

Machine Learning


Logistic Regression

Classification

Based on ML Coursera
course by Andrew Ng

Tabular Data

Sample	Feature x_1	Feature x_2	Feature x_3	Feature x_n	Target y
1	$x_1^{(1)}$	$x_2^{(1)}$.	.	.	$x_n^{(1)}$	$y^{(1)}$
2	.					.	.
3	.					.	.
...	.					.	.
...	.					.	.
...	.					.	.
m	$x_1^{(m)}$	$x_n^{(m)}$	$y^{(m)}$



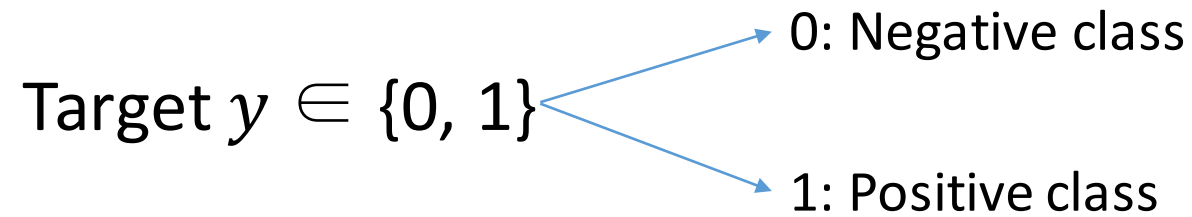
Matrix X (m, n)

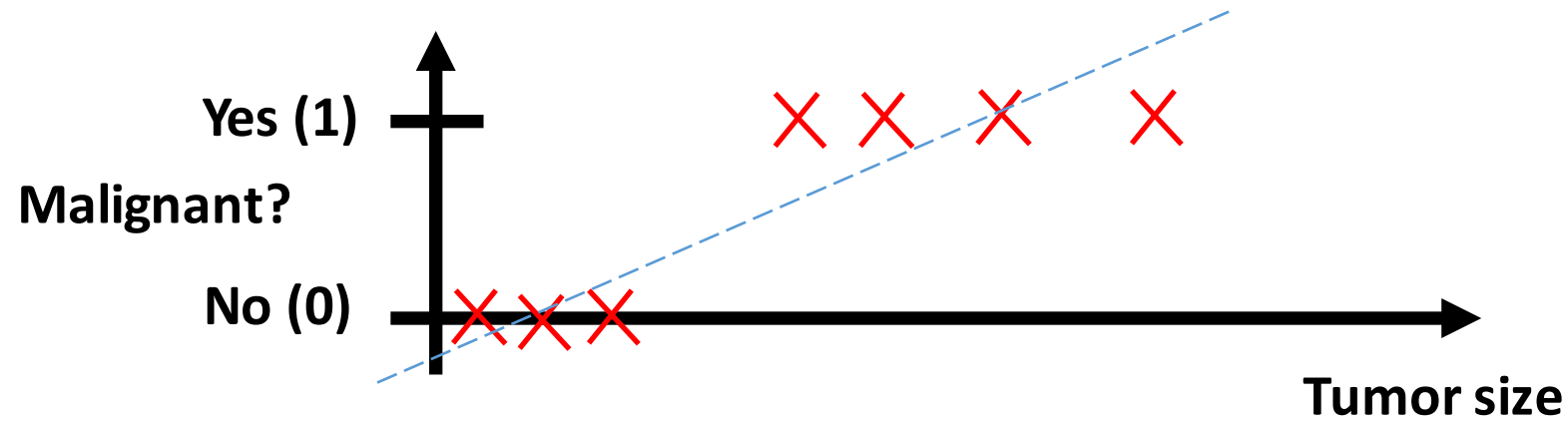
Vector Y ($m, 1$)

Binary Classification

Examples:

- Email: Spam/ Not spam?
- Bank transaction: Fraud/ Not fraud?
- Tumor: Benign/ Malignant?





Linear Regression function $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n = \theta^T x$ cannot perfectly stay in the range (0,1).

1	$x_1^{(1)}$	$x_2^{(1)}$.	.	.	$x_n^{(1)}$
1	.					.
1	.					.
1	.					.
1	.					.
1	.					.
1	.					.
1	$x_1^{(m)}$	$x_n^{(m)}$



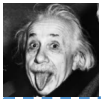
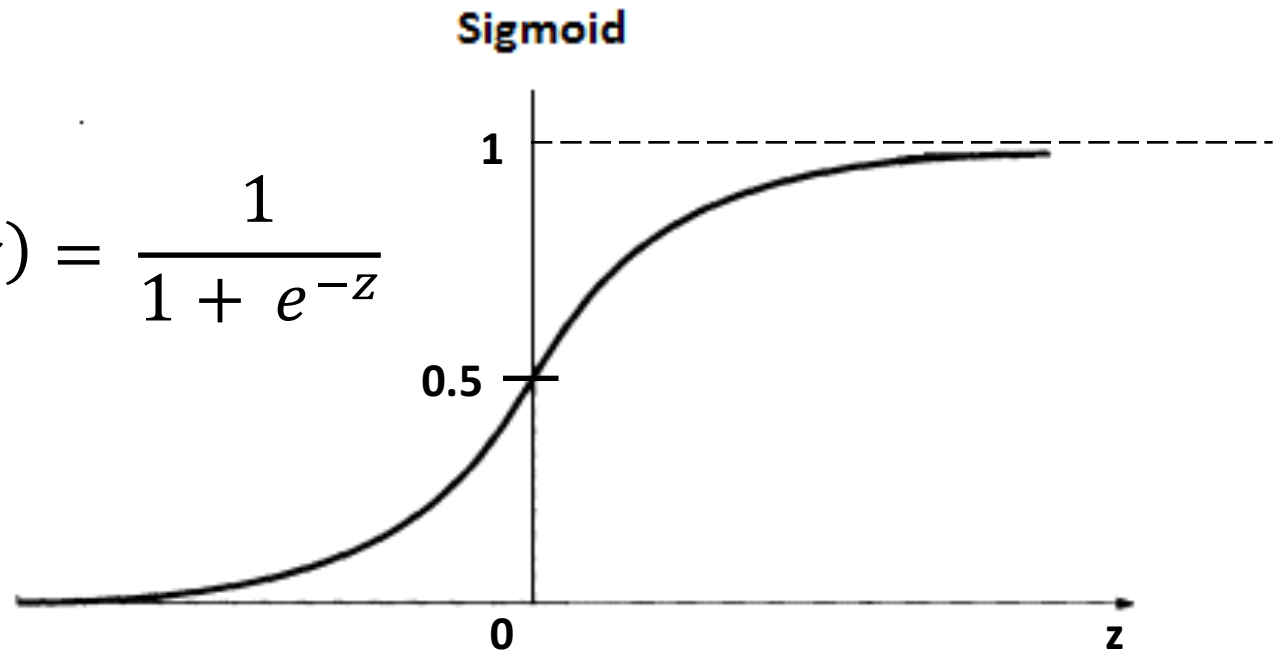
$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n = \theta^T x$$

Simple trick to help with calculation: we add a feature x_0 to the matrix of X , which are all equal to 1.

Need to warp a function around $\theta^T x$

$$h_{\theta}(x) = \text{sigmoid}(\theta^T x)$$

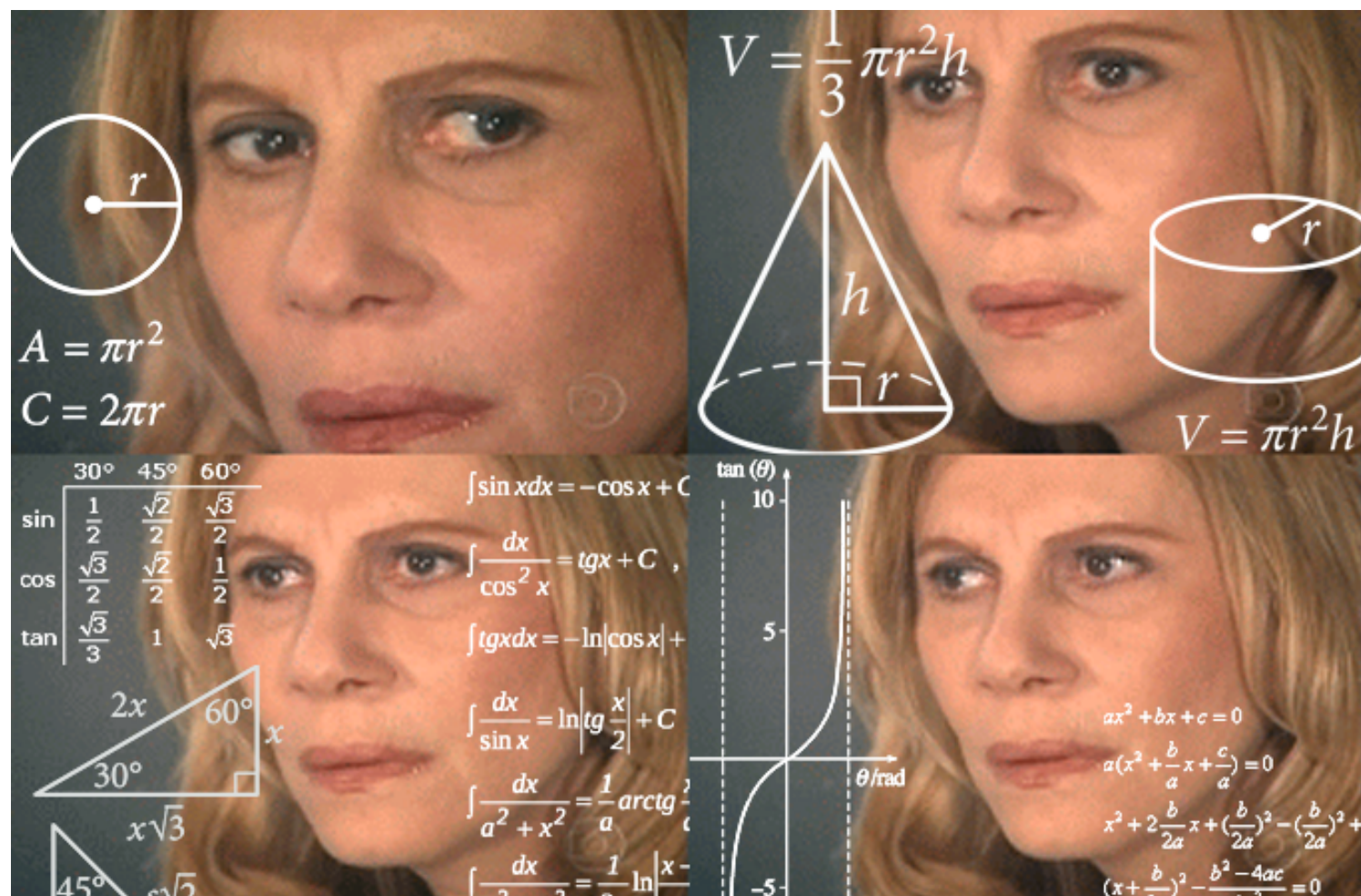
$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}}$$



Mathematically, this logistic sigmoid function is the one that satisfies the relationship:

$$\text{Log odds}(P(y = 1|x)) = \theta^T x$$

Too much maths?



Interpretation

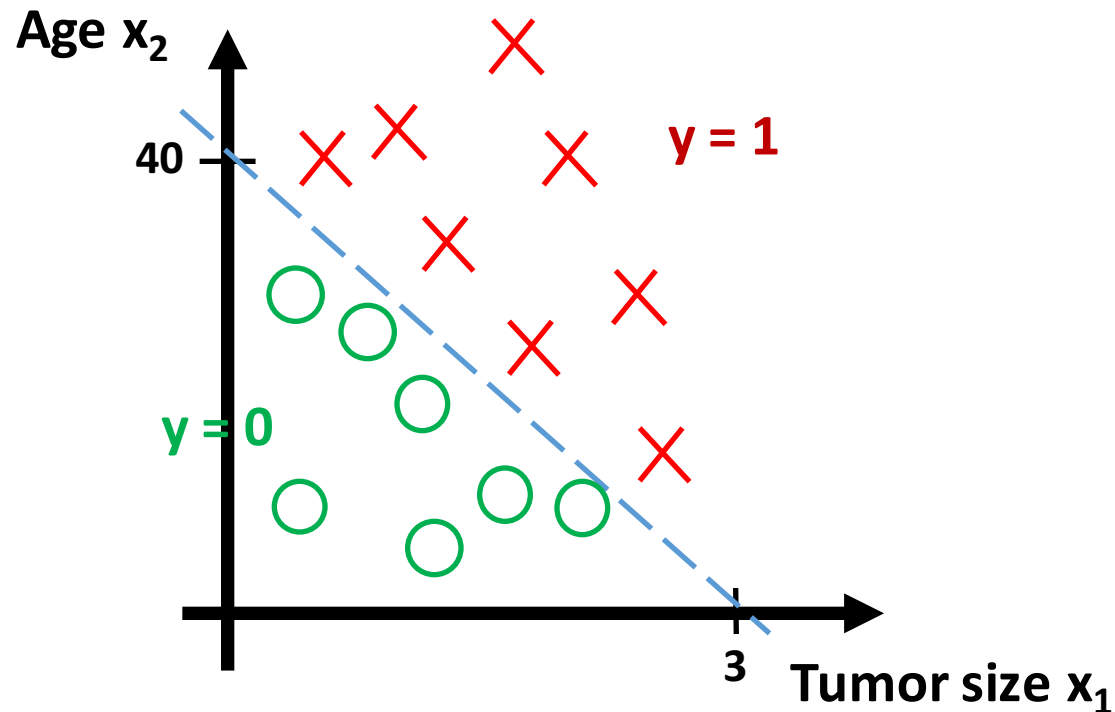
- $h_{\theta}(x)$ = estimated probability that $y = 1$ with input x

For example, if $h_{\theta}(x) = 0.7$, we say it's a 70% chance this tumor is malignant.

Translate to label:

- If $h_{\theta}(x) \geq 0.5 \rightarrow y = 1$
- If $h_{\theta}(x) < 0.5 \rightarrow y = 0$

Decision boundary



$$\begin{aligned}\theta_0 &= -40 \\ \theta_1 &= \frac{40}{3} \\ \theta_2 &= 1\end{aligned}$$

Predict $y = 1$ if
 $\text{sigmoid}\left(-40 + \frac{40}{3}x_1 + x_2\right) \geq 0.5$,
which is the same as
 $-40 + \frac{40}{3}x_1 + x_2 \geq 0$

Decision boundary doesn't have to be a straight line.

Big Idea

- We want to choose θ so that $h_{\theta}(x)$ “fits” the true label y as much as possible.

- But how do you know which one fits better?

$h = 0.6$ and $y = 1$

or

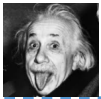
$h = 0.9$ and $y = 1$?

- We have something called a cost function $J(\theta)$, and the smaller it is, the better the fit. Basically, we need to find θ to minimize this Cost $J(\theta)$.

Cost function

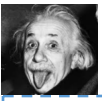
Cost function is the average of the Cost function for each sample, which will explain shortly after:

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



Mathematically, this cost function maximizes the likelihood of X given Y

$$P(X|Y)$$



Cost function for 1 sample

- If $y = 1$, Cost is: $-\log(h_{\theta}(x))$

Intuition:

If $h_{\theta}(x)$ is large (close to 1), $\log(h_{\theta}(x))$ will be close to 0, $-\log(h_{\theta}(x))$ will be close to 0 -> **good**

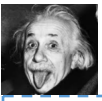
However, if $h_{\theta}(x)$ is small (close to 0), $\log(h_{\theta}(x))$ will be a large negative number, $-\log(h_{\theta}(x))$ will be positively large -> **bad**

- If $y = 0$, Cost is: $-\log(1 - h_{\theta}(x))$

Intuition:

If $h_{\theta}(x)$ is large (close to 1), $1 - h_{\theta}(x)$ will be close to 0, $\log(1 - h_{\theta}(x))$ will be a large negative number, $-\log(1 - h_{\theta}(x))$ will be positively large -> **bad**

However, if $h_{\theta}(x)$ is small (close to 0), $1 - h_{\theta}(x)$ will be close to 1, $-\log(1 - h_{\theta}(x))$ will be close to 0 -> **good**



Cost function for 1 sample

Thus, in a simplified way, Cost for 1 sample is:

$$\text{Cost}(h_{\theta}(x), y) = -[y * \log(h_{\theta}(x)) + (1 - y) * \log(1 - h_{\theta}(x))]$$

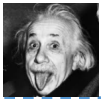
Pluck in $y = 1$ or $y = 0$, and you will see that it is exactly the cost we defined in the last slide.

Gradient

Now we have the cost, how do we minimize it? The answer is Gradient Descent.

To start off, we need to calculate the partial derivative of $J(\theta)$ over θ_j :

$$\frac{d}{d\theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$



Mathematically, this equation utilizes the chain rule and the property of sigmoid function:

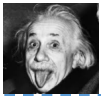
$$\frac{d}{dz} \text{sigmoid}(z) = \text{sigmoid}(z)(1 - \text{sigmoid}(z))$$

Gradient Descent

Then we update θ with the opposite direction of the gradient, by repeating:

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

(with α a pre-set number called the learning rate)



Choosing the learning rate has an impact on the algorithm:

- α too small make the gradient descent process very slow
- α too large can return bad result

Summary

- Step 1: Calculate $h_{\theta}(x) = \text{sigmoid}(\theta^T x)$
- Step 2: Calculate cost $J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$
- Step 3: Calculate gradients $\frac{d}{d\theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$
- Step 4: Update θ based on gradient $\theta_j := \theta_j - \lambda * \frac{d}{d\theta_j} J(\theta)$
- Step 5: Repeat step 1, 2, 3, 4 many many times to get “optimal” θ

Mind = blown?

