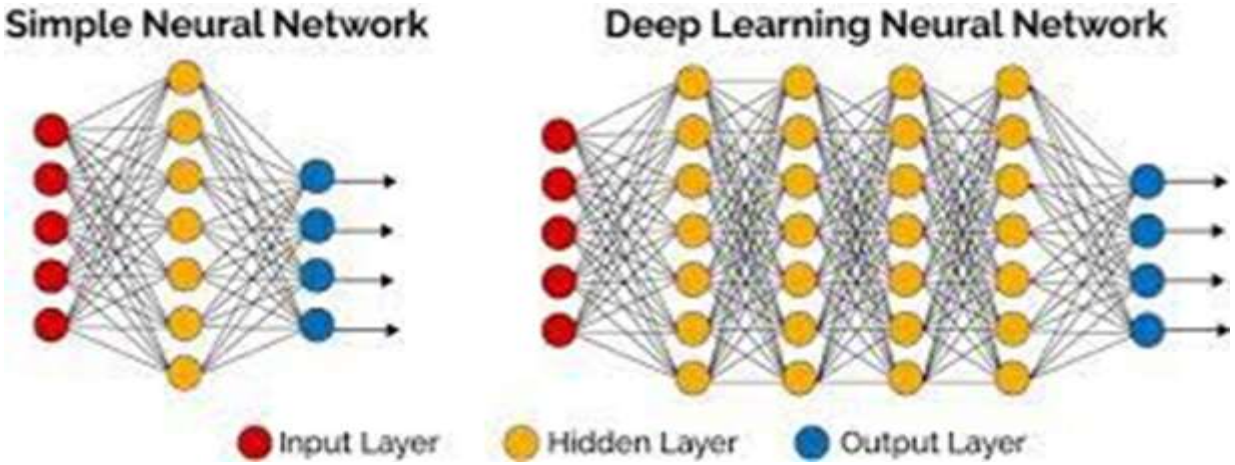
Elements of Neural Network & Deep Learning

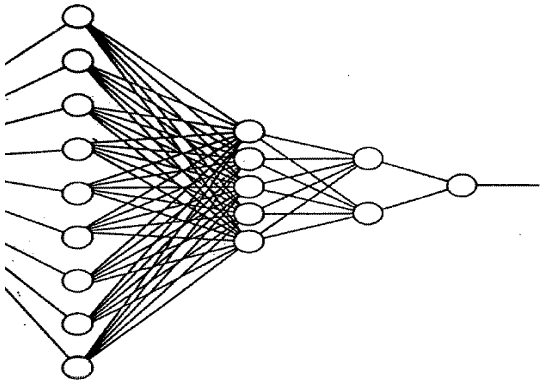


Neural Network

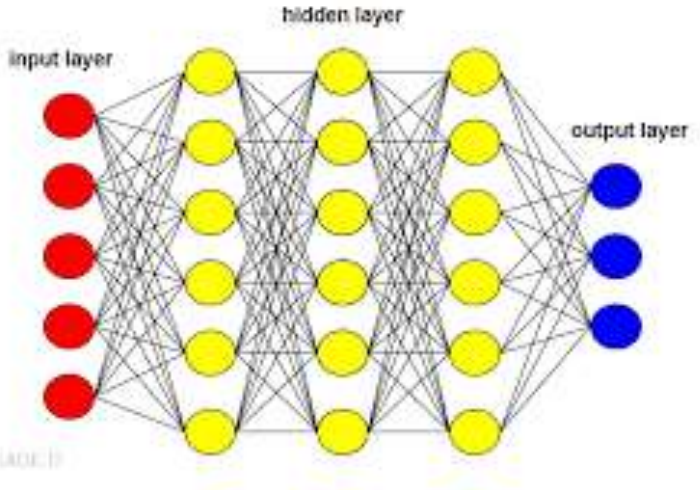
- Connectionism is a cognitive science approach that hopes to explain the functioning of the mind using artificial neural networks
- Neural networks simulate the behavior of the brain at a subsymbolic level by modeling the single neuron (in a simplified way) and a network of interconnected neurons.
- In an artificial neural network, each "node" represents the oversimplification of a biological neuron.
- It attempts to simulate the behavior of neurons with connections similar to the synapses of a biological neuron and via a activation function, which determines when the neuron sends a signal.

Aritficial Neural Network

IN



OUT

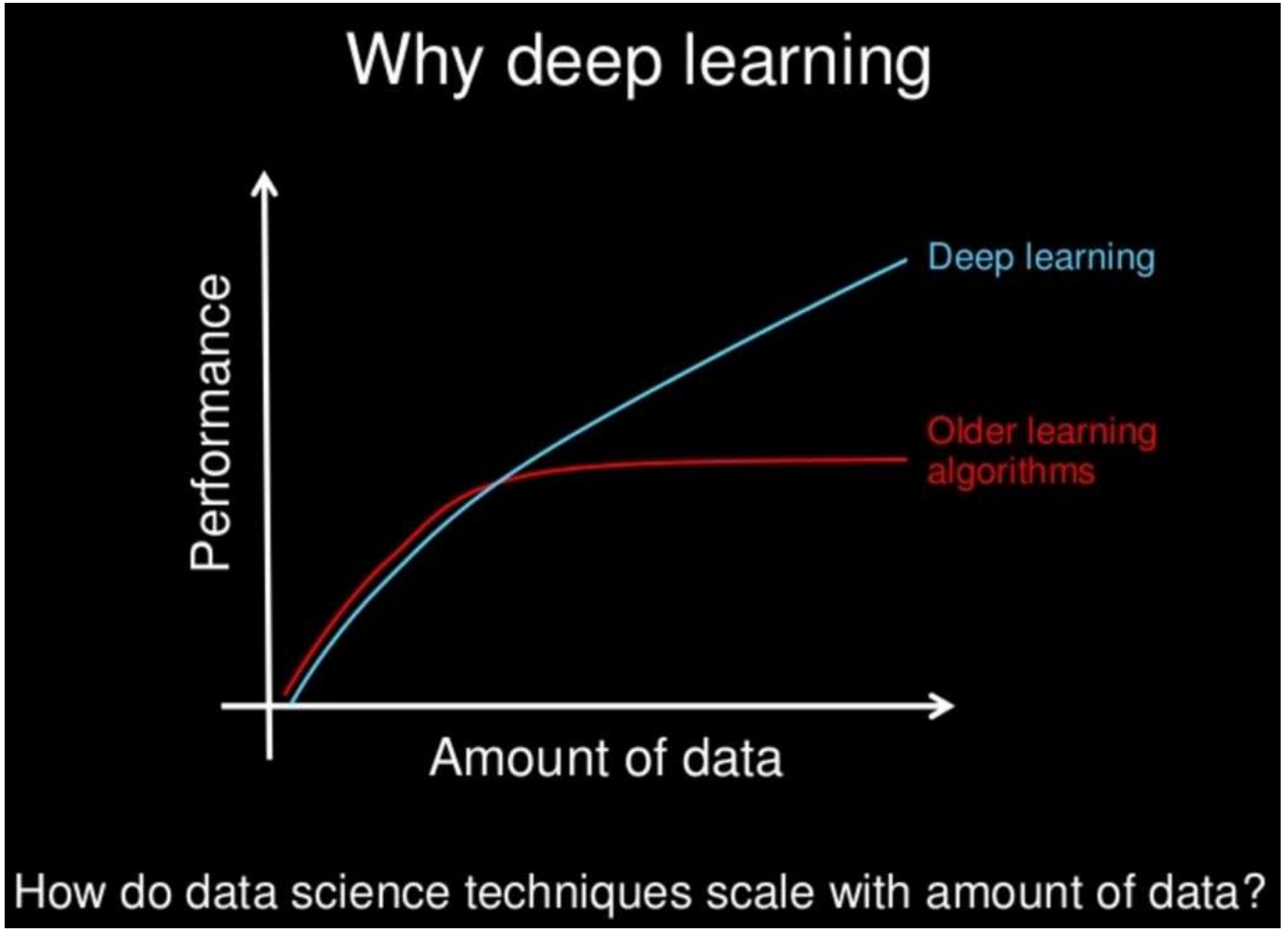


By modifying the weights associated with the connections between neurons it is possible to modify the behavior of the network.

The network, if large enough, can approximate any function.

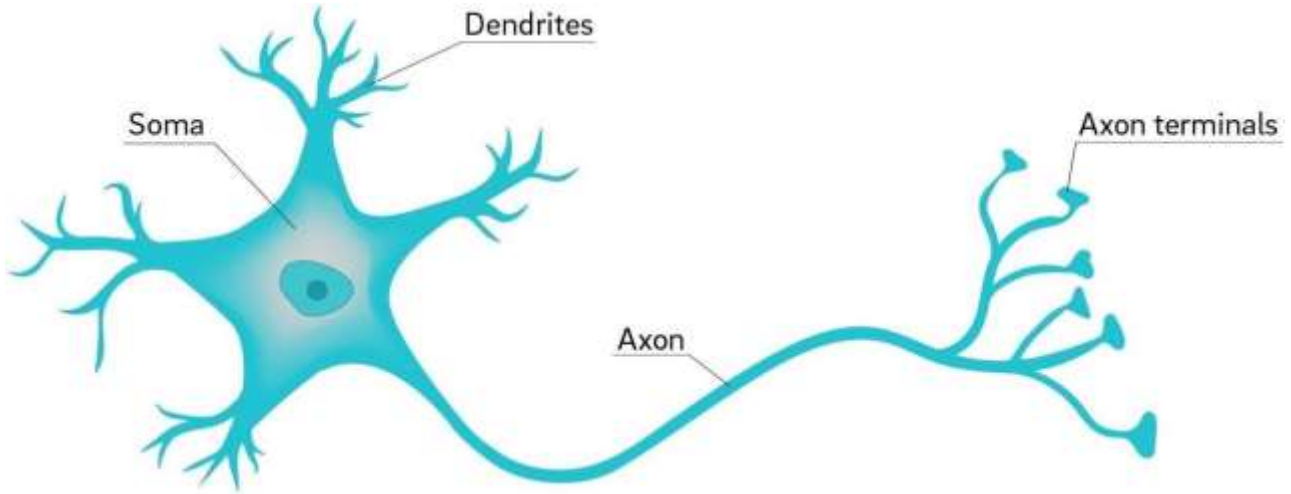
Given a training set in which the desired output is specified for each data (example), I can replicate the behavior described by the data with the network

NN & Deep Learning ?



Biological neural networks

- Network of neurons (about 10^{11} in humans)
- Each neuron receives impulses from dendrites
- Soma is excited from these impulses and it propagates a new electric signal through the axon to other neurons



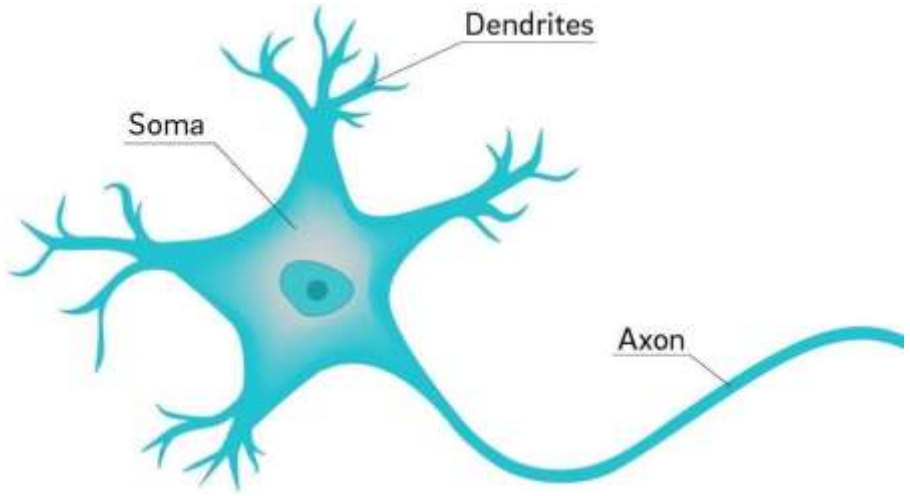
Artificial Neural Network (ANN)

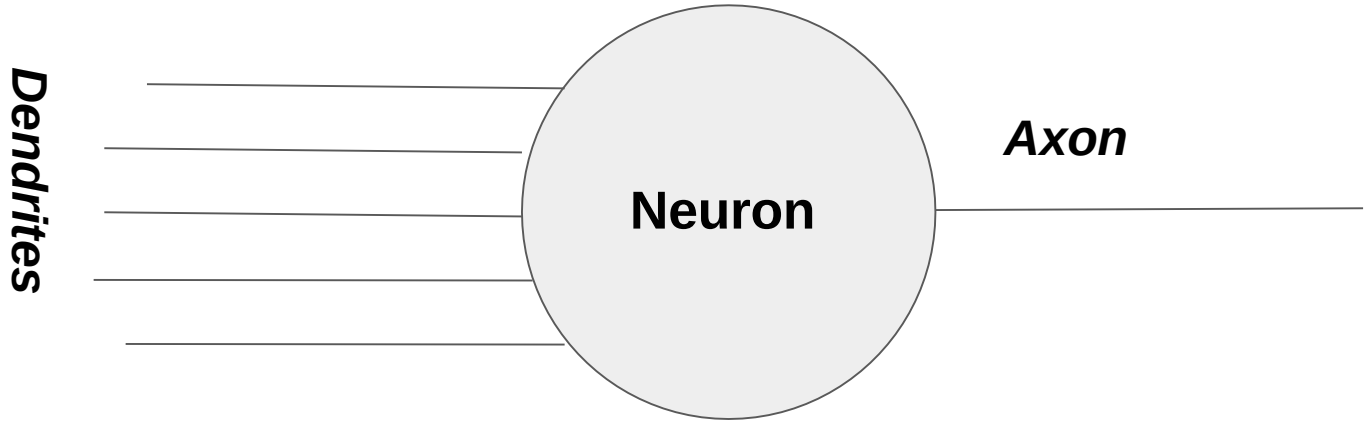
- Artificial Neural Network(ANN): Computational paradigm inspired by a mathematical model of the neuron (McCulloch & Pitts 1943) devised to study the computational abilities of biological neurons and neural networks.

- It takes inspiration from the architecture of human brain for building an intelligent machine.

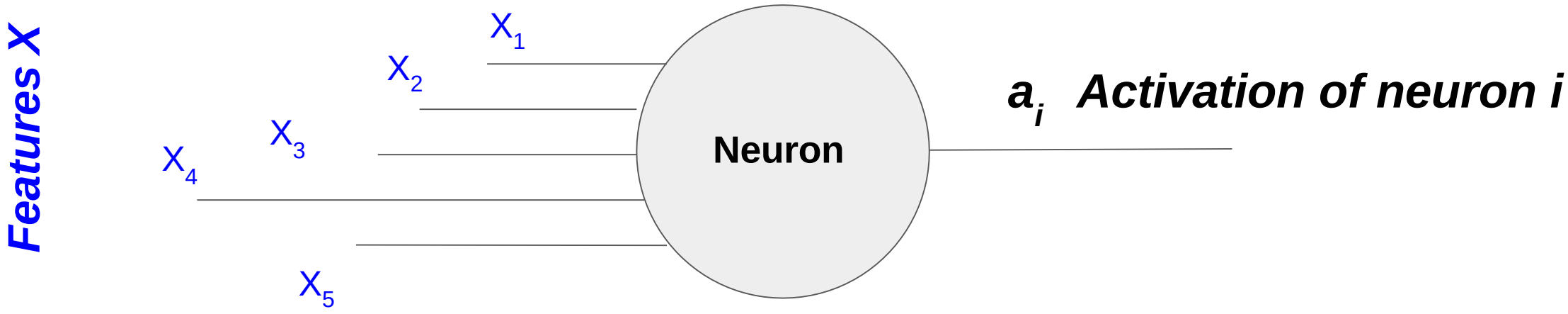
- Network of nodes (artificial neurons)

Towards the Artificial Neuron



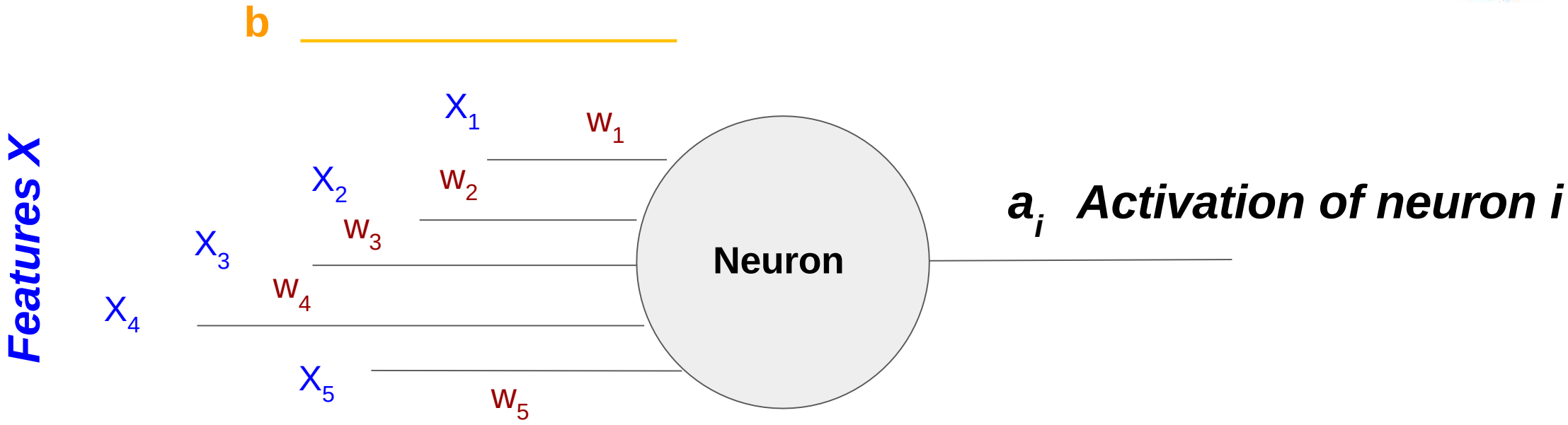


Towards the Artificial Neuron



- To simulate the biological neuron's behavior we should collect and accumulate the input from the "dendrites". We can do a weighted sum of the inputs by associating a weight ***w*** to each connection

Towards the Artificial Neuron



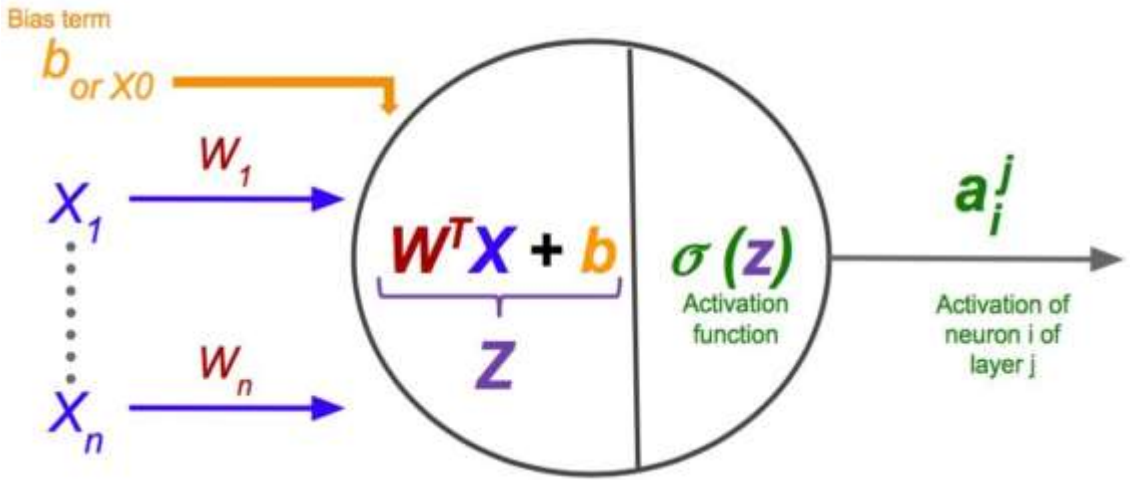
- So we have: $w_1x_1 + w_2 x_2 + w_3 x_3 + w_4x_4 + w_5 x_5$
- This weighted sum can be written as $W^T X$
- Let's add another input **b** called "bias" to play this role. Usually b=1
- $W^T X + b$

Towards the Artificial Neuron

- What's new? *Until now we are practically speaking about a Linear Regression*
- Indeed, a neuron to accumulate stimuli from dendrites (input features X) exploits a linear model
- After that, we have to decide what is the output of the neuron along the axon
    - We can apply a linear or non-linear function to the result of the linear model to define the output of the neuron (The Activation)

Artificial Neuron

- An artificial neuron is a function that maps an *input vector* $\{x_1, \ldots, x_k\}$ to a *scalar output* y via a *weight vector* $\{w_1, \ldots, w_k\}$ and a *function* f (typically non-linear).
- Where the input vector represents the dendrites
- The output scalar value represents the activation of the neuron and the signal propagated over the axon
- Neuron receives all the stimuli (vector X), it computes a weighted sum and then applies an activation function that defines a threshold to define the output value



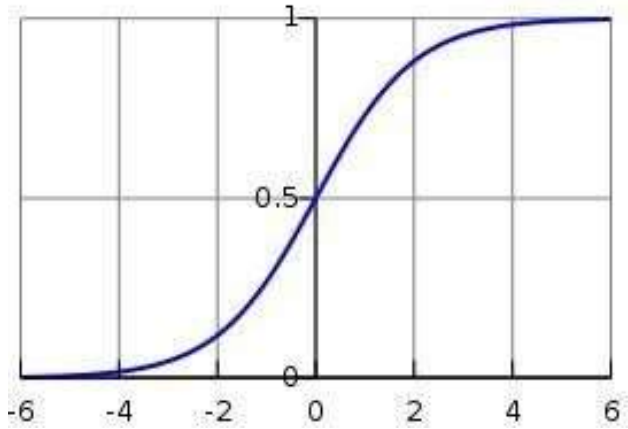
$$a_i^j = f(\sum_{i=0}^{K} w_i x_i) = f(\mathbf{w}^\mathbf{T}\mathbf{x})$$

Activation function

- The *function f* is called the **activation function** and generates a non-linear input/output relationship.
- A common choice for the activation function is the **Logistic function** (or **Sigmoid**).

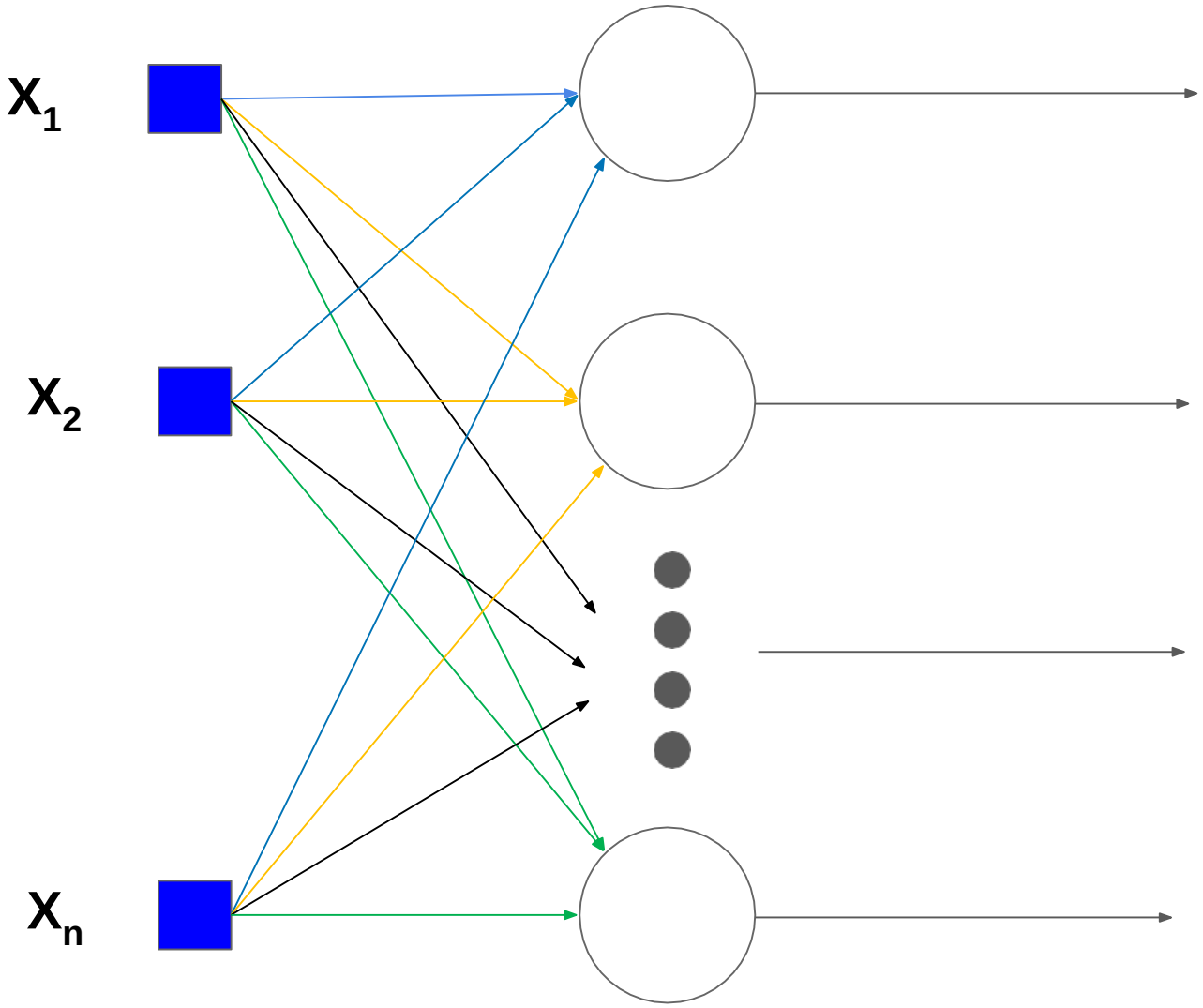
Sigmoid

$$f(\mathbf{w}^T\mathbf{x}) \rightarrow y = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$



- In this way our neuron will have an output between 0 and 1
  - *Practically is a Logistic Regression*
- But we will see that is not the only option we have
- In general Neuron := Linear + activation

More neurons: A layer



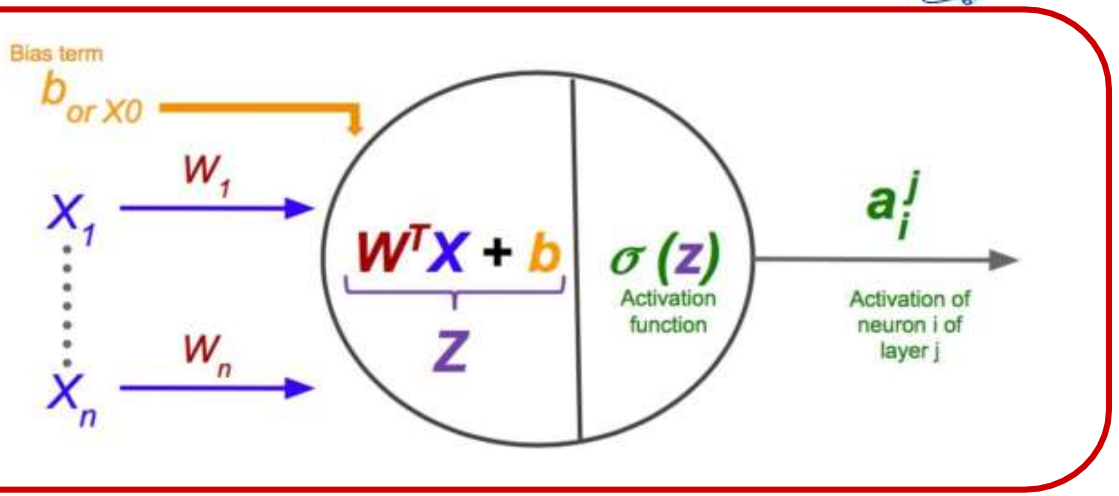
$$a_1 = \sigma(W_1^T X + b_1)$$

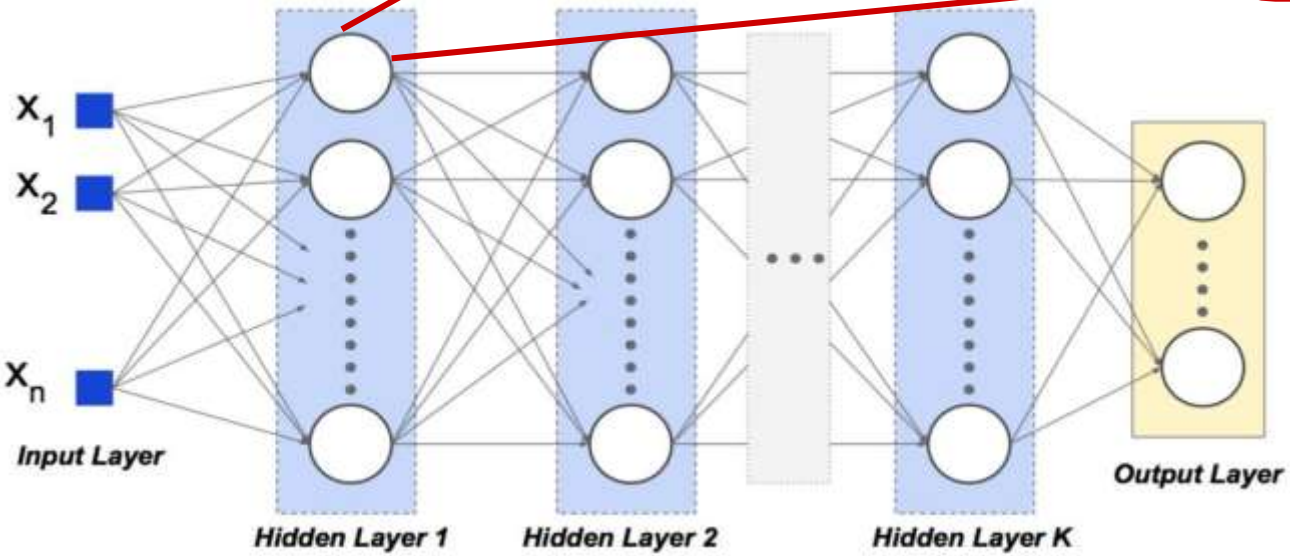
$$a_2 = \sigma(W_2^T X + b_2)$$

$$a_n = \sigma(W_n^T X + b_3)$$

Artificial Neural Network (ANN)

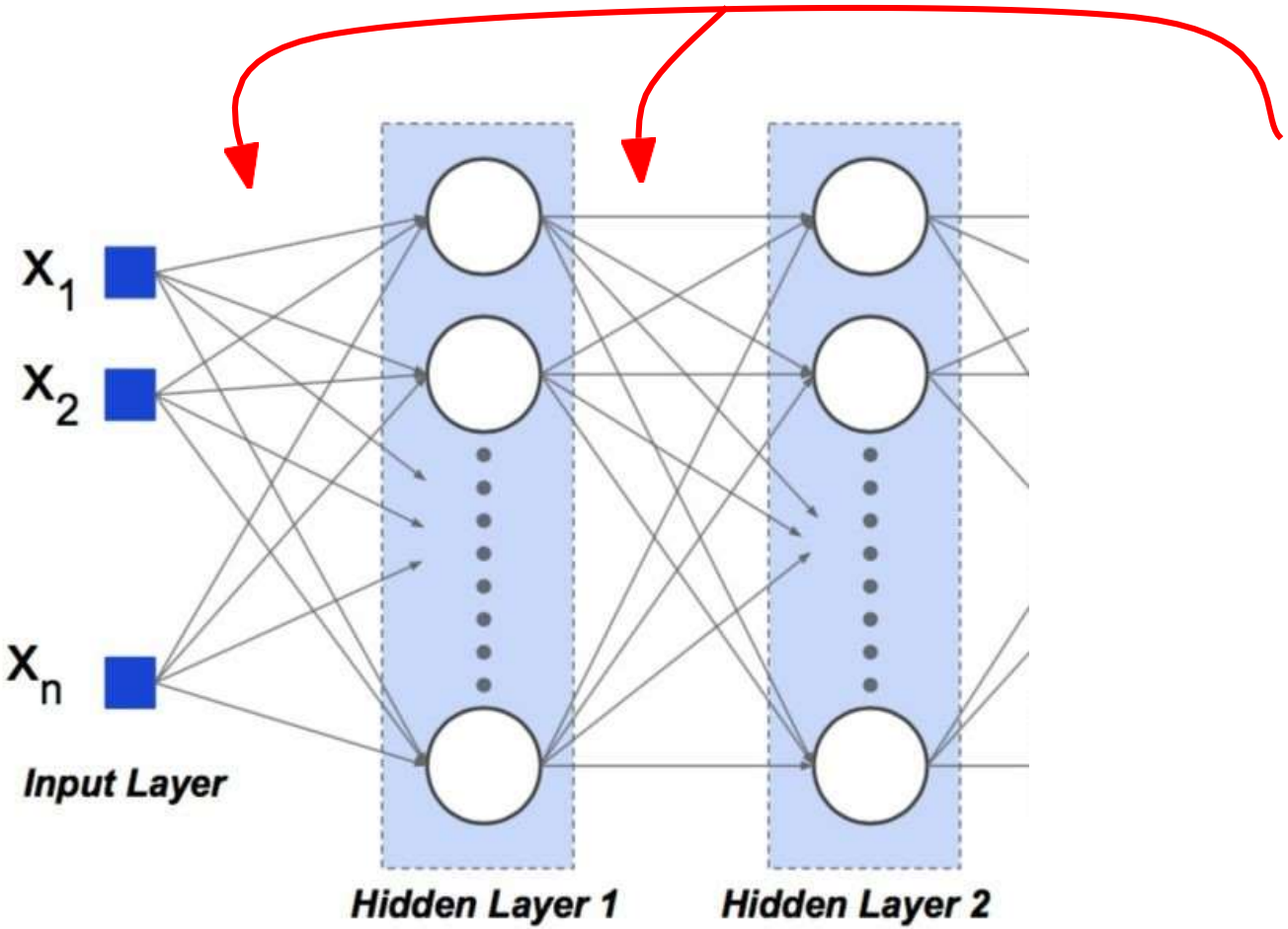
- Neurons in the same layer don't have to communicate with each other





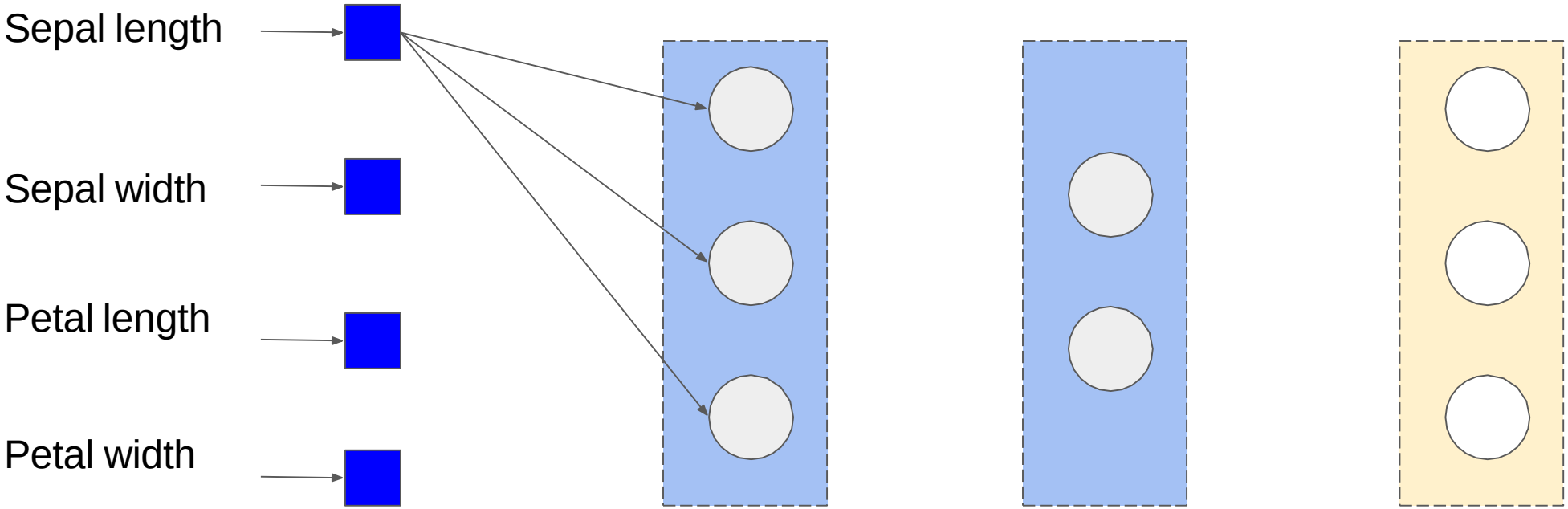
- Each neuron at layer j -1 is connected with all the neurons in the next layer j

Artificial Neural Network (ANN)



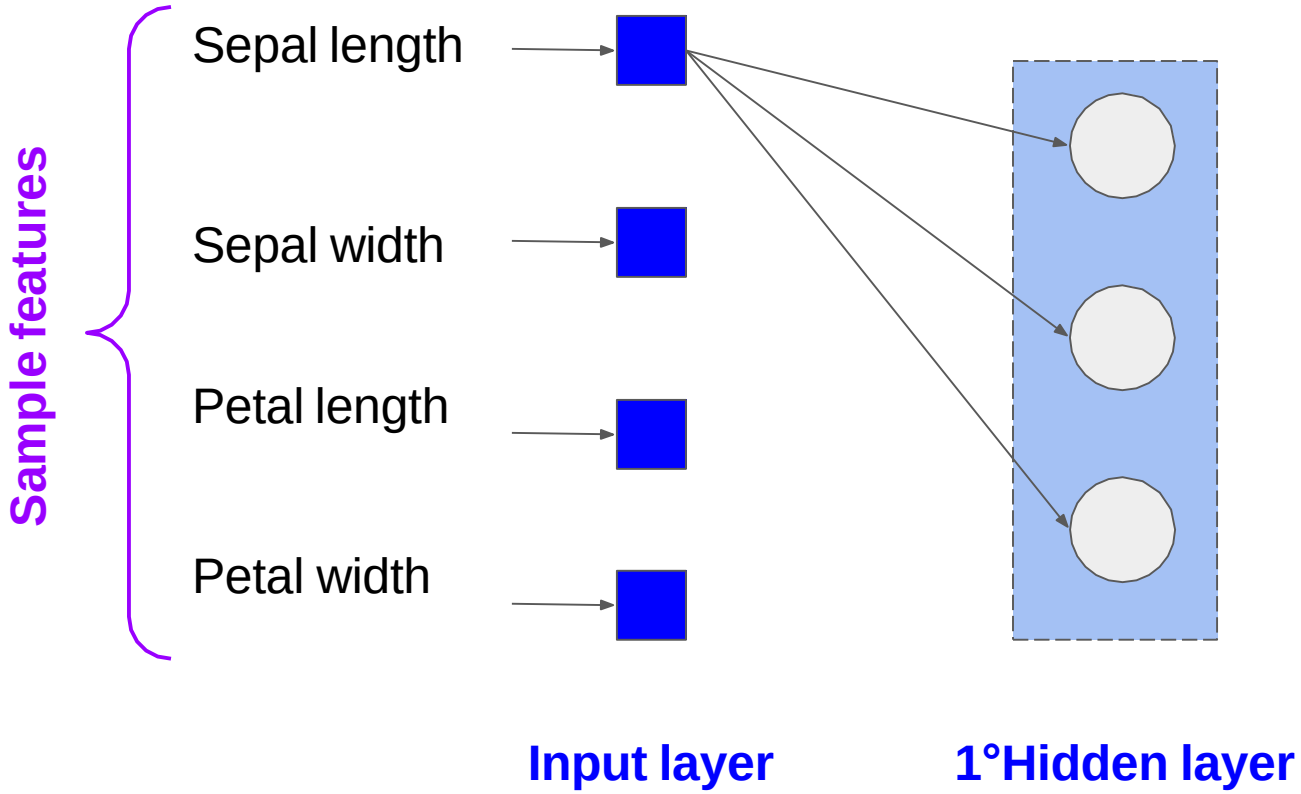
- Each connection between input and neurons, and between neurons and neurons has an associated weight **w**
- Weights are randomly initialized
- ***Our goal during the training step is to learn these weights***
- All weights between two layers are organized in a matrix ***W***

Some examples: Iris classification

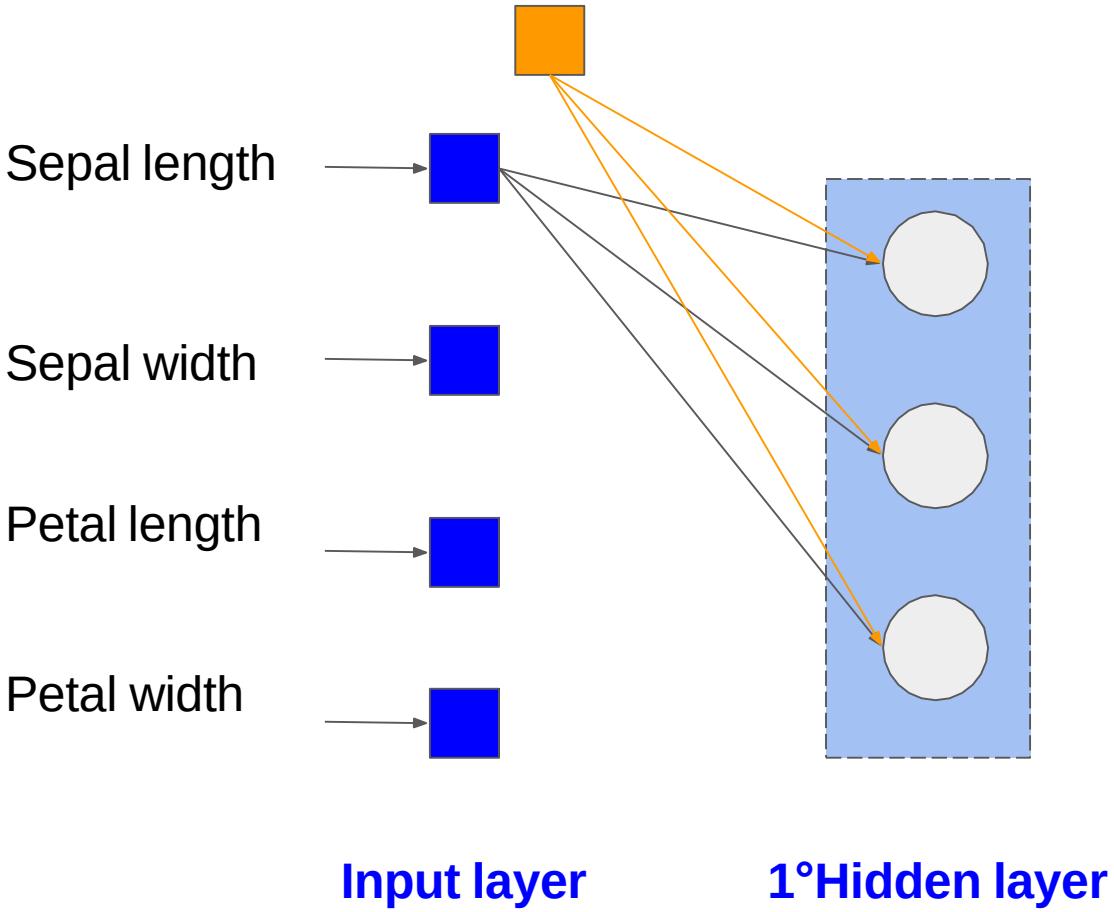


Input layer

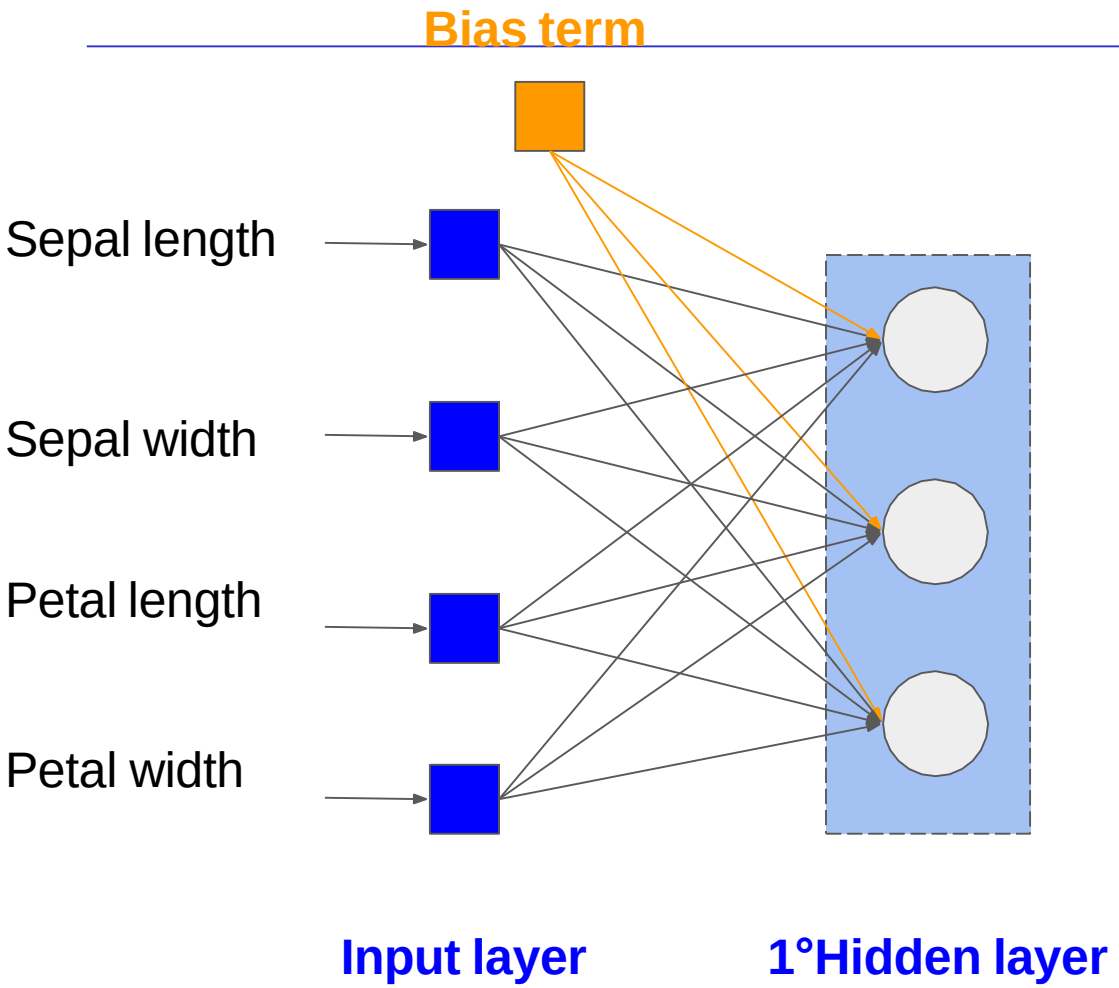
Some examples: Iris classification



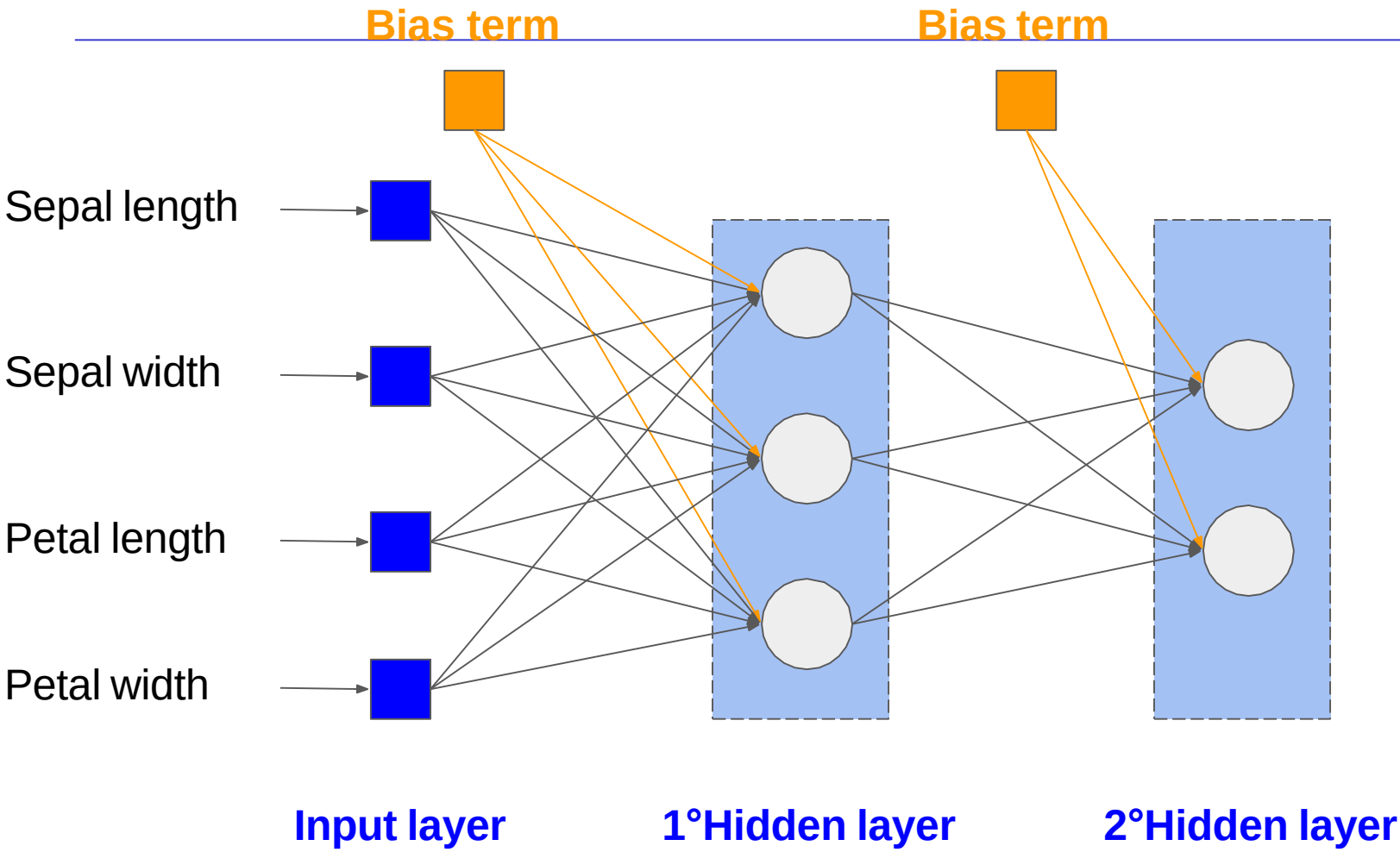
Some examples: Iris classification



Some examples: Iris classification

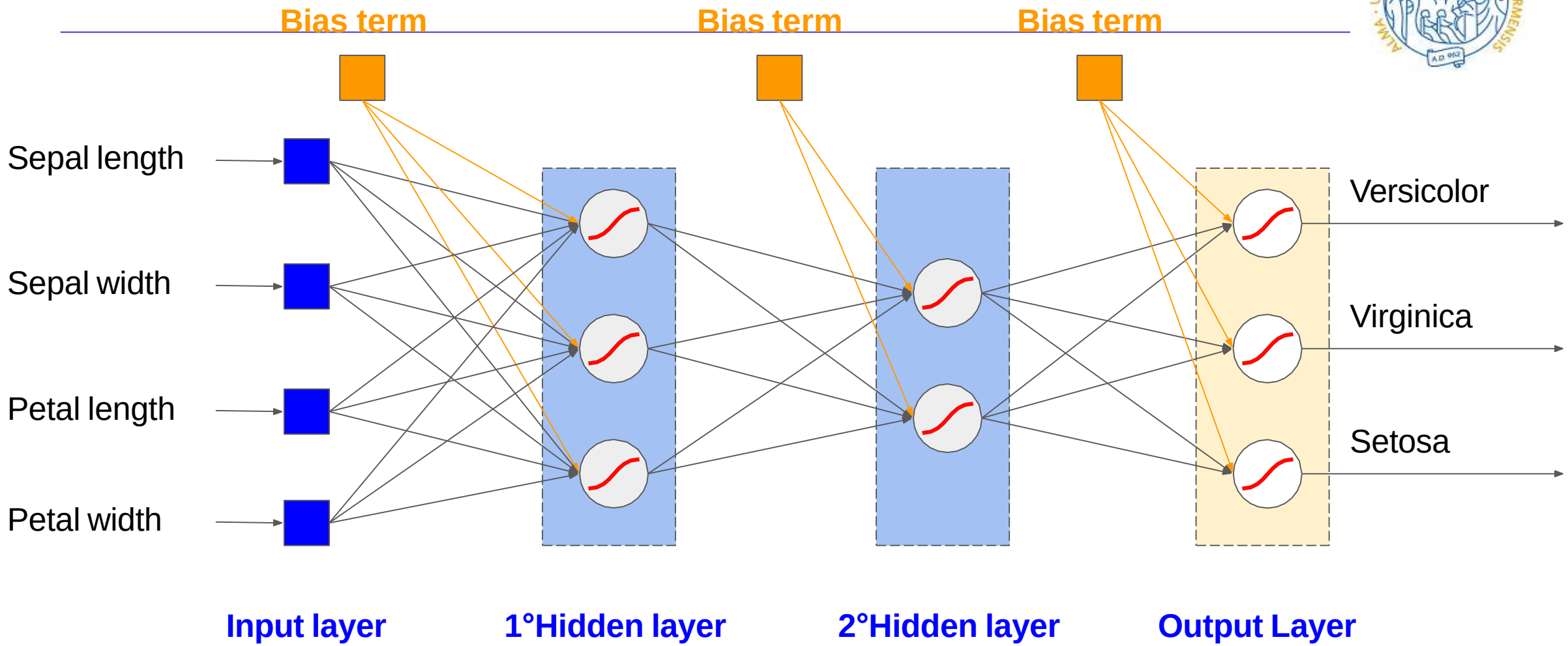


Some examples: Iris classification



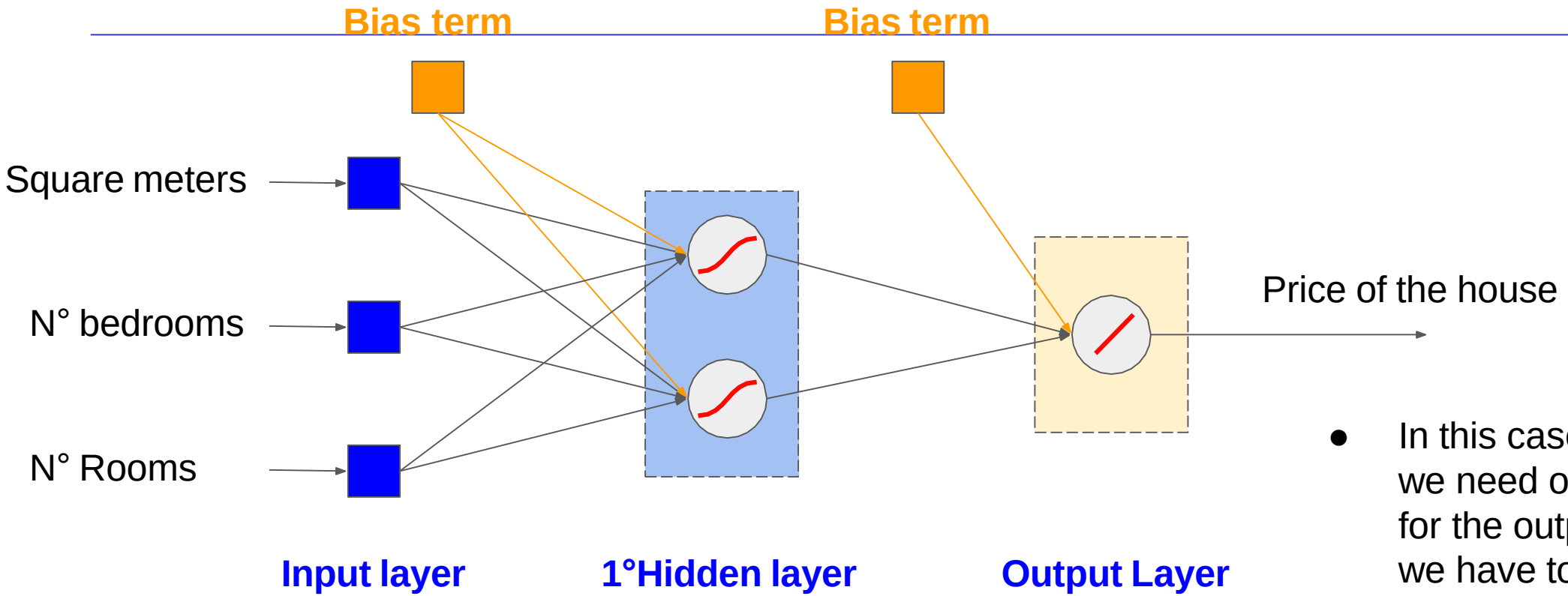
- Now we want to add the output layer to make predictions
- How many neurons we need ?

Some examples: Iris classification



is the Sigmoid Activation function

Another example: House price prediction



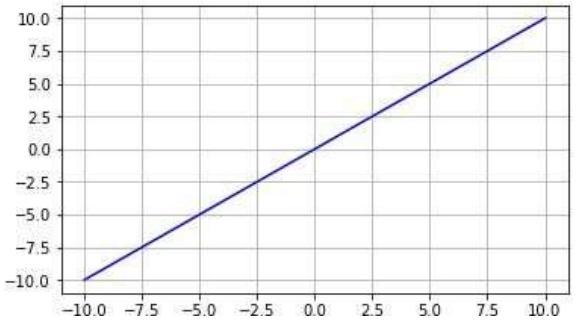
- In this case of regression we need only one neuron for the output layer since we have to predict only one value
- We cannot use the Sigmoid activation function since the value we are predicting is not between 0-1 but can be any value

is the Sigmoid Activation function

is the Linear Activation function

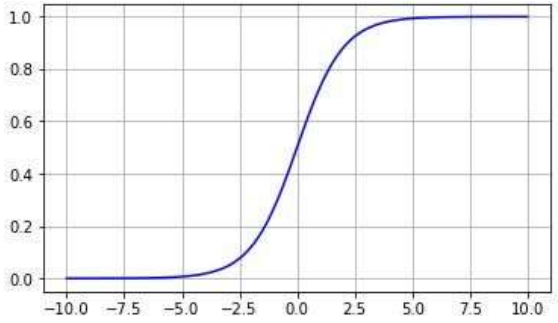
Activation functions in Neural Networks

Linear Activation



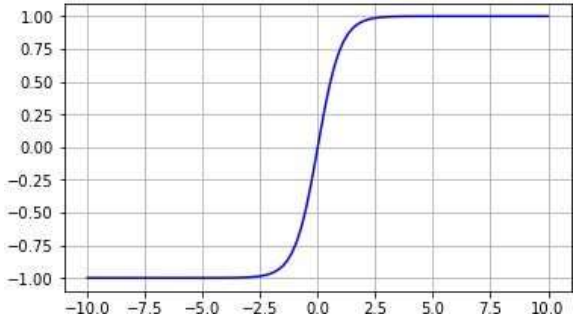
f(u) = u

Sigmoid Activation



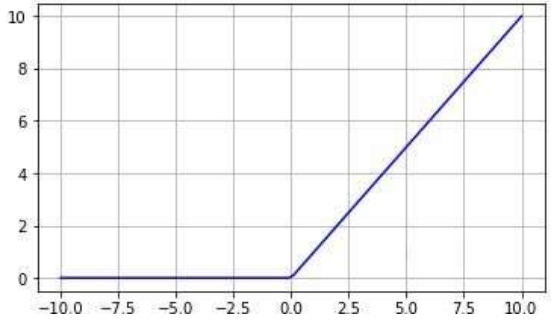
$$f(u) = \frac{1}{1+e^{-u}}$$

Tanh Activation



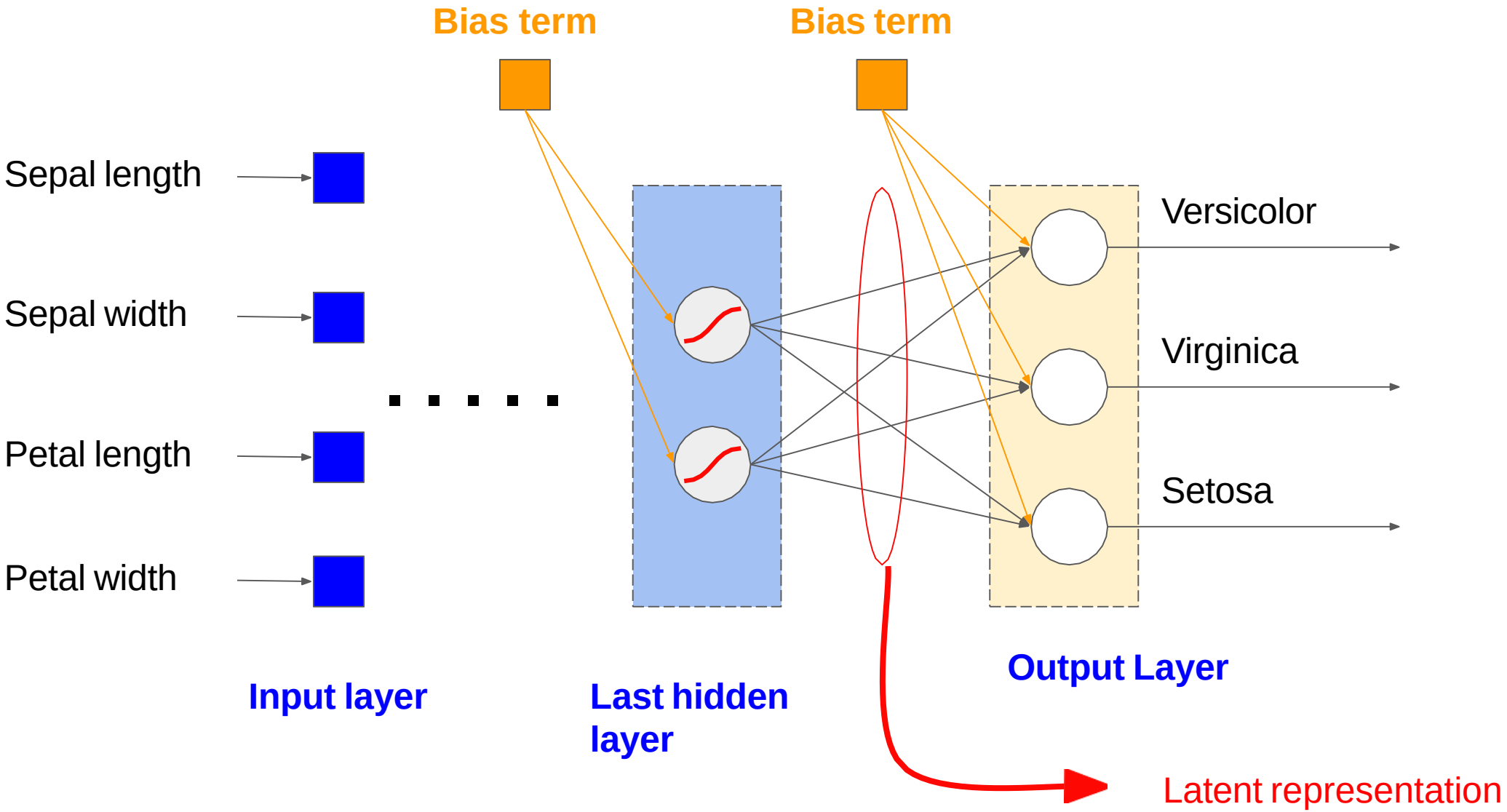
$$f(u) = \frac{e^{u}-e^{-u}}{e^{u}+e^{-u}}$$

ReLU Activation

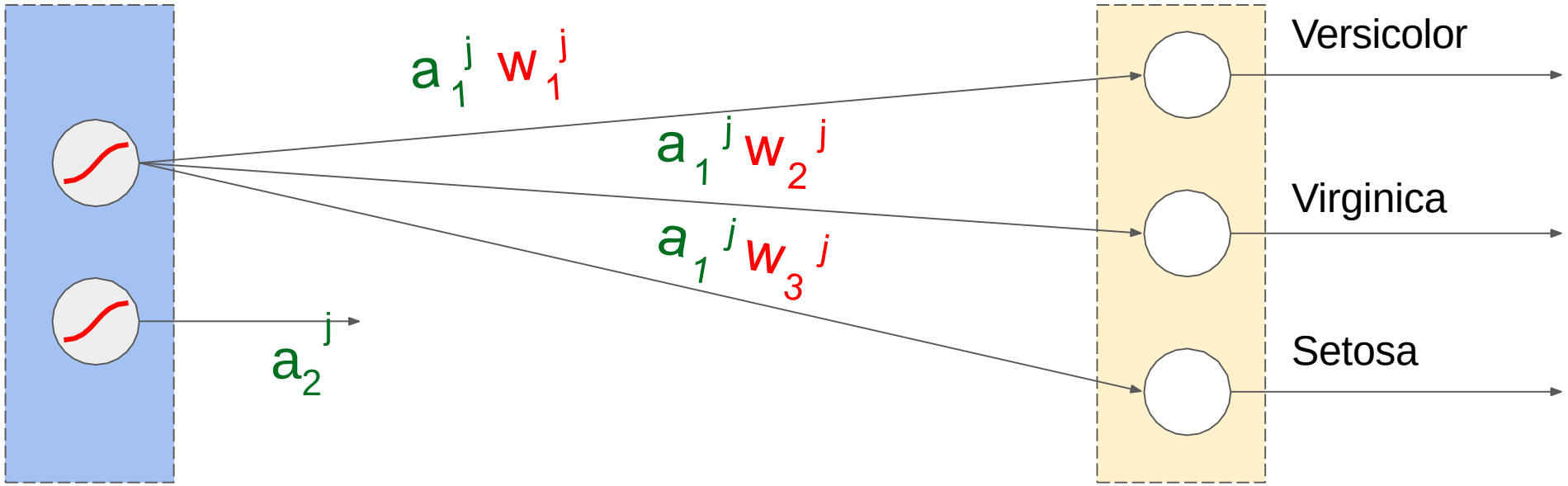


f(u) = max(0,u)

Note: The Latent representation (Embedding)



Note: The Latent representation (Embedding)



Last hidden layer (Layer j)

Output Layer

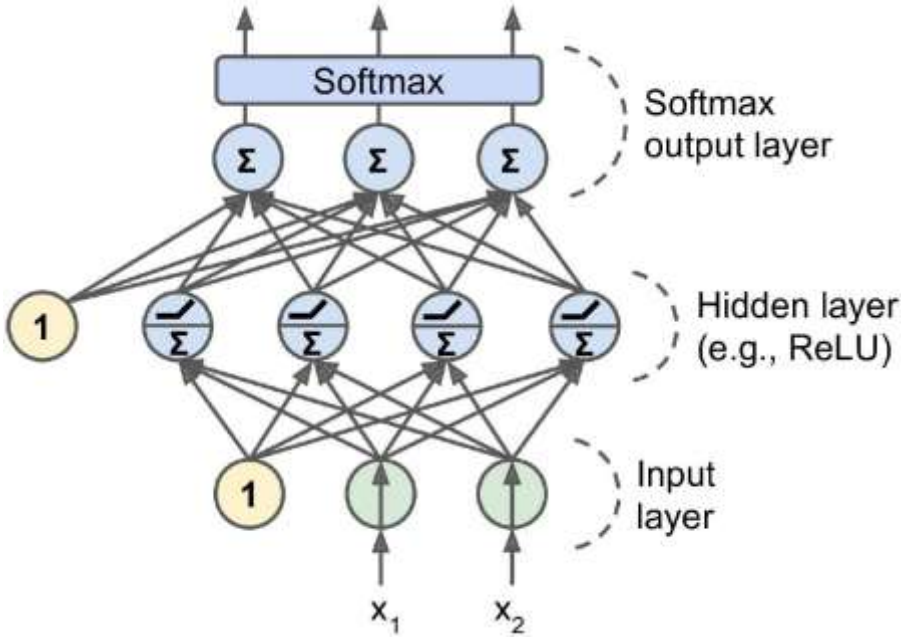
Note: The Latent representation (Embedding)

- a_1^j and a_2^j represent the resulting ***Latent representation*** that the ANN learnt about the input

- Supervised training leads to a representation of the input at each layer of the ANN
- The representation in the last hidden layer usually has the property of make the classification/regression task easier
- This capability of the Neural Network is called "***Representation Learning***"
- Several application:
  - Dimensionality reduction
  - Anomaly detection and Signal reconstruction
  - Representation of items that are not naturally a vector (Graphs, words etc..)

The Softmax output layer: Only for Classification!

- When the classes are exclusive (class 0 is a dog, class 1 is a cat, class 2 is a mouse) and the problem is not multi-label (to each sample we assign one and only one label), another type of neuron is used
- We can replace individual activation functions with a shared Softmax function
- The output of each neuron corresponds to the estimated probability of the corresponding class
- *The Softmax function is a generalization of Logistic Regression in order to support multiple classes directly without having to train and combine multiple binary classifiers*



The Softmax output layer

- The idea is the following:
  - Given an instance x or its representation in the penultimate layer of a neural network
  - The Softmax model compute a score $\mathbf{s_k(x)}$ for each class **k**
  - Then it estimates the probability of each class by applying the softmax function to these scores

- $\mathbf{s_k(x)} = W^{(k)\,T} X$
  - Each class has its own dedicated parameter vector $W^{(k)}$

- Now we can compute the probability p_k that the instance belongs to class k

$$\hat{p}_k = \sigma(\mathbf{s}(\mathbf{x}))_k = \frac{\exp\left(s_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(s_j(\mathbf{x})\right)}$$

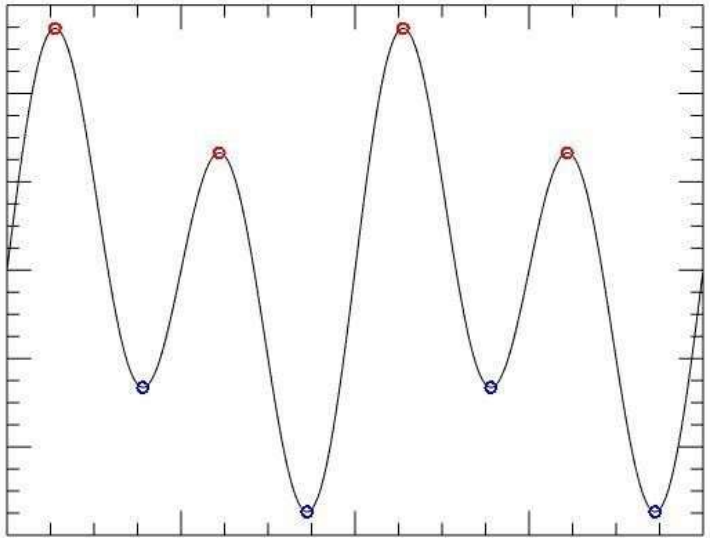
- It predicts the class with the highest probability (class with the highest score)

Neural networks: Optimization problem

- Our goal is to **minimize** an **objective function**, which measures the difference between the actual output t and the predicted output y.
    - In this case we will consider as the objective function the **squared loss function**.

Squared loss function

$$E = \frac{1}{2}(t - y)^2 = \frac{1}{2}(t - f(\mathbf{w}^\mathbf{T}\mathbf{x}))^2$$



Loss functions

Squared loss function

$$E = \frac{1}{2}(t - y)^2$$

Mean squared error

$$E = \frac{1}{n}\sum_{i=1}^{n}(t_i - y_i)^2$$

Mean absolute error

$$E = \frac{\sum_{i=1}^{n} | t_i - y_i |}{n}$$

Cross entropy

$$E = -\sum_{i=1}^{n} t_i \log(y_i)$$

Kullback Leibler divergence

$$E = \sum_{i=1}^{n} t_i \log\left(\frac{t_i}{y_i}\right)$$

Cosine proximity

$$E = \frac{t \;\; y}{||t|| \;\; ||y||}$$

Loss functions

- **For Regression task**
  - Mean Squared Error Loss
  - Mean Absolute Error Loss

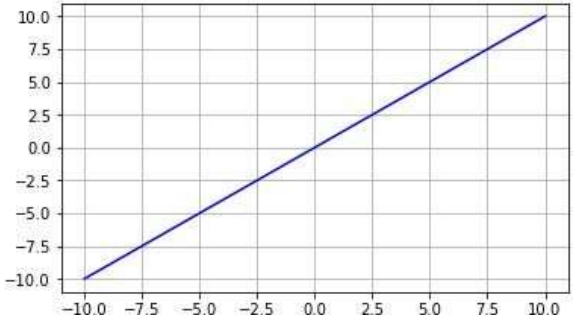
- **For Binary Classification**
  - Binary Cross-Entropy

- **For Multi-Class Classification**
  - Multi-Class Cross-Entropy Loss
  - Kullback Leibler Divergence Loss

- **<u>Note: This list is not exhaustive</u>**

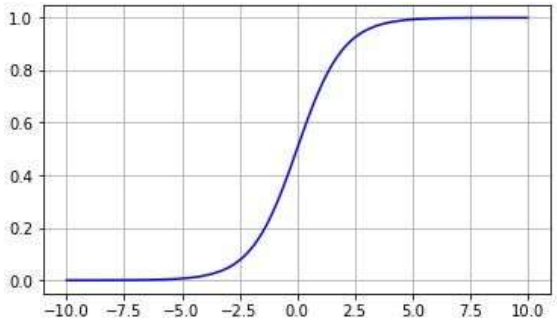
Activation functions in Neural networks

Linear Activation



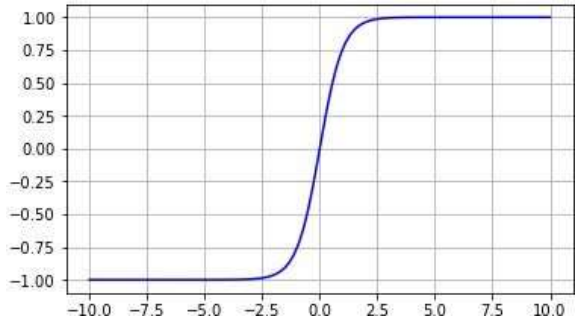
f(u) = u

Sigmoid Activation



$f(u) = \frac{1}{1+e^{-u}}$

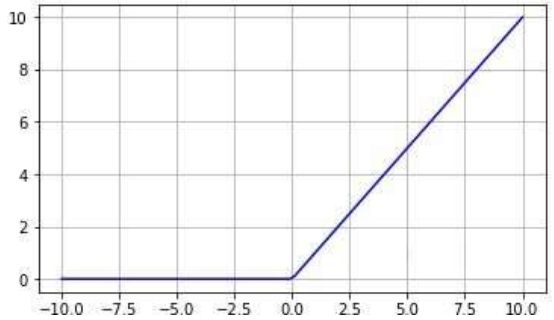
Tanh Activation



$f(u) = \frac{e^{u}-e^{-}}{e^{u}+e^{-}}$

- Avoid **vanishing gradient** in flat components of activation functions

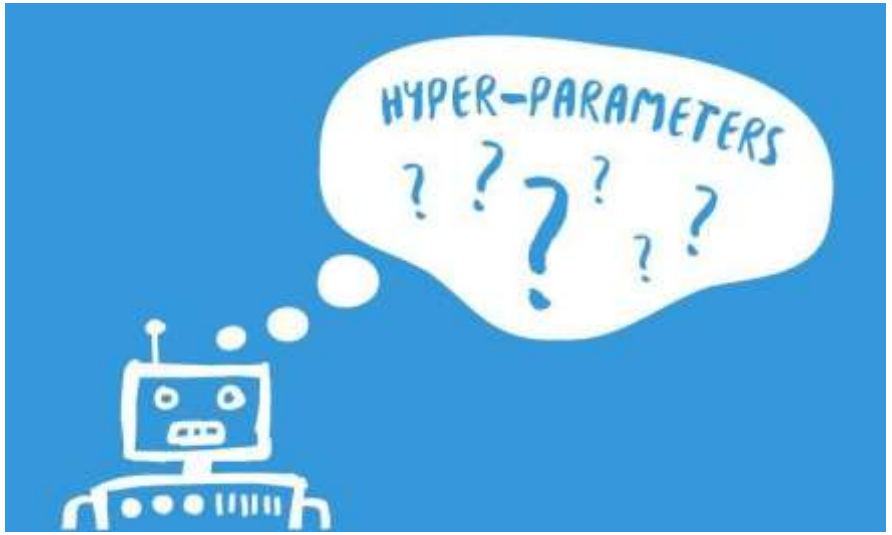
ReLU Activation



f(u) = max(0,u)

Hyper-parameters

- Hyperparameters are the parameters which determine the **network structure** (e.g. Number of Hidden Units) and the parameters which determine **how** the **network** is **trained** (e.g. Learning Rate)
  - Number of neurons
  - Number of layers
  - Learning rate
  - Batch size
  - Number of epochs
  - others

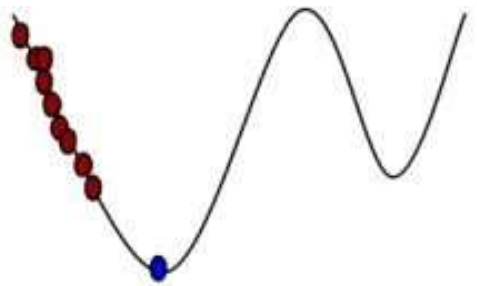


Learning rate

- Training parameter that controls the size of weight changes in the learning phase of the training algorithm.
- The learning rate determines how much an updating step influences the current value of the weights.

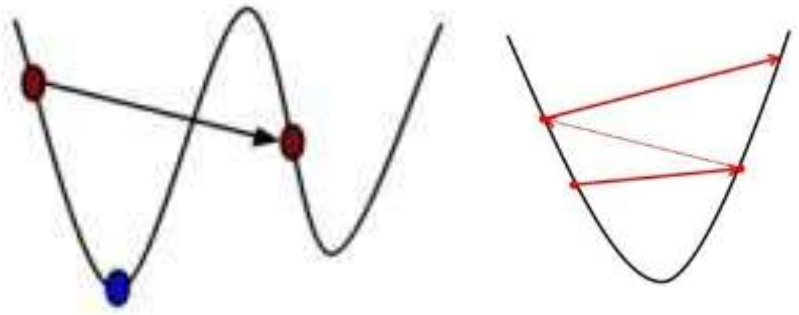
$$w_i^{new} = w_i^{old} - \eta \frac{\partial E}{\partial w_i}$$

Very small learning rate



Many updates required before reaching the minimum.

Too big learning rate



Drastic updates can lead to divergent behaviors, missing the minimum.

Hyper-parameters

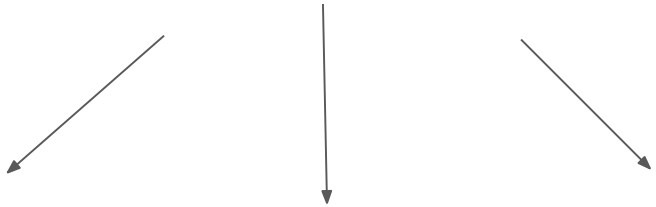
Number of epochs

- The number of epochs is the number of times the whole training data is shown to the network while training.

- Remember that at the beginning weights are randomly initialized. Our training is sensitive to this initialization

Batch size

- The number of samples shown to the network before the gradient computation and the parameter update.



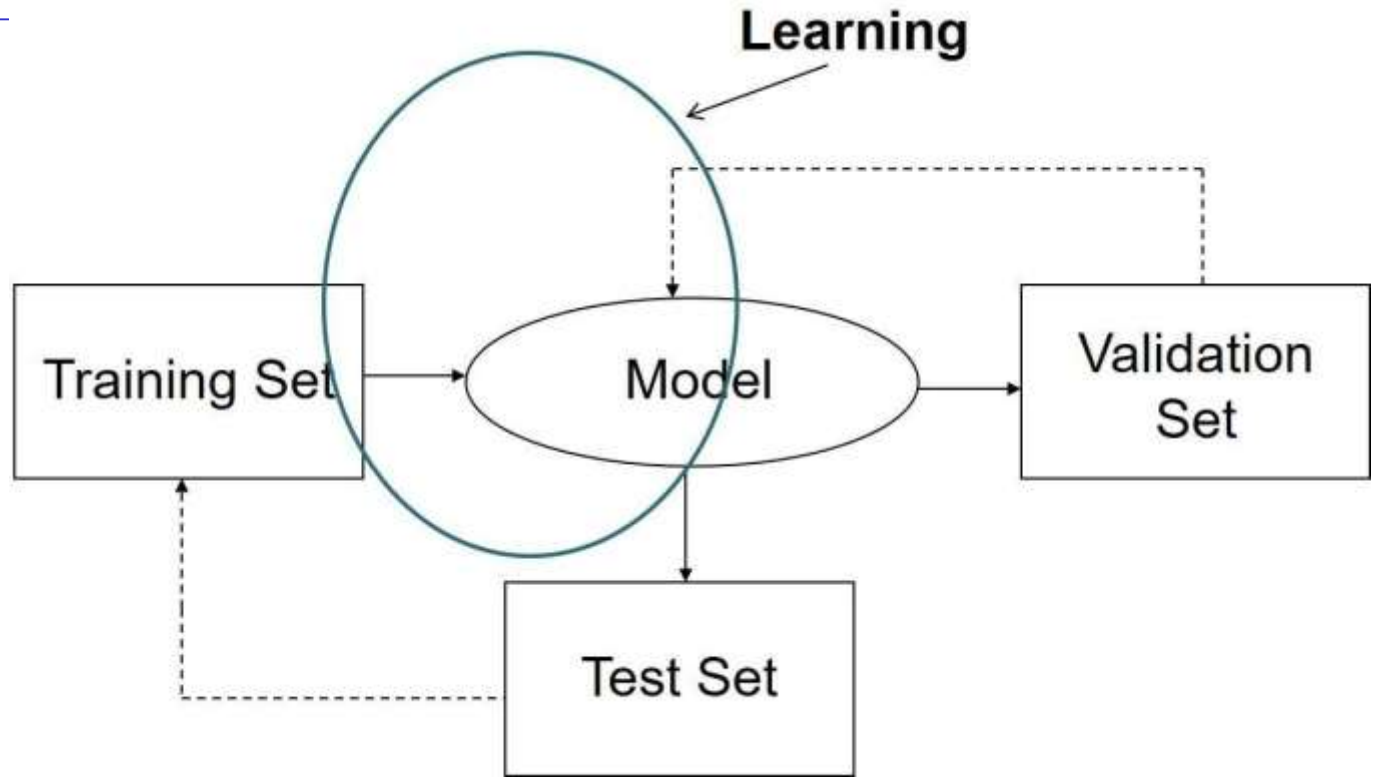
Stochastic Gradient Descent

Gradient Descent

Batch Gradient Descent

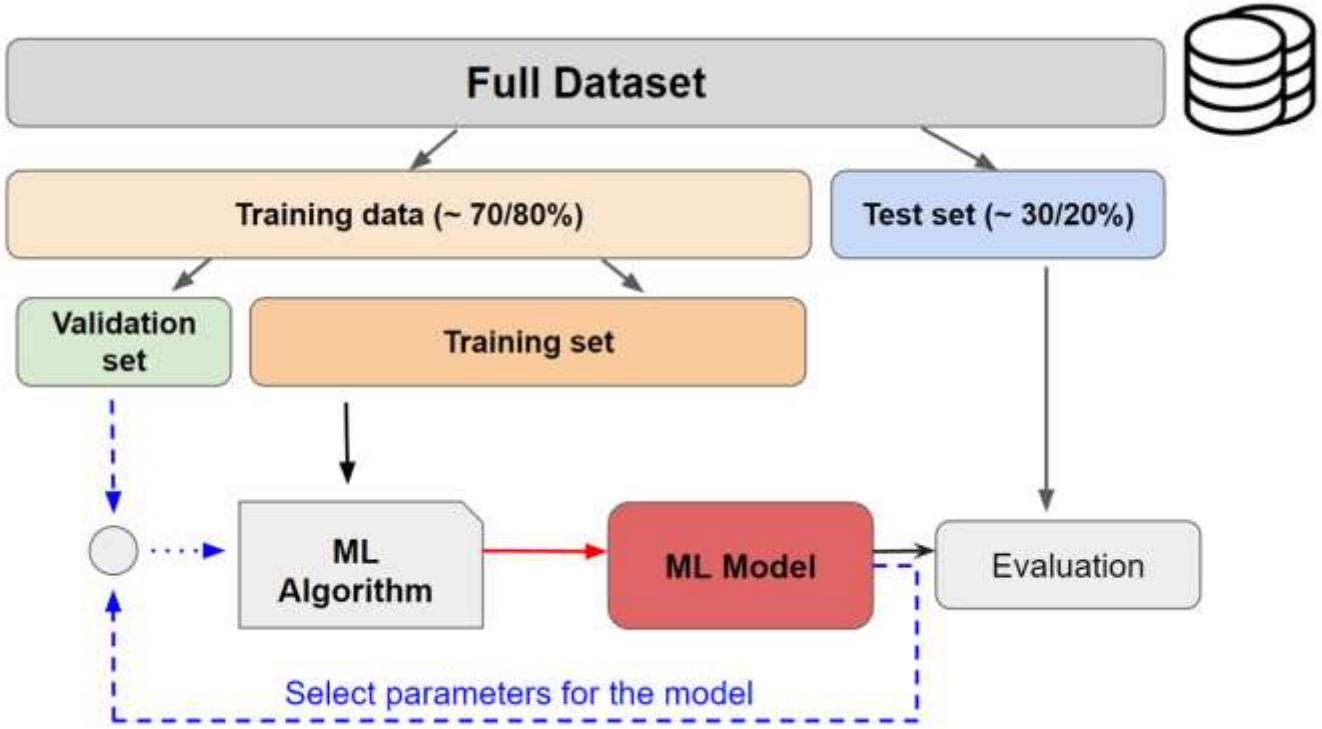
Validation set

- Data set with the ‘same’ goal of the test set (verifying the quality of the model which has been learnt), but not as a final evaluation, but as a way to fine-tune the model.
- Its aim is to provide a feedback which allows one to find the best settings for the learning algorithm (parameter tuning).



Reminder: How to use data

- **Training Set**: the dataset from which we learn
- **Validation set:** A dataset we use to tune model's parameters
- **Test Set**: A dataset which must include patterns describing the same problem which do not belong to the training set
    - The test set is used to verify whether what has been learnt in the training phase can be correctly **generalized** on new data



How to choose hyper-parameters ?

Grid-search

- We try every combination of a preset list of values of the hyper-parameters and evaluate the model for each combination.
- Each set of parameters is taken into consideration and the accuracy is noted.
- Once all the combinations are evaluated, the model with the set of parameters which give the top accuracy with the VALIDATION SET is considered to be the best.
- The number of evaluations required for this strategy increases exponentially with each additional parameter

Random search

- Random combinations of the hyperparameters are used to find the best solution
- It works well for lower dimensional data since the time taken to find the right set is less with less number of iterations
- In Random Search for Hyper-Parameter Optimization by Bergstra and Bengio, the authors show empirically and theoretically that random search is more efficient for parameter optimization than grid search
- When dimension is high it is better to combine with a Grid-search

Early stopping

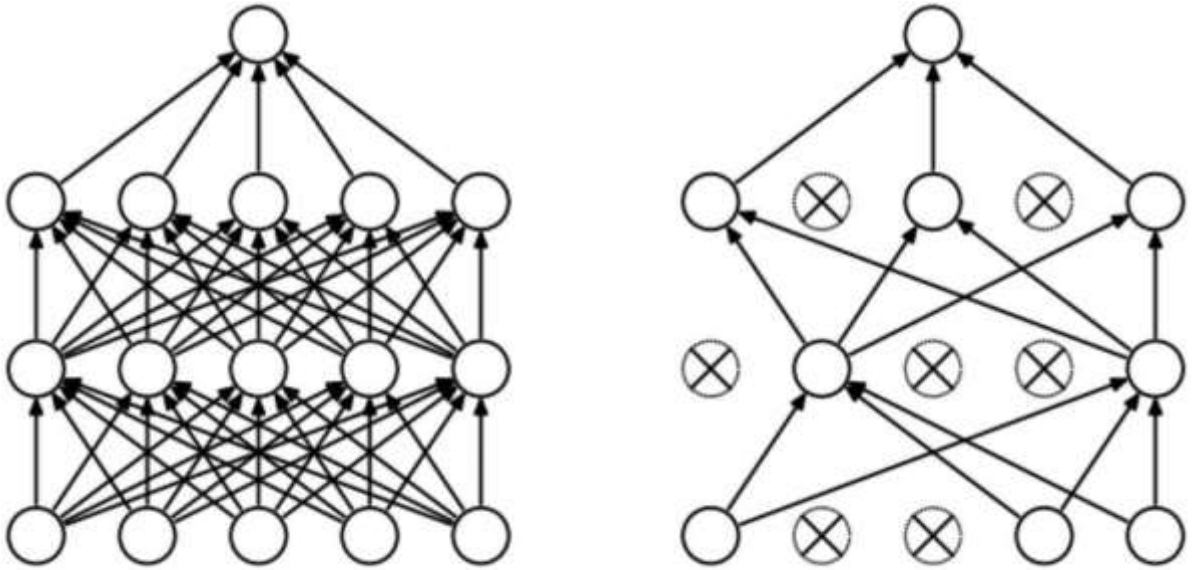
- Early stopping is a form of regularization used to avoid overfitting when training a learner with an iterative method, such as gradient descent

- Stop training as soon as the error on the validation set is higher than it was the last time it was checked
  - We can define a patient parameters: We accept that a patient number of times the validation error can be higher than the previous iteration. After this number is reached, training will be stopped.

- Use the weights the network had in that previous step as the result of the training run

Dropout

- It is another form of regularization for Neural Networks
- At each update during training time, randomly setting a fraction rate of input units to 0.
- It helps to prevent overfitting.



(a) Standard Neural Net

(b) After applying dropout.

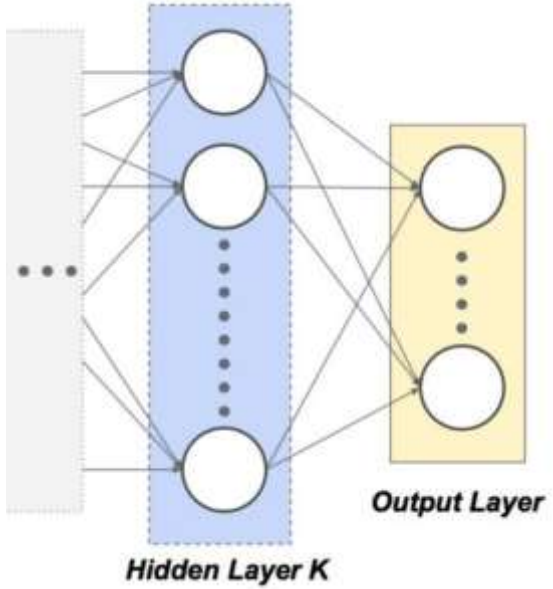
The output layer

- We should say something more about the output layer.
- The structure of this last layer depends on the task you want to perform.
- y in our dataset has to be formatted depending on this layer
- Usually the number of neurons is equal to the number of expected possible outputs, but we should pay attention:

  - **Regression**: For example price prediction of a house, we need only one neuron with an activation function that is able to produce the value we need (for example a linear function)

  - **Binary classification**: We can have one neuron with a sigmoid activation function. <u>But it is not the only possibility!</u>

  - **Multi-class or multi-label classification :** We will have a number of neurons equal to the number of the possible classes. Each neuron is like a binary classifier.

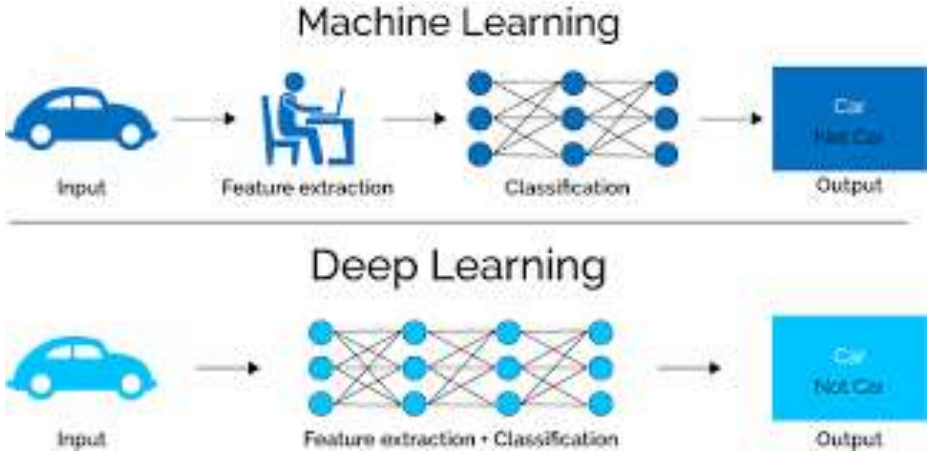


Deep Learning

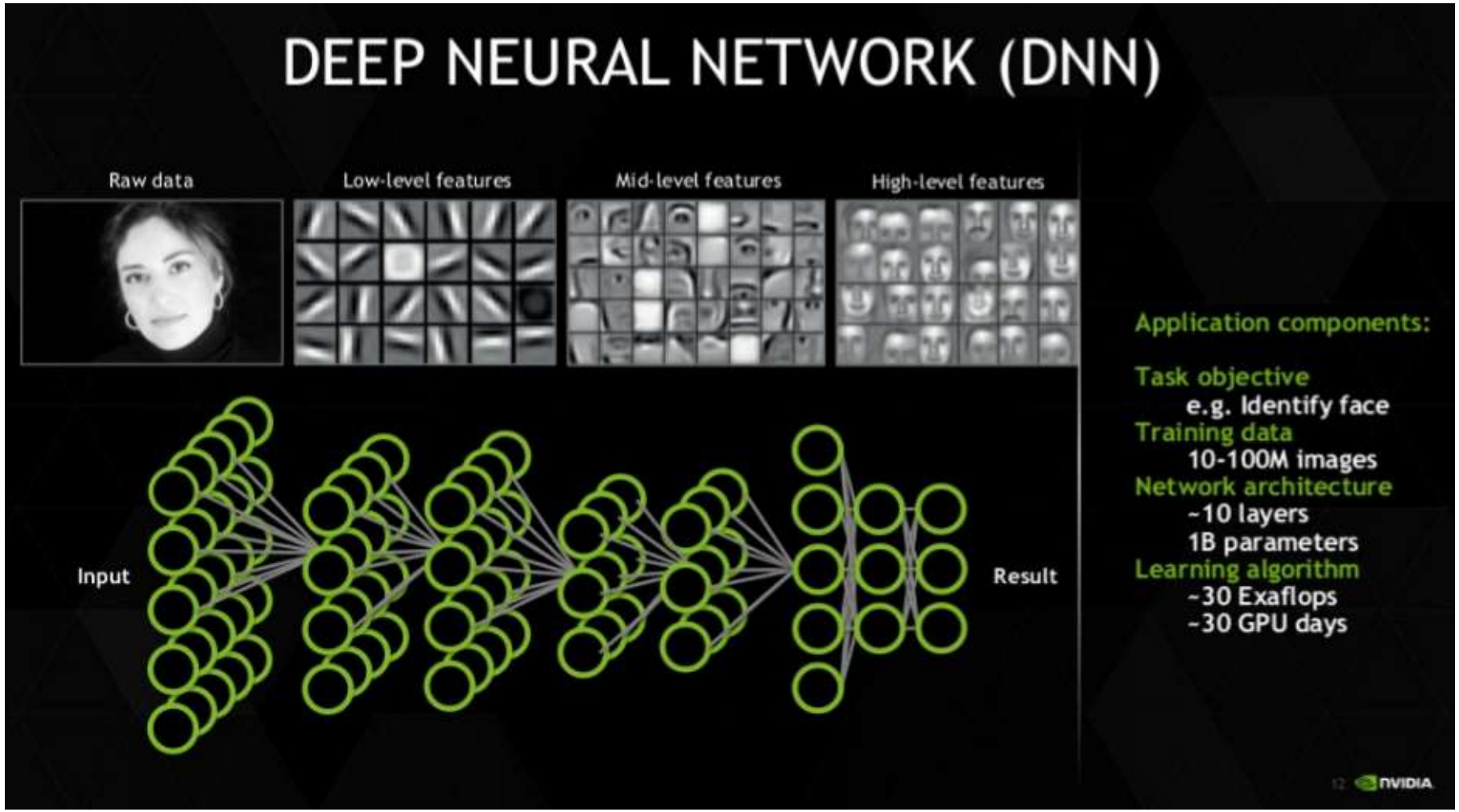
Usually, the input data derives from a pre-processing of the available data (feature extraction) in order to represent even complex data (e.g. an image) with a relatively limited vector of numerical values.

By using networks with a very large number of hidden layers and with layers that perform more complex functions, very powerful networks are obtained that are able to carry out the data pre-processing phase, freeing the user from the need to calculate a more "compact" representation of the input data.

These networks provide, in many cases, excellent results.



Deep Learning



Deep Learning

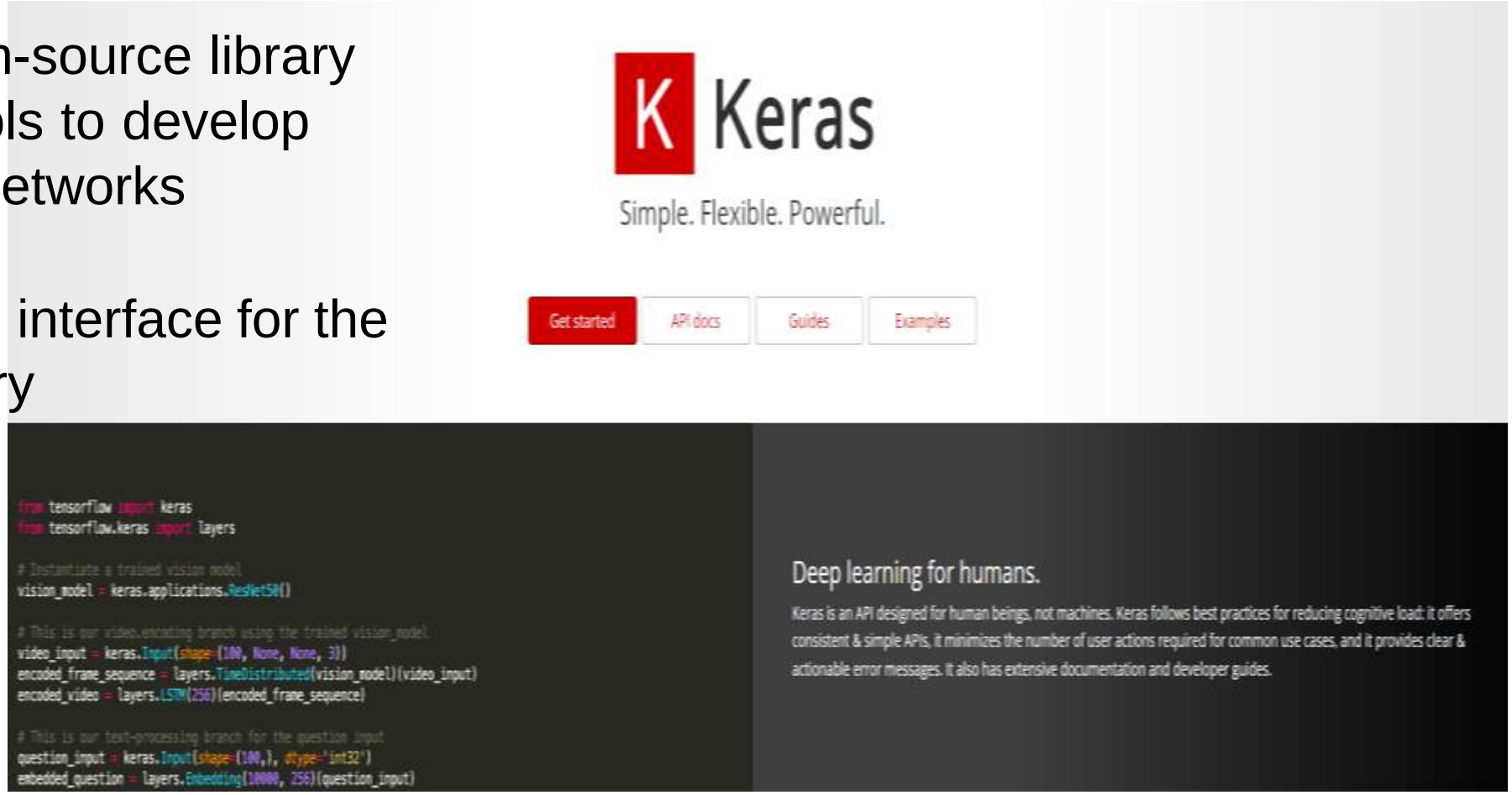
- **Deep neural networks are made up of dozens of layers of interconnected neurons.**
  - Require computational power
  - -Require lot of data
- There are many different types of networks, each of which is suited to solving a certain class of problems.

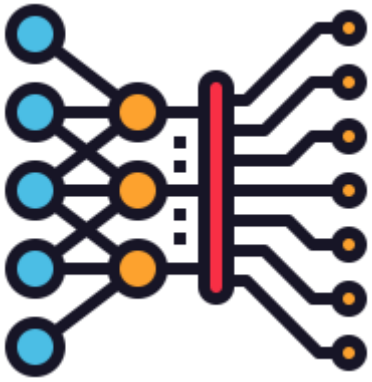
- Several models belong to this group:
  - Multi-layer Perceptron with several hidden layers
  - Convolutional Neural Networks (CNN)
  - Recurrent Neural Networks (RNN)
  - Auto-encoder for unsupervised learning
  - Graph Neural Network (GNN)
  - Graph Convolutional Network (GCN)
  - Transformers and Attention Models
  - Others

ANN in python

https://keras.io/

- Keras is an open-source library that provides tools to develop artificial neural networks
- Keras acts as an interface for the TensorFlow library





let's sum it up

AI & DEEP LEARNIG & INTELLIGENT MACHINES

Deep Learning

- The Electronic Frontier Foundation has developed a methodology for measuring the progress of artificial intelligence by comparing its results with human performance in certain fields (2018-2020)
- https://www.eff.org/it/ai/metrics

- Finding that
- Machines surpass man in the ability to **identify objects in a photograph**, in recognizing handwritten characters (particularly numbers), in **playing chess**, go and **poker** and in many video games (by autonomously creating their own strategy), in **recognizing words during a spoken dialogue**.
  - Impressive results are also in the driving of autonomous vehicles, and in the diagnosis of diseases of various kinds.

Deep Learning

- Instead, machines are doing worse than men, but they are improving, in translating from one language to another, in some role-playing games, in understanding what is represented in a photograph, in holding an interview ***taking into account the context*** ...

- ✓ *In practice we are still at narrow artificial intelligence.*
- ✓ *And far from realizing a general artificial intelligence equipped with a generalist intelligence capable of passing from one context to another*

Deep Learning

Deep Learning is the paradigm that is producing the most striking results today

But:

- **they need a huge number of examples..**
- ***Humans need only a few***:
- A child is able to generalize having seen a few examples of an animal, such as a cat.
- The network today is able to autonomously classify an object (even a cat) without anyone explaining it to it. But it needs an enormous number of examples and consequently of time and processing power with consequent use of enormous amounts of energy
- All we need is the energy taken with a sandwich.

Deep Learning

limits:

- **have difficulty generalizing.**
  - They are able to generalize within a specific context, but do not know, as yet, how to transfer this ability to a different context;
- **have difficulty explaining their own behavior.**
  - They are often black boxes;
- **have difficulty distinguishing between correlations and cause and effect.**
  - It's not easy for us either…