MACHINE LEARNING

DIANE LINGRAND

2022-2023

Torres Ramos, Juan Luis

SI4 - Computer Science

INDEX

EXAM

Information

- **Next week**: class from 8am to 9am for all your questions (it disappeared from the edt but should come back).
- from 9:15 to 10:45: DS (1/3 time until 11:15)
- For the DS: no documents allowed. I do not ask to know the formulas of the course by heart. I may give some of them again and ask you to explain them. There will be some MCQ type questions with certainty and maybe some open questions. Don't forget to write and erase and avoid erasable pens.

In response to individual questions: I will not be giving corrections for the practical exercises. If you have any questions about the tutorials, you should ask them now!
And I never said that the questions of the DS will not be about the practical exercises...
In last week's tp, some mistakes have been deliberately slipped in to force you to read and understand the code ....
When reading the dataset for the test, many did not see that an i increment was missing. To realise it: the results of the confusion matrix were aberrant (black images ranked before as many chances in each class).
For the DS, I can give you a code and ask you to understand it. For example, I can give you the description in keras of a neural network and ask you to draw it, to know what is the task (classification or regression) ....
I can also give you some hole code.
If some documentation is needed, it will be in the subject.
You should know the functions fit/predict/compile/add, the objects Model, Dense, Conv2D, Pooling, Flatten …

Videos

Machine Learning - YouTube

Introduction
A Gentle Introduction to Machine Learning
Machine Learning Fundamentals: Cross Validation
Machine Learning Fundamentals: The Confusion Matrix
Machine Learning Fundamentals: Sensitivity and Specificity
ROC and AUC, Clearly Explained!
Decision trees
Decision and Classification Trees, Clearly Explained!!!
StatQuest: Decision Trees, Part 2 - Feature Selection and Missing Data
Regression Trees, Clearly Explained!!!
Machine Learning Fundamentals: Bias and Variance

DImension Reduction
StatQuest: PCA main ideas in only 5 minutes!!!
StatQuest: Principal Component Analysis (PCA), Step-by-Step
StatQuest: t-SNE, Clearly Explained

Clustering algorithms
StatQuest: K-means clustering
StatQuest: Hierarchical Clustering
StatQuest: K-nearest neighbors, Clearly Explained
Clustering with DBSCAN, Clearly Explained!!!

Linear regression and logistic regression
Gradient Descent, Step-by-Step
Linear Regression, Clearly Explained!!!
StatQuest: Logistic Regression

SVM
Support Vector Machines Part 1 (of 3): Main Ideas!!!
Support Vector Machines Part 2: The Polynomial Kernel (Part 2 of 3)
Support Vector Machines Part 3: The Radial (RBF) Kernel (Part 3 of 3)

Neural networks
Neural Networks Pt. 1: Inside the Black Box

Neural Networks Pt. 2: Backpropagation Main Ideas
Neural Networks Pt. 3: ReLU In Action!!!
Neural Networks Pt. 4: Multiple Inputs and Outputs
Neural Networks Part 5: ArgMax and SoftMax

CNN

Convolutional Neural Networks (CNNs) explained
Neural Networks Part 8: Image Classification with Convolutional Neural Networks (CNNs)

Ensemble learning

Ensemble learners
Bootstrapping Main Ideas!!!
AdaBoost, Clearly Explained

INTRODUCTION

1. Learning Algorithms
2. Data Preparation
3. Analysis of the results

1. Learning Algorithms

What is Machine learning?

- Is the field of study that gives computers the capability to learn without being explicitly programmed
- Ability to learn
- Data analyst and data scientist

Algorithm Families. / CATEGORIES

- Unsupervised Learning (REQUIRES DATA)
    - clustering , Partitioning: “you ask the computer to separate similar data into groups (clusters)”.
    - Representation: use the computer to visualise data
    - Generation of new data: a model captures the probability distribution of your input, and can generate new data.

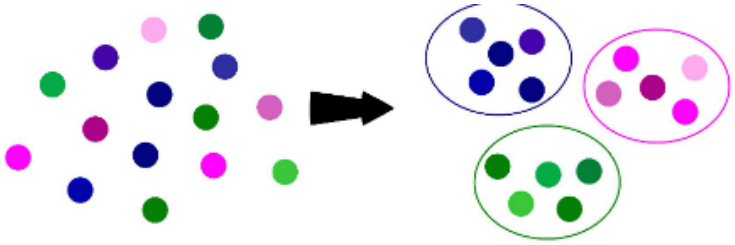
- Supervised Learning (REQUIRES DATA AND LABELS, data each pabel)
    - The model or algorithm is presented with example inputs and their desired outputs and then finding patterns and connections between inputs and outputs.
        - Objective -> learn general rule that maps inputs/outputs
        - Types
            - Regression (Linear/non Linear, SVM)
            - Classification (logistic, SVM, trees and forest)
            - Legends, Segmentation, Comprehension

Algorithms Syllabus

- *Clustering Algorithm*

A set of inputs is to be divided into groups “clusters”, Unlike classification, the groups are not known beforehand (is unsupervised).

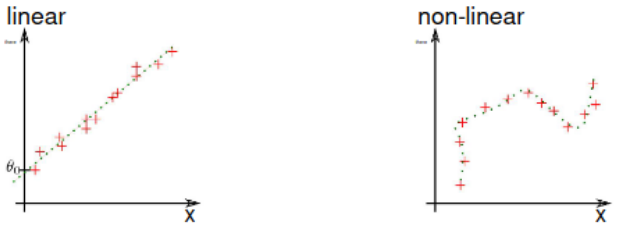
- K-means, Gaussians, hierarchical, density-based



- *Regression algorithms*

Predict the numeric value and outputs are continuous rather than discrete
“We draw a graphic, a line that predicts values” Supervised

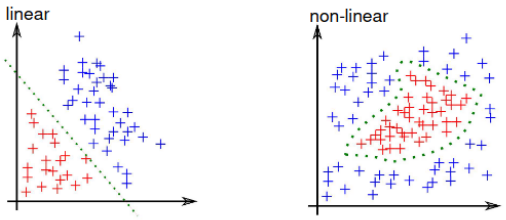
- Linear, decision trees, SVM, neural networks



- *Classification algorithms*

Inputs are divided into 2 or more classes, and the learner must produce a model that assigns unseen inputs to one or more of these classes and predicts later. Supervised

- two groups: binary classification, multiple classification
- Lineas, decision trees, SVM, neural networks



- *Set algorithms*
  - Boosting, random trees

- *Introduction to deep learning*
  - CNN (convolutional networks)

2. Data Preparation

To process the data to apply our algorithm to implement Machine Learning we follow these steps

- Collection data, we select the data
- Preparation, we do a data cleaning
- Codign, transforming variables, Inputs convert to be able to read the machine (Convert Image to a matrix N*N)
  For digits we always apply normalisation or standaritation.

3. Analysis of the Results

How do we split the data in Machine Learning? / Data Separation

- Training data: Party of data we use to train our model. Must include elements of each class
- Validation data: Part of data that is used to do a frequent evaluation of the model, fit on the training dataset along with improving involved hyperparameters (set parameter) before the model begins.
- Testing data: When we have a model trained, testing the data provides, we need to evaluate the performance of the algorithm with it

Proportion used:

60% training	20% validation	20% testing
80 “”	0 “”	20 “”

If the number of data is low, we use **cross-validation** (which is going our test data and our train data) ***We can also say that it is a technique to check how a statistical model generalizes to an independent dataset***. (puntuación a lo bruto)
CrossValidation allows us to compare different machine learning methods and get a sense of how well they will work in practice.
Is a technique in which we train our model using the subset of the dataset and then evaluate using the complementary subset of the dataset.

the three steps involved in cross-validation are :

1. Reverse some portion of sample data-set
2. Using the rest data-set train model
3. Test the model using the reverse portion of the data-set

Evaluation

When we want to evaluate how well and algorithm is applied for a case, we can calculate a punctuation (we use it on sklearn) or we apply a Confusion Matrix:
Real (the actual values of our dataset); predicted(the values predicted by our classifier)

Predicted / Real	has heart disease	does not have heart disease
has heart disease	True Positive (142)	False positive (22)
doesn have heart disease	False negative(29)	true negative (110)

Real / Predicted	has heart disease	does not have heart disease
has heart disease	TP	FN
doesn have heart disease	FP	TN

true positive: positive data(real) calculated as positive (predicted) 142 TP
142 samples that we expect to be positive our model predicted to be positive

true negative: negative data calculated as negative 110 TN
110 samples that we expect to be negative came back negative

false positive: negative data calculated as positive 22 FP
22 samples that we expect to be negative came back positive

false negative: positive data (real) calculated as negative(predicted) 29 FP
29 samples that we expect to be positive came back negative

With this data we can calculate sensitivity,specificity, , ROC and AUC, with this we can decide which algorithm we are going to use

- Sensitivity

  Tell us what % of patient with heart disease were correctly identified (+ proportion corrected)

$$S = \frac{TP}{TP + FN}$$

- Specificity

  Tell us what percentage of patient without heart disease were correctly identified (- proportion corrected)

$$SP = \frac{TN}{TN + FP}$$

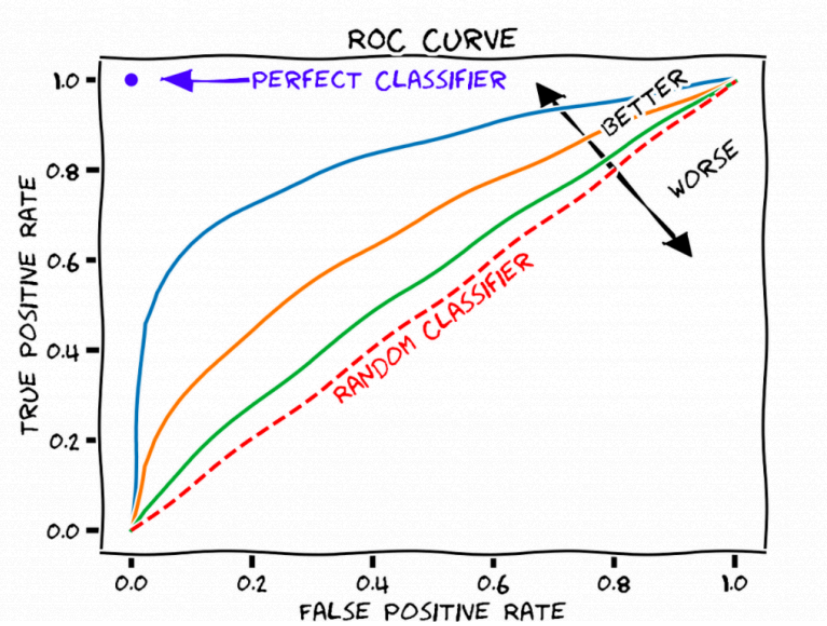
ROC Curve AUC and PR
Receiver Operating Characteristic ROC

Area Under Curve AUC

- FPR: PR rate, tell us ratio op negative examples that are incorrectly classified

  - $$FPR = \frac{FP}{TN + FP}$$

  Model evaluation: ROC is nothing but the plot between sensitivity and PR rate across all possible threshold(limites)
  AUC: is the entire area beneath this ROC curve, measures how well a model is able to distinguish between classes



Metrics For multi-class classification
For classification we use:

- Precision accuracy
- Recall recall
- Calculation of VP FP FN
- Macro
- weight

Supervised and unsupervised

Supervised learning is a machine learning approach that's defined by its use of labelled datasets. These datasets are designed to train or "supervise" algorithms into classifying data or predicting outcomes accurately. Using labelled inputs and outputs, the model can measure its accuracy and learn over time.

Unsupervised learning uses machine learning algorithms to analyse and cluster unlabeled data sets. These algorithms discover hidden patterns in data without the need for human intervention (hence, they are "unsupervised")
https://www.geeksforgeeks.org/supervised-unsupervised-learning/

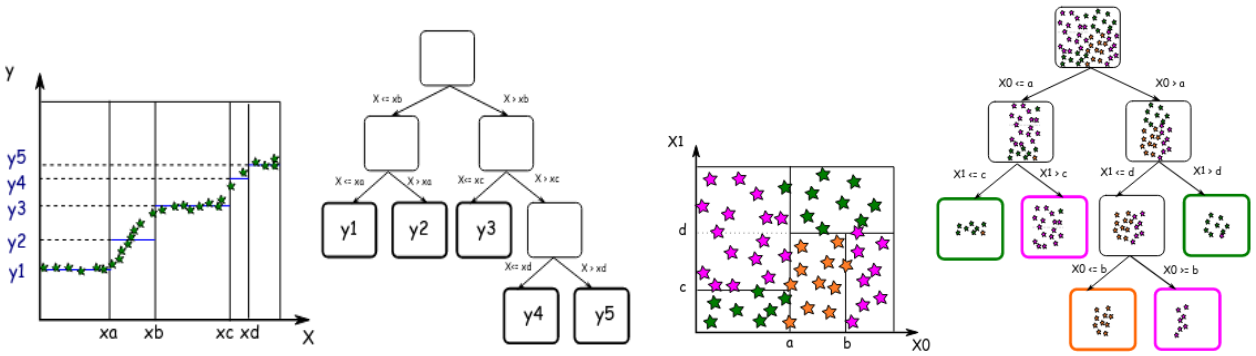
Supervised(data and labels)	**Unsupervised(just data)**
Linear Regression	Clusterning(k means, etc)
Logistic Regression	PCA and tSNE, dimension reduction
SVM	
Decision trees	
Neural networks/CNN	

DECISION TREES

1. Impurity
2. Building a tree

What is a decision tree?

- Is a supervised learning, can be used to solve both regression and classification problems
- Decision trees use the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree. We can represent any boolean function on discrete attributes using decision trees



With Regression

With Classification

When would the tree stop growing?

- Don't build very deep trees in order to avoid overfitting.

How to select a feature and a threshold?

- Choose the decision that will split the data in the equally size subsets
- Choose the decision that will lower the global error
- Need to choose a metric

1. Impurity

- We need to look and measure the quality of a node
- Always look for a low impurity (needs to be pure a node = same value)
- If the results are more uniform it equals low impurity, it equals low deep.

How to Measure impurity in

- **Classification Tree**

In a classification tree, if its continues splitting into more nodes, if there are a lots of nodes it equals high impurity

- EX: {1,1,0,1,1} is pure than {0,1,0,1,1} “The first is more uniform”
- Percentage of majority class

  {1,1,0,1,1} is pure class 1 at 80% while {0,1,0,1,1} is pure class 1 at 60%
- Entropy
- Gini Index

- **Regression tree**

In a regression tree if it continues to split if the cost is too high, we use MSE (Metric of Cost) to handle it.

Gini Index

It's a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower Gini Index equals low impurity, that’s what we’re looking for.

$$GINI(n) = 1 - \sum p^2(c|n)$$

$p(c|n)$ is probability of class c at node n(total posibilities)

Ex: A node contains samples from 3 classes. Calculate Gini Index for
{0,0,1,1,1,2,2,2,2,2}
Class 0: p(0) = 2/10 = 0.2 -> 0.02^2=0.04
Class 1: p(1) = 0.3 -> 0.09
Class 2: p(2) = 0.5 -> 0.25
GINI = 1 - (0.04 +0.09+0.25) = 0.62 / 0 = pure, 1 = impure, 0.5 equally distributed

NBCL = total = 10 “number of classes”
Max Value = 1- 1/NBCL min value = 0

Entropy (or Log Loss as measure of impurity)

$$E\,(n) \;=\; -\sum p(c|n)\,log(p(c|n))$$

Ex: Anode contains samples from 3 classes. Calculate the Entropy for {0,0,1,1,1,2,2,2,2,2}

Class 0: p(0) = 2/10 = 0.2
Class 1: p(1) = 0.3
Class 2: p(2) = 0.5
ENTROPY= - [0.2*log2(0.2) + 0.3*log2(0.3) + 0.5*log2(0.5)] = 1.4

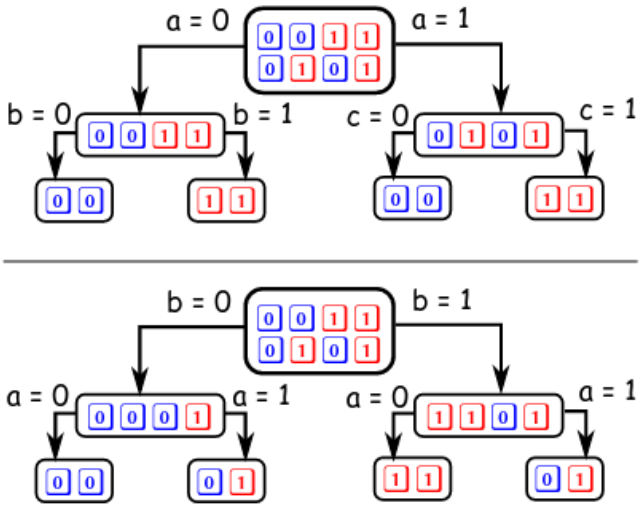
Max Value = 3* log (nbCL)/nbCL

2. Building Tree

We build all the possibilities of the tree and selecting the best ones we are going to find the bst solution, but we cannot do it, it takes so much time
So, we apply "greedy algorithm" to build a tree

- Best decision are taken locally, at each split (node)
- Compare the split candidate by a metric
- Is a viable solution, not the best one
- Be intuitive

a	b	c	class
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



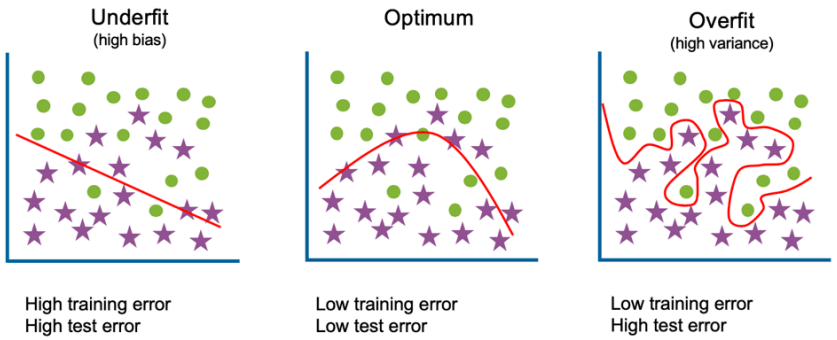
Solution for a boolean tree

Pruning

During the learning process we apply pruning to our tree in order to match the constraints, and after the learning ,it will reduce overfitting.

Underfit and overfitting

In the real world, the dataset present will never be clean and perfect. It means each dataset contains impurities, noisy data, outliers, missing data, or imbalanced data. Due to these impurities, different problems occur that affect the accuracy and the performance of the model. One of such problems is Overfitting



An overfitted model doesn't perform accurately with the test/unseen dataset and can't generalise well.

Overfitting occurs when the model fits more data than required(is complex), and it tries to capture each and every datapoint fed to it. Hence it starts capturing noise and inaccurate data from the dataset, which degrades the performance of the model.

Advantages	Disadvantages
Easy to interpret	Unstable (small number of data can change the tree and thus the prediction)
No need for preprocessing (normalisation)	Easi biassed with unbalanced dataset
Easy work with numeric and categorical data	No guaranty to have an optimal tree
Good with large dataset	Not precise with ML algorithm (only thresholding components)
Way for select important features	Causes normally overfitting
Fast inference	

DIMENSION REDUCTION: PCA, tSNE

1. Introduction

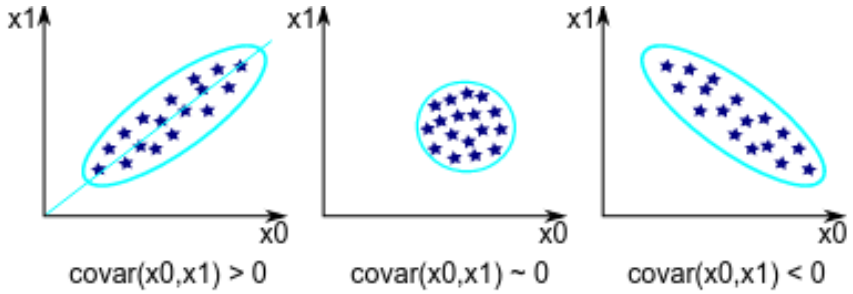
Mean (media) Variance and Covariance

With n samples of dimension 1 (scalars) {x0, x1 . . . x n−1} (n total)

Mean: $\mu = 1/n * \sum x$ (Add all data values and divide by sample size n).

Variance: $var(x) = \sigma^2 = 1/n \sum (x - \mu)^2$ (find mean, then calculate each $(x - \mu)^2$, sum that, divide by the number of data points)

Covariance: $(x0\ x1) = 1/n \sum (x0 - \mu0)\ (x1 - \mu1)$



What is Dimensionality Reduction?

In ML classification problems, there are often too many factors on which the final classification is done. These factors are basically called features, higher features equals harder it gets to visualise the training set and then work on it.

Dimensionality reduction (Reduction data) is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into selection and feature extraction.

There are two components of dimensionality reduction:

- Feature selection: In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem

- Feature Extraction: This reduces the data in a high dimensional space to a lower dimension space.

principal methods used for dimensionality reduction:

- Principal Component Analysis(PCA)
- tSNE

2. PCA Principal Component Analysis

- Unsupervised
- Analysis of variance-covariance matrix
- Reducing the dimension of data
- Visualisation of data of the reduce dimension: 2 or 3 Dimension
- Interpretation: dependency between variables
- PCA: often as pre-processing

Is an unsupervised linear dimensionality reduction and data visualisation technique for very high dimensional data. Main idea is to reduce the dimensionality of data that is highly correlated by transforming the original set of vectors to a new set: Principal Component.

I works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum

Steps

1. Construct the covariance matrix of the data
2. Compute the eigenvector of this matrix.
3. Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of the original data

PCA tries to preserve the global Structure of data (when converting n-dimensional data to n'-dimensional data then it tries to map all the clusters as a whole due to which local structures might get lost.)

eigenvectors :orthogonal vectors where the covariance matrix is diagonal
eigenvalue, often denoted by λ

Idea of PCA

- diagonalisation of Σ (matrix)
    - order eigenvalues by decreasing order
- if 0 is a eigenvalue: the corresponding dimensions can be removed
- the lower eigenvalues do not contribute a lot to the variance

Variance: $tr(\Sigma) = \sum \sigma 2$ (1° calculate variance then summarise all)

3. tSNE: t-distributed Stochastic Neighbour Embedding

is also an unsupervised non-linear dimensionality reduction and data visualisation technique. the maths. Behind t-SNE is complex but the idea is simple

- It embeds the points from a higher dimension to a lower dimension trying to preserve the neighbourhood(local structure) of that point. It involves Hyperparameters.

Idea of t-SNE:

- build map in which distances between points reflect similarities in the data
    - preserve local structure , try to avoid all points collapsing
- non linear dimension reduction
    - convert affinities of data points to probabilities represented by Gaussian joint probabilities
    - affinities in the embedded space are represented by Student's t-distribution

- Stochastic algorithm: multiple restarts with different seeds can yield different results.

why Gaussian distribution

(original space) we want to capture close elements and do no care of distant elements

why Student's t-distribution

(embedded space) the samples are initially randomly projected . We need to
be able to capture them.

Unlike PCA, tSNE tries to preserve the local structure of data by minimising the Kullback-Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the map.

tSNE: Is a iterative algorithm

- random initialisation of samples in the embedded space
- iterative minimisation of the KL divergence:
    - compute the distance between embedded points
    - use the t-distribution to transform these values + normalisation
    - compute the gradient of KL and move the sample points in the embedded space.

parameters of tSNE

- perplexity (between 5 and 50)
- early exaggeration factor -> optimization in 2 step -> exaggeration phase + final optimization
- learning rate : not too small not too large
- maximum number of iterations 5000
- angle

Differences between PCA and tSNE

PCA - linear reduction

- tries to preserve the global structure
- i does not work well as t-SNE
- does not involve hyperparameters
- can handle outliers
-determinist algorithm
- it work by rotating the vectors for preserving variance
- we can find decide on nd<t how much the variance to preserve using eigen-values

tSNE - non linear-reduction

- tries to preserve the local structure
- good
- it involves hyperparameters -> perplexity learning rate and number of steps
- handle outliers
- non-determinism / randomised algorithm
- it works by minimising the distance between the point in a gaussian
- we cannot preserve variance instead we can preserve distance using hyperparameters

CLUSTERING ALGORITHMS

1. Similarity - Distance
2. Partitioning Approaches to clustering
3. Hierarchical clustering Method
4. Density Based clustering
5. Evaluation

Clustering

Is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups

- Collection of objects on the basis of similarity and dissimilarity between them.
- unsupervised learning (no labels)
- find the class labels directly from the data
- cluster (collection of data object)
    - similar to one another in the same cluster / dissimilar to object in different cluster
    - need a way to calculate object similarity/distance
- typical application
    - a stand-alone tool to get insight into data distribution
    - preprocessing step for other algorithm

Ex: Marketing: help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs

What is good clustering

- A good clustering will produce high quality cluster with
    - high intra-class similarity
    - low inter-class similarity
- the quality of the cluster result -> depend on
    - the similarity measured used
    - his implementation
- the quality is measured by its ability to discover some or all of the hidden patterns

1. SIMILARITY- DISTANCE

Similarity
How would you do a binary clustering of these images?

food : muffin, bagel and fried chicken
pet : chihuahua, labradoodle and puppy

¡¡ CHOOSING THE DISTANCE IS CRUCIAL IMPORTANCE

Interval.scaled variables
How to standardise data

1. calculate the mean absolute deviation sj

$$s_j = 1/n \sum |xij - \mu j|$$ "media absoluta desviación, variance"

$$\mu_j = 1/n \sum xij$$ "media"

2. calculate the standardised measurement (z-score)

$$z(ij) = (x(ij) - \mu j) / sj$$

1. media: age = 60 media, tmb se haría para el salario
2. media absoluta desviación
   1/4 * (|50-60| + |70-60 | + 0 + 0) =20/4 = 5

3. z-score
   z(1) = (50-60)/5 = -2
   z(2) = (70-60)/5 = 2
   z(3) = 0 / 5 = 0
   z(4) = 0 / 5 = 0

age salary	age salary
alice 50 11000	alice -2 -2
bob 70 11100	bob 2 0.7
charly 60 11122	charly 0 1.3
dany 60 11074	dany 0 0

standardise data -is more easy to use

Distance Cluster

we calculate distance between cluster with -> Minkowski distance

- Minksowski distance

  dq(i,j) -> FORMULA

  $$dq\ (i, j) = q\sqrt{(\sum|xik - xjk\ |q)}$$

  reuse formula

  If q=1: Manhattan distance - raiz 1 elevado a 1

  If q=2: Euclidean distance - raiz 2 elevado a 2

  If q = inf -> MAX DISTANCE

  else use minkowski distance

Alternative Distance between cluster

- single linkage -> smallest distance between an element in one cluster and a element in other

- complete linkage -> largest distance between an element in one cluster and a element in other

- average linkage -> average distance between an element in one cluster and a element in other

- centroid -> distance between the centroids of 2 clusters

  centroid = midpoint of a cluster

- Menoid : distance between the menoid of two cluster

  menoid: one chosen, centrally located object in the cluster

Major clustering approaches

- partitioning (k means k medoids) *
- hierarchical *
- density-based *
- grid-based
- model-based

- spectral clustering
- frequent pattern based

2. PARTITIONING APPROACHES TO CLUSTERING

Method: Construct a partition of a dataset D of n objects into a set of k clusters, minimising the sum of squared distances

given a k : find partition of k clusters that optimises the chosen partitioning criterion

- global optimal : exhaustively enumerate all partitions
- heuristic method: use k-means and k medoids algorithm

- k-means: each cluster is represented by the centroid of the cluster
- k-medoids or PAM: each cluster is represented by one of the objects in the cluster(menoids)

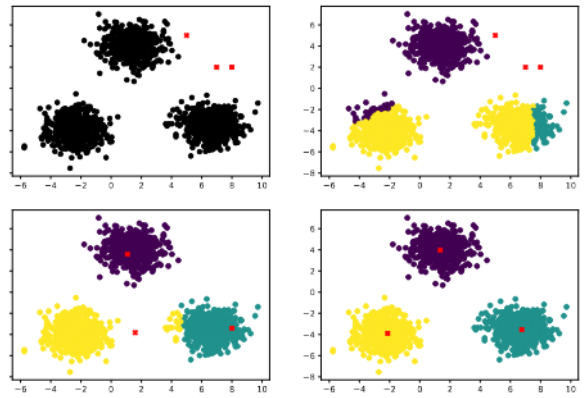
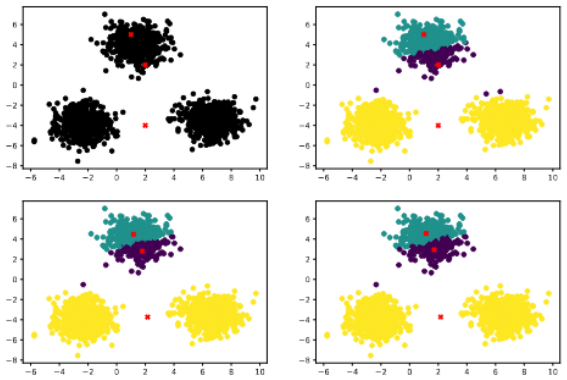
k-means

given k
1. choose k initial centroids, each labelled as k
2. assign each object to the cluster having the nearest centroid point
3. Compute centroids of the cluster of the current partition
4. back to step 2 , stop when no more change (until its good)

- strength - efficient O(tkn) / PAM is more heavy

  Comment: Often terminates at a local optimum
  - global optimum may be found using optimisation methods -> annealing and evolutionary algorithm
  - practice : we need to do several trials using random initialization

- Weakness
  - applicable when only mean is defined
  - need to specify k, number of cluster in advance (or use elbow method)
  - unable with noisy data
  - no suitable to discover clusters with non-convex shapes

K-means good example	K-means bad initialisation
	

mini-batch kmeans

a variation of kmean for scalability improving

mini-batch = subset of dataset

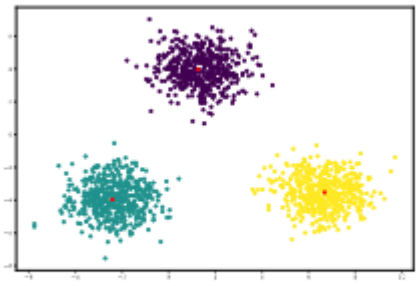
randomly sampled at each iteration of the training process

K Medoids PAM

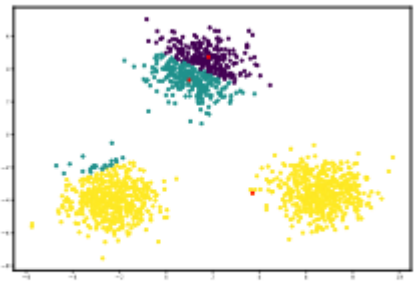
- find representative objects, called medoids in cluster
- start from a initial set of medoids and iteratively replaces one of the medoids by one of the

non medoid if it improves the total distance of the resulting clustering

- less sensitive to outliers that kmean
- work great with small data set, don't scale well with big ones
- CLARA / CLARANS



good initialization



bad initialization

C-mean clustering method

- fuzzy extension k-means
- a record can belong to more than one cluster to a degree (soft vs hard assignment)

3. HIERARCHICAL CLUSTERING METHODS

hierarchical clustering methods

- Input: distance matrix
- Output : a dendrogram (tree of clusters)
- this method does not require the number of clusters k as an input, but may need a termination condition

- 2 approaches
  - bottom up approach -> agglomerative
  - top down approach (DIANA = divide analysis)

- Algorithm
  - start with all single data as their own cluster
  - merge clusters together according to a linkage criteria
  - ends when all data are in the same cluster

- the linkage criteria -> determines the metric used for the merge strategy
  - ward - minimises the sum of squared differences within a cluster
  - maximum or complete linkage
  - average linkage
  - single linkage

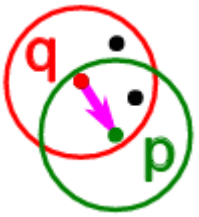
4. DENSITY BASED CLUSTERING METHODS

cluster = areas of high density separated by areas of low density
clustering based of density (local cluster criterion) such as density-connected points
Solve the problem of nesting. / NBSC Scan in sklearn

features

- discover cluster of arbitrary shape
- handle noise
- one scan
- need density parameter as a termination condition
- DBSCAN, OPTICS, DENCLUE, CLIQUE

basic concepts



- 2 parameters
  - e: Maximum radius of the neighbours
  - MinPts: minimum number of points in an e-neighbors of that point

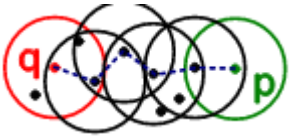
- Neighbourhood: N

- core point: if at least MinPts points are within distance e of p(including p)

- directly density-reachable; A point p is directly density-reachable from a point q w.r.t MinPts if:
  - p belong to N(q)
  - core point condition N(q) >= MinPts

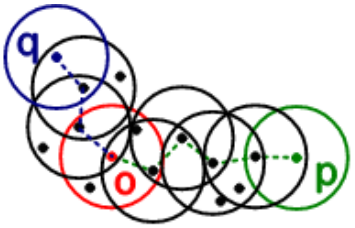
Density reachable

- a point p is density reachable from a point q w.r.t e MinPts if there is a chain of points
- p1 .. pn -> p1 = 1 pn = p such that i+1 is directly density reachable from pi



Density connected

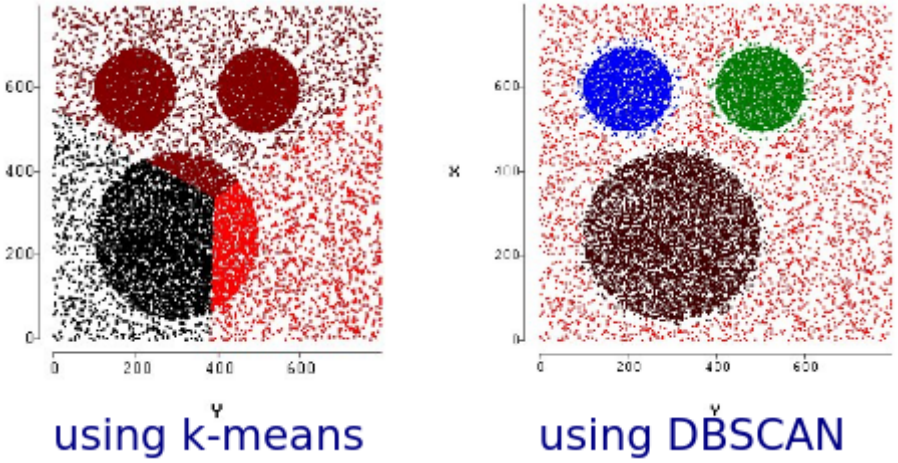
- a point p is density connected to a point q w.r.t e, MinPts if there is a point o such that both q and p are density reachable from o w.r.t e and MinPts



DBSCAN

1. choose a core point p , not already assigned to a cluster
2. add all density- reachable points from p to this cluster
3. go back to step 1 until no more points without cluster
4. all points non reachable from any other point are outliers or noise point

see image difference using k means and using dbscan



5. EVALUATION

evaluation of clustering types

- ground truth known - ARI MI V-measure
- ground truth not known - davies-bouldin index dunn index silhouette

measures

internal validation : Separation, Compactness
we use *silhouettes coefficients*

s = b-a / max (a,b)
a - mean distance between a sample all other samples from same cluster
b - mean distance between a sample all other samples in the next nearest cluster

for a dataset:: mean of silhouettes coefficients for each sample
metrics.silhouette score

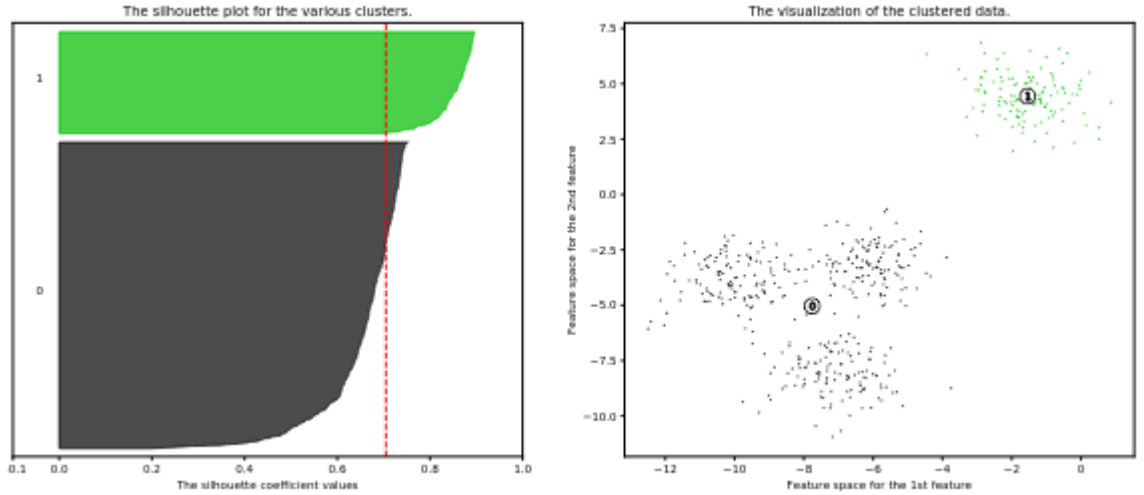
higher -> more compact

REMEMBER

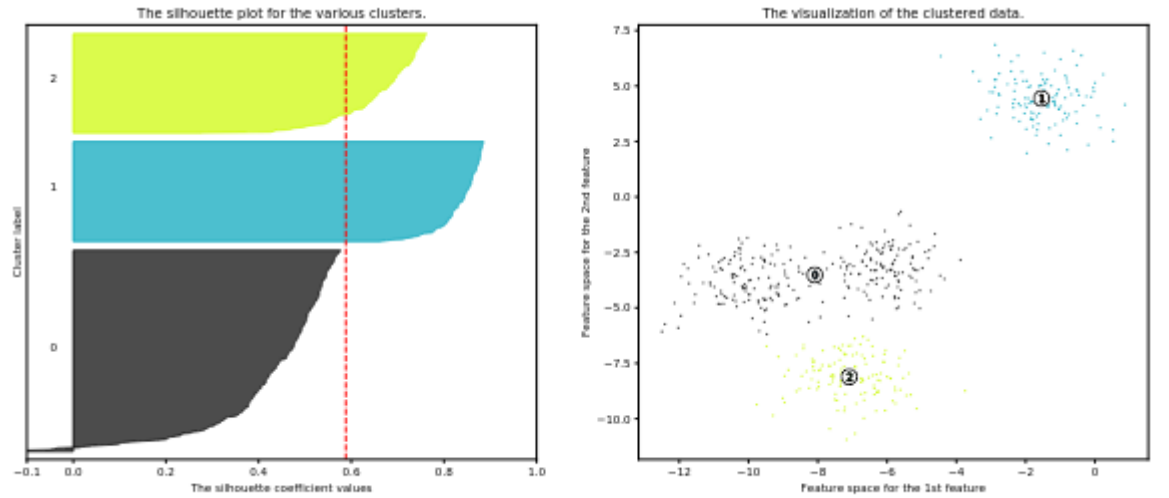
- numeric data -> standardisation is important
- many algorithm exist with different advantages and drawbacks - find the correct one
- parameters of the algorithm are important - tune them
- evaluation metrics are useful but should be considered carefully

Examples of silhouettes with k-means, k number of groups you want to discover

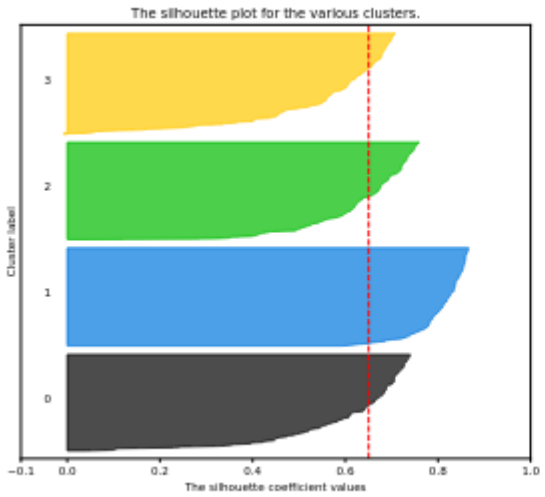
K = 2



K = 3

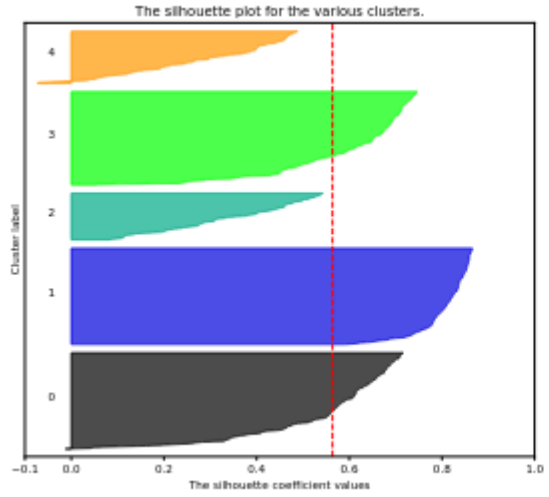


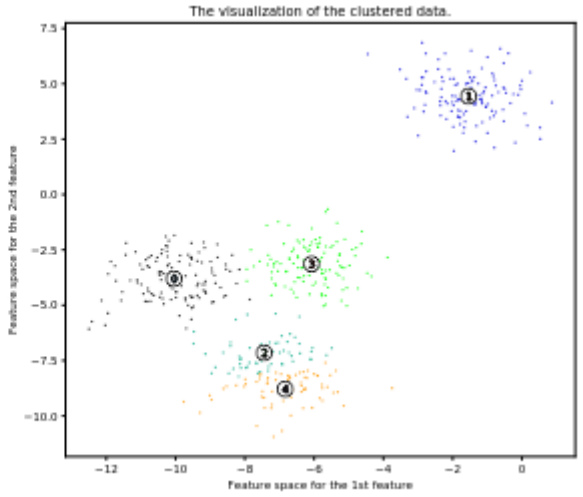
K = 4


The silhouette plot for the various clusters.
Cluster label
The silhouette coefficient values


The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature

K = 5


The silhouette plot for the various clusters.
Cluster label
The silhouette coefficient values


The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature

LINEAR REGRESSION AND LOGISTIC REGRESSION

1. Linear Regression
2. Logistic Regression

1. Linear Regression

What is Linear Regression?

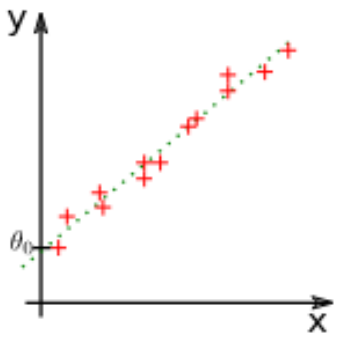
Is a Machine Learning algorithm, based on supervised learning that performs a regression task.

Y =mX + b

- Y: depend variable we want to predict
- X is the depend variable we are using for prediction
- B: is a constant, known as the Y-intercept. If X = 0. Y would be equal b,

- Data: There is a M data of dimension:
  - 1: Scalar value: x
  - N: Many scalar values: vector X [x1,x2 ... xn]

The goal is that we want to predict the y value. Ex: we want to predict the amount of rain from altitude in the Alps.



The regression consist of determine the value y with respect to X

Linear (DIM 1): the function is a line parameterised by θ = [θ0 θ1]

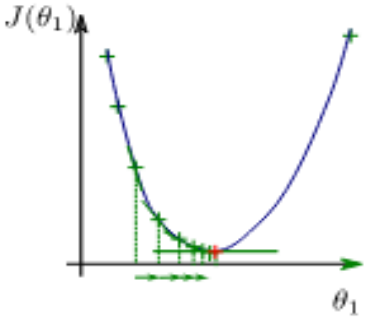
$y = \theta 0 + x * \theta 1 \; (y = mx + n)$

Learning: determine θ0 and θ1.

Cost Function of error $j(\theta) = 1/2m \sum (h\theta(x\hat{}i) - y\hat{}i)\hat{}2 \; ; \; i = 0 \; -> \; m - 1$

Determine θ (constant), we apply gradient descend (to calculate θ0).

Gradient Descend



We need to minimise the loss to make a good predicting algorithm. Our objective is to find the deepest point (global minimum) of this function

Two thing are required to find the deepest point

- Derivative - to find the direction of next step
- Learning rate - magnitude of next step

The idea is you first select any random point from the function. Then you need to compute the derivative $dJ/d\theta 1$, calculate the loss. This will point to the direction of the local minimum. Now multiply that with the learning rate, we stop when we find a value close to 0

Optimise clusters, lines . etc.

α -> learning rate

has to be chosen carefully

- if to small -> slow the convergence (small step)
- too big -> oscillations

3 types of gradient descent

- *batch gradient descent*
  - all training samples for each step

- *stochastic gradient descent*
  - one training sample for each step (need to shuffle training data) (choose a subset of data)

- *minibatch (b=10)*
  - a set of b training samples for each step

2. Logistic Regression

What is Logistic Regression?

Is a machine learning algorithm based on supervised learning, learning to predict an output when it's given an input vector. We know the class/labels y from all training data.
The dependent variable y is binary in nature having coded as either 1 or 0, can be multinomial if it has more than 2.

- Logistic regression is a classification method
- Linear regression leads to values $h\theta(x) \in R$
- the idea : values in (0 1) then thresholding at 0.5

we do a data separation according to labels (0 or 1)

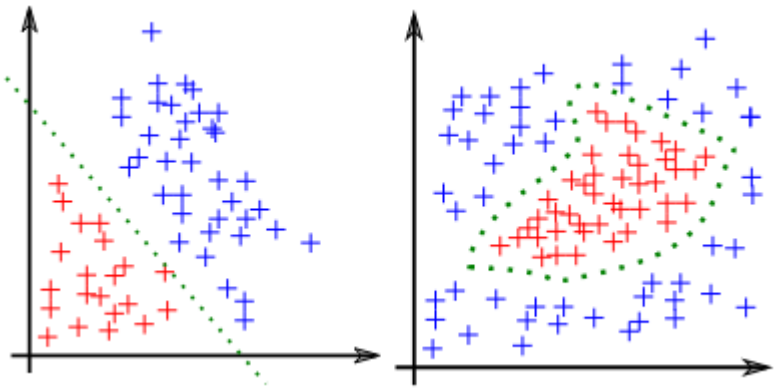
- linear separation : we draw a line, plane or hyperplane (2D,3D,4D)
- Non-linear separation: polynomial or gaussian (cuando no son líneas)

Notation

data : x = [x1 x2...]

labels : $y \in \{0, 1\}$

decision criteria $h\theta$ parameterised by θ $\theta = [\theta 0 \ \theta 1 \ ...]$



Decision boundary (como dividimos la grafica)

line of equation : $\theta 0 + \theta 1 \ x1 + \theta 2 \ x2 = 0$ or $\theta T \ x = 0$

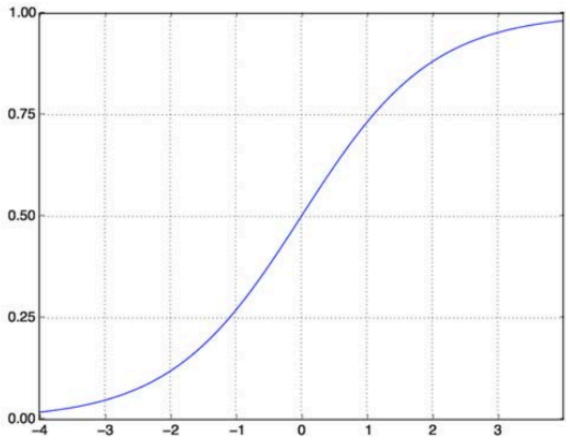
Decision

if $\theta T \ x >= 0$ then y = 1 else y = 0

Sigmoid Function

The sigmoid function to draw an Logistic function is s(z) = 1/1 + e^−z

if hθ(x) x >= 0.5 then y = 1 else y = 0.5



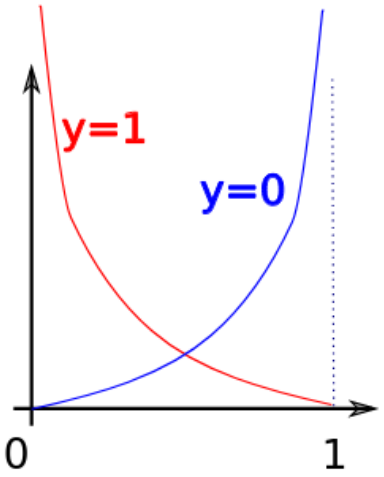
Learning process

we get data (m data), we try find θ applying error minimisation and the gradient ascent

Binary cross-entropy Error minimisation

The most common loss function for training a binary classifier is binary cross entropy

$$J = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)})))$$



Non linear logistic regression

- polynomial model

  x = [x0x1 . . . xn x0^2 x^2 . . . x0x1x2 . .

  - under-fitting
    - add parameters
  - over-fitting
    - reduce the number of parameters

Regularisation

-> to avoid over-fitting

add $\|\theta\|$ to the cost

* norm : L1 (Lasso), L2 (Ridge), Elastic-Net

* many solvers:
  - coordinate descent (C++)
  - Stochastic Average Gradient SAG
  - Broyden-Fletcher-Goldfrab-Shanno algorithm

SVM SUPPORT VECTOR MACHINE

3. SVM Classification

Introduction

Is a relatively simple Supervised Machine Learning algorithm used for classification and/or regression.

- SVM stands for support vectors (data points that are closest to the hyperplane).

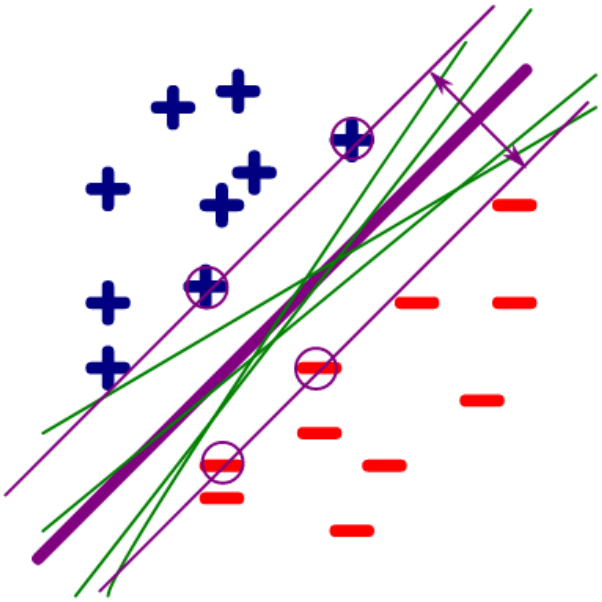
SVM finds a hyper-plane that creates a boundary between the types of data (2D is a line)
We plot each data item in the dataset in a N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyperplane to separate the data.

- SVM performs binary classification (choose between 2 classes)
- SVM multi-class problems - technique to solve multi-class problem

We can difference:

- Binary Classification
- Multiclass Classification

Concepts



Support vectors (circle purple)
Data Points that are closest to the hyperplane

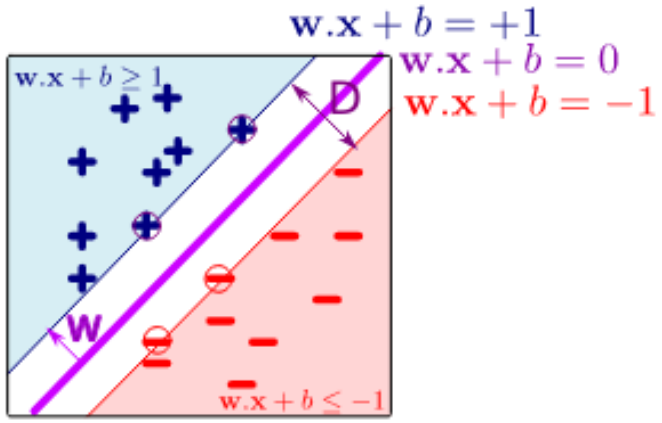
Hyperplane (line purple)
Is a decision plane or solace which is divided between a set of objects having different classes

Margin (arrows purple)
Gap between two lines on the closest data point on different classes

Goal

Is to divide the datasets into classes to find a Maximum Marginal Hyperplane MMH

Problem Formalisation



Margin Maximisation D (What are we looking for)

$$D = \frac{2}{\|w\|}$$

We need to minimise

$$\|w\| \; or \; \frac{1}{2}\|w\|^2$$

Hyperplane: W*x + b = 0 is our hyper plane (a line because is 2D)

Labelled data

- Positive data: y = +1 $w.x + b \geq 1$
- Negative data: y = -1 $w.x + b \leq -1$
- Thus : $y\,(w.x + b) \geq 1$

Prediction: Sign(w*x + b)

SVM Problem

Minimisation of $\frac{1}{2}\|w\|^2 \; or \; \|w\|$ under the constraint $\forall i,\; yi\,(w.xi + b) \geq 1$

Dual Problem

Maximisation of $\sum i\; \alpha i - 1/2 * \sum i \sum k\; \alpha i\; \alpha k\; yi\; yk\; xixk$ under the constraint $\forall i\; \alpha i \geq 0$ and $\sum i\; \alpha i\; yi = 0$

SVM: Optimal solution

To the dual problem we add the Karush-Kuhn-Tucker condition

$$\forall i\; \alpha i\; (yi\,(w.xi + b) - 1) = 0$$

And we can look for the optima solution

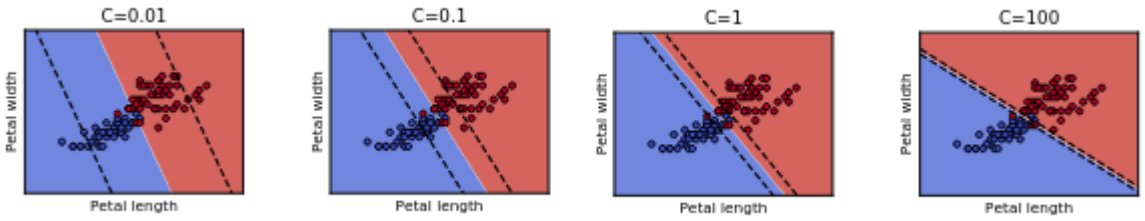
- If $\alpha i > 0$: $yi\,(w.xi + b) - 1 = 0$: on the margin
- If $yi\,(w.xi + b) > 1$: $\alpha i = 0$
- Thus: $w = \sum i, \alpha i > 0 \alpha i\; yi\; xi$

Soft Margin

Represent the error. Minimisation of $1/2\ \|w\|2\ +\ C\sum i\ \zeta i$ under the constraint $i\ yi\ (w.xi\ +\ b) \geq 1\ -\ \zeta i\ and\ \forall i\ \zeta i \geq 0$

When we allow misclassification, the distance between the observations and the threshold is called a Soft Margin

- Parameter C (Ck = C * w)
  * C is a hyperparameter determining the penalty for misclassification
  * A small C will give a wider margin , at the cost of some misclassifications
  * A huge C value will give the hard margin classifier and tolerates zero constraint violation
  * Find the C value such that noisy data does not impact the solution too much



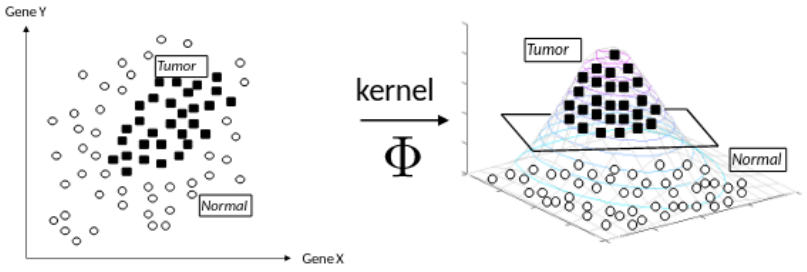
Unbalanced data

The idea is to increase the cost of bad classification for the smallest class to prevent them for being overwhelmed by the majority

C is replaced by C+ for positive data and C- for negative data.

Kernel, for non-linear data

In practice , the SVM algorithm is implemented with a kernel that transforms an input data space into the required form. SVM uses a technique called kernel trick in which the kernel takes a low dimensional input space and transforms it into a higher dimensional space. In simple words, Kernel converts non-separable problems into separable problems by adding more dimensions to it. **We transform the data to implement a Support vector classifier with kernels, kernels just calculate the relationships between point(don't do the tranforma.)**



Data is not linearly separable in the input space

Data is linearly separable in the feature space obtained by a kernel

$$\Phi : \mathbf{R}^N \rightarrow \mathbf{H}$$

Types of kernel

Calculating the high-dimensional relationship without transforming the data -> Kernel trick

- Linear Kernel
- Polynomial Kernel (degree and gamma)
- Radial Basis Function (RBF) Kernel (C)
- Sigmoid (C and gamma)

In practice we need to find the best kernel for our problem, we should compare kernels with best parameters.

Here, gamma ranges from 0 to 1. We need to manually specify it in the learning algorithm. A good default value of gamma is 0.1 (RBF)

Methods of parameters estimation

- Exhaustive search (GridSearchCV)
- Randomised Search (RandomizedSearchCV)

There're also custom kernels. We will use it for text or genes. Images, videos ….

Multiclass

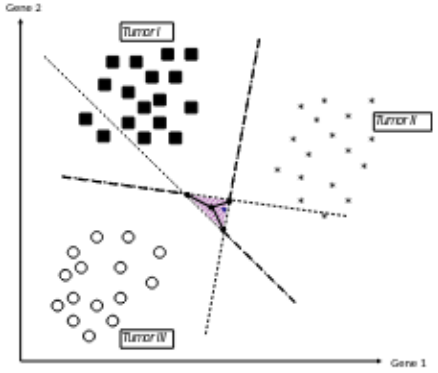
When the machine wants to classify an instance as only one of three classes or more it's called multiclass classification.

Natively SVM doesn't support multiclass classification natively. It supports binary classification and separating data points into two classes. For multiclass, the same principle is utilised after breaking down the multiclass problem into multiple binary classification problems.

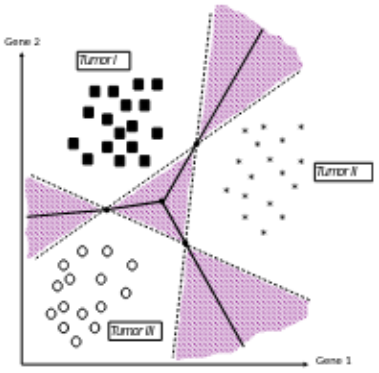
Approaches

- *One-to One*: map data points to high dimensional space to gain mutual linear separation between every two classes. Breaks down the multiclass problem into multiple binary problems. (we separate between 2 classes, binary)
- *One to Rest*. the breakdown is set to a binary classifier per each class (we separate between a class and all other at once)

- n classes
- two methods (`decision_function_shape(...)`)



- one versus one (`'ovo'`)
  - build $n(n-1)/2$ SVM classifiers
  - maximum vote from classifiers



- one versus the rest (`'ovr'`)
  - build n SVM classifiers
  - unbalanced classes

Each n would predict membership in one of the n classes

Active Learning with SVM

- Hyperplane estimation using small set of labelled data
- Estimation of distance for unlabelled data
- Selection of data close to the hyperplane (constraint)
- Redefine the estimation

1 dimension: the support vector classifier is a point
2 dimension: the support vector classifier is a line
3 dimension: the support vector classifier is a plane
4 dimension: the support vector classifier is a hyperplane

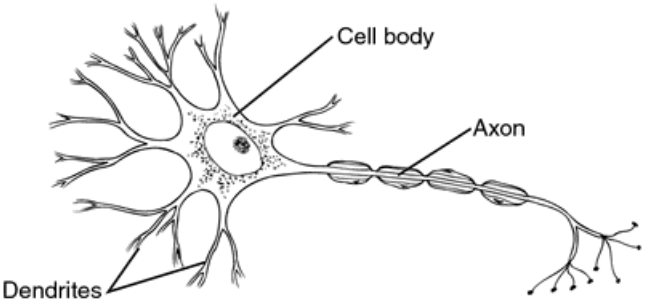
NEURAL NETWORKS

1. A Single Neuron
2. Perceptron
3. Multi Layer Perceptron
4. Experimentations

1. A Single Neuron

First we start with what it is a biological neuron

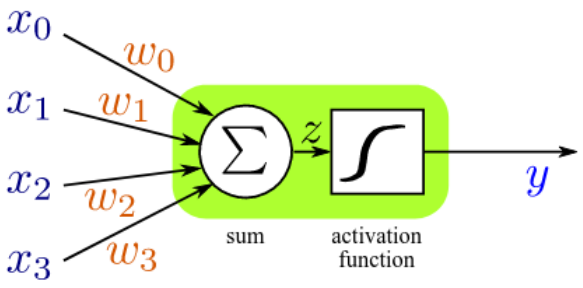
- Dendrites: They're our inputs.
- Soma or cell body: summation function
- Axon: activation function and output



An artificial neuron performs a weighted sum of its inputs, followed by an activation function

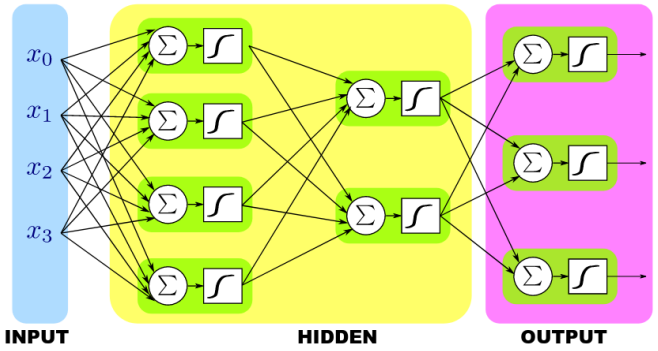
Now we look at an artificial network, the cells of our neural network. Parts:

- Inputs Xi (also outputs of other neurons)
- Associate weights Wi (synaptic modulation. Parameters we multiply
- Bias: special inputs X0 = 1 with weight Wo (the premier its bios W0) Parameters that we add.
- Cell body: sum of weight inputs
- Axo: its our activation function
- Generates and output y



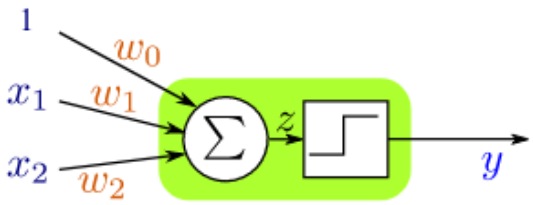
$$y = s(z) = s\left(\sum_{i=0}^{n} w_i x_i\right)$$

How It looks a full neural network



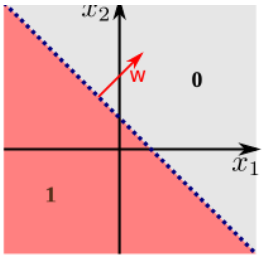
Formula for a simple single neuron:

We have a single neuron with 3 inputs: Bios, X1 with W1 and X2 with X2.



Activation Function/output is gonna be:

$y = s(z) = s(w0 + w1x1 + w2x2)$



The output its equivalent to divide the 2D plane by a line of equation, like this:

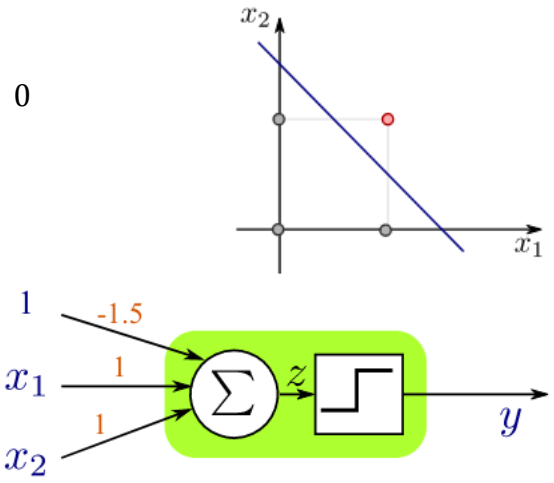
$w0 + w1x1 + w2x2 = 0$ or $w.x = 0$

Examples of single neurons are the operations AND, OR, NAR, NOR, XOR-

Boolean operation AND

Activation function: $-1.5 + x1 + x2 = 0$

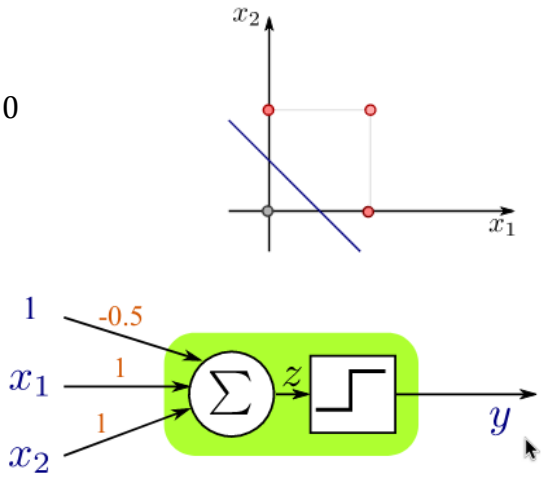
AND	0	1
0	0	0
1	0	1



Boolean operation OR

Activation Function: $-0.5 + x1 + x2 = 0$

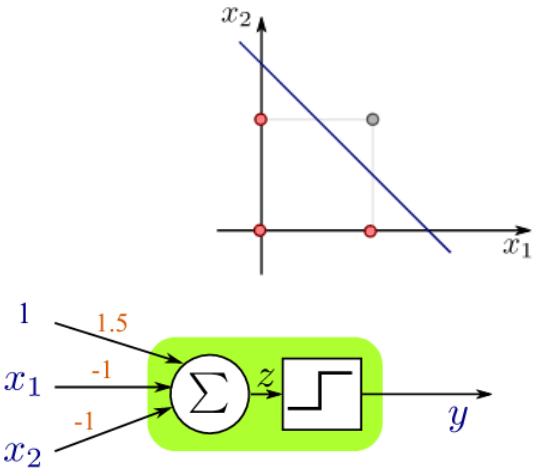
OR	0	1
0	0	1
1	1	1



Boolean operation NAND

Activation function: $1.5 - x1 - x2 = 0$

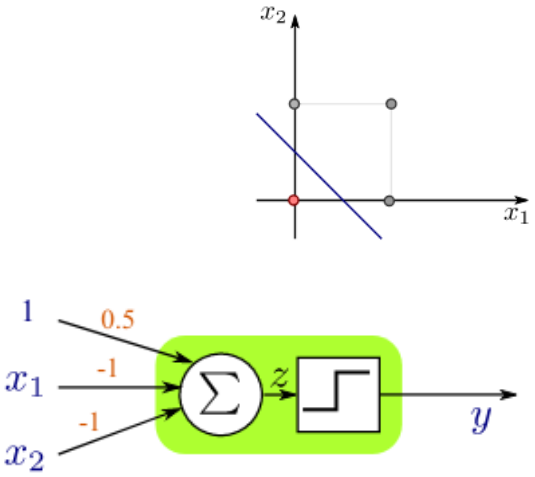
NAND	0	1
0	1	1
1	1	0



Boolean operation NOR

Activation Function: $0.5 - x1 - x2 = 0$

NOR	0	1
0	0	0
1	1	0



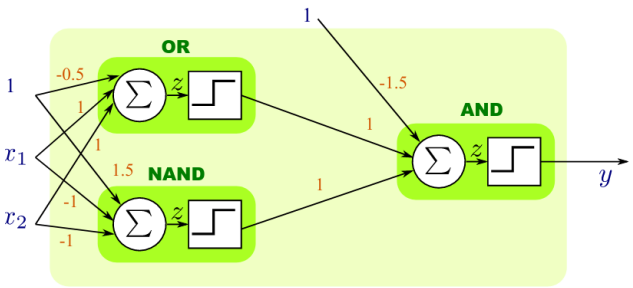
Boolean operation XOR

Imposible with a single neuron !!!

Solution express XOR using a combination of boolean operators

Activation Function: A XOR B = (A NOR B) AND (A OR B)

XOR	0	1
0	0	1
1	1	0

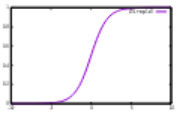


2. Perceptron

Activation Function

- Must be continuous and differentiable
- Is sigmoids, ReLU, softplus

- logistic function $s(z) = \frac{1}{1+e^{-z}}$ and $s'(z) = \frac{ds}{dz}(z) = \frac{e^{-z}}{(1+e^{-z})^2} = s(z)(1 - s(z))$



3. Multi Layer Perceptron (MLP)

A MLP neural network (Multi Layer Perceptron) is made of successive layers of neurons in which the outputs of the neurons of the layer (n) are connected to all the inputs of the layers (n+1). Hidden layers

Multilayer feed-foward neural network as universal approximator

It is proven that

- All boolean functions can be represented using a single hidden layer
- All borned continuous functions can be represented using a single hidden layer, with arbitrary precision
- All function can be represented using 2 hidden layers, with arbitrary precision

But

- How to determine the topology of the neural network?
- How many neurons per layer
- How to tune the weights? (learning phase)

Back Propagation Algorithm (Retro Propagation)

1. Initialise weights to small random values
2. Choose a random sample training item (Xj, Yj)
3. Compute total input Zi and output Yi for each unit (forward prop)
4. Compute δ for output layer δ = (1- Yp)(Yp - Yj)
5. Compute δ for preceding layers by backprop rule
6. Compute weight change by descent rule (repeat for all weights)

For efficiency , simplicity and location we use backpropagation

Input encoding

The number of neurons in the input layer is related to the structure of your data (Iris involve 4 neurons cause -> y dimension 4, 4 types of flower)
All values are real values from 0 and 1, if not we need to scale it

$$x = \frac{u - u\,(min)}{u(max) - u(min)}$$

	x_1	x_2	x_3
flour	1	0	0
butter	0	1	0
sugar	0	0	1

Output encoding

The number of neurons in the output layer depends on the problem to be solved
For a regression:

- One output neuron if its scalar value
- N output neurons if the dimension of the vectors to predict is n

For a multi classification

- N classes is n output neurons

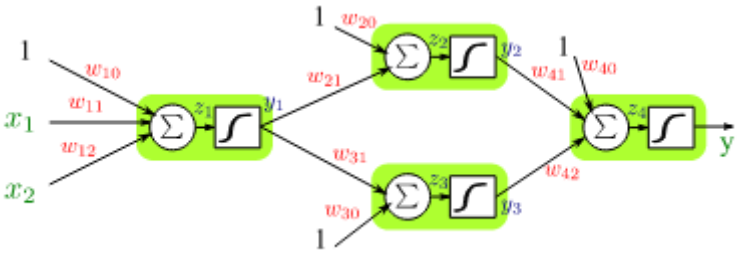
Training, en Neural network usamos el 20 20 60

1. Split the data set (randomly) into three subsets
    - Training set : used for adjust weights
    - Validation set: used to stop training
    - Test set: used to evaluate performance
2. Pick random, small weights as initial values
3. Perform iterative minimization of error over training set
4. Stop when error on validation set reaches a minimum (avoid overfitting)
5. Repeat step 2 several times (avoid local minima)
6. Use best weights to compute error on the test set

If data to small -> increase ir or divide it into 3 subsets, use cross-validation

Exercise

Network for regression:



Values of weights:

w_{10}	w_{11}	w_{12}	w_{20}	w_{21}	w_{30}	w_{31}	w_{40}	w_{41}	w_{42}
0.1	0.1	0.1	0.15	0.12	0.09	0.07	0.2	0.2	0.1

New incoming data:

$$x_1 = 0.4,\ x_2 = 0.6,\ y = 0.2$$

Considering η = 0.01 compute the weights at next step:

w_{10}	w_{11}	w_{12}	w_{20}	w_{21}	w_{30}	w_{31}	w_{40}	w_{41}	w_{42}

Step 1: Forward propagation. Compute the output of each neuron(there are 4 neurons in the image -> y1 y2 y3 and y). Sum of each weights

$s(z) = \frac{1}{1+e^{-z}}$

Adobe is the activation function for sigmoid, Relu is $s(z) = max(0, z)$

$y1 = s(z1) = s(w10 + w11 * x1 + w12 * x2)$

$y1 = s(0.1 + 0.1 * 0.4 + 0.1 * 0.6) = s(0.2) = \frac{1}{1+e^{-0.2}} = 0.55$

$y2 = s(z2) = s(w20 + w21 * y1)$

$y2 = s(0.15 + 0.12 * 0.550) = 0.554$

$y3 = s(z3) = s(w30 + w31y1)$

$y3 = s(0.09 + 0.07 * 0.550) = 0.5532$

$y = s(w40 + w41 * y2 + w42 * y3)$

$y = s(0.2 + 0.2 - 0.554 + 0.1 * 0.5532) = 0.59,\ higher\ than\ the\ expected\ value\ 0.2\ yj$

Step 2: Compute the δ for each neuron backwards, with delta rule

$yj = 0.2\ (el\ resultado\ que\ debería\ haber\ salido)$

$\delta = (y - yj)y(1 - y)$

Delta rule : $\delta i = yi(1 - yi) * \sum wik * \delta ik$

$\delta 4 = (y - yj)y(1 - y)$

$\delta 4 = (0.59 - 0.2) * 0.59 * (1 - 0.59) = 0.094$

$\delta 3 = \delta 4 * w42 * y3(1 - y3)$

$\delta 3 = 0.094 * 0.1 * 0.554()1 - 0.554 = 0.0023$

$\delta 2 = \delta 4 * w41 * y2(1 - y2)$

$$\delta 2 = 0.094 * 0.2 * 0.554(1 - 0.554) = 0.0023$$

$$\delta 1 = (\delta 2 * w21 + \delta 3 * w31) * y1(1 - y1)$$
$$\delta 1 = (0.0023 * 0.12 + 0.07 * 0.0023) * 0.55 * (1 - 0.55) = 0.00015$$

Step 3: update the weights
η: Learning rate = 0.01
Gradient descent: w ← w − η δ x
$w42 \leftarrow w42 - \eta\delta 4y3 = 0.995$
$w41 \leftarrow w41 - \eta\delta 4y2 = 0.199$
$w40 \leftarrow w40 - \eta\delta 4 = 0.1991$

$w30 \leftarrow w30 - \eta\delta 3 = 0.089$
$w31 \leftarrow w31 - \eta\delta 3y1 = 0.069$

$w20 \leftarrow w20 - \eta\delta 2 = 0.1499$
$w21 \leftarrow w21 - \eta\delta 2y1 = 0.1199$

$w10 \leftarrow w10 - \eta\delta 1 = 0.099$
$w11 \leftarrow w11 - \eta\delta 1x1 = 0.099$
$w12 \leftarrow w12 - \eta\delta 1x = 0.099$

w_{10}	w_{11}	w_{12}	w_{20}	w_{21}
0.099998	0.099999	0.099999	0.14998	0.11999
w_{30}	w_{31}	w_{40}	w_{41}	w_{42}
0.08995	0.06997	0.1991	0.1995	0.0995

Activation functions
the activation function can be the identity, a slot, a sigmoid or a ReLU function
Main Activation functions:

- Sigmoid
- ReLu: Rectified Linear Unit
- Softmax: transforms output values into values that can be interpreted as probabilities
- Others : (tankh, leakyReLU, PReLU …)

Why ReLU

- Problem of vanishing gradient
    - Delta rule and gradient descent
    - Different solutions

- Change to activation function to ReLU
- Residual networks
- Batch normalisation

Loss functions

- Classification: binary cross entropy, categorical cross entropy
- Regression : mse, rmse …

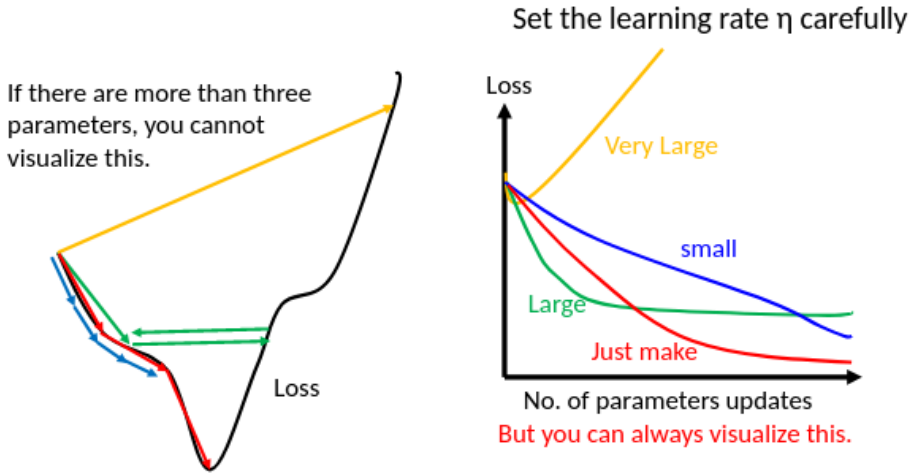
Learning rate

Learning a neural network consist in learning the values of the input weight of each neuron of the network

For learning, it is necessary to minimise a cost function which depends on the problem to solve.

In the case of regression we can use : MSE (loss function) mean square error

1 epoch corresponds to the number of iterations necessary to see the whole dataset (mini-batch case)



Layers of Nodes between the Input and the output Nodes are called Hidden Layers.

Conclusion

You must remember that:

- An **artificial neuron** performs a weighted sum of its inputs, followed by an activation function
- For the moment, the **activation function** can be the identity, a slot, a sigmoid or a ReLU function (we will check others)
- A **MLP neural network** (Multi Layer Perceptron) is made of successive layers of neurons in which the outputs of the neurons of the layer (n) are connected to all the inputs of the layers (n+1)
- The **number of neurons in the input layer** is related to the structure of your data (Iris involve 4 neurons cause -> y dimension 4, 4 types of flower)
- The **number of neurons in the output layer** depends on the problem to be solved
    - If it's a regression value: 1 neuron, try to guess if yes or no,
    - If it's a regression vector of dimension d : d neurons
    - If it's a classification into n classes: n classes
- **Learning a neural network** consist in learning the values of the input weight of each neuron of the network
- For l**earning**, it is necessary to minimise a cost function which depends on the problem to solve.
    - In the case of **regression we can use : MSE**
- **1 epoch** corresponds to the number of iterations necessary to see the whole dataset (divide by mini-batch case)

CONVOLUTIONAL NEURAL NETWORK

1. CNN
2. Architectures

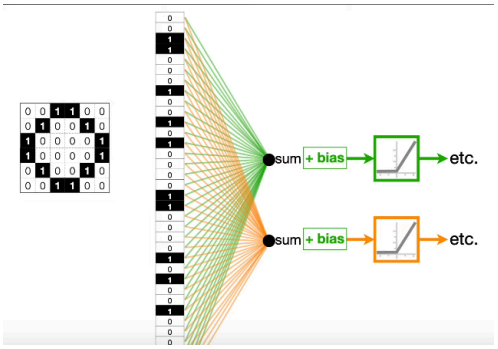
[Convolutional Neural Networks (CNNs) explained]
[Neural Networks Part 8: Image Classification with Convolutional Neural Networks (CNNs)]
[Introduction to Convolutional Neural Network (CNN) using Tensorflow | by Govinda Dumane | Towards Data Science]

To image processing methods:
CNN, SIFT, SURF,

1. CNN

Motivation

The main idea for using CNN is Image classification (also image descriptor or transfert learning)
For example lets have 2 images X, and O, can be represented as a matrix we're value 1 is black and value 0 is white. How can we differentiate these 2 images?
The size of the image is 6x6 pixels,



What if we use neural networks as a function for image classification?

- Huge number of weight to learn (nb of inputs x nb neuros)
- Back to the example, we can translate the image to a single column of 36 input nodes(flattering), each node has a neuron , it increases so fast, not worth it.
- Also can't differentiate shifted (moved) images with neural networks

Convolutional network do 3 things to make image classification practical:

- Reduce the number of input node
- Tolerate small shifts where the pixels are in the image
- Take advantage of the correlations that we observer in complex images

Convolution layer (Create a feature map with a filter)

A convolution is used instead of matrix multiplication in at least one layer of the CNN. Convolutions take two functions and return a function, do the feature map + bias

Learning the convolution filters

- Convolution: sum of weighted entries
- 1 convolution filter = 1 neuron
- The weights are the filter coefficient

Sharing weight with different neurons

- The same convolution is applied to every pixel in the image
- Less weight to learn!
- Ex: one 3x3 filter: 10 coefficients (9 for filters (3x3) + 1 bias)

In keras: We use Conv2D instead of Dense

- Parameters: number of filters, size of filters (usually k*k, 3x3), stride
- Input: size of image(an image 6x6 pixels) and number of channels (3 if RGB, 2 black and white)
- Output : tensor of dimension number of filters, (new) dim of images
- Conv1D and Conv3D also exist

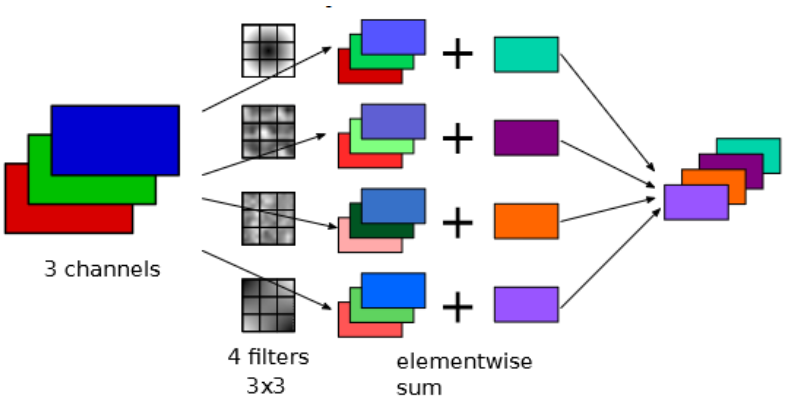
Example on dimension for a Conv2D layer

Parameters:

- Input: tensor of dimension w * h * ch (size image * channels)
- Filters: nb filters, dimension k * k (for example can be 3x3)
- Output: tensor of dimension w' * h' * nb (we're looking for a yes or type of dog)

Questions. *How did the number of channel ch dimensions vanished?*

- For each filter, element wise addition of the result of the convolution of each channel by this filter.

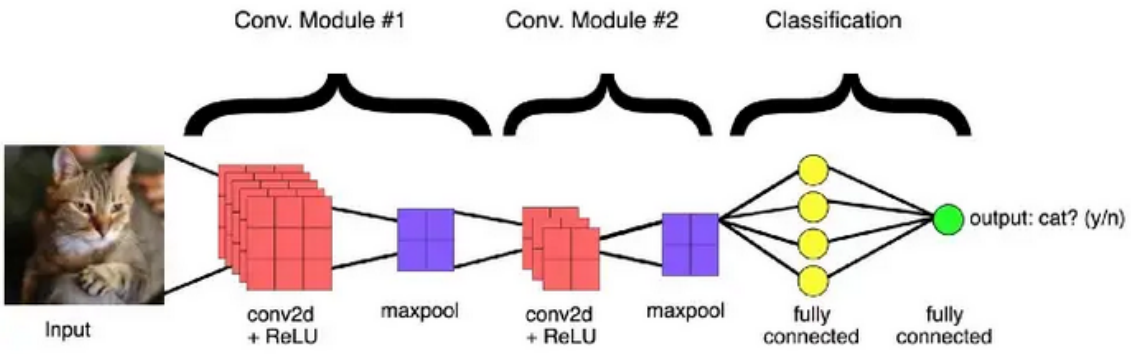


- Number of parameters to learn
    - Each filter: 3x3x(number of input channels) + 1 ,(number of channel is 3 cause is RGB)
    - 4 filters
    - Thus -> 4x(3x3x3+1) = 112 parameters to learn

Convolutional Neural Network

A convolutional neural network is a specific kind of neural network with multiple layers. It processes data that has a grid-like arrangement then extracts important features, discover the pattern in an image.

A big difference between a CNN and a regular neural network is that CNNs use convolutions to handle the maths behind the scenes. A convolution is used instead of matrix multiplication in at least one layer of the CNN. Convolutions take two functions and return a function.



The first thing a convolutional Neural network does is apply a filter to the Input Image (smaller square of 3x3 for example), trained by backpropagation. We overlay the filter onto the image and then we apply a dot product.

By computing the dot product between the input and the filter,then add a bias. We can say that the Filter is Convolved with the input (That why is called Convolutional neural network)

With that values we create a Feature map (another matrix by the size of the results, that case we have a 4x4 feature map by an image of 6x6)
Now we run the feature Map through a ReLU activation function.

Max Pooling: other filter,not overlapping itself like a convolder filter. When we select the maximum value in each region, we're applying max pooling, selects the spots where the filter did the best job matching the input image

Alternative we can calculate the average value for each region Average/Mean Pooling

Now with the input, that is 2x2 we can apply a normal neural network, that 4 values are our inputs, we can use 1 hidden layer (ReLu activation function) and the output is If the image is a X or a O.

And two hidden fully connected layers at the output
1º convolutional + bias, applies a filter (Con2D + activation function)
2º Maxpooling

- In practice we used : 1 conv2D layer + 1 pooling + 1 dense layer.

2 hidden fully connected layers at the output (the function of classification)

Activation functions

Can be Sigmoid, ReLU(rectified Linear Unit), leaky Relu and softmax (transforms output values into values that can be interpreted as probabilities)

Softmax layers

For the output value, we can use softmax to be easier to interpreted
Raw outputs in a neural network are not values that are we looking for, for example 1 or 0
Argmax = set the largest value to 1 and all the others to 0
Ex = out1 = 0.65 / out2 = 1.2 / out3=-1.4, so out2 is 1 and the rest is 0

But we can't use to back propagation due to his limitation in neural networks, so we use softmax
What it does = apply a formula, then the values are always between 0 and 1 , there are not problem then, an the total is 1, depend of the weights and biases

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Why ReLU

Solve the problem of vanishing gradient (can be solve also with residual networks or batch normalisation)

Pooling layers

Comes after a convolutional layer, it reduces the size of data, maxpooling or mean pooling

- Padding ('VALID' means discard borders, 'SAME' zero padding)
- default : 2x2 with stride 2, stride (factor of reducing data "downsampling")

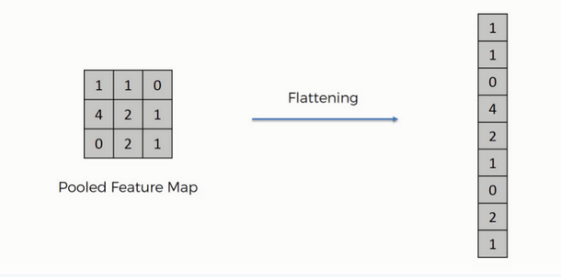
Why 3 channels ?
Because Images are in RGB.

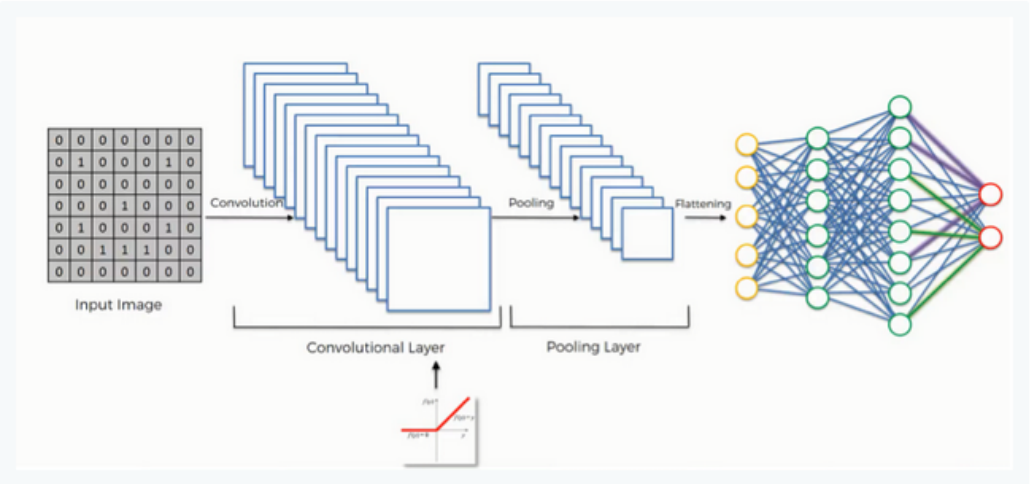
What is the batch?
The batch size defines the number of samples that will be propagated through the network
The batch size is the number of samples that are passed to the network at once.
An epoch is one single pass over the entire training set to the network. The batch size and an epoch are not the same thing.
1000 images of dogs that we want to train our network in order to identify different breeds of dogs. Now, let's say we specify our batch size to be 10. This means that 10 images of dogs will be passed as a group, or as a batch, at one time to the network.
Given that a single epoch is one single pass of all data through the network, it will take 100 batches to make up the full epoch. We have 1000 images divided by a batch size of 10, witch equals 100 total batches
Batches in epoch = training set size / batch_size

Convolutional Neural Networks are primarily used for binary and multi-class, multi-label classification. There are some p*re-trained models* like *Inception, VGG16, VGG19, mobilenet*, etc

Steps

- Input Image
- Convolution Operation (create a feature map) + ReLU Convolutional layer
- MaxPooling (mean, max, sum/global) Pooling layer
- Flattening (we do step 1 and 2 until we can have a handle data for the input)





ENSEMBLE LEARNING

[Ensemble Learning: Stacking, Blending & Voting | by Fernando López | Towards Data Science]

What is ensemble learning?
Meta-algorithm combining different learners. Can be homogeneous (use like 10 decision trees) or heterogeneous (using SVM, decision tree and logistic regression for a classification problem)
Combining the results of multiple models, ex: decision trees

Different Methods:

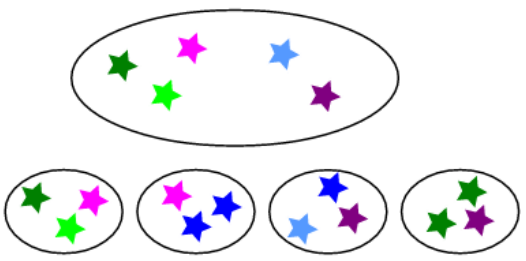
- Bagging
  - Weak learners learned independently
  - voting(classification) or averaging (regression)
  - Bootstrapping - features sampling - Random Forest
- Boosting
  - Weak learners learned sequentially
  - Focus on erroneous samples
- Stacking
  - Weak learners learned independently
  - Meta-model built on top of the output of the leak learners

1. Bagging

Bagging: Combining the result of multiple models (various decision trees)

- But we need different models
- For regression: average the different results
- For classification: vote for the classes

Bootstrapping: repeatedly selects a subset with replacement of the training set.



Random forest

- Bootstrapping of the m data
- Bootstrapping on the n features

2. Boosting algorithm

Is a supervised learning, for binary classification(separation between data corresponding to some criterions and data not corresponding).

Main idea

The main idea is to build a strong learner from weak learners (green correspond to fir tree, vertical shapes correspond to fir tree …)

Example of sport bet and heuristic

- We ask experts simple rules that are true more than 50%
- We focus on examples for which a rule fails and we ask other rules to experts
- And loop the process

“Is it possible to make as good as possible a weak learner” - YES boosting
We need to choose the 3 training subsamples with respect to its performance

Probabilistic boosting

3 main ideas

1. A set of specialised experts and ask them to vote to take a decision
2. Adaptive weighting of votes by multiplicative update
3. Modifying example distribution to train each expert, increasing the weights iteratively of examples misclassified and previous iteration

Adaboost (Adaptive boosting) -> minimisation of exponential loss
Is fast, easy, uses 1 parameter, but depends on the data , noise sensitive and could fail.

3. Stacking

- For each sample i
    - For each learner
        - Compute the prediction of the sample by the learner
    - Arrange them in a vector xi
- Learn a new machine learning algorithm
    - With the dataset xi
    - Could be a logistic regression or any other ml algorithm

SKLEARN

For the DS, I can give you a code and ask you to understand it. For example, I can give you the description in keras of a neural network and ask you to draw it, to know what is the task (classification or regression) ....
I can also give you some hole code (we need to write in it).
If some documentation is needed, it will be in the subject.
You should know the functions fit/predict/compile/add, the objects Model, Dense, Conv2D, Pooling, Flatten …

In the context of data science (DS), I can provide you with code and ask you to analyze or interpret it. For example, I might share a Keras-based neural network description and ask you to visualize its architecture or identify whether the task is classification or regression. Additionally, I may present incomplete code where you'll need to fill in the gaps.

If any documentation is required, it will be provided as part of the assignment. You should be familiar with common Keras functions such as fit(), predict(), compile(), and add(), as well as key components like Model, Dense, Conv2D, Pooling, and Flatten.

Introduction

Load the dataset

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

Useful data

```
print("there are", len(y), "data. ")
print("Type of data: ", type(X[0][0]))
print("Dimension of data ", X.shape[1])
print("Type of label: ", type(y[0]))
```

Load the dataset MNIST

```
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
# Split the dataset into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Plot a data

```
import matplotlib.cm as cm

colors = ['b', 'r']

col = [colors[c] for c in y_train]

plt.figure(figsize=(10,6))
plt.scatter(X_train[:,2], X_train[:,3], c=col, marker="x")
```

Confusion Matrix

```
ConfusionMatrixDisplay.from_predictions(y,ypred)
```

Score

```
print("train accuracy of a classification",
accuracy_score(y_train,ypred_train))
```

Decision trees

Function predict:

Once the model is trained, we can use predict() to predict an output value, on the basis of input values.

We call it from an existing instance of a machine learning model that's already been trained

```
model_instance.predict(x_text);
```

Model_instance: (Linear regression, SVM, decision trees ,...)

Predict: the predict method

X_test: the features of the test data (input needs to be 2D numpy format)

Train data

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4)
```

Basic tree algorithm in python and plotA

```
myTree = tree.DecisionTreeClassifier()
myTree.fit(X_train,y_train)
ypred = myTree.predict(X_test)
tree.plot_tree(myTree)

print("Accuracy of a test classification", accuracy_score(y_test,ypred))


ConfusionMatrixDisplay.from_predictions(y_test,ypred)
```

Dimension Reduction

Fit method

"Fit" a machine learning model in Python.

The scikit learn 'fit' method is one of those tools. The 'fit' method trains the algorithm on the training data, after the model is initialised. Uses the training data X as input to train the machine learning model .

PCA:

n_components : Number of components to keep, number of dimensions.

If we set it to 2: we transform to 2 dimensions.

```
#X dimension: (1797, 64) is 64
X,y = datasets.load_digits(return_X_y=True)
pca = PCA(n_components = 29)

pca.fit(X)

X_ = pca.transform(X)
```

tSNE

```
tsne = manifold.TSNE(n_components=2, init='pca', random_state=0)
X_tsne = tsne.fit_transform(X)
```

Clustering algorithms

k-means

```
from sklearn.cluster import KMeans
nbClusters = 3
#Algorithm basic kmena
km = KMeans(n_clusters=nbClusters, max_iter=10, n_init=1,
init='random')
#learning and computing the result:
y_pred = km.fit_predict(X)

#Basic Plot
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
c=km.cluster_centers_
```

K-menoids

```
km1 = KMedoids(n_clusters=nbClusters, max_iter=10, init='random')
```

Agglomerative clustering

```
km1 = KMedoids(n_clusters=nbClusters, max_iter=10, init='random')
```

DBSCAN

```
db = DBSCAN(eps=0.5, min_samples=10)
y_pred=db.fit_predict(X)
```

Silhouettes Coefficient

```
silhouetteScore = metrics.silhouette_score(X, y_pred)
print("Silhouette Coefficient: %.2f" %silhouetteScore)
```

Linear regression and logistic regression

Linear Regression

```
lr = LinearRegression()

# learning
lr.fit(X_train,y_train)

# prediction
y_pred = lr.predict(X_test)
Logistic REgression

# score
score_train = lr.score(X_train, y_train)
print("train score =", score_train)
```

Logistic Regression

```
lg = LogisticRegression(n_jobs = -1, max_iter=5000,
random_state=0)

# learning
lg.fit(A_train,b_train)
```

```
# prediction
b_test_pred = lg.predict(A_test)

print(classification_report(b_test, b_test_pred))
```

SVM

SVM example

```
classification = svm.SVC(kernel = 'linear', C = 1)
classification.fit(X_train, y_train)
y_pred_test = classification.predict(X_test)

y_pred_train = classification.predict(X_train)

accuracy = metrics.accuracy_score(y_train, y_pred_train)
print("Acurracy Test: ", format(accuracy))
ConfusionMatrixDisplay.from_predictions(y_train, y_pred_train)
```

Binary and multiclassification

```
binary = svm.SVC(kernel = 'linear', C = 1)
binary.fit(X_train, y_train)

multiclassification = svm.LinearSVC(multi_class="ovr",
max_iter=1000000)
multiclassification.fit(X_train,y_train)
```

Neural networks as Approximation

Basic neural network with sequential

```
# We have 3 modes - Sequentail : the easiest but limited
# Funtional API: for creating more complex networks
# Model Subclasing : new for tensorflow 2.0
model = Sequential()

#Input
model.add(Dense(64, input_dim=1,activation='relu')) # 64 nodes

# hidden layer
model.add(Dense(64,activation='relu')) # 64 nodes with activation
function
model.add(Dense(128,activation='relu')) # 128 nodes
model.add(Dense(256,activation='relu')) # 256 nodes
model.add(Dense(512,activation='relu')) # 512 nodes
model.add(Dense(256,activation='relu')) # 256 nodes
model.add(Dense(128,activation='relu')) # 128 nodes
model.add(Dense(64,activation='relu')) # 64 nodes

#Output
model.add(Dense(1)) # LAST ONE NEEDS TO BE 1 withour activation -
linear
# dense 1 for sumarize the input or n depend on the output

model.compile(optimizer='rmsprop',loss='mse')

#learning
model.fit(xTrain, yTrain, epochs=80, batch_size=128)
```

Neural Networks as Classification

This model has a single hidden layer with 4 nodes using the sigmoid activation function. has 32 nodes so it has 32 biases output = 4*10 = 40 weights

Train our network with fit: only 20 epoch and batch size 128, so the epoch contais 60000/128 iterations

```
nbClasses=10
image_size = 28*28
model = Sequential()

# input as the size of the data, now its a image of 28*28
model.add(Dense(1, input_dim =image_size, activation='sigmoid'))
#input

model.add(Dense(4,activation='relu')) #hidden layers

# output is the number of clases
model.add(Dense(nbClasses, activation = 'softmax'))  #output

model.compile(optimizer='rmsprop',loss='categorical_crossentropy',
metrics=['accuracy'])

# normal fit
history = model.fit(xTrain, yTrain, epochs=20, batch_size=128)

# fit with call back callback to stop before
ourCallback = EarlyStopping(monitor='val_accuracy',
min_delta=0.0001, patience=20, verbose=0, mode='auto',
baseline=None, restore_best_weights=True)


model.fit(xTrain, yTrain, epochs=2000, batch_size=128,
validation_split=0.2, callbacks=[ourCallback])
```

CNN

```
# classes = ['mucca', 'elefante'] tipos de imagenes
nbClasses = 2

model = Sequential()
model.add(Conv2D(16,(3,3),padding='same',activation='relu'))
#convultional layer, apply a filter + relu

model.add(MaxPooling2D(pool_size=(2, 2),padding='same')) # pooling
# a filter but a max value + not overlap

model.add(Flatten()) # flatten is important for the dimension
# convert the image matrix to a vector posible to make output

model.add(Dense(nbClasses, activation='softmax')) # dense layer,
hidden layer, I apply a neural network with softmax, 2 neurons.

# En vez de usar fit uso build
input_shape=(None,224,224,3) #size of image 224x224 and nb
channels 3 rgb
model.build(input_shape)

model.summary()

#we need to define the loss function for the training, the
optimisation method (RMSprop) and the accuracy as a metric
model.compile(optimizer='rmsprop',loss='categorical_crossentropy',
metrics=['accuracy'])

# fit, train the algorithm
history = model.fit(xTrain, yTrain, epochs=20, batch_size=128)
score = model.evaluate(xTrain,yTrain)
```

REVISION

Questions

Introduction

- *Difference between supervised and unsupervised learning? Give examples*
  Unsupervised learning just needs data and supervised learning needs data and labels.
  Unsupervised (dimension reduction or clustering algorithm)
  Supervised (classification and regression algorithm)

- *How are data generally separated?*
  Can be separate by train and data set (20% 80 %) or train, validation and test set (20 20 60)

- *Difference in classification/regression?*
  Regression focus on prediction and classification divides the labelled data into the corresponding groups.

Decision Trees

- *How do we measure impurity?*
  We use Gini Index or Log loss/Entropy

- *What is pruning? Why do we use it?*
  During the learning process we use it in order to match the constraints (nº samples), and after the learning in order to reduce overfitting.

Dimension Reduction

- *explain in a few sentences what PCA is*
  Is an unsupervised linear dimensionality reduction and data visualisation technique for very high dimensional data. Main idea is reduce the dimensionality of the data that is highly correlated using eigenvector to preserve the data and create a new structure

- *Explain in a few sentences what t-SNE is*
  It is an unsupervised linear dimensionality reduction and data visualisation technique. The main idea is to transform from a higher dimension to a lower dimension trying to preserve the local structure, from its similarity. It involver hyperparameters

- *Why choose one rather than the other?*

  why Gaussian distribution

  (original space) The case when we want to capture close elements and do no care of distant elements

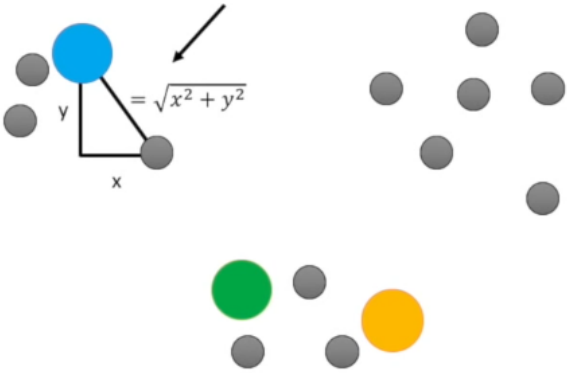
  why Student's t-distribution

  (embedded space) the case when the samples are initially randomly projected . We need to be able to capture them.

Clustering

- *how to measure the similarity between 2 data? Give examples*

  To calculate the similarities between data we calculate their distances, if their distances are low it equals that these data are highly similar. We calculate it with the Euclidean distance (Pitagoras) 2 data, generally the Minkowski distance(depend on the dimension) .



- *give examples of approaches for clustering*

  Partitioning (K-means, K-medoids), Hierarchical (DIANA), Density-base(DBSCAN). Grid-based, model-based, spectral clustering , etc.

- *Describe k-means*

  given k, number of clusters that we want to find:

  1. We choose a centroid, each labelled as k
  2. For each object we calculate the distance between them to the other centroids
  3. Assign the data to the closest cluster
  4. Do it for all the data

- *what are the 2 approaches used with Hierarchical clustering*
  - bottom up approach , agglomerative
  - top down approach (DIANA = divide analysis)

- *How we evaluate the clustering*
  We use ground truth known(like ARI) and ground truth not known (like silhouette) we use silhouettes coefficients at the end

- *slide 45 to remember*
  - For numeric data, standardisation is really important
  - Many algorithms exist with different advantages and drawbacks
  - Parameters of algorithm are important
  - Evaluation metrics are useful but should be considered carefully.

Linear Regression and Logistic Regression

SVM

- *What SVM stands for, is supervised or unsupervised?*
  Is supervised. Stands for Support vector machine

- *What are the support vector*
  There are the points closest to the hyperplane. In SVM we look for Maximum Marginal Hyperplane MMH reducing the margin. Can agave a soft margin (an loss error) that is with the variable C

- What is soft margin
  When we allow misclassification, the distance between the observations and the threshold is called a Soft Margin Its with the variable C, higher the value, smaller the misclassification.

Neural networks / CNN

- *I have 500 images of size 28 by 28 belonging to 5 classes, give the size of the input and output of the MLP (enural network)*
  Input is 28*28 and the output is 5.

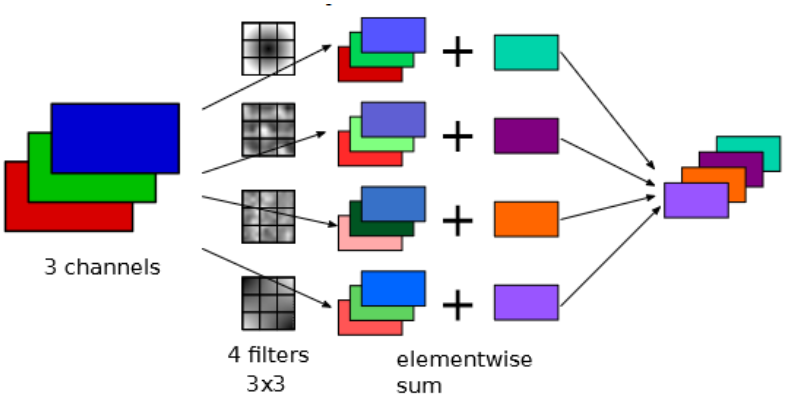
- *Example of activation function*
  Relu(recta pa arriba), Softmax(la S otra), Sigmoid(La S), softplus(s pa arriba)

- *What loss for the classification? Which loss for the regression?*

- *What is the softmax used for?*
  For the output value, we can use softmax to be easier to interpreted

- *What is pooling?*
  Comes after a convolutional layer, it reduces the size of data

- *explain this shema*



  In this schema, there is a CNN with an image RGB, it applies the filters 3x3 and then depending on the values of the RBG it will interpret a different color.For each filter, element wise addition of the result of the convolution of each channel by this filter. Convolutional layer (filer + sum then the activation function)

-

- *How can we avoid overtraining/overfitting?*

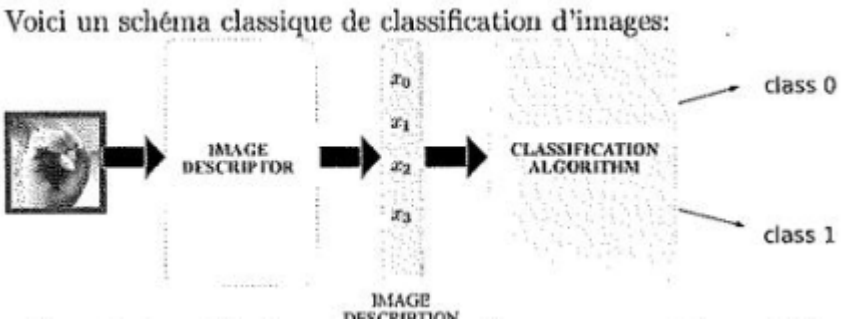
- *What is an epoch and a batch?*
  The batch size defines the number of samples that will be propagated through the network.
  The batch size is the number of samples that are passed to the network at once.
  An epoch is one single pass over the entire training set to the network. The batch size and an epoch are not the same thing. N batches = n° iteration/batch

Ensemble learning

- *What is ensemble learning?*
  Meta-algorithm combining different learners.

- Different methods
  Bagging boosting and stacking

Exam Questions



Question 1: select the algorithms that can be used in the "Image descriptor" box

- What is happening in the image description, we're doing a flattering image, we're depurating the outputs to be possible to apply a classification algorithm
- We usually apply a flattening layer in the image descriptions
- We don't use boosting, or MLP or SVM for regression (thats other thing) OR ann
- We use autoencoder (bottleneck), SIFT +BoW, HoG, CNN without the last 2 columns
    - HoG (Histogram of oriented gradients), a feature descriptor
    - SIFT (Scale Invariant and feature Transform)
    - SIFT + Bow (a variant of SIFT)
    - Autoencoder (type of neural network that can be used to learn a compressed representation of raw data "image description")
    - We have study to apply CNN a convolutional neural network to descript the image, the newest.
    -

A feature descriptor is used in computed vision and image processing of gradient orientation in the localised portion of an image.

- Linear Regression
- SVM
- ANN
- Autoencoder
- CNN without the last 2 columns
- HoG
- SIFT+BoW
- MLP

Question 1.5: Select the algorithms that can be used in the "classification algorithm"

- Boosting
- MLP
- SVM
- Régression logistique

We cannot use the regression linear because it solves regression problems, not classifications.

Question 2: We recall the definitions of specificity s and recall r (or sensitivity) .

- *Sensitivity:* Tell us what % of patient with heart disease were correctly identified (+ proportion corrected)

$$S = \frac{TP}{TP + FN}$$

- *Specificity*: Tell us what percentage of patient without heart disease were correctly identified (- proportion corrected)

$$SP = \frac{TN}{TN + FP}$$

A new rapid covid test leads to a specificity of 1 and a recall of 0.9 What can you say?

- Specify says to us the amount % of corrected positive result of our algorithm (% of TP) and the % of sensitivity tell us the % amount of correct negative result (% of TN). If its 1 the sensitivity all the tet covid that vade been positive are true and the 0.9 of specificity tell us that the 90% of the test that says no covid are true
- In conclusion

- We cannot say anything
- all people with covid-19 will test positive
- people without covid-19 will have a negative test

Question 3: confusion matrix obtained during a TP rendering:

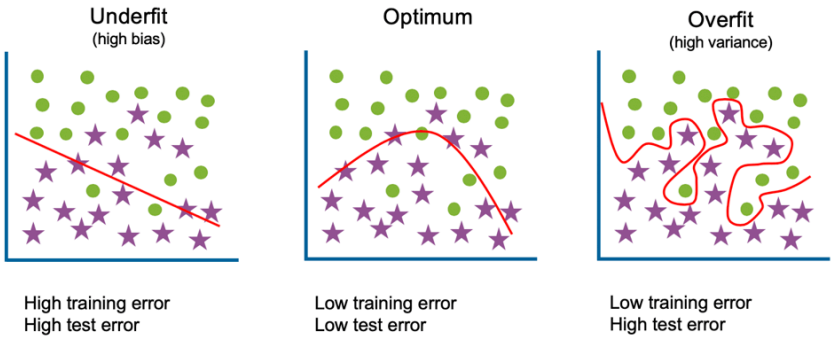
	E1	E2	E3	E4
E1	90	3	3	4
E2	1	99	0	0
E3	0	0	50	50
E4	0	0	60	40

We can look at the confusion matrix , it has 4 classes E1, E2 , E3, E4. We focus on the TP and TN. We look that E1 and E2 have a good relation because has a High positives and low negatives. On the other hand E3 and E4 has a high negative values (50 and 60), equal a great confusion

- the classes are unbalanced: it would be necessary to modify the parameter concerning the balancing of the classes in the algorithm used
- It's all very confusing
- There are mistakes: learning has failed (or is not finished)

- the data of the classes E1 and E2 are classes with good performances. On the other hand, there is a great confusion between the classes E3 and E4
- Nothing can be concluded from this

Question 4: We speak under-fitting(sous-apprentissage) when the learned model



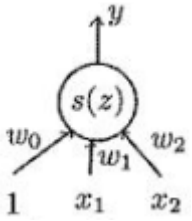
- has poor performance on training and test data
- has too good performance on test data
- has a much better performance on training data than on test data
- Is too complicate
- is not complex enough

Overfitting is the opposite, has a much better performance on training data than on test data cause is too complex

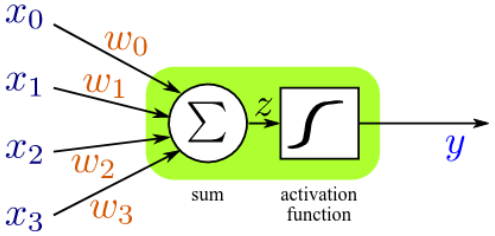
Question 4.4: We speak over-fitting when the learned model

- is too complex
- performs much better on training data than on test data.
- is not complex enough
- performs poorly on both training and test data
- has too good performances on test data

Question 6: Look the following artificial neuron, X = (x0,x1,x2) and w = (w0 w1 w2) the output y is expressed as:
In the activation function "s" we do a sum of the weight*input, so ints the first one



- $y = s(xw^T)$
- $y = xw^T$
- $y = s(x)w^T$



$$y = s(z) = s\left(\sum_{i=0}^{n} w_i x_i\right)$$

Question 5: The supervises learning (label + data)

- requires human intervention for each iteration
- is an autonomous algorithm, unlike unsupervised learning
- requires that each learning data has a label
- requires only learning data (NO)

Question 5.5: The unsupervised learning.

- requires that each training data has a label.
- requires only training data.
- requires a hierarchy of classifiers.
- requires a self-organisation capacity of the weak classifiers in cascade.

Question 7: back propagation:

- Indicates that the gradient descent propagates from the output to the input
- moves the suppressed vectors back from the array
- is an old method that was fashionable in the early days of artificial intelligence but is no longer used today
- allows learning the weights of the neural network

Question 8 : We want to estimate the price of an apartment according to its location (longitude, latitude), its surface, the floor and its exposure (angle.) A customer wishes to estimate the price of 4 apartments. How to choose the topology of this network?
In between the number of neurons is:, ESTIMAR EL PRECIO ES REGRESSION
To this kind of problem we need to understand what is asking

- What is the data that we have?: we have latitude, longitude, surface, floor, angle
- If we have a photo of 28*28: the input is 28*28 = 784.
- 4 appartements is not useful in a Neural network WE DONT USE IT

- 1
- 4
- 5
- 20 (4*5)

Question 9: The output the number of neurons is

- What is the problem?: estimate the price of a apartment 1 data

- 5
- 4
- 1
- 20

Question 10: Why do CNN-type networks perform well with images?
What it does is convolution, obtaining a good description of the image to apply to a neural network.

- They take fixed intervals in input and the pixel values are always between 0 and 255
- they are efficient with all types of data because they are deep
- the convolution layers allow to obtain a very good description of the images
- there are many images used on the information channel CNN

Question 11: what are the different possible natures for the layers of a neural network

- convulsion
- boosting
- backpropagation
- Convolution Conv2D
- Pooling MaxPooling
- dense

Question 13: what is the main purpose of an autoencoder (is a image description)
Autoencoder: A machine learning model that learns a lower-dimensional encoding of data", is like a maxpooling
Like an adaptation of PCA on neural networks.

- coding of a neural network by itself
- reduce the size of the data
- neural network for autonomous driving of cars
- encoding the weights of a neural network

Question 14:

```
Model = Sequential()
model.add(Conv2D(16,(5,5), strides=(1,1), padding = 'same',
activation='relu'))
model.add(MaxPooling2D(pool_size(2,2),strides=None,padding='same'))
```

We remind the following information found in the tensorflow documentation

pool_size: integer or tuple of 2 integers, window size over which to take the maximum. (2, 2) will take the max value over a 2x2 pooling window.

strides: Integer, tuple of 2 integers, or None. Strides values. For pooling : Specifies how far the pooling window moves for each pooling step. If None, it will default to pool_size. For convolution: An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width.

padding: One of "valid" or "same". "valid" means no padding. "same" results in padding evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.

what will be the output dimension of this network if the input is 10 images of size 224*224 in color RGB(10,224,224,3)
We're adding a Convolution and then a max pooling, the max pooling will determine the output of the image, we look that is a pool o size of 2*2, so if we have an image of 224*224 it will reduce to its half: 112*112 will be the image
*Why 16 the RGB? 3*5 + 1 = 15 + 1 = 16*

- (10,110,110,16)
- (10,22,22,16)
- (10,112,112,16)
- (10,224,224,26)

Question 14: Now we want to pass in this same network (without modifying it) 10 new images of half size in width and height (10,112,112,3) What will be the dimension in output

- it is possible if we modify the network
- You will have to enlarge these images again because the input size is fixed. We will find the dimension of the previous question
- (10,56,56,16)

We're still applying a maxpooling of 2x2 so we reduce the imagen to its half 112/2 = 56

Question 15: a recognition of musical instruments has been performed in concert images. This recognition is based on the Faster R-CNN algorithm. Among the musical instruments is the ukelele. You also have a good number of images of ukuleles with the name of the model. You are asked to implement an automatic recognition of ukulele models in concert images.
It's like the example of Christmas trees on the slices.
We apply a CNN algorithm, we're training a filter with back propagation to apply if it's a ukelele or not, we train with all the examples of ukuleles we have until we can have a proper filter. Then it's just apply the convolutional network based on R-CNN, flattering it

and then do a neural network with an algorithm method to determine if its a ukelele or no CNN handle variance, it's the same as the image its move if we have a good filter trained with backpropagation

Question 16: The algorithm SVM consist:

- find the minimum number of points to calculate the linear regression (or a more complex model)
- find the separation between two classes that maximises the margin MMH
- represent the data in the most compact form possible with the least amount of information

Question 17: active learning allows.
Active learning is a special case of machine learning in which a learning algorithm can interactively query a user (or some other information source) to label new data points with the desired outputs. SVM

- ask the user for the data he/she thinks should be labelled first
- to be less passive in front of the computer
- users can modify the topology of a neural network during training
- to label as few data as possible by users

Question 18: Kernels on SVM allows:
Use kernel for dimension and stuff (non-linear)

- to determine the points whose class is certain
- to have convolution layers
- learn non-linear separations
- to accelerate learning

Question 19: We have numerous data coming from 2 classes (positive and negative) that we wish to separate by a neural network (the topology is imposed).
that we want to separate by a neural network (the topology is imposed). Assuming that this is possible, how do we organise the data?

- The first one is the one that makes sense, to organise the data we cannot use all dat a for training (the learning), remember sklearn that we divided the data first into test and train to perform neural networks.

- We learn with 60% of the data (train). The stopping criterion uses 20%(validation) of the data. The performances will be calculated on the remaining 20%(test).

- We learn using all the data. We then test each of the data and rank the data according to the data according to their errors. The score is calculated on the best ranked data.
- We learn by using all the data. We then test each of the data and rank the data according to their errors. We calculate the score on the lowest ranked data.
- We put the positive data in the train and learn to recognize them. We put the negative data in the validation set to validate the performances. The test set is made of the two previous sets to test the classification.
- The data being numerous, we will take only 20% of the data for the training. We 20% for the validation and 60% for the test

Question 20: We know that we can identify a person by his typing style by measuring only three measuring only three parameters: the time of pressure on each key, the time of release as well as the flight time between two keys. We want to identify the 24 students of your group in this way (24 tipos de estudiantes, no si es un estudiante o no), using a neural network. How to choose the topology of this, de si o no = 1 neuron
network?
As input, the number of neurons is:
3 parameters to identify a person, 3 output 1 input

- 3
- 24
- 3*24=72
- It depends on the number of data

Question 21: As output the number of neuron is (deberia ser 1, por que es 1 estudiante)

- 3
- 24
- 3*24=72
- It depends on the number of data

Question 22: We try to estimate the presence of air pollution based on the response to 6 detectors (each detector is sensitive to a different pollutant and returns a numerical value). We have data on one week with an acquisition per hour as well as the information of air pollution or not for each data. We wish to estimate the presence or absence of air pollution by a neural network. How to choose the topology of this network?
As input, the number of neurons is
What is the problem?: presence of pollution or no (1 neuron)
What do we have?: 6 detectores

- 6

- 1
- 2
- 7*24*6
- 7*6

Question 23: As output is:

- 6
- 1
- 7*24
- 2*7*24

Question 24: Deep neural networks often replace sigmoid activation functions with activation functions by RELU functions. What does RELU mean?

- Rectified linear Unit
- Retro Estimation Loss Unit
- Real Error Lightly Uniform
- Regression Error or Loss Unit

Question 25: Why are RELU activations used instead of sigmoid activations?
sigmoid activations?

- Avoid gradient loss during training. (solve problem gradient descent)
- Accelerate the prediction of new data.
- Avoid overlearning.

Question 26: An adversary image is an image that:

- looks like an image of a class but is identified by classifier as being of another class, with a high degree of certainty.
- is an image whose class has not been learned.
- is the opposite image of an image in the training base.
- is part of the negatives in the binary classification.

Question 27: With which algorithm is it easiest to do active learning?

- SVM
- HOG
- MLP
- Boosting
- CNN

Question 28: For a problem of binary classification of images of dimensions (780x560), whose number of images for training is 10 images per class, what do you choose a priori as a classification algorithm (we specify that the problem is not simple because the images are all different, contrary to the are all very different, contrary to the fruit base for example):

- SVM
- MLP
- CNN