Strojno učenje
3 (II dio)
Struktura metoda/algoritama strojnog učenja

Tomislav Šmuc

PMF, Zagreb, 2013

Sastavnice (nadziranog) problema učenja

Osnovni pojmovi

- Ulazni vektor varijabli (engl. *attributes, features*): $\mathbf{x} = (x_1, x_2, \ldots, x_d)$
  - 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*): $\mathrm{D} = \{(\mathbf{x_i}, y_i); i = 1 \ldots N\}$
  - Broj primjera za učenje: N
  - Dani podaci

- Nepoznata ciljna (idealna) funkcija (koncept): $f\colon \mathcal{X} \longrightarrow \mathcal{Y}$, $y = f(\mathbf{x})$

Sastavnice (nadziranog) problema učenja

- Cilj problema nadziranog učenja:
  - za dani skup primjera za učenje
  - „naučiti” funkciju h: $\mathcal{X} \longrightarrow \mathcal{Y}$ 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 smo mali broj diskretnih vrijednosti

- Hipoteza: h: $\mathcal{X} \longrightarrow \mathcal{Y},\ h \in \mathcal{H}$
- Skup, prostor hipoteza (klasa hipoteze): $\mathcal{H}$
  - npr. skup svih klasifikatora:
    $\mathcal{H} = \left\{h_{\mathrm{w}}: h_{\mathrm{w}}(\mathbf{x}) = 1\{\mathbf{w}^{\mathrm{T}}\mathbf{x} \geq 0\}, \mathbf{w} \in \mathbb{R}^{\mathrm{d}+1}\right\}$
  - Skup, prostor konzistentnih hipoteza: $VS_{\mathcal{H},D}$
    - Podskup hipoteza iz $\mathcal{H}$ konzistentnih s primjerima za učenje iz D
    - $VS_{\mathcal{H},D} = \{h \in \mathcal{H} | konzistentna(h|D)\}$

Skup hipoteza

- Veličina skupa hipoteza:
  - fiksna
  - varijabilna
- Parametrizacija hipoteze – opis hipoteze:
  - skupom simboličkih (diskretnih) varijabli
  - skupom kontinuiranih parametara
- Evaluacija hipoteze
  - 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 i klasifikacija

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

$y \in \mathbb{R}$

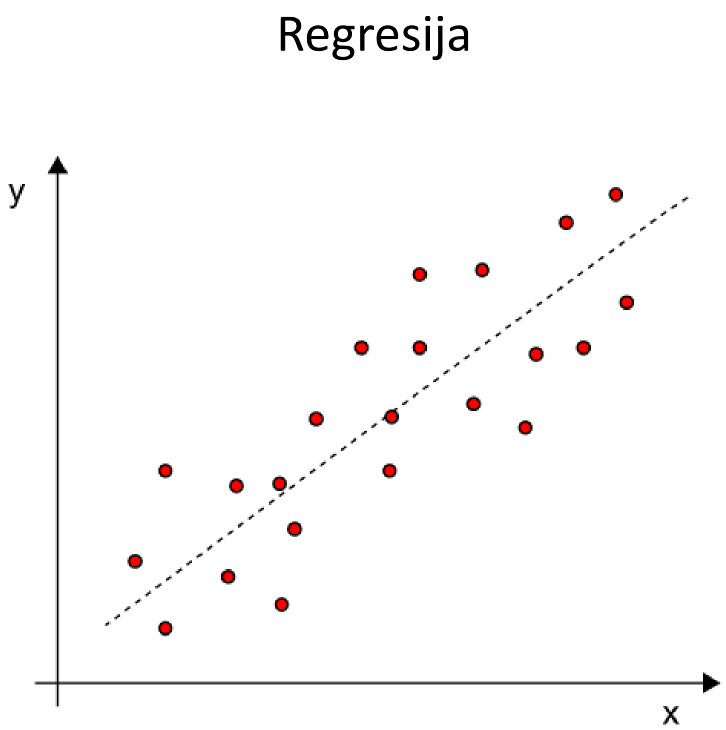
Razlika je u obliku ciljne varijable!

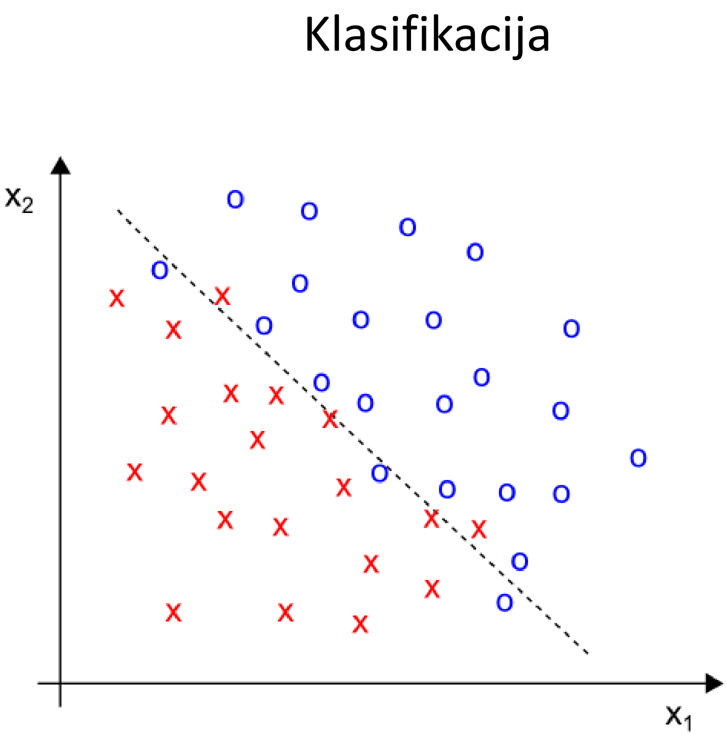
Klasifikacija

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

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

Regresija i klasifikacija

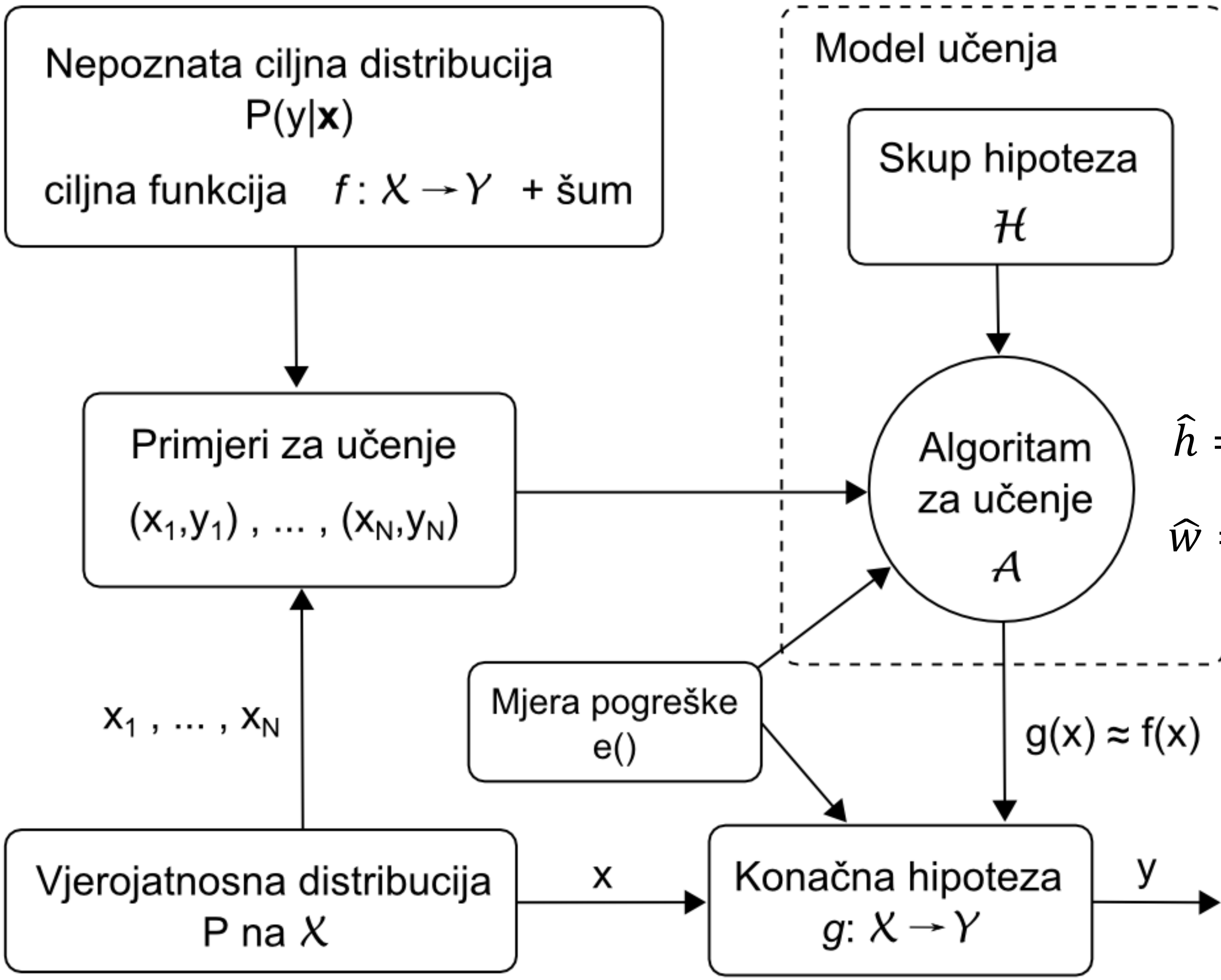




Algoritam učenja

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

Dijagram učenja



$$\hat{h} = \underset{h \in \mathcal{H}}{\arg\min}\, J(h)$$

$$\hat{w} = \underset{w}{\arg\min}\, J(h_w(\mathbf{x}), y)$$

Reprezentacija modela za učenje: primjer

- Hipoteza:

$$h_w(x) = w_0 + w_1 x_1$$

- Parametri:

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

- Mjera greške ili funkcija troška:

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

- Cilj:

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

optimizacijski problem => algoritmi optimizacije

Reprezentacija modela: linearni modeli

regresija

Linearna regresija (engl. *linear regression*)

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

klasifikacija

Logistička regresija (engl. logistic regression)

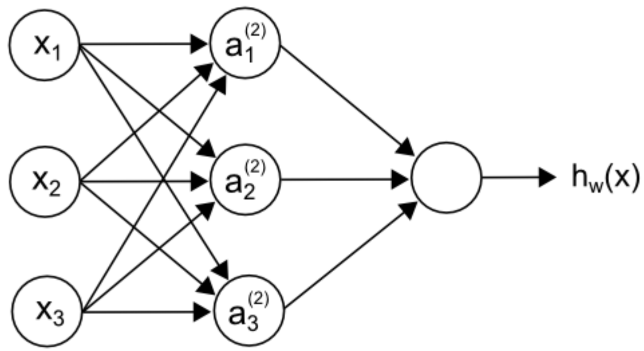
Želimo: $h_{\mathrm{w}}(\mathbf{x}) \in [0,1]$ za **binarnu klasifikaciju**: $y \in \{0,1\}$

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

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

Reprezentacija modela: nelinearna klasifikacija

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



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

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

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

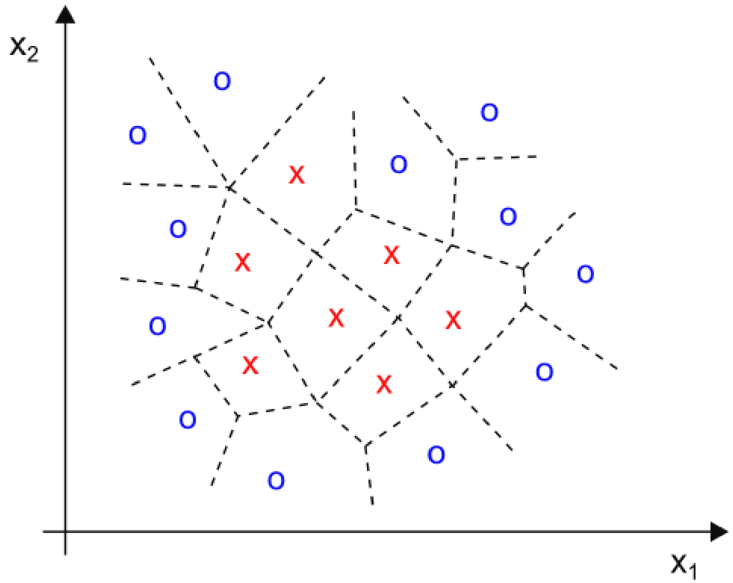
$$h_w(x) = a_1^{(3)} = g(w_{10}^{(2)} a_0^{(2)} + w_{11}^{(2)} a_1^{(2)} + w_{12}^{(2)} a_2^{(2)} + w_{13}^{(2)} a_3^{(2)})$$

Tipično *g(x)*:

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

Reprezentacija modela: nelinearna klasifikacija

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

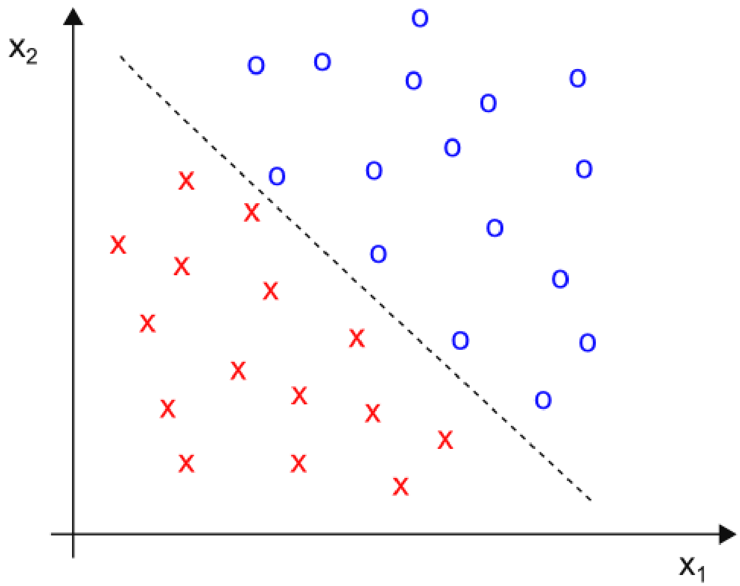


primjer s 1-NN

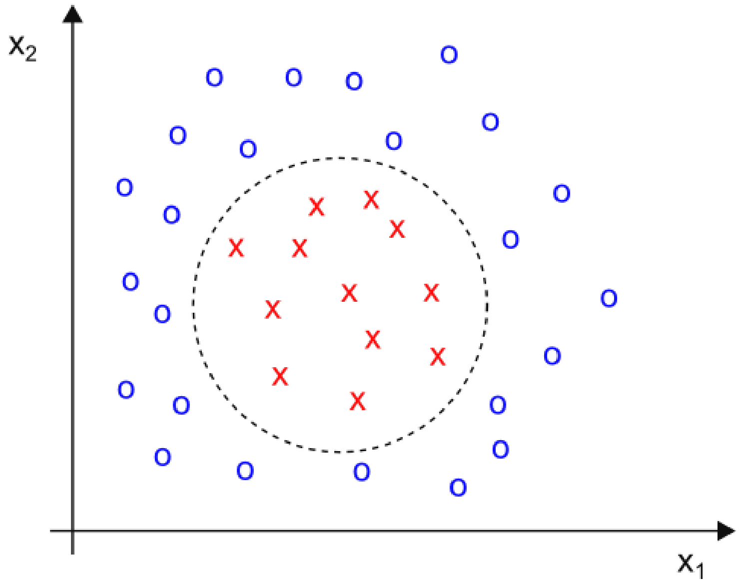
ne-parametarski algoritam učenja

Oblici hipoteza: klasifikacija

Linearni model

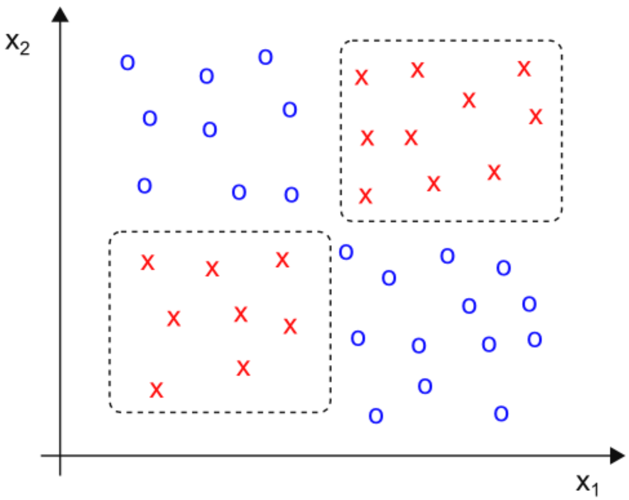


Nelinearni model



Reprezentacija modela: klasifikacija

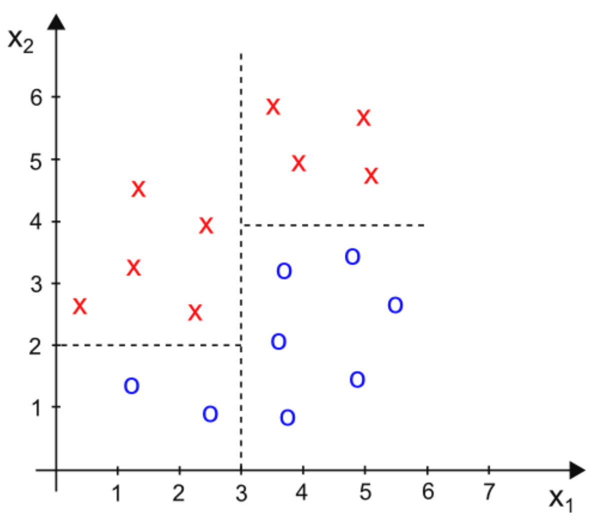
Pravila (engl. *Rules*)

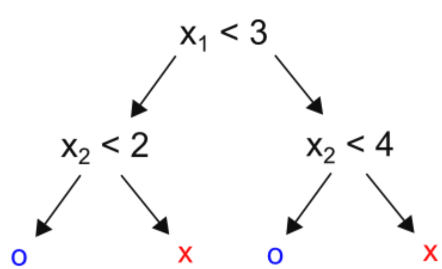


IF (x1>0.5) AND (x1<4.5) AND (x2>0.5) AND (x2<4.0) THEN y = x

IF (x1>5.0) AND (x1<7.0) AND (x2>4.5) AND (x2<6.5) THEN y = x

Stablo odluke (engl. *Decision tree*)





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

Parametarski algoritmi

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

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 (primjer => k-NN)

- SVM ?

Slobodni parametri algoritama

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

D ={x, y**}**

A) Određivanje/optimizacija slobodnih parametara (**S**) (korisnik, ili korisnički definirana metoda pretraživanja/optimizacije)

B) Optimizacija parametara modela (**w**)

M =f (x, y, **w*)**

$\boldsymbol{w}^* = g(\mathbf{D},\mathbf{S})$

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

Slobodni parametri tipično određuju kompleksnost modela/hipoteza

složenost modela

$$h_w(x) = w_0 + w_1 x_1$$

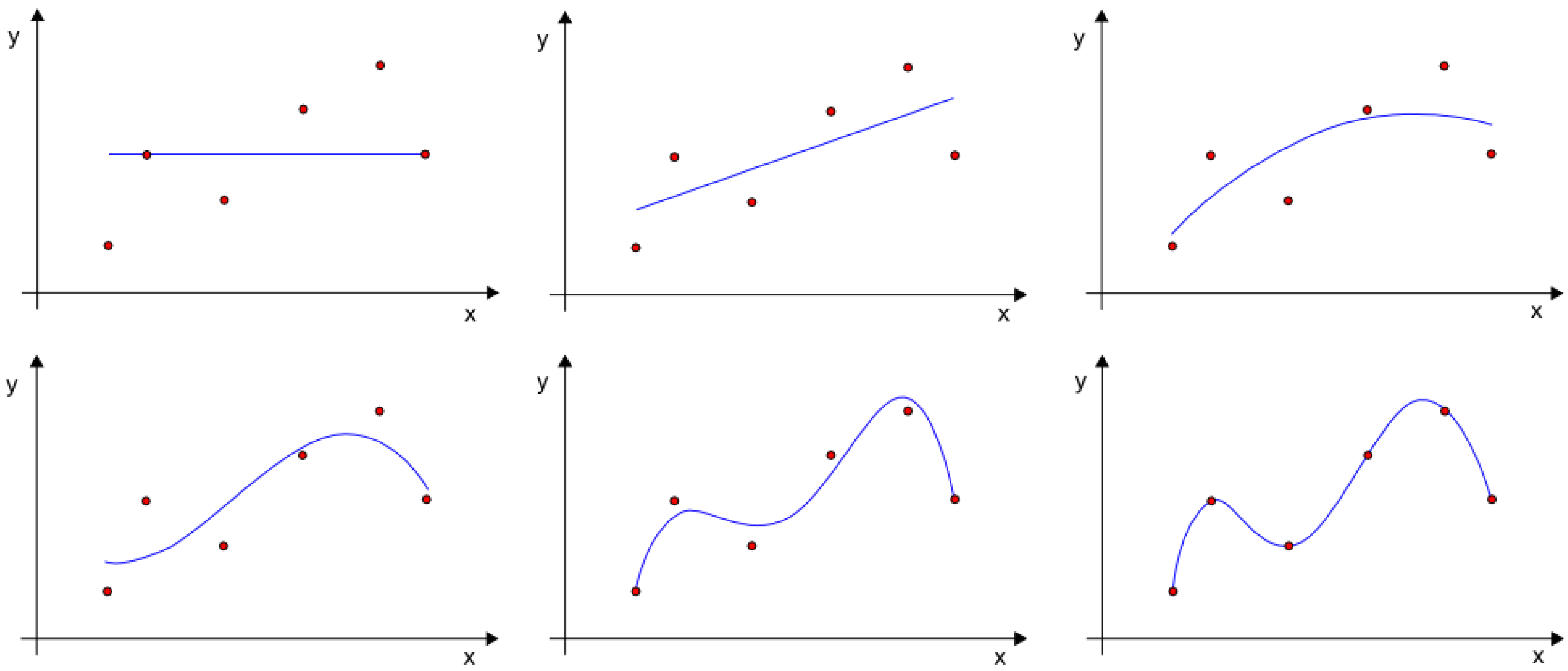
$$h_w(x) = w_0 + w_1 x_1 + w_2 {x_1}^2$$

$$\vdots$$

$$h_w(x) = w_0 + \sum_{j=1}^{n} w_j {x_1}^j$$

Problem - Regresija

Slobodni parametri – određuju stupanj polinoma

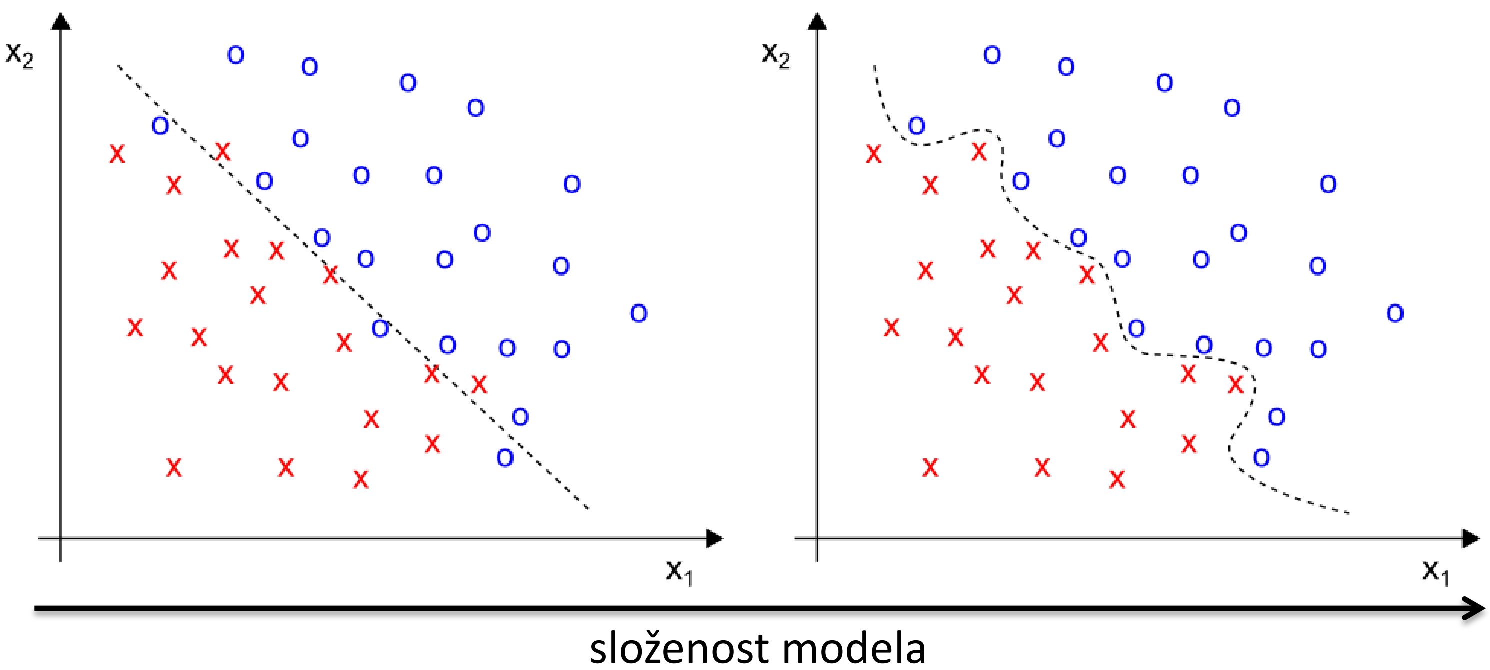


http://arachnoid.com/polysolve/index.html

Struktura | Slobodni parametri algoritama

Problem - Klasifikacija

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



Generativni i diskriminativni algoritmi

Generativni algoritmi/modeli (Bayes modeli)

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

$$p(C, \boldsymbol{x}) = f(\boldsymbol{x})\ ; \qquad C \text{ – oznaka klase}$$

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

Taksonomija algoritama strojnog učenja

Diskriminativni

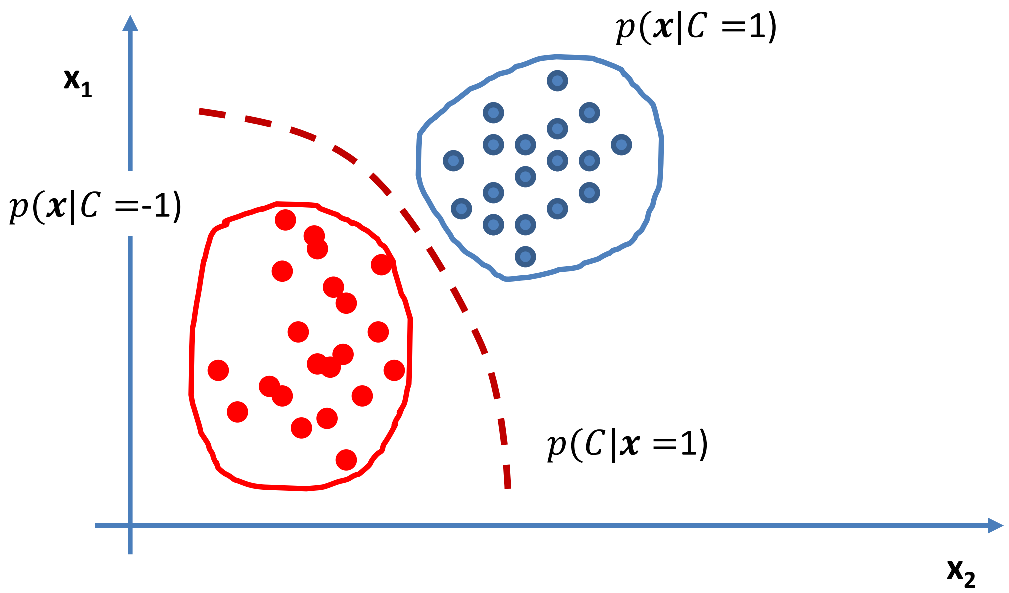
- Ne modeliraju eksplicitno distribucije varijabli, nego direktno mapiraju ulazne varijable na ciljnu varijablu
- (granična ploha koja razdvaja klase, ili funkcija koja aproksimira ponašanje ciljne varijable – regresija)
- Modelira se direktno aposteriori vjerojatnost:

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

- Nema dodatnog zadatka - „Black-box” model !

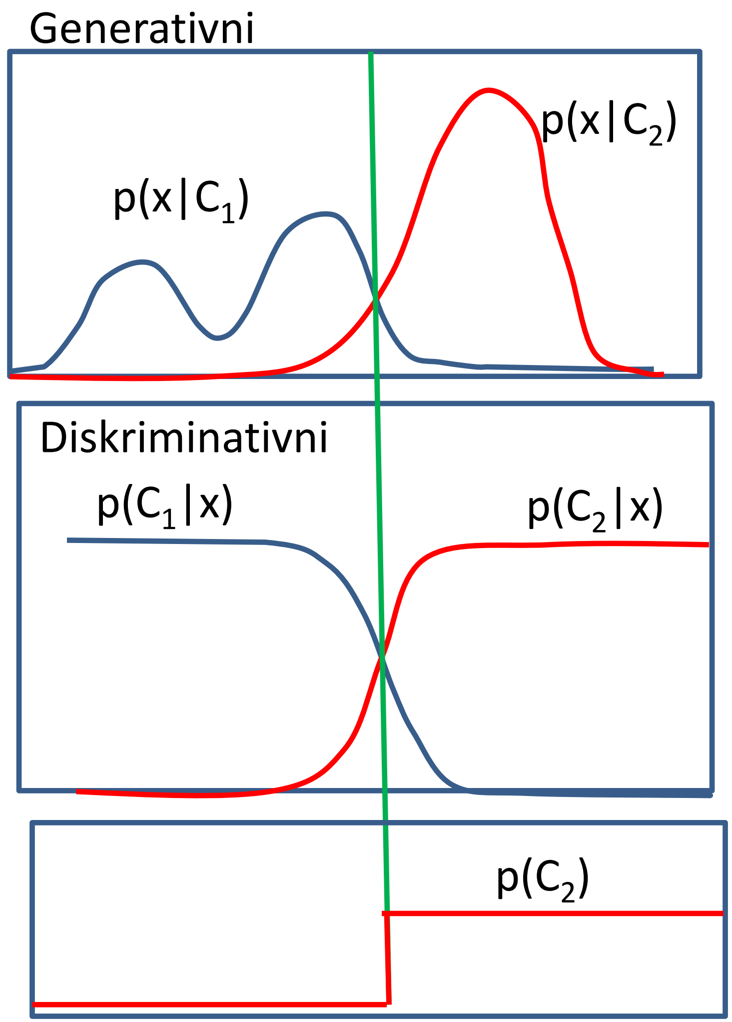
Algoritmi: NN-neuralne mreže, SVM-metoda potpornih vektora, Logistička regresija….

Generativni i diskriminativni modeli



Generativni i diskriminativni modeli

C_1



C_2

Klasifikacijska funkcija

Generativni algoritmi/ modeli

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

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

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

ML procjena **w** :

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

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

Podaci koje imamo
- najvjerojatniji su uz parametre $\widehat{\boldsymbol{w}}_{ML}$

Bayesovo pravilo za generativne algoritme

Izvođenje generativnih modela radi njihove usporedbe – „unatrag” – kolika je vjerojatnost parametara uz dane podatke ?

Vjerojatnost podataka (likelihood)

Posteriorna vjerojatnost

$$p(\boldsymbol{w}|C_k,\boldsymbol{x}) = \frac{p(\boldsymbol{x}|Ck,\boldsymbol{w})}{\int p(\boldsymbol{x}|\boldsymbol{w})p(\boldsymbol{w})} \cdot p(\boldsymbol{w}|Ck)$$

apriorna vjerojatnost

Evidencija (podaci)= p(**x**)

- Posteriorna vjerojatnost je ono što nam treba:
  - Kombinacija apriorne vjerojatnosti parametara za svaku klasu sa vjerojatnosti podataka za tu klasu
  - normaliziranom sa podacima (∫ poster. vj. podataka =1 !)

Metode optimizacije modela u algoritmima strojnog učenja

Većina algoritama u SU predstavljaju određeni oblik optimizacije odnosno pretraživanja

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

A) Određivanje/optimizacija slobodnih parametara (**S**)
- Korisnik zadaje **S** (iskustveno)
- Pretraživanje (Grid search)
- Optimizacija (meta-heuristike: genetski algoritmi)

B) Optimizacija/pretraživanje parametara modela (**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:

<u>stabla odlučivanja (DT), učenje pravila (RL)</u>

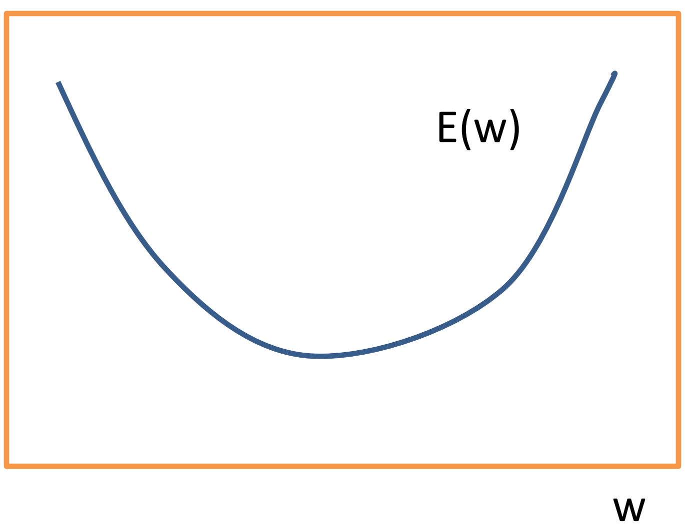
- Ovi algoritmi baziraju se na razdvajanje ili prekrivanje primjera iz skupa za učenje
- U određenom trenutku (iteraciji) tražimo najbolji uvjet, ili pravilo koje:

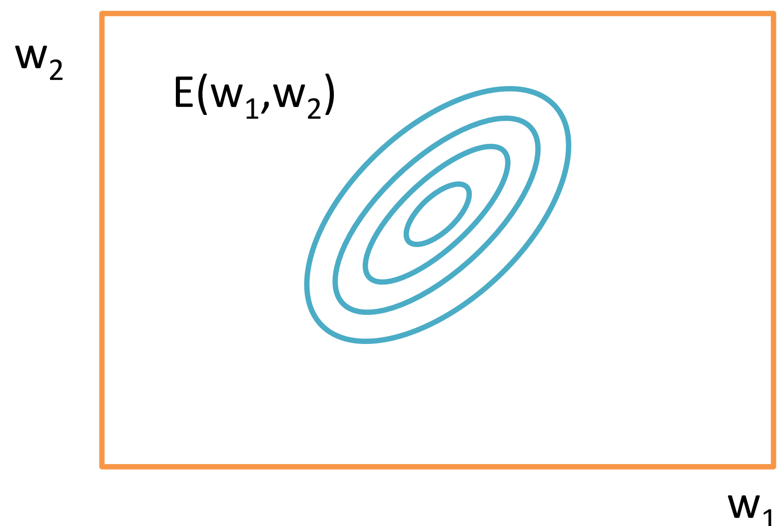
a) Najbolje razdvaja dvije klase primjera (DT)
b) Najbolje pokriva određeni dio podataka neke klase (RL)

- 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
- ! Ako su podaci i.i.d. – funkcija greške je suma individualnih funkcija greške (za svaku točku trening skupa zasebno (E_n) !





Gradijent na površini funkcije greške

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

Tri osnovna problema:

1. Kako efikasno izračunati gradijent
2. Kako minimizirati grešku – kad imamo gradijent
3. Gdje ćemo završiti na površini *E(**w**)*

Gradijent na površini funkcije greške – parcijalne derivacije

- Najstrmiji gradijent:

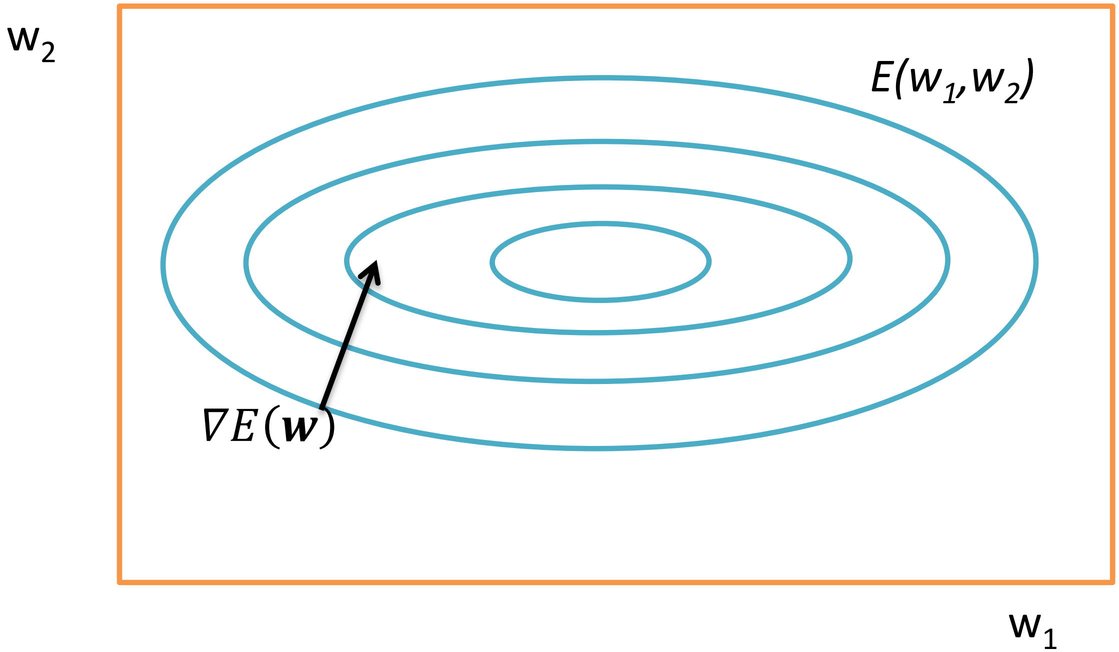
$$\boldsymbol{w}^{\boldsymbol{t+1}} = \boldsymbol{w}^{\boldsymbol{t}} - \epsilon \nabla E(\boldsymbol{w})$$

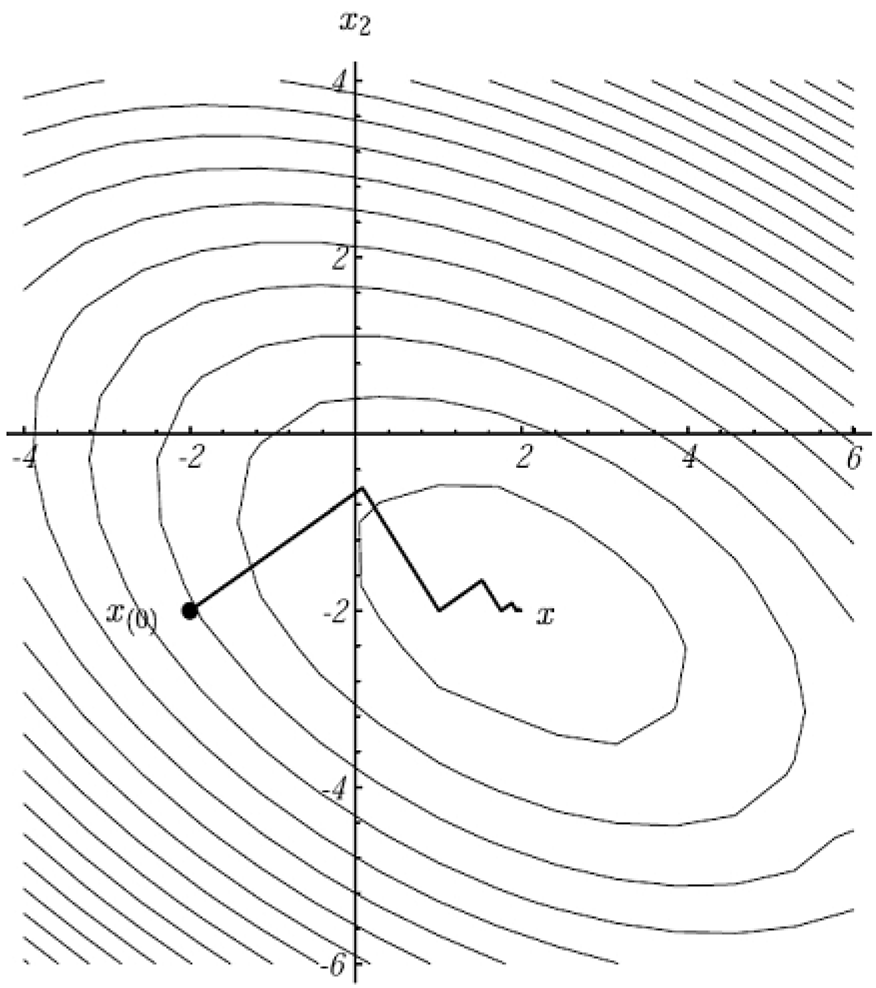
- Ako su koraci promjene ***w*** dovoljno mali – postupak će konvergirati u minimum (barem lokalni)

- No, ako nas brine brzina konvergencije onda:
  - Veličina koraka ϵ (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 spuštanje

Adaptacija koraka – gradijentno spuštanje

- Nema generalnog recepta za adaptaciju koraka
- „**Bold driver**" heuristika:
  - Nakon svakog pomaka (epohe – prolaz-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

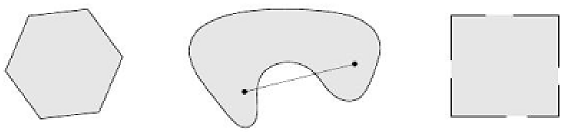
$$\boldsymbol{w}^{t} = \boldsymbol{w}^{t-1}$$

  - Tipične vrijednosti $\rho{\sim}1.1;\ \sigma{\sim}0.5$

Gradijentno spuštanje 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 $\lambda \in [0,1]$

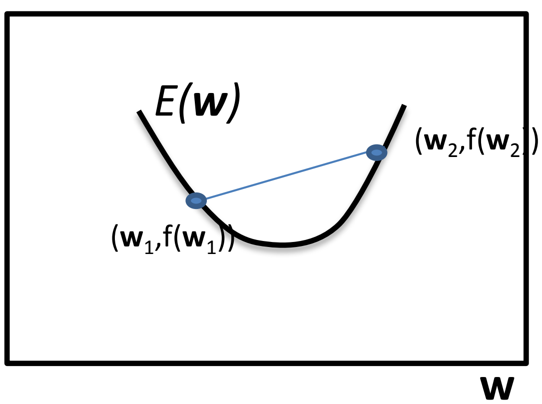


  - Konveksna funkcija je ona za koju vrijedi

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

$$E(\lambda \boldsymbol{w}_1 + (1-\lambda)\boldsymbol{w}_2) \leq E(\lambda f(\boldsymbol{w}_1) + (1-\lambda) f(\boldsymbol{w}_2)$$

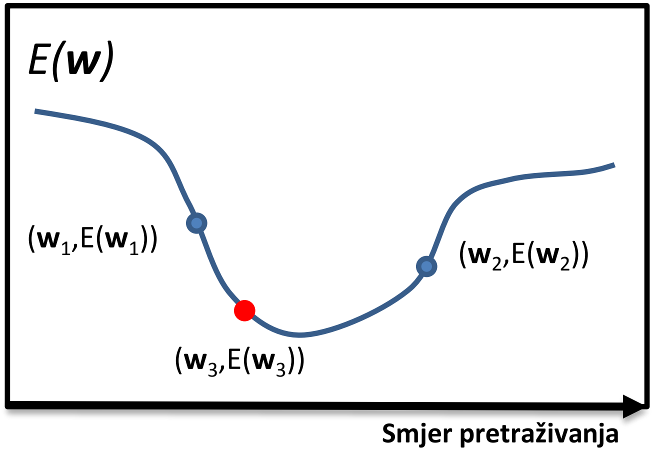
- Prostor funkcije greške za neke algoritme (linearna regresija, logistička regresija) je konveksan – samo jedan minimum (globalni)
- U većini slučajeva u alg. SU – nemamo konveksni optimizacijski problem !



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:

$$E(\boldsymbol{w}_3) < E(\boldsymbol{w}_2) \ \wedge E(\boldsymbol{w}_3) < E(\boldsymbol{w}_1)$$



Gradijentno spuštanje - ubrzanje konvergencije

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

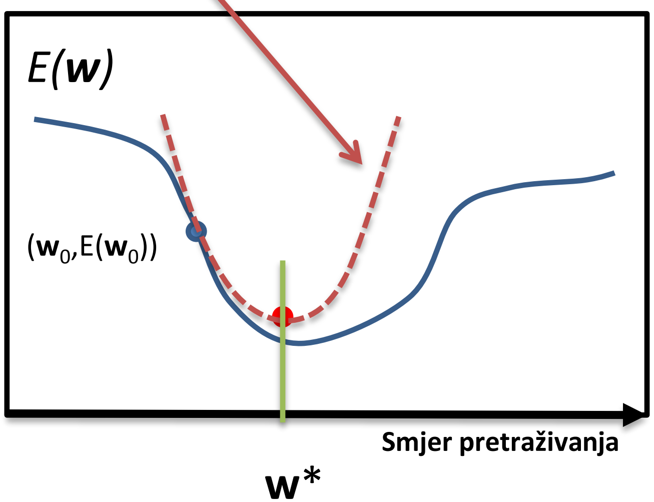
$$H_{ij} = \partial^2 E / \partial w_i w_j$$

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

Lokalna kvadratna aproksimacija (Newton's method)

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

$$E'(\boldsymbol{w}-\boldsymbol{w}_0) \approx E(\boldsymbol{w}_0) + (\boldsymbol{w}-\boldsymbol{w}_0)^T \left(\frac{\partial E}{\partial \boldsymbol{w}}\right)_{w_0} + (\boldsymbol{w}-\boldsymbol{w}_0)^T \frac{\boldsymbol{H}(\boldsymbol{w}_0)}{2}(\boldsymbol{w}-\boldsymbol{w}_0)$$



$$\boldsymbol{w}^* = w_0 - \boldsymbol{H}^{-1}(\boldsymbol{w}) \left(\frac{\partial E}{\partial \boldsymbol{w}}\right)_{w_0}$$

NM – „skok” u minimum lokalne kvadratne aproksimacije

Newtonova metoda i metode drugog reda

- Sve metode koje koriste *zakrivljenost* funkcije greške ili **H** (Hessian) matricu
- Brže konvergiraju – ali mogu biti vrlo skupo (računanje i spremanje **H**)
- Najčešći slučaj - kompromis: Nešto između metoda prvog i drugog reda !
- Npr. sekvenca gradijenata (prve derivacije) može poslužiti za aproksimiranje zakrivljenosti (**H**) – može biti i bolje od prave **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 **H** i koriste je zajedno sa smjerom (gradijentom) uz kombiniranje sa nekom od metoda fiksnih-koraka, analitički određenih koraka ili linijskog pretraživanja.

Metoda konjugiranih (vezanih) gradijenata

- Iskustveno pravilo: na kraju iteracije jednog linijskog pretraživanja, novi gradijent ($\partial E/\partial \boldsymbol{w}$) je približno ortogonalan smjeru prethodnog linijskog pretraživanja
- Ako odredimo novi gradijent – uvijek ćemo se kretati u ortogonalnim smjerovima ….

- Stoga odredimo novi smjer tako da gradijent paralelan sa starim smjerom bude 0! To praktično znači da trenutni negativni gradijent treba kombinirati sa dosadašnjim smjerom ***s(t)***:

$$\boldsymbol{s}(t+1) = -\boldsymbol{\nabla}_{\boldsymbol{w}} E(t+1) + \beta(t)\boldsymbol{s}(t)$$

što za sobom povlači

$$\boldsymbol{\nabla}_{\boldsymbol{w}} E(t)\boldsymbol{s}(t) = 0$$

- Različite varijante određivanja $\beta(t)$ i varijante CG metode:
  - Fletcher-Reeves, Polak-Ribiere, Hestenes-Stiefel…

BFGS Metoda

- Kvazi-Newtonova metoda
- Iterativno aproksimiranje **H**

$$\boldsymbol{s}(t) = -\boldsymbol{g}(t) + \boldsymbol{A}(t)\Delta\boldsymbol{w}(t) + \boldsymbol{B}(t)\Delta\boldsymbol{g}(t)$$

gdje su:

$$\boldsymbol{g}(t) = \boldsymbol{\nabla}_{\boldsymbol{w}} E(t); \Delta\boldsymbol{g}(t) = \boldsymbol{g}(t) - \boldsymbol{g}(t-1)$$

$$\Delta\boldsymbol{w}(t) = \boldsymbol{w}(t) - \boldsymbol{w}(t-1)$$

$$\boldsymbol{A}(t) = -1\left(1 + \frac{\Delta\boldsymbol{g}(t)^T\Delta\boldsymbol{g}(t)}{\Delta\boldsymbol{w}(t)^T\Delta\boldsymbol{g}(t)}\right)\cdot\frac{\Delta\boldsymbol{w}(t)^T\boldsymbol{g}(t)}{\Delta\boldsymbol{w}(t)^T\Delta\boldsymbol{g}(t)} + \frac{\Delta\boldsymbol{g}(t)^T\boldsymbol{g}(t)}{\Delta\boldsymbol{w}(t)^T\Delta\boldsymbol{g}(t)}$$

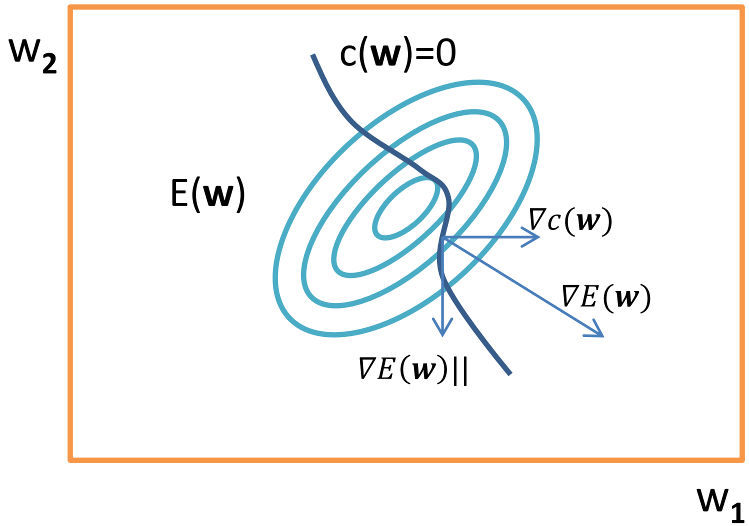
$$\boldsymbol{B}(t) = \frac{\Delta\boldsymbol{w}(t)^T\boldsymbol{g}(t)}{\Delta\boldsymbol{w}(t)^T\Delta\boldsymbol{g}(t)}$$

- Svakih N koraka, pretraživanje se restarta u smjeru negativnog gradijenta

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(**w**)* želimo ići po gradijentu koji istovremeno leži na toj plohi.

- To zapravo znači da gradijent mora biti okomit na normalu površine ograničenja
- gradijent - $\nabla E(\boldsymbol{w})||$ - komponenta $\nabla E(\boldsymbol{w})$ paralelna gradijentu plohe ograničenja



Što kada imamo ograničenja u funkciji greške (SVM)?

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

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

- Kontanta λ se naziva ***Lagrangeov multiplikator***. Funkcija koja se optimira u slučaju kada minimiziramo funkciju greške uz ograničenja naziva se ***Lagrangeovom funkcijom***
- Svojstvo LF je da kada je njen gradijent jednak 0, ograničenja su zadovoljena, a isto vrijedi i za $\nabla E(\boldsymbol{w})$:

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

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

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

Što kada imamo velike količine podataka ?

- Sve ove metode pretpostavljaju izračunavanje gradijenta na cijelomskupu 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…

Online pristup

- Ako mini-skup postane jedan primjer – **online learning**
- Drugo ime za tu metodu jest 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.

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