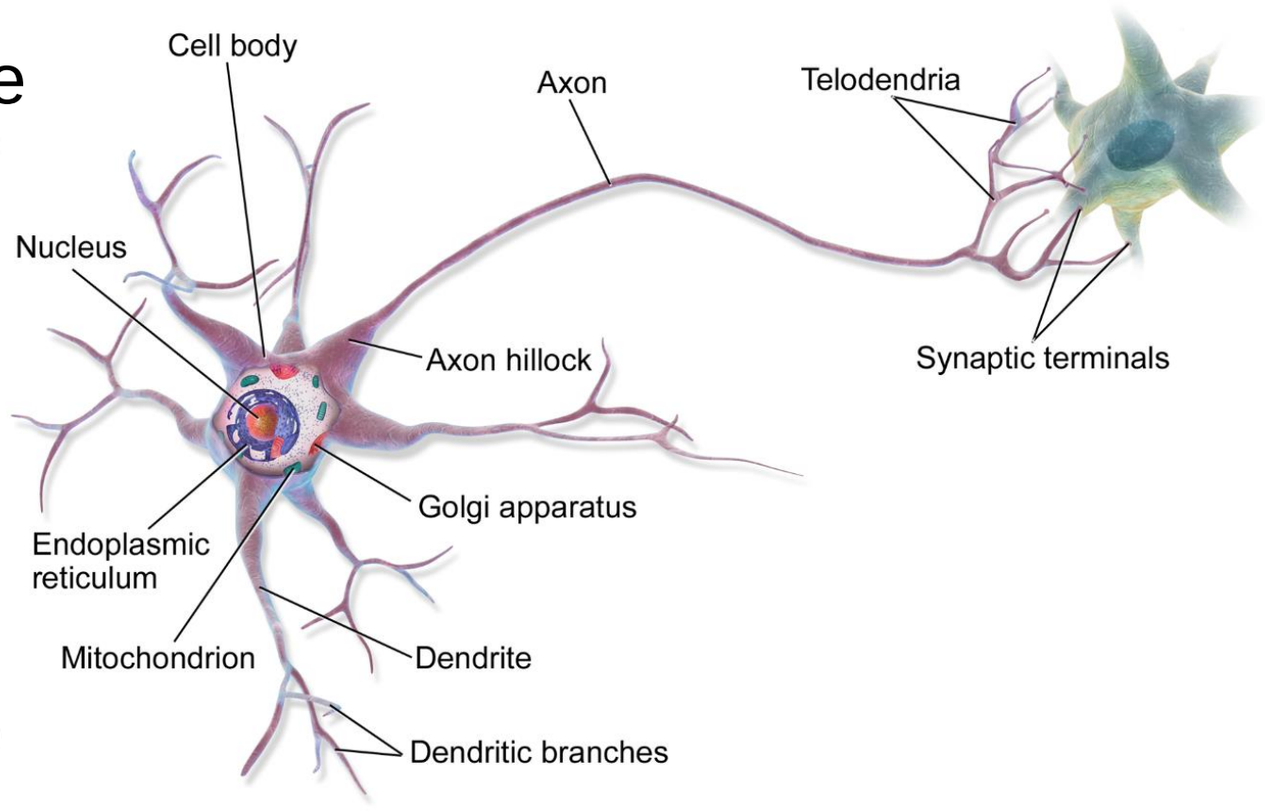
Neural Networks

Artificial Neural Networks

- Vaguely inspired by the biological neural networks that constitute animal brains

- The human brain is made of close to 100 billions of neurons interconnected by synapses.



- A neuron processes and transmits information through electrical and chemical signals that are carried via the synapses

- Neurons can connect to each other to form neural networks – each neuron can be connected with about 5,000 other neurons

Artificial NNs Vs Biological NN

greatlearning
Learning for Life

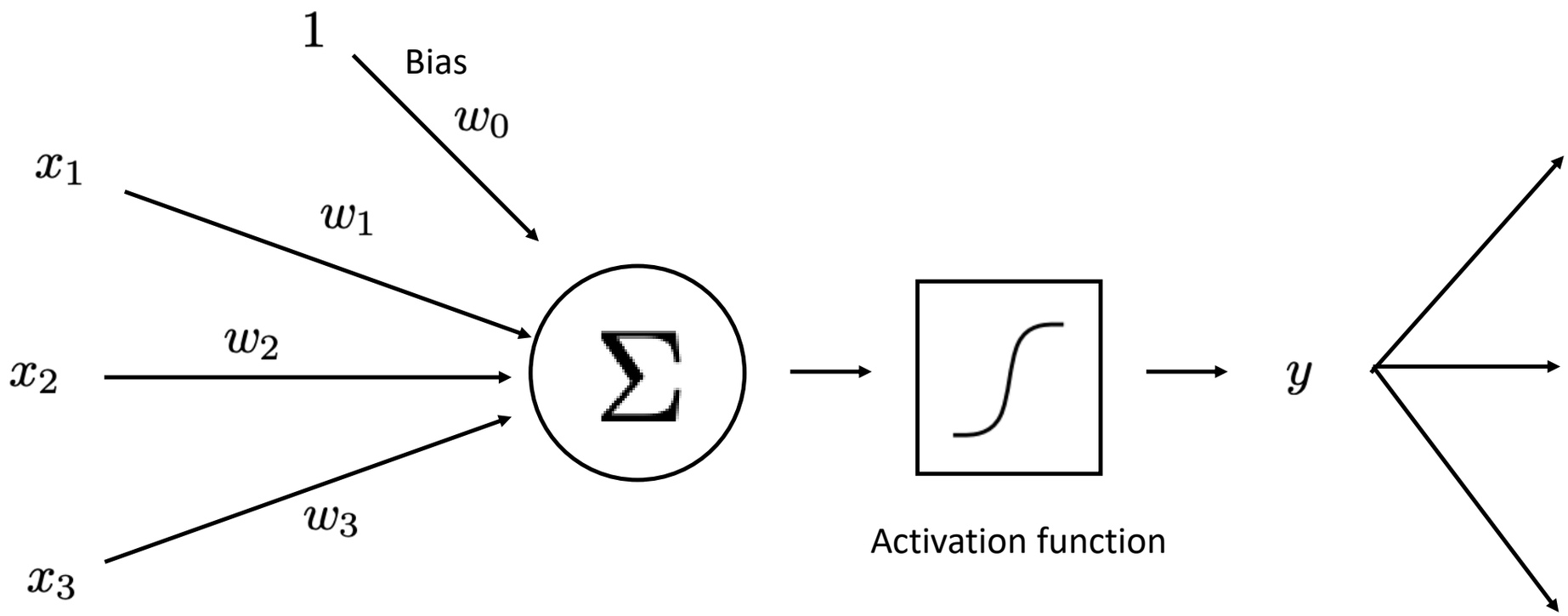
	ANN	BNN
Size	10-1000 Neurons & 1000's of synapses	86 Billion Neurons & >100 Trillion synapses
Topology	Usually Feed-forward, that is computed Layer by Layer	Complex Network that is computed asynchronously
Calculation Speeds	nano seconds	milli seconds
Power	100 watts (~100Watts)	20 watts
Others	Not Fault Tolerant, Learning	Fault Tolerant, Learning?

Applications

- Neural Nets have done exceptionally well at tasks like
  - Image Recognition, Character Recognition, Face Recognition
  - Feature Extractions, Finger print processing, Signature matching
  - Speech Recognition
- Other modest successes in: Stock market predictions, Combinatorial optimization, Medicine etc.

The Perceptron

- Perceptron: The main building block

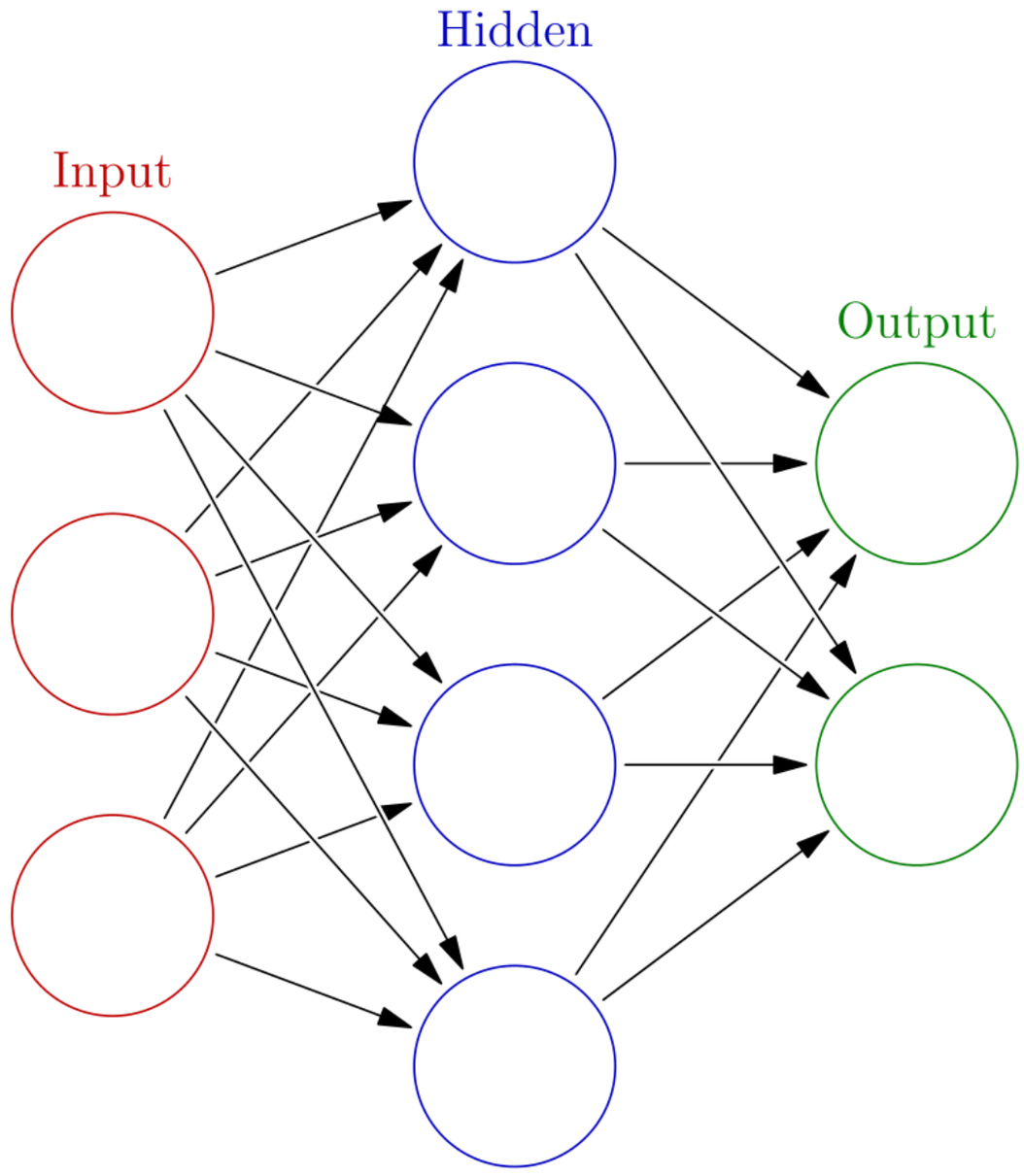


$$y = f\left(\sum w_i x_i\right)$$

Activation function

The Artificial Neural Net

- Number of Layers
  - Single Vs
  - Multi Layer
- Number of Nodes in each Layer
- Weights/connections
- Activation or Transfer function



Examples of Activation functions

greatlearning
Learning for Life

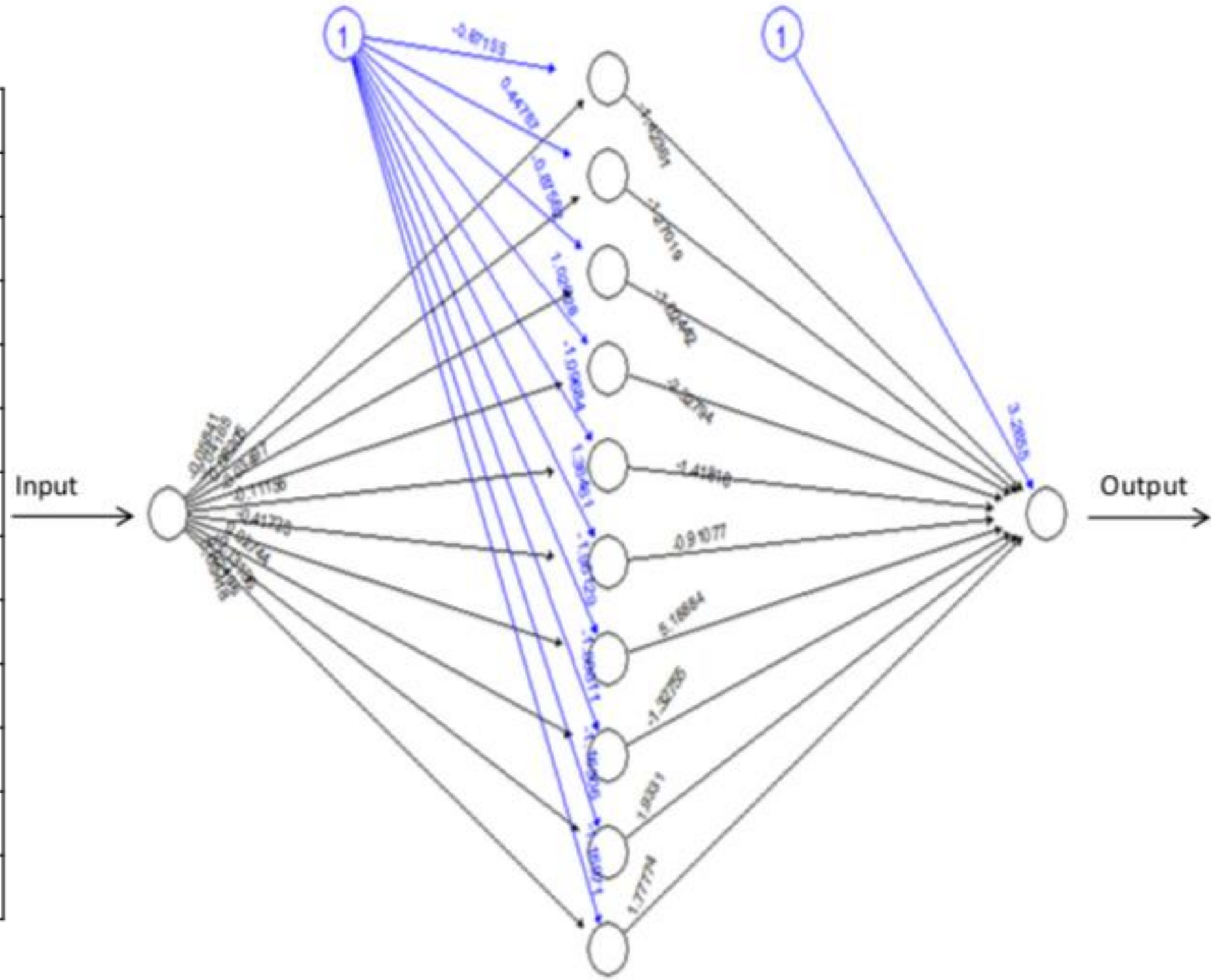
- ReLu (with Softmax/Linear)
- Sigmoid (Logistic)
- Hyperbolic Tangent (tanh)
- Step function (Heaviside)
- Softmax (Generalized Logistic)
- Linear

- Which one do we use?
  - There is no set procedure/Rule
  - ReLu has become very popular

An Example

greatlearning
Learning for Life

Training Data	
Input	Output
1.0	1.0
4.0	2.0
9.0	3.0
90.25	9.5
57.74	7.6
14.44	3.8
23.04	4.8
5.29	2.3
3.61	1.9
5.76	2.4
13.69	3.7



Learning

- How to determine the weights?
  - Start with guess values for weights
  - Calculate outputs from inputs
  - Compare outputs to desired outputs: Calculating errors
  - Training algorithms update the weights in a way to minimize the errors (cost functions)
- Cost (loss) functions: measures how close an output is to the desired output. Preferably:
  - Non-negativity
  - Globally continuous and differentiable

Training Also: Back Propagation

- Back propagation of errors, is a common algorithm to train artificial neural networks used in conjunction with an optimization method such as gradient descent.
- The method calculates the gradient of a loss function with respect to all the weights in the network.
- The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.

Gradient Descent

- A first-order optimization algorithm.
- Essentially equivalent to falling down on a slope to eventually find the minimum (lowest point in the valley).
- To find a minimum (valley) of a function, take a small step along the steepest direction. And keep iterating.
- For finding maximums we'd do gradient ascent.

$$w_i^{new} = w_i - \eta \frac{\partial E}{\partial w_i}$$

Learning rate

- Choosing the Learning rate (η)
  - Too small, we will need too many iterations for convergence
  - Too large, we may skip the optimal solution
- Adaptive Learning Rate : start with high learning rate and gradually reduce the learning rate with each iteration

$$\eta_n = \frac{\eta_1}{n}$$

- Can also be chosen by Trial & Error – Use a range of learning rate (1, 0.1, 0.001, 0.0001) and use the results as a guide

Epochs, Batch size, Iterations

- When dataset is too large, passing all the data through a Neural net before we make weight updates is computationally expensive
- Instead we would create data batches with smaller batch size.
- After each batch is passed and weights updated, we will count it as one iteration.
- When an entire dataset is passed forward and backward (weights updated) through the neural network, we will count it as one epoch.
  - Too few epochs: under fitting, Too many: overfitting
  - Batch training: All of the training samples pass through the neural net, before weights are updated
  - Sequential training: Weights are updated after each training vector is passed through the neural net.

Scaling

- Scaling the Variables
  - The non-linearities in the activation function and the numerical rounding errors make input scaling quite important
  - Scaling can accelerate learning and improve performance

Overfitting

- Neural Network Models are susceptible to overfitting
  - Large number of weights and biases
  - Excessive learning (Epocs) on training data

- Ways to avoid Overfitting
  - Increase sample size
  - Early stopping
  - Reduce Network Size
  - Regularization

Regularization (weight decay)

- Regularization is a technique used to avoid this overfitting problem.
- The idea behind regularization is that models that overfit the data are complex models that have for example too many parameters.
- Regularization penalizes the usual loss function by adding a complexity term that would give a bigger loss for more complex models.
- Types of Regularization
  - LASSO
  - Ridge
- Optimal value of λ, the decay rate or penalty coefficient, is determined through cross-validation

Hyper parameters and tuning

- Hyper parameters are the variables which determines the network structure and the variables which determine how the network is trained
  - Number of hidden layers
  - Number of neurons in hidden layers
  - Decay factor
  - Number of Epoch
  - Learning Rate
  - The Activation Function
- Tuning these to ensure that you don’t overfit is an art.