STROJNO UČENJE

⇒ **strojno učenje**:

- programiranje računala da **optimizira** nek **kriterij uspješnosti** temeljem podatkanih primjera

PRIMJENE:

1) UI-PROBLEMI
   - ↳ čak teži od NP-potpunih problema
   - ↳ ovdje niti heuristika ne pomaže
2) SUSTAVI KOJI SE DINAMIČKI MIJENJAJU
   - ↳ područje robotike i „pametnih" korisničkih sučelja
3) DATA-MINING
   - ↳ dubinska analiza podataka u poslovnom i znanstvenom svijetu
   - ↳ financije i kreditna sposobnost
   - ↳ medicina i dijagnosticiranje

⇒ VRSTE STROJNOG UČENJA:

a) NADZIRANO UČENJE (većinom se ovim bavimo)

↳ podac su parovi (ulaz, izlaz)

↳ treba pronaći izlaz = f(ulaz)

b) NENADZIRANO UČENJE

↳ dan su podac bez izlaza, tj. bez ciljne vrijednosti

↳ treba pronaći povezanost među podacima

c) POTPORNO UČENJE

↳ uči se optimalna strategija ponašanja u prostoru podataka

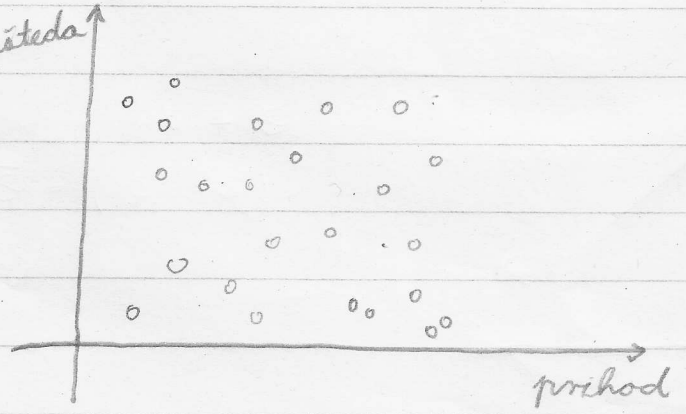
OSNOVNI KONCEPTI

⇒ odabir modela → učenje modela → vrednovanje modela
→ dijagnostika i ispravljanje

⇒ PRIMJER (engl. INSTANCE):
- jedinica obrade
- jedan redak u bazi podataka
- prikazan vektorom značajk (engl. features):

$$\vec{X} = (x_1, \ldots, X_n)$$

⇒ PROSTOR PRIMJERA (engl. INSTANCE SPACE):
- dimenzionalnost u kojem se primjeri mogu naći
- primjer je neki vektor u prostoru primjera
- mi ćemo se baviti dvodimenzionalnim i trodimenzionalnim zbog jednostavnosti prikaza
- oznaka: X
- npr. kreditna sposobnost:



⇒ OZNAKA (engl. LABEL):

- oznaka klase kod klasifikacije:

$y \in Y$, imamo K klasa:

$$Y = \{0, \dots, K-1\}$$

- oznaka klase kod regresije:

$$Y = \mathbb{R}$$

⇒ ukupan broj primjera je N i tipično vrijedi da je broj primjera puno veći od broja značajki: $N \gg n$

⇒ SKUP OZNAČENIH PRIMJERA:

- matrica s podacima ulaz → izlaz

$$D = \{(\vec{x}^i, y^i)\}_{i=1}^{N}$$

- primjet da vrijedi: $D \subseteq X \times Y$

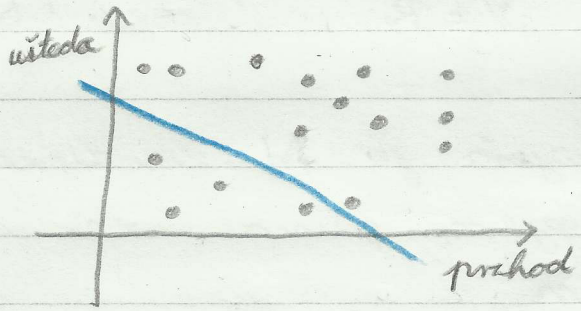
⇒ HIPOTEZA:

- funkcija koja preslikava:

$$h: X \to Y$$

- PR: za binarnu klasifikaciju:

$$h: X \to \{0, 1\}$$



⇒ VEKTOR PARAMETARA:

- modificira hipotezu kako bi je bolje prilagodio problemu
- oznaka: $\vec{\Theta}$

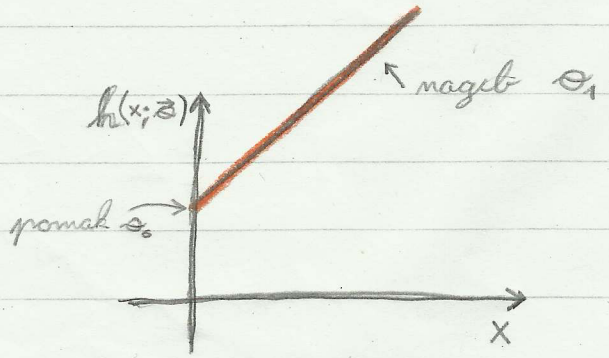
PR: Predviđamo broj ugleda na temelju dobi osobe?

$$h: X \to R \quad (n = 1)$$

↑ broj godina

$$h(x) = \Theta_1 x + \Theta_0$$

$$\vec{\Theta} = (\Theta_0, \Theta_1)$$



PR: Viden primjer predvidanja kreditne sposobnosti na temelju prihoda i ušteđe?

↳ definiramo hipotezu:

$$h(x_1, x_2; \Theta_0, \Theta_1, \Theta_2) = \mathbb{1}(\Theta_1 x_1 + \Theta_2 x_2 + \Theta_0 \geq 0)$$

$$\mathbb{1}: \{T, \bot\} \to \{0, 1\}$$

$$\mathbb{1}(p) = \begin{cases} 1 \text{ ako } p \equiv T \\ 0 \text{ inače} \end{cases}$$

⇒ MODEL:

- skup hipoteza
- oznaka: $\mathcal{H}$
- definicija:

$$\mathcal{H} = \{ h(\vec{x}; \vec{\theta}) \}_{\vec{\theta}}$$

⇒ UČENJE MODELA:

↳ pretraživanje modela $\mathcal{H}$ u nad da nađemo najbolju hipotezu

↳ to je cilj strojnog učenja

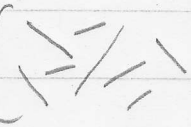
↳ primjetimo da je ovo zapravo optimizacijski problem

↳ zapravo, tražimo najbolje parametre $\vec{\theta}$

PR: Što je model na primjeru s procjenom kreditne sposobnost na temelju prihoda i uštede?

↳ model su zapravo svi mogući pravci koj razdvajaju primjere

↳ slikovit prikaz:

$\mathcal{H} = \{$  $\}$

⇒ EMPIRIJSKA POGREŠKA:

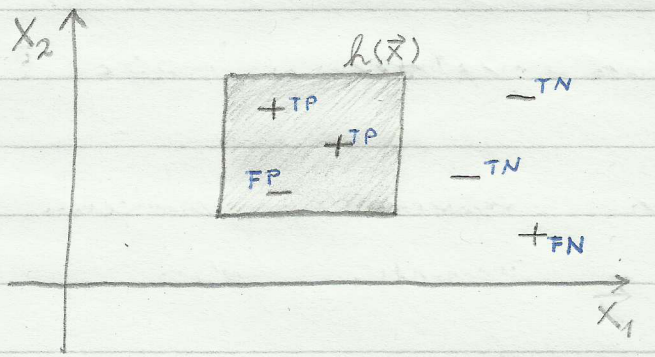
• oznaka: $E(h|D)$

čitaj: koliko hipoteza „h" griješi na skupu primjera „D"

• za klasifikaciju pišemo:

$$E(h|D) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{1\{h(\vec{x}^i) \neq y^i\}}_{\text{FUNKCIJA GUBITKA - definira gubitak na pogrešnoj klasifikaciji}}$$

PR: Izračunaj empirijsku pogrešku ako hipoteza $h(\vec{x})$ kaže da je sve unutar pravokutnika pozitivno:



$$E(h|D) = \frac{2}{6}$$

$$E(h|D) = \frac{1}{3}$$

TP - true positive (2)

TN - true negative (2)

FP - false positive (1)

FN - false negative (1)

⇒ SLOŽENOST MODELA:

- u idealnom slučaju, postoj neka hipoteza koja savršeno klasificira sve primjere:

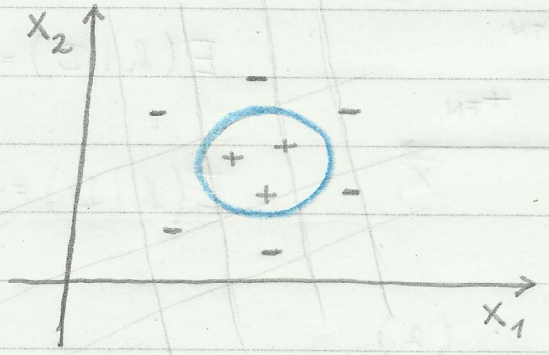
$$\exists h \in \mathcal{H} \text{ za koj } E(h|D) = 0$$

- no, moće b-t da to nije slučaj:

$$\forall h \in \mathcal{H} \text{ vrijed } E(h|D) > 0$$

↳ tada kažemo da naš model nije dovoljne složenosti il dovoljnog kapaciteta

PR: Pogledajmo sljedeći skup primjera i dvije hipoteze?



$\mathcal{H}_1$ = „skup pravaca" $\mathcal{H}_2$ = „skup kružnica"

↓

NEDOVOLJNOG KAPACITETA

⇒ ŠUM:

- nepravilnost u podacima koja otežava klasifikaciju
- uzroci: a) nepreciznost u mjerenju
  b) pogreška u označavanju
  c) postojanje okruženja i nepredviđenih značajki
  d) nejasne granice između klasa
- ideja je da se model ne prilagodi šumu nego da se šum predvidi i uvažere se model koji se ne prilagođava previše podacima nego **dobro generalizira**
  ↳ treba spriječiti OVERFITTING!

⇒ ODABIR MODELA:

↳ iz familje modela trebamo izabrati model koj dobro generalizira

$\{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_m\}$ ← familija modela

↳ kod klasifikacije, najčešće biramo kolčinu nelinearnosti u modelu (pravac, krivnica, neki čudni oblici, ...)

↳ mi zapravo ovdje biramo jedan parametar (model) te ga nazivamo HIPERPARAMETAR

↳ stoga, možemo reći da hiperparametar određuje složenost modela

↳ PREFERIRAJ JEDNOSTAVNIJI MODEL!

a) bolja generalizacija

b) lakše učenje i primjena

c) lakše tumačenje

↳ <u>druge krajnosti</u>:

① Podnaučenost („underfitting")

⇒ $\mathcal{H}$ je prejednostavan za podatke

⇒ loša predikcija na viđenim i na neviđenim podacima

② Prenaučenost („overfitting")

⇒ $\mathcal{H}$ je presložen za podatke

⇒ dobra predikcija na viđenim podacima, ali loša na neviđenim (model ne generalizira)

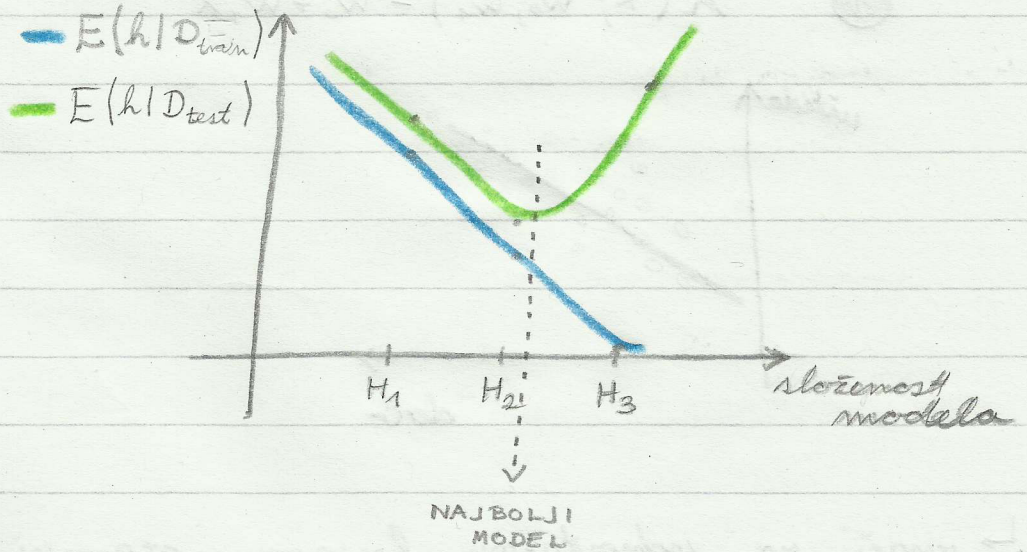
⇒ UNAKRSNA PROVJERA (engl. CROSS VALIDATION):

- ideja je podijelit skup primjera na dva dijela: a) skup za učenje
  b) skup za provjeru

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$$

- „$\mathcal{D}_{train}$" i „$\mathcal{D}_{test}$" su disjunktni
- pogreška učenja (engl. train error):
$$E(h \mid \mathcal{D}_{train})$$
- pogreška testiranja (engl. test error):
$$E(h \mid \mathcal{D}_{test})$$

⇒ okvirna slika pogreške u ovisnosti o složenosti modela:



REGRESIJA

⇒ **Jednostavna regresija:**

$$\mathbb{D} = \left\{ (\vec{x}^i; y^i) \right\}_{i=1}^{N}$$

$$h : \mathbb{R}^n \to \mathbb{R}$$

$\vec{x}^i \to$ ulazne / neovisne varijable

$y \to$ izlazna / ravisna varijable

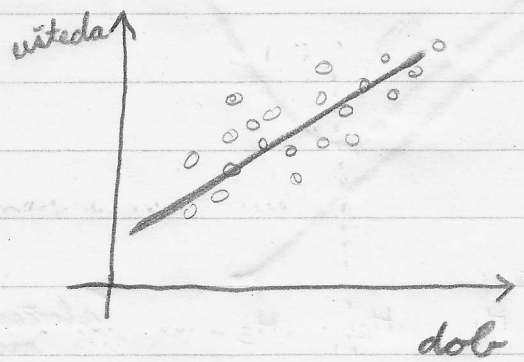
⇒ **Linearna regresija:**

$$h(\vec{x}; \vec{w}) = w_0 + w_1 x_1 + x_2 x_2 + \dots + w_n x_n$$

↳ u općenitom slučaju, ovo je hiperravnina

↳ za jednostavnu regresiju, model je: $(n = 1)$

(1) $$h(x; w_0, w_1) = w_0 + w_1 x$$

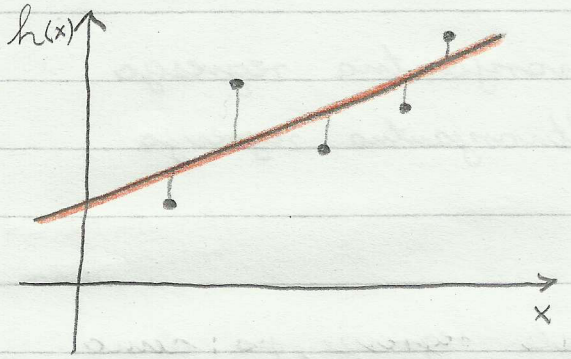


↳ vidiš, za jednostavnu linearnu regresiju model je pravac !

• time ćemo se mi baviti !

⇒ imamo hipotezu h, pa još trebamo funkciju gubitka

⇒ FUNKCIJA GUBITKA:



$$L(Y, h(x)) = (Y - h(x))^2$$

⇒ FUNKCIJA POGREŠKE:

- očekivanje funkcije gubitka

(2) $$E(h|D) = \frac{1}{2} \sum_{n=1}^{N} (Y - h(\vec{x}^i))^2$$

⇒ OPTIMIZACIJSKI POSTUPAK:

(3) $$h^* = \underset{h \in H}{\operatorname{argmin}}\, E(h|D)$$

↳ uvrstimo model i funkciju pogreške:

$$(w_0, w_1)^* = \underset{w_0, w_1}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{N} (y^i - w_0 - w_1 x_1)^2$$

↳ kako bi ovo dobili, treba derivirat i ratim dobivenu potrebne težine w_0 i w_1 !

⇒ VRSTE REGRESIJE:

BROJ ULAZA
- a) $n = 1$ ⇒ jednostavna regresija
- b) $n > 1$ ⇒ višestruka regresija

BROJ IZLAZA
- a) $f(\vec{X}) = Y$ ⇒ univarijantna regresija
- b) $f(\vec{X}) = \vec{Y}$ ⇒ multivarijantna regresija

⇒ sada znamo različite vrste regresije, pa ćemo pogledati tri komponente linearne regresije u općem slučaju !!

TRI KOMPONENTE LINEARNE REGRESIJE

(1) MODEL:

$$h(\vec{x}; \vec{w}) = w_0 + w_1 x_1 + \ldots + w_m x_m = \sum_{i=1}^{m} w_i x_i + w_0$$

↳ prelazimo na matrični račun:

- da bi mogli uključiti w_0 u skalarni umnožak, trebamo „dummy-feature": $x_0 = 1$

$$h(\vec{x}; \vec{w}) = \vec{w}^T \cdot \vec{x}$$

$$\left.\begin{aligned} \tilde{\vec{w}} &= (w_0, \vec{w}) \\ \tilde{\vec{x}} &= (x_0, \vec{x}) \end{aligned}\right\} \text{nekad se koriste ove oznake}$$

(2) FUNKCIJA GUBITKA I POGREŠKE:

$$L(y^i, h(\vec{x}^i)) = (y^i - h(\vec{x}^i))^2$$

$$E(h|D) = \frac{1}{2} \sum_{i=1}^{N} (y^i - h(\vec{x}^i))^2$$

(3) OPTIMIZACIJSKI POSTUPAK:

$$\vec{w}^* = \underset{\vec{w}}{\operatorname{argmin}}\, E(\vec{w}|D)$$

} postupak najmanjih kvadrata („ordinary least squares") OLS

POSTUPAK NAJMANJIH KVADRATA

$D = (X, Y)$

$$X = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \cdots & x_m^1 \\ 1 & x_1^2 & x_2^2 & \cdots & x_m^2 \\ \vdots & & & & \\ 1 & x_1^N & x_2^N & \cdots & x_m^N \end{bmatrix}$$

← prvi primjer
← drugi primjer
← N-ti primjer

} MATRICA DIZAJNA

$$Y = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix}$$

→ sada ćemo „naštelat" množenje matrica da dobijemo model

⇒ EGZAKTNO RJEŠENJE:

$(\vec{x}, \vec{y}) \in D \quad h(\vec{x}^i) = y^i$

↳ naš „naštelani" sustav:

$$X\vec{w} = \vec{y} \quad / \cdot X^{-1}$$

$$\boxed{\vec{w} = X^{-1} \cdot \vec{y}}$$

↓

no ovo je problem („ill defined")
jer X mora biti kvadratna $(N = m+1)$

$N > n+1 \Rightarrow$ preodređen sustav

$N < n+1 \Rightarrow$ pododređen sustav

↳ čak i kad bi „N" bio jednak „n+1", možda matrica nije nije punog ranga (tj. sva linearno ravnine retke il stupce)

↳ tada smo u dalnjem problemima jer matrica nije regularna što znači da nema inverz, pa sustav ne možemo riješiti na ovaj način

TREBAMO DRUGI PRISTUP!

(tražimo približno rješenje metodom / portuphom najmanjih kvadrata)

OKRENI!

$X\vec{w} = \vec{y}$ ← to treba rješit?

$$E(\vec{w}|D) = \frac{1}{2} \cdot \sum_{i=1}^{N} (\vec{w}^T \cdot \vec{x}^i - y^i)^2$$

⋮ (objasni ovo sam!)

$$E(\vec{w}|D) = \frac{1}{2} (X\vec{w} - \vec{y})^T (X\vec{w} - \vec{y})$$

$$(AB)^T = B^T A^T$$
$$(A^T)^T = A$$

$$E(\vec{w}|D) = \frac{1}{2} \left(\vec{w}^T X^T X \vec{w} - \vec{w}^T X^T \vec{y} - \vec{y}^T X \vec{w} + \vec{y}^T \vec{y} \right) =$$

$$= \frac{1}{2} \left(\vec{w}^T X^T X \vec{w} - 2 \cdot \vec{y}^T X \vec{w} + \vec{y}^T \vec{y} \right)$$

↳ trebamo ovo derivirat, pa trebamo pravila za deriviranje matrica:

a) $\frac{\partial}{\partial \vec{x}} \vec{x}^T A \vec{x} = \vec{x}^T (A + A^T)$

b) $\frac{\partial}{\partial \vec{x}} A\vec{x} = A$

↳ sada možemo derivirat:

$$\nabla_{\vec{w}} E(\vec{w}|D) = \frac{1}{2} \left(\vec{w} \left(X^T X + \overbrace{(X^T X)^T}^{X^T X} \right) - 2\vec{y}^T X \right) =$$

$$= \vec{w}^T X^T X - \vec{y}^T X$$

↳ ovo je ispalo redak, a nama treba vektor stupac, pa cijeli izraz transponiramo:

$$\nabla_{\vec{w}} E(\vec{w}|D)^T = X^T X \vec{w} - X^T \vec{y}$$

↳ izjednačimo s nul-vektorom da dobijemo nul-točke:

$$X^T X \vec{w} - X^T \vec{y} = \vec{0}$$

$$X^T X \vec{w} = X^T \vec{y} \quad / \cdot (X^T X)^{-1}$$

} SUSTAV NORMALNIH JEDNADŽBI

$$\vec{w} = \underbrace{(X^T X)^{-1} X^T}_{X^+} \vec{y}$$

$X^+ \Rightarrow$ PSEUDOINVERZ

↳ vrijedi: $X^+ = X^{-1}$

↳ „generalna stvar" iz linearne algebre koja je potrebna za metodu najmanjih kvadrata

↳ pseudoinverz zapravo minimizira kvadratni gubitak:

- on minimizira ovo: $\| X \vec{w} - \vec{y} \|_2$

↳ da u funkciji greške nemamo kvadrat, nebi dobili zatvorenu formu u jednom koraku računa i nebi došli do pseudoinverza i onda smo u problemima

⇒ još jedan pojam:

- matrica $X^T X$ se naziva **Gramova matrica**
- ona može biti velika i može utjecati na performanse
- X je $N \times (n+1)$, pa je $X^T X$ dimenzija koja ne ovisi o N:

$$(n+1) \times (n+1)$$

- iz tog zaključujemo da Gramova matrica ne ovisi o broju ulaznih primjera već samo o vektoru značajki $\vec{x}$

↳ o tome ovisi performanse

PROBABILISTIČKA INTERPRETACIJA

$$h(\vec{x}; \vec{w}) = \vec{w}^T \vec{x}$$

$f: X \to R$ $\Rightarrow$ imamo nekakav šum u podacima

↳ izlaz je tada: $y' = f(\vec{x}) + \varepsilon_i$

↳ šum se „pokorava" normalnoj distribuciji:

$$\varepsilon \sim N(0, \sigma^2)$$

$$P(X = x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

↳ funkcija generira podatke s nekim normalnim rasipanjem:

$$y^i \mid \vec{x}^i \sim f(\vec{x}^i) + N(0, \sigma^2)$$

$$\sim N(f(\vec{x}^i), \sigma^2)$$

↑

generiranje podataka oko $f(\vec{x}^i)$ s normalnim rasipanjem

⇒ podaci su dani:

- mi pretpostavljamo šum po gausu
- tražimo hipotezu koja najvjerojatnije daje podatke koje vidimo

$$P(\vec{y} \mid \vec{w}) = \prod_{i=1}^{N} P(y^i \mid \vec{x}^i)$$

IZGLEDNOST

↳ IZGLEDNOST: vjerojatnost podataka za danu hipotezu

- mi maksimiziramo vjerojatnost da su podaci baš onakvi kakve mi vidimo ∇ //

⇒ **LOG-IZGLEDNOST:**

$$\ln P(\vec{y} \mid \vec{w}) = \ln \prod_{i=1}^{N} P(y^i \mid \vec{w})$$

$$\ln P(\vec{y} \mid \vec{w}) = \sum_{i=1}^{N} \ln P(y^i \mid \vec{w})$$

↳ dalje raspisujemo koristeći normalnu razdiobu:

$$\ln P(\vec{y} \mid \vec{w}) = \sum_{i=1}^{N} \ln N\left(f(\vec{x}^i), \sigma^2\right) =$$

$$= \sum_{i=1}^{N} \ln N\left(n(\vec{x}; \vec{w}), \sigma^2\right)$$

↳ raspisujemo normalnu razdiobu:

$$\ln P(\vec{y} \mid \vec{w}) = \sum_{i=1}^{N} \ln \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^i - h(\vec{x}^i; \vec{w})^2}{2\sigma^2}\right)$$

$$= \underbrace{-N \ln(\sqrt{2\pi}\sigma)}_{\text{KONST}} - \underbrace{\frac{1}{2\sigma^2}}_{\substack{\text{NIJE} \\ \text{BITNO}}} \sum_{i=1}^{N} \left(y^i - h(\vec{x}^i, \vec{w})\right)^2$$

↓

ovo rješimo maksimizirati jer je to vjerojatnost da su podaci bili generirani baš onakvi kakve ih mi vidimo (ovi podaci koji su nam dani)

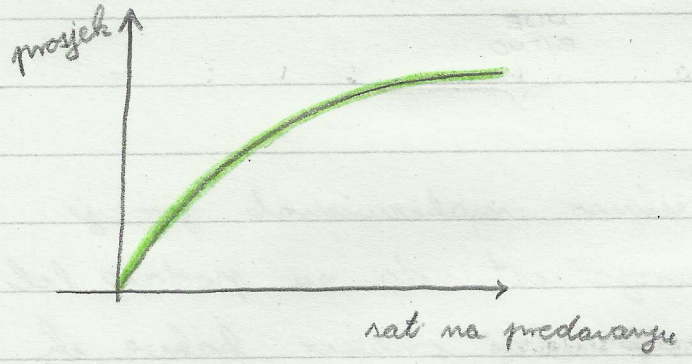
⇒ ovaj probabilistički pogled nam je dao opravdanje za kvadratnu funkciju pogreške

(u strojnom učenju često stvari možemo promatrati na više načina)

NELINEARNA REGRESIJA

$$h(\vec{x}; \vec{w}) = \sum_{i=0}^{m} w_i x_i = \vec{w}^T \vec{x}$$

⇒ predviđamo nelinearne funkcije na temelju podataka:



⇒ JEDNOSTRUKA NELINEARNA REGRESIJA: $(m = 1)$

$$h(x; w_0, w_1, w_2) = \underbrace{w_0 + w_1 x}_{\text{LINEARNI MODEL}} + w_2 x^2 \qquad \in H_2$$

⇒ VIŠESTRUKA NELINEARNA REGRESIJA: $(m = 2)$

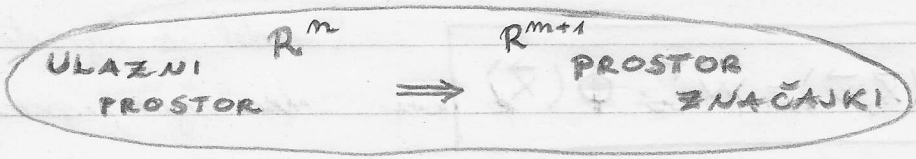
$$h(x_1, x_2; \vec{w}) = \underbrace{w_0 + w_1 x_1 + w_2 x_2}_{\text{LINEARNI MODEL}} + \underbrace{w_3 x_1 x_2}_{\substack{\text{interakcijska} \\ \text{značajka}}} + w_4 x_1^2 + w_5 x_2^2 \qquad \in H_2$$

⇒ sada želimo unificirat model kako bi mogli njih isto tretirati u postupku optimizacije

⇒ no, model ne možemo mijenjati, pa ćemo **mijenjati podatke** (preslikavanje u prostor značajki)

⇒ PRESLIKAVANJE PODATAKA:

↳ podatke preslikavamo u viši prostor kako bi problem postao linearan u tom višem prostoru



↳ Bazne funkcije ($m+1$): $\{\phi_0, \phi_1, \ldots, \phi_m\}$

$\phi_j : R^n \to R$

↳ Funkcija preslikavanja:

$$\boxed{\phi(\vec{x}) = \left(\phi_0(\vec{x}), \phi_1(\vec{x}), \ldots, \phi_m(\vec{x})\right)}$$

$\downarrow \quad \phi_0(\vec{x}) = 1$

$$\phi(\vec{x}) = \left(1, \phi_1(\vec{x}), \ldots, \phi_m(\vec{x})\right)$$

↓

ovo je „$m+1$" dimenzionalni vektor što je veći od „$n+1$" u tipičnoj primjeni jer mi težimo linearnoj odvojivosti, a u višedimenzionalnom prostoru je za to puno veća šansa

⇒ model regresije s preslikanim podacima:

$$h(\vec{x}, \vec{w}) = \sum_{j=0}^{m} w_j \phi_j(\vec{x})$$

$$\boxed{h(\vec{x}, \vec{w}) = \vec{w}^T \cdot \phi(\vec{x})}$$

(model s ugrađenom funkcijom preslikavanja)

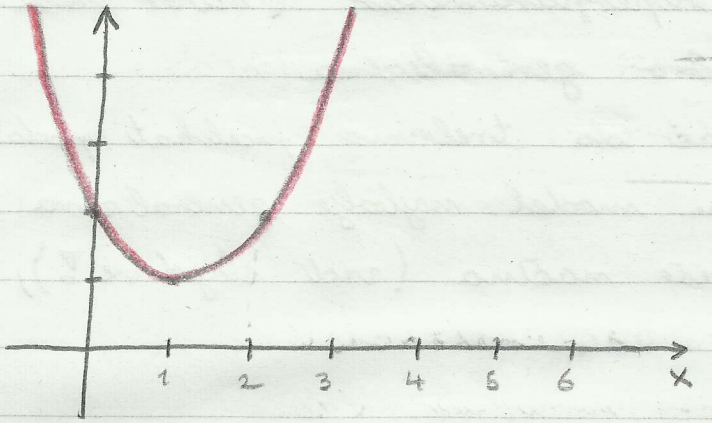
↓

pri tome da s preslikavanjem nismo uopće promijenili model od standardne regresije

↳ ovo se naziva LINEARAN MODEL REGRESIJE ili ponekad LINEARNA REGRESIJA (ne miješaj s jednostavnom linearnom regresijom od prije!)

PR: Pogledajmo primjer s realnim podacima po nekoj kvadratnoj funkciji ?

$X = R$, $n = 1$



↳ preslikavanje u 3D prostor:

$$\Phi(x) = (1, x, x^2)$$

$\Phi_1(x)$ $\Phi_2(x)$

$0 \mapsto (1, 0, 0)$

$1 \mapsto (1, 1, 1)$

$2 \mapsto (1, 2, 4)$

↳ funkcija greške i optimizacijski postupak se <u>ne mijenjaju</u> u odnosu na standardnu linearnu regresiju bez preslikavanja

(TO JE BIO CILJ?)

PRENAUČENOST

⇒ trebamo optimizirati hiperparametar (izbor modela) tako da model dobro generalizira

↳ e sad, to znači da trebamo preslikati podatke u prostor u kojem model najbolje generalizira

↳ za to ima više načina (vid slajdove ∇b), no mi ćemo obradit REGULARIZACIJU

⇒ NAPOMENA: ako imamo 100 primjera, nemoj te primjere preslikat u 100 dimenzijski prostor jer su onda primjeri svakako linearno razdvojivi, al model neće generalizirat (savršeno će odvojit podatke, al će se prenaučit)

↳ tipično, to će se manifestirat tako da veće dimenzije imaju ogromne težine ur njih jer je model shvatio da savršeno može „sjest" na podatke ako se prebac u veće dimenzije

↳ to ne želimo, pa ćemo u postupku REGULARIZACIJE kažnjavati hipoteze koje imaju takve velike težine ur neke dimenzije (tipično visoke)

REGULARIZACIJA

⇒ u praksi, probleme rješavamo ovako:

1) uzmi što složeniji model (bojim se podnaučenosti)
2) regularizacijom „smiri" taj model

⇒ ako su neke težine gotovo nula, efektivno se gase značajke te to nazivamo RIJETKI MODEL

↳ to je dobro jer znači da te značajke ne trebamo, pa ih i možemo maknuti !

⇒ **REGULARIZIRANA FUNKCIJA POGREŠKE:**

$$E_R(\vec{w}|D) = E(\vec{w}|D) + \lambda \Omega(\vec{w})$$

„λ" → faktor regularizacije

$\lambda \Omega(\vec{w})$ — regularizacijski izraz

- ako je λ jednak 0, nemamo regularizaciju i ne sprječavamo prenaučenost
- ako je λ jako velik, model teži k linearnom modelu

⇒ sada ćemo pogledat kako izabrati prikladnu funkciju zaglađivanja $\Omega(\vec{w})$

⇒ **trebamo p-norme:**

- L2-norma: $p=2$ $\quad \|w\|_2 = \sqrt{\sum_{j=1}^{m} w_j^2} = \sqrt{\vec{w}^T \vec{w}}$
- L1-norma: $p=1$ $\quad \|w\|_1 = \sum_{j=1}^{m} |w_j|$
- L0-norma: $p=0$ $\quad \|w\|_0 = \sum_{j=1}^{m} 1\{w_j \neq 0\}$

↳ NAPOMENA: w_0 ne regularizramo („j" ide od 1, a ne od nule) jer on samo pomiče graf gore-dolje (određuje pomak) i ne želimo izgubiti tu slobodu te osim toga – on ne uzrokuje prenaučenost!

⇒ **L2-regularizirane regresije:** („hrbatne regresije", „ridge regression"):

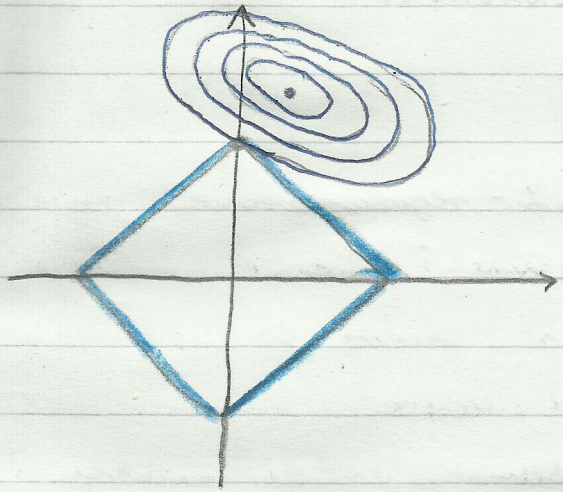
$$E_R(\vec{w}|D) = \frac{1}{2}\sum_{i=1}^{N}\left(\vec{w}^T \phi(\vec{x}^i) - y^i\right)^2 + \frac{1}{2}\lambda \|\vec{w}\|_2^2$$

⇒ **L1-regularizirane regresije:** („Lasso")

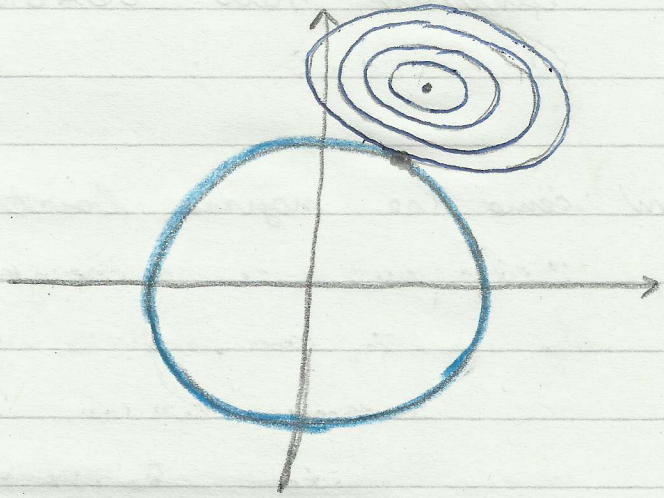
$$E_R(\vec{w}|D) = \frac{1}{2}\sum_{i=1}^{N}\left(\vec{w}^T \phi(\vec{x}^i) - y^i\right)^2 + \lambda \|\vec{w}\|_1$$

⇒ NAPOMENA: jer L1 regularizacija mjeri · kvadrat, =
ona više kažnjava male težine (blizke nul)
od L2 regularizacije

↳ zbog toga L1 regularizacija puno lakše dođe rijetke modele od L2 regularizacije



$\|w\|_1$



$\|w\|_2$

↳ ovo je pojednostavljeni prikaz u dvije dimenzije

↳ ako se susretne elipse s objektom na osi, značajka se ugasila jer je težina uz nju jednaka nul

↳ veća je vjerojatnost susreta na osi kod L1 regularizirajućih regresija jer imaju „šiljastost" višedimenzijskih hiper-oblika, a ne hiper-kugli

⇒ L0 - regularizirane regresije

$$E_R(\vec{w}|D) = \frac{1}{2}\sum_{i=1}^{N}\left(w^T\phi(\vec{x}^i) - y^i\right)^2 + \frac{1}{2}\sum_{j=1}^{m} 1\{w_j \neq 0\}$$

↳ kažnjava za svaku ne-uzaćenu značajku, pa efektivno radi ODABIR ZNAČAJKI

⇒ mi ćemo se najviše baviti L2 - regulariziranim regresijama

L2-REGULARIZIRANA REGRESIJA (OPTIMIZACIJA)

⇒ optimizacijski postupak L2-regulariziranih regresija je vrlo sličan već izvedenom postupku za regresije bez regularizacije

(vidi slajdove za izvod!)

⇒ PAZI: λI je matrica s λ na dijagonali, ali na prvom mjestu je 0 jer w_0 ne regulariziramo

$$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda \end{bmatrix}$$

↳ regularizacija je zapravo ispala samo zbrajanje

⇒ Grammove matrice sa λI

BITNO!

KONDICIJA MATRICE

⇒ za sad znamo: $\vec{W} = (\Phi^T \Phi)^{-1} \Phi^T \vec{Y}$

$$\Phi = \underbrace{\begin{bmatrix} | & | & | & | & | \\ | & | & | & | & | \\ | & | & | & | & | \end{bmatrix}}_{m+1} N$$

↳ ako su dvije značajke linearno zavisne, matrica nije punog ranga te nema inverz

↳ to zovemo MULTIKOLINEARNOST

↳ ako nešto tako izračunamo, jer su značajke „skoro" linearno zavisne i matrica je blizu singularne, dobit ćemo NESTABILNO RJEŠENJE:

- uz malu promjenu ulaznih podataka, rješenje $\vec{W}$ se drastično mijenja!
- to je obilježje prenaučenog modela!

⇒ KONDICIJSKI BROJ:

↳ matematički alat koji mjeri nestabilnost rješenja

⇒ na sreću, REKONDICIONIRANJE radimo automatski kad radimo regularizaciju jer tada množimo Grammovu matricu sa λI

↳ stoga, regularizaciju možemo gledati i tako (kao rekondicioniranje)

↳ funkcija pogreške prilikom množenja Grammove matrice postaje konveksnija pa se zato zove hrbatna regresija

VIDI SLIKU S PREDAVANJA!

LINEARNI DISKRIMINATIVNI MODELI

⇒ od sada nadalje, bavit ćemo se **klasifikacijom**

⇒ **LINEARNI**: znači da je granica linearna u prostoru značajki

↳ stoga, model će biti:

$$h(\vec{x}; \vec{w}) = \vec{w}^T \vec{x} \quad ; \quad x_0 = 1$$

↳ ovo promatramo na način da je granica tamo gdje je model jednak nul:

$$h(\vec{x}; \vec{w}) = \vec{w}^T \vec{x} = 0$$

↳ granica je dimenzionalnost „n-1" (jedan manje od ulaznog prostora)

↳ funkcija $h(\vec{x}) = 0$ naziva se:

DISKRIMINATIVNA FUNKCIJA

↳ ako želimo nelinearnost, radimo preslikavanje u viš prostor (VIDI PROŠLO PREDAVANJE!)

⇒ **DISKRIMINATIVNI**: znači da model generira samo granicu između skupina primjera

↳ suprotni model:

• GENERATIVNI: generira vjerojatnosti značajki na temelju primjera, pa daje više informacija (u drugom članku!)

⇒ GEOMETRIJA LINEARNOG MODELA:

↳ recimo da imamo dvije klase primjera: $K=2$

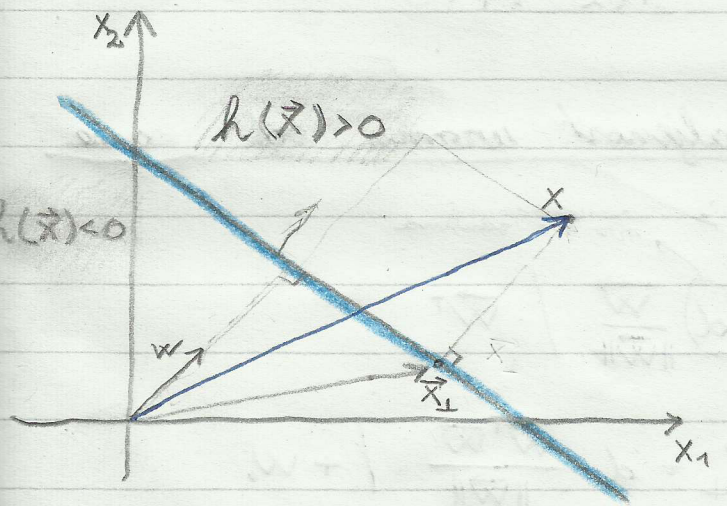
↳ razmatrajmo <u>jednodimenzijski</u> primjer, pa je naš model:

$$h(\vec{x}; w_0, w_1) = w_0 + w_1 x$$

• granica je tada točka jer je ona u „n-1" dimenzijskom prostoru

↳ <u>dvodimenzijski</u> primjer: $n=2$

$$h(\vec{x}; w_0, w_1, w_2) = w_0 + w_1 x_1 + w_2 x_2$$



⇒ u sve slike cijelo vrijeme neke stvari koje nas zanimaju i proširujemo to na proizvoljan broj dimenzija ∇

a) <u>zanima nas vektor $\vec{w}$:</u>

• znamo: $h(\vec{x}^1) = h(\vec{x}^2) = 0$ (točke na granici)

$$w_0 + \vec{w}^T \vec{x}^1 = w_0 + \vec{w}^T \vec{x}^2$$

$$\boxed{\vec{w}^T (\vec{x}^1 - \vec{x}^2) = 0}$$

↓

vidimo da je $\vec{w}^T$ normala hiperravnine ∇

b) **ravnina me udaljenost hiperravnine od ishodišta sustava:**

↳ krenimo od jednadžbe hiperravnine jer je tada $\vec{x}$ točka na hiperravnini:

$$\vec{w}^T \vec{x} + w_0 = 0 \quad / : \|\vec{w}\|$$

$$\frac{\vec{w}^T}{\|\vec{w}\|} \vec{x} = -\frac{w_0}{\|\vec{w}\|}$$

↓

predznačena udaljenost hiperravnine od ishodišta (može biti pozitivna i negativna)

c) **ravnina me udaljenost uzorka od same decizijske granice:**

ovo me zanima

$$\vec{x} = \vec{x}_\perp + d \frac{\vec{w}}{\|\vec{w}\|} \quad / \cdot \vec{w}^T$$

$$\vec{w}^T \vec{x} = \vec{w}^T \vec{x}_\perp + d \cdot \frac{\vec{w}^T \vec{w}}{\|\vec{w}\|} \quad / + w_0$$

$$\underbrace{\vec{w}^T \vec{x} + w_0}_{h(\vec{x})} = \underbrace{\vec{w}^T \vec{x}_\perp + w_0}_{0} + d \cdot \frac{\vec{w}^T \vec{w}}{\|\vec{w}\|} = \frac{\|\vec{w}^2\|}{\|\vec{w}\|} = \|\vec{w}\|$$

↳ dobivamo:

$$h(\vec{x}) = d \cdot \|\vec{w}\| \Rightarrow \boxed{d = \frac{h(\vec{x})}{\|\vec{w}\|}}$$

↑

predznačena udaljenost od hiperravnine

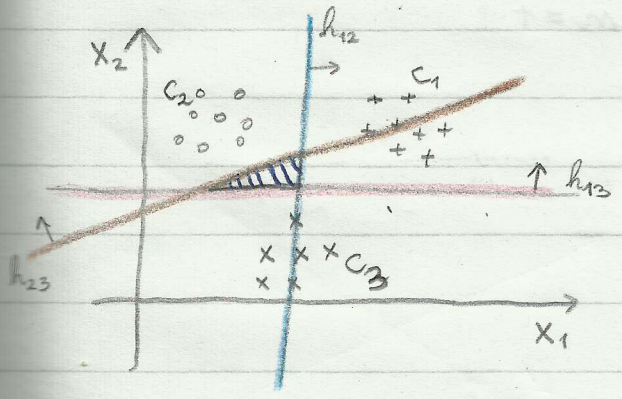
VIŠEKLASNA KLASIFIKACIJA

⇒ mnogi algoritmi u strojnom učenju su binarni, pa trebamo višeklasni problem rastaviti na više binarnih

⇒ **dva pristupa:**

a) OVO

↳ „one-vs-one"



↳ svaki model treba samo podijeliti dvije klase

↳ imamo $\binom{K}{2}$ decizijskih granica

↳ odluka na temelju glasova:

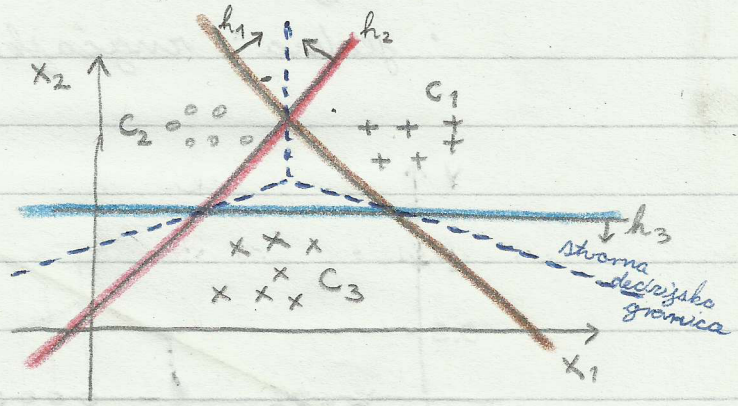
$$h(\vec{x}) = \arg\max_i \sum_{i \neq j} \operatorname{sgn}\left(h_{ij}(\vec{x})\right)$$

$$h_{ij}(\vec{x}) = -h_{ji}(\vec{x})$$

↳ trokut područje je „undefined" (jednak broj glasova)

b) OVR

↳ „one-vs-rest"



↳ svaki model razdvaja jednu klasu od ostalih

↳ klasificiram tamo gdje je izlaz određenog modela najveći:

$$h(\vec{x}) = \arg\max_j h_j(\vec{x})$$

↳ ovo rahtjeva samo K decizijskih funkcija

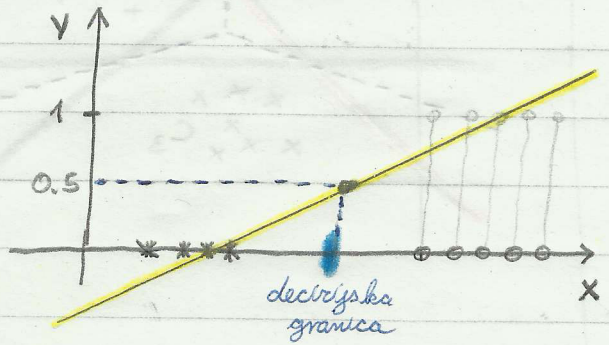
KLASIFIKACIJA REGRESIJOM

⇒ znamo: $\vec{E}(\vec{w}|D) = \frac{1}{2}(\Phi\vec{w} - \vec{Y})^T(\Phi\vec{w} - \vec{Y})$

$$\vec{w} = \underbrace{(\Phi^T\Phi)^{-1}\Phi^T}_{\Phi^+}\vec{Y}$$

⇒ <u>pogledajmo najjednostavniji slučaj</u>:

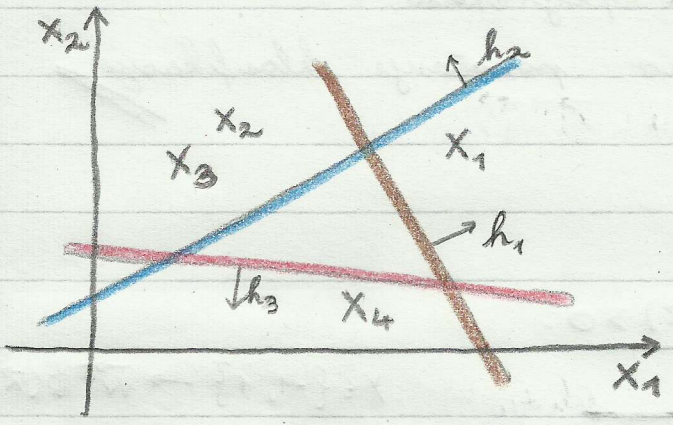
- dvije klase: $K = 2$
- jedna značajka: $n = 1$



- <u>ideja</u> – svi iz jedne klase imaju 1, a iz druge 0
  - pravac minimizira kvadratnu pogrešku jer pokušava spojit te točke (uč regresiju)
  - tamo gdje je polovica vrijednosti pravca je mjesto decizijske granice

⇒ **kompliciraniji slučaj:**

- tri klase: $K = 3$
- dvije značajke: $n = 2$



⇒ stavimo jedinice samo na ono mjesto gdje primjer spadaju u tu klasu

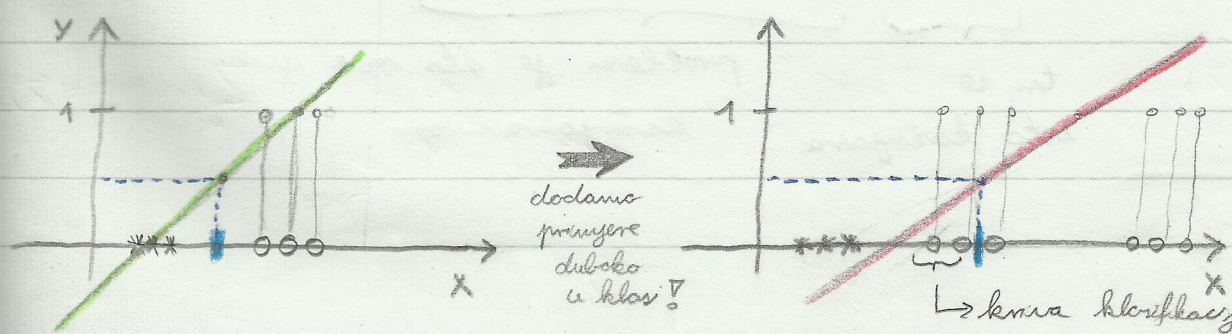
- „h_1" ima vektor $y_1 = (1, 0, 0, 0)^T$
- „h_2" ima vektor $y_2 = (0, 1, 1, 0)^T$
- „h_3" ima vektor $y_3 = (0, 0, 0, 1)^T$

⇒ PROBLEM: klasifikacija regresijom je loša jer:

a) izlazi modela nemaju vjerojatnosnu interpretaciju

b) NEROBUSNOST ∇:

↳ uzasna osjetljivost na primjere koj odskacu (jer regresija smanjuje kvadratni gubitak)



⇒ <u>gdje je problem?</u>

- problem je u tome što regresija kažnjava čak i dobro klasificirane primjere kako bi smanjila kvadratnu pogrešku
- to nije robusno ponašanje klasifikacije

IZLAZ / OZNAKA	$\vec{W}^T \Phi(\vec{x}) < 0$	$W^T \Phi(\vec{x}) > 0$
$y = -1$	+	−
$y = 1$	−	+

$y \in \{-1, 1\} \rightarrow \vec{W}^T \Phi(\vec{x}) \cdot y$

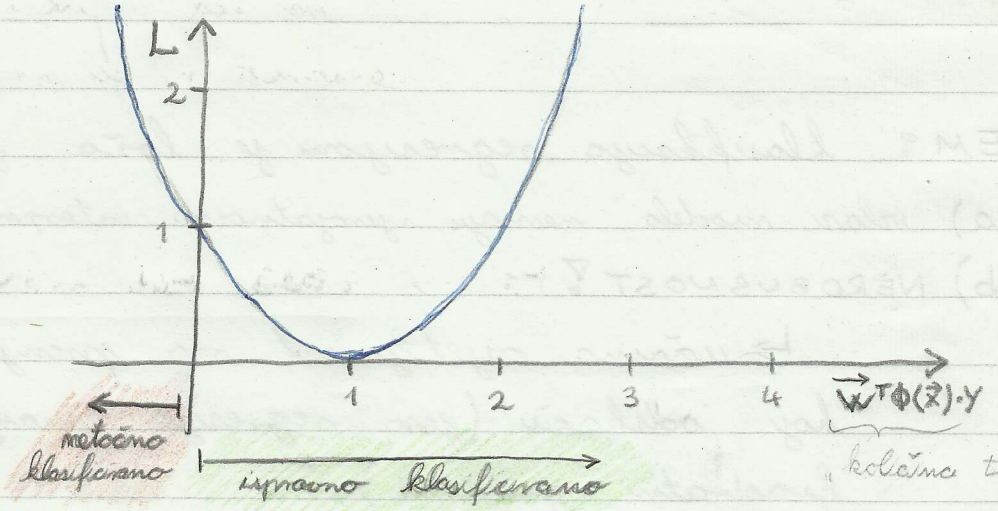
„+" ⇒ točna klasifikacija

„−" ⇒ netočna klasifikacija

⇓

- skicirajmo okvirno ponašanje funkcije gubitka:

$$L(y^i, h(\vec{x}^i)) = (y^i - h(x^i))^2$$



„količina točnost klasifikacije"

tu se isto kažnjava

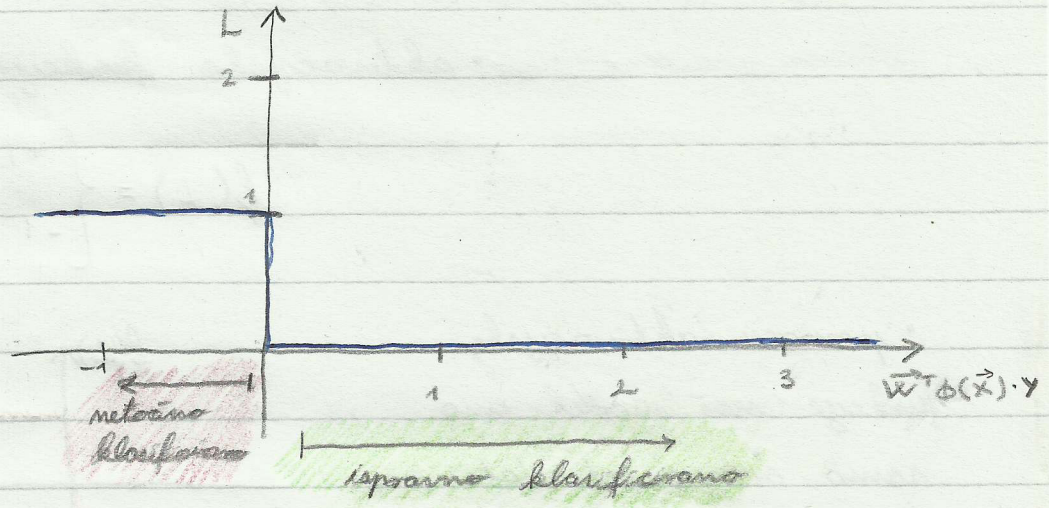
problem je što ove sve kažnjavamo

• ova funkcija gubitka (kvadratna) je razlog za loše ponašanje regresije za klasifikator

• stoga, trebamo bolje funkcije gubitka koje neće kažnjavati dobro klasificirane primjere

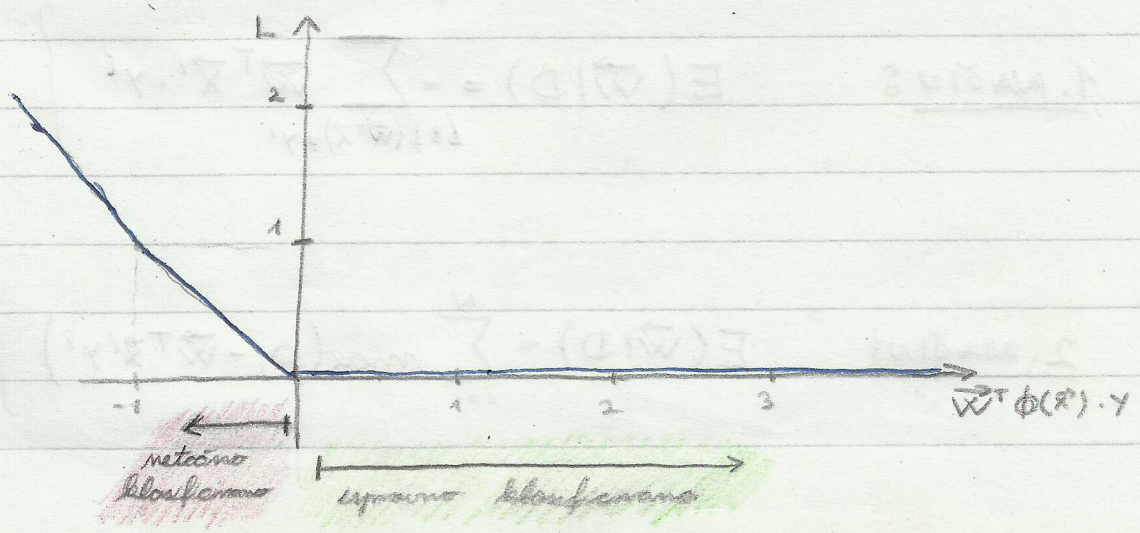
• neke takve funkcije gubitka:

a) ZERO-ONE LOSS



⇒ problem ove funkcije je to da se ne može derivirat, pa ju ne možemo minimizirat

b) PERCEPTRON LOSS FUNCTION



bolje jer se može derivirat (nije nula)

PERCEPTRON

⇒ <u>model je dan na sljedeći način:</u>

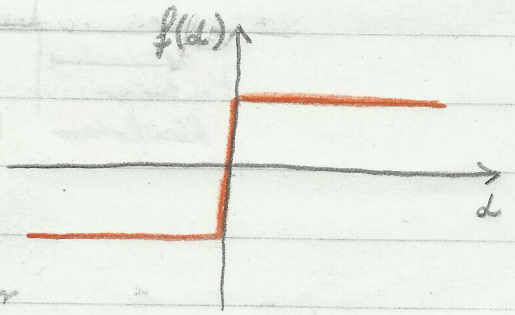
$$h(\vec{x}) = f(\vec{w}^T \vec{x})$$

↓

• aktivacijska funkcija:

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & \text{inače} \end{cases}$$

• pomoću aktivacijske funkcije naš model ima samo mogućnost dati 1 il -1 što je dobro jer jedan ekstreman primjer ne mijenja ponašanje



⇒ <u>funkcija pogreške:</u>

<u>1. NAČIN:</u> $$E(\vec{w}|D) = -\sum_{i \,\&\, f(\vec{w}^T \vec{x}) \neq y^i} \vec{w}^T \vec{x}^i \cdot y^i$$

} koristimo samo negativne primjere

<u>2. NAČIN:</u> $$E(\vec{w}|D) = \sum_{i=1}^{N} \max(0, -\vec{w}^T \vec{x}^i y^i)$$

} opis grafa ▵ prošle stranice

⇒ **optimizacijski postupak:**

$\frac{\partial E(\vec{w}|D)}{\partial \vec{w}} = 0$ ⇒ nema rješenje u zatvorenoj formi, pa trebamo iterativnu optimizaciju

⇓

najjednostavnija iterativna optimizacija je GRADIJENTNI SPUST

↳ ideja gradijentnog spusta je da ažuriramo $\vec{x}$ sve dok ne dođemo do minimuma:

• krećemo se u malim koracima u smjeru suprotnom od gradijenta:

$$\vec{x}(k+1) = \vec{x}(k) - \eta \nabla f(\vec{x})$$

WIDROW-HOFF DELTA PRAVILO

η ⇒ stopa učenja

↳ o gradijentu:

a) $\nabla f(\vec{x}) = \left(\frac{\partial f(\vec{x})}{\partial x_0}, \frac{\partial f(\vec{x})}{\partial x_1}, \dots, \frac{\partial f(\vec{x})}{\partial x_m} \right)$

b) $\nabla f(\vec{x}) = \vec{0}$ ⇒ je nek ekstrem il' sedlo

• možemo: 1) sve članove sume funkcije pogreške derivirati, pa zbrojiti („batch error calculation")

2) jednog deriviranog, napravit korak („on-line error calculation")

⇒ DOBRE STRANE PERCEPTRONA:

- daje robusnu klasifikaciju
- jednostavan postupak

⇒ LOŠE STRANE PERCEPTRONA:

- izlaz modela nema vjerojatnosnu interpretaciju jer je samo -1 il 1
- rezultat može ovisit o početnoj točki mutor hiper-ravnine (ako minimum ima više točaka, tj. minimum je hiperravnina)
- ne konvergira ako primjeri nisu linearno razdvojivi (GLAVNI NEDOSTATAK)

DOPUNA 1. DOMAĆE ZADAĆE

(3) ⇒ PRISTRANOST JEZIKA - samim odabirom modela mi ograničavamo jezik (izražajnost) kojim se mogu opisati primjeri u prostoru

- ne znamo da se cijeli prostor može opisati modelom
- vezana je uz odabir modela

⇒ PRISTRANOST PRETRAŽIVANJA - ugrađena u algoritam minimizacije pogreške jer znamo uvijek raditi s nekom odabranom hipotezom (može se varijkovat između algoritama)

(4) $E_{p(x)}[X] = \sum_{X} X\, p(X)$

↖ očekivanje ovisi o distribuciji slučajne varijable X

↳ problem je što mi ne znamo distribuciju primjera, pa u praksi pretpostavljamo uniformnu distribuciju što znači da je naš uzorak reprezentativan

↳ što više primjera imamo, tim više će on padati na stvarnu distribuciju podataka

LOGISTIČKA REGRESIJA

⇒ algoritam rad vrlo dobru klasifikaciju (ne regresiju!)

↳ trebao bi riješiti probleme perceptrona i regresije za klasifikaciju

⇒ prisjetimo se modela perceptrona:

$$h(\vec{x};\vec{w}) = f(\vec{w}^T\vec{x})$$

↑

„f" je aktivacijska funkcija koja preslikava vrijednosti od $-\infty$ do ∞ u neki stisnuti interval

npr. $f: \mathbb{R} \to [0,1]$

$f: \mathbb{R} \to [-1,+1]$

$f: \mathbb{R} \to \{-1,+1\}$

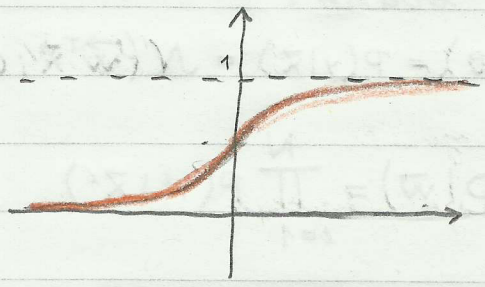
↳ stoga, ovo se zove „squashing function"

⇒ ovdje nećemo funkciju praga nego želimo nešto što je derivabilno!

↳ nazivat ćemo ju LOGISTIČKA FUNKCIJA

⇒ SIGMOIDALNA FUNKCIJA:

- jedna jako dobra logistička funkcija



$$\sigma(\lambda) = \frac{1}{1 + \exp(-\lambda)}$$

- derivacija sigmoide sadrži samu sigmoidu što je jako korisno:

$$\frac{\partial \sigma(\lambda)}{\partial \lambda} = \sigma(\lambda) \cdot [1 - \sigma(\lambda)]$$

⇒ model logističke regresije koristeći sigmoidalnu funkciju:

$$h(\vec{x}, \vec{w}) = \sigma(\vec{w}^T \phi(x)) = \frac{1}{1 + \exp(-\vec{w}^T \phi(x))}$$

↳ vidimo da je izlaz modela na intervalu od nule do jedan, pa izlaz možemo tumačit kao vjerojatnost:

$$h(\vec{x}, \vec{w}) = P(y = 1 \mid \vec{x})$$

BITNO!!

⇒ funkcija gubitka / pogreške klasične regresije:

↳ probabilistički, pogrešku **klasične** regresije smo tumačil ovako:

$$h(\vec{x};\vec{w}) = P(y|\vec{x}) = \mathcal{N}(\vec{w}^T\vec{x};\sigma^2)$$

$$P(D|\vec{w}) = \prod_{i=1}^{N} P(y^i|\vec{x}^i)$$

⋮ pomoću toga smo dobili log-vjerodostojnost

$$E(\vec{w}|D) = -\ln P(D|\vec{w})$$

↳ znači, maksimiziramo vjerojatnost da su podaci takvi kakve ih mi vidimo

- to je čest način izvođenja funkcije gubitka, pa ćemo ga primjenit i ovdje ☺

⇒ izvod funkcije gubitka logističke regresije:

BERNOULIJEVA VARIJABLA

$Y \in \{0, 1\}$

$$P(Y=1|\mu) = \begin{cases} \mu, & \text{ako } Y=1 \\ 1-\mu, & \text{ako } Y=0 \end{cases} = \overbrace{\mu^{Y}(1-\mu)^{1-Y}}^{\text{bolj zapis za izvod}}$$

↳ što daje vjerojatnost „μ"?

- pa, daje ju model jer on određuje vjerojatnost pozitivne klasifikacije
- stoga, uvrstimo model umjesto „μ":

$$P(Y^i=1|\vec{X}^i;\vec{w}) = h(\vec{X}^i;\vec{w})^{Y^i} \cdot (1-h(\vec{X}^i;\vec{w}))^{1-Y^i}$$

↳ sada treba maksimizirati izglednost kao što smo to radili kod klasične regresije:

$$P(D|\vec{w}) = \prod_{i=1}^{N} P(Y^i|\vec{X}^i) \quad / \ln$$

$$\ln P(D|\vec{w}) = \sum_{i=1}^{N} \ln P(Y^i|\vec{X}^i) = \sum_{i=N}^{N} \ln\left[h(\vec{X}^i;\vec{w})^{Y^i} \cdot (1-h(\vec{X}^i;\vec{w}))^{1-Y^i}\right]$$

$$= \sum_{i=1}^{N} \left[Y^i \ln h(\vec{X}^i;\vec{w}) + (1-Y^i)\ln(1-h(\vec{X}^i;\vec{w}))\right]$$

↳ sada, funkcija pogreške je samo negativna log-vjerodostojnost:

$$E(\vec{w}|D) = \frac{1}{N} \cdot \sum_{i=1}^{N} \left[-y^i \ln h(\vec{x}^i; \vec{w}) - (1-y^i) \ln \left(1 - h(\vec{x}^i; w)\right) \right]$$

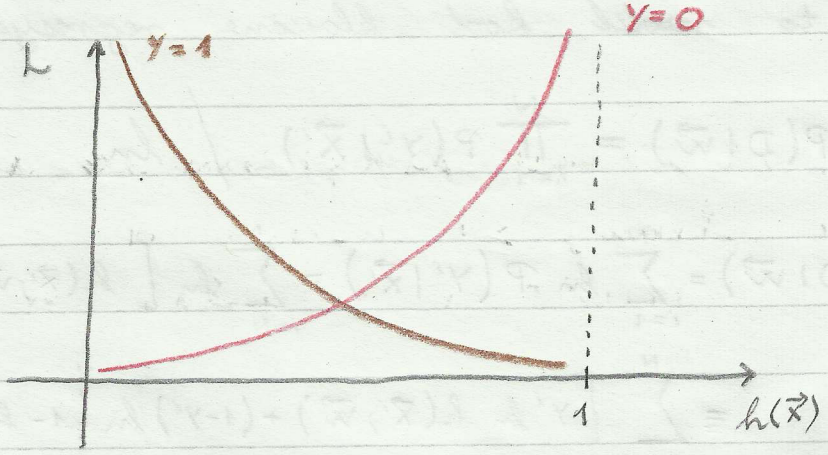
↑ samo uprosječujemo po svim primjerima

↳ ovo se naziva POGREŠKA UNAKRSNE ENTROPIJE

↳ funkcija gubitka je očekivanje funkcije pogreške:

$$L\left(y, h(\vec{x}; \vec{w})\right) = -y \ln h(\vec{x}; \vec{w}) - (1-y) \ln \left(1 - h(\vec{x}; \vec{w})\right)$$

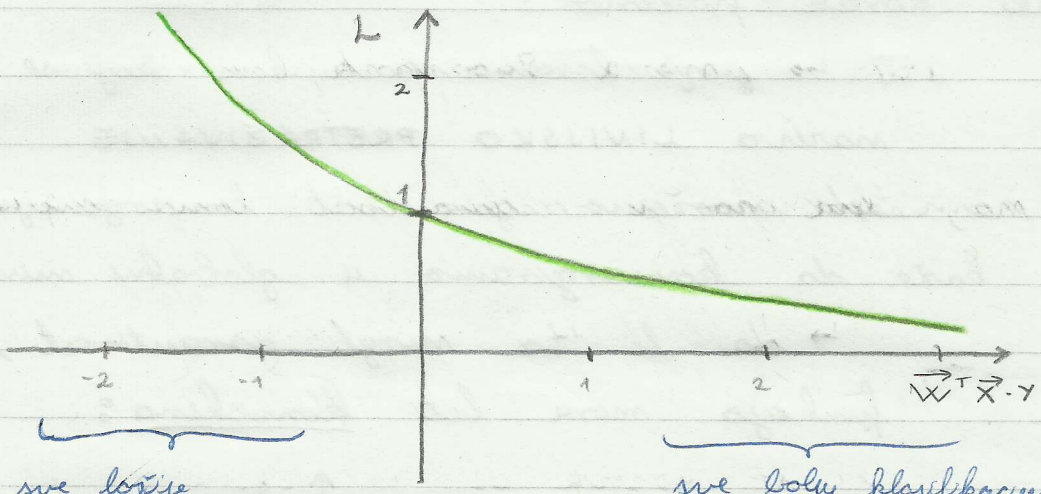
↳ ovo se naziva GUBITAK UNAKRSNE ENTROPIJE i izgleda ovako



⇒ sada bi još bilo super da nacrtamo funkciju gubitka u ovisnosti o „točnost klasifikacije" $\vec{w}^T \vec{x} \cdot y$:

• no, kako bi estetski bilo ljepše, funkciju gubitka ćemo podijeliti sa $\ln 2$ (ionako skalirano, pa to ne mijenja ništa):

$$L(y, h(\vec{x})) = \frac{1}{\ln 2} \cdot \ln\left(1 + \exp(-y\, \vec{w}^T \vec{x})\right)$$



sve lošije kad sve više

sve bolja klasifikacija kvačinog sve bolje (manje)

⇒ e sad, derivacija funkcije pogreške neće dati zatvorenu formu, pa trebamo drugačiji pristup

↳ koristit ćemo metodu **gradijentnog spusta**:

$$\vec{w} \leftarrow \vec{w} - \eta \nabla_{\vec{w}} E(\vec{w} | D)$$

⇒ **gradijentni spust**:

⇒ problem: preveliki „η" može uzrokovat iskakanje iz minimuma i postupak divergira, a premali daje jako sporu konvergenciju

⇒ želimo garanciju konvergencije (GLOBALNA KONVERGENCIJA), pa „η" moramo birat za svak korak posebno

↳ „η" biramo postupkom koj se nariva LINIJSKO PRETRAŽIVANJE

⇒ no, sada možemo garantirat konvergenciju, no tko kaže da konvergiramo u globalni minimum?

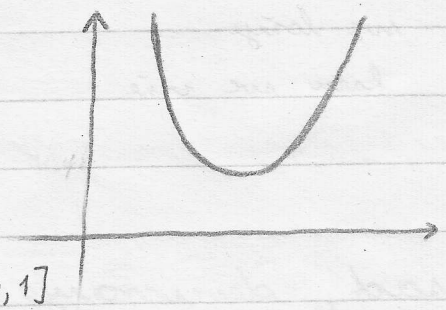
↳ da bi to mogl garantirat, funkcija mora bit **konveksna**:

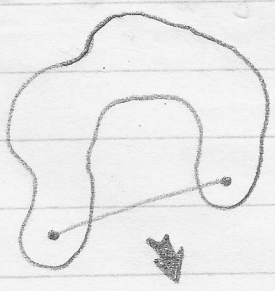
$f: \mathbb{R}^n \to \mathbb{R}$

① domena funkcije „f" je konveksan skup

② $\forall x_1, x_2 \in \text{domena}(f)$, $\lambda \in [0,1]$

$$f(\lambda x_1 + (1-\lambda) x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$





NIJE KONVEKSNO!

⇒ konveksne funkcije su nam jako poželjne u strojnom učenju

⇒ Izvod gradijentnog spusta

$$E(\vec{w}|D) = \frac{1}{N} \sum_{i=1}^{N} \left[-y^i \ln h(\vec{x}^i; \vec{w}) - (1-y^i) \ln \left(1 - h(\vec{x}^i; \vec{w}) \right) \right]$$

$$\nabla_{\vec{w}} E(\vec{w}|D) = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\vec{w}} L\left(y^i; h(\vec{x}^i; \vec{w}) \right)$$

$$\nabla_{\vec{w}} L\left(y, h(\vec{x}) \right) = \left(-\frac{y}{h(\vec{x})} + \frac{1-y}{1-h(\vec{x})} \right) h(\vec{x}) \left(1 - h(\vec{x}) \right) \vec{x}$$

$$\nabla_{\vec{w}} L\left(y, h(\vec{x}) \right) = \left(h(\vec{x}) - y \right) \cdot \vec{x}$$

REGULARIZIRANA LOGISTIČKA REGRESIJA

⇒ PREDNOSTI:

1) sprječavanje prejakane nelinearnosti
2) uklanjanje nepotrebnih značajki

} ovo je isto kao i kod klasične regresije

3) sprječavanje „otvrdnjavanje" sigmoide:

$\|\vec{w}\|$ raste ⇒ $\vec{w}^T\vec{x}$ raste ⇒ sigmoida je strmija

↓

ne želimo izgubit blag prijelaz, pa će regularizacija pomoć jer neće dati da težine nekontrolirano rastu

⇒ <u>L2 - REGULARIZACIJA:</u>

$$E(\vec{w}|D) = \frac{1}{N}\sum_{i=1}^{N}\left[-y^i \ln h(\vec{x}^i;\vec{w}) - (1-y^i)\cdot\ln\left(1-h(\vec{x}^i;\vec{w})\right)\right] + \frac{\lambda}{2}\vec{w}^T\vec{w}$$

↳ gradijentni spust je tada:

$$\vec{w} \leftarrow \vec{w} - \eta\left[\sum_{i=1}^{N}\left(h(\vec{x}^i) - y^i\right)\vec{x}^i + \lambda\vec{w}\right]$$

$$\vec{w} \leftarrow \vec{w}\underbrace{(1-\eta\lambda)} - \eta\sum_{i=1}^{N}\left(h(\vec{x}^i) - y^i\right)\vec{x}^i$$

„WEIGHT DECAY"
(odumiranje težina)

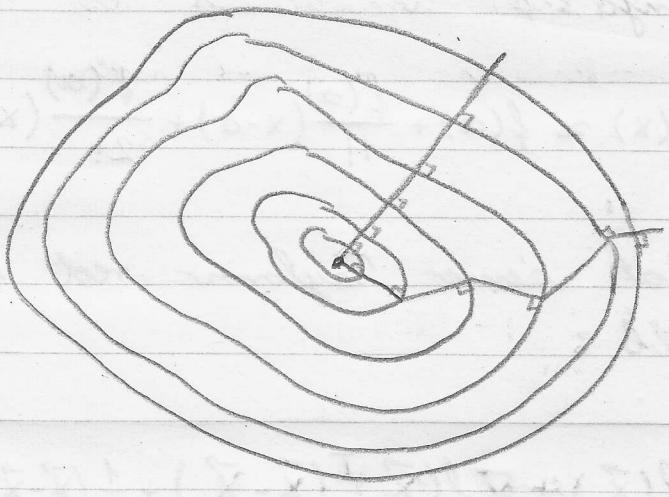
w_0 ne regulariziraj!

ALTERNATIVE GRADIJENTNOM SPUSTU

⇒ problem gradijentnog spusta je da može divergrati ako smo uzel prevelik pomak

⇒ dodatno, problem je do linijsk gradijentni spust može ići „cik-cak" po funkciji ako početna smjer (točka) nije dobar

↳ u svakom koraku idemo suprotno od gradijenta (okomito na linije funkcije)



⇒ TREBAMO PROMJENU GRADIJENTA (druga derivacija)

⇒ <u>Newton-Raphsonov postupak:</u>

- minimiziramo kvadratno odstupanje u svakom koraku

$$\vec{x}_{t+1} \leftarrow \vec{x}_t - \eta \nabla f(\vec{x}_t)$$

• trebamo funkciju rastavit u Taylorov red:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots$$

$$\boxed{f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^2}$$

↳ ovo je općenit Taylorov red za funkciju jedne varijable oko točke „a"

• nama će trebat samo Taylorov red drugog stupnja:

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2$$

• no, trebat ćemo Taylorov red za funkciju više varijabli:

$$\boxed{f(\vec{x}) \approx f(\vec{x}_t) + \nabla f(\vec{x}_t)^T(\vec{x}-\vec{x}_t) + \frac{1}{2}(\vec{x}-\vec{x}_t)^T H_t(\vec{x}-\vec{x}_t)}$$

ovo ima rješenje u zatvorenoj formi jer je ova kvadratna funkcija

↳ ovo je razvoj za oko točke $\vec{x}_t$

↳ H_t ⇒ Hesseova matrica (matrica drugih parcijalnih derivacija)

• <u>učenje</u>:

$$\vec{x}_{t+1} \leftarrow \vec{x}_t - \eta H_t^{-1} \nabla f(\vec{x}_t)$$

⇒ PRIMJENA NEWTON-RAPHSONOVOG POSTUPKA NA LOGISTIČKU REGRESIJU 8

↳ ber izvoda, možemo vidjeti da je:

$$H = \Phi^T S \Phi \qquad \Phi \to N \times (n+1)$$

$$S = diag\left(h(\vec{x}^{(i)})(1-h(\vec{x}^{(i)}))\right) \qquad S \to N \times N$$

↳ pokazuje se da je H pozitivno definitna, a to znači da je $\vec{x}^T H \vec{x} > 0$ za $\vec{x} \neq \vec{0}$

↳ ovo ubacimo gradijent i minimiziramo, dobivamo:

$$\vec{w} \leftarrow \vec{w} - H^{-1} \nabla E(\vec{w} | D) \qquad (\eta = 1)$$

↳ ovo se naziva IRLS!

↳ problem je u inverzu Hesseove matrice (može biti skupo jer je ova dimenzija $(n+1)\times(n+1)$)

↳ to se rješava aproksimacijom matrice H drugom potpuku:

- Quasi-Newton
- algoritam koj to daje je BFSG

↳ ako pak je H prevelika da stane u memoriju, sama aproksimacija inverza nije dovoljna, pa smanjujemo dimenzionalnost algoritmom L-BFSG

⇒ REGULARIZACIJA ZA NEWTON-RAPHSONOV POSTUPAK:

↳ samo ovoj za formule regularizacija bez izvoda

↳ L2-REGULARIZACIJA:

$$\nabla E_R(\vec{w}|D) = \nabla E(\vec{w}|D) + \lambda \vec{w}$$

$$H_R = H + \lambda I$$

↳ L1-REGULARIZACIJA:

- nije nam derivabilno, pa minimum oblično tražimo metodom koordinatnog spusta
- nije derivabilno zbog apsolutne vrijednosti iz L1 norme

VIŠEKLASNA LOGISTIČKA REGRESIJA

⇒ problem OVO i OVR klasifikacije što suma vjerojatnosti pripadnosti ne daje 1

↳ loše za interpretaciju

⇒ klasifikacija u dvije klase (podsjetnik):

$$h(\vec{x}; \vec{w}) = \sigma(\vec{w}^T \Phi(\vec{x})) = P(y=1 \mid \vec{x})$$

⇒ <u>multinomijalna klasifikacija:</u>

$$h_k(\vec{x}; \underbrace{\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_K}_{W}) = \underbrace{\frac{\exp(\vec{w}_k^T \Phi(\vec{x}))}{\sum_{j=1}^{K} \exp(\vec{w}_j^T \Phi(\vec{x}))}}_{\text{SOFT-MAX POSTUPAK}} = P(y=k \mid x, W)$$

⇒ sada trebamo funkciju pogreške jer gore imamo dobar model:

↳ no za to trebamo dodatne pojmove iz matematike / vjerojatnosti

↳ to nam treba za prikaz varijable za više klasa ($K > 2$) jer nam jedna slučajna varijabla više dovoljna

⇒ KATEGORIČKA VARIJABLA

↳ slučajna varijabla za više klasa

↳ „multi-nuljeva" varijabla

$$\vec{y} = (y_1, y_2, \dots, y_K) \qquad (\text{npr. } \vec{y} = (0,0,1,0))$$

$$P(\vec{y} = j \mid \vec{\mu}) = \prod_{k=1}^{K} \mu_k^{y_k}$$

↳ napišimo log-izglednost:

$$\ln P(D \mid W) = \ln \prod_{i=1}^{N} P(y^i \mid \vec{x}^i) =$$

$$= \ln \prod_{i=1}^{N} \prod_{k=1}^{K} \mu_k^{y_k^i} \longrightarrow h_k(\vec{x})$$

(model koji kaže kojoj klasi pripada)

$$= \ln \prod_{i=1}^{N} \prod_{k=1}^{K} h_k(\vec{x}^i \mid W)^{y_k^i} =$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} \ln \left[h_k(\vec{x}^i \mid W)^{y_k^i} \right] =$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} y_k^i \ln \left[h_k(\vec{x}^i \mid W) \right]$$

↳ funkcija pogreške je stoga:

$$E(W \mid D) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_k^i \ln h_k(\vec{x}^i; W)$$

↳ funkcija gubitka:

$$L(\vec{y}; h_k(\vec{x})) = -\sum_{k=1}^{K} y_k \ln h_k(\vec{x}; W)$$

⇒ posljednje, treba nam optimizacijski postupak:

$$\nabla_{\vec{w}_k} E(\mathbf{w}|\mathcal{D}) = \sum_{i=1}^{N} \left[h_k(\vec{x}^i; \mathbf{w}) - y_k^i \right] \vec{x}^i$$

↳ iz ovog dobijemo izraz za ažuriranje težina koj najčešće koristimo:

$$\vec{w}_k \leftarrow \vec{w}_k - \eta \left[h_k(\vec{x}; \mathbf{w}) - y_k^i \right] \vec{x}^i$$

ADAPTIVNE BAZNE FUNKCIJE

⇒ **BAZNE FUNKCIJE**: funkcije koje smo koristili za preslikavanje u viši prostor

$$\Phi(\vec{x}) = \left(\Phi_0(\vec{x}), \Phi_1(\vec{x}), \dots, \Phi_m(\vec{x}) \right)$$

$\{ \Phi_0, \Phi_1, \dots, \Phi_m \}$ ➡ bazne funkcije

$$\Phi_j : \mathbb{R}^M \to \mathbb{R}$$

PR: Funkcija za preslikavanje funkcije dvije varijable u stupanj dva:

$$\Phi(\vec{x}) = \left(1, x_1, x_2, x_1^2, x_2^2 \right)$$

⇒ **POOPĆENI LINEARNI MODEL**:

$$h(\vec{x}; \vec{w}) = f\left(\vec{w}^T \Phi(\vec{x}) \right) = f\left(\sum_{j=0}^{m} w_j \Phi(\vec{x}) \right)$$

⇒ ideja je da učimo i funkcije za preslikavanje i to ugradimo u model

↳ to ćemo probati ugraditi u opći linearni model

$$h(\vec{x};\vec{w}) = f\left(\sum_{j=0}^{m} w_j \cdot \underbrace{f\left(\sum_{i=0}^{n} w_{ji} x_i\right)}_{\Phi_j(\vec{x})}\right)$$

adaptivna bazna funkcija

$$h(\vec{x};\vec{w}) = f\left(\vec{w}^T f(W\vec{x})\right)$$

gornju sumu možemo prepoznati kao funkciju za izlaz neuronske mreže!

⇒ iz rečenog, zaključujemo da je neuronska mreža složenija od linearnog modela te je time izražajnija

↳ no, to plaćamo slabijom mogućnosti interpretacije, složenijim optimizacijskim postupkom, većom mogućnosti prenaučenosti, ...

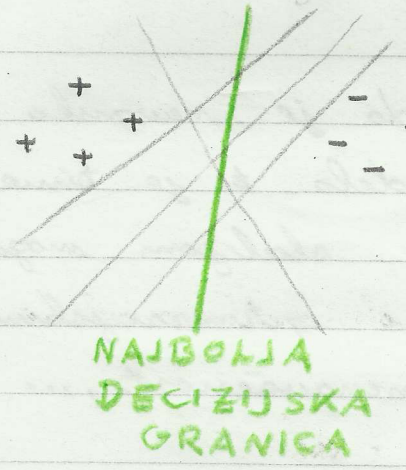
STROJ POTPORNIH VEKTORA

⇒ **model SVM-a:**

↳ nešto drugo no jednostavni linearni model

$$h(\vec{x}; \vec{w}) = \vec{w}^T \vec{x}$$

⇒ pretpostavimo da su primjeri linearno odvojivi:

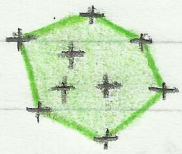


cilj SVM-a je pronaći najbolju decizijsku granicu

⇒ MARGINA ⇒ udaljenost najbližih primjera od razdvajajuće hiperravnine

⇒ cilj SVM-a je naći najbolju razdvajajuću hiperravninu, a to je ona koja ima najveće margine

⇒ KONVEKSNA LJUSKA ⇒ spajanje najvanjskijh primjera nekog razreda



} ovo je primjer konveksne ljuske

↳ ideja je naći takvu descrizku granicu koja je na polovici spojnica konveksnih ljusaka



⇒ kreemo vidjeti sada kako doći do najbolje razdvajajuće hiperravnine:

$$h(\vec{x}; \vec{w}) = \vec{w}^T \vec{x} + w_0$$

↳ pretpostavka: primjeri su linearno razdvojivi, pa sigurno postoje težine za koje model ne griješi

↳ pišemo: $\exists\, w_0, \vec{w}$ t.d. $\forall (\vec{x}^i, r^i) \in D$, $y^i h(\vec{x}) \geq 0$

$h(\vec{x}) = 0$ ⇒ jednadžba hiperravnine

$y \in \{-1, 1\}$ ⇒ domena izlaza

$y = \text{sgn}(h(\vec{x}))$ ⇒ izlaz modela

↳ prisjetimo se:

• udaljenost primjera od hiperravnine:

$$d = \frac{h(\vec{x})}{\|\vec{w}\|}$$

↳ stoga, udaljenost do najbližeg primjera (tj. udaljenost do konveksne ljuske):

$$\min_i \left\{ \frac{y^i \cdot h(\vec{x}^i)}{\|\vec{w}\|} \right\} = (*)$$

$\|\vec{w}\|$ ne ovisi o „i"

$$(*) = \frac{1}{\|\vec{w}\|} \min_i \left\{ y^i \cdot h(\vec{x}^i) \right\} = \frac{1}{\|\vec{w}\|} \min_i \left\{ y^i \cdot (\vec{w}^T \vec{x}^i + w_0) \right\}$$

↳ stoga, maksimalna margina je dana za sljedeće težine:

$$\arg\max_{\vec{w}, w_0} \left\{ \frac{1}{\|\vec{w}\|} \min_i \left\{ y^i \cdot (\vec{w}^T \vec{x}^i + w_0) \right\} \right\}$$

⇓

onako je definiran

problem maksimalne margine

⇒ za najbliže primjere vrijedi: $y^i(\vec{w}^T \vec{x}^i + w_0) = 1$

(zato jer vektor $\vec{w}$, w_0 vrijedi množimo skalirati da to vrijedi!)

⇒ za ostale primjere vrijedi: $y^i(\vec{w}^T \vec{x}^i + w_0) \geq 1$

⇒ sada, vratimo se u problem maksimalne margine i uvrstimo izraz za najbliže primjere:

$$\underset{\vec{w}, w_0}{\text{argmax}} \left\{ \frac{1}{\|\vec{w}\|} \cdot 1 \right\}$$

ovo mora vrijediti! ← $y^i(\vec{w}^T \vec{x}^i + w_0) \geq 1 \qquad i = 1, \ldots, N$

⇒ <u>stoga, PROBLEM MAKSIMALNE MARGINE glasi:</u>

$$\underset{\vec{w}, w_0}{\text{arg min}} \left\{ \frac{1}{2} \|\vec{w}\|^2 \right\}$$

tako da vrijedi $y^i(\vec{w}^T \vec{x}^i + w_0) \geq 1 \qquad i = 1, \ldots, N$

⇓

konveksna optimizacija uz ograničenja

↳ kvadriramo $\|\vec{w}\|$ da dobijemo konveksnu funkciju

↳ ovo se još naziva kvadratno programiranje

LAGRANGEOVA DUALNOST

⇒ metoda rješavanja problema kvadratnog programiranja s prošle stranice

⇒ POČETNI PROBLEM:

- minimizirati $f(\vec{x})$

  ↳ uz ograničenja $g_i(\vec{x}) \leq 0 \quad i = 1, \dots, m$

  $h_j(\vec{x}) = 0 \quad j = 1, \dots, p$

- ugradimo ograničenja u Lagrangeovu funkciju:

$$L(\vec{x}, \vec{\lambda}, \vec{\beta}) = f(\vec{x}) + \sum_{i=1}^{m} \lambda_i g_i(\vec{x}) + \sum_{i=1}^{p} \beta_i h_i(\vec{x}) \quad ; \lambda_i \geq 0$$

λ_i, β_i ⇒ Lagrangeovi multiplikatori

- rješenje je stacionarna točka Lagrangeove funkcije, pa nju trebamo derivirati i izjednačiti s nulom:

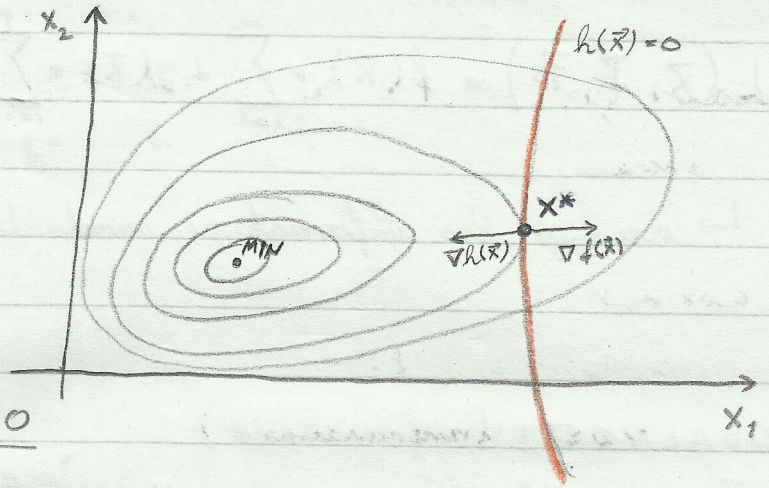
$$\nabla L = 0$$

↳ u točki minimuma (ostvarenog minimuma uz dana ograničenja) znamo da su $\nabla h(\vec{x})$ i $\nabla f(\vec{x})$ kolinearni, pa možemo napisati:

$$\nabla f(\vec{x}) + \beta \nabla h(\vec{x}) = 0$$

⇒ ovo napisano vrijedi za ograničenja jednakosti

↳ ilustracija kolinearnosti u dvije dimenzije:



$$\nabla L = 0$$

$$\nabla f(\vec{x}) + \beta \nabla h(\vec{x}) = 0$$

$$\nabla L = \nabla f(\vec{x}) + \beta \nabla h(\vec{x}) \quad //$$

x^* ⇒ najbolji mogući minimum uz ograničenja

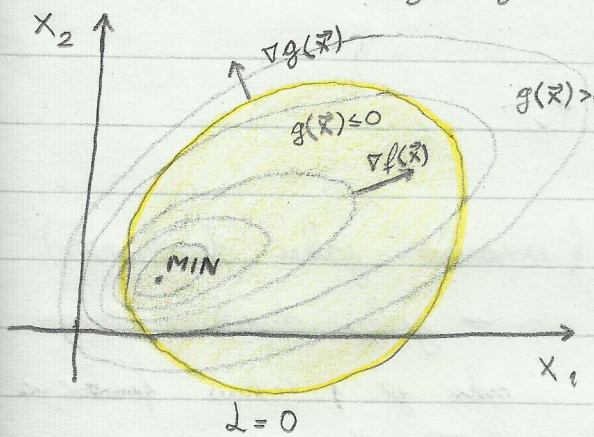
⇒ za ograničenja nejednakosti imamo dva slučaja:

a) NEAKTIVNO OGRANIČENJE

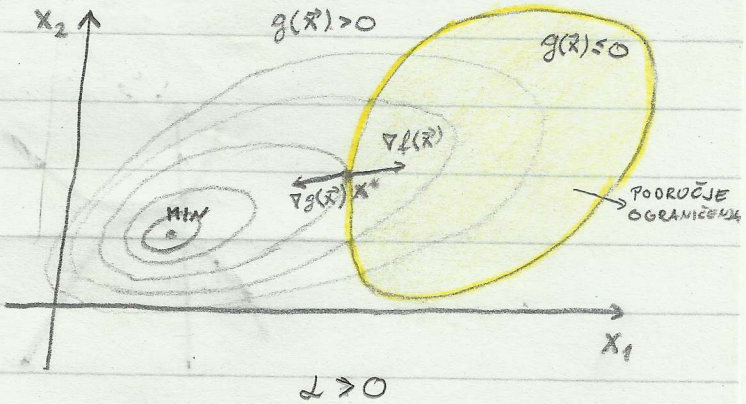
↳ minimum je moguće doseći

b) AKTIVNO OGRANIČENJE

↳ minimum nije moguće doseći



$\lambda = 0$



$\lambda > 0$

↳ vidimo da vrijedi:

$$\nabla f(\vec{x}) = -\lambda \nabla g(\vec{x}) = 0$$

⇒ ukombiniramo ograničenja nejednakost i jednakost te dobivamo baš Lagrangeovu funkciju:

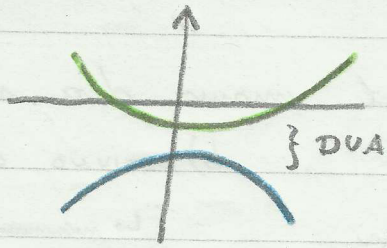
$$L(\vec{x}, \vec{\lambda}, \vec{\beta}) = f(\vec{x}) + \sum_{i=1}^{m} \lambda_i g_i(\vec{x}) + \sum_{i=1}^{n} \beta_i h_i(\vec{x})$$

↳ ovo je bio informativni uvod Lagrangeove funkcije

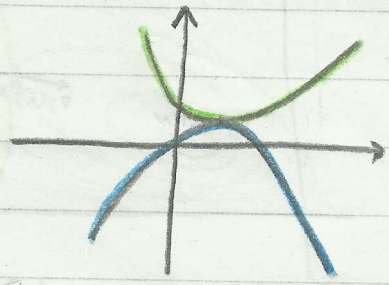
⇒ NAČELO DUALNOSTI (informativno)

a) primarni problem: minimizacija $f(\vec{x})$

b) dualni problem: nalaženje donje ograde primarnog problema



(imamo tada slabu dualnost)



(imamo jaku dualnost)

↳ jaku dualnost imamo ako je početna funkcija $f(\vec{x})$ konveksna

↓

to će biti slučaj jer optimiramo problem kvadratnog programiranja

⇒ LAGRANGEOVA DUALNOST:

↳ uvjetno minimum po $\vec{x}$: $\nabla_{\vec{x}} L(\vec{x}, \vec{\lambda}, \vec{\beta}) = 0$

$$\widetilde{L}(\vec{\lambda}, \vec{\beta}) = \min_{\vec{x}} L(\vec{x}, \vec{\lambda}, \vec{\beta})$$

⇓

dualna Lagrangeova funkcija

(zbog načela dualnosti, ovo je donja ograda primarnog problema, pa da minimiziram po $\vec{x}$ početnu funkciju, trebam maksimizirati njenu dualnu funkciju, što je (nadamo se) znatno lakše)

OPTIMIZACIJA MAKSIMALNE MARGINE

⇒ rješavamo problem maksimalne margine korištenjem prethodno izvedene i pokazane Lagrangeove funkcije i njezine dualnosti

⇒ $$\arg\min_{\vec{w}, w_0} \left\{ \frac{1}{2} \|\vec{w}\|^2 \right\}$$

$$y^i \left(\vec{w}^T \vec{x}^i + w_0 \right) \geq 1 \quad , \quad i = 1, \ldots, N$$

↳ pišemo Lagrangeovu funkciju koja uvrne u obzir ograničenja:

$$L\left(\vec{w}, w_0 ; \vec{\lambda}\right) = \frac{1}{2} \|\vec{w}\|^2 - \sum_{i=1}^{N} \lambda_i \cdot \left(y^i \left(\vec{w}^T \vec{x}^i + w_0 \right) - 1 \right)$$

$$\vec{\lambda} = \left(\lambda_1, \ldots, \lambda_N \right) \; ; \; \lambda_i \geq 0$$

↳ stoga, kao što smo rekli, deriviramo po $\vec{w}$ i tražimo stacionarnu točku tako da izjednačimo s nulom:

$$\frac{\partial L\left(\vec{w}, w_0, \vec{\lambda}\right)}{\partial \vec{w}} = 0 \quad \xrightarrow{\text{dobivamo}} \quad \vec{w} = \sum_{i=1}^{N} \lambda_i y^i \vec{x}^i$$

$$\frac{\partial L\left(\vec{w}, w_0, \vec{\lambda}\right)}{\partial w_0} = 0 \quad \xrightarrow{\text{dobivamo}} \quad \sum_{i=1}^{N} \lambda_i y^i = 0$$

↳ sa dobivenim derivacijama po primarnim parametrima dobivamo dualnu Lagrangeovu funkciju:

$$\tilde{L}(\vec{I}) = \ldots = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y^i y^j \vec{x}^{iT} \vec{x}^j$$

⬇

dualna Lagrangeova funkcija,
pa nju želimo maksimizirat
↳ to nećemo radit pješice jer ovo
radi algoritam SMO (engl. „Sequential Minimal
Optimisation") jer je ovo, dalje problem
kvadratnog programiranja
(složenost $O(N^2)$)

⇒ zašto smo ovo radili ako smo dobili opet problem kvadratnog programiranja?
↳ pa, prebacili smo se na drugu varijablu
u dualnoj formulaciji (na $\vec{I}$)
↳ $\vec{I}$ je dimenzije N (broj primjera),

⇒ VEZA PRIMARNOG I DUALNOG MODELA:

a) primarni:

$$h(\vec{x}; \vec{w}, w_0) = \vec{w}^T \vec{x} + w_0$$

b) dualni:

$$h(\vec{x}; \vec{\lambda}; D) = \sum_{i=1}^{N} \lambda_i y^i \vec{x}^{iT} \vec{x} + w_0$$

sličnost vektora

↳ ako je $\lambda_i = 0$, i-ti vektori su u ovoj sumi preskočeni

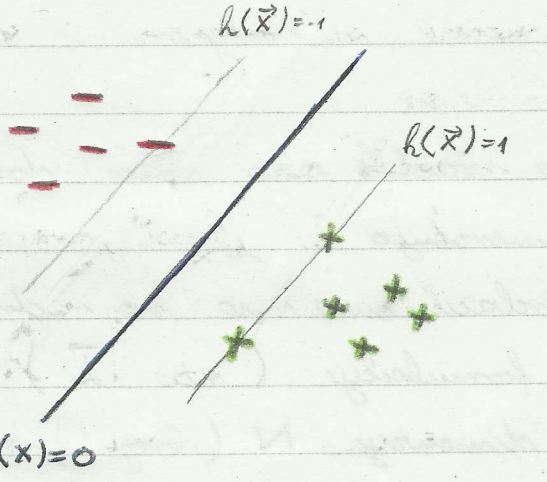
↳ ako je $\lambda_i \neq 0$, tada je:

$$y^i h(\vec{x}^i) - 1 = 0$$

(prirodan uz ograničenja:

$$\lambda_i (y^i h(\vec{x}^i) - 1) = 0$$)

$y^i h(\vec{x}^i) = 1$



potporni vektori su oni primjeri koji daju na izlaz modela jedan ili minus jedan:

$$h(\vec{x}) = \pm 1$$

MEKA MARGINA

→ dualna varijabla

⇒ vektor λ će biti takav da maksimizira margine ako su primjeri linearno razdvojivi

↳ no što ako primjeri nisu linearno razdvojivi?

- tada nećemo dobiti rješenje, pa trebamo drugo rješenje!

⇒ neke ideje rješenja problema linearne nerazdvojivosti:

a) preslikavanje u viši prostor

- pa, nije loše, ali nemamo garanciju da ćemo uspjeti naći taj prostor na kojoj model generalizira i uspjećemo linearno odvojiti

b) meka margina

- kao i kod prijašnjih modela, dopustit ćemo pogrešnu klasifikaciju primjera, ali ćemo ih kažnjavati
- stoga, moramo malo izmijeniti uvjete optimizacije kako bi bili u mogućnosti prihvatiti pogrešno klasificirane primjere

⇒ dopuštanje krivo klasificiranih primjera:

"SLACK" varijabla ξ_i

$$\boxed{\xi_i \geq 0} \qquad i = 1, \ldots, N$$

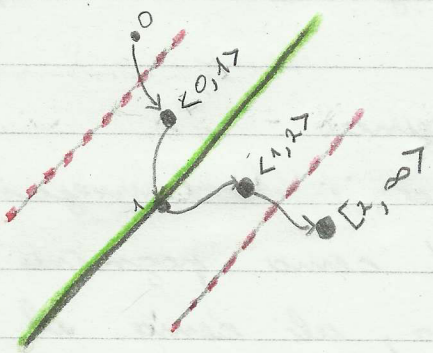
$\xi_i = 0$ ⇒ primjer je ispravno klasificiran

$0 < \xi_i < 1$ ⇒ ispravno klasificiran, ali unutar margine, pa ovo malo kažnjavamo

$\xi_i = 1$ ⇒ primjer je na granici klasifikacije

$1 < \xi_i < 2$ ⇒ primjer je na drugoj strani, ali između dedrijske granice i druge margine

$\xi_i > 2$ ⇒ primjer je prešao skroz na drugu stranu margine



⇒ računanje slack varijable:

$$\boxed{\xi_i = |y^i - h(\vec{x}^i)|}$$

y^i: $+1$, -1 ; $h(\vec{x}^i)$: $\in \mathbb{R}$

⇒ mi želimo minimizirati sljedeće:

$$\underset{\vec{w}, w_0}{\arg\min} \left\{ \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^{N} \xi_i \right\}$$

↳ hiperparametar C: $C > 0$

- određuje kompromis između širine margine i tvrdoće margine (ne možemo imati oboje)
- veći C daje složeniji model jer on više kažnjava krivu klasifikaciju, pa se model bolje prilagođava podacima

$$\underset{w_0, \vec{w}, \vec{\xi}}{\arg\min} \left\{ \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^{N} \xi_i \right\}$$

- ograničenja: $y^i (\vec{w}^T \vec{x}^i + w_0) \geq 1 - \xi_i$

$$\xi_i \geq 0$$

↳ da je $\xi_i < 0$, tada nagrađujemo dobru klasifikaciju, a mi samo želimo kazniti pogrešku

⇒ sada ovo treba minimizirati, a to radimo tako da napišemo Lagrangeovu funkciju i prijeđemo u dualni problem (isto kao prije)

⇒ **MINIMIZACIJA:**

• prvo Lagrangeovu funkciju:

$$L(\vec{w}, w_0, \vec{\xi}, \vec{\alpha}, \vec{\beta}) = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y^i(\vec{w}^T\vec{x}^i + w_0) - 1 + \xi_i\right] - \sum_{i=1}^{N}\beta_i\xi_i$$

KKT: (uvjeti)

$$y^i(\vec{w}^T\vec{x}^i + w_0) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$
$$\alpha_i \geq 0$$
$$\beta_i \geq 0$$
$$\alpha_i(y^i(\vec{w}^T\vec{x}^i + w_0) - 1 + \xi_i) = 0$$
$$\beta_i\xi_i = 0$$

$i = 1, \ldots, N$

• sada ovo moramo derivirat po primarnim parametrima, izjednačimo sa nulom jer je rješenje ekstrem il sedlo kod La Grangea:

$$\frac{\partial L}{\partial \vec{w}} = 0 \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^{N}\alpha_i y^i \vec{x}^i$$

$$\frac{\partial L}{\partial w_0} = 0 \quad \Rightarrow \quad \sum_{i=1}^{N}\alpha_i y^i = 0$$

$$\frac{\partial L}{\partial \xi} = 0 \quad \Rightarrow \quad \alpha_i = C - \beta_i$$

sada s ovime izlenemo natrag u La Grangeovu funkciju

• umstimo dobiveno u LaGrangeovu funkciju:

$$L(\vec{\lambda}) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y^i y^j (\vec{x}^i)^T \vec{x}^j$$

↳ uz ograničenja: $\beta_i \geq 0$, $\lambda_i = C - \beta$, $\lambda_i \geq 0$ $\Rightarrow$ $0 \leq \lambda_i \leq C$

„BOXED CONSTRAINTS"

• ovo je dualni problem, pa tražimo maksimum:

$$\underset{\vec{\lambda}}{\operatorname{argmax}} \left\{ L(\vec{\lambda}) \right\}$$

⇒ PREDIKCIJA:

• predikciju radimo na sljedeći način:

$$h(\vec{x}) = \sum_{i=1}^{N} \lambda_i y^i (\vec{x}^i)^T \vec{x} + w_0$$

↳ **potporni vektori:**

• oni primjeri koj su u margini:

$$0 < \lambda_i \leq C$$

$\lambda_i < C \Rightarrow$ leže na margini

$\lambda_i = C \Rightarrow$ leže u margini

GUBITAK ZGLOBNICE

⇒ prisjetimo se primarnog (ne dualnog) optimizacijskog problema:

$$\underset{\vec{w}, w_0, \xi}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\vec{w}\|^2 + C \sum_{i=1}^{N} \xi_i \right\}$$

↳ prisjetimo se funkcijom pogreške za L2-regularizirani regresiju:

$$E_R(\vec{w}|D) = \sum_{i=1}^{N} L\left(y^i, h(\vec{x}^i)\right) + \frac{\lambda}{2}\|w\|^2 \qquad \left(\text{uz } C = \frac{1}{\lambda}\right)$$

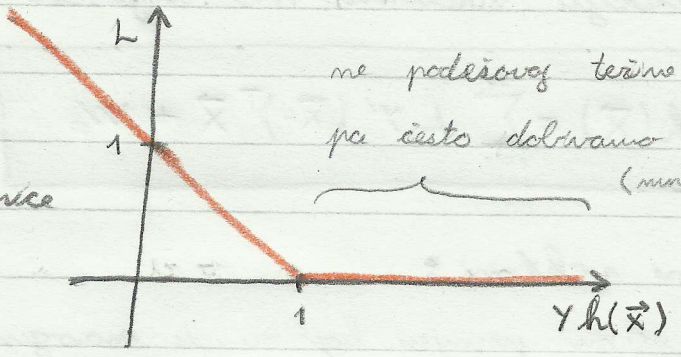
$$y^i h(\vec{x}^i) \geq 1$$

$$\xi_i = |y^i - h(\vec{x}^i)| = 1 - y^i h(\vec{x}^i)$$

↳ **funkcija gubitka:**

$$L\left(y^i, h(\vec{x}^i)\right) = \max\left(0, 1 - y^i h(\vec{x}^i)\right)$$

"HINGE LOSS"
gubitak zglobnice



ne podešavaj težine na dobru klasifikaciju
pa često dobivamo rijetke modele
(mnoge težine jednake nuli)

kazni one ispravno klasificirane, ali koji su uš unutar margine

BITNE NAPOMENE O SVM-U

⇒ NELINEARNOST: samo preslikamo u viši prostor (isto ko prije)

⇒ SVM REGRESIJA:

- mi smo naprovli klasifikaciju, no moguće je naprovit regresiju:
  - ↳ ideja je sada smjestit sve primjere unutar margine, a ne odvajamo ih

⇒ HIPERPARAMETAR C:

- SVM je osjetljiv na promjene C-a, pa ga „štelamo" unakrsnom provjerom

⇒ SKALIRANJE:

- npr. kreditna sposobnost

  $\vec{x} = (\langle dob \rangle, \langle prihod \rangle)$

  mali brj, veliki brj

  (treba normalizirati !)

⇒ VIŠEKLASJE:

- SVM je osjetljiv na razlike u broju uzoraka između klasa, pa je OVO bolj

⇒ PROBABILISTIČKI IZLAZ:

- baš ga nema, pa trebamo sljedeći izraz (logistička regresija):

$$P(y=1 \mid \vec{x}) = \sigma\left(\vec{w}\, h(\vec{x}) + w_0\right)$$

↑ izlaz SVM-a

JEZGRENE METODE

⇒ JEZGRENE FUNKCIJE:

↳ do sada smo uvijek pretpostavljal da su nam ulazi vektori $\vec{x}$:

$$\vec{x} \in X, \quad X = R^n$$

↳ no, kako bi, na primjer, riječi prikazal kao vektor značajki il DNA ljudi prikazal kao vektor

↳ moguće je, ali je možda nepotrebno i preteško

↳ možda je tada lakše pričati o sličnosti među primjerima

↳ JEZGRINA FUNKCIJA („kernel function"):

- mjera sličnost između primjera

$$K: X \times X \to R \qquad K: X \times X \to [0, 1]$$

$$K(\vec{x}, \vec{x}') \geq 0 \qquad \text{(similarity measure)}$$

$$K(\vec{x}, \vec{x}') = K(\vec{x}', \vec{x})$$

⇒ <u>neke jezgrine funkcije</u>:

a) LINEARNA JEZGRA:

$$K(\vec{x}, \vec{x}') = \vec{x}^T \vec{x}'$$

b) GAUSSOVA JEZGRA:

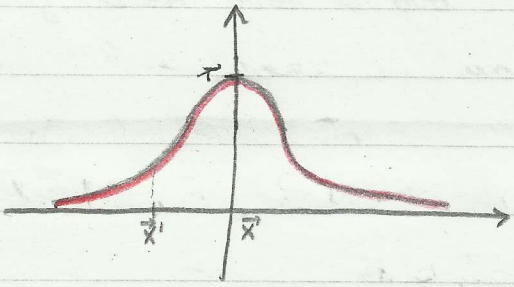
↳ najčešće korištena jezgra

$$K(\vec{x}, \vec{x}') = \exp\left(-\frac{\|\vec{x}-\vec{x}'\|^2}{2\sigma^2}\right) = \exp\left(-\gamma\|\vec{x}-\vec{x}\|^2\right)$$

$$\|\vec{x}-\vec{x}'\|^2 = (\vec{x}-\vec{x}')^T(\vec{x}-\vec{x}') = \sum_{j=1}^{n}(x_j - x'_j)^2$$

↳ koeficijent γ se naziva **preciznost**:

- što je γ veći, gaussova funkcija je uža



⇒ <u>radijalne bazne funkcije</u>:

- općenita klasa koja u sebi sadrži jezgrene funkcije
- ima mnogo tipova:
  - ↳ gaussova je primjer eksponencijalne
  - ↳ postoje inverzne:

$$K(\vec{x}, \vec{x}') = \frac{1}{1+\|\vec{x}-\vec{x}'\|^2}$$

- ne moramo nužno koristiti Euklidsku udaljenost (ova je Euklidska udaljenost $\|\vec{x}-\vec{x}'\|$)

• ako koristimo Mahalanobisovu udaljenost, dobivamo sljedeću Gaussovu barem funkciju:

$$K(\vec{x}, \vec{x}') = \exp\left(-\frac{1}{2}(\vec{x}-\vec{x}')^T \cdot \underbrace{\Sigma^{-1}}_{\text{kovarijacijska matrica}} (\vec{x}-\vec{x}')\right)$$

$$\Sigma = \text{dijag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_m^2)$$

$$K(\vec{x}, \vec{x}') = \exp\left(-\frac{1}{2}\sum_{j=1}^{m} \frac{1}{\sigma_j^2}(x_j - x_j')^2\right)$$

ovo se zove ARD jezgra

⇒ sada smo malo pokarali varijante norme i jezgrene funkcije no još nismo pokarali kako raditi sa riječima:

(i dalje ćemo navike pišemo vektore iako riječi nisu vektori)

⇒ STRING KERNEL:

- služi za usporedbu sličnosti među riječima tako da uočimo supstringove gdje se riječi poklapaju

Zagreb – Zadar: Za, a, r

$$K(\vec{x}, \vec{x}') = \sum_{s \in A^*} c(s, \vec{x}) \cdot c(s, \vec{x}')$$

⇒ za kompliciranije probleme koristimo još:

- TREE KERNELS
- GRAPH KERNELS

JEZGRENI STROJEVI

⇒ umjesto polinomijalnih baznih funkcija za preslikavanje u viši prostor, sada ćemo koristiti jezgrine funkcije

⇒ prisjetimo se: $\Phi(\vec{x}) = \left(\Phi_0(\vec{x}), \Phi_1(\vec{x}), \ldots, \Phi_m(\vec{x})\right)$

$\Phi_j : R^n \to R$

↑ bazna funkcija

⇒ sada koristimo jezgrene funkcije:

$$\Phi(\vec{x}) = \left(1, K(\vec{x}, \vec{\mu}_1), K(\vec{x}, \vec{\mu}_2), \ldots, K(\vec{x}, \vec{\mu}_m)\right)$$
$$\vec{\mu}_j \in X$$

↳ ovo je JEZGRENI STROJ
(kernel machine)

⇒ <u>μ_j su centroidi</u> i njih treba pametno stavit kako bi u tom novom prostoru dobili linearno odvojene primjere

↳ centroidi definiraju točke u odnosu na koje se odredjuje sličnost i tako „razapinju" prostor

⇒ stoga, problem je sada redefiniran:
Kako razmjestiti centroide?

⇒ **RJEŠENJE:**

• pa ne želim stavit centroid negdje gdje nema ni jednog primjera, nego ih stavljam na svaki primjer:

↳ od ulaza se gleda udaljenost do najbližeg primjera:

$$\Phi(\vec{x}) = \left(1, K(\vec{x}, \vec{x}^1), K(\vec{x}, \vec{x}^2), \ldots, K(\vec{x}, \vec{x}^N)\right)$$

↳ no, sada imamo velku dimenzionalnost jer je centroid na svakom primjeru ($m = N$)

↳ to rješavamo **L1-regularizacijom:**

• L1-regularizacija strogo pritiše mnoge težine prema nuli i time će puno centroida otpasti

• ovi centroid koj ostaju su ostali zbog najbitnijih primjera za učenje (ovi koj nose najviše informacije)

JEZGRENI TRIK

⇒ kernel funkcije možemo primjeniti direktno unutar algoritma SVM-a

⇒ prisjetimo se modela dualnog SVM-a:

ovo ugrađivanje u SVM je jezgreni trik

$$h(\vec{x}; \vec{\lambda}, \mathcal{D}) = \sum_{i=1}^{N} \lambda_i y^i \underbrace{(\vec{x}^i)^T \vec{x}}_{\text{tu ugrađujemo kernel}} + w_0$$

↳ učenje:

$$\underset{\vec{\lambda}}{\text{argmax}} \left(\frac{1}{2} \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y^i y^j \underbrace{(\vec{x}^i)^T \vec{x}^j}_{\text{tu ugrađujemo kernel}} \right)$$

⇒ u SVM-u se samo na ova dva mjesta pojavljuje $\vec{x}\vec{x}$, pa tu ugrađujemo kernel:

$$K(\vec{x}, \vec{x}') = \phi(\vec{x})^T \phi(\vec{x}')$$

↳ kernel će biti ona kutija:

- unutra ubacimo primjere i on izbaci sličnost

⇒ ovaj jezgreni trik se još naziva i INVERZNO OBLIKOVANJE

⇒ PREDNOSTI INVERZNOG OBLIKOVANJA:

1) smanjenje računalne složenosti jer je lakše ovo nego preslikat u viš prostor

2) jednostavnost oblikovanja

3) veća složenost modela koja daje odvojivost primjera za učenje

⇒ no, ne možemo koristit sve jezgrene funkcije za inverno oblikovanje:

↳ one koje možemo zovu se **Mercerove jezgre**

↳ ideno ove jezgre pmkarati kao matrice:

$$K_{N\times N} = \begin{pmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix} \Rightarrow \text{„kernel matrix"}$$

↳ ako je K pozitivno - semidefinitna, znam da ona mora odgovarat nekom skalarnom produktu primjera:

$$x^T A x \geq 0$$

jezgrena matrica ⇒ Grammova matrica

↳ sažimamo ovo u **MERCEROV TEOREM:**

• ako je K pozitivno semidefinitna za svaki D, onda K mogu rastavit na skalarni produkt dvaju vektora:

$$K(\vec{x}, \vec{x}') = \Phi(\vec{x})^T \Phi(\vec{x}')$$

↑ Mercerova jezgra

⇒ u općenitosti Mercerove jezgre dolaze iz Hilbertovog prostora koj su poopćenje $\mathbb{R}^n$ jer mogu biti $\mathbb{R}^\infty$ samo trebaju imati definiran skalarni produkt

⇒ kako ćemo mi sada graditi jezgre?

↳ pa, uzmat ćemo neke pravila:

- linearne: $K(\vec{x}, \vec{x}') = \vec{x}^T \vec{x}'$ (nema preslikavanja, ista kao bez jezgre)
- Polinomijalne: $K(\vec{x}, \vec{x}') = (\vec{x}^T \vec{x}' + 1)^p$
- „String" jezgra
- ⋮

PR: Pokažimo da je polinomijalna jezgra Mercerova!

$$K(\vec{x},\vec{z}) = (\vec{x}^T\vec{z})^2 = \left((x_1,x_2)^T(z_1,z_2)\right)^2 =$$

$$= (x_1z_1 + x_2z_2)^2 =$$

$$= (x_1z_1)^2 + 2(x_1z_1)(x_2z_2) + (x_2z_2)^2 =$$

$$= x_1^2z_1^2 + \sqrt{2}(x_1z_1)\sqrt{2}(x_2z_2) + x_2^2z_2^2 =$$

$$= x_1^2z_1^2 + \sqrt{2}\,x_1x_2\sqrt{2}\,z_1z_2 + x_2^2z_2^2 =$$

$$= \left(x_1^2, \sqrt{2}\,x_1x_2, x_2^2\right)^T \left(z_1^2, \sqrt{2}\,z_1z_2, z_2^2\right)$$

$$= \phi(\vec{x})^T\,\phi(\vec{z})$$

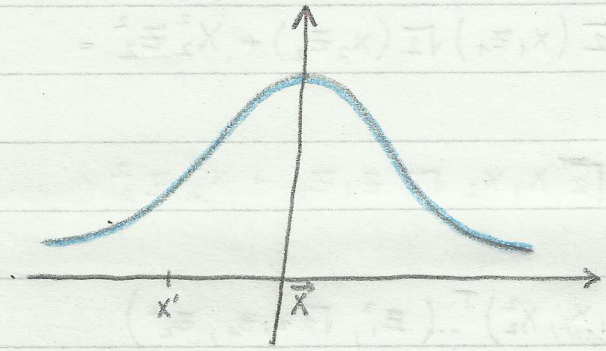
⇓

vidimo da ovaj kernel možemo napisati kao skalarni umnožak nekih preslikavanja u viši prostor, pa je ova jezgra Mercerova

↳ također, vidimo da je za preslikavanje potrebno izračunati $x_1^2, \sqrt{2}\,x_1x_2, x_2, z_1^2, \sqrt{2}\,z_1z_2, z_2^2$, pa je to puno teže nego računanje s jezgrenom funkcijom

⇒ O RBF JEZGRI:

↳ kako γ raste, sličnost parova primjera ide prema nuli, pa su vektori u novom prostoru ortogonalni jer je skalarni umnožak jednak nuli - svaki primjer je poslan u svoju dimenziju, pa sigurno ovo možemo odvojiti hiperravninom (model postaje prenaučen jer je pre-složen)



$$K(\vec{x}, \vec{x}') = \exp\left(-\gamma \|\vec{x} - \vec{x}'\|^2\right)$$

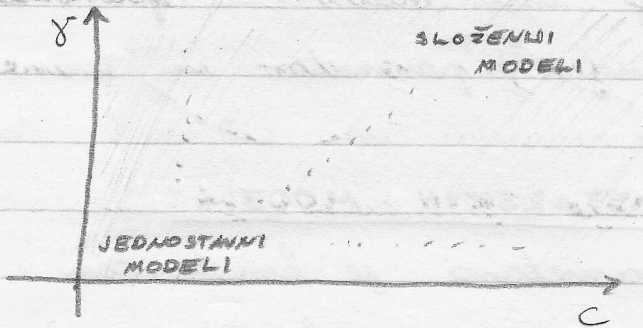
⇒ ako trebamo složenije jezgre, možemo ih dobit kombiniranjem Mercerovih jezgri

(„Multiple Kernel Learning")

⇒ NAPOMENA:

- sada optimiramo dva hiperparametra SVM-a

C i γ



složenost modela u ovisnosti o dva parametra SVM-a:

„C" i „γ"

NEPARAMETARSKI MODELI

⇒ većina modela koje smo radili su **parametarski**:
↳ naprimjer, parametar su težine $\vec{w}$

⇒ **ZNAČAJKE PARAMETARSKIH MODELA:**

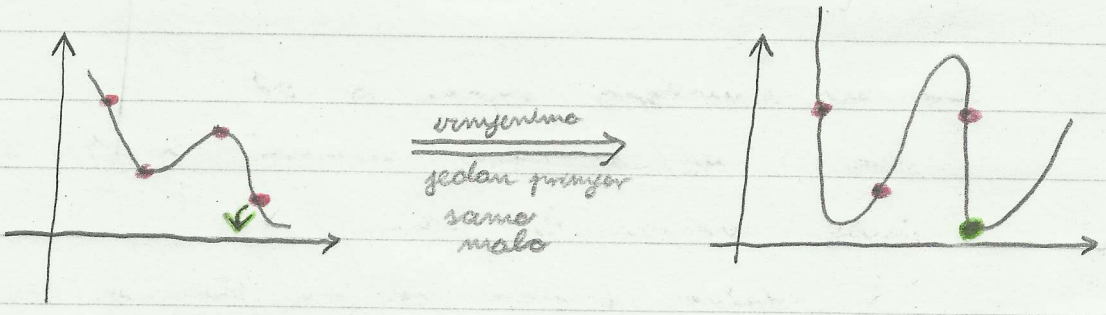
1. Broj parametara je konstantan.
   ↳ ne ovisi o broju primjera

P_R: $\mathcal{X} = \mathbb{R}^R$

$$h(\vec{x}; \vec{w}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5$$

$$m = 6$$

2. Globalnost
   ↳ primjer ima utjecaj na cijelo područje definicije modela



3. Pretpostavljaju da se podac pokorava nekom idealnom modelu / raspodjeli
   ↳ regresija: $P(y \mid \vec{x}) = \mathcal{N}(\vec{w}^T \vec{x}, \sigma^2)$ } normalna raspodjela
   ↳ logistička regresija: $P(y \mid \vec{x}) = \sigma(\vec{w}^T \vec{x})^2 \left(1 - \sigma(\vec{w}^T \vec{x})\right)^{1-y}$
   Bernoullijeva varijabla

⇒ no, jesu li ova svojstva uvijek dobra?

↳ pa, nekad nisu! Osobito druga karakteristika!

⇒ **ZNAČAJKE NEPARAMETARSKIH MODELA:**

(3) Ne pretpostavljaju distribuciju podataka

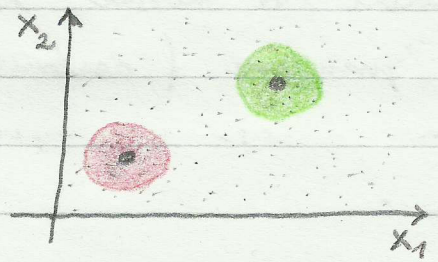
(2) Rade lokalne aproksimacije hipotere

(gledaju sličnost primjera s postojećim primjerima i pretpostavljaju da su slični primjeri slično označeni:

↳ gledaju LOKALNO SUSJEDSTVO)

(ovo lokalno susjedstvo je intuitivna ideja jer tako i ljudi razmišljaju:

„Pogledaj okolnu!")



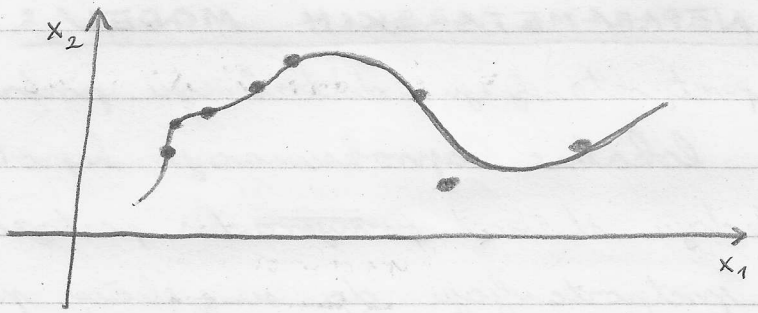
(1) Broj parametara ovisi o N

(primjet da se zove neparametarski modeli, a imaju parametre (PAZI!))

↳ dodatno možemo reći da broj parametara raste kako broj primjera (N) raste:

VEĆI „N" ⇒ VEĆI BROJ PARAMETARA

⇒ kao posljedica navedenih značajki, neparametarski modeli daju veću preciznost (bolje prijanjanje) tamo gdje imamo više primjera:



⇒ KADA KORISTITI KOJI MODEL?

↳ pa, parametarski model ima manje parametara, pa kada znamo (otprilike) distribuciju primjera onda njih koristimo

↳ neparametarski model se koristi kada nismo sigurni u distribuciju primjera (često u „big-data" analizi)

⇒ promatramo jedan neparametarski model SVM

↳ pa, s jednim smo se već upoznali, a to je SVM:

$$h(\vec{x}) = \vec{w}^T \vec{x} + w_0 = \sum_{i=1}^{N} L_i Y^i \vec{x}^T \vec{x}^i + w_0$$

PRIMARNO
parametarski
$O(n^3)$

DUALNO
neparametarski
↳ prednosti:
- SMO rješava u $O(N^2)$
- LAZY model (izračunavamo granicu tek kad dobijemo konkretni $\vec{x}$)
- ovisnost o broju primjera

⇓

neparametarski model koristimo kada imamo malo primjera:

$N << n$

⇒ NAPOMENA: stabla odluke koje smo radili na predmetu Umjetna inteligencija su također neparametarski modeli

↳ broj parametara (čvorova stabla) ovisi o broju primjera

↳ no, nećemo ih baš raditi jer su sklona prenaučenosti i nestabilnosti

ALGORITAM k-NEAREST NEIGHBOUR

⇒ IDEJA: „ s kim s, takav si ! "

⇒ ovo je neparametarski model

$$NN_k(\vec{x}) : X \longrightarrow 2^{1 \times 1}$$

} funkcija za dohvaćanje susjeda

⇒ <u>napišimo model:</u>

$$h(\vec{x}) = \underset{j \in \{0,\dots,K-1\}}{\arg\max} \sum_{(\vec{x}^i, y^i) \in NN_k(\vec{x})} 1\{y^i = j\}$$

K ⇒ broj klasa

k ⇒ broj susjeda koj gledamo

⇒ ovaj model ima sve ugrađeno u model, pa ne trebamo funkciju pogreške i optimizacijski postupak

⇒ <u>„k" je hiperparametar:</u>

- odredtuje koliko susjeda gledamo
- za binarnu klasifikaciju je dobro uzeti neparan „k" kako bi izbjegli izjednačenje
- ako je „k" mali, model je složeniji jer gledamo samo uski područje (finije odlučuje)

  ↳ povećanjem „k", model postaje jednostavniji

WEIGHTED k-NN

⇒ želimo da nam bliži susjedi imaju veći utjecaj na klasifikaciju od onih dalnjih

⇒ stoga, u model treba ugraditi udaljenost

↳ udaljenost računamo JEZGRENIM FUNKCIJAMA

⇒ <u>napišimo model:</u>

$$h(\vec{x}) = \underset{j \in \{0,\dots,K-1\}}{\arg\max} \sum_{(\vec{x}^i, y^i) \in NN_k(\vec{x})} K(\vec{x}^i, \vec{x}) \, \mathbb{1}\{j = y^i\}$$

PROBLEM NAJBLIŽEG SUSJEDA

⇒ k-NN je super, no pretpostavimo da imamo dosta zahtjevan problem:

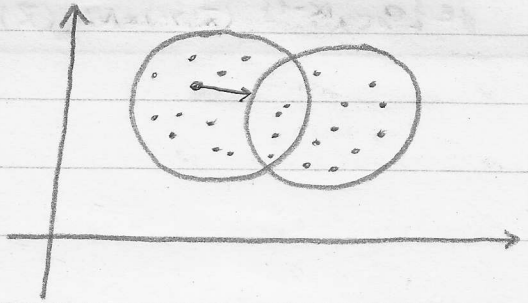
- **kako pronaći najbližeg susjeda** (il „k" susjeda)

⇒ ideja je sagraditi neku stabilnu strukturu

↳ neki pristupi:

a) BALL-TREE:

- grad stablo lopti



- udaljenost do svih u lopti je sigurno veća od udaljenosti do same lopte, pa u loptu nemoj ići tražiti
- smanjuj lopte po koracima

b) LHS:

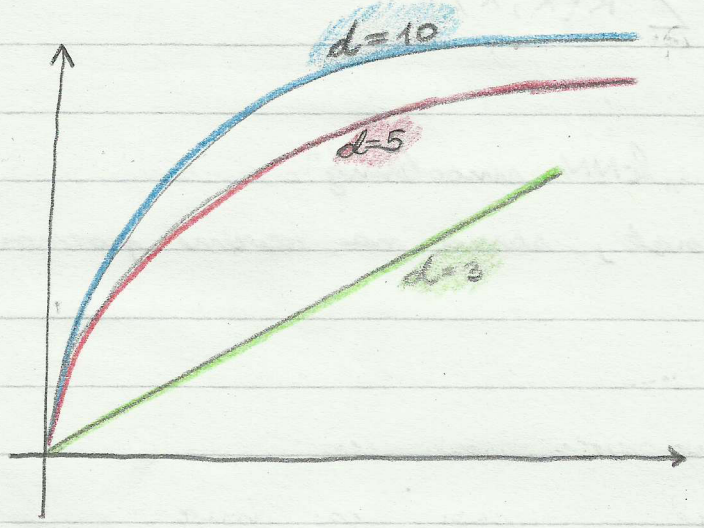
- samo informativno

PROKLETSTVO DIMENZIONALNOSTI

⇒ ako želimo uzeti recimo 50% primjera, u 3-dimenzionalnoj kocki trebamo linearno povećati stranice da l uzel tih 50%

↳ no u sve višim dimenzijama trebamo sve više povećati stranice kocke jer su primjer sve više udaljeni

(nije skroz intuitivno)



REGRESIJA POMOĆU NEPARAMETARSKIH MODELA

⇒ koristimo k-NN algoritam:

$$h(\vec{x}) = \frac{1}{k} \cdot \sum_{(\vec{x}^i, y^i) \in NN_k(\vec{x})} y^i$$

$$h(\vec{x}) = \frac{\sum_{i=1}^{N} K(\vec{x}^i, \vec{x}) \cdot y^i}{\sum_{i=1}^{N} K(\vec{x}^i, \vec{x})}$$

⇒ ovo se još zove „k-NN smoothing"

⇒ ne treba detaljno znati, samo na razini poznavanja

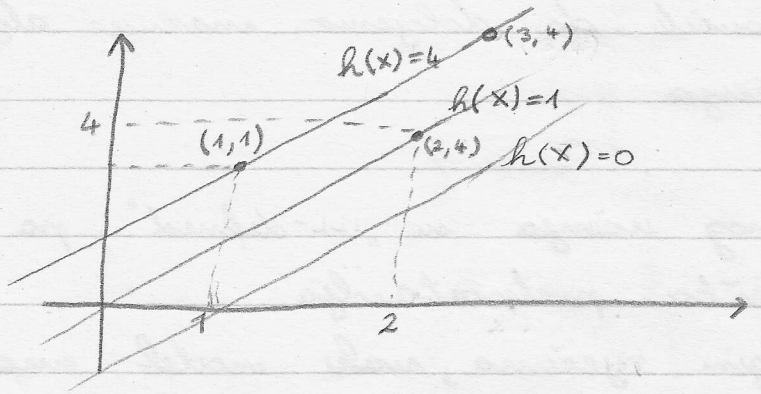
DOPUNA 2. DOMAĆE ZADAĆE

④ a) $y = \frac{3x^2}{w_2} - \frac{12x}{w_1} + \frac{13}{w_0} \Rightarrow$ $w_2 = 3$

$w_1 = -12$

$w_0 = 13$

b)



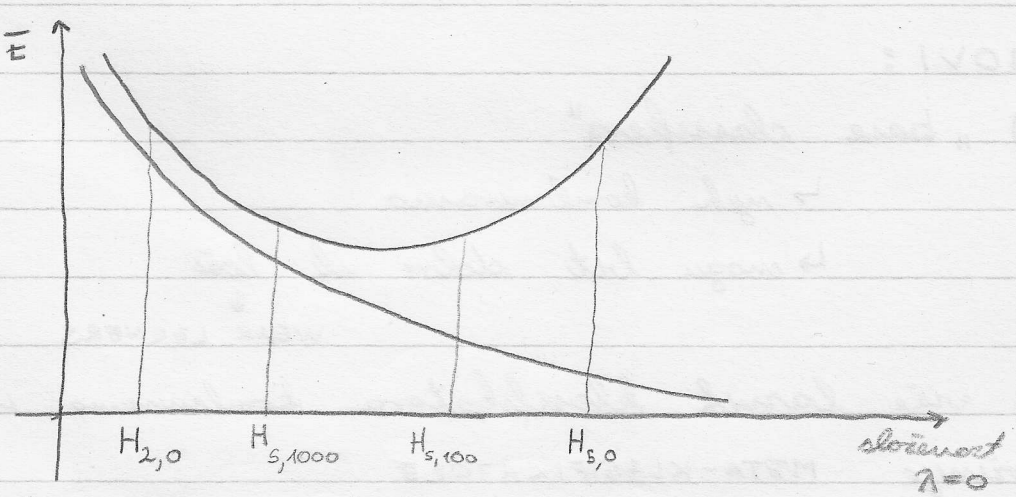
odnos na grafu nisu baš dobri

⑤ d) pa, očekujemo do 3 stupnja jer imamo 3 multiplikatora

↳ pa, 4 stupnja je dobar jer ga malo regulariziramo, no nije ni nužno jer u ulaznim podacima nema šuma

⑥ MEĐUISPIT

⑦ b)



samo kvalitativna skica je dovoljna

ANSAMBLI

⇒ „bogatstvo zvuka orkestra dolazi od pojedinačnih instrumenata"

↳ želimo iskombinirat neke algoritme koje smo do sada naučil da dobijemo snažnije algoritme strojnog učenja

⇒ problemi iz strojnog učenja su „ILL-defined", pa svaki model u sebi nešto pretpostavlja

↳ drugim riječima, svaki model ima svoju INDUKTIVNU PRISTRANOST

↳ zbog razlike u tim pristranostima, neki model je bolji za neki problem, al nije najbolji za sve probleme („no-free-lunch")

⇒ želimo imat komplementarne algoritme:

↳ tamo gdje je jedan loš, drugi je dobar

⇒ POJMOVI:

a) „base classifiers"

↳ njih kombiniramo

↳ mogu biti dobri il loši

↓

WEAK LEARNERS

b) više različnih klasifikatora kombiniramo u složenije META-KLASIFIKATORE

⇒ no, bazne klasifikatore treba DIVERZIFICIRATI:

- malo ih treba izmjenit i to se obično radi na više načina:

1) manipulacija skupa za učenje
2) manipulacija značajkama
3) manipulacija algoritmima (npr. hiperparametrima)

⇒ da bi mogli kombinirati klasifikatore, svaki mora bit malo bolji od slučajnog

↳ barem više od pola mora dobro klasificirati!

KOMBINIRANJE ALGORITAMA

⇒ radimo na više različitih načina

① GLASANJE:

- imamo L osnovnih algoritama:

↳ klasifikacija:

$$h(\vec{x}; \vec{w}) = \arg\max_y \sum_{j=1}^{L} w_j \, 1\{h_j(\vec{x}) = y\}$$

$$w_j > 0$$

$$\sum w_j = 1$$

↑ težine za svaki glas, a ne težine modela

↳ regresija:

$$h(\vec{x}; \vec{w}) = \frac{1}{L} \sum_{j=1}^{L} w_j \, h_j(\vec{x})$$

↳ za uniformno glasovanje:

$$w_j = \frac{1}{L}$$

PR: Imamo 25 binarnih osnovnih klasifikatora koj su međusobno neovisni. Svaki ima vjerojatnost da pogriješi jednaku $E = 0.35$. Kolika je vjerojatnost da ansambl pogriješi?

↳ ansambl griješi ako pogriješi 13 il više osnovnih klasifikatora (više od pola):

$$\sum_{i=13}^{25} \binom{25}{i} E^i (1-E)^{25-i} = 0.06$$

⬇

ovo je puno manje od pogreške jednog klasifikatora, pa intuicija da glasovanje ima smisla (daje bolje rezultate) je točna

⇒ pretpostavka o neovisnost je malo rastegnuta, al nije previše netočna

⇒ ansambl <u>smanjuje varijancu</u> rješenja, tj. povećava stabilnost rješenja:

↳ pokažimo to:

- uprosječimo izlaze ansambla:

$$h = \frac{1}{L} \sum_{j=1}^{L} h_j(\vec{x})$$

- trebamo varijancu iz statistike:

$$Var(h) = Var\left(\frac{1}{L} \sum_{j=1}^{L} h_j(\vec{x}) \right) = (*)$$

$$Var(aX) = a^2 Var(X)$$
$$Var(X+Y) = Var(X) + Var(Y)$$

$$(*) = \frac{1}{L^2} Var\left(\sum_{j=1}^{L} h_j(\vec{x}) \right) = (*)$$

↳ pretpostavi da su modeli isti:

$$\underbrace{h_j(\vec{x}) + h_j(\vec{x}) + \dots + h_j(\vec{x})}_{L} =$$

$$(*) = \frac{1}{L^2} \cdot L \cdot Var(h_j) = \frac{1}{L} Var(h_j)$$

⇓

varijanca ansambla je manja L puta od varijance jednog slabog klasifikatora

(2) STACKING:

- koristi se u big-data analizi ali nije popularan u teorijskim razmatranjima
- vidi informativno na Internetu

(3) BAGGING: (manipulacija skupa za učenje)

- radi se uzorkovanje s ponavljanjem (SAMPLING WITH REPLACEMENT):
  - ↳ uzmi element iz skupa primjera, zabilježi ga i vrati ga natrag
  - ↳ tako iz malog skupa primjera možemo dobiti veći (početni skup primjera mora biti dobar!)
- generiramo „L" skupova podataka, svaki veličine „N" primjera:
  - ↳ ovo se naziva „0.632 bootstrap" jer će u prosjeku biti izabrano 63.2% primjera za uzorkovanje u svakom skupu:
    - ↳ pogledajmo izračun:

$\frac{1}{N}$ ⇒ vjerojatnost odabira primjera

$1-\frac{1}{N}$ ⇒ vjerojatnost da nekuda izabran

$\left(1-\frac{1}{N}\right)^N$ ⇒ vjerojatnost da ne bude izabran u N ponavljanja

$$\lim_{N\to\infty}\left(1-\frac{1}{N}\right)^N = \frac{1}{e} \approx 0.368$$

↳ stoga, vjerojatnost uključivanja je:

$1 - 0.368 = 63.2\%$

PR: Generiranje 3 skupa podataka od 4 elemenata iz početnog skupa elemenata:

1 2 5 8

↳ znamo: $L = 3$, $N = 4$

↳ sada nasumično biramo:

(1) 1 2 2 8

(2) 2 5 8 8

(3) 1 2 5 8

⇒ RANDOM FORESTS:

↳ bagging + stabla odluke

↳ radi vrlo dobro

(4) **BOOSTING:**

- fokusiraj se na one primjere koje smo teško klasificirali u prethodnoj iteraciji
- **ideja:**
  - ↳ početna, vjerojatnost odabira primjera je $\frac{1}{N}$
  - ↳ radimo L iteracija, svaki primjer dobiva svoju težinu u iteracijama
  - ↳ one koje teže klasificiramo, damo im veću težinu, tako da se ovi klasifikatori pri kraju iteriranja specijaliziraju na klasifikaciju onih najtežih (najproblematičnijih) primjera
  - ↳ ako su ti radniji klasifikatori uvrok prenaučenosti jer prijanjaju na šum, onda smanji L (broj iteracija)
  - ↳ dodatno, ako je model dobar, i „fulao" je na dva primjera, dodaj puno težinu ta dva primjera, a ako je model loš, nemoj puno dodati jer je model loš preko svih primjera
  - ↳ to je drugo (intuitivno) objašnjenje
- ovo je algoritam **ADABOOST:**

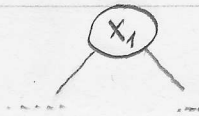
$$h(\vec{x};\vec{z}) = \arg\max_y \sum_{j=1}^{L} z_j \mathbb{1}\{h_j(\vec{x}) = y\}$$

↓

za krajnju odluku se opet radi glasovanje

- kod glasovanja se koristi stabla odluke dubine 1 što se naziva PANJ („stump")

(vidi online!)

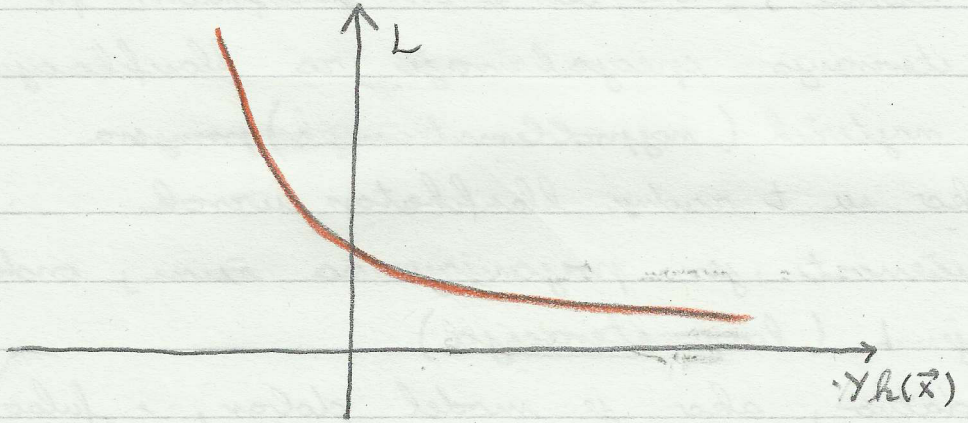


- AdaBoost je dosta jeftin i dobro radi
- AdaBoost raspravo minimizira eksponencijalni gubitak:

$$L(y, h(\vec{x})) = \exp(-y h(\vec{x}))$$

↓

mogli smo i ovako izvesti AdaBoost (minimizirano pogrešku gradijentnim spustom)



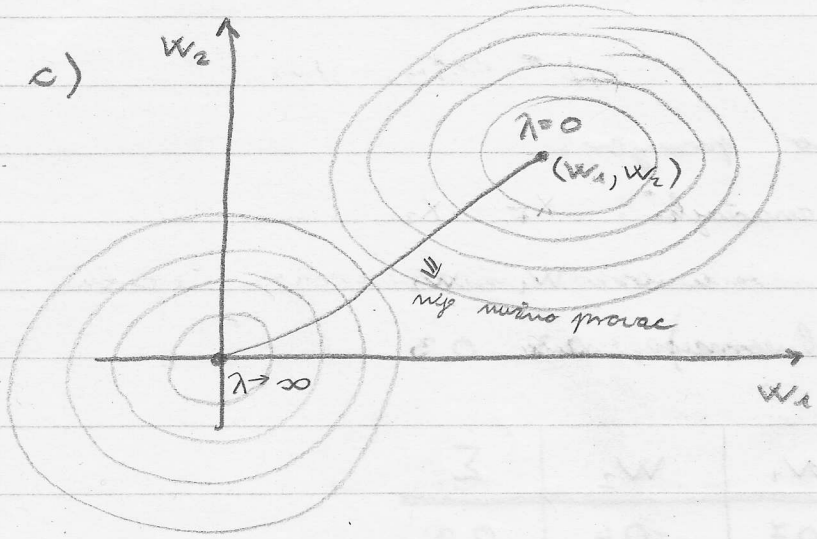
↓

jako kažnjava loše klasifikacije, pa je sklon prenaučenosti (može se spriječit)

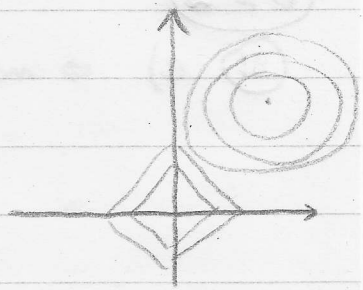
↳ dodatno, ne daje mjerljive modele jer se malo kažnjava rub i točno klasificirani primjeri (za razliku od SVM-a)

DOPUNA 2. DOMAĆE ZADAĆE

(8) c)



d)



(9) c) 1) nebitna je ona s malom težinom

2) model ima velike težine na velikim stupnjevima

e) redundantne ⇒ kolinearne, suvišne

relevantne ⇒ nešto nebitno (npr. temperatura u Japanu)

↳ očekivane težine koje su usporedive nad tim značajkama (ako su skalirane na istu veličinu)

↳ regularizacija samo po sebi ne rješava problem interpretabilnosti

PONAVLJANJE ZA ISPIT

DZ 2

9. c) ⇒ naprovimo primjer:

↳ 2 značajke: X_1 X_2

W_1 W_2

(kolinearne) 0.3 0.3

	W_1	W_2	Σ
bez reg	-0.7	-0.4	0.3
L2	0.15	0.15	0.3
L1	0.3	0	0.3
	0	ili 0.3	

vidimo da nas regularizacija ne spašava od redundantnih značajki, pa matematičar ne rade regularizaciju nego pažljivo biraju značajke kako bi izbjegli nebitne, kolinearne značajke

DZ 3

(2) a) zato jer želimo istu kaznu za sve pogrešne klasifikacije

↳ stoga, želimo jednaku udaljenost među labelama razreda:

• npr. $\vec{Y} = \{[1,0,0], [0,1,0], [0,0,1]\}$

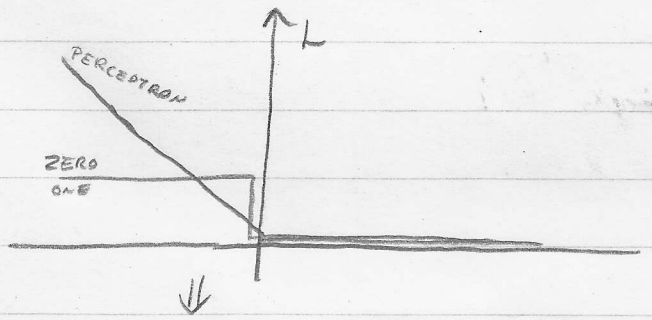
↳ ti brojevi su samo oznake klasa, a ne želimo da imaju utjecaj na kaznu

(npr. ne želimo veću kaznu za pogrešnu klasifikaciju primjera iz 0 u 1 nego za pogrešnu klasifikaciju primjera iz 0 u 2)

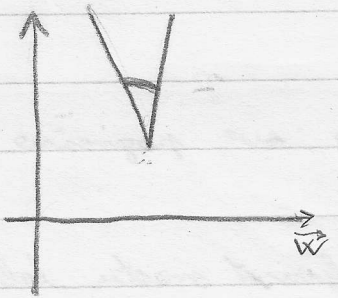
g) zato jer kvadratni gubitak uopće ne aproksimira zero-one, a zero-one loss je ono što želimo kod klasifikacije (on nam je idealan)

(3) $$E(\vec{w}|D) = \frac{1}{N}\sum_{i=1}^{N}(Y - sgn(\vec{w}^T\vec{x}))$$

↳ deriviramo i dobijemo perceptron loss:



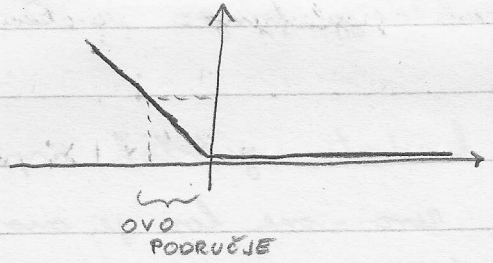
niže gornja granica, pa niže bolja aproksimacija zero-one loss-a



nikad ne može doseći minimum i taj minimum nije na nuli (pa, i može doseć minimum, ali on nije na nuli, pa algoritam ne staje)

) sve osim perceptrona imaju prosjek barem jednak broju pogrešnih klasifikacija

↳ perceptron ne mora imati veću grešku jer ima dio gdje je L manji od 1



DZ.4

BITNO!

① c) $P(D \mid h(\vec{x}, \vec{w}))$ ⇒ *izglednost hipoteze*

↳ možemo pisati $P(D \mid \vec{w})$

→ *izglednost hipoteze ovisna i vjerojatnost podataka uz dane primjere*

$$P(D \mid \vec{w}) = \prod_{i=1}^{N} P(y^i \mid \vec{x}^i)$$

↓

radi toga smijemo prijeći sa D na Y:

↳ *pa pogledajmo:* (*matrica primjera*)

$$P(D \mid \vec{w}) = P(\mathbf{X}, \vec{y} \mid \vec{w}) =$$

$$= P(\vec{y} \mid \mathbf{X}, \vec{w}) \cdot P(\mathbf{X} \mid \vec{w}) =$$

$$\ln P(D \mid \vec{w}) = \ln P(\vec{y} \mid \mathbf{X}, \vec{w}) + \ln P(\mathbf{X} \mid \vec{w})$$

deriviramo po $\vec{w}$, *a ovo ne ovisi o* $\vec{w}$ *pa on ispada* (*samo izbor modela* $\vec{y}$ *ovisi o* $\vec{w}$ *pa ga to opravdanje za prijelaz s D na Y*)

e) $h(\vec{x}) = \frac{1}{1+e^{-\vec{w}^T\vec{x}}}$

$$1 - h(\vec{x}) = \frac{1+e^{-\vec{w}^T\vec{x}} - 1}{1+e^{-\vec{w}^T\vec{x}}} = \frac{1}{1+e^{\vec{w}^T\vec{x}}}$$

razlika je samo u predznaku što ima smisla jer smo uzel -1, 1 za Y'

↳ uvrstimo u funkciju gubitka:

$$L(Y', h(\vec{x}')) = -Y \ln h(\vec{x}) - (1-Y)\ln(1-h(\vec{x}))$$

$$= -Y \ln\left(\frac{1}{1+e^{-\vec{w}^T\vec{x}}}\right) - (1-Y)\ln\left(\frac{1}{1+e^{\vec{w}^T\vec{x}}}\right)$$

$$= \underbrace{Y \ln\left(1+e^{-\vec{w}^T\vec{x}}\right)}_{Y=1} + \underbrace{(1-Y)\ln\left(1+e^{\vec{w}^T\vec{x}}\right)}_{Y=0} = (*)$$

↳ sada trebamo pametno šaltanje između grana:

- izbacimo Y pa nešto drugo smislimo za naše oznake
- „ako je $Y=0$, aktiviran desni granu, a za $Y=1$ aktiviran lijevu granu"

$$(*) = \ln\left(1 + e^{-Y\vec{w}^T\vec{x}}\right)$$

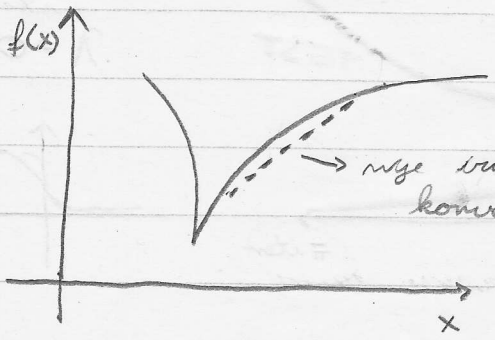
↓

podjelimo sa $\ln 2$ kako li zadovoljil uvjet:

$$L(0, Y) = 1$$

(2) a) funkcija mora biti ispod spojnice i domena je konveksan skup ▽○

b)



→ nije iznad funkcije pa nije konveksna iako je unimodalna ▽○

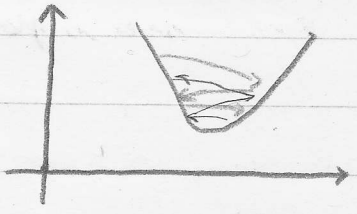
↳ NIJE ISTO?

c) mata što ako je gubitak konveksan, onda je i pogreška konveksna

↳ a konveksne gubitke znamo optimirati postupcima konveksne optimizacije

(3) a) mata da možemo garantirati globalnu konvergenciju

↳ neću napraviti prevelik korak u nekom smjeru, pa neću skakati oko optimuma



} ovo s nečim desiti ▽!

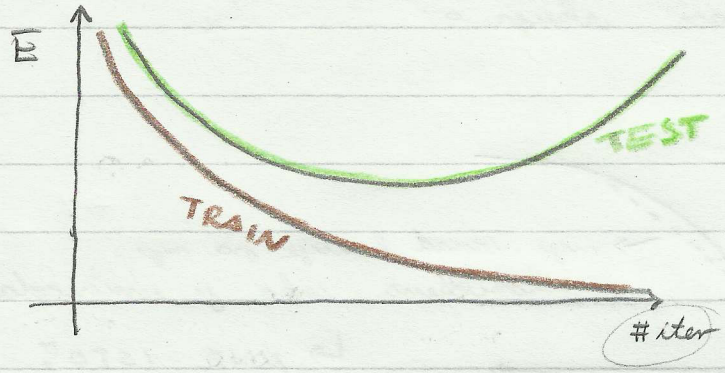
c) BITNO! Pogledaj!

d) Brže konvergirat, al računalno zahtjevnij, al nekad uvijek imat rješenje (ne mora postojati inverz Hessijanove matrice?)

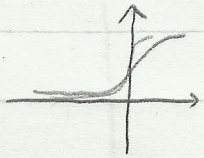
BITNO!

(4) Aka imamo linearne odvojive primjere!

a)



$\lambda = 0$



nepravilne
sigmoide



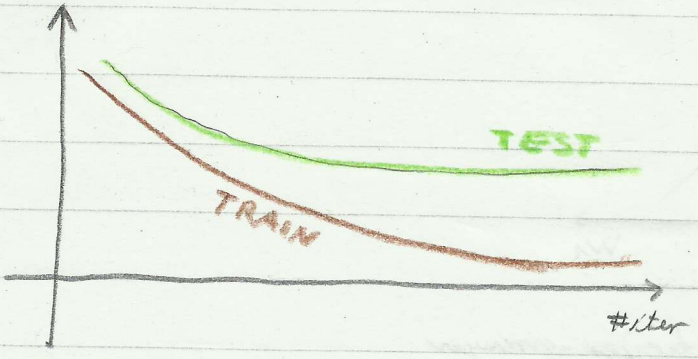
$\lambda = 100$

↘ blago raste

⇓

čim imamo $\lambda > 0$, train više ne može doći do nule te postoji neki minimum greške na train setu, što nije istina za $\lambda = 0$ jer tada $\vec{w}$ možemo povećavati u beskonačnost

Ako imamo linearno NE-odvojive primjere!



$\lambda = 0$

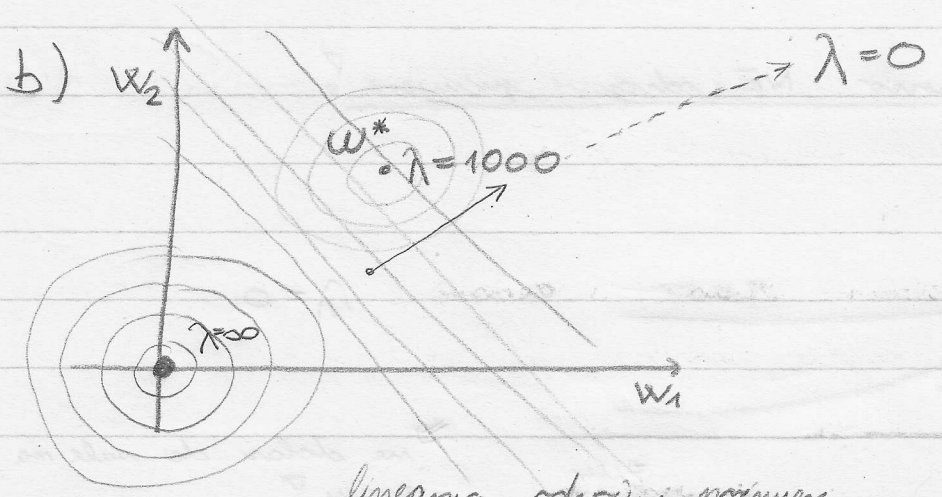
⇒ ne dolazi do nule za train!



$\lambda = 100$

⇒ ne raste!

b)



Linearna odhojna primjer

⑥ a) Linearna regresija:

$$h(\vec{x}; \vec{w}) = \vec{w}^T \vec{x}$$

$$P(y / \vec{x}, \vec{w}) = \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(h(\vec{x}), \sigma^2)$$

$$L(y, h(\vec{x})) = (h(\vec{x}) - y)^2$$

Normalna razdioba

Logistička regresija:

$$h(\vec{x}; \vec{w}) =$$

⋮ bernoullijeva razdioba

⋮

Multinomijalna regresija:

- samo je prilagođena za više klasa

" MULTI - NULIJEVA RAZDIOBA" (softmax)

⇓

VEZA: gubitak smo izveli na iste načine za sve regresije, a razlika je samo u razdiobama

b) probabilistički pogled : $P(D|\vec{w})$

diskriminativni pogled : $E(\vec{w}|D)$

↓

tipično ovako možemo

uvodit strani u strojnom

učenju

(dva načina ?)

DZ 5.

(1) b) bitno je znati za KKT uvjete od kuda su oni došli (primarni il' dualni problem)

$$y^{(i)}(\vec{w}^T\vec{x}^i + w_0) \geq 1$$

$$\alpha_i \geq 0$$

$$\alpha_i(y^i f(\vec{x}) - 1)$$

↳ konačni KKT:

$y^i(\vec{w}^T\vec{x} + w_0) \geq 1 - \xi$ iz orginalnog problema

dualna ograničenja

komplementarna labavost

OGRANIČENJA:

- ono što treba zadovoljavat neki minimum

KKT:

- uvjeti što vrijede vrijed u točki minimuma

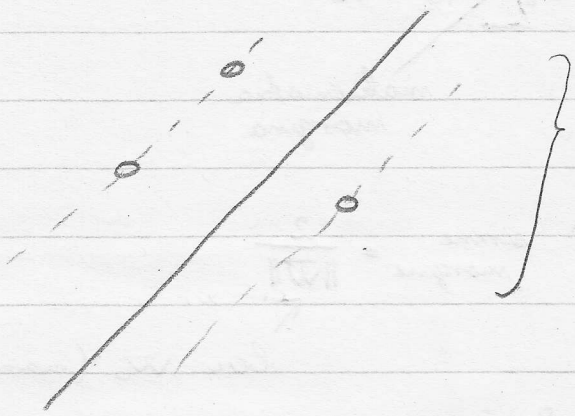
d) Jer možemo koristit jezgreni trik

↳ koristimo jezgru direktno u optimizacijskom problemu

↳ želimo imat umnožak vrlara i potpornih vektora

d) $\Rightarrow$ pravac se w_0 može napisati kao zbroj dualnih parametara (al nije nužno) te se tako možemo

e) potporni vektori su oni za koje je $\alpha^{(i)} > 0$ tj. to su oni primjeri koji utječu na predikciju



oni definiraju hiperravninu koja razdvaja primjere

f) pogledamo radnj KKT:

$$\alpha^i (y^i h(\vec{x}^{(i)}) - 1) = 0$$

$\downarrow$

ako je $\alpha^i > 0$, ovo u zagradi mora biti nula, pa ako je točnost klasifikacije 1 ili -1, on je na margini:

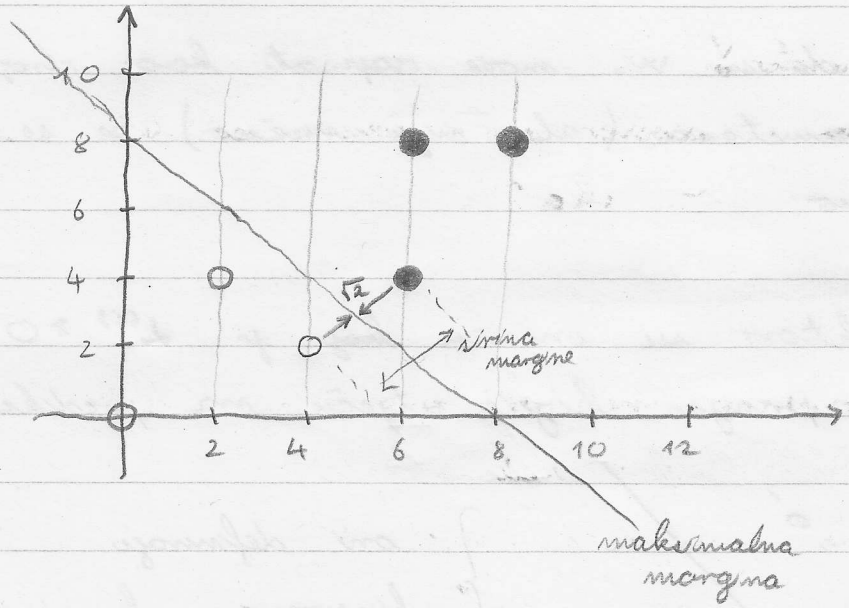
$$y^i h(\vec{x}^{(i)}) - 1 = 0$$

$$y^i h(\vec{x}^{(i)}) = 1$$

u dualnom problemu

g) množimo $\vec{x}^T \vec{x}$, pa već značajke imaju puno veći utjecaj jer se vrijednosti kvadriraju

(2)



$$d = \frac{h(\vec{x})}{\|\vec{w}\|}$$

$$\text{širina margine} = \frac{2}{\|\vec{w}\|}$$

↖ bez w_0 (naravno)

$$h(\vec{x}) = x_1 + x_2 - 8$$

$$w_1 = 1 \qquad w_2 = 1 \qquad w_0 = -8$$

$$d = \frac{h(\vec{x})}{\|\vec{w}\|} = \frac{h((4,2))}{\|\vec{w}\|} = \frac{4+2-8}{\sqrt{1+1}} = \frac{-2}{\sqrt{2}} \cdot \frac{\sqrt{2}}{\sqrt{2}} = -\sqrt{2}$$

⇓

nije dobro jer nam je selar modela 2, a u SVM-u želimo jedan!

↳ ako tražimo podjelbu s 2, dobivamo:

$$h(\vec{x}) = 0.5x_1 + 0.5x_2 - 4$$

$$\|\vec{w}\| = \frac{1}{\sqrt{2}}$$

$$d' = \frac{h((4,2))}{\|\vec{w}\|} = \frac{\frac{1}{2}\cdot 4 + \frac{1}{2}\cdot 2 - 4}{\frac{1}{\sqrt{2}}} = -\sqrt{2}$$

⇒ dobili smo isto!

↳ to je dobro jer udaljenost treba biti neovisno o skali!

⇓

ni zapravo ne želimo tu udaljenost jednaku 1 nego želimo da na rubu margine daju imaju izlaz modela 1 (znači, potporni vektori moraju imati izlaz modela jednak 1)

⇒ ovo je demonstracija kako dobiti potporne vektore (kako dobit da najbliži primjer daju izlaze 1 il -1)

⇒ također, demonstrirali smo da se udaljenost od granice ne mijenja ako težine množimo s nekim faktorima

↳ širina margine: $2\sqrt{2}$

c) ovdje je uvrštavljen jedan potporni vektor, ali nema vere (ne mijenja radatak)

↳ ovaj uvrar u radatku omogućava prijelaz iz dualnog u primarni problem:

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y^i \vec{x}^i$$

ovdje nema w_0

↳ sada, radatak je invertan:

- grafički smo dobili $\vec{w}$, a trebamo dobiti α koeficijente

$\vec{w} = (\frac{1}{2}, \frac{1}{2})$

$\vec{x}^3 = (4,2) \quad y^3 = -1$

$\vec{x}^4 = (6,4) \quad y^4 = 1$

$$\vec{w} = \alpha_3 (-1) \begin{bmatrix} 4 \\ 2 \end{bmatrix} + \alpha_4 \cdot (1) \cdot \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

⇓

iz ove dvije jednadžbe dobijemo α_3 i α_4:

$$\alpha_3 = \alpha_4 = 0.25$$

↳ za ostale pomnožene je nula jer nisu potporni:

$$\vec{\alpha} = (\underset{\alpha_1}{0} \; \underset{\alpha_2}{0} \; \underset{\alpha_3}{0.25} \; \underset{\alpha_4}{0.25} \; \underset{\alpha_5}{0} \; \underset{\alpha_6}{0})$$

ovdje nebi trebala biti nula

VIDI SKRIPTU !

d) nije točno $\left(w_0 = -4\right)$

↳ w_0 se može izračunat jer je skalar za potporne vektore jednak 1

e) PRIMARNI: $h(\vec{x}) = \vec{w}^T\vec{x} + w_0$

DUALNI: $h(\vec{x}) = \sum_{i=}^{N} \alpha_i y^i \vec{x}^T\vec{x}^i + w_0$

↳ formula za d podatak:

$$w_0 = y^i - \sum_{j \in S} \alpha_j y^j (\vec{x}^i)^T \vec{x}^j$$

↑ support vector

↑ „i" ⇒ bilo koj potporni vektor

$w_0 = -4$

⇒ umjesto dualni problem:

$$h((5,6)) = 0.25\,(-1)\begin{bmatrix}5 & 6\end{bmatrix}\begin{bmatrix}4\\2\end{bmatrix} + 0.25\,(+1)\begin{bmatrix}5 & 6\end{bmatrix}\begin{bmatrix}6\\4\end{bmatrix} - 4$$

$$h((5,6)) = 1.5$$

↓

pozitivan primjer

trebla pisati $\alpha_i < C$, a ne manje-il-jednako (kao u radatku)

③ c) Potporni vektori zadovoljavaju $\alpha_i > 0$

$$\alpha_i \left(y^i h(\vec{x}) - 1 + \xi_i \right) = 0$$

ako je primjer potporni vektor, vidimo da on u nagradi mora biti nula ▽:

$$y^i h(\vec{x}) - 1 + \xi_i = 0$$

ako $\alpha_i < C$

$$\alpha_i = C - \beta_i$$

$$\beta_i = C - \alpha_i \Rightarrow \beta_i > 0$$

$$\beta_i \xi_i = 0$$

$\Downarrow$

ξ_i mora biti nula

$\xi_i = 0$

(to znači da je primjer na margini)

ako $\alpha_i = C$

$$\alpha_i = C - \beta_i$$

$$\beta_i = C - \alpha_i \Rightarrow \beta_i = 0$$

$$\beta_i \xi_i = 0$$

$\Downarrow$

ξ_i može biti i veći od nula

$\xi_i > 0$

(to znači da je primjer unutar margine)

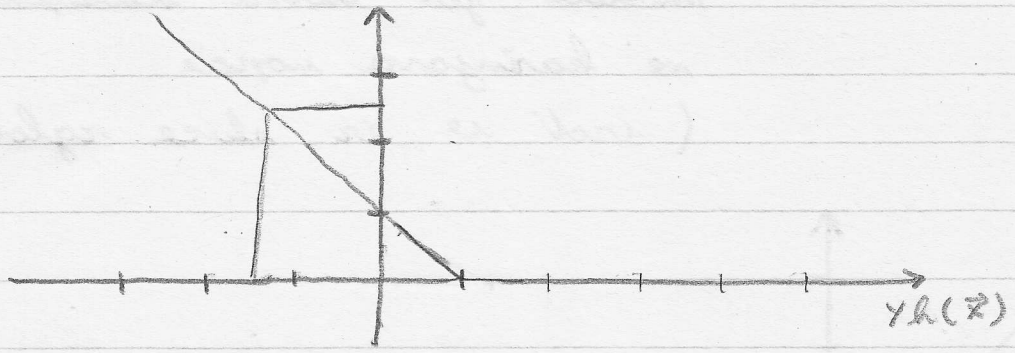
(4) c) $\vec{x}^7 = (5, 6)$ $y^7 = -1$

↳ stvarna klasifikacija

↳ model je dao: $h(\vec{x}^7) = 1.5$

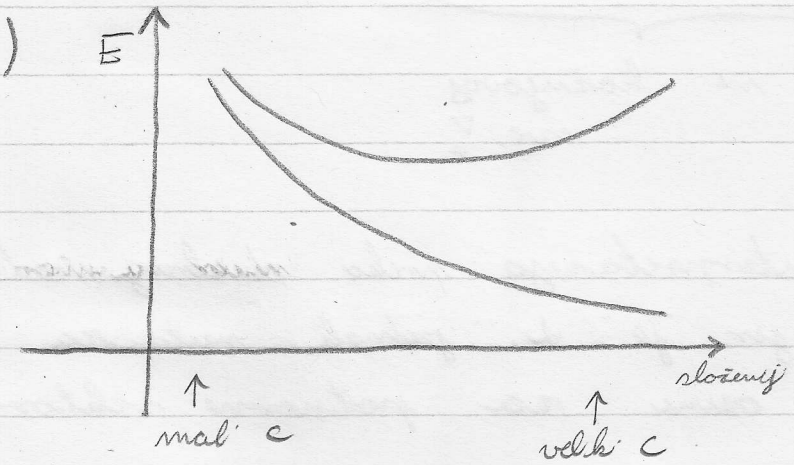
↳ primjetimo sa gubitka zglobnice:

$$L(y, h(\vec{x})) = \max(0, 1 - y\,h(\vec{x}))$$



$$L(y^7, h(\vec{x}^7)) = \max(0, 1 - (-1) \cdot 1.5) = 2.5$$

d)



OBJAŠNJENJE:

↳ velik C znači da puno kažnavamo uloshe u margini, pa je model složenij

• vid se iz formule:

$$E(\vec{w}|D) = \|\vec{w}\|^2 + C \sum_{i=1}^{N} \xi_i$$

e) $E(\vec{w} \mid D) = \|\vec{w}\|^2 + C \sum_{i=1}^{N} \xi_i$

↑

ima L2 - regularizaciju

po defaultu iako nismo

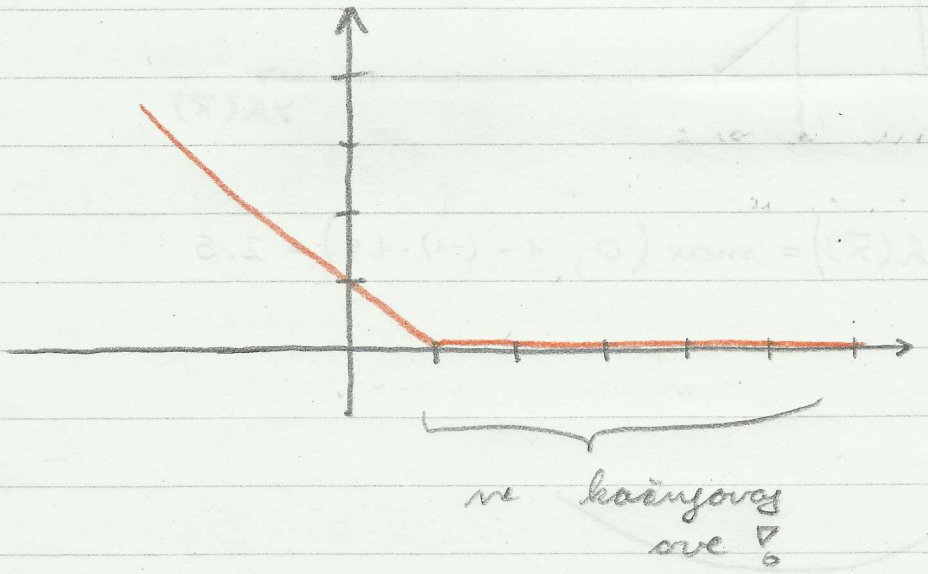
to ugradivali u SVM

↳ ipak, SVM daje rijetke

modele jer dobru klasifikaciju

ne kažnjava nepće

(vidi se iz skice roglobnice)



ne kažnjava ove ▽

↳ postoji i interpretacija preko dualnog modela:

- jer je α_i jednak nuli za

sve osim za potporne vektore

⇒ NAPOMENA:

↳ ako je model rijedak u jednoj

formulaciji, ne mora biti rijedak u drugoj ▽

DZ 6

① c) Opdenit linearan model:

$$h(\vec{x}) = f\left(\vec{w}^T \phi(\vec{x})\right)$$

$$\phi(\vec{x}) = \left(\phi_0(\vec{x}), \dots, \phi_m(\vec{x})\right)$$

Jezgreni stroj:

$$\phi(\vec{x}) = \left(1, K(\vec{x}, \vec{\mu}_1), \dots, K(\vec{x}, \mu_m)\right)$$

↳ isti model kao gore, samo razlika u preslikavanju

PARAMETARSKI?

(ne ovisi o broju primjera)

Rijetki jezgreni stroj:

$$\phi(\vec{x}) = \left(1, K(\vec{x}, \vec{x}^1), \dots, K(\vec{x}, \vec{x}_N)\right)$$

↳ isti model kao gore

NEPARAMETARSKI?

(ovisi o broju parametara)

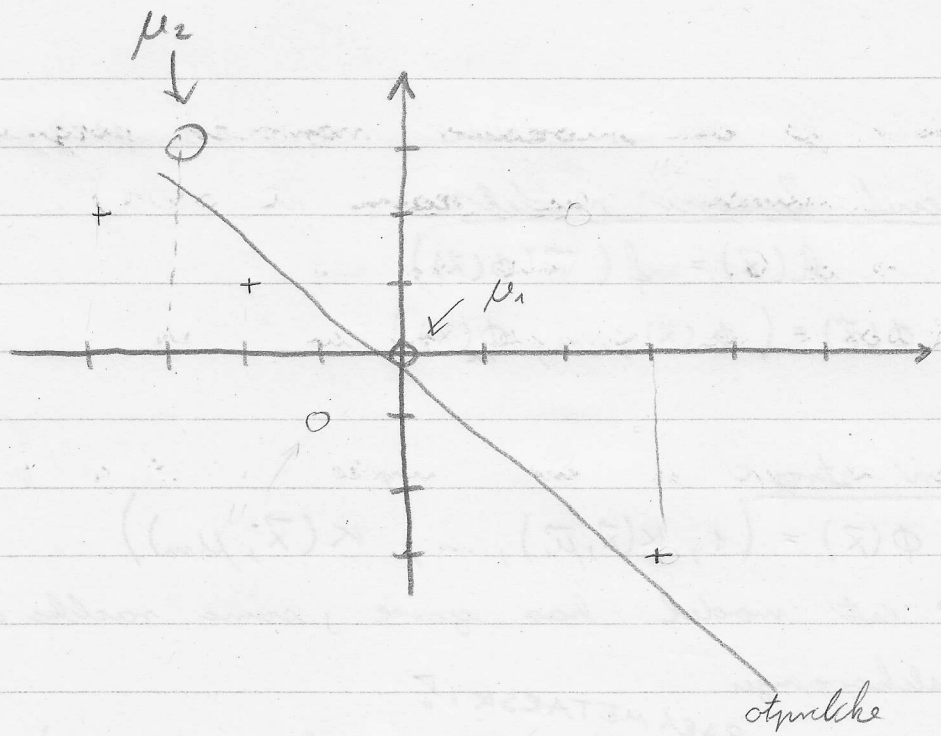
⇒ SVM je zapravo rijetki jezgreni stroj jer dualni model odgovara ovom linearnom modelu uz „rijetka" preslikavanja

$$h(\vec{x}) = \sum_{i=1}^{N} (\lambda_i y^i) \vec{x}^T \vec{x}^i + w_0$$

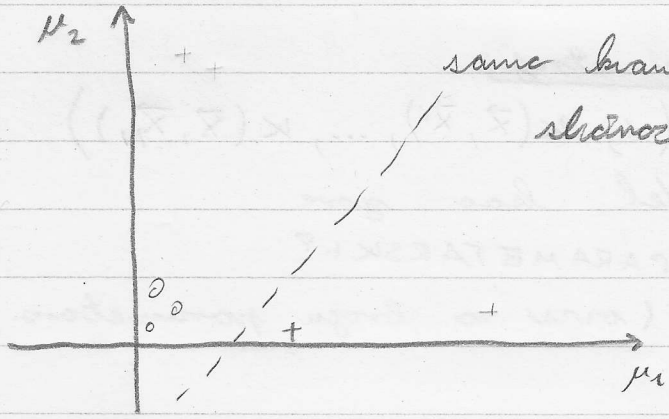
ovo su zapravo težine

(2)

a)



b)



same krivuljasta slika složenosti !?

c) ⇒ prije preslikavanja:

$$h(\vec{w}; \vec{x}) = \sigma(\vec{w}^T \vec{x}) \quad \Big\} \; n = 3 \text{ parametra}$$

⇒ nakon preslikavanja:

$$h(\vec{w}; \vec{x}) = \sigma(\vec{w}^T \phi(\vec{x}))$$

$$\phi(\vec{x}) = (1, \; \kappa(\vec{x}, \vec{\mu_1}), \; \kappa(\vec{x}, \vec{\mu_2})) \quad \Big\} \; n = 3 \text{ parametra}$$

dodatna dva hiperparametra

⇒ hiperparametri ne utječu na složenost (imamo 2)

③ a) bitno je da možemo napravit jezgreni trik (ugradit tu jezgru u SVM)

↳ dokaz je isti kao na predavanjima (samo raspiši skalarni umnožak!)

b) trebamo iz jezgrene funkcije iskopat Φ (preslikavanje)

↳ iz a) podzadatka vidimo:

$$\Phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

↳ dalje je lagano!

c) neda nam se preslikavat, pa koristimo jezgreni trik!

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y^i \Phi(\vec{x}^i) \qquad (X' \text{ je } Z \text{ u zadatku})$$

$$K(\vec{x}, \vec{x}') = (\vec{x}^T\vec{x}' + 1)^3$$

$\vec{x}^1 = (-2, 3, 5)$ $\quad y^1 = -1$ $\quad \alpha_1 = 0.131$

$\vec{x}^2 = (6, 4, 3)$ $\quad y^2 = +1$ $\quad \alpha_2 = 0.048$

$\vec{x}^3 = (8, 8, 2)$ $\quad y^3 = +1$ $\quad \alpha_3 = 0.013$

↳ ranima me oznaka od: $\vec{x}^4 = (1, 2, 3)$

⇒ pa, krenimo:

- $h(\vec{x}) = \vec{w}^T\Phi(\vec{x}) + w_0 = \sum_{i=1}^{N} \alpha_i y^i \underbrace{\Phi(\vec{x})^T\Phi(\vec{x}^i)}_{K(\vec{x}, \vec{x}')} + w_0$

$$h(\vec{x}) = 0.131 \cdot (-1) \cdot \left([1 \;\; 2 \;\; 3] \cdot \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix} + 1 \right) + \ldots \text{ostali potpori vektori}$$

$$-0.51$$

④ e) (1) da, kada god dovedemo nov primjer, on će uvesti novu dimenziju

50 primjera ⇒ 50 dimenzija

(2) ne, jedino ako je gama dovoljno velik

(3) ne, jedino ako je gama dovoljno velik

(4) ne, jedino ako je model dobro odabran!

PROCJENA PARAMETARA (UVOD)

⇒ probabilistički modeli su super jer je teorija vjerojatnosti jako dobro razrađena, pa možemo puno domnat iz rezultata

↳ dodatno, oni dobro rade ako imamo malo podataka

⇒ za motivaciju, prisjetimo se BAYESOVOG KLASIFIKATORA:

$$P(y|\vec{x}) \sim P(\vec{x}|y) \cdot P(y)$$

⇒ kada pričamo o probabilističkim modelima, procjena parametara je u stvari učenje modela



STATISTIČAR:

„Ja procjenjujem parametre."

=



PROGRAMER:

„Ja učim model."

⇒ SLUČAJNE VARIJABLE:

↳ oznaka: X

↳ moguće vrijednosti: $\{x_i\}$

↳ vjerojatnost neke vrijednosti:

$$P(X = x_i) = P(x_i) \geq 0$$

$$P(X) = \sum P(x_i) = 1$$

} diskretne vrijednosti

↳ za kontinuirane vrijednosti koristimo oznaku malo „p"

↳ tu pričamo o gustoći vjerojatnosti:

$$p(x) \geq 0$$

$$\int_{-\infty}^{\infty} p(x)\,dx = 1$$

} kontinuirane vrijednosti

⇒ OČEKIVANJE:

a) <u>diskretno</u>: $E[X] = \sum x P(x)$

b) <u>kontinuirano</u>: $E[X] = \int_{-\infty}^{\infty} x\, p(x)\, dx$

⇒ VARIJANCA:

↳ mjeri koliko neka varijabla pleše oko svoje očekivane vrijednosti:

$$Var(X) = \sigma_x^2 = E\left[(x - E[x])^2\right] = E[x^2] - E[x]^2$$

⇒ KOVARIJANCA: (kakva je vera među varijabla)

$$Cov(x, y) = \sigma_{x,y} = E\left[(x - E[x])(y - E[y])\right]$$

$$Cov(x, y) = Cov(y, x) \quad ; \quad Cov(x, x) = Var(x)$$

Pr: Izračunaj očekivanja, varijance i kovarijance!

$X = \{1, 2\}$ $\quad P(X=1) = 0.6 \quad P(X=2) = 0.4$

$Y = \{1, 2\}$ $\quad P(Y=1) = 0.7 \quad P(Y=2) = 0.3$

X \ Y	1	2	
1	0.5	0.1	0.6
2	0.2	0.2	0.4
	0.7	0.3	

$P(X=1, Y=1) = 0.5 \quad P(X=1, Y=2) = 0.1$

$P(X=2, Y=1) = 0.2 \quad P(X=2, Y=2) = 0.2$

$E[X] = 0.6 \cdot 1 + 0.4 \cdot 2 = 1.4$

$E[Y] = 0.7 \cdot 1 + 0.3 \cdot 2 = 1.3$

$Var(X) = 0.6 \cdot (1 - 1.4)^2 + 0.4(2 - 1.4)^2 = 0.24$

$Var(Y) = \ldots = 0.21$

$$Cov(X, Y) = \sum_{x \in \{1,2\}} \sum_{y \in \{1,2\}} P(x, y)(x - E[x])(y - E[y]) = \ldots = 0.08$$

↓

vel do varijable „kovarijacija", al sama vrijednost nam ne govori je li to puno il malo, pa to treba normirat / standardizirat:

$$\rho_{X,Y} = \frac{Cov(X, Y)}{\sigma_X \cdot \sigma_Y}$$

PEARSONOV KOEFICIJENT KORELACIJE

$\rho_{X,Y} \in [-1, 1]$

$$\rho_{X,Y} = \frac{0.08}{\sqrt{0.24} \cdot \sqrt{0.21}} = 0.35$$

⇒ u strojnom učenju, mi radimo s vektorima, pa njih modeliramo s **vektorima slučajnih varijabli**:

$$\vec{X} = (X_1, X_2, \dots, X_n)$$

⇒ **KOVARIJACIJSKA MATRICA**:

↳ matrica kovarijanci svih parova varijabli unutar slučajnog vektora $\vec{X}$

$$\Sigma = \begin{bmatrix} Cov(X_1, X_1) & Cov(X_1, X_2) & \dots & Cov(X_1, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_n, X_1) & Cov(X_n, X_2) & \dots & Cov(X_n, X_n) \end{bmatrix}$$

↳ pretpostavljamo da je ona simetrična i da su na dijagonali varijance

↳ ekstremni slučajevi:

a) dijagonalna kovarijacijska matrica

↳ nema ravnoteže među parovima

$$\Sigma = \begin{bmatrix} \sigma_{X_1}^2 & 0 \\ 0 & \sigma_{X_2}^2 \dots \end{bmatrix} = diag(\sigma_i^2)$$

b) izotropna kovarijacijska matrica

↳ sve imaju isto raspršenje

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \dots \end{bmatrix} = \sigma^2 I$$

↳ uvijek je poželjno semi-definitna:

$$\vec{x}^T \Sigma \vec{x} \geq 0$$

↳ inverz kovarijacijske matrice:

- ne mora nužno postojat
- Σ je singularna u dva slučaja:
  1. neki od elemenata na dijagonal je jednak nul ($\sigma_i^2 = 0$)
  2. postoje linearno ovisni retci/stupci

⇒ NEZAVISNOST VARIJABLI:

↳ za nezavisne varijable vrijedi:

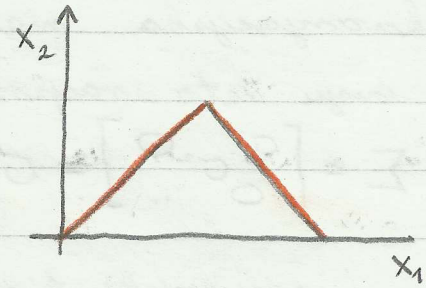
$$P(X,Y) = P(X) \cdot P(Y)$$

↳ vrijedi slijedeća tvrdnja, ali ne i njen obrat:

$$X \text{ i } Y \text{ su nezavisne} \Rightarrow Cov(X,Y) = 0$$

↳ kovarijanca je jednaka nul ako su varijable linearno nezavisne, te tvrdnja vrl da ako su varijable (vjerojatnosno, stohastički) nezavisne, onda su sigurno i linearno nezavisne

↳ **obrat** ne vrijedi:



kovarijanca je nula, a nisu nezavisne

OSNOVNE VJEROJATNOSNE RAZDIOBE

⇒ ponovno se bavimo parametarskim modelima, pa se pretpostavlja da se podaci pokoravaju nekoj razdiobi

⇒ DISKRETNE:

a) jednodimenzijska:

- binarna: Bernoullijeva
- viševrijednosna: Kategorička (multinuljeva)

b) višedimenzijska:

- vektor binarnih / viševrijednosnih varijabli

⇒ KONTINUIRANE:

a) jednodimenzijska:

- univarijantna normalna (Gaussova)

b) višedimenzijska:

- multivarijantna normalna (Gaussova)

⇒ napomena: čim su brojevi na ulazu, rabi kontinuirane razdiobe kako bi imao vezu u kontinuiranom prostoru (npr. veza između ocjena - neću da je udaljenost između 2 i 3 isto kao i između u 1 i 3, što bi bilo da sam koristio diskretnu razdiobu)

⇒ BERNOULLIJEVA VARIJABLA:

- $P(X=x|\mu) = \begin{cases} \mu & \text{ako } X=1 \\ 1-\mu & \text{inače} \end{cases} = \mu^x(1-\mu)^{1-x}$

- $E[X] = \mu$

- $Var(X) = \mu(1-\mu)$

↳ za više-dimenzionalni prostor imamo vektor Bernoullijevih varijabli:

$$\vec{X} = (x_1, x_2, x_3, \ldots, x_M)$$

0 0 0 1
0 0 1 0
0 1 0 0
1 0 0 0

} 1-hot-coding za 4 moguće vrijednosti

$$P(X=\vec{x}|\vec{\mu}) = \prod_{k=1}^{K} \mu_k^{x_k}$$

$$\vec{\mu} = (\mu_1, \mu_2, \ldots, \mu_K)$$

NA TAJ NAČIN OSTVARUJEMO MULTINULIJEVU VARIJABLU POMOĆU BERNOULLIJEVE

1

PR: Imamo sljedeći vektor: $\vec{x} = x_3 = (0, 0, 1, 0)$

$$\vec{\mu} = (0.2, 0.3, 0.4, 0.1)$$

$$P(\vec{x}=x_3) = 0.2^0 \cdot 0.3^0 \cdot 0.4^1 \cdot 0.1^0 = 0.4$$

⇒ UNIVARIJANTNA GAUSSOVA VARIJABLA:

↳ koristi se za modeliranje šuma

- $P(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
- $E[x] = \mu$
- $Var(x) = \sigma^2$

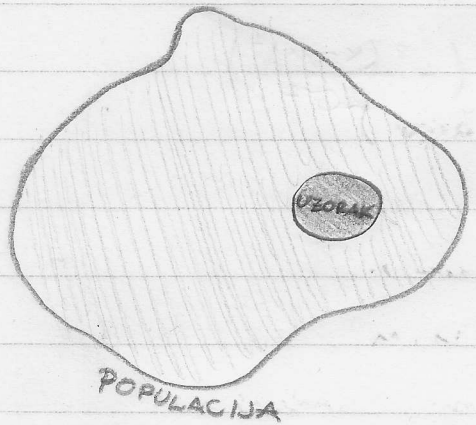
⇒ MULTIVARIJANTNA GAUSSOVA VARIJABLA:

- $P(\vec{x}|\vec{\mu},\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} \cdot (\vec{x}-\vec{\mu})\right)$

↳ trebamo inverz kovarijacijske matrice, pa ne smije biti singularna

↳ ako je Σ blizu singularne, imamo nestabilno rješenje (treba identificirat značajke koje to uzrokuju i eliminirat ih ☺)

PROCJENA PARAMETARA



⇒ želimo da je naš uzorak reprezentativan - da smo slučajno uzorkovali populaciju

↳ uzorak je zapravo vektor slučajnih varijabli:

$$(X_1, X_2, \ldots, X_N)$$

↳ da bi uzorak bio reprezentativan, želimo da vrijedi skraćenica „IID":

- „Independently and identically distributed"

⇒ **STATISTIKA:** (*statistic*)

- slučajna varijabla koja je funkcija vektora slučajnih varijabli:

$$\Theta = g(X_1, X_2, \ldots, X_N)$$

$\hat{\mu} = \frac{1}{N} \sum x^i$

→ procjena za Gaussa

⇒ PROCJENITELJ: (estimator)

- statistika koja računa parametar Θ distribucije

BITNO!

⇒ PRISTRANOST PROCJENITELJA: (bias)

- greška koju dobivamo zbog toga što je uzorak ograničen (nemamo pogled na sve)
- točnije, to je razlika između očekivane vrijednosti procjenitelja i stvarne vrijednosti parametra populacije (njega ne znamo):

$$b_\Theta(\hat{\Theta}) = E[\hat{\Theta}] - \Theta$$

↳ za nepristran procjenitelj vrijedi:

$$E[\hat{\Theta}] = \Theta$$

Pr: Izračunaj procjenitelje s pristranost za Gaussa!

$$E[x] = \mu$$
$$Var(x) = \sigma^2$$
$$x \in \mathbb{R}$$

↳ ne znamo μ niti σ^2!

$$\mu, \sigma^2 = ?$$

↳ izračunajmo procjenitelje:

$$\hat{\mu} = \frac{1}{N} \sum_i x^i \qquad \hat{\sigma}^2 = \frac{1}{N} \sum_i (x^i - \mu)^2$$

↳ pitamo se da li je naša procjena dobra:

$$E[\hat{\mu}] \stackrel{?}{=} \mu \qquad E[\sigma^2] \stackrel{?}{=} \sigma^2$$

↳ računamo (ne znam što?!):

$$E[\hat{\sigma}^2] = \frac{N-1}{N} \sigma^2$$

$$b(\hat{\sigma}^2) = E[\hat{\sigma}^2] - \sigma^2 = -\frac{\sigma^2}{N}$$

PROCJENITELJ MLE

⇒ raspolažemo skupom podataka: $\mathcal{D} = \{\vec{x}^i\}_{i=1}^N$

⇒ dvije pretpostavke:

a) primjeri se vuku po razdiobi:

$$\vec{x} \sim p(\vec{x} \mid \vec{\theta})$$

b) primjeri su „iid"

⇒ izračunajmo vjerojatnost naših uzoraka:

$$P(\mathcal{D} \mid \vec{\theta}) = p(x^1, x^2, \ldots, x^N \mid \vec{\theta})$$

↓ primjenimo „iid" pretpostavku

$$P(\mathcal{D} \mid \vec{\theta}) = \prod_{i=1}^{N} (x^i \mid \vec{\theta})$$

⇒ izvukli smo neki uzorak iz populacije, pa je izglednost parametara jednaka vjerojatnosti uzorka:

$$\boxed{P(\mathcal{D} \mid \vec{\theta}) = \mathcal{L}(\vec{\theta} \mid \mathcal{D})}$$

⇐ IZGLEDNOST (likelihood)

varijabla ↑ fiksno ↑

BITNO!

⇒ stoga, izglednost je: $\mathcal{L}: \vec{\theta} \to P(\mathcal{D} \mid \vec{\theta})$

PR: Pogledajmo pokus bacanja novčića 10 puta u kojem je 8 puta pala glava, a 2 puta pismo. Odredimo izglednost?

↳ ovo je binarna Bernoullijeva varijabla:

$$P(X = x \mid \mu) = \mu^x (1-\mu)^{1-x}$$

↳ izglednost je tada:

$$\mathcal{L}(\mu \mid D) = P(D \mid \mu) = p(x^1, x^2, \ldots, x^{10} \mid \mu)$$

$$= \prod_{i=1}^{10} p(x^i \mid \mu) = \mu^8 (1-\mu)^2$$

⇒ iz primjera zaključujemo da je izglednost za općenite binarne Bernoullijeve varijable:

↳ ako je $m = \sum_{i=1}^{N} x^i$ broj glava, onda izglednost je:

$$\boxed{\mathcal{L}(\mu \mid D) = \mu^m (1-\mu)^{N-m}}$$

⇒ **PROCJENITELJ MLE:**

- maksimiziramo funkciju izglednosti ("maximum likelihood estimator")
- ovo je oportunistički jer pretpostavljamo da su primjeri koje vidimo najizgledniji ☺

↳ ako to napišemo formalno, dobivamo formulu:

$$\hat{\vec{\theta}}_{MLE} = \arg\max_{\vec{\theta}} \mathcal{L}(\vec{\theta} | D)$$

↳ raspisujemo:

$$\hat{\vec{\theta}}_{MLE} = \arg\max_{\vec{\theta}} \prod_{i=1}^{N} p(x^i | \vec{\theta})$$

↳ ako ovo želimo maksimizirati, lakše je maksimizirati sumu, pa logaritmiramo:

$$\arg\max_{\vec{\theta}} \ln(\mathcal{L}(\vec{\theta} | D)) = \arg\max_{\vec{\theta}} \sum \ln(p(x^i | \vec{\theta}))$$

↳ ovo ćemo htjeti obično maksimizirati analitički, ali kada neće biti moguće, radit ćemo iterativno !

⇒ izvedimo MLE za bitne varijable kako bi malo stekli osjećaj za ovo:

a) **MLE za Bernoullijevu varijablu:**

$$\ln \mathcal{L}(\mu|D) = \ln(P(D|\mu)) = \ln\left(\prod_{i=1}^{N} p(x^i|\mu)\right)$$

$$= \sum_{i=1}^{N} \ln(p(x^i|\mu)) = \sum_{i=1}^{N} \ln\left(\mu^{x^i}(1-\mu)^{1-x^i}\right)$$

$$= \sum_{i=1}^{N} \left(x^i \ln\mu\right) + \left(N - \sum_{i=1}^{N} x^i\right) \ln(1-\mu)$$

↳ ovo želimo maksimizirati:

$$\frac{d \ln \mathcal{L}(\mu|D)}{d\mu} = 0$$

⋮ nije teška derivacija

↳ dobiva se:

$$\hat{\mu}_{MLE} = \frac{1}{N} \sum_{i=1}^{N} x^i$$

$$\hat{\mu}_{MLE} = \frac{m}{N}$$

↓

relativna frekvencija

b) MLE za kategoričku varijablu:

$$\ln \mathcal{L}(\vec{\mu} | D) = \ln P(D | \vec{\mu}) = \ln \left(\prod_{i=1}^{N} p(x^i | \vec{\mu}) \right)$$

$$= \ln \left(\prod_{i=1}^{N} \prod_{j=1}^{K} \mu_K^{x_K^i} \right)$$

↳ ovo ne možemo derivirat i izjednačit s nulom jer imamo ograničenje:

$$\sum_{j=1}^{K} \mu_K = 1$$

↳ stoga, ovo minimiziramo uz Lagrangeovu funkciju (ne treba znat)

↳ dobiva se:

$$\hat{\mu}_{K, MLE} = \frac{1}{N} \sum x_K^i$$

$$\hat{\mu}_{K, MLE} = \frac{N_K}{N}$$

c) MLE za Gaussovu varijablu:

$$\ln \mathcal{L}(\mu, \sigma^2 | D) = \ln P(D | \mu, \sigma^2) = (*)$$

↳ primjena „iid" pretpostavke:

$$(*) = \ln\left(\prod_{i=1}^{N} p(x^i | \mu, \sigma^2) \right) =$$

$$= \ln\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(x^i - \mu)^2}{2\sigma^2} \right) \right)$$

↳ deriviramo po dane varijable:

$$\frac{\partial \ln(\mathcal{L}(\mu, \sigma^2 | D))}{\partial \mu} = 0 \qquad \frac{\partial \ln(\mathcal{L}(\mu, \sigma^2 | D))}{\partial \sigma^2} = 0$$

$$\hat{\mu}_{MLE} = \frac{1}{N} \sum_{i=1}^{N} x^i \qquad \sigma^2_{MLE} = \frac{1}{N} \sum_{i=1}^{N} (x^i - \hat{\mu}_{MLE})^2$$

⇒ NAPOMENA:

↳ težine koje daju MLE su iste one koje bi dobili minimizacijom empirijske pogreške (prisjeti se izvoda pogreške na linearnu regresiju)

↳ stoga projektelji su napravo 3. dio algoritma strojnog učenja: optimizacijski postupak

PROCJENITELJ MAP

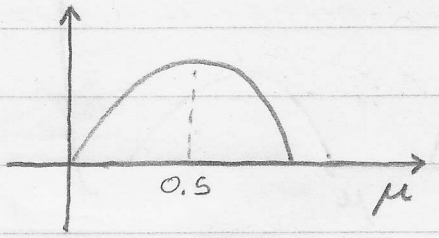
⇒ problem MLE-a je prenaučenost:

- ako 5 puta bacimo novčić, dobijemo 5 puta glavu, MLE će reći da je vjerojatnost pisma nula što je ekstremno ‼

↳ želimo ugraditi neko znanje u procjenitelja (npr. očekujemo da je vjerojatnost glave na novčiću 0.5)

⇒ stoga, uvodimo APRIORNU DISTRIBUCIJU PARAMETARA:

- oznaka: $p(\vec{\theta})$
- npr, za novčić je najvjerojatnije da je vjerojatnost glave 0.5:



ovo je gustoća, pa je integral 1, a ne amplituda

⇒ **procjenitelj MAP:**

$$p(\vec{\theta}|D) = \frac{p(D|\vec{\theta})\,p(\vec{\theta})}{p(D)}$$ } FIKSNO

$$\hat{\theta}_{MAP} = \arg\max_{\vec{\theta}} p(\vec{\theta}|D)$$

znanje iz podataka

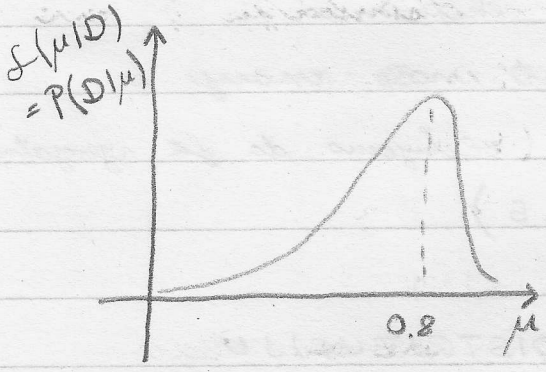
$$\boxed{\hat{\theta}_{MAP} = \arg\max_{\vec{\theta}} p(D|\vec{\theta})\,p(\vec{\theta})}$$

apriorno znanje

PRS Poglejmo MAP na primeru s novčićem!

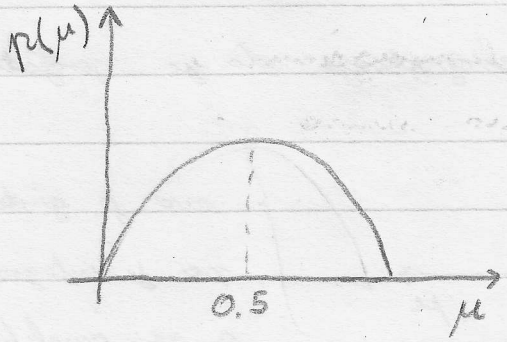
$N = 10$, $n = 8$

$$P(\mu|D) \sim P(D|\mu) \cdot p(\mu)$$



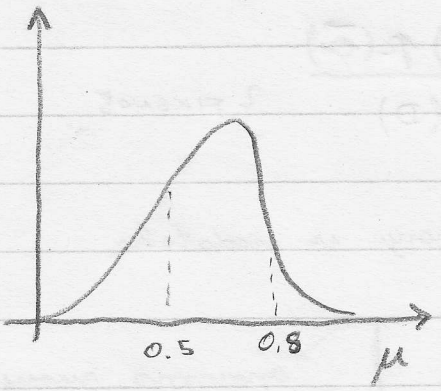
} znanje iz podataka

X



} znanje iz prakse

=



} rezultat dobiven kombiniranjem znanja

(pomaknute u lijevo)

⇒ pokažimo sada zašto uopće MAP radi: strogim matematičkim postupkom:

- želimo dobiti: $P(\theta|D)$

$$P(\theta|D) \sim P(D|\theta)\, p(\theta)$$

ako je apriorna distribucija parametra konjugatna distribucija za $P(D|\theta)$, onda su $P(\theta|D)$ i $p(\theta)$ iste distribucije, te nam to omogućava puno raklgućaka

↳ kažemo da su tada $P(\theta|D)$ i $p(\theta)$ konjugatne

- mi ćemo htjet birat konjugatne distribucije (ne znam zoš zašto, al ćemo htjet pažljivo birat distribucije)

BETA - BERNOULLIJEV MODEL

⇒ koristimo beta distribuciju jer je ona konjugatna na Bernoullijevu ▽

⇒ **beta distribucija:**

$$p(\mu \mid \alpha, \beta) = \frac{1}{B(\alpha, \beta)} \mu^{\alpha-1} (1-\mu)^{\beta-1}$$

$$\alpha, \beta > 0$$

↳ pogledaj kako ona izgleda i pokušaj to razumjeti (npr. za $\alpha = \beta = 1$ imamo ravnu crtu, pa to znači da nemamo nikakve apriorne znanje o problemu te se MAP-a degenerirano u MLE)

↳ korisno za beta distribuciju je da uvijek znamo gdje je maksimum:

$$\text{mod} = \frac{\alpha - 1}{\alpha + \beta - 2}$$

PR: Pogledajmo pokus s novčićem u kojem je od „N" bacanja „m" puta pala glava:

$$\mathcal{L}(\mu|D) = P(D|\mu) = \mu^m (1-\mu)^{N-m}$$

↳ prema Bayesu vrijedi:

$$P(\mu|D,\alpha,\beta) = \underbrace{\mu^m (1-\mu)^{N-m}}_{\text{vjerodostojnost}} \cdot \underbrace{\frac{1}{B(\alpha,\beta)} \mu^{\alpha-1}(1-\mu)^{\beta-1}}_{\text{apostemorna vjerojatnost}} \cdot \frac{1}{P(D)}$$

$$P(\mu|D,\alpha,\beta) = \underbrace{\mu^{m+\alpha-1}(1-\mu)^{N-m+\beta-1}}_{\text{vidimo da smo opet dobili beta distribuciju, samo s drugim parametrima } (\alpha' = m+\alpha,\ \beta' = N-m+\beta)} \cdot \underbrace{\frac{1}{B(\alpha,\beta)\cdot P(D)}}_{\text{ovo je sama normalizacijska faktor jer integral svega ovoga mora biti 1 (te je onda } \frac{1}{B(\alpha',\beta')})}$$

↳ stoga, maksimum ove distribucije je ovdje:

$$\text{mod} = \frac{\alpha'-1}{\alpha'+\beta'-2} = \frac{m+\alpha-1}{\alpha+N+\beta-2}$$

⇒ stoga, MAP procjenitelj je:

$$\boxed{\hat{\mu}_{MAP} = \frac{m+\alpha-1}{\alpha+N+\beta-2}}$$

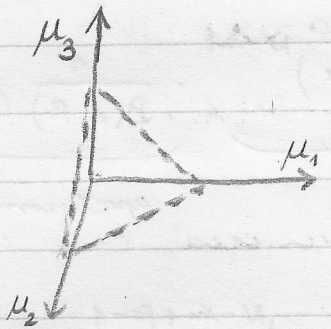
(vidi da za $\alpha = 1 = \beta$ ovo degradira u MLE)

⇒ Dirichletova distribucija:

- poopćenje beta distribucije

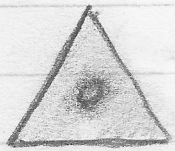
(slično kao što je multinulljeva distribucija poopćenje bernoullijeve)

- vid simpleksa:



$$\mu_1 + \mu_2 + \mu_3 = 1$$

↳ ako su sva tri „μ" ista, najvećа vjerojatnost je u sredini:



↳ više o tome u domaćoj radaći!

BAYESOV KLASIFIKATOR

⇒ <u>o pravilima vjerojatnosti:</u>

↳ ideja je da se algebra vjerojatnosti može svest na dva jednostavna pravila:

a) PRAVILO ZBROJA:

$$P(X) = \sum_{Y} P(X, Y)$$

↳ marginalizacija za više varijabli:

$$P(X, Y) = \sum_{Z} P(X, Y, Z)$$

$$P(X) = \sum_{Y} \sum_{Z} P(X, Y, Z)$$

b) PRAVILO UMNOŠKA:

↳ uvodi se uvjetne vjerojatnosti:

$$P(Y|X) = \frac{P(X, Y)}{P(X)}$$

$$P(X, Y) = P(X) \cdot P(Y|X)$$

$$P(X, Y) = P(Y) \cdot P(X|Y)$$

⇒ NAPOMENA: neki korisni nazivi koji će se pojavljivati:

$P(X)$ ⇒ marginalna vjerojatnost

$P(X, Y)$ ⇒ zajednička vjerojatnost („joint")

⇒ BAYESOVO PRAVILO:

↳ složeno pravilo koje će nam pomoći jer pomoću njega možemo abduktivno zaključivati (što u logici nije raspoloživo)

· tu je snaga Bayesovog pravila

$$P(X,Y) = P(Y|X)\,P(X) = P(X|Y)\,P(Y) \quad / : P(X)$$

$$\boxed{P(Y|X) = \frac{P(X|Y)\,P(Y)}{P(X)}}$$

⇒ PRAVILO LANCA:

↳ također korisno složeno pravilo — faktori:

$$P(X,Y,Z) = P(X)\,P(Y|X)\cdot P(Z|X,Y)$$

↓ generalizirano!

$$P(X_1,\ldots,X_n) = \prod_{k=1}^{n} P(X_k \mid X_1,\ldots,X_{k-1})$$

⇒ BAYESOV KLASIFIKATOR:

↳ direktno poopćenje Bayesova pravila!

$$P(Y|\vec{X}) = \frac{p(\vec{X}, Y)}{p(\vec{X})} = \frac{p(\vec{X}|Y)\ P(Y)}{p(\vec{X})}$$

$P(Y|X)$ ⇒ aposteriorna vjerojatnost (koliko je oznaka vjerojatna uz primjere)

$p(\vec{X}|Y)$ ⇒ izglednost oznake/klase (vjerojatnost podataka uz svu klasu)

$P(Y)$ ⇒ izglednost klase (apriorna vjerojatnost klase - „class prior")

$p(\vec{X})$ ⇒ gustoća vjerojatnosti primjera (nekada zovemo PDF → „Probability Density Function")

↳ problem Bayesa u ovom obliku je to da su ove gustoće vjerojatnosti kompliciranе za modelirati (to su $p(\vec{X}|Y)$ i $p(\vec{X})$)

↳ dodatno, u praksi ne znamo PDF primjera, pa poopćenjem pravila zbroja:

$$p(\vec{X}) = \sum_Y p(\vec{X}|Y) = \sum_Y p(\vec{X}|Y)\,P(Y)$$

OKRENI

↳ konačan oblik Bayesovog klasifikatora:

$$h_k(\vec{x};\vec{\theta}) = P(Y=k|\vec{x}) = \frac{p(\vec{x}|Y=k)\,P(Y=k)}{\sum_j p(\vec{x}|Y=j)\,P(Y=j)}$$

⇓

problemi slijedeće iz formule:

- može biti problema raditi na proizvoljno mnogo klasa
- dosta vjerojatnosni izrazi

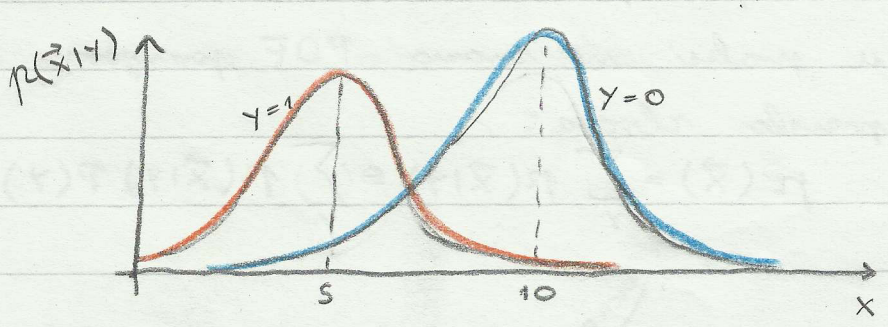
↳ ako nas ne zanimaju vjerojatnosti pripadnosti primjera nekoj klasi, pišemo:

$$h(\vec{x};\vec{\theta}) = \arg\max_y p(\vec{x}|y)\,P(y)$$

PR: Pretpostavlja se Gaussova distribucija za dvije klase:

$K = 2$

$n = 1 \Rightarrow x \in \mathbb{R}$



više brda znači da imamo manje varijanci, pa te znači da ima manje šuma

⇒ za Bayesa funkciju pogreške i optimizacijski postupak rješavamo nekim od procjeniteljima (MAP, MLE, ...)

↳ stoga, primjet da su sve 3 komponente algoritma strojnog učenja cijelo vrijeme tu!

⇒ GENERATIVNI MODELI:

- prvo izaberi „Y", a zatim opisujemo kako su primjeri nastali:

$$P(Y) \longrightarrow p(\vec{x} \mid Y)$$

- ovo se naziva generativna priča jer opisuje kako podaci nastaju
- upoznat ćemo još generativnih modela, ali zasad će nam Bayes biti dosta!
- još jednom, generativni model modeliraju zajedničku distribuciju

⇒ <u>napomena</u>: većinom ćemo se bavit diskretnim domenama (pa pišemo veliki „P"), no isto vrijedi za kontinuirane (samo se radi integral umjesto sume)

PR: Klasificirajmo diskretan primjer u 3 klase!

$$K = 3$$

$$\left.\begin{aligned} P(Y=1) &= P(Y=2) = 0.3 \\ P(Y=2) &= 0.4 \\ P(\vec{x} \mid Y=1) &= 0.9 \\ P(\vec{x} \mid Y=2) &= 0.4 \\ P(\vec{x} \mid Y=3) &= 0.4 \end{aligned}\right\} \text{ovo je zadano!}$$

↳ sada samo uvrštavamo u Bayesovu formulu:

$$P(x \mid Y=1) \cdot P(Y=1) = 0.27$$
$$P(x \mid Y=2) \cdot P(Y=2) = 0.12$$
$$P(x \mid Y=3) \cdot P(Y=3) = 0.16$$

$$\Downarrow$$

vidimo da spada u klasu 1!

↳ ako želimo vjerojatnost pripadnost klasama, trebamo računat:

$$P(\vec{x}) = \sum_{Y} P(\vec{x} \mid Y) \cdot P(Y) = 0.55$$

$$P(Y=1 \mid \vec{x}) = 0.49$$
$$P(Y=2 \mid \vec{x}) = 0.22$$
$$P(Y=3 \mid \vec{x}) = 29$$

↳ oznake nisu najbolje (proboj pazit na malo „p" i veliko „P")

⇒ GENERATIVNI VS DISKRIMINATIVNI:

↳ prisjetimo se logističke regresije:

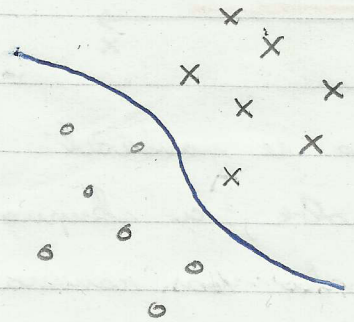
$$h(\vec{x}; \vec{w}) = P(y \mid \vec{x}) = \sigma(\vec{w}^T \vec{x})$$

↓

samo modelira aposteriornu vjerojatnost ▽!

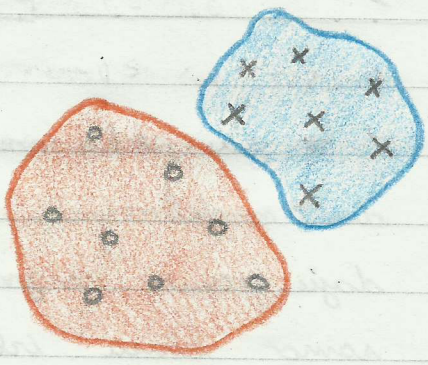
↳ Bayes u sebi ima više vjerojatnost, pa se tu uklapa generativna priča

DISKRIMINATIVNO



↳ generiraju samo granicu među klasama

GENERATIVNO



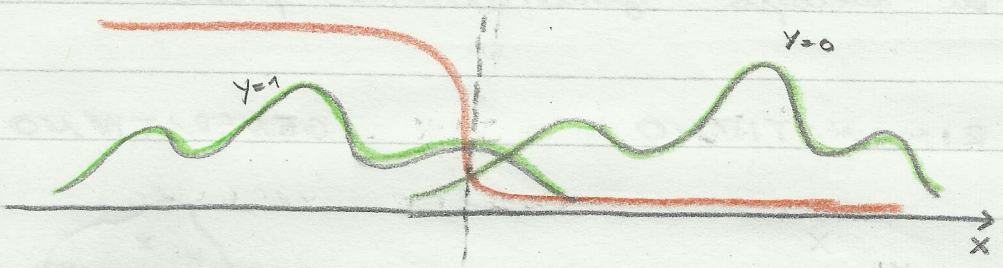
↳ generiraju „oblake", a granica slijedi implicitno ▽!

⇒ prednosti generativnih modela:

- lako ugradnja znanja stručnjaka
- interpretabilnost rezultata (zbog teorije vjerojatnosti)
- ocjena pouzdanosti (zbog vjerojatnosti na izlazu)
- možemo odbiti klasifikaciju ako nam vjerojatnost na izlazu nije dovoljno visoka
- nalaženje stršećih vrijednosti (outliers)

⇒ **nedostaci generativnih modela:**

- iziskuju mnogo primjera (trebamo puno podataka da bi imali dobre vjerojatnosti)
- nepotrebna komplikacija modela ako nam nisu potrebne vjerojatnosti na izlazu nego nas zanima samo granica:



↳ sa slike vidimo da smo se namučili da modeliramo zelene razdiobe, a skupno daju samo sigmoidu, pa ako nas zanima samo klasa izlaza, bolji je izbor logistička regresija (idemo zaobilaznim putom)

⇒ „OBIČNA" NEZAVISNOST:

↳ varijable su (marginalno / standardno) nezavisne ako vrijedi:

$$P(X,Y) = P(X) \cdot P(Y)$$

$$P(X|Y) = P(X)$$

$$P(Y|X) = P(Y)$$

↳ oznaka: $X \perp Y$

⇒ UVJETNA NEZAVISNOST:

↳ varijable su uvjetno nezavisne ako vrijedi:

$$P(X|Y,Z) = P(X|Z)$$

$$P(Y|X,Z) = P(Y|Z)$$

$$P(X,Y|Z) = P(X|Z) \cdot P(Y|Z)$$

↳ oznaka: $X \perp Y | Z$

↳ kako treća varijabla može utjecat na veru ove dvoje?

· pogledaj primjer:

$X =$ „student je primljen na FER"

$Y =$ „student je primljen na PMF-MAT"

$$P(Y|X) \neq P(Y)$$

↳ intuitivno su zavisni ∇

↳ dodajmo tvrdnju:

$Z =$ „student je bio na državnom iz MAT"

↳ sada su varijable uvjetno nezavisne jer je primljen na oba fakulteta:

$$P(Y|X,Z) = P(Y|Z) \quad \Big\} \quad X \perp Y | Z$$

NAIVAN BAYESOV KLASIFIKATOR

⇒ krenimo odmah s izvodom jer smo ovo radili na umjetnoj inteligenciji:

$$P(Y|\vec{X}) = \frac{P(\vec{X}|Y)\,P(Y)}{\sum_{Y'} P(\vec{X}|Y')\,P(Y')} = \frac{P(X_1,\ldots,X_n|Y)\cdot P(Y)}{\sum_{Y'} P(X_1,\ldots,X_n|Y')\,P(Y')}$$

⇒ problem je brojnik: $P(X_1,\ldots,X_n|Y)$ jer nam on rahtjeva eksponencijalan broj parametara:

↳ npr. za $K=2$, $X_i \in \{0,1\}$, n-dimenzionalni prostor: broj parametara:

$$(2^n-1)\cdot 2.$$

↳ stoga za crno bijelu sliku veličine 100 piksela u širinu, visinu imamo jako-jako puno parametara, pa to neće ić

⇒ drugi problem je to što se model neće dobro ponašat na nevidenim primjerima (ovo se vidi iz formule, ali ne znam kako)

↳ ako se ta konkretna kombinacija značajki nije pojavila, vjerojatnost je nula - valjda se tako vidi iz modela

⇒ uvodimo naivnu pretpostavku:

$$X_k \perp X_1, \ldots, X_{k-1} \mid Y$$

↳ time možemo dalje za brojnik rastavit ovako:

$$P(X_k \mid X_1, \ldots, X_{k-1}, Y) = P(X_k \mid Y)$$

↳ a samim time vrijedi:

$$P(X_1, \ldots, X_n \mid Y) = \prod_{k=1}^{n} P(X_k \mid Y)$$

⇒ „NAIVE B. C.":

$$h(\vec{x} \mid \vec{\theta}) = \operatorname{argmax}_y P(y) \prod_{k=1}^{n} P(X_k \mid y)$$

↳ broj parametara za faktor $P(X_k \mid Y)$:

- X_k poprima K_k vrijednosti:

$$(K_k - 1)K$$

- za ukupno „n" faktora:

$$\sum_{k=1}^{n} (K_k - 1)K$$

LINEARNA OVISNOST ☺

⇒ napomena: ovaj „−1" u $(K_k - 1)K$ je zato što jednu vjerojatnost ne moramo znat jer je njihov zbroj jednak 1

POLUNAIVAN BAYESOV KLASIFIKATOR

⇒ prisjetimo se običnog „naivnog" Bayesa:

$$h(x_1, \ldots, x_n) = \arg\max_y P(y) \prod_{k=1}^{n} P(x_k | y)$$

↳ dodatno, prisjetimo se uvjetne nezavisnosti:

$$X \perp Y \mid Z$$

$$P(X \mid Y, Z) = P(X \mid Z)$$

$$P(Y \mid X, Z) = P(Y \mid Z)$$

$$P(X, Y \mid Z) = P(X \mid Z) P(Y \mid Z)$$

↳ pitanje je, jesu li varijable X_k, X_j i Y stvarno uvjetno nezavisne?

$$X_k \perp X_j \mid Y$$

(hint: <u>nisu</u>, to nam je motivacija za polunaivnog Bayesovog klasifikatora jer se želimo približiti stvarnosti tako da ne pretpostavimo nezavisnost svih komponenti vektora $\vec{X}$, nego neke parove komponenti promatramo kao zavisne parove)

PR: Novinski članak sastavljen od riječ:

$\vec{X}$ je novinski tekst

Y su rubrike

$$Y = \{ \text{Sport}, \text{Politika}, \text{Kriminal} \}$$

↳ Značajke:

$$X_1 = 1\{\text{„rezultat"} \in \vec{X}\}$$

$$X_2 = 1\{\text{„lopta"} \in \vec{X}\}$$

$$X_3 = 1\{\text{„gol"} \in \vec{X}\}$$

↳ vrijedi da lopta i gol nisu uvjetno nezavisne:

$$X_2 \not\perp X_3 \mid Y$$

$$P(X_3 \mid X_2, Y) \neq P(X_3 \mid Y)$$

$\Downarrow$

- pokušaj si objasniti ovaj primjer jer je on motivacija za polunaivan Bayesov klasifikator

⇒ prijeđimo sada na polunaivan Bayesov klasifikator:

- on je nešto složenije nego skroz naivan Bayesov klasifikator jer združujemo neke varijable kako bi se približil stvarnom svijetu:

$$P(Y \mid X_1, X_2, X_3) \propto P(X_1 \mid Y) \underbrace{P(X_2 \mid Y) \cdot P(X_3 \mid Y)} P(Y)$$

združujemo ih jer ovdje naivna pretpostavka nije baš dobra:

$$P(X_2 X_3 \mid Y)$$

↳ <u>no, kako odlučiti koje varijable stopiti?</u>

- kako saznati koje varijable su najviše zavisne?
  - sličnost varijabli bi mogli izmjeriti nekom vrstom mjere zavisnosti (korelacija nije najbolja jer mjeri samo linearnu zavisnost)

- drugi način je da isprobavamo stapanje različitih varijabli (unakrsna provjera, ali bez povratka jer je prostor pretraživanja jako velik - broj particija skupa:
  - Bellov broj: $B_3 = 5$

    $B_4 = 15$

    $\vdots$

    $B_{10} = 115975$)

- ovaj „drugi način" nas upućuje da isprobamo sparat dvije varijable, ako je model bolji, nastavo sparivat druge, a ako je gori, vratit se na model bez sparenih varijabli
  - ↳ ne isprobavamo sve moguće parove jer se ne vraćamo na već isprobane varijable

⇒ MJERENJE ZAVISNOSTI:

↳ treba nam za prvi način

↳ pogledajmo jednakost:

$$P(x,y) = P(x) \cdot P(y)$$

↳ gledamo lijevu i desnu stranu jednakosti kao dvije distribucije i što su one različitije, to su varijable više međusobno ravisne

↳ za mjeru sličnosti dvije distribucije trebamo Kullback-Leiblerovu divergenciju:

$$D_{KL}(P \| Q) = -\sum_x P(x) \ln\left(\frac{P(x)}{Q(x)}\right)$$

↳ primijenimo KL divergenciju na dvije varijable:

$$I(x,y) = D_{KL}\left(P(x,y) \| P(x) \cdot P(y)\right)$$

$$I(x,y) = -\sum_{x,y} P(x,y) \ln\left(\frac{P(x,y)}{P(x)\,P(y)}\right)$$

↖ drugi način je RELATIVNA ENTROPIJA

↑ uzajamna informacija (MUTAL INFORMATION)

} ako je međusobna informacija veća, postoj protok informacija među varijablama i ravisnost je veća! //

GAUSS - BAYESOV KLASIFIKATOR

$$p(\vec{x} \mid y)$$

$\vec{x}$ ↑ vektor kontinuiranih značajki

y ↑ Gaussova varijabla
↳ ona služi za modeliranje šuma

⇒ zbog jednostavnosti, radit ćemo se bavit jednom dimenzijom:

$$p(x \mid y=j) = \frac{1}{\sqrt{2\pi}\,\sigma_j} \exp\left(-\frac{(x-\mu_j)^2}{2\sigma_j^2}\right)$$

$$h(x \mid \vec{\theta}) = \arg\max_y p(x \mid y)\, p(y)$$

↳ pogledajmo model jedne klase:

$$h_j(x \mid \vec{\theta}) = p(x, y=j) = p(x \mid y=j)\, p(y=j)$$

↑ gledaj kao posteriornost da primjer pripada klasi „j"

(ima smisla gledati samo ovdje vjeru relativno s drugima, i tamo gdje je najveća, tamo klasificiramo!)

RELATIVNA MJERA!

↳ ovo možemo logaritmirati zbog traženja maksimuma po klasama, pa se rezultat neće promjenit:

$$h_j(x|\vec{\theta}) = \ln\left(p(x|y=j)\cdot p(y=j)\right) =$$

$$= \ln\left(p(x|y=j)\right) + \ln\left(p(y=j)\right)$$

$$= -\frac{1}{2}\ln(2\pi) - \ln\sigma_j - \frac{(x-\mu_j)^2}{2\sigma_j^2} + \ln\left(p(y=j)\right)$$

- što su nam sve parametri $\vec{\theta}$?

  ↳ pa, sigurno su „μ_j", „σ_j", ali još iz apriorne distribucije sigurno dolazi jedan parametar (to je ovaj zadnj član), njega nazovimo „μ'":

$$\vec{\theta} = (\mu_j, \sigma_j, \mu_j')$$

$$\Downarrow$$

ovo obično procjenjujemo pomoću nekog od procjenitelja

⇒ za višedimenzionalni prostor, trebamo multivarijantnu Gaussovu distribuciju:

$$p(\vec{x} \mid y=j) = \frac{1}{(2\pi)^{m/2} |\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu}_j)^T \Sigma^{-1} (\vec{x}-\vec{\mu}_j)\right)$$

$$h_j(\vec{x} \mid \vec{\theta}) = \ln p(\vec{x} \mid y=j)\, p(y=j) = \dots$$

$$\vdots$$

$$h_j(\vec{x} \mid \vec{\theta}) = -\frac{1}{2} \ln |\Sigma_j| - \frac{1}{2}(\vec{x}-\vec{\mu}_j)^T \Sigma_j^{-1} (\vec{x}-\vec{\mu}_j) + \ln\left(P(y=j)\right)$$

↳ pogledajmo da imamo jako puno parametara, pa trening modela može biti zahtjevan:

$$\underbrace{\frac{m}{2}(m+1)}_{\text{trebamo polu } \Sigma_j \text{ jer je ta matrica simetrična}} \cdot K + \underbrace{m \cdot K}_{\text{za } \mu_j} + \underbrace{K - 1}_{\text{množimo s } K \text{ jer je ovo gore model samo za jednu klasu}}$$

⇓

imamo $O(m^2)$ složenost parametara

⇒ pronađimo sada granicu između druge klase:

$$h_1(\vec{x}) = h_2(\vec{x}) \Rightarrow \quad h_1(\vec{x}) - h_2(\vec{x}) = 0$$

$$h_{12}(\vec{x}) = h_1(\vec{x}) - h_2(\vec{x}) = -\frac{1}{2}\ln|\Sigma_1| - \frac{1}{2}\left(\vec{x}^T\Sigma_1^{-1}\vec{x} - 2\vec{x}^T\Sigma_1^{-1}\vec{\mu}_1 + \vec{\mu}_1^T\Sigma_1^{-1}\vec{\mu}_1\right) + \ln\left(P(Y=1)\right)$$

$$- \left(-\frac{1}{2}\ln|\Sigma_2| - \frac{1}{2}\left(\vec{x}^T\Sigma_2^{-1}\vec{x} - 2\vec{x}\Sigma_2^{-1}\vec{\mu}_2 + \vec{\mu}_2^T\Sigma_2^{-1}\mu_2\right) + \ln\left(P(Y=2)\right)\right.$$

...

⇓

ovdje se dosta toga poništi, ali ostaje član:

$$\vec{x}^T\left(\Sigma_1^{-1} - \Sigma_2^{-1}\right)\vec{x}$$

↳ taj član govori da je granica u stvari **kvadratna**

GRANICA JE NELINEARNA!

↳ nelinearnost granice je dobra jer je model ekspresivan, ali je pretežak za učenje jer ima previše parametara, pa trebamo pojednostavljenja modela!

VARIJANTE GAUSS-BAYESOVOG KLASIFIKATORA

① Dijeljena kovarijacijska matrica

↳ umjesto da svaka klasa ima svoju kovarijacijsku matricu, svi modeli imaju istu (dijeljenu)

↳ stoga, kvadratni član se poništava te dobivamo linearnu granicu između klasa

$$\hat{\Sigma} = \sum_j \mu_j \Sigma_j$$

$\Downarrow$ suma

$\Downarrow$ kovarijacijska matrica PAZI!

↳ no, još uvijek imamo kvadratnu ovisnost u broja parametara o broju značajki, samo sada nije taj kvadrat pomnožen sa K jer sve klase imaju istu kovarijacijsku matricu

② Dijagonalna kovarijacijska matrica (isto dijeljena)

↳ kovarijacijsku matricu aproksimiramo dijagonalnom

$$\hat{\Sigma} = diag(\sigma_i^2) \quad \Rightarrow \text{za svaku klasu!}$$

$$|\hat{\Sigma}| = \prod_i \sigma_i$$ } to je zato jer je matrica dijagonalna

(ne postoje kovarijacije između značajki!)

$$\Sigma^{-1} = diag\left(\frac{1}{\sigma_i^2}\right)$$ } isto jer je matrica dijagonalna

↳ ako ovo primjenimo, vidimo da se multivarijantni Gauss raspada na umnožak univarijantnih:

$$P(\vec{x}|y) = \prod_{i=1}^{n} p(x_i|y)$$

↓

primjeti da je ovo zapravo ista formula za Bayesa (naivnog)

· znači, model se raspao na naivnog Bayesa

↳ ovaj model ima linearnu granicu i ima linearnu ovisnost o broju značajki:

$$n + nK + K - 1$$

↳ $O(n)$!

⇒ NAPOMENA: iako smo rekli da su značajke samo linearno neovisne kada smo rekli da je kovarijacijska matrica dijagonalna (jer ona mjeri samo linearnu neovisnost), dobili smo da se model raspada na naivnog Bayesa koji govori da su značajke potpuno neovisne, a ne samo linearno neovisne

↳ ZAŠTO ???

• zato jer izborom Gaussovog modela mi pretpostavljamo linearnost značajki, pa druge ovisnosti ne postoje !

⇒ NAPOMENA: ovdje iz edukativnih razloga gradimo na dijeljenu kovarijacijsku matricu, al možemo imati dijagonalnu kovarijacijsku matricu ber da su dijelimo istu klasu
(onda imamo množenje sa K u broju parametara)

③ Izotropna kovarijacijska matrica (također dijeljena)

↳ dijagonalna, rojednačka i sve varijable imaju isto rasipanje

$$\hat{\Sigma} = \sigma^2 I$$

↳ vrkonture su sada kružnice, a ne elipse jer su rasipanja jednaka za sve varijable

↳ ovo ima smisla ako varijable skaliramo da dobijemo sličnu rasipanja !

↳ uvrstimo pojednostavljenje u model, dobivamo:

$$h_j(\vec{x}) = \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu_{ij})^2 + \ln(P(Y=1))$$

↳ linearan model, najjednostavniji za sada !

⇒ NAPOMENA: postoji još varijacija Gauss-Nexitanovog klasifikatora - neki složeniji, neki jednostavniji, za najbolji za noi problem biramo unakrsnom provjerom !

PROBABILISTIČKI GRAFIČKI MODELI

⇒ sažeti zapis zajedničke („joint") distribucije pomoću grafa
↳ čvorovi grafa su varijable, a bridovi su ovisnosti između tih varijabli
↳ svrha je **probabilističko zaključivanje** - određivanje vrijednosti neopaženih varijabli

⇒ aspekt probabilističkih grafičkih modela su:

a) **reprezentacija**:
↳ lagan prikaz zajedničke (engl. „joint") distribucije
↳ **Bayesove mreže su usmjereni aciklički grafovi**, a Markovljeve mreže su neusmjereni grafovi

b) **zaključivanje**:
↳ određivanje vrijednosti neopaženih varijabli na temelju vrijednosti opaženih varijabli

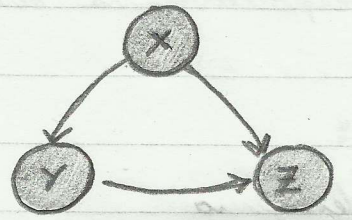
c) **učenje**:
↳ procjena parametara ili učenje strukture - mreže (PGM-a) na temelju podataka
↳ generativno za Bayesove mreže, a diskriminativno za Markovljeve mreže

BAYESOVE MREŽE

⇒ brid povezuje varijable koje utječe s onom koja je uvjetovana



PR: Pogledajmo primjer i na njemu primjenimo pravilo lanca!



$$\Rightarrow P(X,Y,Z) = P(X) \cdot P(Y|X) \cdot P(Z|X,Y)$$

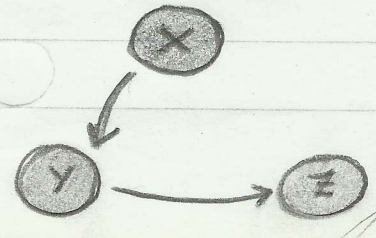
↳ primjetimo da je ovaj graf samo zapis rastavljanja distribucije $P(X,Y,Z)$

⇒ <u>pogledajmo pravilo lanca</u>: $P(X_1, \dots, X_M) = \prod_{k=1}^{M} P(X_k | \overbrace{X_1, \dots, X_{k-1}}^{pa(X_k)})$

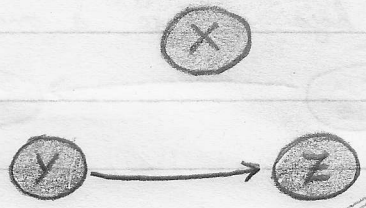
↳ $pa(X)$ su čvorovi roditelji čvora X

↳ primjetimo da je struktura računanja rastavljanja distribucije prvrše složeno (Bayesova mreža je potpuno povezani graf), pa uvodimo INDUKTIVNE PRETPOSTAVKE u obliku uvjetnih neovisnosti

⇒ ako na gornju mrežu pretpostavimo $X \perp Z | Y$, te znači da je X nebitan na član $P(Z|X,Y)$, te taj član postaje $P(Z|Y)$ što se manifestira kao graf:



⇒ ako pak pretpostavimo marginalnu nezavisnost $X \perp Y$, tada član $P(Y|X)$ postaje $P(Y)$, te $P(Z|X,Y)$ postaje $P(Z|Y)$ što daje sljedeći graf:



⇒ zaključujemo da ako fali bridova, postoji pretpostavljena nezavisnost

⇒ UREĐAJNO MARKOVLJEVO SVOJSTVO:

↳ svaki čvor ovisi samo o roditeljima

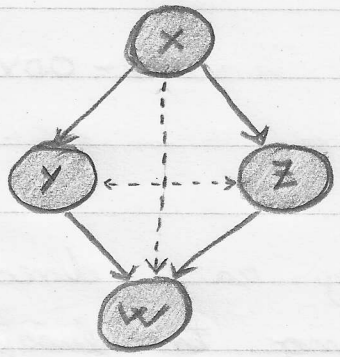
$$X_k \perp pred(X_k) \setminus pa(X_k) \mid pa(X_k)$$

$pred(X_k)$ ⇒ skup prethodnika

↳ ono nam definira uredaj i osigurava da je neki usmjeren graf stvarno Bayesova mreža

⇒ skup prethodnika $pred(X)$ je definiran u odnosu na topološke uredaje

Pr: Pogledaj sljedeći graf:



$$P(X,Y,Z,W) = P(X)\,P(Y|X)\,P(Z|X)\,P(W|Y,Z)$$

↳ topološki uredaj:

a) X, Y, Z, W b) X, Z, Y, W

↳ iz slike vidimo: $Y \perp Z \mid X$; $W \perp X \mid Y,Z$

↳ to bi dobili da primjenimo uredaju Markovljevo svojstvo na topološki uredaj „a)":

1) $Y \perp X \setminus \{X\} \mid X \Rightarrow Y \perp \emptyset \mid X$ (nema nezavisnosti)

2) $Z \perp \{X,Y\} \setminus \{X\} \mid X \Rightarrow Z \perp Y \mid X \to Y \perp Z \mid X$

3) $W \perp \{X,Y,Z\} \setminus \{Y,Z\} \mid Y,Z \to W \perp X \mid Y,Z$

↳ primjenom na topološki uredaj „b)" dobivamo isto!

↳ sada možemo koristeći svojstva 2) ; 3) pokušati dobit $P(X,Y,Z,W)$ iz desne strane početnog izraza kako bi pokazali da napisane nezavisnosti stvarno vrijede:

$$P(X)\,P(Y|X)\,P(Z|X)\,P(W|Y,Z) = P(X,Y)\,P(Z|X)\,P(W|Y,Z) = (*)$$

iz 2) raključujemo:
$P(Z|Y,X) = P(Z|X)$

$$(*) = P(X,Y)\,P(Z|Y,X)\,P(W|Y,Z) = (**)$$

iz 3) raključujemo:
$P(W|Y,Z) = P(W|X,Y,Z)$

$$(**) = P(X,Y,Z)\cdot P(W|X,Y,Z) = P(X,Y,Z,W)$$

D - ODVOJENOST

⇒ „D" stoji za „directed"

⇒ <u>definiramo tri slučaja</u>

① $p(x, y, z) = p(x \mid z) \cdot p(y \mid z) \cdot p(z)$

↳ ovo je račvanje:



② $p(x, y, z) = p(x)\ p(z \mid x)\ p(y \mid z)$

↳ ovo je lanac:



③ $p(x, y, z) = p(x) \cdot p(y) \cdot p(z \mid x, y)$

↳ ovo je srar:



⇒ pretpostavka se uredenog Markovljevog svojstva:

$$X \perp pred(x) \setminus pa(x) \mid pa(x)$$

⇓

ovo primjenjujemo
na gornja 3 slučaja
da bi dobili 3 pravila

(pred ⇒ „predecessor" ; pa ⇒ „parent")

⇒ PRAVILA:

① $Y \perp X \mid Z \iff X \perp Y \mid Z$

- račvanje je odvojeno akko je Z opaženo

② $Y \perp X \mid Z \iff X \perp Y \mid Z$

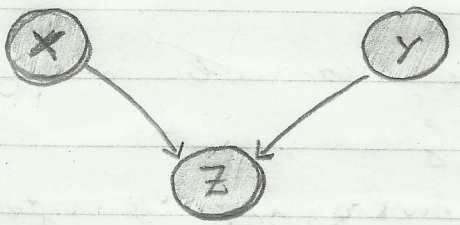
- lanac je odvojen akko je Z opaženo

③ $Y \perp X \mid \emptyset \iff X \perp Y$

- sraz je odvojen akko je Z neopažena

⇒ ovo treće pravilo (sraz) je malo neintuitivno, pa ga objasnimo:

- varijable X i Y se natječu da objasne Z
- no, čim vidimo Z, varijable postoju zavisne jer znamo rezultat i jednu varijablu



↓

ovo se zove

EFEKT OBJAŠNJENJA

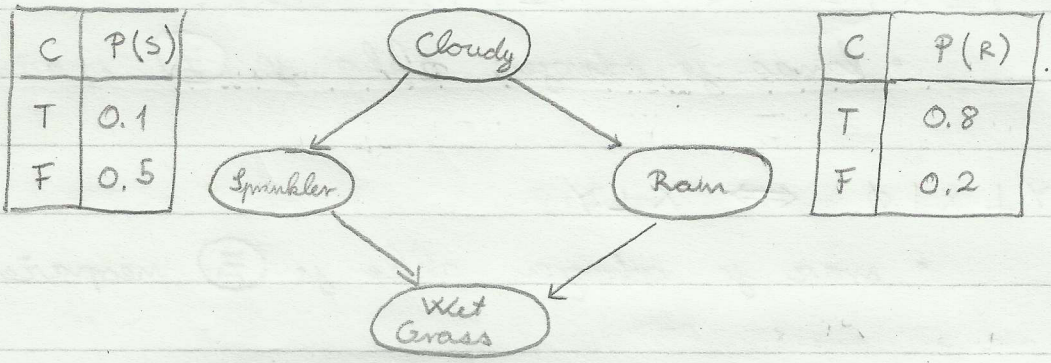
(„Explaining Away")

$$p(X \mid Z) \neq p(X \mid Y, Z)$$

PR: Bacamo dva novčića te ako znamo zbroj Z, jedan novčić (npr. X), onda znamo i drugi novčić (npr. Y) te varijable postoju zavisne!

ZAKLJUČIVANJE POMOĆU PROBABILISTIČKIH GRAFIČKIH MODELA

$P(C) = 0.5$



C	P(S)
T	0.1
F	0.5

C	P(R)
T	0.8
F	0.2

S	R	P(W)
T	T	0.99
T	F	0.9
F	T	0.9
F	F	0

⇒ na gornju mrežu postavljamo upit:

- ako je trava mokra, koja je vjerojatnost da je uzrok kiša, a koja da je uzrok prskanje?

$$P(S=1 \mid W=1) = \frac{P(S=1, W=1)}{P(W=1)} = \frac{\sum_{C,T} P(C, S=1, T, W)}{\sum_{C,T,S} P(C, S, T, W=1)} = (*)$$

$P(W=1)$ → vjerojatnost dokaza

$(*)$ ↓

$$P(c,s,r,w) = p(c) \cdot p(s|c) \cdot p(r|c) \cdot p(w|s,r)$$

$$(*) = \frac{0.2781}{0.6471} = 0.43$$

$$P(r=1 \mid w=1) = \frac{P(r=1, w=1)}{P(w=1)} = \frac{\sum_{s,r} P(c,s,r=1,w=1)}{0.6471} = 0.708$$

⇒ **tipovi varijabli:**

$\vec{X}_o$ ⇒ opažene varijable („observed")

$\vec{X}_q$ ⇒ varijable upita („query")

$\vec{X}_m$ ⇒ varijable smetnje („nuisance")

⇒ **tipovi upita:**

a) POSTERIORNI UPITI:

$$P(\vec{X}_q \mid \vec{X}_o) = \frac{\sum_{\vec{X}_m} p(\vec{X}_q, \vec{X}_o, \vec{X}_m)}{\sum_{\vec{X}_m', \vec{X}_q'} p(\vec{X}_q', \vec{X}_o, \vec{X}_m')}$$

↳ ovo smo radili u prvom delu!

b) MAP - UPITI:

$$\vec{X}_q^* = \arg\max_{\vec{X}_q} P(\vec{X}_q \mid \vec{X}_o)$$

⇓

MPE - „Most probable explanation"

↳ ovakav način rakljućivanja funkcionira samo na akademskim primjerima jer se događa eksplozija varijabli (trebamo bolje načine rakljućivanja)

⇒ ZAKLJUČIVANJE UZ ELIMINACIJU VARIJABLI:

$$P(w) = \sum_c \sum_s \sum_r p(c, s, r, w) =$$

$$= \sum_c \sum_s \sum_r p(c) \cdot p(s|c) \cdot p(r|c) \cdot p(w|c, r) =$$

$$= \sum_i p(c) \sum_s p(s|c) \sum_r p(r|c)\, p(w|s, r)$$

$t_1(c, s, w)$

$t_2(c, w)$

KONAČNO!

↳ ideja je marginalizirat varijable jednu po jednu

↳ rakljućivanje eliminacije varijabli kod skrivenih Markovljevih modela naziva se VITERBIJEV ALGORITAM (samo trebaš znat da postoji)

↳ rakljućivanje uz eliminaciju varijabli je egzaktan postupak jer daje točne rezultate

↳ ako ne trebamo točne rezultate, za ubrzanje možemo koristiti aproksimativne postupke:

a) slučajno rakljućivanje

b) „sampling - based"

⇒ ZAKLJUČIVANJE UZ METODE UZORKOVANJA

↳ aproksimativni postupci

↳ uči na raznim pornovanjo

↳ IDEJA: uzmi uzorke iz distribucije i pomoću njih računaj vjerojatnosti

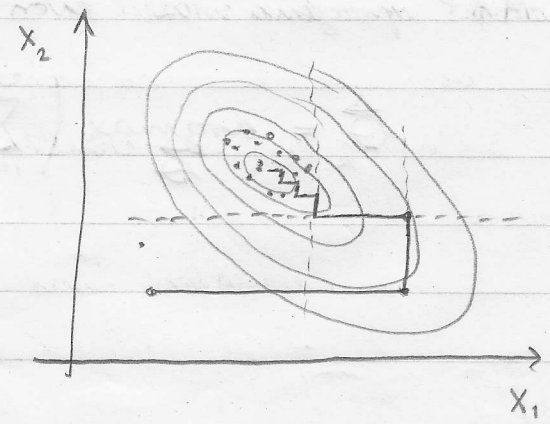
$$\vec{X} \sim P(\vec{X})$$

$$P(\vec{X}=x) = \frac{1}{N}\sum_{i=1}^{N} 1\{\vec{X}=x\}$$

$$\mathbb{E}[f] = \frac{1}{N}\sum_{i=1}^{N} f(\vec{x})$$

↳ načini uzorkovanja:

- Forward sampling (vidi primjer online!)
  - ↳ problem je što nas zanima $P(X_2 | X_0)$, a to ne možemo ovim načinom
- Rejection sampling - kao Forward samo što odbacujemo one koji nam ne trebaju
- Likelihood weighting - „importance sampling"
- MCMC - „Markov Chain Monte Carlo"
  - to se najčešće koristi!
  - ovo se koristi u GIBBS SAMPLING



⇒ UČENJE BAYESOVE MREŽE:

↳ model je probabilistički, pokorava se nekoj distribuciji koja ima svoje parametre, pa model učimo upravo procjenom parametara

↳ Log - vjerodostojnost za Bayesovu mrežu:

$$\ln \mathcal{L}(\vec{\Theta} \mid D) = \ln P(D \mid \vec{\Theta}) = \ln \prod_{i=1}^{N} P(\vec{X}^i \mid \vec{\Theta}) = (*)$$

(iid)

faktorizacija $\vec{X}^i$ prema Bayesovoj mreži

$$(*) = \ln \prod_{i=1}^{N} \prod_{k=1}^{K} P(X_k \mid pa(X_k), \vec{\Theta}) =$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} \ln P(X_k \mid pa(X_k), \vec{\Theta}) =$$

$$= \sum_{k=1}^{K} \sum_{i=1}^{N} \ln P(X_k \mid pa(X_k), \vec{\Theta})$$

- stoga, MLE procjena za čvor „k":

$$\hat{\vec{\Theta}}_k = \arg\max_{\vec{\Theta}} \left(\sum_{i=1}^{N} \ln P(X_k \mid pa(X_k)) \right)$$

- MAP procjena za čvor „k"

$$\hat{\vec{\Theta}}_k = \arg\max_{\vec{\Theta}} \left(\sum_{i=1}^{N} \ln P(X_k \mid pa(X_k)) + \ln p(\vec{\Theta}) \right)$$

⇒ NAPOMENA:

- izračunato je na predovanjima MAP procjena za čvor „Wet grass", pa to pogledaj na slajdovima ili prepiši od nekoga!

NENADZIRANO UČENJE

⇒ sada krećemo na **nenadzirano učenje** - nemamo klase za primjere za učenje

⇒ NEOZNAČENI SKUP PRIMJERA:

$$D = \{\vec{x}^i\} \qquad (\text{„unlabeled dataset"})$$

⇒ neoznačeni skup primjera imamo kada klase ne znamo ili ih možemo saznati, ali je previše posla

↳ npr. grupiranje klijenata prema ponašanju, grupiranje članaka prema temama, grupiranje gena sa sličnom funkcionalnošću, klasifikacija objekata na fotografiji, otkrivanje čudnog ponašanja korisnika na mreži („intrusion detection")

⇒ ZADACI NENADZIRANOG STROJNOG UČENJA:

a) grupiranje („clustering")
b) procjena gustoće („density estimation")
c) „outlier / novelty detection"
d) smanjenje dimenzionalnosti

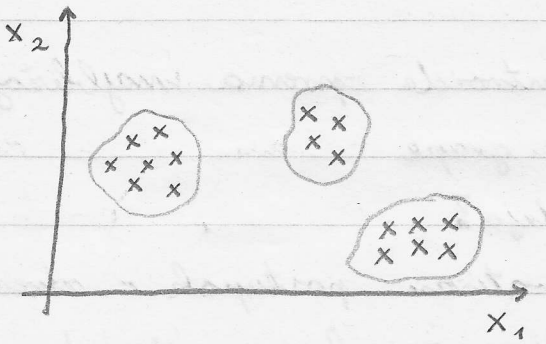
⇒ postoji i **polunadzirano** strojno učenje koje možemo koristiti kada imamo nešto označenih primjera i nešto (obično većina) neoznačenih primjera, a želimo kvalitetno iskoristiti označene primjere

↳ npr. imamo tri grupe s malo oznaka i samo propagiramo većinske oznake:



GRUPIRANJE

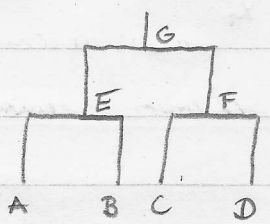
⇒ razdjeljivanje primjera u grupe (engl. „clusters") primjera, tako da su slični primjeri svrstani u istu grupu



⇒ VRSTE GRUPIRANJA:

↳ <u>prva podjela</u> (razv il stablast rezultat):

- PARTICIJSKO - spaja uzorke u grupe/particije
  - osnovna vrsta grupiranja
- HIJERARHIJSKO - spaja particije i tako daje stablo kao rezultat



npr. ovo je rezultat hijerarhije grupa

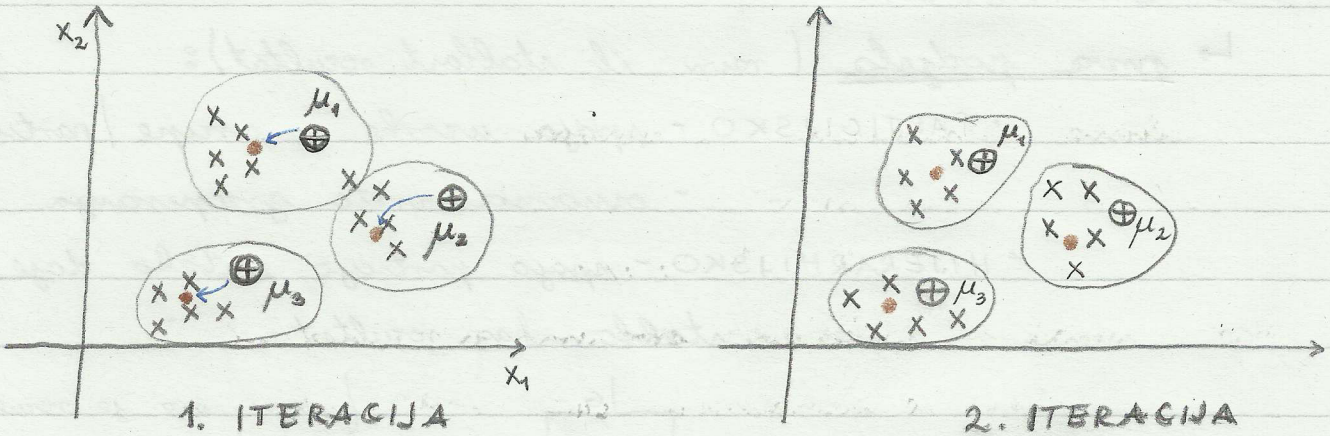
↳ <u>druga podjela</u> (može l primjer biti član jedne il više klasa):

- ČVRSTO - primjer je član jedne klase
  - engl. „hard"
- MEKO - primjer je član više klasa (il samo jedne)
  - engl. „soft"

ALGORITAM K-SREDINA

⇒ želimo grupirati u K grupa
↳ K unesek trebamo dati algoritmu

⇒ algoritam pomiče centroide prema najbližoj grupi, tj. prema sredini najbliže grupe
↳ to je ideja!
↳ ovo je iterativan postupak - odredi koji primjeri su najbliži i ratim pomakni centroid prema središtu tih najbližih primjera



⇒ dva ključna pitanja:
- kako odabrati K?
- kako odabrati centroide?

} dok ćemo do ovoga!

⇒ pogledajmo pogrešku koju oraj algoritam minimizira:

$$J = \sum_{k=1}^{K} \sum_{i=1}^{N} b_k^i \cdot \| \vec{x}^i - \vec{\mu}_k \|^2$$

$$b_k^i = \begin{cases} 1, & \text{ako je } x^i \text{ grupiran u grupu } K \\ 0, & \text{inače} \end{cases}$$

↳ pogreška „J" će biti tim veća što je primjer dalje od svog centroida

↳ stoga, za minimizaciju pogreške primjer svrstavamo najbližem centroidu, pa imamo novu definiciju za „b":

$$b_k^i = \begin{cases} 1, & \text{ako je } k = \arg\min_j \| \vec{x}^i - \vec{\mu}_j \|^2 \\ 0, & \text{inače} \end{cases}$$

↳ želimo minimizirat pogrešku:

$$\underset{b_1, b_2, \dots, b_K, \mu_1, \mu_2, \dots, \mu_K}{\arg\min} \; J$$

↳ ovo nejde analitički jer imamo ovisnost između „b" i „μ", pa se izvlačimo tako da jedno od njih fiksiramo, a po drugom deriviramo:

$$\nabla_{\vec{\mu}_k} J = 0 \quad \Rightarrow \quad 2 \sum_{i=1}^{N} b_k^i \left(\vec{x}^i - \vec{\mu}_k \right) = 0$$

$$\vdots$$

$$\vec{\mu}_k = \frac{\sum_{i=1}^{N} b_k^i \vec{x}}{\sum_{i=1}^{N} b_k^i}$$

⇒ ovo je korak za izračunavanje centroida, pa smo izveli algoritam

⇒ složenost ovog algoritma je linearna po sva 4 parametra:

$$O(T \cdot m \cdot N \cdot K)$$

↳ „T" je broj iteracija

⇒ zapitajmo se sada **konvergira li ovaj algoritam?**

↳ pa, da!

↳ prvi način da ovo zaključimo:

- algoritam pretražuje prostor stanja veličine K^N (velik, ali konačan) - to je broj particija jer prvi primjer možemo svrstati u K grupa, drugi u K grupa i tako N primjera
- kako je ovo konačno, algoritam to pretražuje, pa on konvergira

↳ drugi način da ovo zaključimo:

- smanjuje se funkcija pogreške

⇒ zapitajmo se sada **nalazi li algoritam optimalno rješenje?**

↳ **ovisi o odabiru početnih centroida tako da je odgovor: „ne, u općenitosti"**

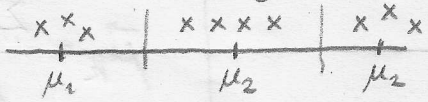
↳ <u>**tehnike odabira centroida:**</u>

a) nasumičan odabir primjera

↳ loše je to da možda odaberem „outliera"

b) centroid skupa podataka + slučajan vektor

c) K-means++ algoritam



$$P(\vec{\mu}_{K+1} = \vec{x}^i \mid \vec{\mu}_1, \ldots, \vec{\mu}_K) = \frac{\min_k \|\vec{\mu}_k - \vec{x}^i\|}{\sum_j \min_k \|\mu_k - \vec{x}^j\|}$$

ALGORITAM K-MEDIOIDA

⇒ problem s algoritmom K-sredina je u tome što on radi samo sa vektorskim prostorima jer on treba moći izračunati euklidsku udaljenost prostora do centroida

⇒ ovaj algoritam je jako sličan, al koristi neku drugu mjeru sličnosti

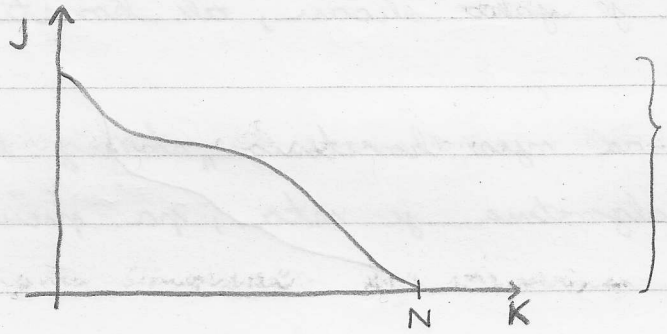
↳ npr. za riječi koristimo „string kernel"

↳ ideja algoritma je ista, pa pseudokod vrlo sličan slajdovima

PROVJERA GRUPIRANJA

⇒ trebamo mehanizam ocjene dobrote / kvalitete grupiranja

⇒ npr. za algoritam K-sredina, želimo odabrat optimalan K, a to je onaj koji minimizira J:



funkcija monotono pada

↑
svaki podatak ima svoj centroid i grupiran je u svoju grupu

↳ to ne želimo jer je to prenaučenost!

↳ stoga, stroktni izbor najboljeg K za minimalnu greške nije dobar način, pa trebamo pametnije metode

⇒ **bolji pristup** je npr. **metoda lakta**:

↳ gornja funkcija ima platoe jer kad počnemo cijepat grupe na manje grupe, počnemo dijelit grupe na manje prirodan način što daje manji doprinos smanjenju funkcije pogreške

VIZUALNA METODA (nije egzaktna)

⇒ drugi pristup je preslikavanje u 2D prostor nekom metodom redukcije dimenzionalnosti (npr. PCA ili raspoređivanja uzoraka)
VIZUALNA METODA

⇒ treći pristup je analiza silueta:

↳ računamo siluete:

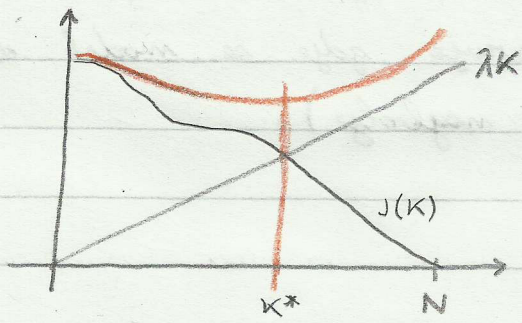
$$\vec{x}^i \rightarrow s(i) = \frac{b(i) - a(i)}{\max(b(i), a(i))}$$

↳ ako je neka od silueta grupa ispod prosjeka, tu podjelu nećemo uzeti

↳ dodatno, želimo da su siluete što ravnopravnije jer onda svaka grupa ima otprilike isti broj primjera

⇒ četvrti pristup je minimizacija regularizirane funkcije pogreške:

$$K^* = \arg\min_K \left(J(K) + \lambda K \right)$$



za λ obično uzimamo 2m što je dokazano Akhaike information centrom

⇒ peti pristup je provjera točnosti na označenom podskupu primjera:

↳ ideja je ispitati točnost na onim primjerima koje smo uspjeli označit

↳ tipične mjere točnosti: NMI, Randov Index

↳ Randov Index: $R = \frac{a+b}{\binom{N}{2}}$

što su ovi brojevi veći, to je grupiranje točnije

a → broj jednako označenih parova koji su u istoj grupi

b → broj različito označenih parova koji su u različitoj grupi

PR: Recimo da grupiramo u dvije grupe ($K=2$) te je grupiranje dano. Odredi Random index!

$$\{\{1,1,1,1,2\},\{1,1,2,2,2,2,2,2\}\}$$

$N = 13$

$\binom{N}{2}$ ⇒ broj parova

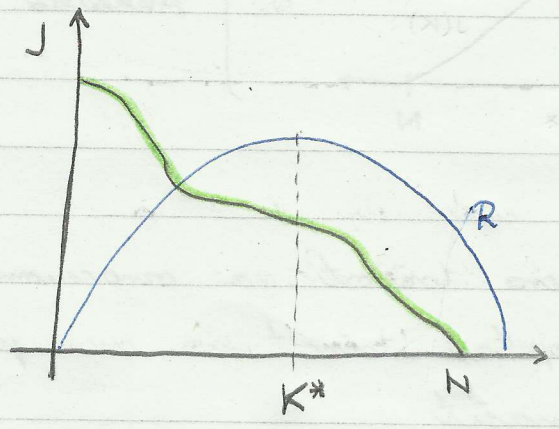
$$a = \binom{4}{2} + \binom{1}{2} + \binom{2}{2} + \binom{6}{2} = 22$$

$$b = 4 \cdot 6 + 1 \cdot 2 = 26$$

$$R = \frac{a+b}{\binom{N}{2}}$$

$$R = 0.62$$

⇒ Random index dobro pokazuje gdje se model dobro ponaša (tamo gdje se grupiranje najbolje):



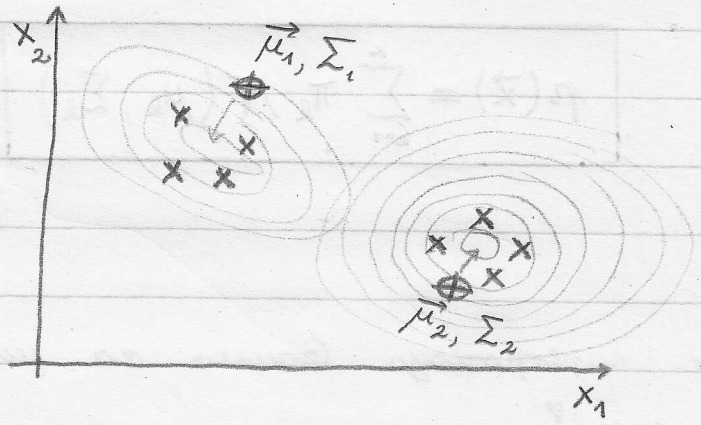
MODEL GAUSSOVE MJEŠAVINE

⇒ ponekad ne želimo čvrsto grupiranje, nego neki primjeri bi mako spadali u jednu, a mako u drugu grupu

⇒ dodatno, korisno nam je meko grupiranje kada ne znamo broj grupa, pa svoju nesigurnost modeliramo tako da primjer pripada u više grupa

⇒ engleski naziv je „Gauss Mixture Model" (GMM)

↳ ideja je **namjestiti** multivarijantnog Gaussa na svaku grupu



↳ pretpostavimo da sada ne K centroida imamo i K kovarijacijskih matrica, pa je ovaj model složeniji od modela K-centroida

① **MODEL**:

$$p(\vec{x}) = \sum_{k=1}^{K} \pi_k \, p(\vec{x} \mid \Theta_k)$$

$\pi_k \Rightarrow$ komponenta mješavine $\left(\sum_{k=1}^{K} \pi_k = 1 \right)$

$p(\vec{x} \mid \Theta) \Rightarrow$ gustoća komponente

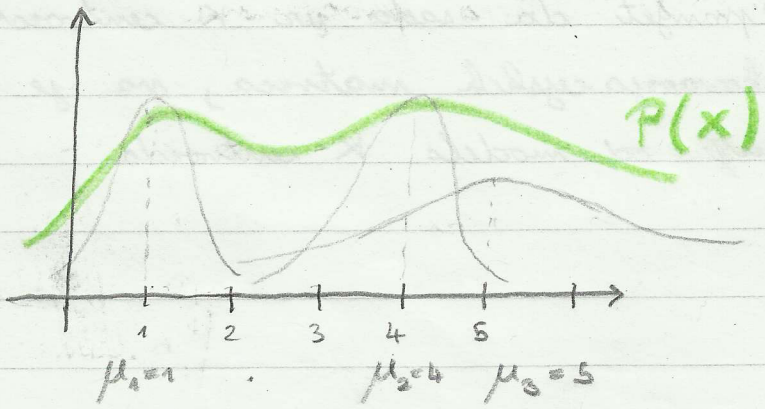
$\pi_k = P(G_k)$

$$p(\vec{x}) = \sum_{k=1}^{K} P(G_k) \, p(\vec{x} \mid G_k)$$

↳ ovo smo napisali u općenitosti za neko grupiranje, a za Gaussovu mješavinu vrijedi normalna razdioba:

$$\boxed{p(\vec{x}) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mu_k, \Sigma_k)}$$

PR: Nacrtajmo superpoziciju Gaussa za jednu dimenziju i neko grupiranje 6



$\sigma_1 = \sigma_2 = 1 \qquad \sigma_3 = 6$

⇒ nas zanima u praksi vjerojatnost da neki primjer pripada nekoj grupi:

$$h_k = P(G_k | \vec{x}^i) = \frac{P(\vec{x}^i, G_k)}{P(\vec{x}^i)} = \frac{P(G_k) \cdot P(\vec{x}^i | G_k)}{\sum_j P(G_j) P(\vec{x}^i | G_j)}$$

↳ zapis u notaciji sa „π":

$$\boxed{h_k = \frac{\pi_k P(\vec{x}^i | G_k)}{\sum_j \pi_j P(\vec{x}^i | G_j)}}$$

h_k ⇒ ODGOVORNOST

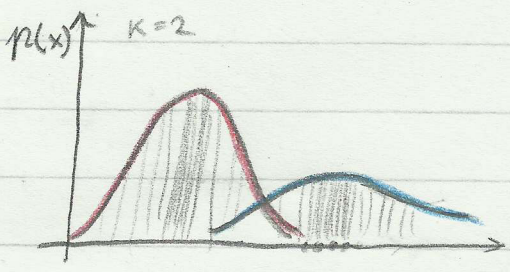
(vjerojatnost da primjer $\vec{x}^i$ dolazi iz grupe / komponente G_k)

⇒ PARAMETRI MODELA:

$$\vec{\Theta} = \left\{ \pi_k, \vec{\mu}_k, \Sigma_k \right\}_{k=1}^{K}$$

↳ model učimo procjenom parametara MLE procjeniteljem:

$$D = \left\{ \vec{x}^i \right\}_{i=1}^{N}$$



↳ kako bi procijenili parametre MLE : procjeniteljem nalazimo log-izglednost i pokušavamo ga maksimizirati:

$$\ln \mathcal{L}(\theta|D) = \ln P(D|\theta) \overset{\text{IID}}{=} \ln \prod_{i=1}^{N} p(\vec{x}^i|\theta) =$$

$$= \ln \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k p(\vec{x}^i|\theta_k) =$$

$$= \sum_{i=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k p(\vec{x}^i|\theta_k) \right)$$

$\Downarrow$

nemoguće rješenje u zatvorenoj formi:

↳ stoga, trebamo <u>drugi pristup</u>:

- GRADIJENTNI SPUST
- ALGORITAM MAKSIMIZACIJE OČEKIVANJA

ALGORITAM MAKSIMIZACIJE OČEKIVANJA

⇒ <u>uvodimo latentne varijable</u>:

- svaki primjer ima jednu varijablu (vektor):

$$\vec{z}^i = (z_1^i, z_2^i, \dots, z_k^i)$$

↳ „one-hot-coding":

$$(0, 0, 1, \dots, 0)$$
$$\uparrow_{k=3}$$

(mnogo je generiran od samo jedne komponente / grupe, ali mi ne znamo to - i zato znamo varijablu $\vec{z}^i$ koja to modelira)

$$P(\vec{z}^i = k) = \prod_{k=1}^{K} \pi_k^{z_k^i}$$

↓

pomoću ovoga dalje pišemo model:

$$P(\vec{x}, z \mid \theta) = P(z) \cdot P(\vec{x} \mid z, \theta)$$

$$= \prod_{k=1}^{K} \pi_k^{z_k} \prod_{k=1}^{K} P(\vec{x} \mid \theta_k)^{z_k}$$

$$= \prod_{k=1}^{K} \pi_k^{z_k} \cdot p(\vec{x} \mid \theta)^{z_k}$$

↳ sada ovo pokušajmo procesirat MLE procesirteljem (očekujemo da će ići u zatvorenu formu):

OKRENI

$$\ln \mathcal{L}(\Theta \mid D, Z) = \ln P(D, Z \mid \Theta) = \ln \prod_{i=1}^{N} P(\vec{x}^i, z^i \mid \Theta) =$$

$$= \ln \prod_{i=1}^{N} \prod_{k=1}^{K} \pi_k^{z_k^i} \cdot p(\vec{x}^i \mid \Theta_k)^{z_k^i}$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} z_k^i \left(\ln \pi_k + \ln p(\vec{x}^i \mid \Theta_k) \right)$$

↑ slučajna varijabla koja je nepoznata

↳ ovaj izvod ne treba baš znati, ali treba znati neku ideju:

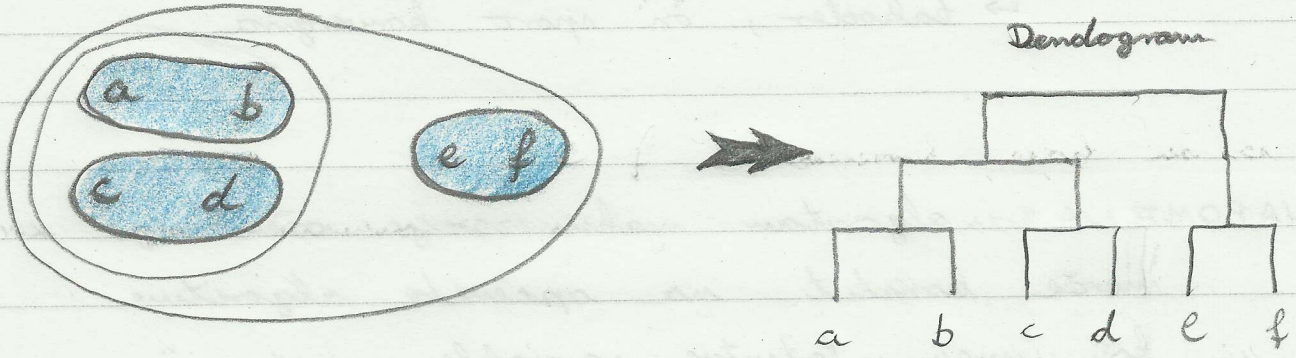
- ovo se odvija u dva koraka:
  1) E - „step" (korak)
  2) M - maksimizacija
- prvo inicijaliziramo sve parametre: π_k, $\vec{\mu}_k$, Σ_k
- radimo E korak:
  ↳ računamo vjerojatnost da primjer pripada u svaku od K grupa / komponenti (h_k)
- radimo M korak:
  ↳ računamo nove $\vec{\mu}_k$ i Σ_k za svaku komponentu na temelju h_k
- na temelju izračunatoga, računamo log-izglednost i ponavljamo se po njoj
- ponavljamo do konvergencije

⇒ NAPOMENA: algoritam GMM ne mora nužno pronaći globalni optimum

↳ također, on sporo konvergira

⇒ NAPOMENA: algoritam maksimizacije očekivanja se može koristiti na općenite algoritme koji imaju latentne varijable

HIJERARHIJSKO GRUPIRANJE



⇒ tipovi gradenja dendogramu / algoritama:

1) aglomerativno - odozdo prema gore
   - grupiraj manje pa sve vece
   ↳ spada algoritam HAC
2) divizivno - odozgo prema dolje
   - vrsi sve pa postupno dijeli

⇒ trebamo udaljenost dva primjera, pa uvedimo neke mjere udaljenosti: $d(x, x')$

a) udaljenost centroida:

$$d_{cent}(G_i, G_j) = \left\| \frac{1}{N_i} \sum_{\vec{x} \in G_i} \vec{x} - \frac{1}{N_j} \sum_{\vec{x} \in G_j} \vec{x} \right\|$$

b) jednostruka povezanost: „single linkage"

$$d_{min}(G_i, G_j) = \min_{\vec{x}^i \in G_i \; \vec{x}^j \in G_j} d(\vec{x}^i, \vec{x}^j)$$

c) potpuna povezanost: „complete linkage"

$$d_{max}(G_i, G_j) = \max_{\vec{x}^i \in G_i, \vec{x}^j \in G_j} d(\vec{x}^i, \vec{x}^j)$$

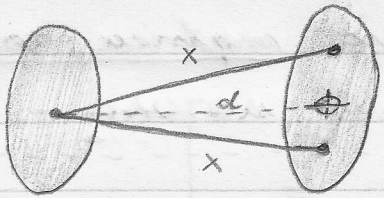
⇒ ovo su dvije krajnosti!

⇒ maksimum daje sklone grupe, a minimum razvučene

d) osrednja povezanost: „average linkage"

$$d_{avg}(G_i, G_j) = \frac{1}{N_i N_j} \sum_{\vec{x}^i \in G_i} \sum_{\vec{x}^j \in G_j} d(\vec{x}^i, \vec{x}^j)$$

⇒ NAPOMENA: centroid i osrednja povezanost nisu isti slučajevi:



VREDNOVANJE MODELA

⇒ mnogo kompliciranije i značajnije nego što se na prvi pogled čini

↳ u ovom dijelu strojnog učenja se radi puno pogrešaka

⇒ OSNOVNE MJERE VREDNOVANJA:

↳ pogledajmo binarni klasifikacijski primjer:

$K = 2 \quad N = 150$

↳ napišimo <u>matricu zabune</u>:

		ACTUAL +	ACTUAL −
PREDICTION	+	TP	FP
	−	FN	TN

↳ recimo da su za naš primjer podaci ispali oraku u matrici zabune:

	+	−
+	6	12
−	2	130

① TOČNOST ⇒ engl. „accuracy"

⇒ problem je da se lagano može izigrati ako su primjeri neuravnoteženi (puno pozitivnih, a malo negativnih primjera i obratno)

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

(za naš primjer: $ACC = \frac{136}{160} = 90.7\%$)

⇒ još i vrijedi: $ACC = 1 - E(h|D)$

② PRECIZNOST ⇒ engl. „precision"

$$P = \frac{TP}{TP + FP}$$

⇒ razlikuje od točnosti!

③ ODZIV ⇒ engl. „recall", „sensitivity"

$$R = \frac{TP}{TP + FN}$$

④ FALL-OUT

$$FPR = \frac{FP}{FP + TN}$$

(„false-positive rate")

⑤ SPECIFIČNOST

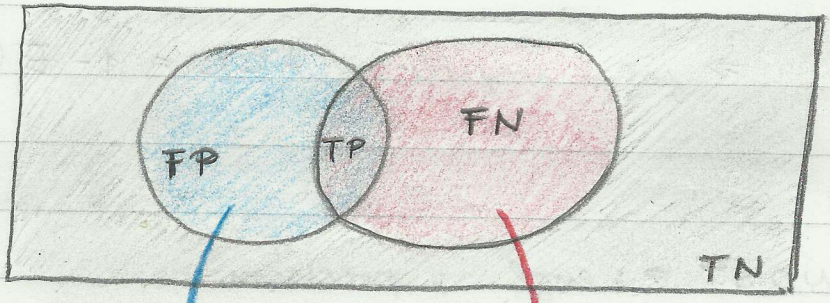
$$S = \frac{TN}{FN + TN}$$

⇒ obično se koristi točnost, jedina ako primjeri nisu dobro balansirani

↳ najbolje je koristiti onu mjeru koja se ustaljena u zajednici područja na kojem primjenjujemo strojno učenje

⇒ mi ćemo se skoncentrirati na **preciznost** i **odziv**:

↳ daju realističniju procjenu „točnost"



$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN}$$

↳ u ekstremnom disbalansu možemo i ovdje imati problem (npr. dobijemo u nazivniku nulu)

ZAD: Od N=1000 primjera 100 je pozitivno. Klasifikator ispravno klasificira 50 pozitivnih i 850 negativnih. Izračunaj točnost, preciznost i odziv.

PREDICTED \ ACTUAL	+	-
+	50	50
-	50	850

↓ 100

P = 50%

R = 50%

ACC = 90%

ZAD: Za zadanu matricu zabune odredi točnost, preciznost i odziv.

PREDICTED \ ACTUAL	+	-
+	90	250
-	10	650

P = 0.265 = 26.5%

R = 0.9 = 90%

ACC = 0.74 = 74%

↳ primjetit varijaku u P i R (preciznost i odziv) pa ta dva broja želimo ujediniti u jedan broj (uvodimo harmonijsku sredinu):

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2 \cdot P \cdot R}{P + R} = 0.458$$

⇒ HARMONIJSKA SREDINA:

↳ služi za objedinjavanje preciznosti i odziva u jedan broj

↳ nju koristimo jer je strožija od aritmetičke i geometrijske sredine što želimo jer želimo pesimističniju mjeru dobrote klasifikatora

↳ stoga, ova mjera je negdje između preciznosti i odziva, ali je bliže lošijoj mjeri (lošija više utječe)

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{P + R}$$

⇒ poopćenje harmonijske sredine je β-mjera koja nam omogućava dodjeljivanje važnosti preciznosti ili odzivu - ovo što nam treba:

$$F_\beta = \frac{(1 + \beta^2) \cdot PR}{\beta^2 P + R}$$

$\beta = 0.5$ ⇒ odred što se preferira (preciznost il odziv)

$\beta = 2$ ⇒ odred što se preferira (preciznost il odziv)

↳ na detekciju spama preferiramo preciznost jer želimo smanjiti broj dobrih mailova koj su završil u spamu

↳ na detekciju raka, preferiramo obratno!

MJERENJE DOBROTE VIŠEKLASNE KLASIFIKACIJE

PR: Razlegajmo primjer s 3 klase sa 13 primjera!

PREDICTED \ ACTUAL	Y=1	Y=2	Y=3
Y=1	1 (TP_1)	1	0
Y=2	2	2	3
Y=3	0	0	4

$\Rightarrow FP_1$ (1 + 0)

FN_1 (2 + 0)

TN_1 (2 + 3 + 0 + 4)

} matrica zabune za više klasa

↳ **napomena:** sve izvan dijagonale je FP za jednu klasu; a FN za drugu

↳ naprazimo sada matricu zabune za svaku klasu:

- ruši, matricu K×K transformiramo u K matrica dimenzija 2×2:

Y=1	+	−
+	1	1
−	2	9

Y=2	+	−
+	2	5
−	1	5

Y=3	+	−
+	4	0
−	3	6

- sada iz ovih matrica možemo računati sve razne izvedene mjere (točnost, preciznost, ...) za svaku klasu zasebno
- kada imamo te mjere za svaku klasu zasebno, možemo ih kombinirati na više načina: „MAKRO" i „MIKRO" način

OKRENI

⇒ MACRO - NAČIN kombiniranja:

↳ izračunaj željenu veličinu za svaku klasu zasebno i zatim samo uprosječi tu veličinu kako bi dobio mjeru za višeklasni klasifikator

↳ brojevi za prošli primjer:

$ACC_1 = 0.77$ $P_1 = 0.5$ $R_1 = 0.33$ $F_{1,1} = 0.4$

$ACC_2 = 0.54$ $P_2 = 0.29$ $R_2 = 0.67$ $F_{1,2} = 0.4$

$ACC_3 = 0.77$ $P_3 = 1$ $R_3 = 0.55$ $F_{1,3} = 0.73$

$$ACC^M = \frac{1}{3}(0.77 + 0.54 + 0.77) = 0.69$$

$$P^M = 0.6 \quad R^M = 0.52 \quad F_1^M = 0.51$$

↳ ova mjera je dosta stroga jer ako klasifikator „kiksa" na jednoj klasi to jako puno prosjek, pa je ova mjera preporučena ☺

⇒ MICRO - NAČIN kombiniranja:

↳ zbroj matrica zabune svih klasa i onda izračunaj mjere koje te razumiju

↳ brojevi za prošli primjer:

	+	-
+	7	6
-	6	20

⇒ $Acc^\mu = 0.69$

$P^\mu = 0.54$ } $F_1^\mu = 0.54$

$R^\mu = 0.54$

↳ napomena: ova mjera je manje stroga $(F_1^\mu > F_1^M)$

↳ napomena: P^μ i R^μ su uvijek isti jer se zbroje FP i FN



VREDNOVANJE KLASIFIKATORA S PRAGOM

⇒ pogledajmo sada krivulje modela u **P-R ravnini**:

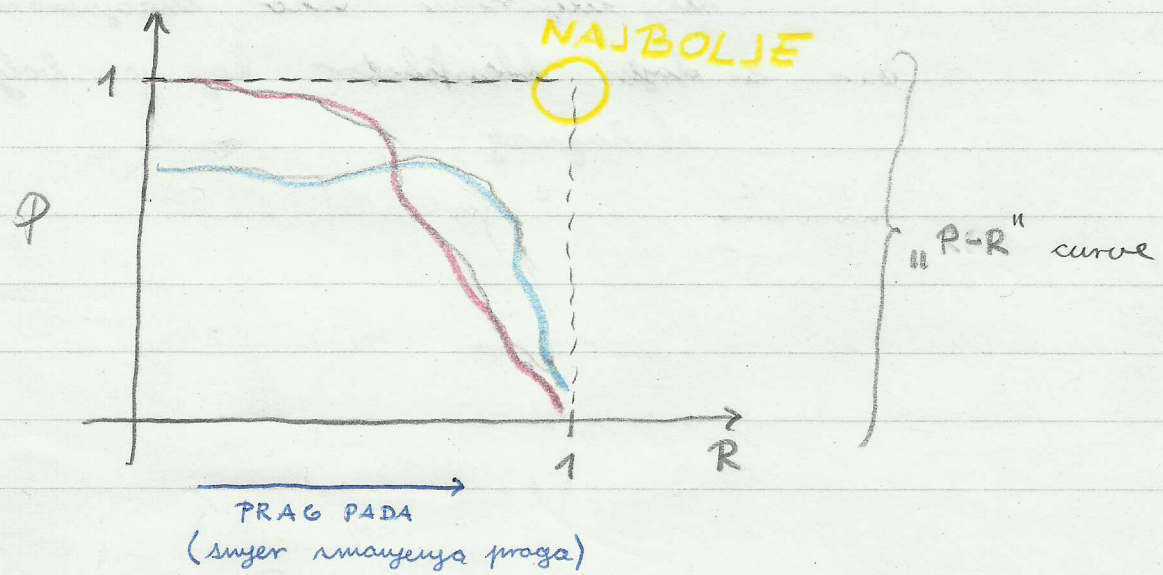
↳ ako je jedna krivulja stalno ispod druge, taj model je lošij i nema razloga da ga uzmemo

↳ najbolja krivulja / najbolji model:

- tamo gdje je integral (površina ispod krivulje) najbliži 1
- ta mjera površine se naziva „Average Precision" (AP)

↳ obično modeli imaju različite krivulje koje se sjeku, pa uzmemo onaj koji daje najbolju vrijednost (najveću vrijednost) na našu radnu točku

- radna točka je odredena našom preferencijom preciznosti i odziva

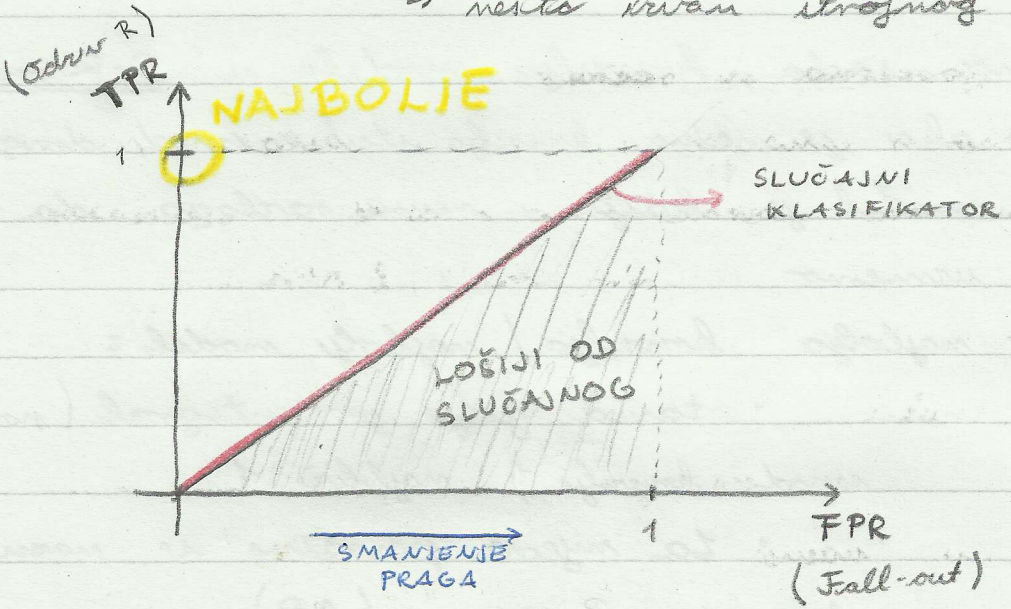


⇒ drugi način je AUC:

↳ „Area under ROC curve"

↳ ROC ⇒ „Receiver Operating Characteristics"

⇒ nešto kriva strojnog učenja



$$FPR = \frac{FP}{FP+TN}$$

↳ ako dobijemo klasifikator lošij od slučajnog, okrenemo njegove predikcije i dobivamo bolj od slučajnog jer tc rad simetriju u odnosu na pravac slučajnog klasifikatora

• čudno je ako se to desi jer su algoritmi tako dizajnirani da daju klasifikator koj je bolji od slučajnog

PROCJENA POGREŠKE MODELA

⇒ **nepoštena procjena pogreške:**

- ako računamo TP, FP i ostalo na istom skupu na kojem smo i učili klasifikator

TO NE RADITI !!! NIKAD !!!

⇒ **Holdout metoda:**

- podjelimo primjere na ispitni skup i na skup za učenje
- dobro je da ispitujemo pogrešku generalizacije, no imamo dva problema:
  1) smanjujemo skup za učenje
  2) da smo drugačije razdijelili primjere, možda bi dogodili puno drugačije mjere (nije statistički prihvatljivo)
- oba ova problema rješavamo unakrsnom provjerom (slučajno razdijeli primjere i ponovljaj Holdout više puta)

↳ ovo je „Repeated Holdout"

⇒ **k-struka unakrsna provjera:**

- „folded cross-validation"
- radi unakrsne provjere na „k"-različitih podjela skupa za učenje i ispitivanje
- ako je „k" velik, puno primjera se ponovlja (visoka varijanca)

⇒ **leave-one-out cross validation:**

- ekstreman primjer „k"-struke provjere gdje trenriramo na k-1 primjeru, a testiramo na jednom primjeru
- problem: ne možemo koristit ni jednu drugu mjeru osim točnosti jer imamo samo jedan zapitan primjer
  ↳ također, visoka varijanca rezultata testa na jednom primjeru (točnost varira)
- ova koristi kada imaš „srednji" broj primjera

⇒ k-struka unakrsna provjera je najbolje što imamo (u velikoj većini slučajeva)

⇒ teorijski mi želimo i pronaći najbolji model, i ocjenit ga, pa dijelimo skup na 3 dijela:

D_1 ⇒ skup za učenje
D_2 ⇒ skup za provjeru
D_3 ⇒ skup za testiranje

⇓

kako bi izbjegli fikaciju ova 3 skupa, želimo radit „k"-folding te dobivamo ugniježđeni „k"-folding

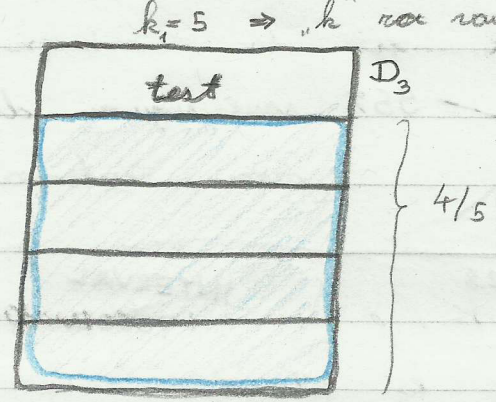
⇒ NESTED k-FOLDED : CROSS-VALIDATIONS

↳ najbolji način pronalaženja najboljih parametara (hipoteze)

↳ također, i najpoštenji način

↳ imamo dvije petlje

• vanjska petlja radi standardni postupak „k"-foldinga, i ona u jednoj iteraciji daje ovakav skup:



• unutarnja petlja uzima ove 4/5 podataka i njega dijeli na recimo 10 dijelova (jer je $k_2 = 10$, na primjer)

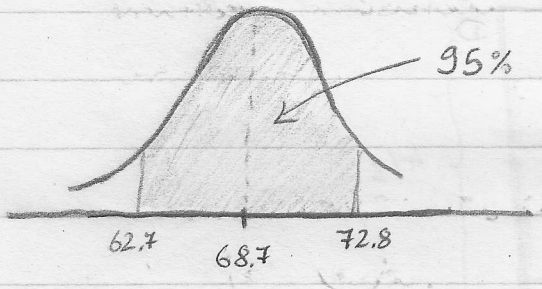
• tada na tih 10 dijelova traženimo najbolji model (uspoređuje se sa Grid Search Cross-Validacijom)

↳ ovo dobro naučit i skužit sve!

• skužit unutarnju „for" petlju!

STATISTIČKO ZAKLJUČIVANJE

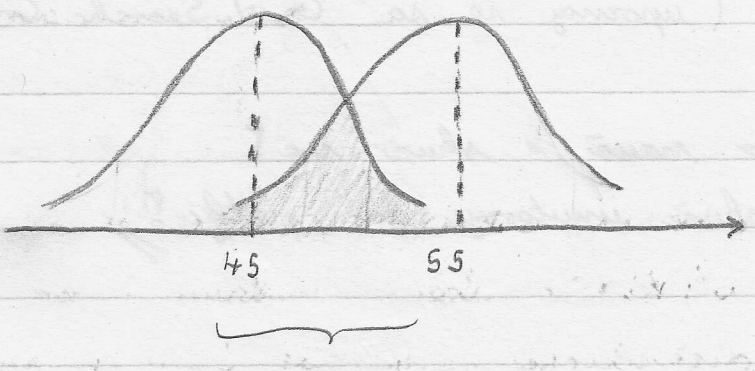
⇒ želimo reći kolika smo sigurni da je naša ocjena modela dobra jer nikada ne možemo biti sigurni u naše brojke jer je uzorak na kojem istraživanje slučajan



95% sam siguran da je točnost u ovom intervalu

INTERVAL POUZDANOSTI

↳ druga ideja / alat je STATISTIČKO TESTIRANJE HIPOTEZE (SHT) koj nam daje mogućnost da testiramo kolika je neka hipotera točna:



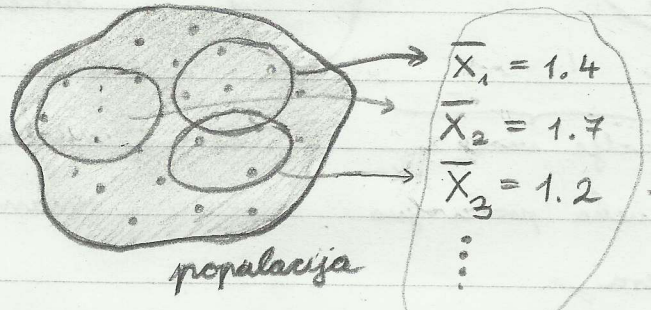
npr. 25% smo sigurni da su A i B iste stvari

⇒ TRI TEMELJNA POJMA:

1) **populacija** → svi promatrani
2) **uzorak** → ispitni skup (mora biti reprezentativan)
3) **statistika** → neka mjera koju mi ispitujemo
(npr. F_1, Acc, P, ...)

⇒ da bi mi mogli izračunati interval pouzdanosti i statističko testiranje hipoteze, trebamo znati distribuciju statistike

↳ to se još zove DISTRIBUCIJA UZORKOVANJA i ona govori kako bi se statistika (ta naša mjera) ponašala kada bi ponovljali uzorkovanje



statistika koju ovdje koristimo je srednja vrijednost uzetog uzorka

→ ravnamo nas distribucija ovih podataka (to je distribucija uzorkovanja)

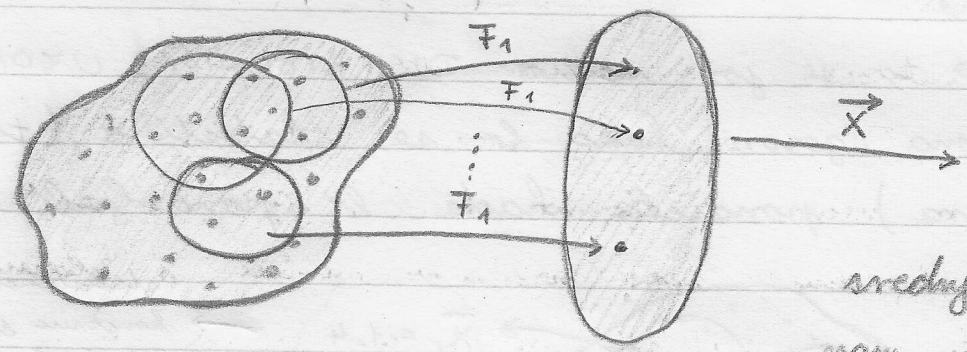
↳ DISTRIBUCIJA UZORKOVANJA ovisi o 4:

1. statistici koju koristimo (npr. srednja vrijednost)
2. distribuciji populacije (što uzorkujemo)
3. postupak uzorkovanja („with replacement" ili ne)
4. veličini uzorka N

⇒ no, mi trebamo vrat neku mjeru za koju znamo distribuciju (to je statistika), a nažalost za puno mjera to ne znamo

↳ na primjer, za F_1 to ne znamo

↳ no, možemo se truditi na način da iz populacije vučemo uzorke, računamo F_1 i onda nad tim vrijednostima mjere F_1 računamo srednju vrijednost jer znamo distribuciju srednje vrijednosti



srednja vrijednost nam je korisno jer njenu distribuciju znamo uvijek //

srednje može doći neka proizvoljna mjera //

⇒ BITNO SVOJSTVO:

↳ ako je statistika srednja vrijednost i populacija se ravna po normalnoj razdiobi $N(\mu, \sigma^2)$ i uzorci su veličine N, tada je distribucija srednje vrijednosti također normalna:

$$\bar{X} \sim N\left(\mu, \frac{\sigma^2}{N}\right)$$

⇒ normalno, srednja vrijednost je: $\bar{X} = \frac{F_1^1 + F_2^2 + \ldots + F_1^k}{K}$

⇒ standardna pogreška:

$$SE = \sqrt{\frac{\sigma^2}{N}}$$

⇒ e sad, problem je da se populacija mora ravnati po normalnoj varijabli što u stvarnosti ne znamo
↳ no, tu nas štiti CENTRALNI GRANIČNI TEOREM:

- „Ako je statistika srednja vrijednost i ako $N \to \infty$, onda je distribucija uzorkovanja također normalna $\mathcal{N}\left(\mu, \frac{\sigma^2}{N}\right)$"
- u praksi, ako je $N \geq 30$, ovo radi dovoljno dobro (no, 30 može biti teško dosegnut ako radimo nested „k"-folding jer je N jednak „k", a ra $k \geq 30$ ovo je jako sporo).

⇒ drugi problem je što ne znamo standardnu devijaciju populacije (ne znamo varijancu):

- tada koristimo studentovu „t"-distribuciju jer ona ima samo jedan parametar ⇒ **stupanj slobode**

$t(x, df)$ „df" je stupanj slobode

- za više od 30 primjera, studentova varijabla sjeda na normalnu i vrijedi da je stupanj slobode jednak varijanci normalne varijable ($df = N - 1$)

⇒ AKO IMAMO < 30 PRIMJERA, OVA PRIČA NE ŠTIMA JER NE ZNAMO DA LI JE POPULACIJA NORMALNA I NEMOJ NIKAD KORISTITI OVE PRETPOSTAVKE!

INTERVAL POUZDANOSTI

⇒ radimo ocjenu pouzdanosti:

$$\hat{\mu} = \bar{X} = \frac{1}{N} \sum_{i=1}^{N} F_i^i$$

$$\mu = \bar{X} \pm SE \quad (\text{sa } \langle \text{nekoliko} \rangle \% \text{ pouzdanosti})$$

$$\bar{X} \sim \mathcal{N}\left(\mu, \underbrace{\frac{\sigma^2}{N}}_{SE^2}\right)$$

↳ sada želimo standardizirati ovu distribuciju (središte u ishodištu i varijance jednake 1):

$$Z = \frac{\bar{X} - \mu}{SE} \sim \mathcal{N}(0, 1)$$

⇒ npr. recimo da želimo 95%-tni interval pouzdanosti:

$$P\left(-1.96 \leq \frac{\bar{X} - \mu}{SE} \leq 1.96\right) = 0.95$$

$$P\left(\bar{X} - 1.96\,SE \leq \mu \leq \bar{X} + 1.96\,SE\right) = 0.95$$

$$P\left(\bar{X} - 1.96 \frac{\sigma}{\sqrt{N}} \leq \mu \leq \bar{X} + 1.96 \frac{\sigma}{\sqrt{N}}\right) = 0.95$$

↳ znači, interval je 1.96 na jednu i drugu stranu

PR: Odredi 95%-tni interval pouzdanosti na slijedeće podatke:

$N = 100$

$\bar{X} = 4.9$

$\hat{\sigma} = 0.5$ ⇒ ne znamo pravu vrijednost, pa smo ju procijenili (uzorak je velik, pa procjenjujemo pravu σ i radimo s normalnom razdiobom umjesto da radimo s egzaktnom studentovom razdiobom)

$$\mu = \bar{X} \pm 1.96\,SE = \bar{X} \pm 1.96\,\hat{\sigma}/\sqrt{N} = 4.9 \pm 0.1$$

$$4.802 \leq \mu \leq 4.998 \quad \text{(sa 95\% pouzdanosti)}$$

PR: Odredi 95%-tni interval pouzdanosti za manji uzorak:

$N = 10$, ali znamo da su iz normalne populacije

$\bar{X} = 4.9$ (da nije, ZABORAVI SVE ŠTO ZNAŠ I BACI U SMEĆE ☺)

$\hat{\sigma} = 0.5$

↳ prelazimo na t distribuciju zbog male uzorka te očekujemo da će taj interval pouzdanosti biti širi (manje podataka daje širu studentovu krivulju):

$$\bar{X} \sim t(df = 9) \qquad P(\bar{X} - 2.262\,SE \leq \mu \leq \bar{X} + 2.262\,SE) = 0.95$$

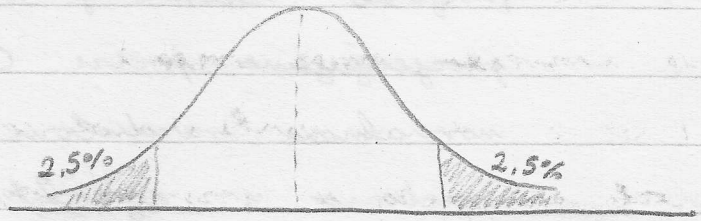
($df = N - 1$)

$$\hat{\mu} = 4.9 \pm 0.36$$

STATISTIČKO TESTIRANJE

HIPOTEZE

⇒ na neki način obrnuta ideja od intervala pouzdanosti:



$\alpha = 5\%$

↓

NIVO ZNAČAJNOSTI

(vjerojatnost odbacivanja ispravne hipoteze)

↓

računa nas vjerojatnost da dobijemo nešto u ovom intervalu izvan

PR: Ako je doista $\mu = 5$, koja je vjerojatnost da sam ja dobio 4.9 ili manje (ili 5.1 i više)

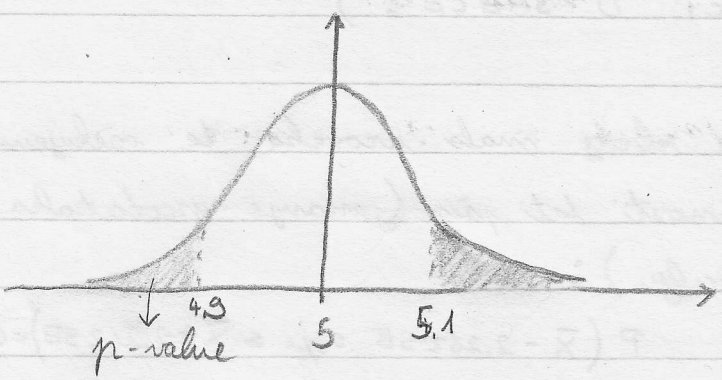
$N = 100$

$\hat{\sigma} = 0.5$

$\bar{x} = 4.9$

↓

ta vjerojatnost je „p"-value i ako je ona manja od α, tada odbacujemo hipotezu (tj. odbacujemo nultu hipotezu da je $\mu = 5$)



⇒ samo integriramo jer je „p"-value površina ovog razjenjenog područja

(DVOSTRANI TEST jer se gleda s obje strane)

⇒ KORACI:

1) formuliraj hipotezu H_0 i njoj alternativnu hipotezu
   ↳ sada nadalje želimo odbaciti H_0
2) odaberi nivo značajnosti α (npr. $\alpha = 5\%$)
   ↳ to je vjerojatnost False-positiva → u koliko slučajeva odbacujemo H_0, a to nismo smjeli (to je pogreška prve vrste)
   ↳ oko je α premali, možemo ne-odbaciti H_0, a trebali smo (to je pogreška druge vrste)
3) provjeri vrijede li pretpostavke
4) odaberi prikladan test i odgovarajuću testnu statistiku (mi koristimo t-test)
5) izračunaj testnu statistiku (npr. t-statistiku)
6) a) usporedi vrijednost testne statistike s kritičnom vrijednošću: upada li vrijednost testne statistike u kritično područje?
   b) izračunaj p-vrijednost statistike: je li p-vrijednost $< \alpha$?
   (oni su alternative, danas koristi b) jer je lako integrirati, a prije se koristio a))
7) formuliraj zaključak
   " možemo li odbaciti H_0 il ne možemo nešto?

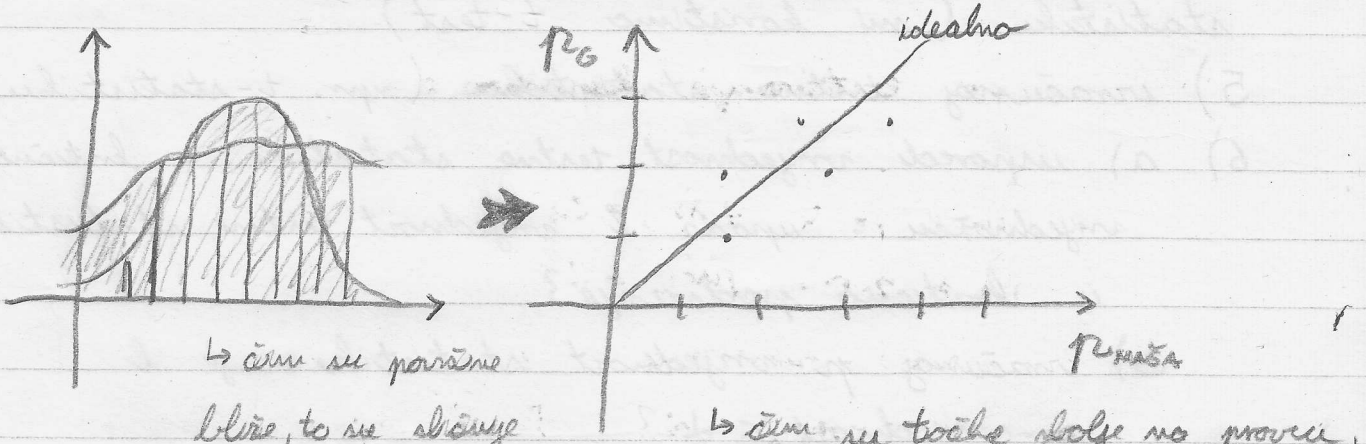
USPOREDBA MODELA

⇒ test za normalnu razdiobu ako je $N < 30$:

↳ CGT ne vrijedi, pa trebamo neki test

↳ postoji vizualni test:

- integriraj normalnu i svoju distribuciju po kvantilima
- ako u kvantilnom prostoru dobivamo pravac (ili približno), ključ normalne distribucije smo



↳ čim su površine bliže, to su sličnije

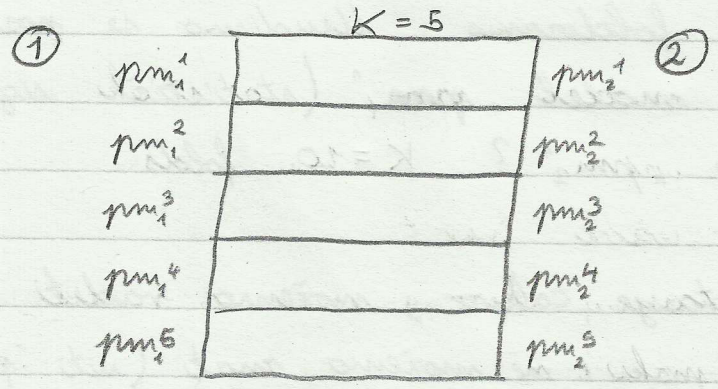
↳ čim su točke dalje na pravcu, to su sličnije

↳ ovo nije strogo vezano uz usporedbu modela

⇒ **usporedba dva klasifikatora:**

- **UPARENI t-TEST:**

„pm" - performance measure



} radimo sa srednjim vrijednostima (nažalost manje od 30 uzoraka)

↳ radimo testiranje hipotere:

$H_0: \overline{pm_1} - \overline{pm_2} = 0$

$H_1: \overline{pm_1} - \overline{pm_2} \neq 0$

$pm_1^1 - pm_2^1 = d^1$

$pm_1^2 - pm_2^2 = d^2$

$\vdots$

$pm_1^5 - pm_2^5 = d^5$

$\bar{d} = \overline{pm_1} - \overline{pm_2}$

↓

ovo znam da je t-statistika sa K-1 stupnjeva slobode (K je veličina uzorka - nekad je označa N)

↳ pišemo t-statistiku (procjenjujemo σ):

$$t = \frac{\bar{d} - \mu^{0}}{\hat{\sigma}/\sqrt{K}}$$

$$\sigma = \sqrt{\frac{\sum (d_i - \bar{d})^2}{K-1}}$$

PR: Neka su „pm_1" i „pm_2" točnosti dvaju klasifikatora. Vidimo da je „pm_1" uvijek veća od „pm_2" u našim foldovima. Mislimo se postavit pitanje: je li model „pm_1" (statistički signifikantno) bolji od modela „pm_2"? $K = 10$ folds

↳ jer je pitanje takvo, možemo raditi jednostrani test, al u praksi ne možemo znati (niti pretpostavit) da je neki model bolji od drugog, pa uvijek radimo dvostrani test ☺

$H_0 \dots \overline{pm}_1 - \overline{pm}_2 = 0$ → nisam siguran

$K = 10$ folds

$d_1 = pm_1^1 - pm_2^1 = 0.17$

$\vdots$

$d_{10} = \dots$

$\overline{pm}_1 = 0.2175$

$\overline{pm}_2 = 0.0649$

$\bar{d} = \overline{pm}_1 - \overline{pm}_2 = 0.1526$

$\hat{\sigma} = 0.05969$

$$t = \frac{0.1526 - 0}{0.05969/\sqrt{10}} = 8.0844$$

↳ pogledamo u tablicu i dobivamo da je p manje od 5% te odbacujemo hipotezu

ODABIR ZNAČAJKI

⇒ želimo smanjiti broj značajki zbog redundantnosti, irelevantnosti i šuma

⇒ druga motivacija je nemogućnost izračuna ako imamo više značajki nego primjera

⇒ treći razlog je lakša interpretacija modela (možemo napraviti vizualizaciju ako je $n \leq 3$)

⇒ DVA PRISTUPA SMANJENJA DIMENZIONALNOSTI:

a) **odabir značajki**

- učimo samo neki podskup značajki
- prednost je da zadržimo interpretabilnost
- negativna strana je da može uzrokovat prenaučenost

b) **transformacija značajki**

- transformira značajke u nove značajke kombinacijom značajki
- gubi se interpretabilnost

⇒ mi se bavimo ovime pod a) (odabir značajki)

UNIVARIJANTNA METODA FILTRA

⇒ ocenjujemo značajke
↳ **Merit** - visok merit, visoka korisnost značajke

⇒ nakon ocene, sortiramo značajke
⇒ nakon sortiranja, uzimamo najboljih M značajki

⇒ DOBRE STRANE:
- neovisno o modelu

⇒ LOŠE STRANE:
- neovisno o modelu (možda ne koristimo neke informacije)
- neovisno odredjujemo Merit za svaku varijablu (tako zanemarujemo neke informacije)
↳ univarijantni filtar uzima u obzir samo jednu varijablu

⇒ mjerenje zavisnosti klase Y i značajke X_i ($X \to Y$):

① **UZAJAMNA INFORMACIJA:**

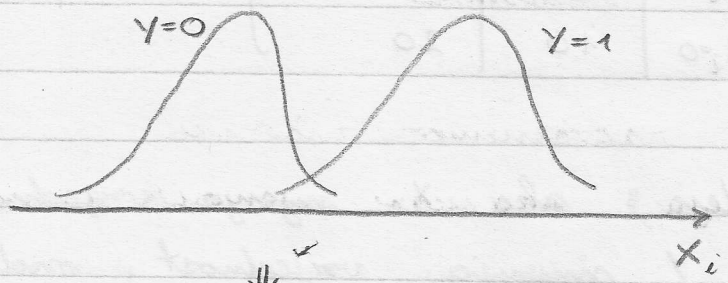
$$I(X,Y) = D_{KL}\left(P(X,Y) \mid P(X)P(Y)\right)$$
$$= \sum_{X,Y} P(X,Y) \ln \frac{P(X,Y)}{P(X)P(Y)}$$

↳ ako je informacija velika, postoji veliki protok

informacije između značajke i oznake klase pa tu značajku želimo ukloniti

(2) t-TEST:

↳ omogućava da usporedimo jesu li dva uzorka iz iste populacije



} test za dvije klase (za više klasa koristi ANOVA metodu)

⇓

ove distribucije se malo preklapaju što znači da ova značajka dobro razdvaja razrede

↳ želimo što manju „p"-vrijednost (to je za one dobre značajke)

$$t = \frac{\bar{x}_i^1 - \bar{x}_i^2}{\hat{\sigma} \cdot \sqrt{\frac{1}{N_1} + \frac{1}{N_2}}}$$

} malo drugačiji test od onog ranije

↳ problem je to što ga možemo koristiti samo za kontinuirane značajke //

③ χ^2-TEST:

↳ koristo se za diskretne značajke

PR: $K=2$, $n=2$

	$Y=1$	$Y=0$	
$X_i=1$	30	5	$N=65$
$X_i=0$	10	20	

↳ ideja: ako X_i mjerja vrijednost 1 ako tada Y mjerja vrijednost, onda je X_i dobra značajka

↳ marginalizirano vjerojatnosti:

$P(X_i=1) = 0.54$

$P(X_i=2) = 0.46$

$P(Y=1) = 0.62$

$P(Y=2) = 0.38$

$$E\{X=1, Y=1\} = N \cdot \underbrace{P(X=1) \cdot P(Y=1)}$$

pretpostavimo nezavisnost, pa ovo pokušavamo dobiti (to je hipotera H_0)

ovo računamo za svaku ćeliju

	$Y=1$	$Y=0$
$X_i=1$	21.76	13.34
$X_i=0$	18.53	11.36

↳ pišemo χ^2 vardiobu:

$$\chi^2 = \sum_{j=1}^{K} \sum_{i=1}^{K_i} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}} \sim \chi^2_{(K-1)(K-1)}$$

↳ za naš primjer:

$$\chi^2 = \frac{(30-21.76)^2}{21.76} + \frac{(5-13.34)^2}{13.34} + \ldots = 18.53$$

↳ želimo dobiti malu „p"-vrijednost kako bi mogli odbaciti hipotezu da su značajka i oznaka klase neravisne - tada tu značajku želimo uzeti u obzir

④ p - METRIC:

$$P(X_i, Y) = \frac{\overline{X}_i^{Y=0} - \overline{X}_i^{Y=1}}{\sigma^2_{Y=0} + \sigma^2_{Y=1}}$$

↳ želimo da P vrijednost bude što veća jer su tada klase udaljenije i zgusnutije!

↳ slično kao Fisher iz raspoznavanja uzoraka

⑤ VIF:

↳ ove metode do sada ne gledaju zavisnost između varijabli - VIF gleda redundantnost (ostale metode relevantnost)

↳ VIF se pita:

- je li X_i redundantna?
  (da li je X_i multikolinearan u odnosu na druge značajke)

↳ VIF traži odgovor tako da radi regresiju pomoću ostalih značajki da predvidi X_i:

- ako dobro predvidi X_i, ta značajka je loša jer se lako predvida

$$h(x_1 | \vec{w}) = w_1 x_2 + \dots + w_{n-1} x_n + w_0$$

ovo je za značajku X_1 (analogno za X_i)

$$\boxed{VIF = \frac{1}{1 - R^2}}$$

$R^2 \rightarrow$ koeficijent determinacije

$$VIF \in [1, \infty\rangle$$

↳ što manje to bolje
(ako je veći od 10, tipično nećemo uzeti tu varijablu)

↳ o koeficijentu determinacije:

$$R^2 = \frac{TSS - RSS}{TSS}$$

$$R^2 \in [0, 1]$$

najbolji model

$$RSS = \sum_{i=1}^{N} (y^i - h(x^i))^2$$ ⇒ zbroj kvadrata reziduala

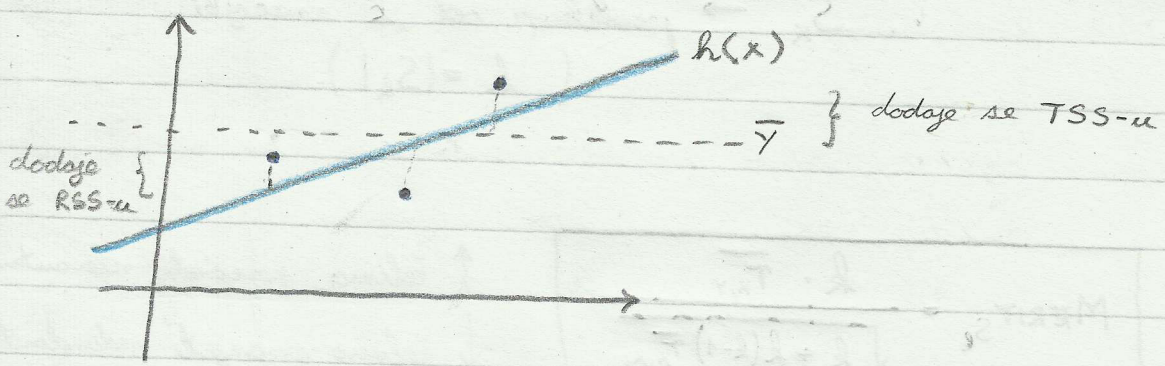
↓

zbroj kvadrata odstupanja od stvarne funkcije koju smo modelirali regresijom

$$TSS = \sum_{i=1}^{N} (y^i - \bar{y})^2$$

↓

zbroj kvadrata odstupanja od srednje vrijednosti (trebalo bi biti manje od RSS jer bi naš model dobiven regresijom trebao bolje predvidjeti tok od srednje vrijednosti)

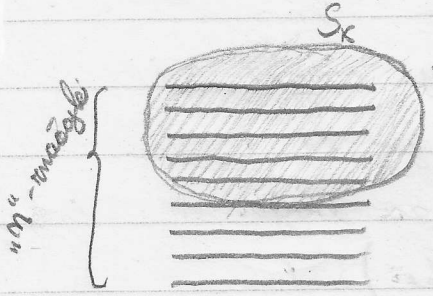


⇒ NAPOMENA: ove metode se najčešće koriste kako vrsu idealne jer gledaju značajke u izolaciji

MULTIVARIJANTNA METODA FILTRA

① CFS - „Corelation Feature Selection"

↳ ideja je maksimizirati relevantnost u odnosu na „Y" i minimizirati korelaciju između značajki



⇒ uzmemo neki skup značajki kako bi izbjegli kombinatornu eksploziju (jer za „n" značajki imamo 2^n skupova)

S_k → podskup od „k" značajki ($k = |S_k|$)

$$MERIT_{S_k} = \frac{k \cdot \overline{r_{x,y}}}{\sqrt{k + k(k-1)\,\overline{r_{x,x}}}}$$

↑ želimo povećat relevantnost

↓ želimo smanjit redundantnost

↳ ovo dalje optimiramo / pretraživanjem „forward - best - first - search" algoritmom

② RELIEF

↳ iterativno gradimo vektor relevantnost značajki

↳ ako je udaljenost značajki primjera iz **iste klase**:

a) VELIKA:

↳ ta značajka je loša i njoj ćemo smanjit težinu u vektoru relevantnost značajki

b) MALA:

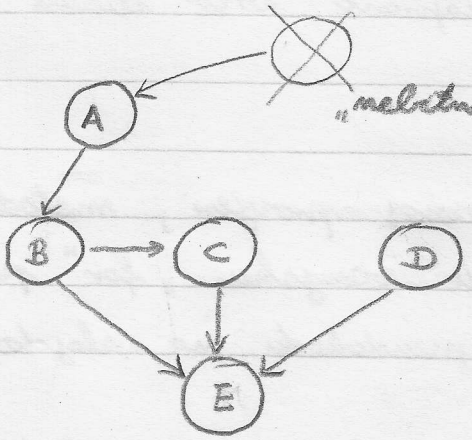
↳ ta značajka je dobra i nećemo kažnjavat - štoviše, malo ćemo povećat njenu vrijednost u vektoru relevantnosti značajki

↳ parove primjera za usporedbu biramo tako da nasumično izaberemo primjer i jedan primjer najbliže njemu

③ METODA RELEVANTNOSTI ZA PGM:

↳ **Markovljev omotač**

- na klasifikaciju čvora utječu njegovi **roditelji, djeca i roditelji njegove djece** (efektivno susjedi tog čvora)

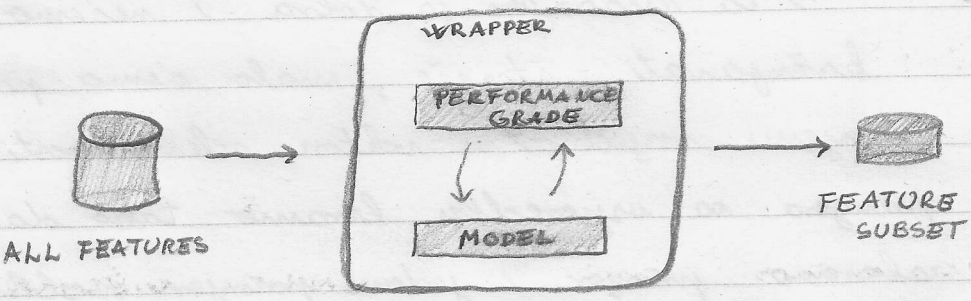


svi čvorovi od A-E su bitni za klasifikaciju čvora B!

METODA OMOTAČA

⇒ ova metoda je ovisna o klasifikatoru nad kojim radi (nalazi najbolje značajke za ocjenu dobrote danog modela)

⇒ IDEJA: isprobavaj model nad različitim podskupovima svih značajki dok ne nađeš nešto dobro



↳ pretražujemo velik prostor stanja, pa se možemo koristiti npr. genetskim algoritmima

⇒ problem ovoga:
- ne daje uvijek dobro rješenje
- radi specifično za dani model i ocjenu performansi

⇒ polunaivni Bayes je naprava vrlo sličan metodi omotača (probaj ovo skužiti)

⇒ u ugniježđenoj unakrsnoj provjeri, metode odabira značajki ugrađujemo u vanjsku „for" petlju BITNO!
(usmedu linija 3. i 4. u pseudokodu na slajdovima)

UGRAĐENE METODE

⇒ to su sve one metode / modeli gdje model sam vrši odabir značajki

↳ po učenju modela rezultat dobivamo podskup značajki.

⇒ neki od tih algoritama:

- BOOSTING
- STABLA ODLUKE
- RANDOM FOREST
- sve metode koje koriste L1-regularizaciju
- SVM uzima samo potporne vektore u obzir, pa on implicitno bira značajke

KAKO RADITI EVALUACIJU
(skraćeno)

⇒ napraviti tablicu:

	P	R	F_1^M	Acc
MODEL$_1$	....	....	....	....
MODEL$_2$	....	....	70.8	....
MODEL$_3$	....	....	....	....
BASELINE	....	....	69.2	....

} sve ove brojke treba dobiti usrednjenom unakrsnom provjerom

neki dovoljno lagan model za provjeru, ali ne preglup da se ne osjećamo predobro u vezi rezultata

usporedi ove modele s t-testom i donesi neki zaključak (dobiti ćemo da je jedan model bolji od drugoga)

↓

reci je li ta razlika od baseline u praksi bitna!

⇒ ABLACIJSKA STUDIJA:

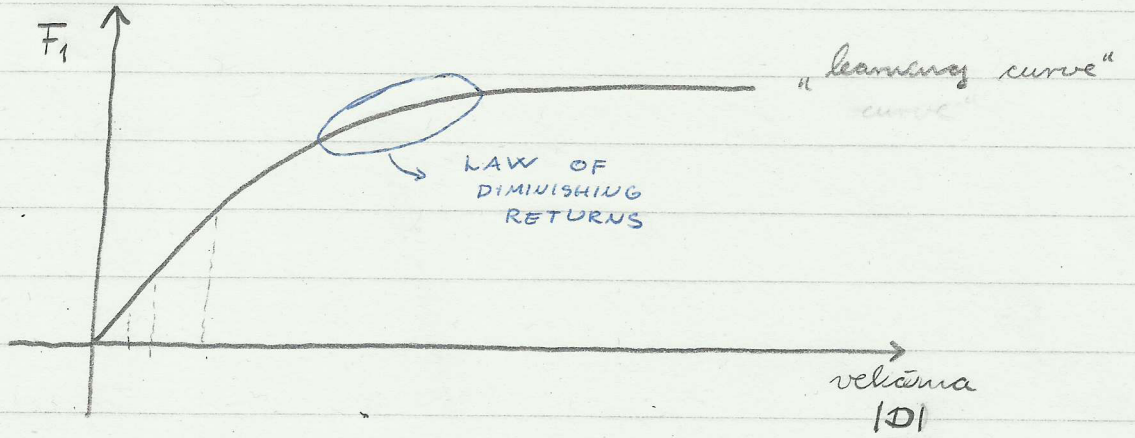
↳ napravi model sa svim značajkama i model bez te jedne značajke

↳ ako je model bez te značajke znatno lošiji, ta značajka je bitna

↳ te modele opet usporedi t-testom

⇒ **KRIVULJA UČENJA:**

↳ u praksi, nemamo dovoljno vremena ili novca da označimo jako puno primjera, pa želimo naći optimum prikladne označenih podataka



↳ ako je krivulja učenja strma, još ima potencijala za model, pa ima smisla još uložiti u označavanje primjera