Classification

To attempt classification, 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. However, this method doesn't work well because classification is not actually a linear function.

The classification problem is just like the regression problem, except that the values we now want to predict take on only a small number of discrete values. For now, we will focus on the **binary classification** **problem** in which y can take on only two values, 0 and 1. (Most of what we say here will also generalize to the multiple-class case.) For instance, if we are trying to build a spam classifier for email, then x, start superscript, left parenthesis, i, right parenthesis, end superscriptx^{(i)}*x*(*i*) may be some features of a piece of email, and y may be 1 if it is a piece of spam mail, and 0 otherwise. Hence, y∈{0,1}. 0 is also called the negative class, and 1 the positive class, and they are sometimes also denoted by the symbols “-” and “+.” Given x, start superscript, left parenthesis, i, right parenthesis, end superscriptx^{(i)}*x*(*i*), the corresponding y, start superscript, left parenthesis, i, right parenthesis, end superscripty^{(i)}*y*(*i*) is also called the label for the training example.

Hypothesis Representation

We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm to try to predict y given x. However, it is easy to construct examples where this method performs very poorly. Intuitively, it also doesn’t make sense for h, start subscript, theta, end subscript, left parenthesis, x, right parenthesish\_\theta (x)*hθ*​(*x*) to take values larger than 1 or smaller than 0 when we know that y ∈ {0, 1}. To fix this, let’s change the form for our hypotheses h, start subscript, theta, end subscript, left parenthesis, x, right parenthesish\_\theta (x)*hθ*​(*x*) to satisfy 0, is less than or equal to, h, start subscript, theta, end subscript, left parenthesis, x, right parenthesis, is less than or equal to, 10 \leq h\_\theta (x) \leq 10≤*hθ*​(*x*)≤1. This is accomplished by plugging theta, start superscript, T, end superscript, x\theta^Tx*θTx* into the Logistic Function.

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

|  |
| --- |
| *hθ*(*x*)=*g*(*θTx*)*z*=*θTxg*(*z*)=11+*e*−*z* |

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



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.

h, start subscript, theta, end subscript, left parenthesis, x, right parenthesish\_\theta(x)*hθ*​(*x*) will give us the **probability** that our output is 1. For example, h, start subscript, theta, end subscript, left parenthesis, x, right parenthesis, equals, 0, point, 7h\_\theta(x)=0.7*hθ*​(*x*)=0.7 gives us a probability of 70% that our output is 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%).

|  |
| --- |
| *hθ*(*x*)=*P*(*y*=1|*x*;*θ*)=1−*P*(*y*=0|*x*;*θ*)*P*(*y*=0|*x*;*θ*)+*P*(*y*=1|*x*;*θ*)=1 |

Decision Boundary

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

|  |
| --- |
| *hθ*(*x*)≥0.5→*y*=1*hθ*(*x*)<0.5→*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*)≥0.5*whenz*≥0 |

Remember.

|  |
| --- |
| *z*=0,*e*0=1⇒*g*(*z*)=1/2*z*→∞,*e*−∞→0⇒*g*(*z*)=1*z*→−∞,*e*∞→∞⇒*g*(*z*)=0 |

So if our input to g is theta, start superscript, T, end superscript, X\theta^T X*θTX*, then that means:

|  |
| --- |
| *hθ*(*x*)=*g*(*θTx*)≥0.5*whenθTx*≥0 |

From these statements we can now say:

|  |
| --- |
| *θTx*≥0⇒*y*=1*θTx*<0⇒*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**:

|  |
| --- |
| *θ*=⎡⎣5−10⎤⎦*y*=1*if*5+(−1)*x*1+0*x*2≥05−*x*1≥0−*x*1≥−5*x*1≤5 |

In this case, our decision boundary is a straight vertical line placed on the graph where x, start subscript, 1, end subscript, equals, 5x\_1 = 5*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, start superscript, T, end superscript, X\theta^T X*θTX*) doesn't need to be linear, and could be a function that describes a circle (e.g. z, equals, theta, start subscript, 0, end subscript, plus, theta, start subscript, 1, end subscript, x, start subscript, 1, end subscript, start superscript, 2, end superscript, plus, theta, start subscript, 2, end subscript, x, start subscript, 2, end subscript, start superscript, 2, end superscriptz = \theta\_0 + \theta\_1 x\_1^2 +\theta\_2 x\_2^2*z*=*θ*0​+*θ*1​*x*12​+*θ*2​*x*22​) or any shape to fit our data.