Logistic Regression Model

Cost Function

For linear regression, we had our cost function as:

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

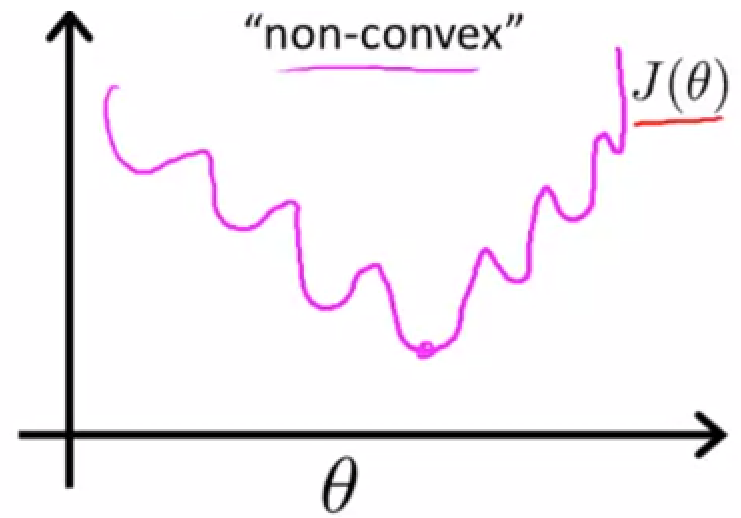
We'll use an alternateway of writing the cost function. Instead of using the squared error term as above, we'll use:

$$J(\theta) = \frac{1}{m} \cdot \sum_{i=1}^{m} cost(h_\theta(x^{(i)}), y^{(i)})$$
$$=> cost(h_\theta(x), y) = \frac{1}{2}(h_\theta(x) - y)^2$$

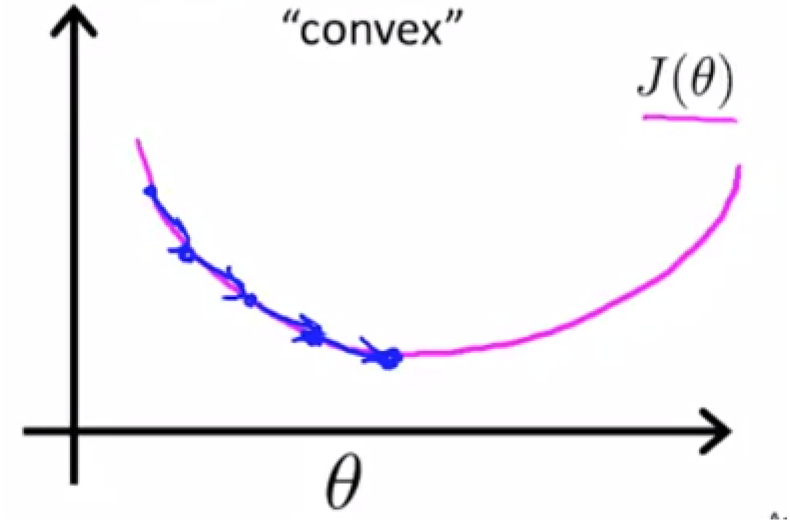
If we plot this function for logistic regression, when

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

We'll get a plot of non-convex function like this:

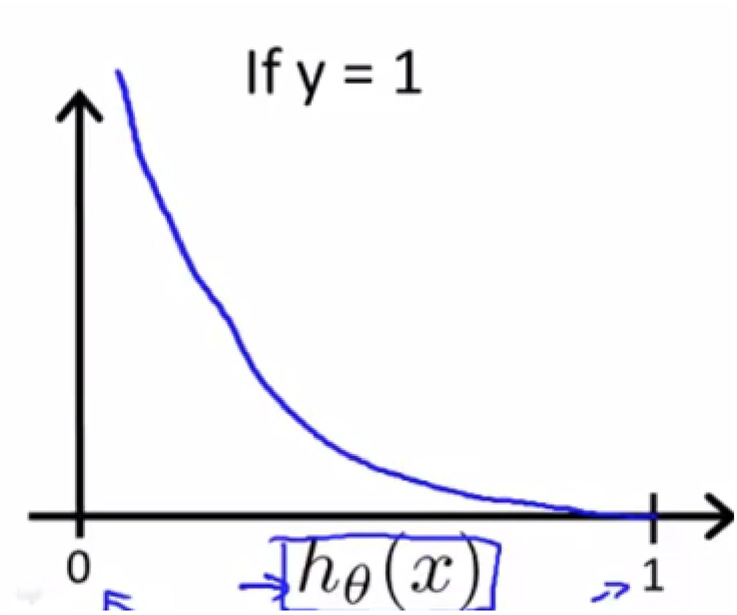


This function will have many local optima, and we cannot run gradient descent on this sort of function. Instead, we want a convex function which will guarantee that gradient descent will converge on the global minimum.



We'll therefore use,

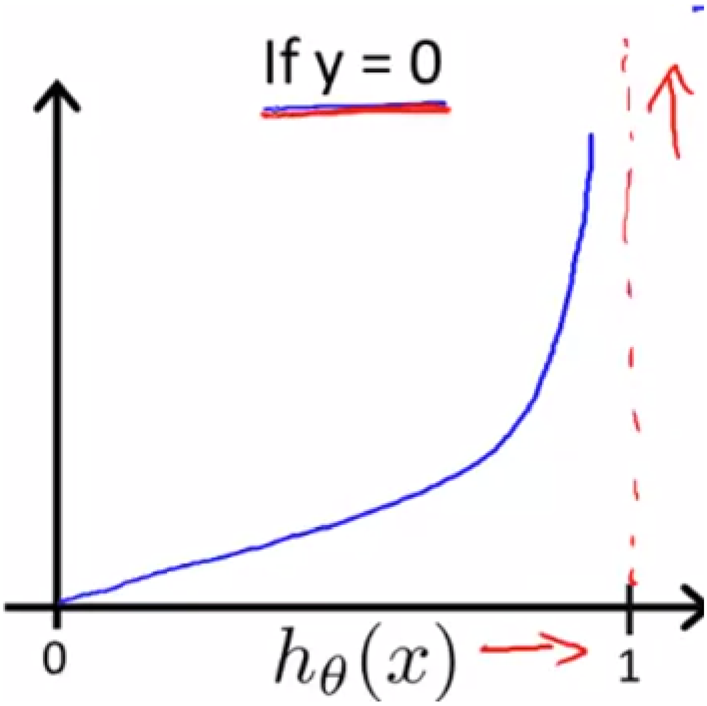
$$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



If $y = 1$ and $h_\theta(x) = 1$, then $cost = 0$

But as $h_\theta(x) \to 0$, $cost \to \infty$

It captures intuition that if $h_\theta(x) = 0$, [predict $P(y = 1|x; \theta) = 0$], but $y = 1$, we'll penalize the learning algorithm by a very large cost. It's like the probability of a patient to have a malignant tumor is 0, even though the value of $y = 1$.



Therefore,

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

> In logistic regression, the cost function for our hypothesis outputting (predicting) $h_\theta(x)$ on a training example that has label $y \epsilon \{0, 1\}$ is:
>
> $$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$
>
> Which of the following are true? Check all that apply.
>
> - [x] If $h_\theta(x) = y$, then $cost(h_\theta(x), y) = 0$ (for $y = 0$ and $y = 1$)
> - [x] If $y = 0$, then $cost(h_\theta(x), y) \rightarrow \infty$ as $h_\theta(x) \rightarrow 1$
> - [ ] If $y = 0$, then $cost(h_\theta(x), y) \rightarrow \infty$ as $h_\theta(x) \rightarrow 0$
> - [x] Regardless of whether $y = 0$ or $y = 1$, if $h_\theta(x) = 0.5$, then $cost(h_\theta(x), y) > 0$

Simplifying the Cost Function & Gradient Descent

We have our cost function:

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

We can write the cost function in a simpler way as follows:

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

Therefore,

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

To fit the parameters θ, minimize $J(\theta)$ for θ

To give a prediction from new x, output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

Gradient Descent

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

We want to minimize θ in $J(\theta)$:

repeat {

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

} [Simultaneously update all θ_j]

This update rule is exactly the same as Linear Regression, except the fact that the hypothesis has changed

θ updates on every iteration:

$$
\begin{aligned}
\theta_0 &:= \theta_0 - \alpha \cdot \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\
\theta_1 &:= \theta_1 - \alpha \cdot \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \\
\theta_2 &:= \theta_2 - \alpha \cdot \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)} \\
&\vdots \\
\theta_n &:= \theta_n - \alpha \cdot \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_n^{(i)}
\end{aligned}
$$

Vectorized implementation of θ updates (instead of using a 'for' loop):

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

$$
(or)
$$

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

Advanced Optimization

Let's revise what gradient descent actuatlly does:

- We have a cost function $J(\theta)$ and we want to minimize θ.
- For minimizing θ, we write code that can compute:
    - $J(\theta)$
    - $\frac{\partial}{\partial\theta_j}J(\theta)$ [For $j = 0, 1, 2, \cdots, n$]
- The above gets plugged into gradient descent

Optimization algorithms which can be used:

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

The three other algorithms doesn't need manual selection of the learning rate α and often converges much faster than gradient descent. However, they're more complex.

Let's consider an example,

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

For implementation in Octave,

```
function [jVal, gradient] = costFunction(theta)
    jVal = (theta(1)-5)^2 + (theta(2)-5)^2;
    gradient = zeros(2,1);
    gradient(1) = 2*(theta(1)-5);
    gradient(2) = 2*(theta(2)-5);
```

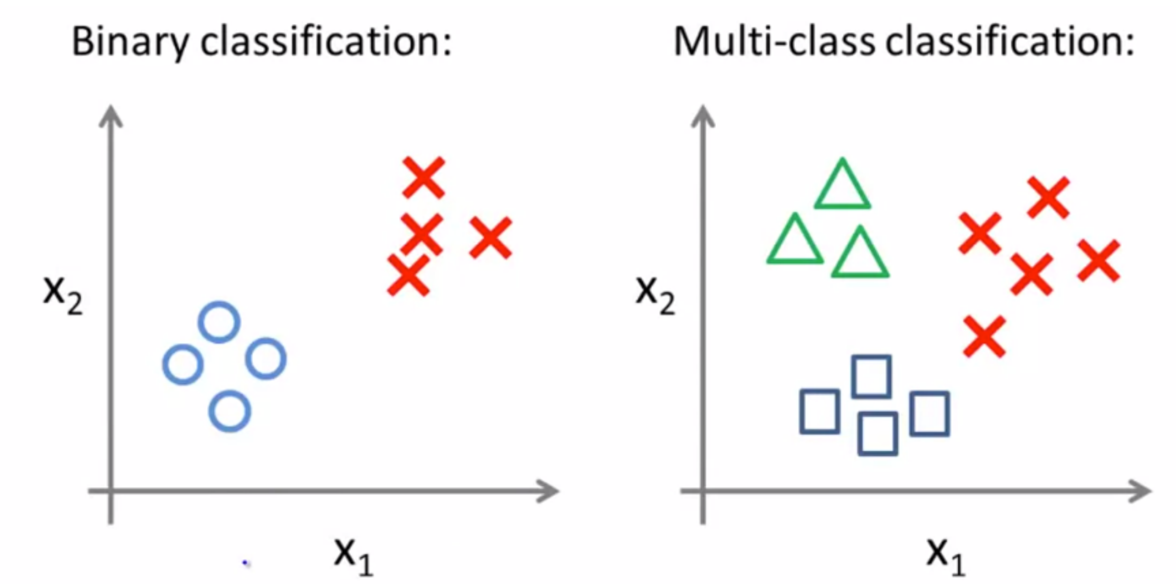
Now to call the advanced optimization functions in Octave,

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

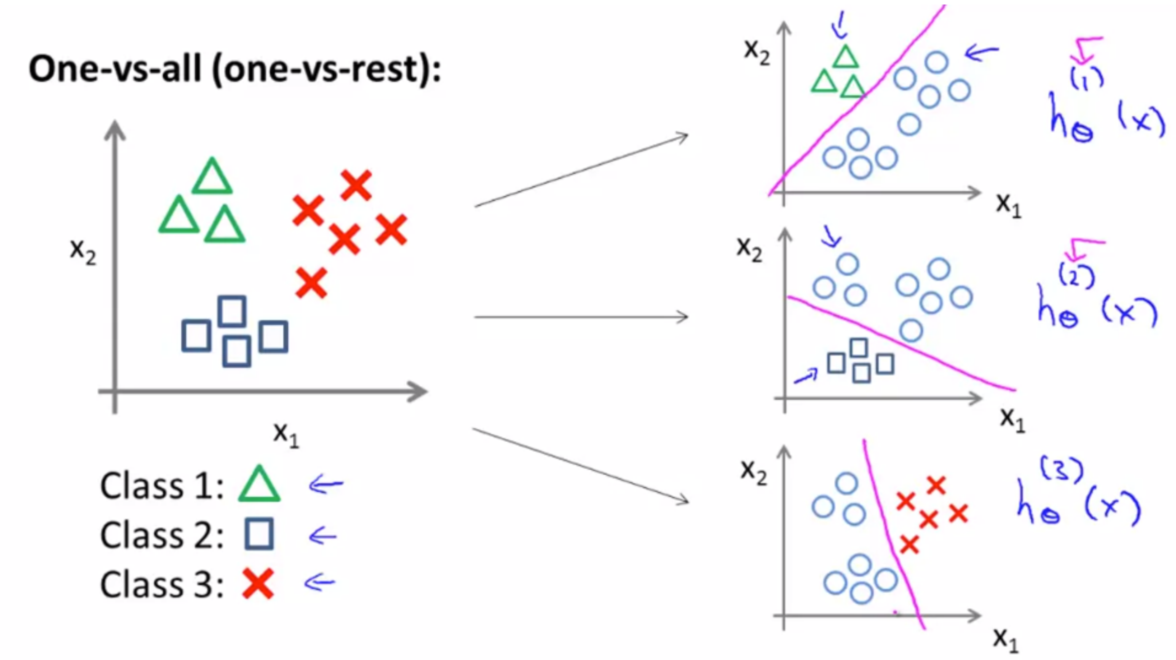
Multiclass Classification

Suppose we have classification problems such as email tagging or classifying weather, y can have different values for each type, such as:

Example	y=1	y=2	y=4	y=4
Email Tagging	Work	Friends	Family	Promotions
Medical Diagnosis	No Disease	Cold	Flu	
Weather	Sunny	Rain	Snow	Cloudy



One vs Rest



We can have a problem where we might want to differentiate one class from the rest. We will have a hypothesis such as:

$$h_\theta^{(i)}(x) = (y = i \mid x; \theta) \qquad (i = 1, 2, 3)$$

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 a prediction, pick the class i that maximizes:

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