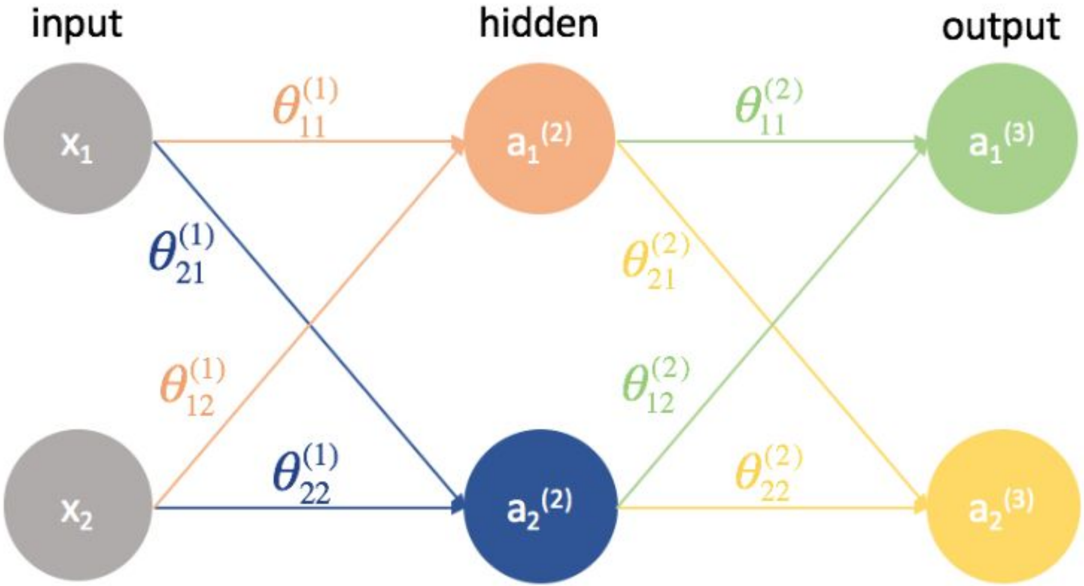
This is CS4045!

GCR : dxuxugo

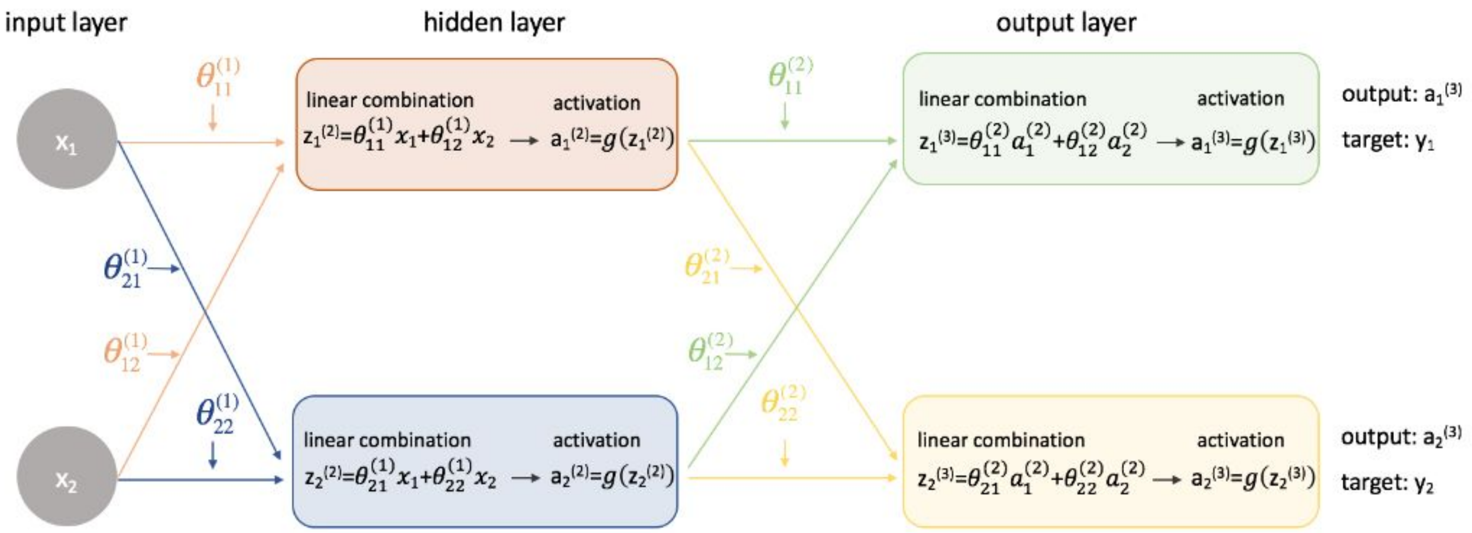
P.S. These slides are useless if you do not attend classes

Neural Networks

Neural network with multi Output



Neural network with multi Output



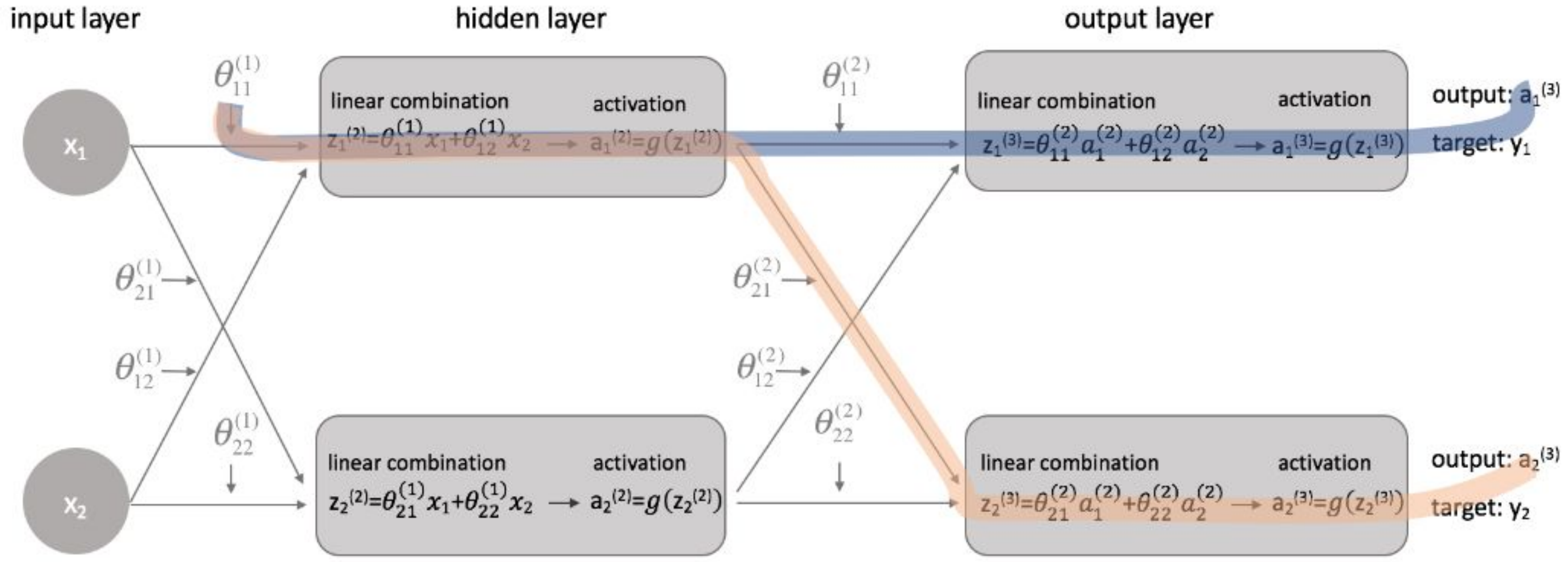
Neural network with multi Output

Loss calculation

m=number of samples

$$J(\theta) = \frac{1}{2m} \sum \left(y_i - \mathrm{a}_i^{(2)} \right)^2$$

Neural network with multi Output



The derivative chain for the blue path is:

$$\left(\frac{\partial J(\theta)}{\partial \mathrm{a}_1^{(3)}}\right)\left(\frac{\partial \mathrm{a}_1^{(3)}}{\partial z_1^{(3)}}\right)\left(\frac{\partial z_1^{(3)}}{\partial \mathrm{a}_1^{(2)}}\right)\left(\frac{\partial \mathrm{a}_1^{(2)}}{\partial z_1^{(2)}}\right)\left(\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}}\right)$$

The derivative chain for the orange path is:

$$\left(\frac{\partial J(\theta)}{\partial \mathrm{a}_2^{(3)}}\right)\left(\frac{\partial \mathrm{a}_2^{(3)}}{\partial z_2^{(3)}}\right)\left(\frac{\partial z_2^{(3)}}{\partial \mathrm{a}_1^{(2)}}\right)\left(\frac{\partial \mathrm{a}_1^{(2)}}{\partial z_1^{(2)}}\right)\left(\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}}\right)$$

Combining these, we get the total expression for $\frac{\partial J(\theta)}{\partial \theta_{11}^{(1)}}$.

$$\frac{\partial J(\theta)}{\partial \theta_{11}^{(1)}}=\left(\frac{\partial J(\theta)}{\partial \mathrm{a}_1^{(3)}}\right)\left(\frac{\partial \mathrm{a}_1^{(3)}}{\partial z_1^{(3)}}\right)\left(\frac{\partial z_1^{(3)}}{\partial \mathrm{a}_1^{(2)}}\right)\left(\frac{\partial \mathrm{a}_1^{(2)}}{\partial z_1^{(2)}}\right)\left(\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}}\right)+\left(\frac{\partial J(\theta)}{\partial \mathrm{a}_2^{(3)}}\right)\left(\frac{\partial \mathrm{a}_2^{(3)}}{\partial z_2^{(3)}}\right)\left(\frac{\partial z_2^{(3)}}{\partial \mathrm{a}_1^{(2)}}\right)\left(\frac{\partial \mathrm{a}_1^{(2)}}{\partial z_1^{(2)}}\right)\left(\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}}\right)$$

Layer 2 Parameters

$$\frac{\partial J(\theta)}{\partial \theta_{11}^{(2)}} = \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_1^{(3)}}\right)\left(\frac{\partial \mathrm{a}_1^{(3)}}{\partial z_1^{(3)}}\right)\left(\frac{\partial z_1^{(3)}}{\partial \theta_{11}^{(2)}}\right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{12}^{(2)}} = \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_1^{(3)}}\right)\left(\frac{\partial \mathrm{a}_1^{(3)}}{\partial z_1^{(3)}}\right)\left(\frac{\partial z_1^{(3)}}{\partial \theta_{12}^{(2)}}\right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{21}^{(2)}} = \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_2^{(3)}}\right)\left(\frac{\partial \mathrm{a}_2^{(3)}}{\partial z_2^{(3)}}\right)\left(\frac{\partial z_2^{(3)}}{\partial \theta_{21}^{(2)}}\right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{22}^{(2)}} = \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_2^{(3)}}\right)\left(\frac{\partial \mathrm{a}_2^{(3)}}{\partial z_2^{(3)}}\right)\left(\frac{\partial z_2^{(3)}}{\partial \theta_{22}^{(2)}}\right)$$

Layer 1 Parameters

$$\frac{\partial J(\theta)}{\partial \theta_{11}^{(1)}} = \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_1^{(3)}}\right)\left(\frac{\partial \mathrm{a}_1^{(3)}}{\partial z_1^{(3)}}\right)\left(\frac{\partial z_1^{(3)}}{\partial \mathrm{a}_1^{(2)}}\right)\left(\frac{\partial \mathrm{a}_1^{(2)}}{\partial z_1^{(2)}}\right)\left(\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}}\right) + \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_2^{(3)}}\right)\left(\frac{\partial \mathrm{a}_2^{(3)}}{\partial z_2^{(3)}}\right)\left(\frac{\partial z_2^{(3)}}{\partial \mathrm{a}_1^{(2)}}\right)\left(\frac{\partial \mathrm{a}_1^{(2)}}{\partial z_1^{(2)}}\right)\left(\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}}\right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{12}^{(1)}} = \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_1^{(3)}}\right)\left(\frac{\partial \mathrm{a}_1^{(3)}}{\partial z_1^{(3)}}\right)\left(\frac{\partial z_1^{(3)}}{\partial \mathrm{a}_1^{(2)}}\right)\left(\frac{\partial \mathrm{a}_1^{(2)}}{\partial z_1^{(2)}}\right)\left(\frac{\partial z_1^{(2)}}{\partial \theta_{12}^{(1)}}\right) + \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_2^{(3)}}\right)\left(\frac{\partial \mathrm{a}_2^{(3)}}{\partial z_2^{(3)}}\right)\left(\frac{\partial z_2^{(3)}}{\partial \mathrm{a}_1^{(2)}}\right)\left(\frac{\partial \mathrm{a}_1^{(2)}}{\partial z_1^{(2)}}\right)\left(\frac{\partial z_1^{(2)}}{\partial \theta_{12}^{(1)}}\right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{21}^{(1)}} = \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_1^{(3)}}\right)\left(\frac{\partial \mathrm{a}_1^{(3)}}{\partial z_1^{(3)}}\right)\left(\frac{\partial z_1^{(3)}}{\partial \mathrm{a}_2^{(2)}}\right)\left(\frac{\partial \mathrm{a}_2^{(2)}}{\partial z_2^{(2)}}\right)\left(\frac{\partial z_2^{(2)}}{\partial \theta_{21}^{(1)}}\right) + \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_2^{(3)}}\right)\left(\frac{\partial \mathrm{a}_2^{(3)}}{\partial z_2^{(3)}}\right)\left(\frac{\partial z_2^{(3)}}{\partial \mathrm{a}_2^{(2)}}\right)\left(\frac{\partial \mathrm{a}_2^{(2)}}{\partial z_2^{(2)}}\right)\left(\frac{\partial z_2^{(2)}}{\partial \theta_{21}^{(1)}}\right)$$

$$\frac{\partial J(\theta)}{\partial \theta_{22}^{(1)}} = \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_1^{(3)}}\right)\left(\frac{\partial \mathrm{a}_1^{(3)}}{\partial z_1^{(3)}}\right)\left(\frac{\partial z_1^{(3)}}{\partial \mathrm{a}_2^{(2)}}\right)\left(\frac{\partial \mathrm{a}_2^{(2)}}{\partial z_2^{(2)}}\right)\left(\frac{\partial z_2^{(2)}}{\partial \theta_{22}^{(1)}}\right) + \left(\frac{\partial J(\theta)}{\partial \mathrm{a}_2^{(3)}}\right)\left(\frac{\partial \mathrm{a}_2^{(3)}}{\partial z_2^{(3)}}\right)\left(\frac{\partial z_2^{(3)}}{\partial \mathrm{a}_2^{(2)}}\right)\left(\frac{\partial \mathrm{a}_2^{(2)}}{\partial z_2^{(2)}}\right)\left(\frac{\partial z_2^{(2)}}{\partial \theta_{22}^{(1)}}\right)$$

Discussion on Assignment Network

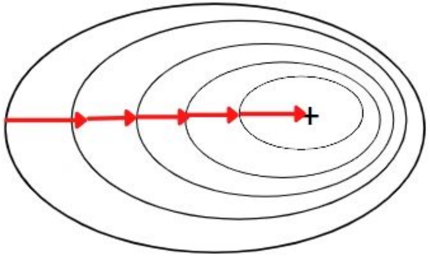
1. [20 Points] Draw 3-layer Neural Networks (Input layer is not counted here)with 2 nodes in each hidden layer and 1 output node. Input size is 3. Activation function is relu for 1st hidden layer and sigmoid for 2nd hidden and last layer. Write mathematical equations of all hidden and output layer nodes for a forward pass. Write down mathematical equations of the partial derivative of Loss w.r.t. all weights and bias for a backward pass. Simplify all these equations. Use (i) Squared Loss (ii) Cross Entropy Loss

Discussion on Assignment Network

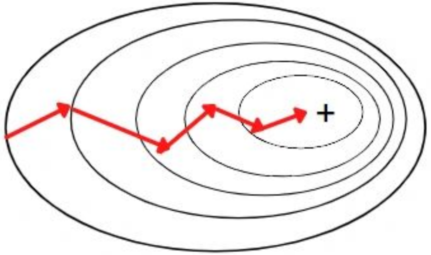
2. [10 Points] Draw 2-layer Neural Networks with 1 node in the hidden layer and 2 output nodes. Input size is 2. The activation function in the hidden layer is tanh while in the output layer, sigmoid is used. Write mathematical equations of all hidden and output layer nodes for a forward pass. Write down mathematical equations of the partial derivative of Loss w.r.t. all weights and bias for a backward pass. Simplify all these equations. Use Cross Entropy Loss.

Gradient Descent Variations - How to update parameters?

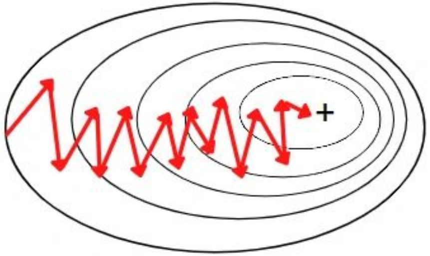
Batch Gradient Descent



Mini-Batch Gradient Descent

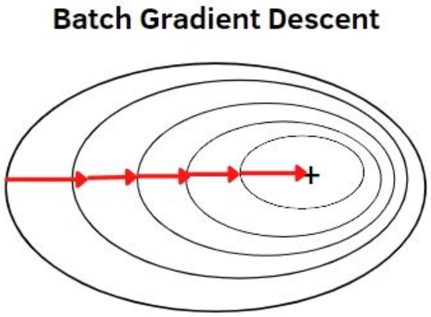


Stochastic Gradient Descent



Gradient Descent Variations - How to update parameters?

Batch gradient descent, also known as vanilla gradient descent
Calculates the error for each example within the training dataset.

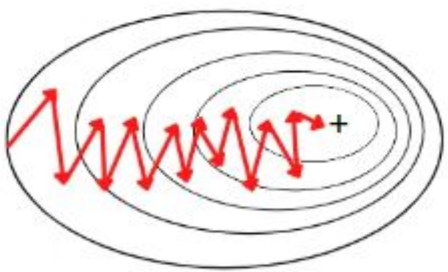


Still, the model is not changed until every training sample has been assessed. The entire procedure is referred to as a cycle and a training epoch.

Gradient Descent Variations - How to update parameters?

SGD changes the parameters for each training sample one at a time for each training example in the dataset.

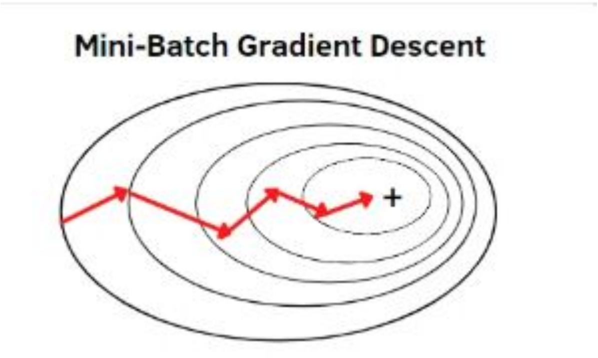
It makes SGD faster than batch gradient descent.



One benefit is that the regular updates give us a fairly accurate idea of the rate of improvement.
The frequency of such updates can also produce noisy gradients, which could cause the error rate to fluctuate rather than gradually go down.

Gradient Descent Variations - How to update parameters?

Mini-batch gradient descent combines ideas of batch gradient descent with SGD, it is the preferred technique.

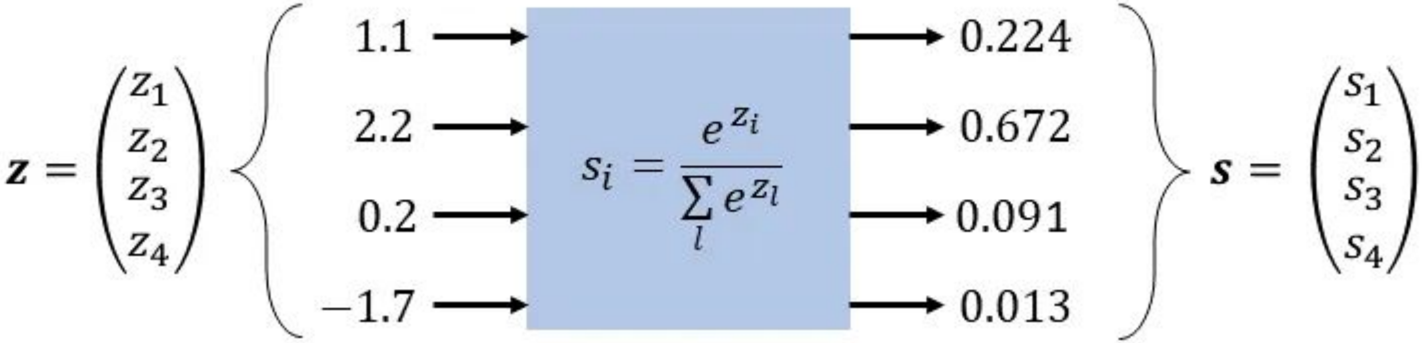


It divides the training dataset into manageable groups and updates each separately.

This strikes a balance between batch gradient descent’s effectiveness and stochastic gradient descent’s durability.

SoftMax Activation Function for MultiClass

$$\boxed{\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}}$$



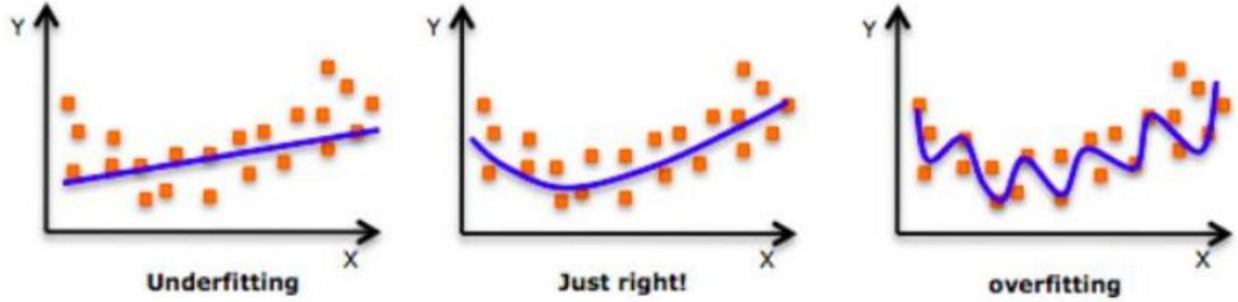
SoftMax Activation Function for MultiClass - Derivative

$$J_{softmax} = \begin{pmatrix} \frac{\partial s_1}{\partial z_1} & \frac{\partial s_1}{\partial z_2} & \cdots & \frac{\partial s_1}{\partial z_n} \\ \frac{\partial s_2}{\partial z_1} & \frac{\partial s_2}{\partial z_2} & \cdots & \frac{\partial s_2}{\partial z_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial s_n}{\partial z_1} & \frac{\partial s_n}{\partial z_2} & \cdots & \frac{\partial s_n}{\partial z_n} \end{pmatrix}$$

$$J_{softmax} = \begin{pmatrix} s_1 \cdot (1 - s_1) & -s_1 \cdot s_2 & -s_1 \cdot s_3 & -s_1 \cdot s_4 \\ -s_2 \cdot s_1 & s_2 \cdot (1 - s_2) & -s_2 \cdot s_3 & -s_2 \cdot s_4 \\ -s_3 \cdot s_1 & -s_3 \cdot s_2 & s_3 \cdot (1 - s_3) & -s_3 \cdot s_4 \\ -s_4 \cdot s_1 & -s_4 \cdot s_2 & -s_4 \cdot s_3 & s_4 \cdot (1 - s_4) \end{pmatrix}$$

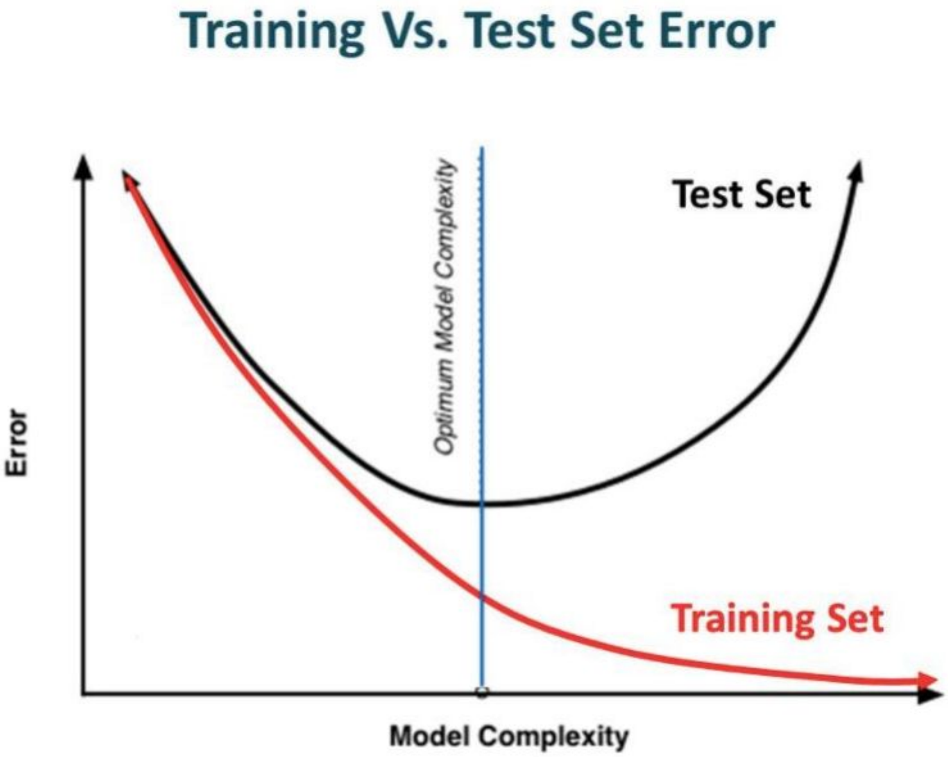
Regularization

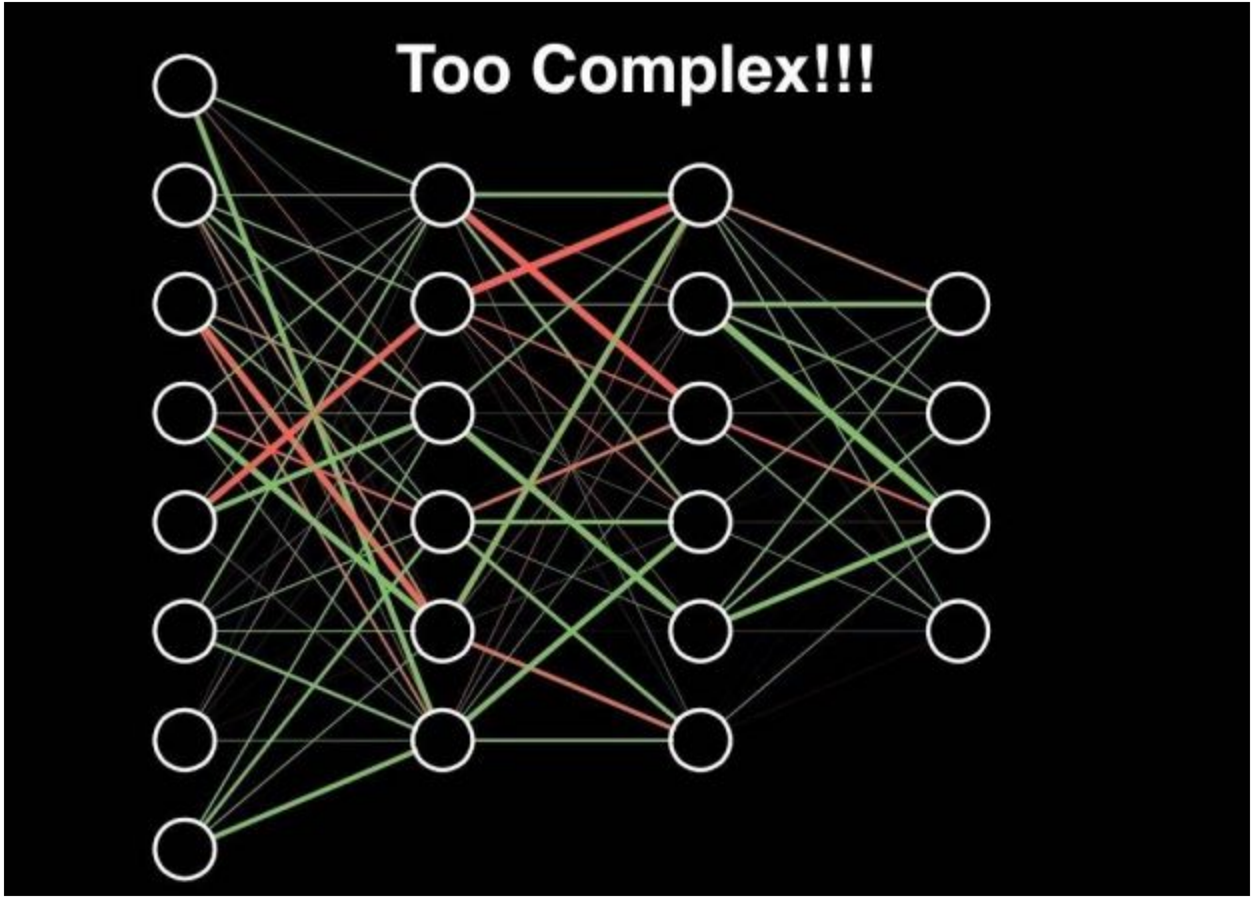
Regularization



As move towards right, poor performance on unseen data

Regularization



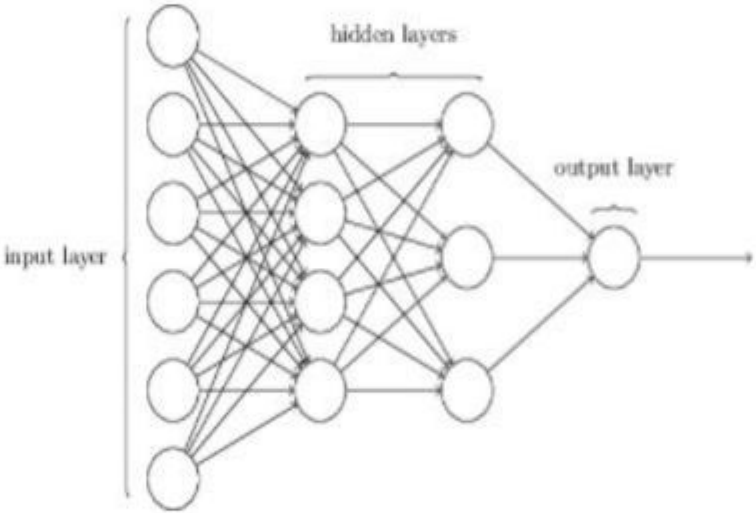

Too Complex!!!

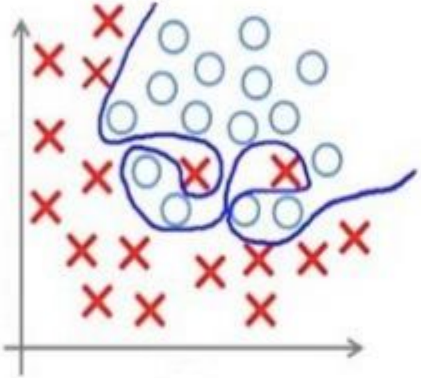
What is Regularization ?

Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better

This in turn improves the model's performance on the unseen data as well

Regularization





Over-fitting

Regularization

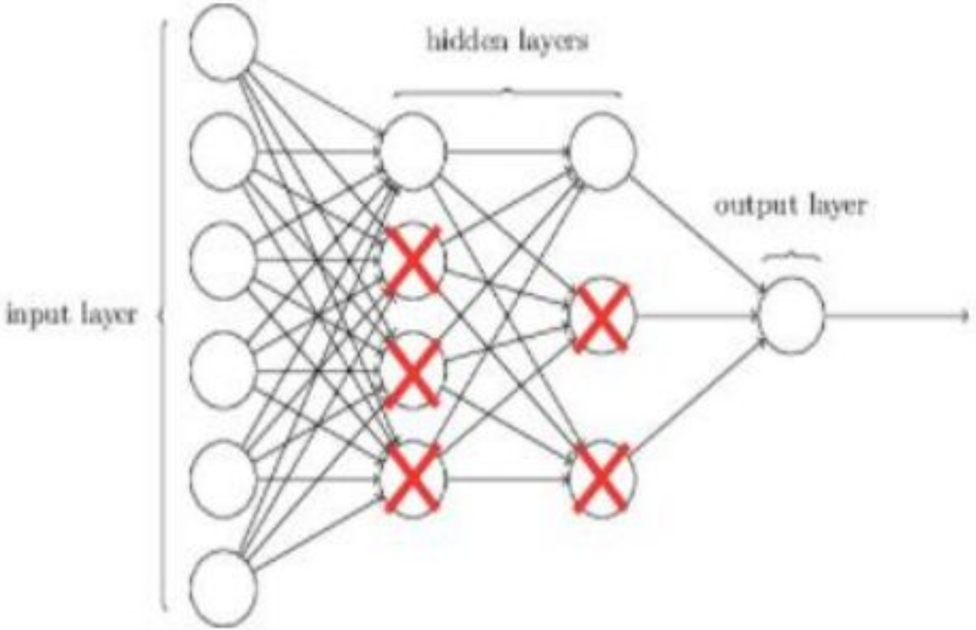
In machine learning, regularization penalizes the coefficients

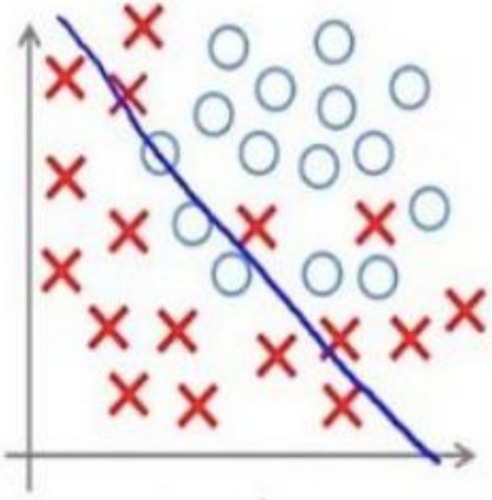
In deep learning, it actually penalizes the weight matrices of the nodes

Assume that our regularization coefficient is so high that some of the weight matrices are nearly equal to zero

This will result in a much simpler linear network and slight underfitting of the training data.

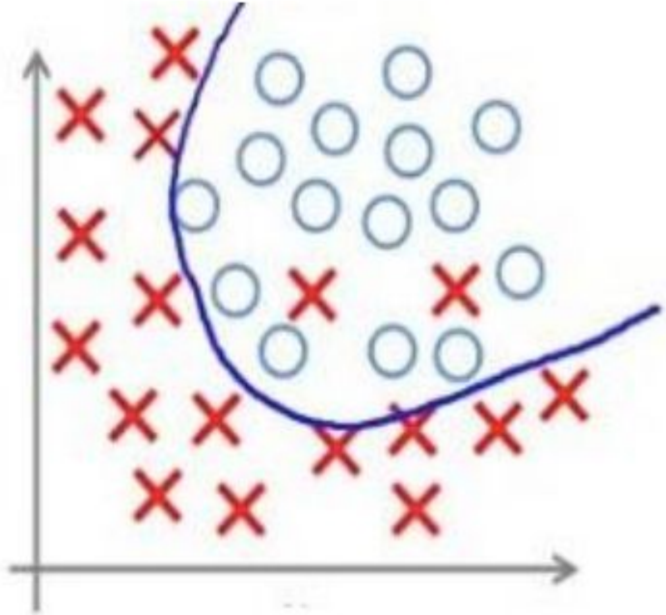
Regularization





Under-fitting

Regularization



Appropriate-fitting

Such a large value of the regularization coefficient is not that useful

We need to optimize the value of regularization coefficient in order to obtain a well-fitted model as shown in the image below

Regularization

- L1 and L2 regularization
- DropOut
- Data Augmentation
- Early Stopping

Reading Assignment

Read Neural Network L2 Regularization Using Python -- Visual Studio Magazine.pdf

Read Neural Network L1 Regularization Using Python -- Visual Studio Magazine.pdf

L1 L2 Regularization

L1 Regularization

$$\text{Modified loss function} = \text{Loss function} + \lambda \sum_{i=1}^{n} |W_i|$$

L2 Regularization

$$\text{Modified loss function} = \text{Loss function} + \lambda \sum_{i=1}^{n} W_i^{2}$$

L1

$$E = \underbrace{\frac{1}{2} * \sum (t_k - o_k)^2}_{\text{squared error}} + \underbrace{\lambda * \sum |w_i|}_{\text{L1 weight penalty}}$$

$$\Delta w_{jk} = -1 * \underbrace{\eta}_{\text{learning rate}} * \overbrace{[\, x_j * \underbrace{(o_k - t_k) * o_k * (1 - o_k)\,] \pm \lambda\,]}_{\text{signal}}}^{\frac{\partial E}{\partial w_{jk}} \text{ gradient}}$$

$$w_{jk} = w_{jk} + \Delta w_{jk}$$

L2

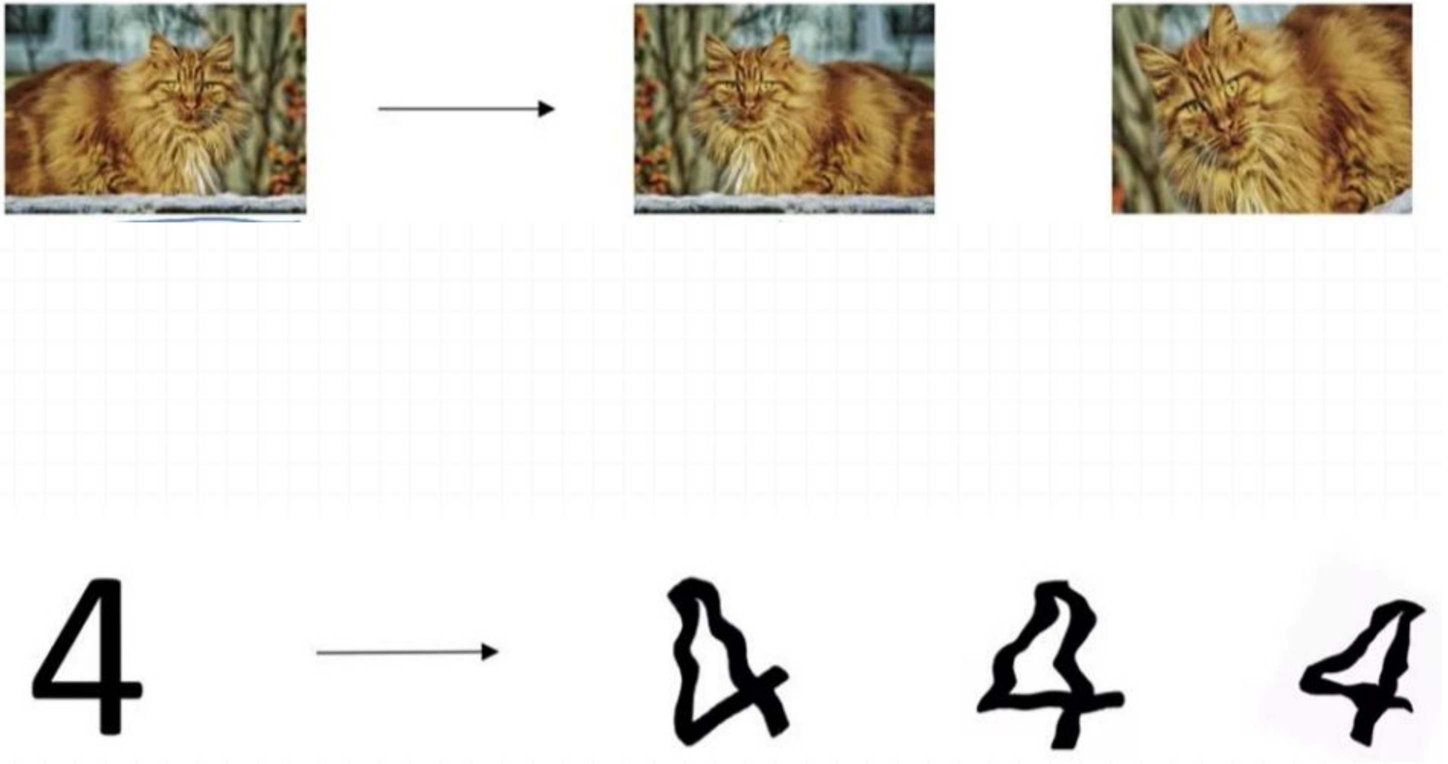
$$E = \underbrace{\frac{1}{2} * \sum (t_k - o_k)^2}_{\text{plain error}} + \underbrace{\frac{\lambda}{2} * \sum {w_i}^2}_{\text{weight penalty}}$$

elegant math

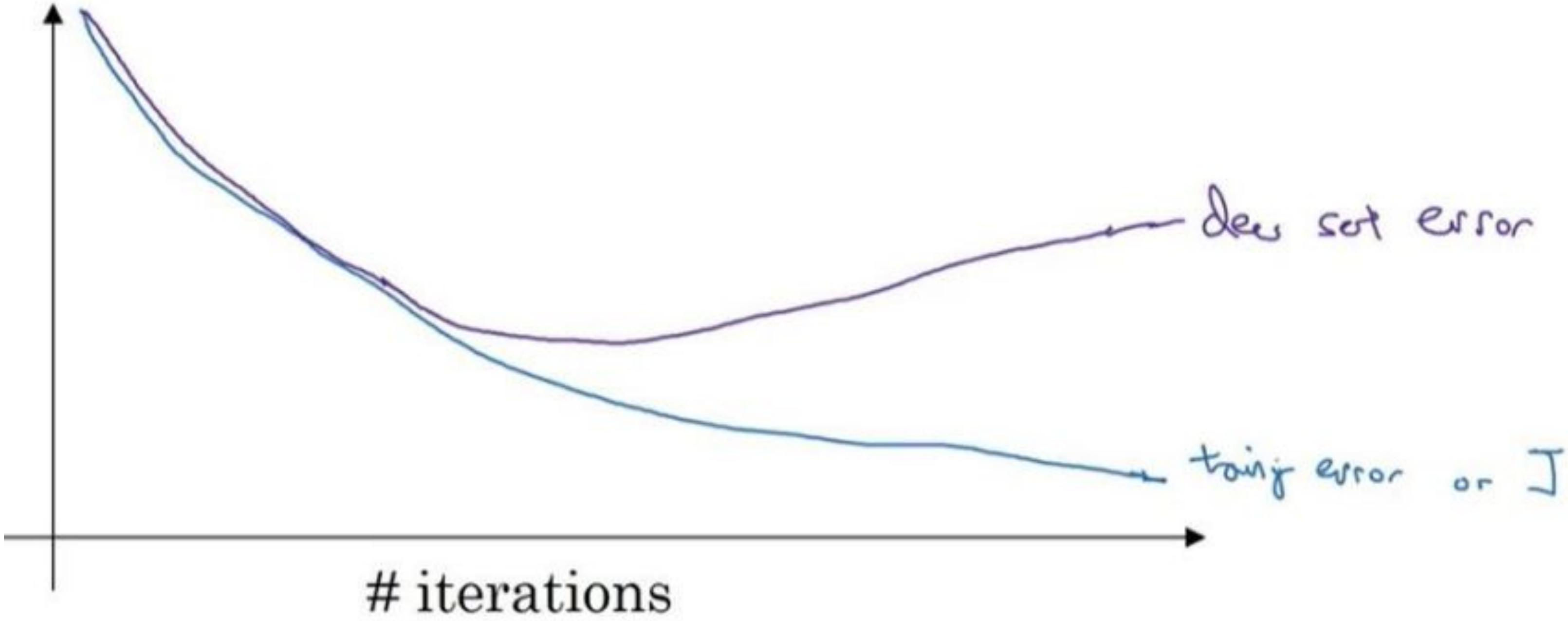
simple math

$$\Delta w_{jk} = \underset{\text{learning rate}}{\eta} * \overbrace{\left[x_j * \underbrace{(o_k - t_k) * o_k * (1 - o_k) \right] + [\lambda * w_{jk}]}_{\text{signal}}}^{\frac{\partial E}{\partial w_{jk}} \ \text{gradient}}$$

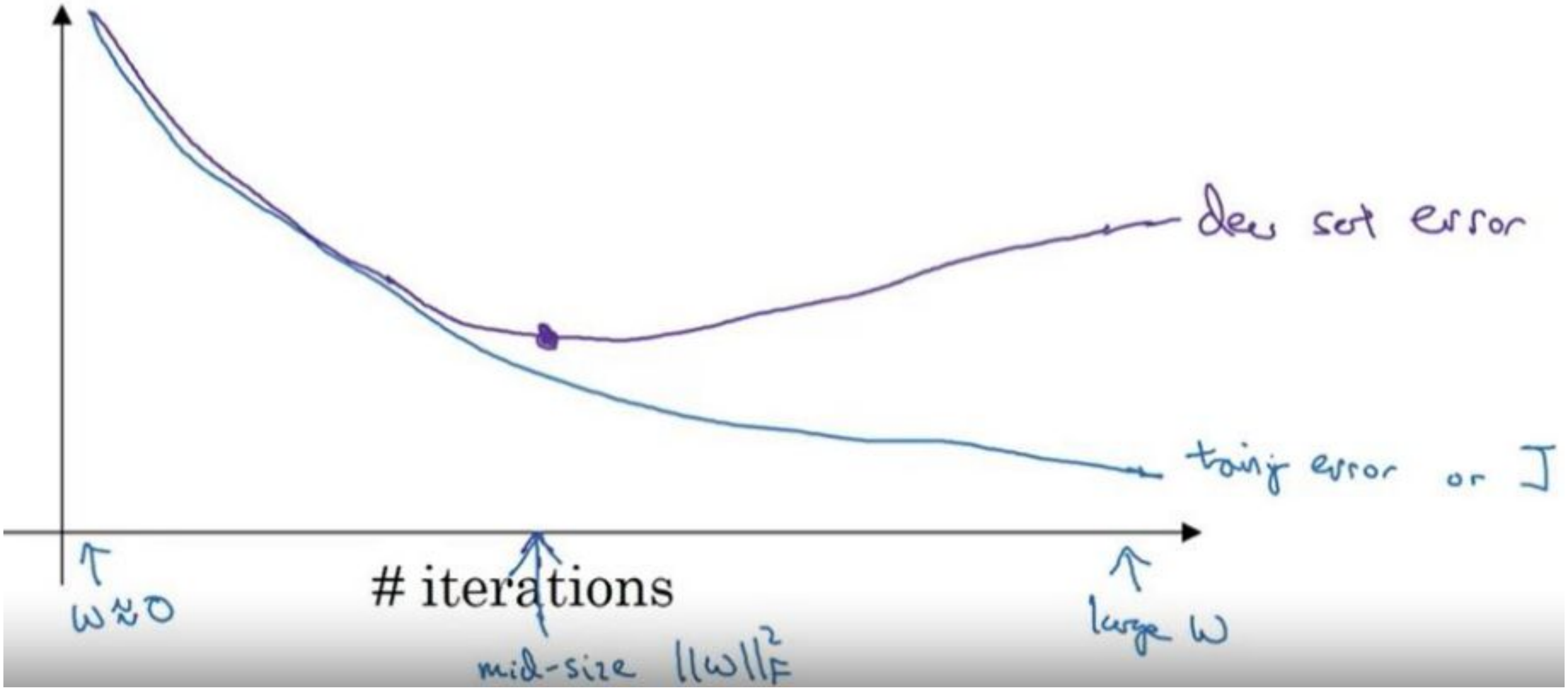
Data Augmentation



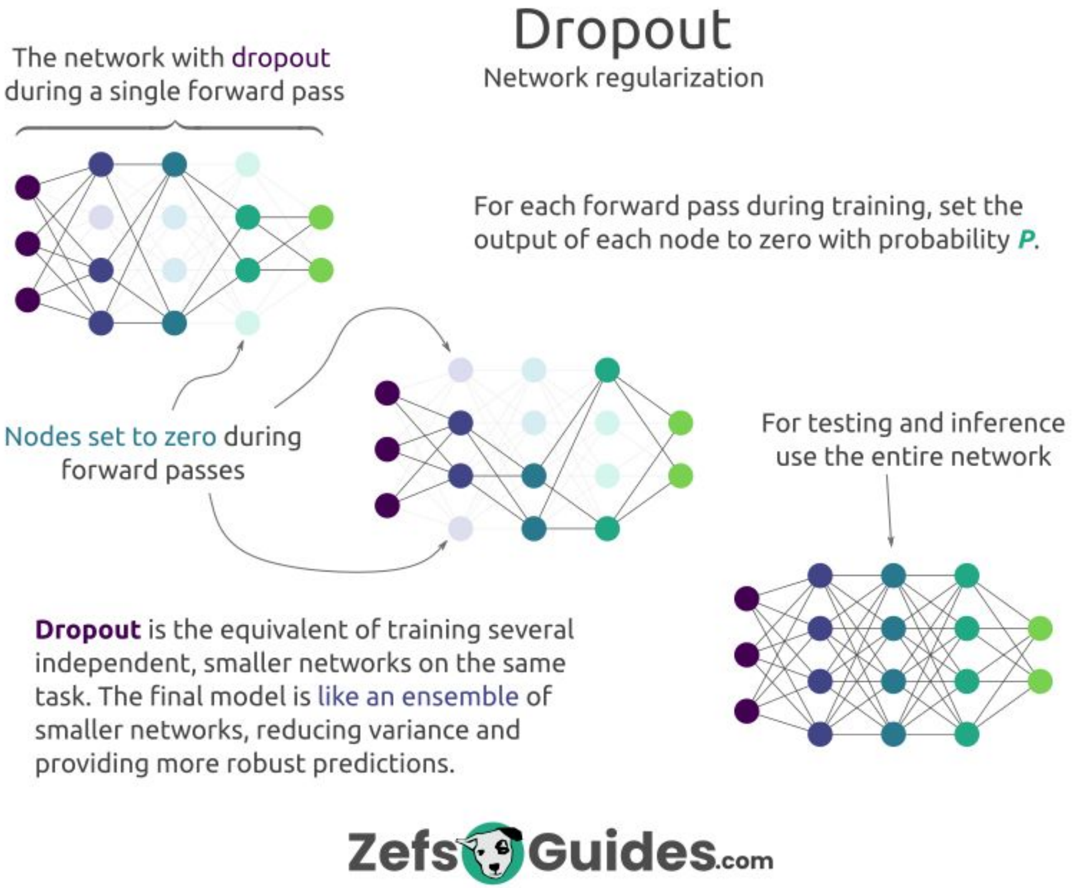
Early Stopping



Early Stopping



Drop Out



Dropout
Network regularization
The network with dropout during a single forward pass
For each forward pass during training, set the output of each node to zero with probability P.
Nodes set to zero during forward passes
For testing and inference use the entire network
Dropout is the equivalent of training several independent, smaller networks on the same task. The final model is like an ensemble of smaller networks, reducing variance and providing more robust predictions.
ZefsGuides.com

References

https://www.jeremyjordan.me/neural-networks-training/

https://www.reddit.com/r/learnmachinelearning/comments/x89qsi/dropout_in_neural_networks_what_it_is_and_how_it/?rdt=55837

https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

https://medium.com/towards-data-science/derivative-of-the-softmax-function-and-the-categorical-cross-entropy-loss-ffceefc081d1