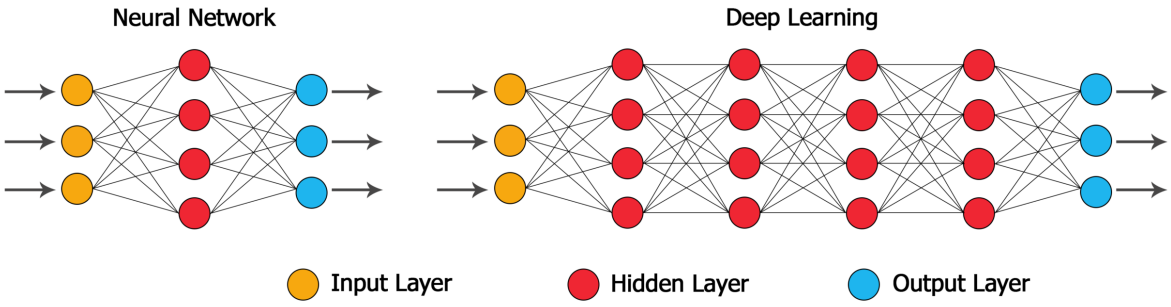
Artificial Neural Network (ANN)

Artificial Neural Networks (ANN) is a supervised learning system built of a large number of simple elements, called neurons or perceptrons. Each neuron can make simple decisions, and feeds those decisions to other neurons, organized in interconnected layers. Together, the neural network can emulate almost any function, and answer practically any question, given enough training samples and computing power.

A shallow neural network has only three layers of neurons:

- An input layer that accepts the independent variables or inputs of the model
- One hidden layer
- An output layer

A Deep Neural Network (DNN) has a similar structure, but it has two or more hidden layers of neurons that process inputs. Additional layers are useful up to a limit of 9-10, after which their predictive power starts to decline. Today most neural network models and implementations use a deep network of between 3-10 neuron layers.



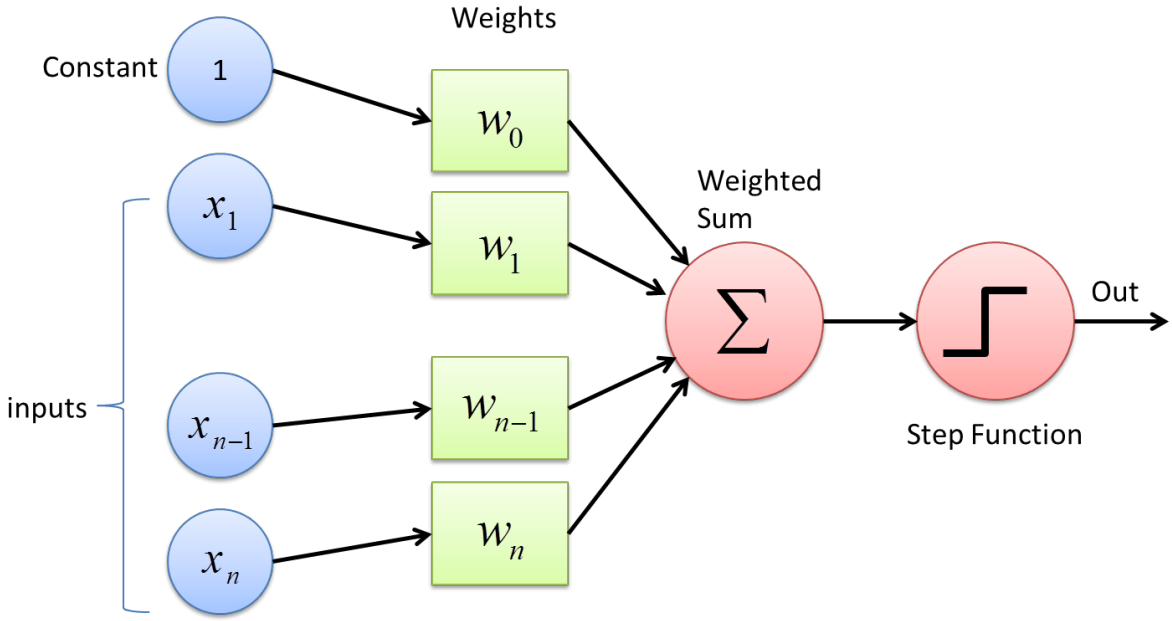
Neural Network Concepts

- **Inputs**: Inputs to a neural network are typically a set of real values. Each value is fed into one of the neurons in the input layer.
- **Training Set**: A set of inputs for which the correct outputs are known, used to train the neural network.
- **Outputs**: Neural networks generate their predictions in the form of a set of real values or boolean decisions. Each output value is generated by one of the neurons in the output layer.
- **Perceptron**: The basic unit of the neural network. Accepts an input and generates a prediction. Each neuron accepts part of the input and passes it through the activation function. Common activation functions are sigmoid, TanH and ReLu. Activation functions help generate output values within an acceptable range, and their non-linear form is crucial for training the network.
- **Weight Space**: Each neuron is given a numeric weight. The weights, together with the activation function, define each neuron's output. Neural networks are trained by fine-tuning weights, to discover the optimal set of weights that generates the most accurate prediction.

- **Forward Pass**: Neurons generate their outputs and pass them on to the next layer, until eventually the network generates an output.
- **Error Function**: Defines how far the actual output of the current model is from the correct output. When training the model, the objective is to minimize the error function and bring output as close as possible to the correct value.
- **Backpropagation**: In order to discover the optimal weights for the neurons, we perform a backward pass, moving back from the network's prediction to the neurons that generated that prediction. This is called backpropagation. Backpropagation tracks the derivatives of the activation functions in each successive neuron, to find weights that brings the loss function to a minimum, which will generate the best prediction. This is a mathematical process called gradient descent.
- **Bias and Variance**: When training neural networks, like in other machine learning techniques, we try to balance between bias and variance. Bias measures how well the model fits the training set—able to correctly predict the known outputs of the training examples. Variance measures how well the model works with unknown inputs that were not available during training.
- **Hyperparameters**: A hyperparameter is a setting that affects the structure or operation of the neural network. In real deep learning projects, tuning hyperparameters is the primary way to build a network that provides accurate predictions for a certain problem. Common hyperparameters include the number of hidden layers, the activation function, and how many times (epochs) training should be repeated.

Perceptron

A perceptron is a binary classification algorithm modeled after the functioning of the human brain—it was intended to emulate the neuron. The perceptron, while it has a simple structure, has the ability to learn and solve very complex problems.



Backpropagation

After a neural network is defined with initial weights, and a forward pass is performed to generate the initial prediction, there is an error function which defines how far away the model is from the true prediction. There are many possible algorithms that can minimize the error function—for example, one could do a brute force search to find the weights that generate the smallest error. However, for large neural networks, a training algorithm is needed that is very computationally efficient. Backpropagation is that algorithm—it can discover the optimal weights relatively quickly, even for a network with millions of weights.

Procedure

- **Forward pass**: Weights are initialized and inputs from the training set are fed into the network. The forward pass is carried out and the model generates its initial prediction.
- **Error function**: The error function is computed by checking how far away the prediction is from the known true value.
- **Backpropagation**: The backpropagation algorithm calculates how much the output values are affected by each of the weights in the model. To do this, it calculates partial derivatives, going back from the error function to a specific neuron and its weight. This provides complete traceability from total errors, back to a specific weight which contributed to that error. The result of backpropagation is a set of weights that minimize the error function.
- **Weight update**: Weights can be updated after every sample in the training set, but this is usually not practical. Typically, a batch of samples is run in one big forward pass, and then backpropagation performed on the aggregate result. The batch size and number of batches used in training, called iterations, are important hyperparameters that are tuned to get the best results. Running the entire training set through the backpropagation process is called an epoch.

Activation Functions

An activation function is a mathematical equation that determines the output of each element (perceptron or neuron) in the neural network. It takes in the input from each neuron and transforms it into an output, usually between one and zero or between -1 and one. Classic activation functions used in neural networks include the step function (which has a binary input), sigmoid and tanh. New activation functions, intended to improve computational efficiency, include ReLu and Swish.

In a neural network, inputs, which are typically real values, are fed into the neurons in the network. Each neuron has a weight, and the inputs are multiplied by the weight and fed into the activation function. Each neuron's output is the input of the neurons in the next layer of the network, and so the inputs cascade through multiple activation functions until eventually, the output layer generates a prediction. Neural networks rely on nonlinear activation functions—the derivative of the activation function helps the network learn through the backpropagation process (see backpropagation above).

Methods to Avoid Overfitting

- **Retraining neural networks**: Running the same model on the same training set but with different initial weights, and selecting the network with the best performance.

- **Multiple neural networks**: Training several neural network models in parallel, with the same structure but different weights, and averaging their outputs.
- **Early stopping**: Training the network, monitoring the error on the validation set after each iteration, and stopping training when the network starts to overfit the data.
- **Regularization**: Adding a term to the error function equation, intended to decrease the weights and biases, smooth outputs and make the network less likely to overfit.
- **Dropout**: Randomly stop a certain percentage of neurons in every training iteration. This ensures some information learned is randomly removed, reducing the risk of overfitting.

Methods to Avoid Underfitting

- **Adding neuron layers or inputs**: Adding neuron layers, or increasing the number of inputs and neurons in each layer, can generate more complex predictions and improve the fit of the model.
- **More training samples**: Adding more training samples or improving quality—the more training samples you feed into the network, and the better they represent the variance in the real population, the better the network will perform.
- **Decreasing regularization parameter**: Regularization can be overdone. By using a regularization performance parameter, you can learn the optimal degree of regularization, which can help the model better fit the data.