LOGISTIC REGRESSION

→ Classification

We will focus on the **binary classification** problem, where:

$y \in \{0, 1\}$

0 : Negative class (ex: benign tumor)
1 : Positive class (ex: malignant tumor)

Applying linear regression to a classification problem is **not effective**.

Hypothesis

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

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

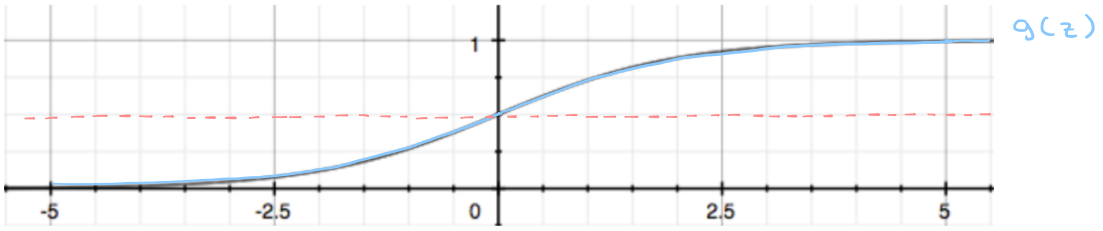
Interpretation

$h_\theta(x) = P(y=1 \mid x; \theta)$ → estimated probability that $y=1$ on input x

Because $y=0$ or $y=1$: $P(y=0 \mid x; \theta) + P(y=1 \mid x; \theta) = 1$

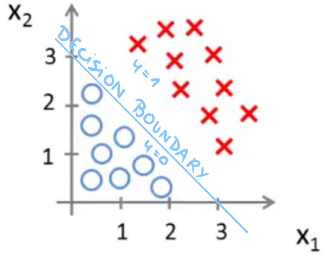
$P(y=0 \mid x; \theta) = 1 - P(y=1 \mid x; \theta)$

Decision Boundary



$g(z) < 0$ when $z < 0$ $\qquad$ $g(z) > 0$ when $z > 0$

So, $h_\theta(x) = g(\theta^T x) \geq 0.5$, whenever $\theta^T x \geq 0$ ($= z$)



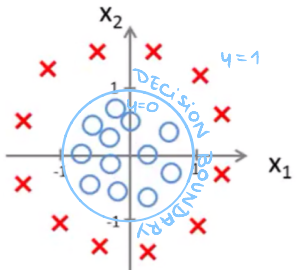
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

Say we choose $\theta_0 = -3$ and $\theta_1 = \theta_2 = 1$

Predict $y=1$ if $-3 + x_1 + x_2 \geq 0$ ($\Leftrightarrow x_1 + x_2 \geq 3$)

The Decision Boundary is a property of the hypothesis and its parameters, not of the dataset.

Non-linear Decision Boundaries



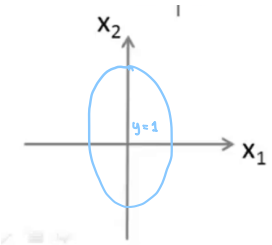
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

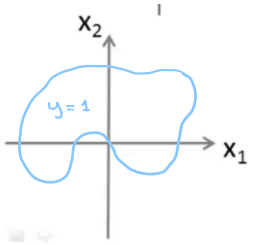
Say we choose $\theta_0 = -1$, $\theta_1 = \theta_2 = 0$ and $\theta_3 = \theta_4 = 1$

Predict $y=1$ if $-1 + x_1^2 + x_2^2 \geq 0 \iff x_1^2 + x_2^2 \geq 1$

Higher Order Polynomial Features

Can produce all kinds of more or less complex shapes.





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

→ Logistic Regression

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

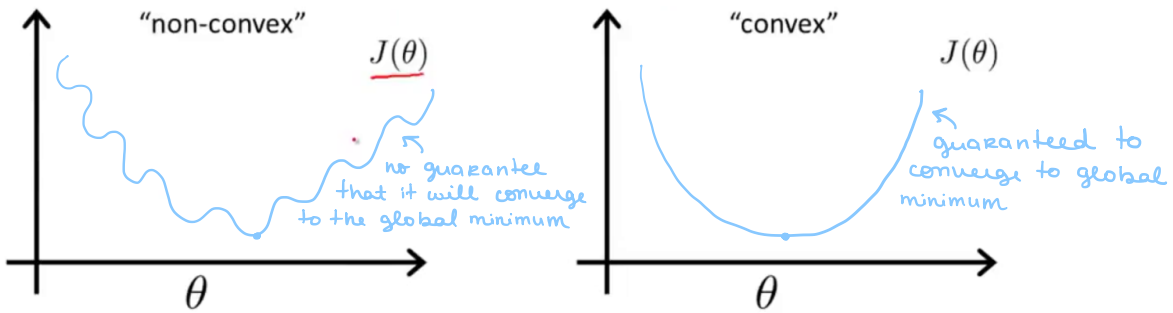
$x_0 = 1, y \in \{0, 1\}$

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

How to choose the parameters θ?

Cost Function

We cannot use the same Cost Function that we use for linear regression because the logistic Function will cause the output to be wavy, causing many local optima. In other words, it will not be a convex function.

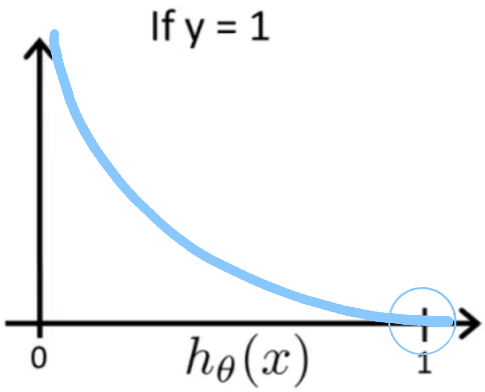


$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}Cost(h_\theta(x^{(i)})-y^{(i)})$$

$$Cost(h_\theta(x),y)=-\log(h_\theta(x)) \quad \text{if } y=1$$

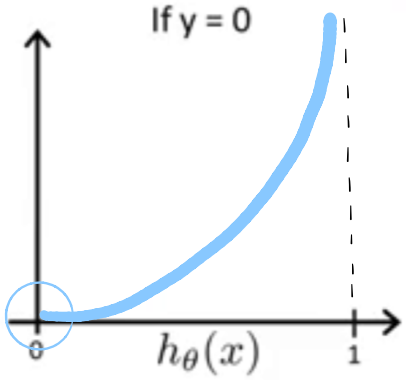
$$Cost(h_\theta(x),y)=-\log(1-h_\theta(x)) \quad \text{if } y=0$$

Plotting $-\log(h_\theta(x))$



Cost = 0 if $y=1$, $h_\theta(x)=1$
But as $h_\theta(x)\to 0$, Cost $\to \infty$

Plotting $-\log(1-h_\theta(x))$



Cost = 0 if $y=0$, $h_\theta(x)=0$
But as $h_\theta(x)\to 1$, Cost $\to \infty$

Simplified Cost Function

$$Cost(h_\theta(x),y)=-y\ \log(h_\theta(x))-(1-y)\log(1-h_\theta(x))$$

Suppose $y=1$ $\rightarrow$ $Cost(h_\theta(x),y)=-1\log(h_\theta(x))-\overbrace{(1-1)\log(1-h_\theta(x))}^{0}=-\log(h_\theta(x))$

Suppose $y=0$ $\rightarrow$ $Cost(h_\theta(x),y)=-\overbrace{0\log(h_\theta(x))}^{0}-(1-0)\log(1-h_\theta(x))=-\log(1-h_\theta(x))$

So, we can write the **Cost Function** as:

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

A **vectorized** implementation is:

$$h=g(X\theta)$$
$$J(\theta)=\frac{1}{m}\cdot(-y^{T}\log(h)-(1-y)^{T}\log(1-h))$$

Gradient Descent

We will use Gradient Descent to minimize the Cost Function.

Repeat until convergence {

$$\theta_j:=\theta_j-\alpha\ \frac{\delta}{\delta\theta_j}\ J(\theta_0,\ldots,\theta_n)$$

}

Working the derivative part, we get:

Repeat until convergence {

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

}

A vectorized implementation is:

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$

Notice that the algorithm is identical to the one used in Linear Regression! So what is the difference?
What has changed is the definition for $h_\theta(x)$:

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

→ Advanced Optimization

Optimization Algorithm

Different approaches to compute $J(\theta)$ and $\frac{\partial}{\partial \theta_j} J(\theta)$:

- Gradient Descent
- Conjugate gradient
- BFGS
- L-BFGS

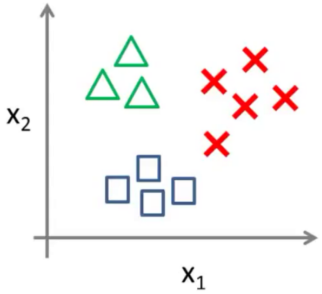
ADVANTAGES:
- No need to manually pick α
- Often faster than Gradient Descent

DISADVANTAGES:
- More complex

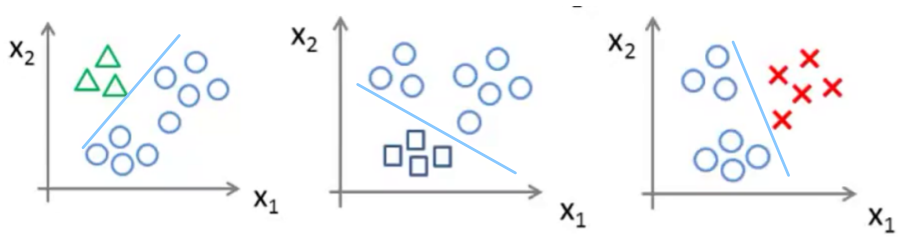
→ Multiclass Classification: One-vs-All

Multiclass Classification Problems:

- E-mail foldering / tagging → Work ($y=1$), Friends ($y=2$), Family ($y=3$), ...
- Medical diagnosis → Not ill ($y=1$), Cold ($y=2$), Flu ($y=3$), ...
- Weather → Sunny ($y=1$), Cloudy ($y=2$), Rain ($y=3$), ...



Using one-vs-all classification, we turn a multiclass classification problem into several binary:



$h_\theta^{(1)}(x)$ $P(y=1 \mid x;\theta)$ $\quad$ $h_\theta^{(2)}(x)$ $P(y=2 \mid x;\theta)$ $\quad$ $h_\theta^{(3)}(x)$ $P(y=3 \mid x;\theta)$

- Train a logistic gre ression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y=i$.

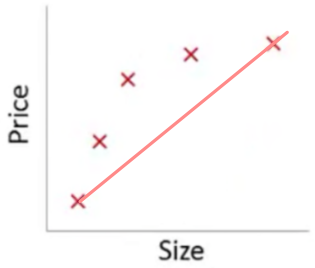
- On a new input x, to make a prediction, pick the class i that maximizes $\max_i h_\theta^{(i)}(x)$

REGULARIZATION

→ The problem of Overfitting

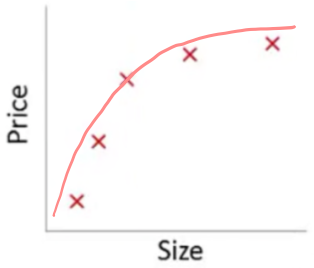
Example: Linear Regression

Housing Prices

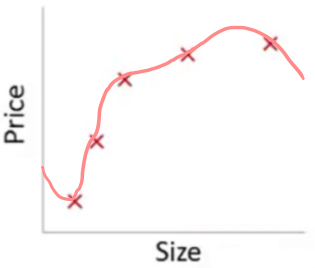


$\theta_0 + \theta_1 x$

Underfit / High bias



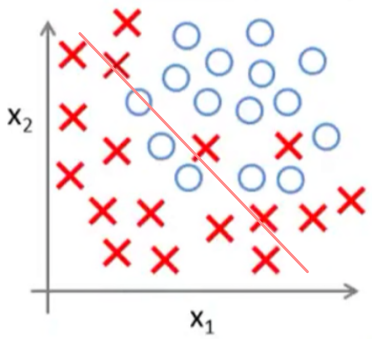
$\theta_0 + \theta_1 x + \theta_2 x^2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

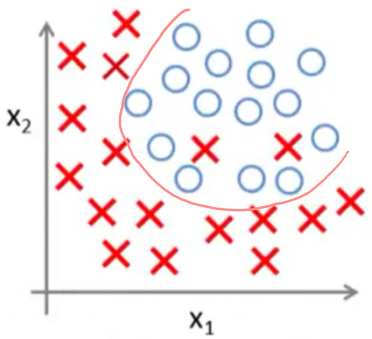
Overfit / High variance

Example: Logistic Regression

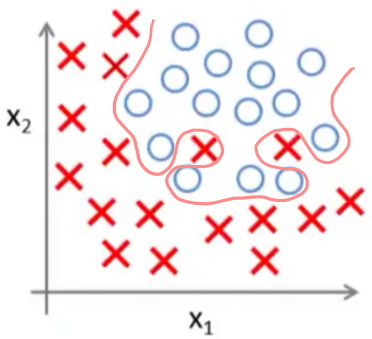


$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

Overfitting

If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) \approx 0$), but fail to generalize to new examples.

Addressing the overfitting

1. Reducing the number of features

- Manually select which features to keep
- Model selection algorithm → algorithms that automatically decide which features to keep and which features to lose

2. Regularization

- Keeping all the features, but reducing magnitude / values of parameters θ_j.
- Works well when we have a lot of features, each of which contributes a bit to predicting y.
- "Simpler" hypothesis.
- Less prone to overfitting.

→ Cost Function

Suppose we penalise and make θ_3 and θ_4 really small:

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

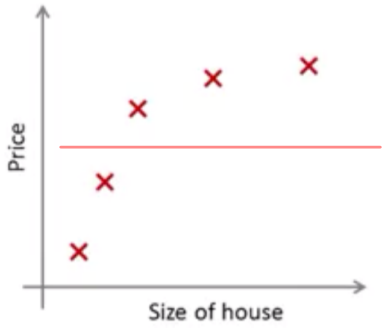
When we minimise this function we end up with θ_3 and θ_4 close to zero, which leads to a quadratic function.
So, we modify the Cost Function:

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

λ: regularization parameter

$j=1$ so we don't penalise θ_0

If λ is ~~too~~ large, the algorithm may underfit. We end up penalising the parameters too highly — all end up close to zero, and all we have left is $h_\theta(x) = \theta_0$ which is like fitting a straight horizontal line through the data.



→ "underfit"
too high bias.

For regularization to work well, we need some care with λ!

→ Regularized Linear Regression

Gradient Descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

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

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

$\frac{\partial}{\partial \theta_j} J(\theta)$ regularized

}

Grouping the terms that relate to θ_j:

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

Normal Equation

$$= (X^T X + \lambda . L)^{-1} X^T y$$

$$\text{Where } L = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \dots \\ & & & & 1 \end{bmatrix}$$

If $\lambda > 0$ the matrix is invertible.
Using regularization, we also take care of the non-invertibility issue of the $X^T X$ matrix.

→ Regularized logistic Regression

Cost Function

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

Gradient Descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

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

}

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

→ it is not the same algorithm as linear regression because the hypothesis is $h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$