrediti - za svaki problem!
  - Površina greške može biti zakrivljena sasvim različito za različite dimenzije  $w_k$  – to znači da gradijent ne pokazuje u smjeru najbližeg minimuma !



## Gradijent na površini funkcije greške – parcijalne derivacije

- Površina greške može biti zakrivljena sasvim različito za različite dimenzije  $w_k$  – to znači da gradijent ne pokazuje u smjeru najbližeg minimuma !





## Gradijentno spužanje

## Adaptacija koraka – gradijentno spužtanje

- Nema generalnog recepta za adaptaciju koraka
- „**Bold driver**” heuristika:
  - Nakon svakog pomaka (epohe – sumiranje greške preko cijelog skupa primjera za učenje):
    - Ako se greška smanjuje:
      - povećanje  $\epsilon \Rightarrow \epsilon = \epsilon \cdot \rho$
    - Ako se greška povećava:
      - smanjenje  $\epsilon \Rightarrow \epsilon = \epsilon \cdot \sigma$ ;
      - vraćanje prethodne vrijednosti parametara
$$\mathbf{w}^t = \mathbf{w}^{t-1}$$
- Tipične vrijednosti  $\rho \sim 1.1$ ;  $\sigma \sim 0.5$

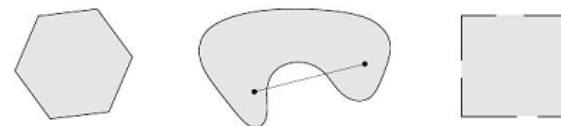
## Gradijentno spužtanje i konveksnost prostora

- Konveksnost – poželjno svojstvo
  - konveksni skup sadrži segment između bilo koje dvije točke u skupu



$$x_1, x_2 \in S \rightarrow (\lambda x_1 + (1 - \lambda)x_2) \in S;$$

gdje je  $\lambda \in [0,1]$



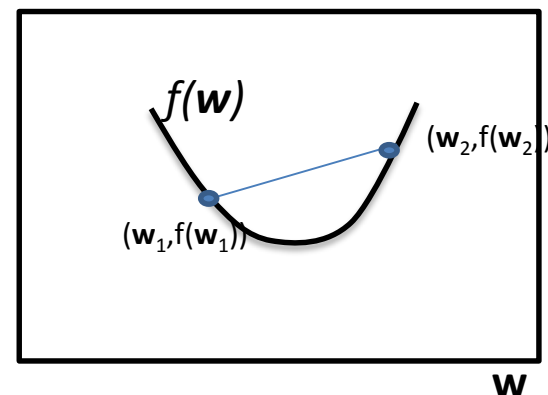
- Konveksna funkcija je ona za koju vrijedi



$f : \mathbb{R}^d \rightarrow \mathbb{R}$  je konveksna funkcija ako je njena domena konveksni skup za sve  $\lambda \in [0,1]$ :  
**(Jensenova nejednakost)**

$$f(\lambda w_1 + (1 - \lambda)w_2) \leq (\lambda f(w_1) + (1 - \lambda)f(w_2))$$

- Prostor funkcije greške za neke algoritme (linearna regresija, logistička regresija) je konveksan – samo jedan minimum (globalni)
- U mnogim algoritmima SU – nemamo konveksni optimizacijski problem !

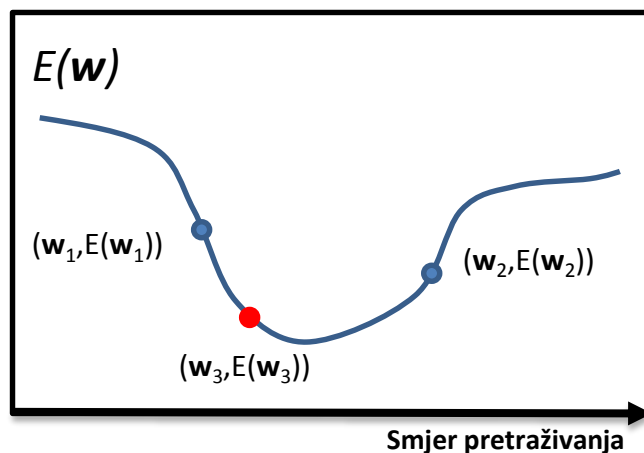


## Linijsko pretraživanje (Line search)

LP: Umjesto da radimo fiksne korake u smjeru negativnog gradijenta, radimo pretraživanje po tom smjeru dok ne nadjemo minimum (na toj liniji)

Tipično – bisekcija, kojom se traži treća točka koja  $w_3$  koja zadovoljava:

$$E(w_3) < E(w_2) \wedge E(w_3) < E(w_1)$$



## Gradijentno spuštanje - ubrzanje konvergencije

- Korištenje informacije o **zakrivljenosti prostora**
- Matrica parcijalnih derivacija drugog reda (Hessian matrix)

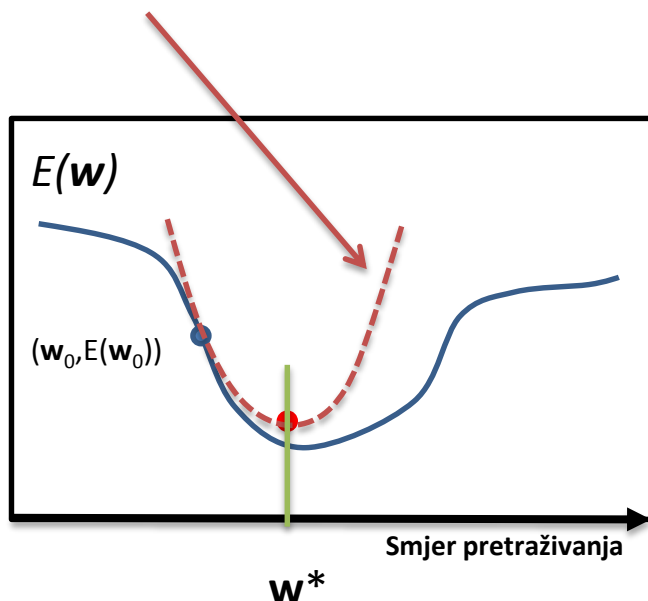
$$H_{ij} = \partial^2 E / \partial w_i \partial w_j$$

- Svojstveni vektori/vrijednosti  $H$  određuju smjer i veličinu zakrivljenosti u svakom (max korak  $\sim 2/\lambda_{max}$ )
- U praksi računanje Hessian m. Preskupo – **adaptacija koraka** !

## Lokalna kvadratna aproksimacija (Newton's method)

Taylorov red za vrijednost  $E$  oko neke točke  $\mathbf{w}_0$ :

$$E'(\mathbf{w} - \mathbf{w}_0) \approx E(\mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)^T \left( \frac{\partial E}{\partial \mathbf{w}} \right)_{\mathbf{w}_0} + (\mathbf{w} - \mathbf{w}_0)^T \frac{\mathbf{H}(\mathbf{w}_0)}{2} (\mathbf{w} - \mathbf{w}_0)$$



$$\mathbf{w}^* = \mathbf{w}_0 - \mathbf{H}^{-1}(\mathbf{w}) \left( \frac{\partial E}{\partial \mathbf{w}} \right)_{\mathbf{w}_0}$$

NM – „skok” u minimum lokalne kvadratne aproksimacije

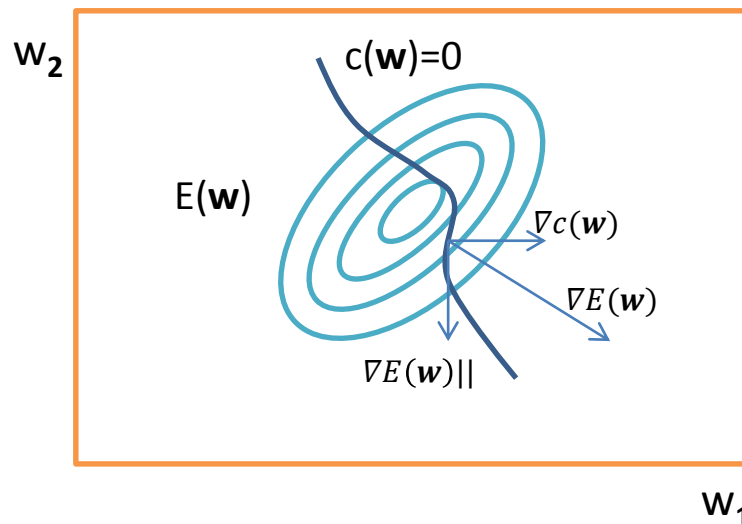
## Newtonova metoda i metode drugog reda

- Sve metode koje koriste *zakrivljenost* funkcije greške ili  $\mathbf{H}$  (Hessian) matri brže konvergiraju – ali mogu biti vrlo skupe (računanje i spremanje  $\mathbf{H}$  )
- Najčešći slučaj - kompromis: Nešto između metoda prvog i drugog reda !
- Npr. sekvenca gradijenata (prve derivacije) može poslužiti za aproksimiranje zakrivljenosti  $\mathbf{H}$  (tzv. metode s momentom) može biti i bolje od prave  $\mathbf{H}$ , jer možemo ograničiti aproksimaciju
- Kvazi Newtonove metode:
  - Metoda Konjugiranih Gradijenata (CG)
  - Broyden-Fletcher-Goldfarb-Shanno (BFGS)
  - Davidon-Fletcher-Powell (DVP)
  - Levenberg-Marquardt (LM)
- Sve aproksimiraju matricu  $\mathbf{H}$  i koriste je zajedno sa smjerom (gradijentom) uz kombiniranje sa nekom od metoda fiksnih-koraka, analitički određenih koraka ili linijskog pretraživanja.



## Što kada imamo ograničenja u funkciji greške (SVM)?

- Što znači ograničenje ?  
Npr. parametri koje optimiramo očekujemo da budu u nekom intervalu
  - To se može geom. predstaviti da leže **unutar ili na nekoj  $d$ -dimenzionalnoj plohi**
  - U tom slučaju da bi optimirali  $E(\mathbf{w})$  želimo ići po gradijentu koji istovremeno leži na toj plohi!
- 
- To zapravo znači da **gradijent mora biti okomit na normalu površine ograničenja**
  - gradijent -  $\nabla E(\mathbf{w})$  - komponenta  $\nabla E(\mathbf{w})$  paralelna gradijentu plohe ograničenja



## Što kada imamo ograničenja u funkciji greške (SVM)?

- U (ograničenom) optimumu -  $\nabla E(\mathbf{w})$  je paralelan gradijentu plohe ograničenja  $\nabla c(\mathbf{w})$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \lambda \frac{\partial c(\mathbf{w})}{\partial \mathbf{w}}$$

- Konstanta  $\lambda$  se naziva **Lagrangeov multiplikator**. Funkcija koja se optimira u slučaju kada minimiziramo funkciju greške uz ograničenja naziva se **Lagrangeovom funkcijom**

$$L(\mathbf{w}, \lambda) = E(\mathbf{w}) + \lambda^T c(\mathbf{w})$$

- Svojstvo LF je da kada je njen gradijent jednak 0, ograničenja su zadovoljena, a za  $\nabla E(\mathbf{w})$  vrijedi :

$$\frac{\partial L(\mathbf{w}, \lambda)}{\partial \mathbf{w}} = \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} + \lambda \frac{\partial c(\mathbf{w})}{\partial \mathbf{w}} = 0 \Rightarrow \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = -\lambda \frac{\partial c(\mathbf{w})}{\partial \mathbf{w}}$$

$$\frac{\partial L(\mathbf{w}, \lambda)}{\partial \lambda} = c(\mathbf{w}) = 0$$

## Što kada imamo velike količine podataka ?

- Sve ove metode pretpostavljaju izračunavanje gradijenta na cijelom skupu podataka....Izračunavanje gradijenta postaje vrlo skupo !
- Možemo li aproksimirati gradijent na jeftiniji način – a da zadržimo ugrubo smjer kretanja algoritma?

### Mini-batch pristup

- Podijelimo skup podataka u manje, mini-skupove (mini-batch), te računamo gradijent za takav manji skup, pa u slijedećoj iteraciji koristimo drugi skup, u slijedećoj iteraciji treći...

### SGD („online” pristup)

- Ako mini-skup postane jedan primjer – **online learning**
- Drugo ime - Stohastičko Gradjentno Spuštanje (**Stochastic Gradient Descent - SGD**)

Ove metode su vrlo brze u usporedbi sa metodama u kojima se gradijent računa egzaktno – no potrebno je dodatno kontrolirati konvergenciju.

## Gradijentne metode u učenju dubokih modela

Kontrola konvergencije gradijentnih metoda:

- a) adaptiranjem parametra brzine učenja  $\alpha$  (learning rate)
- b) adaptiranjem gradijenta  $\partial L / \partial \mathbf{w}$
- c) adaptiranjem  $\alpha$  i  $\partial L / \partial \mathbf{w}$  (tj a) i b))



Vanilla SGD

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$

Momentum

$$w_{t+1} = w_t - \alpha V_t$$

$$V_t = \beta V_{t-1} + (1 - \beta) \frac{\partial L}{\partial w_t}$$

Nesterov

$$w_{t+1} = w_t - \alpha V_t$$

$$V_t = \beta V_{t-1} + (1 - \beta) \frac{\partial L}{\partial w^*}$$

$$w^* = w_t - \alpha V_{t-1}$$

Adagrad

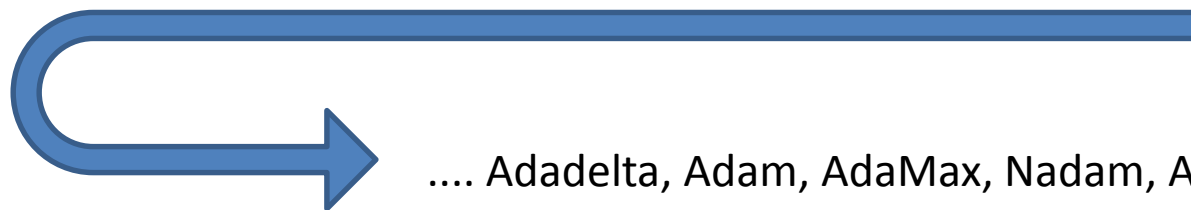
$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

$$S_t = S_{t-1} + \left[ \frac{\partial L}{\partial w_t} \right]^2$$

RMSprop

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

$$S_t = \beta S_{t-1} + (1 - \beta) \left[ \frac{\partial L}{\partial w_t} \right]^2$$



.... Adadelata, Adam, AdaMax, Nadam, AMSGrad....

## Literatura:

### Reprezentacija modela, struktura algoritama:

- [Lecture notes](#) 1,2,4,5; CS 229 Machine learning, Andrew Ng, Stanford
- [Machine learning class @ Coursera](#), Andrew Ng, Stanford
- Lecture 1: „Basic concepts in machine learning”, [Machine learning class @ Coursera](#), Pedro Domingos, Washington University

### Optimizacijske metode:

- Wikipedia
- S. Boyd, lecture notes , Convex Optimization II
- S. Ruder „An overview of gradient descent optimization algorithms”