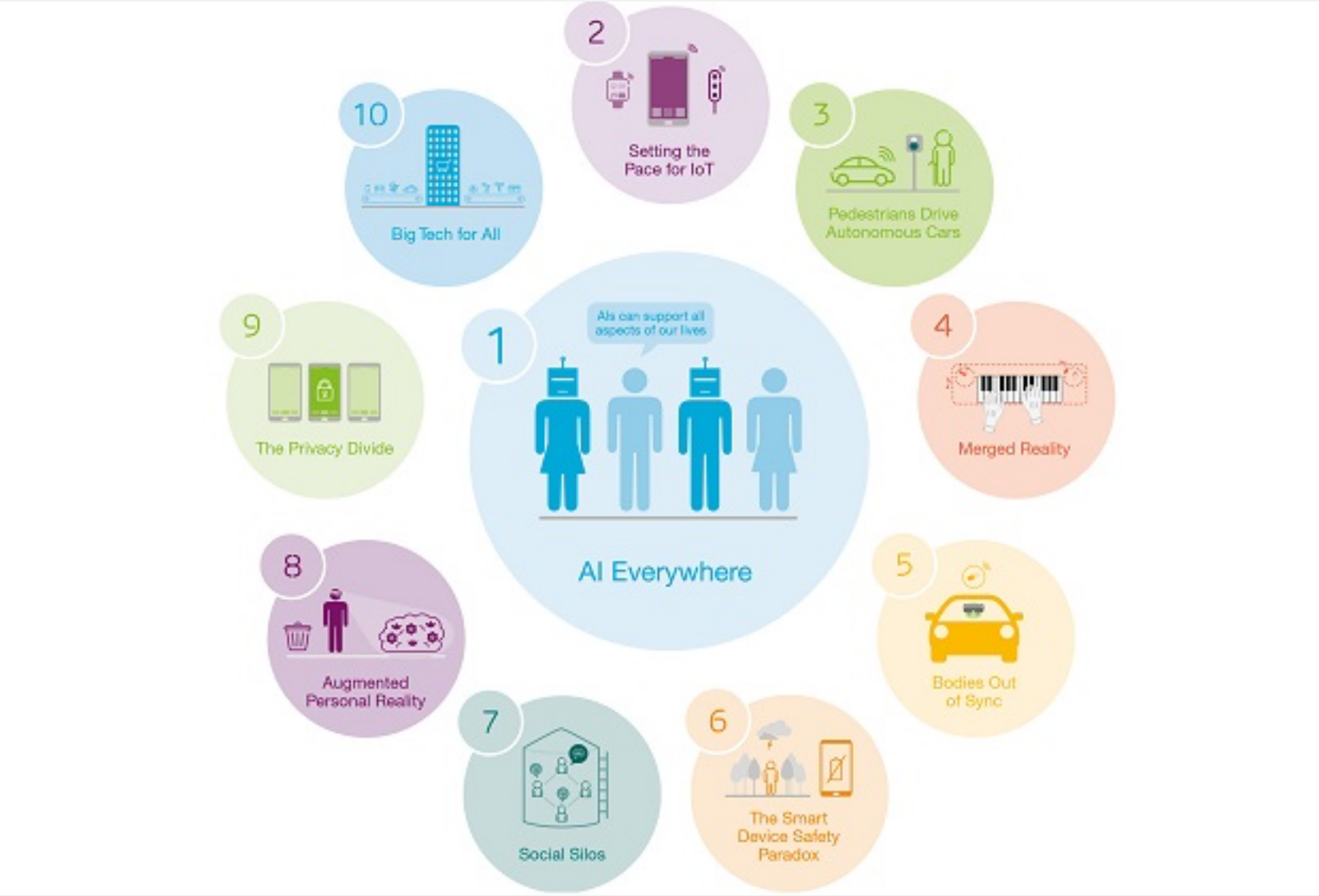
POLITECNICO
MILANO 1863

IMAGE AND SOUND
ISPG
PROCESSING GROUP

CREATIVE PROGRAMMING AND COMPUTING

Deep Learning

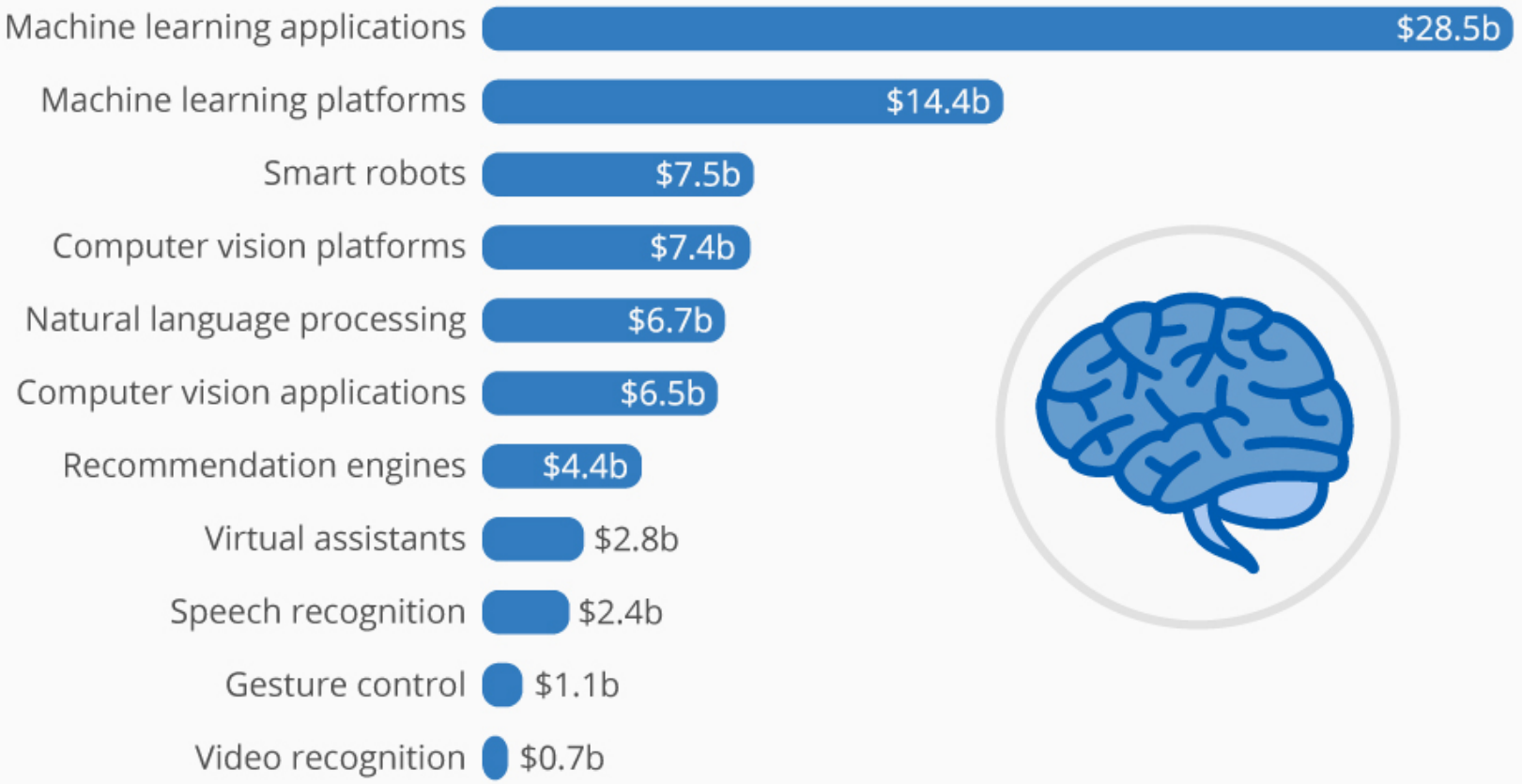
DEEP LEARNING: A TREND



DEEP LEARNING: A TREND

Machine Learning Tops AI Dollars

AI funding worldwide cumulative through March 2019 (in billion U.S. dollars), by category



Sources: Venture Scanner, Statista estimates

statista

WHAT'S DEEP LEARNING ?

Supervised Learning —Essentially, a strategy that involves a teacher that is smarter than the network itself. For example, let's take the facial recognition.

Unsupervised Learning —Required when there isn't an example data set with known answers. Imagine searching for a hidden pattern in a data set.

Reinforcement Learning —A strategy built on observation. Think of a little mouse running through a maze. If it turns left, it gets a piece of cheese; if it turns right, it receives a little shock. Presumably, the mouse will learn over time to turn left.

Its neural network makes a decision with an outcome (turn left or right) and observes its environment (yum or ouch). If the observation is negative, the network can adjust its weights in order to make a different decision the next time.

WHAT'S DEEP LEARNING ?

Imagine an organism or machine that experiences some of sensory inputs: E = x1, x2, x3, x4, . . .

- **Supervised learning**: given the desired outputs y1, y2, . . . , learn to produce the correct output given new input

- **Unsupervised learning**: exploit regularities in E to build a representation that can be used for reasoning or prediction

- **Reinforcement learning**: producing actions a1, a2, . . . which affect the environment, and receiving rewards r1, r2, . . . learn to act in a way that maximizes rewards in the long term

WHY DEEP LEARNING ?

- **Rule-based/Grammar-based systems**
  - Simulate how way of reasoning
  - Cons: cannot be exhaustive

- **Classical Machine Learning systems**
  - Simulate how way of perceive –> Feature extraction
  - Simulate how way of learning -> Training process
  - Simulate how way of reasoning -> Application of the learnt model to a specific function
  - **Cons**: features set not exhaustive, functions are not learnt –> very specific to the task

WHY DEEP LEARNING ?

- **Deep Learning**
  - Simulate the brain functioning
  - Starts from the more informative representation of the elements
  - The network structure of the net will learn
    - How to produce synthetic descriptive features (i.e. Spectrum)
    - How to apply the features for a specific task (function)
    - How to semantically abstract from the signal

NSYNTH

Sounds generated by AI starting from some timbres

Nsynth
https://nsynthsuper.withgoogle.com/

Nsynth blog description
https://magenta.tensorflow.org/nsynth

Build Nsynth
https://github.com/googlecreativelab/open-nsynth-super

https://www.youtube.com/watch?v=0fjopD87pyw&feature=emb_logo

https://magenta.tensorflow.org

DEEP DREAM

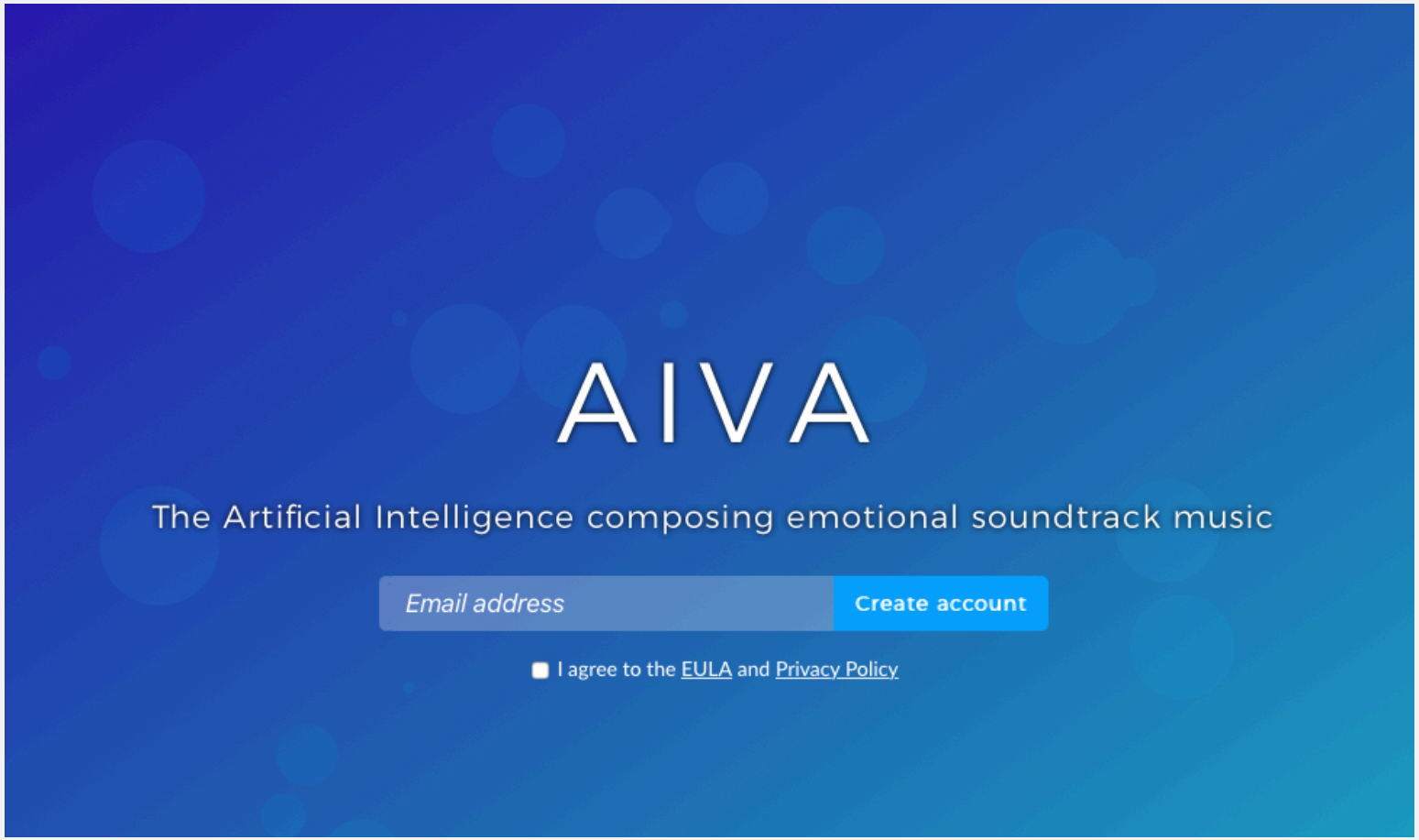
Create Images

https://deepdreamgenerator.com

https://www.youtube.com/watch?v=SCE-QeDfXtA

AIVA

Music composition for sound tracks



https://www.aiva.ai

https://www.youtube.com/watch?v=KPunHZtquoQ

ANNA RIDLER “TULIPMANIA”

https://www.linkedin.com/posts/fusefactory_generative-particle-artificialneuralnetworks-activity-6592439908839763968-vKmD

https://vimeo.com/287645190

MEMO – LEARNING TO SEE

https://vimeo.com/260612034

http://www.memo.tv/works/

VIDEO STYLE TRANSFER

1 Upload photo

The first picture defines the scene you would like to have painted.

2 Choose style

Choose among predefined styles or upload your own style image.

3 Submit

Our servers paint the image for you. You get an email when it's done.

https://www.youtube.com/watch?v=fcnjHmBcLNQ

FLOW-MACHINES

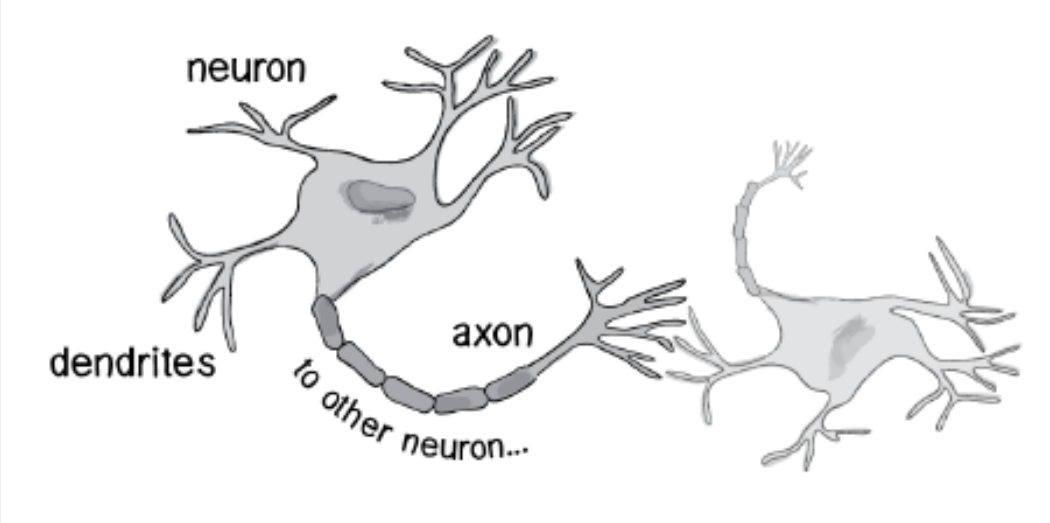
Automatic music composition in a style – Daddy's car

https://www.flow-machines.com

https://www.youtube.com/watch?v=LSHZ_b05W7o

PERCEPTRON

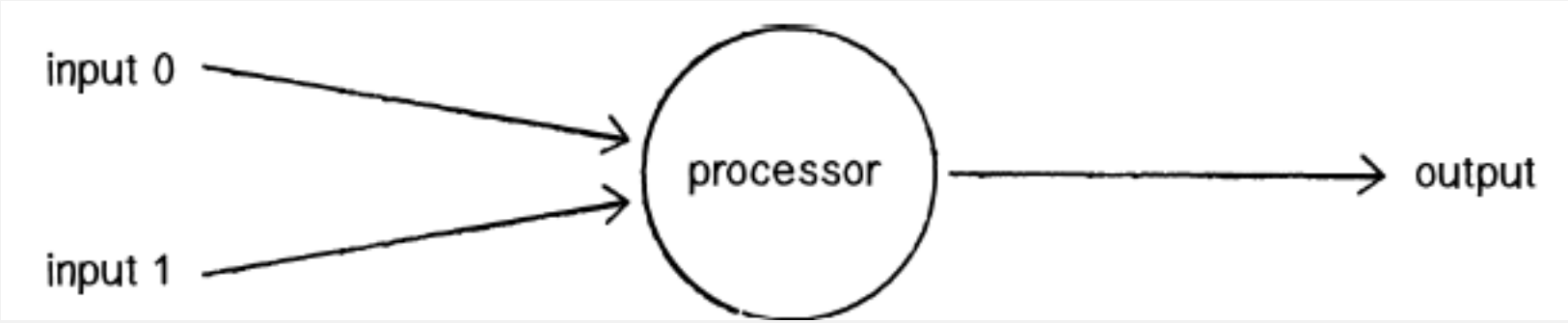
PERCEPTRON



Information is transmitted through chemical mechanisms:

- Dendrites collect input charges from synapses
  - Inhibitory synapses with different weight
  - Excitatory synapses with different weight
- Axon transmit accumulated charges through synapses
  - Once the charge is above a threshold the neuron fires (activation)

PERCEPTRON

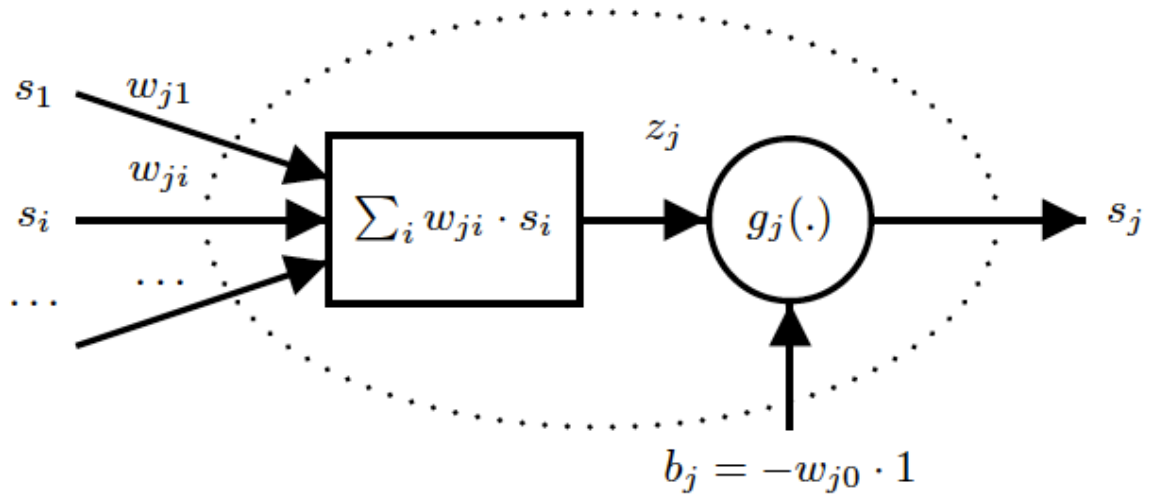


- A perceptron follows the "**feed-forward**" model, meaning inputs are sent into the neuron, are processed, and result in an output

- Artificial **Dendrites ->** neuron input
- Each input that is sent into the neuron must first be weighted, i.e. multiplied by some value (often a number between -1 and 1) – Inhibitory (negative weights) – Excitatory (positive weights)
- The weighted inputs are then summed
- **Axon ->** output and are activated according to an activation procedure

PERCEPTRON

- Given a generic neuron j



- S_i are input
- The synaptic weights are w_{ij}
- The activation procedure is divided in:
  - Activation value $z_{ij} = \sum_i w_{ij} s_i$
  - Activation function $g(.)$
  - Activation threshold (bias) $b_j = -w_{j0} * 1$

Final Output

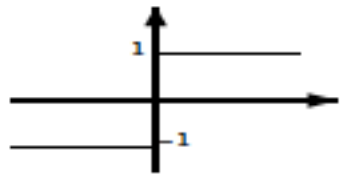
$$s_j = g_j(\sum_{i=1}^{N} w_{ij} s_i - b_j) = g_j(\sum_{i=0}^{N} w_{ij} s_i)$$

PERCEPTRON

$$s_j = g_j(\sum_{i=1}^{N} w_{ij} s_i - b_j) = g_j(\sum_{i=0}^{N} w_{ij} s_i)$$

- **The output of a perceptron is generated by passing that sum through an activation function.**
- In the case of a **simple binary output**, the activation function is what tells the perceptron whether to "fire" or not.
- Example: a LED connected to the output signal: if it fires, the light goes on; if not, it stays off.
- An example of activation function is the st**ep function**

Step function: $g(z_j) = \begin{cases} 1 & \text{if } z_j > 0 \\ 0 & \text{if } z_j = 0 \\ -1 & \text{if } z_j < 0 \end{cases}$



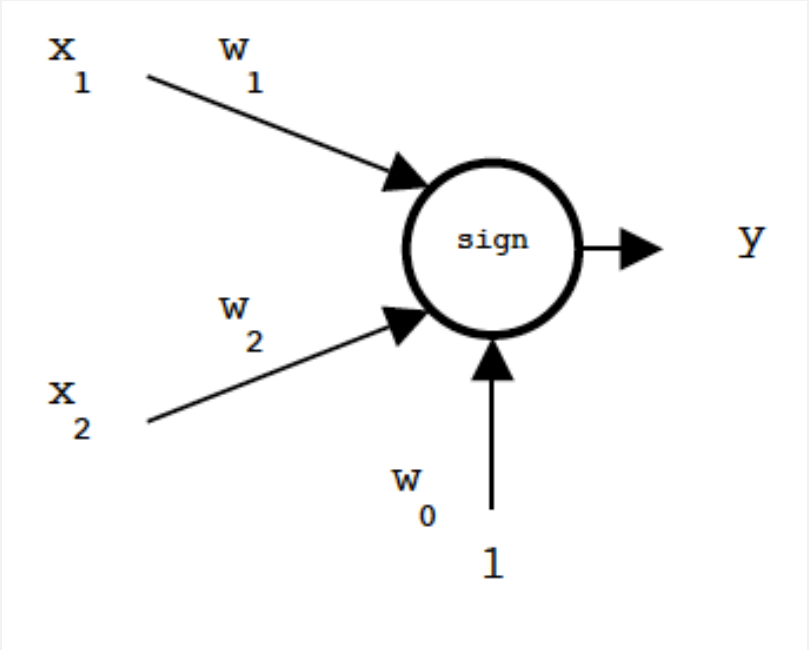
LOGISTIC REGRESSION

PERCEPTRON AS LOGICAL OPERATOR

- A perceptron can implement the AND operator

x_1	x_2	y
0	0	−1
0	1	−1
1	0	−1
1	1	1

- Example:
  - $W_1 = 3/2$
  - $W_2 = 1$
  - $W_0 = -2$

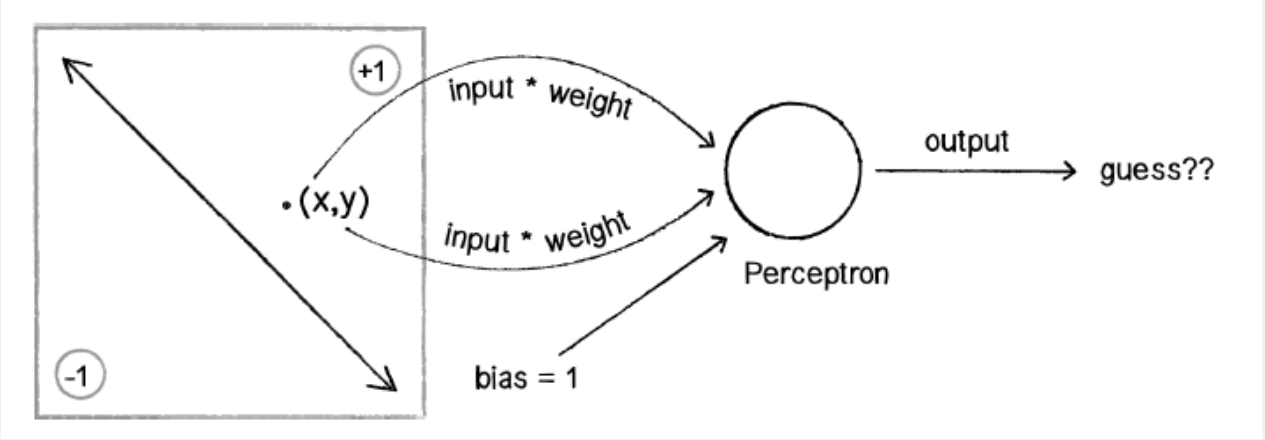


- A perceptron can implement the OR operator
- Example
  - $W_1 = 1$
  - $W_2 = 1$
  - $W_0 = -1/2$

x_1	x_2	y
0	0	−1
0	1	1
1	0	1
1	1	1

PERCEPTRON – CLASSIFICATION

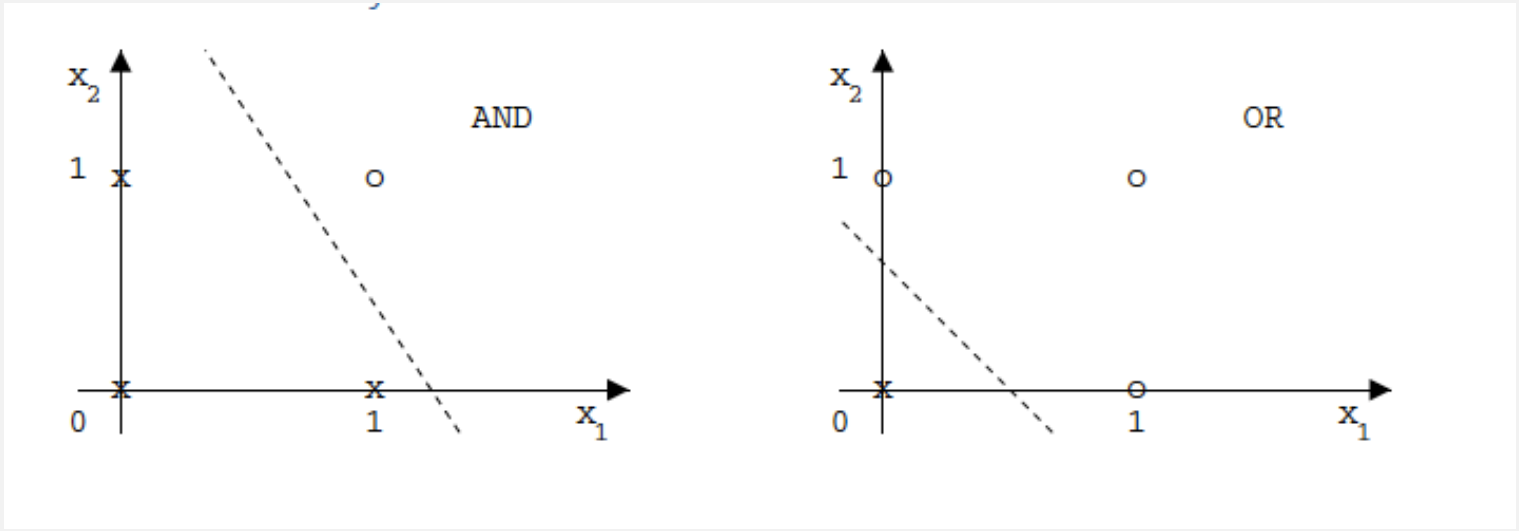
- Let's imagine the output as a class in the classification problem
- Let's think to have a line as decision boundary



$$0 = x_1 w_1 + x_2 w_2 + w_0$$

$$x_2 w_2 = -x_1 w_1 - w_0$$

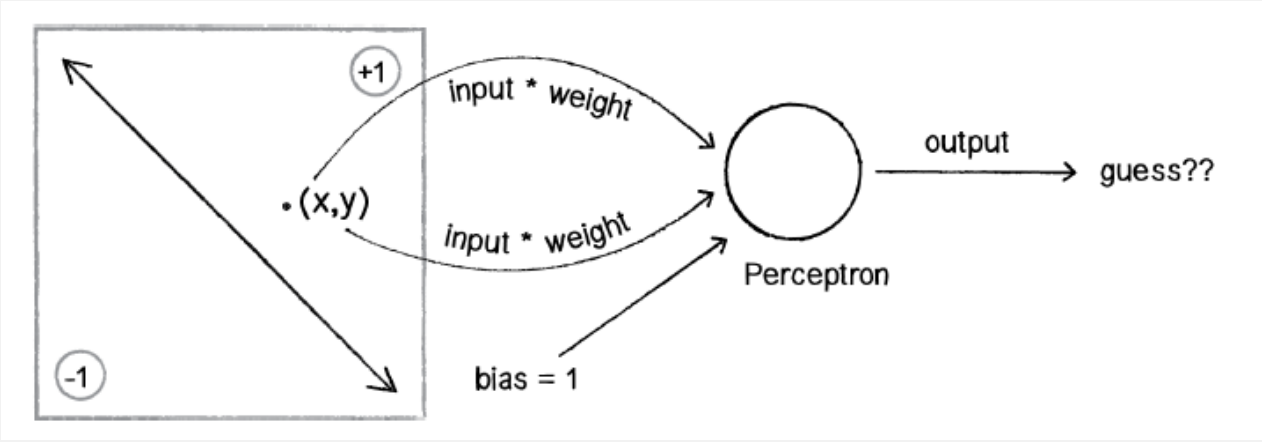
$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$



- The classification problem is to find the weights for the best linear decision boundary

PERCEPTRON – CLASSIFICATION

- Generalize the classification problem:
  - Give a set of labelled data (training data)
  - Find the best boundary that discriminate the two classes -> **find the best weights**

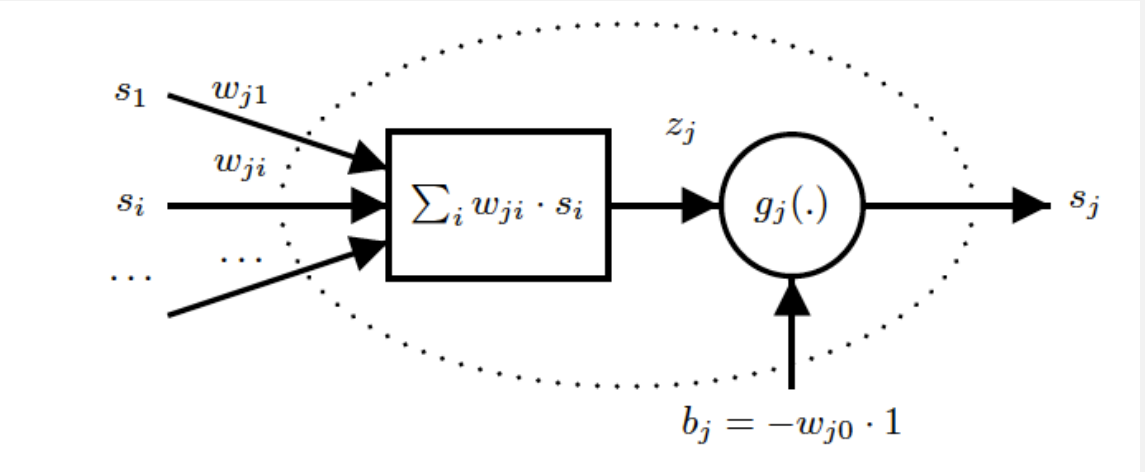


- **The perceptron learn from mistakes and update the weights accordingly**:
  - Provide the perceptron with inputs for which there is a known answer
  - Ask the perceptron to guess an answer
  - Compute the error: Did it get the answer right or wrong?
  - Adjust all the weights according to the error (tuning)
  - Return to Step 1 and repeat!

PERCEPTRON – CLASSIFICATION

- **Provide the perceptron with inputs for which there is a known answer ->** Activation value give a w_{ij} (there are some initial weights)

$$z_{ij} = \sum_i w_{ij} s_i$$



- In our case x1 = s1 and x2 = s2 and the initial weights are randomly generated

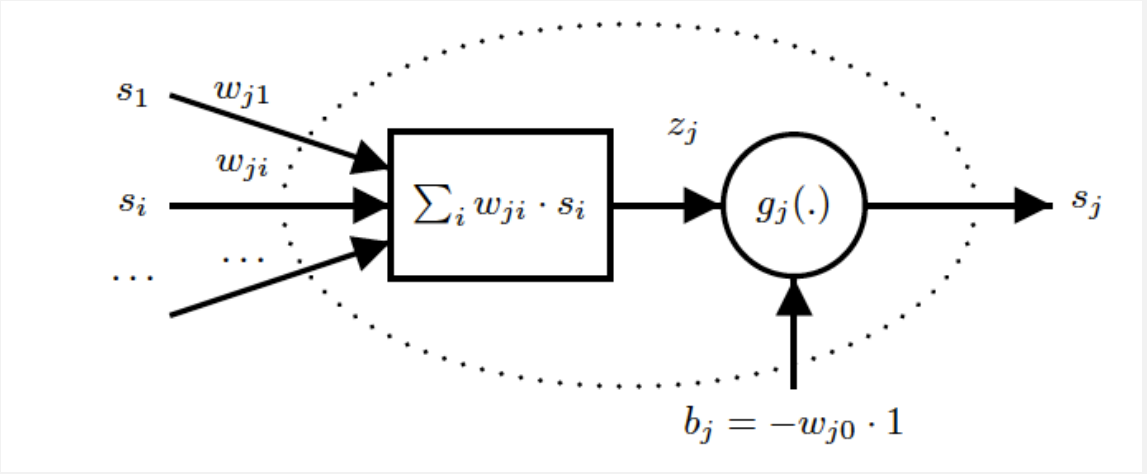
- **Ask the perceptron to guess an answer**

$$s_j = g_j(\sum_{i=1}^{N} w_{ij} s_i - b_j) = g_j(\sum_{i=0}^{N} w_{ij} s_i)$$

PERCEPTRON – CLASSIFICATION

- **Ask the perceptron to guess an answer**

$$s_j = g_j(\sum_{i=1}^{N} w_{ij} s_i - b_j) = g_j(\sum_{i=0}^{N} w_{ij} s_i)$$



- In our case, since the output is only -1 o 1, $g_j(z_j)$ can be

$$\text{Step function: } g(z_j) = \begin{cases} 1 & \text{if } z_j > 0 \\ -1 & \text{if } z_j = 0 \\ -1 & \text{if } z_j < 0 \end{cases}$$

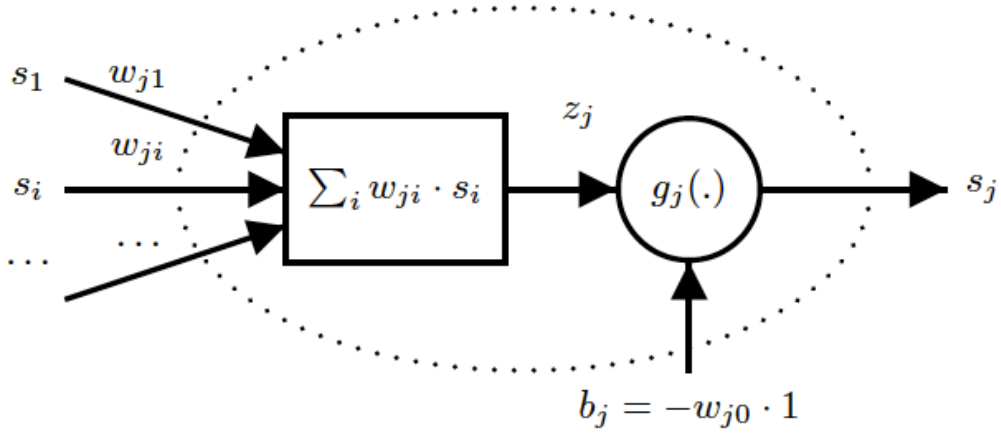
PERCEPTRON – CLASSIFICATION

- **Compute the error.** (Did it get the answer right or wrong?)

$$\epsilon = \hat{s}_j - s_i$$

$\hat{s}_j$ is the desired out s_j is the guess out

- If the output has only two possible values: +1 or -1



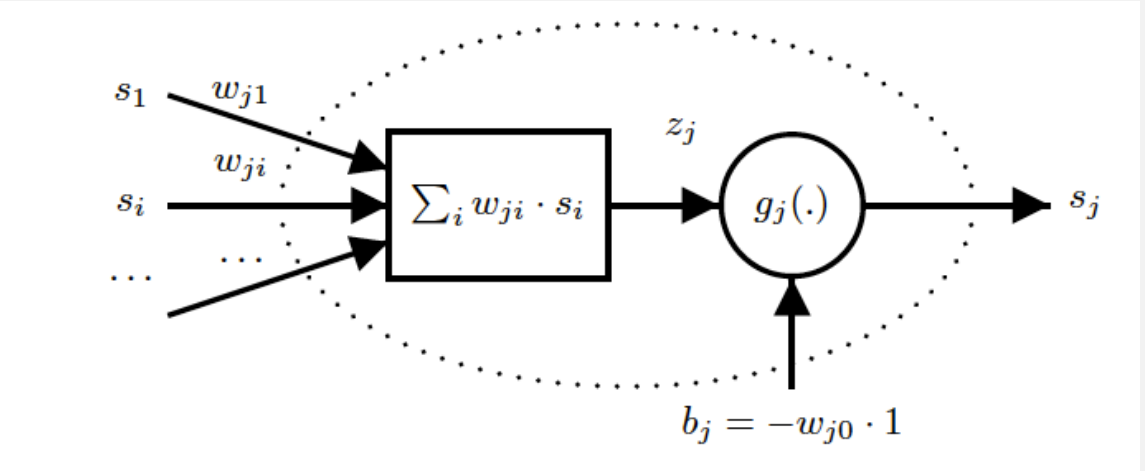
Desired	Guess	Error
-1	-1	0
-1	+1	-2
+1	-1	+2
+1	+1	0

PERCEPTRON – CLASSIFICATION

- **Adjust all the weights according to the error**

$$w_i^* = w_i + \Delta w_i$$
$$\Delta w_i = \epsilon * s_i$$

$$w_i^* = w_i + \epsilon * s_i$$



In order to the changing rate of weights –> add the learning rate

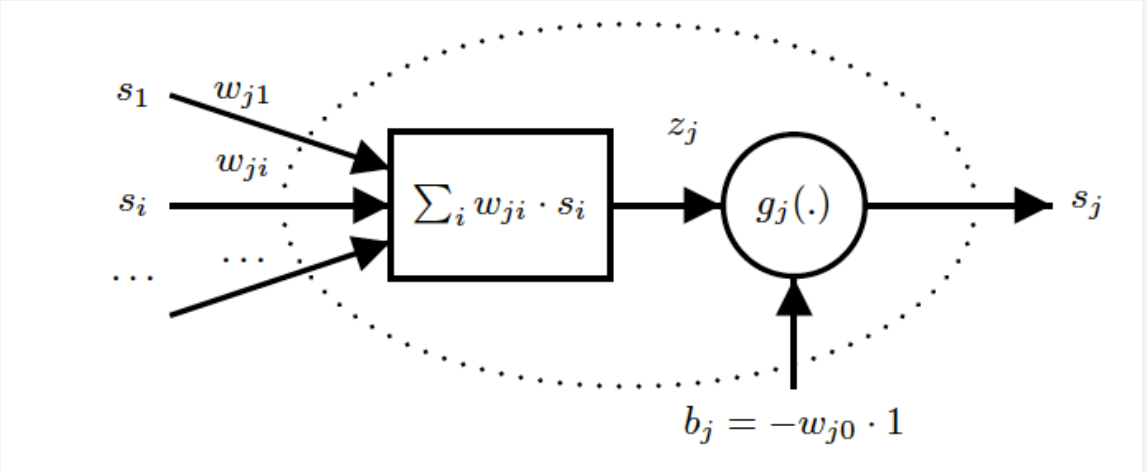
$$w_i^* = w_i + \epsilon * s_i * \eta$$

Trade off between how quick the neurol converges and how accurate will be the weights and the final result

PERCEPTRON – HEBB LEARNING RULE

- Hebb Learning rule is an alternative approach to weight updating
- Based of physiology models

- **"The strength of a synapse increase according to the simultaneous activation of the relative input and the desired target"**



$$w_i^{n+1} = w_i^n + \Delta w_i$$

$$\Delta w_i = \eta * \hat{s}_j * s_i$$

The two methods are equivalent

$\hat{s}_j$ is the desired out

N is the **epoch**

SimplePerceptron

SeekingNeural

LOGISTIC REGRESSION

- Lets generalize …

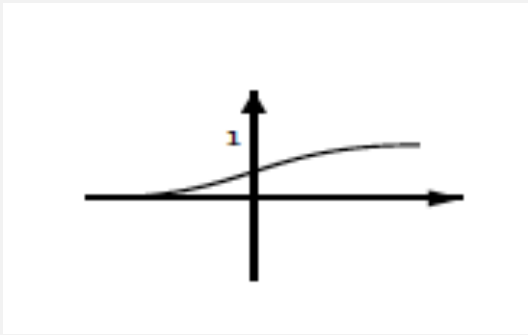
- **Logistic regression** is a regression analysis to conduct when the dependent variable is dichotomous (binary).
- Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more independent variables.
- For this reason, is used for **binary classification**

- Give a feature vector x, binary classification is to find $\hat{y} = P(y = 1|x)$
- Let's use a linear separation manifold $\hat{y} = \omega^T x + b$
- But is not a probability: need a function to transform $\hat{y}$ in the range [0,1]
- Application of an activation function g $\hat{y} = g(\omega^T x + b)$

ACTIVATION FUNCTIONS

Application of an activation function g $\hat{y} = g(\omega^T x + b)$

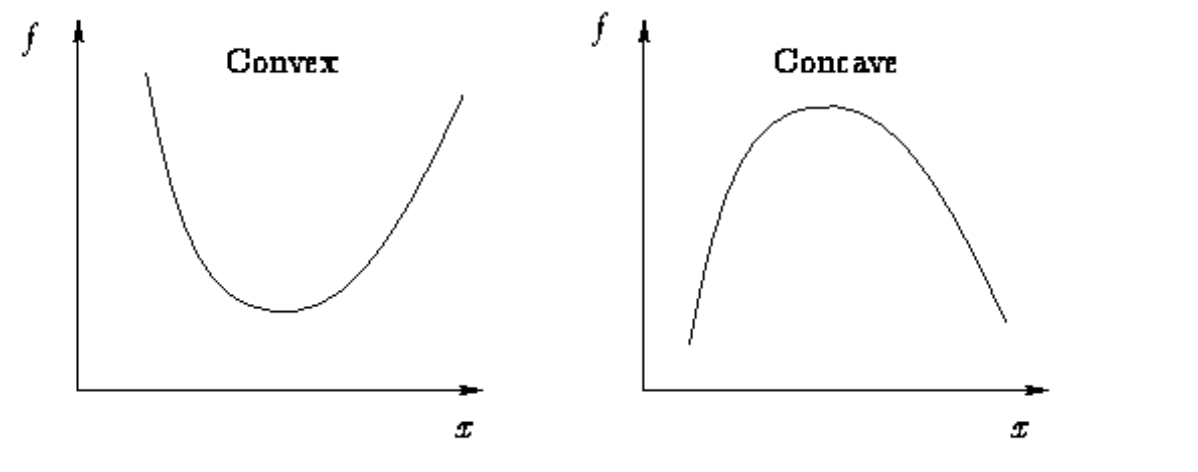
Sigmoid $g(z_j) = \frac{1}{1 + e^{K * z_j}} \quad K < 0$



Training a Logistic regression is learn ω and b such that $\hat{y}$ is a good estimate of $P(y = 1|x)$

LOGISTIC REGRESSION – LOSS FUNCTION

- In order to learn ω and b we need a measure of the error $L(\hat{y}, y)$
- The learning process should minimize $L(\hat{y}, y)$
- Hence the error function should be a convex function such that can converge



$$L(\hat{y}, y) = -(ylog\hat{y} + (1-y)log(1-\hat{y}))$$

LOGISTIC REGRESSION – LOSS FUNCTION

$$L(\hat{y}, y) = -(ylog\hat{y} + (1-y)log(1-\hat{y}))$$

- Example
  - If y =1 -> $L(\hat{y}, y) = -log\hat{y}$ In order to minimize L, $\hat{y}$ should be larger that, in sigmoid function, means 1
  - If y = 0 -> $L(\hat{y}, y) = -log(1-\hat{y})$ In order to minimize L, $\hat{y}$ should be smaller that , in sigmoid function, means 0

LOGISTIC REGRESSION – COST FUNCTION

- The loss function is related to a single feature vector x (single example)
- The **cost function** is related to the entire data set

- Given $x^{(i)}$ the i-th feature vector and $y^{(i)}$ its Ground Truth

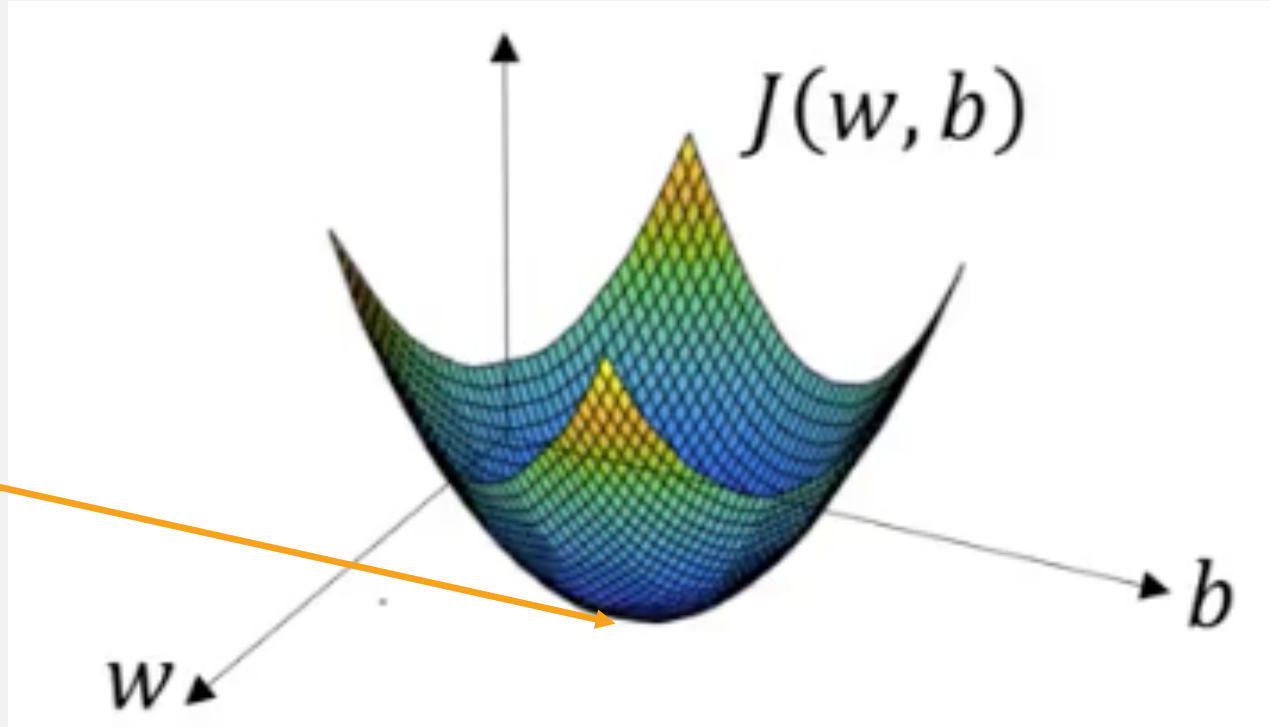
$$J(\omega, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)} log\hat{y}^{(i)} + (1 - y^{(i)})log(1 - \hat{y}^{(i)}))$$

LOGISTIC REGRESSION – BACK PROPAGATION

- Given the cost function, in the training we need a procedure to tune ω and b

to **minimize** $J(\omega, b)$

minimum = global optimum



- The procedure is called **Back Propagation** and **Gradient Descent** is the algorithm we use

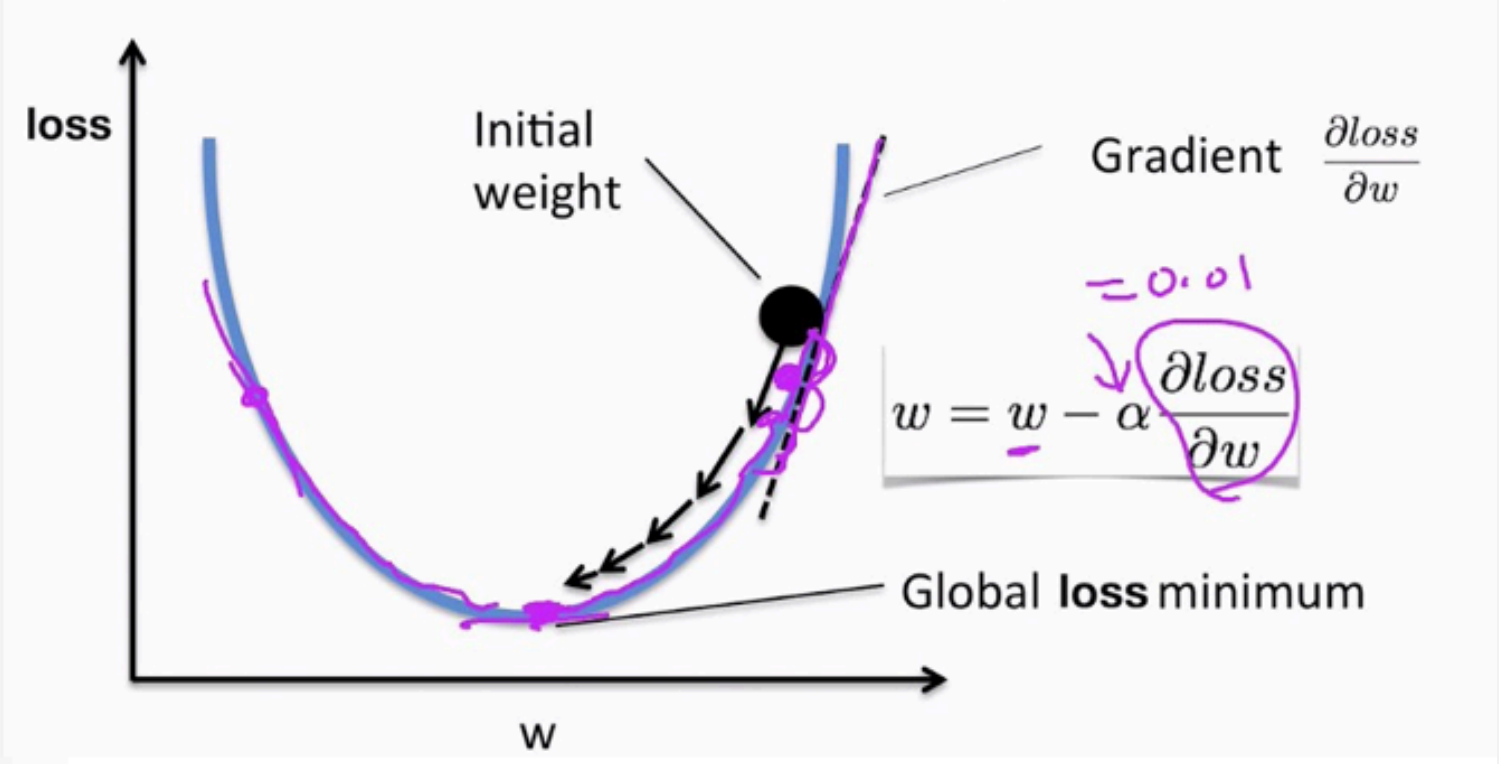
LOGISTIC REGRESSION – BACK PROPAGATION

- Example for ω
- Set a initial random ω
- Compute the gradient

$$G = \frac{dJ(\omega)}{d\omega}$$

- Update ω

$$\omega = \omega - \alpha \frac{dJ(\omega)}{d\omega}$$

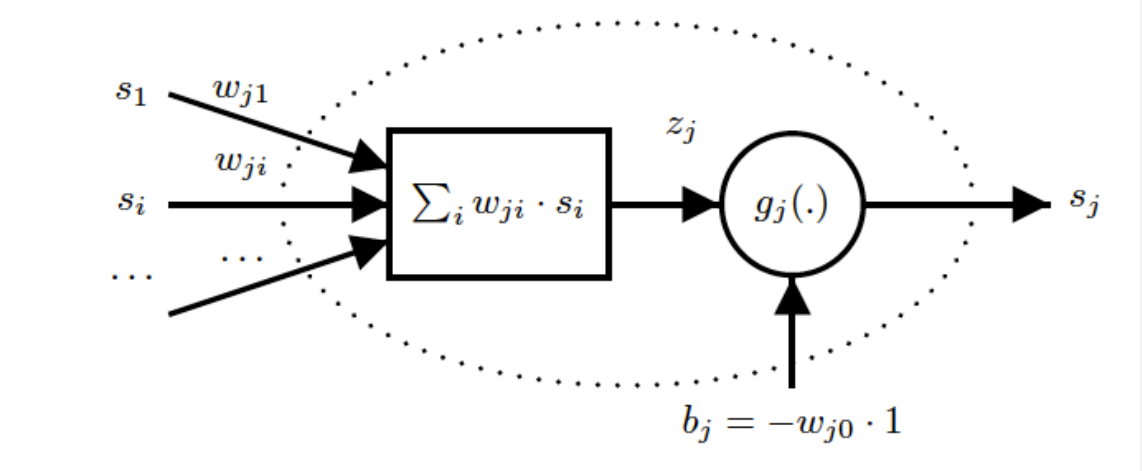


If the initial ω is a the right side of the curve -> G >0 -> ω decrease -> descending in the curve

If the initial ω is a the left side of the curve -> G <0 -> ω increase -> descending in the curve

α = learning rate (linked to size of the descending steps)

LOGISTIC REGRESSION – SUMMARY



Main steps:

- Define the model structure (such as number of input features)
- Initialize the model's parameters
- Loop:
  - Calculate current cost (forward propagation)
  - Calculate current gradient (backward propagation)
  - Update parameters (gradient descent)

LOGISTIC REGRESSION – SUMMARY

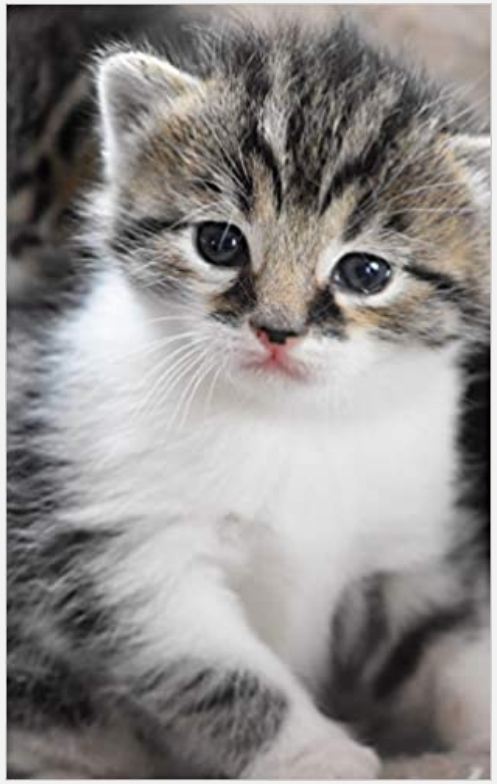
- To summarize:
  - Given a **training set** of feature vector (examples) $(x^{(i)}, y^{(i)}) \forall i \in [1..N]$
  - The **prediction** is given by $\hat{y} = g(\omega^T x + b)$
  - Where **the activation function** is $g(z_j) = \frac{1}{1 + e^{K * z_j}} \quad K < 0$
  - The learning process is given by **cost function** and the **back propagation** procedure

$$J(\omega, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} log \hat{y}^{(i)} + (1 - y^{(i)}) log(1 - \hat{y}^{(i)}))$$

$$\omega = \omega - \alpha \frac{dJ(\omega)}{d\omega} \qquad b = b - \alpha \frac{dJ(b)}{db}$$

IMAGE CLASSIFICATION

- Classify if an image represents a cat or not



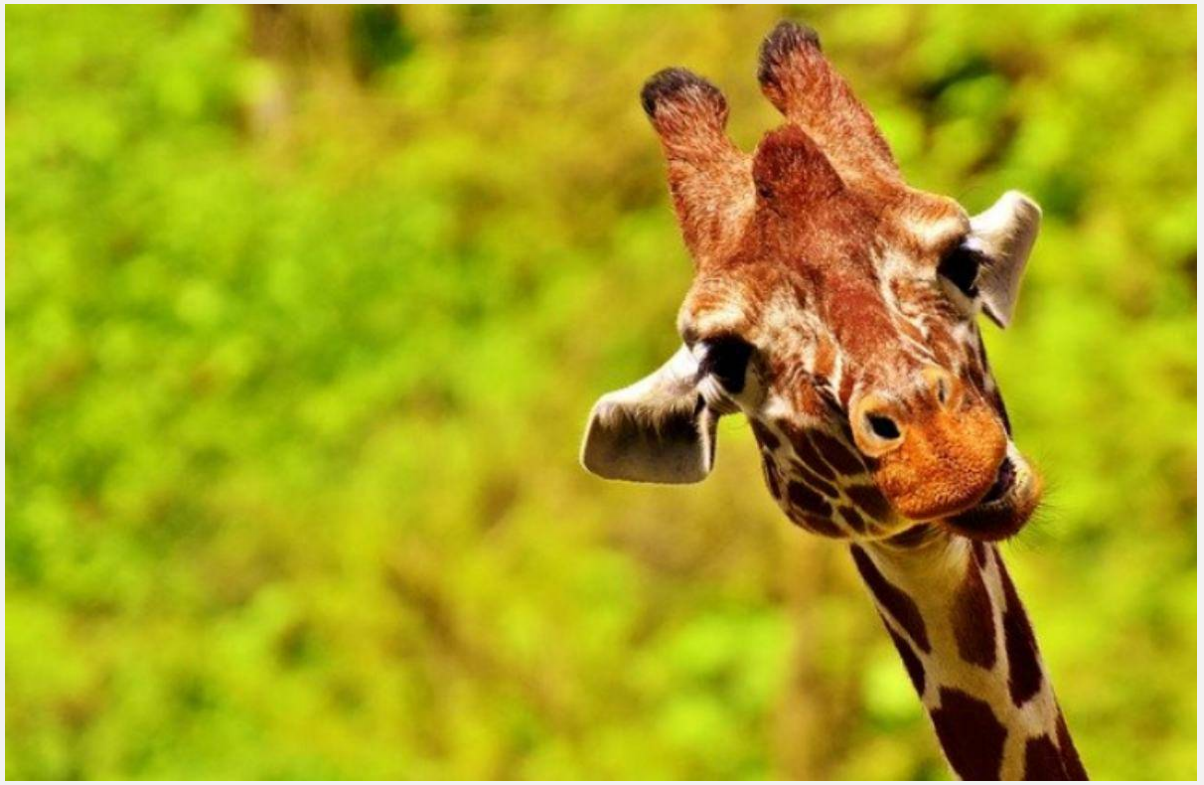
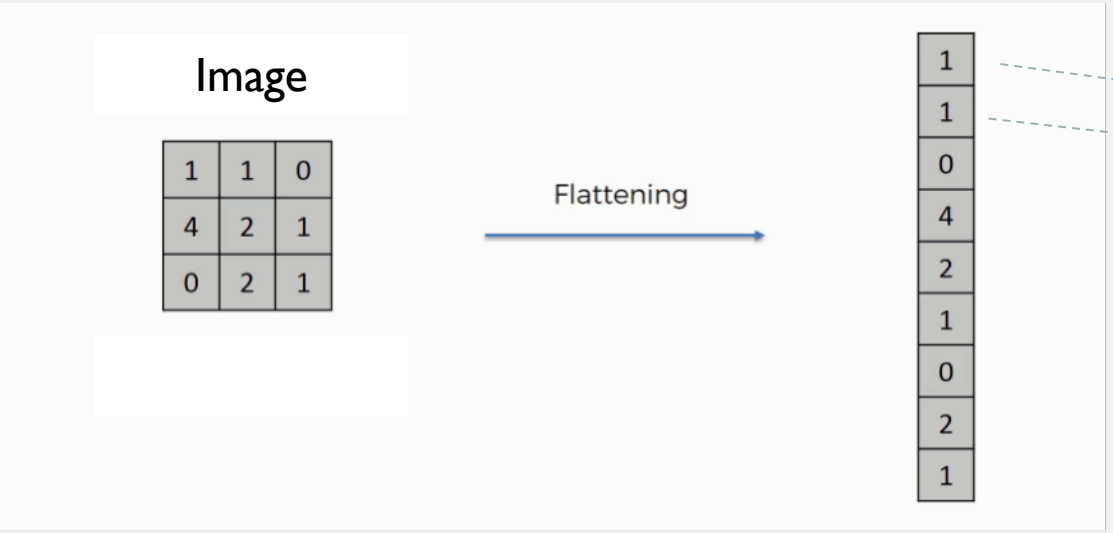


Image classification
logistic regression

IMAGE CLASSIFICATION

- **Input**



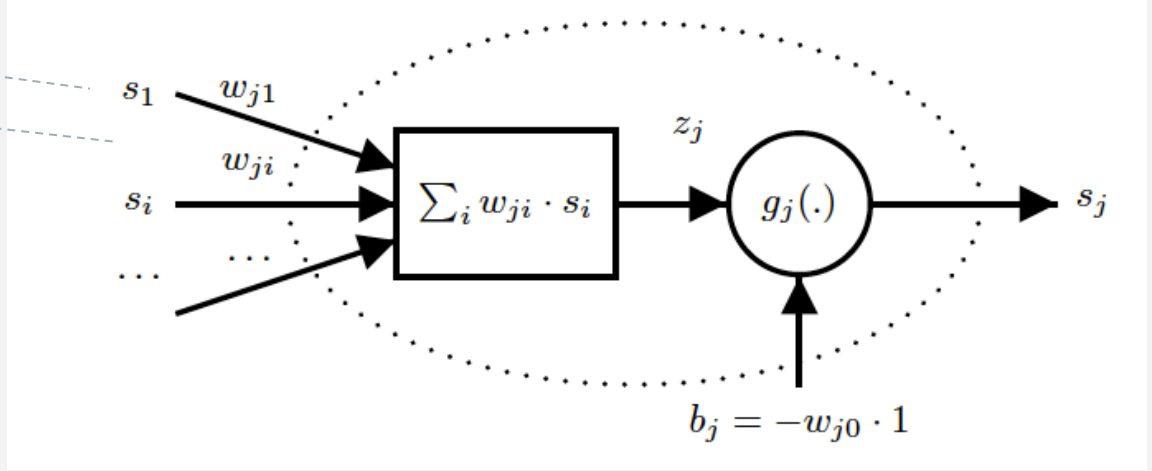
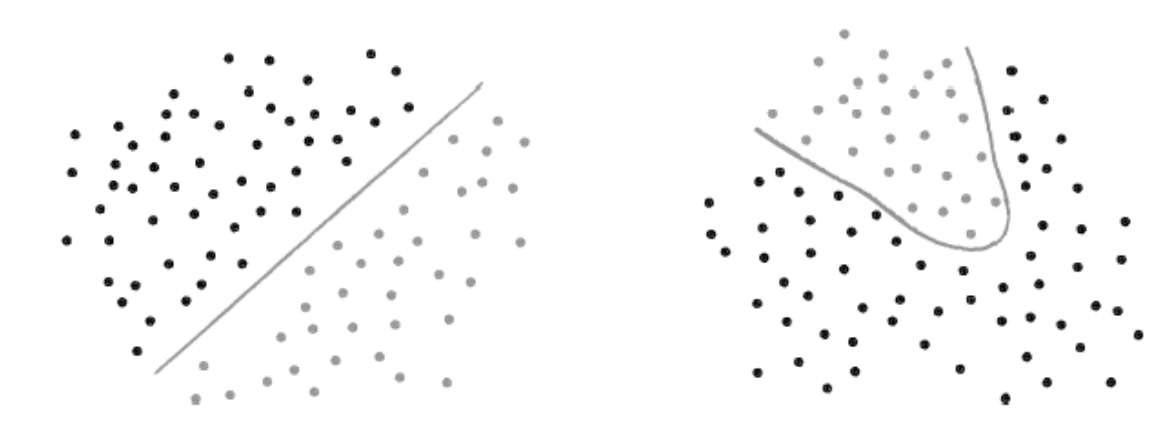


Image dimensions: 64x64x3 → Flattened version dimension: 12288x1

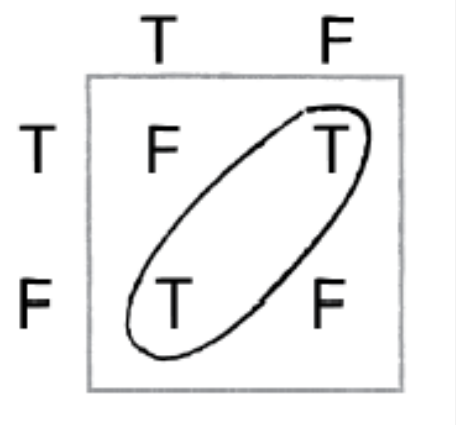
Image classification
logistic regression

PERCEPTRON – THE XOR PROBLEM

What if the space is not linearly separable ?



This called the XOR problem

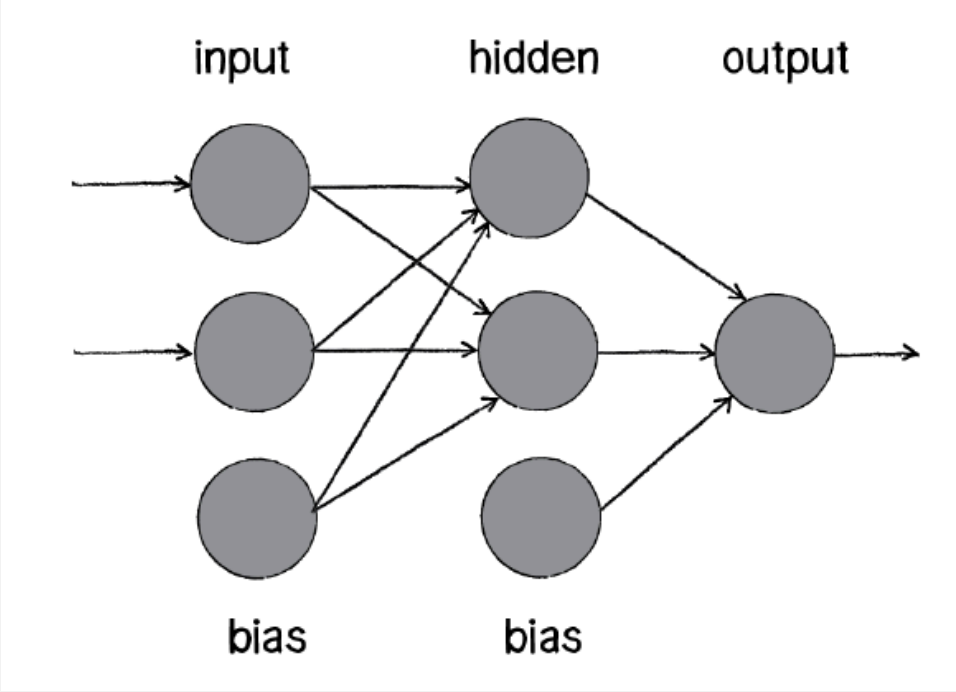


Needs of a more complex networks

MULTILAYER PERCEPTRON

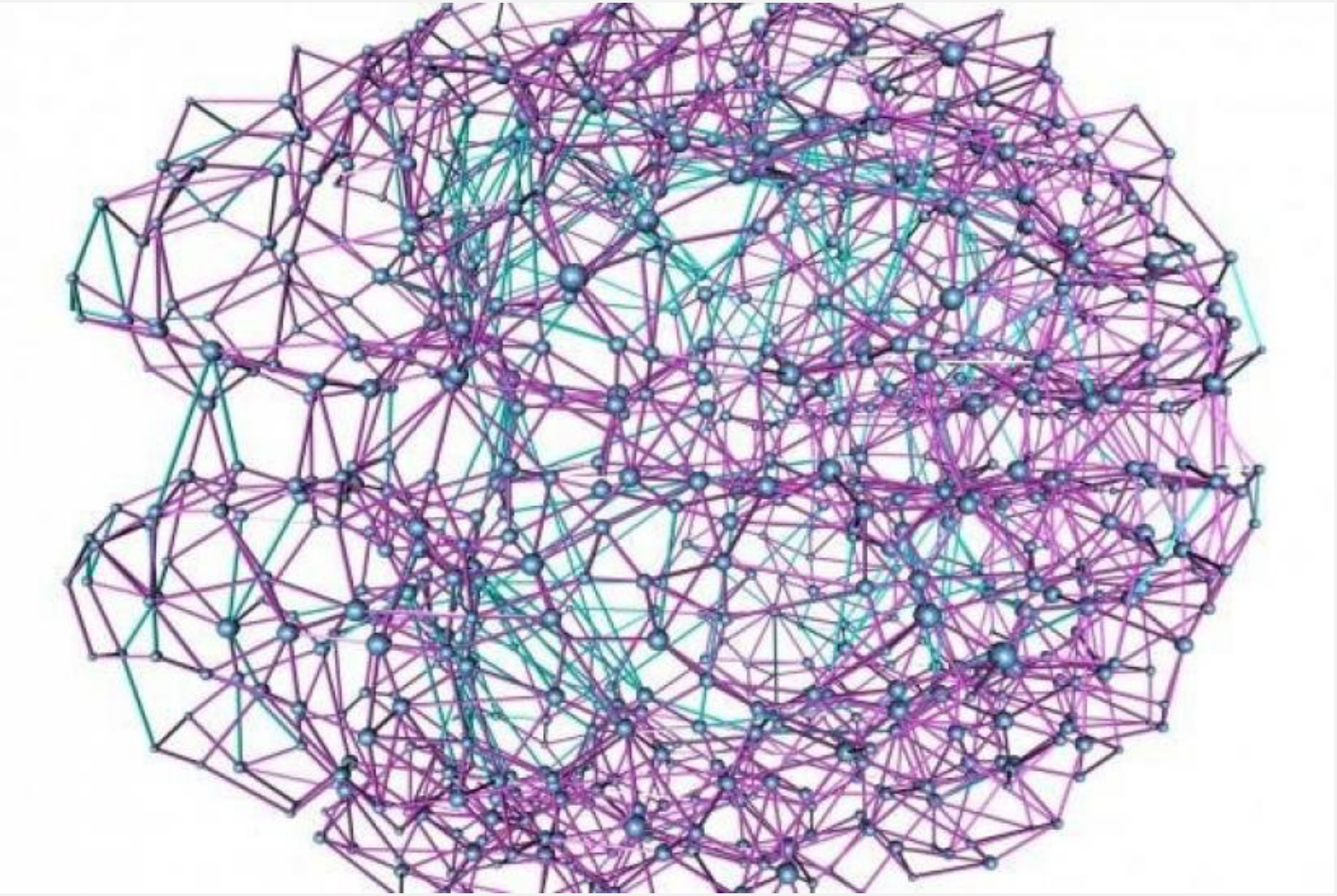
MULTILAYER PERCEPTRON

- Connect more neurons in a structured network (**multi-layered perceptron**)
- Composed by
  - input neurons and receive the inputs
  - a certain number of hidden layers -> perform the layered generalization of the input data
  - output neurons, from which to read the results



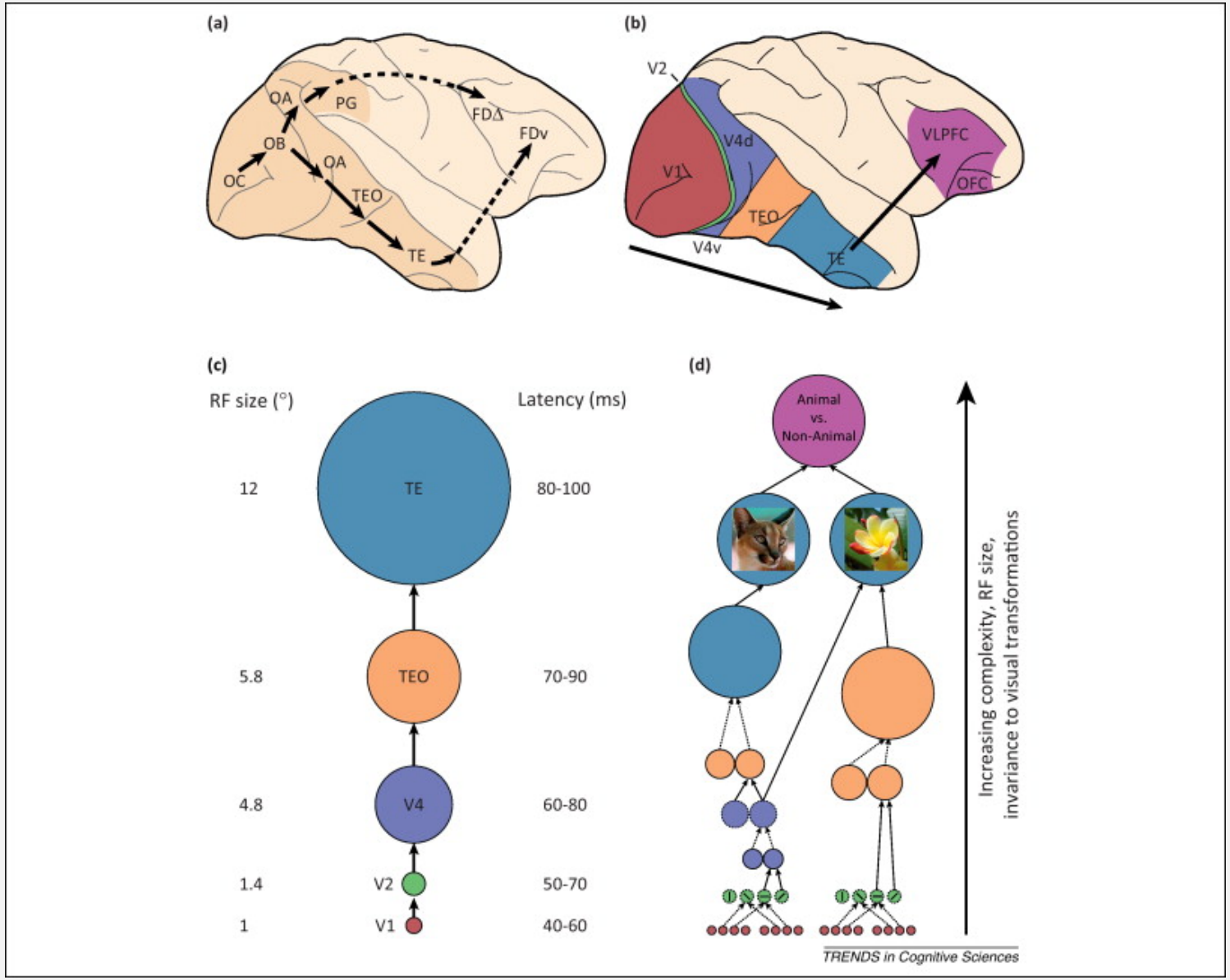
MULTILAYER PERCEPTRON

- Neural Network mimic the brain structure
- Brain is composed by a network of neurons



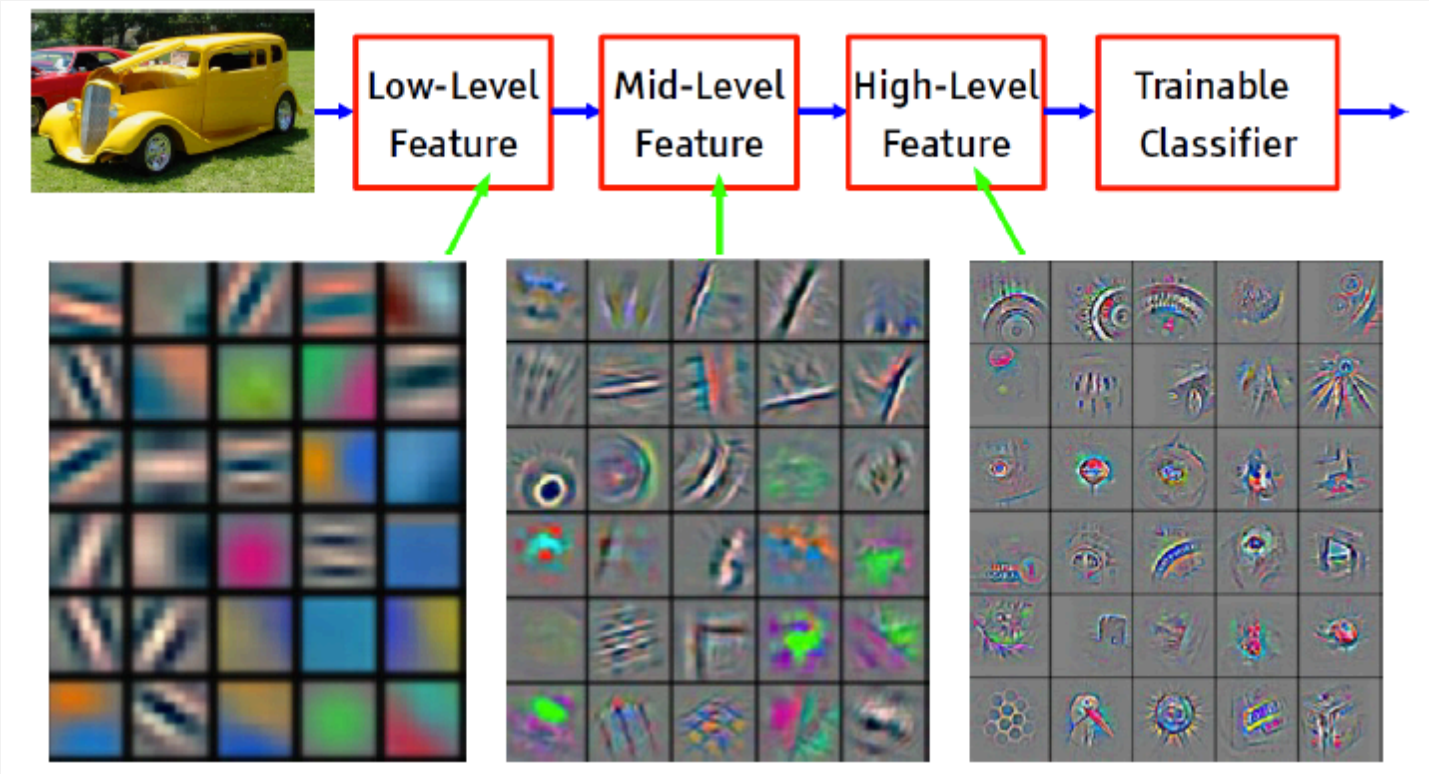
THE VENTRAL (RECOGNITION) PATHWAY IN THE VISUAL CORTEX

- The vision stimuli has a pathway in the brain
- Each node in the path has the ability to infer specific information
- The deeper is the path, the more abstract is the information

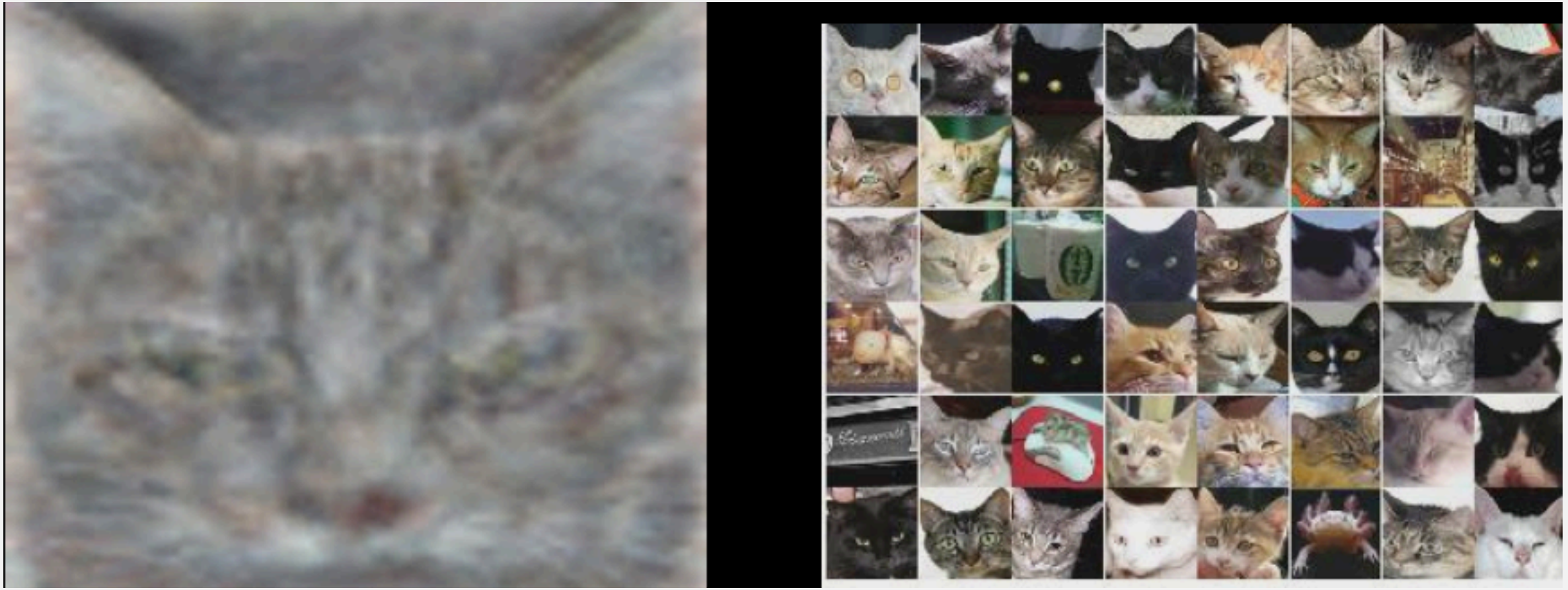


MULTILAYER PERCEPTRON

- Deep learning assumes it is possible to «learn» a hierarchy of descriptors with increasing abstraction, i.e., layers are trainable feature transforms



MULTILAYER PERCEPTRON



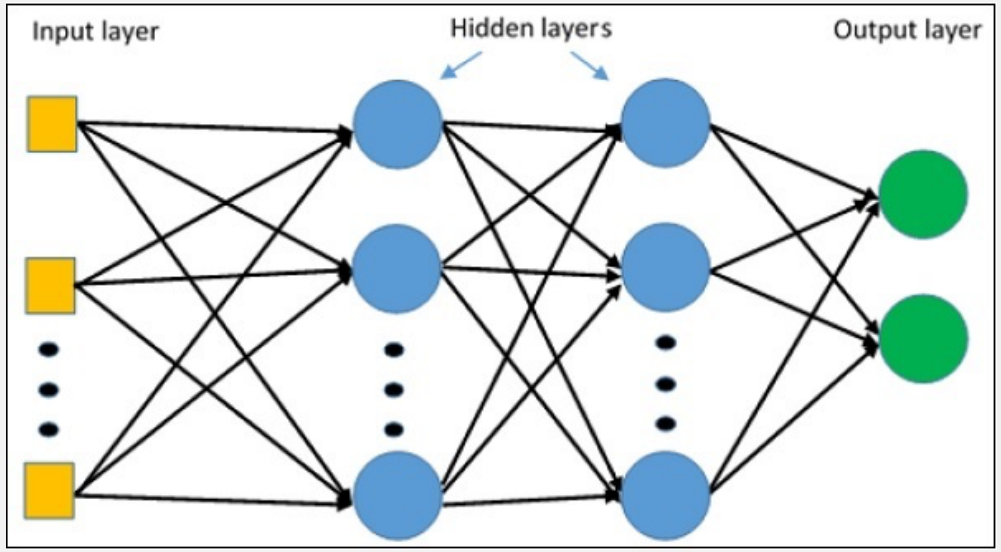
MULTILAYER PERCEPTRON

- In image recognition
  - Pixel → edge → texton → motif → part → object
- In text analysis
  - Character → word → word group → clause → sentence → story
- In speech recognition
  - Sample → spectral band → sound → phone → phoneme → word

MULTILAYER PERCEPTRON OR ARTIFICIAL NEURAL NET (ANN)

- ANN:
  - A set of neurons connected according to a **topology**
  - **Layer**: Neurons at the same distance from the input neurons
  - **Input Layer**: Layer of neurons that receives as input the data to process
  - **Output Layer**: Layer of neurons that gives the final result
  - **Hidden Layers**: Layers of neurons that process data from other neurons to be processed from other neurons

- The choice of the topology as well as the type of input data determine the effectiveness of the network



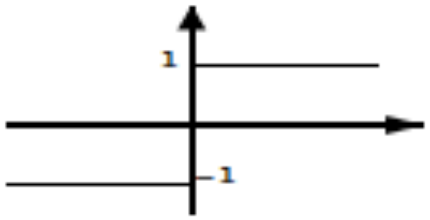
- A Multilayer perceptron is a non-linear system determined by:
  - number of neurons
  - their topology
  - activation functions
  - the values of synaptic weights and biases

ACTIVATION FUNCTIONS

- There exists other activation functions

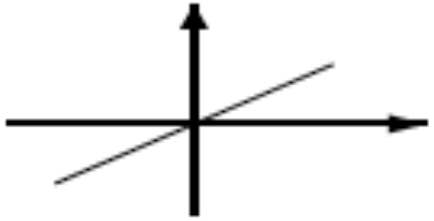
Step Function

$$g(z_j) = \begin{cases} 1 \text{ if } z_j > 0 \\ 0 \text{ if } z_j = 0 \\ 1 \text{ if } z_j < 0 \end{cases}$$



Linear Function

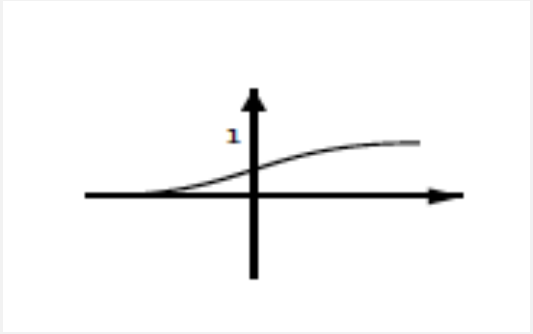
$$g(z_j) = K * z_j$$



ACTIVATION FUNCTIONS

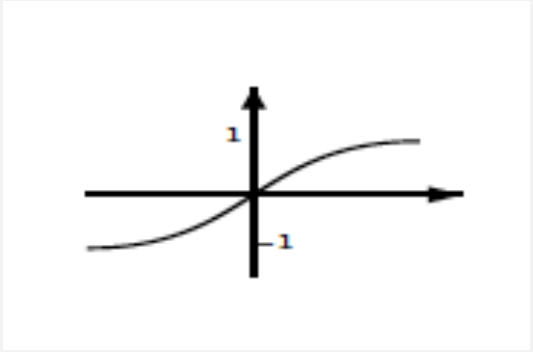
Sigmoid

$$g(z_j) = \frac{1}{1 + e^{K * z_j}} \quad K < 0$$



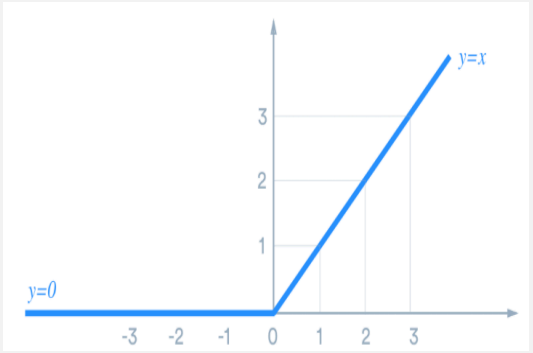
Hyperbolic

$$g(z_j) = \frac{e^{z_j} - e^{-z_j}}{e^{z_j} + e^{-z_j}}$$



ReLU (rectified)

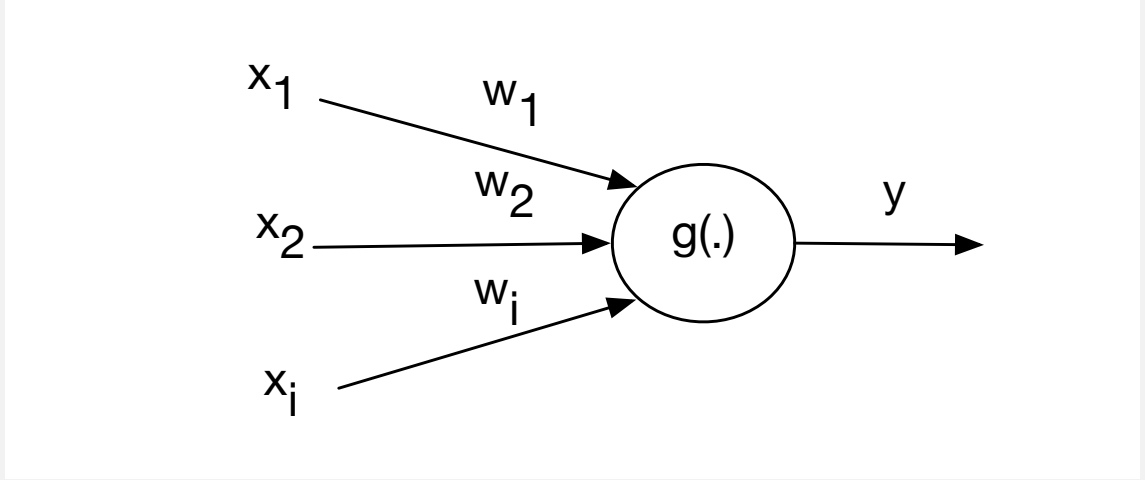
$$g(z_j) = \max(0, z_j)$$



MULTILAYER PERCEPTRON OR ARTIFICIAL NEURAL NET (ANN)

- For a single neuron

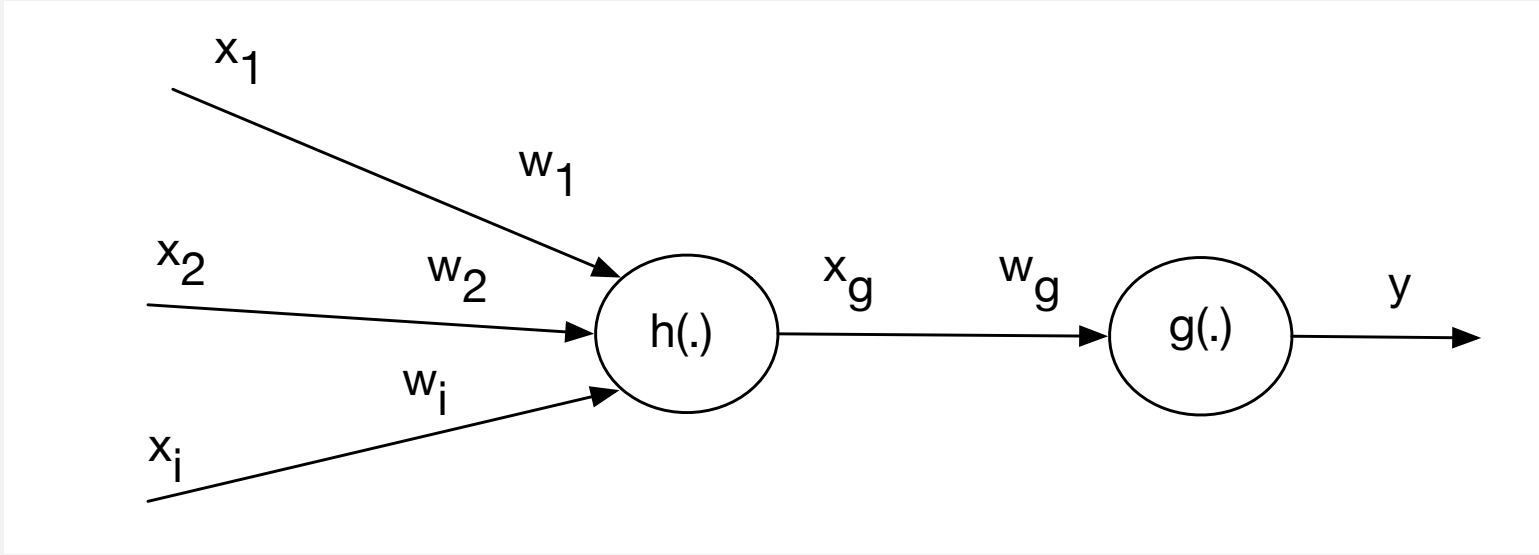
$$y = g(\sum_{i=0}^{I} \omega x_i)$$



- For linked neurons

$$y = g(\omega_g x_g)$$

$$y = g(\omega_g h(\sum_{i=0}^{I} \omega_i x_i)$$

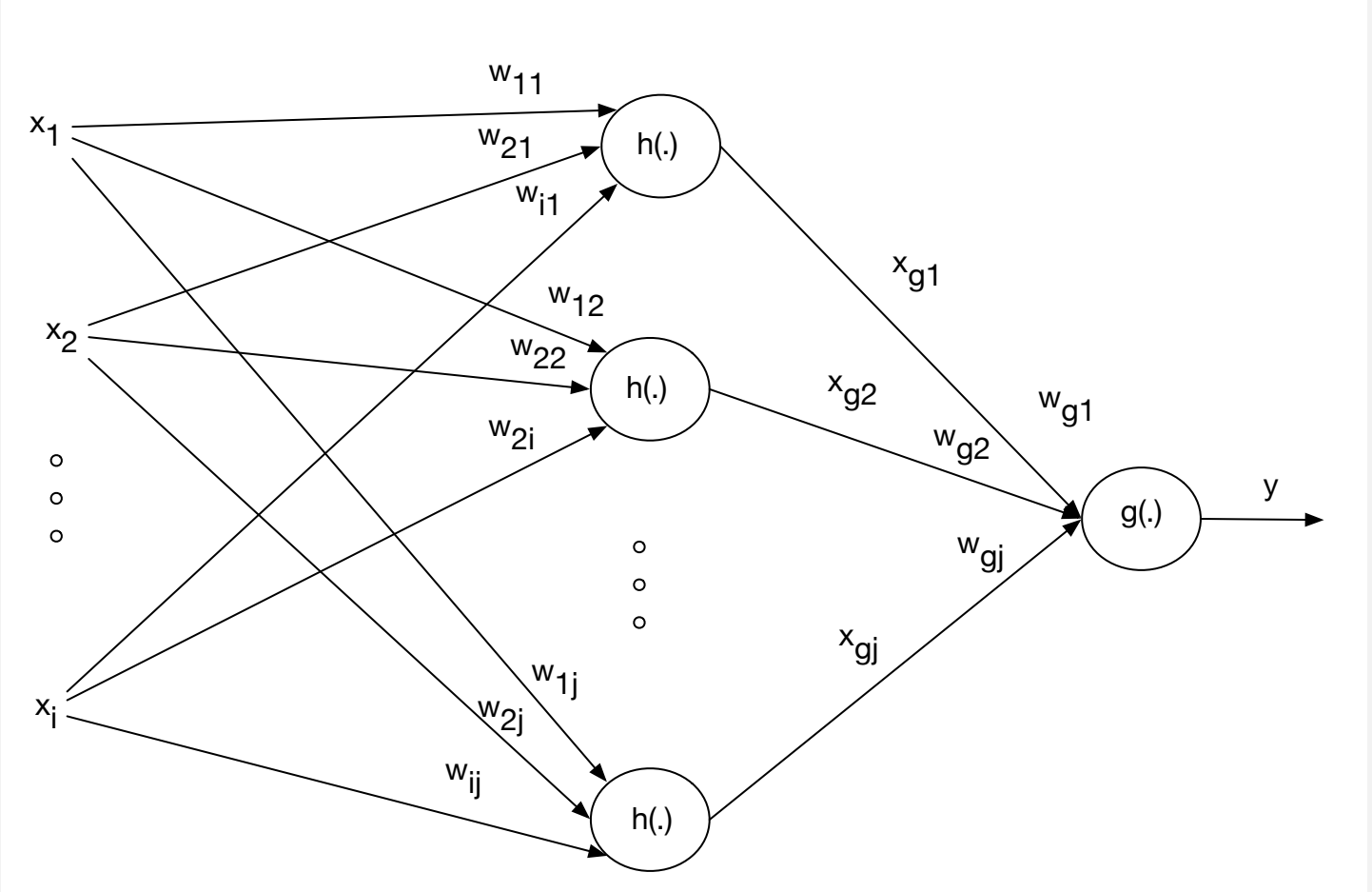


MULTILAYER PERCEPTRON OR ARTIFICIAL NEURAL NET (ANN)

- For a network with 1 hidden layer

$$y = g(\omega_g h(\sum_{i=0}^{I} \omega_i x_i)$$

$$y = g(\sum_{j=0}^{J} \omega_{gj} h(\sum_{i=0}^{I} \omega_{ij} x_i))$$



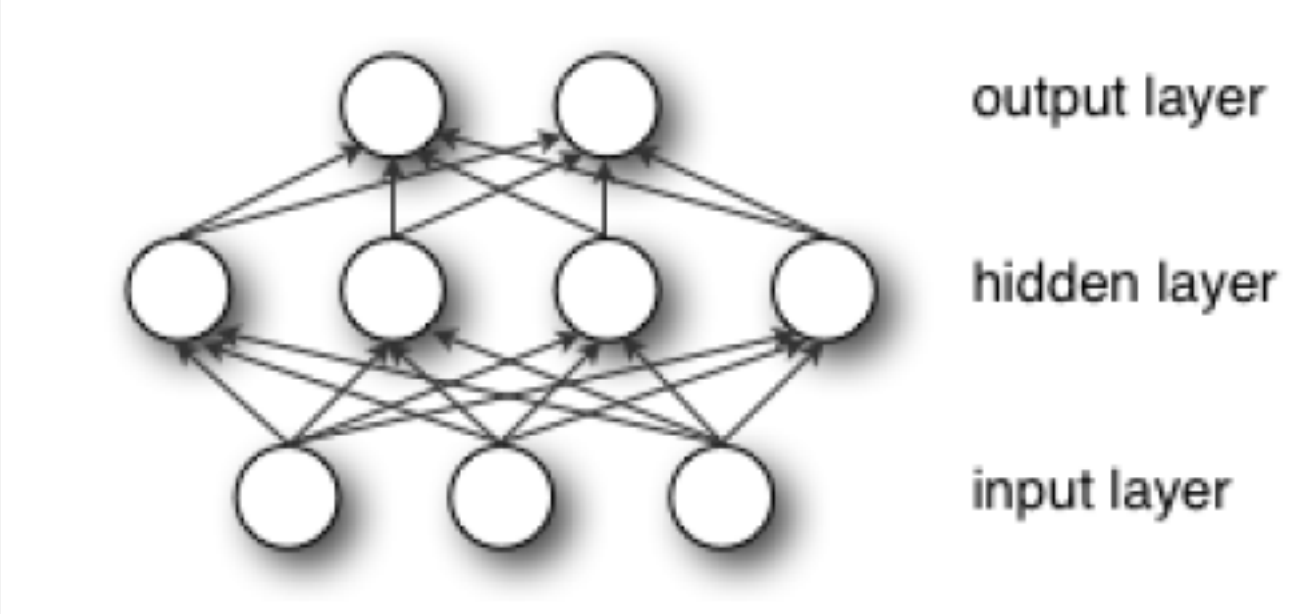
- Where
  - J is number of neurons in the hidden layer
  - I is the number of inputs
  - g() is the output activation function
  - h() if the activation function for hidden layers

MULTILAYER PERCEPTRON OR ARTIFICIAL NEURAL NET (ANN)

- In a more general formulation a **one-hidden-layer MLP** is a function

$$f : R^D \rightarrow R^L$$

- D = number of inputs
- L = number of outputs



$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x))),$$

$$h(x) = \Phi(x) = s(b^{(1)} + W^{(1)}x)$$

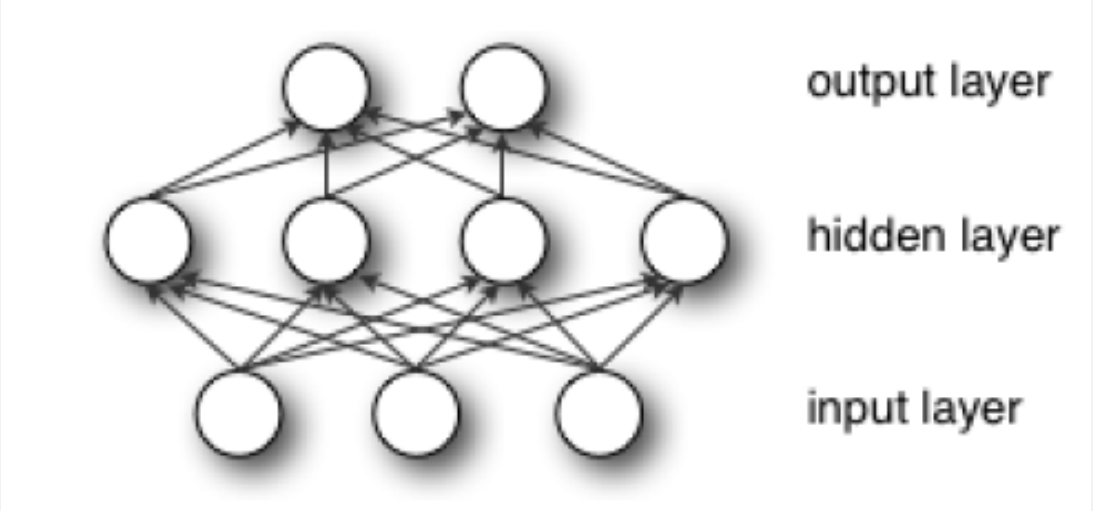
$W^{(1)} \in R^{D \mathrm{x} D_h}$ is the weight matrix connecting the input vector to the hidden layer

TYPICAL INPUT DATA

- Input of the MLP are the higher informative data we have to describe the elements

- Examples
- **Audio**: set of spectrums, set of spectrograms, set of waveforms with a certain windowing resolution
- **Image**: pixels, histogram-based representation

OUTPUT LAYER

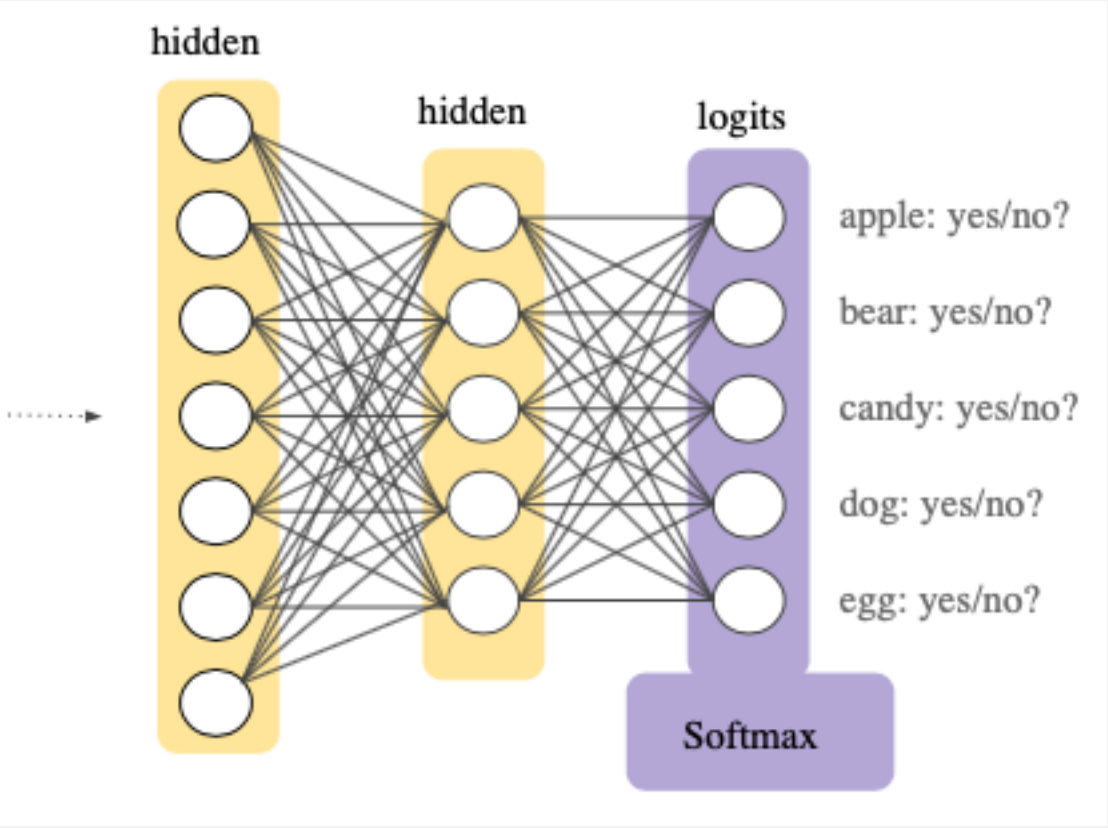


http://playground.tensorflow.org

- In the classification problem:
  - The number of neurons in the output layer is the number of classes
  - The activation value of neurons in the output layer are the probabilities the analysed data to belong the class
- In the regression problem:
  - The number of neurons in the output layer is the number of domains in the regression function
  - Activation values the activation value of neurons in the output layer are the regression values

OUTPUT LAYER

- **Softmax layer**
- Is a generalization of the logistic regression to multiple dimensions.
- The softmax function takes as input a vector *z* of *K* real numbers, and normalizes it into a probability distribution consisting of *K* probabilities proportional to the exponentials of the input numbers
- Softmax layer uses a sigmoid function

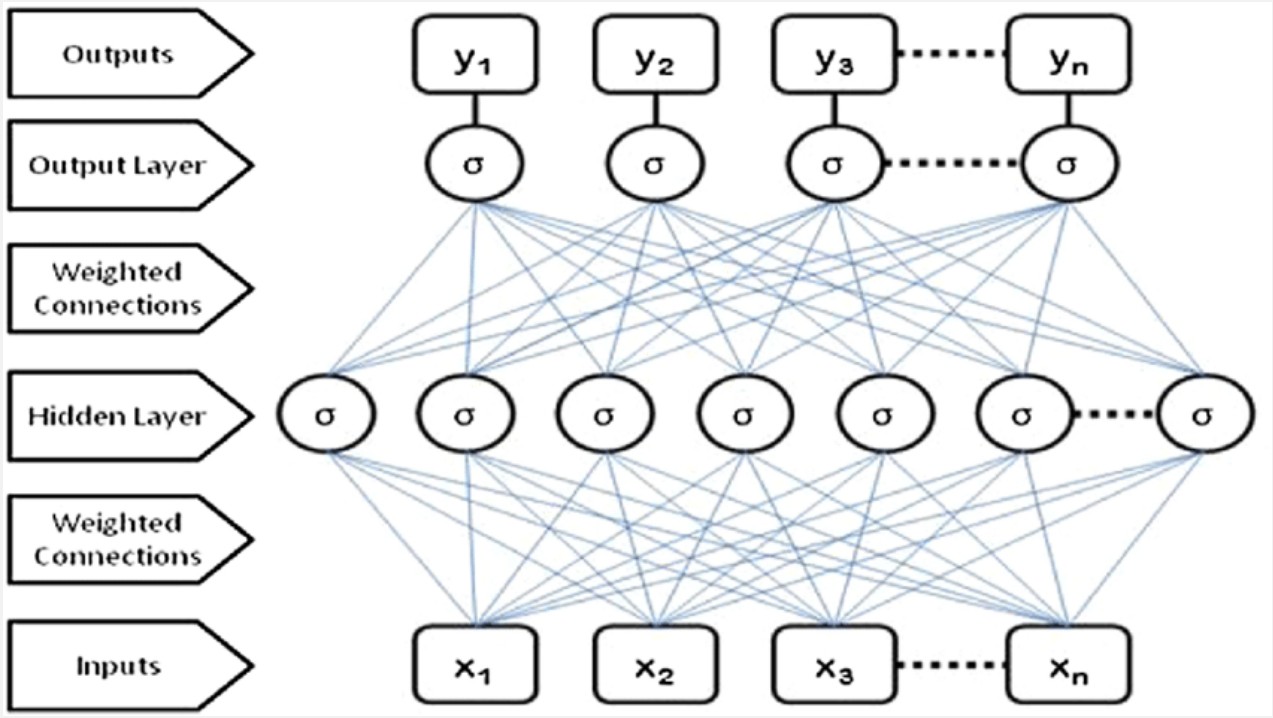


$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

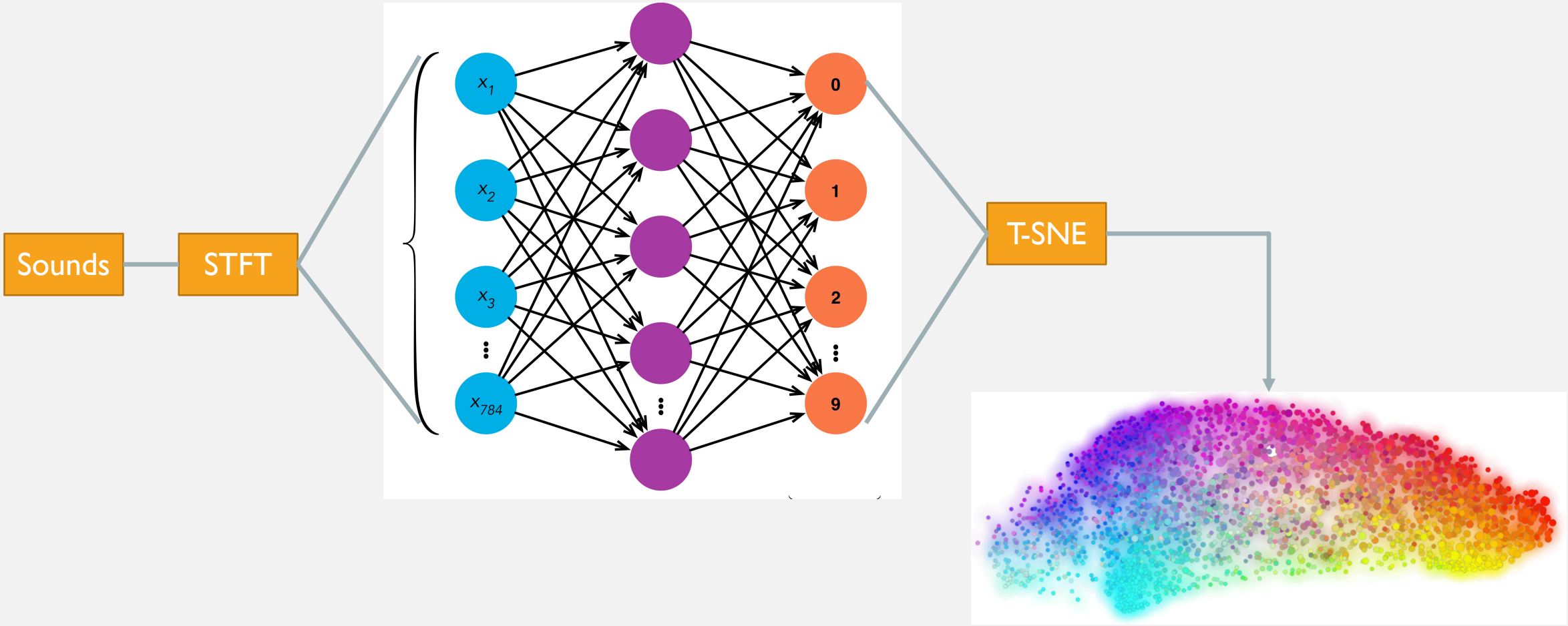
Fashion MNIST classification

OUTPUT LAYER

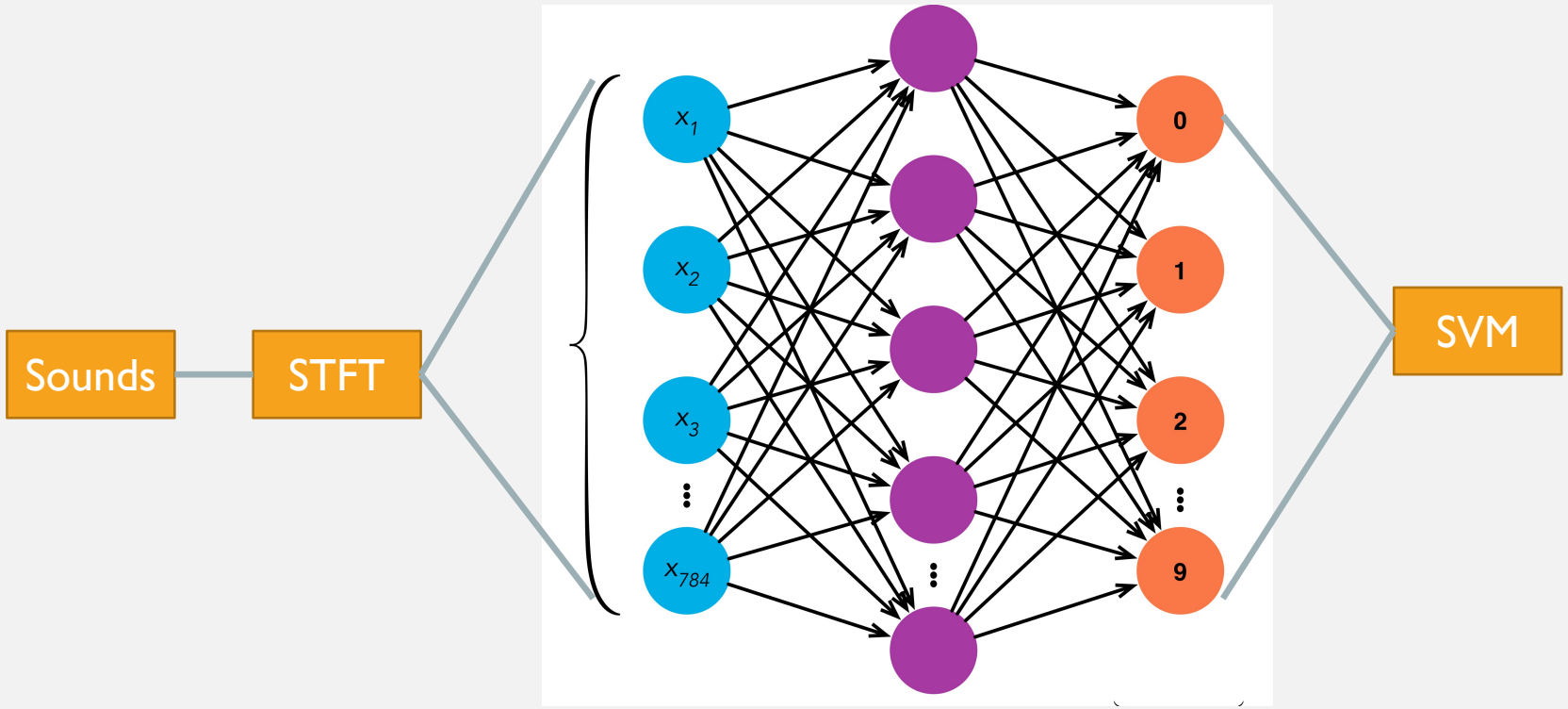
- Neural Networks learn some abstraction of the input
- In the case the output layer is not softmax, the activation value of single output neuron can be seen a **high-level features**
- They can be used in further applications



2D SOUND SPACE

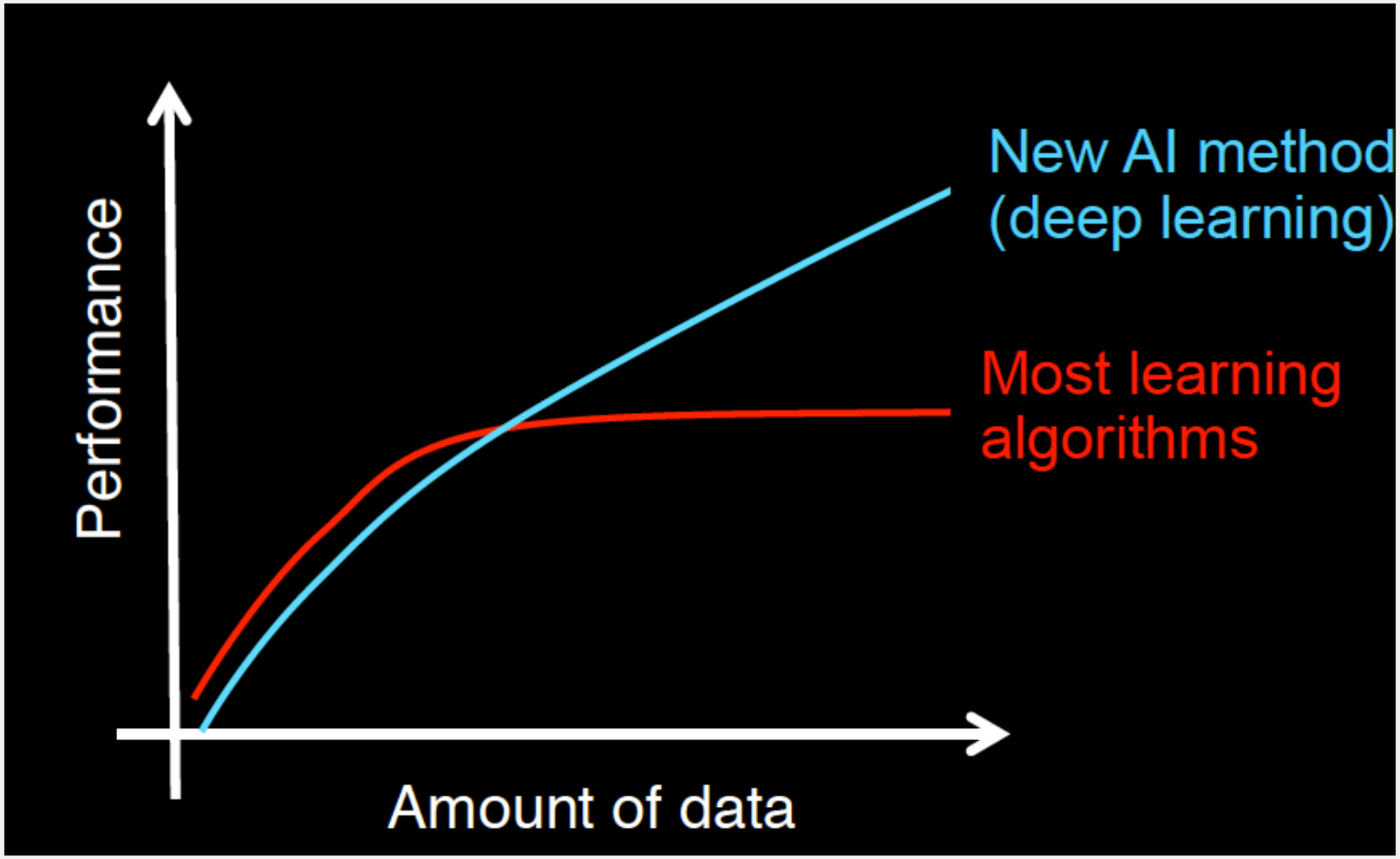


FEATURE GENERATION FOR SOUND CLASSIFICATION



WHY DEEP LEARNING ?

In all problems of classification and regression the use of Deep Learning can be very effective having a huge amount of data –> Higher accuracy with respect to classical machine learning approaches



COMPUTATION

- Deep Learning is very high computationally demanding

- GPU with hundreds of cores