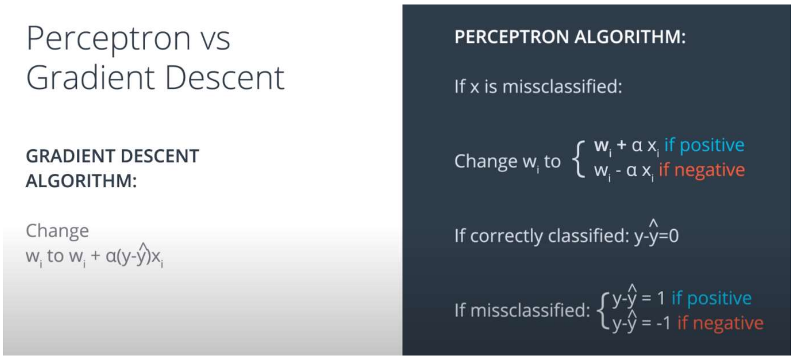
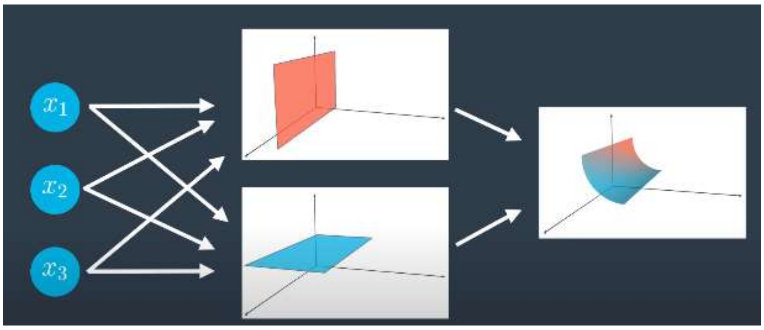
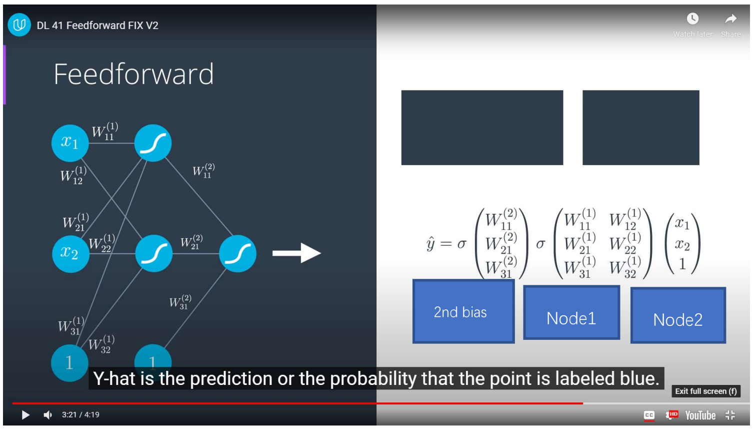
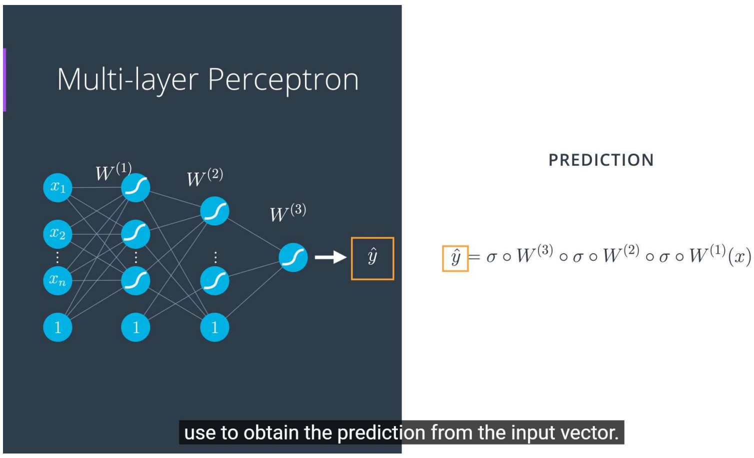


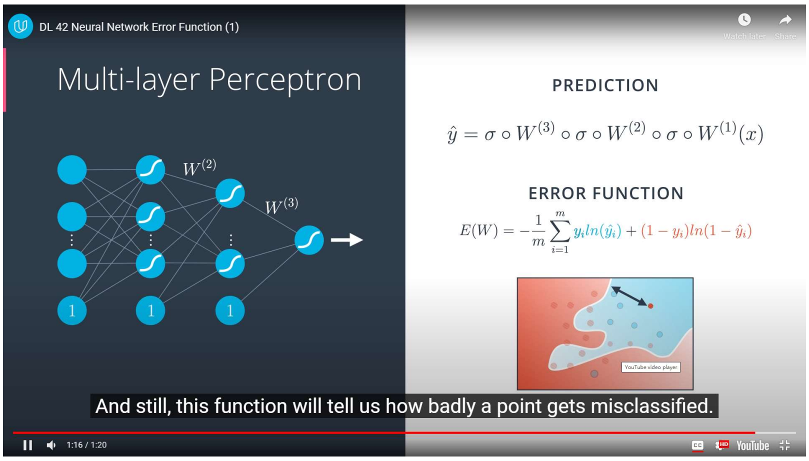
Multi layer

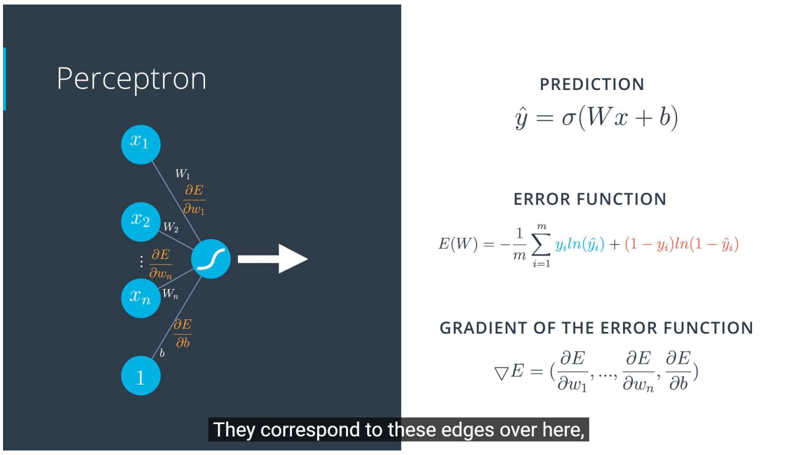


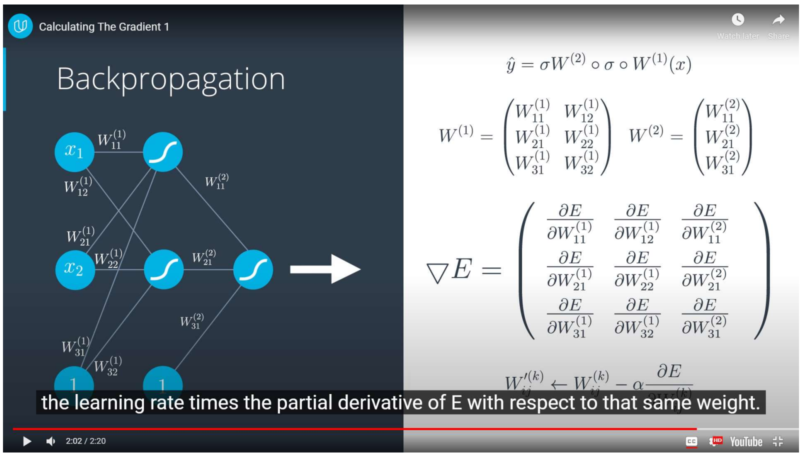
Feedforward

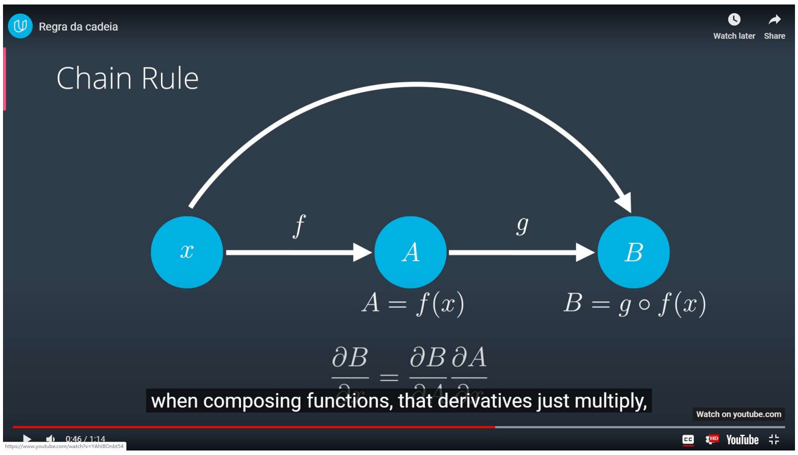


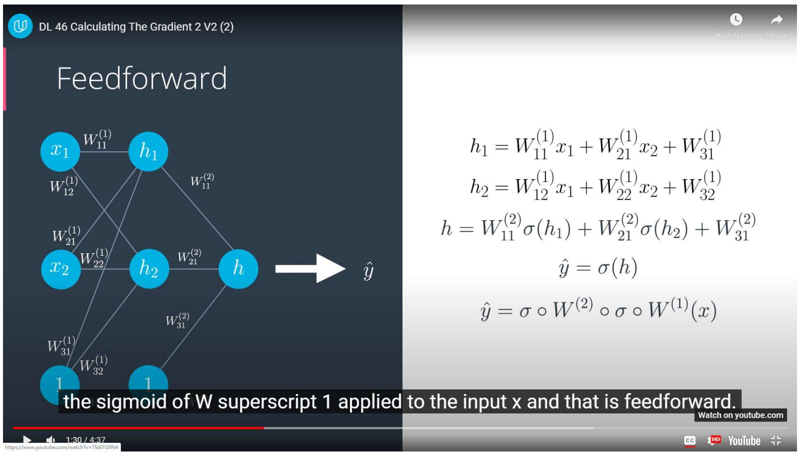


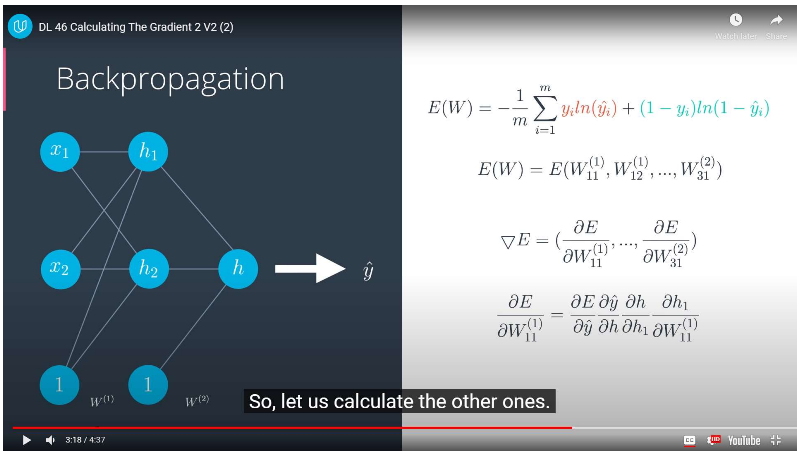

DL 42 Neural Network Error Function (1)
Multi-layer Perceptron
$W^{(2)}$
$W^{(3)}$
PREDICTION
$\hat{y} = \sigma \circ W^{(3)} \circ \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$
ERROR FUNCTION
$E(W) = -\frac{1}{m}\sum_{i=1}^{m} y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)$
And still, this function will tell us how badly a point gets misclassified.
1:16 / 1:20

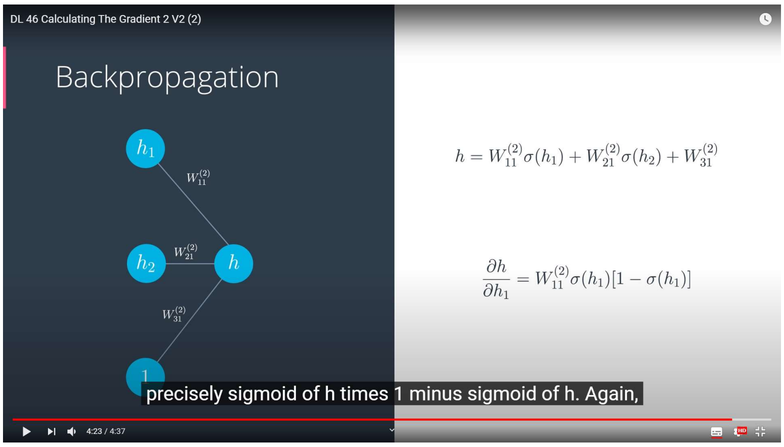

Perceptron
x_1
x_2
x_n
W_1
W_2
W_n
$\frac{\partial E}{\partial w_1}$
$\frac{\partial E}{\partial w_n}$
$\frac{\partial E}{\partial b}$
b
PREDICTION
$\hat{y} = \sigma(Wx + b)$
ERROR FUNCTION
$E(W) = -\frac{1}{m}\sum_{i=1}^{m} y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)$
GRADIENT OF THE ERROR FUNCTION
$\nabla E = (\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}, \frac{\partial E}{\partial b})$
They correspond to these edges over here,


Calculating The Gradient 1
Backpropagation
$\hat{y} = \sigma W^{(2)} \circ \sigma \circ W^{(1)}(x)$
$W^{(1)} = \begin{pmatrix} W_{11}^{(1)} & W_{12}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} \end{pmatrix}$ $W^{(2)} = \begin{pmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{pmatrix}$
$\nabla E = \begin{pmatrix} \frac{\partial E}{\partial W_{11}^{(1)}} & \frac{\partial E}{\partial W_{12}^{(1)}} & \frac{\partial E}{\partial W_{11}^{(2)}} \\ \frac{\partial E}{\partial W_{21}^{(1)}} & \frac{\partial E}{\partial W_{22}^{(1)}} & \frac{\partial E}{\partial W_{21}^{(2)}} \\ \frac{\partial E}{\partial W_{31}^{(1)}} & \frac{\partial E}{\partial W_{32}^{(1)}} & \frac{\partial E}{\partial W_{31}^{(2)}} \end{pmatrix}$
the learning rate times the partial derivative of E with respect to that same weight.
2:02 / 2:20


Regra da cadeia
Watch later
Share
Chain Rule
$x \xrightarrow{f} A \xrightarrow{g} B$
$A = f(x)$
$B = g \circ f(x)$
when composing functions, that derivatives just multiply,
Watch on youtube.com
0:46 / 1:14
YouTube


DL 46 Calculating The Gradient 2 V2 (2)
Feedforward
$h_1 = W_{11}^{(1)}x_1 + W_{21}^{(1)}x_2 + W_{31}^{(1)}$
$h_2 = W_{12}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{32}^{(1)}$
$h = W_{11}^{(2)}\sigma(h_1) + W_{21}^{(2)}\sigma(h_2) + W_{31}^{(2)}$
$\hat{y} = \sigma(h)$
$\hat{y} = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$
the sigmoid of W superscript 1 applied to the input x and that is feedforward.
Watch on youtube.com
1:30 / 4:37
YouTube


DL 46 Calculating The Gradient 2 V2 (2)
Backpropagation
$E(W) = -\frac{1}{m}\sum_{i=1}^{m} y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)$
$E(W) = E(W_{11}^{(1)}, W_{12}^{(1)}, ..., W_{31}^{(2)})$
$\bigtriangledown E = (\frac{\partial E}{\partial W_{11}^{(1)}}, ..., \frac{\partial E}{\partial W_{31}^{(2)}})$
$\frac{\partial E}{\partial W_{11}^{(1)}} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h} \frac{\partial h}{\partial h_1} \frac{\partial h_1}{\partial W_{11}^{(1)}}$
So, let us calculate the other ones.
3:18 / 4:37
YouTube

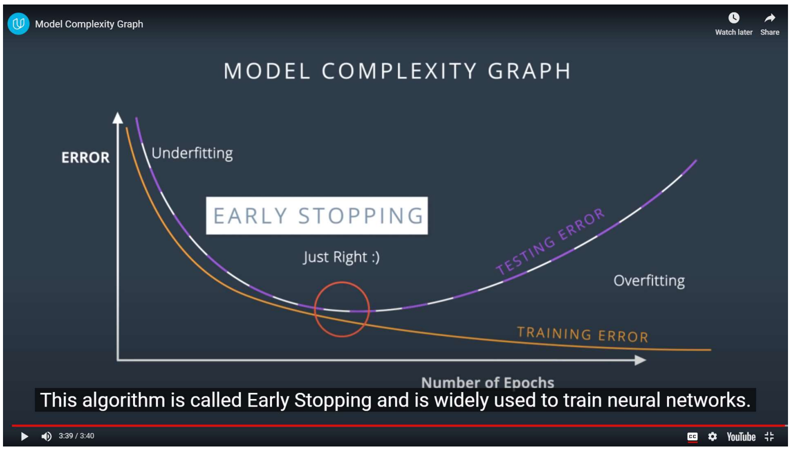


Calculation of the derivative of the sigmoid function

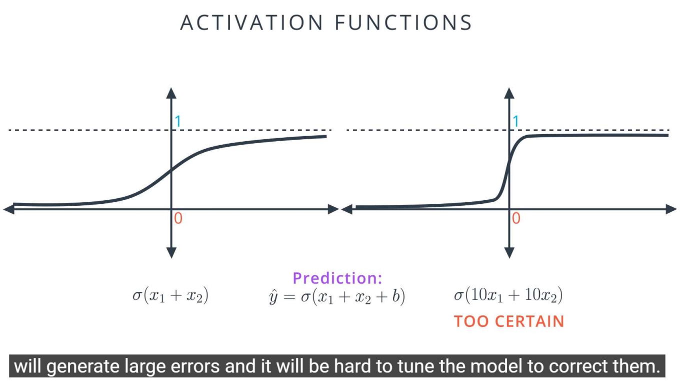
Recall that the sigmoid function has a beautiful derivative, which we can see in the following calculation. This will make our backpropagation step much cleaner.

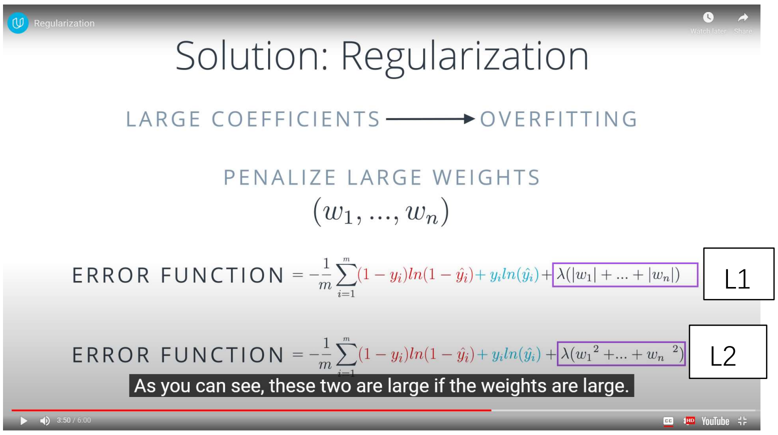
$$
\begin{aligned}
\sigma'(x) &= \frac{\partial}{\partial x}\frac{1}{1+e^{-x}} \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}
$$

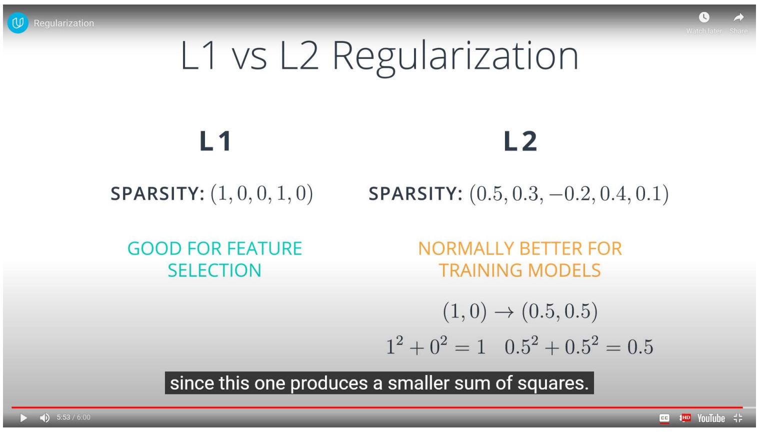
Early Stopping



Regularization：





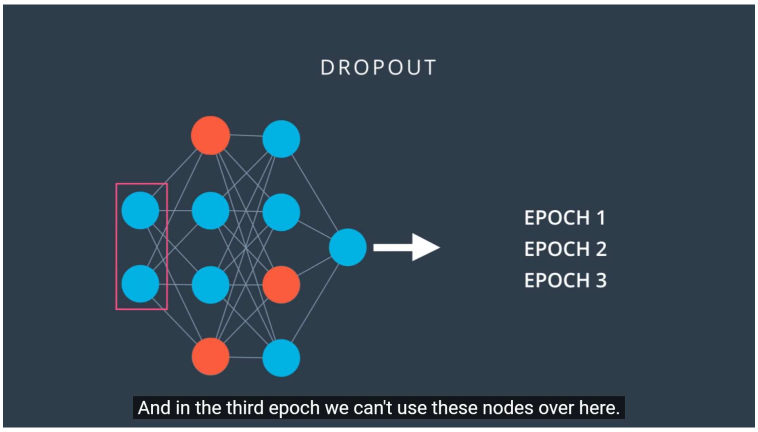


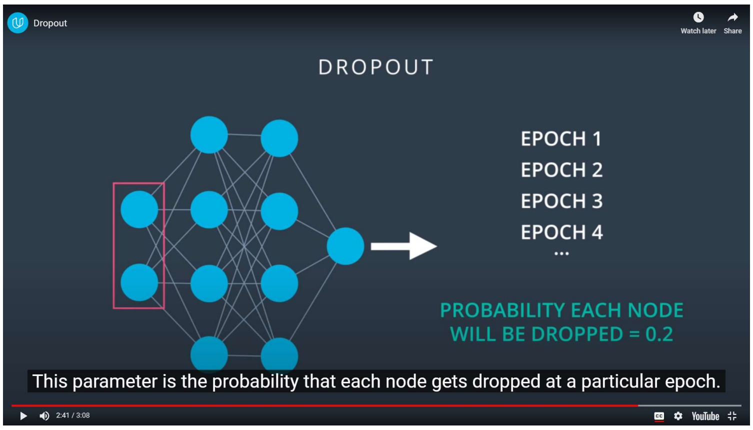
L1 tends to end up with sparse vectors,That means, a small weights will tend to go to zero. So if we want to reduce the number of weights and end up with a small set, we can use L1 This is also good for feature selections and sometimes we have a problem with hundreds of features, and L1 regularization will help us select which ones are important and it will turn the rest into zeros

L2 tries to maintain all the weights homogeneously small, This one normally gives better results for training models. So why would L1 regularization produce vectors with sparse weights and L2 regularization will produce vectors with small homogeneous weights?

L2 regularization will prefer the vector point(0.5,0.5) over the vector(1,0) since this one produces a smaller sum of squares.

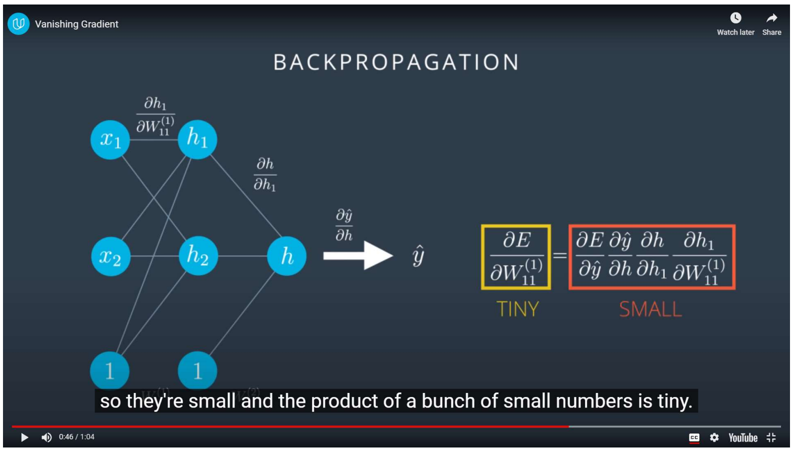
Dropout



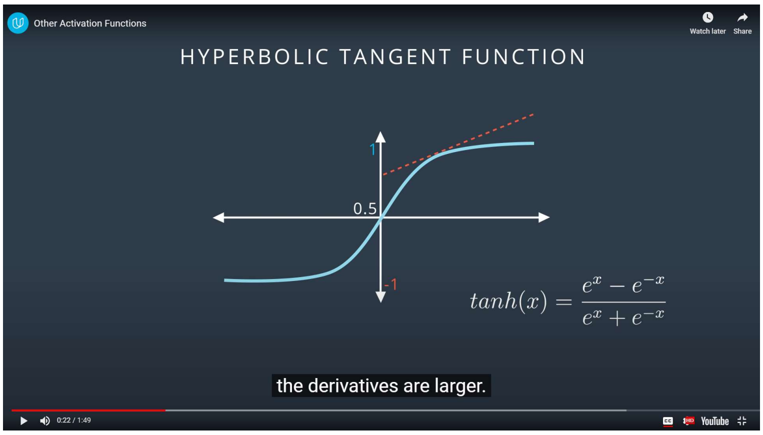


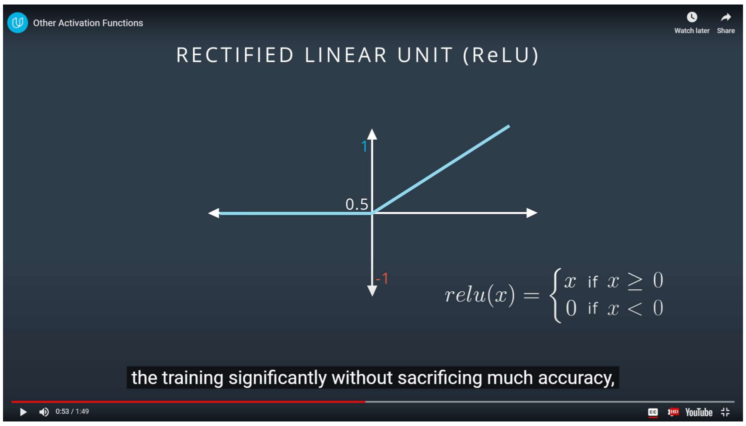
Vanishing Gradient

Sigmoid has very small derivatives when it closes to 1 or -1. This makes weight update very difficult.

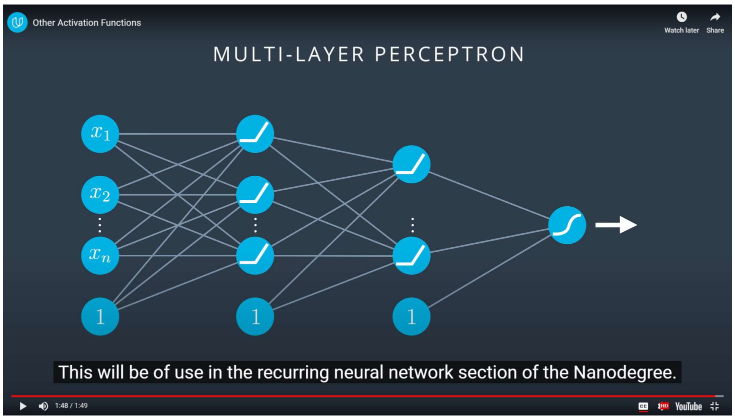


Other activation function

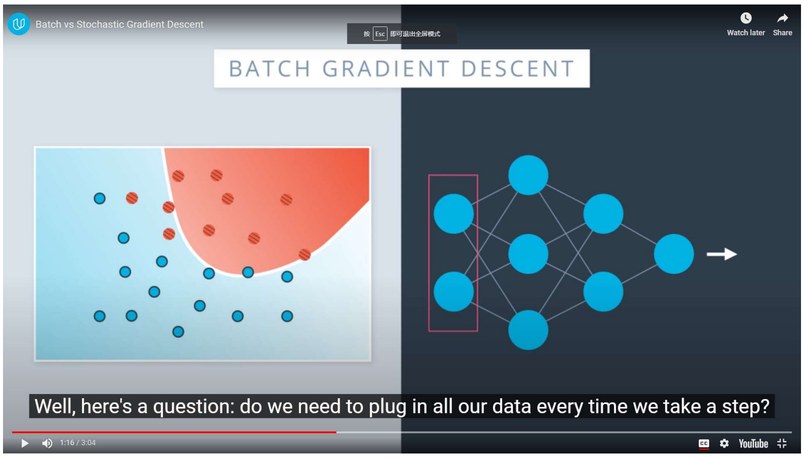




The relu can impove the training significantly without sacrificing much accuracy.

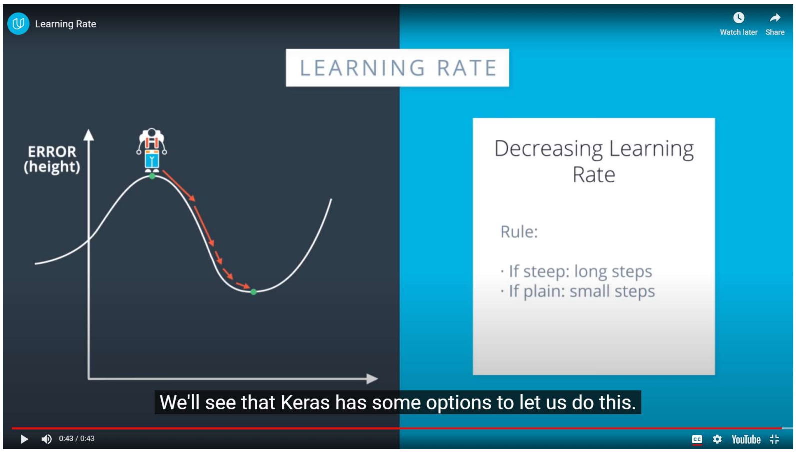


Stochastic Gradient

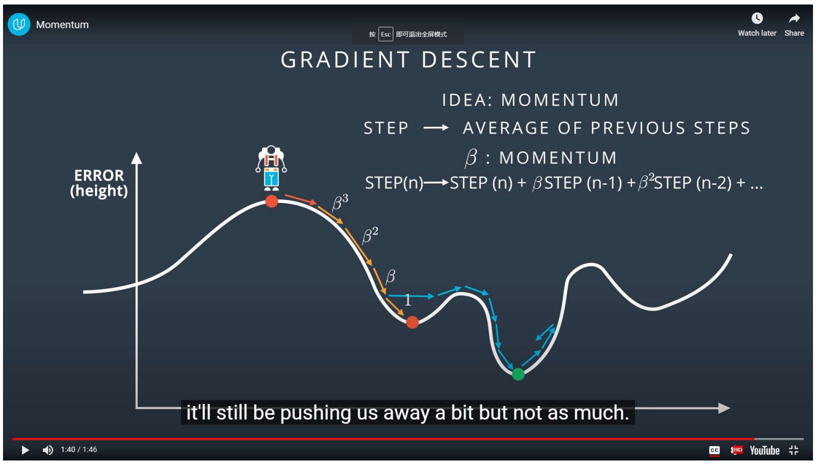


The idea behind stochastic gradient descent is simply that we take small subsets of data,run them through the nn, calculate the gradient of error function based on those points and move one step in that direction. Split the data into several batches.

Learning rate



To solve the local minima to have a random restart or momentum



The previous step matters a lot and the steps before that matter less and less

Neural Network For regression

This is classification: