- Classification Problem -

- Variable y that we want to predict is discrete valued.
- Bu problemin çözümü için logistic regression algoritması kullanılacak.

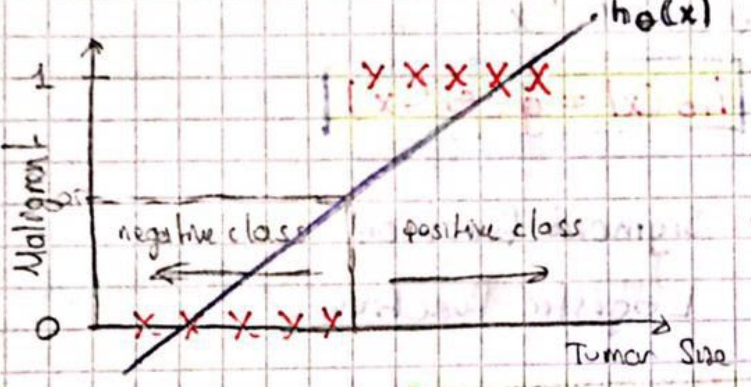
Exs: • Email: Spam / Not Spam • Online Transactions: Fraudulent (Yes/No)? • Tumor: Malignant / Benign?

absence of something

$y \in \{0, 1\}$ 0: "Negative Class" (e.g. benign tumor)
1: "Positive Class" (e.g. malignant tumor)

→ Şimdilik y sadece 0 ve 1 olabilir. bu durumla ilgileniriz. İleride Multiclass Classification Problems göreceğiz, $y \in \{0,1,2,3\}$ olabilecek.

Binaryclass Classification ($y \in \{0,1\}$)

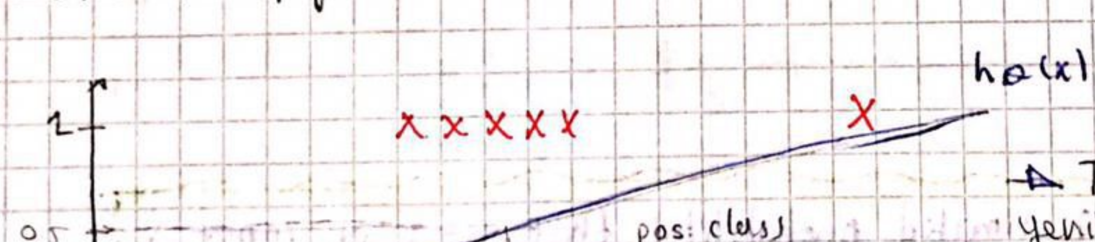


→ Classification problem olmasına rağmen burada linear regression modelini kullanabiliriz.
→ Bir treshold belirleriz ve hypothesis $h_\theta(x) > 0.5$ ise $h_\theta(x) = 1$ deriz, küçükse $h_\theta(x) = 0$ deriz.

→ $\boxed{h_\theta(x) = \theta^T \cdot x}$

Treshold classifier output $h_\theta(x)$ at 0.5:
If $h_\theta(x) \geq 0.5$, predict "$y=1$"
If $h_\theta(x) < 0.5$, predict "$y=0$"

Bu örnek için linear regression modeli işliyor ama biraz modifiye edersek acaba hala kullanışlı olacak mı?



→ Training set'e yeni eklenen data $h_\theta(x)$'i saptırdı ve aynı treshold için, hypothesis artık doğru sonuç vermiyor.

Yani linear regression modeli classification için kullanmak yanlış. İlk örnekte şansına çalıştı.

Classification $y = 0$ or 1

$h_\theta(x)$ can be > 1 or < 0 ... for Linear Regression model

→ Tahmin edilecek output 0 ve 1 olsa dahi hypothesis sonuçları 0'dan küçük ve 1'den büyük olabiliyor bu anlamsız.

→ ileriki bölümlerde Logistic Regression modeline gireceğiz.

Logistic Regression için $0 \le h_\theta(x) \le 1$

↳ İsmin içinde regression olması kafa karıştırmasın bu bir classification algorithm.

Logistic Regression

→ $0 \le h_\theta(x) \le 1$ istiyoruz.

→ Linear regression için $h_\theta(x) = \theta^T \cdot x$ idi.

→ Logistic Regression için $h_\theta(x) = g(\theta^T \cdot x)$

where

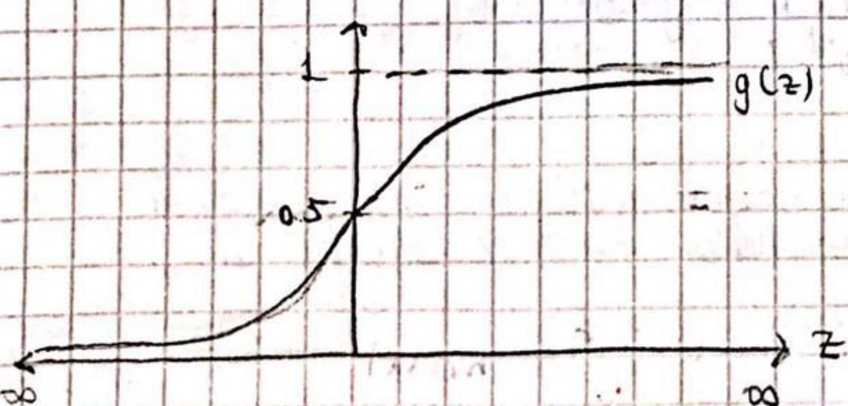
$$g(z) = \frac{1}{1+e^{-z}}$$

→ Sigmoid function
→ Logistic function

synonyms

z is real number

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



→ θ parametrelerini fit etmeyi daha sonra göreceğiz.

$h_\theta(x)$ = Estimated probability that $y=1$ on input x

Ex: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{Tumor Size} \end{bmatrix}$ and $h_\theta(x) = 0.7$

→ Tell patient that 70% change of tumor being malignant.

→ Verilen tumor size için $y=1$ olma olasılığı 70%

→ $\boxed{h_\theta(x) = p(y=1 \mid x;\theta)}$ "probability that $y=1$ given x, parametrized by θ"

→ $p(y=0 \mid x;\theta) = 1 - p(y=1 \mid x;\theta) = 1-0.7 = 0.3$

Ekstra açıklama: Regression problem için, linear regression modelini kullandık ve bir $h_\theta(x)$ atadık. Daha sonra training set ile hypothesis arası farkları minimize ettik bunu da cost function J'i, Gradient Descent algorithm ile veya analitik olarak minimize ederek yaptık. Sonuç olarak elde edilen $h_\theta(x)$'e girilen rasgele bir x bize en gerçekçi output y'yi verdi.

Classification'dan ise, Logistic Regression modeli kullanarak hypothesis oluşturuldu $h_\theta(x)$ ve ile cost function minimize edilerek en iyi $h_\theta(x)$'e ulaşıldı. Fakat burada $h_\theta(x)$ in sonucu $y=1$ olma olasılığını veriyor.

Logistic Regression

- $h_\theta(x) = g(\theta^T.x) = p(y=1 \mid x:\theta)$
- $g(z) = \frac{1}{1+e^{-z}}$

$$\boxed{\begin{array}{l} \text{Suppose predict "}y=1\text{" if } h_\theta(x) \geq 0.5 \\ \qquad\quad \text{predict "}y=0\text{" if } h_\theta(x) < 0.5 \end{array}}$$

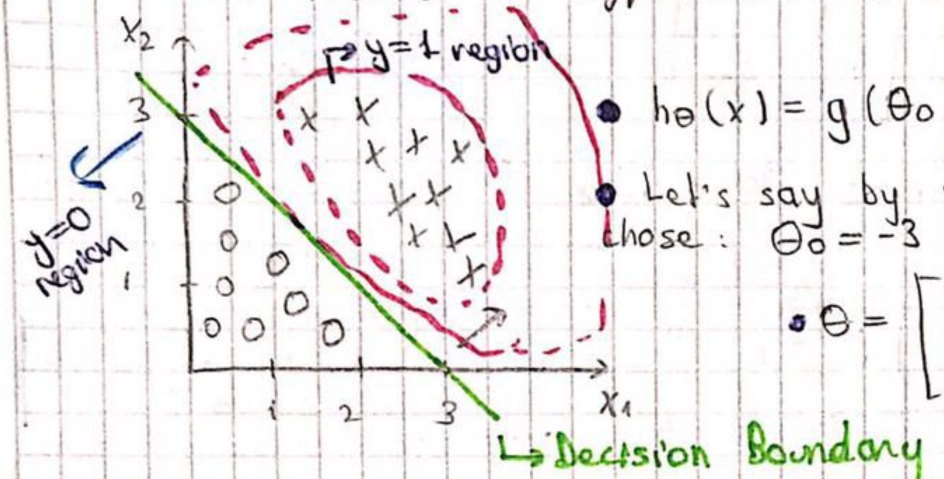
Olay şu $\theta^T x \geq 0$ için $g(\theta^T.x) = h_\theta(x) \geq 0.5$ oluyor. Yani $y=1$ olma olasılığı ≥ 0.5 oluyor o zaman $y=1$ prediction'ı yapabiliriz

$$\boxed{\begin{array}{l} g(z) \geq 0.5 \quad \text{when } z \geq 0 \\ \qquad \text{since} \\ h_\theta(x) = g(\theta^T.x) \geq 0.5 \quad \text{when } \theta^T.x \geq 0 \end{array}}$$

$\theta^T.x = 0$ bir çizgi verir bu çizgi bize $h_\theta(x)=0.5$'i veren x ve θ değerlerini gösterir.

Yani predict $y=1$ if $\theta^T.x \geq 0$
predict $y=0$ if $\theta^T.x < 0$

Ex: Let's suppose we have a training set as below and we have obtained an hypothesis as below:



- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
- Let's say by using an algorithm we chose: $\theta_0 = -3$ $\theta_1 = 1$ $\theta_2 = 1$
- $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

→ Verilen hypothesis'i kullanarak, bu $h_\theta(x)$'in nerelerde $y=1$ nerelerde $y=0$ prediction'ı yaptığını bulalım!

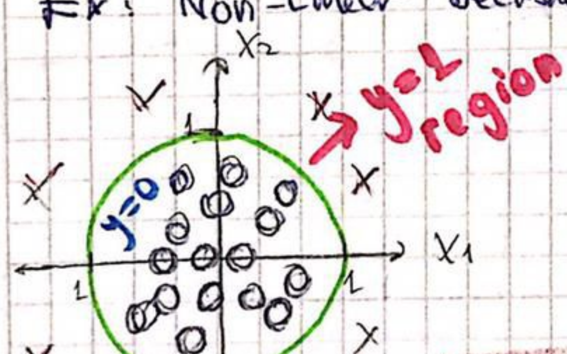
☆ Yada nerelerde $h_\theta(x) = 0.5$ olduğunu bulalım!

We know: • Predict "$y=1$" if $\boxed{\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0}$

↳ $x_1 + x_2 \geq 3$ için $h_\theta(x) \geq 0.5$, predict $y=1$.

☆ Decision Boundary yalnızca $h_\theta(x)$'e bağlıdır, training set'e bağlı değil.

Ex: Non-Linear Decision Boundaries

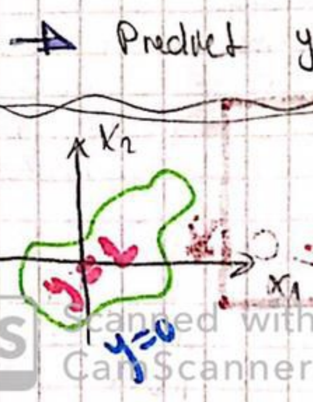


Assume our hypothesis:

- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$
- Let's say $\theta = [-1 \ 0 \ 0 \ 1 \ 1]^T$

- Decision boundary'i bulalım:

→ Predict $y=1$ when $-1 + x_1^2 + x_2^2 \geq 0$ => $\boxed{x_1^2 + x_2^2 \geq 1}$



- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 \cdot x_2 + \theta_5 \cdot x_1^2 \cdot x_2^2 + \theta_6 \cdot x_1^3 \cdot x_2 + \dots)$

☆ Farklı $h_\theta(x)$'ler için daha farklı ve complex Decision Boundaries elde edilebilir.

~ Logistic Regression – Cost Function ~

- Training Set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots (x^{(m)}, y^{(m)})\}$ m examples
- $x \in \begin{bmatrix} x_0 \\ x_1 \\ \ldots \\ x_n \end{bmatrix}$ • $x_0 = 1$ • $y \in \{0, 1\}$
- $h_\theta(x) = g(\theta^T \cdot x) = \dfrac{1}{1 + e^{-\theta^T \cdot x}}$

Question: How to choose parameters θ ?

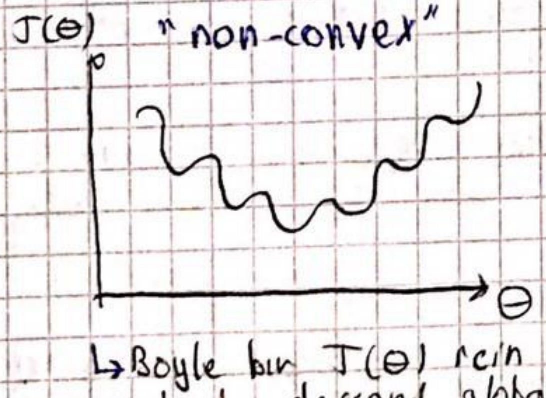
→ Linear Regression için $\boxed{J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} h_\theta(x^{(i)}) - y^{(i)}\right)^2}$

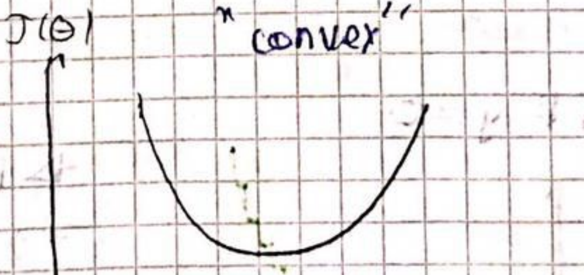
→ Let's say $\boxed{\text{cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}$

↳ $\boxed{\text{cost}(h(x), y) = \frac{1}{2}(h_\theta(x) - y)^2}$ (Simplified)

→ Logistic Regression için bu cost function'ı kullanırsak J non-convex bi form olur yani bir çok local optima olur.

↳ Yani $h_\theta(x) = \dfrac{1}{1 + e^{-\theta^T x}}$ 'i $J(\theta) = \dfrac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ de yerine koyarsak J linear regression'da olduğu gibi convex olmaz şöyle olur:

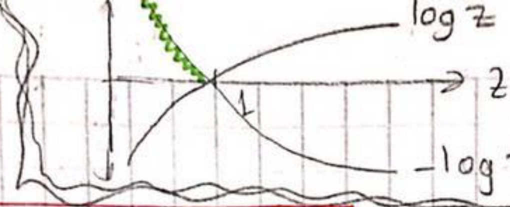




↳ Böyle bir $J(\theta)$ için gradient descent global minimumu bulamayabilir.

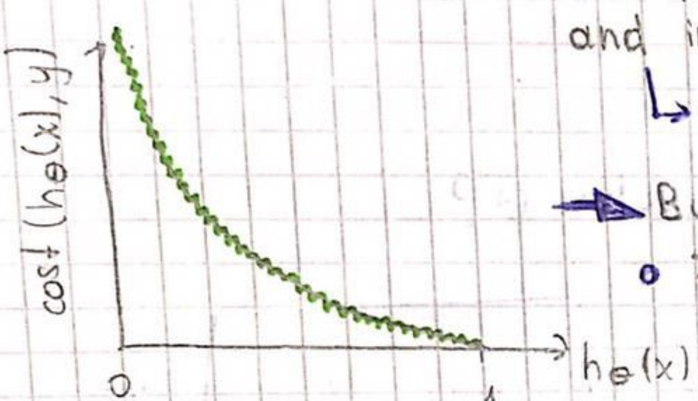
(✱) $h_\theta(x) = \dfrac{1}{1 + e^{-\theta^T x}}$ logistic function very nonliner olduğu için eski cost function tanımımız non-convex oluyo. Yeni bir cost function tanımlayarak $J(\theta)$'yı convex yapacağız.

Logistic Regression Cost Function:



$$\boxed{cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}}$$

If $y=1$



→ If hypothesis $h_\theta(x)$ predicts $y=1$ and indeed $y=1$ then cost $=0$

↳ Cost $=0$ if $y=1$, $h_\theta(x)=1$

→ But as $h_\theta(x) \to 0$, cost $\to \infty$

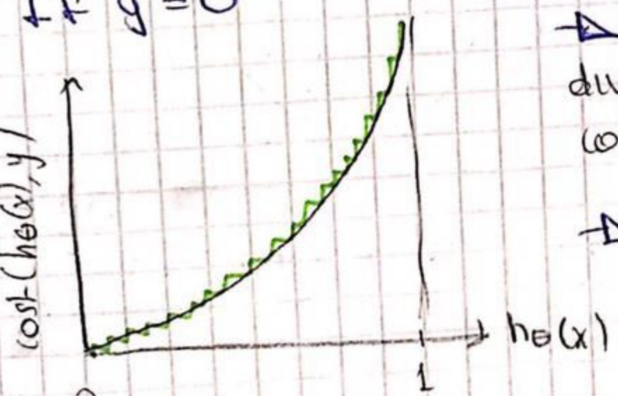
- It means if $h_\theta(x)=0$ (predict $P(y=1|x;\theta)=0$), but $y=1$ we will penalize learning algorithm by a very large cost

→ Logistic regression için $h_\theta(x)$'in 0 ile 1 arasında olduğunu unutma çünkü $h_\theta(x)$ aslında $y=1$ olma olasılığını veriyor.

→ Olay şu, cost'u hesaplarken önceden y değeri fark etmeksizin kare fark alıyorduk. Şimdi $y=1$ ise farklı cost, $y=0$ ise farklı.

→ $y=1$ iken $h_\theta(x)$ bize 1 veriyorsa süper çalışıyor bu $x^{(i)}$ için cost 0! Fakat $y=1$ iken $h_\theta(x)=0$ veriyorsa, yani tümör malignant olduğunu vermemize rağmen hypothesis bunun olasılığı 0 diyorsa cost $\to \infty$ tamamen yanlış çalışıyor bu $h_\theta(x)$, θ'ların değişmesi lazım,

If $y=0$



→ Öncekinin reversi, $h_\theta(x)=0$ diyorsa ve $y=0$ ise sıkıntı yok, cost $=0$

→ Fakat $y=0$ iken $h_\theta(x)=1$ verirse cost $\to \infty$, θ'ların değişmesi şart.

Bu tek bir data line için yani bir $x^{(i)}$ için ve bir $y=0$ or $y=1$ değeri için cost hesaplar.
$J(\Theta)$'yı bulmak için tüm bunları toplam ve $\frac{1}{m}$'e bölen.

~ Simplified Cost Function and Gradient Descent ~

Cost Function for Logistic Regression

$$J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{cost}(h_\Theta(x^{(i)}), y^{(i)})$$

$$\mathrm{cost}(h_\Theta(x), y) = \begin{cases} -\log(h_\Theta(x)) & \text{if } y=1 \\ -\log(1-h_\Theta(x)) & \text{if } y=0 \end{cases}$$

Note: $y = 1$ or 0 Always

ikisi aynı şey!

✲ Şimdi $y=0$ ve 1 için ayrı ayrı iki cost tanımlamak yerine bunları birleştirerek simplify edeceğiz:

$$\mathrm{cost}(h_\Theta(x), y) = -y \cdot (\log(h_\Theta(x)) - (1-y)\log(1-h_\Theta(x))$$

why this cost function? it can be derived from statistics using the principle of maximum likelihood estimation. And it is convex!

Thus

Burada log → ln!

$$J(\Theta) = \frac{-1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_\Theta(x^{(i)})) + (1-y)\log(1-h_\Theta(x^{(i)}))\right]$$

↳ Bu fonksiyon logistic regression için convex olacaktır!

✲ Cost function $J(\Theta)$'yı yazdık, artık amacımız bu $J(\Theta)$'yı minimize edecek Θ parametrelerini bulmak!

↳ $\min_\Theta J(\Theta)$: Get $\Theta = \begin{bmatrix} \Theta_0 \\ \vdots \\ \Theta_n \end{bmatrix}$

✲ After obtaining Θ, then we have $h_\Theta(x) = \frac{1}{1+e^{-\Theta^T x}}$ so we can make a prediction for given new $x^{(i)}$.

$h_\Theta(x) = P(y=1 \mid x;\Theta)$ olduğunu unutma

✲ Cost function $J(\Theta)$'yı minimize etmek için yine gradient descent kullanırız sonuçta bu yine convex bir function. Mantık aynı!

→ W3 3. videoda Advanced Optimization teknikleri gösteriyor α'yı seçmeden daha hızlı converge eden complex methodlar.

Gradient Descent

Want $\min_\theta J(\theta)$:

Repeat {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ Aynı!

}

→ Simultaneously update all θ_j.

$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$ Önceki ile birebir aynı!

$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$ Simultaneously update all θ_j

→ $\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix}$ hepsi için simultaneously update

✯ Sonuç olarak gradient descent algorithm eskisi ile birebir aynı fakat $h_\theta(x)$ değişti!

✯ Önceden gradient descent'i monitor ederek converge ettiğini görmekten bahsetmiştik aynısı geçerli. $J(\theta)$'yı iteration'a göre plot edersek hep azalması lazım.

$$\theta := \theta - \alpha \frac{1}{m} \sum_{i=1}^{m} [h_\theta(x^{(i)}) - y^{(i)}] \cdot x^{(i)}$$

vector formda

✯ Feature scaling'i de logistic regression'da daha hızlı bir convergence için kullanabiliriz.

-Multiclass Classification-

$y=1$ $y=2$ $y=3$ $y=4$

Examples: Email foldering/tagging: Work, Friends, Family, Hobby

Şimdiye kadar binaryclass classification gördük, $y=1$ ya da $y=0$ olmak zorundaydı. Burada $y=0,1,2,3,...$ olabilir.

$y=1$ $y=2$ $y=3$

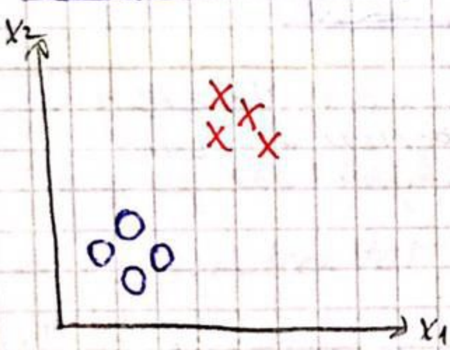
Example: Medical Diagnosis: Not ill, Cold, Flu

$y=1$ $y=2$ $y=3$ $y=4$

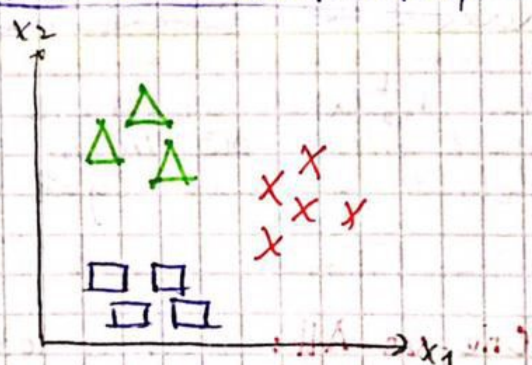
Example: Weather: Sunny, Cloudy, Rain, Snow

y can take small number of discrete values. $y=0$'dan da başlayabilir, önemi yok.

Binary Classification



Multi-class classification



? How to get a learning algorithm?

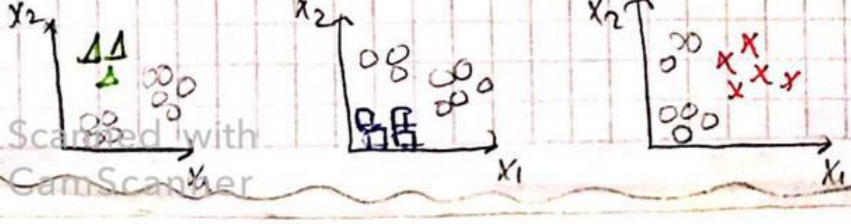
→ Binary için, logistic regression modelini kullanırız ve belki bir çizgi çekeriz, decision boundary olarak. Bir tarafı positive class $y=1$, bir tarafı negatif class $y=0$ olur. Bu line'da $h_\theta(x)=0.5$'i yani $\theta^T \cdot x=0$'ı temsil eder.

"One vs All Classification" adında bir idea kullanarak eski modeli alarak multi-class classification için de çalışır hale getiririz.

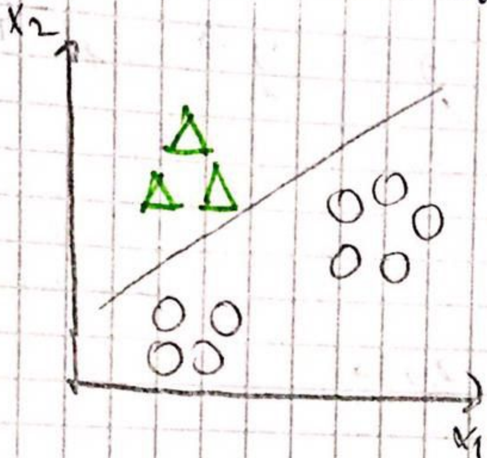
Nasıl Çalışır: Yukarıdaki gibi bir training set olsun.

Class 1: △ ← $y=1$
Class 2: □ ← $y=2$
Class 3: X ← $y=3$

Yapılacak şey, training set'i alıp, 3 ayrı binary classification problemine dönüştürmek!



Önce Class 1'i ele alalım   class 1 vs all:



* Class 1 positive class oldu, digerleri negative class,

* $h_\theta^{(1)}(x)$ → classifier: Bunu training set ile egitecez. assign edicez ve hep yaptigimizi yapicaz ve mesela sekildeki gibi bir decision boundary elde edecegiz.

Sonra Class 2 icin: Yeni bir logistic regression classifier $h_\theta^{(2)}(x)$ fit ederiz.

Sonra Class 3 icin: $h_\theta^{(3)}(x)$

---

* We have fit three classifiers for $i = 1, 2, 3$.

$$\boxed{h_\theta^{(i)}(x) = P(y = i \mid x; \theta) \qquad (i = 1, 2, 3)}$$

Yani $h_\theta^{1}(x)$: girilen x datası için ve θ'lar için $y = 1$ olma olasılığını veriyor. Digerleri de aynı sekilde $y = 2$ ve $y = 3$ olma olasılığını veriyor.

One-vs-All: Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$.

On a new input x, to make a prediction, pick the class i that maximizes

$$\max_i h_\theta^{(i)}(x)$$

Yani en yüksek olasılıklıyı seçeriz.

W3 - Advanced Optimization Algorithms

→ Bunları kullanarak logistic regression'ın GDA ile çok daha hızlı çalışmasını ve large scale problemlere rahat uygulanabilmesini sağlar.

→ GDA'nın dışında başka optimization algorithms de var:

- Conjugate gradient
- BFGS
- L-BFGS

+ No need to manually pick α
+ often faster than gradient descent
− More complex

Example:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

Let's apply an advanced opt. algorithm to minimize $J(\theta)$:

```
function [jVal, gradient] = costFunction(theta)
    jVal = (theta(1)-5)^2 + (theta(2)-5)^2;
    gradient = zeros(2,1);
    gradient(1) = 2*(theta(1)-5);
    gradient(2) = 2*(theta(2)-5);
```

NOW we can call adv. opt. function: fminunc:

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] = fminunc(@costFunction,
        initialTheta, options);
```

Önce üst taraftaki function'a end ekle costFunction ismi ile kaydet! Sonra aşağıdaki kısmı başka mfile'a yaz '100' yerine int istiyor direkt 100 yaz. help fminunc da yardımcı olabilir.

Sonuçta Logistic Regression için:

$$theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \begin{matrix} \text{theta(1)} \\ \text{theta(2)} \\ \\ \text{theta(n+1)} \end{matrix}$$

```
function [jVal, gradient] = costFunction(theta)
    jVal = [code to compute J(θ)]
    gradient(1) = [code to compute ∂/∂θ0 J(θ)]
        ⋮
    gradient(n+1) = [code to compute ∂/∂θn J(θ)]
```

Sonra fminunc'ı kullanırız ve optimized theta values will be obtained.

Ideal for large scale (too many features) problems