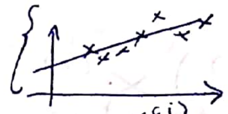
Machine Learning

$m \equiv$ number of training examples

$xs \equiv$ features

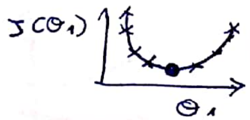
$ys \equiv$ target

- Cost function (J_0)

h_θ  θ_0, θ_1

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$



We have to minimize this

$$\frac{\partial}{\partial \theta_j} J(\theta) = 0$$

- Gradient descent algorithm

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$; $\alpha \equiv$ learning rate

Correct way

$$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

No intercalar

$$\theta_0 := temp0$$

$$\theta_1 := temp1$$

Lo que quiero decir es que se actualiza todo a la vez ...

• When we have multiple features:

$n \equiv$ number of features

$x^{(i)} \equiv$ input (features) of i^{th} training example.

$x_j^{(i)} \equiv$ value of feature j in i^{th} training example.

• $\forall n > 1$ (realmente $n \geq 1$)

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

- Simultaneously update θ_j $\forall j$
- $x_0^{(i)} = 1$

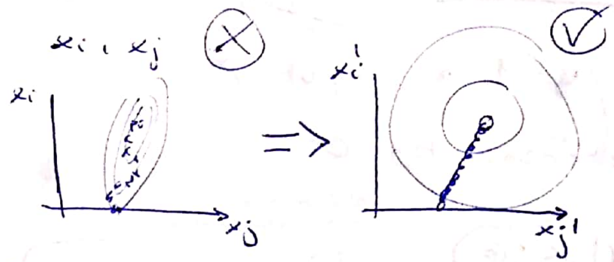
• Mean normalization

We need to do this for all features

$\forall i$ tq $i \neq 0 \Rightarrow \boxed{x_i = \frac{x_i - \mu_i}{s_i}}$

- $i \neq 0$ because $x_0 = 1$
- $\mu_i \equiv$ average value of x_i in training set
- $s_i \equiv$ range (max-min) or stdev of x_i

• Feature scaling



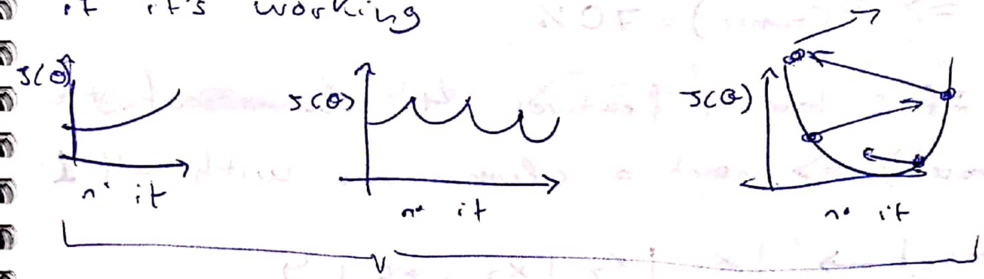
$x_i' = \frac{x_i}{x_{i\,max} - x_{i\,min}}$

$x_j' = \frac{x_j}{x_{j\,max} - x_{j\,min}}$

[This delays the convergence]

[dividing by interval we accelerate the convergence]

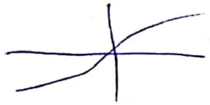
Plot $J(\theta)$ vs number of iterations to check if it's working



This is WRONG. Try lower α

- For sufficiently small α, $J(\theta)$ should decrease every iteration
- If α too small, reaching the convergence takes time.

• Hypothesis representation



$h_\theta(x) \equiv \mathbb{P}$ that $y = 1$ on input x

parametrized by θ

$$h_\theta(x) = \mathbb{P}(y = 1 \mid x ; \theta)$$

$$\mathbb{P}(y = \square_1 \mid x ; \theta) + \mathbb{P}(y = \square_2 \mid x ; \theta) + \dots = 1$$

Ex if $x = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ \text{tumor size} \end{pmatrix}$ and $h_\theta(x) = 0'7$

$\Rightarrow \mathbb{P}(\text{tumor}) = 70\%$

Ex If I have 4 features (4 columns of the dataset) $\Rightarrow$ insert a column x_0 with all 1:

dataset $\rightarrow$

x_1	x_2	x_3	x_4	y
1	2	3	4	40
0	100	50	30	100

$\Rightarrow$

x_0	x_1	x_2	x_3 ...
1	1	2	2
1	0	100	50

• Normal equation

$$X^i = \begin{pmatrix} 1 & 1 & 4 & 3 & 8 \\ 1 & 0 & 2 & 3 & 9 \end{pmatrix} \qquad y = \begin{pmatrix} 40 \\ 100 \end{pmatrix}$$

$m \times (n+1)$ m-dimensional vector

$$\theta = \underbrace{(X^T X)^{-1}}_{\text{inversa de } X^T X} X^T \cdot y \quad \leftarrow \text{when } J(\theta) \text{ minimized}$$

Esto ya está puesto como vector-columna. Los vectores fila están traspuestos

• En ~~clase~~ Matlab $X^+ = X'$

Gradient descent (Numerical solution)	Normal equation (Analytical solution)
- Feature scaling	- No need of feature scaling
- Needs to choose α	- No need of α
- Needs many iterations	- No need of iteration
- Works good when n large	- Slow if n very large $t \sim n^3$

$t \equiv$ time computing

if $n \sim 10^4$

Perfect solution

$$\boxed{\theta = (X^T X)^{-1} X^T Y}$$

• If X^TX not invertible use pinv

pinv(x'·x)·x'·y

pinv → pseudo invertible
inv → invertible

} use always pinv and ok

• If there is an invertibility problem check LD features.

There must be LD features, or if there are too many features, use regularization.

• Gradient descent $\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$

If $\alpha <<$ ⇒ ~~gradient~~ ~~descent~~ ∇descent slow

If $\alpha >>$ ⇒ ∇ descent it may fail to converge

If $\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$

m ≡ batches.

MATLAB

- save □.mat fi → guarda la función f en □.mat
  .txt
- load □.mat lo importa
  .txt
- Matrices: $A|_{3\times 2} \Rightarrow A(3,2)$ es el elem de esa posición

Vector indexes in Matlab starts in 1, not in 0

Logistic regression: Classification problem

∘ Logistic regression → $h_\theta(x) = g(\theta^T x)$

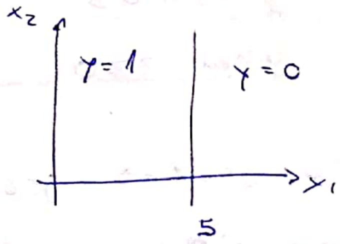
$$g(z) = 1/(1+e^{-z})$$

$$0 \le h_\theta(x) \le 1$$

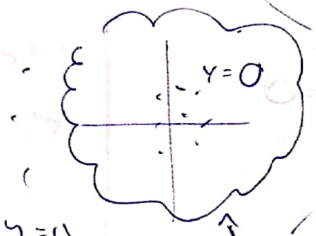
∘ Decision boundary $g(z)$: $z \equiv \theta X$

- If $\theta_0 = 5$, $\theta_1 = -5$, $\theta_2 = 0$, $h_\theta(x) = g(5 - x_1)$

La curva que determina qué set de datos son $y=0$ y ~~que~~ son $y=1$



$(y=0$ if $x_1 > 5)$



(y=1

y=1

decision boundary

To discrete 0 or 1 classification

$h_\theta(x) \geq 0'5 \rightarrow y=1$

$h_\theta(x) < 0'5 \rightarrow y=0$

The logistic function works like

$g(z) \geq 0'5$

when $z \geq 0$

$z=0,\ e^0=1 \Rightarrow g(z)=1/2$

$z \rightarrow \infty,\ e^{-\infty} \rightarrow 0 \Rightarrow g(z)=1$

$z \rightarrow -\infty,\ e^{+\infty} \rightarrow \infty \Rightarrow g(z)=0$

$\Rightarrow \quad h_\theta(x) = g(\theta^T x) = 0'5$

when $\theta^T x \geq 0$

$\Rightarrow \quad \theta^T x \geq 0 \Rightarrow y=1$

$\theta^T x < 0 \Rightarrow y=0$

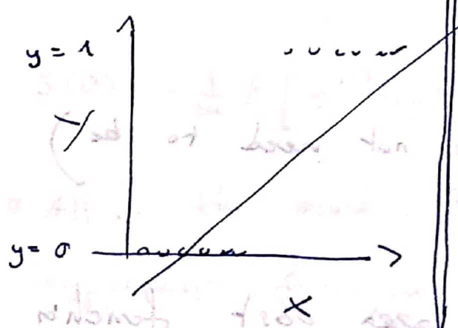
• the decision boundary separates the area where $y=0$ and where $y=1$

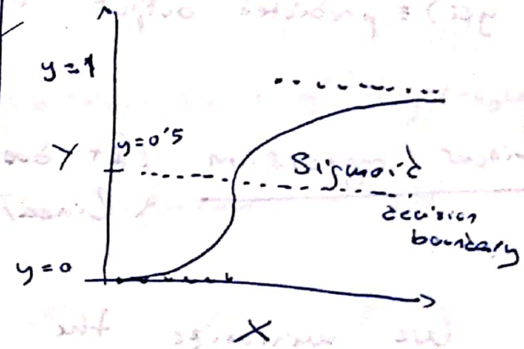
Ex $\theta = \begin{pmatrix} 5 \\ -1 \\ 0 \end{pmatrix}$

$y=1 \Leftrightarrow 5+(-1)x_1+0x_2 \geq 0$

$+5-x_1 \geq 0$; $-x_1 \geq -5$; $x_1 \leq 5)$ boundary decision

Summing up





Predicted Y can exceed 0 and 1

Predicted Y lies within 0 and 1

h_θ = hypothesis function

$h_\theta(x) = \beta_0 + \beta_1 X$

$0 \leq h_\theta(x) \leq 1$

$$h_\theta(x) = \frac{1}{1+e^{\underbrace{-(\beta_0+\beta_1 X)}_{z(x)=\theta^T X}}}$$

Cost function

Cost function ≡ error function

$$\text{Error} = \sum(\text{actual output} - \text{predicted output})^2$$

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

- Our goal is to minimize $J(\theta)$
- h_θ ≡ hypothesis funtion ≡ Y(predicted) value
- $y^{(i)}$ ≡ predicted output

• Linear regression (It does not need to be linear)

We minimize the ~~geen~~ cost function using gradient descent

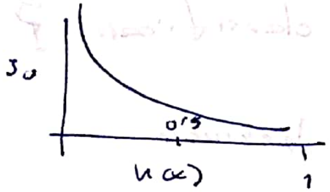
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial\theta_j} J(\theta)$$

α is learning rate. Be careful
Not too big (no converge), not to small (it takes a lot)

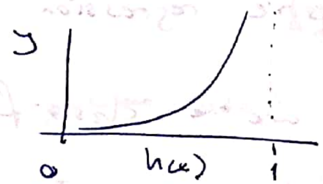
• Logistic regression (classification)

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

y = 1 ⇓



y = 0 ⇓



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

\# All is the same. The difference is the hypothesis function

$$J(h_\theta(x), y) = \begin{cases} 0 & \text{if } h_\theta(x) = y \\ \infty & \text{if } y=0 \text{ and } h_\theta(x)\to 1 \\ \infty & \text{if } y=1 \text{ and } h_\theta(x)\to 0 \end{cases}$$

• Using gradient descent for classification:

$$\frac{d}{dx}\sigma(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{-(1+e^{-x})'}{(1+e^{-x})^2} = \dots =$$

$$= \sigma(x)(1-\sigma(x))$$

Why logistic regression for classification?

Suppose we're classifying between spam or not spam. In linear regression $h(x)$ can be >1 or <1, which is out of context of ~~probabilistic~~ probability.

With logistic regression, $h(x) \in (0,1)$ and we can assign a probability

Using the Coursera nomenclature:

$$h_\theta(x) = g(\theta^T x) \qquad g(z) = \begin{cases} \to 1 & \text{as } z \to \infty \\ \to 0 & \text{as } z \to -\infty \\ \in (0,1) & \end{cases}$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1+e^{-z}} \equiv \text{sigmoid function}$$

$$\Downarrow$$

$h_\theta(x) = \mathbb{P}(y=1 \mid x;\theta)$ The probability of $y=1$ when x is parameterized to θ.

If $h_\theta(x) \geq 0.5 \Rightarrow y=1$

$h_\theta(x) < 0.5 \Rightarrow y=0$

$$g(z) \geq 0.5 \Rightarrow \underbrace{\theta^T x}\geq 0.5 \Rightarrow \underbrace{z} \geq 0.5$$

This is the decision boundary

The decision boundary is the line/curve that distinguishes the area where $y=0$ and $y=1$

Scanned with CamScanner

Understanding logistic regression:

The setting of the threshold is a very important aspect of the Logistic Regressor. and is dependent on the classification problem itself.

The decision for the value of the threshold is majority affected by precision and recall. Ideally, we want a precision and recall to be 1.

- Low precision/high recall

  When we want to reduce the number of false negatives without reducing the number of false positives

Ex Cancer diagnosis. We don't want any affected patient to be classified as not affected. without giving much heed to if the patient is being wrongfully diagnosed with cancer. This is because the absence of cancer can be detected in further tests but no quieres tener cáncer y ser diagnosticado sin él (porque no harán más pruebas y te morirás).

- High precision/low recall

  When we want to reduce the number of false positives without reducing the number of false negatives.

Ex Classifying customers wether they will react positively or negatively to a personalized ad. We want to make sure they'll react positively, otherwise we can ~~mise~~ loss a customer.

Then, logistic regression classification:

- binomial
- multinomial
- ordinal $\begin{cases} \text{very poor} \to 1 \\ \text{poor} \to 2 \\ \text{good} \to 3 \\ \text{very good} \to 4 \end{cases}$

• Multiple features, $n \equiv$ features

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

$$x_0^{(i)} \equiv 1$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\Rightarrow h_\theta(x) = \theta^T x$$

For $n = 1$

update simultaneously

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

For $n \geq 1$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

Simultaneously update $\theta_j \ \forall j$

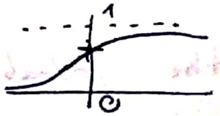
• Making sure gradient descent is working correctly

- For sufficient small α, $J(\theta)$ decrease over every iteration
- It is not slow converging

• Octave tutorial → Lecture 5

Again, Logistic Regression is for classification, where $0 \leq h_\theta \leq 1$ thanks to

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



tangente hiperbólica de toa la vida cabesa (con saltito)

• Logistic regression cost function

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

$$\Rightarrow J(\theta) := \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

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

• Gradient descent

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

• Multiclass

• 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 prediction, pick the class that maximizes

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

• The problem of overfitting

- too many features
  ↳ model selection algorithms
- regularization
  → Keep all features, but reduce magnitude/values of parameters θ_j
  → works well when having a lot of features