Artificial Neural Networks and Self-Organizing Maps-Summary

Previous Lessons Recap

- Fundamentals of Machine Learning
- Introduction to clustering, especially k-means
- Application of clustering algorithms to real-world geospatial problems
- Introduction to open-source Python tools

Learning Objectives

1. Understand the basics of Artificial Neural Networks (ANNs) and how they can be used for classification and regression tasks.
2. Learn the basics of clustering with Self-Organizing Maps (SOMs).
3. Apply open-source Python tools for geodata modeling with ANN.

Artificial Neural Networks (ANNs)

Overview:

- ANNs are inspired by the human brain's structure and functioning, mimicking the central nervous system.
- They consist of interconnected neurons that process data through layers, learning from inputs to produce outputs.

Components:

1. **Neurons and Layers:**
   - Neurons (or nodes) in an ANN are organized into layers: input layer, hidden layers, and output layer.
   - Each layer performs transformations on the data, passing it to the next layer.
2. **Perceptron:**
   - A simple neuron model used for classification tasks.
   - Transforms weighted input sums through an activation function like the logistic (sigmoid) function to approximate a threshold.
3. **Feedforward Neural Networks (FFNs):**
   - Data moves in one direction from input to output.
   - Composed of an input layer, one or more hidden layers, and an output layer.
4. **Multi-layer Perceptron (MLP):**
   - Special type of FFN where each layer is fully connected.
   - Nodes are connected to every node in the subsequent layer and to a bias node.

Learning Process:

- Adjusting connection weights to minimize prediction error.
- The network undergoes multiple iterations (epochs) to optimize weights using a loss function.

Terminology:

- **Size:** Total number of nodes.
- **Width:** Number of nodes in a layer.
- **Depth:** Number of layers.
- **Architecture:** Arrangement of layers and nodes.

Deep Learning:

- Focuses on learning features directly from data.
- Uses multiple layers (deep networks) for hierarchical feature learning.

Self-Organizing Maps (SOMs)

Overview:

- Developed by Teuvo Kohonen in 1992, SOMs are used for unsupervised learning.
- They are effective for clustering and visualizing high-dimensional data.

Structure:

- SOMs map input data to a two-dimensional grid, where similar data points are placed closer together.

Algorithm:

1. **Initialization:**
   - Randomize the weights.
2. **Input Handling:**
   - Calculate distances between input vector and weight vectors.
   - Identify the Best Matching Unit (BMU).
3. **Weight Update:**
   - Adjust weights of BMU and its neighbors.
   - Repeat for many iterations, reducing learning rate and neighborhood radius.

Visualization:

- SOM visualizations include heatmaps and U-Matrix (Unified Distance Matrix), which help in exploring data clusters and relationships between input variables.

Practical Application

Example Scenario:

- A business collects sales data and additional external factors to predict the popularity of tea types.
- Steps involved: Data collection, cleaning, exploratory data analysis (EDA), feature engineering, model training, and evaluation.

Conclusion

- ANNs and SOMs are powerful tools for both supervised and unsupervised learning tasks.
- Understanding their structures and learning processes allows for effective application in various real-world scenarios.

Key Takeaways

- ANN mimics the brain's learning process through interconnected neurons.
- SOMs offer a unique approach to visualizing and clustering high-dimensional data.
- Practical application of these models involves systematic data preparation, model training, and iterative optimization.

Further Questions

- What specific configurations and hyperparameters are optimal for different datasets?
- How can these models be enhanced to handle more complex data structures?

UNIVERSITY OF TWENTE.

ARTIFICIAL NEURAL NETWORKS and SELF-ORGANIZING MAPS

Mahdi KHODADADZADEH
May 2024

With some materials from:

- Raul Zurita-Milla (GIP-ITC)
- https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-s06/www/nn.pdf

ITC FACULTY OF GEO-INFORMATION SCIENCE AND EARTH OBSERVATION

Previous lesson

- We discussed the fundamentals of Machine Learning
- We introduced clustering particularly k-means
- We discussed a real-world geospatial problem which can be addressed by using a clustering algorithm
- We introduced some open-source Python tools

Question #1

Machine learning involves **learning** from **data**.

↳ identify $\underbrace{\text{rules}}_{\downarrow \text{ algorithm}}$ within data

✓ True

False

Question #2

Clustering is a type of **unsupervised** learning that can identify patterns in **unlabeled data**.

[✓ True]

False

Question #3

Supervised learning is a type of Machine Learning where the model is trained on **labeled data** to make predictions or classify new data points.

[✓ True]

False

Supervised Learning

- Data is partially labelled: we have many pairs **(X^i,y^i)** we may also have many **$X^{i'}$** (without known $y^{i'}$)

  X^i → vector of binary, categorical or real valued features

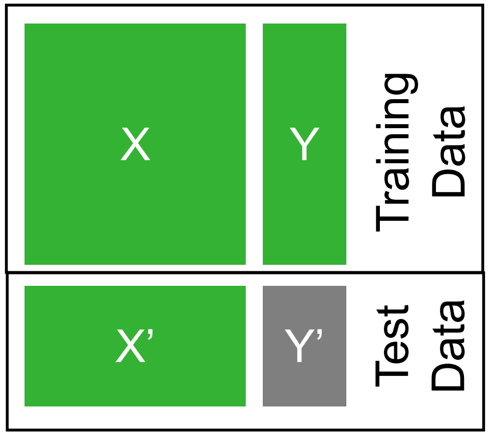
  y^i → class (label) or a real number

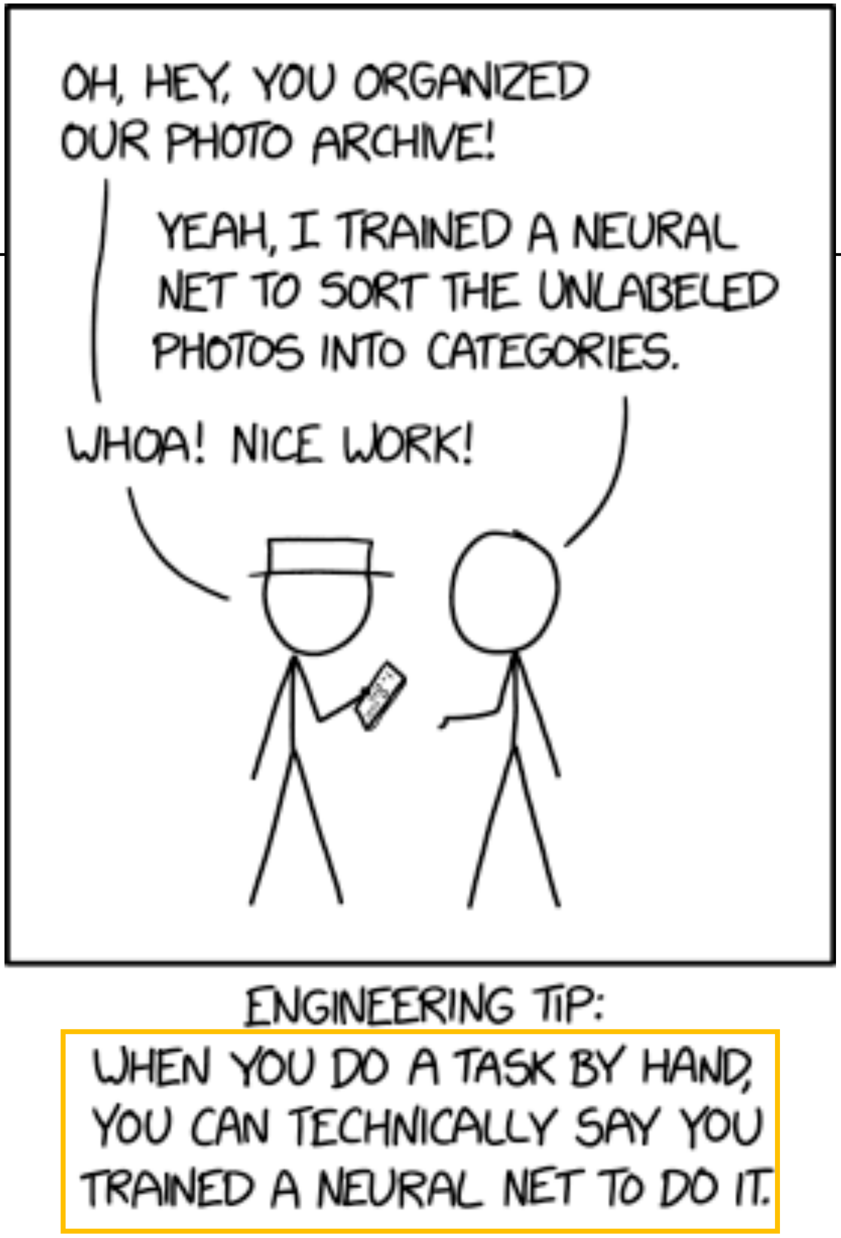
- We would like to estimate a function ***f*()** so that $y^i = f(X^i)$
- Task: given **X** and Y (all the **(X^i, y^i)** pairs) build a model ***f*()** to predict Y' based on **X'**

- When y^i is a real number

  → Regression
- When y^i is class label (or categorical)

  → Classification

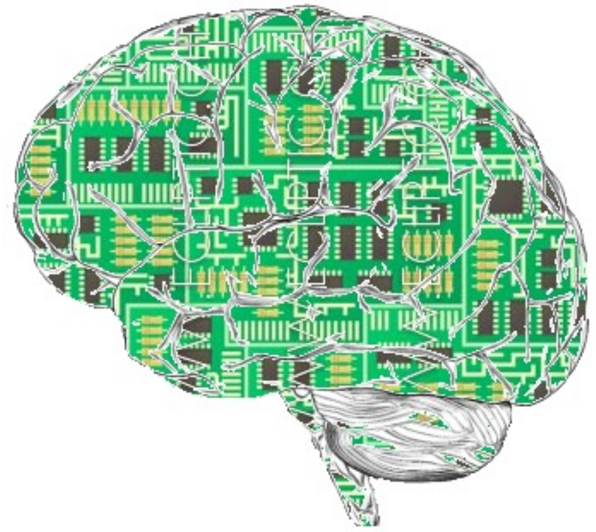



OH, HEY, YOU ORGANIZED OUR PHOTO ARCHIVE!
YEAH, I TRAINED A NEURAL NET TO SORT THE UNLABELED PHOTOS INTO CATEGORIES.
WHOA! NICE WORK!
ENGINEERING TIP:
WHEN YOU DO A TASK BY HAND, YOU CAN TECHNICALLY SAY YOU TRAINED A NEURAL NET TO DO IT.

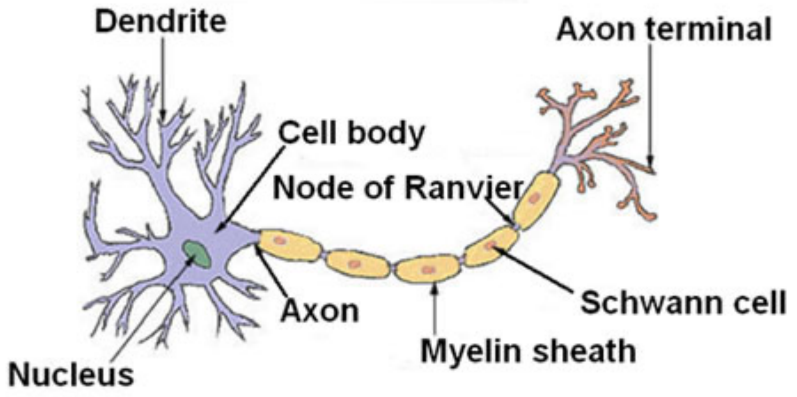
From: https://xkcd.com

Artificial neural networks (ANNs)

ANNs are machine learning models designed to imitate the human brain.



Inspired by the central nervous system and the **neurons** (and their axons, dendrites and synapses)



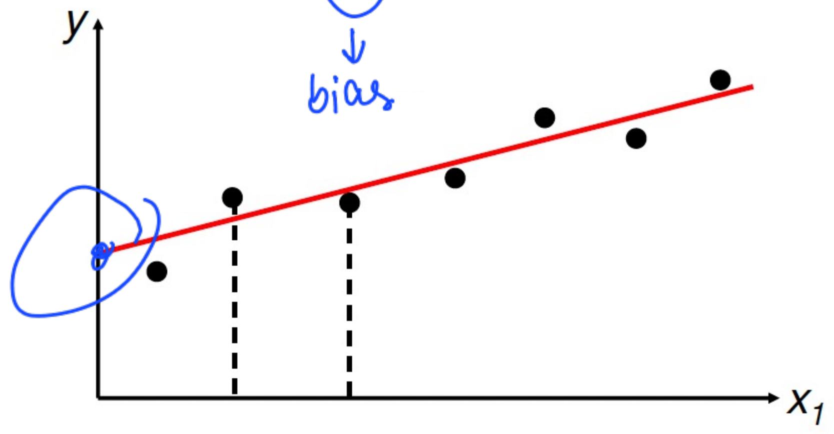
This lesson’s learning objectives

- **Explain** to peers
  - the fundamentals of **Artificial Neural Networks (ANNs)**
  - How we can us ANNs for **classification and regression** tasks
  - the basics of **clustering** with **Self-Organizing Maps (SOMs)**
- **Apply** some **open-source Python tools** to perform geodata modeling with ANN

Linear Regression

- Relationship between variables is described by a Linear function
- The change of one variable causes the other variable to change
- We want to find the parameters that predict the output Y from the data X in a linear fashion

$$Y \approx \underset{\text{bias}}{\boxed{w_0}} + w_1 x_1$$

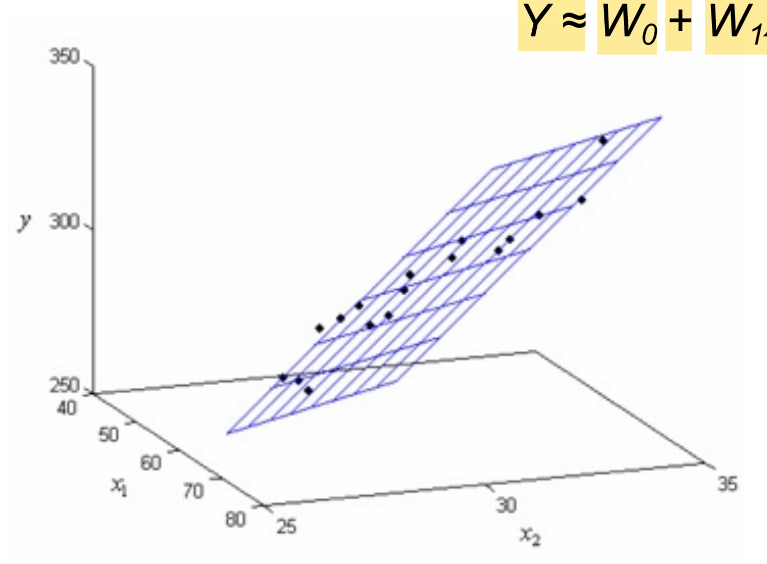


X_1	Y

Linear Regression

- Vector of attributes (feature vector) for each training data point
- We seek a vector of parameters such that we have a linear relation between prediction Y and attributes X

$$Y \approx W_0 + W_1X_1 + W_2X_2$$



X_1	X_2	Y

Linear Regression

- Vector of attributes (feature vector) for each training data point
- We seek a vector of parameters such that we have a linear relation between prediction Y and attributes X

X_1	…	X_M	Y
	…		
	…		
	…		
	…		
	…		
	…		
	…		

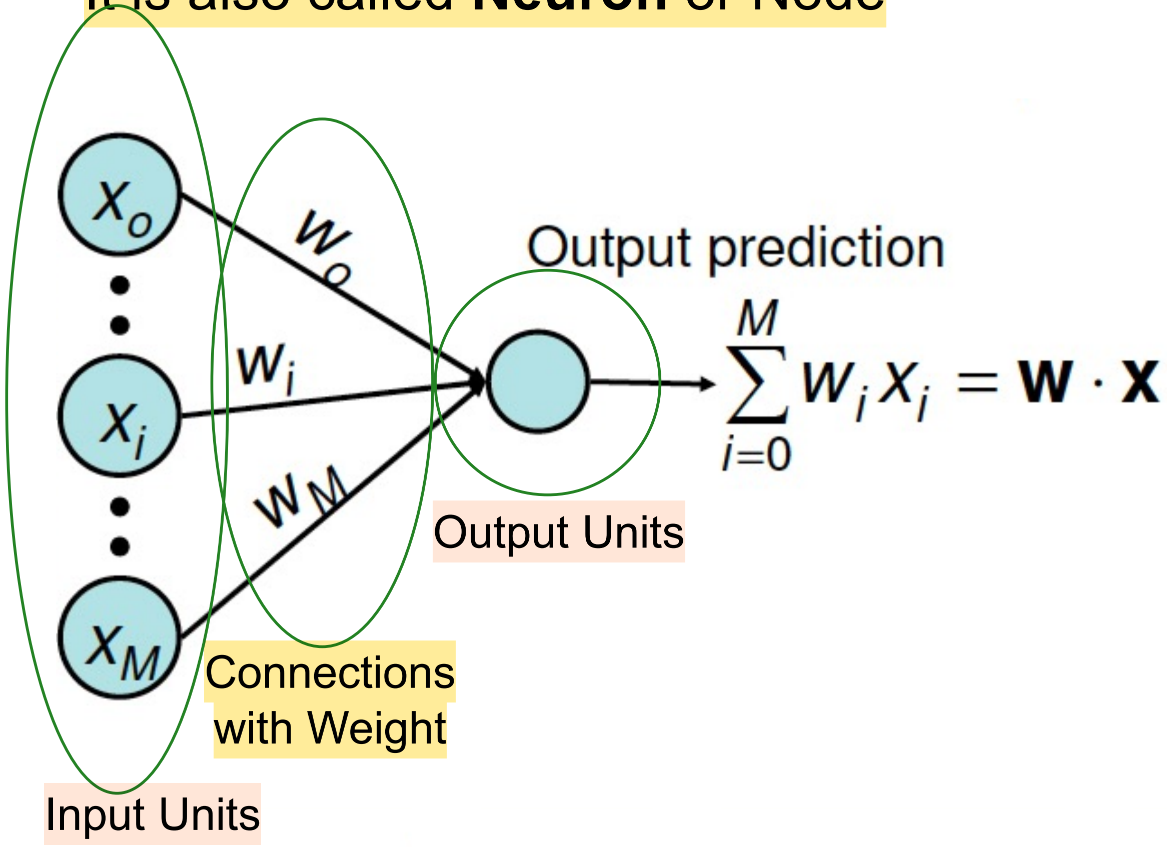
$$\mathbf{w} = [w_o, .., w_M]$$

$$y \approx w_o x_o + w_1 x_1 + \cdots + w_M x_M = \sum_{i=0}^{M} w_i x_i = \mathbf{w} \cdot \mathbf{x}$$

fake attribute for the input data: $x_0 = 1$

Perceptron

- It is also called **Neuron** or Node



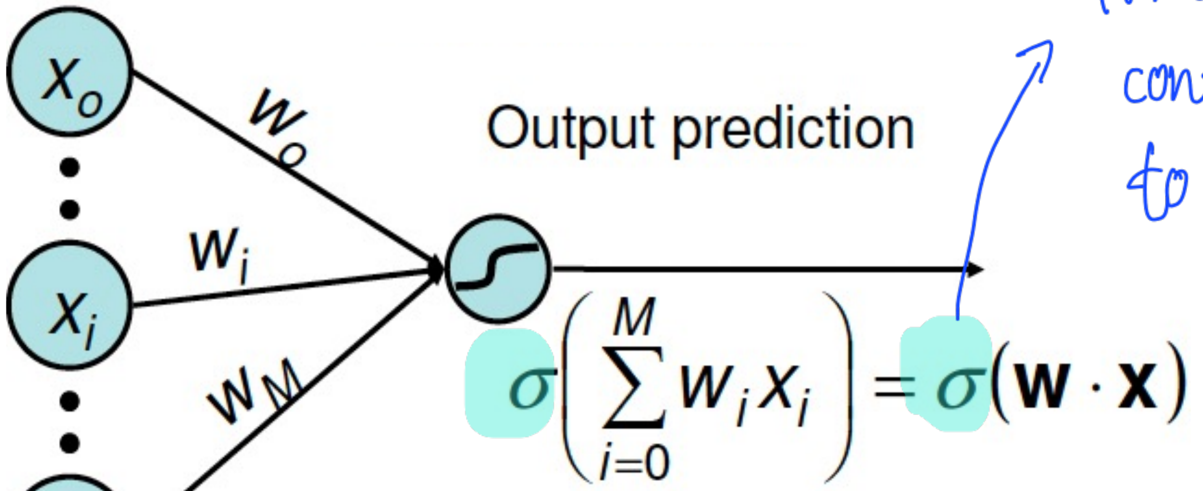
x_1	…	x_M	y
	…		
	…		
	…		
	…		
	…		
	…		
	…		

$$y \approx w_o x_o + w_1 x_1 + \cdots + w_M x_M = \sum_{i=0}^{M} w_i x_i = \mathbf{w} \cdot \mathbf{x}$$

Perceptron for Classification

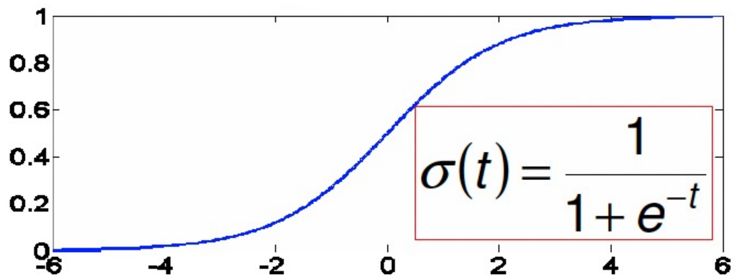
- Input data are weighted, then added up and finally transformed using an activation function

function to classify
continuous output
to classify data



$$\sigma\left(\sum_{i=0}^{M} w_i x_i\right) = \sigma(\mathbf{w} \cdot \mathbf{x})$$

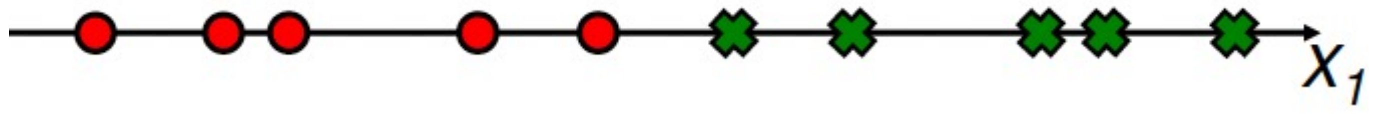
- The logistic (sigmoid) function
- It approximates a hard threshold function at x = 0



$$\sigma(t) = \frac{1}{1+e^{-t}}$$

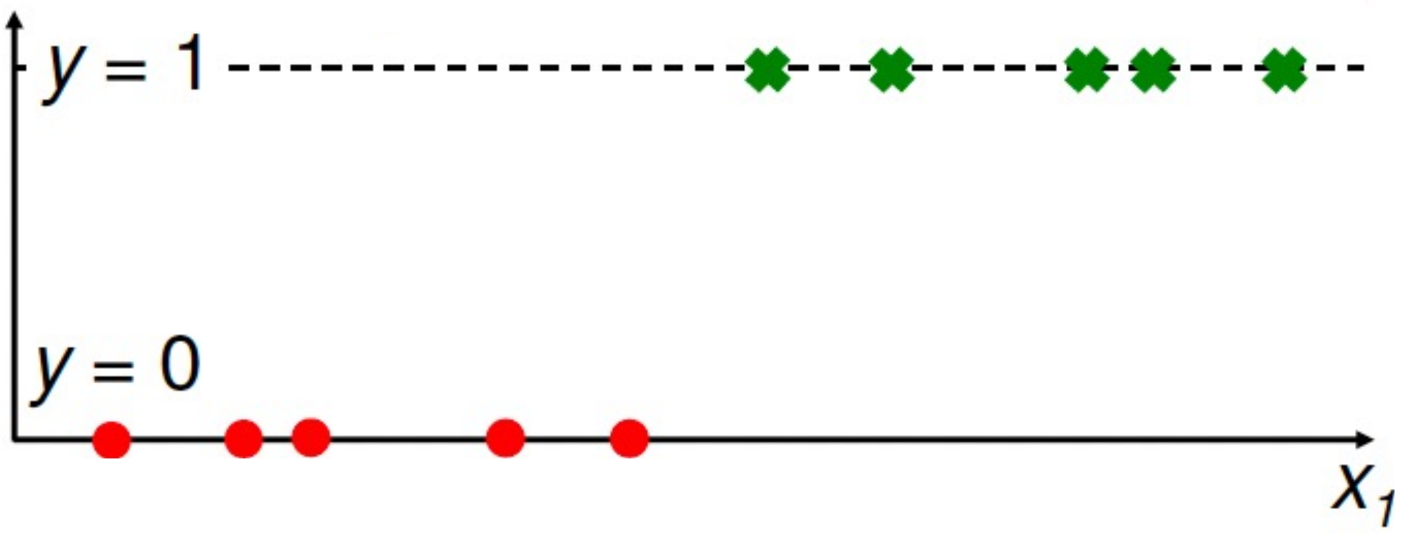
Perceptron for Classification

- Suppose that we have one attribute X_1
- Suppose that the data is in two classes (red dots and green dots)
- Given an input value X_1, we wish to predict the most likely class



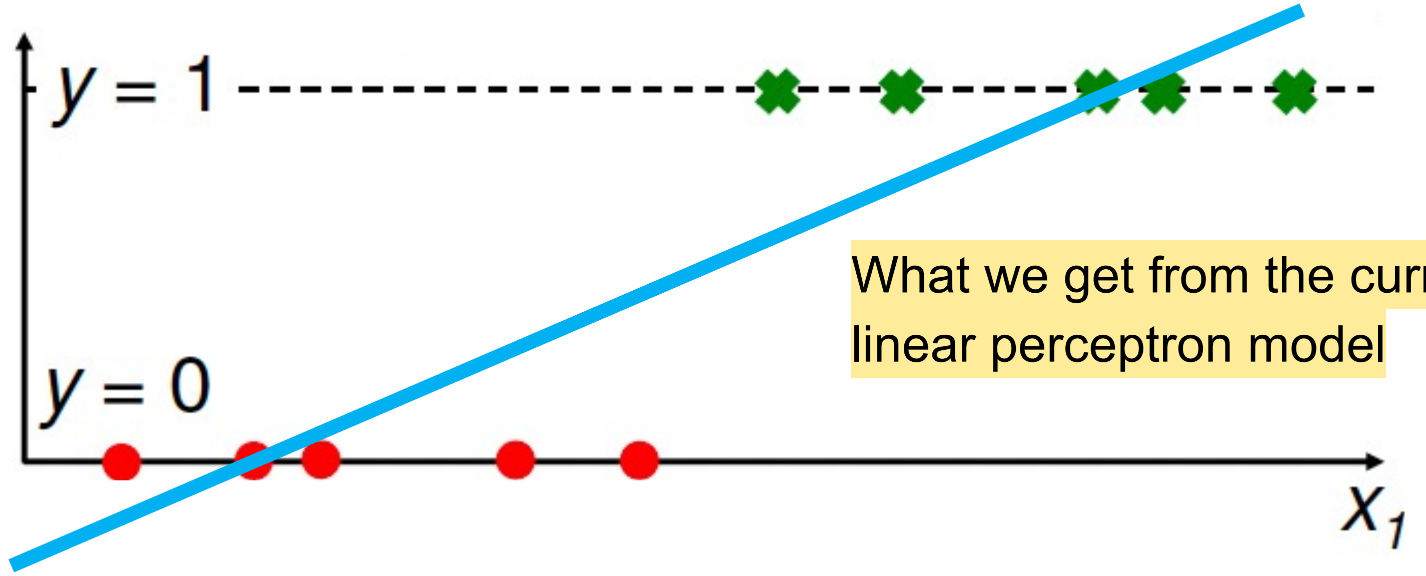
Perceptron for Classification

- We could convert it to a problem similar the regression problem



Perceptron for Classification

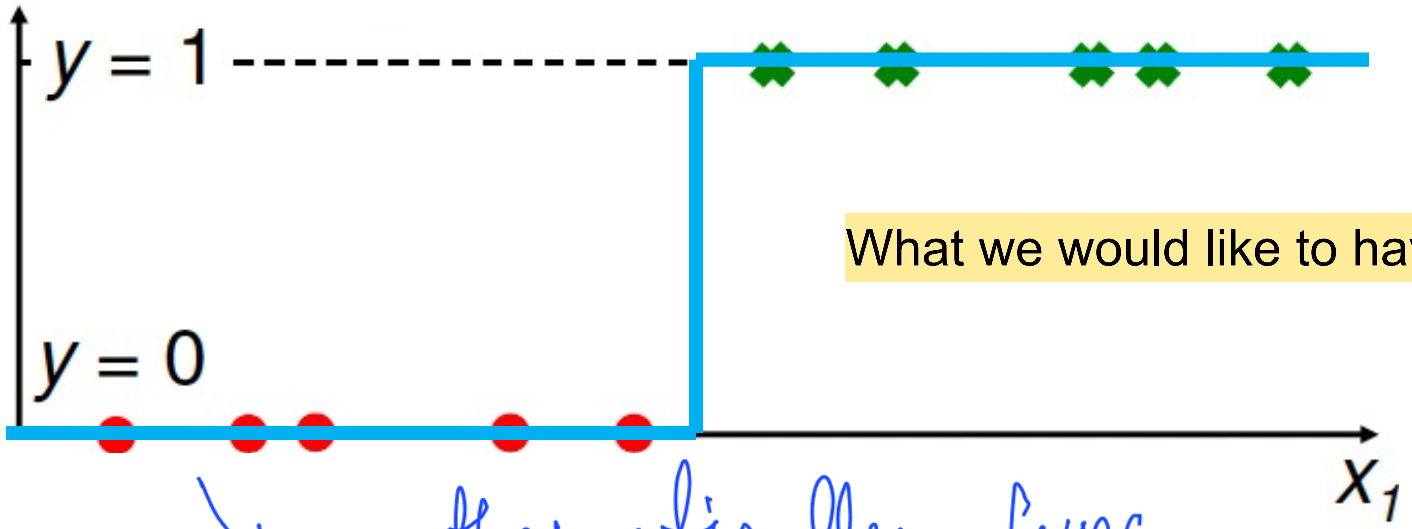
- We could convert it to a problem similar the regression problem



What we get from the current linear perceptron model

Perceptron for Classification

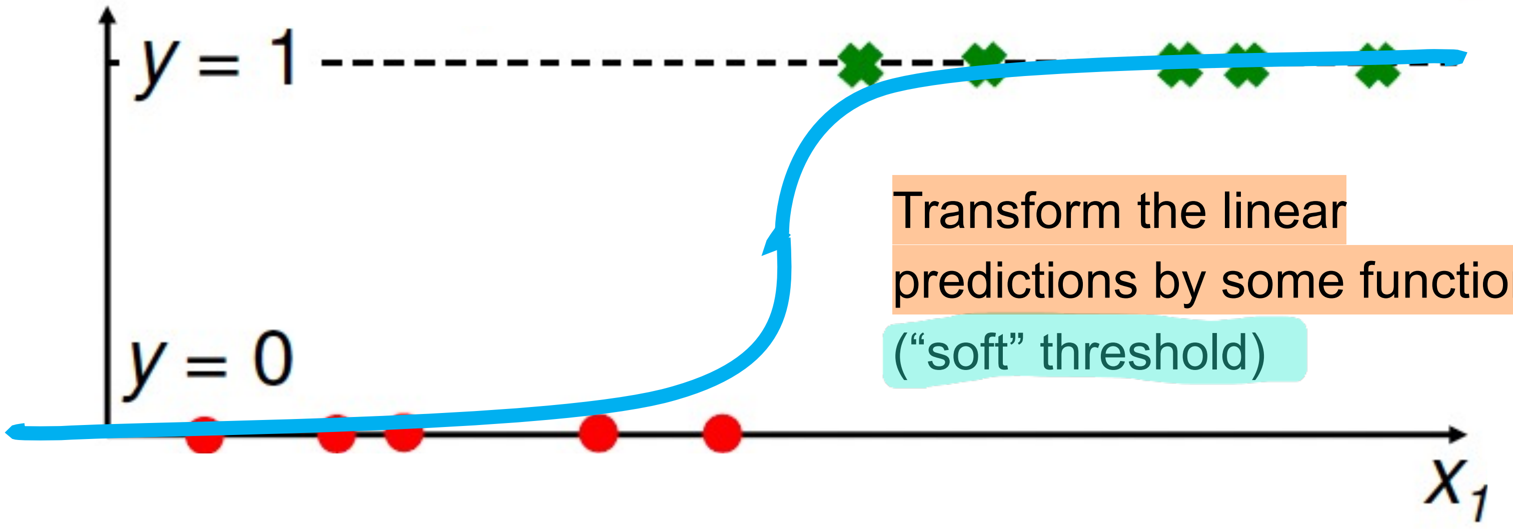
- We could convert it to a problem similar the regression problem



$\searrow$ mathematically func. can't go to this hard threshold

Perceptron for Classification

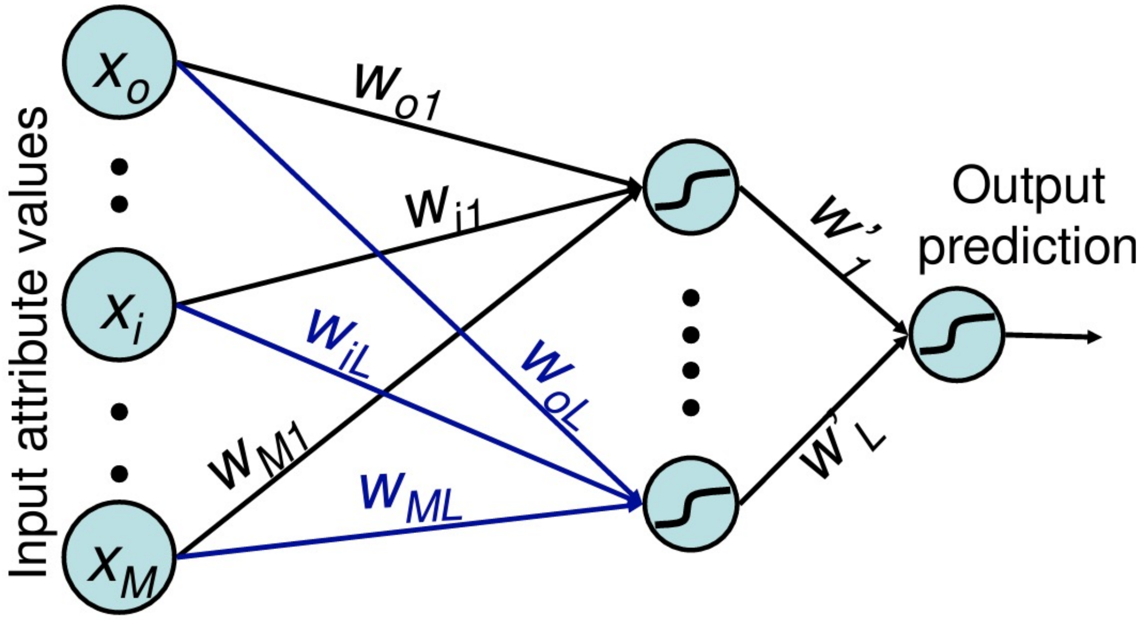
- We could convert it to a problem similar the regression problem



Transform the linear predictions by some function
("soft" threshold)

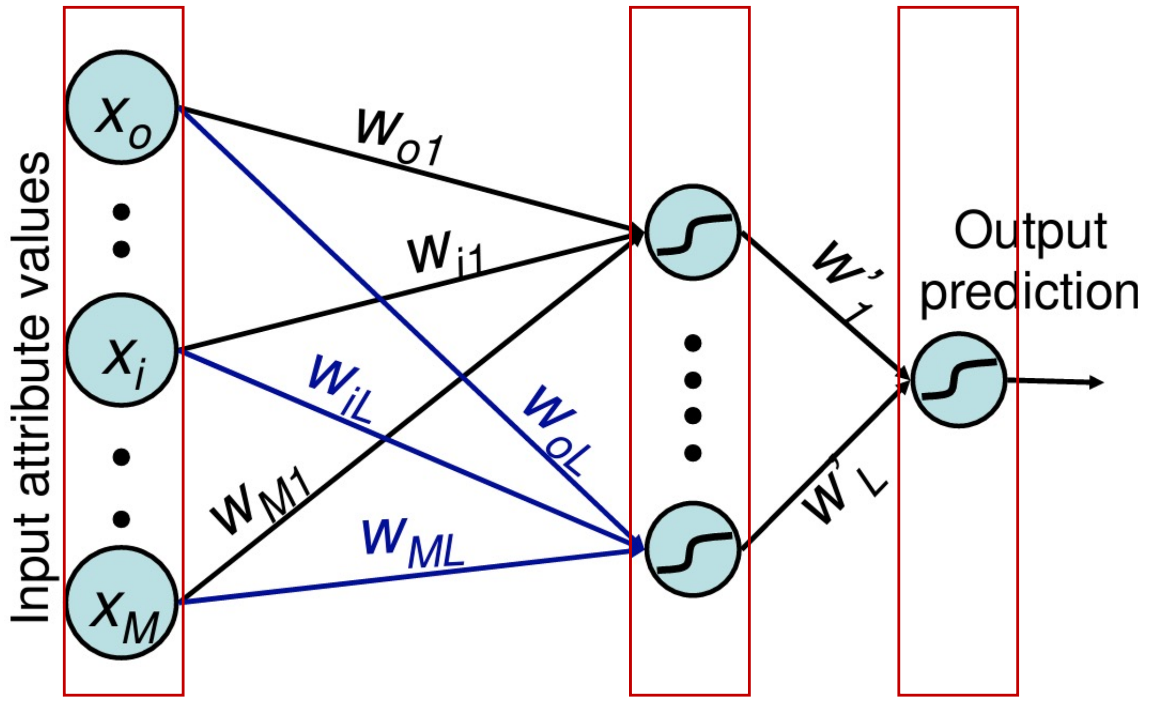
A Neural Network

- A neural network is a combination of perceptrons (or neurons).



A Neural Network

- Neurons are organized in layers
- Each layer is a set of functions



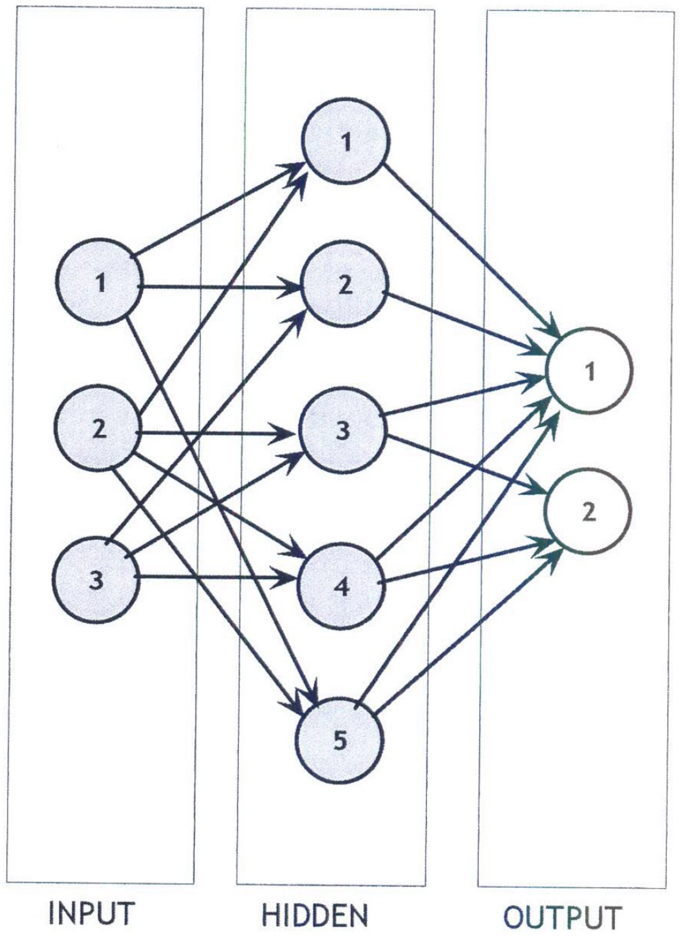
An input layer

One or more hidden layers

An output layer

Feedforward Neural Networks (FFNs)

- FNNs consist of an **input layer**, **one or more hidden layers**, and **an output layer**.
- **Sequential layers**: the connections between the nodes do not form a cycle
- **Data moves in one direction**: from the input nodes, through the hidden nodes (if any), and finally to the output nodes.
- Each hidden layer **outputs** a set of vectors that serve as **input to the next layer**



type of FFN

32 connections = 32 weights that we want to find

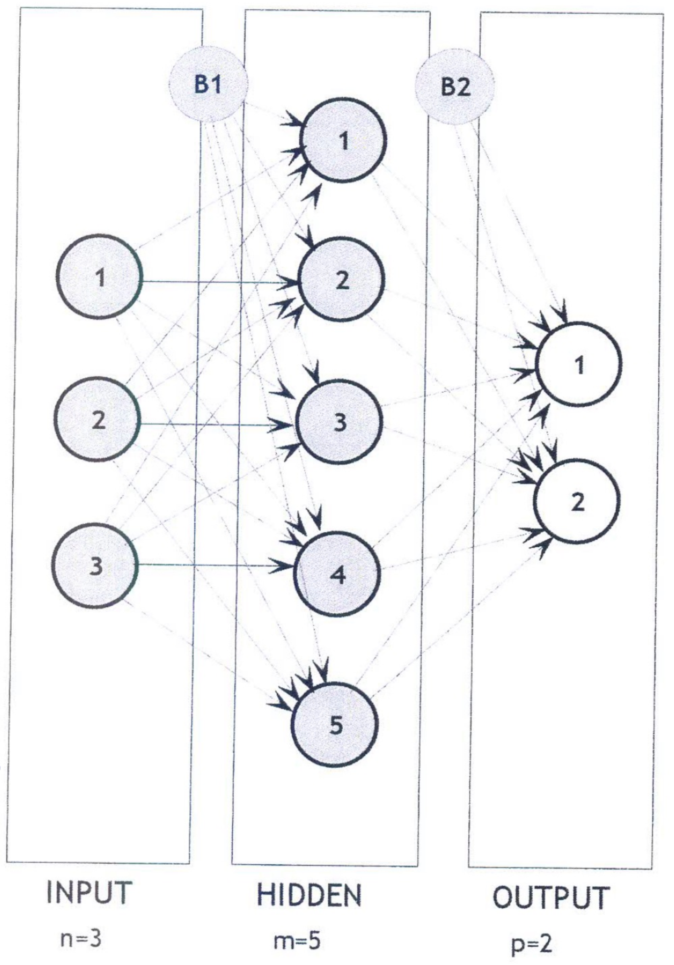
Multi-layer perceptron (MLP)

FFN + fully-connected

- A special case of a feedforward neural network where every layer is a **fully connected layer**
- In **MLP** each node is connected to every node in the next layer
- Nodes in the hidden and output layers are connected to a bias node (feeding a constant value ~ constant in regression models)

why we need bias node?

- bias node → used for shifting activation function, make equation be computable



more neurons $\rightarrow$ more weights that we need to identify $\rightarrow$ we need more data.

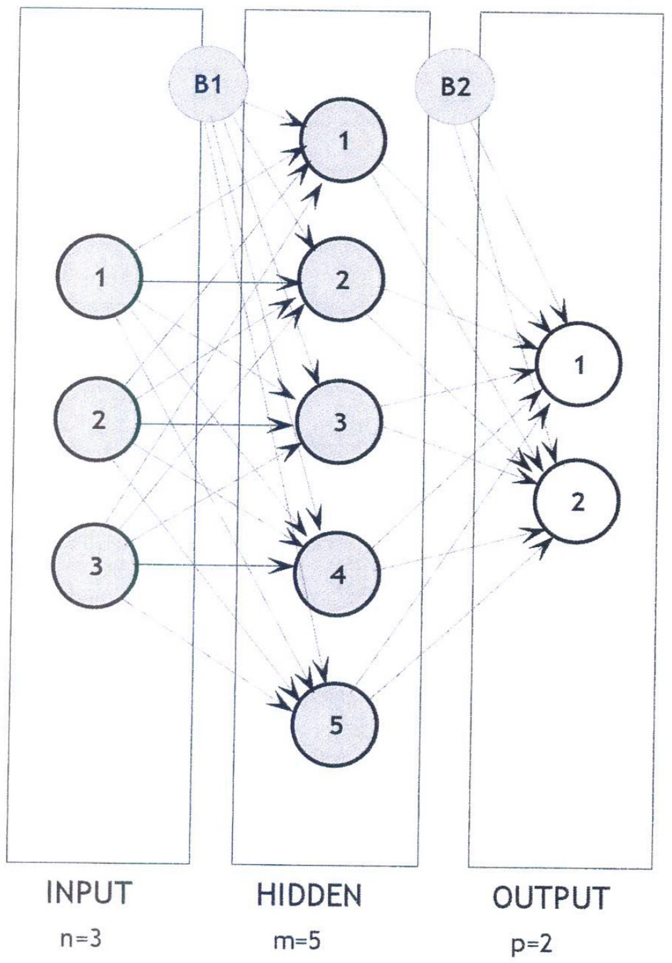
Neural Network Learning problem

- Adjusting the **connection weights** so that the network generates the correct prediction on the **training data**.
  - Start with random weights for all the connections in the neural network.
  - Input data is fed into the input layer.
  - The data is passed through the network layer by layer.
  - Each neuron computes a weighted sum of its inputs and applies an activation function to produce its output.
  - The outputs of one layer become the inputs to the next layer.
  - This process continues until the final output layer produces the network's prediction.
  - Compare the network's prediction with the actual target values (labels) using a loss function.
  - Adjust the weights in the direction that reduces the loss.
  - Repeat for multiple iterations (epochs) until the loss converges to a minimum value.

Multi-layer perceptron (MLP)

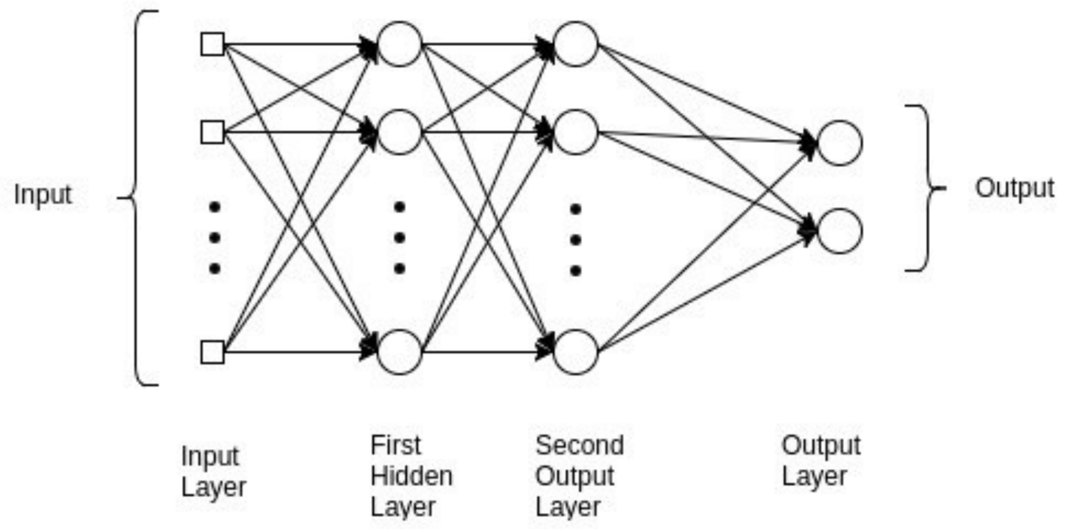
Total number of parameters to be fitted:

20+12 = 32



ANNs: terminology

- **Size**: The number of nodes in the model.
- **Width**: The number of nodes in a specific layer.
- **Depth**: The number of layers in a neural network.
  - The input layer is often not counted
- **Architecture**: The specific arrangement of the layers and nodes in the network.

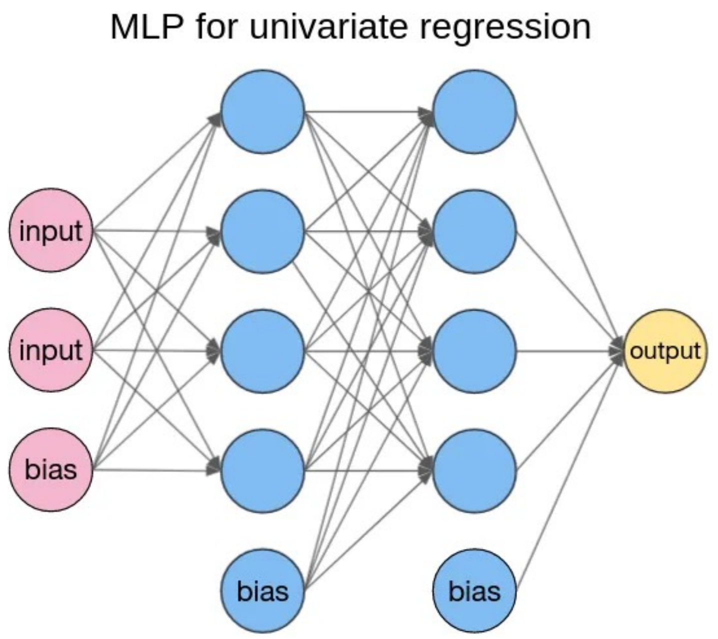


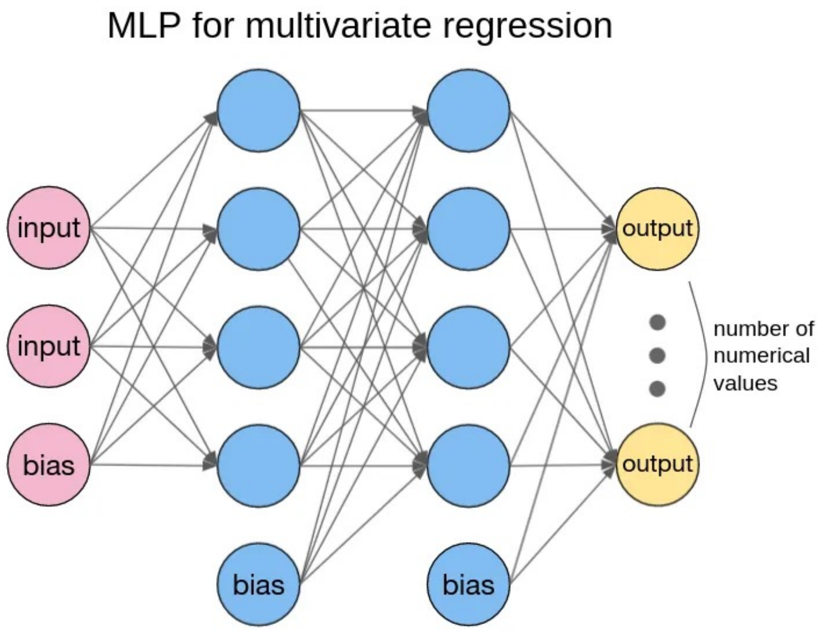
How Many Layers and Nodes to Use?

- The input layer nodes take the values of the input features
- The number of neurons comprising **the input layer is equal to the number of features (columns)** in your data
  - Some configurations add one additional node for a bias term

- The **output layer has one node for each output**
  - A single node in the case of regression
  - K nodes in the case of K-class classification

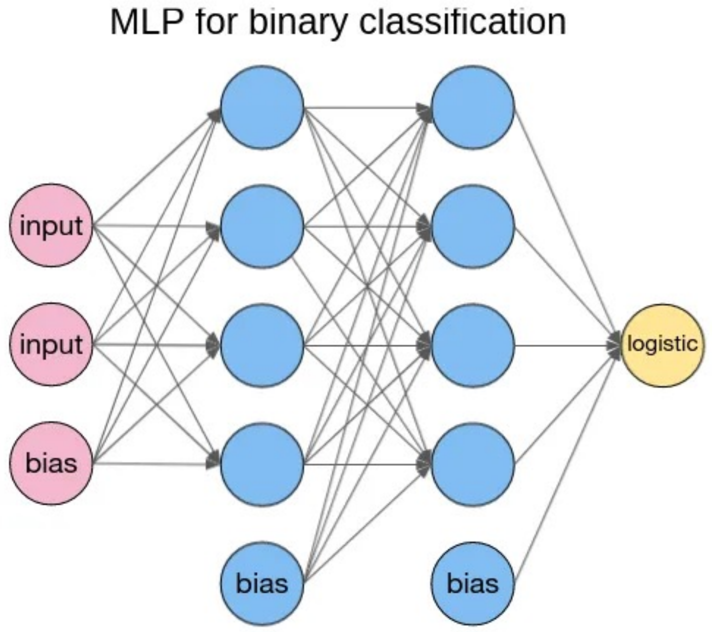
MLP for regression



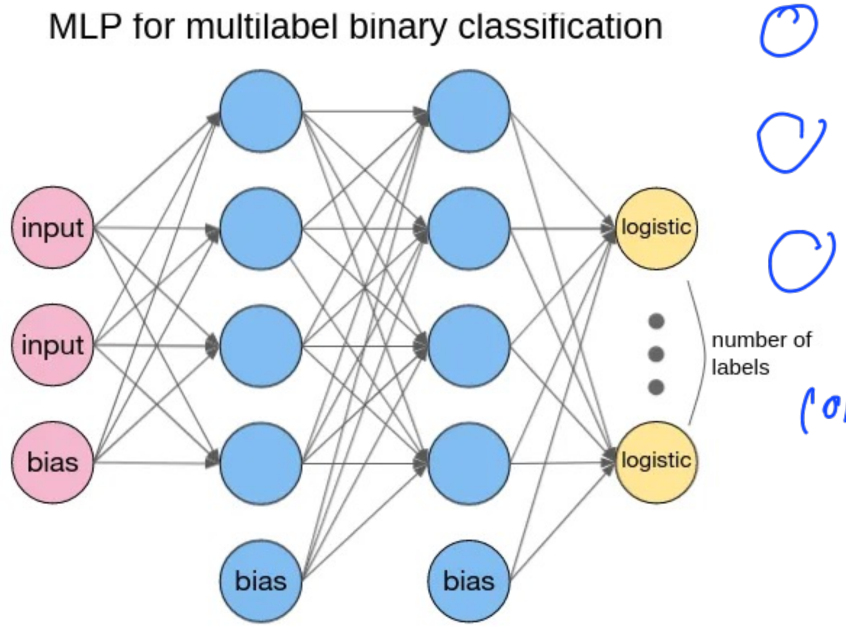


From: https://medium.com/codex/mlps-applications-with-tweaks-in-its-structure-c9aa3f05578

MLP for classification



/ many classes



3 classes.

◯ class 1 vs. not class 1

◯ class 2 vs. not class 2

◯ class 3 vs. not class 3

(one vs. all)

From: https://medium.com/codex/mlps-applications-with-tweaks-in-its-structure-c9aa3f05578

How Many Layers and Nodes to Use?

- For the hidden layers, use systematic experimentation to discover what works best for your specific dataset!
- In general, you cannot analytically calculate the number of layers and nodes

  → **hyperparameters optimization**

- Intuition and experience
- Read the relevant literature

How Many Layers and Nodes to Use?

- There are some rules of thumb that may help you, e.g.:
  - The average of input and output neurons

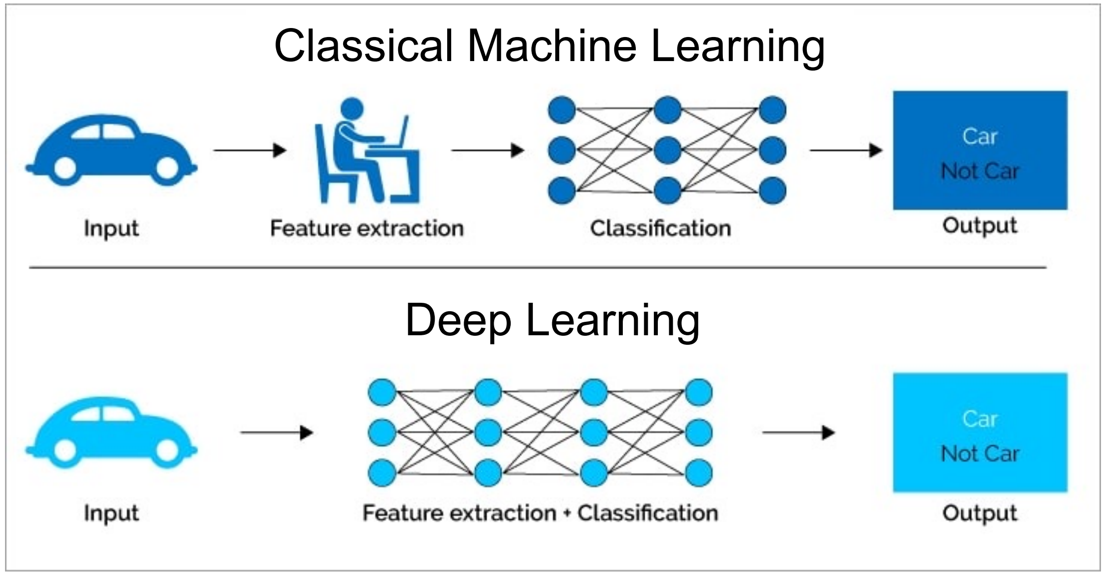
    4 inputs, 2 outputs $\longrightarrow \frac{4+2}{2} = 3 \rightarrow$ 3 neurons for hidden layer
  - Using the following equation:

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

Deep Learning

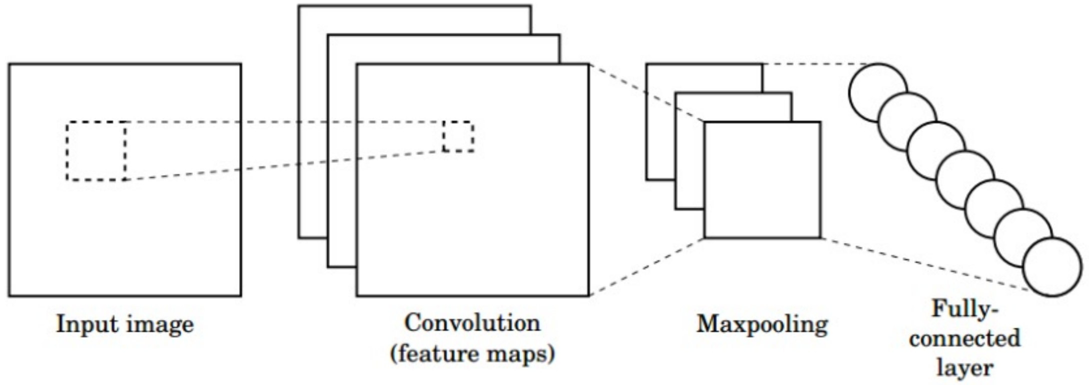
- Hand engineering features is time consuming, brittle, and not scalable in practice.
- Deep Learning aims to learn the underlying features relevant for the task directly from the data through a series of layers in neural networks.



https://levity.ai/blog/difference-machine-learning-deep-learning type of deep learning (it handles feature extraction)

Convolutional Neural Networks (CNNs)

- CNNs are a specialized kind of neural network for processing data that has a known grid-like topology (e.g., images).
- Connecting image patches in input layer to a single neuron in subsequent layer.



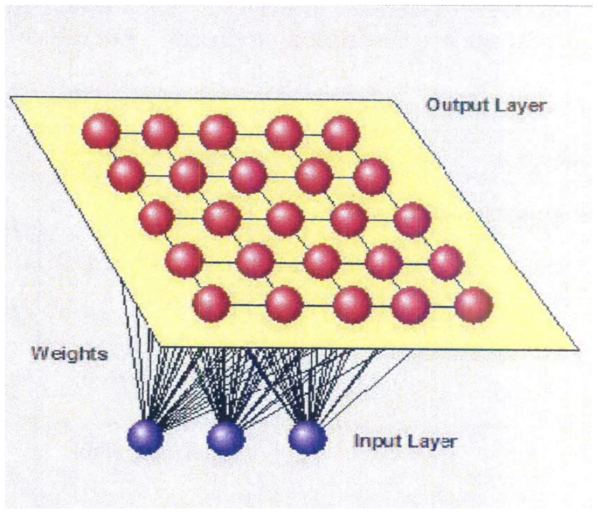
1. **Convolution**: Apply filters with learned weights to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

Train model with image data.
Learn weights of filters in convolutional layers.

Self-Organizing Maps (Kohonen maps)

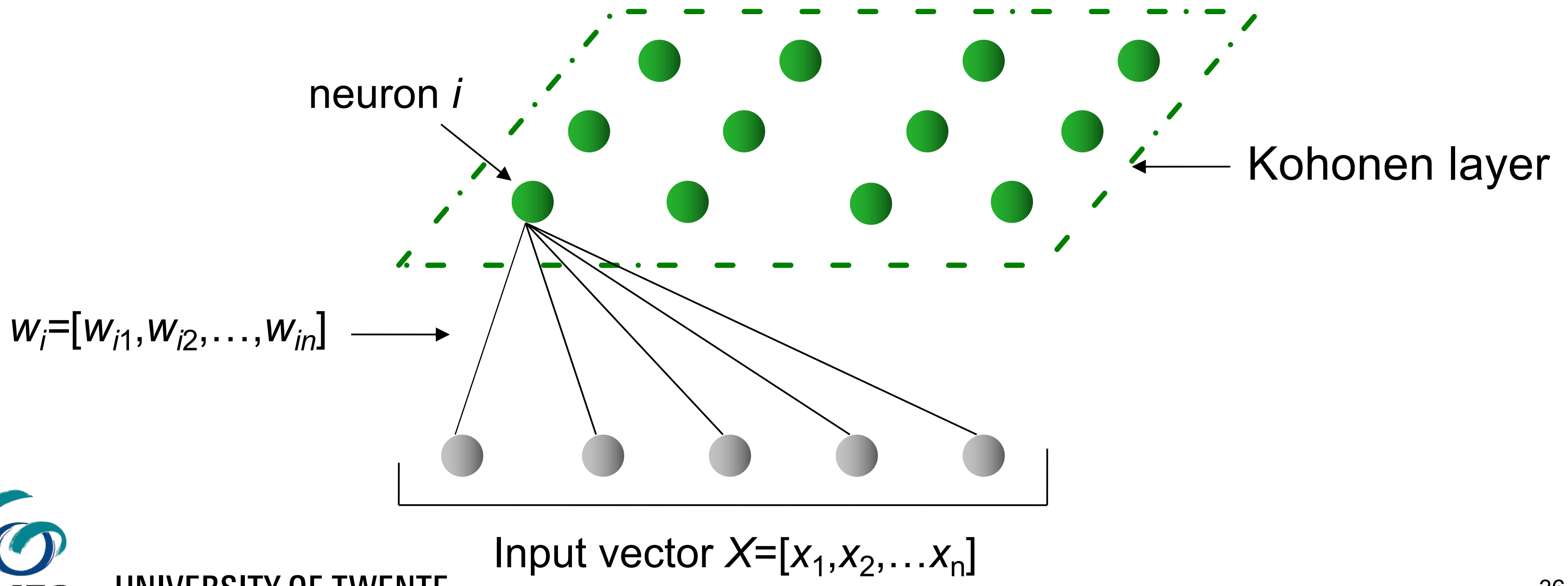
Self-organizing maps (Kohonen maps)

- Developed and formalized in 1992 by **Teuvo Kohonen**
- A type of artificial neural network used for **unsupervised learning**
- **Clustering** and **visualizing** high dimensional data
- **Detecting patterns** in multidimensional data and representing them in much **lower dimensional spaces (typically 2D)**
- **Neural networks** try to figure out patterns in the data on their own



Self-organizing maps

- The input is connected with each neuron of a grid (map).
- SOMs work by mapping input data to a two-dimensional grid.
- Similar data points are mapped closer together.



neuron i
Kohonen layer
$w_i=[w_{i1},w_{i2},\ldots,w_{in}]$
Input vector $X=[x_1,x_2,\ldots x_n]$

SOM Algorithm

- 1) The weights are initialized to random values
- 2) a m-dimensional input vector Xs enters the network;
- 3) The distances di(Wi, Xs) between all the weight vectors on the SOM and Xs are calculated by using (for instance):

$$d_i(W_i, X_s) = \sum_{j=1}^{m} \left(w_j - x_j\right)^2$$

  - Wi denotes the ith weight vector;
  - wj and xj represent the jth elements of Wi and Xi respectively

SOM Algorithm

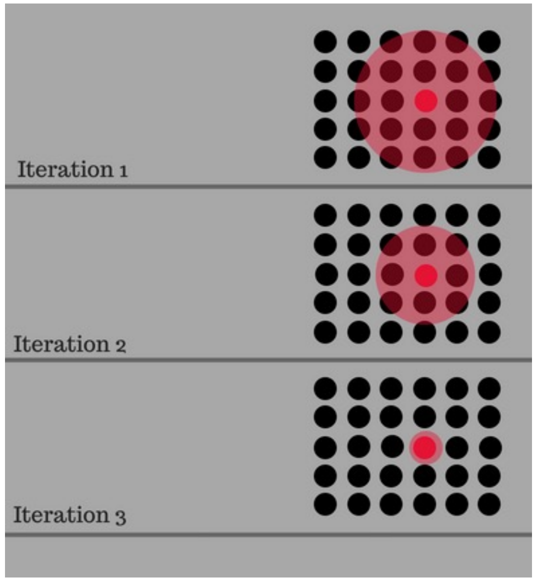
- 4) Find the best matching neuron or "winning" neuron (BMU-Best Matching Unit) whose weight vector W_k is closest to the current input vector X_i ;
- 5) Modify the weights of the winning neuron and all the neurons in the neighbourhood N_k by applying:
    - $W_{jnew} = W_{jold} + \alpha(X_i - W_{jold})$
  - Where α represents the learning rate;
- 6) Next input vector $X_{(i+1)}$, the process is repeated.

➢ Repeat the process for many iterations, gradually reducing the learning rate and the neighborhood radius.

➢ Over time, the weight vectors of the SOM nodes become organized in such a way that they capture the topological structure of the input data.

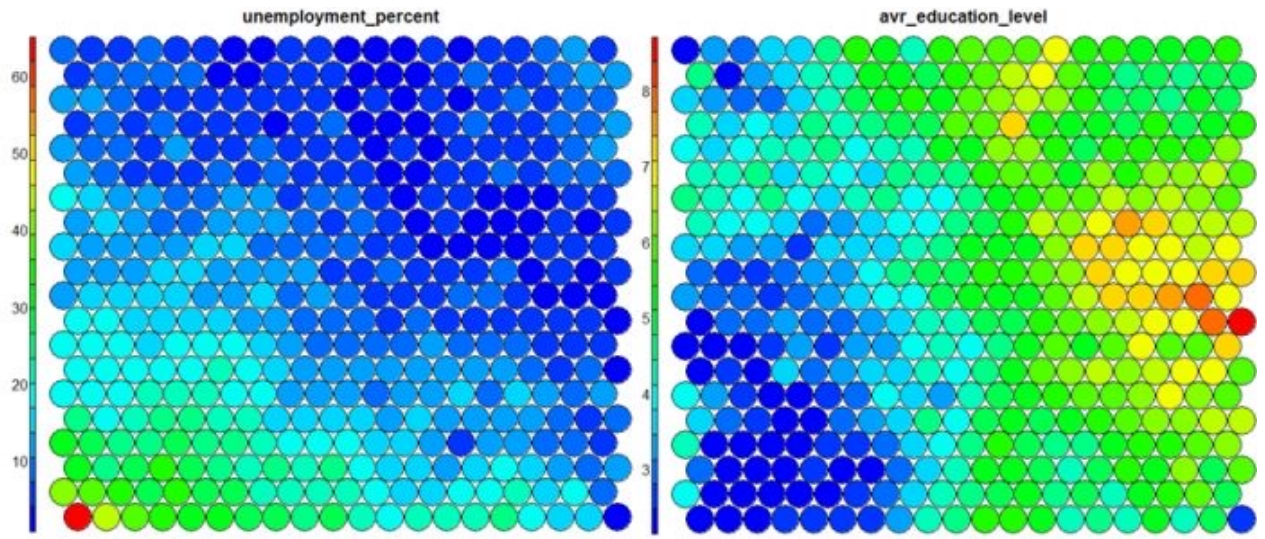
➢ Input are assigned to neurons that are similar to them.

➢ Basically, each neuron becomes the center of a cluster.



SOM Visualizations

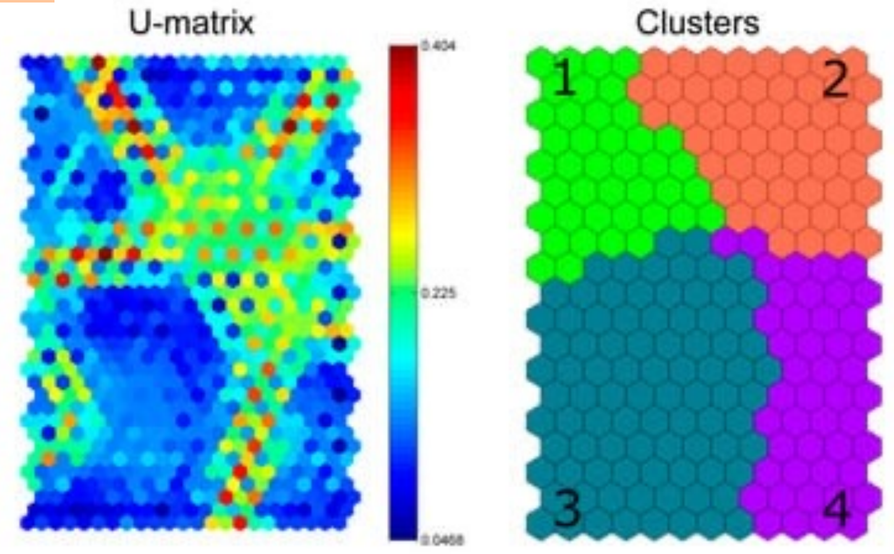
- Typical SOM visualizations are of "heatmaps".
- Visualization of different heatmaps allows one to explore the relationship between the input variables.



https://www.shanelynn.ie/self-organising-maps-for-customer-segmentation-using-r/

SOM Visualizations

- U-Matrix visualization: the distance between each node and its neighbors.
- Useful visualization to find the "natural number" of clusters in the data without any a priori information.
- High value areas work as cluster separators



https://www.mdpi.com/2071-1050/11/5/1314/htm

Team based learning

Ghelgheli decided to improve his business further. This time, he wanted to predict which types of tea would be most popular on any given day. He spent months collecting data from his customers on their orders. He also collected other additional data from different sources.

Next, Ghelgheli used a magic tool that could learn from these past sales data and other factors to predict the future and make more informed decisions. Using this tool, he could understand the relationship between different factors and the sales of certain types of tea and he could predict which types of tea would be most popular on any given day. These predictions helped him to have enough supply for the most popular teas each day and to guarantee his customers' satisfaction. As a result, Ghelgheli's business grew and grew, and he became even more famous for having a wide selection of delicious teas.

- **Which data and methods do you think Ghelgheli utilized for his analysis?**
- **Can you list the potential steps that Ghelgheli took for such an analysis?**
- **Can you provide some real-life examples similar to Ghelgheli's experience?**

Conclusion

- Data: Sales (e.g., daily sales records for each type of tea), customer (e.g., demographics, preferences, feedback and reviews), external factors (e.g., weather, events, social media, seasonality).

- Methods: Data cleaning and preparation, EDA (descriptive statistics and visualizations), feature engineering, classification algorithms (e.g., MLP) for predicting the most popular types of tea.

- Steps: Data collection and integration, data cleaning, EDA, feature engineering, model selection, training and testing, model evaluation.

Conclusion

- Please reply to the questions and write your answers on the yellow post-it

➢ What is the most important idea/insight you will remember from today's lesson?

➢ What questions do you still have?

I can **explain** to my friends the fundamentals of **ANN and SOMs**

not yet ☹ very well ☺