Gradient Descent Implementation note

Mean Square Error

We're going to make a small change to how we calculate the error here. Instead of the SSE, we're going to use the **mean** of the square errors (MSE). Now that we're using a lot of data, summing up all the weight steps can lead to really large updates that make the gradient descent diverge. To compensate for this, you'd need to use a quite small learning rate. Instead, we can just divide by the number of records in our data, m to take the average.

This way, no matter how much data we use, our learning rates will typically be in the range of 0.01 to 0.001. Then, we can use the MSE (shown below) to calculate the gradient and the result is the same as before, just averaged instead of summed.

$$E = \frac{1}{2m} \sum_{\mu} (y^{\mu} - \hat{y}^{\mu})^2$$

Development of the derivative of the error function

Notice that we've defined the squared error to be

$Error = \frac{1}{2} (y - \hat{y})^2$.

Also, we've defined the prediction to be

$\hat{y} = w_1 x + w_2$.

So to calculate the derivative of the Error with respect to w_1 , we simply use the chain rule:

$$\frac{\partial Error}{\partial w1} = \frac{\partial Error}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w1}$$

The first factor of the right hand side is the derivative of the Error with respect to the prediction $\hat{y}$, which is $-(y-\hat{y})$.

The second factor is the derivative of the prediction with respect to w_1, which is simply x. Therefore, the derivative is

$$\frac{\partial}{\partial w_1} Error = -(y - \hat{y})x$$

<u>Here's the general algorithm for updating the weights with gradient descent:</u>

- Set the weight step to zero: $\Delta w_i = 0$
- For each record in the training data:
  - Make a forward pass through the network, calculating the output $\hat{y} = f(\sum_i w_i x_i)$
  - Calculate the error term for the output unit, $\delta = (y - \hat{y}) * f'(\sum_i w_i x_i)$
  - Update the weight step $\Delta w_i = \Delta w_i + \delta x_i$
- Update the weights $w_i = w_i + \eta \Delta w_i / m$ where η is the learning rate and m is the number of records. Here we're averaging the weight steps to help reduce any large variations in the training data.
- Repeat for e epochs.

You can also update the weights on each record instead of averaging the weight steps after going through all the records. Remember that we're using the sigmoid for the activation function, $f(h)=1/(1+e^{-h})$. And the gradient of the sigmoid is $f'(h)=f(h)(1-f(h))$ where h is the input to the output unit, $h=\sum_i w_i x_i$