CPE504

Artificial Neural Networks (ANNs)

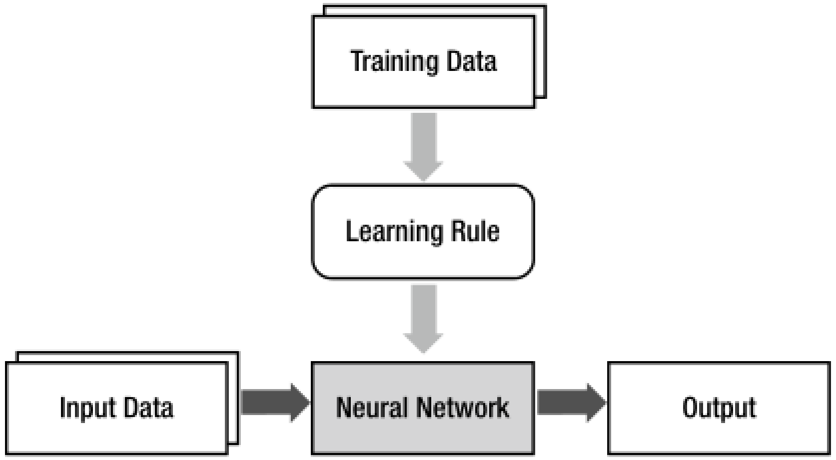
2.0. INTODUCTION: ARTIFICIAL NEURAL NETWORKS

What you will learn

- Relation: ANN to ML
- Relation: ANN to the Brain
- Supervised Learning of ANN
- Delta Rule
- Generalized Delta Rule
- Summary

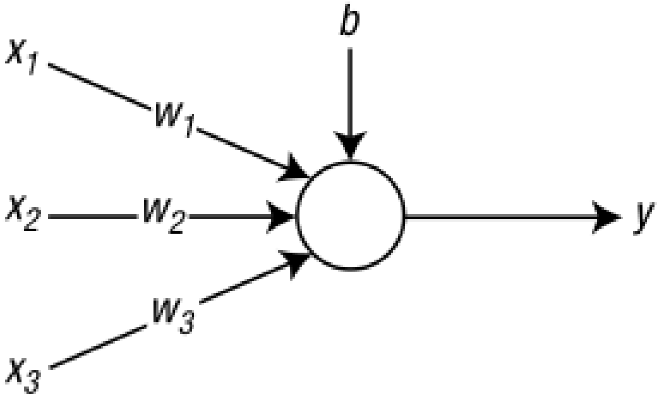
Relation: ANN to ML

- ANNs have a long history of development through numerous research works dating back to the early 1950s.
- A ML Model can be implemented in various forms. An ANN is one of them. The Figure to the left illustrates the relationship between ML and the ANN.
- Note that the **neural network block represents the ML model**, and the **learning rule block represents the actual machine learning** (**optimization**).
- The **algorithm** that determines the parameters or structure of the ANN model with respect to the input data is called the **learning rule**.



Relation: ANN to the Brain

- The brain is a gigantic network of neurons. The connection or association of these neurons forms specific information.
- The brain learns by altering the association of these **neurons**. ANN fundamentally mimics **biological neurons.** This is achieve using connections of **nodes** called **activation functions.**
- The brain stores **knowledge inferred from learning**. Similarly, when learning from input data, the associations between these nodes are controlled using stored **connection weights.**
- A **simple example of this idea** is illustrated by the **single node ANN** figure on the left.
- This network receives an input with 3 features **x**, controlled by three corresponding weights **w** and a bias **b** to give an output **y.**
- **Note:** The bias is an offset connection-weight.
- The fundamental ANN model **stores learned information** using connection weights, which the **computer saves to memory**.



Relation: ANN to the Brain…

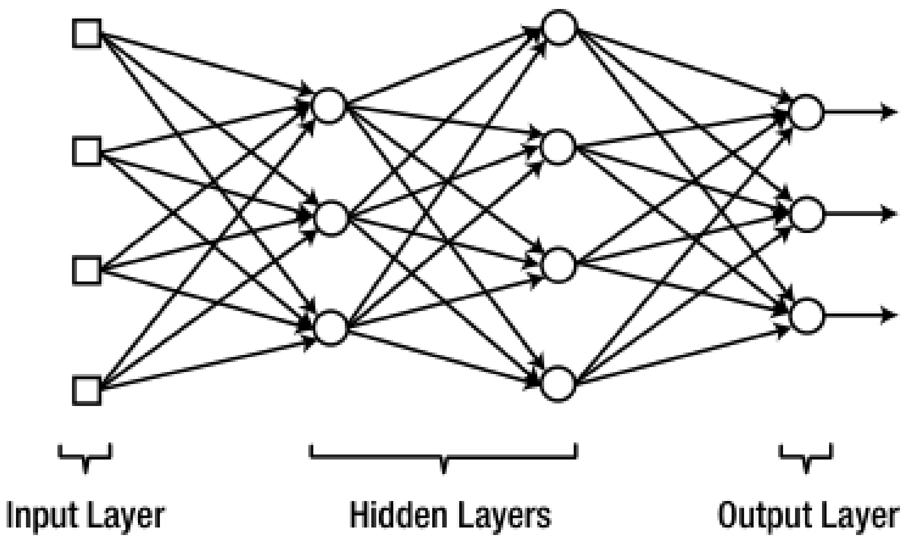
- To obtain **y**, the actual input **v** to the node is a weighted sum of **x**, which can be written in **matrix** form
- $\boldsymbol{v} = \boldsymbol{w}\boldsymbol{x}$
- $\boldsymbol{y} = \Phi(\boldsymbol{v})$
- There are two basic process at the node.

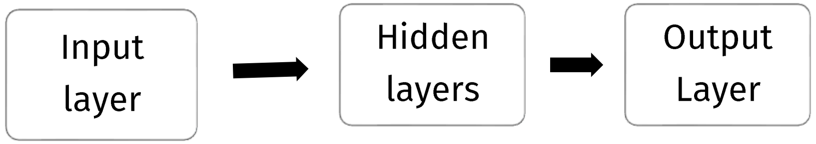
1. **compute** the input **v** to the **node** as a weighted sum of the feature input signals **x**
2. **compute** the output **y** of the **node** (**activation** or **transfer function**)

- $\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ b \end{bmatrix}$
- $\boldsymbol{x} = [1\ x_1 x_2 x_3]$
- As the **brain** is a gigantic **network** of **neurons**, the ANN (we can skip the artificial in the name) is a network of nodes.
- Different type of neural networks can be created depending on how the nodes are connected. This leads to **layers**.
- The most commonly used neural network types employs a **layered structure** of nodes as shown in the Figure, on the next page.

Relation: ANN to the Brain...

- The structure consist of the input layer. Which transmits the input signals to the hidden circular nodes.
- The nodes at the hidden (in between the input and output layers) and the output layer (rightmost nodes) calculate the weighted sum and activation function.
- The output nodes emit the final result of the neural network.
- The hidden layers are named so, because they are not accessible from the outside of the neural network.
- The figure to the left can also be represented by 3 rectangular layers

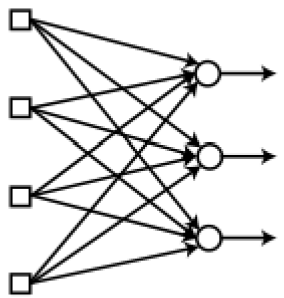




Relation: ANN to the Brain...

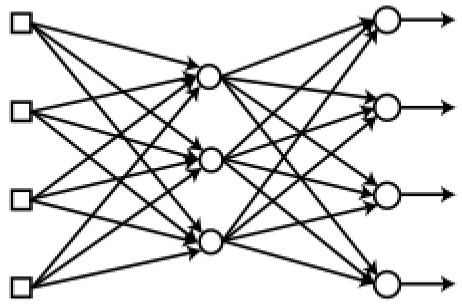
- Neural networks have been developed from **simple** architectures to extremely **complex** structures.
- Initially, neural network pioneers had a very simple architecture with only input and output layers, which are called **single-layer neural networks.**
- When hidden layers are added to a single-layer neural network, this produces a **multi-layer neural network**.
- The multi-layer neural network that has a single hidden layer is called a **shallow neural network** (vanilla ANN)
- A multi-layer neural network that contains two or more hidden layers is called a **deep neural network**. Such as Convolutional Neural Networks, Recurrent Neural Networks, etc..
- Historically, ANNs started as single-layer architectures evolved to the shallow and then the deep architectures.
- Deep architectures had not been seriously highlighted until the early 1990s to mid 2000s, after about 40 years since the development of the shallow architecture.
- Therefore, **for a long time**, the multi-layer neural network meant just the single hidden-layer neural network.
- When the need to distinguish multiple hidden layers arose, they separate name - deep neural network emerged.

Relation: ANN to the Brain…



1. **Single-Layer** Neural Network

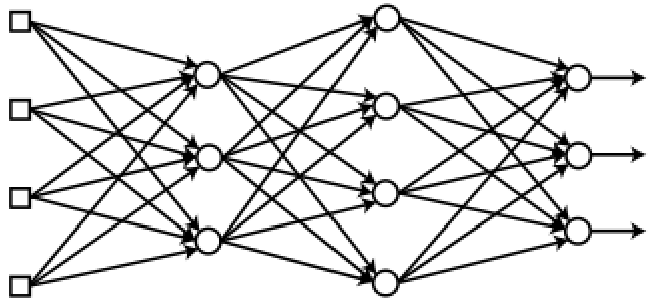
Input Layer – Output Layer



2. **Multi-Layer** Neural Network

- **Shallow** Neural Network

Input Layer – Hidden Layer – Output Layer



- **Deep** Neural Network

Input Layer – Hidden Layers – Output Layers

Relation: ANN to the Brain...

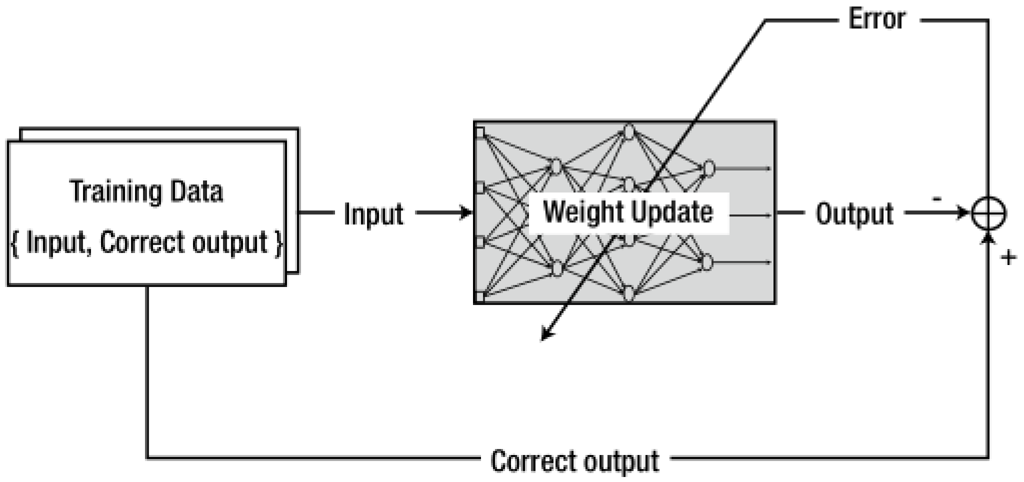
- In the layered network, the signal enters the input layer, the nodes performs nothing more than **signal (data or information) processing**.
- **ANN** is the **leading ML** in areas such as image (or vision) recognition, speech recognition, natural language processing, etc.
- The key part of this processing after the **feature identification input layer** (as seen in ANNs like the convolutional neural net) is the activation function power to aid this processing.
- The **choice of activation functions** is therefore **important** for ANN.

Supervised Learning of ANN

- Generally, **supervised learning** of an **ANN** proceeds as follows:

1. **Initialize** the ANN **weights** with **adequate values**.
2. **Pass** the **input training data**, structured as { **input, correct output** } into the ANN
3. **Compute** the **output** from ANN and its **error** from the **correct output data**.
4. **Adjust** the **weights** to **reduce** the **error**.
5. **Repeat** Steps **2-4** for **all training data**

- The Figure to the left illustrates the concept of this **adaptive supervised learning**
- **A recursive process of TRY-TEST-ADJUST.**



Delta Rule

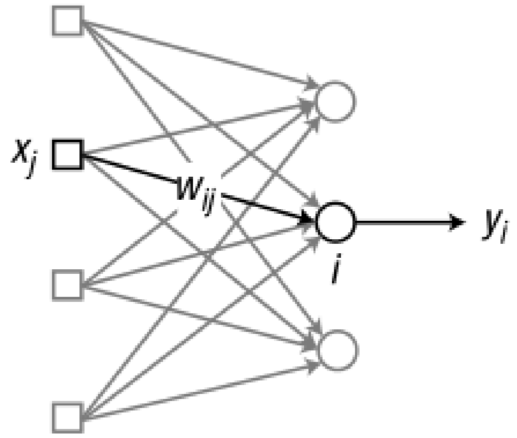
Case Study: A **Single Layer** ANN (**Perceptron**)

- The ANN stores information in the form of weights.
- To train the ANN with new information, the weights should be changed accordingly.
- The **systematic** approach to adapting the weights according to the given error information defined by what is called an **objective function** is called the **learning rule**.
- Since **training** is the **only way** for the neural network to **store the information** systematically, the **learning (or optimization) rule** is a **vital component** of **ANN research**.
- As an introduction, let us look at the **delta rule**, the representative and simple learning rule of the single-layer ANN architecture.
- It is very useful for studying the important concepts of the learning rule.
- It is also referred to as **Adaline** rule as well as **Widrow-Hoff** rule, from **signal processing** literature

Delta Rule…

- Consider a single-layer neural network, as shown in the figure. Here, d_i is the **correct output** of the **i**-th **output node**.
- The activation function at the nodes are taken to be unity.
- $y_i = v_i$
- The **delta rule algorithm** adjusts the weight as follows:
- If an input node contributes to the error of the output node, the weight between the two nodes is adjusted in proportion to the input value x_j and the output error e_i.
- $w_{ij} = w_{ij} + \alpha e_i x_j$

- x_j is the output from the **input** node **j,** (j = 1 2 3),
- e_i is the error of the **output** node **I,**
- w_{ij} is the weight between the output node **i** and input node **j,**
- α is the **learning rate** ($0 < \alpha \leq 1$)



Delta Rule…

- **α** is a very important **hyper-parameter** that determines **how much** the **weight** is **changed per time** during learning.
- If this value is **too high**, the output may wander around the solution and **fail to converge**.
- In contrast, if it is **too low**, the calculation may reach the solution too **slowly.**
- In **summary**, the supervised learning training process using the delta rule for the single-layer neural network is:
- 1. **Initialize** the **weights** at **adequate values**.
- 2. **Pass** an **input pattern** from the training data structure { input, correct output } to the neural network. **Compute** the **error** of the ANN output **y** from the correct output **d**.
- $e_i = d_i - y_i$
- 3. **Compute** the **weight-adjustment** according to the **delta rule**
- $\Delta w_{ij} = \alpha e_i x_j$
- 4. **Adjust** the **weights**
- $w_{ij} = w_{ij} + \Delta w_{ij}$
- 5. **Perform** Steps 2-4 for **all** training **data patterns**.
- 6. **Repeat** Steps 2-5 until the **error** reaches an **acceptable tolerance level**.

Delta Rule...

- The **number of training iterations,** in each of which all patterns in the training data goes through Steps 2-5 once is called an **epoch**.
- Setting epoch=10 means that the ANN goes through 10 repeated training iterations on the same dataset.
- The delta rule is a type of **numerical optimization** method called **gradient descent methods**.
- In fact, it can be stated that almost all structured root-finding algorithms are gradient-based.
- The idea is that the **gradient fall** starts from a current initial value and proceeds to the solution, as if a ball rolls down the hill in a steep path.
- In this analogy, the position of the ball is the error from the model, and the bottom is the solution. It is noteworthy that the **gradient descent method cannot drop the ball to the bottom with just one throw**.
- This is why, it is a **repetitive process** towards the **minimizing** the model's **objective error function**.
- Terms like: **root finding**, **gradient-descent, line-minimization mean the same thing, that is:** optimization or finding a solution to a **problem** expressed by an **objective function.**

Generalized Delta Rule

- The **key limitation** of the Adaline rule considered in the previous section is that it is not general to other forms of activation functions (**AFs**), except linear ones.
- A generalized form, uses the error gradient (**delta)** information in place of just the error information.
- $w_{ij} = w_{ij} + \alpha \frac{\partial E}{\partial w_j} = w_{ij} + \Delta w_{ij}$
- $\frac{\partial E}{\partial w_j} = \delta_i x_j, \quad \delta_i = y_i' e_i$
- y_i' is the **derivative** of the activation function of the **output node i**
- **Therefore,** for instance, using the **standard or simple sigmoid function** as an **AF,** in which the output ranges from 0 to 1.
- $y_i = \frac{1}{1+e^{-v_i}}$
- $y_i' = y_i \ (1 - y_i)$

The general matrix form of the generalized rule is then a kind of proportional control rule.

- $w = w + \alpha \frac{\partial E}{\partial w}$
- E is the objective error function, say $E = \frac{1}{2} \sum_p \sum_i^m e_i^2$ for each m outputs and each p patterns in the training data.
- $\partial E / \partial w$ is the first-order partial derivative of the objective with respect to the weights
- As, we will see in the next slides, this rule is usually applied **stochastically** to train an ANN model. That is: **at every iteration, pick a set or batch of random data patterns and try to line-optimize in that direction.**

Summary

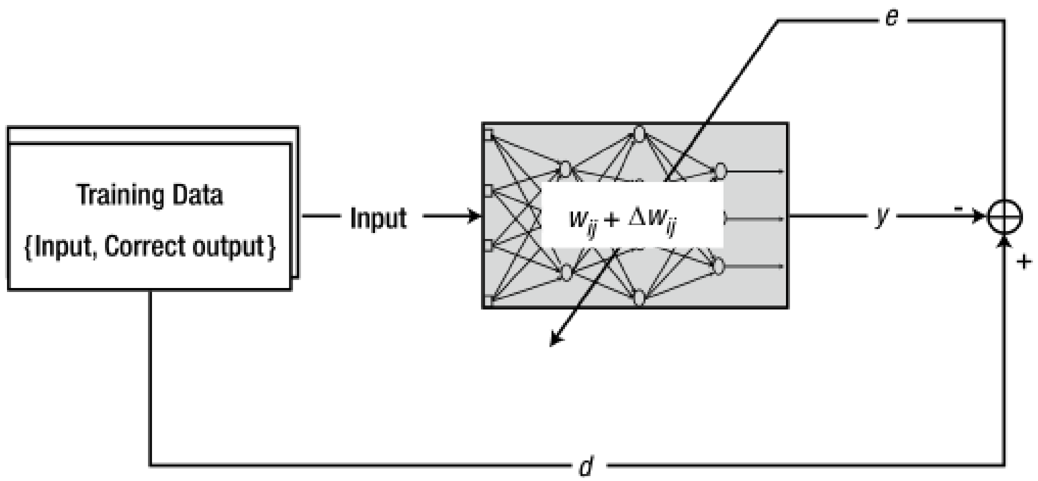
Recap:

- The ANN is a **layered (input to hidden to output) network of nodes**, which **imitate** the neurons of the brain. The **nodes** are **represented by an AF (activation function)**.
- **The choice of activation function is important.** The use of unity linear AFs may negate the effect of the hidden layer as seen in the single-layer NN.
- A **supervised learning rule** adjusts the **connection weights** to minimize the **error** between the correct output data and the ANN output.
- The **TRY-TEST-ADJUST** method used to **adapt or update the weights** according to the error is called the **learning rule**.
- The **delta rule** is the **representative learning rule** of the neural network.
- The **general delta rule** is a **gradient-descent** formula, which varies depending on the **activation function** and the **derivative error information** used.
- The delta rule is an **iterative method** that is, it gradually reaches the solution, by leveraging **repetition**, until the **error is reduced to some satisfactory level**.

Summary

Recap:

- **Setting the learning rate correctly is important.**
- **Note that:** we only considered the learning rule from the viewpoint of the **single-layer ANN.**
- **This supervised learning or training process for the ANN is illustrated as below**



Tools

Recommended Languages

- MATLAB (Fast Matrix Prototyping)
- JavaScript (Language of the Web)

Instructions to Student

- **All ANN functions will be written from scratch in form of custom libraries**
- **Learn to transfer maths to software.**
- **Copying of another person's code (or work) will be heavily penalized.**

Recommended Texts

Main Texts

- **MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence** by Phil Kim
- **Pattern Recognition and Machine Learning** by Christopher M. Bishop
- **Understanding Machine Learning: From Theory to Algorithms** by Shai Shalev-Shwartz and Shai Ben-David
- **PATTERNS, PREDICTIONS, AND ACTIONS: A story about machine learning** by Moritz Hardt and Benjamin Recht
- **Mathematics for Machine Learning** by Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong

Good luck!