Logistic Regression

May 28, 2012

1 Logistic Regression

1.1 Decision Boundary

Decision boundary is a property of the hypothesis and not the data set.

$$h_\Theta(x) = g(\Theta^\top x) = P(y = 1|x; \Theta)$$

sigmoid function:

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

suppose predict $y = 1 \text{if} h_\Theta(x) \geq 0.5$

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

$$h_\Theta(x) = g(\Theta^\top x) \geq 0.5 \, whenever \, \Theta^\top x \geq 0$$

suppose predict $y = 0 \text{if } h_\Theta(x) < 0.5$

$$h_\Theta(x) = g(\Theta^\top x) < 0.5 \, whenever \, \Theta^\top x < 0$$

Non Linear Decision Boundaries
Add higher order polynomials, 2 exra features.

$$h_\Theta(x) = g(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_1^2 + \Theta_4 x_2^2)$$
$$\Theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

predict $y = 1 \text{if} -1 + x_1^2 + x_2^2 \geq 0$, that is $x_1^2 + x_2^2 \geq 1$
if we plot $x_1^2 + x_2^2 = 1$, it's a circle. Inside of the circle, $y = 0$, outside $y = 1$.

With even higher polynomial terms, we can get even more complex decision boundaries.

1.2 Cost Function

Training set:

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

m examples:

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ ... \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \;\; x_0 = 1,\, y \in \{0, 1\}$$

$$h_\Theta(x) = \frac{1}{1 + e^{-\Theta^\top x}}$$

How do we choose parameters Θ?
Cost function for logistic regression, needs to be convex and a cost function for linear regression would not be convex. Instead, use (6:20):

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

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

1.3 Simplified cost function and gradient descent

Logistic regression cost function

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

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

Note: $y = 0\, or\, 1$always
We can rewrite the cost function:

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

So then if $y = 1$or if $y = 0$, this function loses terms accordingly.
Now we can rewrite the cost function like so:

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

To fit parameters Θ:

$$\min_{\Theta} J(\Theta)$$

To make a prediction given new x:
Output

$$h_\Theta(x) = \frac{1}{1 + e^{-\Theta^\top x}}$$

which we interpret as probability that $y = 1$

$$p(y = 1|x; \Theta)$$

We're going to minimize the cost function with **Gradient Descent.**

$$\min_{\Theta} J(\Theta) : \ \mathit{Repeat}\, \{$$

$$\Theta_j = \Theta_j - \alpha \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\}$$

$$\Theta = \begin{bmatrix} \Theta_0 \\ \Theta_1 \\ \vdots \\ \Theta_n \end{bmatrix}$$

Vectorized:

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

This is not the same as Gradient Descent for linear regression, because the definition of $h_\Theta(x)$has changed.

Apply the method from linear regression to make sure logistic regression is converging, plot cost.

Note:
Recall:

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

Vectorized:

$$J(\Theta) = \frac{1}{m} \left[-y^\top \log h_\Theta(X) - (1 - y)^\top \log(1 - h_\Theta(X)) \right]$$

The gradient of the cost is a vector of the same length as $\Theta$where the j^{th}element for $j = (0, 1, 2, \ldots, n)$ is defined as follows:

$$\frac{\partial J(\Theta)}{\partial \Theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Vectorized:

$$\frac{\partial J(\Theta)}{\partial \Theta_j} = \frac{1}{m} X^\top h_\Theta(X) - y$$

1.4 Advanced Optimization

Given Θ, we have code that can compute:

- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_j} J(\Theta)$ (for $j = 0, 1, \ldots, n$)

We can use these to minimize the cost function:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

- No need to manually pick α
  - Line search algorithm automatically picks α, even for each iteration.
- Often faster than gradient descent

Disadvantages:

- More complex

Example:
$\Theta = \begin{bmatrix} \Theta_1 \\ \Theta_2 \end{bmatrix}$
$\Theta_1 = 5$
$\Theta_2 = 5$
$J(\Theta) = (\Theta_1 - 5)^2 + (\Theta_2 - 5)^2$
$\frac{\partial}{\partial \Theta_1} J(\Theta) = 2(\Theta_1 - 5)$
$\frac{\partial}{\partial \Theta_2} J(\Theta) = 2(\Theta_1 - 5)$

```
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(1) = 2 * (theta(2) - 5);
```

jVal gets the result of the cost function
gradient gets a 2x1 vector, the terms of which correspond to the two derivative terms.
We can then call the advanced optimization function, fminunc, function minimization unconstrained. Chooses optimal value of Θautomagically.

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

@ represents a pointer to the above costFunction
set some options:

- 'GradObj', 'on' means that we will provide a gradient
- 'MaxIter', '100' means the number of iterations

$initialTheta \in \mathbb{R}^d$, $d \geq 2$

How do we apply this to logistic regression?

$$theta = \begin{bmatrix} \Theta_0 \\ \Theta_1 \\ \vdots \\ \Theta_n \end{bmatrix}$$

function [jVal, gradient] = costFunction(theta)
jval = [code to compute $J(\Theta)$];
gradient(1) = [code to compute $\frac{\partial}{\partial\Theta_0}J(\Theta)$];
gradient(2) = [code to compute $\frac{\partial}{\partial\Theta_1}J(\Theta)$];
$\vdots$
gradient(n+1) = [code to compute $\frac{\partial}{\partial\Theta_n}J(\Theta)$];

1.5 Multi-class classification: One-vs-all

How to get logistic regression to get to work with multi-class classification problems.
Scenario 1: tag emails for different folders
Scenario 2: Medical diagrams: Not ill, cold, flu
Scenario 3: Weather: Sunny, Cludy, Rainy, Snow

There will be more groups of items on plot.
Turn the problem into 3 different binary classification problems.
We'll create a "fake" trainig set, where Class 1 gets assigned to a positive class and Class 2 and 3 get assigned to the negative class.
$h_\Theta^{(1)}(x)$will denote, say triangles
$h_\Theta^{(2)}(x)$will denote, say squares
$h_\Theta^{(2)}(x)$will denote, say x's

We fit 3 classifiers that try to estimate what is the probability that $y = i$

$$h_\Theta^{(i)} = P(y = i|x; \Theta)\,(i = 1, 2, 3)$$

Train a logistic regression classifier $h_\Theta^{(i)} x$ for each class i to predict the probability that $y = 1$.
On a new input x, to make a prediction, pick the class i that maximizes

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