## Gradient Descent: The Math

We know that the output of a simple neural network is as follows:

$\hat{y} = f(h)$, where $f(h)$ the activation function

Where, $h = \sum_i w_i x_i$

$$\Rightarrow \hat{y} = f(\sum_i w_i x_i)$$

We will use Error function to measure how bad our predictions $\hat{y}$ are.

The obvious choice is to choose $E = (y - \hat{y})$

But we can make some enhancements to our $E$ like follows:

$$E = (y - \hat{y})^2$$

Where $\boldsymbol{E}$ **is the error for one prediction**.

We square the error for multiple reasons:

1- Using of square will penalizes outliers more than small errors
2- Squaring the error make the math of this equation nice and easy

## The error for the entire dataset:

The sum of the squared errors **(SSE)**:

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

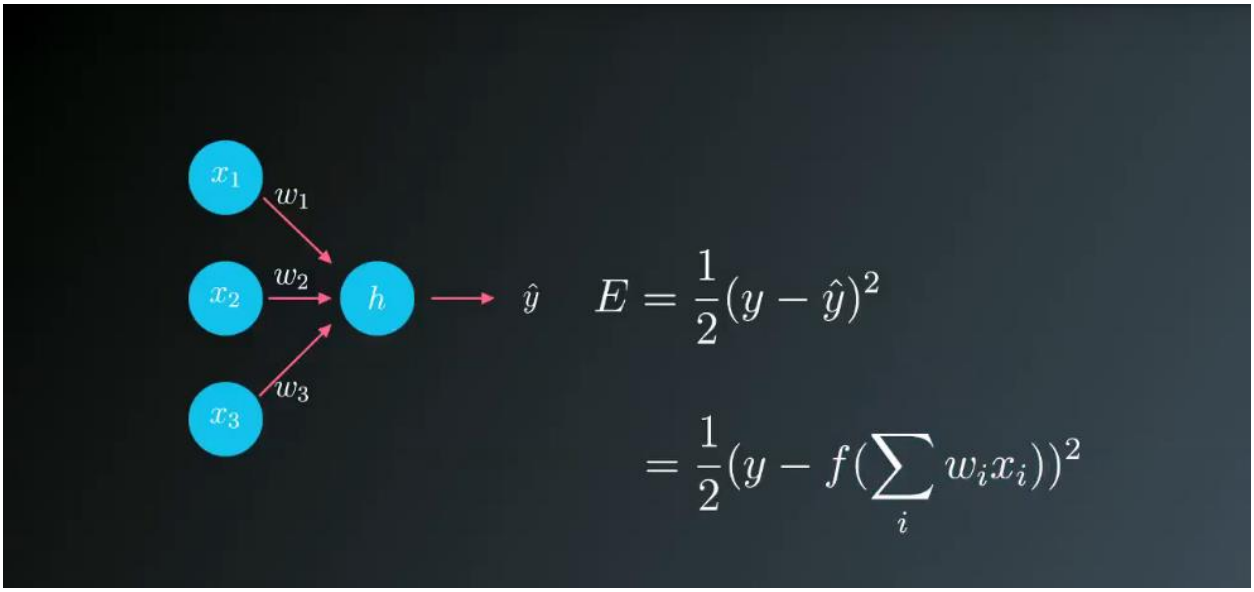
But remember that $\hat{y} = f(\sum_i w_i x_i)$

$$\Rightarrow E = \frac{1}{2}\sum_\mu (y^\mu - f(\sum_i w_i x_i^\mu))^2$$

$\Rightarrow E$ depends on the weights values $w_i$ and inputs values $x_i^\mu$

**Note: (SSE) is a measure of network's performance if it high then the network is making a bad predictions, and it low then the network making good predictions.**

Let take an example with only one data record and only one output uint:

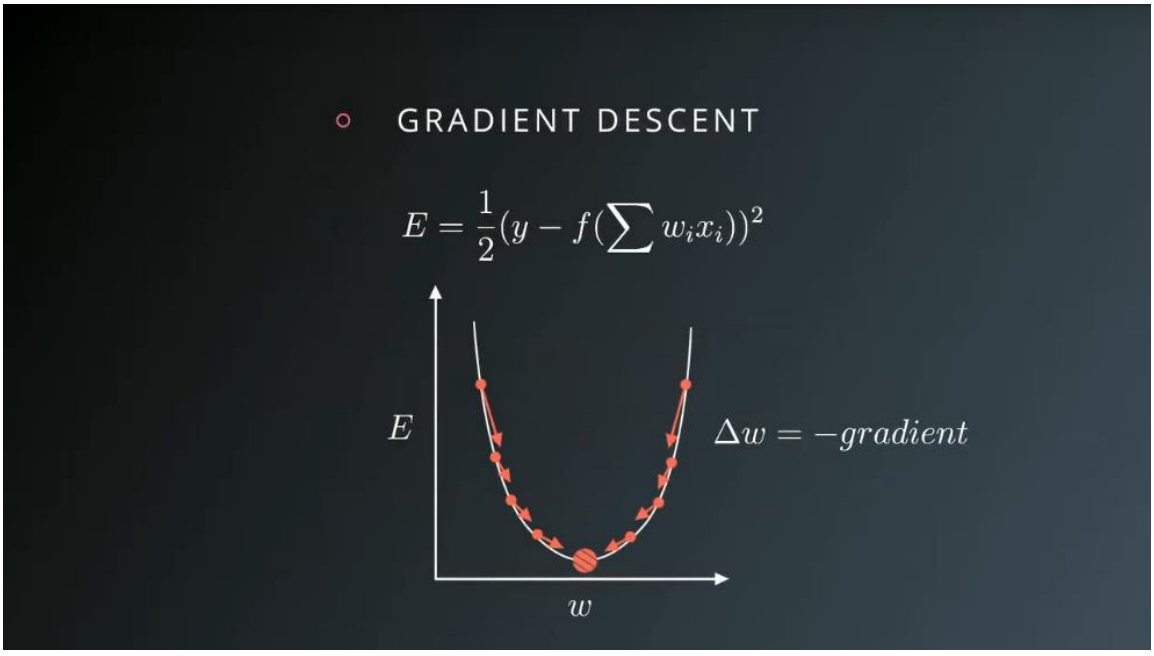



We can see that,

$E = \frac{1}{2}\,(y^{\mu} - f(\sum_i w_i\, x_i))^2$ is the error for one data record and one output unit

From the above equation we can see clearly that the error $E$ is a function of the weights, The weights are the knobs we can use to alter the network's predictions, which in turn effect the overall error, then **our goal to find weights that minimize error**

Here in the image bellow is a simple depiction of the error with one weight



Our goal is to find the weight at the bottom of this bowl, by starting at random weights, we want to make step in the direction towards the minimum.

This direction is the opposite to the gradient, the slope

$$gradient\ descent:$$

$$\Delta w = -gradient$$

If we take many steps, always descending down the gradient, eventually the weight will find the minimum of the error function.

## Updating the weight

The we want to update the weight:

$$w_i = w_i + \Delta w_i$$

**Note:** $\Delta w_i \propto -\frac{\partial E}{\partial w_i}$

The weight step is proportional to the gradient, so we can add some scaling parameter $\eta$ where $\eta$ is the learning rate which allow us to set the size of gradient descent steps.

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

**Calculating the gradient descent:**

So, $\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2}(y - \hat{y})^2$

Recall that the output $\hat{y}$ is a function of the weights ⇒

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2}(y - \hat{y})^2$$

$$= \frac{\partial}{\partial w_i} \frac{1}{2}(y - \hat{y}(w_i))^2$$

Using chain rule: $\frac{\partial}{\partial z} p\big(q(z)\big) = \frac{\partial p}{\partial q} \frac{\partial q}{\partial z}$

We can set $q$ to → $q = (y - \hat{y}(w_i))$ and $p$ to $p = \frac{1}{2} q(w_i)^2$

$$\Rightarrow \frac{\partial E}{\partial w_i} = -(y - \hat{y}) \frac{\partial \hat{y}}{\partial w_i}$$

Where $\hat{y} = f(h)$

$$\Rightarrow \frac{\partial E}{\partial w_i} = -(y - \hat{y}) f'(h) \frac{\partial}{\partial w_i} \sum_i w_i x_i$$

$$\Rightarrow \frac{\partial E}{\partial w_i} = -(y - \hat{y}) f'(h) x_i$$

Recall that $\Delta w = -gradient$

And $w_i = w_i + \Delta w_i$

And $\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$

$$\Rightarrow \Delta w_i = \eta (y - \hat{y}) f'(h) x_i$$

For convenient we can define an **error term**:

$$\delta = (y - \hat{y}) f'(h)$$

$$w_i = w_i + \eta \delta x_i$$

**if multiple output units:**

$$\delta_j = \left(y_j - \hat{y}_j\right) f'(h_j)$$

$$w_{ij} = \eta \delta_j x_i$$

Check out Khan Academy's Multivariable calculus lessons if you are unfamiliar with the subject.