Machine Learning *techniques* *for* Stellar Spectroscopy

Part 1

Lorenzo Spina

INAF - Astronomical Observatory of Padua (Italy)

spina.astro@gmail.com

@lorenzospina

Part 1

- What is Machine Learning?
- How does it work?
- Brief introduction to Artificial Neuronal Networks.

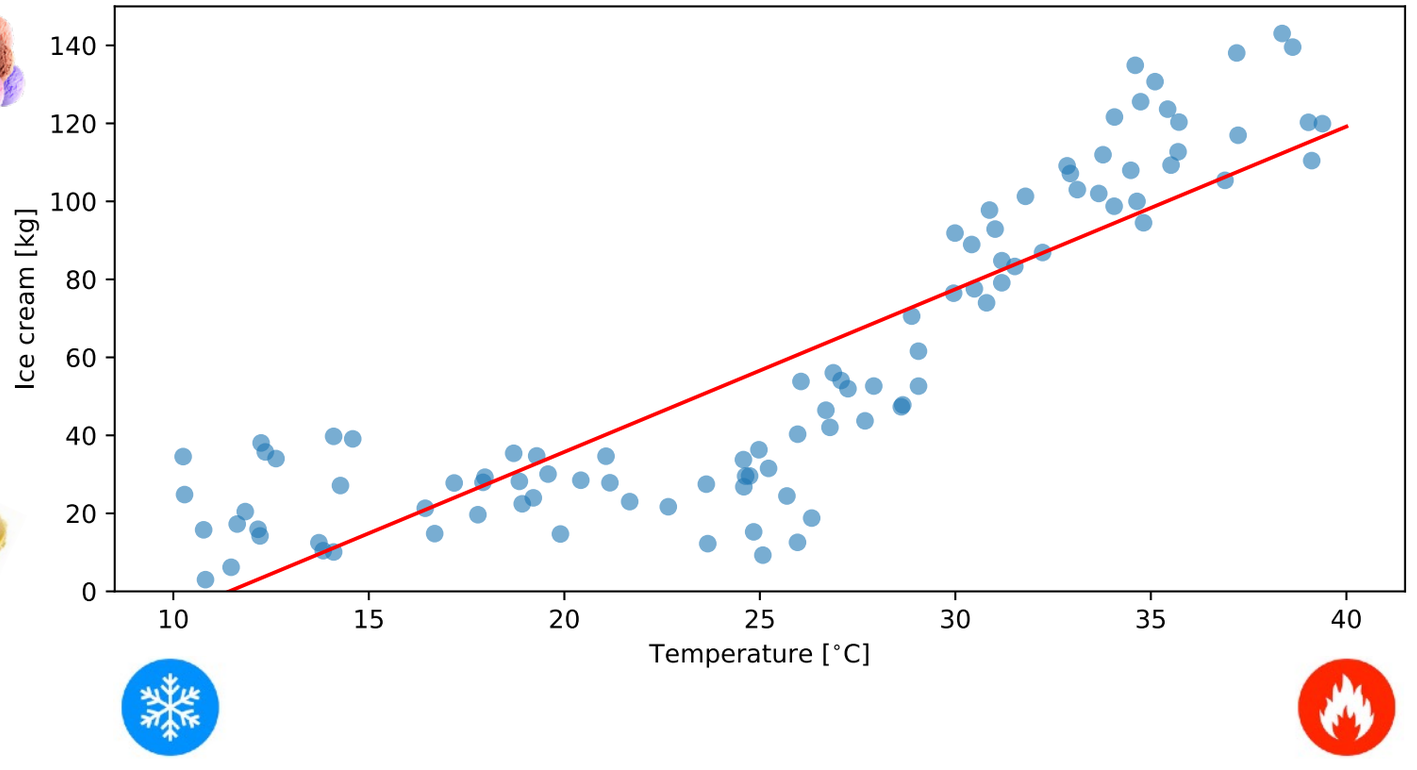
Part 2

- The Cannon
- The Payne
- Issues of using ML for stellar spectroscopy and possible solutions

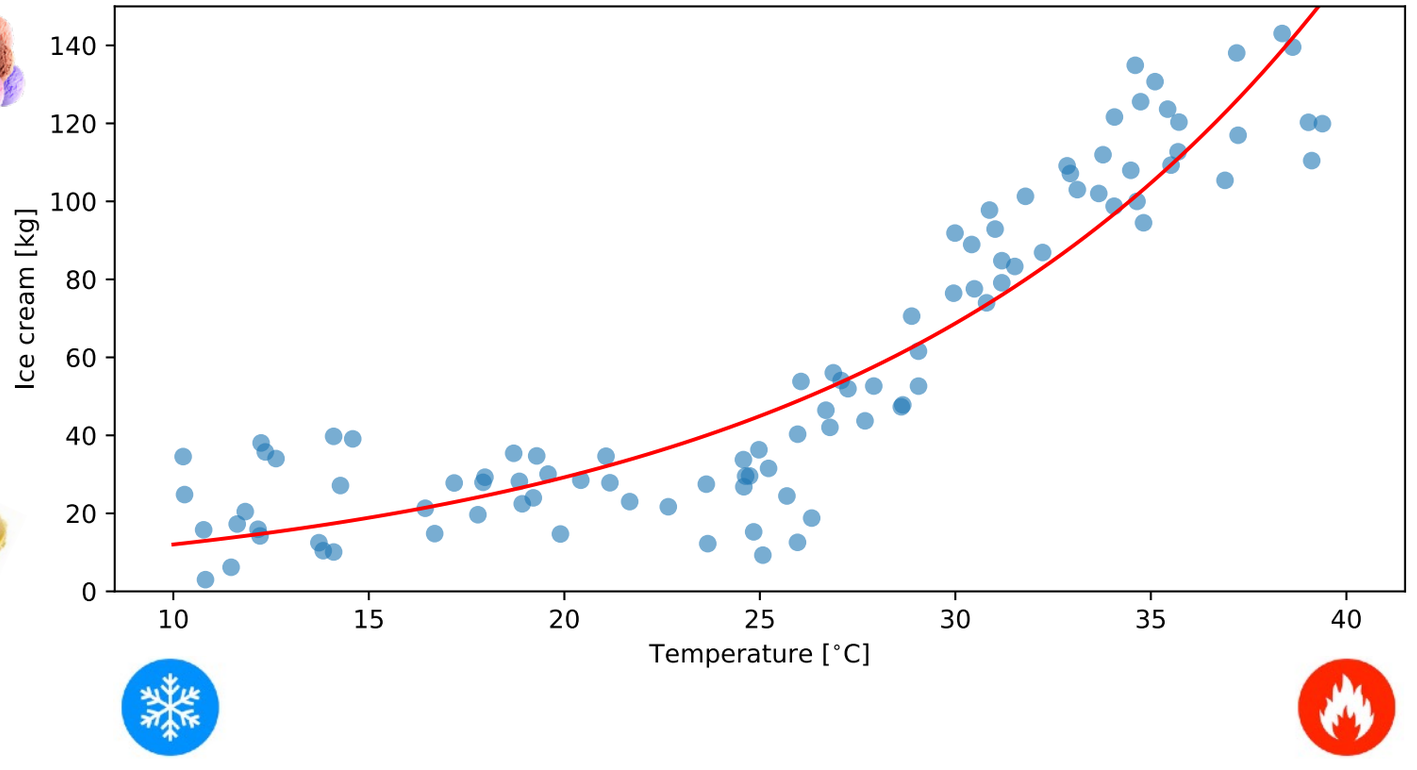

ICE
CREAM


Pr(|)

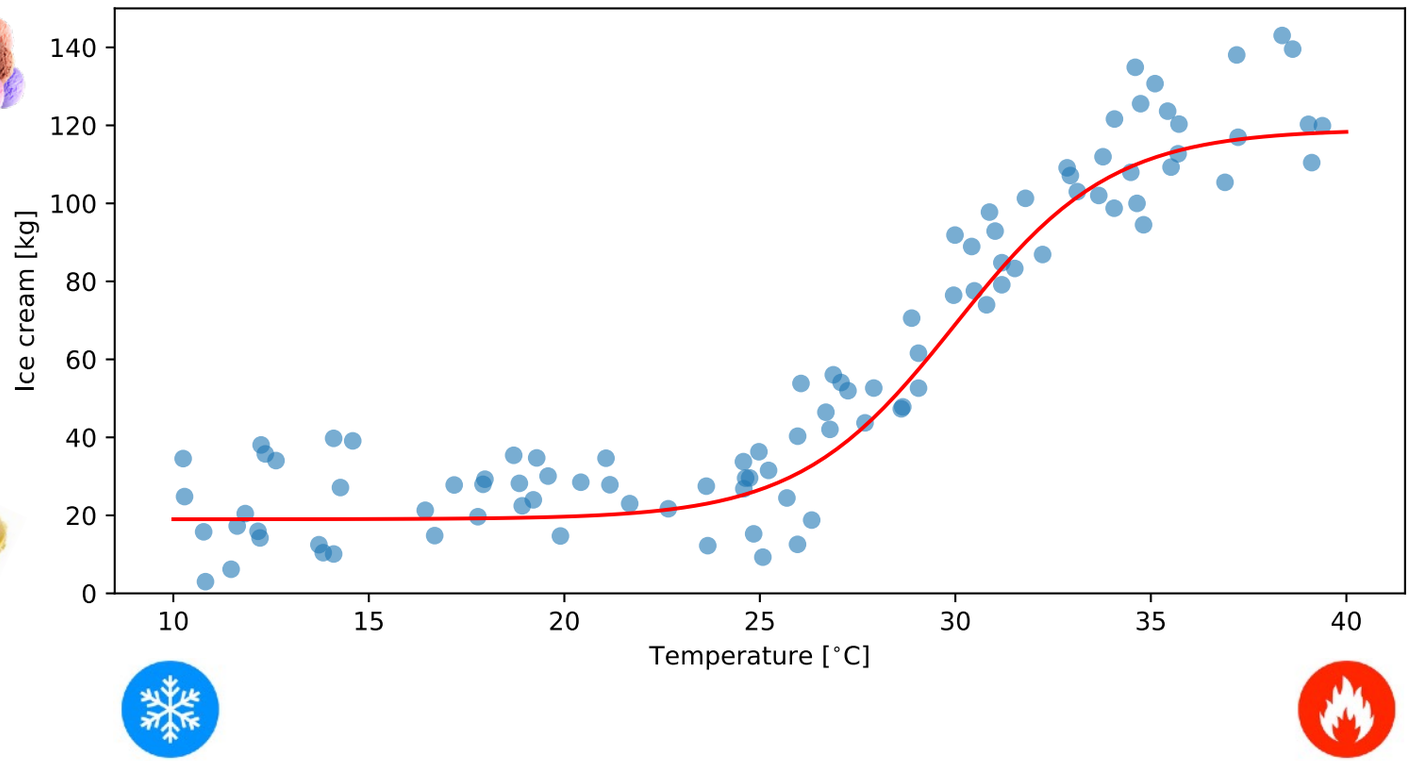

Pr(|)


Ice cream [kg]
Temperature [°C]
0
20
40
60
80
100
120
140
10
15
20
25
30
35
40

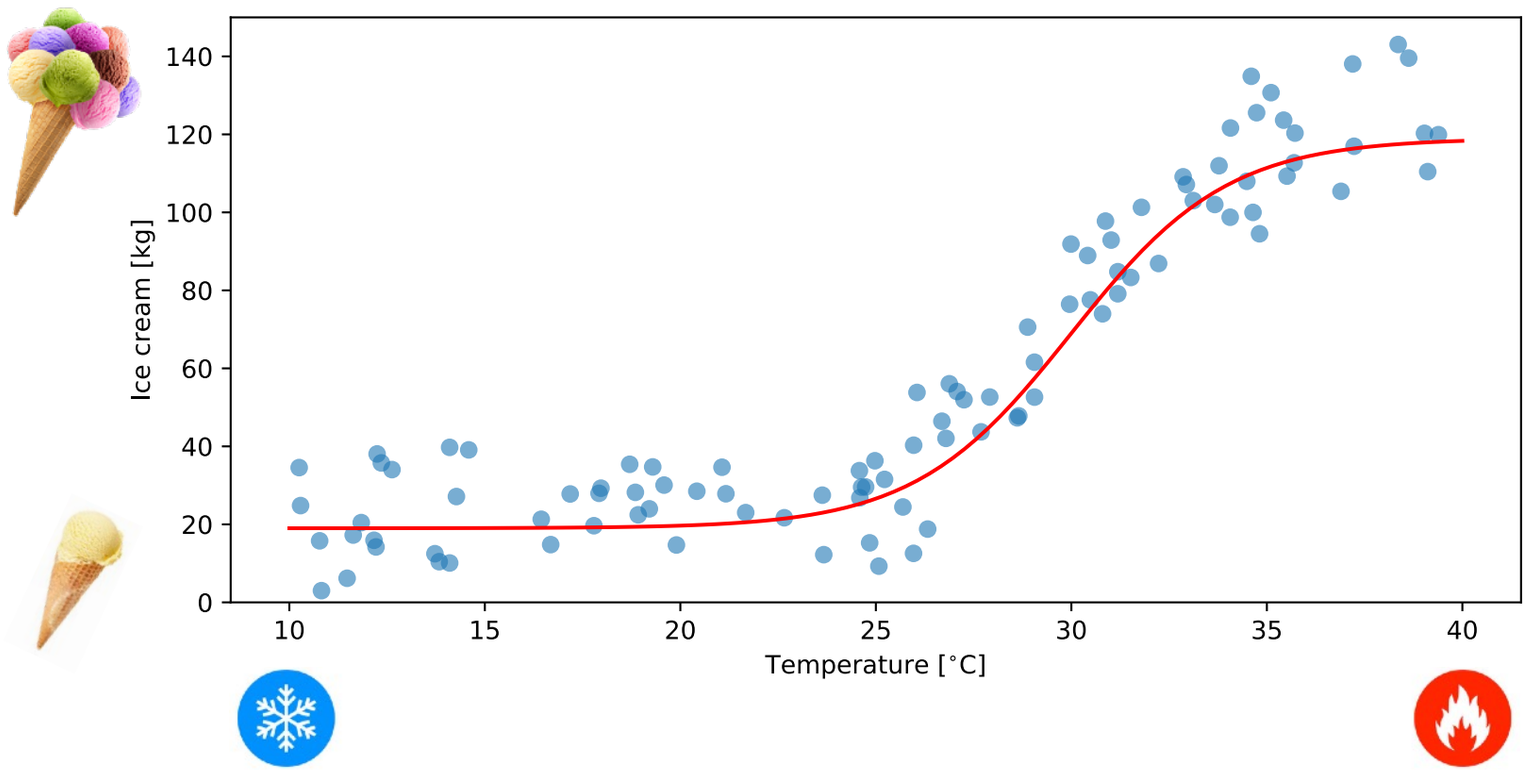

Pr(|)

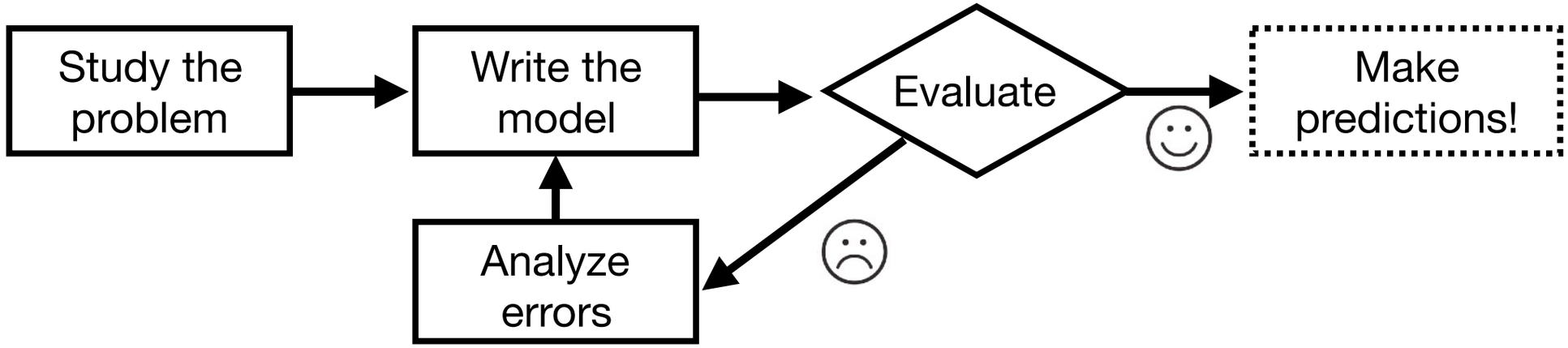

Ice cream [kg]
0
20
40
60
80
100
120
140
10
15
20
25
30
35
40
Temperature [°C]


Pr(|)

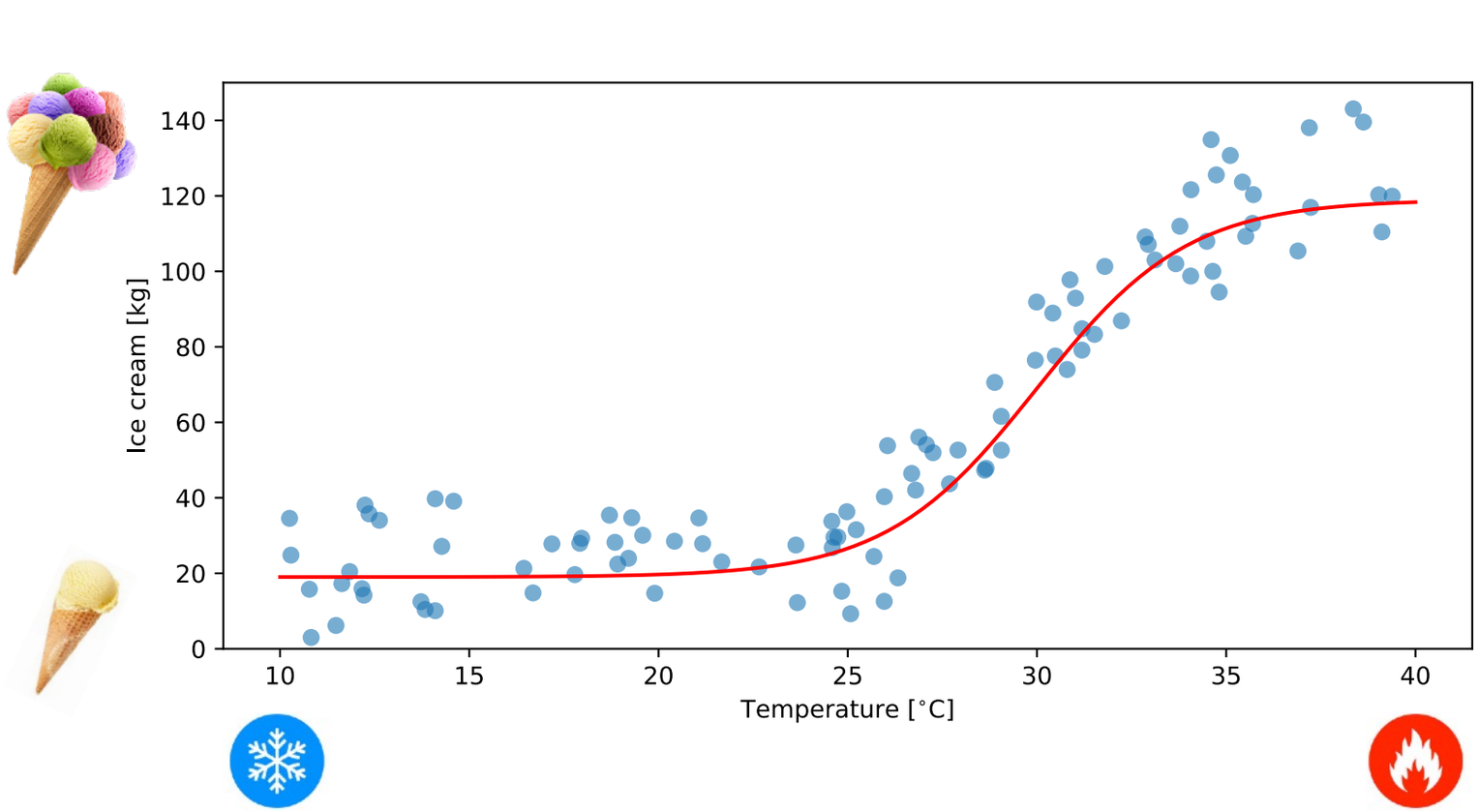

Ice cream [kg]
0
20
40
60
80
100
120
140
10
15
20
25
30
35
40
Temperature [°C]

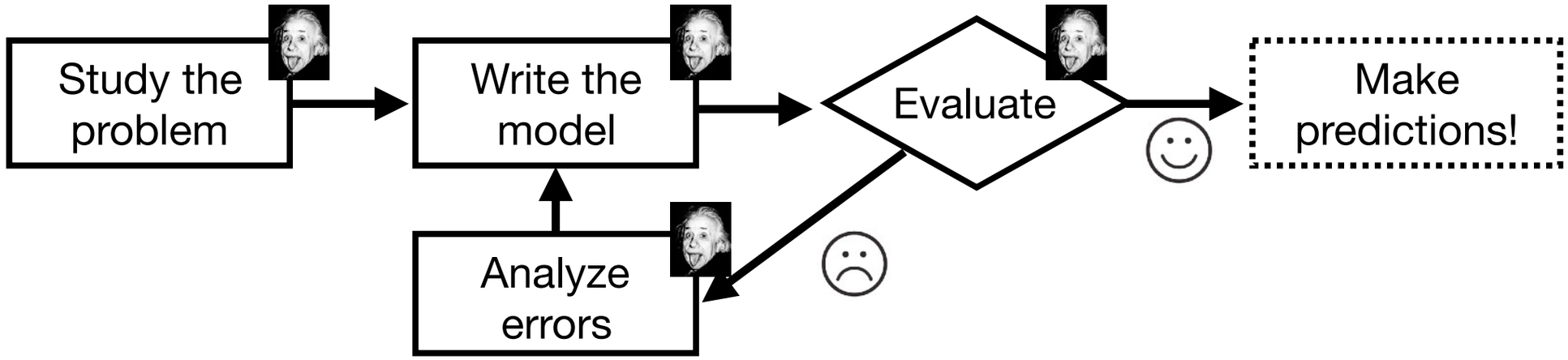

Pr(|)


Ice cream [kg]
Temperature [°C]
0
20
40
60
80
100
120
140
10
15
20
25
30
35
40

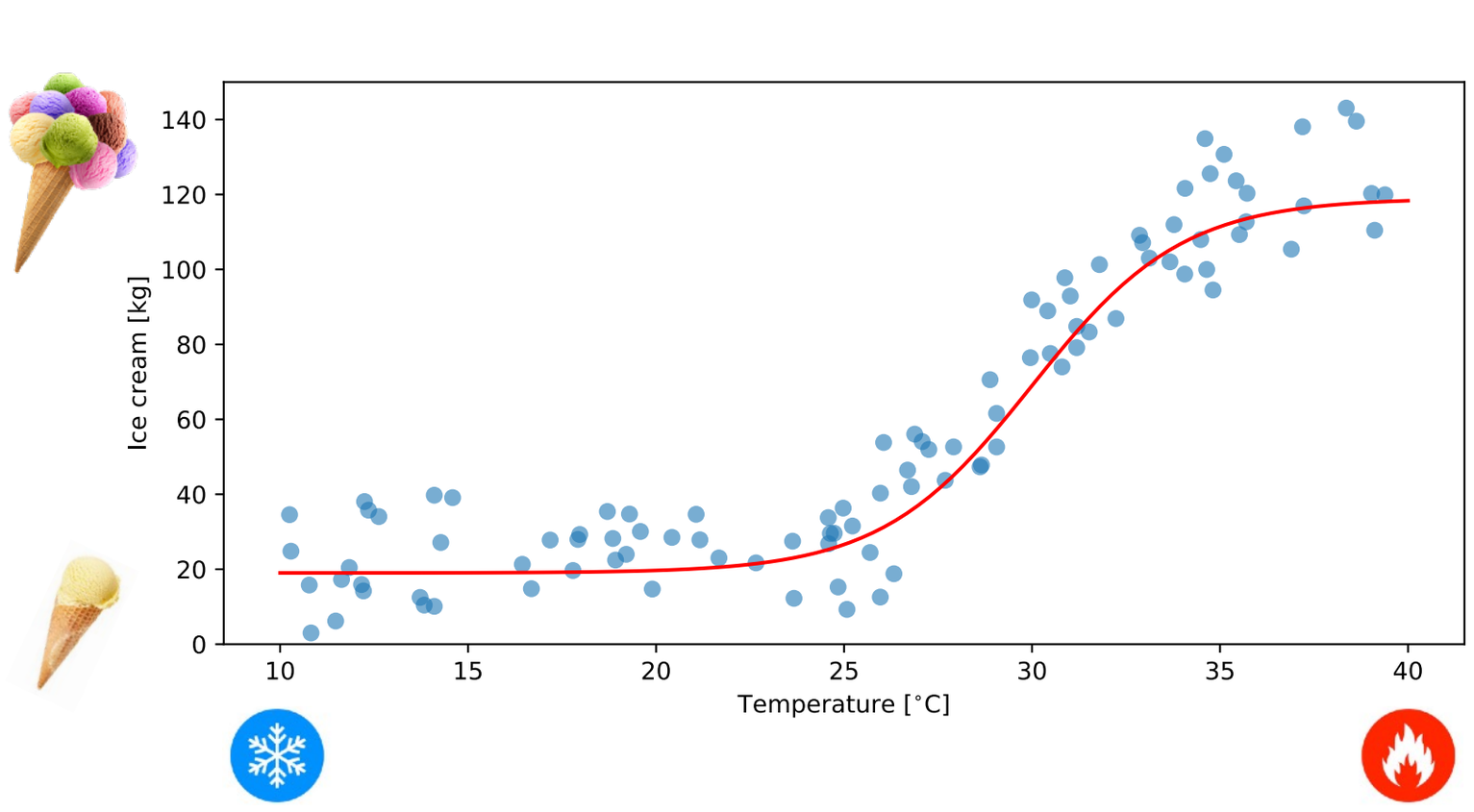

Study the problem
Write the model
Evaluate
Make predictions!
Analyze errors

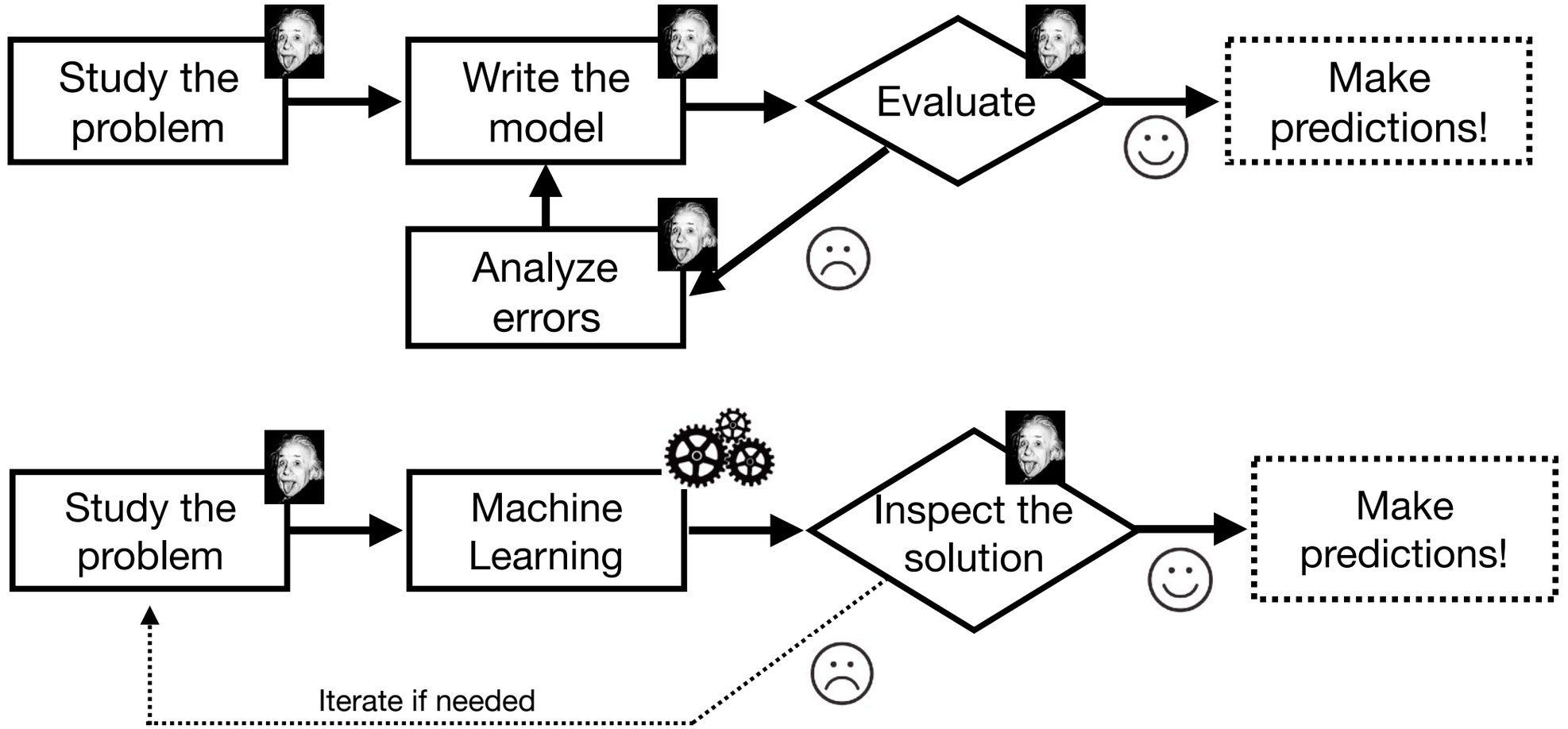

Pr(|)


Ice cream [kg]
Temperature [°C]
0
20
40
60
80
100
120
140
10
15
20
25
30
35
40


Study the problem
Write the model
Evaluate
Make predictions!
Analyze errors


Pr(|)

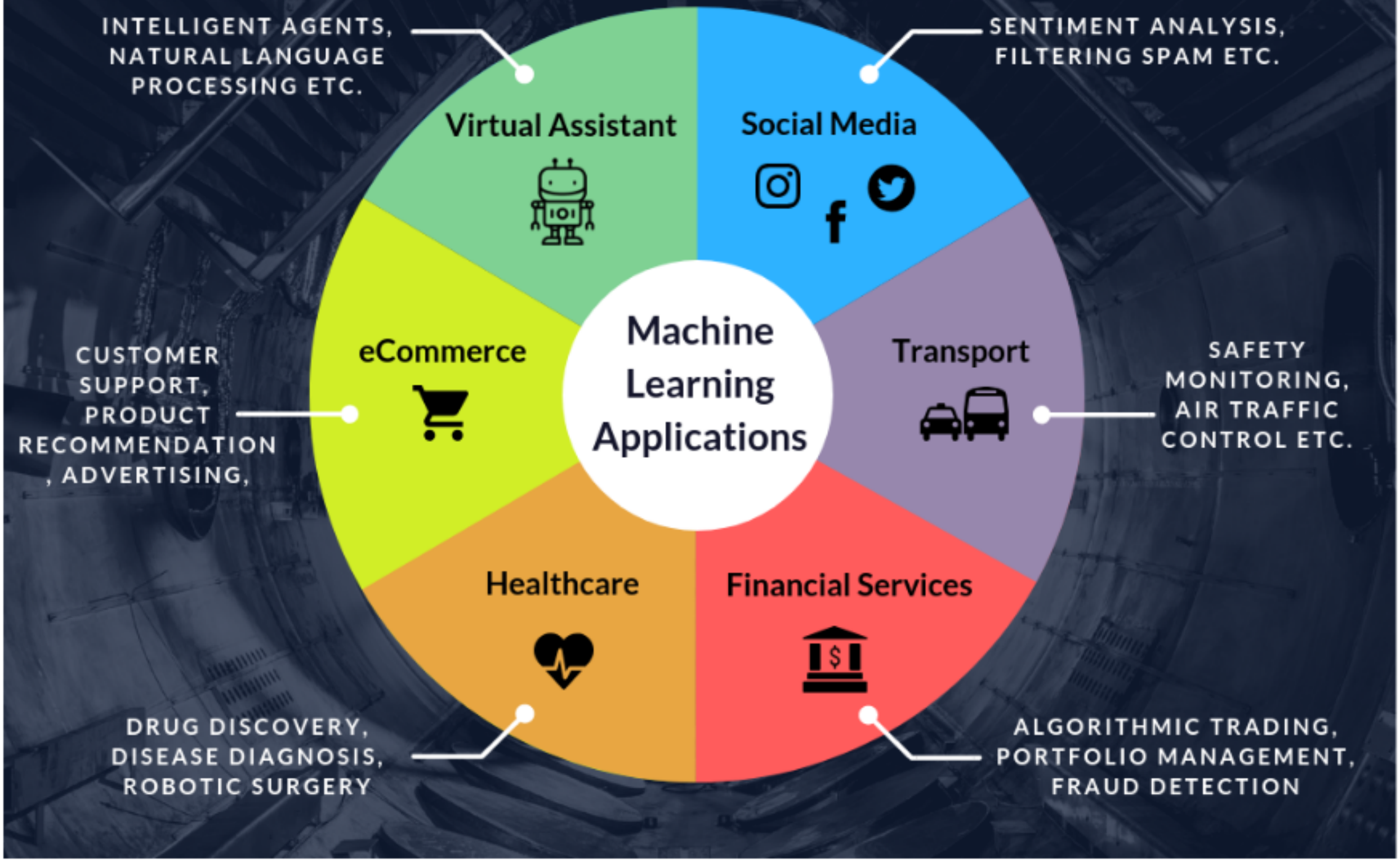

Ice cream [kg]
Temperature [°C]
0
20
40
60
80
100
120
140
10
15
20
25
30
35
40


Study the problem
Write the model
Evaluate
Make predictions!
Analyze errors
Study the problem
Machine Learning
Inspect the solution
Make predictions!
Iterate if needed

''Field of study that gives computers the ability to learn without being explicitly programmed''

Arthur Samuel,
Teaching a computer to play checkers, 1959

APPLICATIONS OF MACHINE LEARNING



APPLICATIONS OF MACHINE LEARNING

QUANT INSTI® GO ALGO

Machine Learning Applications

Virtual Assistant

INTELLIGENT AGENTS, NATURAL LANGUAGE PROCESSING ETC.

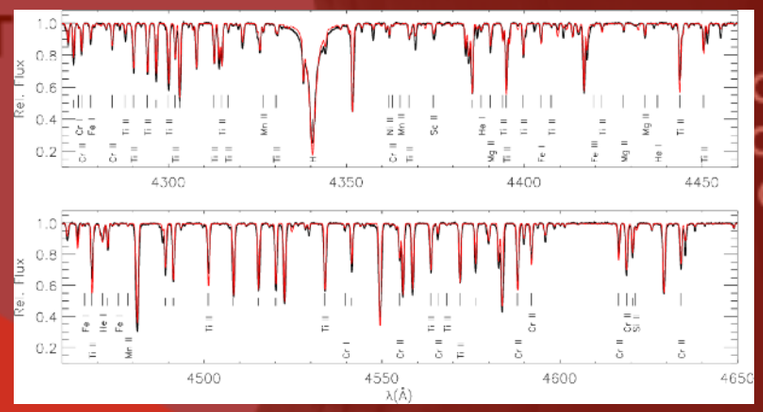
Social Media

SENTIMENT ANALYSIS, FILTERING SPAM ETC.

eCommerce

CUSTOMER SUPPORT, PRODUCT RECOMMENDATION , ADVERTISING,

Stellar spectroscopy



Healthcare

DRUG DISCOVERY, DISEASE DIAGNOSIS, ROBOTIC SURGERY

Financial Services

ALGORITHMIC TRADING, PORTFOLIO MANAGEMENT, FRAUD DETECTION

CLASSICAL MACHINE LEARNING

Data is pre-categorized or numerical

SUPERVISED

Predict a category

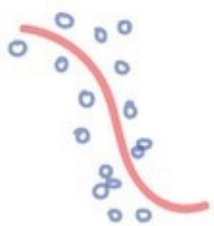
CLASSIFICATION

«Divide the socks by color»

Predict a number

REGRESSION

«Divide the ties by length»



Data is not labeled in any way

UNSUPERVISED

Divide by similarity

CLUSTERING

«Split up similar clothing into stacks»

Find hidden dependencies

DIMENSION REDUCTION (generalization)

«Make the best outfits from the given clothes»

Identify sequences

ASSOCIATION

«Find what clothes I often wear together»

CLASSICAL MACHINE LEARNING

Data is pre-categorized or numerical

SUPERVISED

Predict a category

CLASSIFICATION

«Divide the socks by color»

Predict a number

REGRESSION

«Divide the ties by length»

Data is not labeled in any way

UNSUPERVISED

Divide by similarity

CLUSTERING

«Split up similar clothing into stacks»

Identify sequences

Find hidden dependencies

ASSOCIATION

«Find what clothes I often wear together»

DIMENSION REDUCTION (generalization)

«Make the best outfits from the given clothes»

CLASSICAL MACHINE LEARNING

Data is pre-categorized or numerical

SUPERVISED

Predict a category

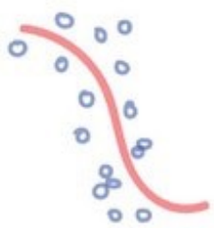
CLASSIFICATION

«Divide the socks by color»

Predict a number

REGRESSION

«Divide the ties by length»



Data is not labeled in any way

UNSUPERVISED

Divide by similarity

CLUSTERING

«Split up similar clothing into stacks»

Identify sequences

Find hidden dependencies

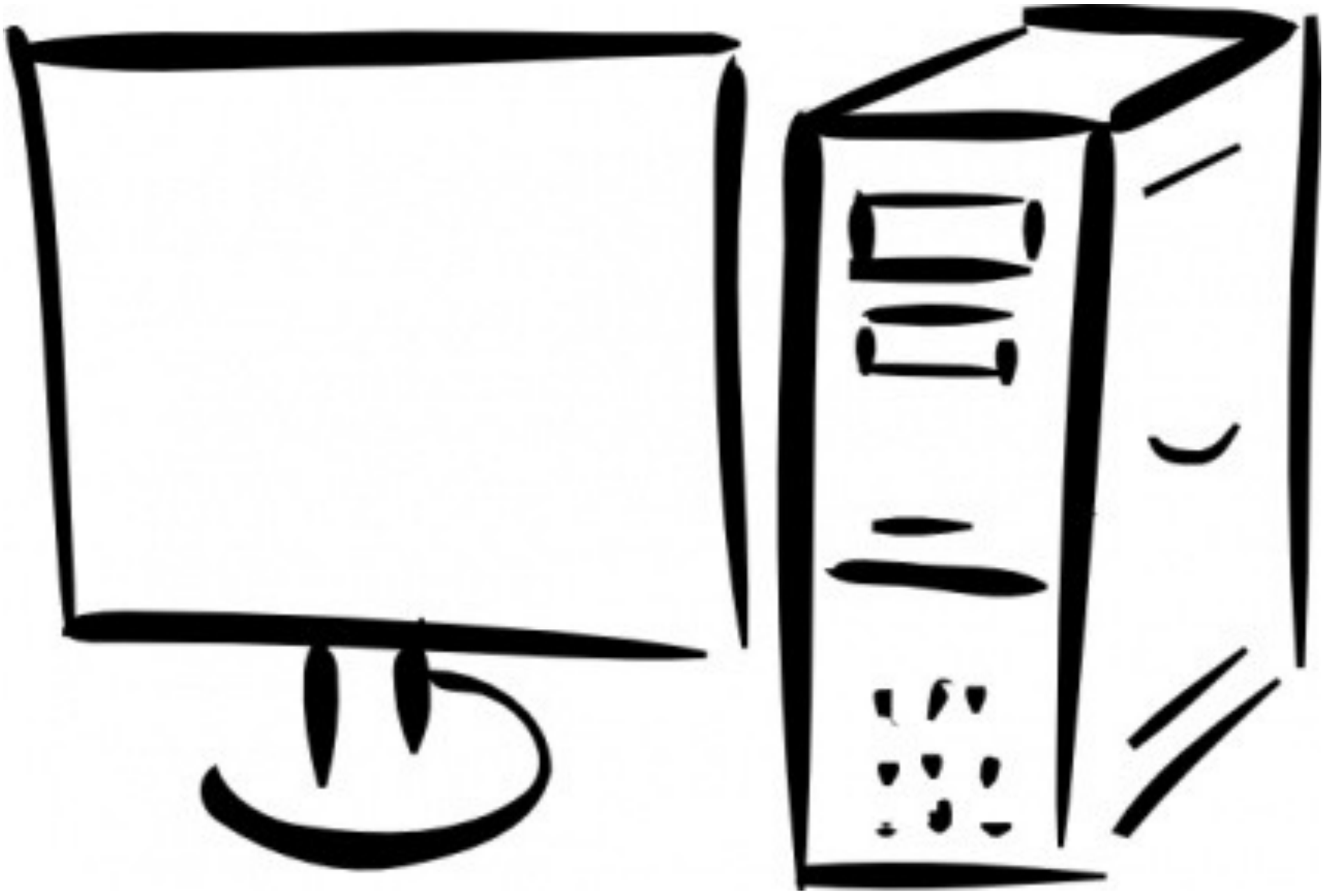
ASSOCIATION

«Find what clothes I often wear together»

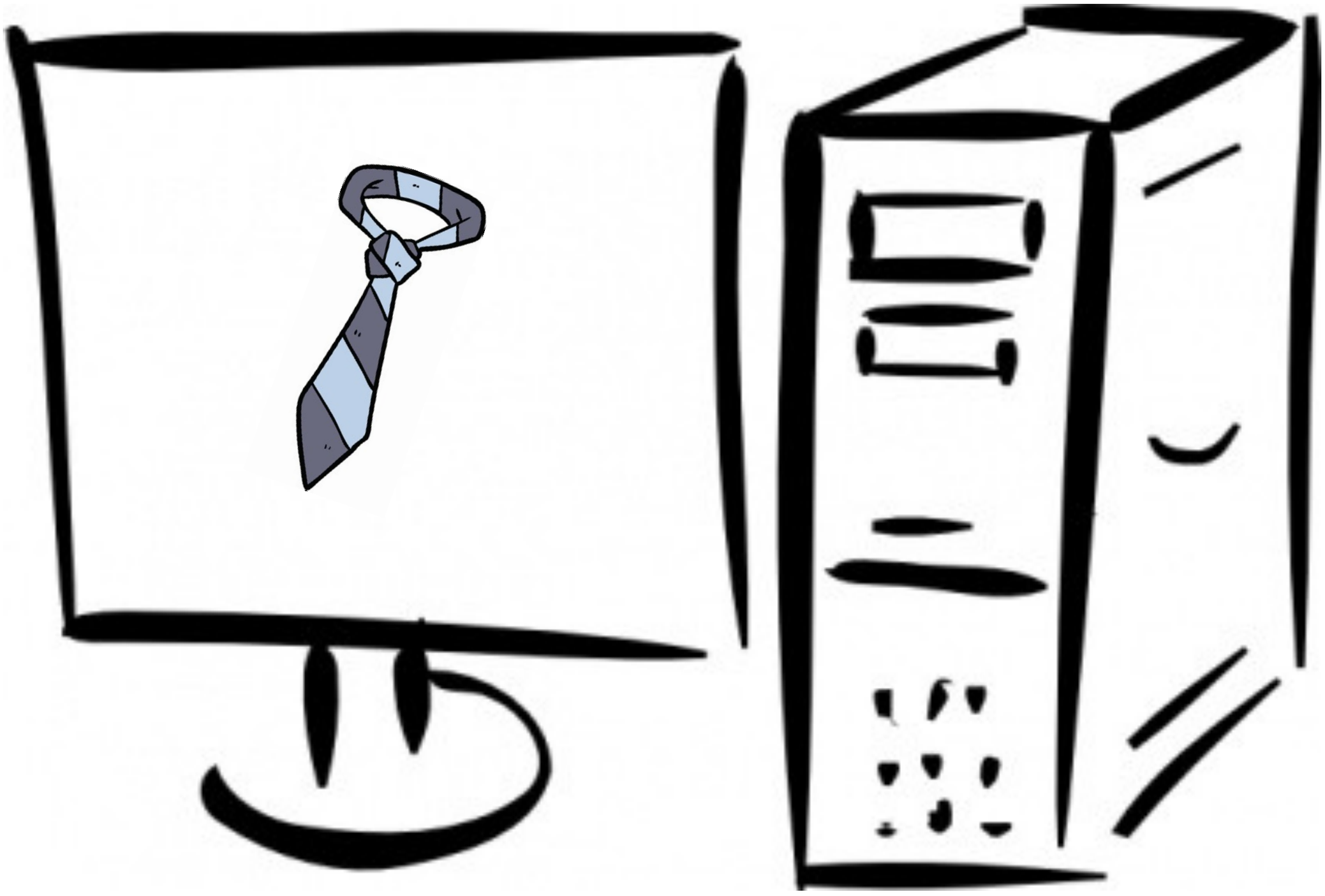
DIMENSION REDUCTION (generalization)

«Make the best outfits from the given clothes»

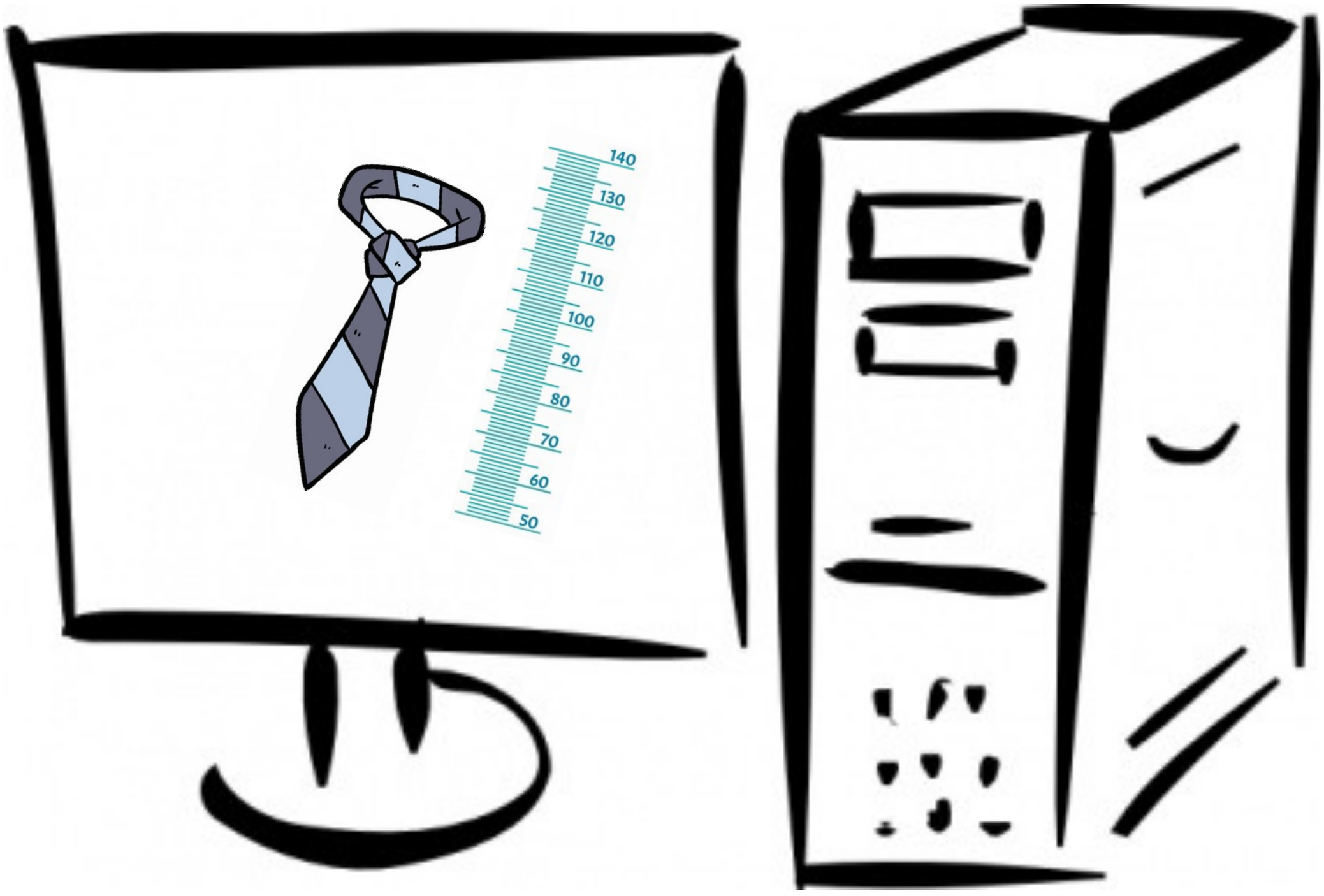
Supervised Machine Learning - Regression Algorithms



Supervised Machine Learning - Regression Algorithms

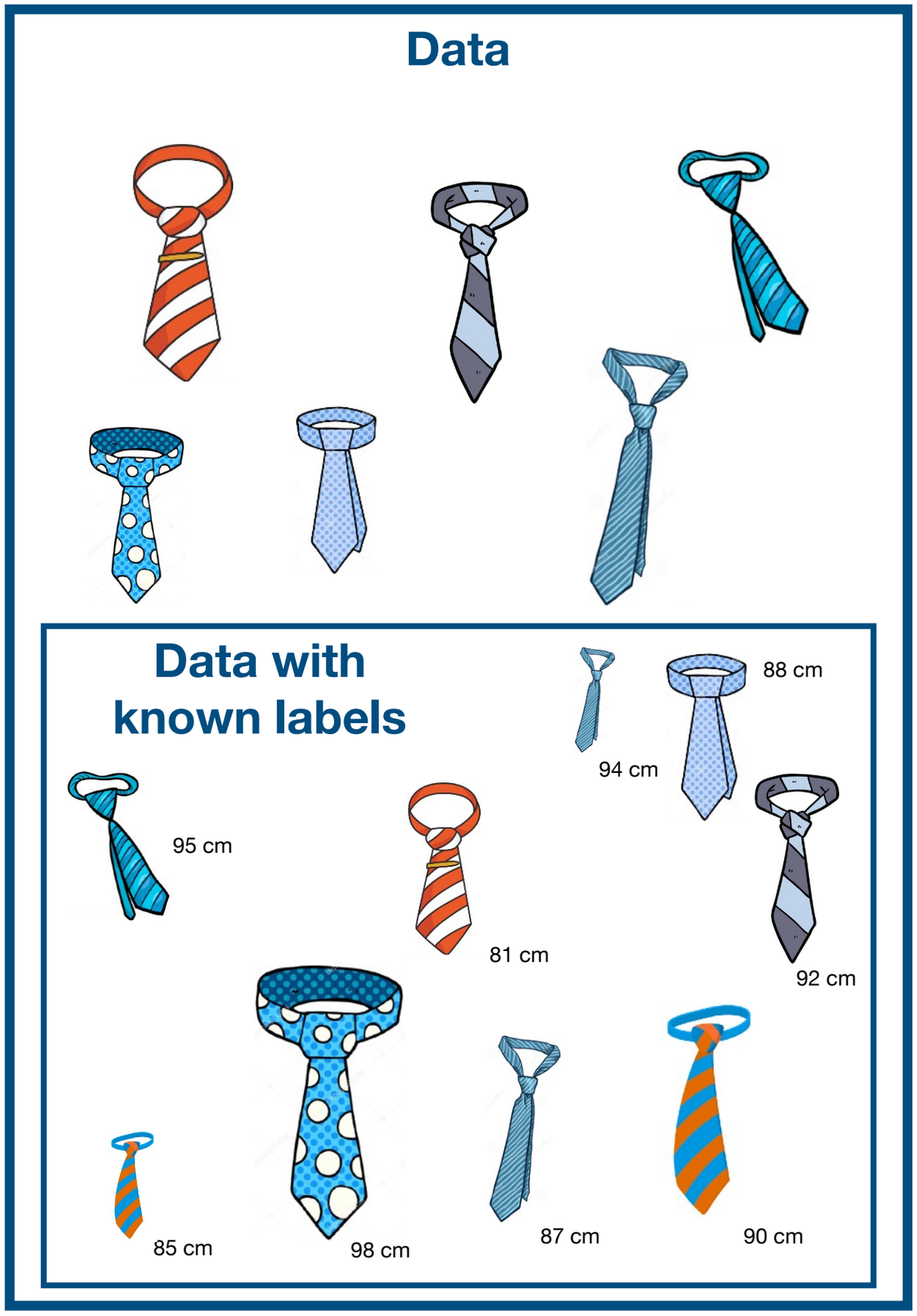


Supervised Machine Learning - Regression Algorithms

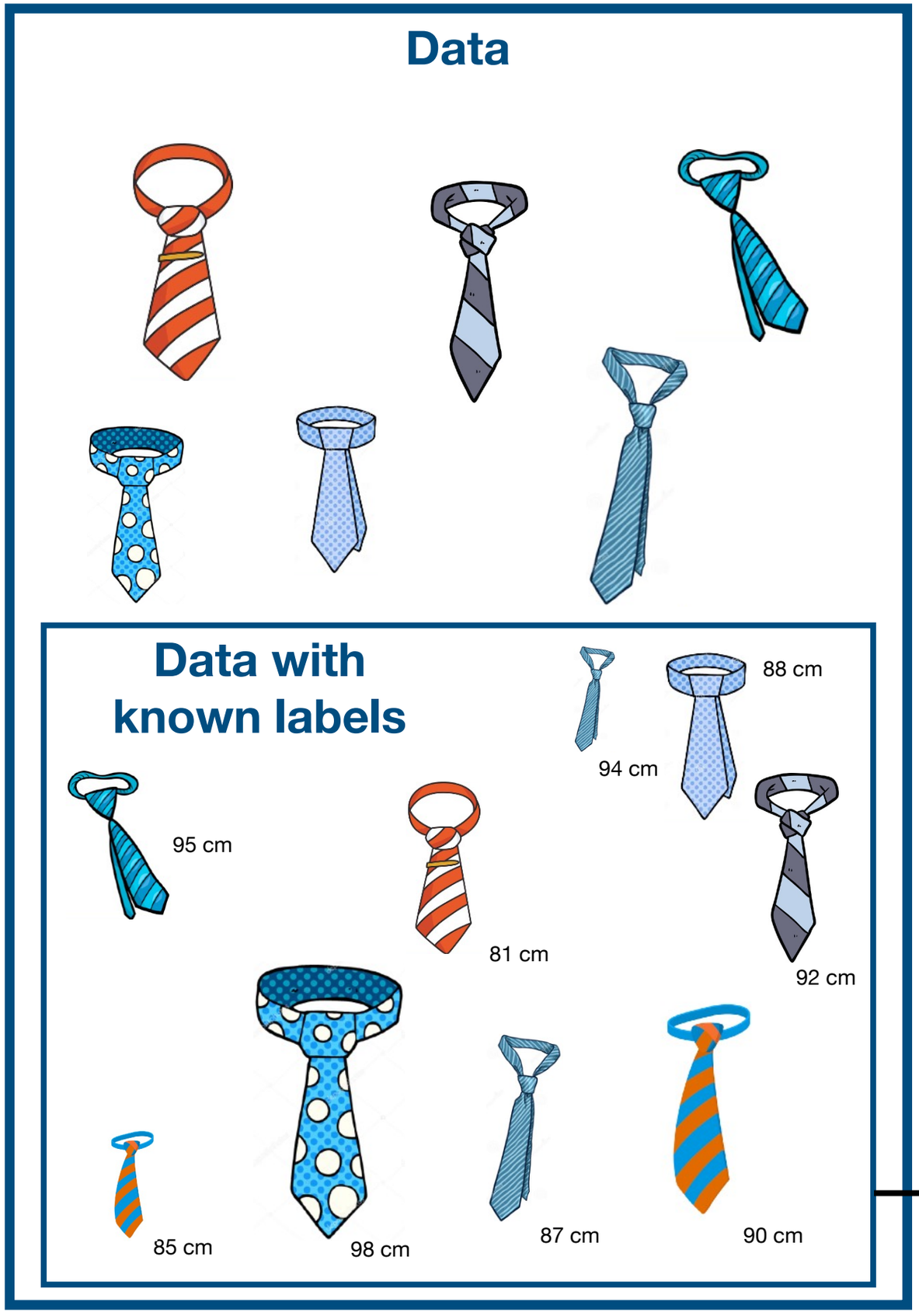


Supervised Machine Learning - Regression Algorithms

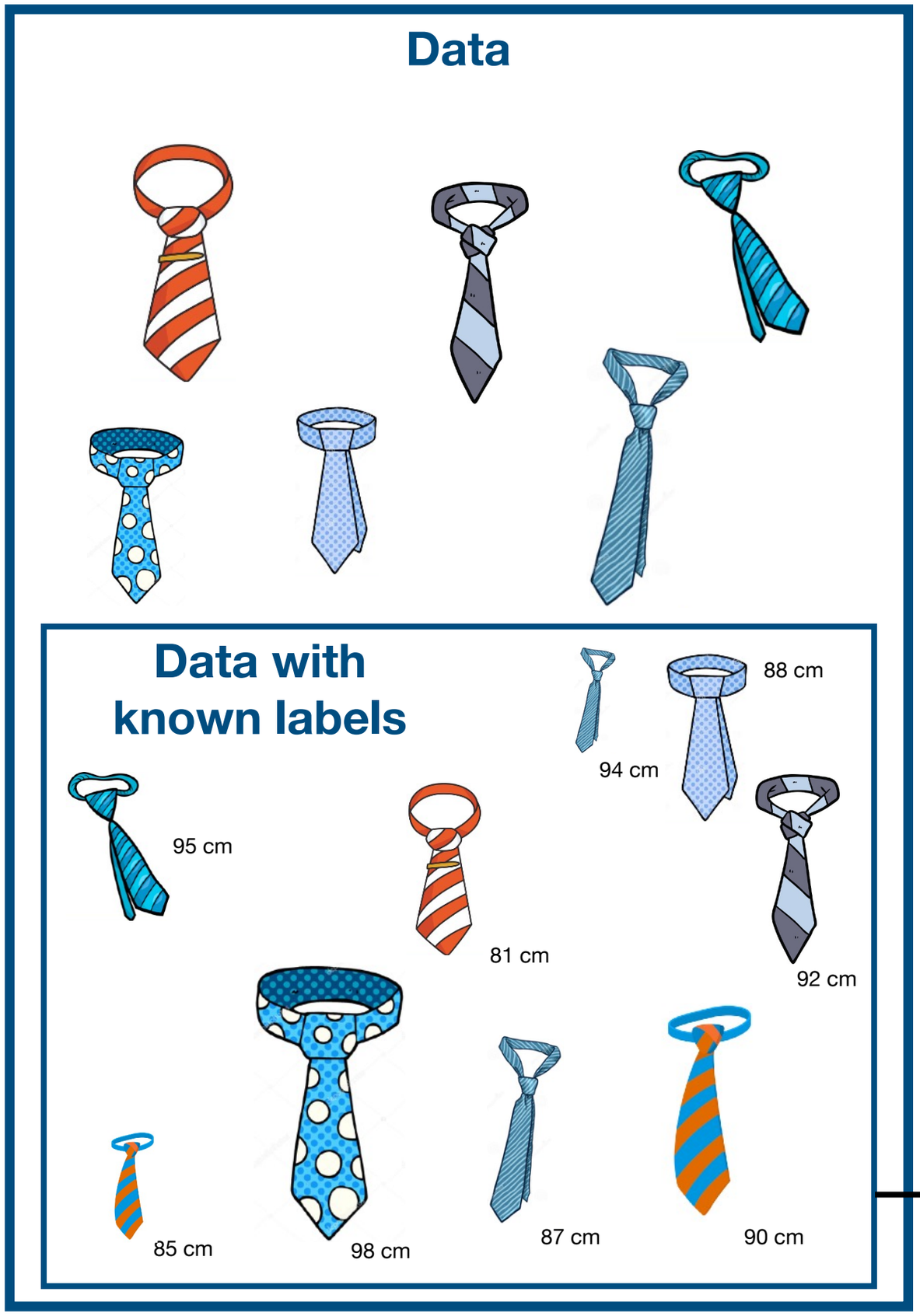
Supervised Machine Learning - Regression Algorithms



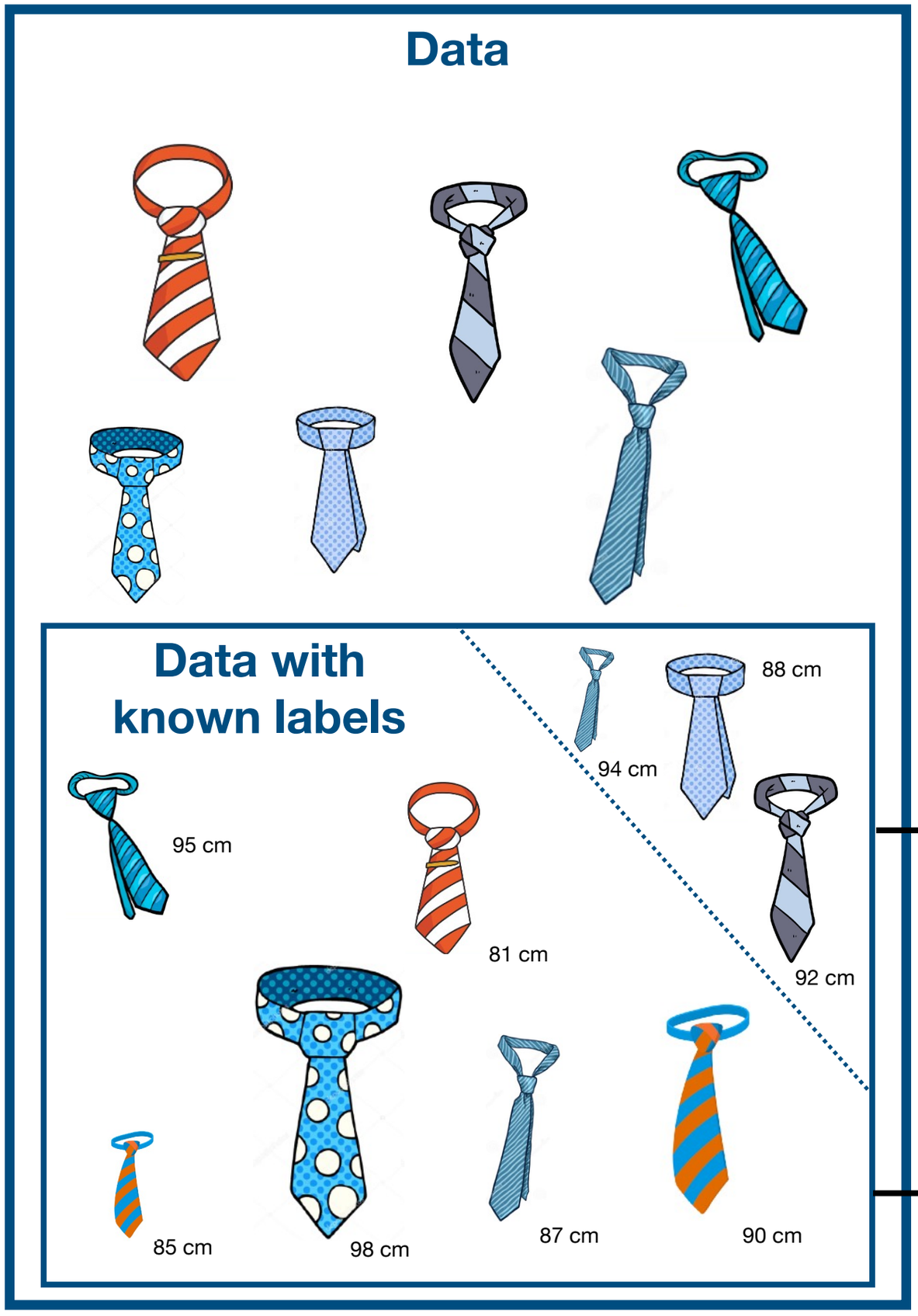
Supervised Machine Learning - Regression Algorithms



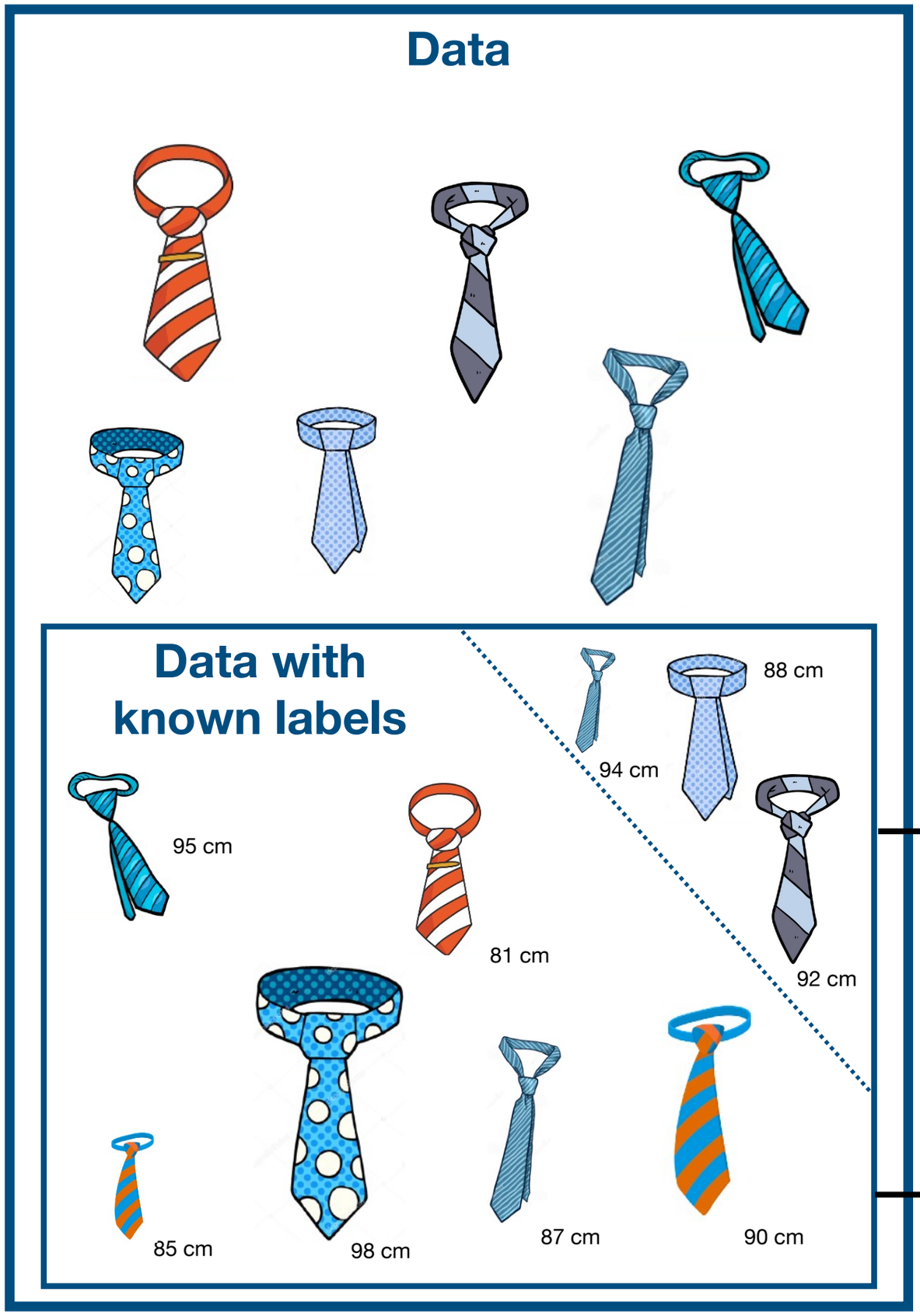
Supervised Machine Learning - Regression Algorithms



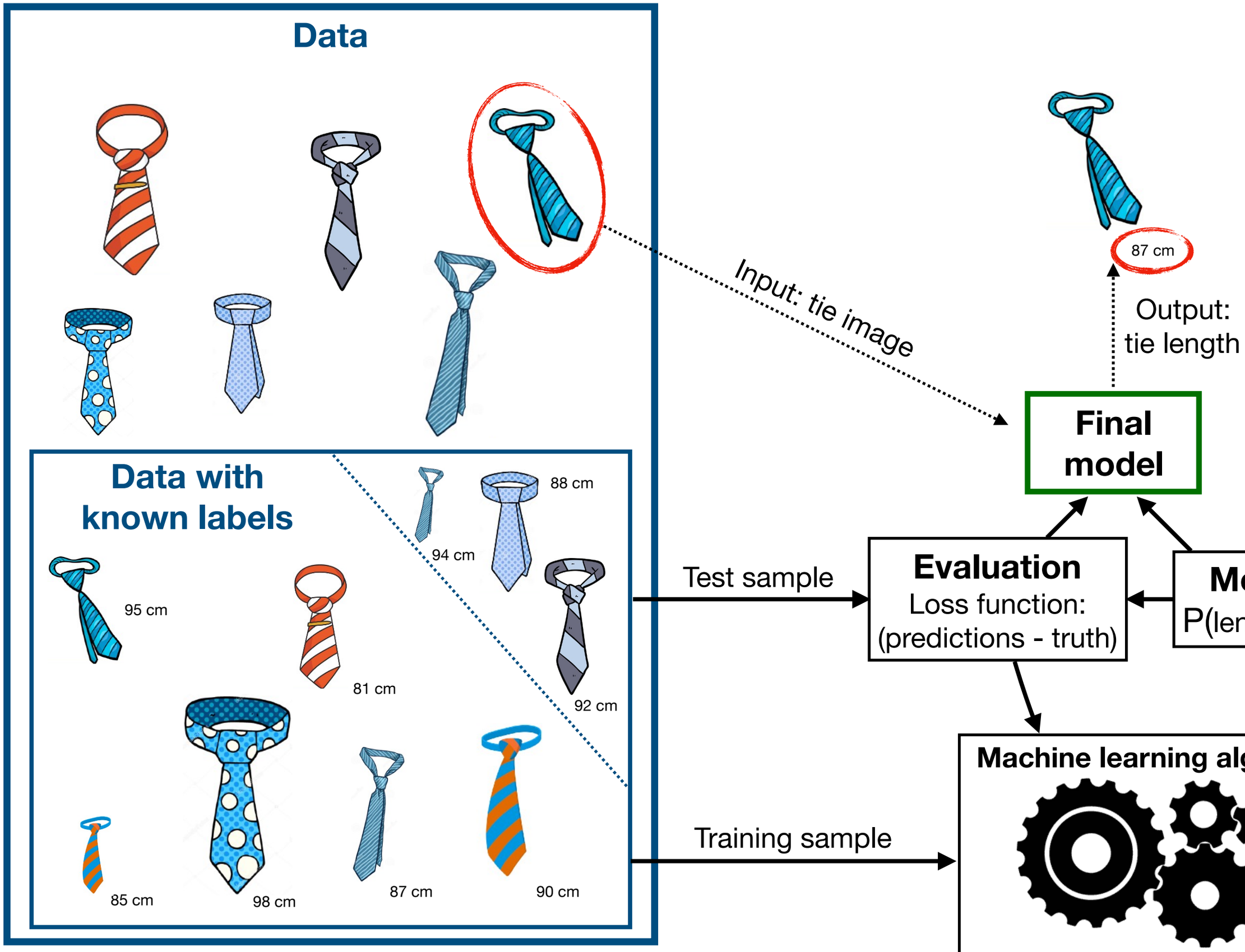
Supervised Machine Learning - Regression Algorithms



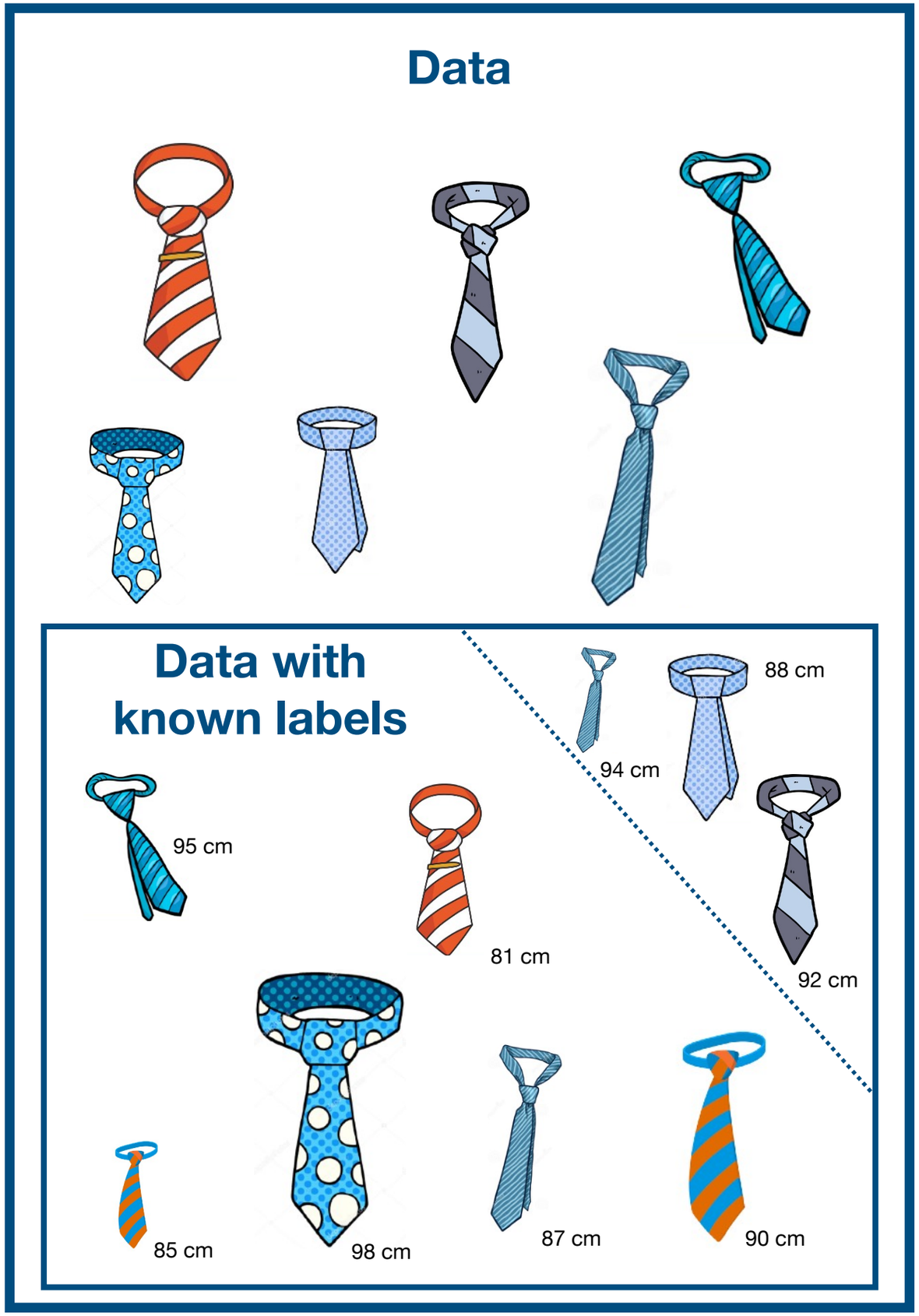
Supervised Machine Learning - Regression Algorithms



Supervised Machine Learning - Regression Algorithms



Supervised Machine Learning - Regression Algorithms

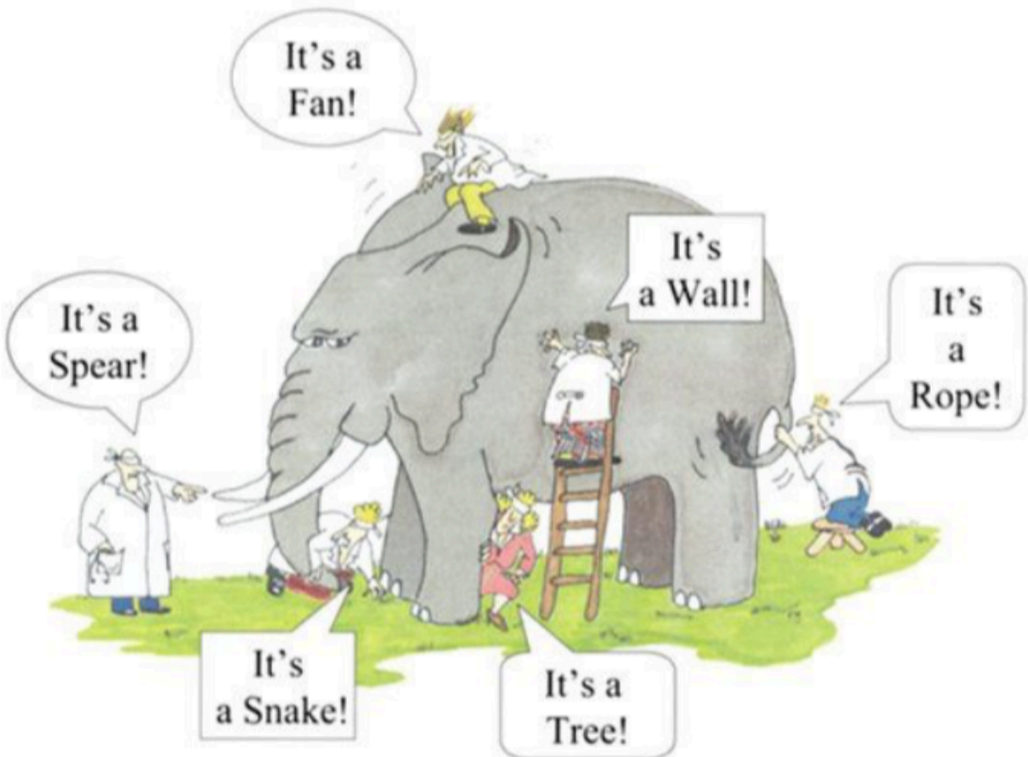


It is a "**generalization problem**".

The algorithm learns concepts from the training sample and then applies these rules to the rest of our dataset.

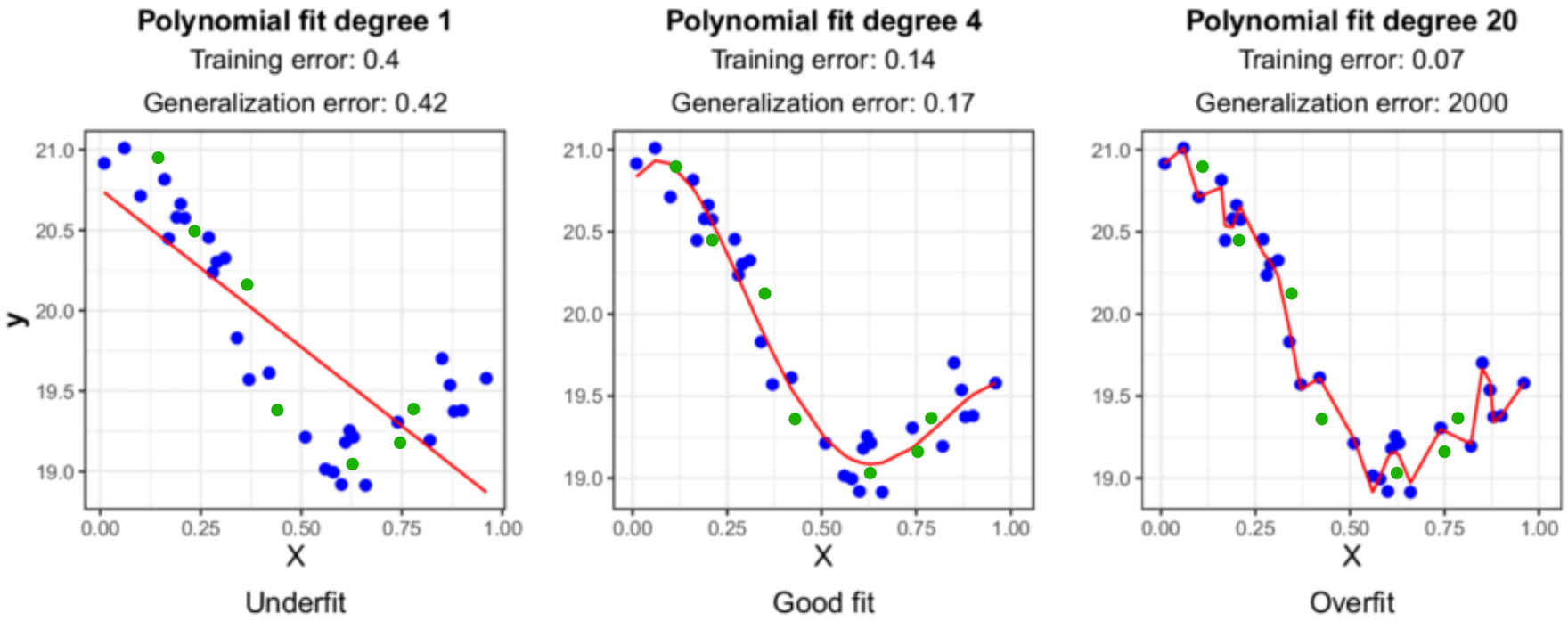
The test set gives you a **generalisation error** (i.e., how well the model is able to generalize).

The training/test sets need to be **representative** of the full picture.



Finding the *best* model

Several types of algorithms (linear/logistic regression, decision tree algorithms, support vector machines, k-means, artifical neuronal networks, etc...). Each of them is recommended for a different type of problem and dataset. Each of them has its own set of **parameters** that can be tuned in order to find the best model.



Sometimes the model performs well on the training data, but doesn't generalise well (fails on new data).

A very simple model is more prone to **underfitting**.

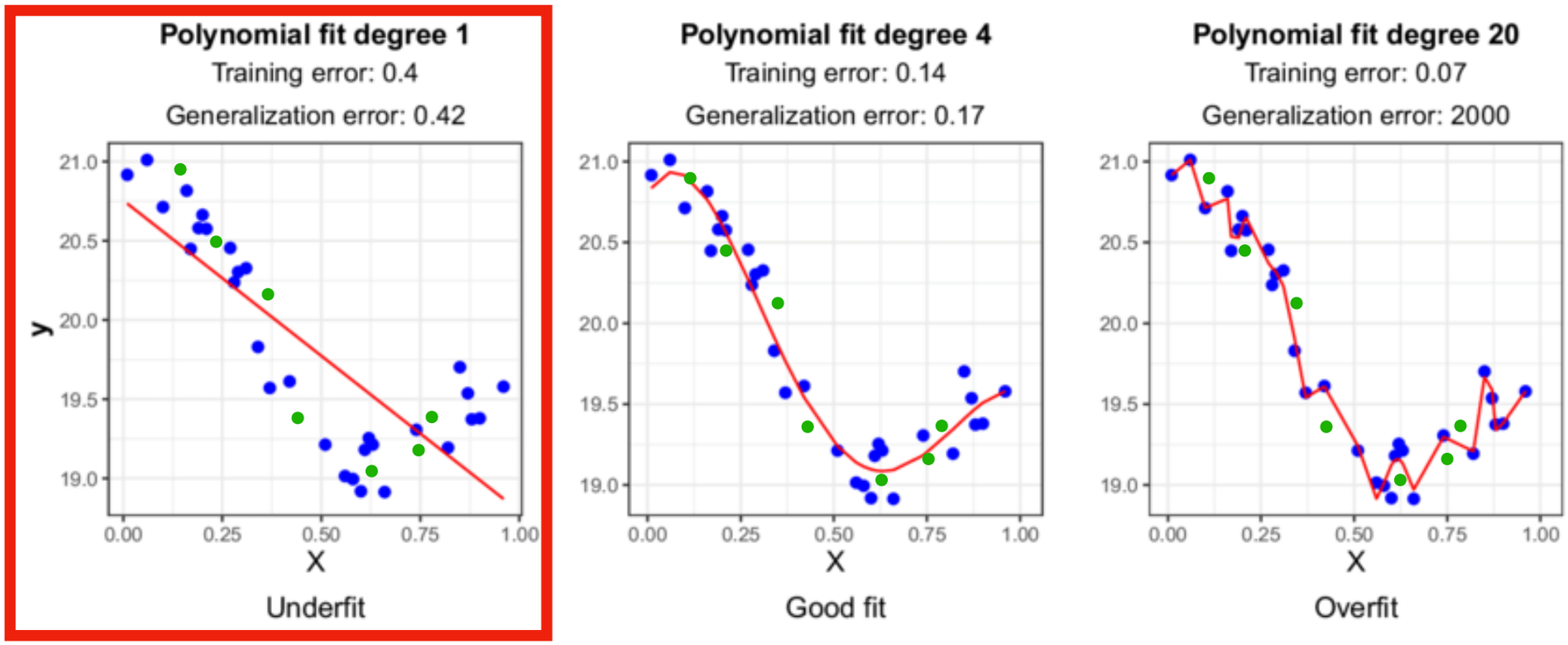
A model with more free parameters is more prone to **overfitting**: they can detect subtle patterns in data, but it is likely to detect pattersn in the noise.

A good model is the one that finds a **balance** between underfitting and overfitting.

Hereafter we will focus on Artificial Neuronal Networks (ANN), which can tacke highly complex ML tasks.

Finding the *best* model

Several types of algorithms (linear/logistic regression, decision tree algorithms, support vector machines, k-means, artifical neuronal networks, etc...). Each of them is recommended for a different type of problem and dataset. Each of them has its own set of **parameters** that can be tuned in order to find the best model.



Sometimes the model performs well on the training data, but doesn't generalise well (fails on new data).

A very simple model is more prone to **underfitting**.

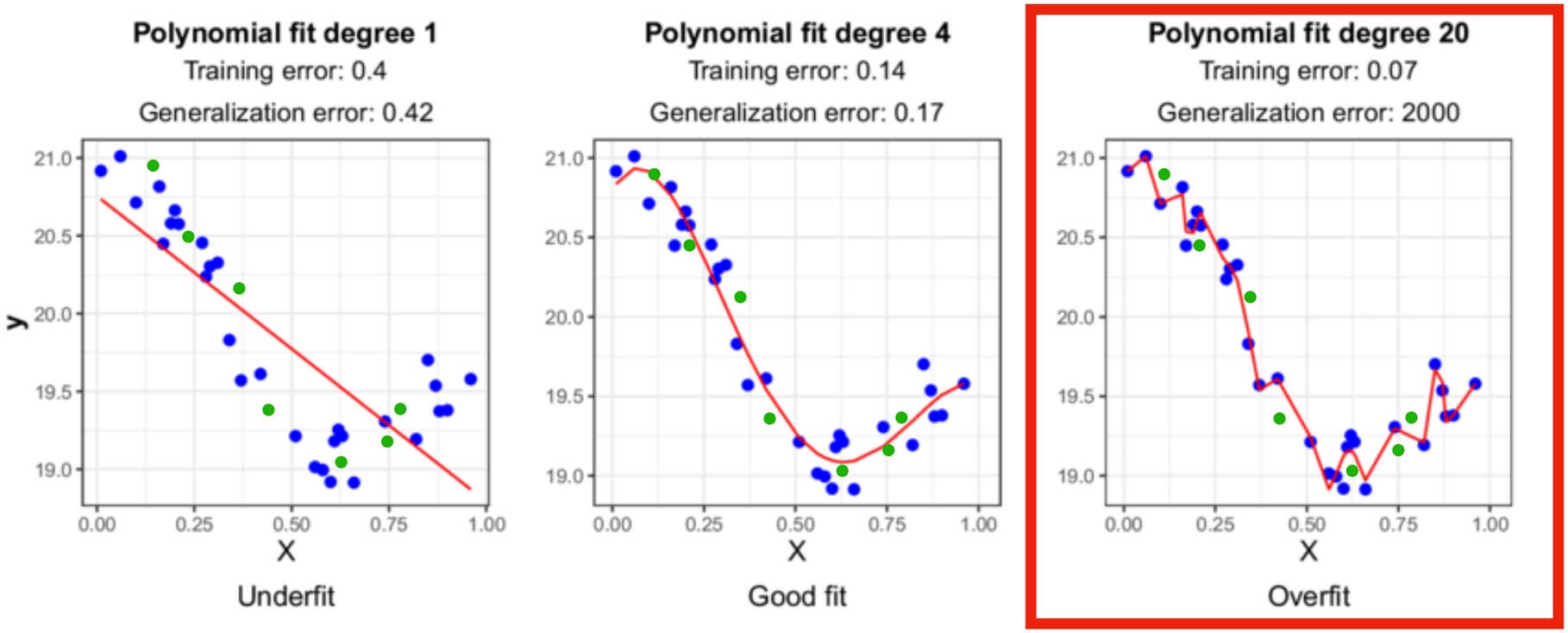
A model with more free parameters is more prone to **overfitting**: they can detect subtle patterns in data, but it is likely to detect pattersn in the noise.

A good model is the one that finds a **balance** between underfitting and overfitting.

Hereafter we will focus on Artificial Neuronal Networks (ANN), which can tacke highly complex ML tasks.

Finding the *best* model

Several types of algorithms (linear/logistic regression, decision tree algorithms, support vector machines, k-means, artifical neuronal networks, etc...). Each of them is recommended for a different type of problem and dataset. Each of them has its own set of **parameters** that can be tuned in order to find the best model.



Sometimes the model performs well on the training data, but doesn't generalise well (fails on new data).

A very simple model is more prone to **underfitting**.

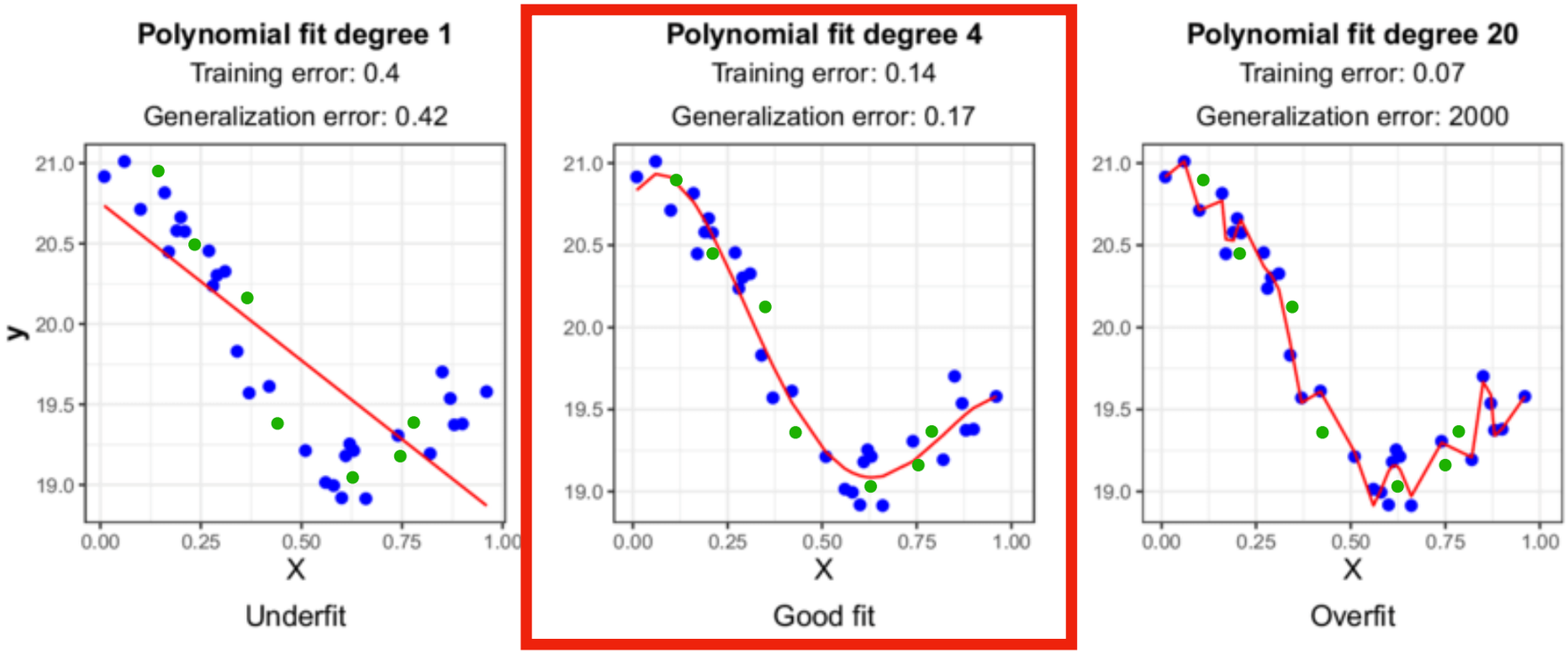
A model with more free parameters is more prone to **overfitting**: they can detect subtle patterns in data, but it is likely to detect pattersn in the noise.

A good model is the one that finds a **balance** between underfitting and overfitting.

Hereafter we will focus on Artificial Neuronal Networks (ANN), which can tacke highly complex ML tasks.

Finding the *best* model

Several types of algorithms (linear/logistic regression, decision tree algorithms, support vector machines, k-means, artifical neuronal networks, etc...). Each of them is recommended for a different type of problem and dataset. Each of them has its own set of **parameters** that can be tuned in order to find the best model.



Sometimes the model performs well on the training data, but doesn't generalise well (fails on new data).

A very simple model is more prone to **underfitting**.

A model with more free parameters is more prone to **overfitting**: they can detect subtle patterns in data, but it is likely to detect pattersn in the noise.

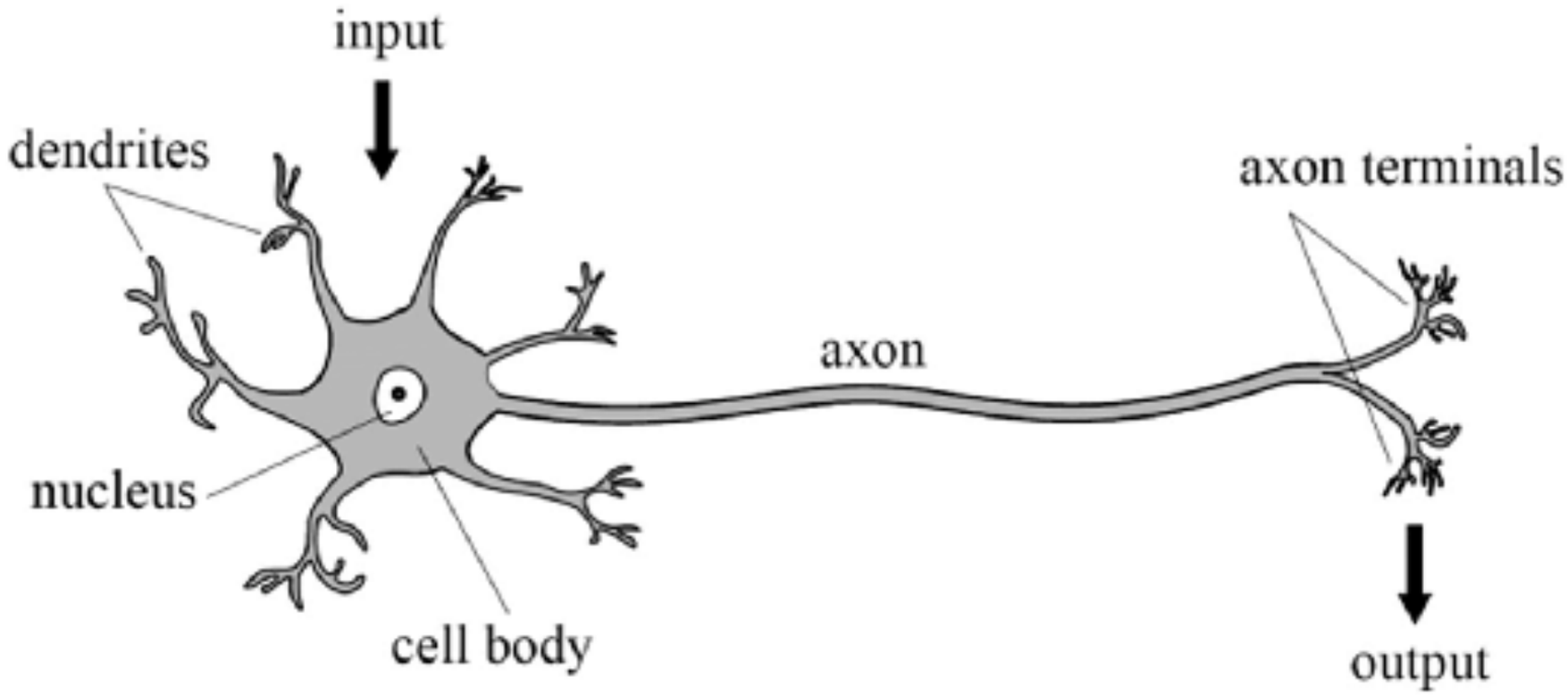
A good model is the one that finds a **balance** between underfitting and overfitting.

Hereafter we will focus on Artificial Neuronal Networks (ANN), which can tacke highly complex ML tasks.

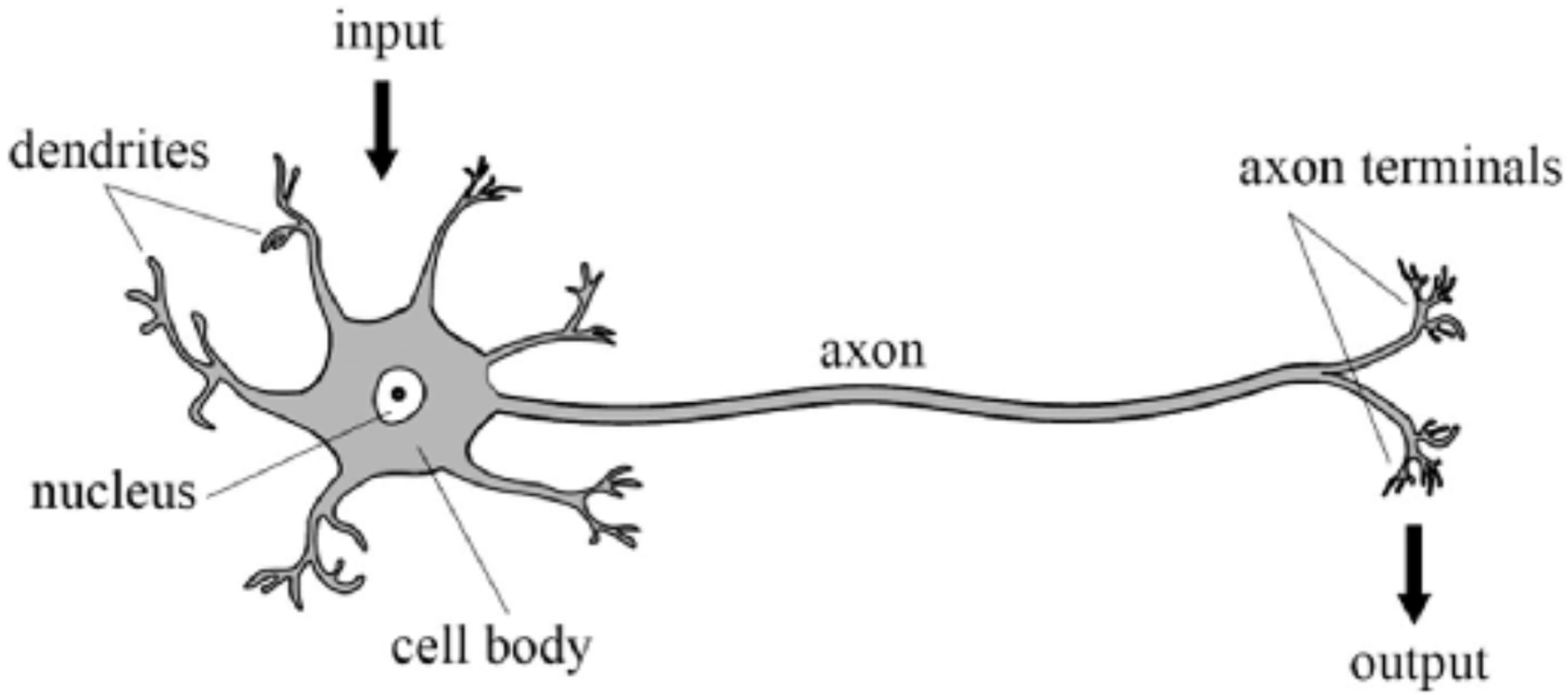
Biological Neurons

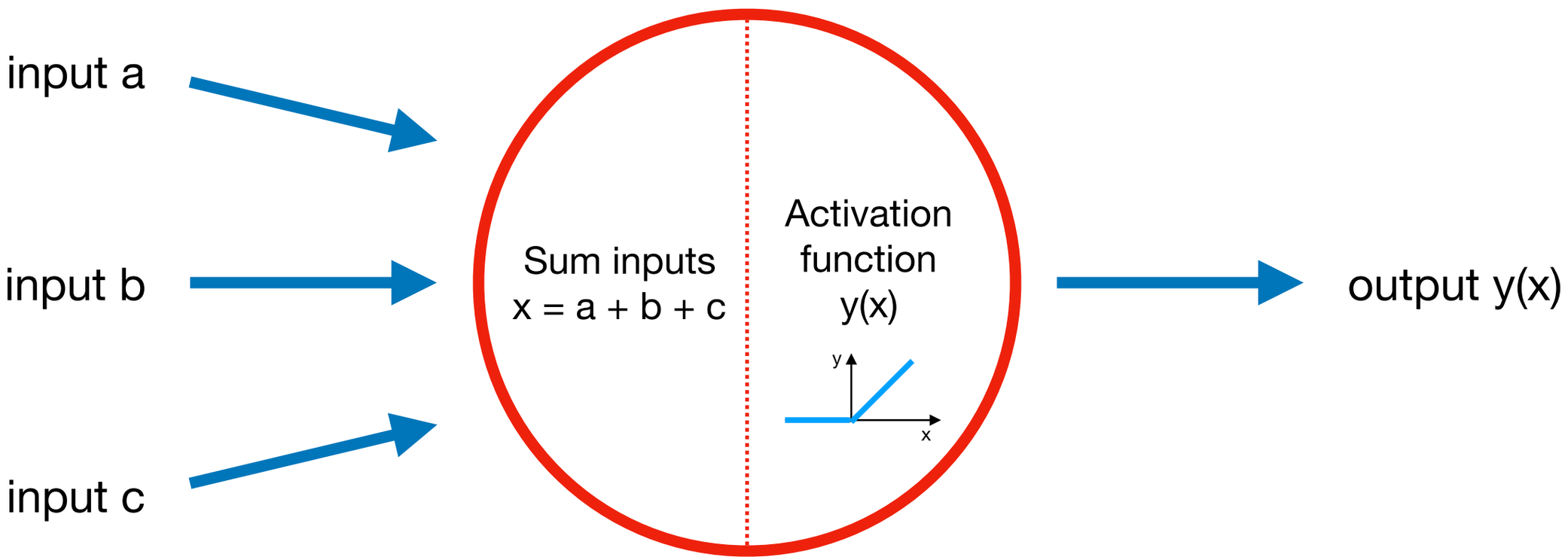
Multiple layers in a biological neural network (human cortex)

Signals (light, sound, etc) are transmitted along the nervous system (i.e. billions of neurons).
Typically organized in layers.

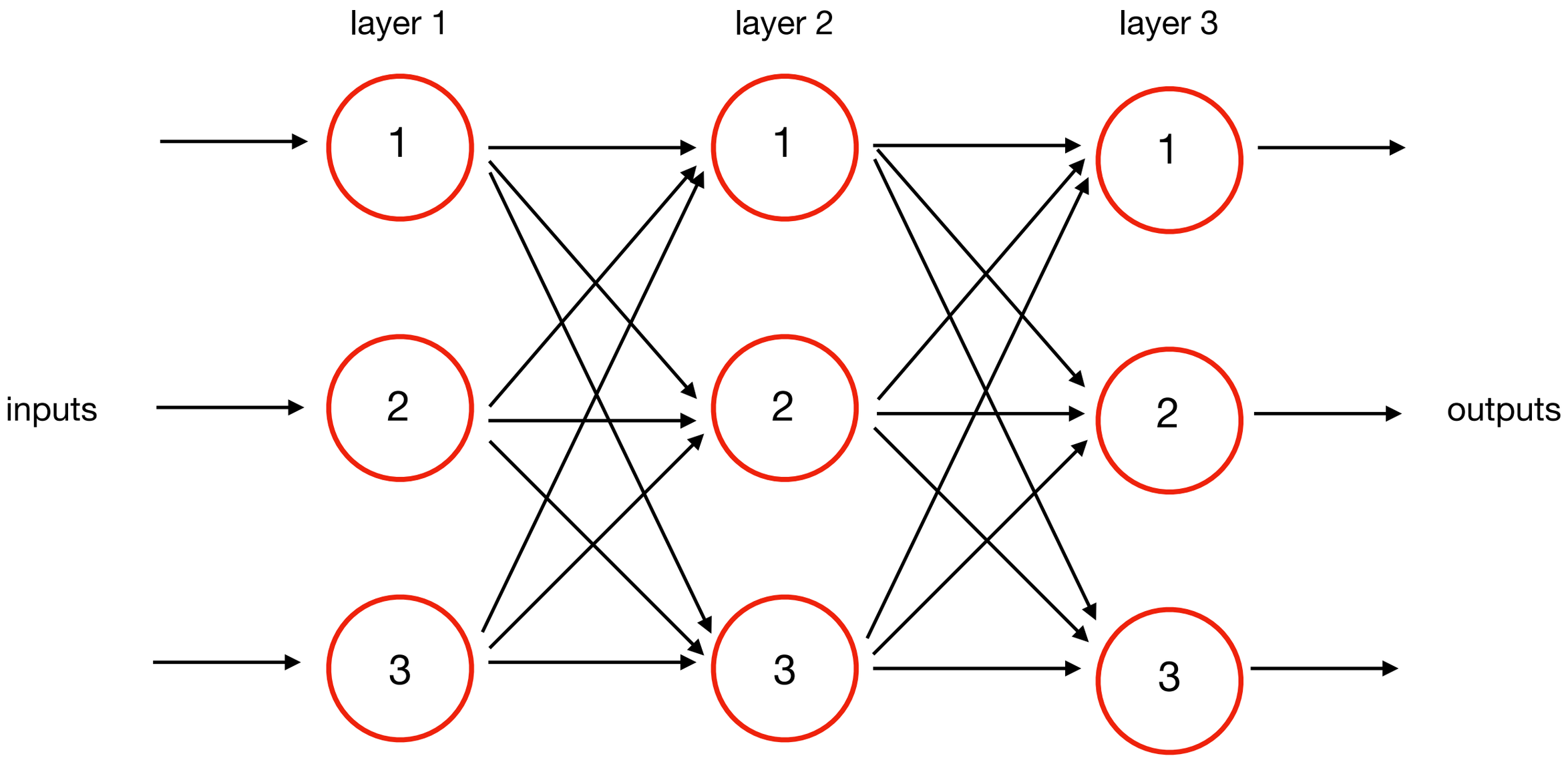


Biological Neurons

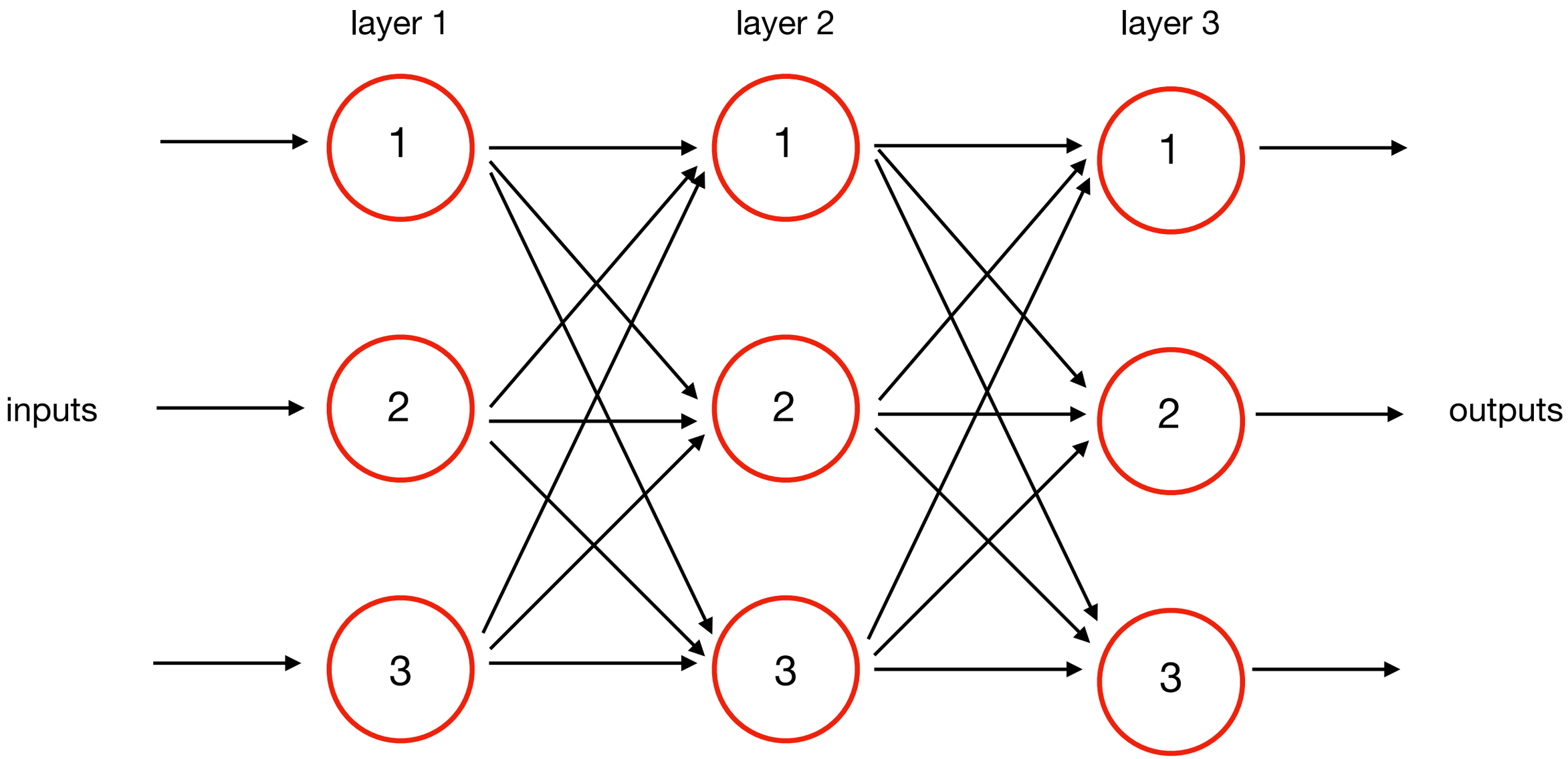


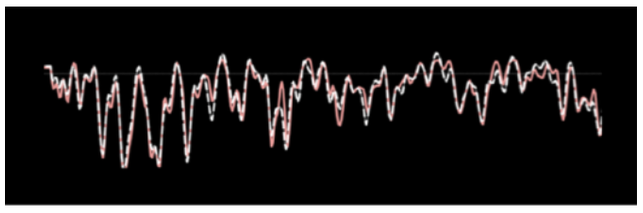


Artificial Neuronal Networks



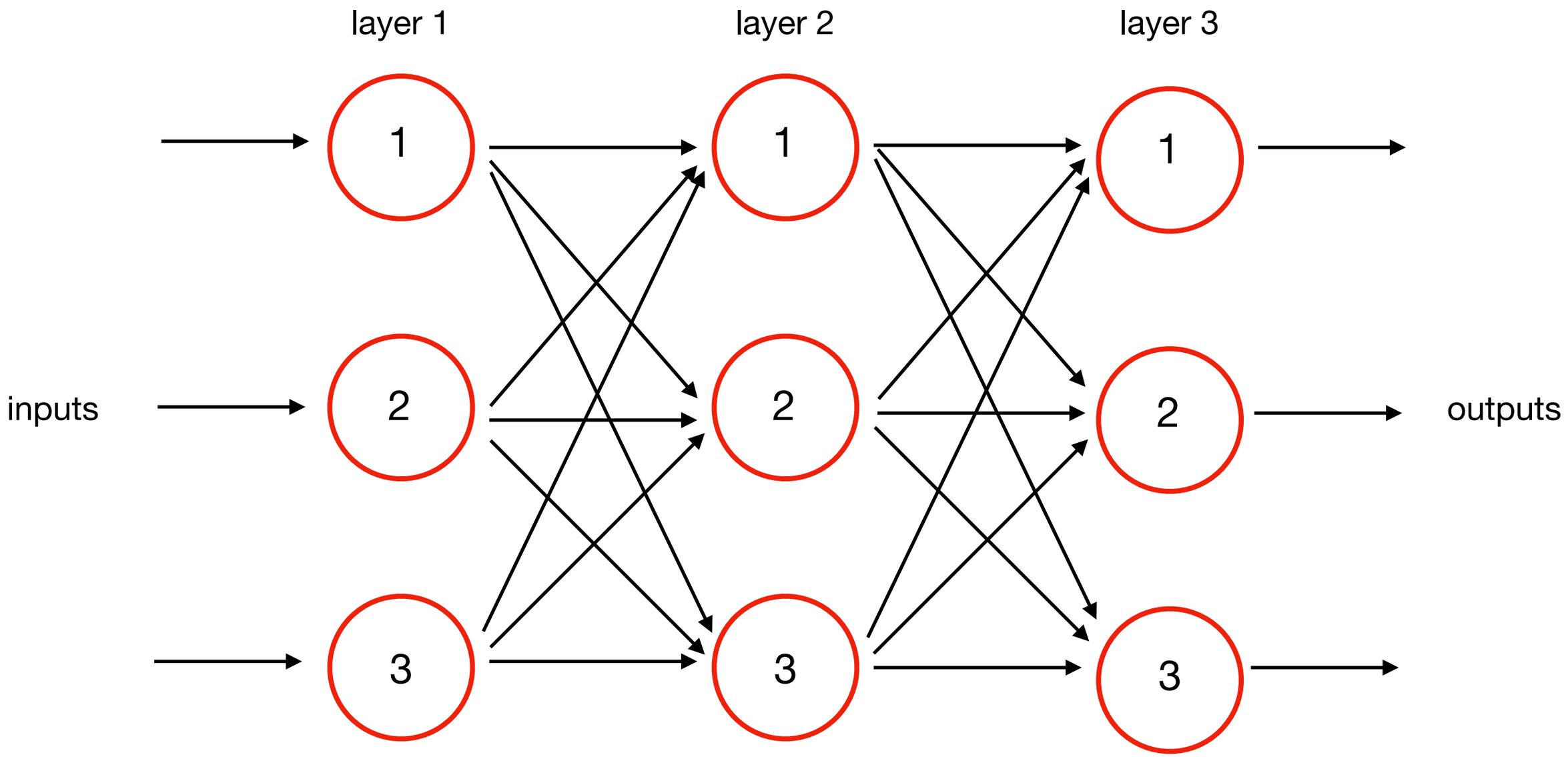
Artificial Neuronal Networks



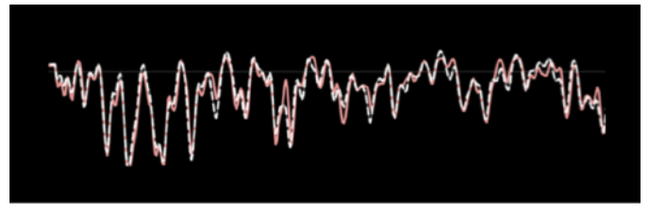


Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])

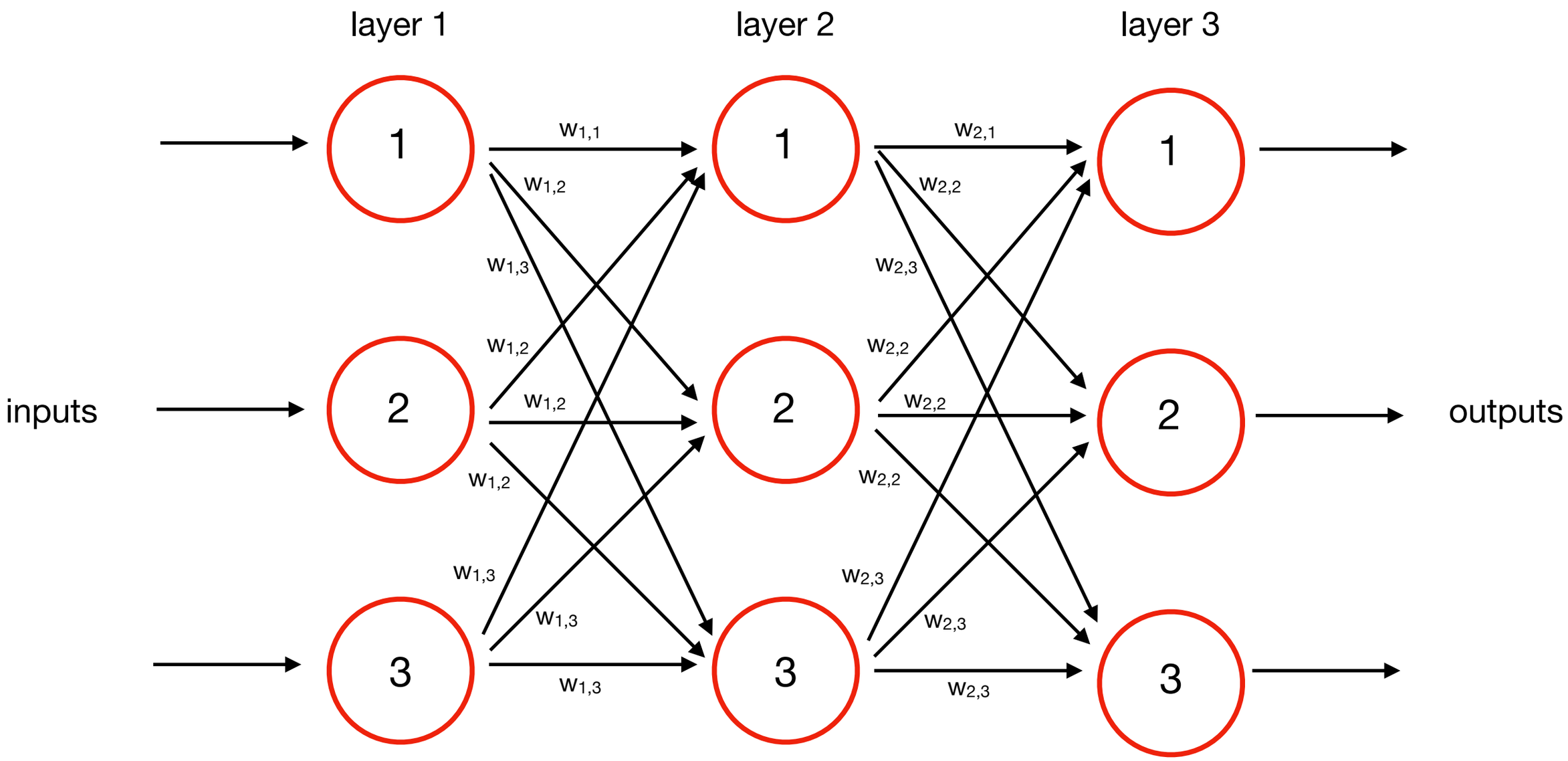
Artificial Neuronal Networks



Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])



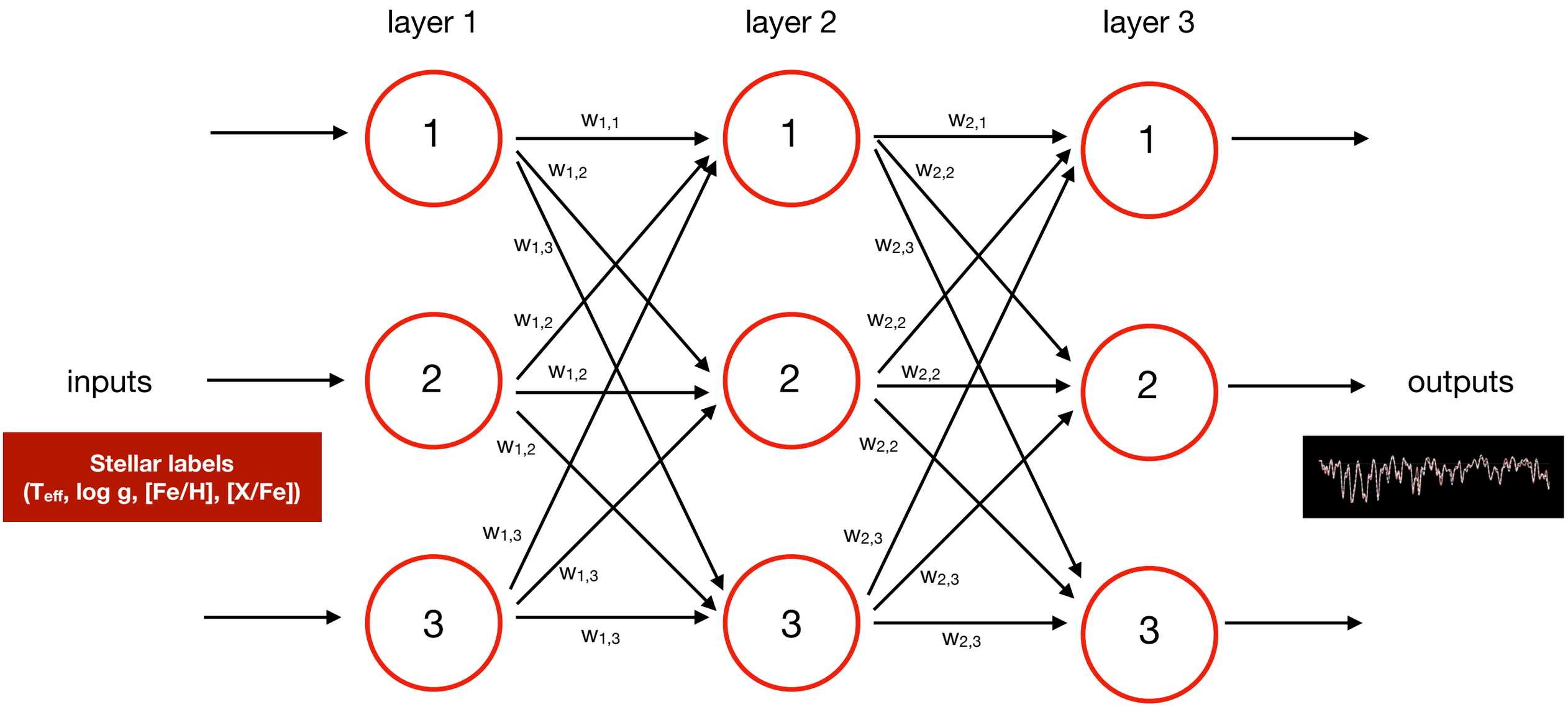
Artificial Neuronal Networks



We can train an ANN to perform a task by adjusting the strenght of each connection with weights.

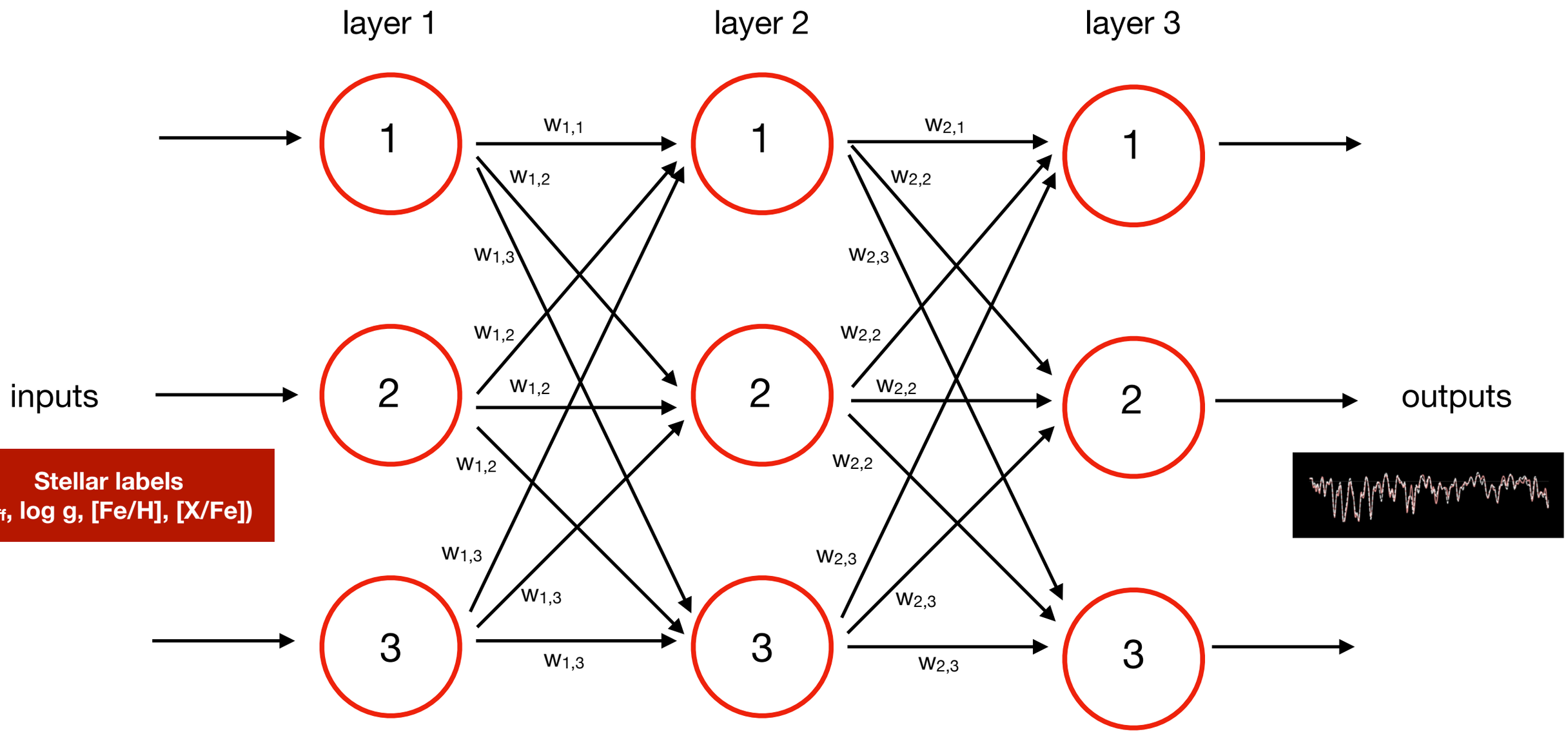
The connection weights are trainable parameters and can be adjusted until we are not satisfied with the performance.

Artificial Neuronal Networks



Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])

Artificial Neuronal Networks

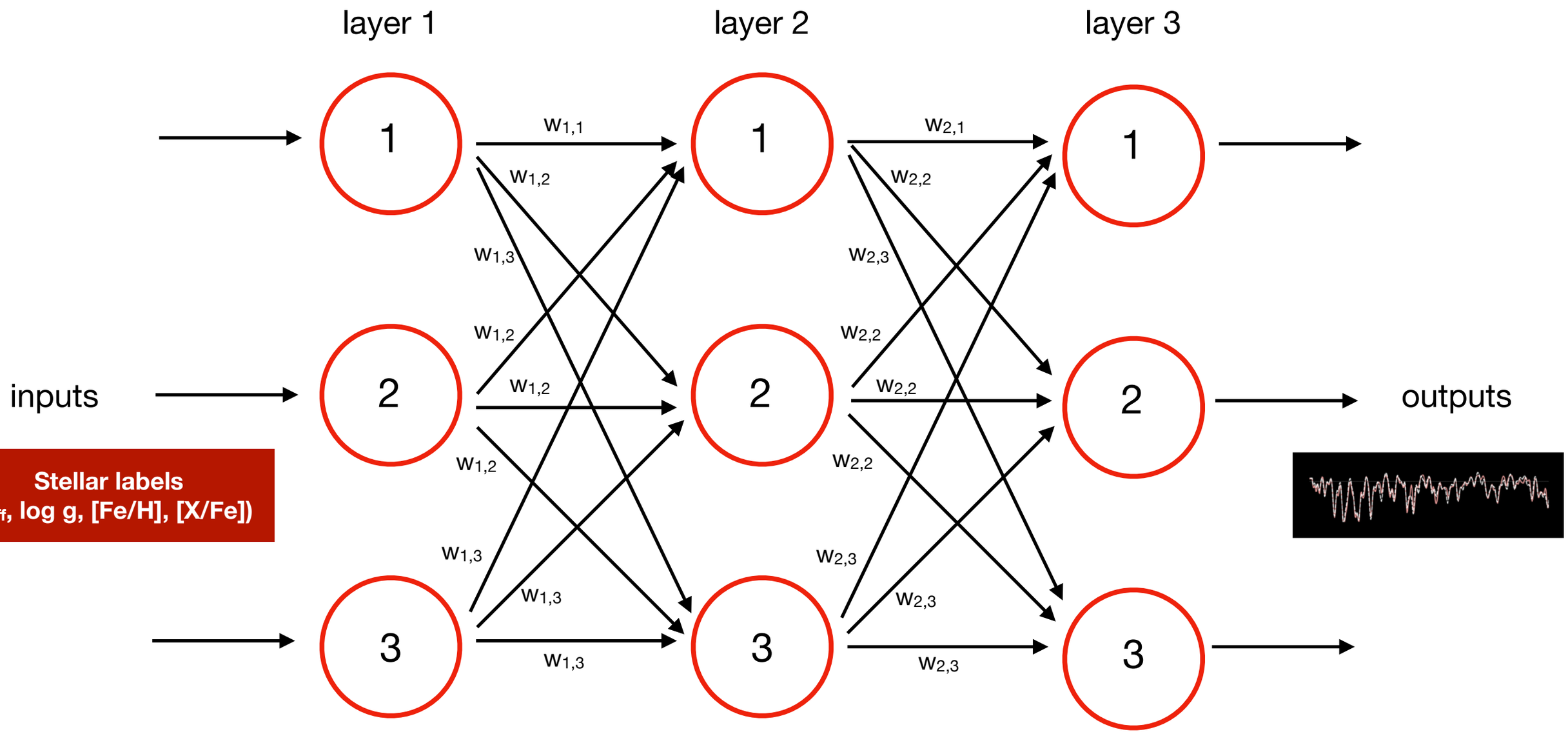


$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

Vector of the final outputs (predictions)

Vector of true values

Artificial Neuronal Networks



Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])

$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

Vector of the final outputs (predictions)

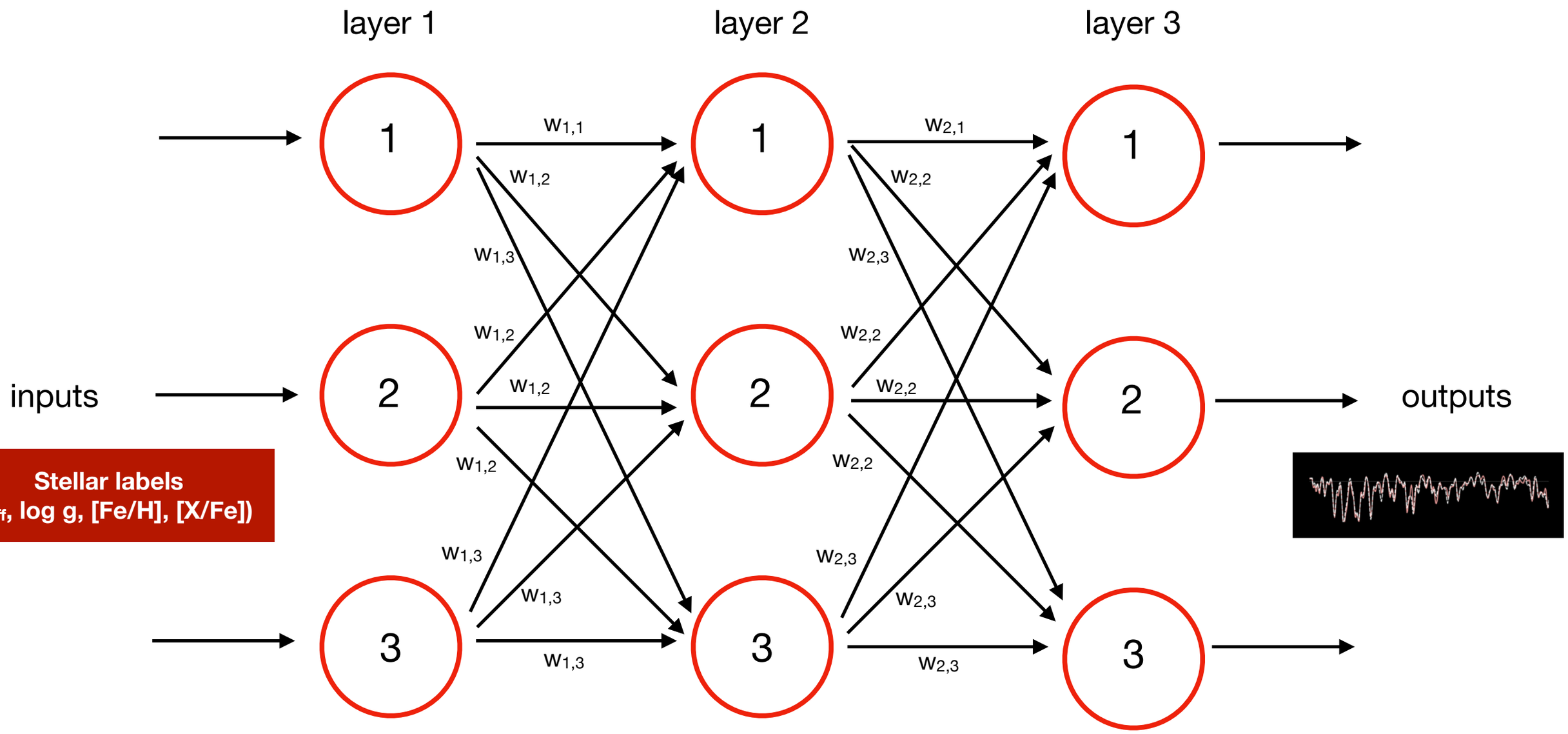
Vector of true values

$$Cost\ function = J(w) = \frac{1}{2n}\sum_{i=1}^{n}[y^i - \hat{y}^i]^2$$

Weights

Number of sample data

Artificial Neuronal Networks



Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])

$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

Vector of the final outputs (predictions)

Vector of true values

$$Cost\ function = J(w) = \frac{1}{2n}\sum_{i=1}^{n}[y^i - \hat{y}^i]^2$$

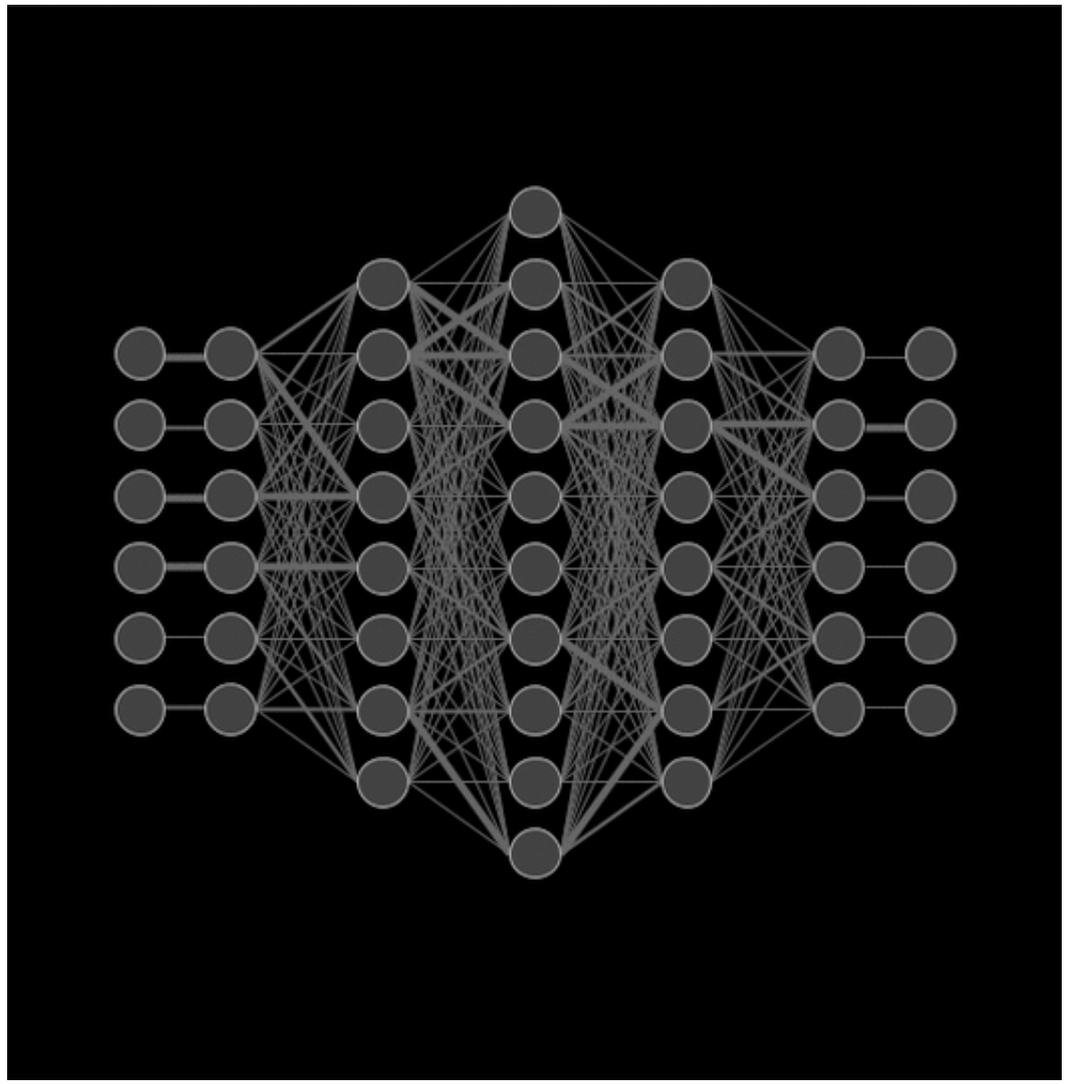
Weights

Number of sample data

$$Derivate\ of\ the\ cost\ function = \frac{\partial J(w)}{\partial w_j} = \frac{1}{n}\sum_{i=1}^{n}[y^i - \hat{y}^i]w_j$$

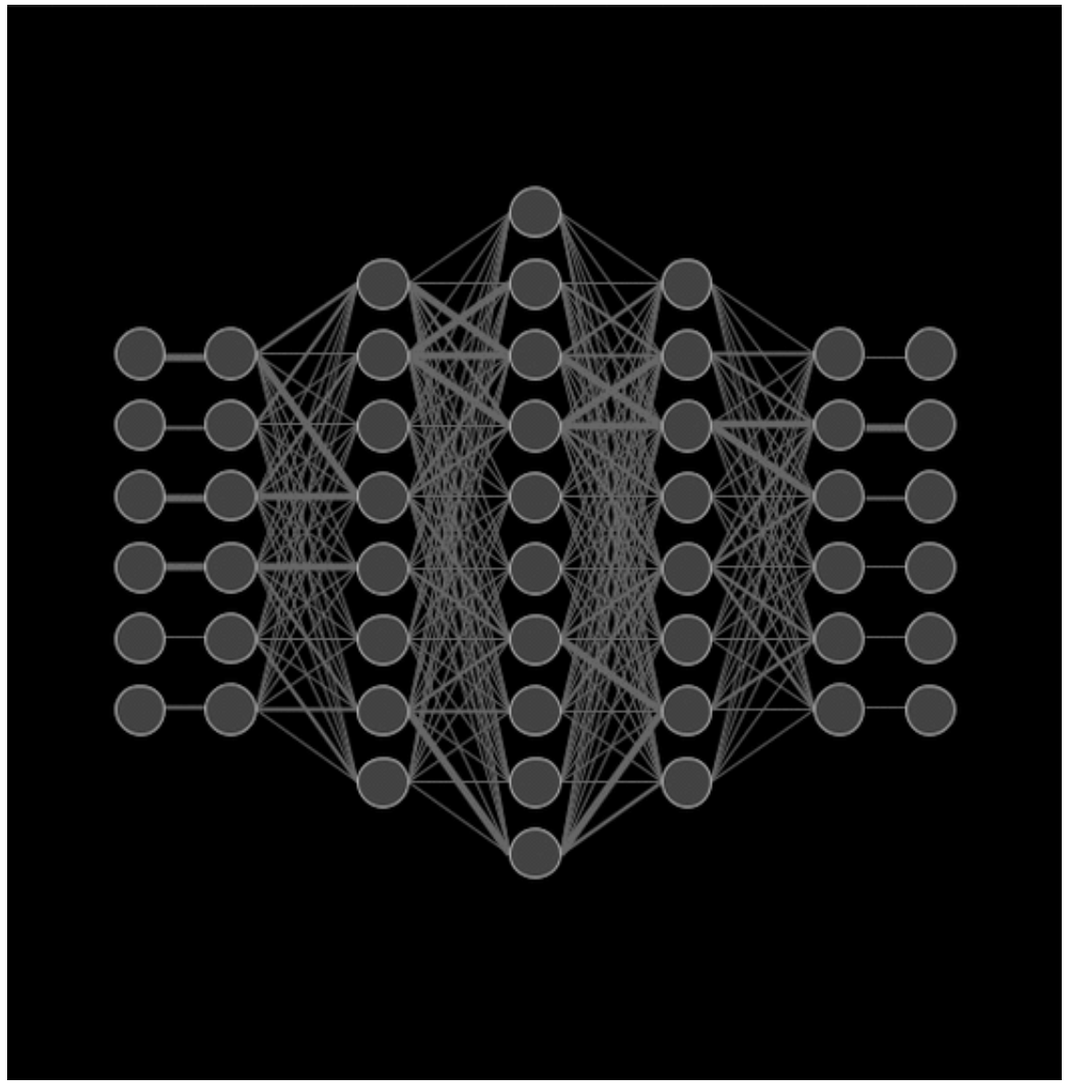
Now you can correct the weights to minimize the cost function.
However, several iterations with small corrections are needed to reach the minimum of J(w).

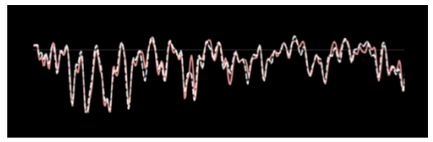
Artificial Neuronal Networks



Artificial Neuronal Networks

Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])

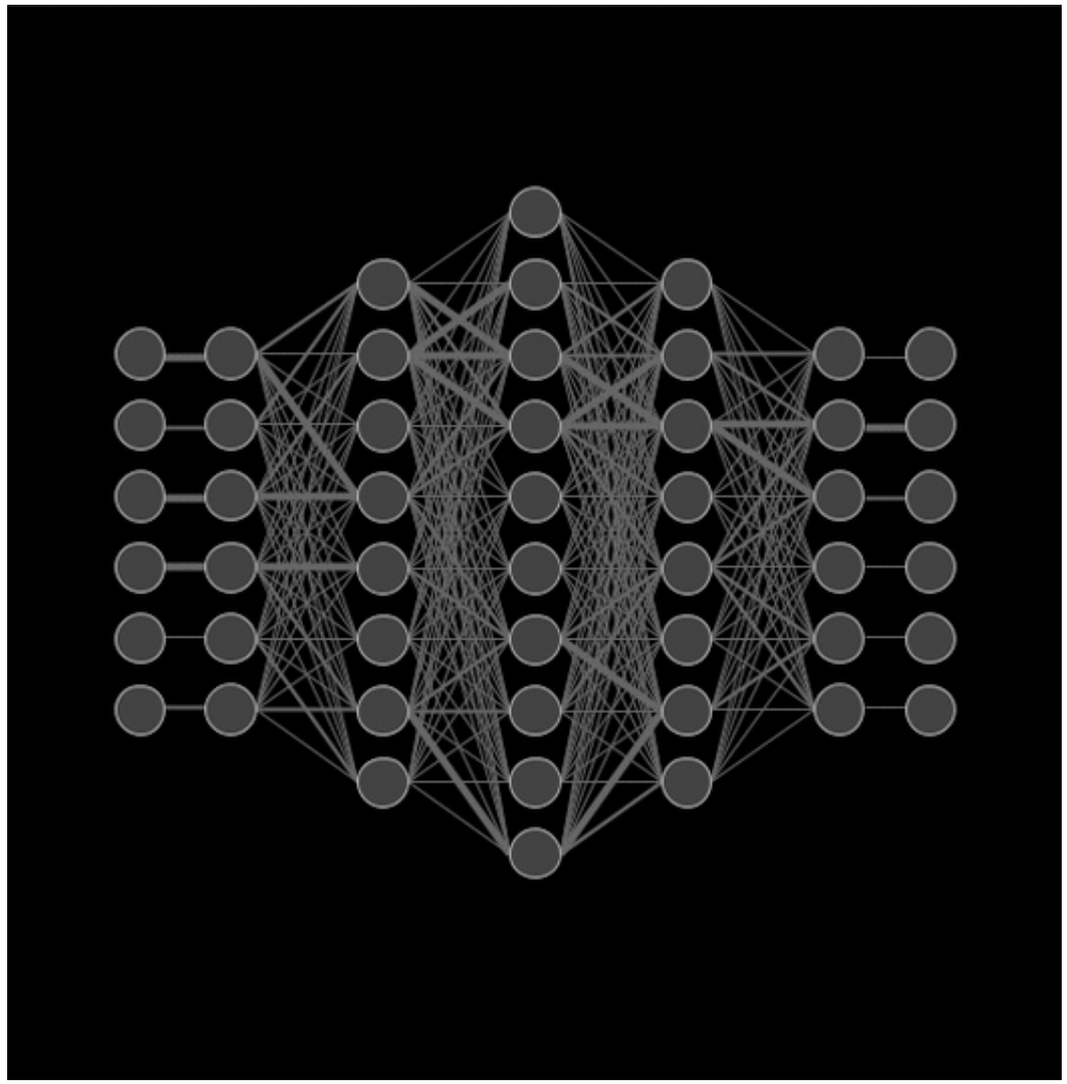


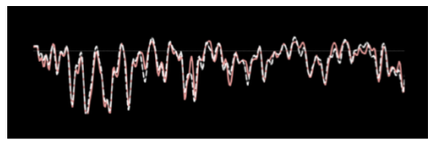


Output: predicted spectrum

Artificial Neuronal Networks

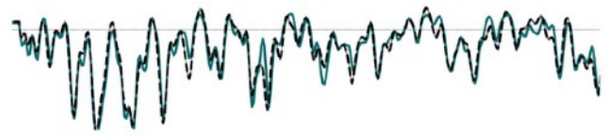
Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])





Output: predicted spectrum

I compare the predicted spectrum with the observed spectrum



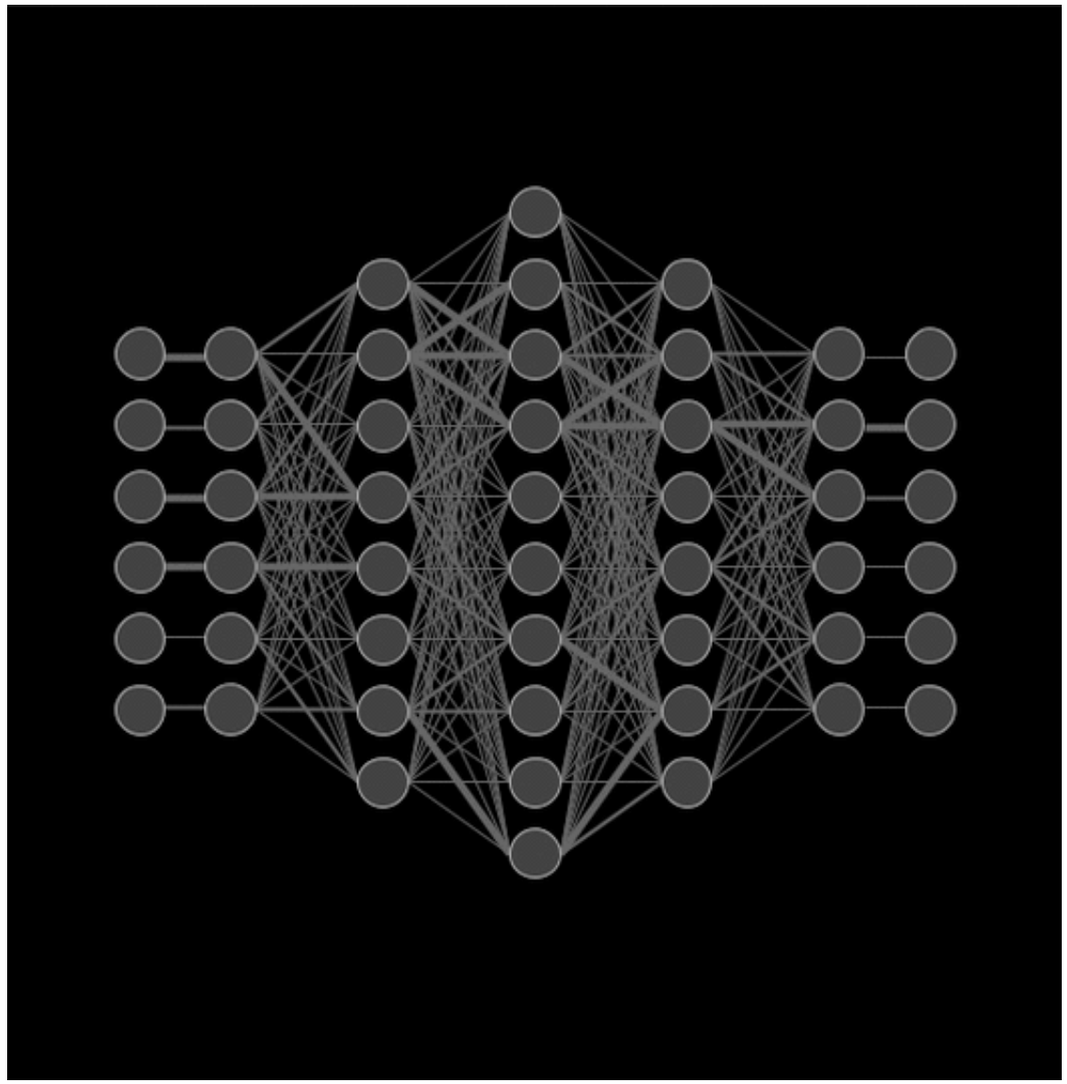
$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

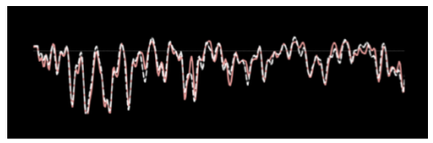
Vector of the final outputs (predictions)

Vector of true values

Artificial Neuronal Networks

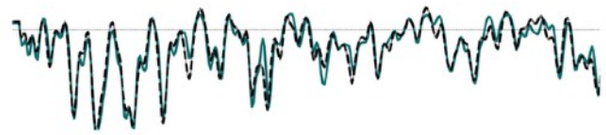
Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])





Output: predicted spectrum

I compare the predicted spectrum with the observed spectrum



$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

Vector of the final outputs (predictions)

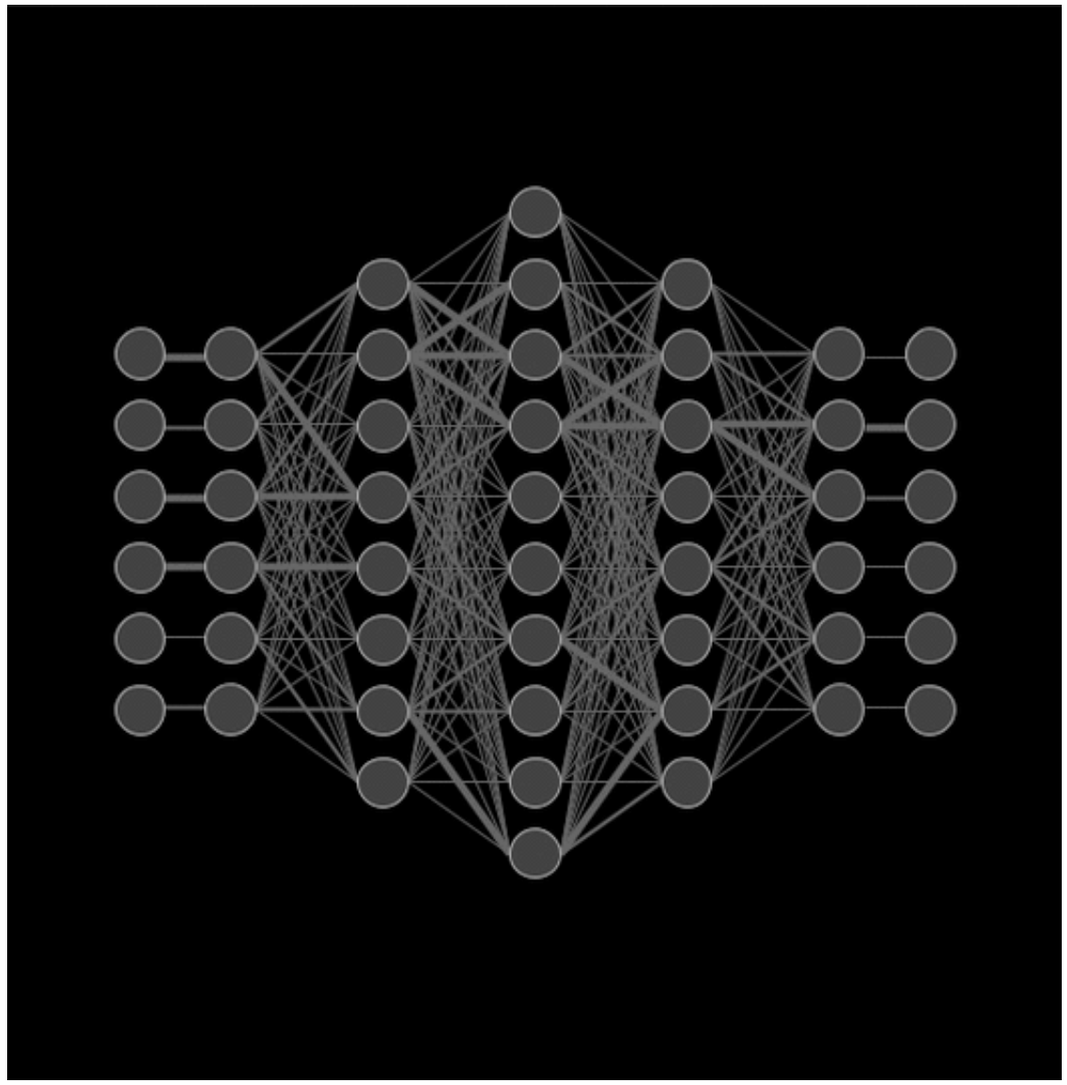
Vector of true values

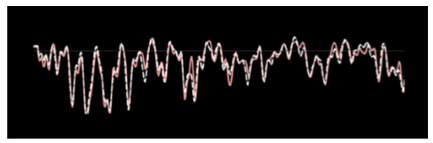
Repeat for every element of the training sample

Calculate the **cost function** and its derivative

Artificial Neuronal Networks

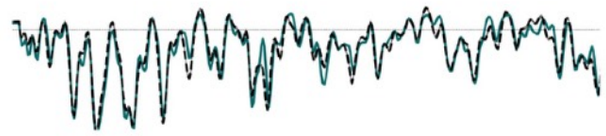
Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])





Output: predicted spectrum

I compare the predicted spectrum with the observed spectrum



$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

Vector of the final outputs (predictions)

Vector of true values

Repeat for every element of the training sample

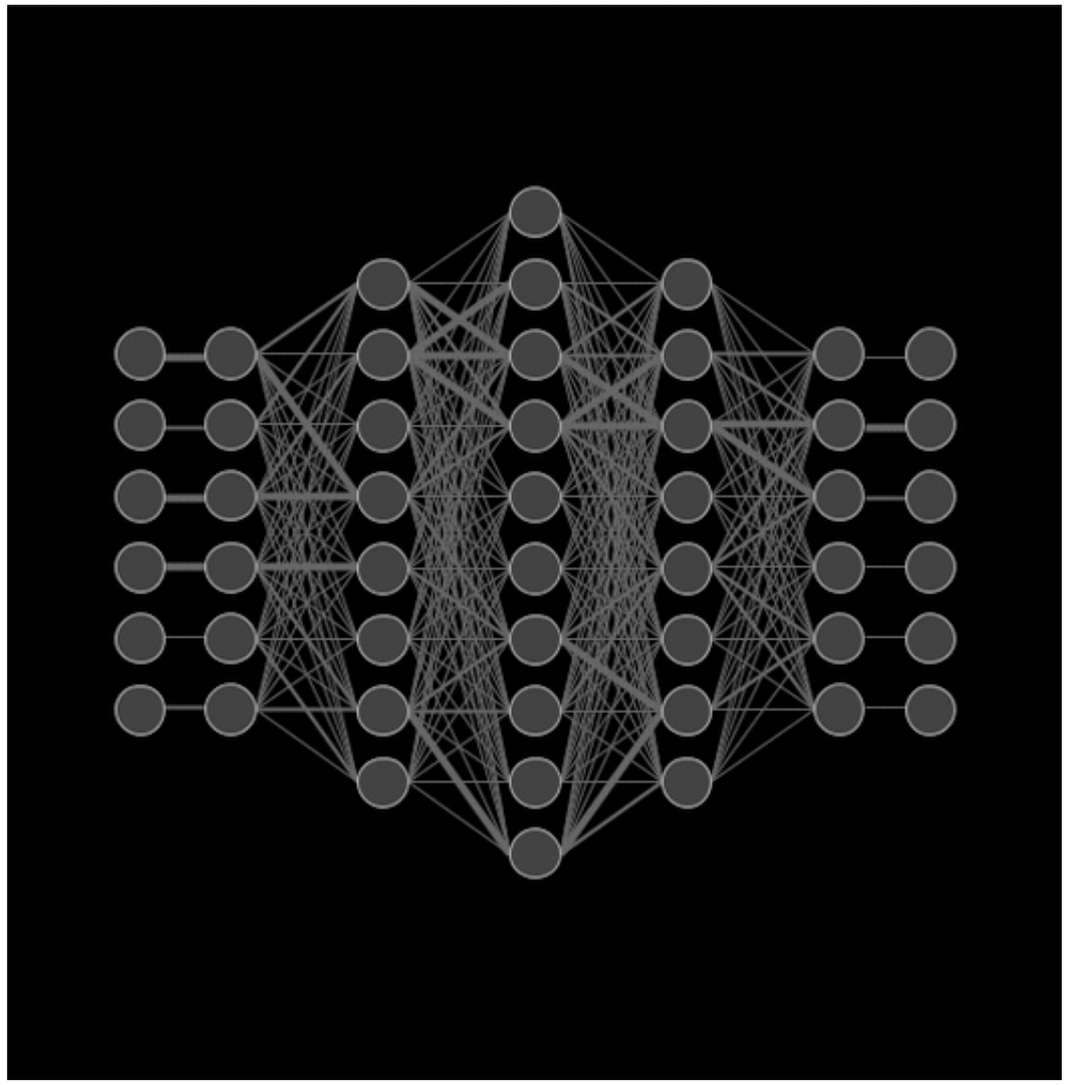
Calculate the **cost function** and its derivative

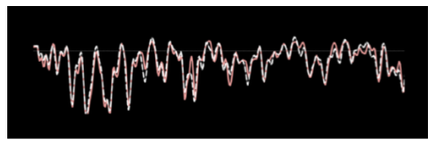
Correct the weights

$$Correction = \frac{\partial J(w)}{\partial w_j} \times learning\ rate$$

Artificial Neuronal Networks

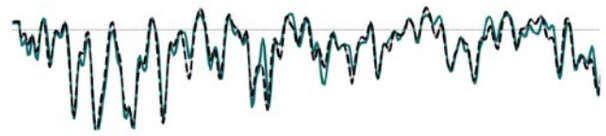
Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])





Output: predicted spectrum

I compare the predicted spectrum with the observed spectrum



$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

Vector of the final outputs (predictions)

Vector of true values

Repeat for every element of the training sample

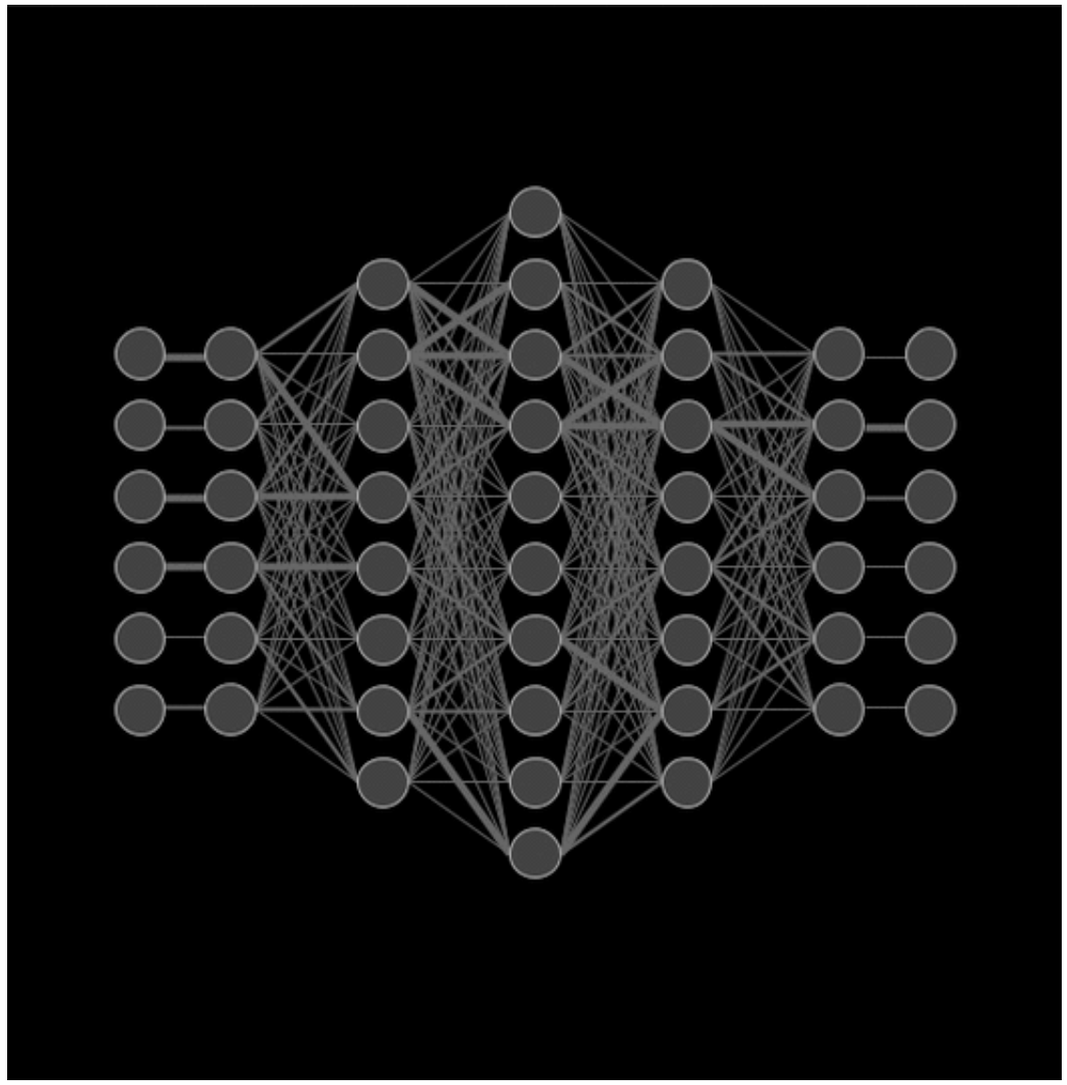
Calculate the **cost function** and its derivative

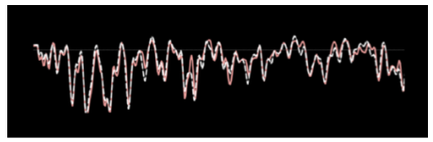
Correct the weights

$$Correction = \frac{\partial J(w)}{\partial w_j} \times learning\ rate$$

Artificial Neuronal Networks

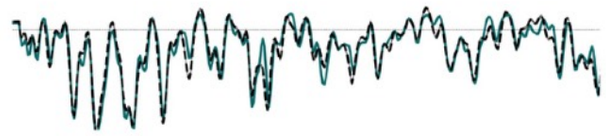
Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])





Output: predicted spectrum

I compare the predicted spectrum with the observed spectrum



$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

Vector of the final outputs (predictions)

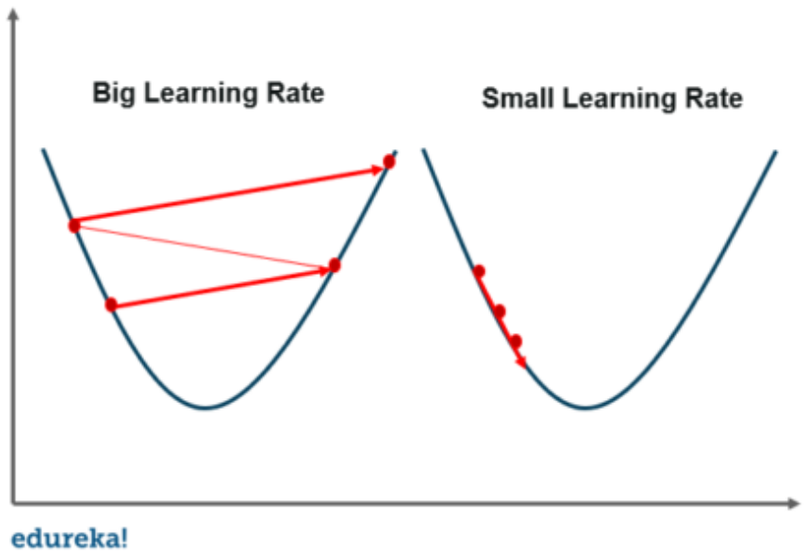
Vector of true values

Repeat for every element of the training sample

Calculate the **cost function** and its derivative

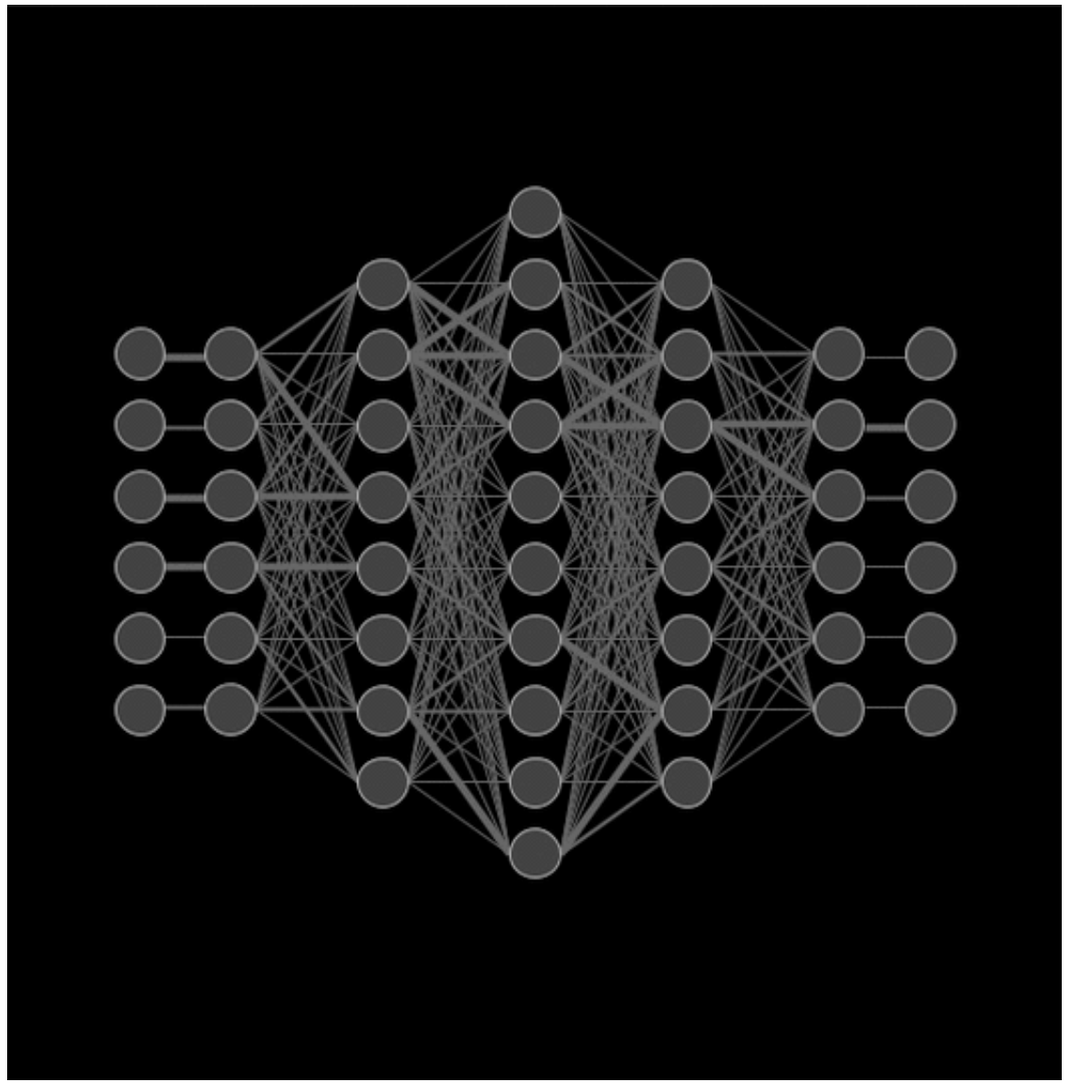
Correct the weights

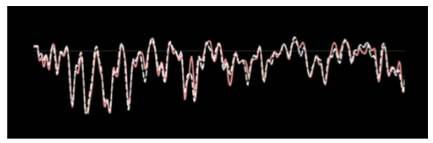
$$Correction = \frac{\partial J(w)}{\partial w_j} \times learning\ rate$$



Artificial Neuronal Networks

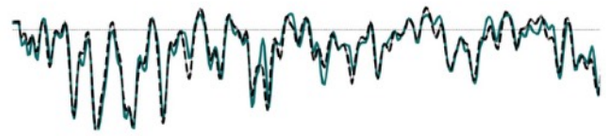
Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])





Output: predicted spectrum

I compare the predicted spectrum with the observed spectrum



$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

Vector of the final outputs (predictions)

Vector of true values

Repeat for every element of the training sample

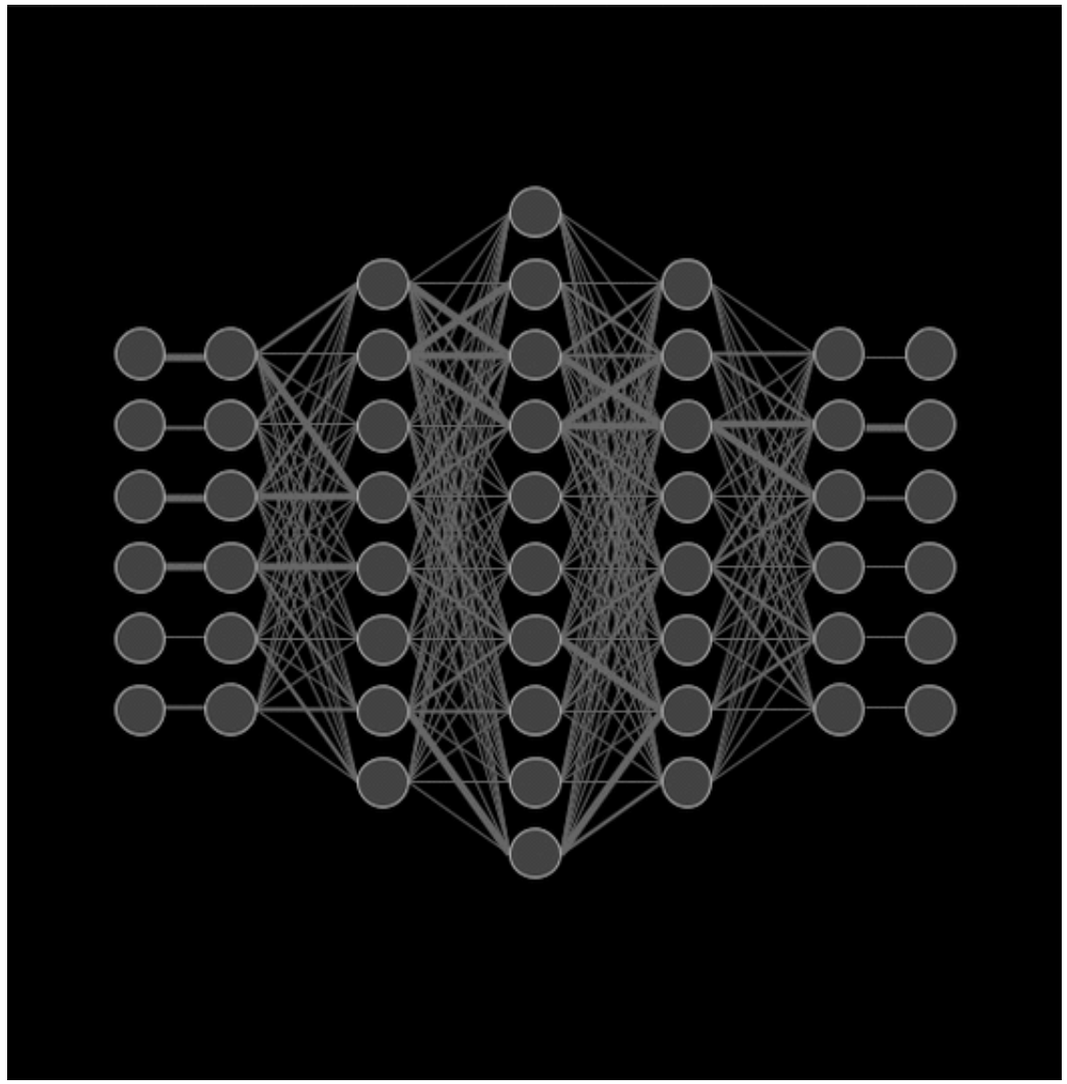
Calculate the **cost function** and its derivative

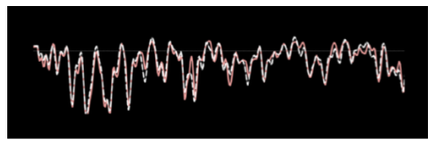
Correct the weights

$$Correction = \frac{\partial J(w)}{\partial w_j} \times learning\ rate$$

Artificial Neuronal Networks

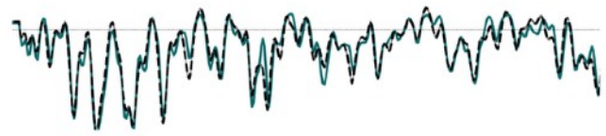
Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])





Output: predicted spectrum

I compare the predicted spectrum with the observed spectrum



$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

Vector of the final outputs (predictions)

Vector of true values

Repeat for every element of the training sample

Calculate the **cost function** and its derivative

Correct the weights

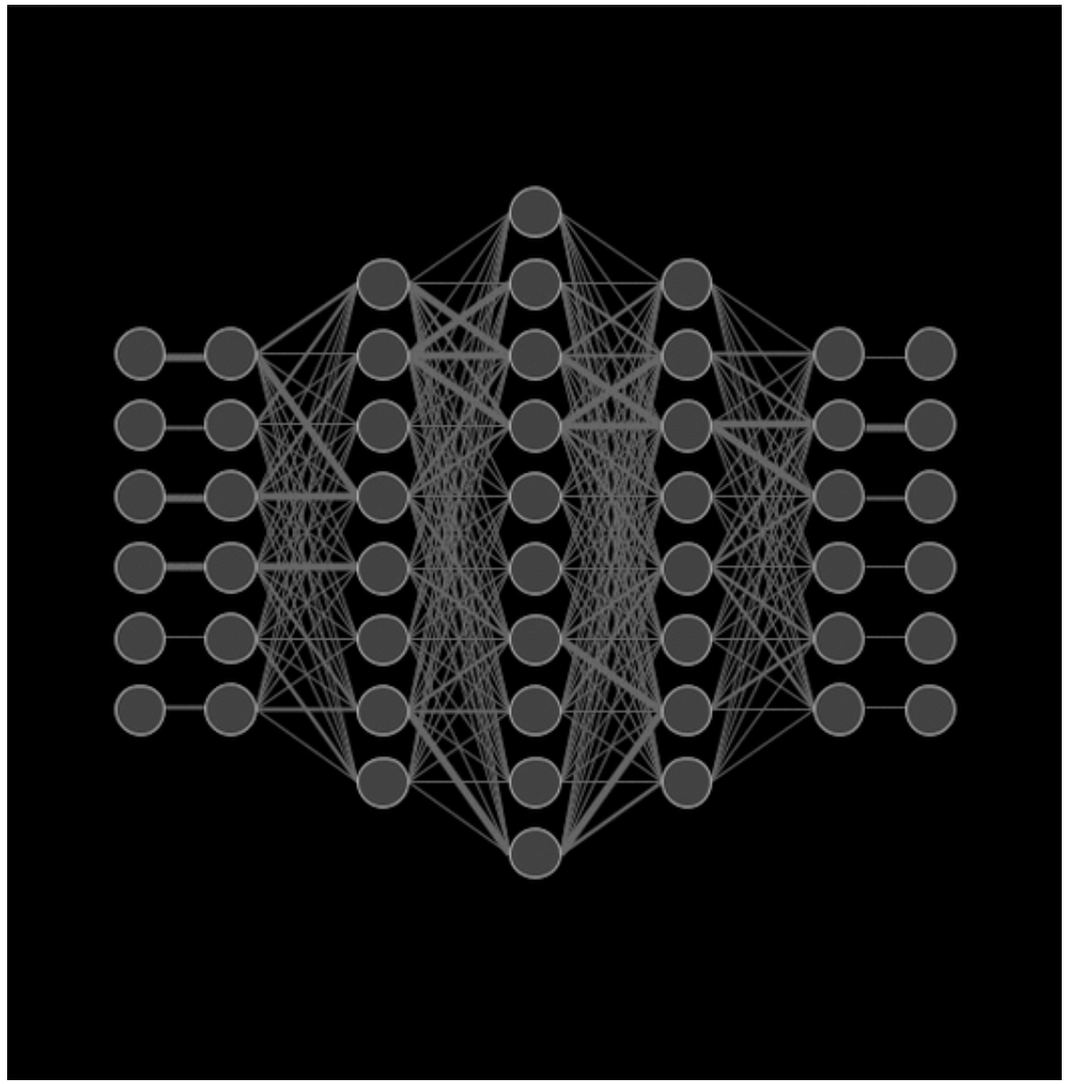
$$Correction = \frac{\partial J(w)}{\partial w_j} \times learning\ rate$$

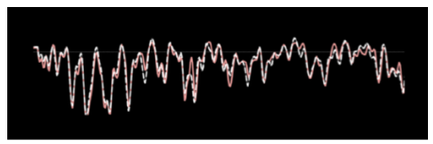
Problem.

Sometimes training samples are composed by thousands of elements. It is highly inefficient to pass the training sample through the ANN all in once.

Artificial Neuronal Networks

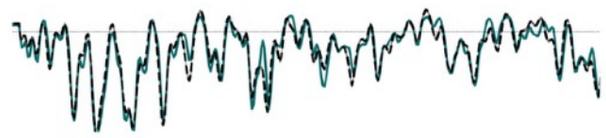
Stellar labels
(T_{eff}, log g, [Fe/H], [X/Fe])





Output: predicted spectrum

I compare the predicted spectrum with the observed spectrum



$$Loss\ function = \frac{1}{2}[y^i - \hat{y}^i]^2$$

Vector of the final outputs (predictions)

Vector of true values

Repeat for every element of the training sample

Calculate the **cost function** and its derivative

Correct the weights

$$Correction = \frac{\partial J(w)}{\partial w_j} \times learning\ rate$$

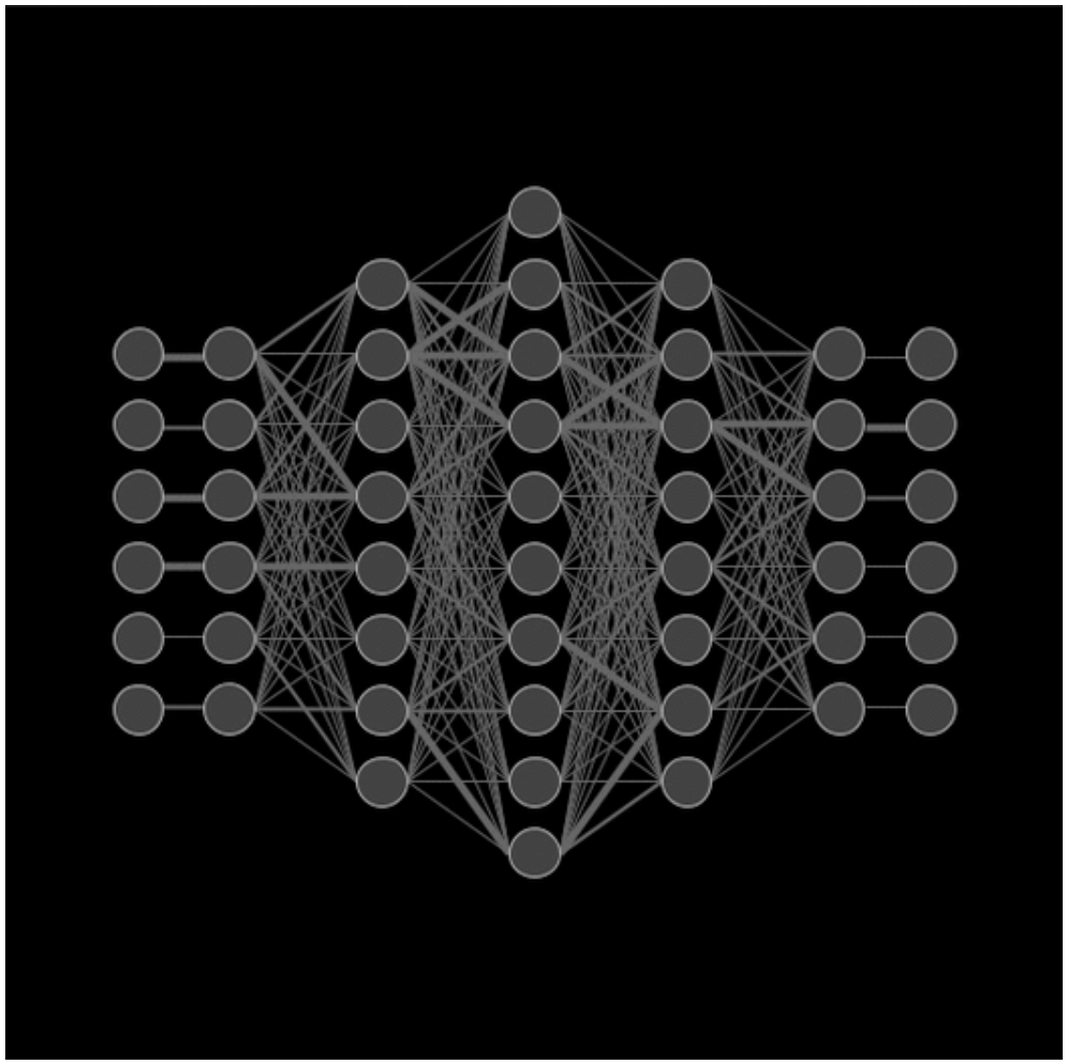
Problem.

Sometimes training samples are composed by thousands of elements. It is highly inefficient to pass the training sample through the ANN all in once.

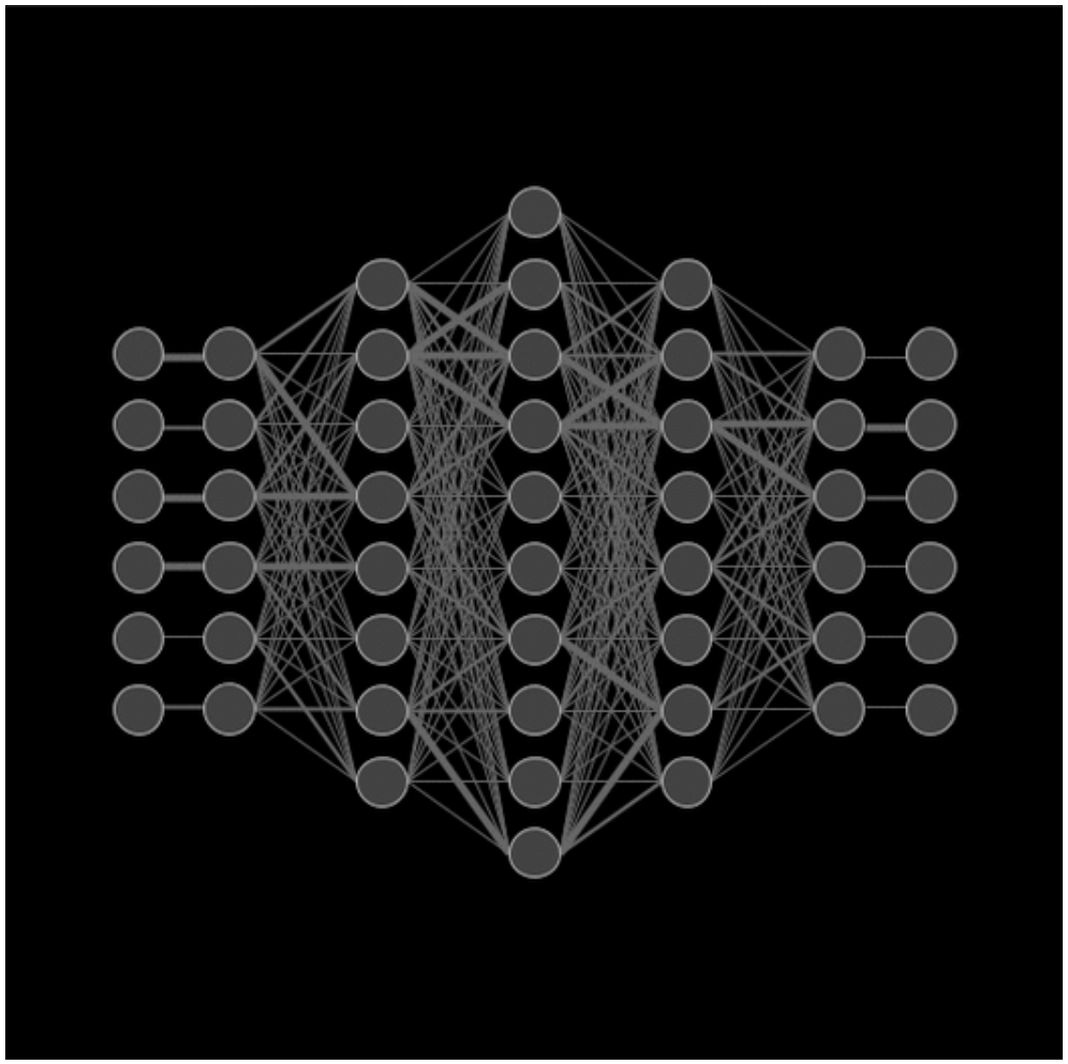
Soution.

The training sample is divided in subsamples of N elements. When each subsamble has passed through the ANN, the weights are corrected. The epoch ends when all the elements of the training sample have passed through the ANN. The size of each subsample is called "batch size".

Artificial Neuronal Networks

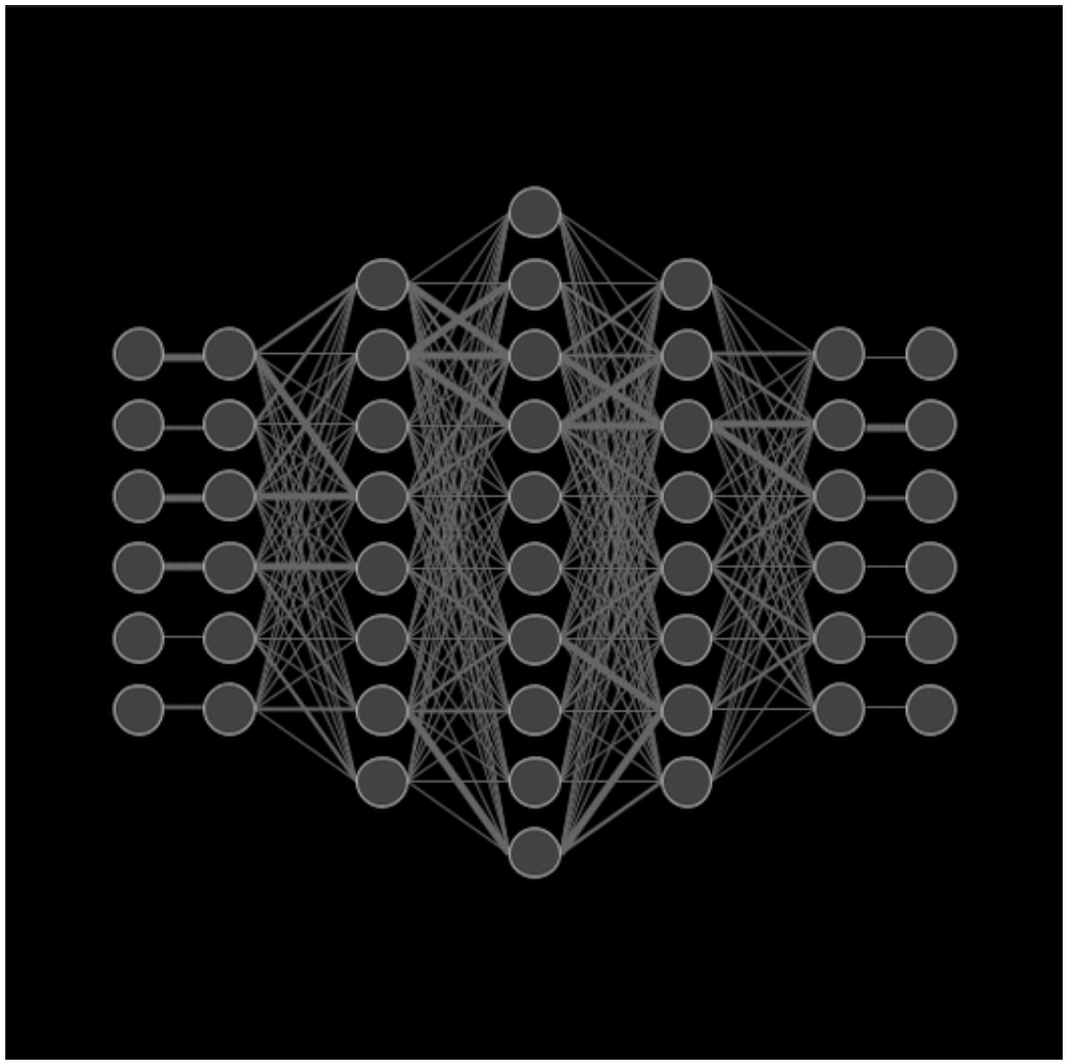


Artificial Neuronal Networks



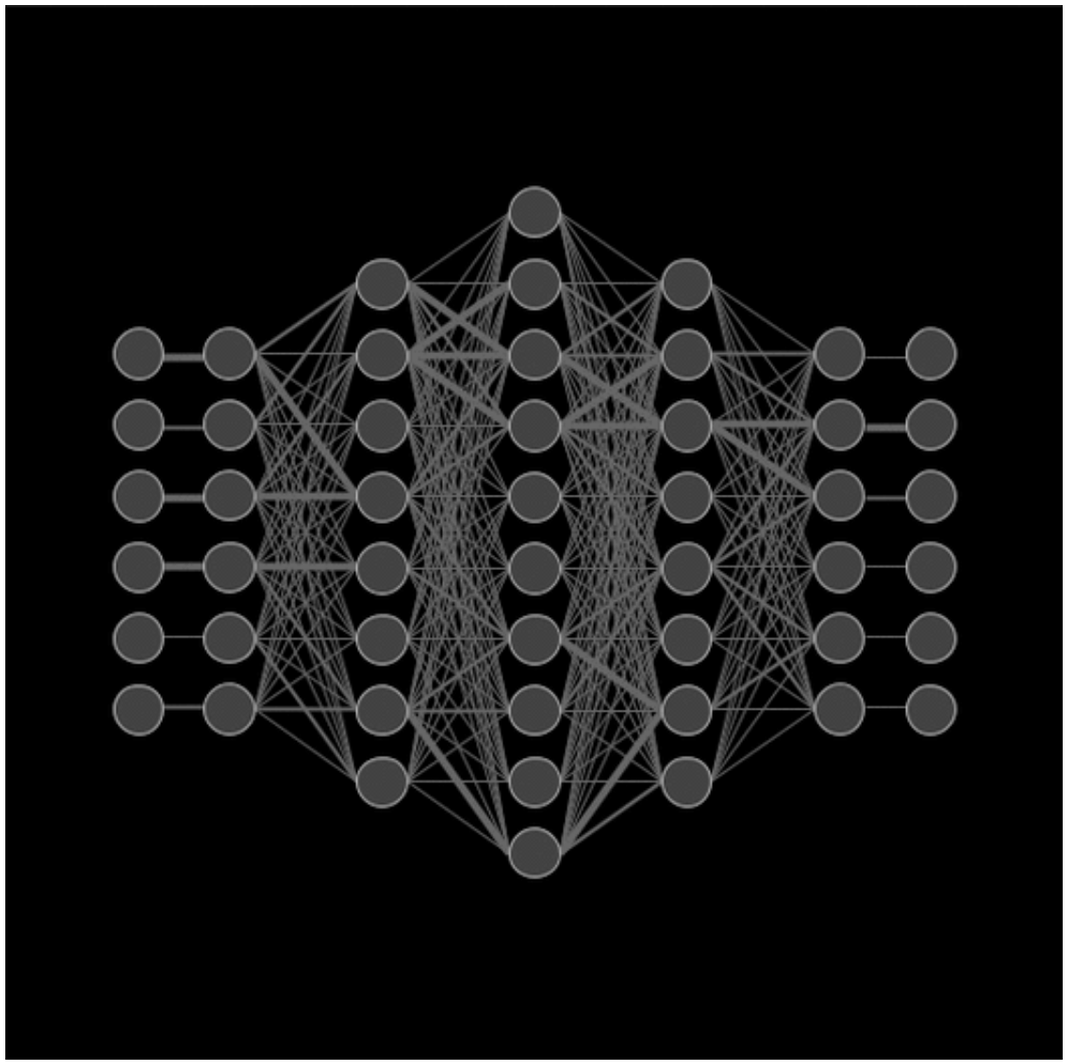
Artificial Neuronal Networks

Training sample



Artificial Neuronal Networks

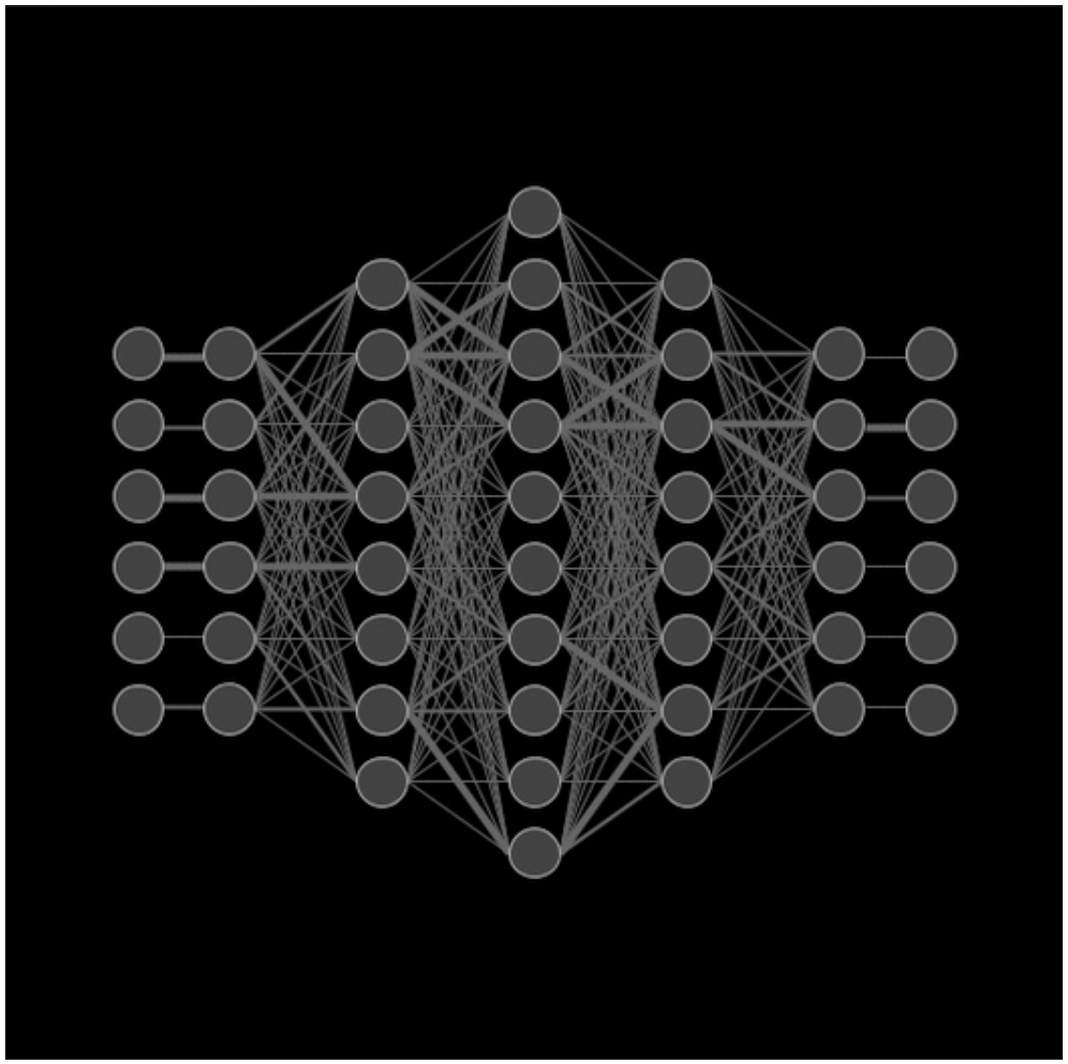
Training sample



Cost function training sample

Artificial Neuronal Networks

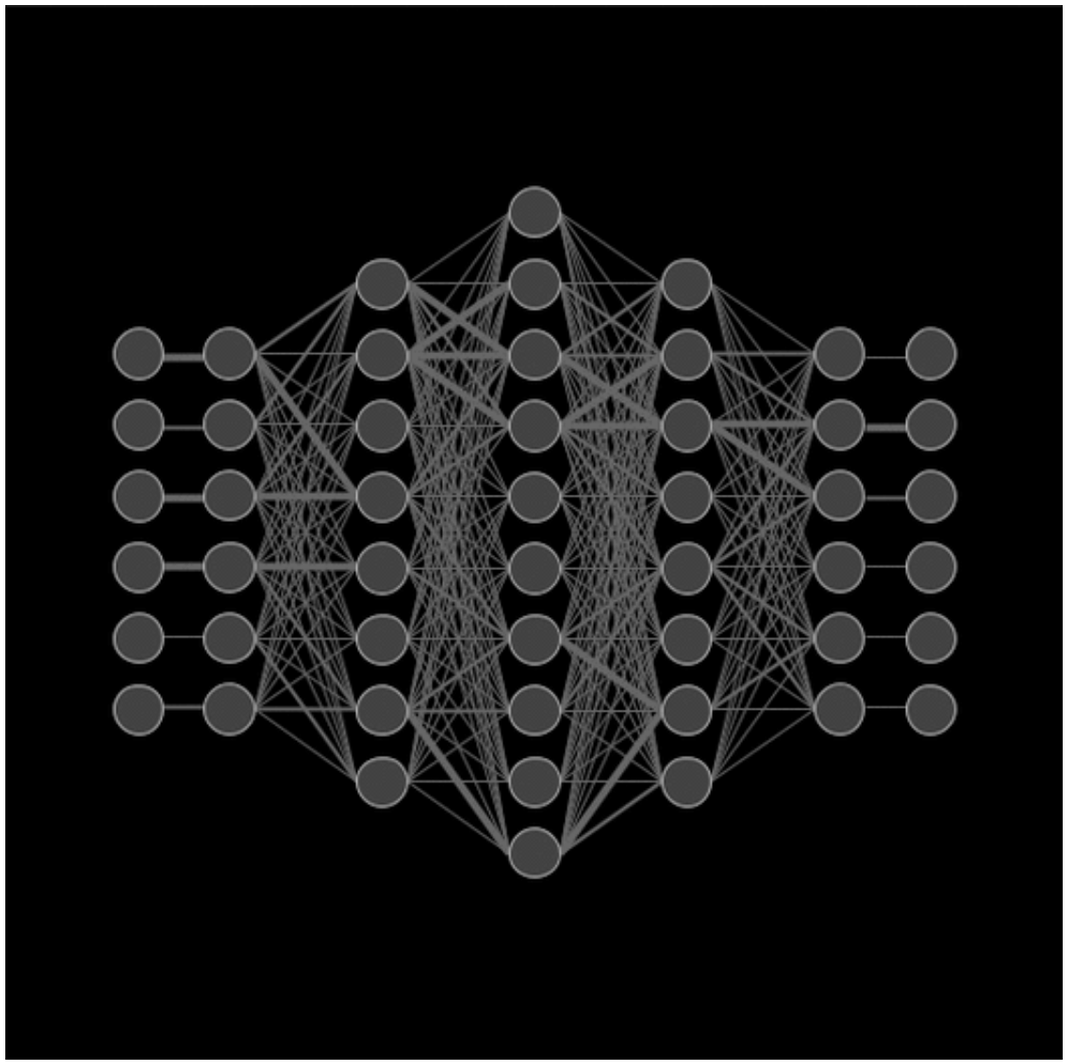
Training sample
[Test sample]



Cost function training sample

Artificial Neuronal Networks

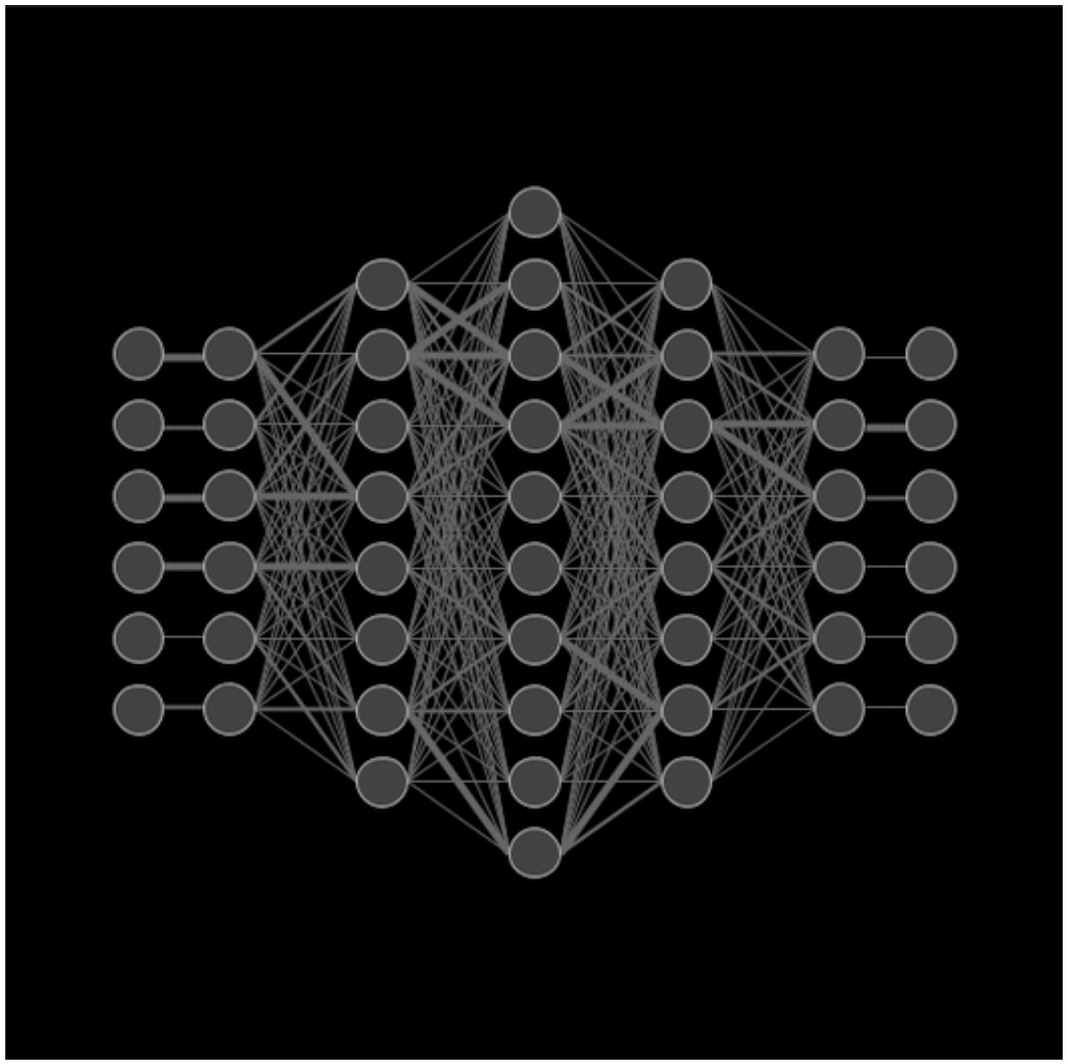
Training sample
[Test sample]



Cost function training sample
[Cost function test sample]

Artificial Neuronal Networks

Training sample
[Test sample]



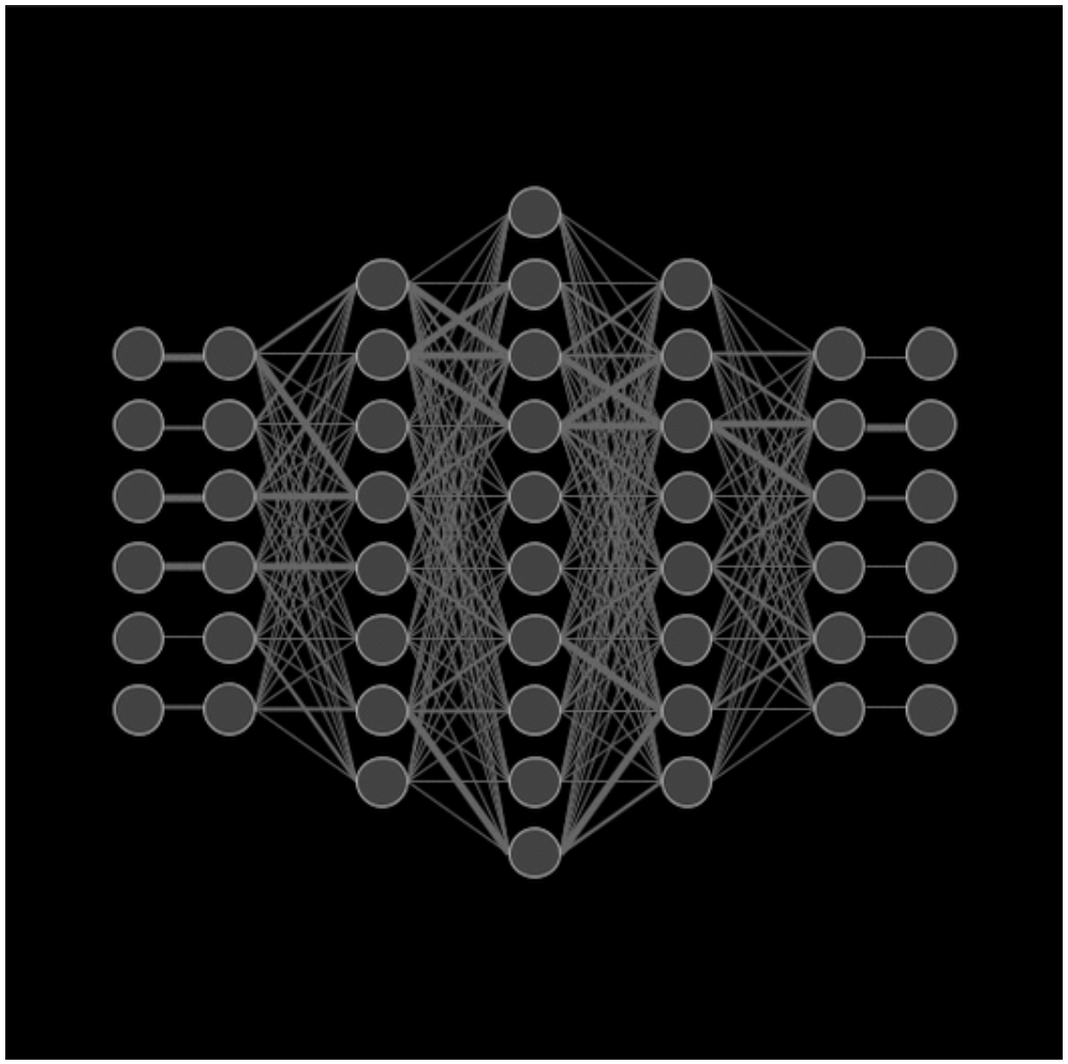
Cost function training sample
[Cost function test sample]
Weights correction

Artificial Neuronal Networks

Epoch 1

Training sample

[Test sample]



Cost function training sample

[Cost function test sample]

Weights correction

Artificial Neuronal Networks

Epoch 1

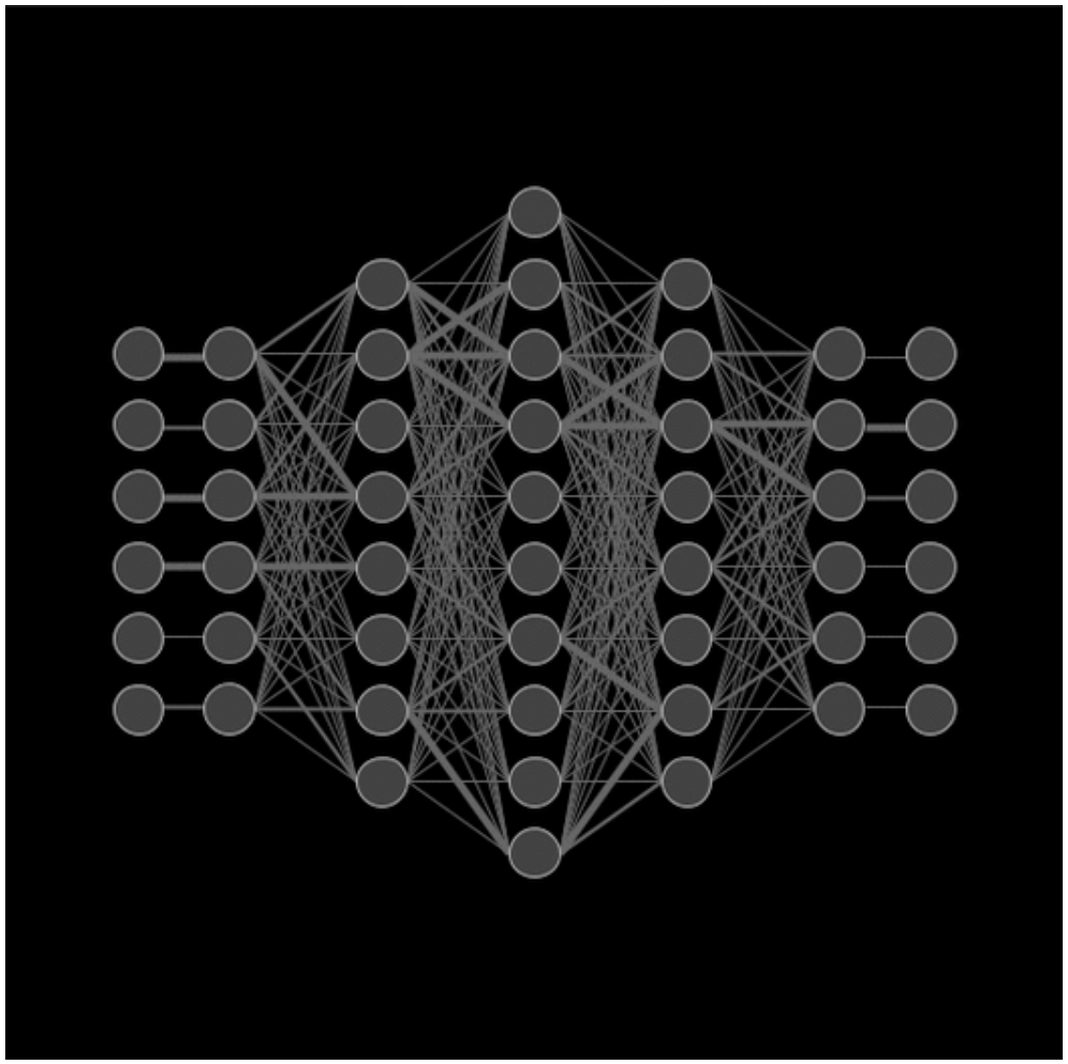
Training sample

[Test sample]

Epoch 2

Training sample

[Test sample]



Cost function training sample

[Cost function test sample]

Weights correction

Cost function training sample

[Cost function test sample]

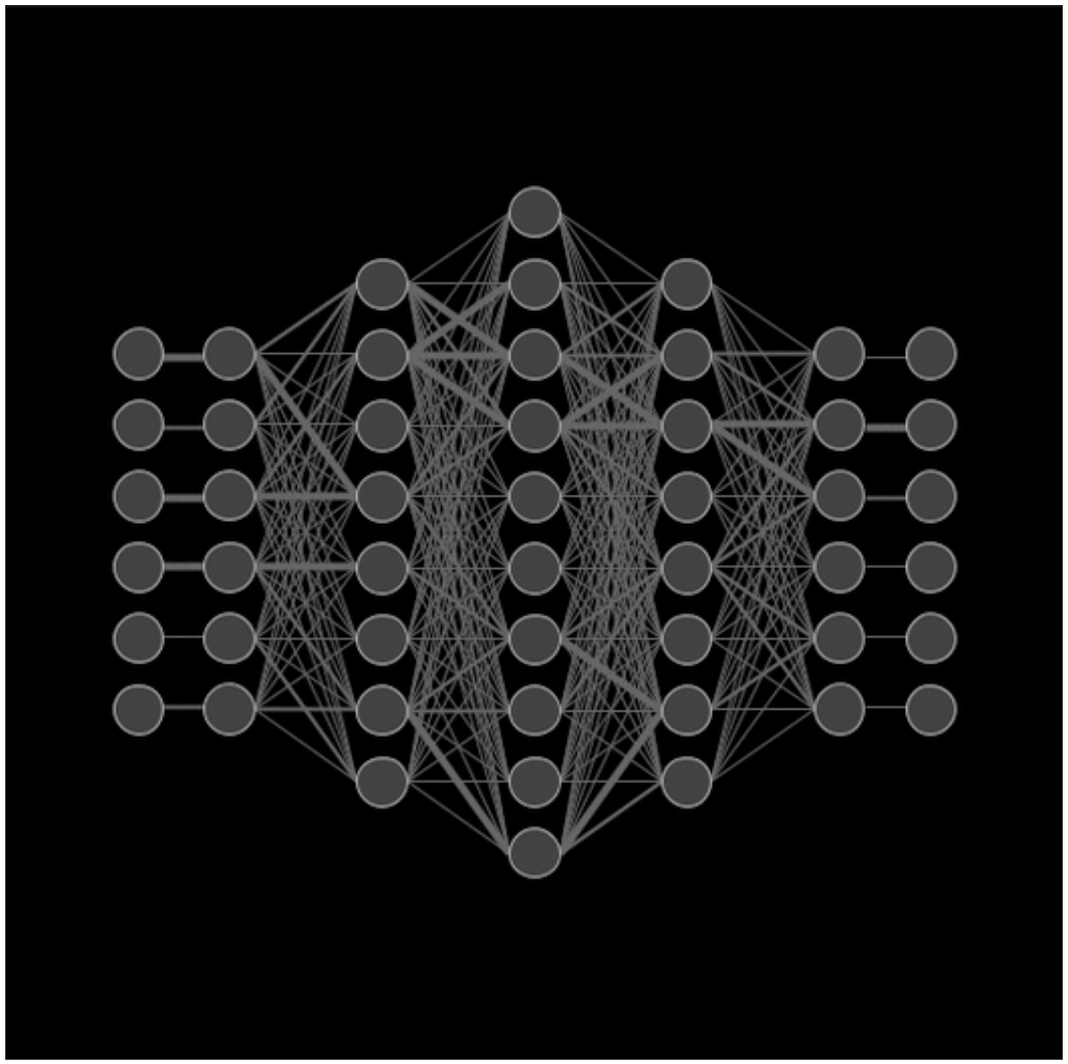
Weights correction

Artificial Neuronal Networks

Epoch 1
Training sample
[Test sample]

Epoch 2
Training sample
[Test sample]

Epoch 3
Training sample
[Test sample]



Cost function training sample
[Cost function test sample]
Weights correction

Cost function training sample
[Cost function test sample]
Weights correction

Cost function training sample
[Cost function test sample]
Weights correction

Artificial Neuronal Networks

Epoch 1
Training sample
[Test sample]

Cost function training sample
[Cost function test sample]
Weights correction

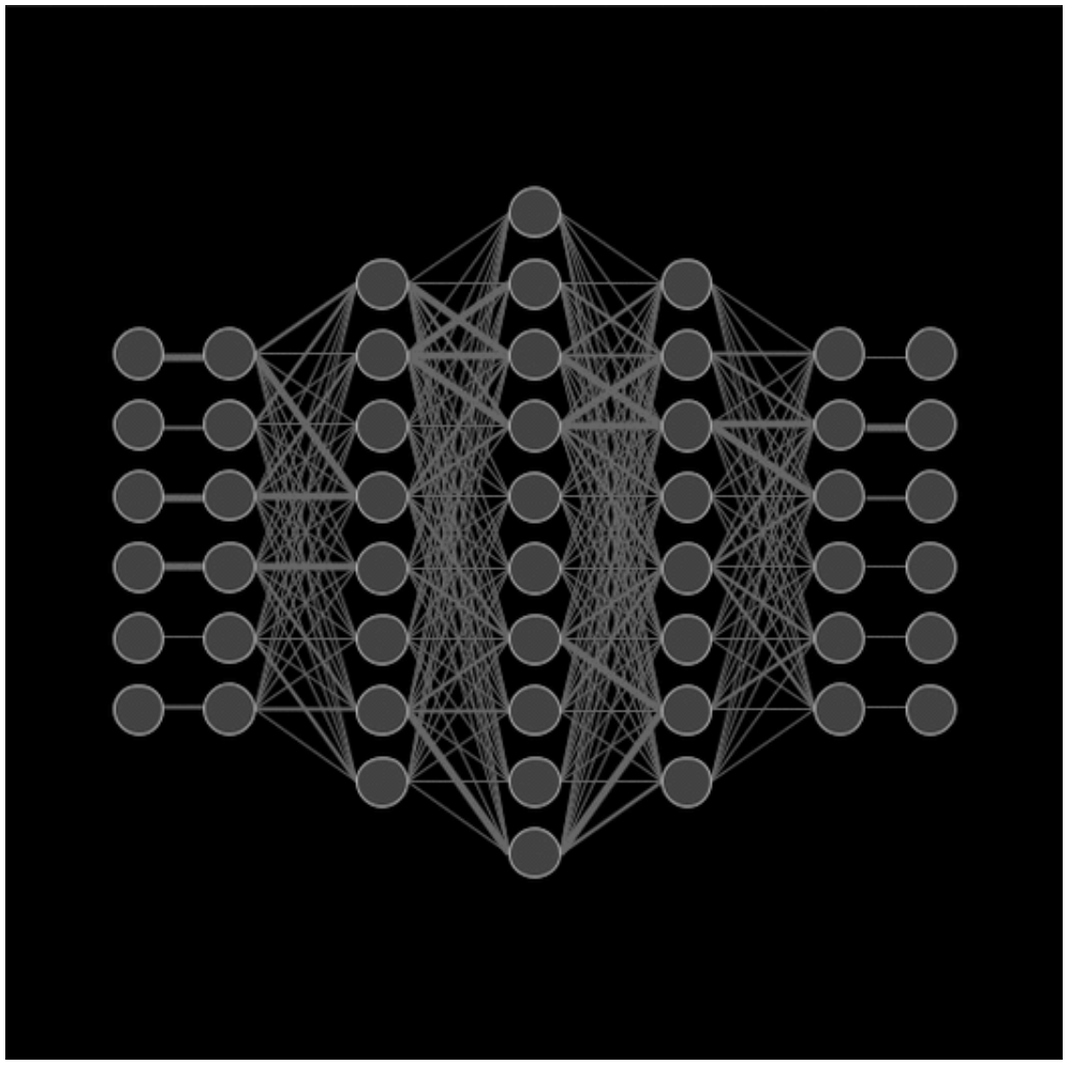
Epoch 2
Training sample
[Test sample]

Cost function training sample
[Cost function test sample]
Weights correction

Epoch 3
Training sample
[Test sample]

Cost function training sample
[Cost function test sample]
Weights correction

Epoch 4
Training sample
[Test sample]



Cost function training sample
[Cost function test sample]
Weights correction

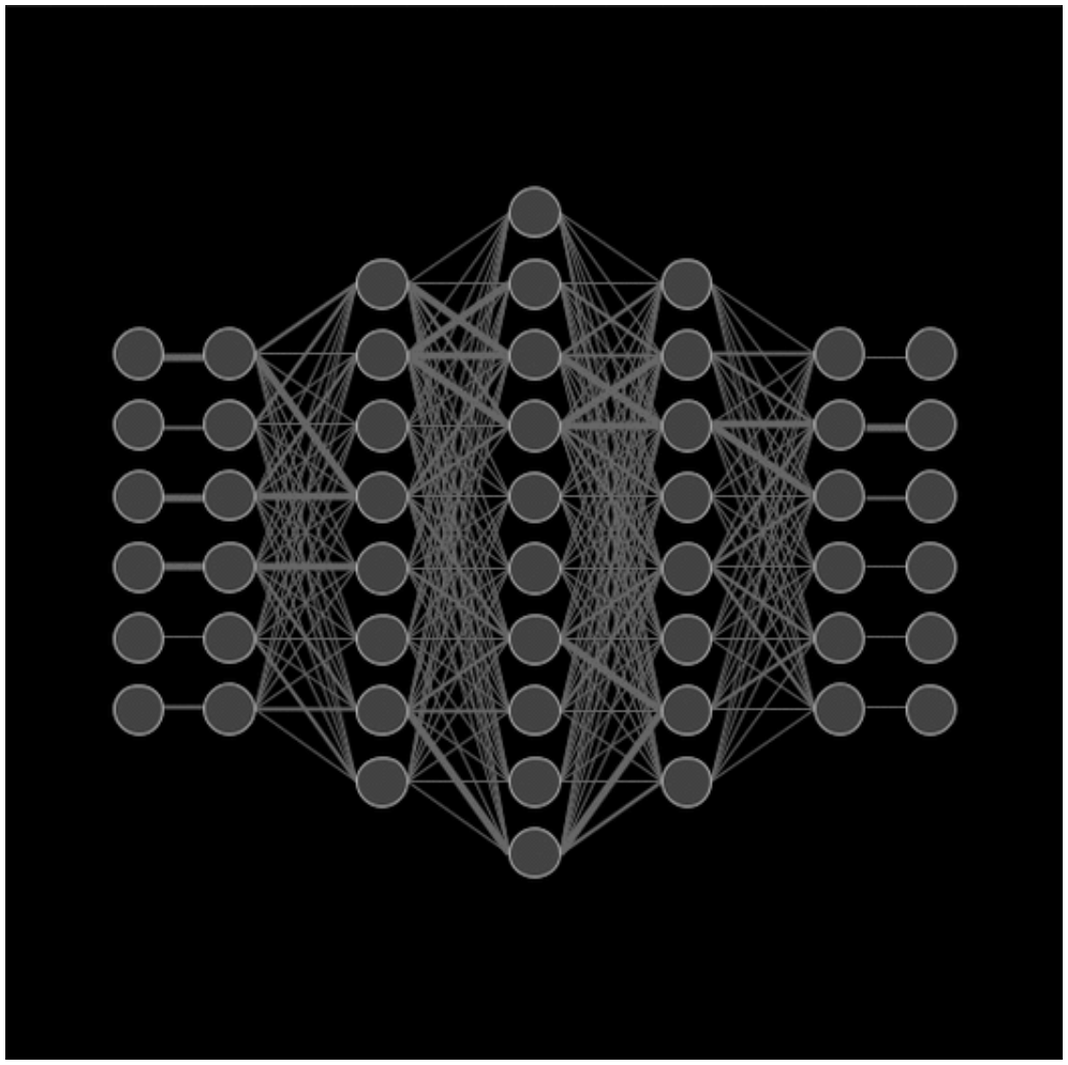
Artificial Neuronal Networks

Epoch 1
Training sample
[Test sample]

Epoch 2
Training sample
[Test sample]

Epoch 3
Training sample
[Test sample]

Epoch 4
Training sample
[Test sample]

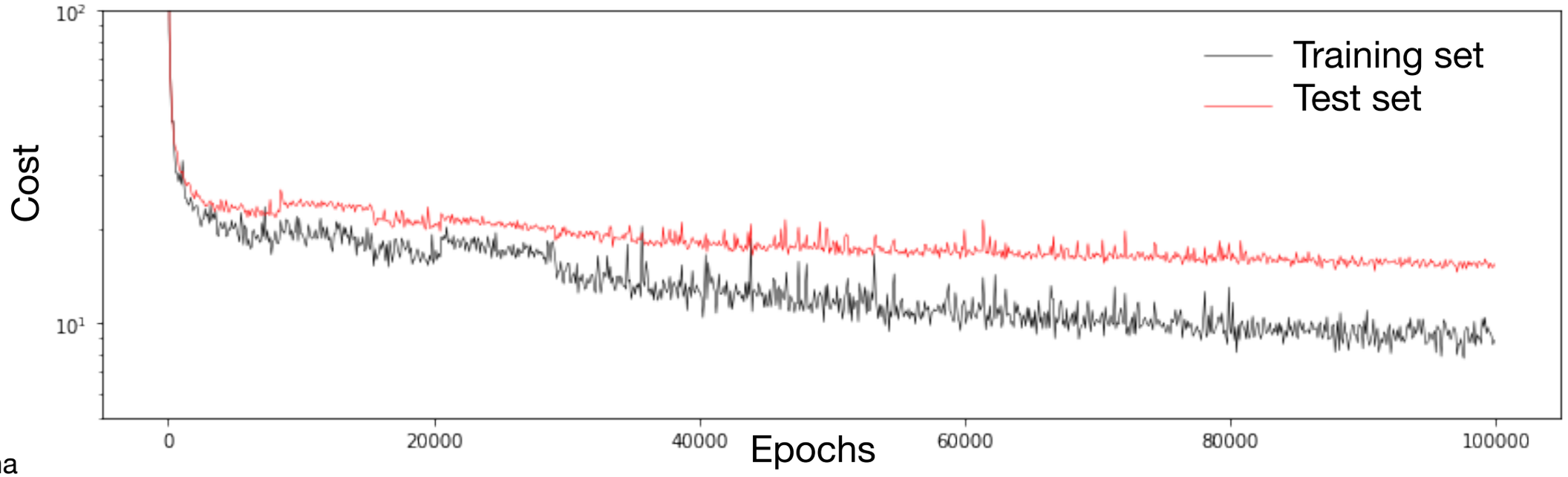


Cost function training sample
[Cost function test sample]
Weights correction

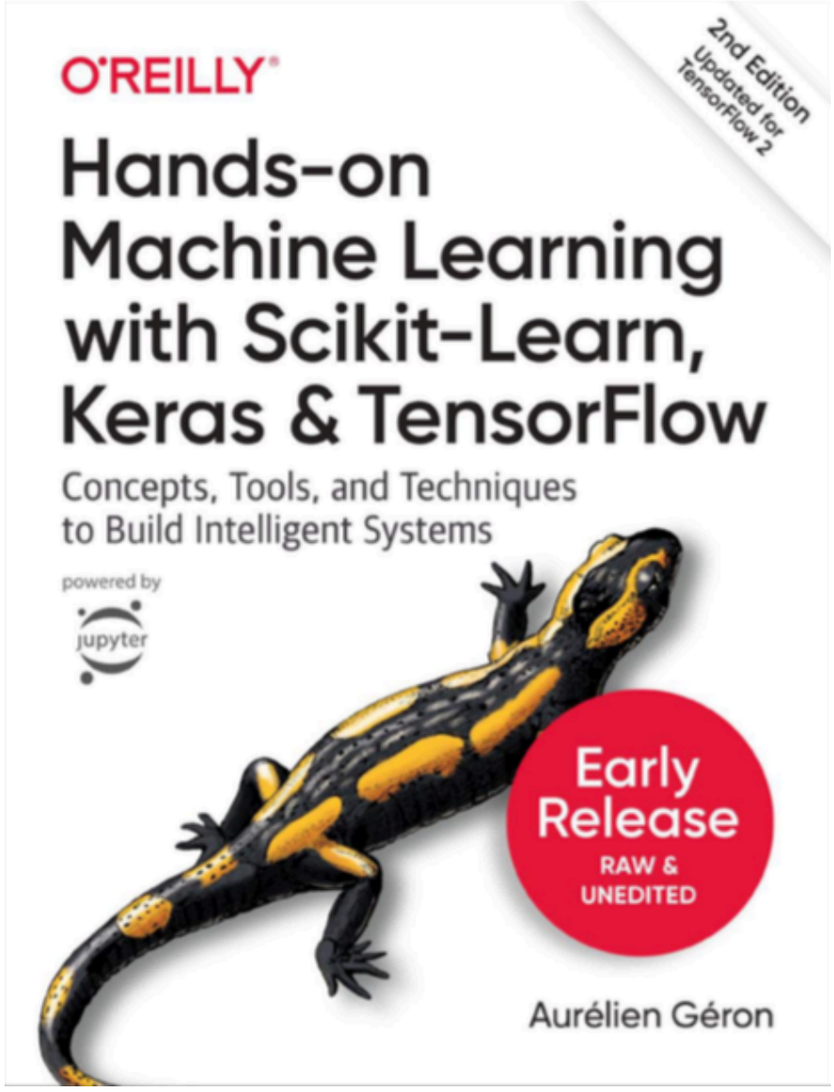
Cost function training sample
[Cost function test sample]
Weights correction

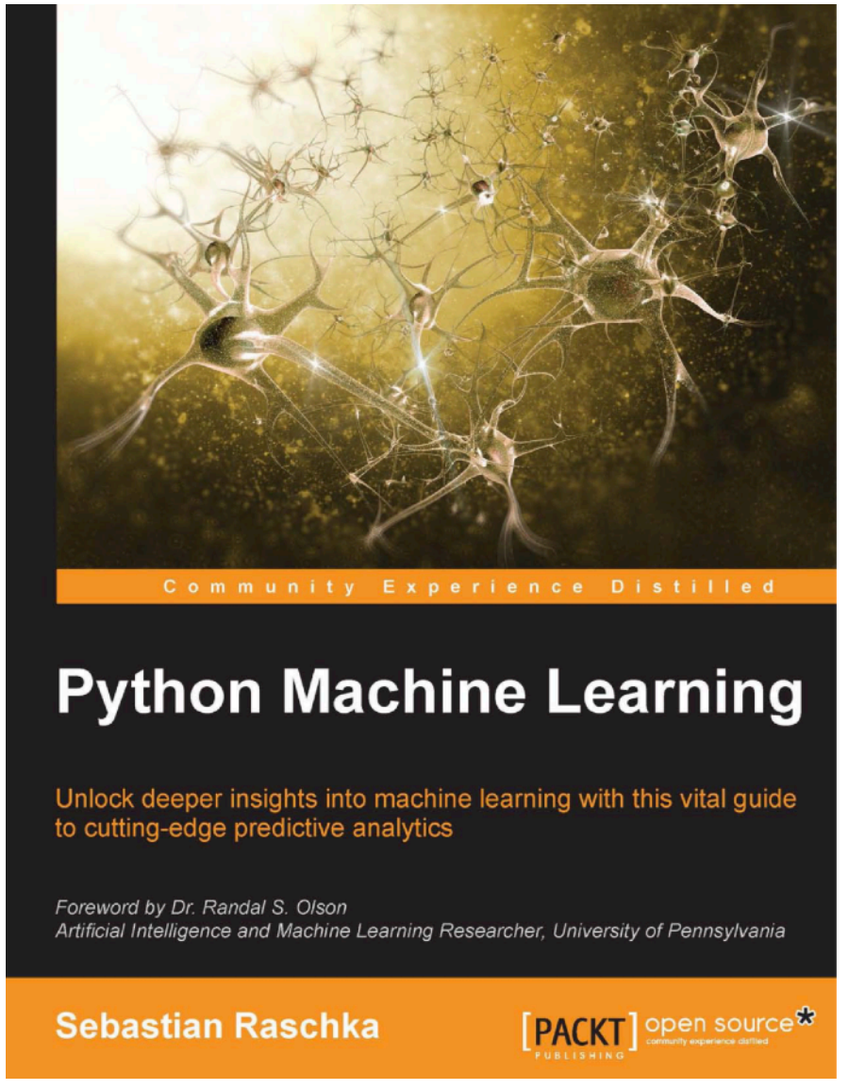
Cost function training sample
[Cost function test sample]
Weights correction

Cost function training sample
[Cost function test sample]
Weights correction



Books





Artificial Neuronal Networks in your browser

https://cs.stanford.edu/people/karpathy/convnetjs/

https://teachablemachine.withgoogle.com/

https://playground.tensorflow.org/