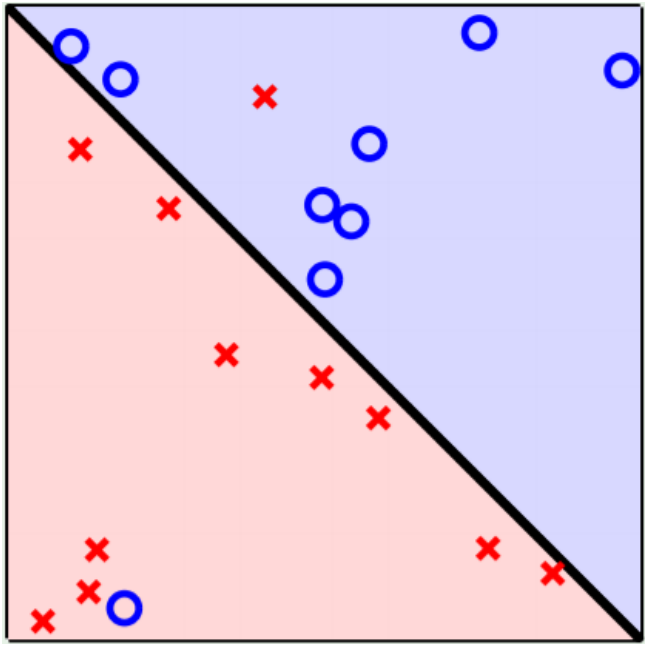
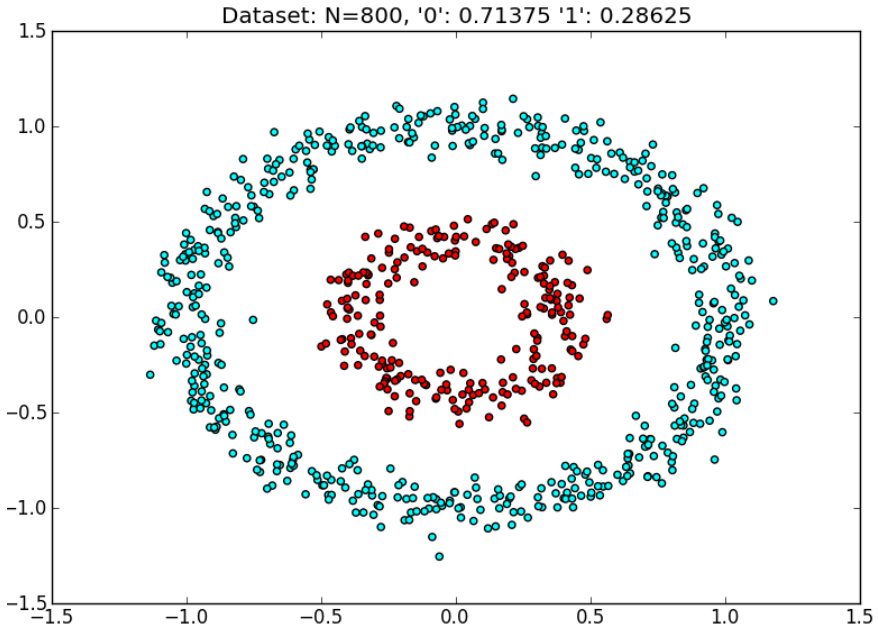


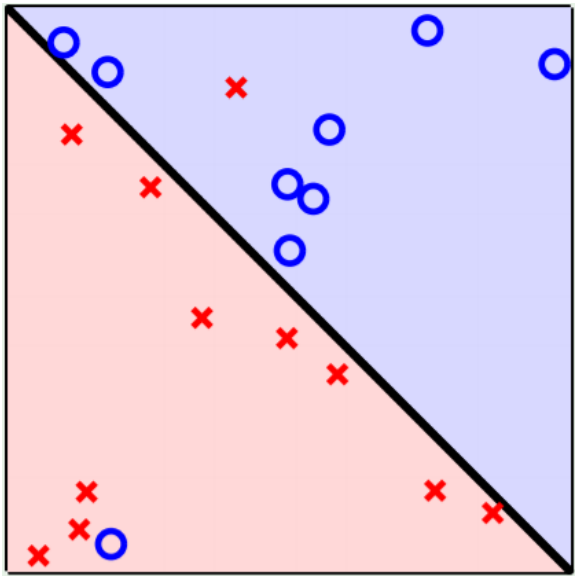


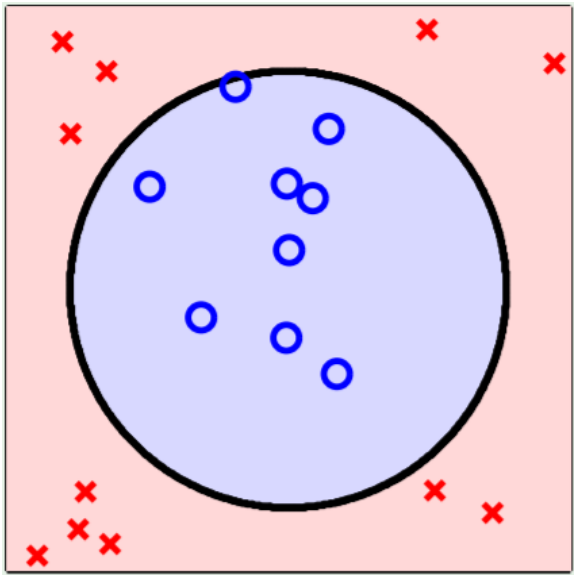
Problemi koji NISU linearno separabilni

Kernel trik

Šta ako problem nije linearno separabilan?

- Dva tipa neseparabilnih podataka





Neznatno

Outlieri (verovatno greške), ne želimo da idemo u visokodimenzioni prostor da bismo korektno klasifikovali ove tačke

Soft margin

Ozbiljno

Nisu outlieri, problem deluje nelinearno – potrebna nam je nelinearna transformacija

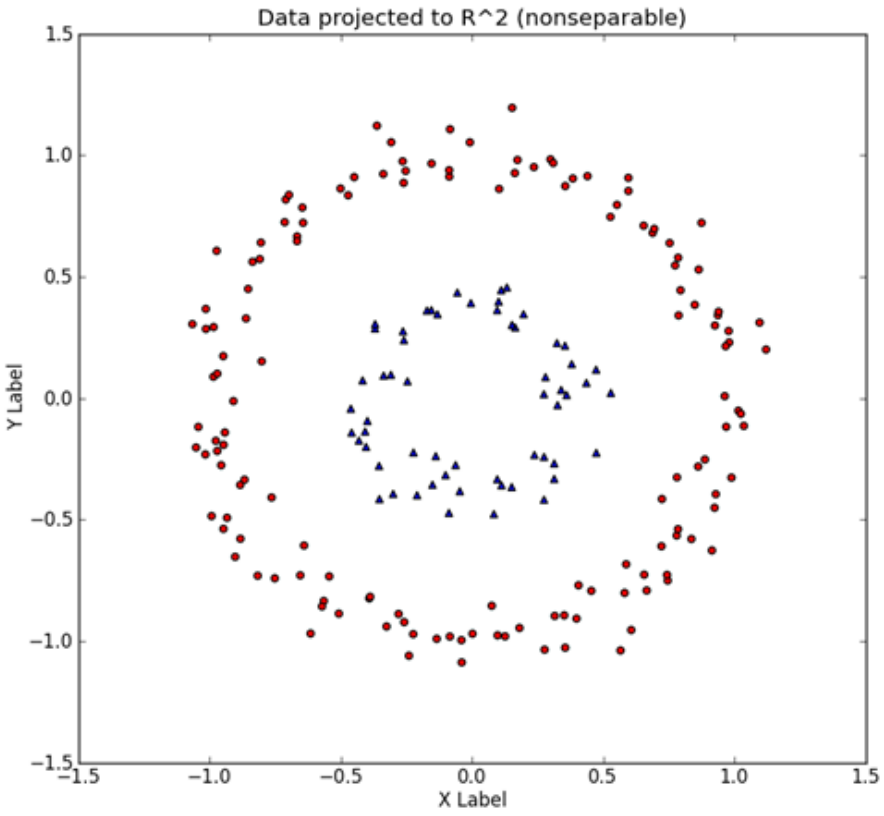
Kerneli

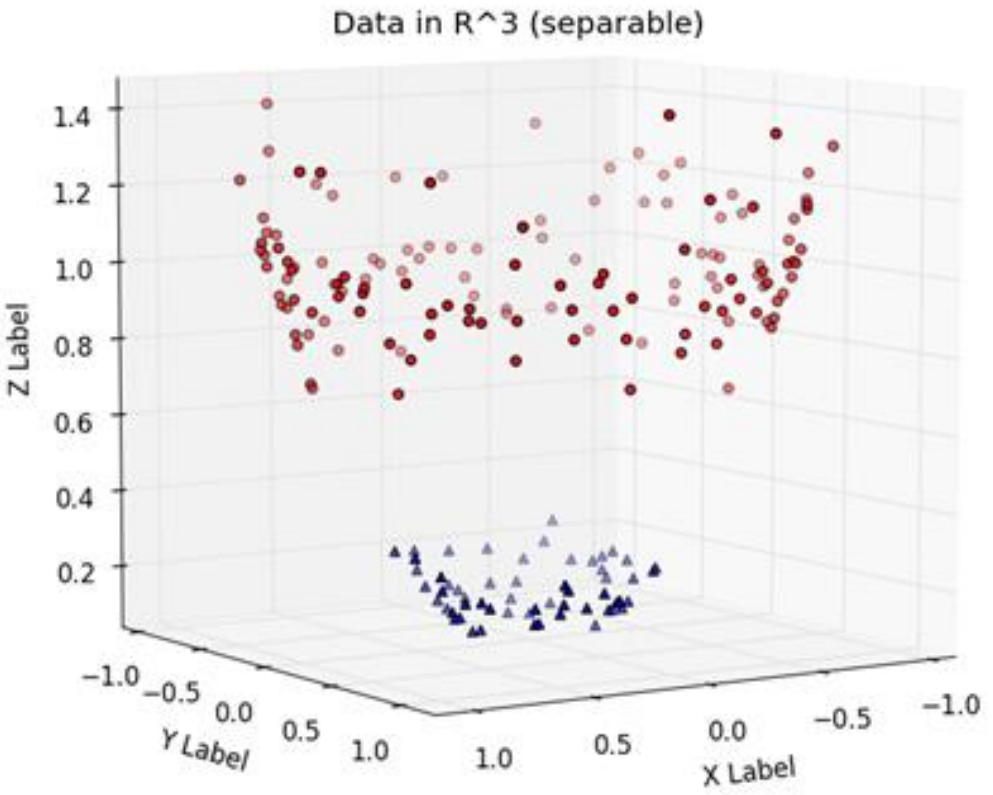
Nelinearna transformacija podataka

- Na raspolaganju imamo samo SVM koji za dati *D*-dimenzioni skup podataka pronalazi $(D-1)$-dimenzionu razdvajajuću hiperravan

- Ali, šta ako bismo mogli da manipulišemo sa *D*?
  - Dodaćemo polinomijalna obeležja (kao što smo radili kod perceptrona i logističke regersije)
  - Ovim prelazimo u višedimenzioni prostor

Nelinearna transformacija podataka



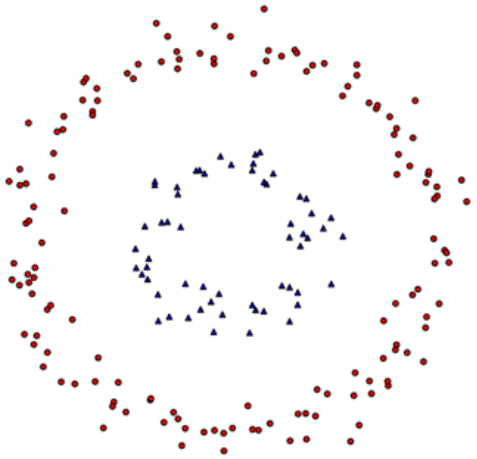


$$(x_1, x_2) \xrightarrow{\phi} (x_1, x_2, x_1^2 + x_2^2) \qquad \phi\colon \mathbb{R}^2 \to \mathbb{R}^3$$

- Podaci su u novom prostoru linearno separabilni i možemo da primenimo SVM!

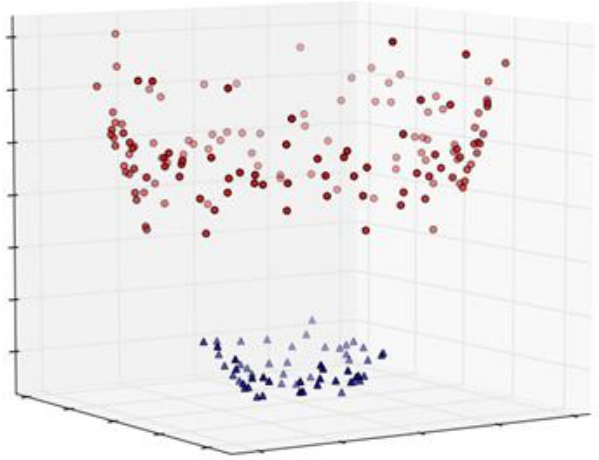
Nelinearne transformacije – ciklus

1. Originalni prostor $X \epsilon \mathbb{R}^2$

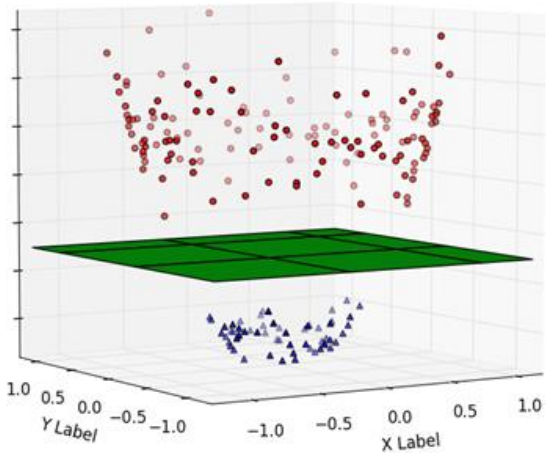


ϕ

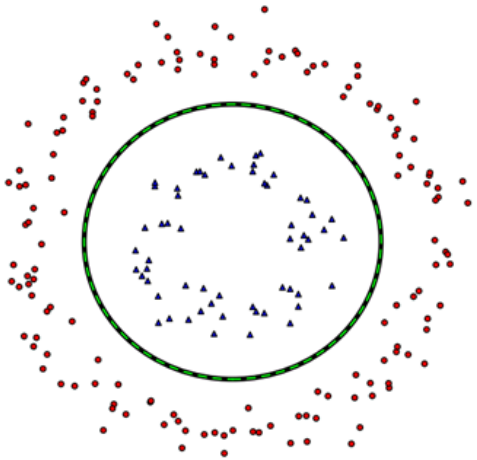
2. Transformisati podatke u novi prostor $Z \in \mathbb{R}^3$



3. Razdvojiti podatke u prostoru Z



$'\phi^{-1}'$



Sumarizacija

- Skup podataka T koji nije linearno separabilan u $\mathbb{R}^d$ može biti linearno separabilan u prostoru $\mathbb{R}^D$ gde je $D > d$
- Ako imamo transformaciju ϕ koja preslikava skup podataka T u višedimenzioni skup podataka T' koji je linearno separabilan:
  - Možemo da treniramo linearni SVM na skupu podataka T' da bismo pronašli granicu odluke $\vec{\theta}$ koja razdvaja klase u prostoru T'
  - Projektovanje granice odluke $\vec{\theta}$ (linearne u prostoru $\mathbb{R}^D$) nazad u originalni prostor $\mathbb{R}^d$ će rezultovati nelinearnom granicom odluke u $\mathbb{R}^d$
- Ovo znači da možemo da naučimo nelinearnu granicu odluke koristeći originalnu linearnu formulaciju SVM-a

Nelinearne transformacije – problem

- Opisana šema je privlačna zbog svoje jednostavnosti, ali...
- Uvećavamo dimenzionalnost sa $\mathbb{R}^d$ u $\mathbb{R}^D$ gde je $D > d$
- Ako D raste veoma brzo u odnosu na d (npr. $D \in O(2^d)$) imaćemo ozbiljnih problema sa nedostatkom računarskih resursa prilikom treniranja SVM-a u novom visokodimenzionom prostoru
  - Često korišćen *kernel* za SVM je *polinomijalni kernel*
  - Za polinomijalni kernel 2. stepena (implicitno) se vrši sledeća transformacija:

$$[x_1, x_2] \rightarrow \left[x_1^2, x_2^2, \sqrt{2} \cdot x_1 \cdot x_2, \sqrt{2c} \cdot x_1, \sqrt{2c} \cdot x_2, c\right]$$

  - Ova transformacija rezultuje sa 3 dodatne dimenzije $\mathbb{R}^2 \rightarrow \mathbb{R}^5$
  - Uopšteno, polinomijalni kernel mapira prostor $\mathbb{R}^N$ u $\binom{N+d}{d}$-dimenzioni prostor
  - Dakle, za visokodimenzione skupove podataka, naivna primena ove transformacije će veoma brzo postati računarski neizvodljiva
- Da li to znači da ne možemo upotrebiti ovaj pristup?

Kernel trik

- Sećate se optimizacije?

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)T} x^{(j)}$$

- Ako primenjujemo SVM u novom prostoru z (gde su podaci linearno separabilni):

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y^{(i)} y^{(j)} \alpha_i \alpha_j z^{(i)T} z^{(j)}$$

- Recimo da transformišemo problem iz 2-dimenzionog prostora u 10^6 dimenzioni prostor. Koliko je optimizacioni problem postao teži?
  - Tražićemo skalarni proizvod vektora dimenzije 10^6 - nije strašno, ovo je skalarni proizvod (broj)
  - Broj α parametara koje treba da odredimo (stvarna dimenzionalnost problema koji rešavamo) zavisi samo oda broja primera u skupu podataka – nema veze sa dimenzionalnošću prostora
  - Možemo da transformišemo u visokodimenzioni prostor, a da ne platimo cenu za to!

Kernel trik

- Sećate se optimizacije?

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)T} x^{(j)}$$

- Ako primenjujemo SVM u novom prostoru z (gde su podaci linearno separabilni):

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y^{(i)} y^{(j)} \alpha_i \alpha_j z^{(i)T} z^{(j)}$$

- Označimo transformaciju prostora sa Φ, tj. $\Phi\left(x^{(i)}\right) = z^{(i)}$
- Ako imamo funkciju k:

$$k\left(x^{(i)}, x^{(j)}\right) = \Phi\left(x^{(i)}\right) \cdot \Phi\left(x^{(j)}\right)$$

ne moramo da znamo šta je transformacija Φ!

Kernel trik

Originalno rešenje SVM optimizacionog problema:

$$f_{\theta,b}(x) = \sum_{i=1}^{N} \alpha_i y_i x_i \cdot x + b$$

Zamena skalarnog proizvoda u $\mathbb{R}^N$ sa kernelom *k*:

$$f_{\theta,b}(x) = \sum_{i=1}^{N} \alpha_i y_i k(x_i, x) + b$$

- Zahvaljujući ovom, **SVM ne mora eksplicitno da radi u više-dimenzionom prostoru** tokom treniranja/testiranja modela
- Reprezentacije podataka u tom prostoru se ne konstruišu eksplicitno
- Postoje kerneli koji odgovaraju skalarnim proizvodima u beskonačnodimenzionalnim prostorima, a izračunavaju se efikasno!

Kernel trik – implikacije

1. Koristeći kernel $k(\vec{v}, \vec{w})$ možemo implicitno transformisati skup podataka u višedimenzioni prostor $\mathbb{R}^M$ bez potrebe za dodatnom memorijom i uz minimalan uticaj na vreme izračunavanja
   - Jedini efekat na vreme izračunavanja jeste dodatno vreme za izračunavanje $k(\vec{v}, \vec{w})$. U zavisnosti od k, ovo vreme može biti minimalno

2. Zahvaljujući 1. možemo **efikasno** naučiti nelinearne granice odluke pomoću SVM-a, jednostavnom **zamenom sklarnih proizvoda vektora u SVM optimizaciji sa $\boldsymbol{k(\vec{v}, \vec{w})}$**

- Korišćenje kernel funkcija zarad postizanja 1. i 2. se naziva kernel trik

Šta je kernel $k\left(x^{(i)}, x^{(j)}\right)$?

- *Kernel funkcije* $k(\vec{v}, \vec{w})$ su funkcije, koje:
  - za data dva vektora v i w u prostoru $\mathbb{R}^N$
  - mogu implicitno da izračunaju skalarni proizvod v i w u prostoru $\mathbb{R}^M$
  - bez eksplicitne transformacije v i w u prostor $\mathbb{R}^M$

- Jednostavnije: kernel je skalarni proizvod u nekom drugom prostoru

Primer polinomijalnog kernela

- Uzmimo kao primer jednostavan polinomijalni kernel 2. stepena

$$k(v, w) = (1 + v^T w)^2$$

gde $v = [v_1, v_2], w = [w_1, w_2] \in \mathbb{R}^2$

- Ovo ne izgleda kao da odgovara bilo kojoj funkciji mapiranja ϕ, ovo je samo funkcija dva vektora koja vraća realan broj

- Međutim, ako rastavimo izraz:

$$k(v, w) = (1 + v^T w)^2 = (1 + v_1 w_1 + v_2 w_2)^2$$
$$= 1 + v_1^2 w_1^2 + v_2^2 w_2^2 + 2v_1 w_1 + 2v_2 w_2 + 2v_1 w_1 v_2 w_2$$

Ispada da je ovo upravo skalarni proizvod dva vektora:

$$\left(1, v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, \sqrt{2}v_1 v_2\right) \text{ i } \left(1, w_1^2, w_2^2, \sqrt{2}w_1, \sqrt{2}w_2, \sqrt{2}w_1 w_2\right)$$

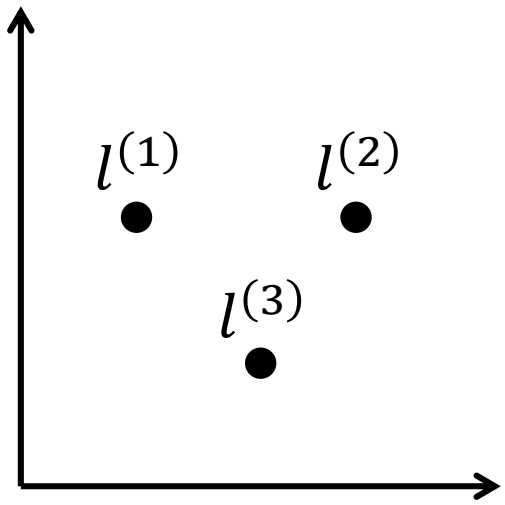
- Dakle, kernel $k(v, w)$ računa skalarni proizvod u 6-dimenzionom prostoru, a da nismo morali eskplicitno posetiti taj prostor

Šta je kernel $k\left(x^{(i)}, x^{(j)}\right)$?

- Postoje mnoge poznate kernel funkcije

- Intuitivno, kernel je funkcija sličnosti
  - Kernel predstavlja skalarni proizvod (u nekom prostoru)
  - A skalarni proizvod je na neki način mera sličnosti

Šta je kernel?



- Odabrati tačke $l^{(1)}$, $l^{(2)}$ i $l^{(3)}$
- Ove tačke ćemo zvati *landmarks*
- Definisaćemo meru sličnosti kao:

Euklidska udaljenost

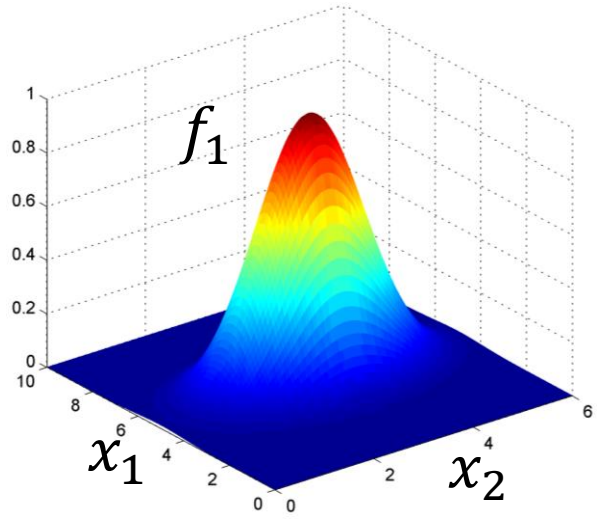
$$\text{similarity}(a, b) = \exp\left\{-\frac{\|a-b\|^2}{2\sigma^2}\right\}$$

Kernel (ova konkretna mera je RBF (Gausov) kernel)

- Za dato x, izračunati nova obeležja na osnovu blizine *landmarks* :

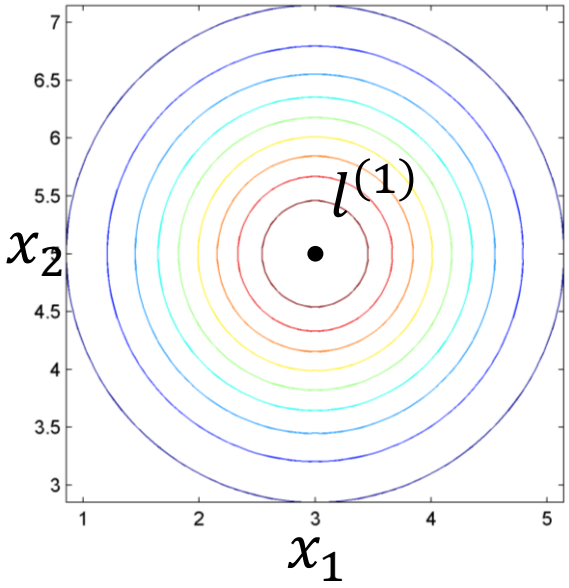
$$f_1 = \text{similarity}\left(x, l^{(1)}\right) = k\left(x, l^{(1)}\right)$$
$$f_2 = \text{similarity}\left(x, l^{(2)}\right) = k\left(x, l^{(2)}\right)$$
$$f_3 = \text{similarity}\left(x, l^{(3)}\right) = k\left(x, l^{(3)}\right)$$

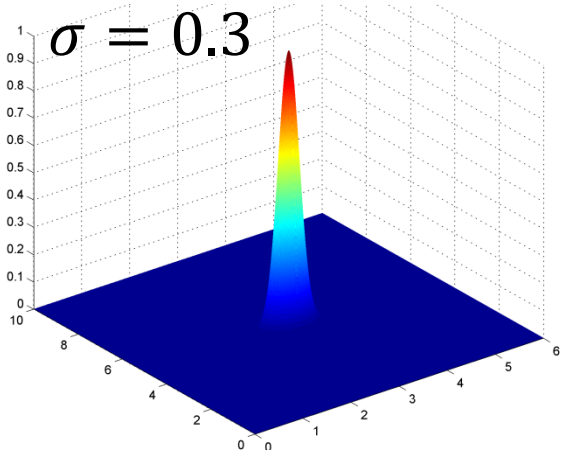
Primer kernela: RBF (Gausov) kernel

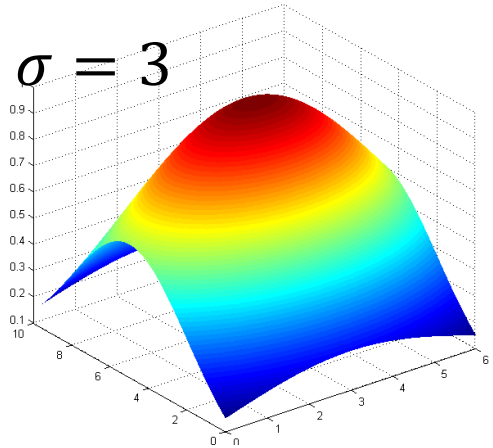


$$f_1 = k\left(x, l^{(1)}\right) = \exp\left(-\frac{\sum_{d=1}^{D}\left(x_d - l_d^{(1)}\right)^2}{2\sigma^2}\right)$$

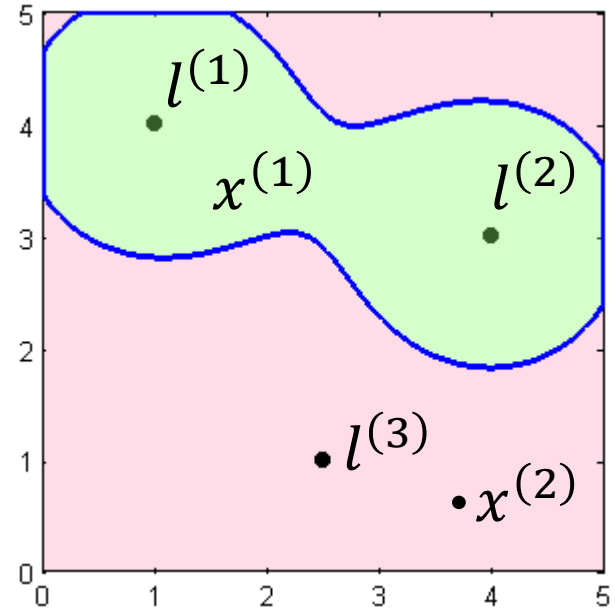
- Za $x \approx l^{(1)} \rightarrow f_1 \approx 1$
- Za x udaljeno od $l^{(1)} \rightarrow f_1 \approx 0$
- σ diktira koliko brzo opada vrednost f_1 sa udaljenošću x od $l^{(1)}$

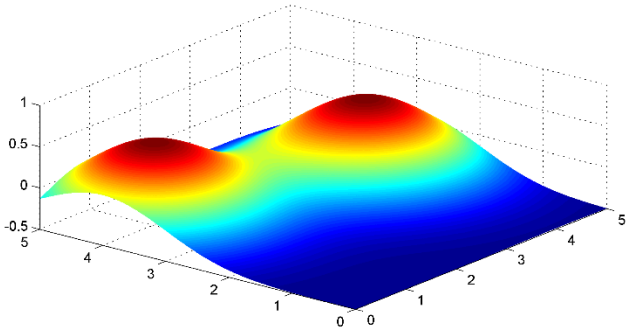






Primer kernela: RBF (Gausov) kernel





- Recimo da su date tačke $l^{(1)}, l^{(2)}$ i $l^{(3)}$
- Za dati primer x ćemo odrediti obeležja f_1, f_2 i f_3
- Naša hipoteza predviđa klasu +1 kada važi

$$h_\theta(x) = \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$$

- Recimo da smo obučili model i dobili vrednosti:

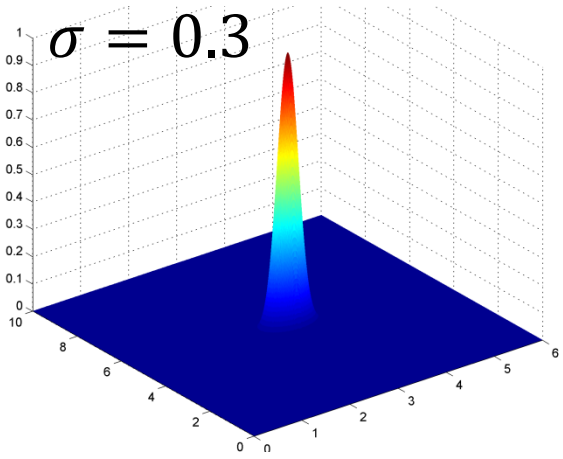
$$\theta_0 = -0.5, \theta_1 = 1, \theta_2 = 1, \theta_3 = 0$$

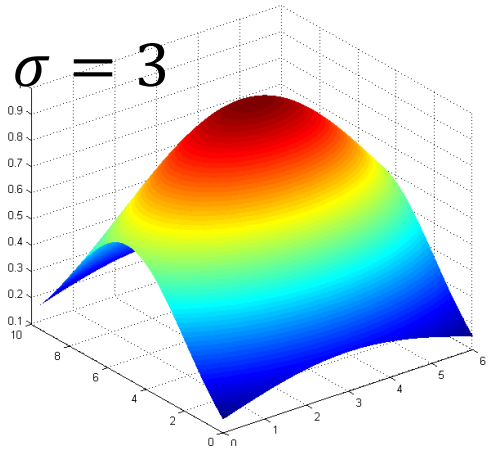
- Za tačku $x^{(1)}$ koja se nalazi u blizini $l^{(1)}$ važi $f_1 \approx 1, f_2, f_3 \approx 0 \rightarrow h_\theta(x) \approx 0.5 > 0$
- Za tačku $x^{(2)}$ koja se nalazi u blizini $l^{(3)}$ važi $f_1, f_2, f_3 \approx 0 \rightarrow h_\theta(x) \approx -0.5 < 0$

Primer kernela: RBF (Gausov) kernel

- Gausov kernel uvodi još jedan dodatan parametar σ^2 koji diktira kako variraju obeležja f_i sa udaljenošću x od $l^{(i)}$

manje sistematsko odstupanje, veća varijansa





veće sistematsko odstupanje, manja varijansa

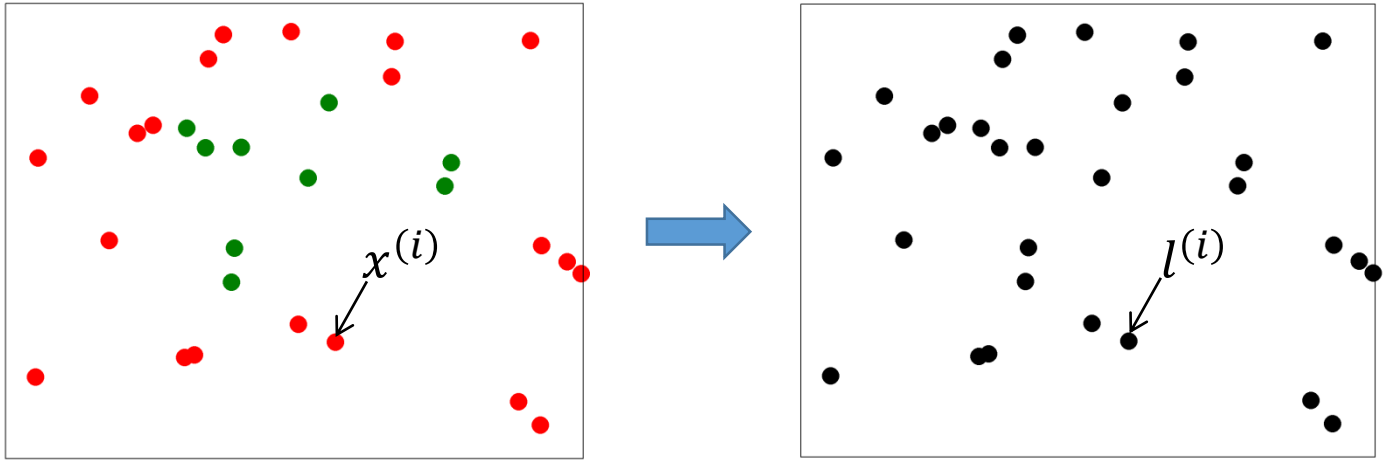
- Parametar σ^2 takođe treba da odredimo iz podataka

Kerneli

- Kako odabrati *landmarks* i koliko ih postaviti?
- Kako odabrati funkciju sličnosti (kernel)?

Kako odabrati *landmarks*

- Na mesto svake instance iz trening skupa postavićemo jedan *landmark:*



- Na ovaj način, svako obeležje meri koliko je novi primer udaljen od primera uočenih u trening skupu
- Broj obeležja je *N* (jednak je broju primera u trening skupu)

SVM sa kernelom sumarizacija

1. Dat je trening skup $T = \{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$. Odabrati *landmarks* tako da:

$$l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(N)} = x^{(N)}$$

2. Svaki primer $x^{(i)}$ iz trening skupa predstaviti putem novih obeležja:

$$f_1 = k(x^{(i)}, l^{(1)})$$
$$f_2 = k(x^{(i)}, l^{(2)})$$
$$\dots$$
$$f_N = k(x^{(i)}, l^{(N)})$$

$$f^{(i)} = \begin{bmatrix} f_0 \\ f_1^{(i)} \\ \dots \\ f_N^{(i)} \end{bmatrix}, f_0 = 1$$

3. Trenirati SVM koristeći trening skup $\{(f^{(1)}, y^{(1)}), \dots, (f^{(N)}, y^{(N)})\}$

Popularni kerneli

- Linearni kernel (bez kernela, standardan linearni klasifikator)
  - Razuman izbor ako imamo veliko D, a malo N
  - Ne želimo kompleksne granice odluke u visokodimenzionom prostoru jer imamo mali broj primera pa se lako može desiti da overfitujemo

- RBF (*Radial Basis Function*, Gausov) kernel
  - $f_i = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$, gde je $l^{(i)} = x^{(i)}$
  - Parametri koje moramo podesiti: σ^2
  - Razuman izbor ako imamo malo *D*, a veliko *N*
  - **Napomena: skalirajte obeležja ukoliko koristite Gausov kernel**
    - Prilikom računanja kernela koristi se Euklidska udaljenost – bez skaliranja će neke dimenzije imati veći uticaj od drugih samo zbog opsega u kome se kreću

Popularni kerneli

- Polinomijalni kernel
  - $k(v, w) = (v^T w + c)^d$, parametri: c, d
  - Obično ima lošije performanse od Gausovog kernela
- Sigmoid kernel
  - $\tanh\left(\left\langle \vec{x}^{(i)}, \vec{x}^{(j)} \right\rangle + r\right)$
- String kernel, chi-square kernel, histogram intersection kernel,...

- U najviše primena se koriste linearni ili RBF kernel
- Izbor „korektnog“ kernela je netrivijalan zadatak i može veoma da zavisi od konkretne primene
- Bez obzira na izbor kernela, važno je korektno odrediti vrednosti njegovih parametara – obično postupkom unakrsne validacije

Šta sve može biti kernel?

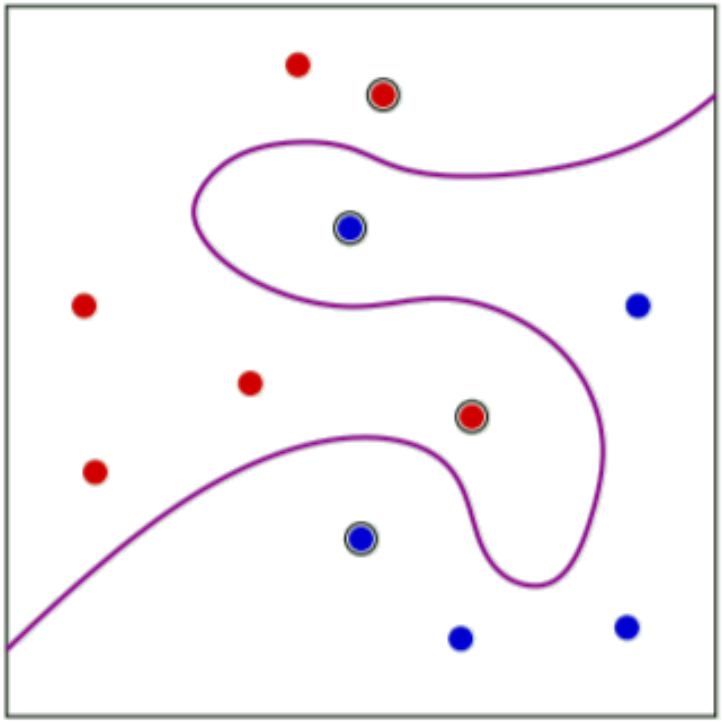
- Do sada smo kernele radi jednostavnosti prikazali kao funkciju sličnosti

- Međutim, nisu sve funkcije sličnosti validni kerneli
  - Validan kernel predstavlja skalarni proizvod u nekom prostoru z. Kako da pokažemo da prostor z postoji (a da ne znamo šta je)?
  - Postoje određena matematička svojstva koja ova funkcija može da ima [2] (Sec. 3.2-3.3)
    - Moraju da zadovolje tehnički uslov koji se zove *Mercer's Theorem*
    - U suprotnom, optimizacija ne mora da bude korektna i može se desiti da dođe do divergencije

- Ali, jednostavna preporuka je
  - Koristite postojeće
  - Ili improvizujte kernel i vidite da li radi

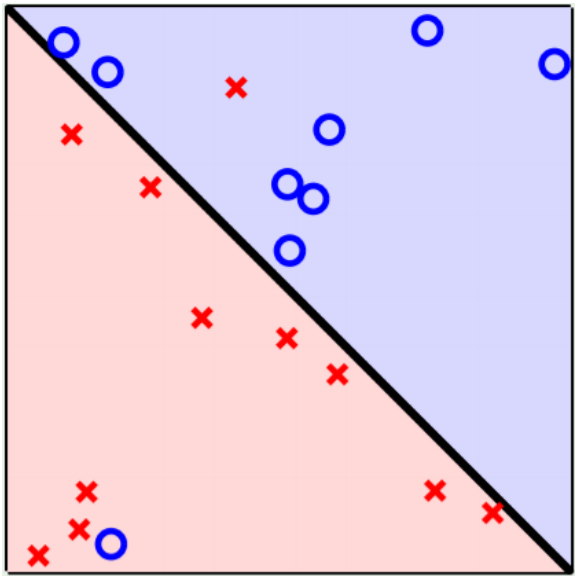
Kernel trik

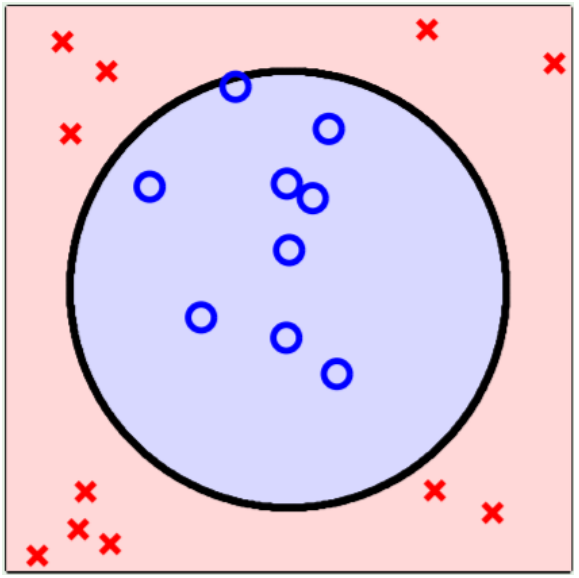
- Potporni vektori su u novom z prostoru
  - Ne moramo da znamo geometriju ovog prostora kako bismo ih identifikovali – znamo koji su primeri u pitanju jer su odgovarajući α_i različi od nule



Šta ako problem nije linearno separabilan?

- Dva tipa neseparabilnih podataka





Neznatno

Outlieri (verovatno greške), ne želimo da idemo u visokodimenzioni prostor da bismo korektno klasifikovali ove tačke

Soft margin

Ozbiljno

Nisu outlieri, problem deluje nelinearno – potrebna nam je nelinearna transformacija

Kerneli

Šta ako problem nije linearno separabilan?

- U praktičnoj primeni, skupovi podataka sadrže oba aspekta (outliere i nelinearnost)

- Zato se kernel trik i *soft margin SVM* često koriste zajedno

SVM parametri

- Potrebno je odrediti:
  - Parametar C
  - Izabrati kernel
  - Odrediti parametre kernela

- Standardan postupak: unarksna validacija ili korišćenje posebnog validacionog skupa