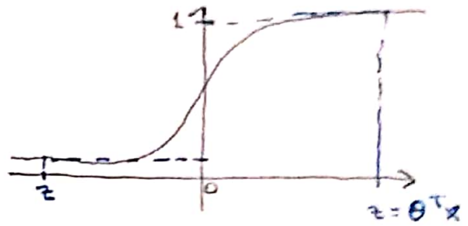
week 7

Support Vector Machine (SVM)

SVM, bazen logistic regresyon veya sinir ağlarına oranla karmaşık doğrusal olmayan işlevleri öğrenmenin daha temiz ve bazen daha güçlü bir yolunu sunar.



logistic regressio:

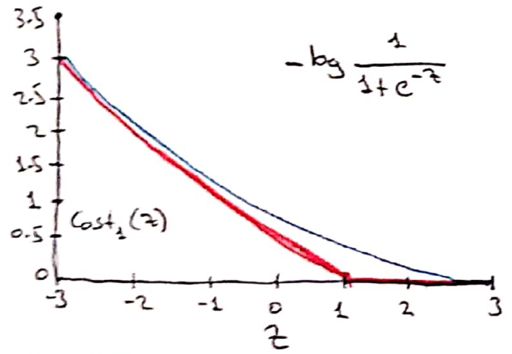
$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Eğer $y=1$ ise, biz $h_\theta(x) \cong 1$ olmasını isteriz. $\theta^T x >> 0$

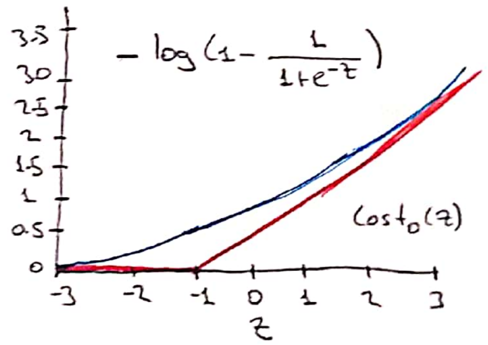
Eğer $y=0$ ise, biz $h_\theta(x) \cong 0$ olmasını isteriz. $\theta^T x << 0$

$$Cost = -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y) \log \left(1 - \frac{1}{1+e^{-\theta^T x}}\right)$$

Eğer $y=1$ ise



Eğer $y=0$ ise



z büyüdükçe $\frac{1}{1+e^{-z}}$ değeri 1'e yaklaşacak ve $\log(1) = 0$ olduğu için cost function değeri 0'a yakın olacaktır.

$$\min_\theta \underbrace{\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log h_\theta(x^{(i)})\right) + (1-y^{(i)})\left(\left(-\log(1-h_\theta(x^{(i)})\right)\right)\right]}_{A} + \underbrace{\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2}_{B}$$

Support vector Machine:

$$A + \lambda B \qquad C = \frac{1}{\lambda}$$

$$CA + B$$

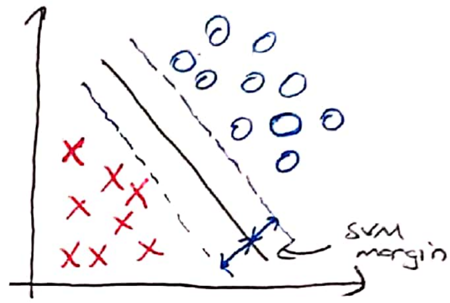
$$\min_\theta C \sum_{i=1}^{m}\left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

Lojistik regresyondaki gibi belli bir örnek sayısına bölerek optimizasyonunu sağlayabileceğimiz gibi vektör destek makinelerinde olduğu gibi C gibi bir katsayıyla çarparakta optimize edebiliriz. $C = \frac{1}{\lambda}$ eşitliğine sahip olarak dizayn edilmiştir.

SVM Hipotez

$$h_\theta(x) \begin{cases} 1 & \text{if } \theta^T x \geq 0, \\ 0 & \text{otherwise} \end{cases}$$

SVM, bazen large margin classifier olarakta adlandırılır. Bunun nedeni sınıflara belli bir uzaklıkta sınıflandırıcı çizgisini çizmesidir. Bu margin optimizasyon sıkıntısından kaynaklanmaktadır.



Large Margin Classification

$$\min_\theta C \underbrace{\sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)}) \right]}_{\to 0} + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

$y^{(i)} = 1$

$\theta^T x^{(i)} \geq 1$

$y^{(i)} = 0$

$\theta^T x^{(i)} \leq -1$

Optimizasyon sonucu örneklerin y değerine göre sonucun yandaki gibi geldiği görülmüştür. Sonuç 0 geldiğinde;

$$\min_\theta \cancel{C \times 0} + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$ (C çok büyük seçilmeli)

Buradaki optimizasyon problemini çözmek için θ'ların küçük seçilmesi gerekir bunun içinde örneklerden uzak bir boundry çizilmelidir. İşte bu nedenle margin oluşur.

$$\min_\theta \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2}(\theta_1^2 + \theta_2^2) = \frac{1}{2}\Big(\underbrace{\sqrt{\theta_1^2 + \theta_2^2}}_{\|\theta\|}\Big)^2 = \frac{1}{2}\|\theta\|^2$$

simple example $\theta_0 = 0$

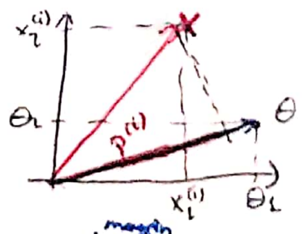
$n = 2$

$\theta^T x^{(i)} \geq 1$ if $y^{(i)} = 1$

$\theta^T x^{(i)} \leq -1$ if $y^{(i)} = 0$

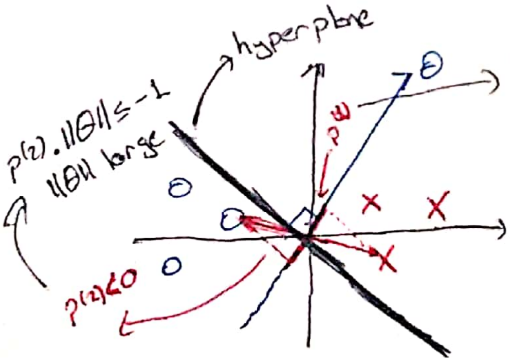
$\theta^T x^{(i)} = ?$

$u^T v$



$\theta^T x^{(i)} = p^{(i)} \cdot \|\theta\|$

$= \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}$

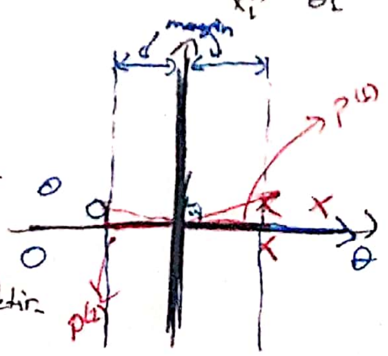


$p^{(1)}.\|\theta\| \geq 1$

$\|\theta\|$ large

Örneklerin izdüşümü arttıkça θ değerleri azalacaktır.

Margin değeri büyüyecektir.

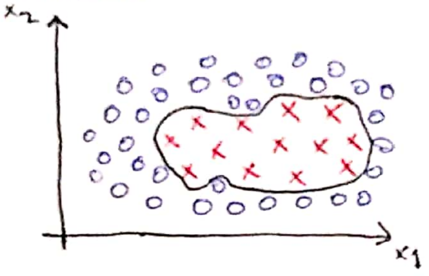


$p^{(1)}.\|\theta\| \geq 1$

$\|\theta\|$ can be smaller.

Kernel 1

Non-linear



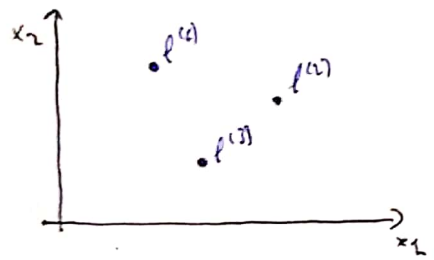
$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \dots \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \dots$$

$$f_1 = x_1,\quad f_2 = x_2,\quad f_3 = x_1 x_2,\quad f_4 = x_1^2,\quad f_5 = x_2^2$$

Yüksek dereceli polinomları hesaplamak oldukça pahalıdır.

Kernel



öklit mesafesi

Given x: $f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$

$f_2 = \text{similarity}(x, l^{(2)}) = \exp\left(-\frac{\|x - l^{(2)}\|^2}{2\sigma^2}\right)$

$f_3 = \text{similarity}(x, l^{(3)}) = \exp\left(-\frac{\|x - l^{(3)}\|^2}{2\sigma^2}\right)$

Yukarıda kullandığımız benzerlik işlemi bir çekirdektir (kernel). Bu çekirdeğe aslında Gauss Kernel denir.

Notation: $k(x, l^{(i)})$

Kernel and Similarity

$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(1)})^2}{2\sigma^2}\right)$$

Eğer $x \cong l^{(1)} \Rightarrow f_1 \cong \exp\left(-\frac{0^2}{2\sigma^2}\right) \cong 1$

Eğer x, $l^{(1)}$'den çok uzaksa $\Rightarrow f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \cong 0.$

Kernel, x'in yer işaretlerinden herhangi birine ne kadar benzediğini ölçmektedir. x, l'ye yakınsa 1'e yakınsar, uzaksa sıfıra yakınsar.

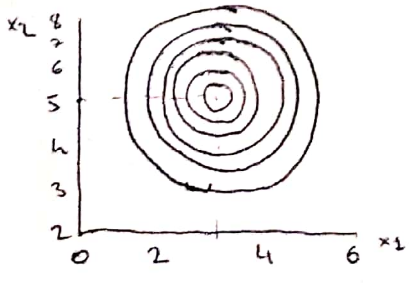
Not: $\|x^{(i)} - l^{(i)}\|^2 = \sum_{k=1}^{n}(x_k^{(i)} - x_k^{(j)})^2$

Generally: $f_{(i)} = \exp\left(-\frac{\|x^{(i)} - l^{(i)}\|^2}{2\sigma^2}\right)$

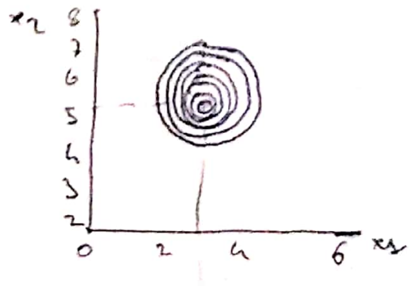
Example

$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$, $f_1 = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$

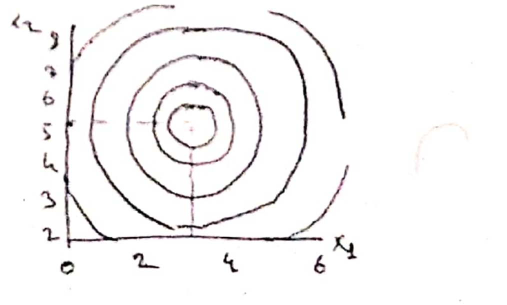
$\sigma^2 = 1$ ise



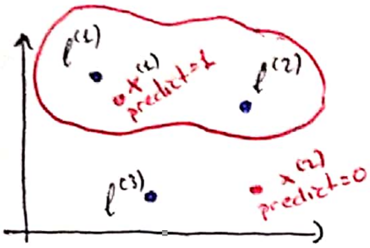
$\sigma^2 = 0.5$ ise



$\sigma^2 = 3$ ise



Not: σ gauss değişkenidir. Azaldıkça kontur grafiği daralır ve 0'a daha hızlı inilir. Arttıkça ise kontur grafiği genişler ve 0'a çok yavaş inilir.



$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$$

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

$x^{(1)}$ için:

$f_1 \cong 1 \,,\ f_2 \cong 0 \,,\ f_3 \cong 0$

$= \theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0$

$= -0.5 + 1 = 0.5 \geq 0 \implies 1$ predict

$x_2^{(1)}$ için:

$f_1, f_2, f_3 \cong 0$

$= \theta_0 + \theta_1 \times 0 + \ldots \cong -0.5 < 0 \implies 0$ predict

Yer işaretlerini nasıl belirleyeceğimiz konusuna gelirsek eğitim örneklerimize yakın noktalar seçeceğiz. (Yani aynısı denilebilir)

SVM with Kernels

Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}$

- x girdisi verildiğinde;

  $f_1 = \text{similarity}(x, l^{(1)})$

  $f_2 = \text{similarity}(x, l^{(2)})$

  $\vdots$

  $$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \qquad f_0 = 1$$

- Eğitim seti için $(x^{(i)}, y^{(i)})$:

$$x^{(i)} \rightarrow \begin{aligned} f_1^{(i)} &= \text{sim}(x^{(i)}, l^{(1)}) \\ f_2^{(i)} &= \text{sim}(x^{(i)}, l^{(2)}) \\ &\vdots \\ f_m^{(i)} &= \text{sim}(x^{(i)}, l^{(m)}) \end{aligned}$$

$\leftarrow f_i^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = \exp\left(-\frac{0^2}{2\sigma^2}\right) \cong 1$ $\quad l^{(i)} = x^{(i)}$

$x^{(i)} \in \mathbb{R}^{n+1}$ (or $\mathbb{R}^n$)

$$f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \qquad f_0^{(i)} = 1$$

Ortaya çıkan $f^{(i)}$ vektörü yeni eğitim örneğimi temsil edecek özellik vektörüdür.

Hipotez: x verildiğinde, hesaplanacak features $f \in \mathbb{R}^{m+1}$

$\rightarrow$ predict "$y=1$" if $\theta^T f \geq 0 \implies \theta_0 f_0 + \theta_1 f_1 + \theta_2 f_2 + \ldots$

Ağırlıkları hesaplayabilmek için SVM algoritmasının cost function'ını kullanırız.

$$\min_{\theta} C \sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{n=m} \theta_j^2$$

$$\theta . \theta^T \Leftarrow \|\theta\|$$

Not: Kernel'i lojistik regresyon gibi algoritmalarda kullanmama nedenimiz oldukça yavaş çalışmasıdır.

SVM Parameters

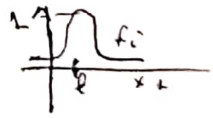
$C (= \frac{1}{\lambda})$ C Büyükse: lojistik regresyondaki küçük lambda değerine yani az regularization kullanmaya denk gelir.
• Düşük bias, yüksek variance.

C Küçükse: lojistik regresyondaki büyük lamda kullanmaya denk gelir.
• Yüksek bias, düşük variance.

C çok büyük seçilirse overfit durumuna düşebilir. Küçük seçilirse underfit durumuna düşebilir.

σ^2 σ^2 Büyükse: f_i feature'ları 0'a yavaşça iner. Yüksek bias, düşük variance.

σ^2 Küçükse: f_i feature'ları 0'a hızlıca iner. Düşük bias, yüksek variance.



Not: SVM algoritması, belirli bir optimizasyon problemi ortaya çıkarır. Bu nedenle parametrelerin θ'sını kendi yazdığınız yazılımla çözülmesi önerilmemektedir.

Eğer yazmak zorunda kalırsanız;
• C parametresini seçmelisiniz.
• Kernel seçmek
- Kernel seçmeyedebilirsiniz. Buna "lineer kernel" denir.
predict "y=1" if $\theta^T x \geq 0$
+ Lineer kernel küçük veri setiniz varsa seçilebilir.
- Gauss Kernel
+ Büyük bir veri setiniz varsa seçmek mantıklıdır.

Not: Çok farklı ölçeklerde feature'larınız varsa, Gauss Kernel'ini kullanmadan önce feature scaling yapmak önemlidir.

Hangi kernel'i seçerseniz seçin, hepsinin teknik bir koşulu karşılaması gerekir. Buna Mercer Teoremi denir. İhtiyaç duymamızın nedeni optimizasyon hilelerine sahip olmasıdır. Yaptığı şey SVM paketlerinin optimizasyon sınıflarını kullanabilmesidir.

Other Kernel

- Polynomial Kernel

  Çok fazla kullanılmaz. $k(x,l) = (x^T l)^2$

  Diğer türleri => $(x^T l)^3$, $(x^T l + 1)^3$, $(x^T l + 5)^4$

  Genel yapısı => $(x^T l + \text{constant})^{\text{degree}}$

  Genelde kötü performans gösterir.

  Negatif olmayan veriler için kullanılır.
- More esoteric: String kernel, chi-square kernel, histogram kernel

Not: Çoklu sınıflandırma için lojistik regresyondaki all in one mantığı kullanılabilir.

Logistic Regression vs. SVM

n => features   m => training set size

- Eğer özellik sayınız (n), eğitim setinizin boyutundan (m) büyükse lojistik regresyon ya da lineer kernel SVM kullanılmalıdır. Bir e postanın spam olup olmadığını metine göre sınıflandırma örnek verilebilir.

- n küçük (1-1000), m ortalama (10-10000) ise SVM Gauss kernel kullanılmalıdır.
- n küçük (1-1000), m büyük (50000+) ise SVM Gauss kernel performans sıkıntısı yaşatır. Bunun için daha fazla özellik ekleyerek lojistik regresyon ya da kernel'sız SVM kullanılmalıdır.

Not: SVM'in optimizasyon problemi dış bükey optimizasyon problemidir.

Optimize edilmiş bir SVM paketi sinir ağından daha iyi ve hızlı çalışabilir.