


Module 1

Introduction to Neural Networks and Deep Learning



Syllabus

Introduction, The Basic Architecture of Neural Networks - Single Computational Layer: The Perceptron, Multilayer Neural Networks. Activation functions – Sign, Sigmoid, Tanh, ReLU, leaky ReLU, Hard Tanh, Softmax. Loss function. Training a Neural Network with Backpropagation. Practical issues in neural network training. Overfitting, Underfitting, Hyper parameters and Validation sets, Estimators -Bias and Variance. Introduction to deep learning, Deep feed forward network.

Introduction

Artificial Neural Network

- An artificial neural network consists of a pool of simple processing units which communicate by sending signals to each other over a large number of weighted connections.

Biological viewpoint: ANNs can be used to replicate and simulate components of the human (or animal) brain, thereby giving us insight into natural information processing.

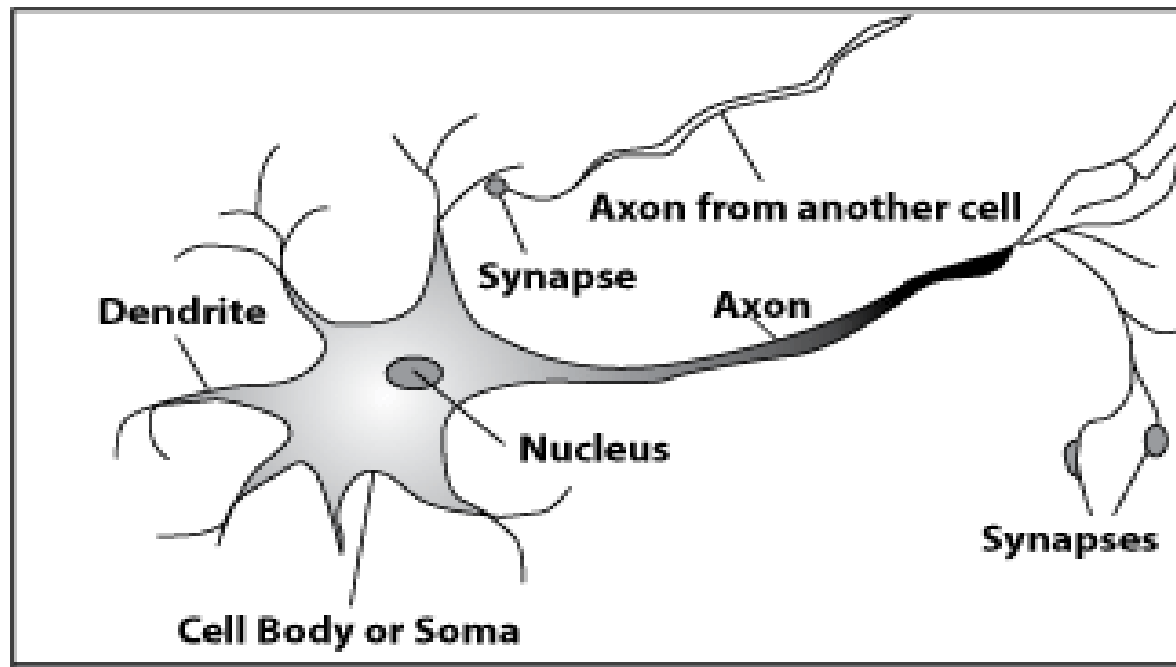
How do our brains work?

- The Brain is a massively parallel information processing system.
- Our brains are a huge network of processing elements. A typical brain contains a network of 100 billion neurons.



How do our brains work?

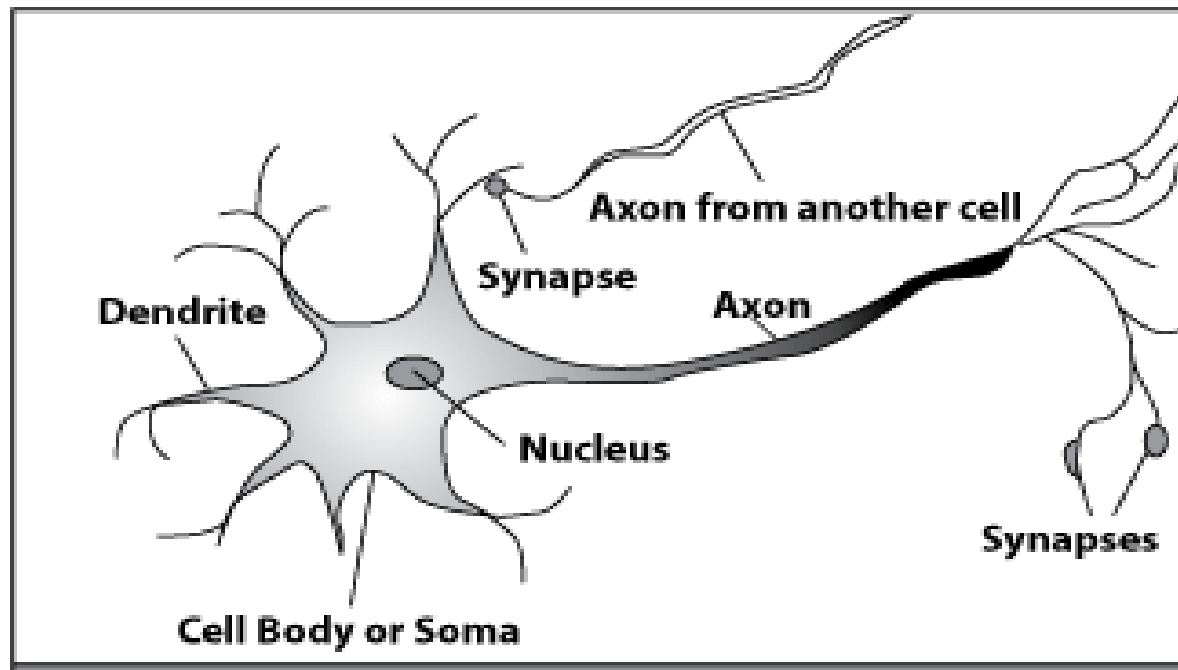
A processing element



Dendrites: Input Cell body: Processor
Synaptic: Link Axon: Output

How do our brains work?

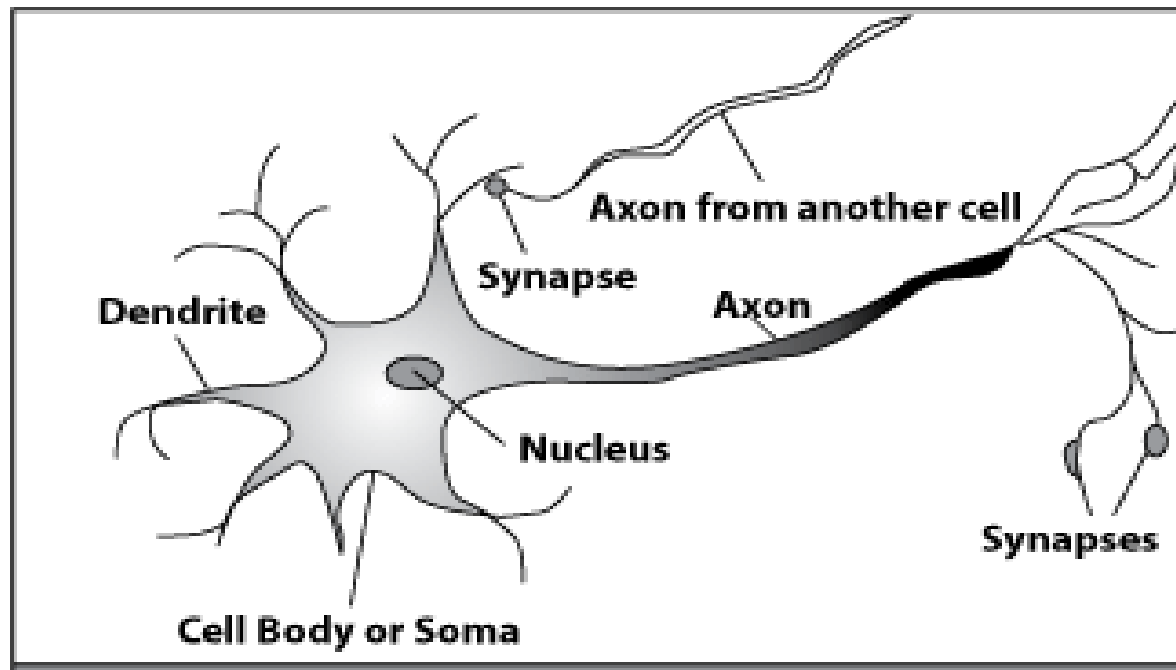
A processing element



A neuron is connected to other neurons through about *10,000 synapses*

How do our brains work?

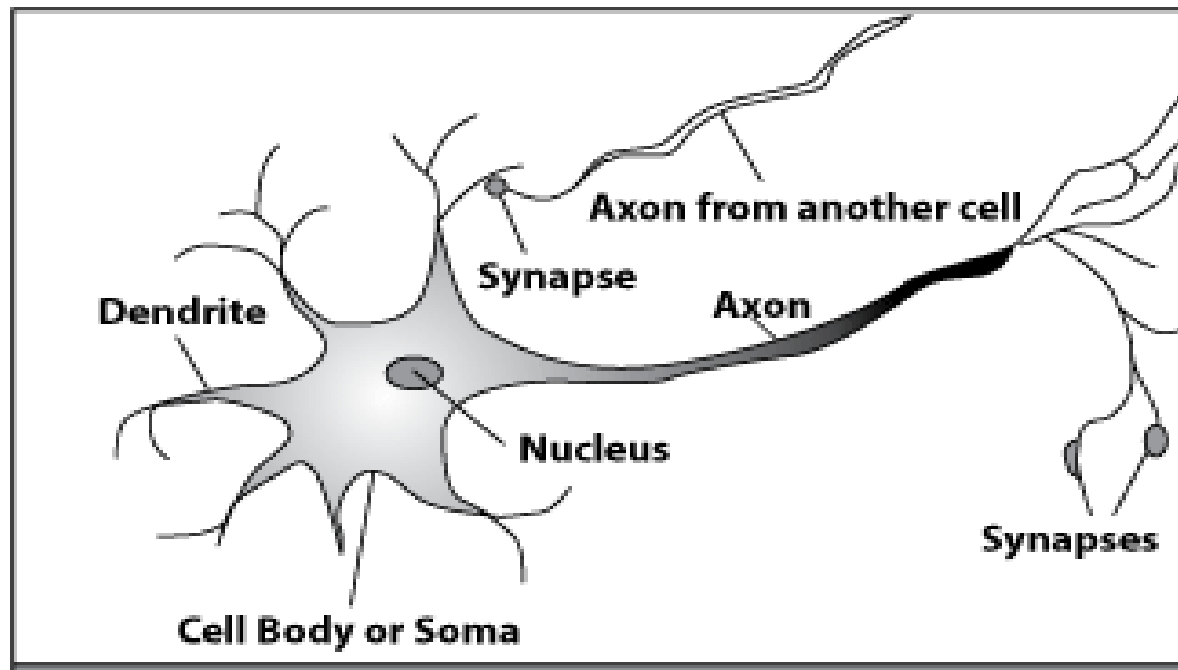
A processing element



A neuron receives input from other neurons. Inputs are combined.

How do our brains work?

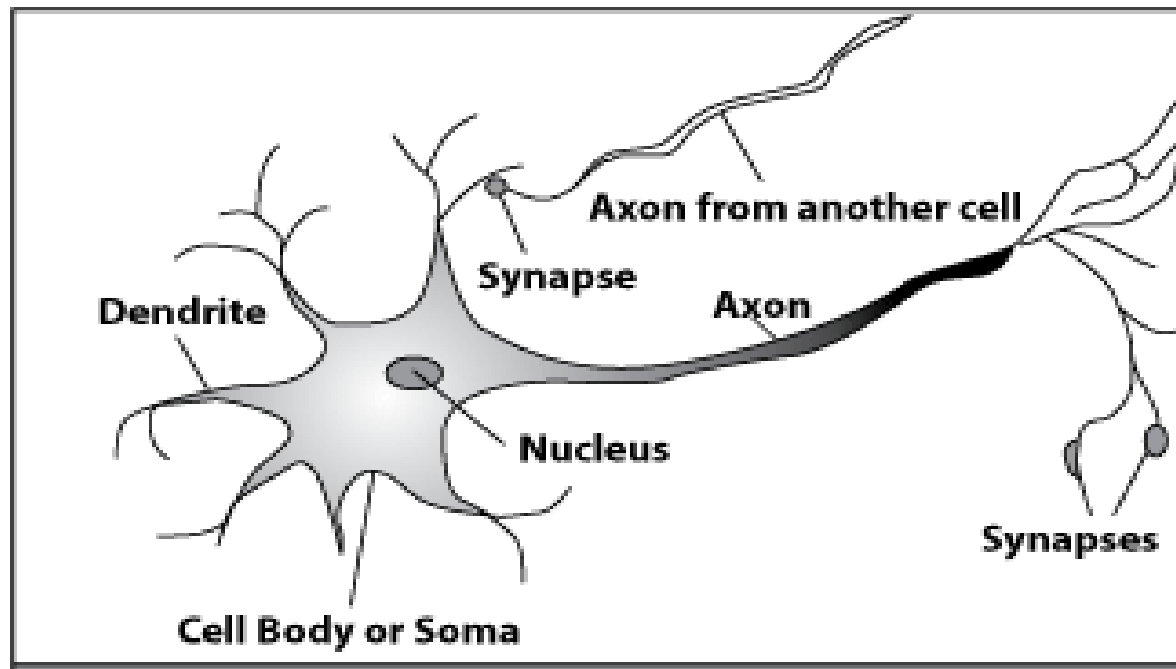
A processing element



Once input exceeds a critical level, the neuron discharges a spike an electrical pulse that travels from the body, down the axon, to the next neuron(s)

How do our brains work?

A processing element



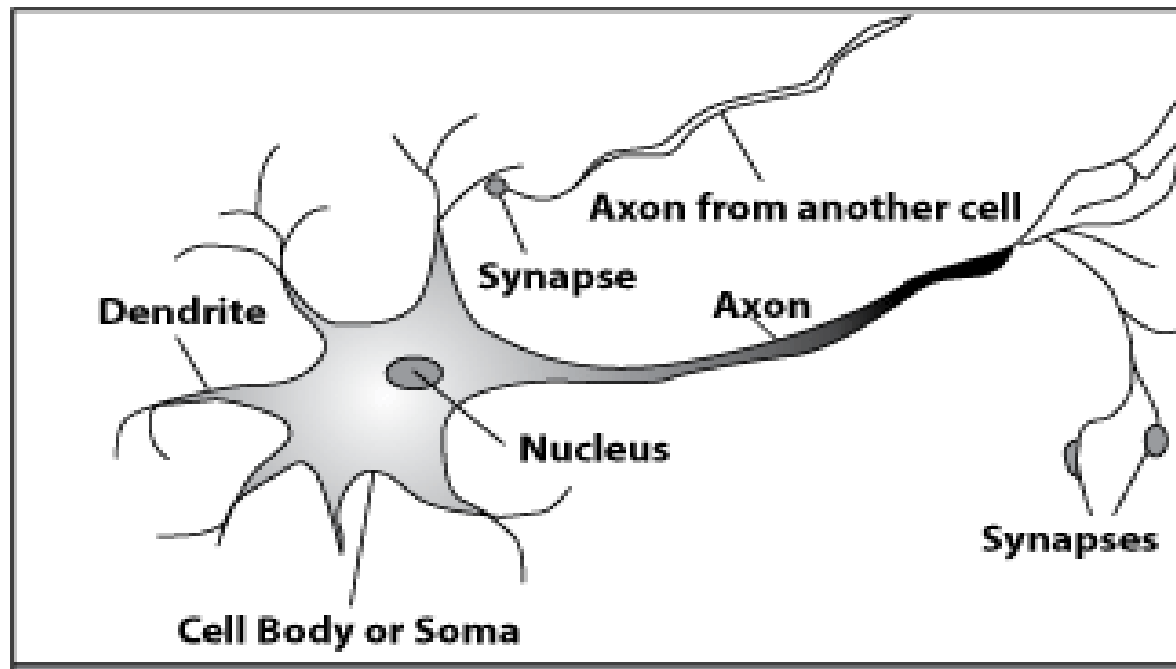
The axon endings almost touch the dendrites of the next neuron.





How do our brains work?

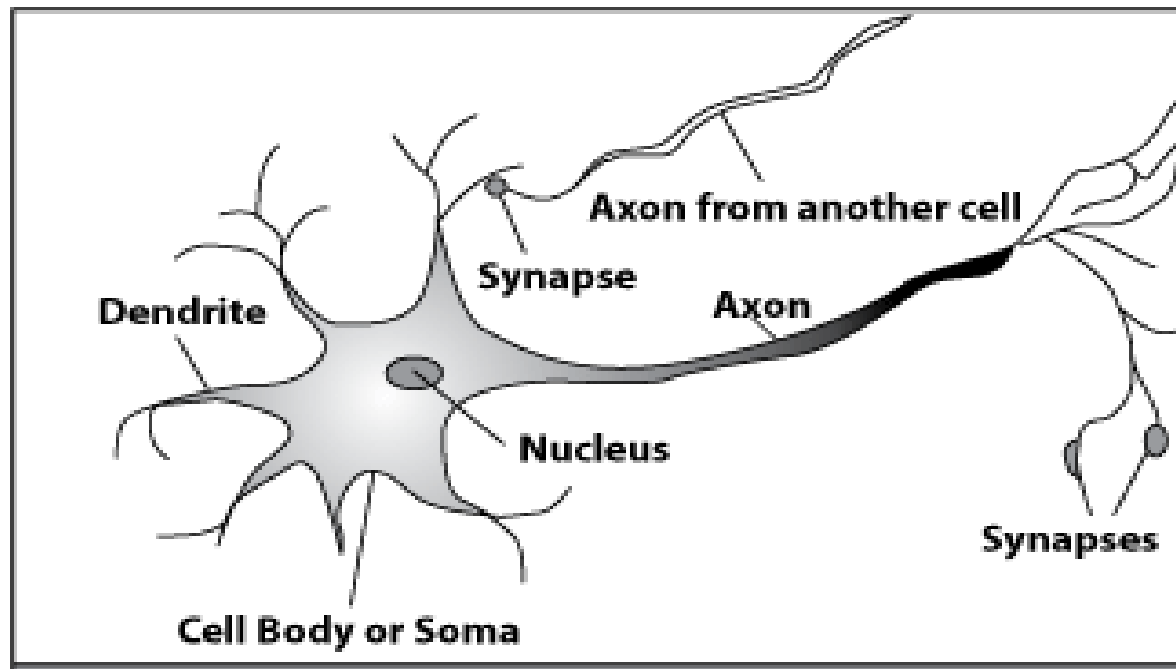
A processing element



Transmission of an electrical signal from one neuron to the next is effected by neurotransmitters.

How do our brains work?

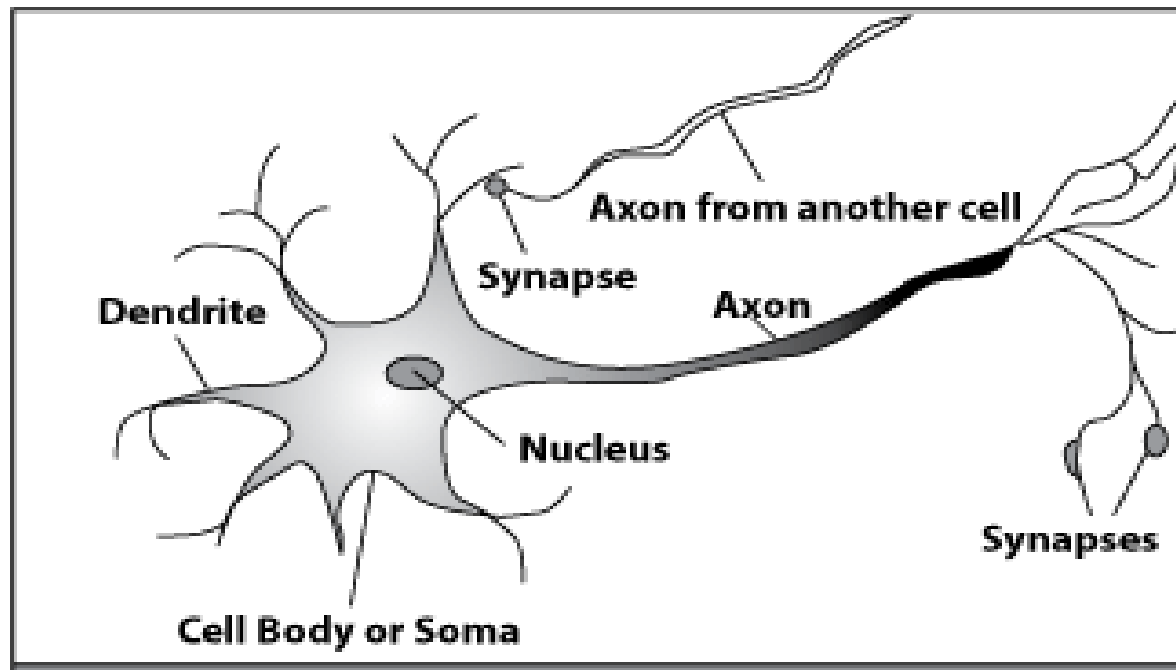
A processing element



Neurotransmitters are chemicals which are released from the first neuron and which bind to the Second.

How do our brains work?

A processing element



This link is called a synapse. The strength of the signal that reaches the next neuron depends on factors such as the amount of neurotransmitter available.

Introduction

Artificial Neural Networks

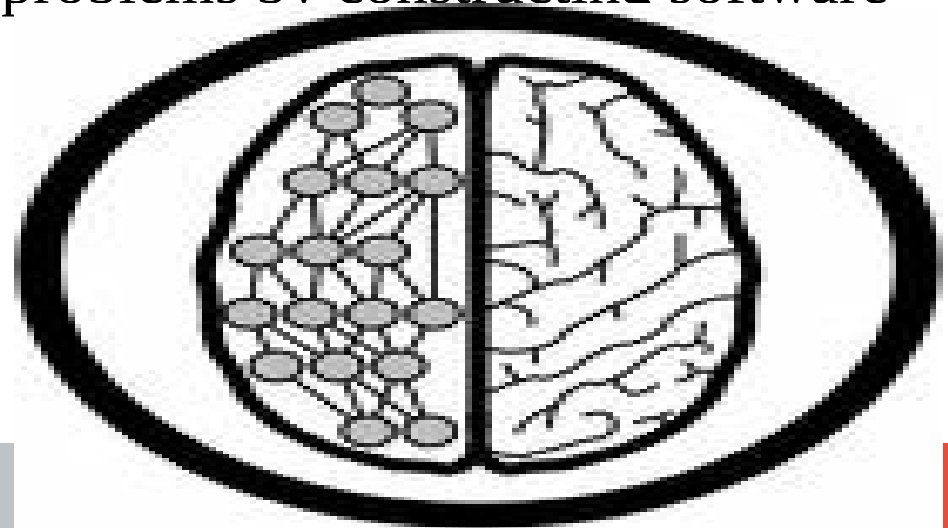
- The “building blocks” of neural networks are the **neurons**.
 - In technical systems, we also refer to them as **units** or **nodes**.
- Basically, each neuron
 - receives **input** from many other neurons.
 - changes its internal state (**activation**) based on the current input.
 - sends **one output signal** to many other neurons, possibly including its input neurons (recurrent network).

Artificial Neural Networks

- Information is transmitted as a series of electric impulses, so-called **spikes**.
- The **frequency** and **phase** of these spikes encodes the information.
- In biological systems, one neuron can be connected to as many as **10,000** other neurons.
- Usually, a neuron receives its information from other neurons in a confined area, its so-called **receptive field**.

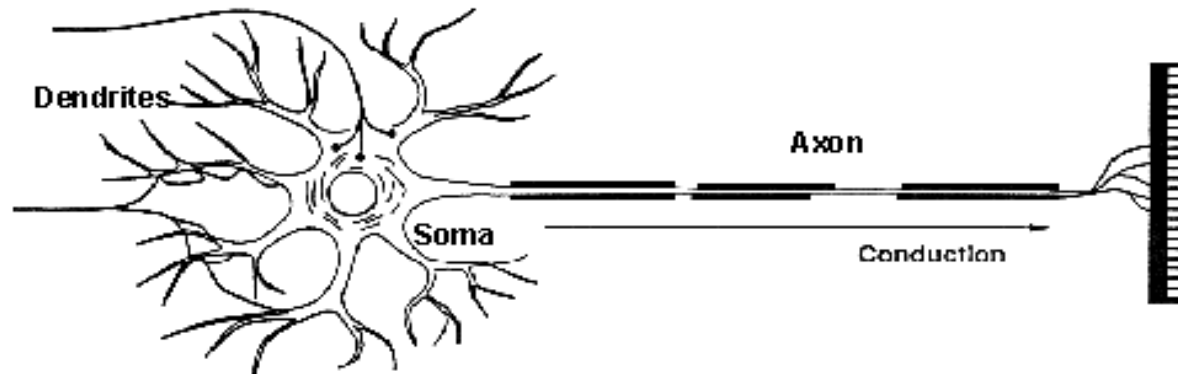
How do ANNs work?

- An artificial neural network (ANN) is either a **hardware implementation** or a **computer program** which strives to simulate the information processing capabilities of its biological exemplar. ANNs are typically composed of a great number of interconnected artificial neurons. The artificial neurons are simplified models of their biological counterparts.
- ANN is a technique for solving problems by constructing software that works like our brains.

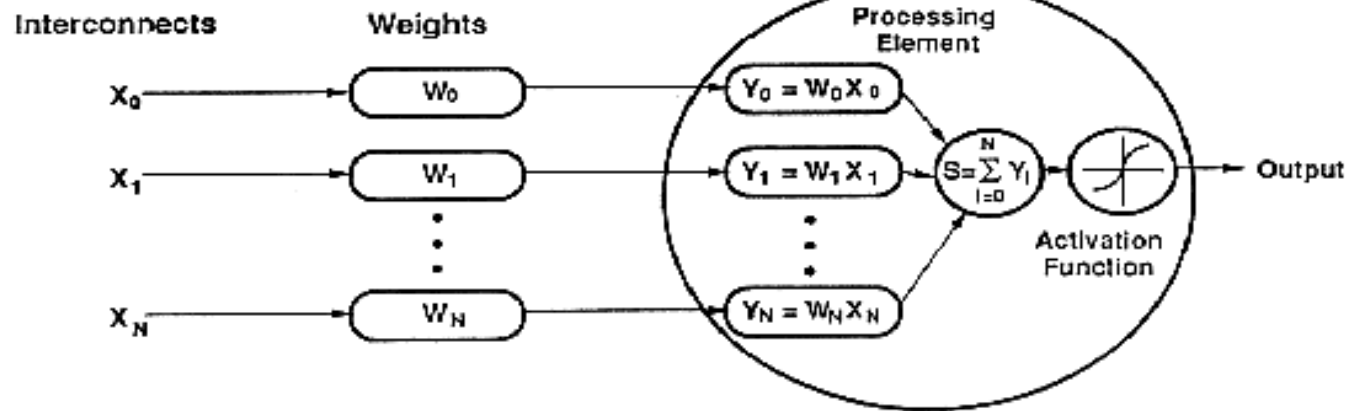


How do ANNs work?

Biological Neuron



Artificial Neuron

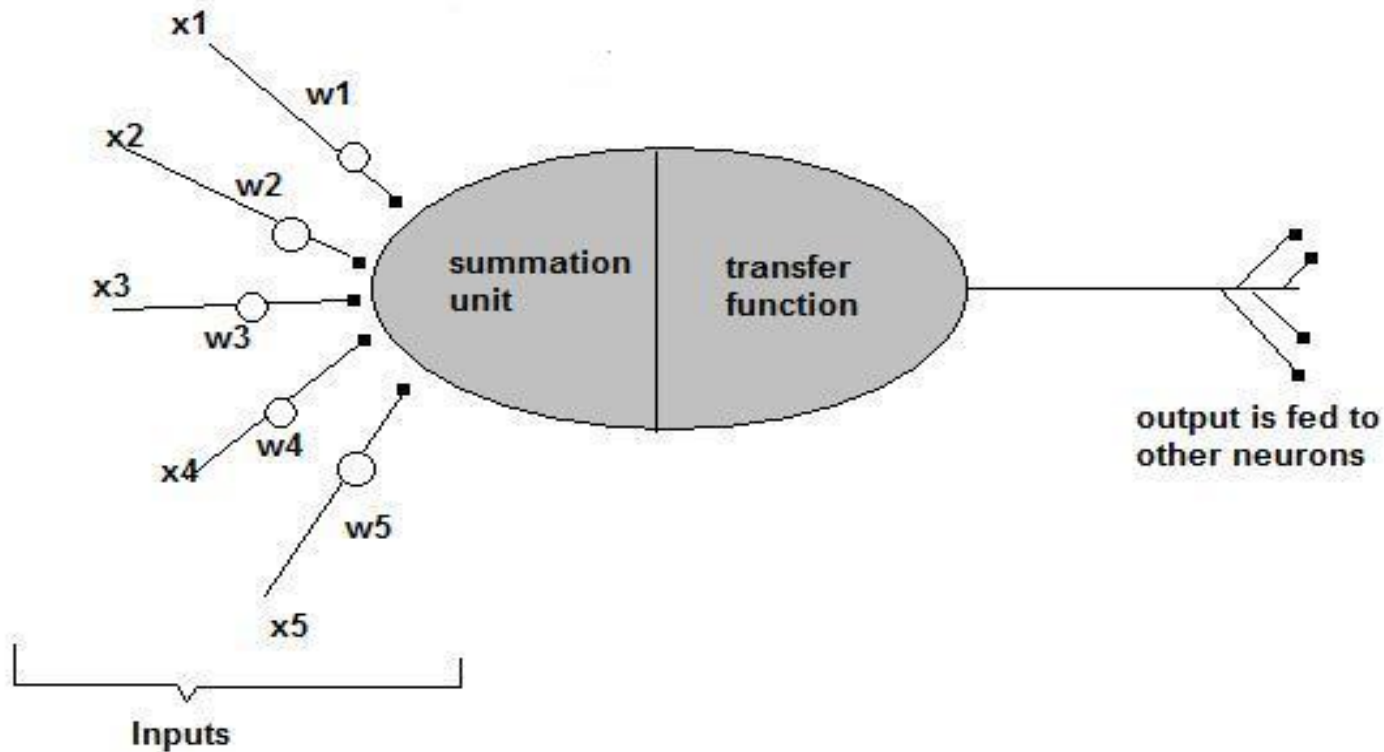


An artificial neuron is an imitation of a human neuron

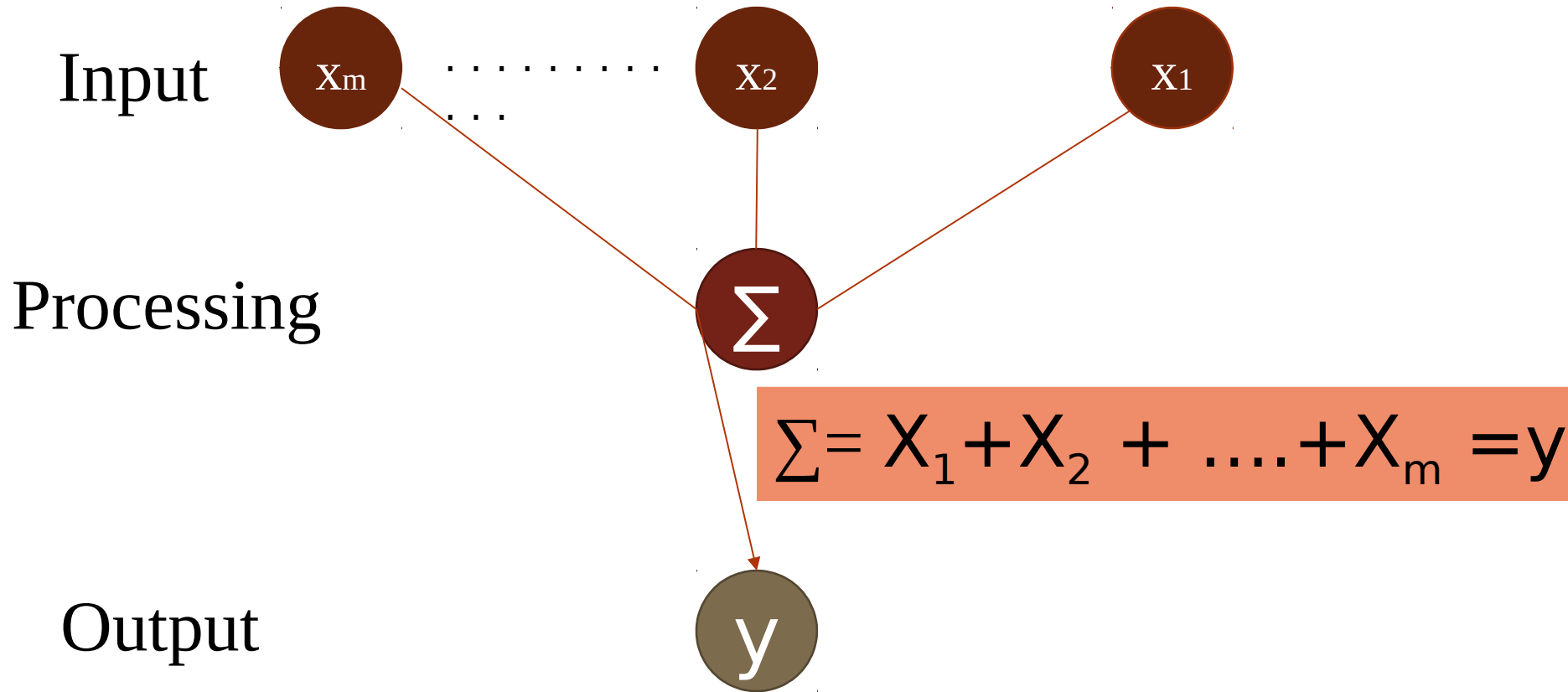
How do ANNs work?

- Now, let us have a look at the model of an artificial neuron.

A Single Neuron

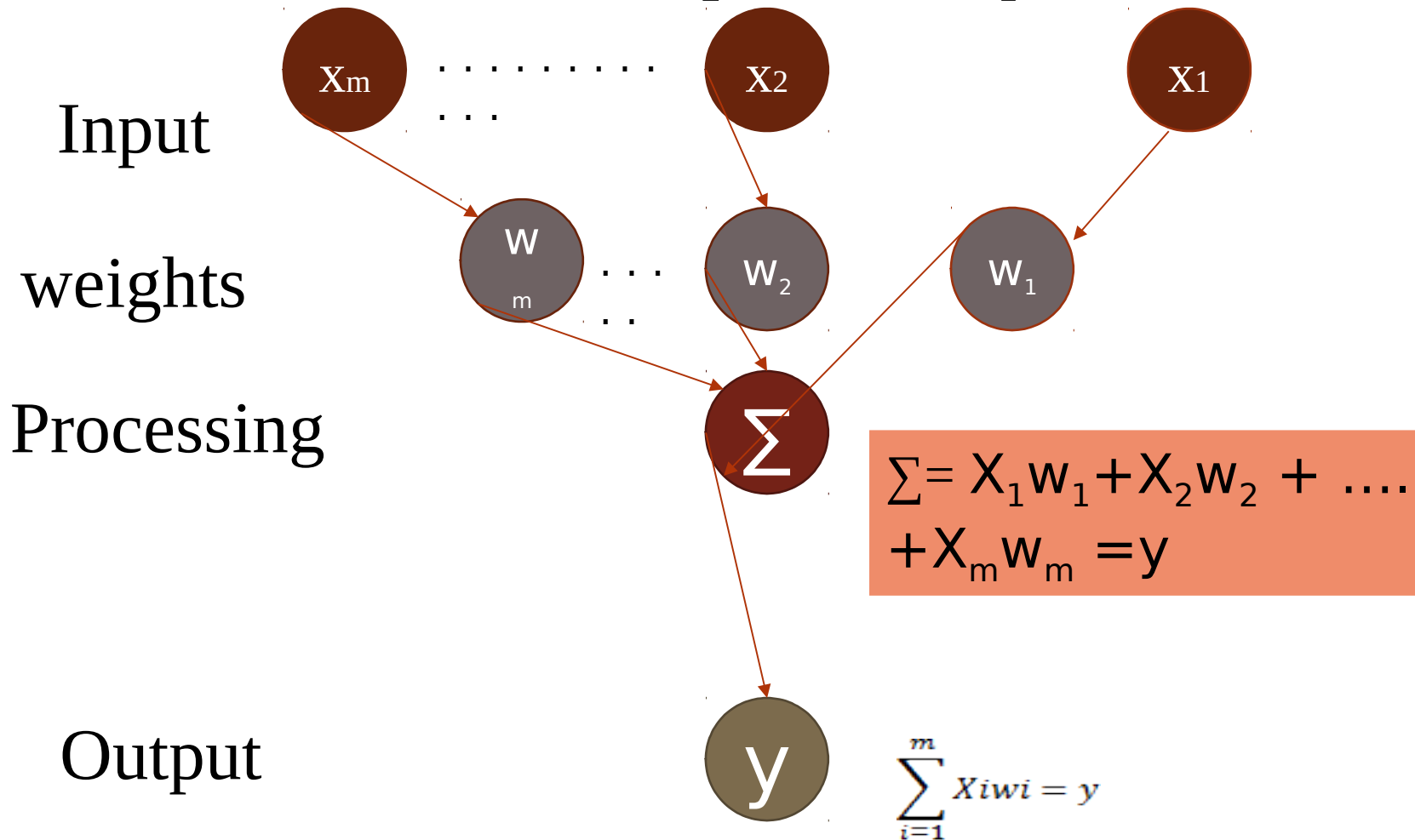


How do ANNs work?

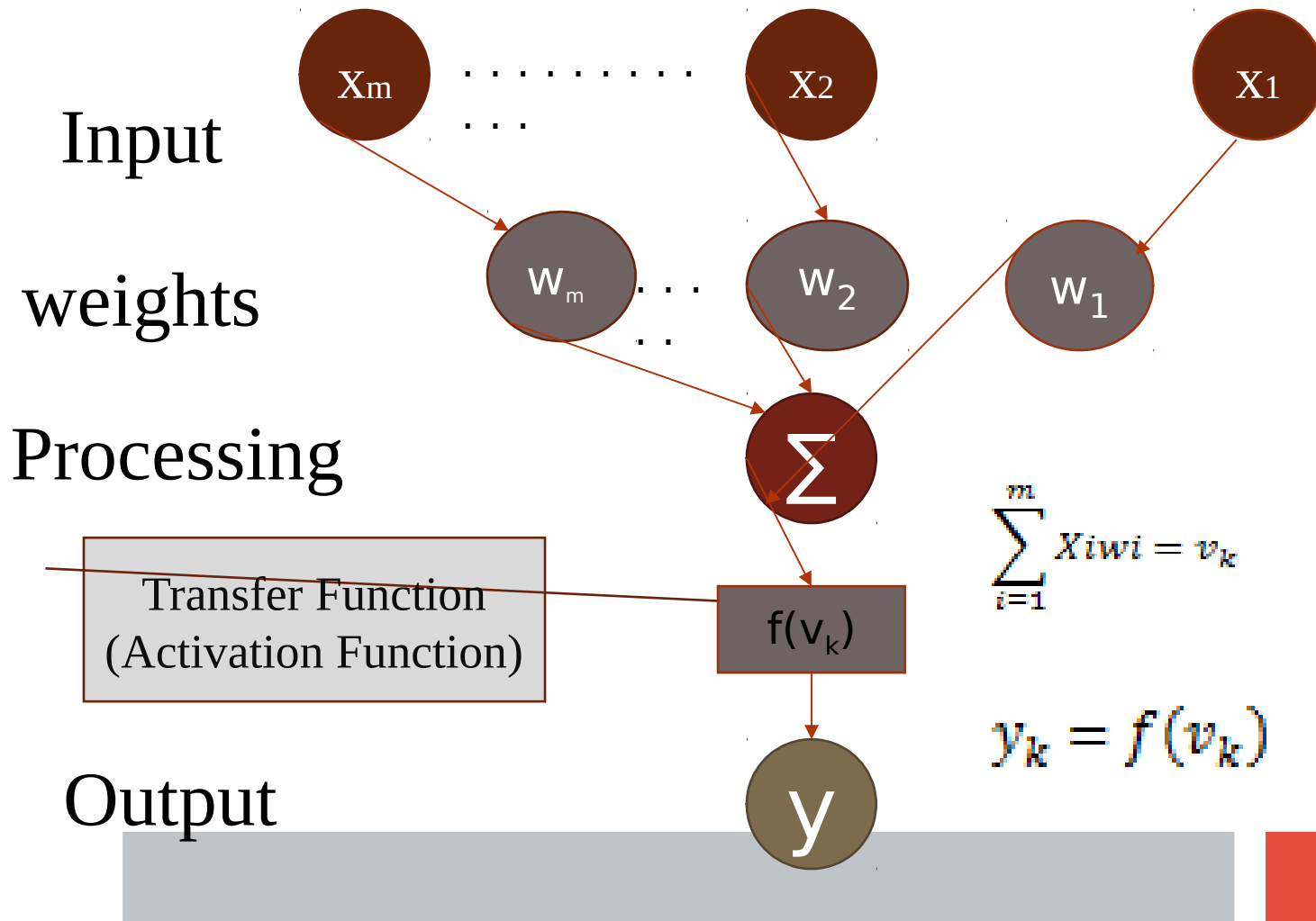


How do ANNs work?

Not all inputs are equal

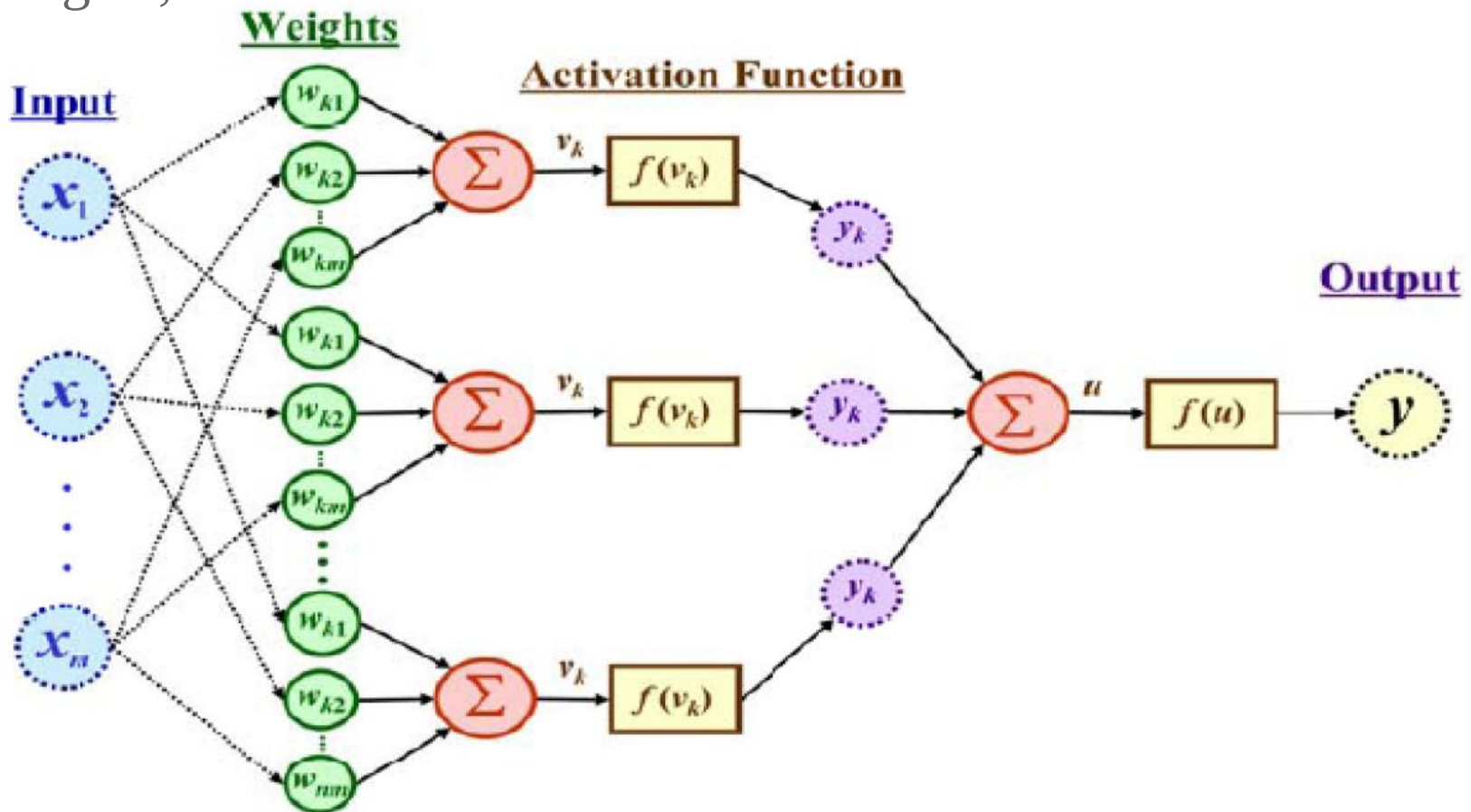


How do ANNs work?



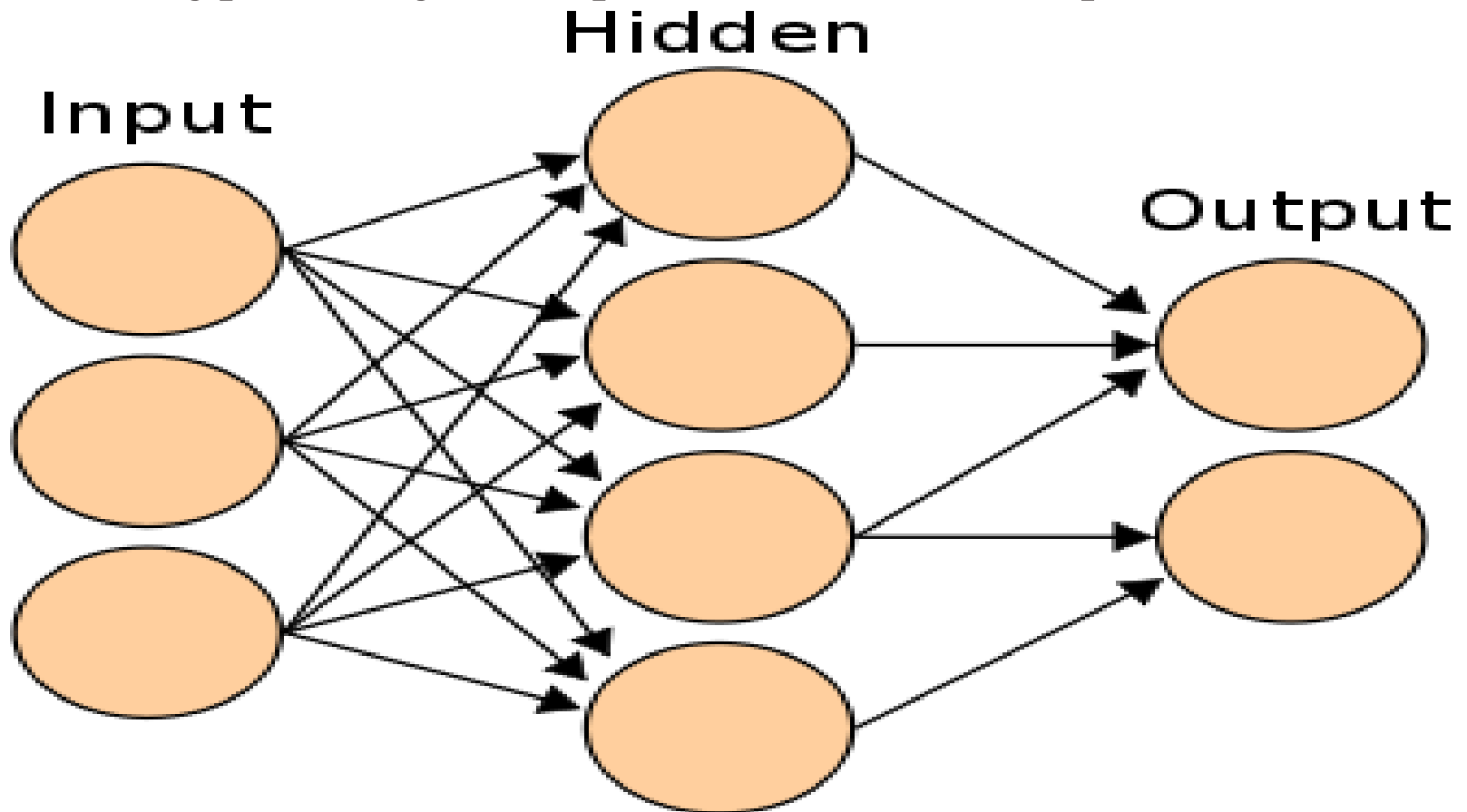
How do ANNs work?

The output is a function of the input, that is affected by the weights, and the transfer functions



How do ANNs work?

Three types of layers: Input, Hidden, and Output



Artificial Neural Networks

- An ANN can:
 1. compute *any computable* function, by the appropriate selection of the network topology and weights values.
 2. learn from experience!
 - Specifically, by trial and error

Learning by trial and error

Continuous process of:

➤ Trial:

Processing an input to produce an output (In terms of ANN:
Compute the output function of a given input)

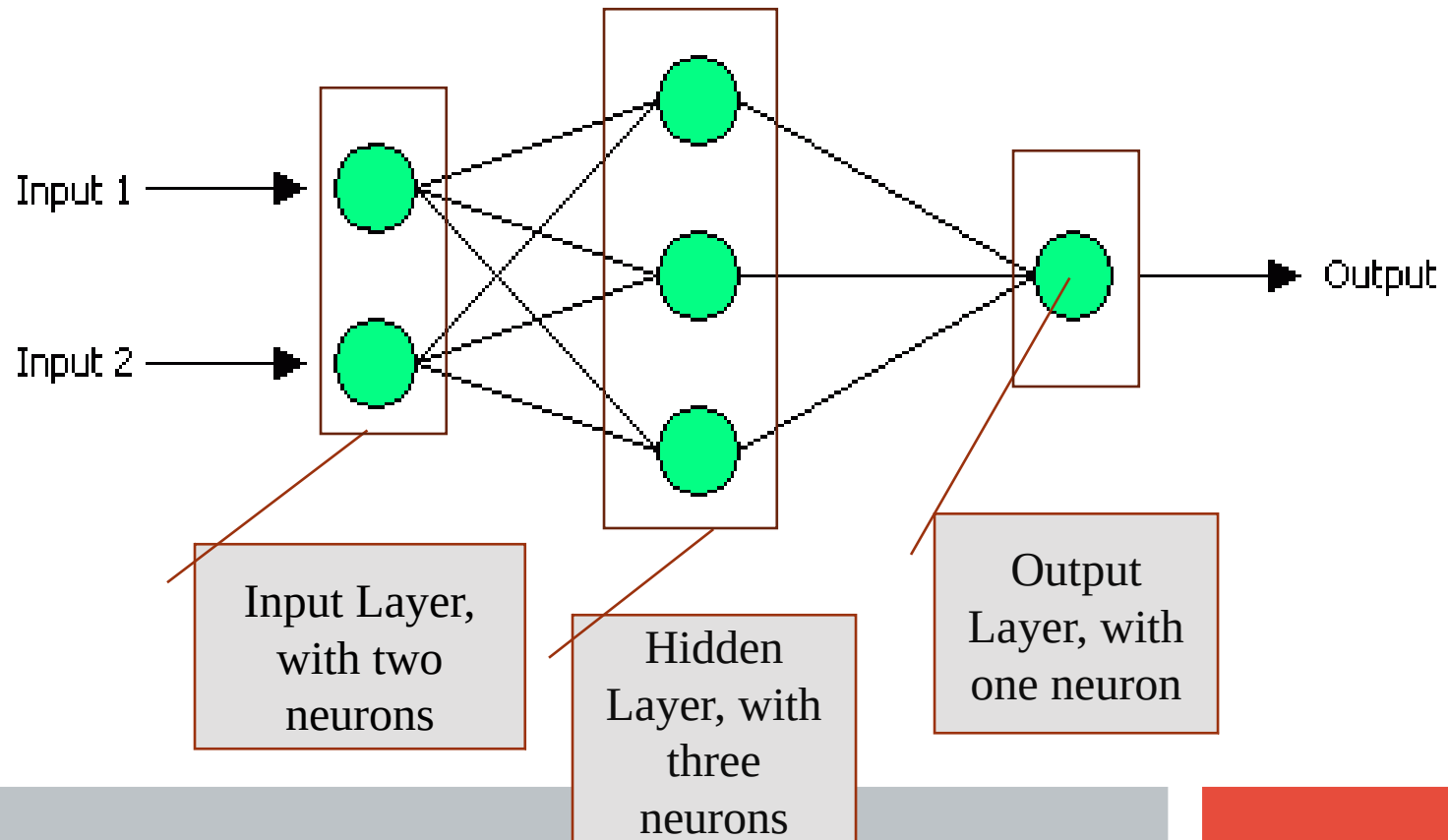
➤ Evaluate:

Evaluating this output by comparing the actual output with the expected output.

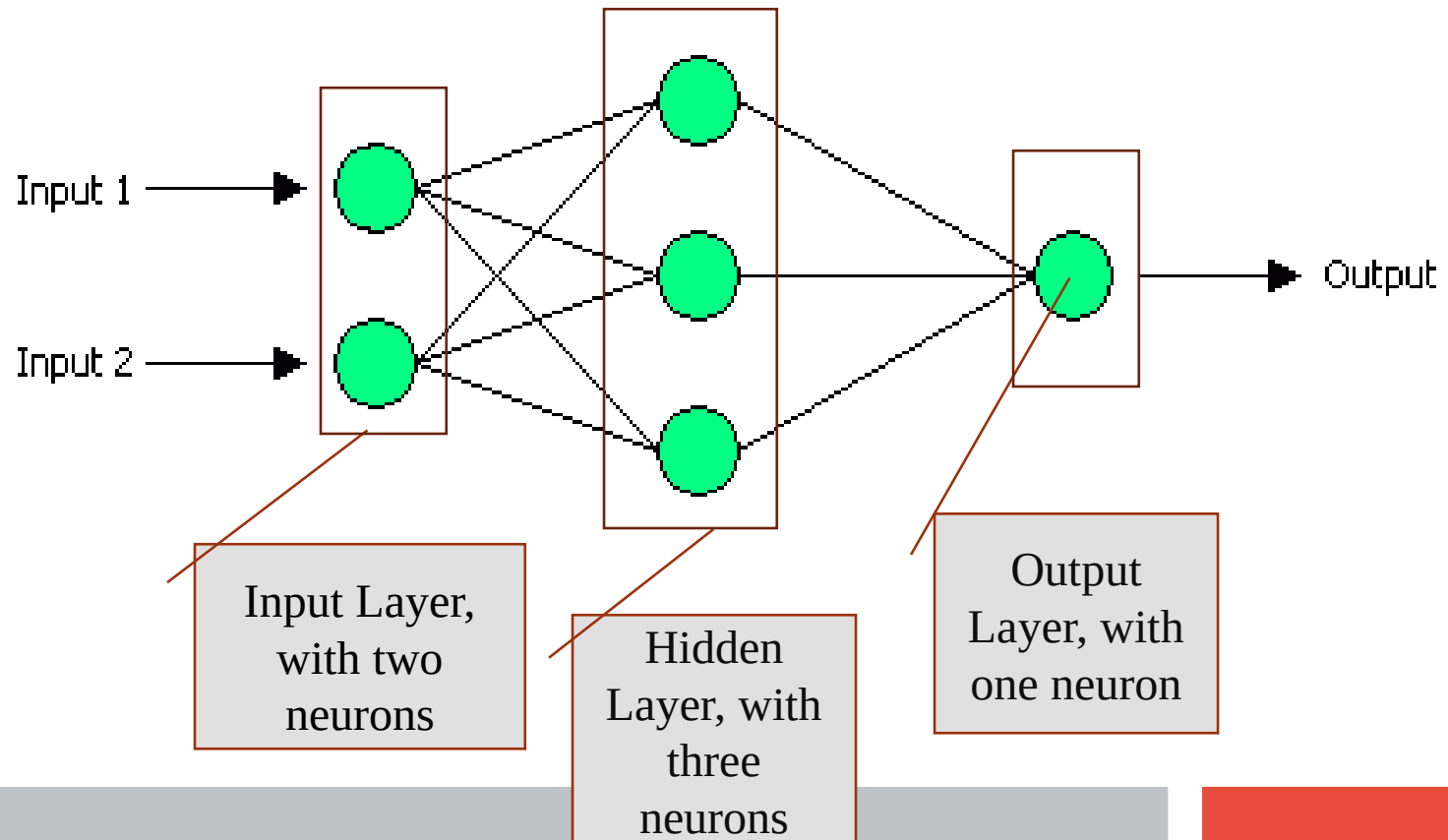
➤ Adjust:

Adjust the *weights*.

Example: XOR



How it works?



How it works?

- Set initial values of the weights randomly.
- Input: truth table of the XOR
- Do
 - Read input (e.g. 0, and 0)
 - Compute an output (e.g. 0.60543)
 - Compare it to the expected output. (Diff= 0.60543)
 - Modify the weights *accordingly*.
- Loop until a condition is met
 - Condition: certain number of iterations
 - Condition: error threshold

Design Issues

- Initial weights (small random values $\in [-1,1]$)
- Transfer function (How the inputs and the weights are combined to produce output?)
- Error estimation
- Weights adjusting
- Number of neurons
- Data representation
- Size of training set

Transfer Functions

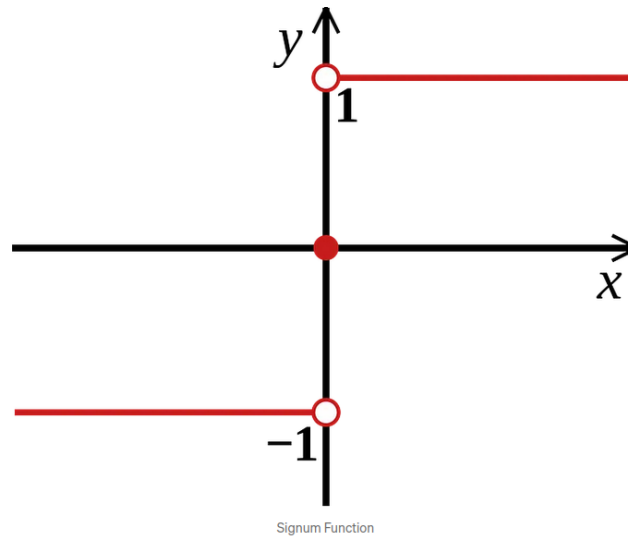
- **Linear:** The output is proportional to the total weighted input.
- **Threshold:** The output is set at one of two values, depending on whether the total weighted input is greater than or less than some threshold value.
- **Non linear:** The output varies continuously but not linearly as the input changes.

Activation Functions

- Sign Function
- Sigmoid Function
- Tanh (Hyperbolic Tangent)
- ReLU (Rectified Linear Unit)
- Leaky ReLU
- Hard Tanh
- Softmax Function

Activation Functions

- Sign Function
- If value is greater than 0 then it will give +1, if value is less than 0 then it will give -1 and if value is equal to 0 then it will also give 0 as an output.



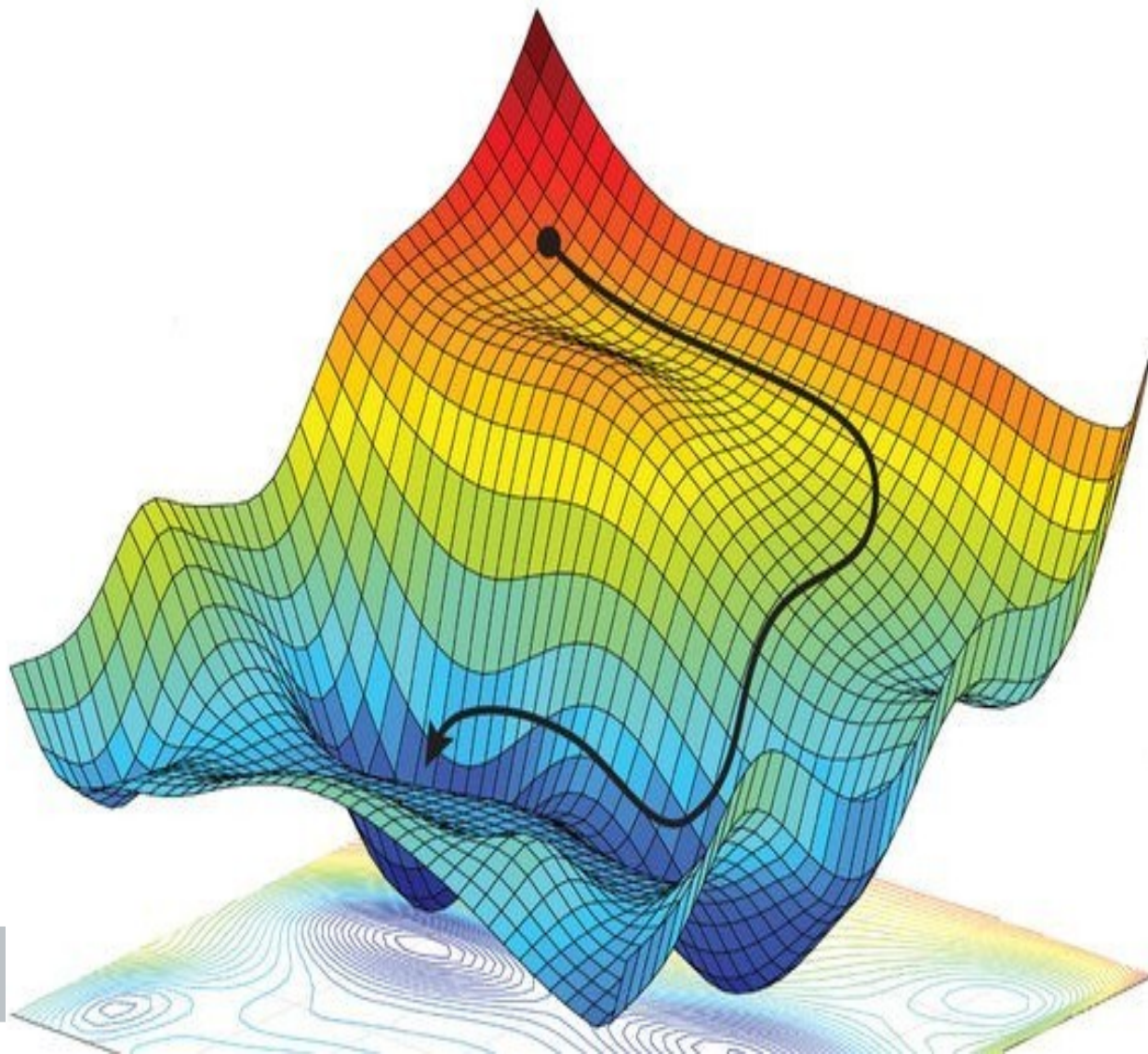
Activation Functions

- Sign Function
- Sigmoid Function
- Tanh (Hyperbolic Tangent)
- ReLU (Rectified Linear Unit)
- Leaky ReLU
- Hard Tanh
- Softmax Function

Transfer Functions

- **Linear:** The output is proportional to the total weighted input.
- **Threshold:** The output is set at one of two values, depending on whether the total weighted input is greater than or less than some threshold value.
- **Non linear:** The output varies continuously but not linearly as the input changes.

Loss functions



Loss Functions

A loss function measures how good a neural network model is in performing a certain task, which in most cases is regression or classification.

We must minimize the value of the loss function during the backpropagation step in order to make the neural network better.

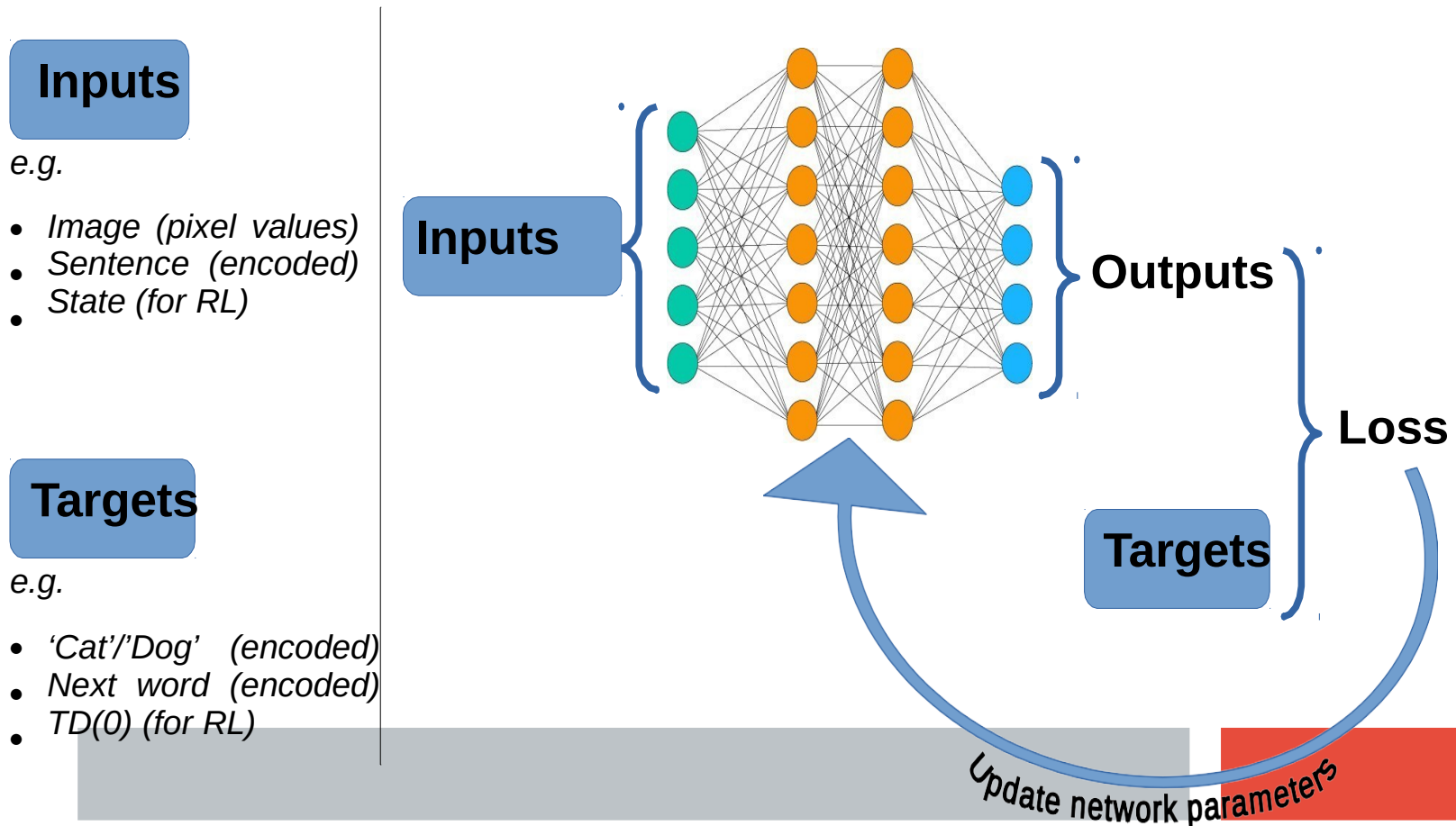
Loss functions

How good is your neural network?



Loss functions

How good is your neural network?



Loss functions

Some pedantry

Loss function and Cost function are often used interchangeably but they are different:

*The **loss function** measures the network performance **on a single datapoint***

*The **cost function** is the **average of the losses** over the entire training dataset*

*Our goal is to **minimize the cost function**.*

*In reality, we actually use **batch losses** as a proxy for the cost function.*

Loss/Cost functions

Some stuff to remember

*The cost function distils the performance of the network down to a **single scalar number**. Generally (although not necessarily) **loss** ≥ 0 (so **cost** ≥ 0 also)*

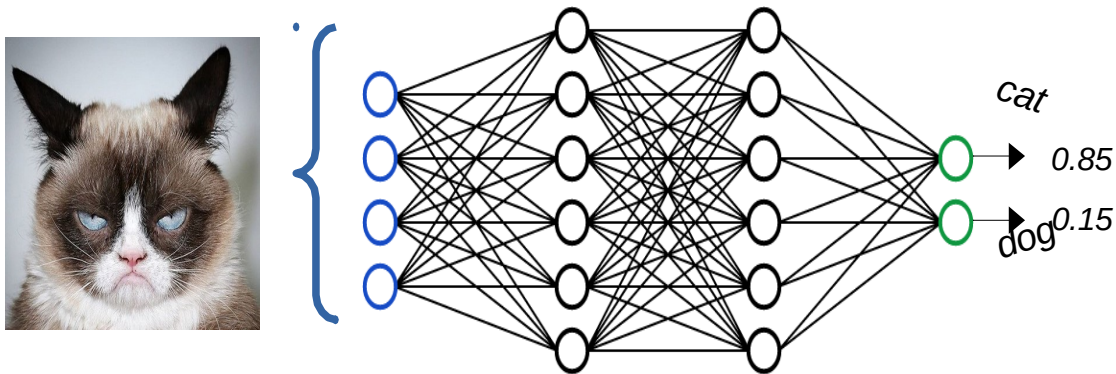
cost=0 \Rightarrow perfect performance (on the given dataset)

*The **lower the value of the cost function**, the **better the performance** of the network.*

Hence gradient descent

*A cost function must **faithfully represent the purpose** of the network.*

Loss/Cost functions



Accuracy: 1
Loss (CCE): 0.07

Loss/Cost functions

Loss functions

- **Classification**
 - *Maximum likelihood*
 - *Binary cross-entropy (aka log loss)*
 - *Categorical cross-entropy*
- **Regression** (i.e. function approximation)
 - *Mean Squared Error*
 - *Mean Absolute Error*
 - *Huber Loss*

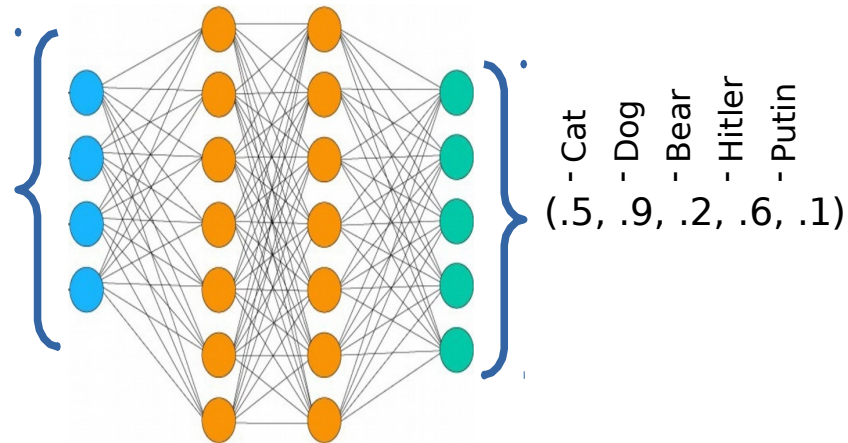
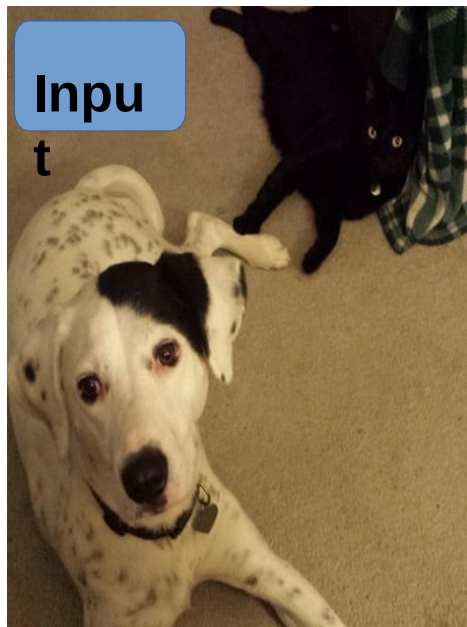
Classification loss functions

Loss functions

- **Classification**
 - Maximum likelihood*
 - Binary cross-entropy (aka log loss)*
 - Categorical cross-entropy*

Classification loss functions

Maximum likelihood

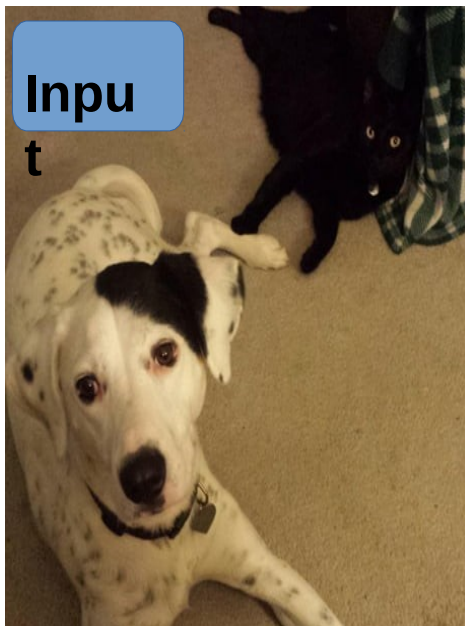


Target (1, 1, 0, 0, 0)

Cat -
Dog -
Bear -
Hitler -
Putin -

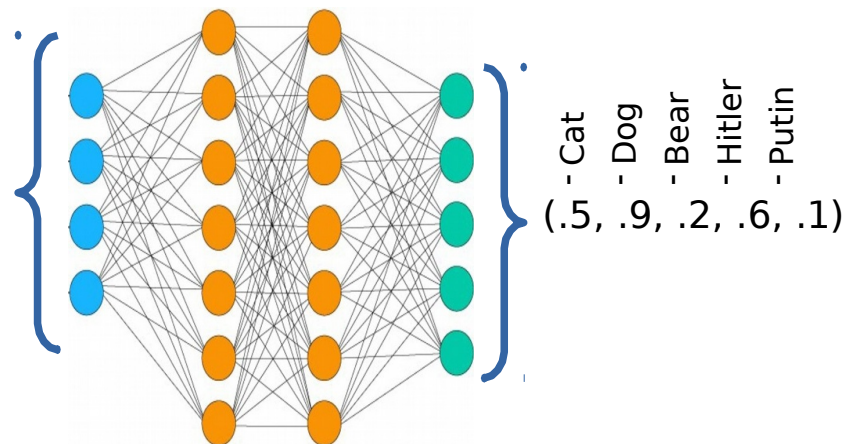
Classification loss functions

Maximum likelihood



Input

t



$$Loss = -\frac{1}{N} \sum_{i=1}^N y_i \cdot (p(y_i)) + (1 - y_i) \cdot (1 - p(y_i))$$

Target

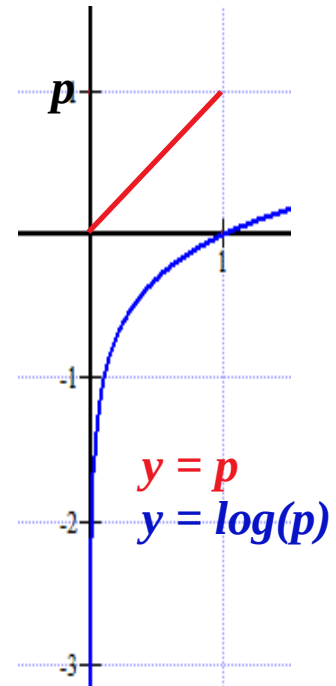
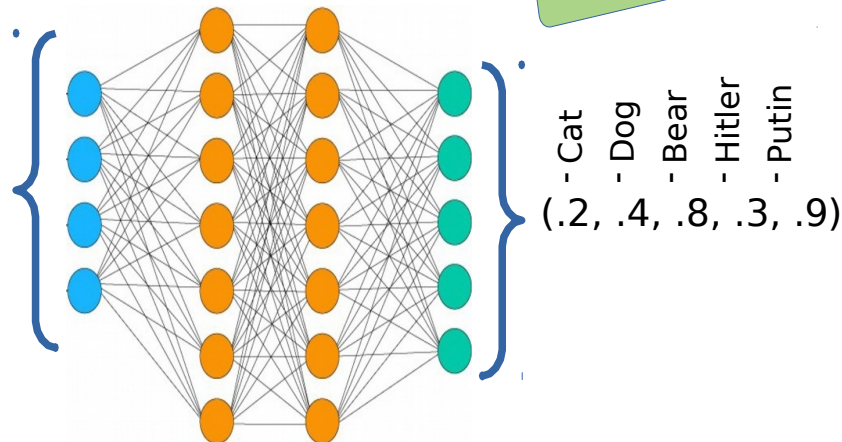
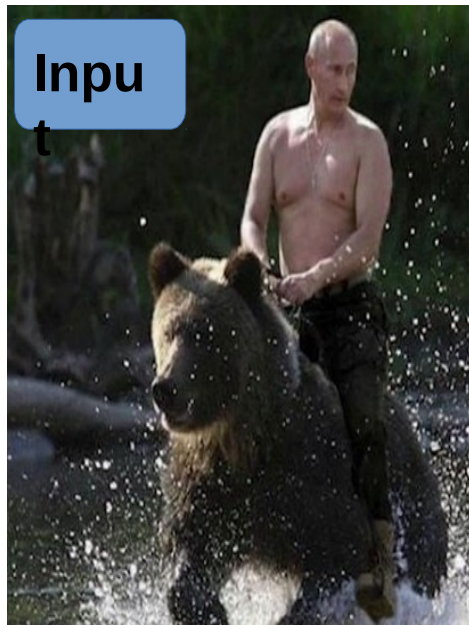
(1, 1, 0, 0, 0)

Cat
 Dog
 Bear
 Hitler
 Putin

$$\begin{aligned}
 &= -1/5 [(1*0.5 + 0*0.5) + (1*0.9 + 0*0.1) + (0*0.2 + 1*0.8) \\
 &\quad + (0*0.6 + 1*0.4) + (0*0.1 + 1*0.9)] \\
 &= -0.7
 \end{aligned}$$

Classification loss functions

Binary cross-entropy (aka Log loss)



$$\text{Loss} = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

$$= -1/5 [(1 \cdot \log(0.8)) + (1 \cdot \log(0.6)) + (1 \cdot \log(0.8)) + (1 \cdot \log(0.7)) + (1 \cdot \log(0.9))]$$

$$= 0.28$$

Target

t

(0, 0, 1, 0, 1)

Cat
Dog
Bear
Hitler
Putin

Classification loss functions

Categorical cross-entropy

Torch.nn.NLLLoss

Torch.nn.CrossEntropyLoss

This one does the softmax for you

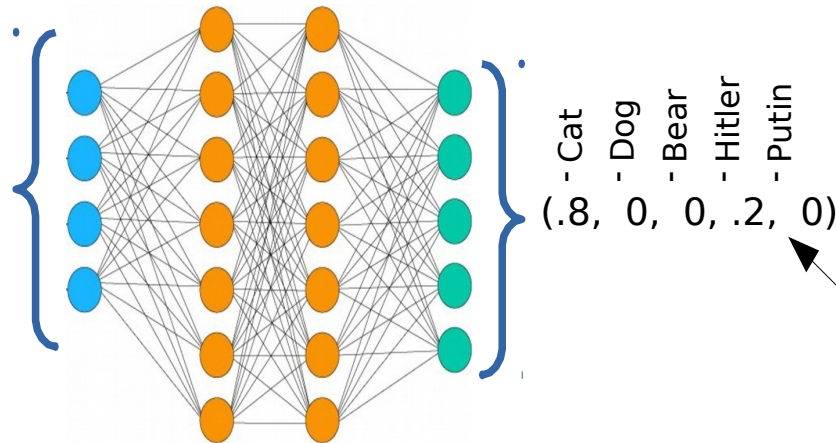
$$= \log \left(\frac{\exp(x_i)}{\sum_j \exp(x_j)} \right)$$

Must be a probability distribution

This loss function can also be used when you have a single binary (e.g. yes/no) output



Input



$$\text{Loss} = - \sum_{i=1}^N y_i \cdot \log(p(y_i))$$

$$= - [1 * \log(0.8)]$$

$$= 0.22$$

Target

(1, 0, 0, 0, 0)

Cat
Dog
Bear
Hitler
Putin

Regression loss functions

Loss functions

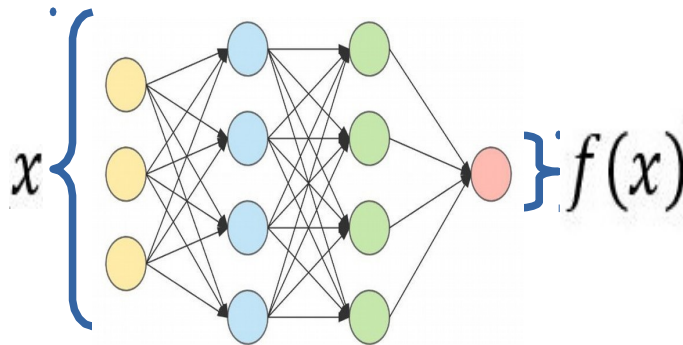
- **Regression** (i.e. function approximation)
 - Mean Squared Error Mean
 - Absolute Error Huber Loss
 -

Regression loss functions

Mean squared error

Torch.nn.MSELoss

Input x



Target y

$$Loss = (y - f(x))^2$$

Regression loss functions

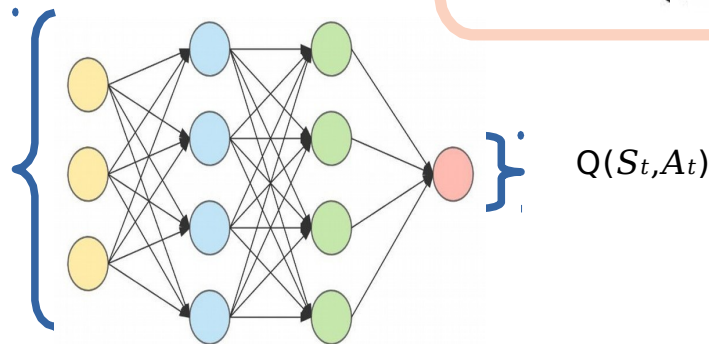
Mean squared error

Torch.nn.MSELoss

$$\begin{aligned} \text{Loss} &= (y - f(x))^2 \\ &= \left((R_{t+1} + \gamma \max_a Q(S_{t+1}, a)) - Q(S_t, A_t) \right)^2 \end{aligned}$$

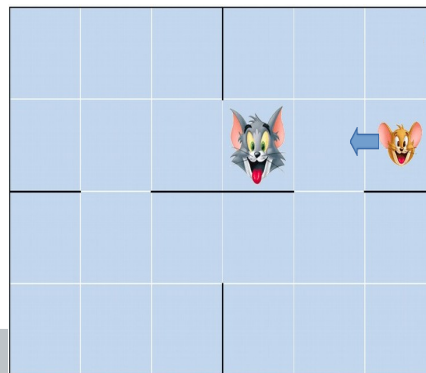


state S_t & action A_t



$Q(S_t, A_t)$

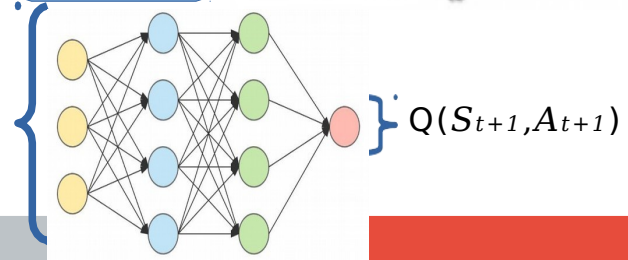
reward R_{t+1}



state S_{t+1} & [best] action A_{t+1}

Target

$$R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$$



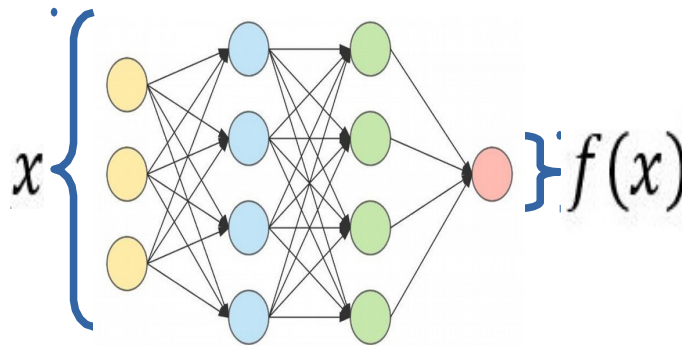
$Q(S_{t+1}, A_{t+1})$

Regression loss functions

Absolute error

Torch.nn.L1Loss

Input x



Target y

$$Loss = |y - f(x)|$$

Regression loss

Huber loss combines the best features of MSE and AE:

- like MSE for small errors.
- like AE otherwise.

This avoids over-shooting:

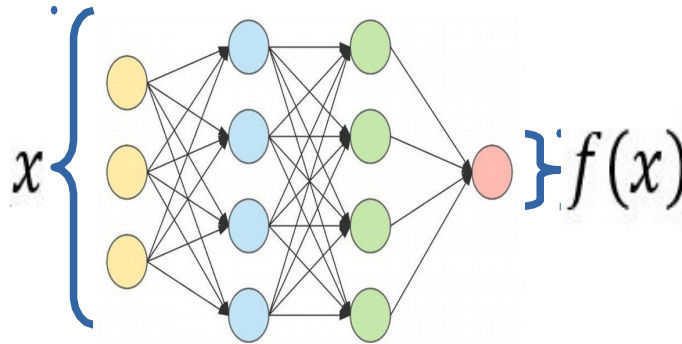
-MSE has a **very steep gradient** when the error is **large** (i.e. for outliers) because of its quadratic form.

- AE has a **steep gradient** when the error is **small** because it doesn't have a 'flat' bottom.

Huber loss

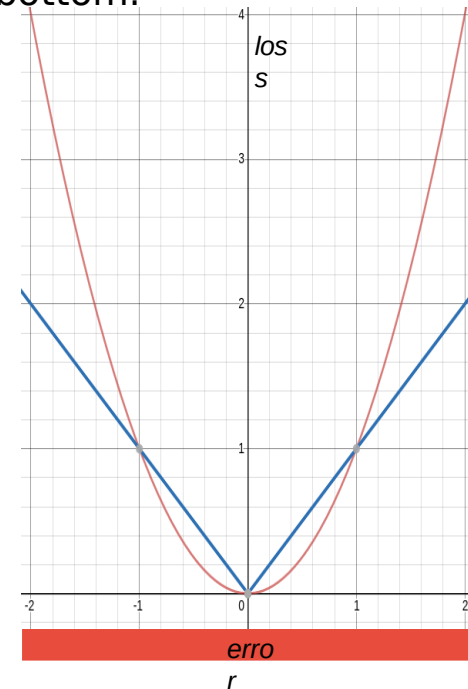
`Torch.nn.SmoothL1Loss`

Input x



Target y

$$L_{os} = \begin{cases} \frac{1}{2}(y - f(x))^2, & \text{if } |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases}$$



Weights Adjusting

- After each iteration, weights should be adjusted to minimize the error.
 - All possible weights
 - Back propagation

Back Propagation

- Back-propagation is an example of supervised learning is used at each layer to minimize the error between the layer's response and the actual data
- The error at each hidden layer is an average of the evaluated error
- Hidden layer networks are trained this way

Back Propagation

- N is a neuron.
- N_w is one of N's inputs weights
- N_{out} is N's output.
- $N_w = N_w + \Delta N_w$
- $\Delta N_w = N_{out} * (1 - N_{out}) * N_{ErrorFactor}$
- $N_{ErrorFactor} = N_{ExpectedOutput} - N_{ActualOutput}$
- This works only for the last layer, as we can know the actual output, and the expected output.

Number of neurons

- Many neurons:
 - Higher accuracy
 - Slower
 - Risk of over fitting
 - Memorizing, rather than understanding
 - The network will be useless with new problems.
- Few neurons:
 - Lower accuracy
 - Inability to learn at all
- Optimal number.

Data representation

- Usually input/output data needs pre processing
- Pictures
 - Pixel intensity
- Text:
 - A pattern

Size of training set

- No one fits all formula
- Over fitting can occur if a “good” training set is not chosen
- What constitutes a “good” training set?
 - Samples must represent the general population.
 - Samples must contain members of each class.
 - Samples in each class must contain a wide range of variations or noise effect.
- The size of the training set is related to the number of hidden neurons

Learning Paradigms

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised learning

- This is what we have seen so far!
- A network is fed with a set of training samples (inputs and corresponding output), and it uses these samples to learn the general relationship between the inputs and the outputs.
- This relationship is represented by the values of the weights of the trained network.

Unsupervised learning

- No desired output is associated with the training data!
- Faster than supervised learning
- Used to find out *structures within data*:
 - Clustering
 - Compression

Reinforcement learning

- Like supervised learning, but:
 - Weights adjusting is not directly related to the error value.
 - The error value is used to randomly, shuffle weights!
 - Relatively slow learning due to ‘randomness’.

Applications Areas

- Function approximation
 - including time series prediction and modeling.
- Classification
 - including patterns and sequences recognition, novelty detection and sequential decision making.
 - (radar systems, face identification, handwritten text recognition)
- Data processing
 - including filtering, clustering blinds source separation and compression.
 - (data mining, e-mail Spam filtering)

Advantages / Disadvantages

- Advantages
 - Adapt to unknown situations
 - Powerful, it can model complex functions.
 - Ease of use, learns by example, and very little user domain specific expertise needed
- Disadvantages
 - Forgets
 - Not exact
 - Large complexity of the network structure

Conclusion

- Artificial Neural Networks are an imitation of the biological neural networks, but much simpler ones.
- The computing would have a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful furthermore there is need to device an algorithm in order to perform a specific task.