Classification

$y = \begin{cases} 0 : \text{Negative class} \\ 1 : \text{positive class} \end{cases}$

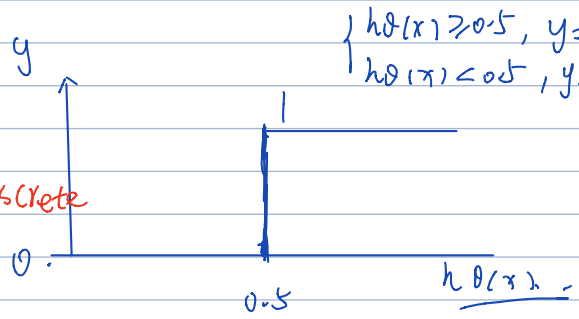
Because sometimes classification is not a linear function, we cannot use linear regression to attempt classification.

multiclass classification Problem:

$y = \{0, 1, 2, 3\}$

定义一种分类算法
↳ 适用于离散情况 discrete

Logistic Regression Model (逻辑回归)



$\begin{cases} h_\theta(x) \geq 0.5, & y = 1 \\ h_\theta(x) < 0.5, & y = 0 \end{cases}$

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

$h_\theta(x) = g(z)$

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

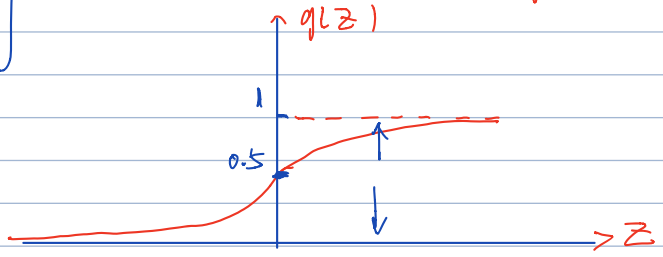
$z = \theta^T x$

$\Rightarrow h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ ↳ Sigmoid Function (S型) / Logistic Function

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

X: 特征向量.

g: 逻辑函数.



g(z) maps any real number into the (0,1) interval

Interpretation of Hypothesis Output

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

根据选择的参数 计算出变量=1的可能性, $h_\theta(x) = p(y=1 \mid x; \theta)$

e.g. if $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorsize} \end{bmatrix}$

$h_\theta(x) = 0.7$ for $y=1$ that patient with feature x, the probability that $y=1$ is 0.7.

Tell patient that tumor 70% chance of tumor being malignant. 恶性的

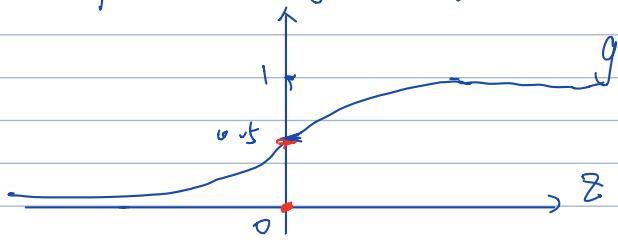
$h_\theta(x) = p(y=1 \mid x; \theta)$ "probability that $y=1$, given x, parameterized by θ.

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

Decision Boundary: the line separate the are where $y=0$ and where $y=1$
it is created by hypothesis function.

suppose predict: $y=1$ if $h_\theta(x) \geq 0.5$.

predict: $y=0$ if $h_\theta(x) < 0.5$.



when $g(z) \geq 0.5$, $y=1$

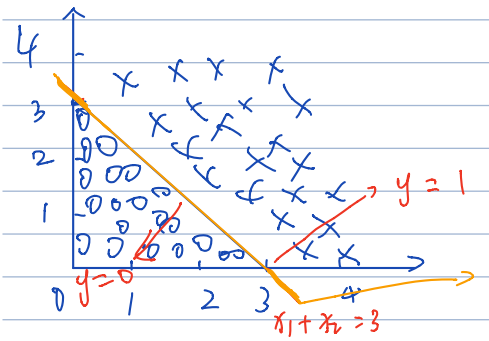
when $z \geq 0$

we know that:

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

$z = \theta^T x \geq 0$.

$\therefore \theta^T x \geq 0$



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = g(-3 + x_1 + x_2)$

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

Remember:

$$\begin{cases} g(z) = \dfrac{1}{1+e^{-z}} \\ z = \theta^T x \end{cases}$$

$\Rightarrow$ $z = 0$, $g(z) = \frac{1}{2}$

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

$z \to -\infty$, $e^{-z} \to \infty$, $g(z) = 0$.

predict $y = 1$ when $-3 + x_1 + x_2 \geq 0$

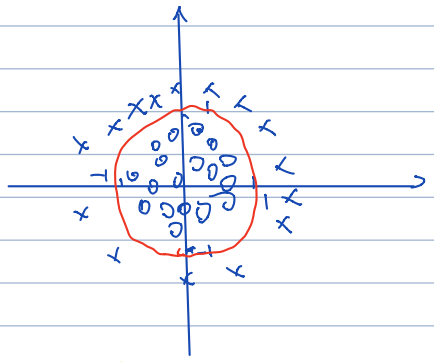
$\Rightarrow$ $x_1 + x_2 \geq 3$

predict $y = 0$ when $-3 + x_1 + x_2 < 0$

$x_1 + x_2 < 3$

Non-linear Decision Boundaries

polynomia regression:



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

assume $\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

$y = 1$, $-1 + x_1^2 + x_2^2 \geq 0$.

$x_1^2 + x_2^2 \geq 1$

Notice: higher order of polynomia, the more complex Decision Boundary.

How to fit the parameters: 定义用来拟合参数的优化目标或者代价函数.

Supervised model:

m examples: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$.

n+1 features: $x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$ $x_0 = 1$, $y \in \{0, 1\}$

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

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

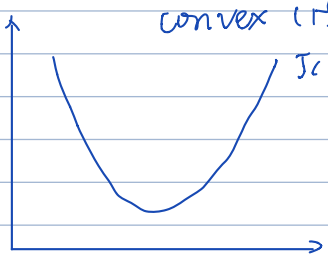
$\frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$ → cost function: $\text{cost}(h_\theta(x^{(i)}), y^{(i)})$

$\text{cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$

Non-convex 非凸函数 $J(\theta)$

cause many local optimum

convex 凸函数 $J(\theta)$.

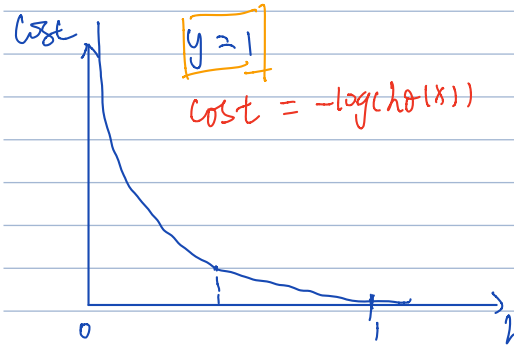


Because $h_\theta(x)$ in classification problem is nontinear ($h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$);

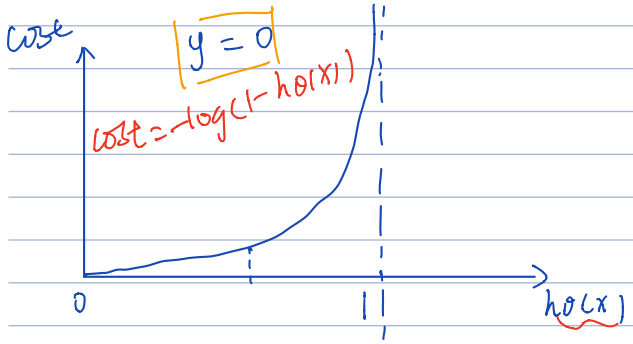
$J(\theta)$ is non-convex; if use gradient decient. there is no global optimization.

Logistic regression cost function: $J(\theta) = \frac{1}{m}\sum Cost(h_\theta(x), y)$

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & \text{if } y=1 \\ -\log(1-h_\theta(x)), & \text{if } y=0 \end{cases} \qquad h_\theta(x) \in [0,1]$$



Notice: Cost = 0, if $y=1$, $h_\theta(x) = 1$
But as $h_\theta(x) \to 0$
Cost $\to \infty$
Captures intuition that if $h_\theta(x) = 0$, predict $P(y=1 \mid x, \theta) = 0$, but $y=1$, we'll penalize learning algorithm by a very large cost.



if $y=0$, $h_\theta(x) = 0$, cost = 0.
if $y=0$, $h_\theta(x) = 1$, cost $\to \infty$

Simplified cost function and Gradien descent.

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

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

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

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

Logistic Regression Cost Function:

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

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

vectorized implementation

$h = g(X\theta)$

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

To fit parameters θ:

$\min J(\theta)$ → Get a set of θ

To make a prediction given new x:

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

$J\theta = \frac{1}{m}\Sigma$

Gradien Descent:

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

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta). \quad \Rightarrow \quad \theta_j := \theta_j - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

(simultaneously update all θ_j)

}

Notice: this equation is identical to linear regression. But they have different meaning.

for linear regression:

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

for logistic regression:

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

the vectorized Gradien Descent:

$$\theta := \theta - \alpha \cdot \boxed{\frac{1}{m} x^T (g(x\theta) - \vec{y})}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$.

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

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

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

Advanced Optimization:

Optimization algorithms:

- Conjugate gradient
- BFGS. 变尺度法
- L-BFGS. 限制变尺度法.

Pros: - No need to manually pick α.
- often faster than gradient descent.

Cons: - more complex.

Example

$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ $J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$

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

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

Logistic Regression:

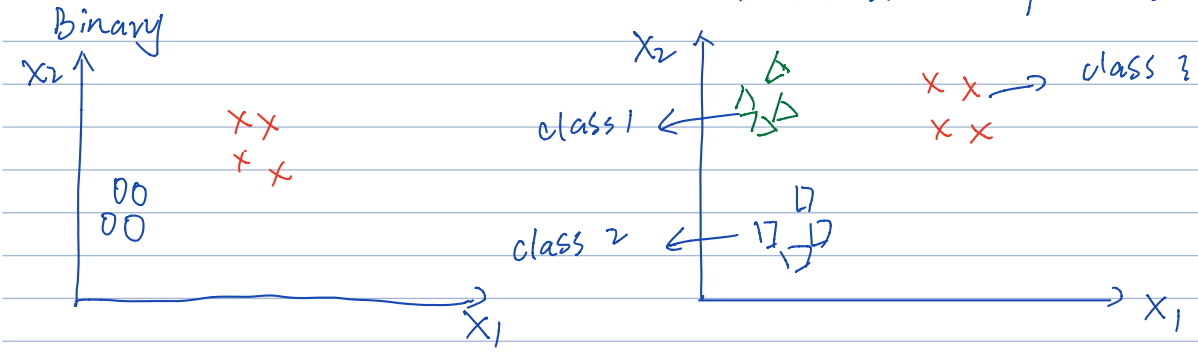
$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ θ_0 ← theta 1, θ_1 ← theta 2

```
function [jVal, gradient] = costFunction(theta):

    jVal = [code to compute J(θ)];
    gradient(1) = [code to compute ∂/∂θ1 J(θ)];
                  :
    gradient(n) = ---           ∂/∂θn J(θ);
```

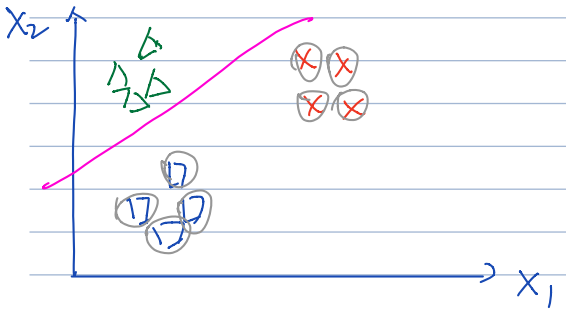
Multi-class classification: one vs all (one-vs-rest)

multiclass classifications.



Train a logistic regression classifier $h_\theta(x^{(i)})$ for each class i to predict the probability that $y=i$ 我们将每个类分别一个标记为正向类，另外的标记为负向类.

On a new input x, to make a prediction, pick the class i that maximizes $\max_i h_\theta^{(i)}(x)$ 输入一个x值，在三个分类器中输入x值，然后我们选择一个让 $h_\theta^{(i)}(x)$ 最大的 i，即 $\max_i h_\theta^{(i)}(x)$



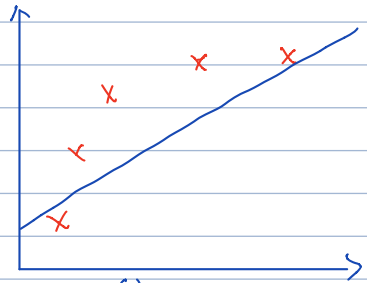
assume, the value of triangles is 1, the circle is 0.

$h_\theta(x)$ to classify the class 1.

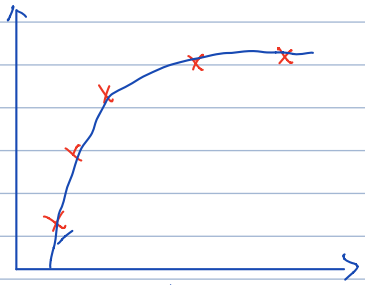
Similarly, we can seperate the other two classes.

Regularization:

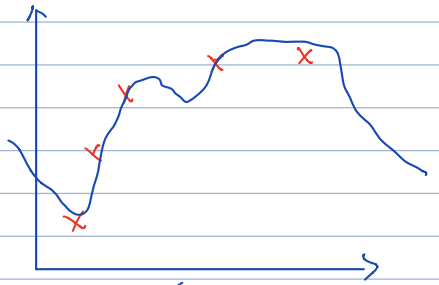
example: linear function (housing price)



$\theta_0 + \theta_1 x$



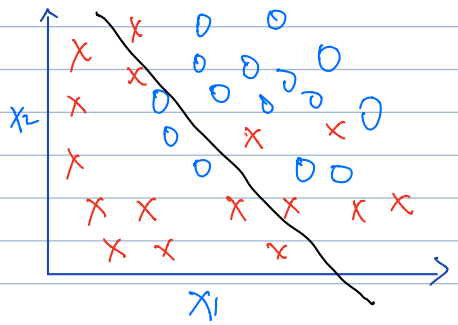
$\theta_0 + \theta_1 x + \theta_2 x^2$



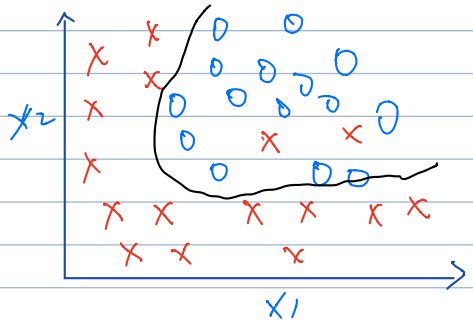
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

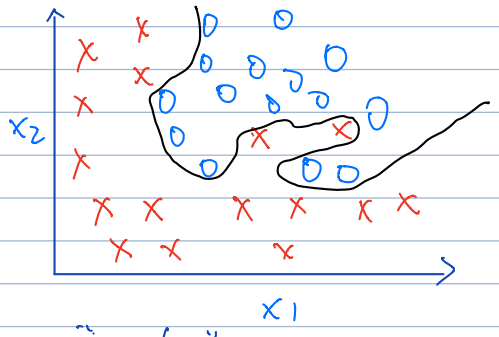
"underfit" "high bias" "overfit" "High variance"

Overfit: If we have too many features, the learned hypothesis may fit the training set very well ($J_\theta = \frac{1}{2m}\sum(h_\theta(x) - y)^2 \approx 0$), but fail to generalize to new examples.



"underfit".





"overfit"

Underfit: When the form of hypothesis function h maps poorly to trend of the data.

Address overfitting problem.

option 1: reduce the number of features:
- manually select which features to keep.
- use a model selection algorithm (e.g. PCA)

option 2: Regularization
- keep all the features, but reduce the magnitude of parameters θ_j.
- Regularization works well when we have a lot of slightly useful features.

regularization:

Small values for parameters $\theta_0, \theta_1, \cdots, \theta_n$
- simpler hypothesis ← penalize.
- less prone to overfitting

example: Housing:
- Features: $x_1, x_2, \cdots, x_{100}$
- Parameters: $\theta_0, \theta_1, \theta_2, \cdots, \theta_{100}$.

we modify the cost function to shrink all parameters. → Penalize

$$J(\theta) = \frac{1}{2m}\left[\sum (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{100} \theta_j\right]$$

不知道要对哪些特征进行惩罚

$\min J(\theta)$

λ: regularization parameter

control trade off between two different goals:
① fit the training data well;
② keep parameters small.

if λ is very large (if $\lambda = 10^{10}$), 一条水平线.

we will penalize $\theta_1, \theta_2 \cdots \theta_{100}$ ..
then $\theta_1 = 0, \theta_2 = 0 \cdots \theta_{100} = 0$

$h_\theta(x) = \theta_0$ → underfit.

正是 $h_\theta(x)$ 中的高次项导致了过拟合的产生，所以我们要做的就是减小这些 θ 的值.

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

我们在 Cost Function 中设置一点 penalty.

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

regularized linear regression:

We will modify the gradient descent to separate out θ_0 from the rest of parameters:

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

第 0 个特征 θ_0 对应的 $x^{(i)}$ 在每个样本中.

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

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

$1 - \alpha\frac{\lambda}{m} < 1$: Reducing the value of θ_j by some amount on every update.

Grad: grad[0] = $\frac{1}{m} * X[:,1]^T * (h_\theta(x^{(i)}) - y^{(i)})$

Normal equation:

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

$$L = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}_{(n+1)\times(n+1)}$$

if $m < n$, $X^T X$ is non-invertible, when add the term λL, $X^T X + \lambda L$ becomes invertible.

Regularized logistic Regression:

Cost Function:

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

Gradient descent

Repeat {

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

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m}\theta_j\right] \quad i = 1, 2, \cdots, n.$$

Regularized logistic Regression:

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