Machine Learning: Collectives of local decision makers

Formulating A Suitable Perceptron Paramaterisation

Daniel Braithwaite

April 17, 2017

The goal of this project is to create neurons which insted of searching for a global minumum they will choose a specific part of the data to place their hyperplane. Consider the following situation in $\mathbb{R}^2$ where we have data split into two classes and the second class is boxed into a chevron shape. A single perceptron will go and place a horizonal line through the plane however this isnt telling us anything interesting. Prehaps something better to do is to align its self to one of the sides of the chevron, then if we added another "boundary hunter" it could align its self to the other side of the chevron.

1 Simplified Situation

Before worrying about this in the general case we aim to get a better understanding of the problem by first trying to solve the case described above. We first make some changes to how we have paramaterised our perceptron. We want to find a way which peforms the same as a standard perceptron before we start making it into a boundary hunter

1.1 Point & Gradient Paramaterisation

We are trying to optimise a point (x,y) and a gradient. This is an alternative way to paramaterise our line. We are using sum squared error for our loss function and sigmoid as our activation. Each "boundary hunter" (BH) has the folowing weights vector $W = [m, x_0, y_0]$ and we consider two inputs to our BH, x and y. Our perceptron computes $z = (y - y_0) - m(x - x_0)$ and then outputs $o = f(z) = \frac{1}{1+e^{-z}}$.

1.1.1 Deriving Gradients

$$\frac{\partial E}{\partial W_a} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial W_a} \tag{1}$$

$$\frac{\partial o}{\partial z} = o(1 - o) \tag{2}$$

$$\frac{\partial E}{\partial o} = -(t - o) \tag{3}$$

$$\frac{\partial z}{\partial y_0} = -1 \tag{4}$$

$$\frac{\partial z}{\partial x_0} = m \tag{5}$$

$$\frac{\partial z}{\partial m} = x_0 - x \tag{6}$$

1.1.2 Comparason To Perceptron

If we use the same set of data to train both our standard perceptron and our modified perceptron we see that our modified dosnt quite peform as well as the standard version. For both models below they where trained for 50000 iterations of the training data.
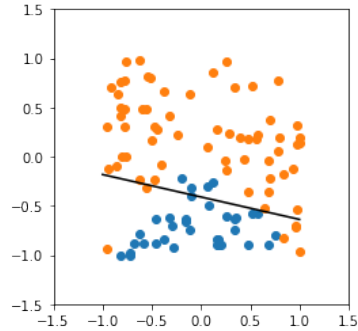

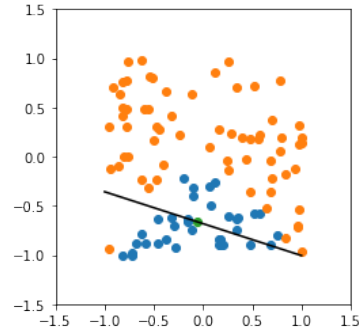Figure 1: Standard Perceptron (SSE = 4.21)

Figure 2: Modified Perceptron (SSE = 7.56)

I propose that this way of constructing our perceptron is less powerful than the standard way. Consider when we are optimising our point-slope representation,

$$y - y_0 = m(x - x_0)$$
$$y = y_0 + mx - mx_0$$
$$y = mx + (y_0 + mx_0)$$

2

We have just shown that (as we would expect) we can convert out point-slope representation into a intercept-slope representation. From here we see we can convert into something that could be represented by our origonal perceptron.

$$y = mx + (y_0 + mx_0)$$
$$\Rightarrow -(y_0 + mx_0) - mx + y$$

So given the generic form of our standard perceptron $A + Bx + Cy$ we see here that our modified perceptron can only ever learn a representation where C = 1, limiting what we can achieve.