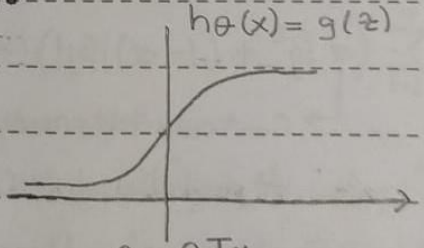
Kernel Methods (SVM)

Here are the methods of Kernel in Machine learning mentioned below: 1) PCA
2) SVM
3) Spectral clustering

SVM (Support Vector Machines)

Logistic Reg. hatırlayalım:

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



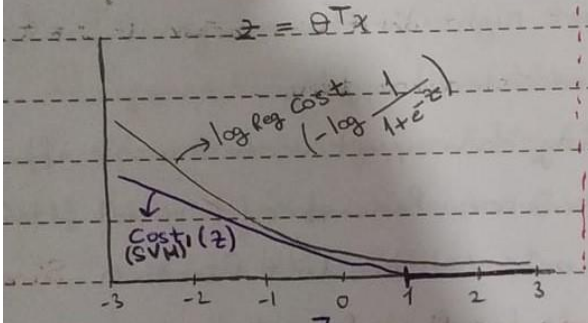
if $y=1$, we want $h_\theta(x) \approx 1$, $\theta^T x \gg 0$

if $y=0$, we want $h_\theta(x) \approx 0$, $\theta^T x \ll 0$

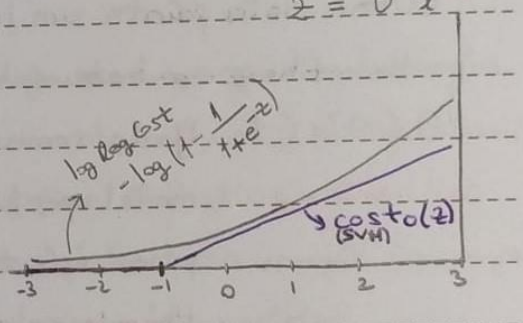
Cost func Logistic Reg: $= -(y \log h_\theta(x) + (1-y)\log(1-h_\theta(x)))$

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

* if $y=1$ (we want $\theta^T x \gg 0$)



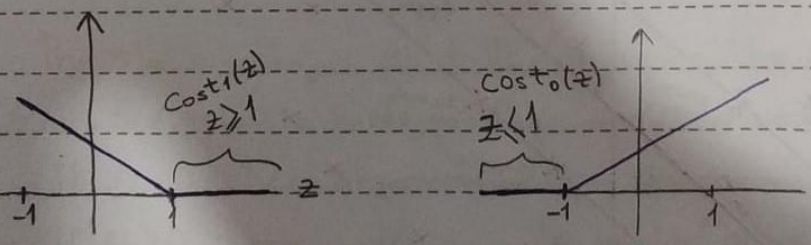
* if $y=0$ (we want $\theta^T x \ll 0$)



Cost Func SVM: Hinge loss

$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}\,\text{Cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\,\text{Cost}_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

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



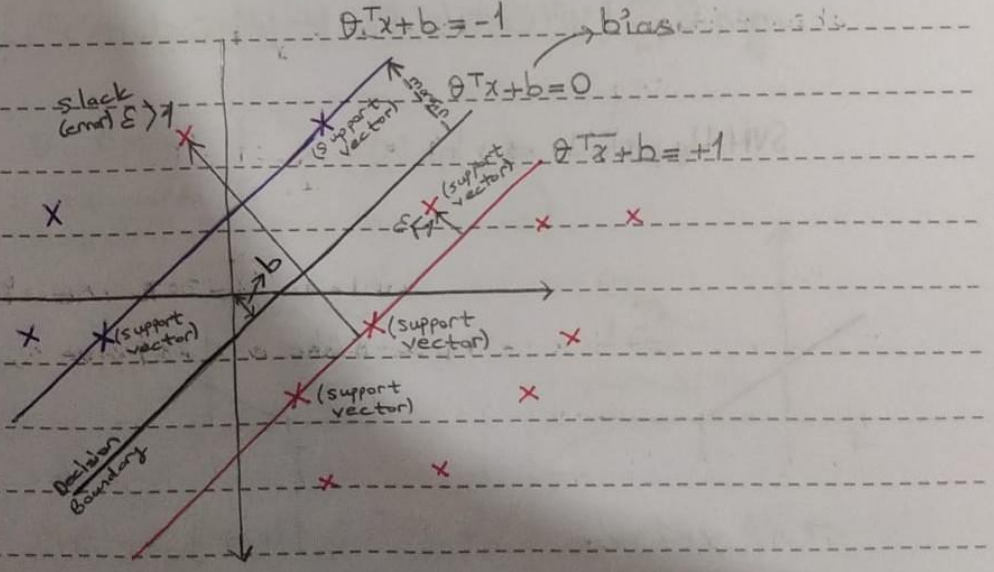
→ if $y=1$, we want $\theta^T x \geq 1$ (not just ≥ 0)

→ if $y=0$, we want $\theta^T x \leq -1$ (not just < 0)

↳ Eğer bir gözlemi doğru sınıflandırıyorsak ve bu gözlem margin'den en az 1 (basically away from decision boundry) uzak ise Cost = 0 olur. Ancak gözlem margin içerisinde veya boundary'nin yanlış tarafında ise Cost ≠ 0 olur.

→ SVM gives some punishment to both incorrect predictions and those close to decision boundary ($0 < \theta^T x < 1$). That is how we call them Support Vectors.

→ When data points are just right on the margin $\theta^T x = 1$, when they are between decision boundary and margin $0 < \theta^T x < 1$. Yani; removing non-support vectors won't affect model performance. Remember!! Model fitting process is to minimize the Cost function. Since there is no cost for Non-Support vectors at all, total value of cost function won't be changed by adding or removing them.



→ Observations on the margin and within the Soft Margin are called Support Vectors.

Şimdiye dek neler öğrendik?

1) Maximal Margin classifier, linear olarak ayrılabilen sınıflar ayırmak için kullanılır. Bunu yaparken öyle bir boundary bulur ki largest margin olur.

2) Separating hyperplane olmadığı zaman, maximal margin classifier'ın general bir formu olan Support vector classifier kullanılır. Support Vector Classifier, Soft Margin classifier olarak da bilinir. Bu durumda, bazı gözlemlerin margin'in veya hyperplane'in yanlış tarafında donarına izin veririz.

$\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ are slack variables, that allow individual observations to be on the wrong side of the margin or hyperplane.

→ $\varepsilon_i \geq 0$;

The slack variable ε_i tells us where the ith observation is located, relative to hyperplane and relative to the margin.

if $\varepsilon_i = 0$ then observation is on the correct side of margin.

if $\varepsilon_i > 0$ then " " " " wrong " " "

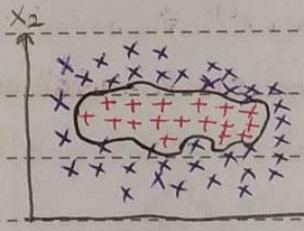
if $\varepsilon_i > 1$ then " " " " " " " hyperplan

* Marjin üzerinde olan veya kendi sınıflarına göre marjin yanlış tarafında olanlar Support Vector'ler olarak bilinir.

3) Support Vector Classifier, boundary between two classes linear ise kullanılır. Ama çoğu kez Non linear class boundary ile karşılaşırız. Yani çoğu zaman Linear Boundary sınıfları ayırmamıza yetmez. Bu durumda Support Vector Machine (SVM) kullanılır. SVM, is extension of the Support Vector Classifier that results from enlarging the feature space in a specific way, using Kernels.

↓ Gelin Kernels neymiş bakalım.

Non-Linear Decision Boundary:



$\rightarrow \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \dots \geq 0$

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \dots + \theta_5 x_2^2 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Yani burada yeni feature'larımız ne oldu?

$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \dots$ } f_i is new feature

$f_1 = x_1$, $f_2 = x_2$, $f_3 = x_1 x_2$ ...

↳ Bu şekilde yaptığımızda computationally expensive'dir

Daha iyi bir çözüm var mıdır? Bu features'ları bulmak için...

* Var. Kernel (similarity) functions

SVM with Kernels:

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

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

→ (landmarks'lar direk training example'lar olacak olsun)

Given example x:

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

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

$\vdots$

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

For training example $(x^{(i)}, y^{(i)})$:

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

$x^{(i)} \rightsquigarrow f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)})$ (Gaussian Kernel)

$\vdots$

$\leftarrow f_1^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = \exp\left(-\frac{0}{2\sigma^2}\right) = 1$ (feature'lardan biri 1 olacak, çünkü kendisi ile benzerliğine baktık)

$f_m^{(i)} = \text{sim}(x^{(i)}, l^{(m)})$

$f^{(i)} = \begin{bmatrix} f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \Rightarrow$ New feature Vector

Cost func: $\min_\theta C \sum_{i=1}^{m} \left[y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{n}$

($\theta^T f^{(i)}$)

~~$\theta^T x^{(i)}$~~

* Predict $y = 1$ if $\theta^T f \geq 0$; $f \in \mathbb{R}^{m+1}$, $\theta \in \mathbb{R}^{m+1}$

→ $\theta_0 f_0 + \theta_1 f_1 + \dots + \theta_m f_m$

* Landmarks'lar training observations ile aynı gözlemler olur.

SVM parameters (Andrew Ng):

C ve σ^2, sklearn kısmında onlatılacak.

SVM General:

1) No Kernel (Linear Kernel)

Predict $y=1$ if $\theta^T x \geq 0$

2) Gaussian Kernel

$f_i = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$, where $l^{(i)} = x^{(i)}$

Note:

→ Perform feature scaling before using Gaussian Kernel.

Choose σ^2

3) Other Choices of Kernel

Polynomial ≈> $sim(x, l) = k(x, l) \Rightarrow (x^T l)^2$, $(x^T l)^3$, $(x^T l + 1)^3$ ...

RBF ≈> $k(x, l) = \exp(-\gamma \|x - l^{(i)}\|^2)$

SVM parameters:

1) C, default=1 (like log Reg, smaller values specify stronger regularization)

C behaves as a Regularization Parameter in SVM.

- A smaller C emphasizes the importance of ξ (slack variables).
- A larger C decrease the importance of ξ (slack variables).

→ Small values of C result in a wider Margin.

→ Large values of C result in Smaller Margin.

$C = \frac{1}{\lambda}$ (similar to regularization)

Large C: Lower Bias, High variance

Small C: Higher Bias, low variance

- Logistic Regresyondan farklı olarak, λ regularization terimi önünde kullanılamaz. SVM'de C'yi fit term'ün önünde kullanır. (Bu bir değiştirmez aynı şey yani)

- Logistic Regression:

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

- Support Vector Machine:

$$\left(C = \frac{1}{\lambda}\right) \leftarrow \min_{\theta} C \sum_{i=1}^{m}\left[y^{(i)}\,\text{Cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\,\text{Cost}_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

fit Term (cost coming from training set) — Regularized Term

→ fit term emphasizes fit the model very well by finding optimal coefficients, and the regularized term controls complexity of the model by constraining the large value of coefficients.

There is a trade off between ① fitting the model well on training dataset and ② the complexity of the model that may lead to overfitting, which can be adjusted by changing the value of λ or C.

2) kernel, default = "rbf"

Note: rbf kernel is also called as Gaussian kernel.

3) degree, default = 3

Degree of the polynomial kernel function.

4) gamma, default = "scale"

Note:

$$\text{Gaussian RBF} = \exp\left\{\frac{-\|x - l^{(i)}\|^2}{2\sigma^2}\right\}$$

$$\text{Gaussian RBF} = \exp\left\{-\gamma \|x - l^{(i)}\|^2\right\} \quad \gamma = \frac{1}{2\sigma^2}$$

* High gama means more Eğrilik (model heavily impacted by support vectors)

Low " " less Eğrilik (model behave like linear SVM)