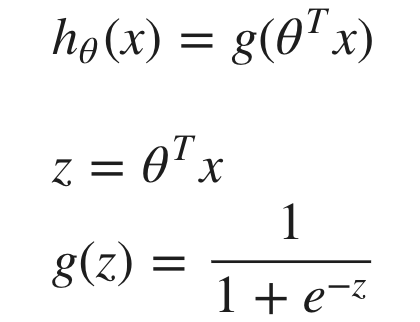
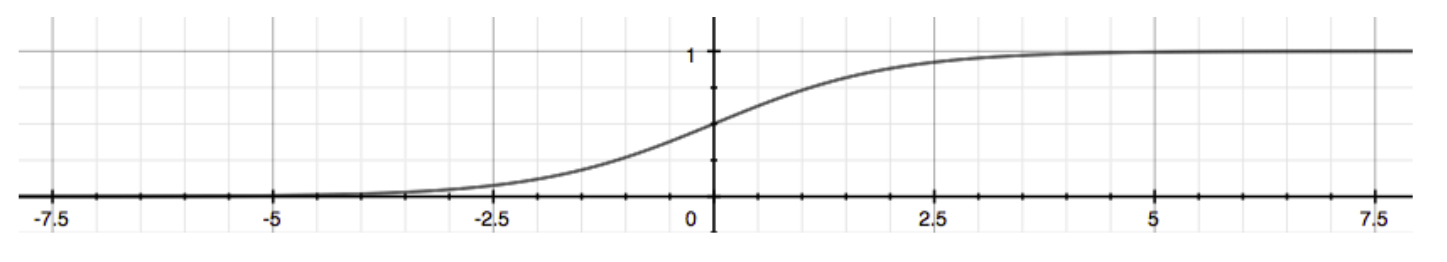
Hypothesis Representation

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



The following image shows us what the sigmoid function looks like:



Decision Boundary

In order to get our discrete 0 or 1 classification, we can translate the output of the hypothesis function as follows:

A close up of a watch

Description automatically generated

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:

A close up of a clock

Description automatically generated

A close up of a clock

Description automatically generated

So if our input to g is *θTX*, then that means:

A picture containing object

Description automatically generated

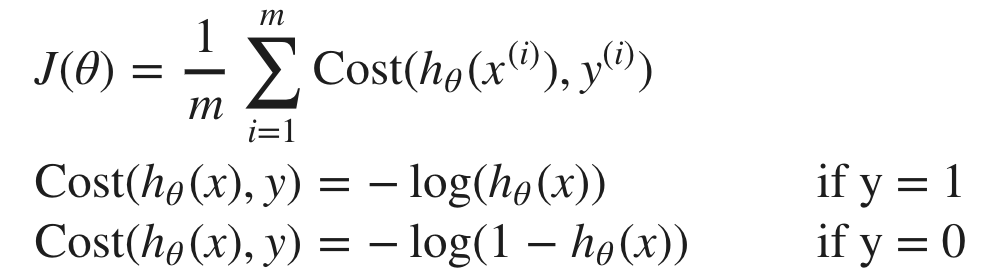
From these statements we can now say:

A close up of a clock

Description automatically generated

In this case, our decision boundary is a straight vertical line placed on the graph where x1 ​=5, and everything to the left of that denotes y = 1, while everything to the right denotes y = 0.

Cost Function



When y = 1, we get the following plot for *J*(*θ*) vs *hθ*​(*x*):

A close up of a logo

Description automatically generated

Similarly, when y = 0, we get the following plot for *J*(*θ*) vs *hθ*​(*x*):

A screenshot of a cell phone

Description automatically generated

A close up of a logo

Description automatically generated

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 fully write out our entire cost function as follows:

A close up of a logo

Description automatically generated

A vectorized implementation is:

# A close up of a logo Description automatically generated

### **Gradient Descent**

A close up of a logo

Description automatically generated

A vectorized implementation is:

A screenshot of text

Description automatically generatedA picture containing object

Description automatically generated

# Advanced Optimization

**"Conjugate gradient", "BFGS", and "L-BFGS"** are more sophisticated, faster ways to optimize θ that can be used instead of gradient descent. We suggest that you should not write these more sophisticated algorithms yourself (unless you are an expert in numerical computing) but use the libraries instead, as they're already tested and highly optimized. Octave provides them.

**regularization**

A close up of a logo

Description automatically generated

**Gradient Descent**

A close up of a logo

Description automatically generated

The term  performs our regularization. With some manipulation our update rule can also be represented as:

A close up of a logo

Description automatically generated

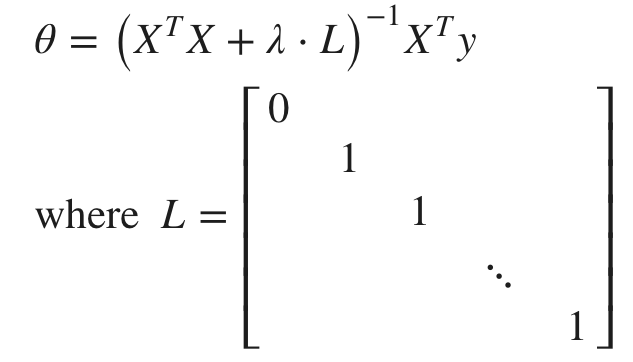
The first term in the above equation,  will always be less than 1. Intuitively you can see it as reducing the value of A close up of a logo

Description automatically generated 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:



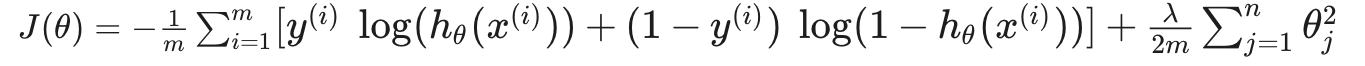
# Regularized Logistic Regression

Recall that our cost function for logistic regression was:

A picture containing object

Description automatically generated

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



Thus, when computing the equation, we should continuously update the two following equations:

A close up of text on a white background

Description automatically generated