Logistic classification

Logistic regression:

ML:Logistic Regression

Now we are switching from regression problems to **classification problems**. Don't be confused by the name "Logistic Regression"; it is named that way for historical reasons and is actually an approach to classification problems, not regression problems.

Binary Classification

Instead of our output vector y being a continuous range of values, it will only be 0 or 1.

y∈{0,1}

Where 0 is usually taken as the "negative class" and 1 as the "positive class", but you are free to assign any representation to it.

We're only doing two classes for now, called a "Binary Classification Problem."

One method is to use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. This method doesn't work well because classification is not actually a linear function.

Hypothesis Representation

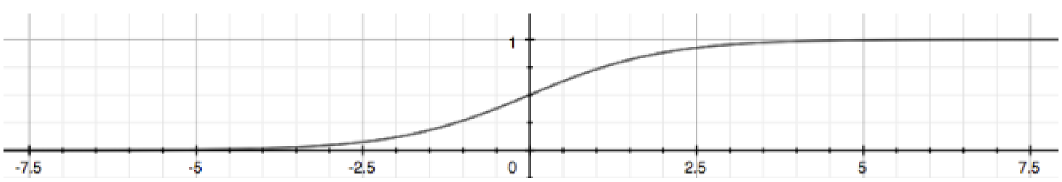
Our hypothesis should satisfy:

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

Our new form uses the "Sigmoid Function," also called the "Logistic Function":

Our new form uses 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 function g(z), shown here, maps any real number to the (0, 1) interval, making it useful for transforming an arbitrary-valued function into a function better suited for classification. Try playing with interactive plot of sigmoid function : (https://www.desmos.com/calculator/bgontvxotm).

We start with our old hypothesis (linear regression), except that we want to restrict the range to 0 and 1. This is accomplished by plugging $\theta^T x$ into the Logistic Function.

h_θ will give us the **probability** that our output is 1. For example, $h_\theta(x) = 0.7$ gives us the probability of 70% that our output is 1.

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

Our probability that our prediction is 0 is just the complement of our probability that it is 1 (e.g. if probability that it is 1 is 70%, then the probability that it is 0 is 30%).

Logistic classification

Decision Boundary

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:

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

Remember.

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

So if our input to g is $\theta^T X$, then that means:

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

From these statements we can now 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 our hypothesis function.

Example:

$$\theta = \begin{matrix} 5 \\ -1 \\ 0 \end{matrix}$$
$$y = 1\ if\ 5 + (-1)x_1 + 0x_2 \geq 0$$
$$5 - x_1 \geq 0$$
$$-x_1 \geq -5$$
$$x_1 \leq 5$$

In this case, our decision boundary is a straight vertical line placed on the graph where $x_1 = 5$, and everything to the left of that denotes y = 1, while everything to the right denotes y = 0.

Again, the input to the sigmoid function g(z) (e.g. $\theta^T X$) doesn't need to be linear, and could be a function that describes a circle (e.g. $z = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$) or any shape to fit our data.

Logistic classification

Cost Function

We cannot use the same cost function that we use for linear regression because the Logistic Function will cause the output to be wavy, causing many local optima. In other words, it will not be a convex function.

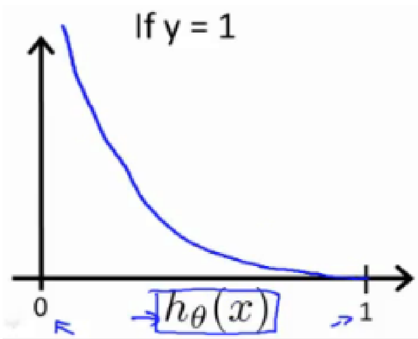
Instead, our cost function for logistic regression looks like:

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

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

$$\mathrm{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \quad \text{if } y = 0$$

When y = 1, we get the following plot for $J(\theta)$ vs $h_\theta(x)$:



Similarly, when y = 0, we get the following plot for $J(\theta)$ vs $h_\theta(x)$:

Cost function

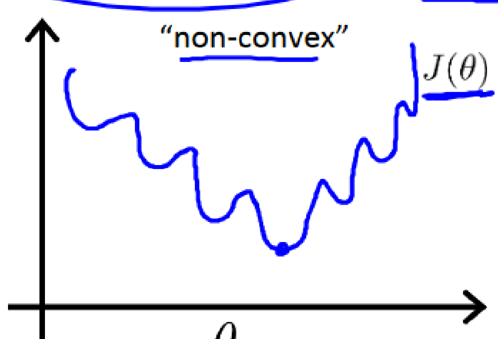
~~Linear~~ regression: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \cancel{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}$

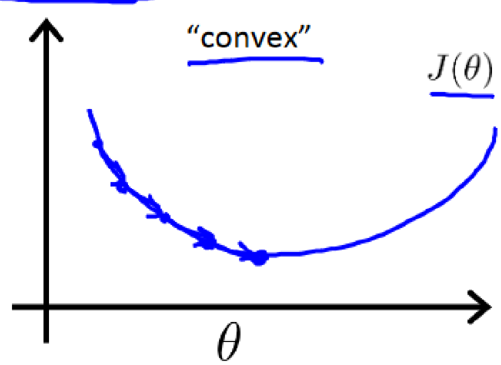
logistic

$\mathrm{cost}(h_\theta(x^{(i)}), y)$

$\mathrm{Cost}(h_\theta(x), y) = \frac{1}{2}\left(h_\theta(x) - y\right)^2$

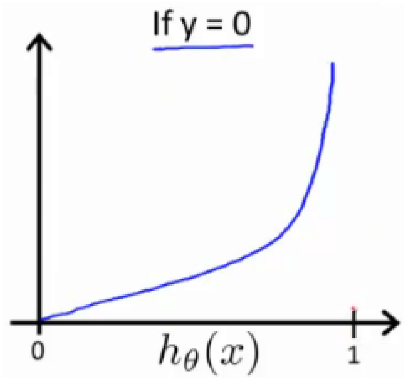
$\frac{1}{1+e^{-\theta^T x}}$





An

Logistic classification



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

If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity.

If our correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function outputs 1. If our 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.

Simplified Cost Function and Gradient Descent

We can compress our cost function's two conditional cases into one case:

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

Notice that when y is equal to 1, then the second term $(1-y)\log(1-h_\theta(x))$ will be zero and will not affect the result. If y is equal to 0, then the first term $-y\log(h_\theta(x))$ will be zero and will not affect the result.

We can fully write out our entire cost function as follows:

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

A vectorized implementation is:

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

Gradient Descent

Remember that the general form of gradient descent is:

$$\begin{aligned}&Repeat\ \{ \\ &\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \\ &\}\end{aligned}$$

Logistic classification

We can work out the derivative part using calculus to get:

$$\begin{aligned}&\textit{Repeat}\ \{\\ &\quad \theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\\ &\}\end{aligned}$$

Notice that this algorithm is identical to the one we used in linear regression. We still have to simultaneously update all values in theta.

A vectorized implementation is:

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

Partial derivative of J(θ)

First calculate derivative of sigmoid function (it will be useful while finding partial derivative of J(θ)):

$$\begin{aligned}\sigma(x)' &= \left(\frac{1}{1+e^{-x}}\right)' = \frac{-(1+e^{-x})'}{(1+e^{-x})^2} = \frac{-1' - (e^{-x})'}{(1+e^{-x})^2} = \frac{0 - (-x)'(e^{-x})}{(1+e^{-x})^2} = \frac{-(-1)(e^{-x})}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2}\\ &= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{e^{-x}}{1+e^{-x}}\right) = \sigma(x)\left(\frac{+1-1+e^{-x}}{1+e^{-x}}\right) = \sigma(x)\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) = \sigma(x)(1-\sigma(x))\end{aligned}$$

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

Logistic classification

The vectorized version;

$$\nabla J(\theta) = \frac{1}{m} \cdot X^T \cdot \left(g\left(X \cdot \theta\right) - \vec{y}\right)$$

hypothesis alag hai

Regularization:

Cost Function

Note: [5:18 - There is a typo. It should be $\sum_{j=1}^{n} \theta_j^2$ instead of $\sum_{i=1}^{n} \theta_j^2$]

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

Say we wanted to make the following function more quadratic:

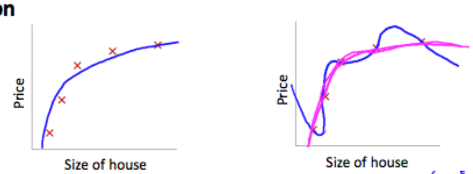
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

We'll want to eliminate the influence of $\theta_3 x^3$ and $\theta_4 x^4$. Without actually getting rid of these features or changing the form of our hypothesis, we can instead modify our **cost function**:

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

We've added two extra terms at the end to inflate the cost of θ_3 and θ_4. Now, in order for the cost function to get close to zero, we will have to reduce the values of θ_3 and θ_4 to near zero. This will in turn greatly reduce the values of $\theta_3 x^3$ and $\theta_4 x^4$ in our hypothesis function. As a 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 $\theta_3 x^3$ and $\theta_4 x^4$.

Intuition



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

Suppose we penalize and make θ_3, θ_4 really small.

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

$\theta_3 \approx 0 \quad \theta_4 \approx 0$

No effect at all in our cost function

We could also regularize all of our theta parameters in a single summation as:

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

The λ, or lambda, is the **regularization parameter**. It determines how much the costs of our theta parameters are inflated.

Using the above cost function with the extra summation, we can smooth the output of our hypothesis function to reduce overfitting. If lambda is chosen to be too large, it may smooth out the function too much and cause underfitting. Hence, what would happen if $\lambda = 0$ or is too small?

Logistic classification

In regularized linear regression, we choose θ to minimize:

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

What if λ is set to an extremely large value (perhaps too large for our problem, say $\lambda = 10^{10}$)

- ◯ Algorithm works fine; setting λ to be very large can't hurt it.
- ◯ Algorithm fails to eliminate overfitting.
- ◉ Algorithm results in underfitting (fails to fit even the training set).

Regularized Linear Regression

We can apply regularization to both linear regression and logistic regression. We will approach linear regression first.

Gradient Descent

We will modify our gradient descent function to separate out θ_0 from the rest of the parameters because we do not want to penalize θ_0.

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

The term $\frac{\lambda}{m}\theta_j$ performs our regularization.

With some manipulation our update rule can also be represented as:

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

The first term in the above equation, $1 - \alpha\frac{\lambda}{m}$ will always be less than 1. Intuitively you can see it as reducing the value of θ_j by some amount on every update.

Notice that the second term is now exactly the same as it was before.

Normal Equation

Now let's approach regularization using the alternate method of the non-iterative normal equation.

To add in regularization, the equation is the same as our original, except that we add another term inside the parentheses:

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

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

Recall that if m ≤ n, then X^TX is non-invertible. However, when we add the term λ⋅L, then X^TX + λ⋅L becomes invertible.

Logistic classification

Regularized Logistic Regression

We can regularize logistic regression in a similar way that we regularize linear regression. Let's start with the cost function.

Cost Function

Recall that our cost function for logistic regression was:

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

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

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

Note Well: The second sum, $\sum_{j=1}^{n}\theta_j^2$ **means to explicitly exclude** the bias term, θ_0. I.e. the θ vector is indexed from 0 to n (holding n+1 values, θ_0 through θ_n), and this sum explicitly skips θ_0, by running from 1 to n, skipping 0.

Gradient Descent

Just like with linear regression, we will want to **separately** update θ_0 and the rest of the parameters because we do not want to regularize θ_0.

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

This is identical to the gradient descent function presented for linear regression.