Machine Learning by Andrew Ng
pdf by CarolusRex

Logistic Regression

Logistic Regression

In fact, it's a **classification problem**. It is named that way for historical reasons.

Binary Classification

Definition: The value Y is a discrete number. Now it only has two values: **0** or **1**.
Target: Given the input X, predict which set the output Y belongs.

One method we have learned is linear regression. However, the output maybe don't belong the range $[0, 1]$. On the other hand, the method doesn't work well when the model isn't linear.

Sigmoid Function / Logistic Function

Hypothesis: $0 \leqslant h_\theta(x) \leqslant 1$
Model:
$h_\theta(x) = g(\theta^T x)$
$z = \theta^T x$
$g(z) = \frac{1}{1+e^{-z}}$

Explication: $h_\theta(x)$ is the probability that the output $Y = 1$. It also can be expressed by follow: $h_\theta(x) = P(y = 1|x; \theta)$

Decision Boundary

$h_\theta(x) \geqslant 0.5 \rightarrow y = 1$
$h_\theta(x) < 0.5 \rightarrow y = 0$
Replace $h_\theta(x)$ by $\theta^T x$, we can find that:
$\theta^T x \geqslant 0 \rightarrow y = 1$
$\theta^T x < 0 \rightarrow y = 0$

CostFunction

The CostFunction of linear regreesion maybe cause many local optima when it's used here, which made the θ is trapped in local minimum. It will not be a convex function.
The CostFunction we choose is not arbitary. In fact, it's the log of probability, which has statistically significance.

$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$
$Cost(h_\theta(x), y) = -log(h_\theta(x)) \; if \; y = 1$
$Cost(h_\theta(x), y) = -log(1 - h_\theta(x)) \; if \; y = 0$
When $y = 1 \; \& \; h_\theta(x) = 0$ or $y = 0 \; \& \; h_\theta(x) = 1$, the costFunction will be infinity.

Simplified CostFunction & Gradient Descend Derivation

$Cost(h_\theta(x), y) = -ylog(h_\theta(x)) - (1 - y)log(1 - h_\theta(x))$
Let $h = g(X\theta)$, we have $J(\theta) = \frac{1}{m} \cdot (-y^T log(h) - (1 - y)^T log(1 - h))$
Let $\sigma(x) = \frac{1}{1+e^{-x}}$, then $\sigma'(x) = (\frac{1}{1+e^{-x}})' = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1+e^{-x}-1}{(1+e^{-x})^2} = \sigma(x)(1 - \sigma(x))$

$$
\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\theta) =& \frac{\partial}{\partial\theta_j}\frac{-1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)})) + (1-y^{(i)})log(1-h_\theta(x^{(i)}))] \\
=& -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\frac{\partial}{\partial\theta_j}log(h_\theta(x^{(i)})) + (1-y^{(i)})\frac{\partial}{\partial\theta_j}log(1-h_\theta(x^{(i)}))] \\
=& -\frac{1}{m}\sum_{i=1}^{m}[\frac{y^{(i)}\frac{\partial}{\partial\theta_j}h_\theta(x^{(i)})}{h_\theta(x^{(i)})} - \frac{(1-y^{(i)})\frac{\partial}{\partial\theta_j}(1-h_\theta(x^{(i)}))}{1-h_\theta(x^{(i)})}] \\
=& -\frac{1}{m}\sum_{i=1}^{m}[\frac{y^{(i)}h_\theta(x^{(i)})(1-h_\theta(x^{(i)}))\frac{\partial}{\partial\theta_j}\theta^Tx^{(i)}}{h_\theta(x^{(i)})} - \frac{(1-y^{(i)})(1-h_\theta(x^{(i)}))h_\theta(x^{(i)})\frac{\partial}{\partial\theta_j}\theta^Tx^{(i)}}{1-h_\theta(x^{(i)})}] \\
=& -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}(1-h_\theta(x^{(i)}))x_j^{(i)} - (1-y^{(i)})h_\theta(x^{(i)})x_j^{(i)}] \\
=& -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} - h_\theta(x^{(i)})]x_j^{(i)} \\
=& \frac{1}{m}\sum_{i=1}^{m}[h_\theta(x^{(i)}) - y^{(i)}]x_j^{(i)}
\end{aligned}
$$

The result is the same as linear regreesion. The reason why it happens isn't discussed in this chapter.

Advanced Optimization

Some more sophisticated, faster method to optimize θ: **Conjugate gradient**, **BFGS**, **L-BFGS**.

Multiclass Classification: One-vs-all

$y \in \{0, 1, ..., n\}$
Make use of **binary classification**.
For set i, we define in the set is **1**, out the set if **0** and excute binary classification. The max probability is the set we predict.
$h_\theta^{(0)}(x) = P(y=0|x;\theta)$
$h_\theta^{(1)}(x) = P(y=1|x;\theta)$
...
$h_\theta^{(n)}(x) = P(y=n|x;\theta)$
$prediction = max(h_\theta^{(i)}(x))$

Regularization

When optimize θ, we often face the problem of overfitting, especially when the training samples is not enough and smaller than the number of parameters. It will cause the good effect on train set, but the bad effect on test set.

There are two main options to address the issue of overfitting.

1. Reduce the number of features.
   a. Manually select which features to keep;
   b. Use a model selection algorithm(not in this chapter)
2. Regularizaion
   a. Keep all the features, but reduce the parameters θ_j

Regularization works well when we have lots of slightly useful features.
To eliminate the influence of θ_j, we can set a large coefficient λ. To make the CostFunction $\lambda\theta_j$ small, the θ_j will be small.
The CostFunction will be $J(\theta) + \lambda \sum_{j=1}^{n} \theta_j^2$

Regularized Linear Regression

Gradient Descent

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

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

Normal Equation

$\theta = (X^TX + \lambda \cdot L)^{-1}X^Ty$
L is an I_{n+1} but $I_{0,0} = 0$
If $m \leqslant n$, then X^TX is non-invertible, however $X^TX + \lambda \cdot L$ is invertible.

Regularized Logistic Regression

CostFunction

$J(\theta) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

Gradient Descent

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

Initial Ones Feature Vector

I don't understand well in this section.