# D. Logistic Regression

## I. Classification and representation

1. Classification

The method is to use linear regression and map all predictions (phỏng đoán) greater than 0.5 as a 1 and all less than 0.5 as a 0. However, this method doesn't work well because classification is not actually (thật) a linear function.

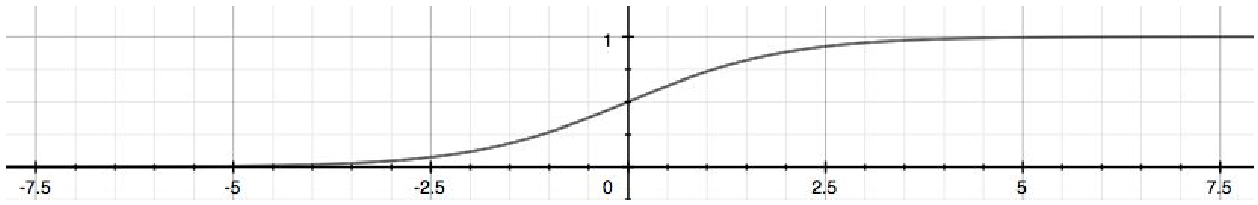
The classification is just like a regression problem. Now, we focus on the "binary classification problem" in which y can take on (đảm nhận) only two values, 0 and 1. Hence (vì thế) $y \in \{0, 1\}$ . 0 is also called the negative class, and 1 the positive class, and they are sometimes also denoted by the symbols "-" and "+". Given $x^i$, the corresponding (tương ứng) $y^i$ is also called the label for the training example.

2. Hypothesis representation (đại diện giả thuyết)

The "Sigmoid function" also called the "Logistic function":

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

The following image shows us what the sigmoid function looks like:



The function **g(z)** maps any real number to the (0, 1) interval (khoảng), making it useful for transforming an arbitrary-valued function (hàm có giá trị tùy ý) into a function better suited (phù hợp) for classification.

$h_\theta(x)$ will give the probability that output is 1.

$$h_\theta(x) = P(y=1|x;\theta) = 1 - P(y=0|x;\theta)$$
$$P(y=0|x;\theta) + P(y=1|x;\theta) = 1$$

3. Decision boundary (Ranh giới quyết định)

In order to get our discrete 0 or 1 classification, we can translate the output of the hypothesis function as follows:

$$h_\theta(x) \geq 0.5 \rightarrow y = 1$$
$$h_\theta(x) < 0.5 \rightarrow y = 0$$

The way our logistic function **g** behaves is that when its input is greater than or equal to zero, its output is greater than or equal to 0.5:

$$g(z) \geq 0.5$$
$$when\ z \geq 0$$

And:

$$z = 0, e^0 = 1 \Rightarrow g(z) = 1/2$$
$$z \to \infty, e^{-\infty} \to 0 \Rightarrow g(z) = 1$$
$$z \to -\infty, e^{\infty} \to \infty \Rightarrow g(z) = 0$$

If input to **g** is $\theta^T X$, that means:

$$h_\theta(x) = g(\theta^T x) \geq 0.5$$
$$when\ \theta^T x \geq 0$$

From these statements, can say:

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

The decision boundary is the line that separates the area where y = 0 and where y = 1. It is created by hypothesis function.

The input to the sigmoid function **g(z)** doesn't need to be linear, and could be a function that describes a circle or any shape to fit data.

**II. Logistic regression model**

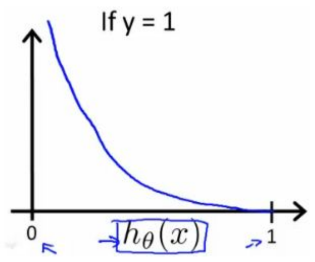
1. Cost function

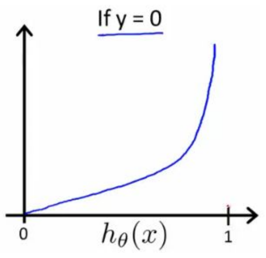
Cannot use the same cost function that we use for linear regression because the Logistic function will cause the output to be <u>wavy</u> (lượn sóng), causing many local optima. In other words, it will not be a <u>convex</u> (lồi) function.

Logistic regression look like:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$\mathrm{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if y = 1}$$
$$\mathrm{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if y = 0}$$

Plot $J(\theta)$ with $h_\theta(x)$





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

If the correct answer 'y' is 0, then the cost function will be 0 if hypothesis function also outputs 0. If hypothesis approaches 1, then the cost function will approach infinity.
If the correct answer 'y' is 1, then the cost function will be 0 if hypothesis function outputs 1. If hypothesis approaches 0, then the cost function will approach infinity.
Note that writing the cost function in this way guarantees that J(θ) is convex for logistic regression.
2. Simplified cost function and Gradient descent
* Simplified cost function
Compress cost function:

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

Entire cost function:

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

Vectorized implementation:

$$\begin{aligned}&h = g(X\theta)\\ &J(\theta) = \frac{1}{m}\cdot\left(-y^T\log(h) - (1-y)^T\log(1-h)\right)\end{aligned}$$

* Gradient descent
General form of gradient descent:

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

The derivative part:

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

Vectorized implementation:

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

3. Advanced optimization
"Conjugate gradient", "BFGS", and "L-BFGS" are more sophisticated, faster ways to optimize θ.

Provide a function that evaluates the following two functions for a given input value θ:

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

We can write a single function that returns both of these:

```
function [jVal, gradient] = costFunction(theta)
   jVal = [...code to compute J(theta)...];
   gradient = [...code to compute derivative of J(theta)...];
end
```

Use octave's "fminunc()" optimization algorithm along with the "optimset()" function that creates an object containing the options we want to send to "fminunc()".

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

**III. Multiclass classification**

1. Multiclass classification: One vs all

The classification of data when we have more than two categories. Instead of y = {0,1} will expand our definition so that y = {0,1...n}, divide our problem into n+1binary classification problems:

$$y \in \{0, 1 \ldots n\}$$
$$h_\theta^{(0)}(x) = P(y = 0|x;\theta)$$
$$h_\theta^{(1)}(x) = P(y = 1|x;\theta)$$
$$\cdots$$
$$h_\theta^{(n)}(x) = P(y = n|x;\theta)$$
$$\text{prediction} = \max_i(h_\theta^{(i)}(x))$$

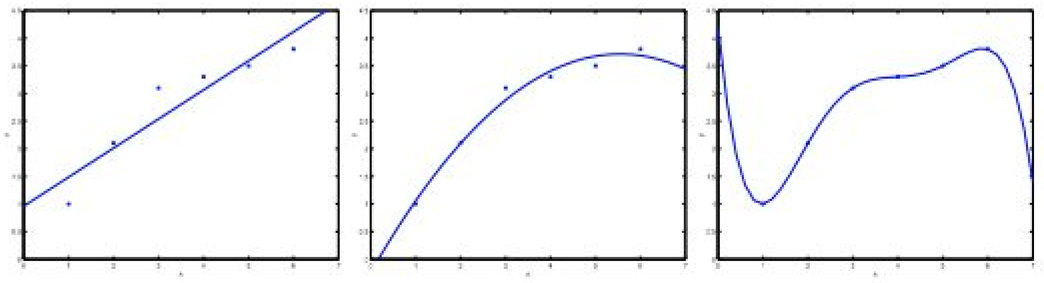
By choosing one class and then lumping all the others into a single second class, do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as prediction.

# E. Regularization

## I. Solving the Problem of overfitting

1. The problem of overfitting

*Eg: Consider the problem of predicting y from x* $\in R$.



*The leftmost shows* $y = \theta_0 + \theta_1 x$. *The fitting is not good.*

*The middle: fit* $y = \theta_0 + \theta_1 x + \theta_2 x^2$. *It's better.*

*The rightmost:* $y = \sum_{j=0}^{5} \theta_j x^j$ *.The best at this example.*

The figure on the left shows an instance of **underfitting**. The figure on the right is an example of **overfitting**.

**Underfitting** (or **high bias**) is when the form of hypothesis function h maps poorly to the trend of the data. It is caused by a function that is too simple or uses too few features.

**Overfitting** (or **high variance**) is caused by a hypothesis function that fits the available data but does not generalize well to predict new data. It is caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data.

They applied to both linear and logistic regression. There are two main options to solve the issue of overfitting:

+ Reduce the number of features:

    Manually select which features to keep.

    Use a model selection algorithm.

+ Regularization

    Keep all the features, but reduce the magnitude of parameters $\theta_j$.

    Regularization works well when it has a lot of useful features.

2. Cost function

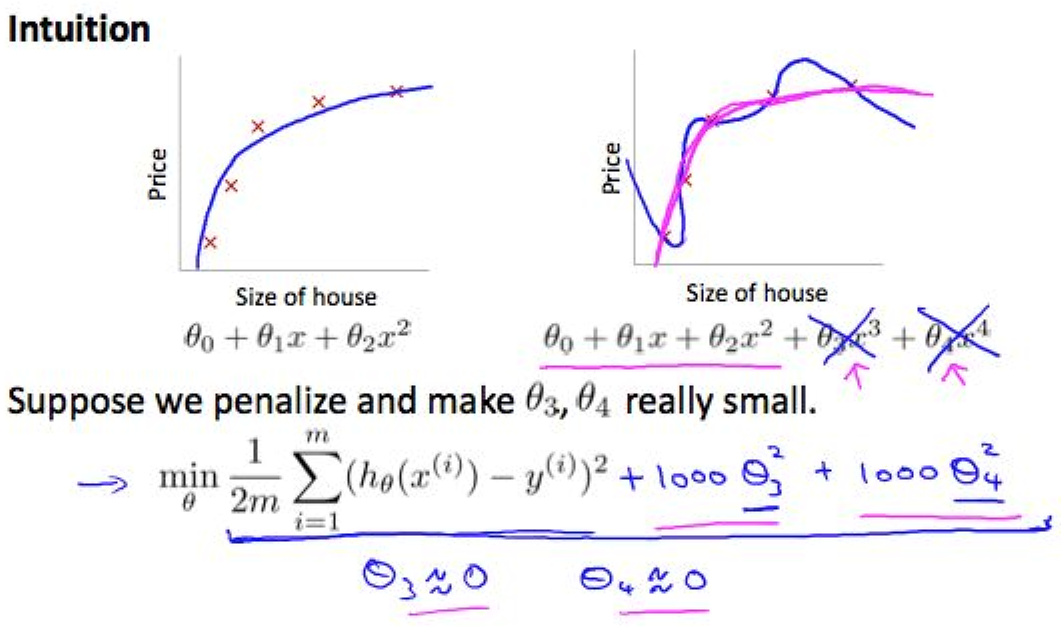
If we have overfitting from our hypothesis function, we can reduce the weight of some terms in function by increasing their cost.

*Eg: Make the function to quadratic:* $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$.

*eliminate the influence of* $\theta_3 x^3 + \theta_4 x^4$, *instead modify cost function:*

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

*Added two extra terms at the end to inflate the cost of $\theta_3 x^3 + \theta_4 x^4$, in order for the cost function to get close to zero, we will have to reduce the values of $\theta_3 + \theta_4$ to near zero. This will in turn greatly reduce the values of $\theta_3 x^3 + \theta_4 x^4$. Result, we see that the new hypothesis (depicted by the pink curve) looks like a quadratic function but fits the data better due to the extra small terms $\theta_3 x^3 + \theta_4 x^4$.*



**Intuition**

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

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

**Suppose we penalize and make $\theta_3, \theta_4$ really small.**

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

$\theta_3 \approx 0 \qquad \theta_4 \approx 0$

Summation:

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

The λ, or lambda, is the regularization parameter. It determines how much the costs of theta parameters are inflated. The above cost function with the extra summation smooths the output of hypothesis function to reduce overfitting.
If lambda is chosen to be too large, it may smooth out the function too much and cause underfitting.
3. Regularized Linear Regression
X is non-invertible if m < n, and may be non-invertible if m = n.
* Gradient Descent
Modify gradient descent function to separate out $\theta_0$ from the rest of the parameters.

$$\text{Repeat } \{$$
$$\theta_0 := \theta_0 - \alpha \ \frac{1}{m} \ \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$
$$\theta_j := \theta_j - \alpha \ \left[ \left( \frac{1}{m} \ \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right] \qquad j \in \{1, 2...n\}$$
$$\}$$

$\frac{\lambda}{m}\theta_j$ performs regularization. With some manipulation, update rule can be represented as:

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

In the above equation, $1 - \alpha\frac{\lambda}{m}$ will alway be less than 1.

* Normal equation

$$\theta = \left( X^T X + \lambda \cdot L \right)^{-1} X^T y$$
$$\text{where } L = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

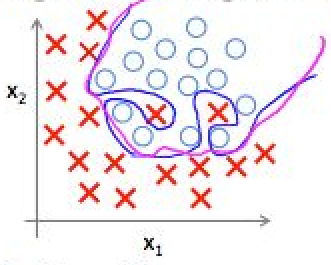
L is a matrix with 0 at the top left and 1's down the diagonal, with 0's everywhere else. Dimension (n+1)×(n+1). This is the identity matrix (though we are not including $x_0$) multiplied with a single real number λ.

If m < n, then $X^TX$ is non-invertible. However, when add the term λL, then $X^TX + \lambda L$ becomes invertible.

4. Regularized logistic regression

Regularized logistic regression likes regularized linear regression. Result, we can avoid overfitting. The image shows how the regularized function, displayed by the pink line, is less likely to overfit than the non-regularized function represented by the blue line:

**Regularized logistic regression.**



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

Cost function:

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \quad \boxed{\theta_1, \theta_2, \dots, \theta_n}$$

Andrew Ng

* Cost function

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

We can regularize this equation by adding a term to the end:

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

$\sum_{j=1}^{n}\theta_j^2$ means to **explicitly exclude** the bias term, $\theta_0$. When computing the equation, we should continuously update the two following equations:

**Gradient descent**

Repeat {

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

$$\theta_j := \theta_j - \alpha \underbrace{\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]}_{\frac{\partial}{\partial\theta_j}J(\theta)} \quad (j = \cancel{0}, 1, 2, 3, \dots, n) \quad \theta_1, \dots, \theta_n$$

}

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