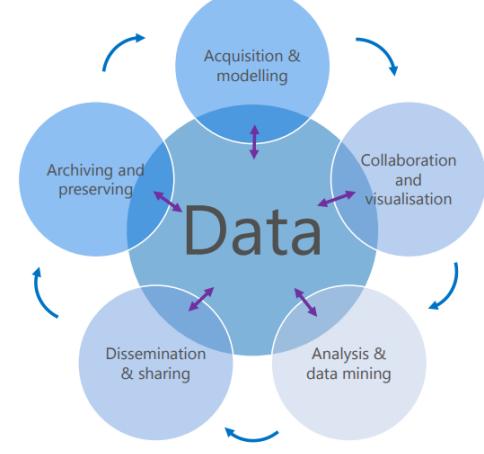
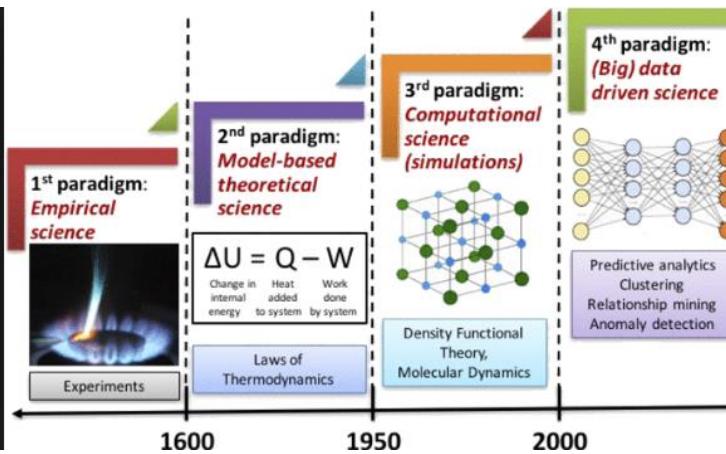
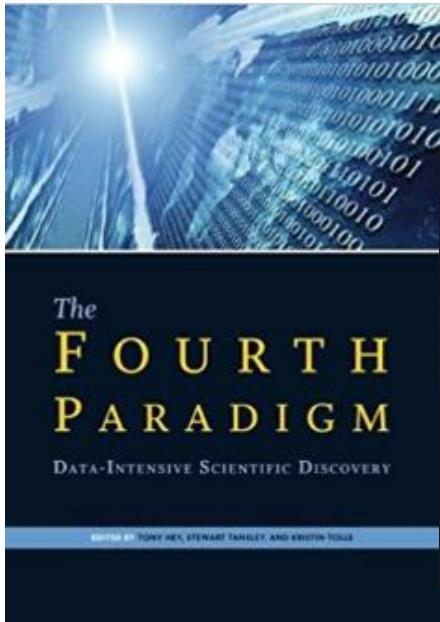


Pr. BEN LAHMAR EL Habib

What is machine learning?

- Algorithms types
- Learning techniques: an overview
- Data Preparation (Data pre-processing)

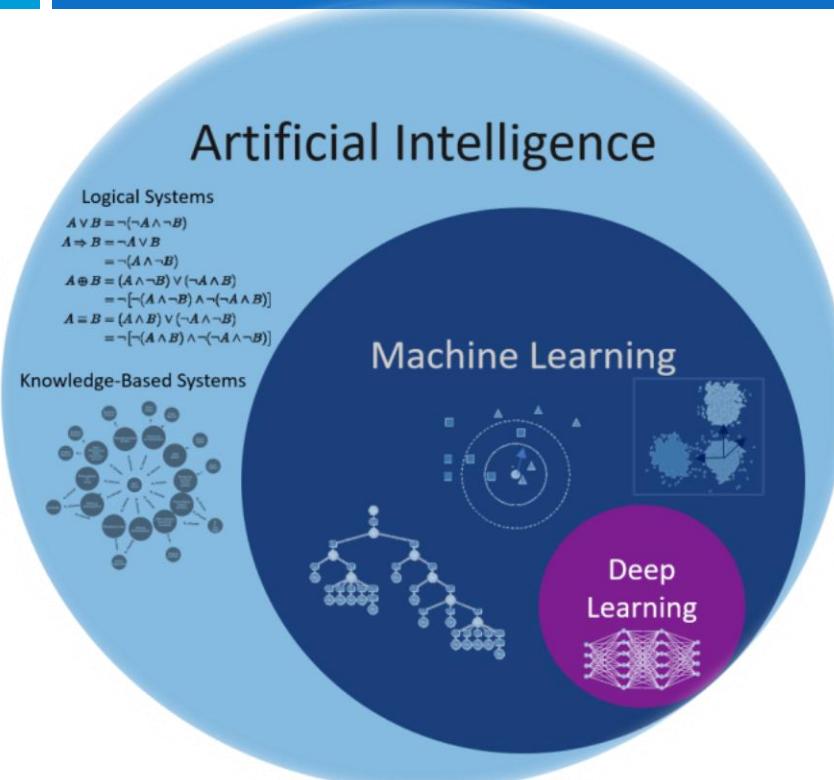
Fourth Paradigm



What We Talk About When We Talk About “Learning”

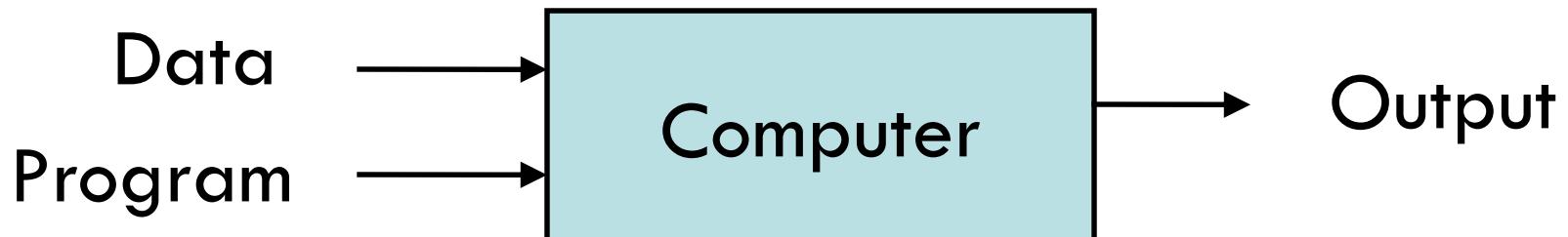
- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.
- Example in retail: Customer transactions to consumer behavior:
People who bought “Da Vinci Code” also bought “The Five People You Meet in Heaven” (www.amazon.com)
- Build a model that is *a good and useful approximation* to the data.

What is machine learning?

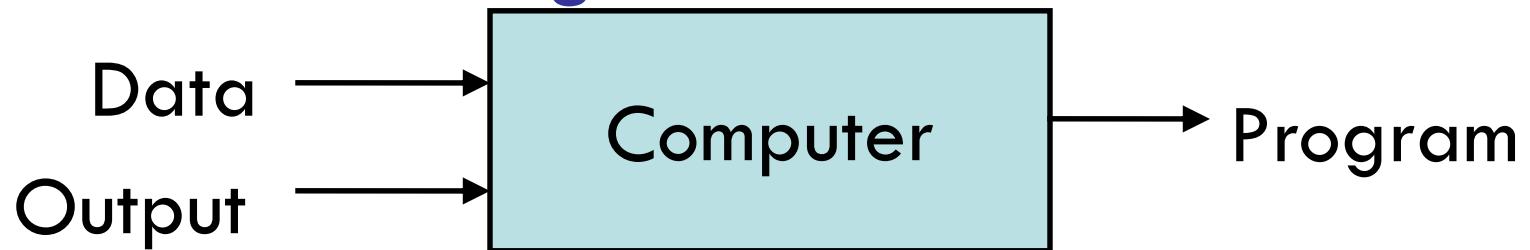


- A branch of **artificial intelligence**, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data.
- As intelligence requires knowledge, it is necessary for the computers to acquire knowledge.

Traditional Programming



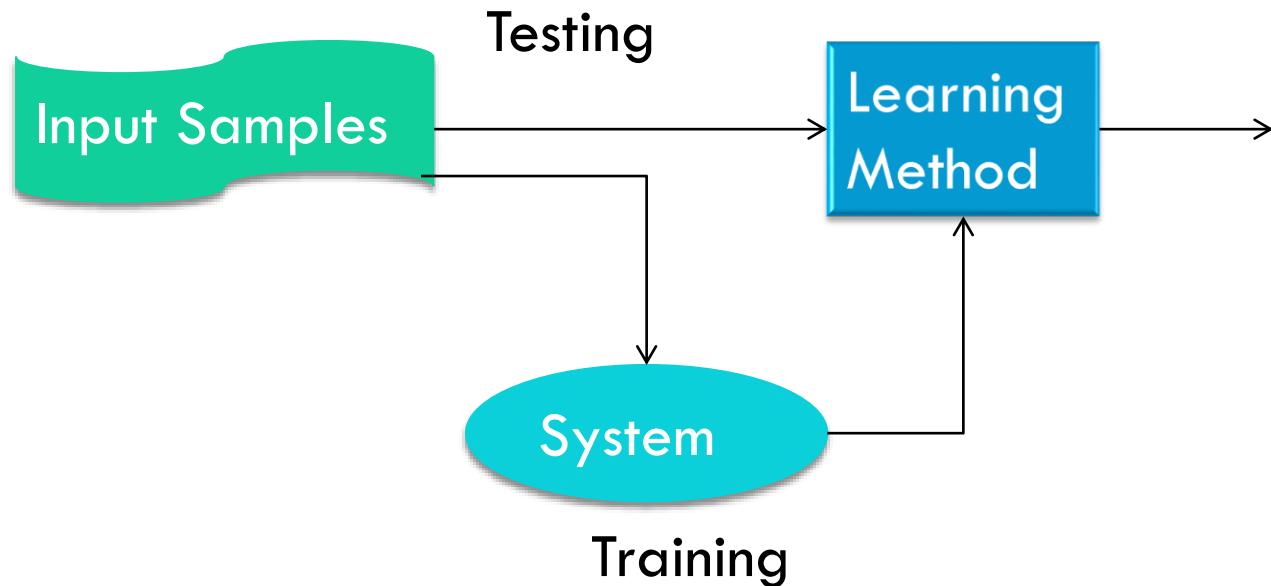
Machine Learning



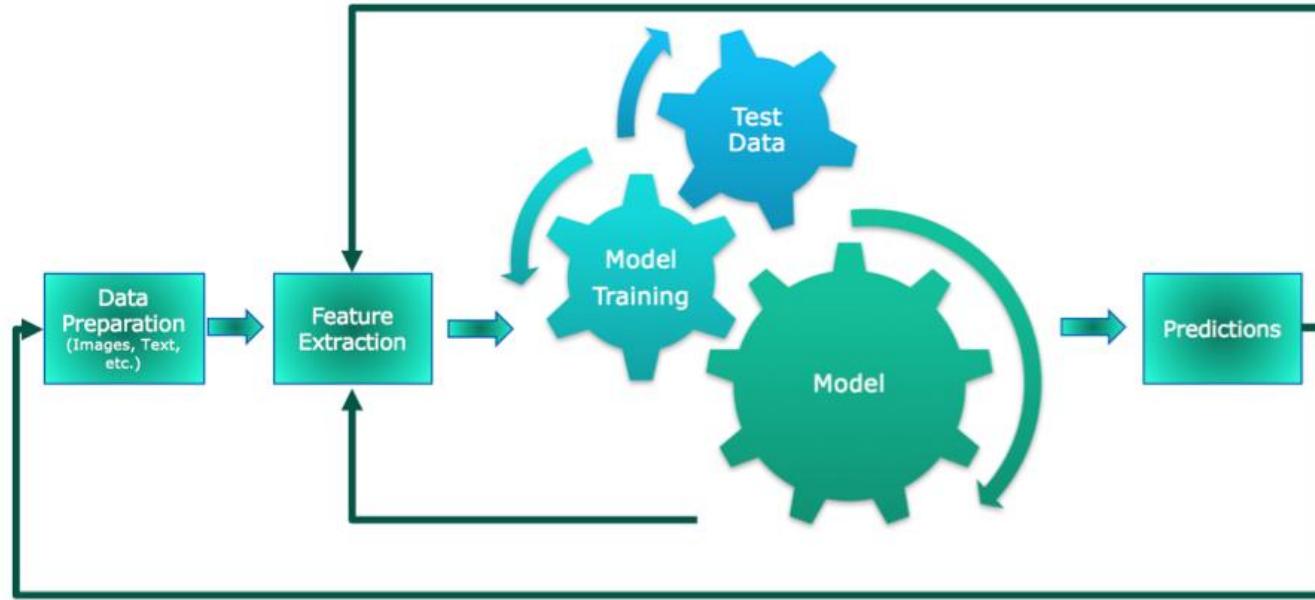
What is machine learning?

- Machine Learning
 - Study of algorithms that
 - improve their performance
 - at some task
 - with experience
- Optimize a performance criterion using example data or past experience.
- Role of Statistics: Inference from a sample
- Role of Computer science: Efficient algorithms to
 - Solve the optimization problem
 - Representing and evaluating the model for inference

Learning system model

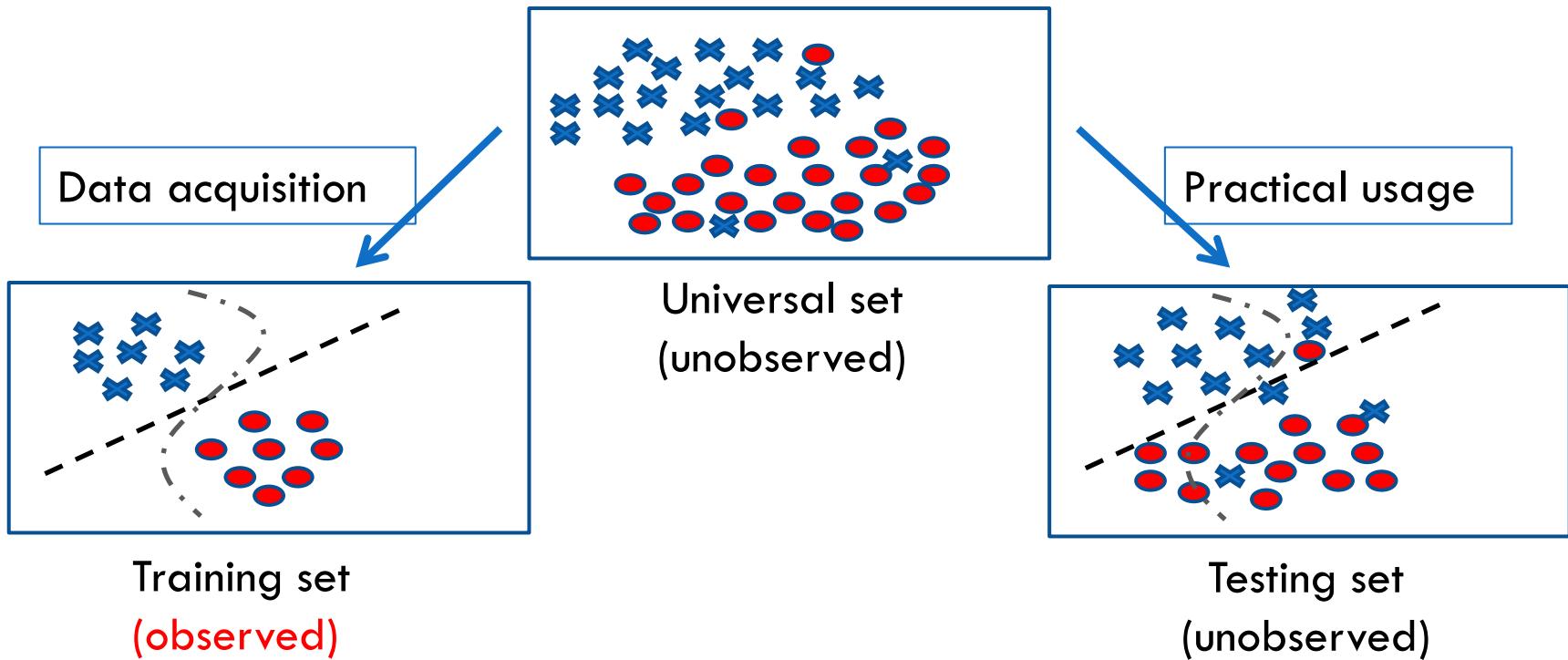


A standard Machine learning pipeline



- Feature extraction is critical for machine learning pipelines (Courtesy: Western Digital)

Training and testing

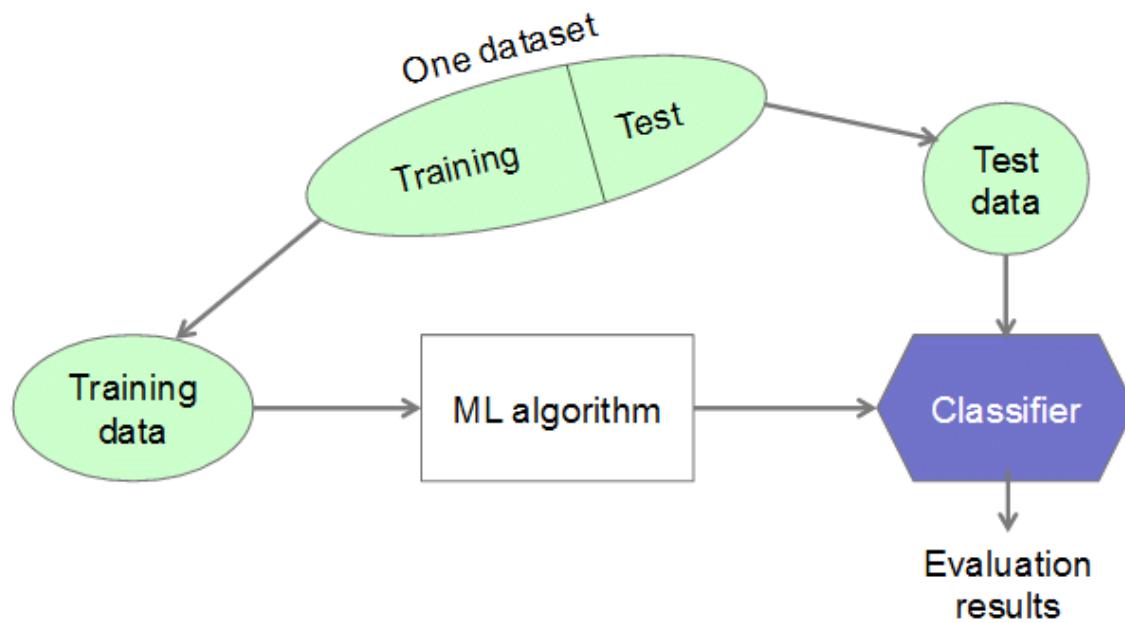


Training and testing

- Training is the process of making the system able to learn.

- No free lunch rule:
 - Training set and testing set come from the same distribution
 - Need to make some assumptions or bias

Training and testing



Performance

- There are several factors affecting the performance:
 - **Types of training** provided
 - The form and extent of any initial **background knowledge**
 - The **type of feedback** provided
 - The **learning algorithms** used
- Two important factors:
 - Modeling
 - Optimization

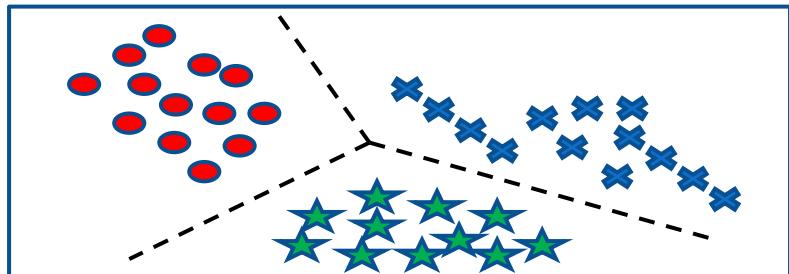
Algorithms

- The success of machine learning system also depends on the algorithms.
- The algorithms control the search to find and build the knowledge structures.
- The learning algorithms should extract useful information from training examples.

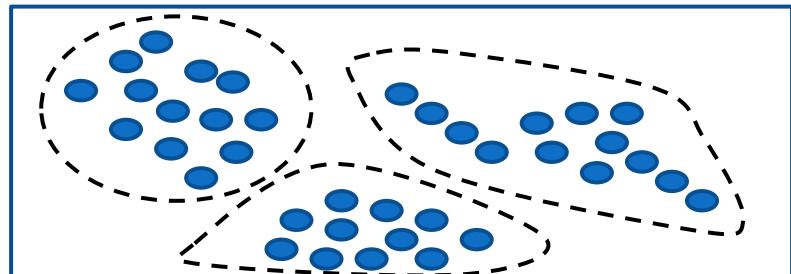
Algorithms

- **Supervised learning** ($\{x_n \in R^d, y_n \in R\}_{n=1}^N$)
 - Prediction
 - Classification (discrete labels), Regression (real values)
 - Training data includes desired outputs
- **Unsupervised learning** ($\{x_n \in R^d\}_{n=1}^N$)
 - Clustering
 - Probability distribution estimation
 - Finding association (in features)
 - Dimension reduction
 - Training data does not include desired outputs
- **Semi-supervised learning**
 - Training data includes a few desired outputs
- **Reinforcement learning**
 - Decision making (robot, chess machine)
 - Rewards from sequence of actions

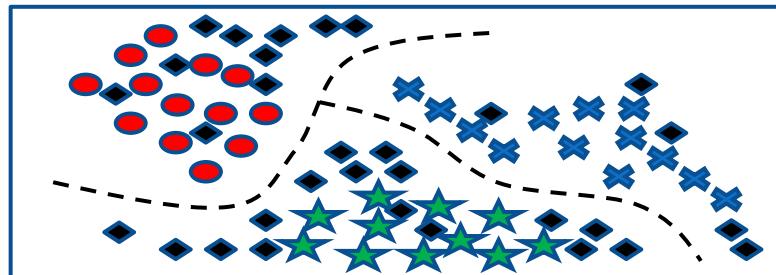
Algorithms



Supervised learning



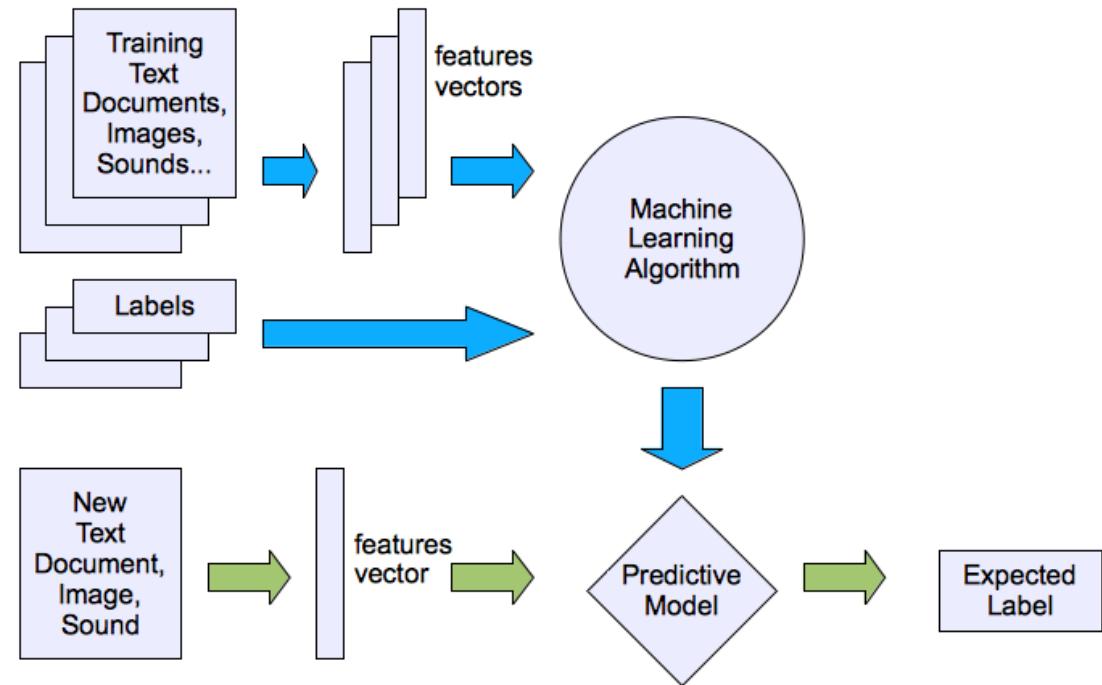
Unsupervised learning



Semi-supervised learning

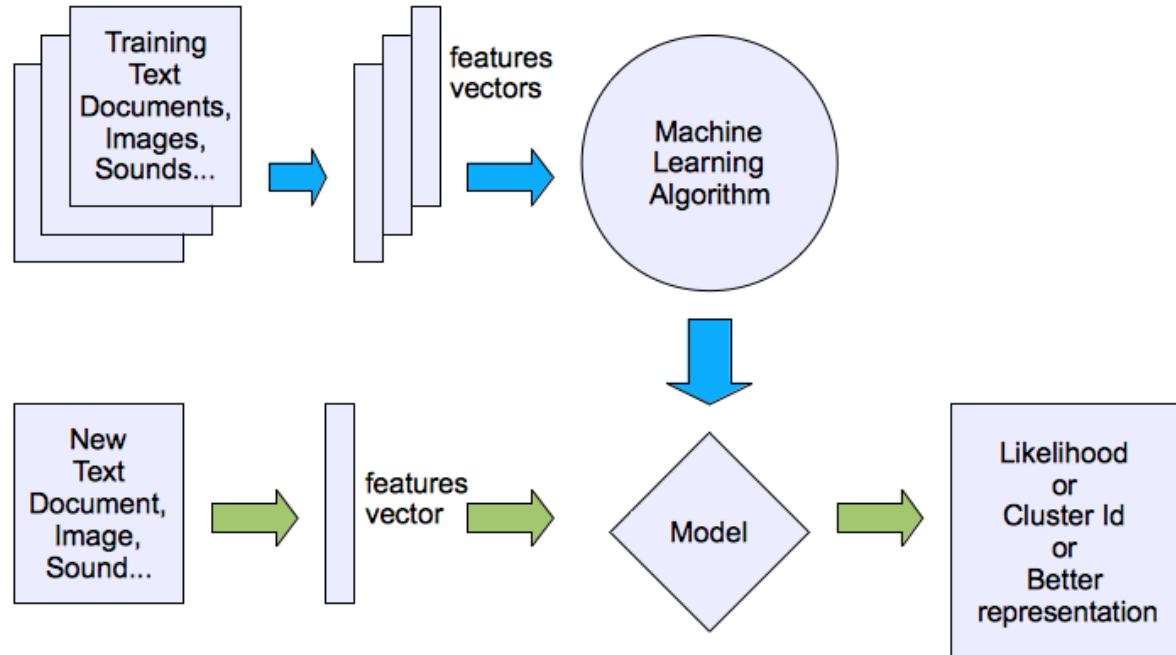
Machine learning structure

Supervised learning



Machine learning structure

Unsupervised learning



What are we seeking?

- Supervised: Low E-out or maximize probabilistic terms

$$\text{error} = \frac{1}{N} \sum_{n=1}^N [y_n \neq g(x_n)]$$

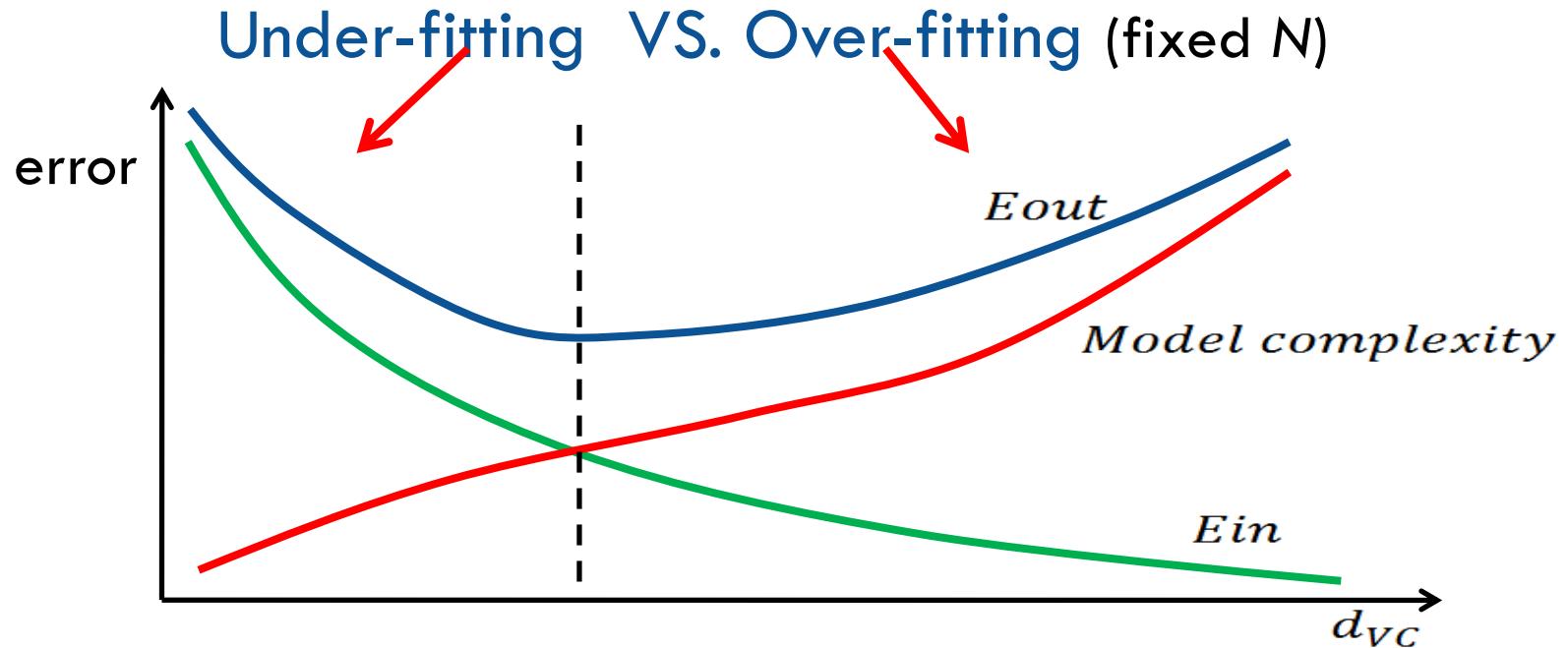
$$Eout(g) \leq Ein(g) \pm O\left(\sqrt{\frac{d_{vc}}{N} \ln N}\right)$$

E-in: for training set

E-out: for testing set

- Unsupervised: Minimum quantization error, Minimum distance, MAP, MLE(maximum likelihood estimation)

What are we seeking?



Learning techniques

- Supervised learning categories and techniques
 - **Linear classifier** (numerical functions)
 - **Parametric** (Probabilistic functions)
 - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
 - **Non-parametric** (Instance-based functions)
 - K-nearest neighbors, Kernel regression, Kernel density estimation, Local regression
 - **Non-metric** (Symbolic functions)
 - Classification and regression tree (CART), decision tree
 - **Aggregation**
 - Bagging (bootstrap + aggregation), Adaboost, Random forest

Learning techniques

- Unsupervised learning categories and techniques
 - **Clustering**
 - K-means clustering
 - Spectral clustering
 - **Density Estimation**
 - Gaussian mixture model (GMM)
 - Graphical models
 - **Dimensionality reduction**
 - Principal component analysis (PCA)
 - Factor analysis

Reinforcement Learning

- Topics:
 - Policies: what actions should an agent take in a particular situation
 - Utility estimation: how good is a state (\rightarrow used by policy)
- No supervised output but delayed reward
- Credit assignment problem (what was responsible for the outcome)
- Applications:
 - Game playing
 - Robot in a maze
 - Multiple agents, partial observability, ...

ML in a Nutshell

- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorithm has three components:
 - **Representation**
 - **Evaluation**
 - **Optimization**

Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

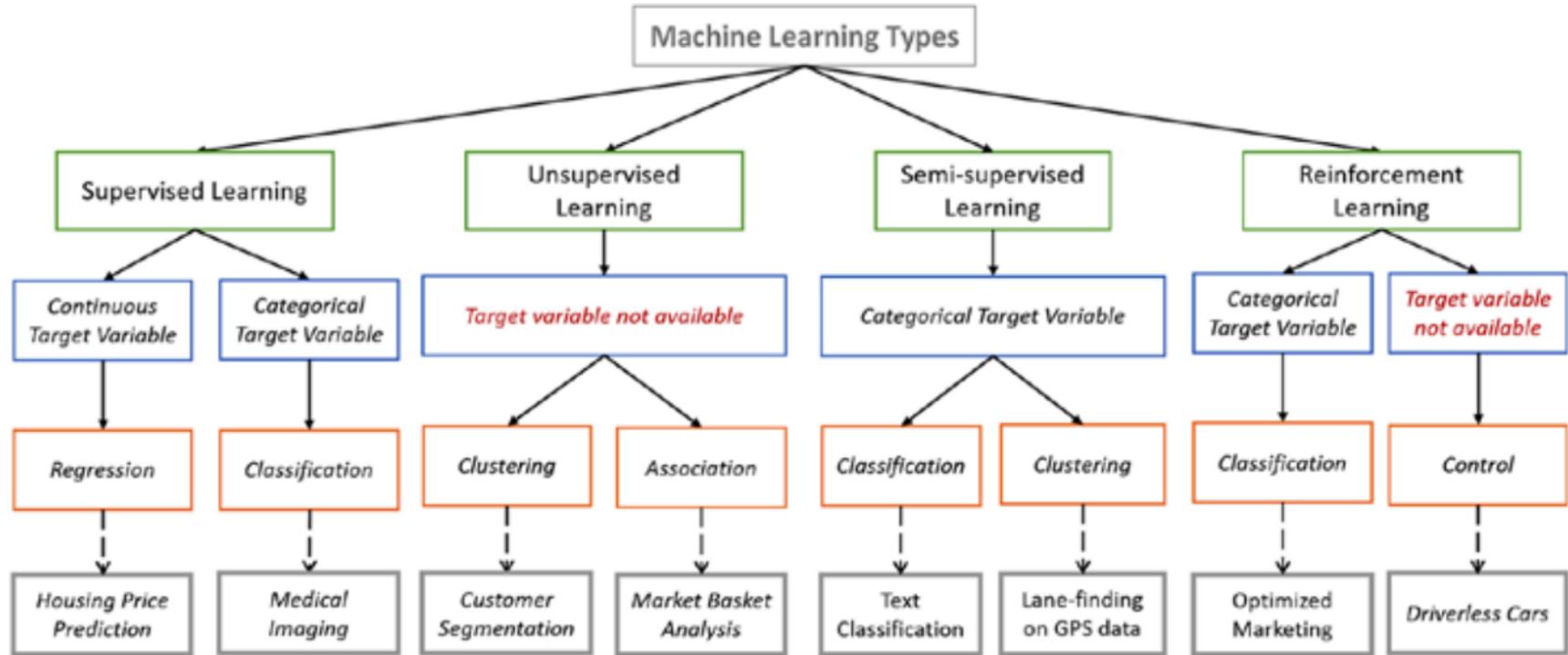
Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.

Optimization

- Combinatorial optimization
 - E.g.: Greedy search
- Convex optimization
 - E.g.: Gradient descent
- Constrained optimization
 - E.g.: Linear programming

Tech. Overview



Steps in developing a machine learning application

1. Collect data.
2. Prepare the input data.
3. Analyze the input data.
4. Filter garbage
5. Train the algorithm.
6. Test the algorithm.
7. Use it.

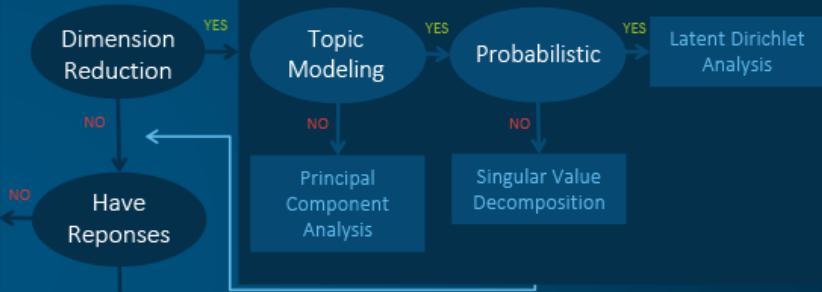
Machine Learning Algorithms Cheat Sheet

Unsupervised Learning: Clustering

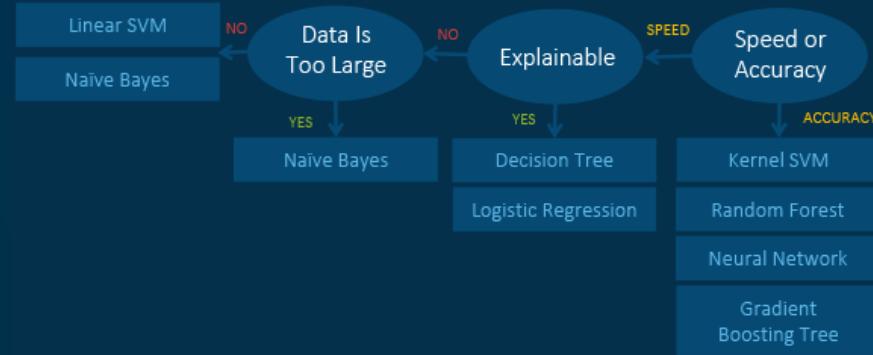


START

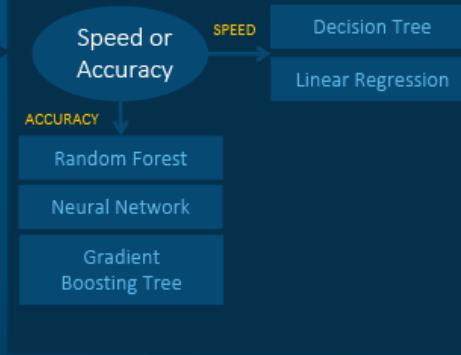
Unsupervised Learning: Dimension Reduction



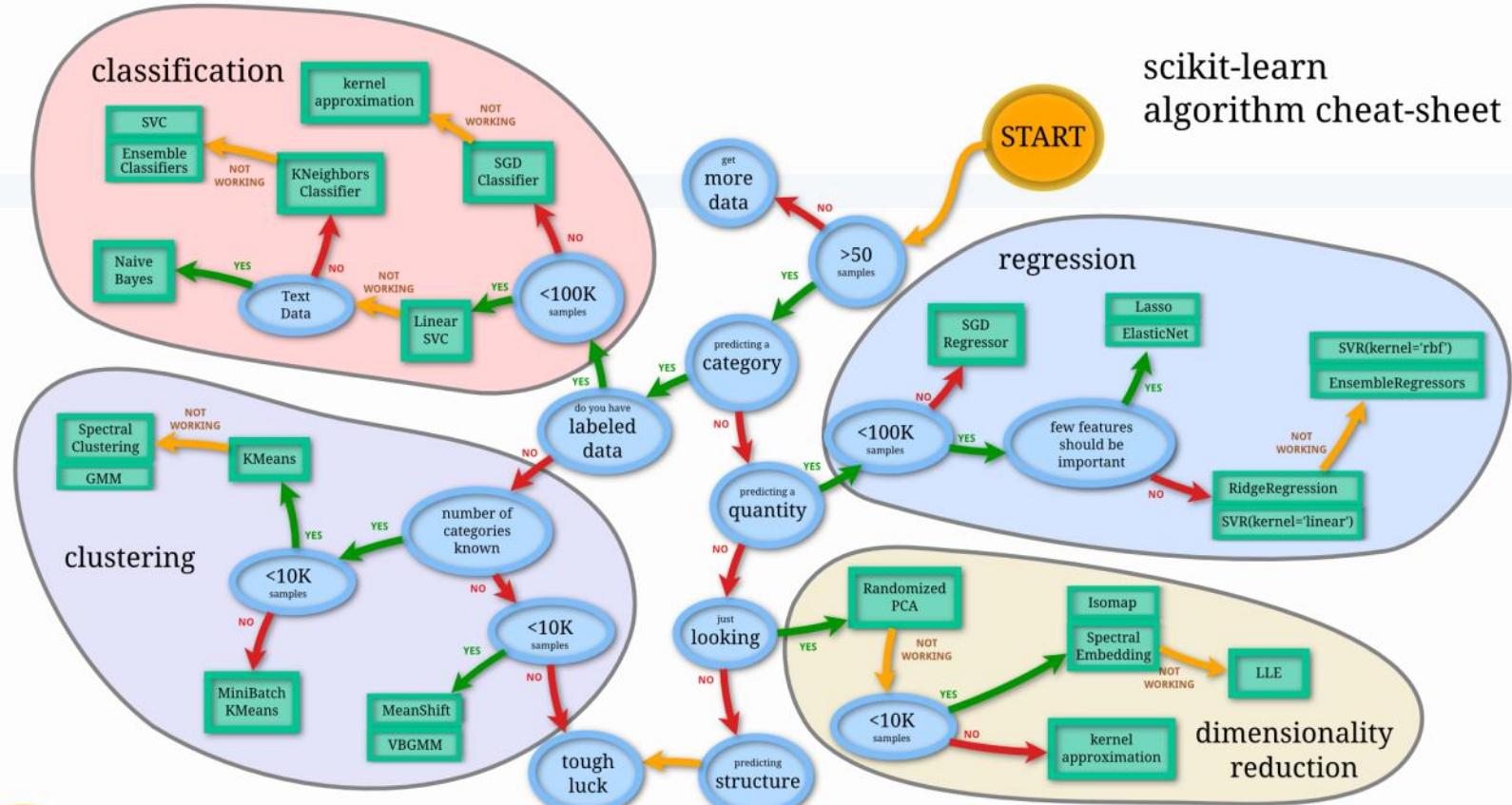
Supervised Learning: Classification



Supervised Learning: Regression



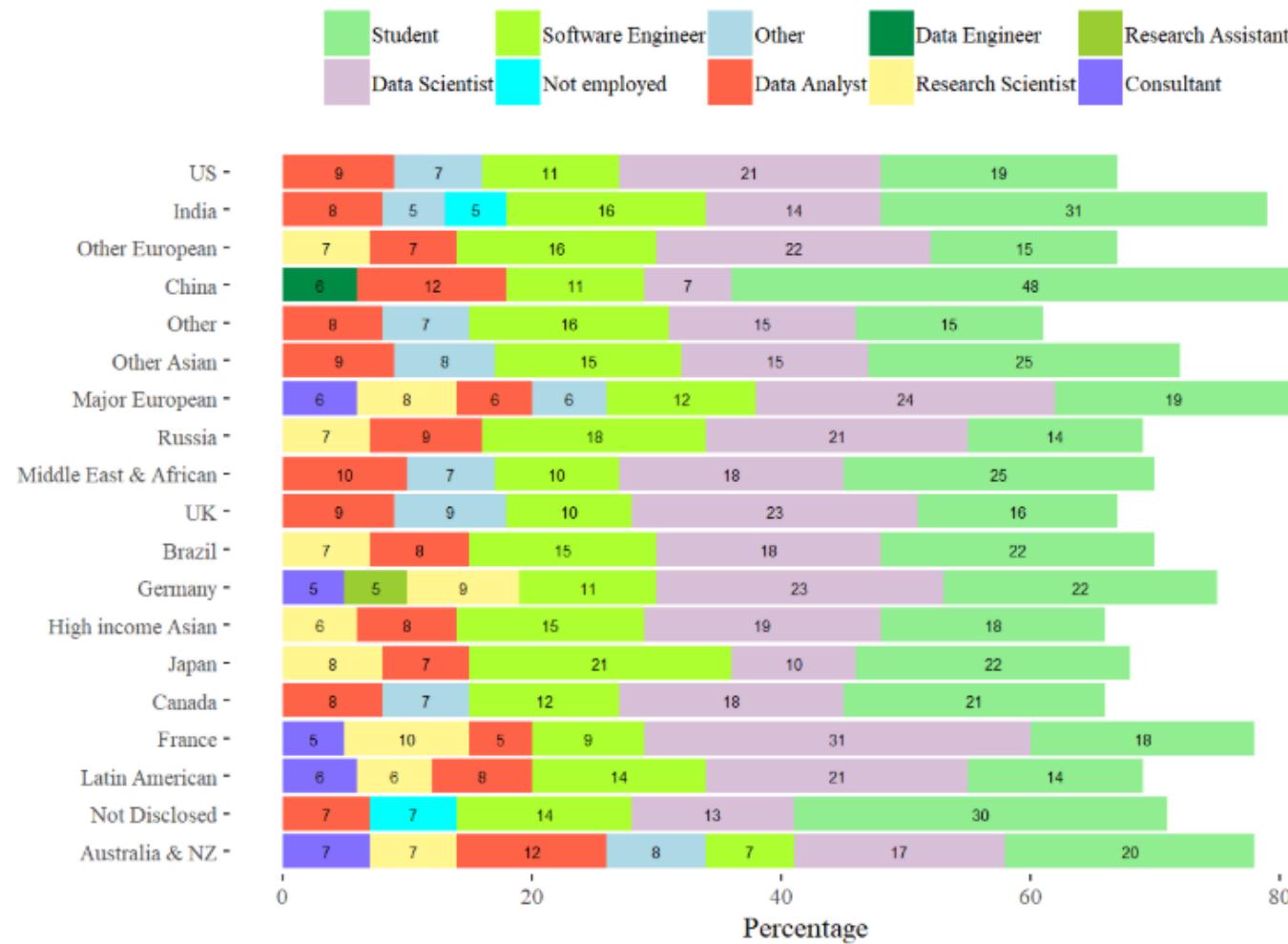
scikit-learn algorithm cheat-sheet



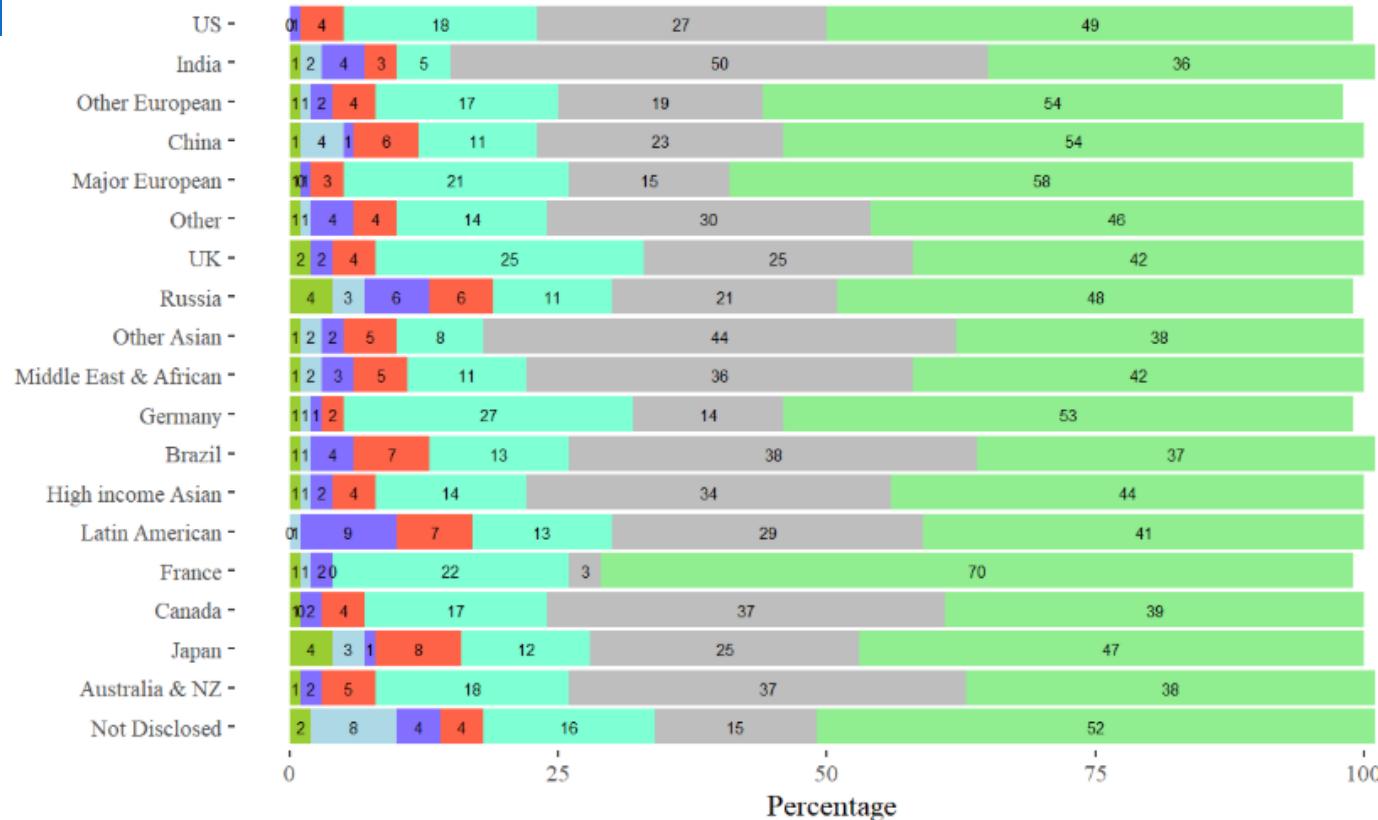
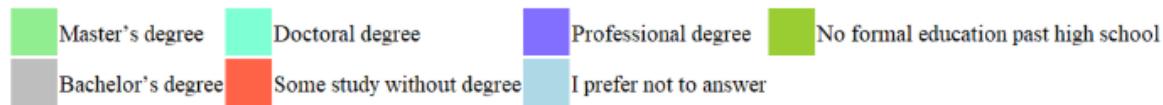
State of art

- Global report on state of Data Science & Machine Learning-2018 Based on Kaggle Survey
 - <https://rpubs.com/cvrajesh/kagglesurvey2018>

Top 5 Designation -By countries



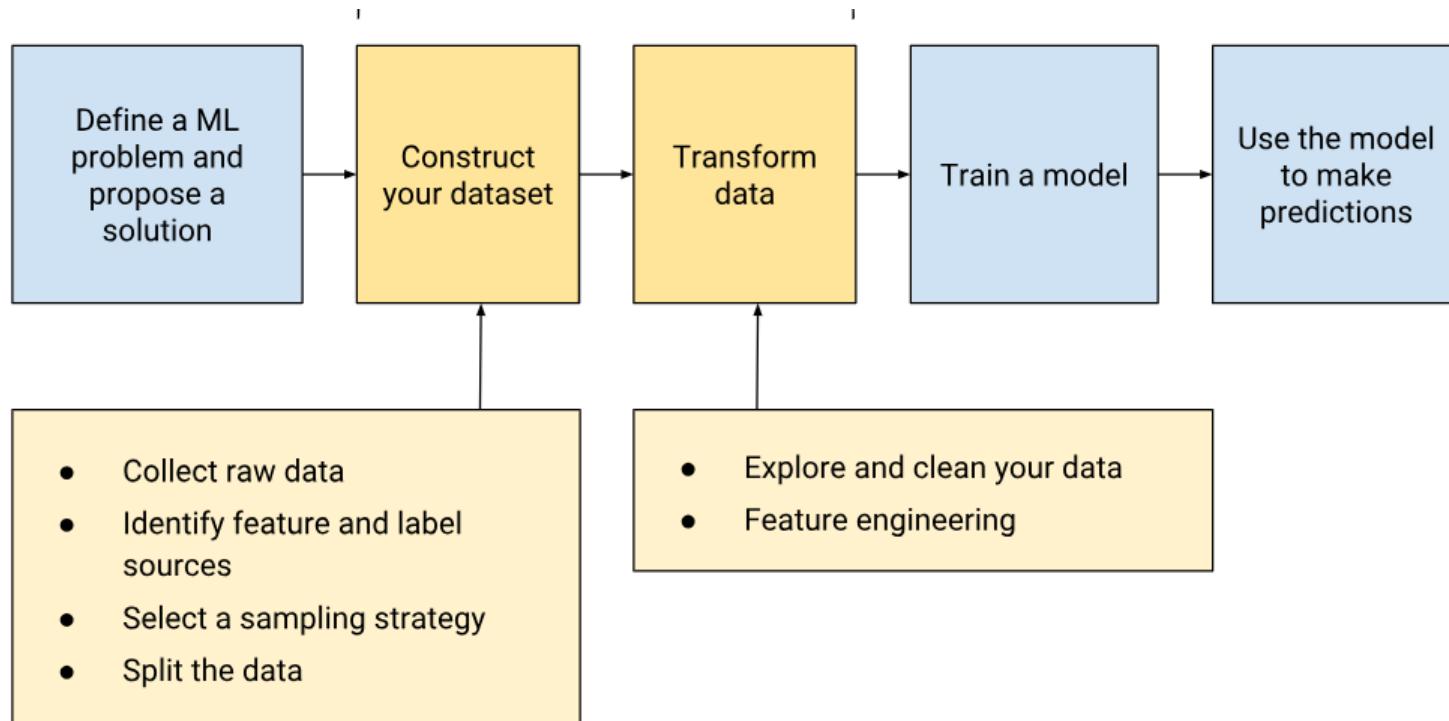
Education levels- By country of residence



Data

Data Preparation (Data pre-processing)

The Process for Data Preparation and Feature Engineering



Steps to Constructing Your Dataset

- To construct your dataset (and before doing data transformation), you should:
 1. Collect the raw data.
 2. Identify feature and label sources.
 3. Select a sampling strategy.
 4. Split the data.
- These steps depend a lot on how you've framed your ML problem. Use the self-check below to refresh your memory about problem framing and to check your assumptions about data collection.

Self-check of Problem Framing and Data Collection Concepts

- You're on a brand new machine learning project, about to select your first features. How many features should you pick?
 - Pick as many features as you can, so you can start observing which features have the strongest predictive power.

Self-check of Problem Framing and Data Collection Concepts

Pick as many features as you can, so you can start observing which features have the strongest predictive power.

- Start smaller. Every new feature adds a new dimension to your training data set. When the dimensionality increases, the volume of the space increases so fast that the available training data become sparse. The sparser your data, the harder it is for a model to learn the relationship between the features that actually matter and the label. This phenomenon is called "the curse of dimensionality."

Pick 4-6 features that seem to have strong predictive power.

- You might eventually use this many features, but it's still better to start with fewer. Fewer features usually means fewer unnecessary complications.

Pick 1-3 features that seem to have strong predictive power.

- It's best for your data collection pipeline to start with only one or two features. This will help you confirm that the ML model works as intended. Also, when you build a baseline from a couple of features, you'll feel like you're making progress!

Self-check of Problem Framing and Data Collection Concepts

- Your friend Sam is excited about the initial results of his statistical analysis. He says that the data show a positive correlation between the number of app downloads and the number of app review impressions. But he's not sure whether they would have downloaded it anyway without seeing the review. What response would be most helpful to Sam?

Self-check of Problem Framing and Data Collection Concepts

You could run an experiment to compare the behavior of users who didn't see the review with similar users who did.

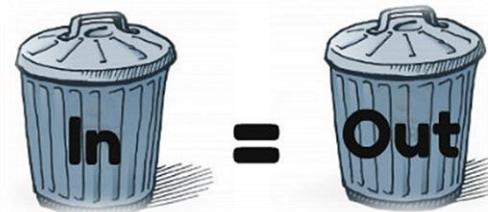
- Correct! If Sam observes that users who saw the positive review were more likely to download the app than those who didn't, then he has reasonable evidence to suggest that the positive review is encouraging people to get the app.

Trust the data. It's clear that that great review is the reason users are downloading the app.

- Incorrect. This response wouldn't lead Sam in the right direction. You can't determine causation from only observational data. Sam is seeing a correlation (that is, a statistical dependency between the numbers) that may or may not indicate causation. Don't let your analyses join the ranks of spurious correlations.

The Size and Quality of a Data Set

- “Garbage in, garbage out”



- After all, your model is only as good as your data.
- But how do you measure your data set's quality and improve it? And how much data do you need to get useful results? The answers depend on the type of problem you're solving.

The Size of a Data Set

- As a rough rule of thumb, your model should train on at least an order of magnitude more examples than trainable parameters. Simple models on large data sets generally beat fancy models on small data sets.

Data set	Size (number of examples)
Iris flower data set	150 (total set)
MovieLens (the 20M data set)	20,000,263 (total set)
Google Gmail SmartReply	238,000,000 (training set)
Google Books Ngram	468,000,000,000 (total set)
Google Translate	trillions

The Quality of a Data Set

- It's no use having a lot of data if it's bad data; quality matters, too.
 - But what counts as "quality"?
 - It's a fuzzy term.
- Consider taking an empirical approach and picking the option that produces the best outcome. With that mindset, a quality data set is one that lets you succeed with the business problem you care about. In other words, the data is good if it accomplishes its intended task.

The Quality of a Data Set

- However, while collecting data, it's helpful to have a more concrete definition of quality. Certain aspects of quality tend to correspond to better-performing models:
 - reliability
 - feature representation
 - minimizing skew

Reliability

- **Reliability** refers to the degree to which you can *trust* your data. A model trained on a reliable data set is more likely to yield useful predictions than a model trained on unreliable data. In measuring reliability, you must determine:
 1. How common are label errors? For example, if your data is labeled by humans, sometimes humans make mistakes.
 2. Are your features noisy? For example, GPS measurements fluctuate. Some noise is okay. You'll never purge your data set of *all* noise. You can collect more examples too.
 3. Is the data properly filtered for your problem? For example, should your data set include search queries from bots? If you're building a spam-detection system, then likely the answer is yes, but if you're trying to improve search results for humans, then no.

Reliability

- What makes data unreliable?
- many examples in data sets are unreliable due to one or more of the following:
 - Omitted values. For instance, a person forgot to enter a value for a house's age.
 - Duplicate examples. For example, a server mistakenly uploaded the same logs twice.
 - Bad labels. For instance, a person mislabeled a picture of an oak tree as a maple.
 - Bad feature values. For example, someone typed an extra digit, or a thermometer was left out in the sun.

Feature Representation

- Representation is the mapping of data to **useful features**. You'll want to consider the following questions:
 - How is data shown to the model?
 - Should you normalize numeric values?
 - How should you handle outliers?

Training versus Prediction

- Let's say you get great results offline. Then in your live experiment, those results don't hold up. What could be happening?
- This problem suggests **training/serving skew**—that is, different results are computed for your metrics at training time vs. serving time.
- Causes of skew can be subtle but have deadly effects on your results. Always consider what data is available to your model at prediction time. During training, use only the features that you'll have available in serving, and make sure your training set is representative of your serving traffic.

Identifying Labels and Sources



Direct vs. Derived Labels

- Machine learning is easier when your labels are well-defined. The best label is a **direct label** of what you want to predict. For example, if you want to predict whether a user is a Taylor Swift fan, a direct label would be "User is a Taylor Swift fan."
- A simpler test of fanhood might be whether the user has watched a Taylor Swift video on YouTube. The label "user has watched a Taylor Swift video on YouTube" is a **derived label** because it does not directly measure what you want to predict. Is this derived label a reliable indicator that the user likes Taylor Swift? Your model will only be as good as the connection between your derived label and your desired prediction.

Label Sources

- The output of your model could be either an Event or an Attribute. This results in the following two types of labels:
- **Direct label for Events**, such as “Did the user click the top search result?”
- **Direct label for Attributes**, such as “Will the advertiser spend more than \$X in the next week?”



Why Use Human Labeled Data?

- There are advantages and disadvantages to using human-labeled data.
- +
 - Human raters can perform a wide range of tasks.
 - The data forces you to have a clear problem definition.
- -
 - The data is expensive for certain domains.
 - Good data typically requires multiple iterations.

Sampling and Splitting Data

- It's often a struggle to gather enough data for a machine learning project. Sometimes, however, there is *too much data*, and you must select a subset of examples for training.

Sampling and Splitting Data

- How do you select that subset?
- As an example, consider Google Search. At what granularity would you sample its massive amounts of data? Would you use random queries? Random sessions? Random users?

Sampling and Splitting Data

- Ultimately, the answer depends on the problem: what do we want to predict, and what features do we want?
- To use the feature *previous query*, you need to sample at the session level, because sessions contain a sequence of queries.
- To use the feature *user behavior from previous days*, you need to sample at the user level.

Imbalanced Data

- A classification data set with skewed class proportions is called imbalanced. Classes that make up a large proportion of the data set are called majority classes. Those that make up a smaller proportion are minority classes.

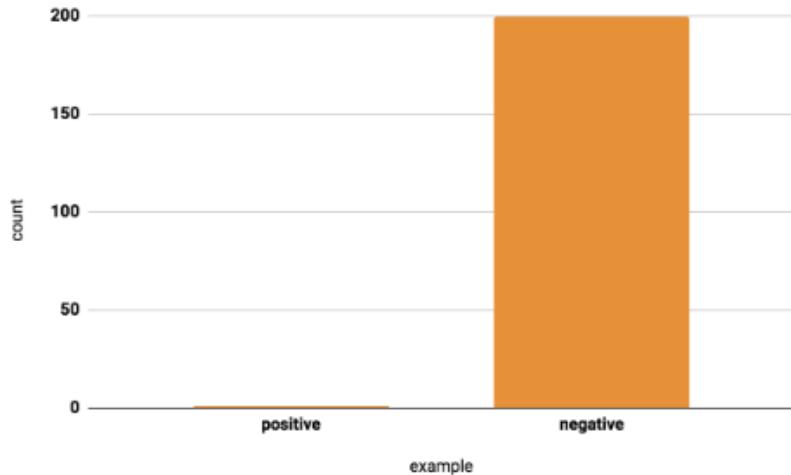
Imbalanced Data

- What counts as imbalanced? The answer could range from mild to extreme, as the table below shows.

Degree of imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

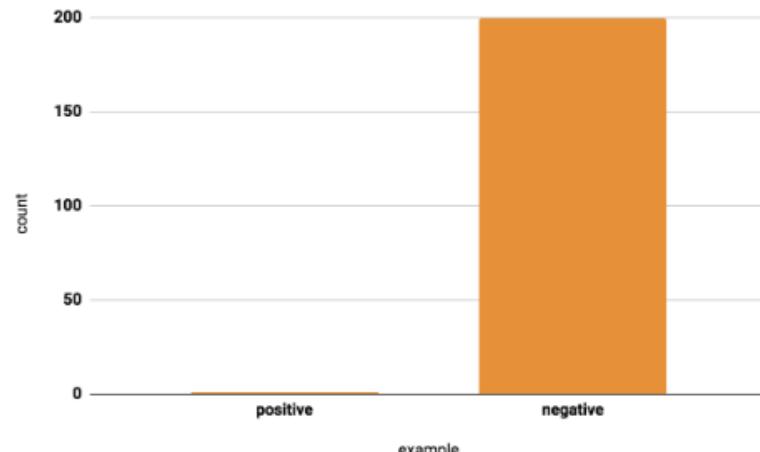
Why look out for imbalanced data?

- Consider the following example of a model that detects fraud. Instances of fraud happen once per 200 transactions in this data set, so in the true distribution, about 0.5% of the data is positive.



Why look out for imbalanced data?

- Why would this be problematic?
- With so few positives relative to negatives, the training model will spend most of its time on negative examples and not learn enough from positive ones.
- For example, if your batch size is 128, many batches will have no positive examples, so the gradients will be less informative.



Why look out for imbalanced data?

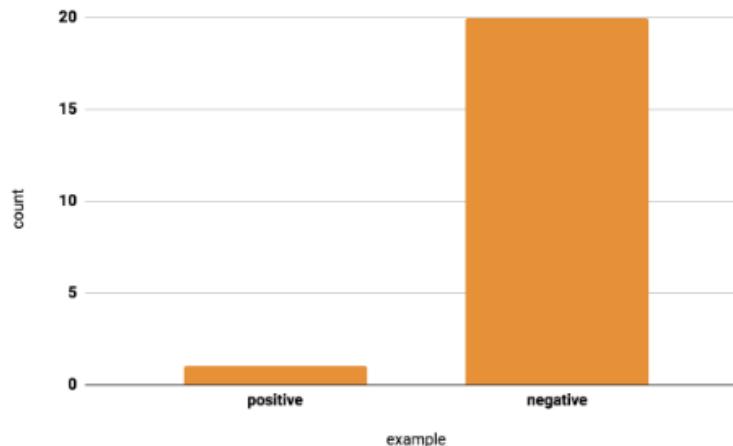
- If you have an imbalanced data set, **first try training on the true distribution**. If the model works well and generalizes, you're done! If not, try the following downsampling and upweighting technique.

Downsampling and Upweighting

- An effective way to handle imbalanced data is to downsample and upweight the majority class. Let's start by defining those two new terms:
 - **Downsampling** (in this context) means training on a disproportionately low subset of the majority class examples.
 - **Upweighting** means adding an example weight to the downsampled class equal to the factor by which you downsampled.

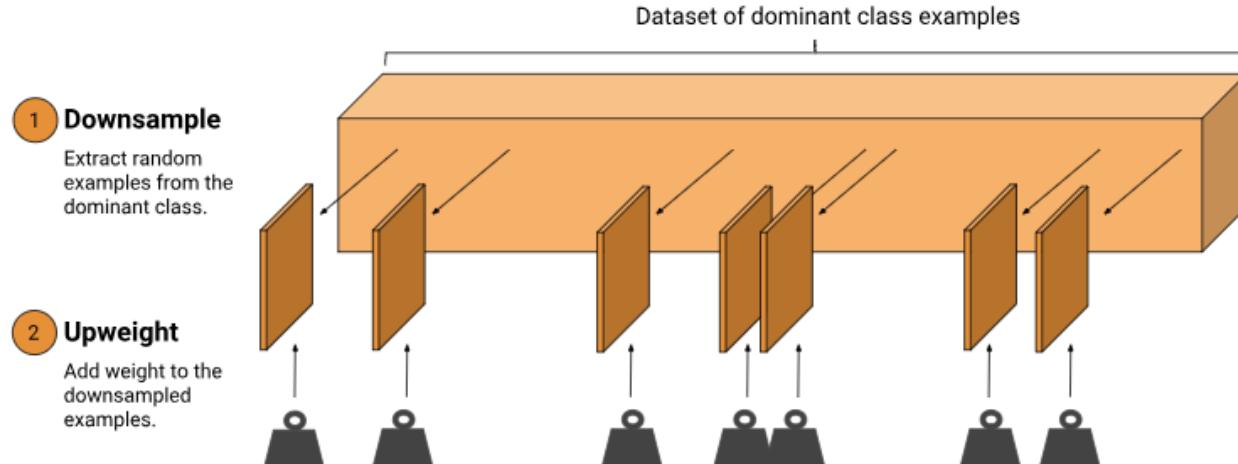
Downsample the majority class

- **Step 1: Downsample the majority class.** Consider again our example of the fraud data set, with 1 positive to 200 negatives. We can downsample by a factor of 20, taking 1/10 negatives. Now about 10% of our data is positive, which will be much better for training our model.



Upweight the downsampled class

- **Step 2: Upweight the downsampled class:** The last step is to add example weights to the downsampled class. Since we downsampled by a factor of 20, the example weight should be 20.



weights

- Here we're talking about *example weights*, which means counting an individual example more importantly during training. An example weight of 10 means the model treats the example as 10 times as important (when computing loss) as it would an example of weight 1.
- The weight should be equal to the factor you used to downsample:

$$\{\text{example weight}\} = \{\text{original example weight}\} \times \{\text{downsampling factor}\}$$

Why Downsample and Upweight?

- It may seem odd to add example weights after downsampling. We were trying to make our model improve on the minority class -- why would we upweight the majority? These are the resulting changes:
 - **Faster convergence:** During training, we see the minority class more often, which will help the model converge faster.
 - **Disk space:** By consolidating the majority class into fewer examples with larger weights, we spend less disk space storing them. This savings allows more disk space for the minority class, so we can collect a greater number and a wider range of examples from that class.
 - **Calibration:** Upweighting ensures our model is still calibrated; the outputs can still be interpreted as probabilities.

Data Split Example

- After collecting your data and sampling where needed, the next step is to split your data into **training sets**, **validation sets**, and **testing sets**.

When Random Splitting isn't the Best Approach

- While random splitting is the best approach for many ML problems, it isn't always the right solution. For example, consider data sets in which the examples are naturally clustered into similar examples.

When Random Splitting isn't the Best Approach

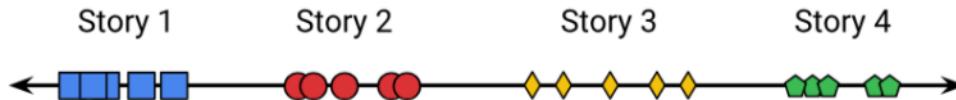


Figure 1. News Stories are Clustered.

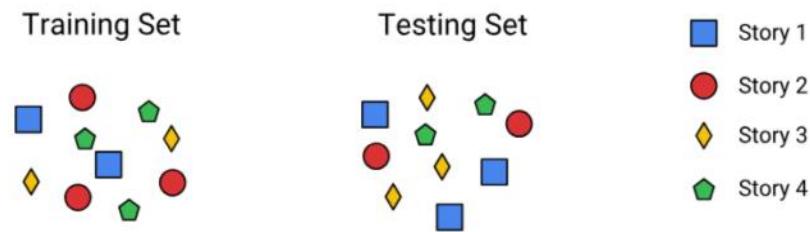
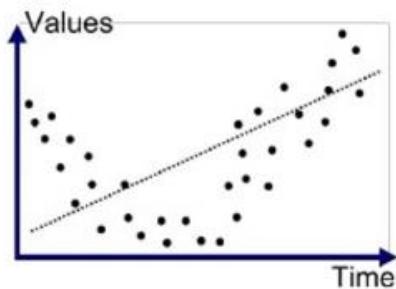


Figure 2. A random split will split a cluster across sets, causing skew.

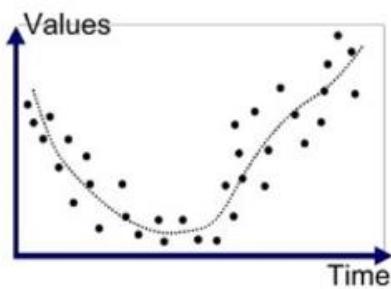


Figure 3. Splitting on time allows the clusters to mostly end up in the same set.

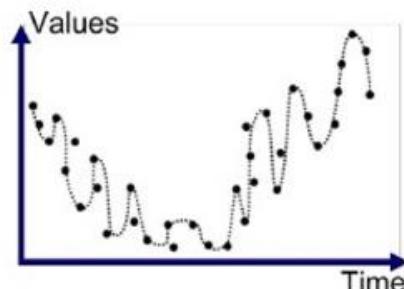
Why Our model perform poor sometime?



Underfitted



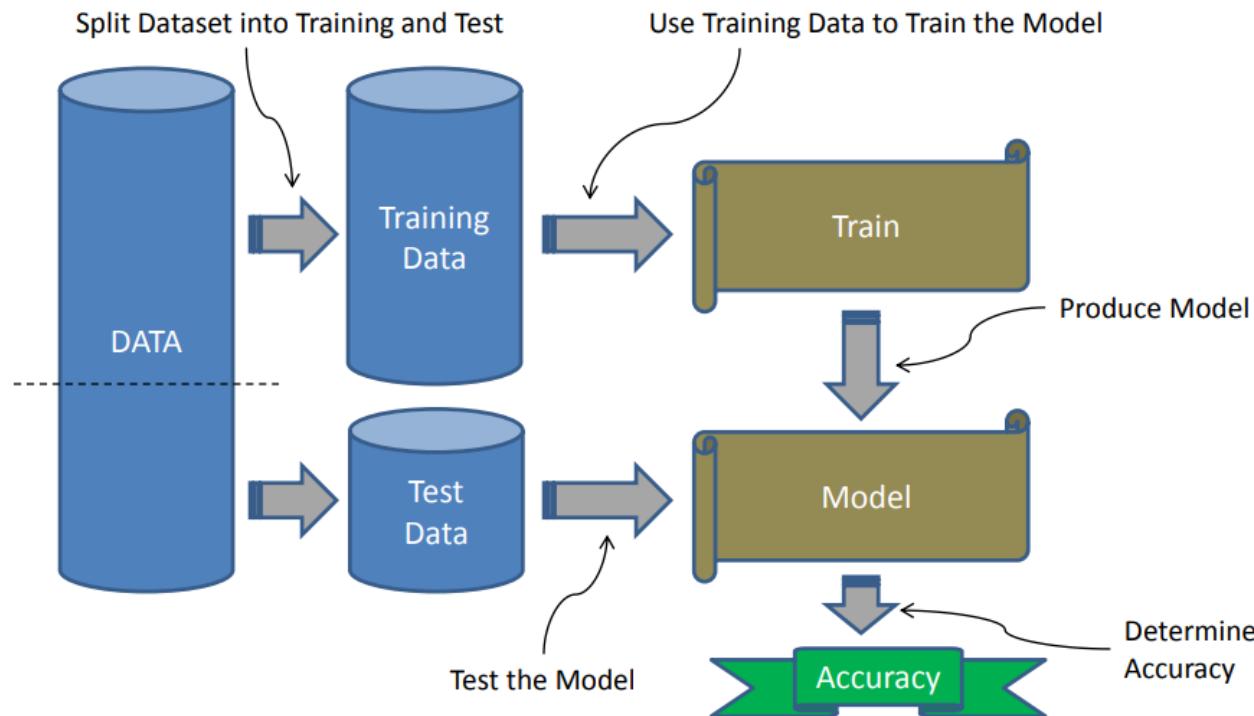
Good Fit/R robust



Overfitted

Image Source: <http://blog.algotrading101.com/design-theories/what-is-curve-fitting-overfitting-in-trading/>

Recall ...



imbalanced data - overfitting

- If the training data is overly unbalanced, then the model will predict a non-meaningful result.
 - For example, if the model is a binary classifier (e.g., cat vs. dog), and nearly all the samples are of the same label (e.g., cat), then the model will simply learn that everything is a that label (cat).
- This is called **overfitting**. To prevent overfitting, there needs to be a fairly equal distribution of training samples for each classification, or range if label is a real value.

Overfitting

- In data science courses, an **overfit** model is explained as having high variance and low bias on the training set which leads to poor generalization on new testing data.

How To Limit Overfitting

- Both overfitting and underfitting can lead to poor model performance. But by far the most common problem in applied machine learning is overfitting.
- Overfitting is such a problem because the evaluation of machine learning algorithms on training data is different from the evaluation we actually care the most about, namely how well the algorithm performs on unseen data.

How To Limit Overfitting

- There are two important techniques that you can use when evaluating machine learning algorithms to limit overfitting:
 1. Use a resampling technique to estimate model accuracy.
 2. Hold back a validation dataset.

Underfitting

- Underfitting refers to a model that can neither model the training data nor generalize to new data.
- An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data.

Basic

- Data science may seem complex but it is really built out of a series of basic building blocks:
 - **Overfitting:** too much reliance on the training data
 - **Underfitting:** a failure to learn the relationships in the training data
 - **High Variance:** model changes significantly based on training data
 - **High Bias:** assumptions about model lead to ignoring training data
 - Overfitting and underfitting cause poor **generalization** on the test set
 - A **validation set** for model tuning can prevent under and overfitting
 - ...

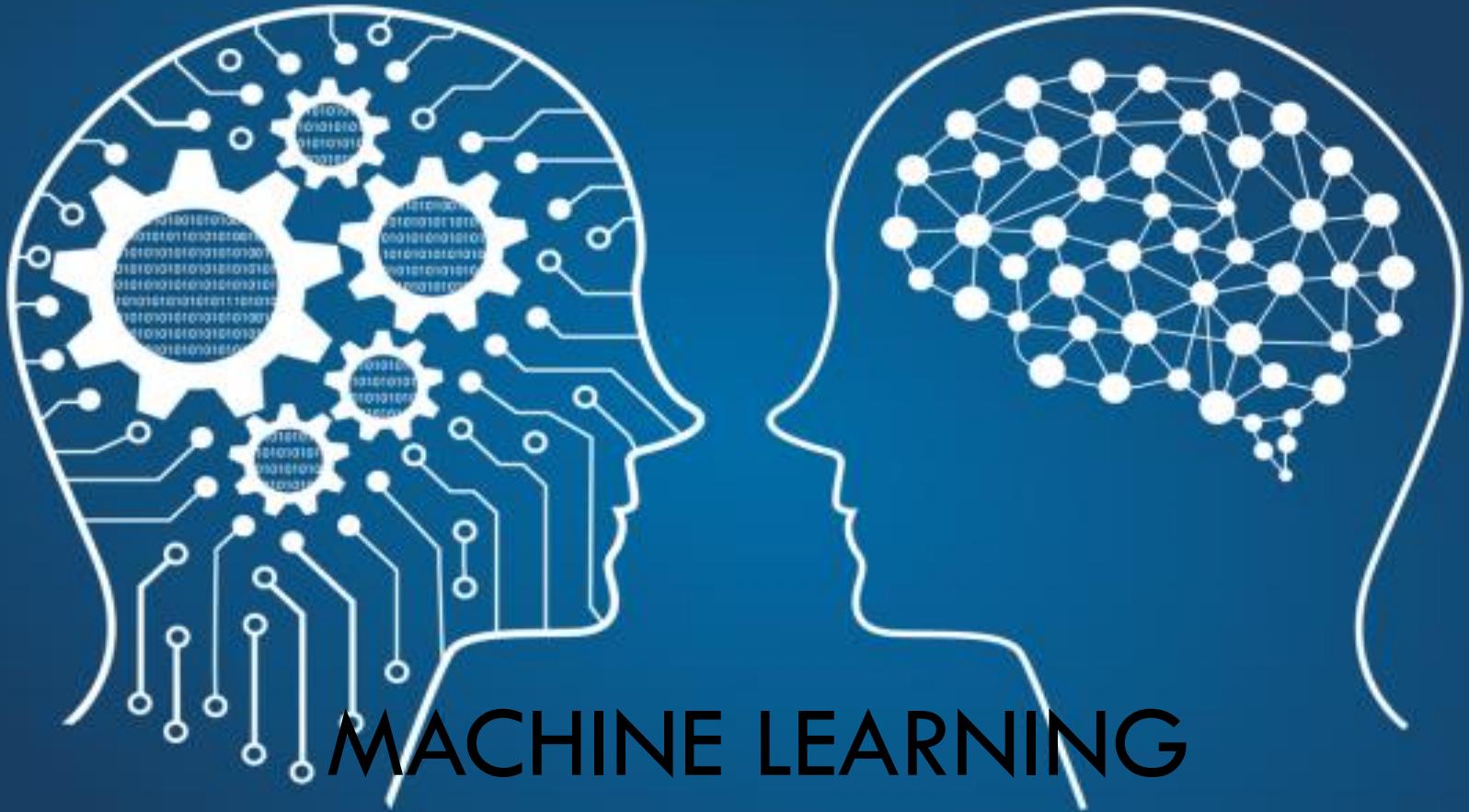
Web Scraping

Web Scraping

API

- **Beautiful Soup**
 - is a tool which help programmer quickly extract valid data from web pages
- **Scrapy**
 - is a web crawling framework

Example

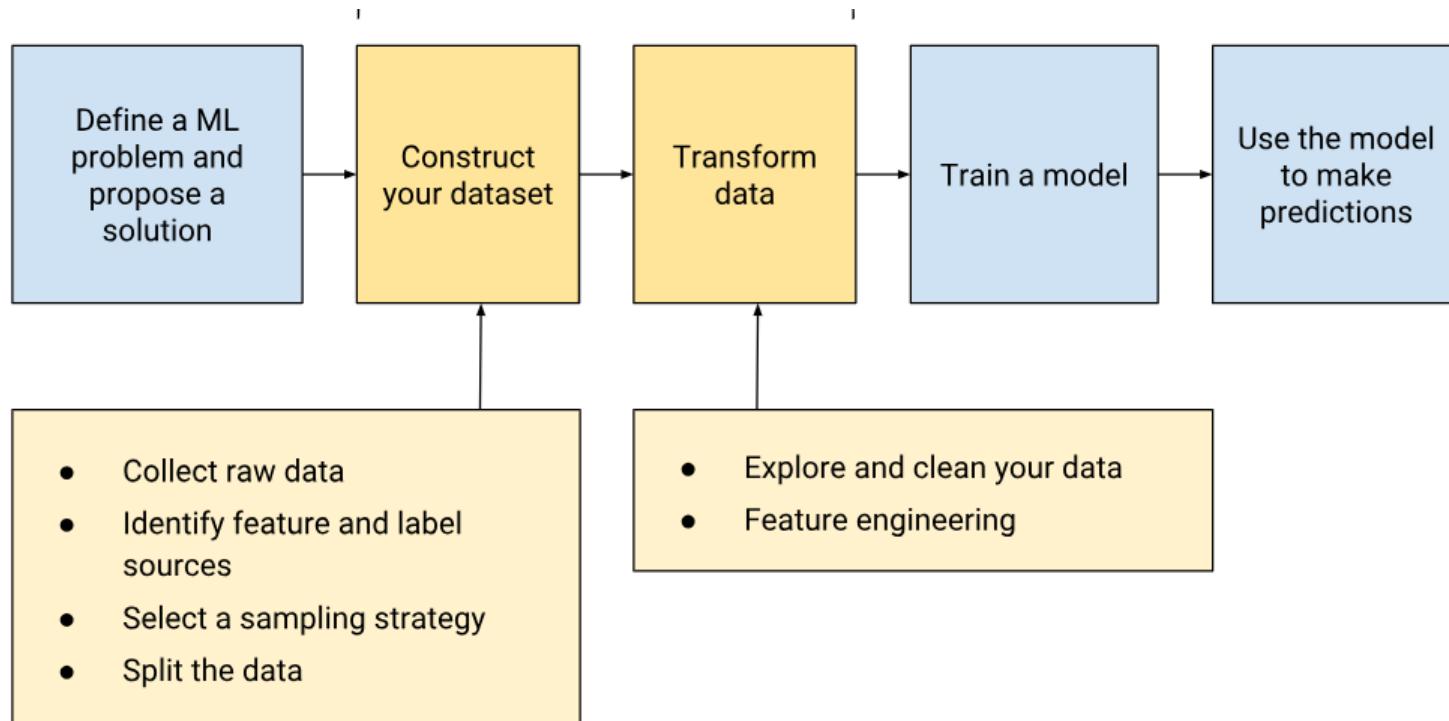


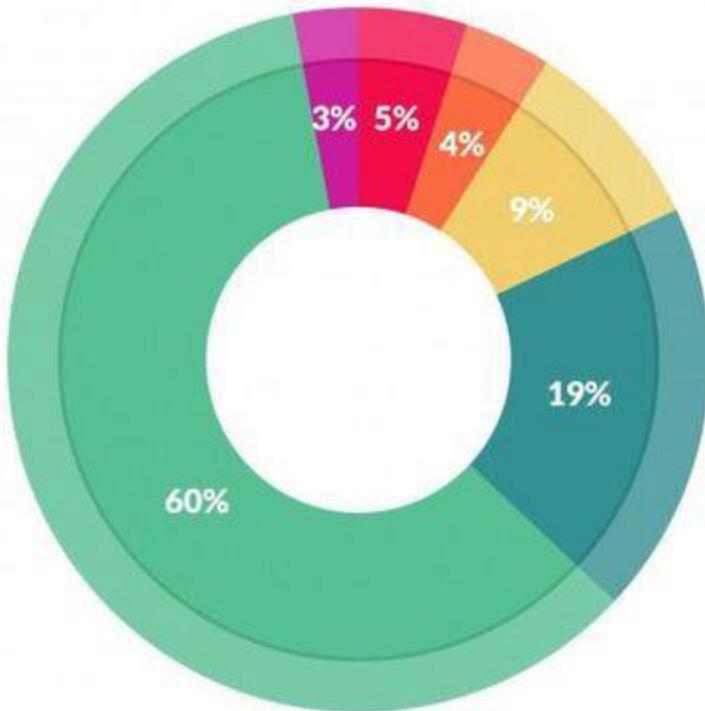
Pr. BEN LAHMAR EL Habib

Features engineering



The Process for Data Preparation and Feature Engineering



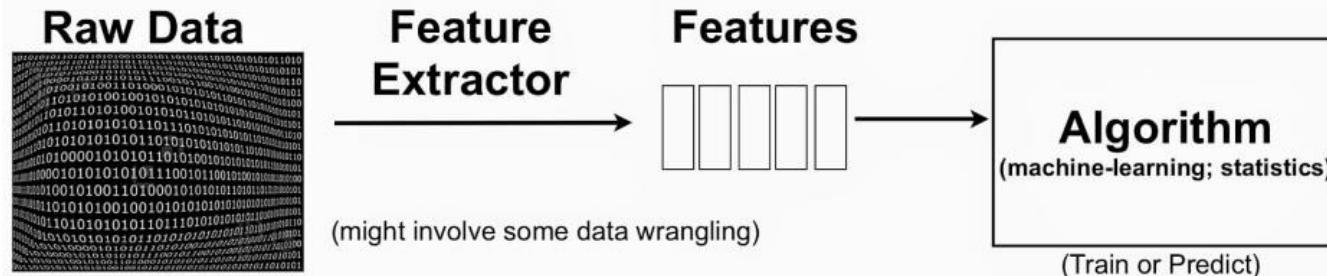


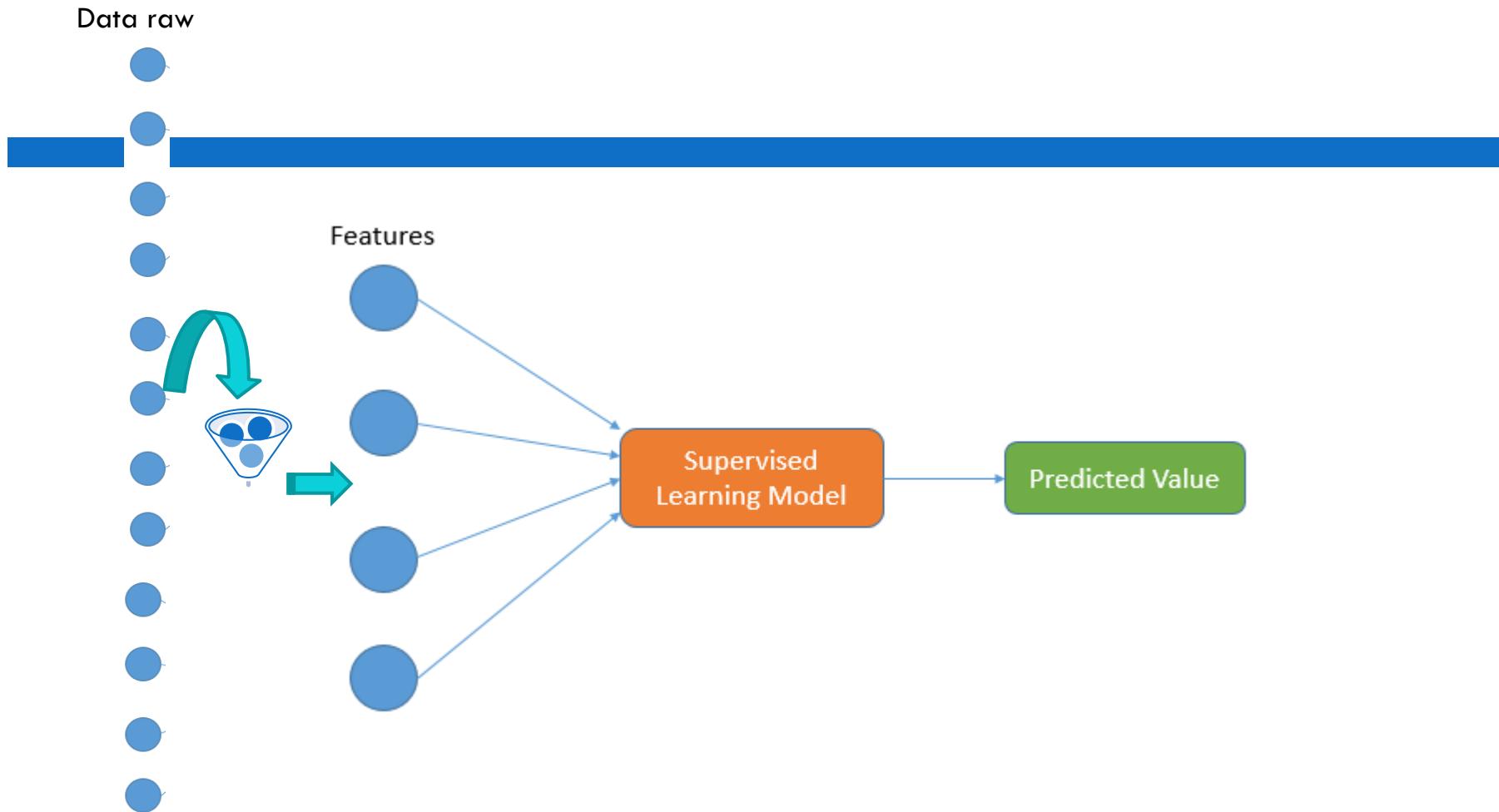
What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

what is Feature Engineering?

- Simply put, it is the art/science of **representing data** is the best way possible.
- Good Feature Engineering involves an elegant blend of domain knowledge, intuition, and basic mathematical abilities.





what ‘best’?

- In essence, the way you present your data to your algorithm should denote the pertinent **structures/properties** of the underlying *information* in the most effective way possible. When you do feature engineering, you are essentially converting your data *attributes* into data *features*.

Attributes

- Attributes are basically all the dimensions present in your data. But do all of them, in the raw format, represent the underlying trends you want to learn in the best way possible? Maybe not.

Feature

- So what you do in feature engineering, is **pre-process** your data so that your model/learning algorithm has to spend minimum effort on wading through noise.
- ‘noise’ is any information that is not relevant to learning/predicting your ultimate goal.
- In fact, using good features can even let you use considerably simpler models.

Feature

- as with any technique in Machine Learning, always use validation to make sure that the new features you introduce really do improve your predictions, instead of adding unnecessary complexity to your pipeline.

Example

- Representing timestamps
- Decomposing Categorical Attributes
- Binning/Bucketing
- Feature Crosses
- Feature Selection
- Feature Scaling (data normalization)
- Feature Extraction

Representing timestamps

- Time-stamp attributes are usually denoted by the EPOCH time or split up into multiple dimensions such as (Year, Month, Date, Hours, Minutes, Seconds)
- But in many applications, a lot of that information is unnecessary.

Representing timestamps

- Consider for example a supervised system that tries to predict traffic levels in a city as a function of Location+Time.



- In this case, trying to learn trends that vary by seconds would mostly be misleading. The year wouldn't add much value to the model as well.

Representing timestamps

- Hours, day and month are probably the only dimensions you need. So when representing the time, **try** to ensure that your model does require all the numbers you are providing it.
- If your data sources come from different geographical sources, do remember to normalize by time-zones if needed.

Decomposing Categorical Attributes

Why

- In many practical data science activities, the data set will contain categorical variables.
- These variables are typically stored as text values". Since machine learning is based on mathematical equations, it would cause a problem when we keep categorical variables as is.

Encoding Methodologies

- **Nominal Encoding:** — Where Order of data does not matter
- In Nominal Encoding we have various techniques:
 - One Hot Encoding
 - One Hot Encoding With Many Categories
 - Mean Encoding
- **Ordinal Encoding:** — Where Order of data matters
 - In Ordinal Encoding we also have various techniques
 - Label Encoding
 - Target Guided Ordinal Encoding

Example

- Examples for Nominal variable:
 - Red, Yellow, Pink, Blue
 - Singapore, Japan, USA, India, Korea
 - Cow, Dog, Cat, Snake
- Example of Ordinal variables:
 - High, Medium, Low
 - “Strongly agree,” Agree, Neutral, Disagree, and “Strongly Disagree.”
 - Excellent, Okay, Bad

Encoding

- 1) One Hot Encoding
- 2) Label Encoding
- 3) Ordinal Encoding
- 4) Helmert Encoding
- 5) Binary Encoding
- 6) Frequency Encoding
- 7) Mean Encoding
- 8) Weight of Evidence Encoding
- 9) Probability Ratio Encoding
- 10) Hashing Encoding
- 11) Backward Difference Encoding
- 12) Leave One Out Encoding
- 13) James-Stein Encoding
- 14) M-estimator Encoding

Exemple (dataset)

No	Section	Semester	Target
0	Smi	S6	1
1	Sma	S5	0
2	Msdbd	S2	1
3	Smi	S5	1
4	Sma	S4	1
5	Smp	S6	0
6	Smi	S4	0
7	Smp	S6	1
8	Msdbd	S1	1
9	Smp	S5	0

One Hot Encoding

- In this method, we map each category to a vector that contains 1 and 0 denoting the presence or absence of the feature.
- The number of vectors depends on the number of categories for features. This method produces a lot of columns that slows down the learning significantly if the number of the category is very high for the feature.

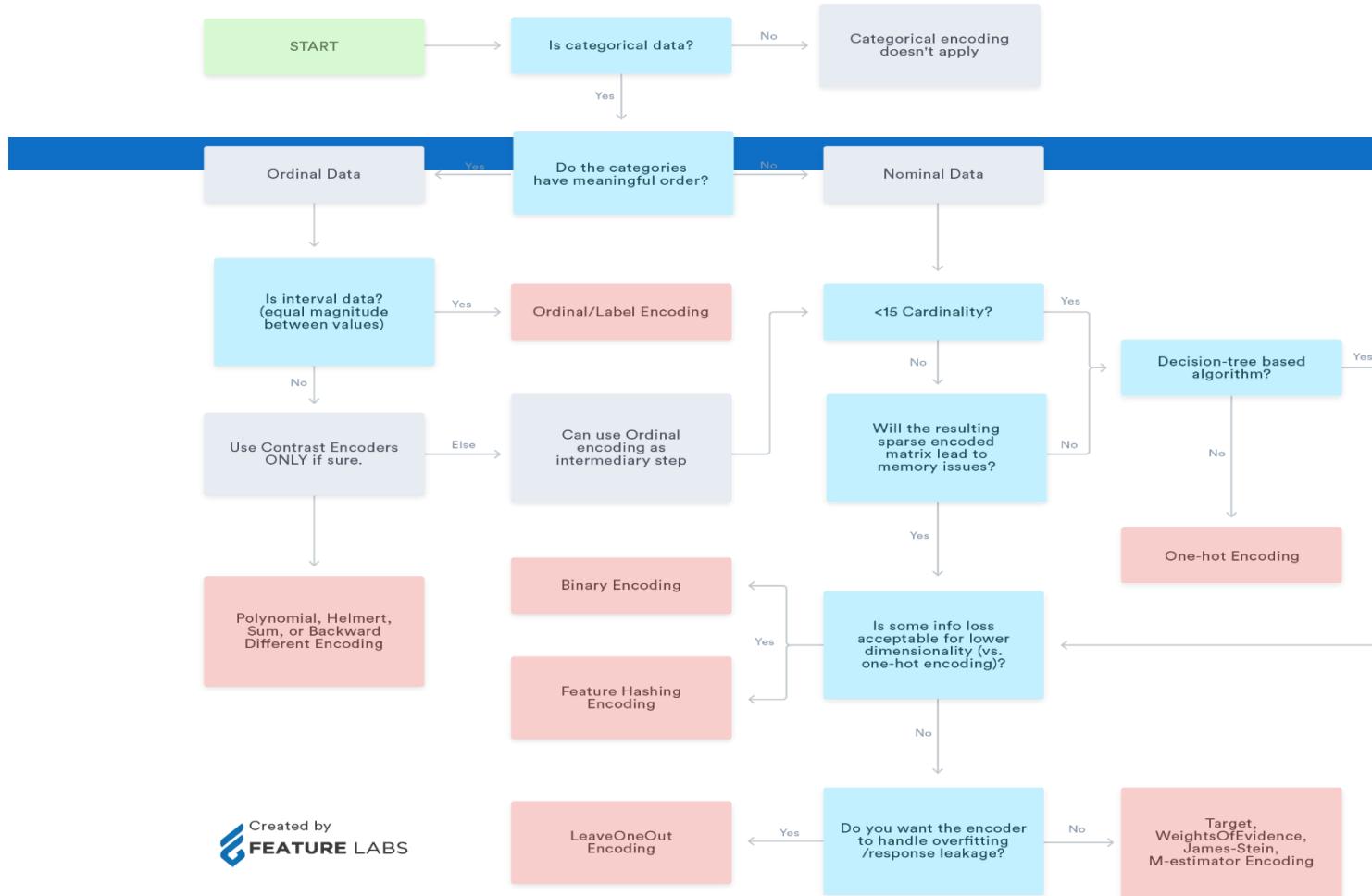
Label Encoding

- In this encoding, each category is assigned a value from 1 through N (here N is the number of categories for the feature).
- One major issue with this approach is there is no relation or order between these classes, but the algorithm might consider them as some order, or there is some relationship.

Ordinal Encoding

- We do Ordinal encoding to ensure the encoding of variables retains the ordinal nature of the variable

Categorical Encoding Methods Cheat-Sheet



Binning/Bucketing

- Sometimes, it makes more sense to represent a numerical attribute as a categorical one.
- Example : Consider the problem of predicting whether a person owns a certain item of clothing or not.
 - Age might definitely be a factor here. What is actually more pertinent, is the **Age Group**.
 - So what you could do, is have ranges such as 1-10, 11-18, 19-25, 26-40, etc.

Binning

- *Binning* or grouping data (sometimes called *quantization*) is an important tool in preparing numerical data for machine learning,

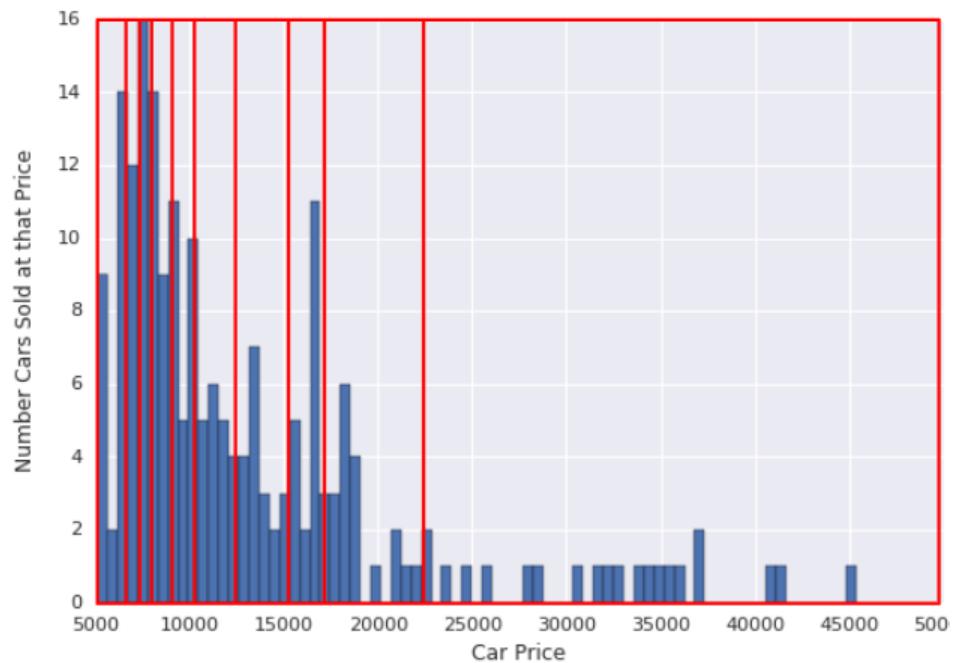
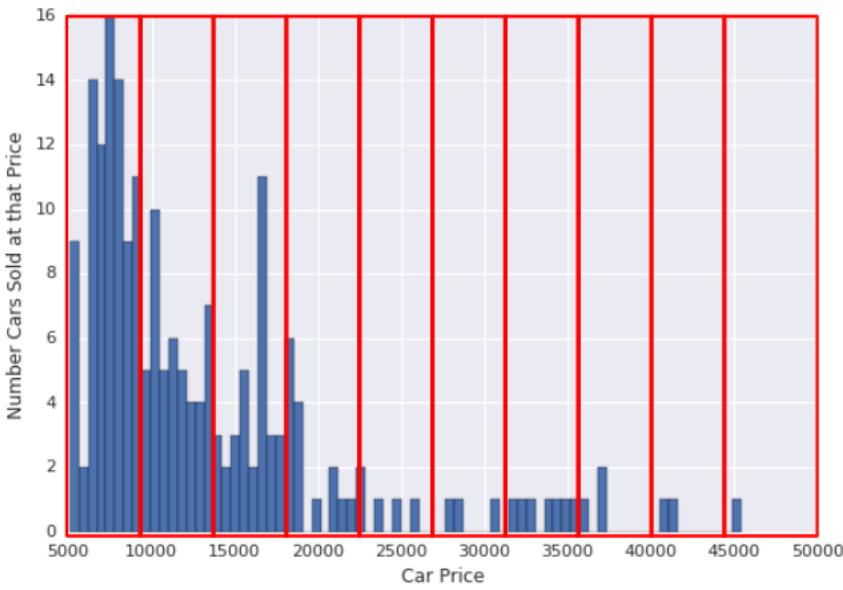
Binning

- Binning also reduces the effect of tiny errors, by 'rounding off' a given value to the nearest representative.
- Binning does *not* make sense if the number of your ranges is comparable to the total possible values, or if precision is very important to you.

Bucketing

- Bucketing makes sense when the domain of your attribute can be divided into **neat ranges**, where all numbers falling in a range imply a **common characteristic**.
 - It reduces overfitting in certain applications

Example

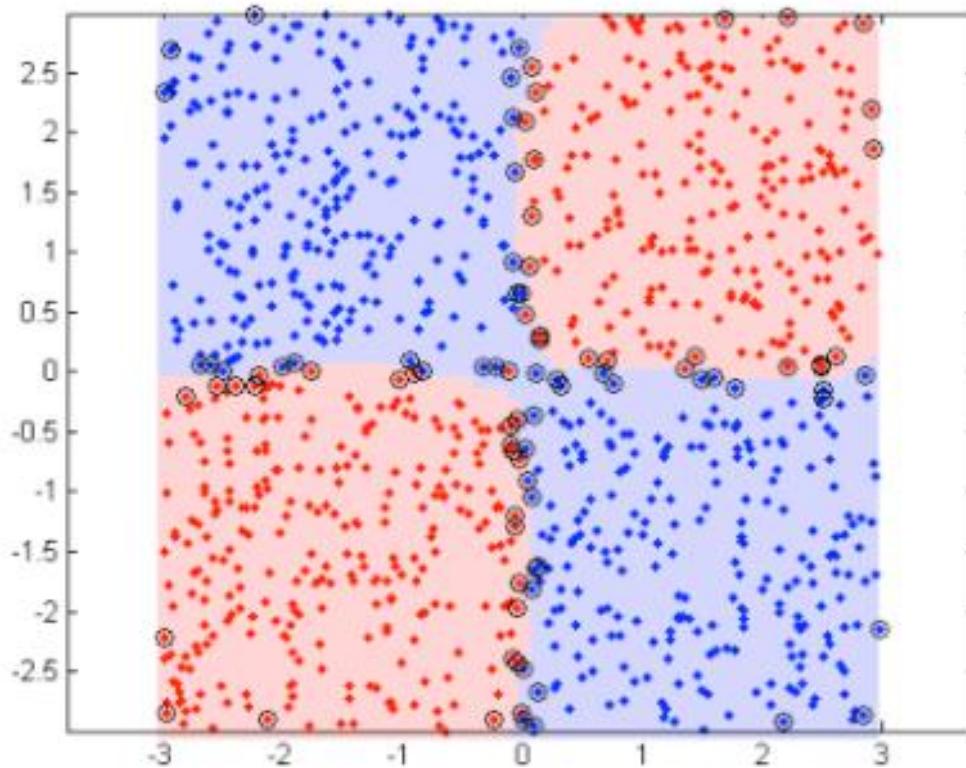


<https://developers.google.com/machine-learning/data-prep/transform/bucketing>

Feature crosses

- **Feature crosses** are a unique way to combine two or more categorical attributes into a *single* one.
- This is extremely useful a technique, when certain features *together* denote a property better than individually by themselves.
 - Mathematically speaking, you are doing a cross product between all possible values of the categorical features.

Example

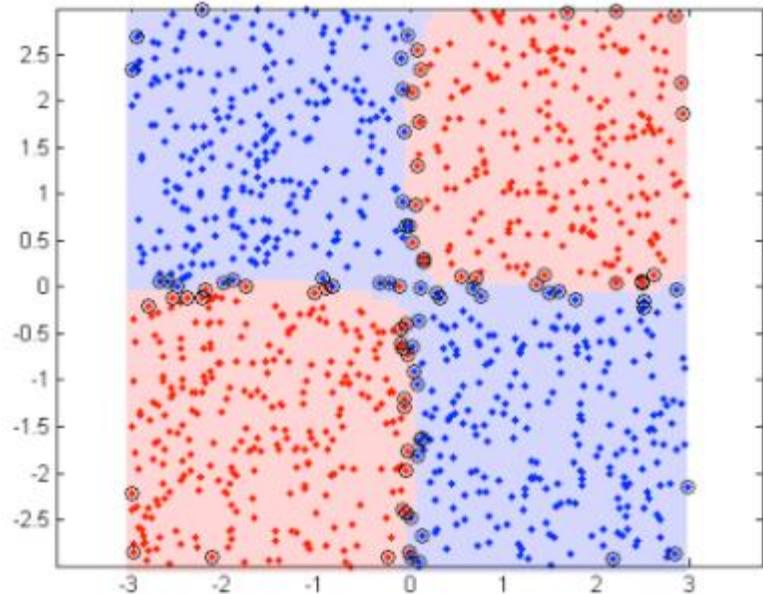


Feature crosses

- Consider a feature A , with two possible values $\{A1, A2\}$. Let B be a feature with possibilities $\{B1, B2\}$. Then, a feature-cross between A & B (lets call it AB) would take one of the following values: $\{(A1, B1), (A1, B2), (A2, B1), (A2, B2)\}$.

Example

- First off, you would benefit from binning the X, Y values into $\{x < 0, x \geq 0\} \text{ & } \{y < 0, y \geq 0\}$ respectively.
- Lets call them $\{X_n, X_p\}$ and $\{Y_n, Y_p\}$.
- It is pretty obvious that **Quadrants I & III** correspond to class Red, and **Quadrants II & IV** contain class Blue.



So if you could now cross features X and Y into a single feature ‘Quadrant’, you would basically have $\{I, II, III, IV\}$ being equivalent to $\{(X_p, Y_p), (X_n, Y_p), (X_n, Y_n), (X_p, Y_n)\}$ respectively.

Example

- suppose you define modified features X_{sign} and Y_{sign} as follows:
- Now, you could just define a new feature as follows: $X_{\text{sign}} = \frac{x}{|x|}$ and $Y_{\text{sign}} = \frac{y}{|y|}$
-
- $\text{Quadrant}_{\text{odd}} = X_{\text{sign}} Y_{\text{sign}}$
- Thats all! If $\text{Quadrant}_{\text{odd}} = 1$ the class is Red. Else, Blue!

Feature selection

- Feature selection... is the *process of selecting a subset of relevant features for use in model construction*
- Feature selection is another key part of the applied machine learning process, like model selection.
- It is important to consider feature selection a part of the model selection process. If you do not, you may inadvertently introduce bias into your models which can result in overfitting.

Feature selection

- Feature selection is different from dimensionality reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction method do so by creating new combinations of attributes, where as feature selection methods include and exclude attributes present in the data without changing them.

Feature Selection

- **Feature Selection** : Using certain algorithms to automatically select a subset of your original features, for your final model.
- Here, you are not creating/modifying your current features, but rather pruning them to reduce noise/redundancy.

Voir: <http://jmlr.csail.mit.edu/papers/volume3/guyon03a/guyon03a.pdf>

Feature selection

- Feature selection is itself useful, but it mostly acts as a filter, muting out features that aren't useful in addition to your existing features.
- Feature selection methods can be used to identify and remove unneeded, irrelevant and redundant attributes from data that do not contribute to the accuracy of a predictive model or may in fact decrease the accuracy of the model.

Objective

- The objective of variable selection is *three-fold*:
 1. Improving the prediction performance of the predictors,
 2. Providing faster and more cost-effective predictors,
 3. and providing a better understanding of the underlying process that generated the data.

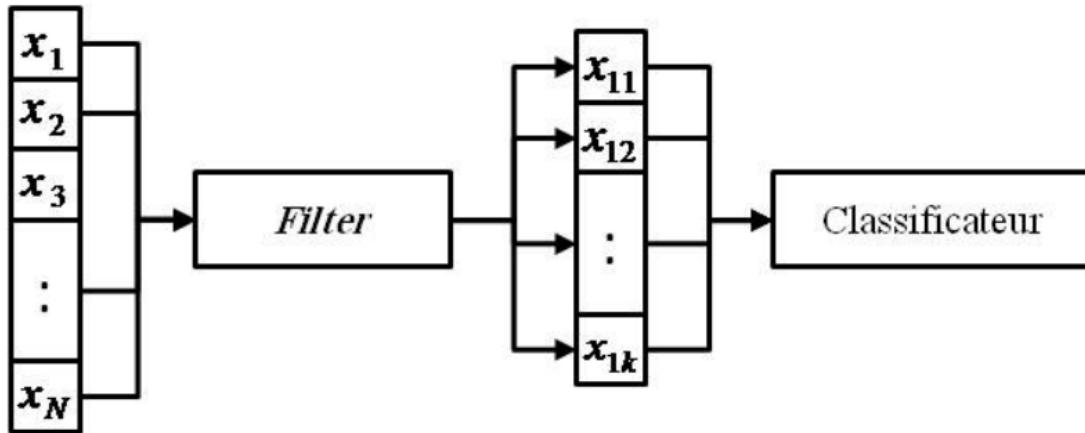
Feature Selection Algorithms

- There are three general classes of feature selection algorithms:
 1. filter methods,
 2. wrapper methods
 3. and embedded methods.

Filter feature selection

- Filter feature selection methods apply a statistical measure to assign a scoring to each feature.
- The features are ranked by the score and either selected to be kept or removed from the dataset.
- The methods are often univariate and consider the feature independently, or with regard to the dependent variable.
 - Some examples of some filter methods include the Chi squared test, information gain and correlation coefficient scores.

Filter feature selection



Consider a set of m examples $\{x_k, y_k\}$ ($k = 1, \dots, m$) consisting of n input variables $x_{k,i}$ ($i = 1, \dots, n$) and one output variable y_k . Variable ranking makes use of a scoring function $S(i)$ computed from the values $x_{k,i}$ and y_k , $k = 1, \dots, m$.

Scores

Correlation Criteria

where the bar notation stands for an average over the i

$$R(i) = \frac{\sum_{k=1}^m (x_{k,i} - \bar{x}_i)(y_k - \bar{y})}{\sqrt{\sum_{k=1}^m (x_{k,i} - \bar{x}_i)^2 \sum_{k=1}^m (y_k - \bar{y})^2}} ,$$

Fisher Criteria

μ_{jk} k and σ_{jk} the mean and standard deviation of k-th class, corresponding to the j-th feature. Let μ_j and σ_j denote the mean and standard deviation of the whole data set corresponding to the j-th feature.

Information Theoretic Ranking Criteria

The probabilities are then estimated from frequency counts

$$F(i) = \frac{\sum_{c=1}^C n_c (\mu_c^i - \mu^i)^2}{\sum_{c=1}^C n_c (\sigma_c^i)^2}$$

SNR(Signal-to-Noise Ratio coefficient)

Golub, T. R., et al. (1999). Molecular classification of cancer : class discovery and class prediction by gene expression monitoring. Science, 286:531–537.

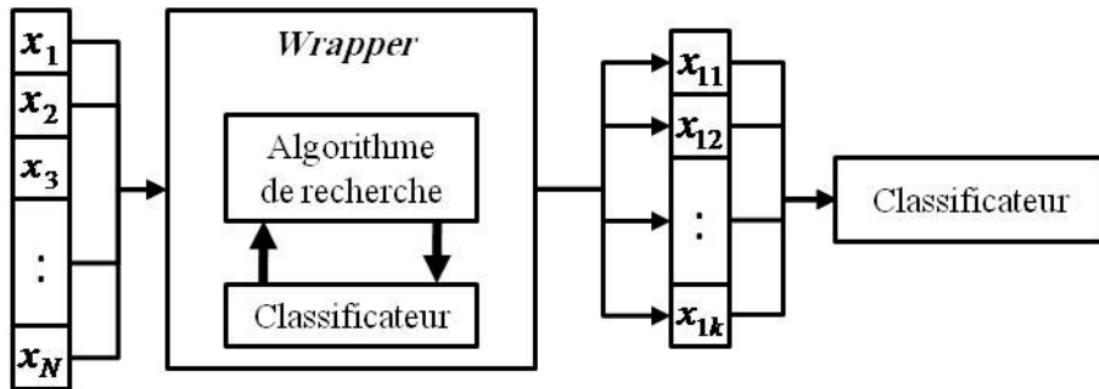
$$SNR(i) = \frac{2 \times |\mu_{C_{i1}} - \mu_{C_{i2}}|}{(\sigma_{C_{i1}} + \sigma_{C_{i2}})}$$

$$I(i) = \sum_{x_i} \sum_y P(X = x_i, Y = y) \log \frac{P(X = x_i, Y = y)}{P(X = x_i)P(Y = y)}$$

Wrapper methods

- Wrapper methods consider the selection of a set of features as a **search problem**, where different combinations are prepared, evaluated and compared to other combinations.
- A predictive model us used to **evaluate** a combination of features and assign a score based on model **accuracy**.
- The search process may be methodical such as a **best-first search**, it may **stochastic** such as a random hill-climbing algorithm, or it may use **heuristics**, like forward and backward passes to add and remove features.
 - An example if a wrapper method is the recursive feature elimination algorithm.

Wrapper methods



Embedded Methods

- Embedded methods learn which features best contribute to the **accuracy** of the model while the model is being created.
- The most common type of embedded feature selection methods are **regularization** methods.
- Regularization methods are also called **penalization** methods that introduce additional constraints into the **optimization** of a predictive algorithm (such as a regression algorithm) that bias the model toward lower complexity (fewer coefficients).
 - Examples of regularization algorithms are the LASSO, Elastic Net and Ridge Regression.

Feature Selection Checklist

1. **Do you have domain knowledge?** If yes, construct a better set of “ad hoc” features
2. **Are your features commensurate?** If no, consider normalizing them.
3. **Do you suspect interdependence of features?** If yes, expand your feature set by constructing conjunctive features or products of features, as much as your computer resources allow you.
4. **Do you need to prune the input variables (e.g. for cost, speed or data understanding reasons)?** If no, construct disjunctive features or weighted sums of feature
5. **Do you need to assess features individually (e.g. to understand their influence on the system or because their number is so large that you need to do a first filtering)?** If yes, use a variable ranking method; else, do it anyway to get baseline results.
6. **Do you need a predictor?** If no, stop

Feature Selection Checklist

7. **Do you suspect your data is “dirty” (has a few meaningless input patterns and/or noisy outputs or wrong class labels)?** If yes, detect the outlier examples using the top ranking variables obtained in step 5 as representation; check and/or discard them.
8. **Do you know what to try first?** If no, use a linear predictor. Use a forward selection method with the “probe” method as a stopping criterion or use the 0-norm embedded method for comparison, following the ranking of step 5, construct a sequence of predictors of same nature using increasing subsets of features. Can you match or improve performance with a smaller subset? If yes, try a non-linear predictor with that subset.
9. **Do you have new ideas, time, computational resources, and enough examples?** If yes, compare several feature selection methods, including your new idea, correlation coefficients, backward selection and embedded methods. Use linear and non-linear predictors. Select the best approach with model selection
10. **Do you want a stable solution (to improve performance and/or understanding)?** If yes, subsample your data and redo your analysis for several “bootstrap”.

Example

- The `sklearn.feature_selection` module implements feature selection algorithms. It currently includes univariate filter selection methods and the recursive feature elimination algorithm.

sklearn.feature selection

<code>feature_selection.GenericUnivariateSelect ([...])</code>	Univariate feature selector with configurable strategy.
<code>feature_selection.SelectPercentile ([...])</code>	Select features according to a percentile of the highest scores.
<code>feature_selection.SelectKBest ([score_func, k])</code>	Select features according to the k highest scores.
<code>feature_selection.SelectFpr ([score_func, alpha])</code>	Filter: Select the pvalues below alpha based on a FPR test.
<code>feature_selection.SelectFdr ([score_func, alpha])</code>	Filter: Select the p-values for an estimated false discovery rate
<code>feature_selection.SelectFromModel (estimator)</code>	Meta-transformer for selecting features based on importance weights.
<code>feature_selection.SelectFwe ([score_func, alpha])</code>	Filter: Select the p-values corresponding to Family-wise error rate
<code>feature_selection.RFE (estimator[, ...])</code>	Feature ranking with recursive feature elimination.
<code>feature_selection.RFECV (estimator[, step, ...])</code>	Feature ranking with recursive feature elimination and cross-validated selection of the best number of features.
<code>feature_selection.VarianceThreshold ([threshold])</code>	Feature selector that removes all low-variance features.
<code>feature_selection.chi2 (X, y)</code>	Compute chi-squared stats between each non-negative feature and class.
<code>feature_selection.f_classif (X, y)</code>	Compute the ANOVA F-value for the provided sample.
<code>feature_selection.f_regression (X, y[, center])</code>	Univariate linear regression tests.
<code>feature_selection.mutual_info_classif (X, y)</code>	Estimate mutual information for a discrete target variable.
<code>feature_selection.mutual_info_regression (X, y)</code>	Estimate mutual information for a continuous target variable.

Removing features with low variance

- VarianceThreshold is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples.

Removing features with low variance

- Suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples. Boolean features are Bernoulli random variables, and the variance of such variables is given by: $\text{Var}[X] = p(1-p)$
- so we can select using the threshold $.8 * (1 - .8)$

EXAMPLE

```
Entrée [1]: In[1]: from sklearn.feature_selection import VarianceThreshold
X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
sel.fit_transform(X)

Out[1]: array([[0, 1],
               [1, 0],
               [0, 0],
               [1, 1],
               [1, 0],
               [1, 1]])
```

As expected, `VarianceThreshold` has removed the first column, which has a probability $p=5/6 > .8$ of containing a zero.

The Recursive Feature Elimination (RFE)

- The Recursive Feature Elimination (RFE) method is a feature selection approach. It works by recursively removing attributes and building a model on those attributes that remain. It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.

The Recursive Feature Elimination (RFE)

This recipe shows the use of RFE on the Iris floweres dataset to select 3 attributes.

```
Entrée [5]: # Recursive Feature Elimination
from sklearn import datasets
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
# load the iris datasets
dataset = datasets.load_iris()
# create a base classifier used to evaluate a subset of attributes
model = LogisticRegression()
# create the RFE model and select 3 attributes
rfe = RFE(model, 3)
rfe = rfe.fit(dataset.data, dataset.target)
# summarize the selection of the attributes
print(rfe.support_)
print(rfe.ranking_)
```

```
[False  True  True  True]
[2 1 1 1]
```

Feature Importance

- Methods that use ensembles of **decision trees** (like Random Forest or Extra Trees) can also compute the **relative importance of each attribute**. These importance values can be used to inform a feature selection process.

Feature Importance

This code shows the construction of an Extra Trees ensemble of the iris flowers dataset and the display of the relative feature importance.

```
Entrée [6]: # Feature Importance
from sklearn import datasets
from sklearn import metrics
from sklearn.ensemble import ExtraTreesClassifier
# Load the iris datasets
dataset = datasets.load_iris()
# fit an Extra Trees model to the data
model = ExtraTreesClassifier()
model.fit(dataset.data, dataset.target)
# display the relative importance of each attribute
print(model.feature_importances_)
```

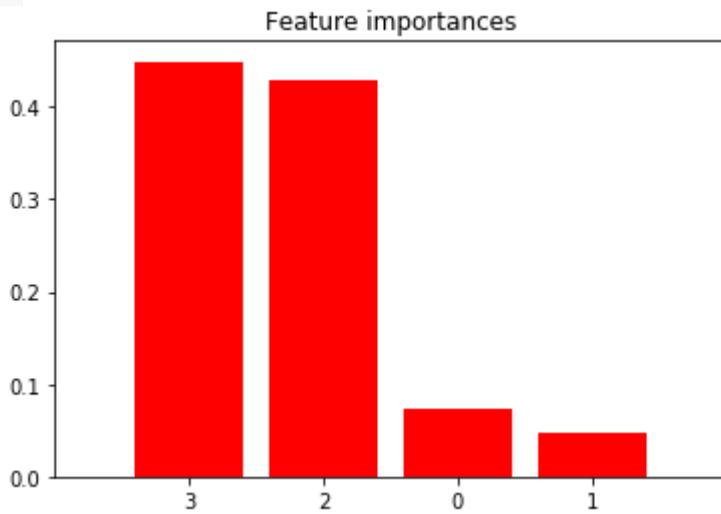
```
[0.06237856 0.04333641 0.57610379 0.31818124]
```

Feature Importance

```
importances = model.feature_importances_
std = np.std([model.feature_importances_ for tree in model.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")
X=dataset.data
for f in range(X.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
        color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.show()
```



Feature scaling

- **Feature scaling** is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

Standardization

- The result of **standardization** (or **Z-score normalization**) is that the features will be rescaled so that they'll have the properties of a standard normal distribution with $\mu=0$ and $\sigma=1$
- where μ is the mean (average) and σ is the standard deviation from the mean; standard scores (also called **z** scores) of the samples are calculated as follows:

$$z = \frac{x - \mu}{\sigma}$$

Standardizing the features

- Standardizing the features so that they are centered around 0 with a standard deviation of 1 is not only important if we are comparing measurements that have different units, but it is also a general requirement for many machine learning algorithms.

Standardizing the features

- with features being on different scales, certain weights may update faster than others since the feature values x_i play a role in the weight updates

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_i (t^{(i)} - o^{(i)}) x_j^{(i)},$$

- so that
- $w_i := w_i + \Delta w_i$, where η is the learning rate, t the target class label, and o the actual output.

Standardizing the features

- Some examples of algorithms where feature scaling matters are:
 - k-nearest neighbors with an Euclidean distance measure if want all features to contribute equally
 - k-means (see k-nearest neighbors)
 - logistic regression, SVMs, perceptrons, neural networks etc. if you are using gradient descent/ascent-based optimization, otherwise some weights will update much faster than others
 - linear discriminant analysis, principal component analysis, kernel principal component analysis since you want to find directions of maximizing the variance

Min-Max scaling

- In this approach, the data is scaled to a fixed range - usually 0 to 1.
 - The cost of having this bounded range - in contrast to standardization - is that we will end up with smaller standard deviations, which can suppress the effect of outliers.
- A Min-Max scaling is typically done via the following equation:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Exemple

$$180 - 160 = 20 / 40 = 0.5$$

- For example, suppose that we have the students' weight data, and the students' weights span [160 pounds, 200 pounds]. To rescale this data, we first subtract 160 from each student's weight and divide the result by 40 (the difference between the maximum and minimum weights).

Mean normalization

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

“Standardization or Min-Max scaling?”

- There is no obvious answer to this question: it really depends on the application.
 - For example, in clustering analyses, standardization may be especially crucial in order to compare similarities between features based on certain distance measures
 - A typical neural network algorithm require data that on a 0-1 scale.



Standardizing and normalizing - how it can be done using scikit-learn

- See Tuto [Scaling.py](#)

DataSet

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Wine Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Using chemical analysis determine the origin of wines



Data Set Characteristics:	Multivariate	Number of Instances:	178	Area:	Physical
Attribute Characteristics:	Integer, Real	Number of Attributes:	13	Date Donated	1991-07-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	1078813

Source:

Original Owners:

Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno,
16147 Genoa, Italy.

Donor:

Stefan Aeberhard, email: stefan '@' coral.cs.jcu.edu.au

<http://archive.ics.uci.edu/ml/datasets/Wine>

Tuto



Feature Extraction?

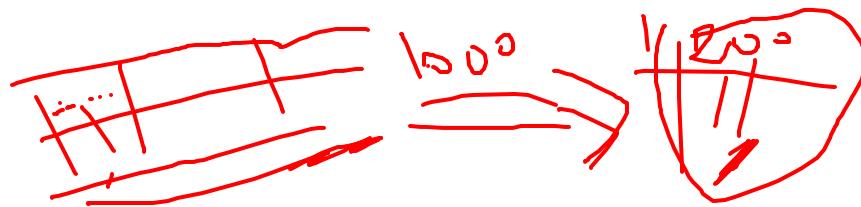


Source: <https://deeppai.org/machine-learning-glossary-and-terms/feature-extraction>

Feature Extraction: What is ?

- Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing.
- A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process.
 - Feature extraction is the name for methods that combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.

What are?



- Dimension Reduction refers to the process of converting a set of data having vast dimensions into data with lesser dimensions ensuring that it conveys similar information concisely.
- These techniques are typically used while solving **machine learning problems** to obtain better features for a classification or regression task.

Dimension reduction techniques

- With more variables, comes more trouble! And to avoid this trouble, **dimension reduction techniques** comes to the rescue.

Human Activity Recognition Using Smartphones Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Human Activity Recognition database built from the recordings of 30 subjects performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors.

Data Set Characteristics:	Multivariate, Time-Series	Number of Instances:	10299	Area:	Computer
Attribute Characteristics:	N/A	Number of Attributes:	561	Date Donated	2012-12-10
Associated Tasks:	Classification, Clustering	Missing Values?	N/A	Number of Web Hits:	773824

Source:

Jorge L. Reyes-Ortiz(1,2), Davide Anguita(1), Alessandro Ghio(1), Luca Oneto(1) and Xavier Parra(2)

1 - Smartlab - Non-Linear Complex Systems Laboratory

DITEN - Università degli Studi di Genova, Genoa (I-16145), Italy.

2 - CETpD - Technical Research Centre for Dependency Care and Autonomous Living

Universitat Politècnica de Catalunya (BarcelonaTech). Vilanova i la Geltrú (08800), Spain

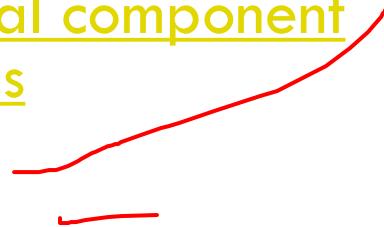
activityrecognition @' smartlab.ws

Why is this Useful?

- The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information.
- Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine's efforts in building variable combinations (features) facilitate the following learning and generalization steps in the machine learning process.

General dimensionality reduction techniques :

- Independent component analysis
- Isomap
- Kernel PCA
- Latent semantic analysis
- Partial least squares
- Principal component analysis
- Multifactor dimensionality reduction
- Nonlinear dimensionality reduction
- Multilinear Principal Component Analysis
- Multilinear subspace learning
- Semidefinite embedding
- Autoencoder



Missing data

- Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behavior and relationship with other variables correctly. It can lead to wrong prediction or classification.

The common methods to perform Dimension Reduction?

- 1. Missing Values: While exploring data, if we encounter missing values, what we do? Our first step should be to identify the reason then impute missing values/ drop variables using appropriate methods.
- But, what if we have too many missing values? Should we impute missing values or drop the variables?

Low Variance



- **Low Variance:** Let's think of a scenario where we have a constant variable (all observations have same value, 5) in our data set. Do you think, it can improve the power of model?
- Ofcourse NOT,because it has zero variance.
 - In case of high number of dimensions, we should drop variables having low variance compared to others because these variables will not explain the variation in target variables.

High Correlation

- **High Correlation:** Dimensions exhibiting higher correlation can lower down the performance of model. Moreover, it is not good to have multiple variables of similar information or variation also known as “**Multicollinearity**”.
 - You can use *Pearson* (continuous variables) or *Polychoric* (discrete variables) correlation matrix to identify the variables with high correlation and select one of them using **VIF** (Variance Inflation Factor). Variables having higher value ($VIF > 5$) can be dropped.

Backward Feature Elimination

- **Backward Feature Elimination:** In this method, we start with all n dimensions. Compute the sum of square of error (SSR) after eliminating each variable (n times). Then, identifying variables whose removal has produced the smallest increase in the SSR and removing it finally, leaving us with $n-1$ input features.
- Repeat this process until no other variables can be dropped.

Forward Feature Selection

- Reverse to this, we can use “**Forward Feature Selection**” method. In this method, we select one variable and analyse the performance of model by adding another variable. Here, selection of variable is based on higher improvement in model performance.

Factor Analysis

- Let's say some variables are highly correlated. These variables can be grouped by their correlations i.e. all variables in a particular group can be highly correlated among themselves but have low correlation with variables of other group(s).
- Here each group represents a single underlying construct or factor. These factors are small in number as compared to large number of dimensions. However, these factors are difficult to observe. There are basically two methods of performing factor analysis:
 - EFA (Exploratory Factor Analysis)
 - CFA (Confirmatory Factor Analysis)

Principal Component Analysis (PCA)

- In this technique, variables are transformed into a new set of variables, which are linear combination of original variables. These new set of variables are known as **principle components**. They are obtained in such a way that first principle component accounts for most of the possible variation of original data after which each succeeding component has the highest possible variance.

Principal Component Analysis (PCA)

- The second principal component must be orthogonal to the first principal component. In other words, it does its best to capture the variance in the data that is not captured by the first principal component. For two-dimensional dataset, there can be only two principal components.

Principal Component Analysis (PCA)

- The principal components are sensitive to the scale of measurement, now to fix this issue we should always standardize variables before applying PCA. Applying PCA to your data set loses its meaning. If interpretability of the results is important for your analysis, PCA is not the right technique for your project.

Tuto

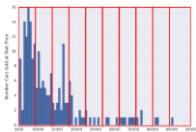


AI Feature Engineering

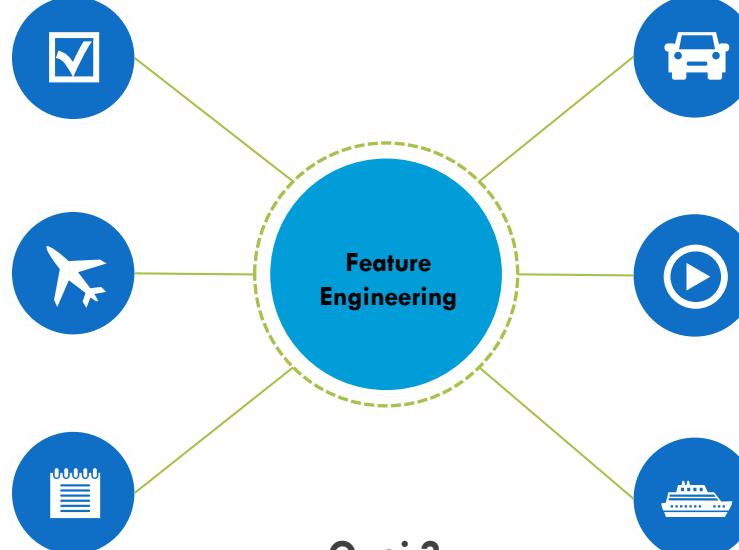
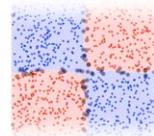
Decomposing Categorical Attributes

CompanyName	Categoricalvalue	Price
VW	1	20000
Acura	2	10011
Honda	3	50000
Honda	3	10000

Binning/Bucketing



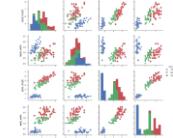
Feature Crosses



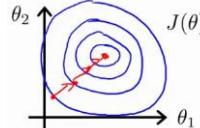
Quoi ?

distinguer les bonnes caractéristiques des mauvaises, prétraiter et transformer ces caractéristiques afin d'optimiser leur efficacité dans vos modèles

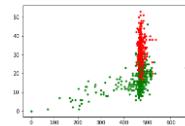
Feature Selection

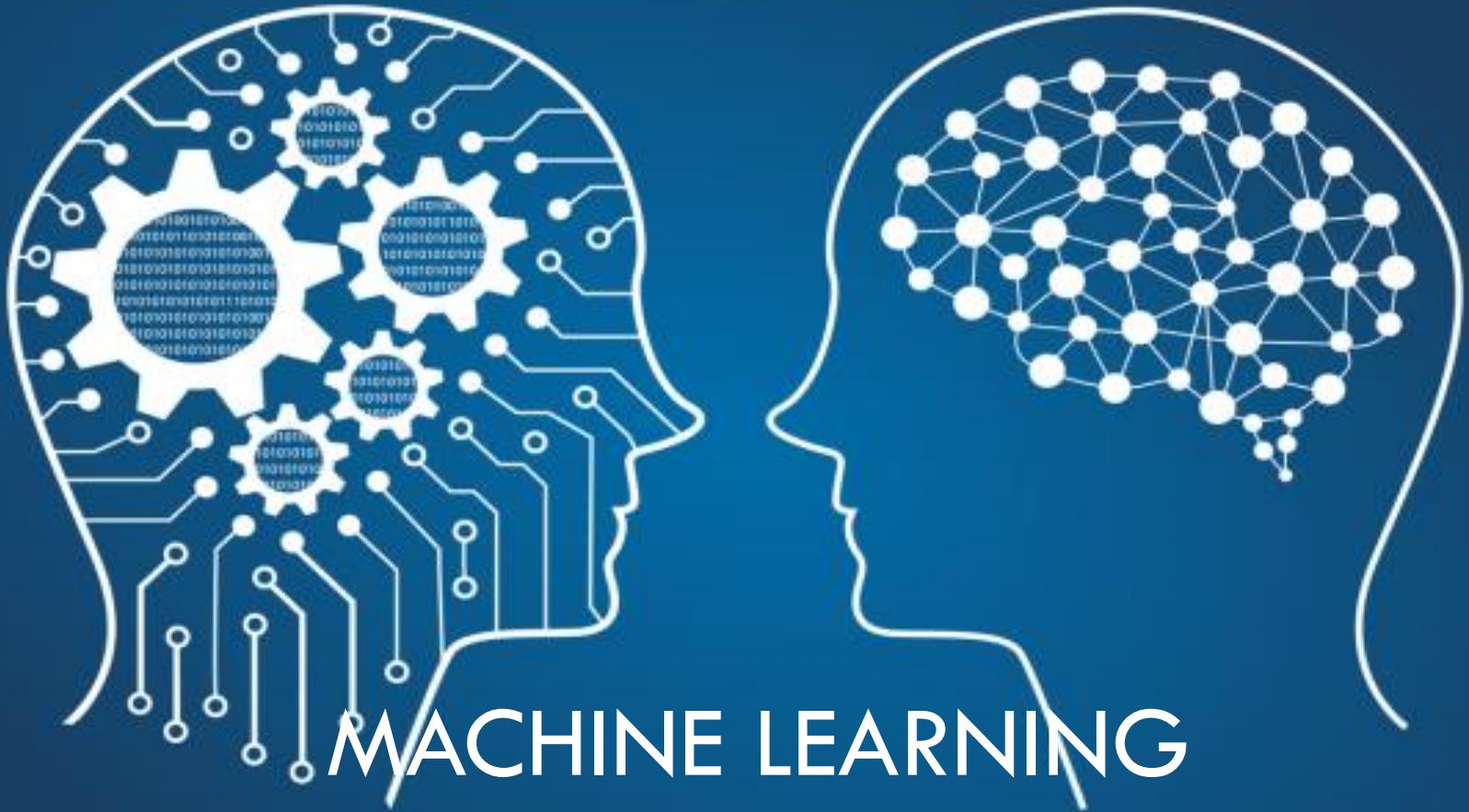


Feature Scaling (data normalization)

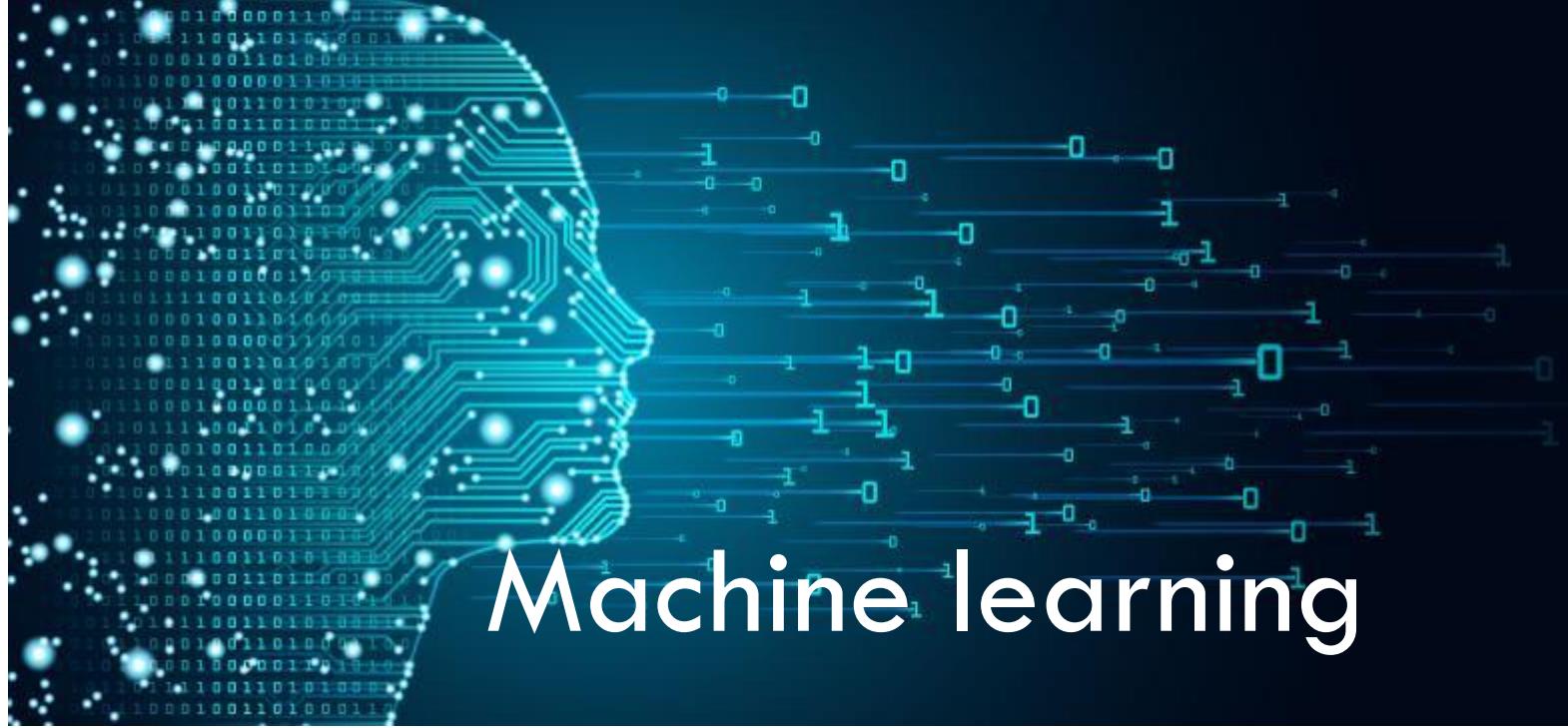


Feature Extraction





Pr. BEN LAHMAR EL Habib



Machine learning

Algorithms

Pr. BEN LAHMAR EL Habib

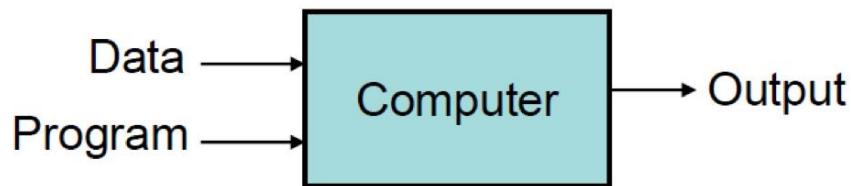


General programming vs Machine learning-model

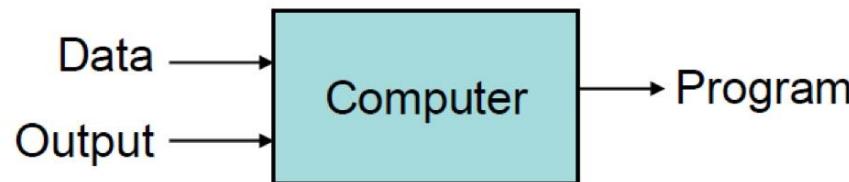
Basic Paradigm

- Observe set of examples: **training data**
- Infer something about process that generated that data
- Use inference to make predictions about previously unseen data: **test data**

Traditional Programming



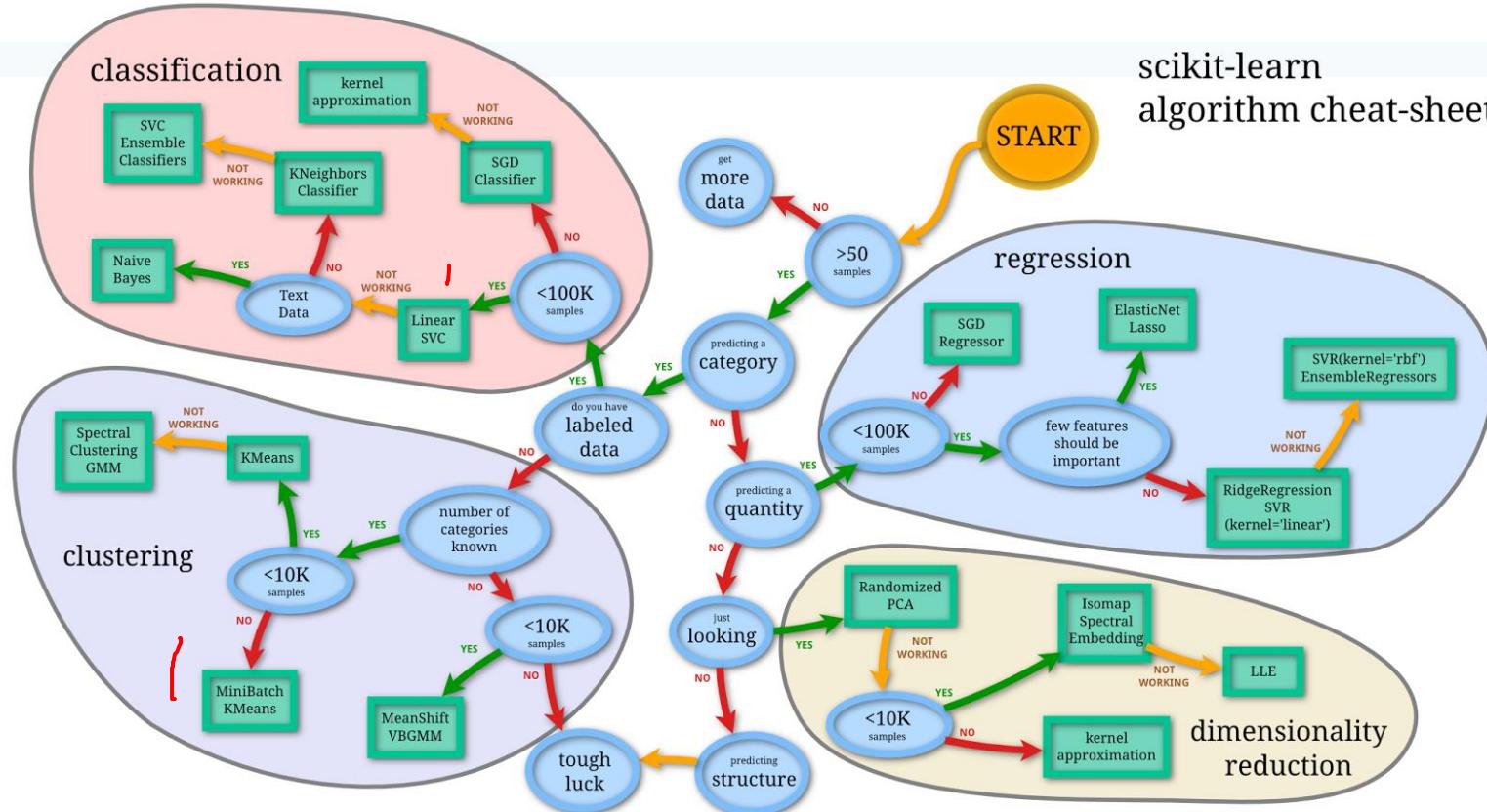
Machine Learning



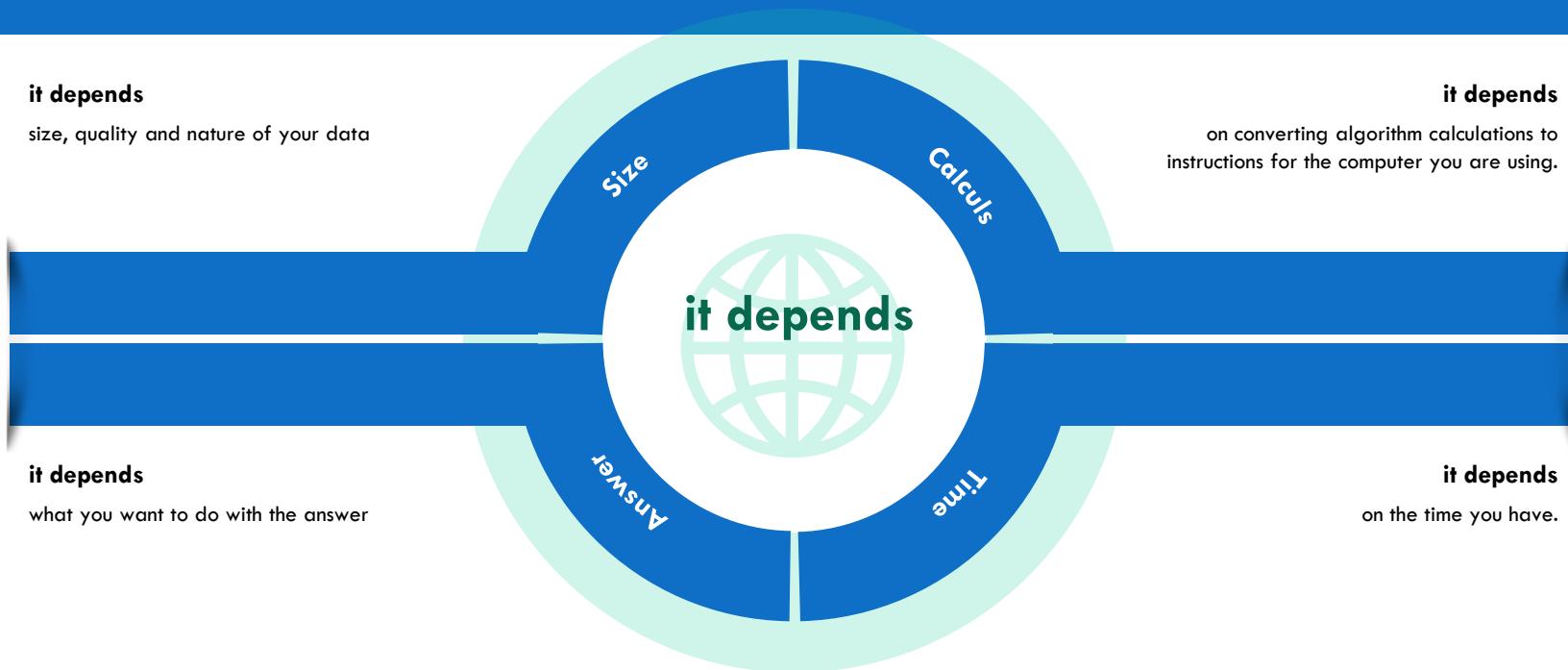
Procedures

1. Representation of the features
2. Distance metric for feature vectors
3. Objective function and constraints
4. Optimization method for learning the model
5. Evaluation method

AI Modèles



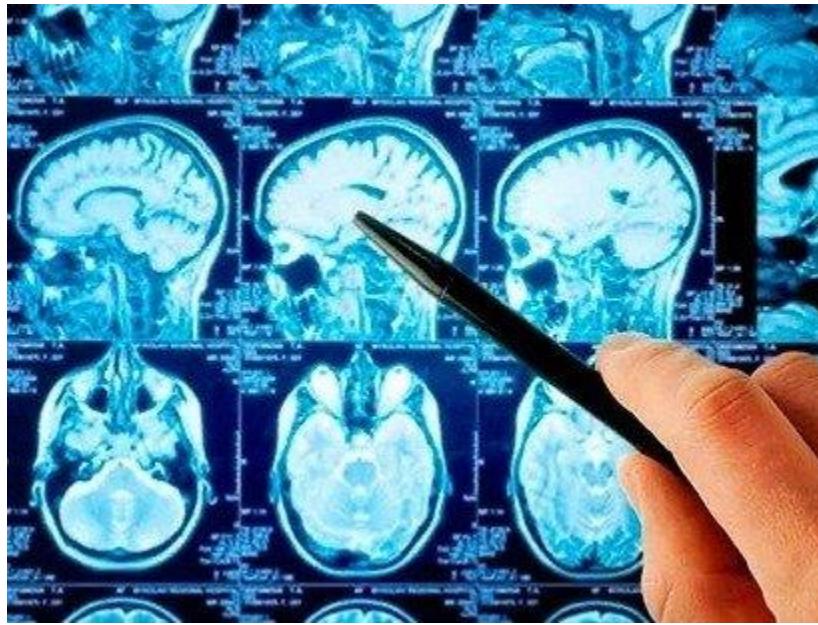
"What's the best machine learning algorithm to use? "



Self drive



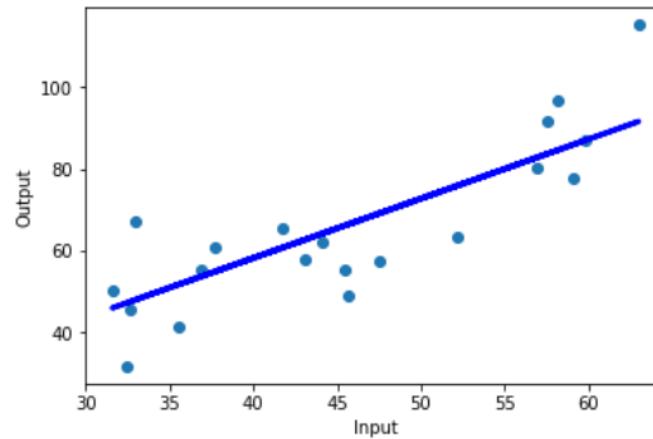
Precision





Linear Regression

- It is used to estimate **real values** (cost of houses, number of calls, total sales etc.) based on **continuous variable(s)**..



Linear Regression

- Linear regression is used for finding linear **relationship** between **target** and one or more **predictors**.
- The core idea is to obtain a **line** that best fits the data.
- The best fit line is the one for which total prediction error (all data points) are as small as possible. Error is the distance between the point to the regression line.

How

- Establish relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation $Y = a * X + b$.

$$Y' = A + B * X$$

SIMPLE REGRESSION EQUATION

- X: predictor (present in data)
- B: coefficient (estimated by regression)
- A: intercept (estimated by regression)
- Y': predicted value (calculated from A, B and X)

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \underline{\varepsilon_i}, \quad i = 1, \dots, n,$$

Types

- Linear Regression is of mainly two types:
 - Simple Linear Regression;
 - Multiple Linear Regression.

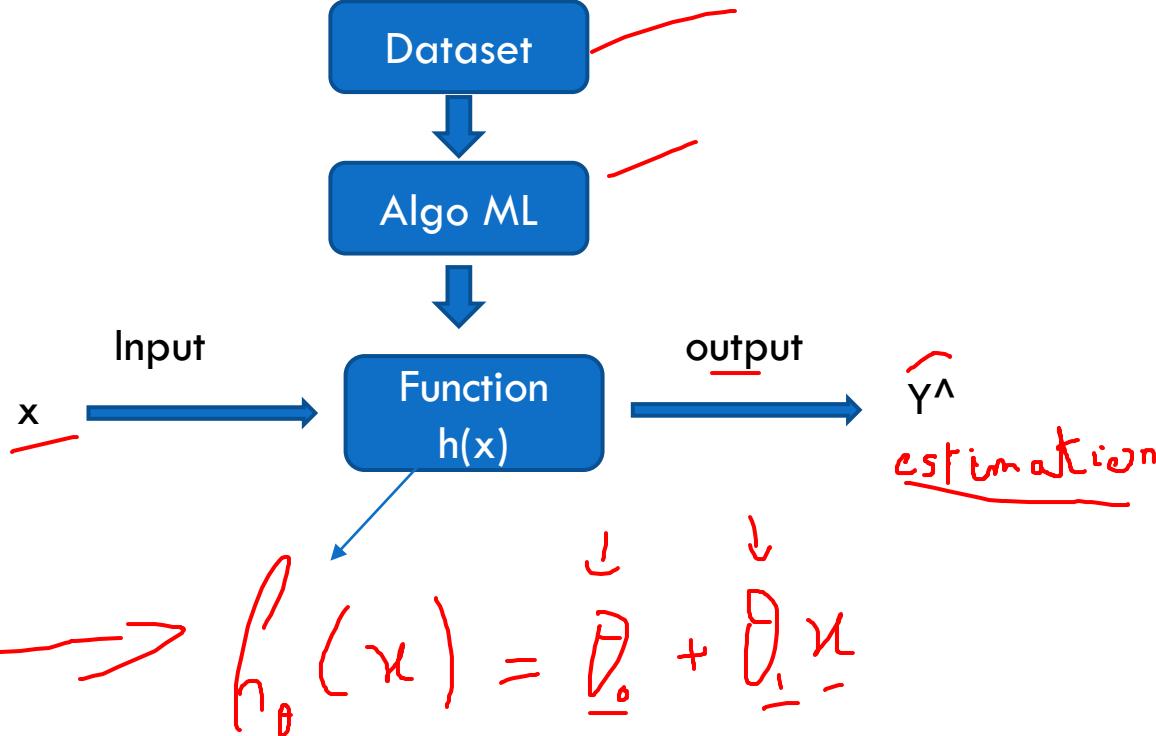
Simple Linear Regression;

- Single dimension linear regression has pairs of x and y values as input training samples.
- It uses these training sample to derive a line that predicts values of y.
- The training samples are used to derive the values of a and b that minimise the error between actual and predicated values of y

$$\hat{y} = ax + b$$

- a is the **slope** and "b" is the y-intercept

Principe



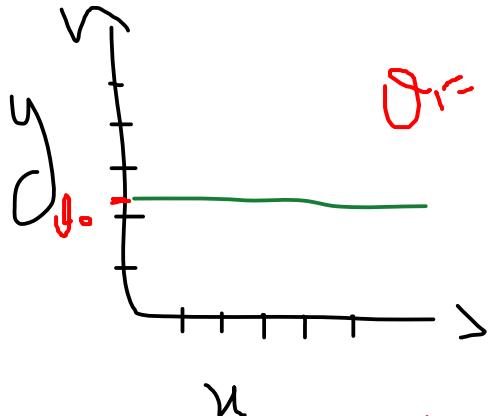
Exemple

x	y
4	12
2	8
4,5	13
5	14
6	14,5
6,5	16
7	16

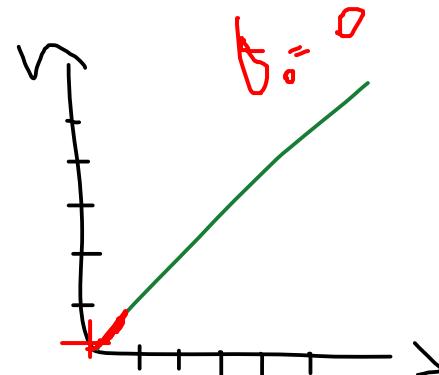
$$f_{\theta}(x) = \theta_0 + \theta_1 x$$
$$\left. \begin{array}{l} \theta_0 = ? \\ \theta_1 = ? \end{array} \right\} \Rightarrow \begin{array}{l} 12 = \theta_0 + \theta_1 \times 4 \\ 8 = \theta_0 + \theta_1 \times 2 \end{array} \Rightarrow \begin{array}{l} 4 = 2 \theta_1 \Rightarrow \theta_1 = 2 \\ \theta_0 = 4 \end{array}$$

$$h_{\theta}(x) = \underline{\theta_0} + \underline{\theta_1}x$$

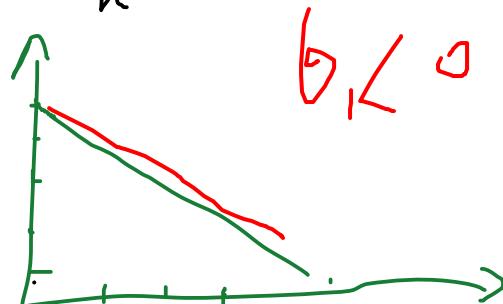
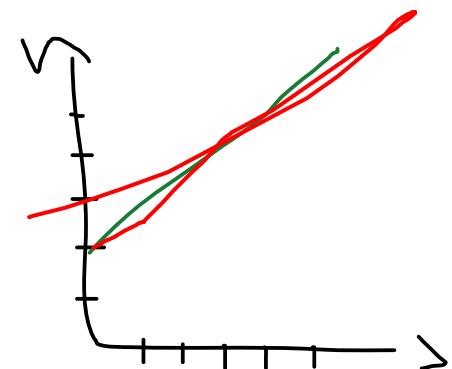
$\theta_0?$ $\theta_1?$



$$\theta_1 = 0$$



$$\theta_0 = 0$$



$$\theta_1 < 0$$

$$h_{\underline{\theta}}(x)$$

Chose $\theta_0?$ $\theta_1?$ so that $h_{\underline{\theta}}(x)$ is close to y for training dataset

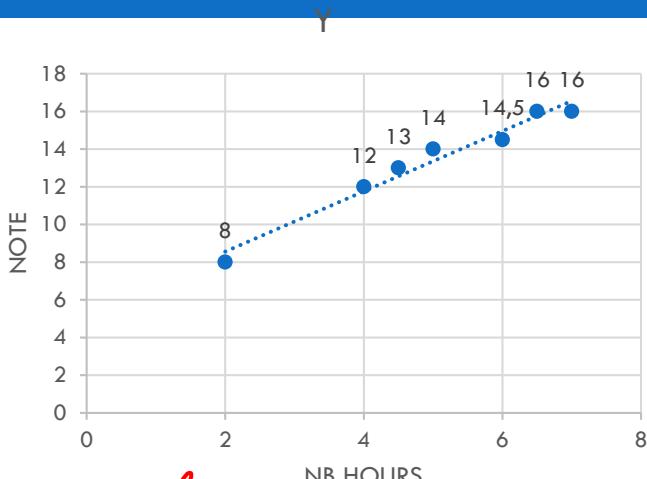
$$\theta_0 = 0$$

$$\theta_1 = 0$$

$$\theta_0 = \theta_1 = \theta_2 = \dots$$

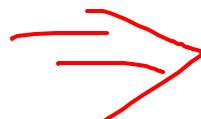
Exemple

X (nb hours)	Y (note)
4	12
2	8
4,5	13
5	14
6	14,5
6,5	16
7	16



$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

Chose θ_0 ? θ_1 ? so that
 $h(x)$ is close to y for training dataset



We want a line that minimises the error between the Y values in training samples and the $h(x)$ values that the line passes through.

So we define the error function for our algorithm so we can minimise that error.

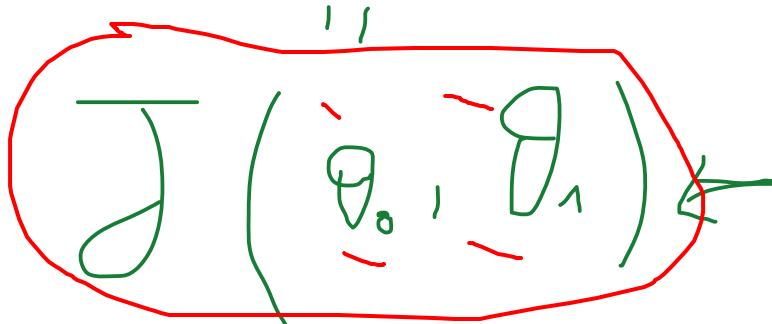
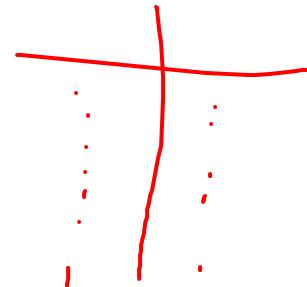
$$E = \frac{1}{n} \sum_{i=0}^n (y_i - h(x_i))^2$$

Cost function

□ Minimise

$$E = \frac{1}{n} \sum_{i=0}^n (y_i - h(x_i))^2$$

$$= \frac{1}{n} \sum_{i=0}^n \left(y_i - (\theta_0 + \theta_1 x_i) \right)^2$$



cost function

Cost Function

- The cost function helps us to figure out the best possible values for a and b which would provide the best fit line for the data points.
- Since we want the best values for a and b, we convert this search problem into a minimization problem where we would like to minimize the error between the predicted value and the actual value.

$$\text{minimize} \frac{1}{n} \sum_{i=1}^n (pred_i - y_i)^2$$

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

Minimize the error

- The values a and b must be chosen so that they minimize the error. If sum of squared error is taken as a metric to evaluate the model, then goal to obtain a line that best reduces the error.
 - Mean Absolute Error
 - Mean Square Error
 - Mean Absolute Percentage Error
 - Mean Percentage Error
 -
- Mean Absolute Error (MAE) is the mean of the absolute value of the errors
- Mean Squared Error (MSE) is the mean of the squared errors
- Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors

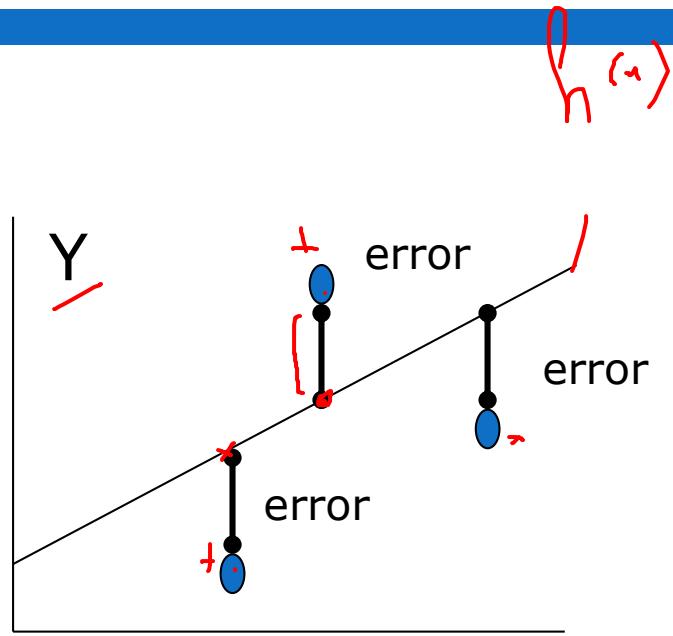
Error

$$BCE = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)$$

$$MSE = \frac{1}{N} \sum_{i=1}^N (f_i - y_i)^2$$

where N is the number of data points,
 f_i the value returned by the model and
 y_i the actual value for data point i .

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$



MSE

- We square the error difference and sum over all data points and divide that value by the total number of data points. This provides the average squared error over all the data points.

$$MSE = \frac{1}{N} \sum_{i=1}^N (f_i - y_i)^2$$

where N is the number of data points, f_i the value returned by the model and y_i the actual value for data point i .

Cost function

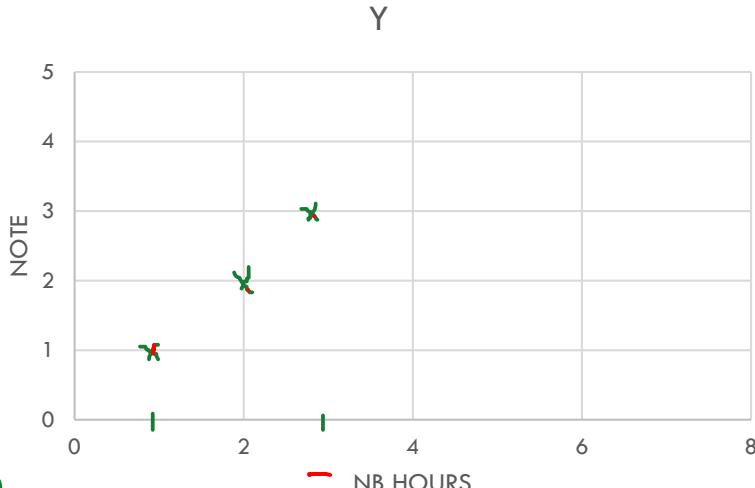
- Goal :
- Minimize

$$J(\theta_0, \theta_1)$$

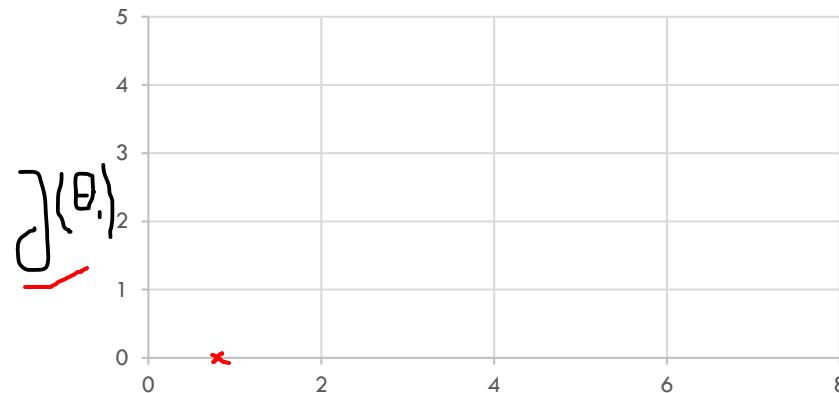
$$\frac{1}{2m} \sum_{i=0}^m \left(y_i - (\theta_0 + \theta_1 x_i) \right)^2$$

Simple example $[(1,1), (2,2), (3,3)]$

Function h



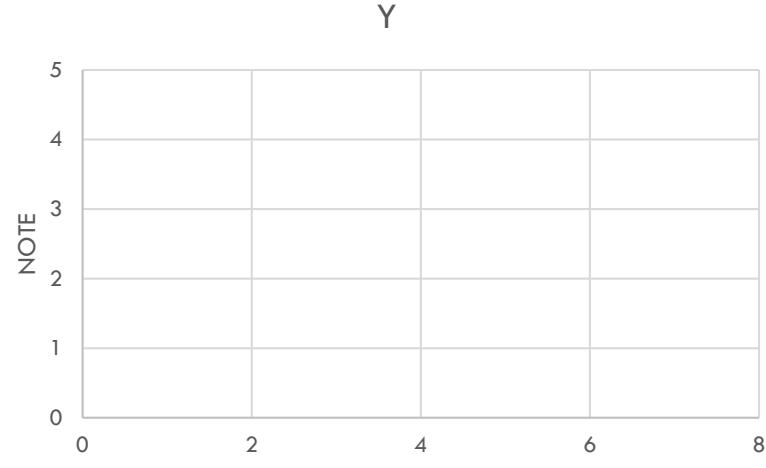
Function $J(\theta_1)$



$$J(\theta_0 = 0, \theta_1 = 1) = ?$$

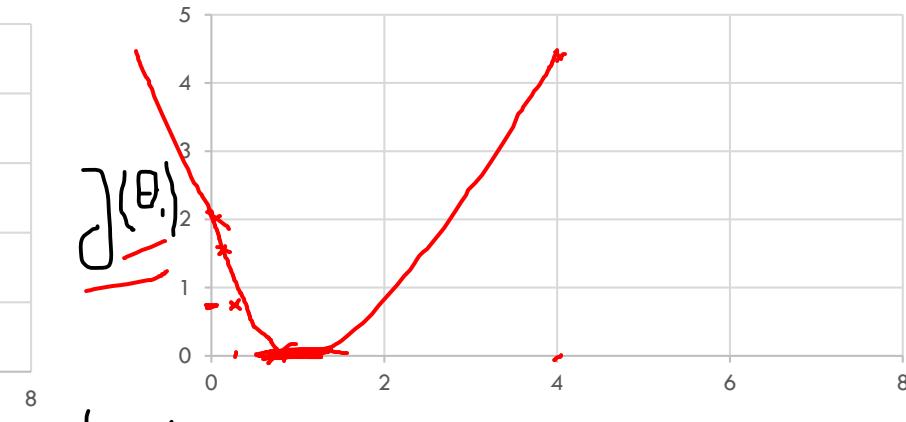
$$J(\theta_0 = 0, \theta_1 = 1) = ? \cdot \frac{1}{2} \left((1-1)^2 + (2-2)^2 \right) = 0$$

Function h



$$f \quad \theta_0 = 0 \quad \theta_1 = \underline{0.5} \rightarrow$$

Function $J(\theta_1)$

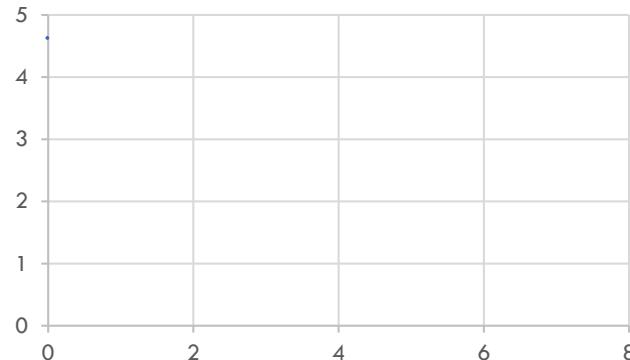


$$J(\theta_1) = ? \cdot \frac{1}{6} \left(|1 - 0.5|^2 + (2 - 1)^2 + (3 - 1)^2 \right)^{\frac{1}{2}}$$

Function h



Function $J(\theta_0, \theta_1)$

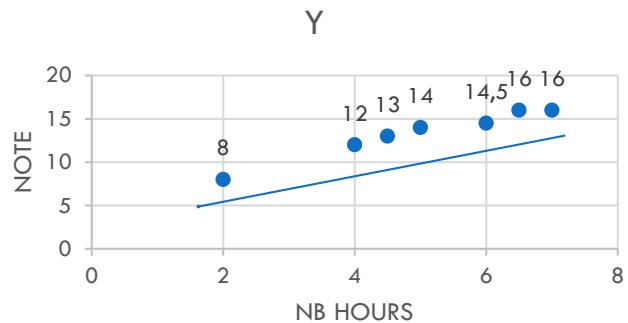


$$\left\{ \begin{array}{l} \theta_0 = 0 \\ \theta_1 = 0 \end{array} \right.$$

$$\theta_1 = 0$$

$$J(0) = ? \quad \frac{1}{2} \left(1^2 + 9^2 + 3^2 \right) = \frac{14}{2} = 7$$

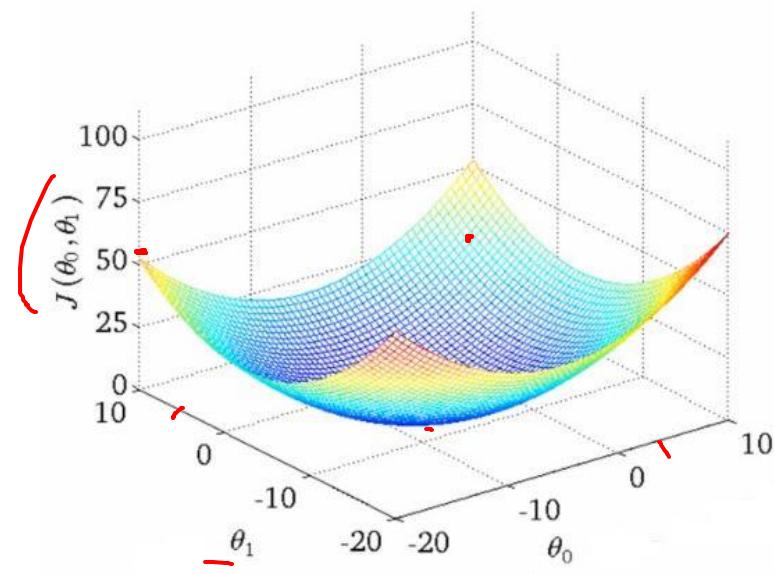
Function h

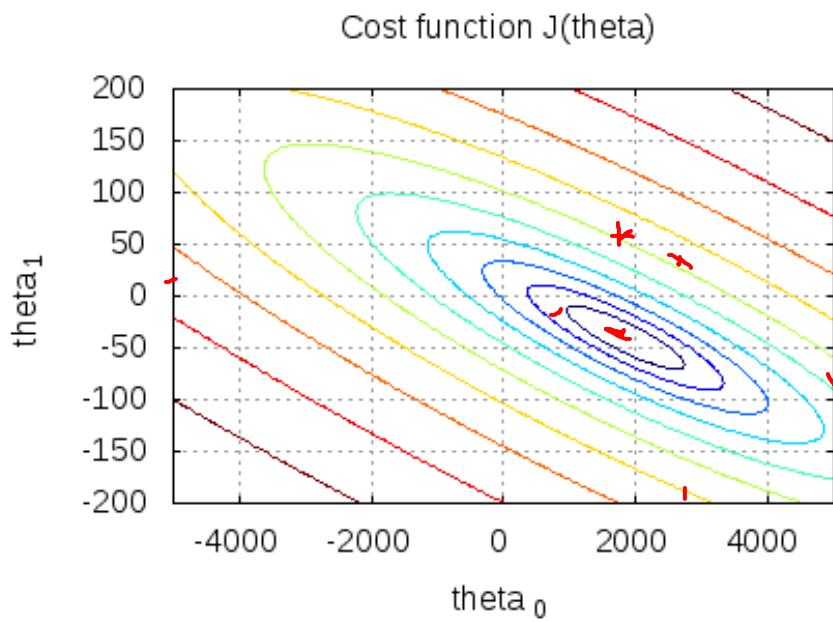
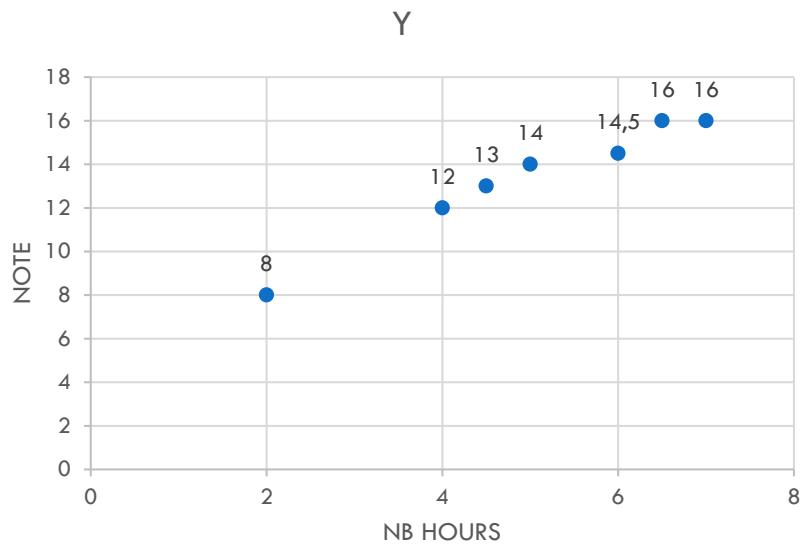


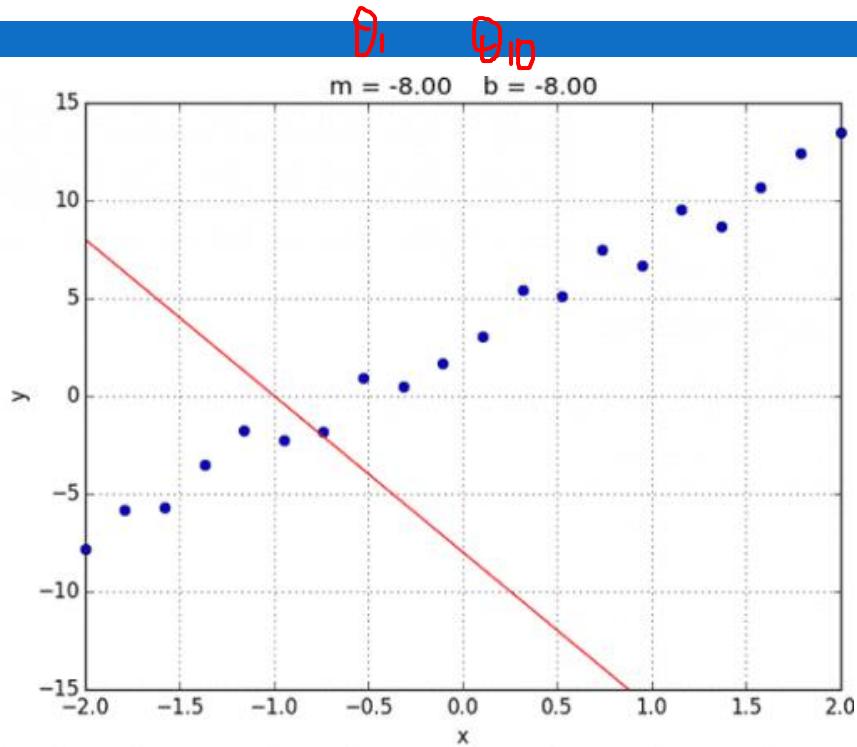
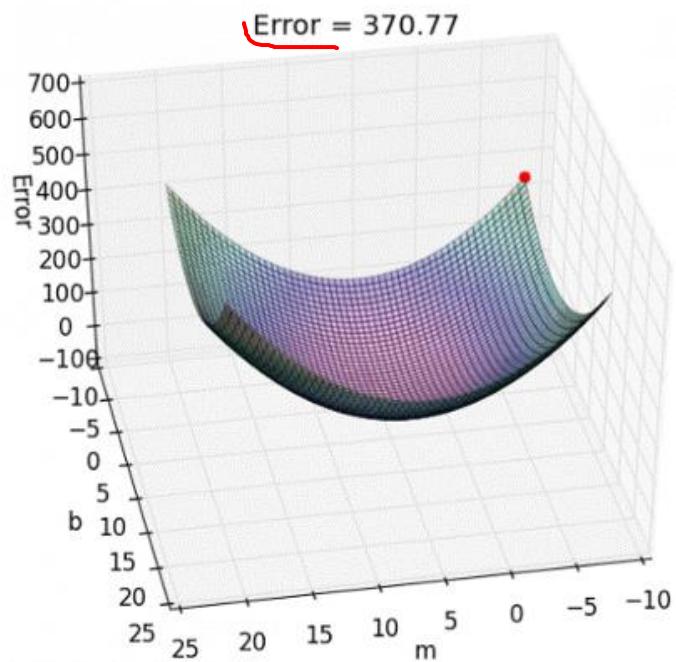
$$\theta_0 = 3$$
$$\theta_1 = 0.5$$

$$h(x) = 3 + 0.5x$$

Function $J(\theta_0, \theta_1)$



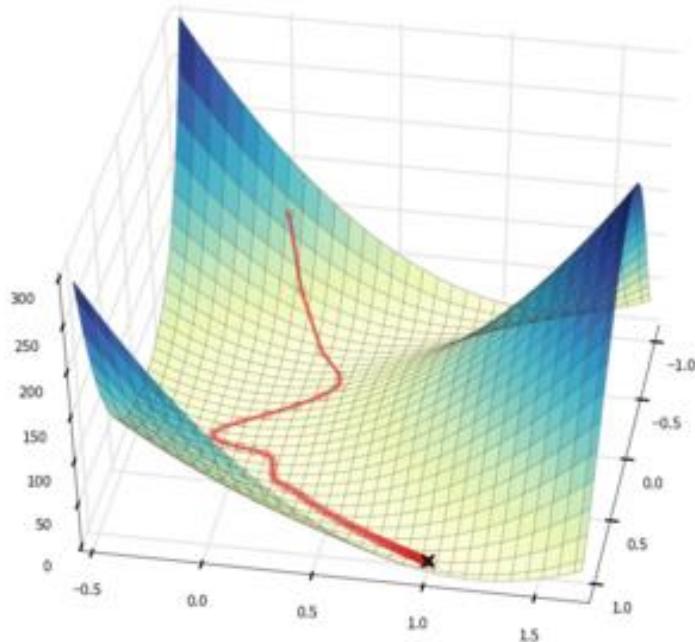




How

- To determine how to best fit our model with given set of points, we want to **minimize** the distance between **each** of these point to our linear model.
- Calculating the **totalError** helps us determine how bad our model is so we can update it every step.
- **but ...**

Minimizing the cost function: Gradient descent



Repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

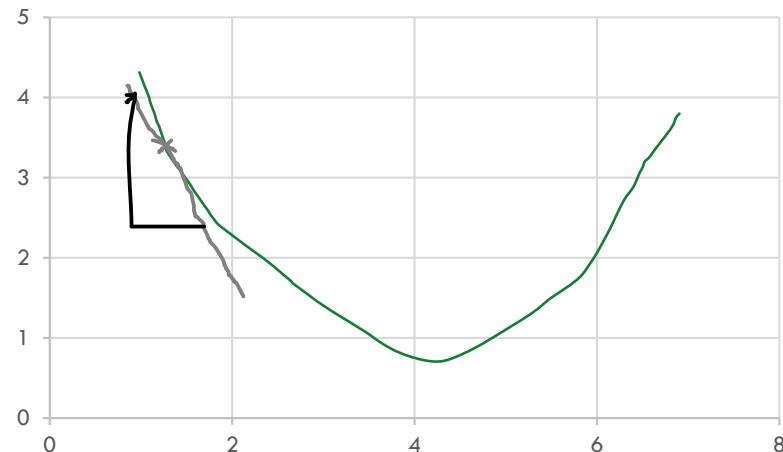
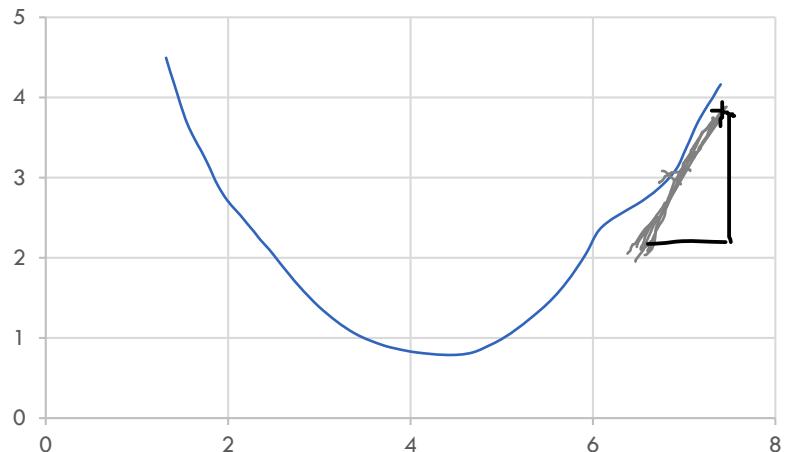
Learning Rate

Partial derivative

The derivative of a function of a real variable measures the sensitivity to change of the function value (output value) with respect to a change in its argument (input value).

Slope

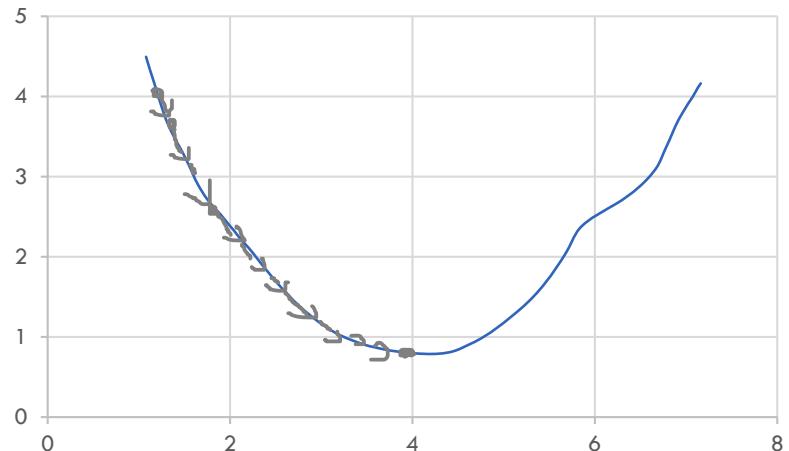
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$



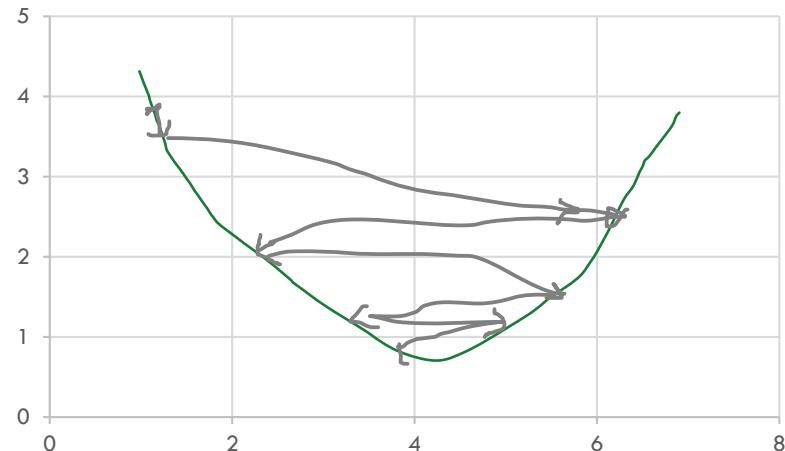
Learning rate

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

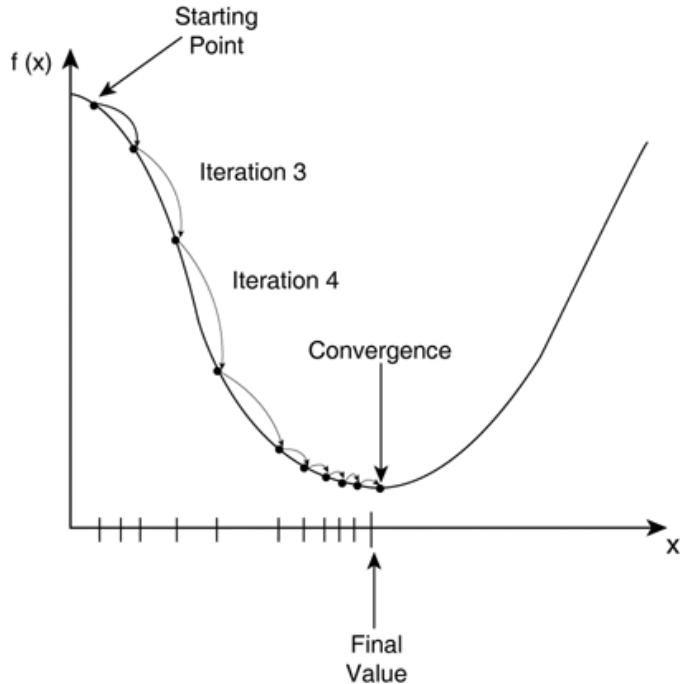
If α is too small



If α is too larger



Minimizing the cost function: Gradient descent



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

A blue bracket encloses the term $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$. A blue arrow points from this bracket to the equation $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = ?$. Another blue arrow points from the bracket to the equation $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = ?$.

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = ?$$
$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = ?$$

$$J = \frac{1}{m} \sum_{i=0}^m (h_\theta(x_i) - y_i)^2$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) =$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=0}^m \left(h_{\theta}(x_i) - y_i \right) \left(\frac{\partial}{\partial \theta_0} + \frac{\partial \theta_1}{\partial \theta_0} x_i - \frac{\partial}{\partial \theta_1} y_i \right)$$

$$= \frac{1}{m} \sum_{i=0}^m \left(h_{\theta}(x_i) - y_i \right) \times x_i$$

Update Θ_0 and Θ_1

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=0}^m (h(x_i) - y_i)$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=0}^m (h(x_i) - y_i) x_i$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=0}^m (h(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=0}^m (h(x_i) - y_i) x_i$$

Example

- Tuto

AI

Multi Dimension Linear Regression

- Each training sample has an x made up of multiple input values and a corresponding y with a single value.
- The inputs can be represented as an X matrix in which each row is sample and each column is a dimension.
- The outputs can be represented as y matrix in which each row is a sample.

$$X = \begin{bmatrix} x_{1,1} & x_{1,\dots} & x_{1,d} \\ x_{\dots,1} & x_{\dots,\dots} & x_{\dots,d} \\ x_{n,1} & x_{n,\dots} & x_{n,d} \end{bmatrix}$$

$X_{ij} \rightarrow \text{sample}_i, \text{dimension}_j$

$$y = \begin{bmatrix} y_1 \\ y_\dots \\ y_n \end{bmatrix}$$

- ❑ Our predicated y values are calculated by multiple the X matrix by a matrix of param, Θ .
- ❑ If there are 2 dimension, then this equation defines plane. If there are more dimensions then it defines a hyper-plane.

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \quad y^{\wedge} = X^* \Theta$$

Generalisation, Over-fitting & Regularisation

Implémentation

Python code

```
#Import Library

#Import other necessary libraries like pandas, numpy...
from sklearn import linear_model

#Load Train and Test datasets

#Identify feature and response variable(s) and values must be numeric and numpy arrays

x_train=input_variables_values_training_datasets
y_train=target_variables_values_training_datasets
x_test=input_variables_values_test_datasets

# Create linear regression object

linear = linear_model.LinearRegression()

# Train the model using the training sets and check score

linear.fit(x_train, y_train)

linear.score(x_train, y_train)

#Equation coefficient and Intercept

print('Coefficient: \n', linear.coef_)

print('Intercept: \n', linear.intercept_)

#Predict Output

predicted= linear.predict(x_test)
```

Implémentation

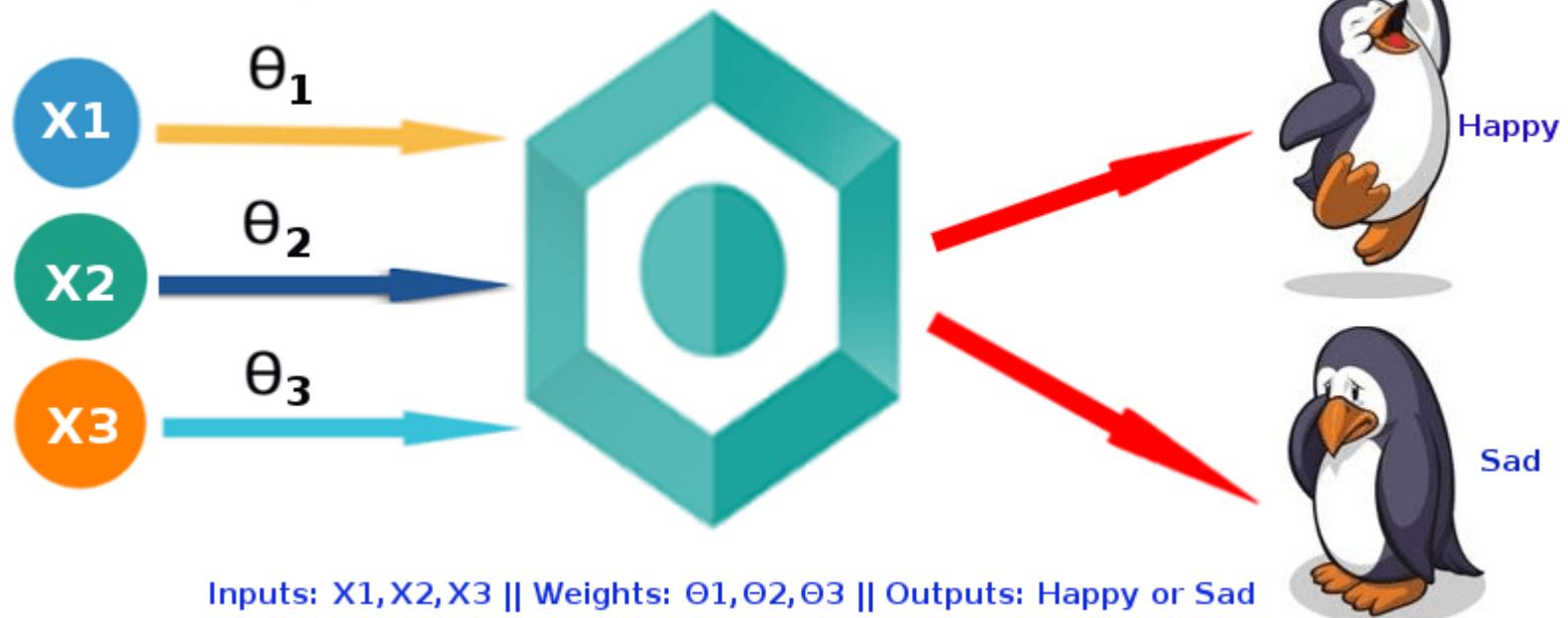
R Code

```
#Load Train and Test datasets  
  
#Identify feature and response variable(s) and values must be numeric and numpy arrays  
  
x_train <- input_variables_values_training_datasets  
y_train <- target_variables_values_training_datasets  
x_test <- input_variables_values_test_datasets  
x <- cbind(x_train,y_train)  
  
# Train the model using the training sets and check score  
  
linear <- lm(y_train ~ ., data = x)  
summary(linear)  
  
#Predict Output  
  
predicted= predict(linear,x_test)
```

Tuto



Logistic Regression Model

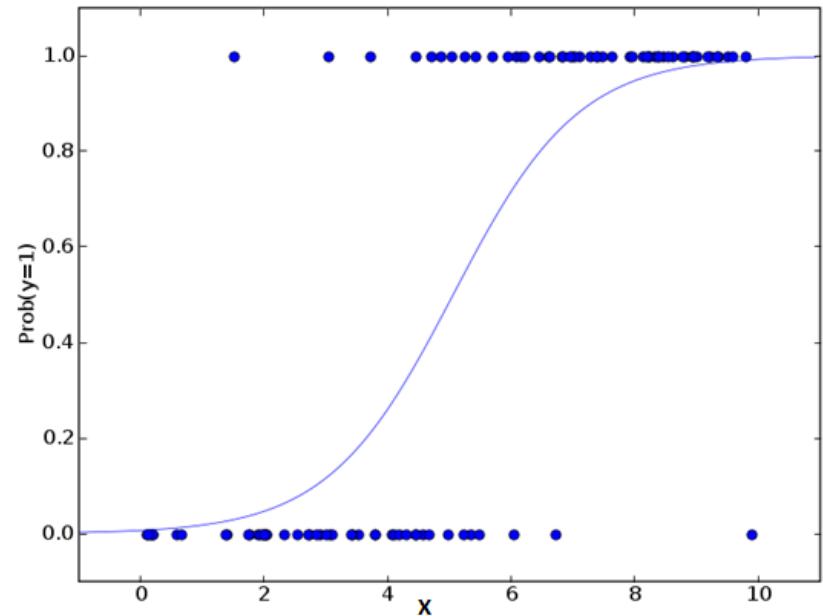


logistic regression

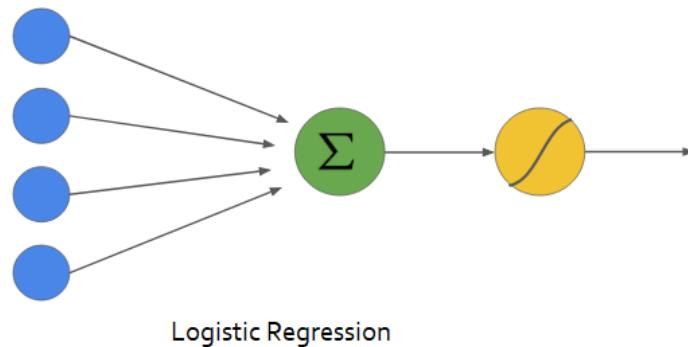
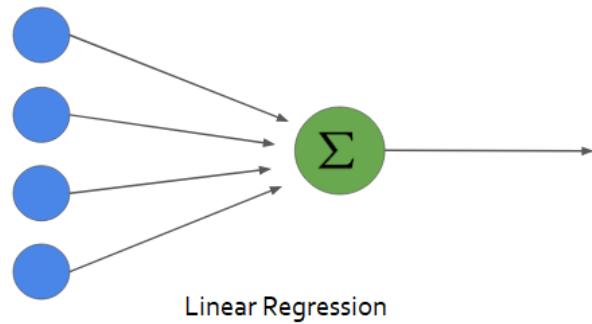
- In a lot of ways, linear regression and logistic regression are similar. But, the biggest difference lies in what they are used for.
- Linear regression algorithms are used to **predict/forecast** values but logistic regression is used for **classification tasks**.

Logistic Regression

- It is a classification not a regression algorithm.
- It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variable(s).



LiR vs LoR



Logistic Regression

- Logistic Regression is used when the **dependent** variable(target) is categorical.
- For example,
 - To predict whether an email is spam (1) or (0)
 - Whether the tumor is malignant (1) or not (0)
 - whether a website is fraudulent (1) or not (0)

Logistic regression

- Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity.

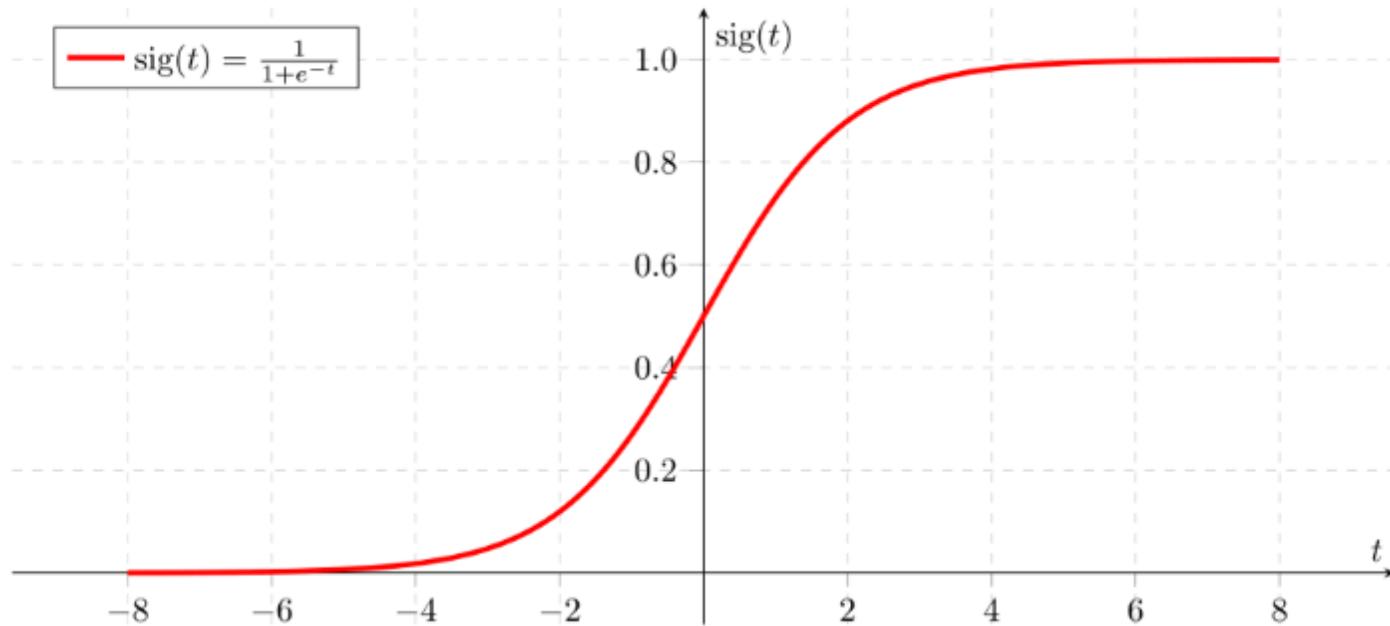
Logistic regression

- The predicted value can be anywhere between **negative infinity** to **positive infinity**. We need the output of the algorithm to be class variable, i.e 0-no, 1-yes.
- Therefore, we are squashing the output of the linear equation into a range of [0,1]. To squash the predicted value between 0 and 1, we use the sigmoid function.

Model

- Output = 0 or 1
 - Hypothesis => $Z = WX + B$
 - $h\Theta(x) = \text{sigmoid}(Z)$
-
- If 'Z' goes to infinity, Y(predicted) will become 1 and if 'Z' goes to negative infinity, Y(predicted) will become 0.

Sigmoid Activation Function



Sigmoid function

$$z = \theta_0 + \theta_1 \cdot x_1 + \theta \cdot x_2 + \dots \quad g(x) = \frac{1}{1 + e^{-x}}$$

Linear Equation and Sigmoid Function

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

Squashed output-h

Mathematically this can be written as

$$h_{\theta}(x) = P(Y=1|X; \theta)$$

Probability that $Y=1$ given X which is parameterized by 'theta'.

$$P(Y=1|X; \theta) + P(Y=0|X; \theta) = 1$$

$$P(Y=0|X; \theta) = 1 - P(Y=1|X; \theta)$$

Types of Logistic Regression

- 1. **Binary Logistic Regression**
 - The categorical response has only two possible outcomes.
Example: Spam or Not
- 2. **Multinomial Logistic Regression**
 - Three or more categories without ordering. Example: Predicting which food is preferred more (Veg, Non-Veg, Vegan)
- 3. **Ordinal Logistic Regression**
 - Three or more categories with ordering. Example: Movie rating from 1 to 5

Cost Function

- Since we are trying to predict class values, we cannot use the same cost function used in linear regression algorithm. Therefore, we use a logarithmic loss function to calculate the cost for misclassifying.

Cost Function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

- The above cost function can be rewritten as below since calculating gradients from the above equation is difficult.

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

Gradients

$$J = \frac{-1}{m} \cdot \left[\sum_{i=1}^m y_i \cdot \log h_i + (1 - y_i) \cdot \log 1 - h_i \right]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{-1}{m} \cdot \left[\sum_{i=1}^m \frac{y_i}{h_i} \cdot h_i^2 \cdot x_n \cdot \frac{1 - h_i}{h_i} + \frac{1 - y_i}{1 - h_i} \cdot -h_i^2 \cdot x_n \cdot \frac{1 - h_i}{h_i} \right]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{-1}{m} \cdot \left[\sum_{i=1}^m x_n \cdot (1 - h_i) \cdot y_i - x_n \cdot h_i \cdot (1 - y_i) \right]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{1}{m} \cdot x_i \cdot \left[\sum_{i=1}^m h_i - y_i \right]$$

Implementation

Python code

```
#Import Library
from sklearn.linear_model import LogisticRegression
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test dataset
# Create logistic regression object
model = LogisticRegression()
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Equation coefficient and Intercept
print('Coefficient: \n', model.coef_)
print('Intercept: \n', model.intercept_)
#Predict Output
predicted= model.predict(x_test)
```

Implementation

R Code

```
x <- cbind(x_train,y_train)

# Train the model using the training sets and check score

logistic <- glm(y_train ~ ., data = x,family='binomial')

summary(logistic)

#Predict Output

predicted= predict(logistic,x_test)
```

Tuto

The screenshot shows a web browser window with the URL archive.ics.uci.edu/ml/datasets/Bank+Marketing. The page displays basic information about the dataset, including its characteristics and associated tasks.

Attribute Characteristics:	Real	Number of Attributes:	17	Date Donated:	2012-02-14
Associated Tasks:	Classification	Missing Values?	N/A	Number of Web Hits:	860529

Source:

[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. *Decision Support Systems*, Elsevier, 62:22-31, June 2014

Data Set Information:

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required. In order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

There are four datasets:

- 1) bank-additional-full csv with all examples (41188) and 20 inputs, ordered by date (from May 2008 to November 2010), very close to the data analyzed in [Moro et al., 2014]
 - 2) bank-additional.csv with 10% of the examples (4119); randomly selected from 1), and 20 inputs.
 - 3) bank-full.csv with all examples and 17 inputs, ordered by date (older version of this dataset with less inputs)
 - 4) bank.csv with 10% of the examples and 17 inputs, randomly selected from 3 (older version of this dataset with less inputs).
- The smallest datasets are provided to test more computationally demanding machine learning algorithms (e.g., SVM).

The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).

Attribute Information:

Input variables:
bank client data
1 - age (numeric)
2 - job : type of job (categorical: 'admin', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
3 - marital: marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high school', 'illiterate', 'professional.course', 'university.degree', 'unknown')
5 - default: has credit in default? (categorical: 'no', 'yes', 'unknown')

<http://archive.ics.uci.edu/ml/datasets/Bank+Marketing>

Attribute Information:

- Input variables:

bank client data:

1 - age (numeric)

2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')

3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)

4 - education (categorical:

'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')

5 - default: has credit in default? (categorical: 'no','yes','unknown')

6 - housing: has housing loan? (categorical: 'no','yes','unknown')

7 - loan: has personal loan? (categorical: 'no','yes','unknown')

related with the last contact of the current campaign:

8 - contact: contact communication type (categorical: 'cellular','telephone')

9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

Attribute Information:

- 12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
 - 13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
 - 14 - previous: number of contacts performed before this campaign and for this client (numeric)
 - 15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
 - # social and economic context attributes
 - 16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
 - 17 - cons.price.idx: consumer price index - monthly indicator (numeric)
 - 18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
 - 19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
 - 20 - nr.employed: number of employees - quarterly indicator (numeric)
- Output variable (desired target):
- 21 - y - has the client subscribed a term deposit? (binary: 'yes','no')





236

Régression Logistique

Régression linéaire

237

- Cette méthode se focalise sur les cas où les valeurs d'une variable à prédire sont continues
- Les valeurs à prédire peuvent être représentées par une fonction linéaire, donc une droite

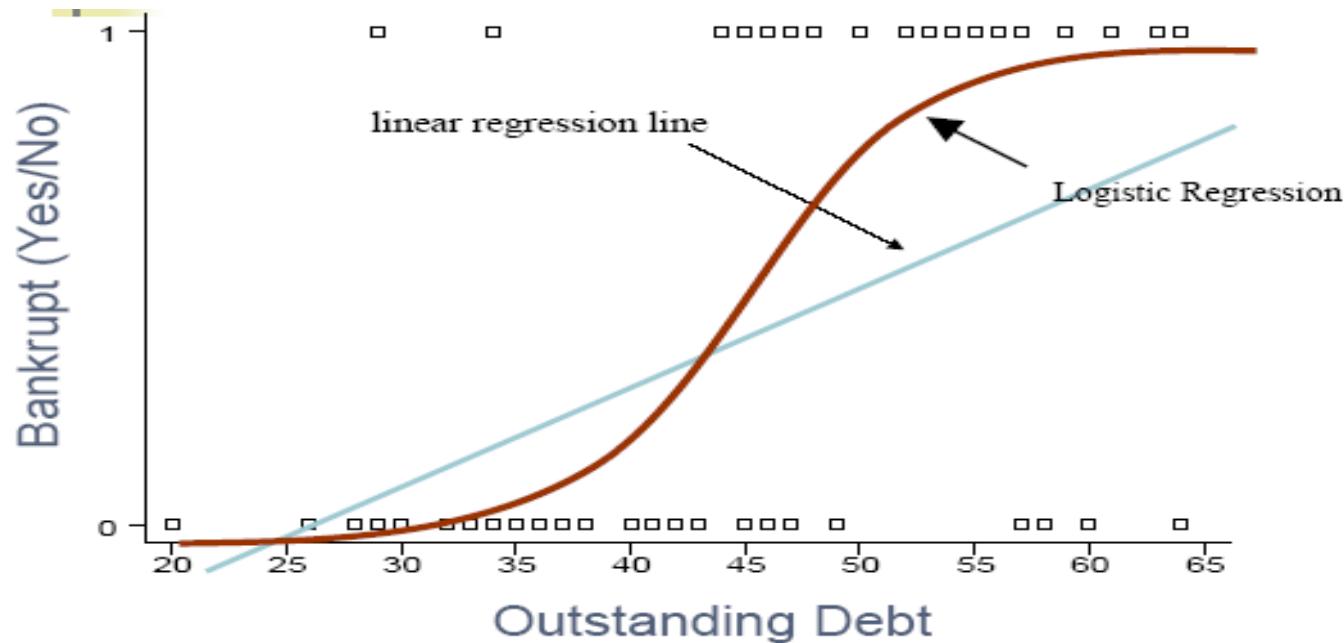
Régression logistique

238

- Cette méthode se focalise sur les situations où les valeurs d'une variable à prédire sont binaires (0 ou 1)
 - Exemple: Une variable booléenne
- Au lieu de prédire la valeur d'une variable, on prédit la probabilité de la variable à être égale à 0 et 1.
- Les probabilités décrivent une sigmoïde (courbe en forme de S) entre 0 et 1

Prédiction de banqueroute

239



Etude de cas: Compagnie de téléphone

- Adoption d'un nouveau service téléphonique (boîte vocale, accès à Internet...) suivant l'éducation, la stabilité de résidence et le salaire
- 10524 personnes ont répondu à un questionnaire sur ce thème réalisé par une compagnie de téléphone
- Comment prédire l'adoption d'un nouveau service téléphonique en fonction de l'éducation, de la stabilité de résidence et du salaire d'une personne?

Réponses au questionnaire

241

	High School or below		Some College or above	
	No Change in Residence during Last five years	Change in Residence during Last five years	No change in Residence during Last five years	Change in Residence during Last five years
Low Income	153/2160 = 0.071	226/1137 = 0.199	61/886 = 0.069	233/1091 = 0.214
High Income	147/1363 = 0.108	139 / 547 = 0.254	287/1925 = 0.149	382/1415 = 0.270

Il y a 2160 personnes qui ont répondu au questionnaire qui ont un niveau d'étude inférieur ou égale au lycée, un bas salaire et qui n'ont pas changé de résidence depuis 5 ans.

Il y a 153 personnes (sur ces 2160 personnes) qui ont adopté un nouveau service téléphonique

Probabilité globale d'adoption d'un nouveau service téléphone pour

Le modèle de régression logistique

242

- Prédire la probabilité de la valeur de Y à partir de variables indépendantes x₁,..., x_k
 - Y = 1: Choisir une option
 - Y = 0: Ne pas choisir une option
- $$\text{Probability}(Y=1|x_1, x_2, \dots, x_k) = \frac{\exp(\beta_0 + \beta_1 * x_1 + \dots + \beta_k * x_k)}{1 + \exp(\beta_0 + \beta_1 * x_1 + \dots + \beta_k * x_k)}$$

Les β_i sont des constantes inconnues à déterminer. Ils sont calculés/estimés par des programmes.

Interprétation des coefficients

243

- Si $\beta_i = 0$, alors le facteur β_i n'a aucun effet sur la chance de succès
- Si $\beta_i > 0$, le facteur β_i augmente la chance de succès
- Si $\beta_i < 0$, le facteur β_i décroît la chance de succès

Poser le problème (1)

244

- On doit calculer les probabilités d'adopter un nouveau service téléphonique en fonction de l'éducation, de la stabilité de résidence et le salaire d'une personne
- Soit Y la variable représentant l'adoption d'un nouveau service téléphonique
 - $Y = 1$ si un nouveau service est adopté, et $Y = 0$ sinon

Poser le problème (2)

- On a trois variables x_1 pour l'éducation, x_2 pour la stabilité de résidence et x_3 pour le salaire
- $X_1 = 1$ pour un niveau d'étude supérieur ou égal à l'université, 0 sinon
- $X_2 = 1$ pour un changement de résidence dans les 5 dernières années, 0 sinon
- $X_3 = 1$ pour un salaire élevé, 0 sinon
- Modèle $Prob(Y = 1|x_1, x_2, x_3) = \frac{\exp(\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_3)}{1 + \exp(\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_3)}.$

Résumé des données

246

x_1	x_2	x_3	# in sample	#adopters	# Non-adopters	Fraction adopters
0	0	0	2160	153	2007	.071
0	0	1	1363	147	1216	.108
0	1	0	1137	226	911	.199
0	1	1	547	139	408	.254
1	0	0	886	61	825	.069
1	1	0	1091	233	858	.214
1	0	1	1925	287	1638	.149
1	1	1	1415	382	1033	.270
			10524	1628	8896	1.000

Calcul de β_0 , β_1 , β_2 et β_3

247

Variable	Coeff.	Std. Error	p-Value	Odds	95% Conf. Intvl. for odds	
Constant	-2.500	0.058	0.000	0.082	0.071	0.095
x_1	0.161	0.058	0.006	1.175	1.048	1.316
x_2	0.992	0.056	0.000	2.698	2.416	3.013
x_3	0.444	0.058	0.000	1.560	1.393	1.746

$$\beta_0 \quad \beta_1$$

$$\beta_2 \quad \beta_3$$

Modèle

$$Prob(Y = 1|x_1, x_2, x_3) = \frac{\exp(-2.500 + 0.161 * x_1 + 0.992 * x_2 + 0.444 * x_3)}{1 + \exp(-2.500 + 0.161 * x_1 + 0.992 * x_2 + 0.444 * x_3)}.$$

x_1	x_2	x_3	# in sample	# adopters	Estimated (# adopters)	Fraction Adopters	Estimated $Prob(Y = l x_1, x_2, x_3)$
0	0	0	2160	153	164	0.071	0.076
0	0	1	1363	147	155	0.108	0.113
0	1	0	1137	226	206	0.199	0.181
0	1	1	547	139	140	0.254	0.257
1	0	0	886	61	78	0.069	0.088
1	1	0	1091	233	225	0.214	0.206
1	0	1	1925	287	252	0.149	0.131
1	1	1	1415	382	408	0.270	0.289

2160 x 0.076 = 164

Estimation du nombre de personnes qui peuvent adopter un nouveau service téléphonique

Nouvelles données

Calculs
d'erreurs

249

598 nouvelle personnes sont sondées

x_1	x_2	x_3	# in validation sample	# adopters in validation sample	Estimated (# adopters)	Error (Estimate - Actual)	Absolute Value of Error
0	0	0	29	3	2.200	-0.800	0.800
0	0	1	23	7	2.610	-4.390	4.390
0	1	0	112	25	20.302	-4.698	4.698
0	1	1	143	27	36.705	9.705	9.705
1	0	0	27	2	2.374	0.374	0.374
1	1	0	54	12	11.145	-0.855	0.855
1	0	1	125	13	16.338	3.338	3.338
1	1	1	85	30	24.528	-5.472	5.472
Totals			598	119	116.202		

$$85 \times 0.289 = 24.5$$

Estimation du nombre de personnes qui peuvent adopter un produit/éducatif

Calcul d'erreurs

250

- Total erreur: -2.8 (or $2.8 / 119 = 2.3\%$)
- La moyenne d'erreur absolue (sommes des erreurs absolues / 119): 24.9%

Tableau de contingence

Matrice de contingence [Kohavi, Provost, 1998]:

Prédit \ Observé	Adopteur	Non Adopteur	Total
Adopteur	103 (TP)	13 (FP)	116
Non adopteur	16 (FN)	466 (TN)	482
Total	119	479	598

TP: true positive, FP: false positive, FN: false negative, TN: true negative

Calcul de taux

□ Vrais positifs:

Cas positifs correctement prédicts

- $103 / 119 = 86.5 \%$

□ Fausses positives:

- Cas incorrectement prédicts positif
- $13 / 479 = 2.7 \%$

□ Exactitude:

- Nombre total de prédictions correctes
- $(103 + 466) / 598 = 95.15\%$

□ Précision:

- Proportion des prédictions positives correctes
- $103 / (103 + 13) = 88.8 \%$

□ Erreurs:

- Proportion des prédictions incorrectes
- $(13+16) / 598 = 4.85 \%$

Quel est le meilleur modèle?

Vrai: Offrir un crédit

Faux: Ne pas offrir un crédit

Modèle 1:

TP 600	FP 75
FN 25	TN 300

Modèle 2:

TP 600	FP 25
FN 75	TN 300

Taux d'erreur pour les 2 modèles: 10%

Le meilleur modèle est Modèle 2 car ce modèle a moins de FP

Conclusion

- Méthode facile à comprendre
- Méthode efficace
- Les prédictions sont faciles à réaliser
- Le bruit peut avoir un effet significatif sur la méthode
- Besoin de plusieurs mesures pour évaluer le modèle



Classification

Apprentissage supervisé

Découverte de règles ou formules (patterns) pour ranger les données dans des classes prédéfinies

- ❑ représentant un groupe d'individus homogènes
- ❑ permettant de classer les nouveaux arrivants
- ❑ Processus en deux étapes
 - ❑ construction d'un modèle sur les données dont la classe est connue (training data set)
 - ❑ utilisation pour classification des nouveaux arrivants

Applications

□ Marketing

- comprendre les critères prépondérants dans l'achat d'un produit
- segmentation automatique des clients pour le marketing direct

□ Maintenance

- aide et guidage d'un client suite à défauts constatés

□ Assurance

- analyse de risques

□ Isolation de populations à risques

- médecine



Les K plus proches voisins (KNN)

Principe de fonctionnement

- Le principe de cet algorithme de classification est très simple.
On lui fournit:
 - un ensemble de données d'apprentissage D
 - une fonction de distance d
 - et un entier k
- Pour tout nouveau point de test x , pour lequel il doit prendre une décision, l'algorithme recherche dans D les k points les plus proches de x au sens de la distance d , et attribue x à la classe qui est la plus fréquente parmi ces k voisins.

Généralités

- la méthode des k plus proches voisins est une méthode de **d'apprentissage supervisé**.
- dédiée à la **classification**.
- En abrégé k-NN ou KNN, de l'anglais *k-nearest neighbor*.
- L'algorithme KNN figure **parmi** les plus simples algorithmes **d'apprentissage artificiel**.
- L'objectif de l'algorithme est de classé les exemples **non étiquetés** sur **la base de leur similarité** avec **les exemples de la base d'apprentissage** .

Domaine d'activité

□ L'algorithme kNN est utilisée dans de nombreux domaines :

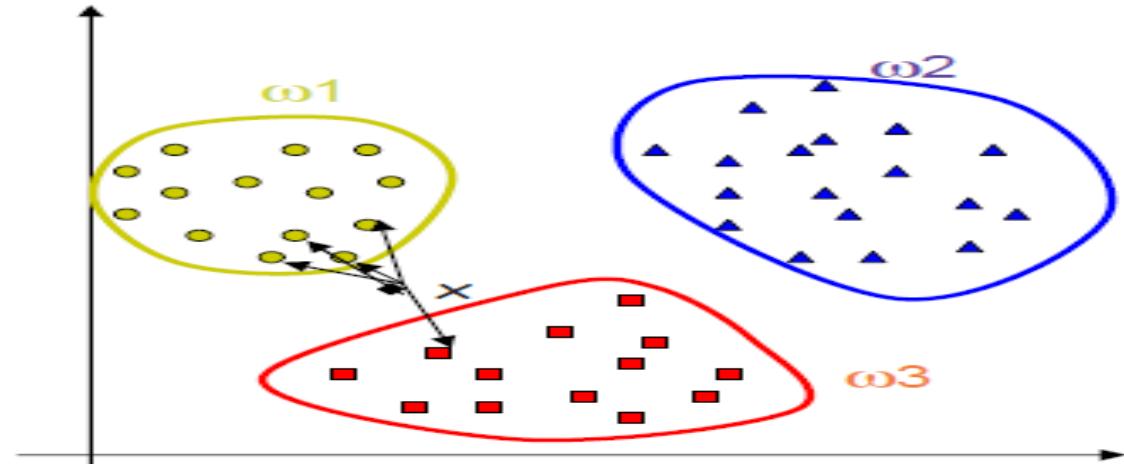
- La reconnaissance de formes.
- La recherche de nouveaux biomarqueurs pour le diagnostic.
- Algorithmes de compression.
- Analyse d'image satellite
- Marketing ciblé

Principe de fonctionnement

- Le principe de cet algorithme de classification est très simple.
On lui fournit:
 - un ensemble de données d'apprentissage D
 - une fonction de distance d
 - et un entier k
- Pour tout nouveau point de test x , pour lequel il doit prendre une décision, l'algorithme recherche dans D les k points les plus proches de x au sens de la distance d , et attribue x à la classe qui est la plus fréquente parmi ces k voisins.

Exemple

- Dans l'exemple suivant, on a 3 classes et le but est de trouver la valeur de la classe de l'exemple inconnu x .
 - On prend la distance Euclidienne et $k=5$ voisins
 - Des 5 plus proches voisins, 4 appartiennent à ω_1 et 1 appartient à majoritaire



Comment choisir la valeur de K ?

□ $K=1$: frontières des classes très complexes

- très sensible aux fluctuations des données (variance élevée).
- risque de sur-ajustement.
- résiste mal aux données bruitées.

□ $K=n$: frontière rigide

- moins sensible au bruit
- plus la valeur de k est grande plus la résultat d'affectation est bien réalisée

Mesures de distance

- Mesures souvent utilisées pour la distance $dist(x_i, x_j)$
- la distance Euclidienne: qui calcule la racine carrée de la somme des différences carrées entre les coordonnées de deux points :

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- la distance de Manhattan: qui calcule la somme des valeur absolue des différences entre les coordonnées de deux points .

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

- la distance de Minkowski

$$d(x, y) = \sqrt[q]{\sum_{i=1}^n |x_i - y_i|^q}$$
 e générale.

Avantages

- Apprentissage rapide
- Méthode facile à comprendre
- Adapté aux domaines où chaque classe est représentée par plusieurs prototypes et où les frontières sont irrégulières (ex. Reconnaissance de chiffre manuscrits ou d'images satellites)

Inconvénients

- prédition lente car il faut revoir tous les exemples à chaque fois.
- méthode gourmande en place mémoire
- sensible aux attributs non pertinents et corrélés
- particulièrement vulnérable au fléau de la dimensionnalité



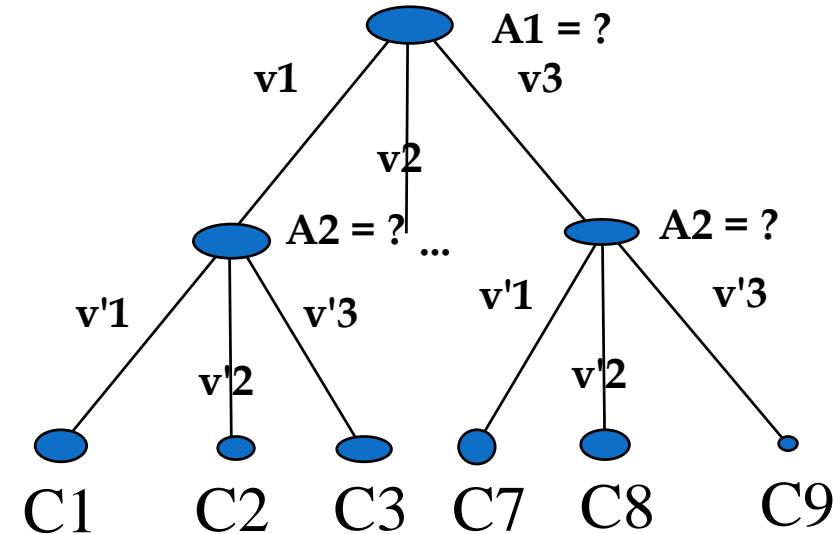
Les arbres de décision

Définition

- Arbre permettant de classer des enregistrements par division hiérarchiques en sous-classes
 - un nœud représente une classe de plus en plus fine depuis la racine
 - un arc représente un prédicat de partitionnement de la classe source
- Un attribut sert d'étiquette de classe (attribut cible à prédire), les autres permettant de partitionner

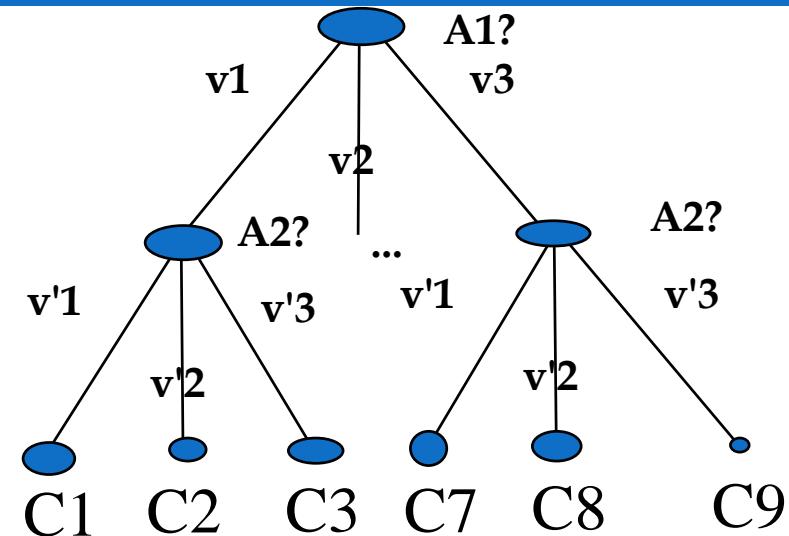
Génération de l'arbre

- Objectif:
 - obtenir des classes homogènes
 - couvrir au mieux les données
- Comment choisir les attributs (A_i) ?
- Comment isoler les valeurs discriminantes (v_i) ?



Arbre = ensemble de règles

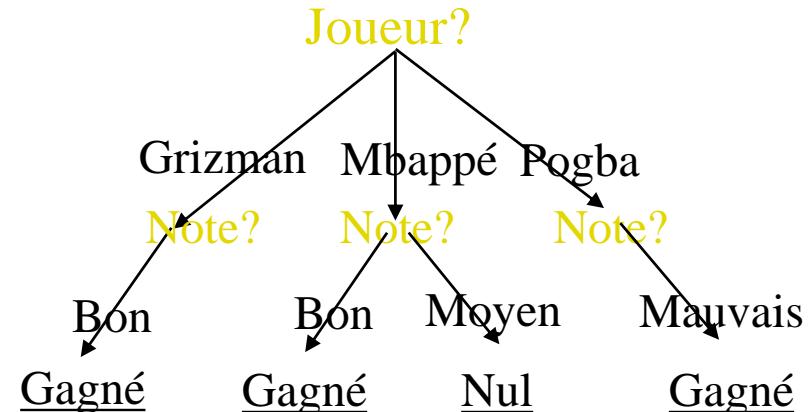
- $(A1=v1) \& (A2=v'1) \rightarrow C1$
- $(A1=v1) \& (A2=v'2) \rightarrow C2$
- $(A1=v1) \& (A2=v'3) \rightarrow C3$
- ...
- $(A1=v3) \& (A2=v'1) \rightarrow C7$
- $(A1=v3) \& (A2=v'2) \rightarrow C8$
- $(A1=v3) \& (A2=v'3) \rightarrow C9$



Exemple codant une table

Attributs ou variables

Joueur	Note	Résultat
Mbappé	Bon	Gagné
Mbappé	Moyen	Nul
Grizman	Bon	Gagné
Grizman	Bon	Gagné
Pogba	Mauvais	Gagné



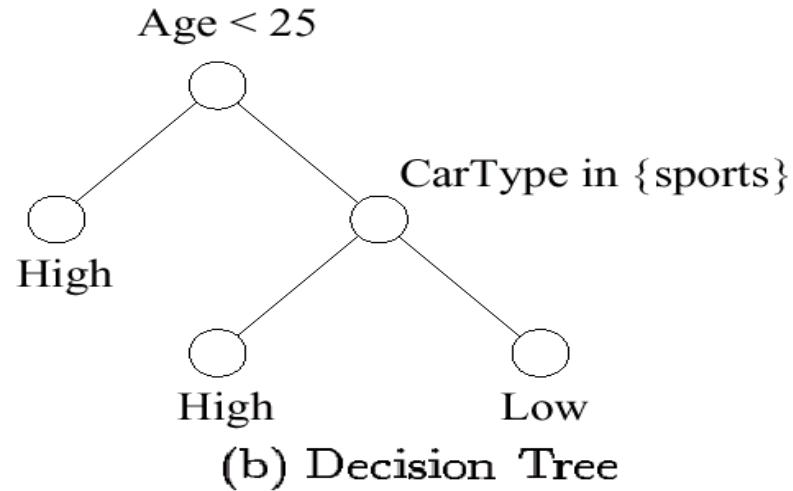
Autre Exemple

<i>rid</i>	Age	Car Type	Risk
0	23	family	High
1	17	sports	High
2	43	sports	High
3	68	family	Low
4	32	truck	Low
5	20	family	High

(a) Training Set



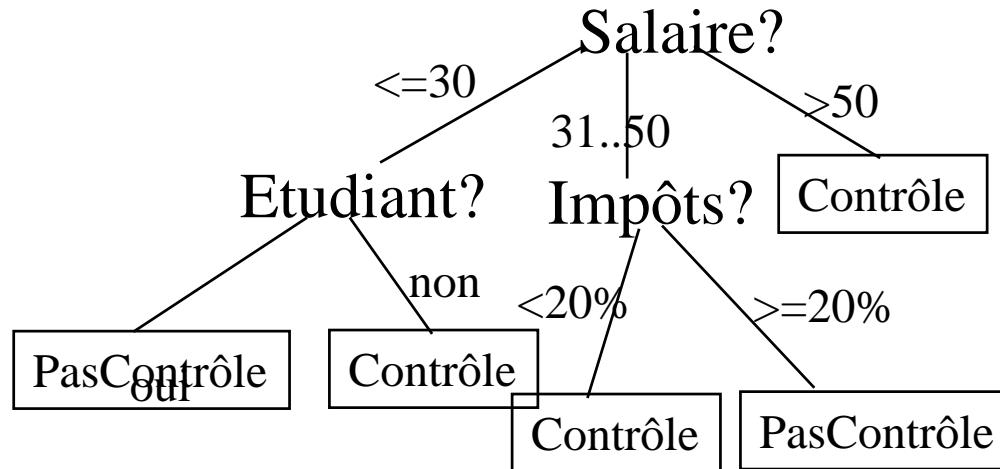
Classes cibles



(b) Decision Tree

Autre Exemple

□ Faut-il vous envoyer un contrôleur fiscal ?



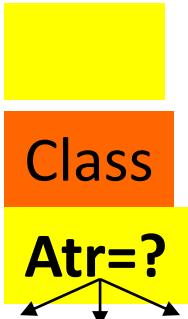
Procédure de construction (1)

- recherche à chaque niveau de l'attribut le plus discriminant
- Partition (nœud P)
 - si (tous les éléments de P sont dans la même classe) alors retour;
 - pour chaque attribut A faire
 - évaluer la qualité du partitionnement sur A;
 - utiliser le meilleur partitionnement pour diviser P en P₁, P₂, ...P_n
 - pour i = 1 à n faire Partition(P_i);

Procédure de Construction (2)

Processus récursif

- L'arbre commence à un nœud représentant toutes les données
- Si les objets sont de la même classe, alors le nœud devient une feuille étiqueté par le nom de la classe.
- Sinon, sélectionner les attributs qui séparent le mieux les objets en classes homogènes => Fonction de qualité
- La récursion s'arrête quand:
 - Les objets sont assignés à une classe homogène
 - Il n'y a plus d'attributs pour diviser,
 - Il n'y a pas d'objet avec la valeur d'attribut



Choix de l'attribut de division

Différentes mesures introduites

- il s'agit d'ordonner le désordre
- des indicateurs basés sur la théorie de l'information
- Choix des meilleurs attributs et valeurs
 - les meilleurs tests
- Possibilité de retour arrière
 - élaguer les arbres résultants (classes inutiles)
 - revoir certains partitionnements (zoom, réduire)

Mesure de qualité

□ La mesure est appelé fonction de qualité

- Goodness Function en anglais
- Varie selon l'algorithme :
 - Gain d'information (ID3/C4.5)
 - Suppose des attributs nominaux (discrets)
 - Peut-être étendu à des attributs continus
 - Gini Index
 - Suppose des attributs continus
 - Suppose plusieurs valeurs de division pour chaque attribut
 - Peut-être étendu pour des attributs nominaux

Mesure d'impureté

(variable nominale)

- Mesure des mélanges de classes d'un nœud N
 - $i(N) = \sum_i \sum_j \{ p_i * p_j \}$ avec $i \neq j$
 - p_i est la proportion d'individus de la classe i dans N .
- La réduction d'impureté de chaque division du nœud N par la variable x_j s'exprime par:
 - $\Delta N = i(N) - \sum_j p_j * i(N_j)$
 - p_j est la proportion d'individus du nœud dans le fils j
- Sur l'ensemble des n variables, la division du nœud t est effectuée à l'aide de la variable qui assure la réduction maximale de l'impureté (\sum minimum)

Mesure d'entropie

- Minimisation du désordre restant
 - π_i = fréquence relative de la classe i dans le nœud N
(% d'éléments de la classe i dans N)
- Mesure d'entropie d'un segment s
 - $E(N) = -\sum \pi_i \log_2(\pi_i)$
- Minimiser son évolution globale [Quinlan]
 - $\Delta N = E(N) - \sum_i P_i * E(N_i)$

Indices de Gini et Twoing

□ Indice de GINI

Si un ensemble de données T contient des éléments de N classes

$$\text{gini}(T) = 1 - \sum_i p_i^2 \text{ où } p_i \text{ est la fréquence relative de la classe } i \text{ dans } T$$

□ Indice de Twoing

$$G(t_g, t_d) = [((n_g/n)(n_d/n))/4][\sum_{i=1}^m |(n_{ig}/n_g) - (n_{id}/n_g)|]^2$$

- t_g : Sommet gauche issu de t.
- t_d : Sommet droit issu de t
- n_d (resp (n_g)) = card $\{t_d\}$ (resp card $\{t_g\}$).
- N : La taille de l'échantillon d'apprentissage.
- M : Le nombre de classe.
- n_{id} : (resp (n_{ig})) : l'effectif de la classe c_i dans t_d (resp (t_g)).

Exemple: Partitions de boules (1)

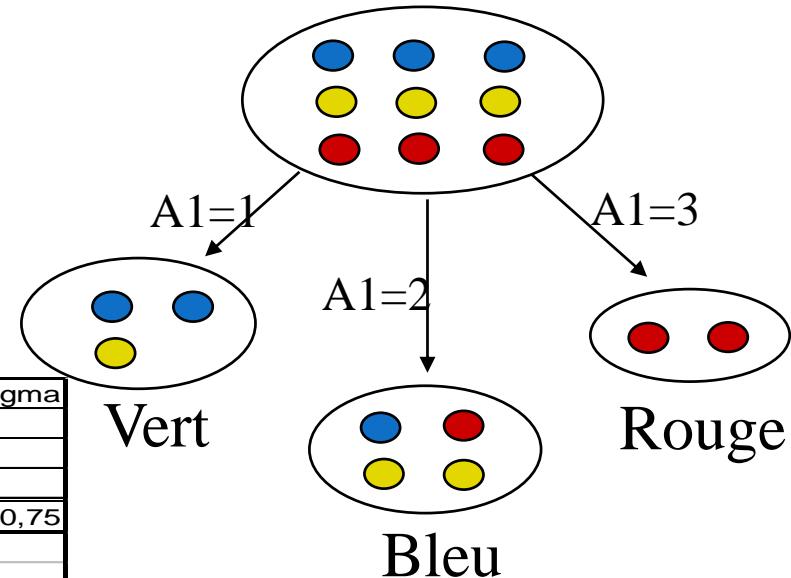
- Partition selon A1 (densité)

- Indice d'impureté :

- $i(N) = \sum_i^k \sum_j^k \{ p_i * p_j \}$ avec $i \neq j$

- P_i est la proportion d'individus de la classe i dans N .

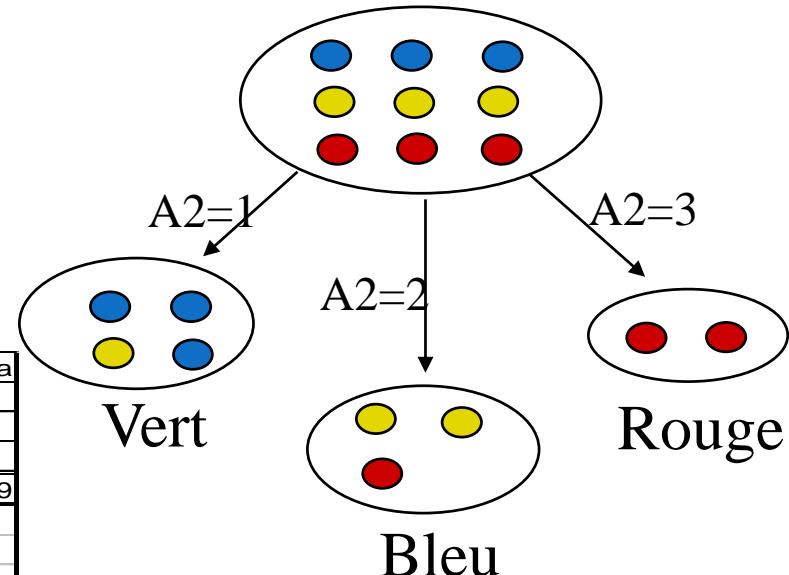
Proportion	C1	C2	C3	Sigma
Vert	0,67	0,25	0,00	
Bleu	0,33	0,50	0,00	
Rouge	0,00	0,25	1,00	
Entropie	0,92	1,00	0,00	0,75
N2 log2(N2)	-0,39	-0,50	0,00	
N3 log2(N3)	-0,53	-0,50	0,00	
N4 log2(N4)	0,00	0,00	0,00	
Impureté	0,44444444	0,625	0	0,43



Exemple: Partitions de boules (2)

- Partition selon A2
 - Position et 4 au plus par partition

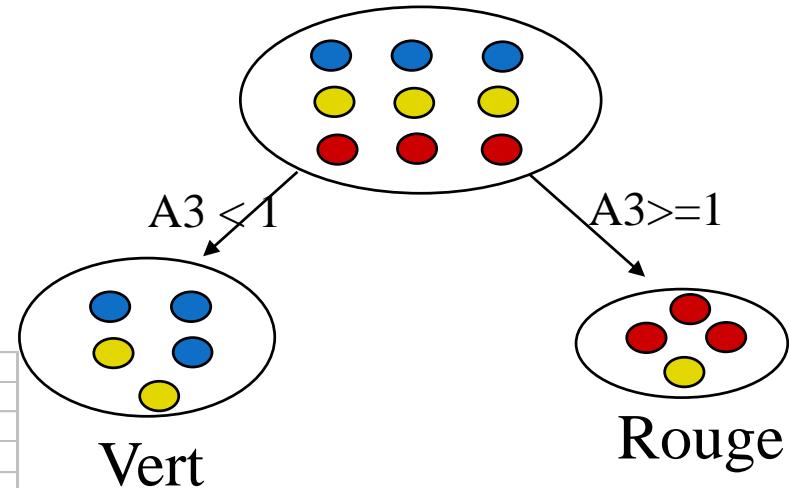
Proportion	C1	C2	C3	Sigma
Vert	0,75	0,00	0,00	
Bleu	0,25	0,67	0,00	
Rouge	0,00	0,33	1,00	
Entropie	0,81	0,39	0,00	0,49
N2 log2(N2)	-0,31	0,00	0,00	
N3 log2(N3)	-0,50	-0,39	0,00	
N4 log2(N4)	0,00	0,00	0,00	
Impureté	0,375	0,444444444	0	0,31



Exemple: Partitions de boules (3)

- Partition selon A3
 - Poids

Proportion	C1	C2	Sigma
Vert	0,60	0,00	
Bleu	0,40	0,25	
Rouge	0,00	0,75	
Entropie	0,97	0,50	0,76
N2 log2(N2)	-0,44	0,00	
N3 log2(N3)	-0,53	-0,50	
N4 log2(N4)	0,00	0,00	
Impureté	0,48	0,375	0,43



Exemple: Partitions de table (1)

Atr=?

Gain(Outlook) = 0.246

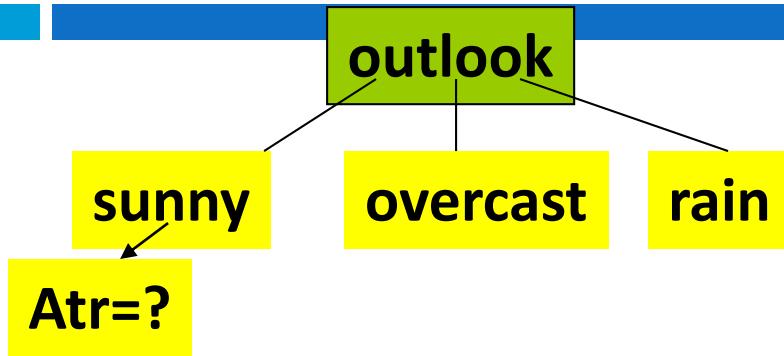
Gain(Temperature) = 0.029

Gain(Humidity) = 0.151

Gain(Windy) = 0.048

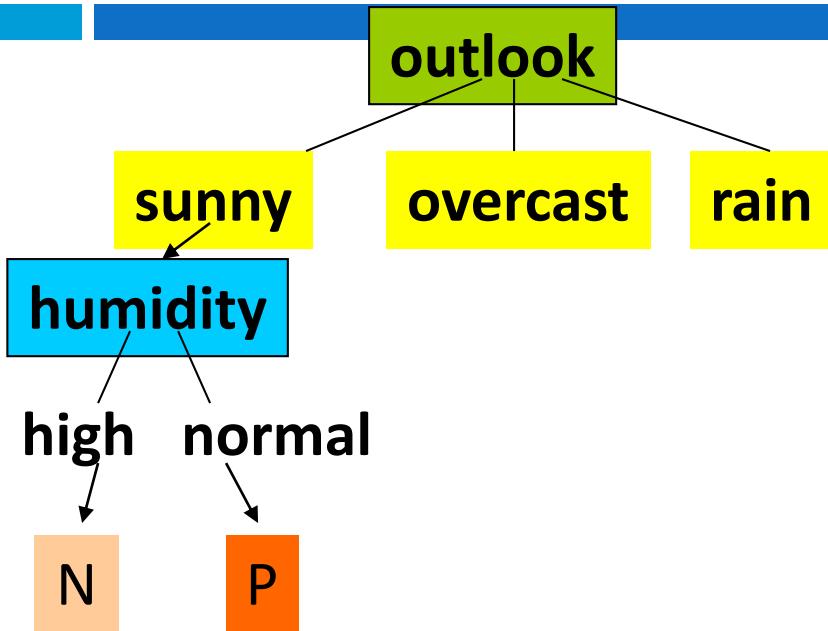
Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Exemple: Partitions de table (2)



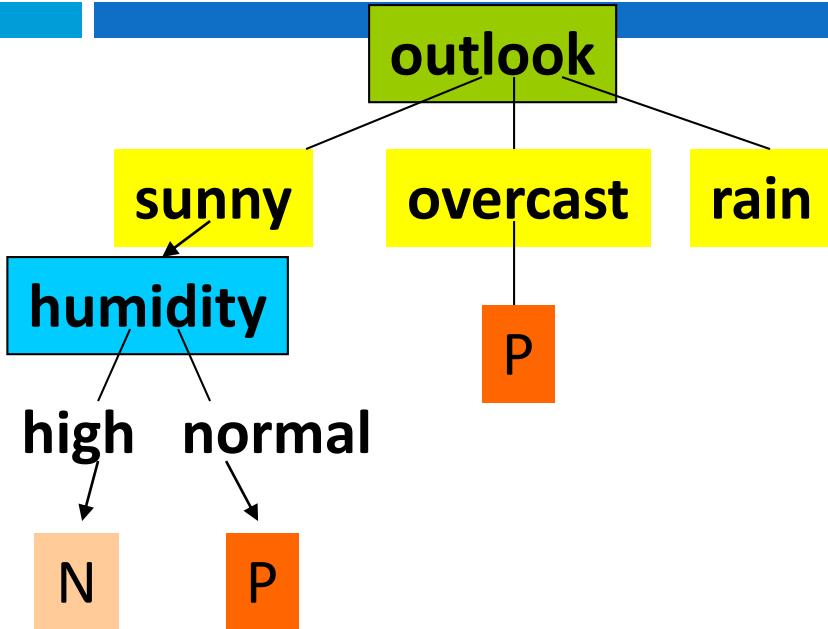
Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Exemple: Partitions de table (3)



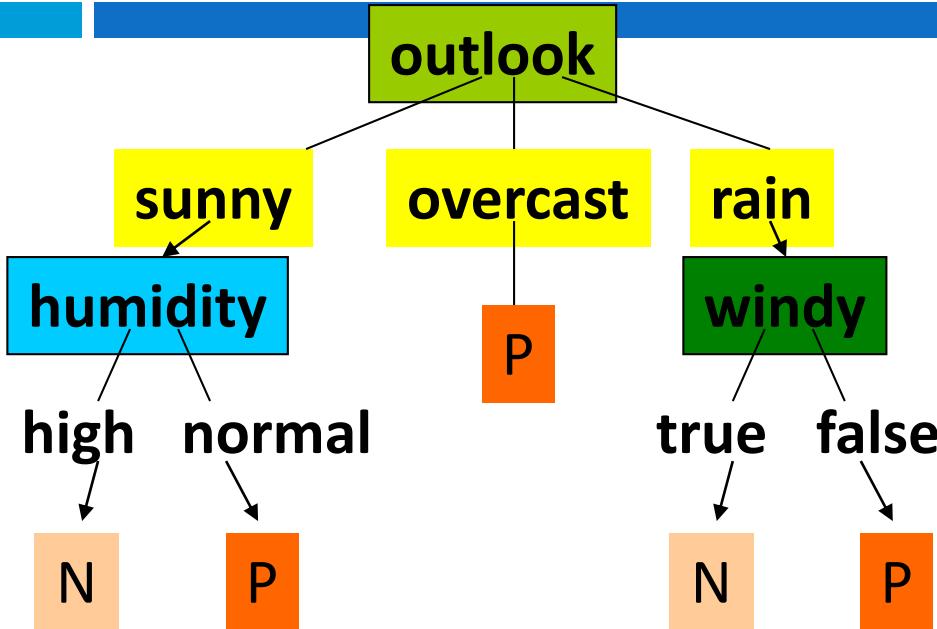
Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Exemple: Partitions de table (4)



Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Exemple: Partitions de table (5)



Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Types de tests

Binaire ou n-aire

- plus ou moins large et profond

Variable nominale

- un prédicat par valeur ou par liste de valeurs ?

Choix par niveau ou par classe

- mêmes tests pour chaque nœud interne d'un niveau

- arbres balancés ou non

Élimination de classes

- vides ou presque, peu représentatives

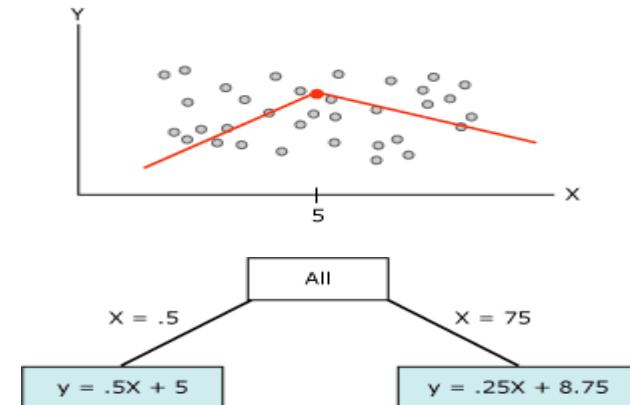
Problème des attributs continus

□ Certains attributs sont continus

- exemple : salaire
- découper en sous-ensembles ordonnés (e.g., déciles)
 - division en segments $[a_0, a_1[, [a_1, a_2[, \dots, [a_{n-1}, a_n]$
- utiliser moyenne, médiane, ... pour représenter
- minimiser la variance, une mesure de dispersion ...
- investiguer différents cas et retenir le meilleur
 - exemple : 2, 4, 8, etc. par découpe d'intervalles en 2 successivement

Attributs continus: Régression

- Partitionnement par droite de régression
- Chaque nœud est représenté par une formule de régression
- Séparation des données = point de non linéarité
- 1 ou plusieurs régresseurs
- Exemple :
 - $\text{salaire} = a + b * \text{tranche_age}$



Procédure d'élagage

- Les arbres trop touffus sont inutiles
- Intérêt d'un élagage récursif à partir des feuilles
 - S'appuie sur un modèle de coût d'utilité
- Possibilité de l'appliquer sur l'ensemble des données ou sur un sous-ensemble réservé à la validation

Exemple d'élagage

□ Exemple :

- arbres vus comme encodage de tuples
- partition utile si gain supérieur à un seuil
- coût d'un partitionnement
 - CP bits pour coder les prédictats de partition
 - Entropie_Après bits pour coder chaque tuple
- partitionnement à supprimer si :
 - $\text{Gain} = n * \text{Entropie_Après} + \text{CP} - n * \text{Entropie_Avant} < \text{seuil}$
- Ce test peut être appliquer lors de la création

Types d'arbres

	Description	Critère de coupe
Segmentation	binaire	<i>variance si variable continue, Indice de pureté si variable nominale</i>
ID3	<i>n</i> -aire	<i>Gain informationnel par entropie de Shannon</i>
C4.5	<i>n</i> -aire dérivé de ID3 + élagage, règles simplifiées, données manquantes,...	<i>Ratio du gain informationnel</i>
CART	binaire régression + élagage	<i>Indice de Gini si 2 valeurs en conclusion</i> <i>Indice de Twoing sinon</i>

Méthodes ID3 et C4.5

□ ID3

- Le pouvoir discriminatoire (ou gain informationnel) d'une variable \leq une variation d'« entropie de Shannon » lors de la partition de S
- C4.5 (ID3++)
 - Support des variables continues
 - Introduit un facteur «Gain ratio» visant à pénaliser la prolifération des nœuds
- Critères d'arrêt :
 - Seuils de gain informationnel, d'effectif dans un nœud
 - Test statistique d'indépendance des variables (Ki2)

Méthode CART

□ Principes

- si problème à 2 classes, cherche la bi-partition minimisant l'indice d'impureté de Gini
- si problème à N classes, cherche celle maximisant le gain d'information donné par l'indice de Towing
- Critères d 'arrêt :
 - Seuil de gain informationnel
 - Seuil d 'effectif dans un nœud
 - Procédure d'élagage

Méthodes passant à l'échelle

- La plupart des algorithmes de base supposent que les données tiennent en mémoire
- La recherche en bases de données a proposé des méthodes permettant de traiter de grandes BD
- Principales méthodes:
 - ▣ SLIQ (EDBT'96 -- Mehta et al.'96)
 - ▣ SPRINT (VLDB96 -- J. Shafer et al.'96)
 - ▣ RainForest (VLDB98 -- J. Hekankho et al.'98)
 - ▣ PUBLIC (VLDB'98 -- R. Rastogi et al.'98)

Méthode SLIQ

□ SLIQ (EDBT'96 -- Mehta et al.'96)

- Supervised Learning In Quest
- Classificateurs CART et C4.5 :
 - Développe l'arbre en profondeur d'abord
 - Tri les données de manière répétée à chaque nœud
- SLIQ:
 - Remplace le tri répété par 1 seul tri par attribut
 - Utilise une nouvelle structure de données (**class-list**)
 - S'applique sur des attributs numériques ou nominaux
- Indicateur: maximiser $\text{gini}_{\text{split}}(T) = \sum_i [n_i/n] \text{gini}(T_i)$

Méthode SPRINT

■ SPRINT (VLDB96 -- J. Shafer et al.'96)

- Scalable PaRallelizable INduction of decision Tree
- SLIQ nécessite de garder la class-list en mémoire
- SPRINT
 - Ne nécessite pas de structure résidente en mémoire
 - Version parallèle passant à l'échelle

Comparaison avec SLIQ

- SLIQ ne divise pas les listes d'attributs lors du split

- Repère le nœud par un pointeur dans la class-list

- Avantages

- Pas de recopie des listes d'attributs lors du split
- Ré-allocation d'articles par déplacement de pointeur

- Désavantage

- La liste des références (class-list) de taille le nombre d'articles doit tenir en mémoire

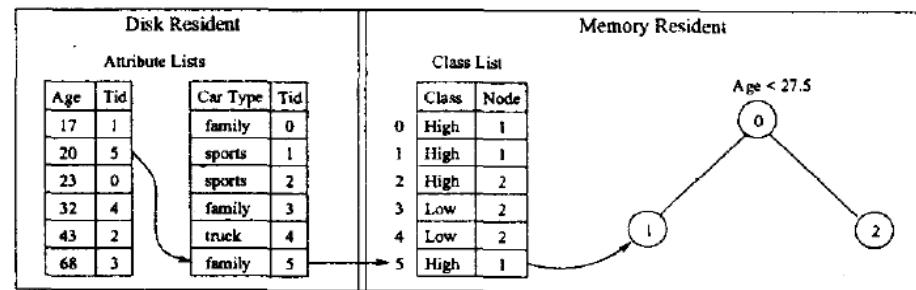


Figure 7: Attribute and Class lists in SLIQ

- SPRINT peut être facilement parallélisé
 - pas de structures partagées en mémoire

Bilan

- De nombreux algorithmes de construction d'arbre de décision
- SPRINT passe à l'échelle et traite des attributs nominaux ou continus
- Autres algorithmes proposés
 - Encore plus rapides ?



Le modèle de Bayes

La classification de Bayes

- Une méthode simple de classification supervisée
- Basée sur l'utilisation du Théorème de Bayes:

$$\Pr[H | E] = \frac{\Pr[E | H] \Pr[H]}{\Pr[E]}$$

Où H est l'hypothèse à tester, et E est l'évidence associée à l'hypothèse

- $\Pr(E | H)$ et $\Pr(H)$ sont facilement calculables
- $\Pr(H)$ est une probabilité a priori: la probabilité de H avant la présentation de l'évidence
- Il n'est pas nécessaire de calculer $\Pr(E)$

Note: $\Pr(E) = P(E)$

Méthode

- Une évidence E est donnée
- On calcule $P(H \mid E)$ pour toutes les valeurs de H
- Si $P(H = h \mid E)$ est maximum, alors on choisit:
 $H=h$

Étude de cas: Météo et match de foot

Données

Outlook	Temp.	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Calcul: $\Pr[\text{yes} | E]$

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Evidence E

$$\Pr[\text{yes} | E] = \Pr[\text{Outlook} = \text{Sunny} | \text{yes}] \times$$

Probability for class “yes”

$$\Pr[\text{Temperature} = \text{Cool} | \text{yes}] \times$$
$$\Pr[\text{Humidity} = \text{High} | \text{yes}] \times$$
$$\Pr[\text{Windy} = \text{True} | \text{yes}] \times \frac{\Pr[\text{yes}]}{\Pr[E]}$$

$$= \frac{2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14}{\Pr[E]}$$

$$= 0.0053 / \Pr[E]$$

Calcul: $\Pr[\text{No} | E]$

$$\boxed{\Pr[\text{No} | E] =}$$

$$= \Pr[\text{Outlook} = \text{Sunny} | \text{No}] \times \Pr[\text{Temperature} = \text{Cool} | \text{No}]$$

$$\times \Pr[\text{Humidity} = \text{High} | \text{No}] \times \Pr[\text{Windy} =$$

$$\text{True} | \text{No}] \times \Pr[\text{No}] / \Pr[E]$$

$$= 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 / \Pr[E]$$

$$= 0.0205 / \Pr[E]$$

Conclusion de l'exemple

On compare $\Pr[\text{Yes} | E]$ et $\Pr[\text{No} | E]$

- $\Pr[\text{Yes} | E] = 0.0053 / \Pr[E]$
- $\Pr[\text{No} | E] = 0.0205 / \Pr[E]$
- Donc le match ne va pas avoir lieu, car $0.0205 > 0.0053$

Cas d'un numérateur égal à 0

- Pour éviter d'avoir un numérateur égal à 0 et donc une probabilité égale à 0, dans le cas où le nombre d'attributs ayant une certaine valeur serait 0, on ajoute une constante k à chaque valeur au numérateur et au dénominateur
 - Un rapport n/d est transformé en $(n + kp)/(d+k)$, où p est une fraction du nombre total des valeurs possibles de l'attribut
 - K est entre 0 et 1
 - Estimation de Laplace: $k = 1$

Exemple: $\Pr[\text{No} \mid E]$

□ $K = 1$

□ $\Pr[\text{No} \mid E] =$

$$= \Pr[\text{Outlook} = \text{Sunny} \mid \text{No}] \times \Pr[\text{Temperature} = \text{Cool} \mid \text{No}] \times \Pr[\text{Humidity} = \text{High} \mid \text{No}] \times \Pr[\text{Windy} = \text{True} \mid \text{No}] \times \Pr[\text{No}] / \Pr[E]$$

$$= (3+1/3)/(5+1) \times (1+1/3)/(5+1) \times (4+1/2)/(5+1) \times (3+1/2)/(5+1) \times 5/14 / \Pr[E]$$

$$= 0.7539 / \Pr[E]$$

Données manquantes

- Les données manquantes sont traitées de façon satisfaisante par la méthode de Bayes
- Les valeurs manquantes sont ignorées, et une probabilité de 1 est considérée

Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	True	?

Le match n'aura pas lieu

$$\text{Likelihood of "yes"} = 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$$

$$\text{Likelihood of "no"} = 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$$

$$P(\text{"yes"}) = 0.0238 / (0.0238 + 0.0343) = 41\%$$

$$P(\text{"no"}) = 0.0343 / (0.0238 + 0.0343) = 59\%$$

Données numériques

- Une fonction de densité des probabilités $f(x)$ représente la distribution normale des données de l'attribut numérique x en fonction d'une moyenne μ et d'une déviation standard σ

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2}$$

- Les valeurs manquantes ne sont pas incluses dans les calculs des moyennes et des déviations standards

Exemple de calcul de $f(x)$



Outlook		Temperature				Humidity				Windy		Play	
	Yes	No		Yes	No		Yes	No		Yes	No	Yes	No
Sunny	2	3		83	85		86	85	False	6	2	9	5
Overcast	4	0		70	80		96	90	True	3	3		
Rainy	3	2		68	65		80	70					
			Yes	73	74.6	mean	79.1	86.2	False	6/9	2/5	9/14	5/14
				6.2	7.9	std dev	10.2	9.7	True	3/9	3/5		
				3/9	2/5								

Temperature: $x = 66$, $\mu = 73$, $\sigma = 6.2$ pour yes

$$f(\text{temperature} = 66 \mid \text{yes}) = \frac{1}{\sqrt{2\pi} 6.2} e^{-\frac{(66-73)^2}{2*6.2^2}} = 0.0340$$

Exemple de classification

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

Likelihood of "yes" = $2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036$

Likelihood of "no" = $3/5 \times 0.0291 \times 0.0380 \times 3/5 \times 5/14 = 0.000136$

$P(\text{"yes"}) = 0.000036 / (0.000036 + 0.000136) = 20.9\%$

$P(\text{"no"}) = 0.000136 / (0.000036 + 0.000136) = 79.1\%$

Le match n'aura pas lieu

Relations entre densité et probabilité

$$\Pr[c - \frac{\varepsilon}{2} < x < c + \frac{\varepsilon}{2}] \approx \varepsilon * f(c)$$

$$\Pr[a \leq x \leq b] = \int_a^b f(t) dt$$

Conclusion

□ Méthode efficace

- La classification ne demande pas des estimations exactes des probabilités, mais seulement que la probabilité maximum soit donnée à la bonne classe
- Les numériques ne sont pas toujours distribués normalement, on a donc besoin d'autres estimations
 - Kernel density estimator

4. Réseaux Bayésiens

Classificateurs statistiques

- Basés sur les probabilités conditionnelles
- Prévision du futur à partir du passé
- Suppose l'indépendance des attributs

Fondements

Dérivé du théorème de Bayes

- permet de calculer une probabilité à postériori $P(C_i/X)$ d'un événement C_i sachant que X s'est produit à partir d'une probabilité à priori $P(C_i)$ de production de l'événement C_i
- $$P(C_i/X) = P(X/C_i)*P(C_i) / \sum P(X/C_j)*P(C_j)$$
- Plus simplement si E est l'événement:
 - $$P(E/X) = P(X/E)*P(E)/P(X)$$

Bayésien Naïf

□ Chaque enregistrement est un tuple

- $X = (x_1, x_2, \dots, x_n)$ sur $R(A_1, A_2, \dots, A_n)$
- Il s'agit de classer X parmi m classes C_1, \dots, C_m
- L'événement C_i est l'appartenance à la classe C_i
- Assignation de la classe la plus probable
 - Celle maximisant $P(C_i/X) = P(X/C_i) * P(C_i)/P(X)$
 - $P(X)$ est supposé constant (équi-probabilité des tuples)
- On cherche la classe maximisant :
 - $P(X/C_i) * P(C_i)$ pour $i = 1 \text{ à } m$

On calcule la probabilité de chaque classe étant donné le tuple X

Calcul de $P(X/C_i)$

■ $P(C_i)$ est déduite de l'échantillon :

- Comptage "training set" = $\text{Taille}(C_i) / \text{Taille}(\text{Ech})$
- $P(X/C_i)$ est approchée comme suit :
 - Indépendance des attributs →
 - $P(X/C_i) = \prod_k P(x_k/C_i)$
- $P(x_k/C_i)$ est estimé comme suit:
 - variable nominale = $\text{Taille}(t=x_k \text{ de } C_i)$
 - distribution gaussienne si variable continue

$P(x_k/C_i)$ est la probabilité d'avoir une valeur donnée x_k pour un attribut d'un tuple dans la classe C_i ; Calculée sur le training set

Exemple de problème

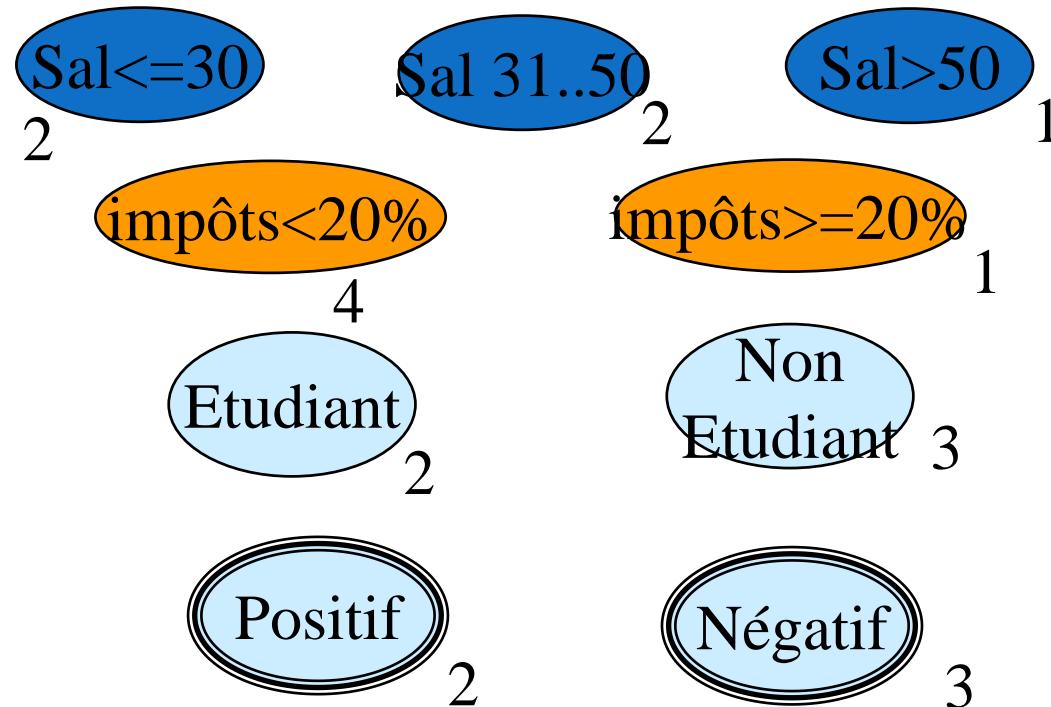
- Faut-il effectuer un contrôle fiscal ?
 - Échantillon de contrôlés

- Faut-il contrôler un nouvel arrivant ?

Salaire	Impôts	Etudiant	Contrôle
20	0	oui	négatif
30	0	non	positif
40	5	oui	positif
40	10	non	négatif
60	10	non	positif

35	2	oui	???
----	---	-----	-----

Les classes nominales



Calcul de Probabilités

- Il s'agit de choisir C_i maximisant $P(C_i/X)$:
 - ▣ $P(\text{Positif}/X) = P(X/\text{Positif})P(\text{Positif})/P(X)$
 - ▣ $P(\text{Négatif}/X) = P(X/\text{Négatif})P(\text{Négatif})/P(X)$
 - ▣ $P(X)$ est supposé constant
- Donc, choisir le plus grand de $\{P(X/\text{Positif})P(\text{Positif}), P(X/\text{Négatif})P(\text{Négatif})\}$
 - ▣ $P(X/\text{Positif}) = \prod_k P(X_k/\text{Positif}) = P(\text{sal30..50}/\text{Positif}) * P(\text{impots}<20\%/\text{Positif}) * P(\text{Etudiant}/\text{Positif}) = 2/3 * 1 * 1/3 = 2/9; P(\text{Positif}) = 3/5$
➔ Produit = 0.13
 - ▣ $P(X/\text{Négatif}) = \prod_k P(X_k/\text{Négatif}) = P(\text{sal30..50}/\text{Négatif}) * P(\text{impots}<20\%/\text{Négatif}) * P(\text{Etudiant}/\text{Négatif}) = 1/2 * 1/2 * 1/2 = 1/8;$
 $P(\text{Négatif}) = 2/5$ ➔ Produit = 0.05
- On effectuera donc un contrôle !

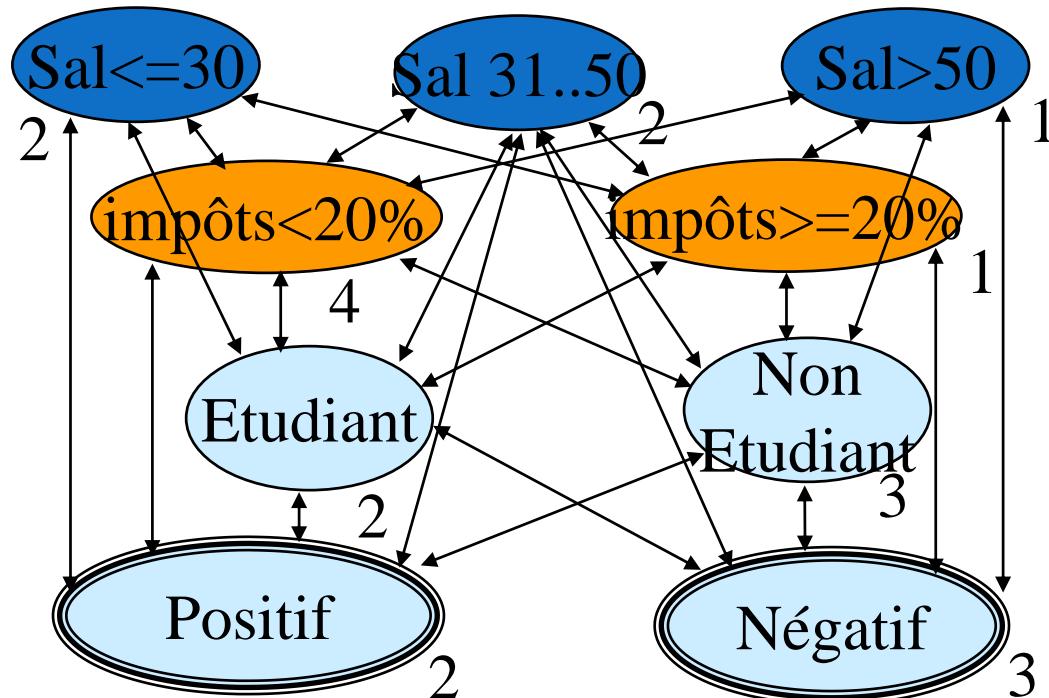
Réseau Bayésien

- Nœuds = Variables aléatoires
- Structure
 - Graphe direct acyclique de dépendance
 - $X \rightarrow Y$ signifie que X est un parent de Y
 - $X \rightarrow \rightarrow Y$ signifie que X est un descendant de Y
 - Les variables non liées sont indépendantes
- Classes à déterminer
 - Nœuds singuliers du réseau
- Probabilités connues
 - à priori et conditionnelles (arcs)

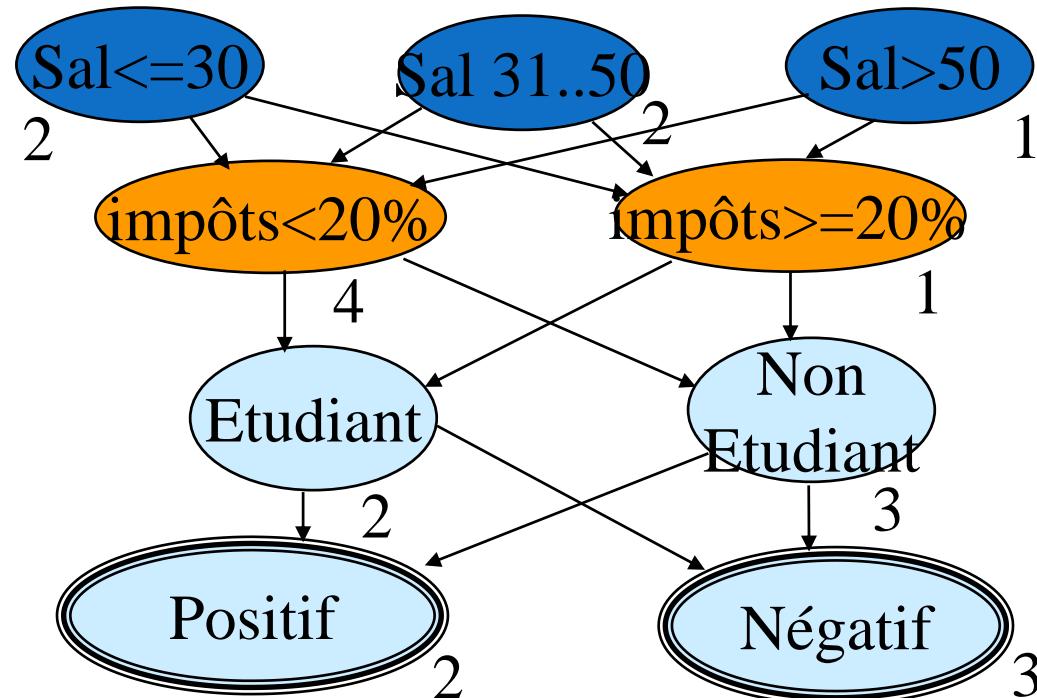
Calculs

- L'instanciation des variables non classes permet de calculer la probabilité des classes
- Application des calculs classiques de probabilité et du théorème de bayes
- Apprentissage à partir d'une base d'échantillons
- Peut être complexe si structure inconnue

Exemple complet



Structure de connaissance



Autre exemple

- Classification de pannes d'ordinateurs
 - Couleur de voyant (Rouge, Vert)
 - Équipement défaillant (UC,MC,PE)
- Envoie d'un dépanneur selon la classe
- Calcul de probabilités sur le training set

P(Couleur/Panne)	Rouge	Vert	P(Panne)
UC	0,70	0,30	0,20
MC	0,40	0,60	0,10
PE	0,20	0,80	0,70
P(Couleur)	0,32	0,68	1,00

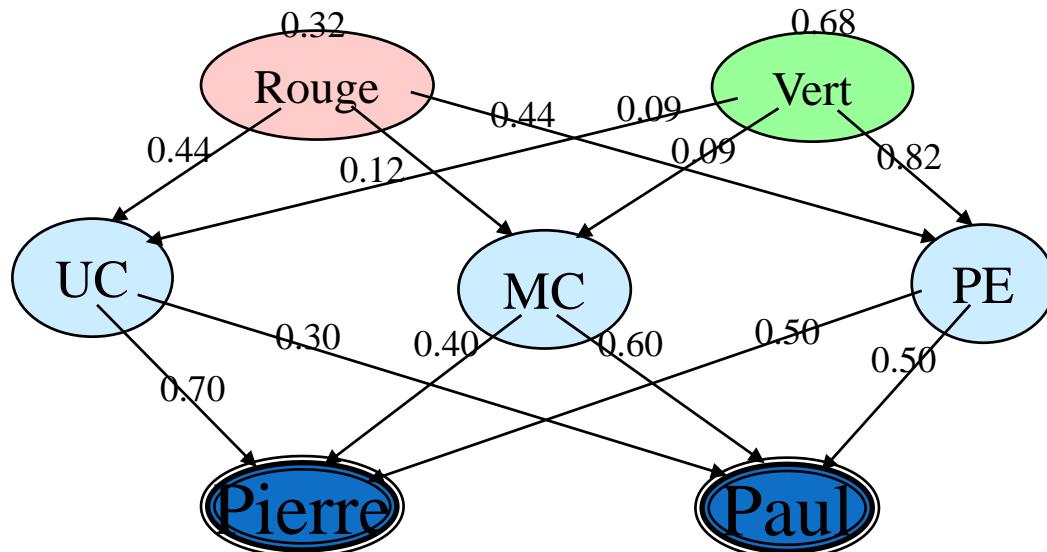
Exemple de réseau

Voyant

Panne

Dépanneur

Rouge



?

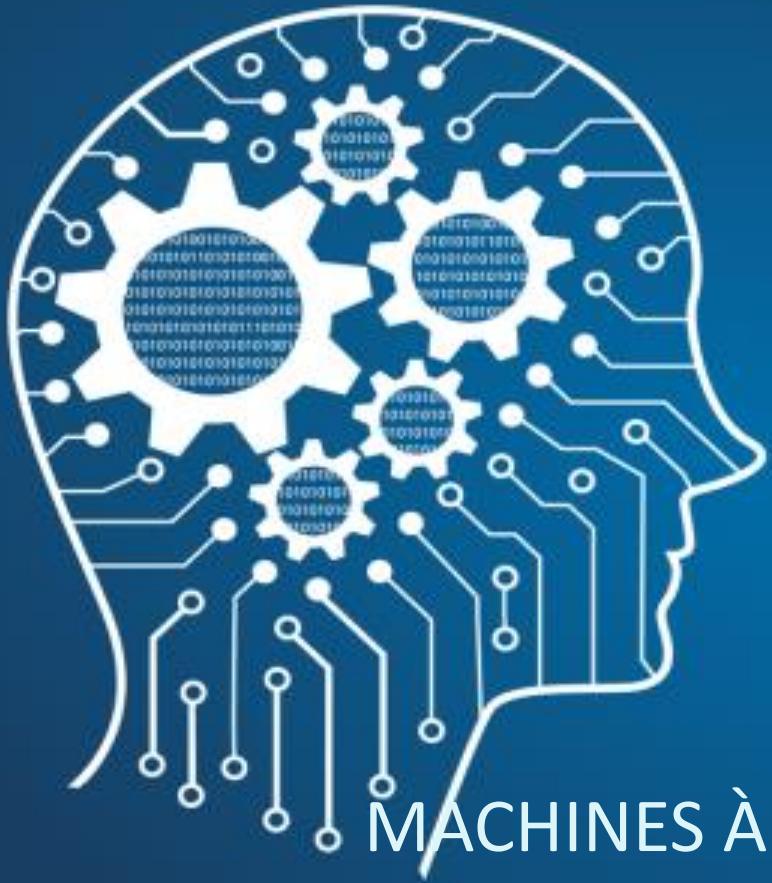
Intérêt

- Permet d'inférer les probabilités dans le réseau
 - méthode d 'inférence du futur à partir du passé
 - les événements X_i doivent être indépendants
 - méthode assez peu appliquée en Data Mining
- Problèmes
 - Comment choisir la structure du réseau ?
 - Comment limiter le temps de calcul ?

Bilan

□ Apprentissage

- si structure connue = calculs de proba.
- si inconnue = difficile à inférer
- Baysien naïf
 - suppose l'indépendance des variables
- Réseaux baysiens
 - permettent certaines dépendances
 - nécessitent des tables d'apprentissage réduites



MACHINES À VECTEURS SUPPORTS (SVM)

Plan

□ Historique

- Qu'est-ce qu'une bonne frontière de séparation pour deux classes linéairement séparables ?
 - ▣ La solution SVM
- Adaptation aux cas non linéairement séparables: l'astuce des fonctions noyau
- Exemples d'application
- Conclusion

Historique du SVM

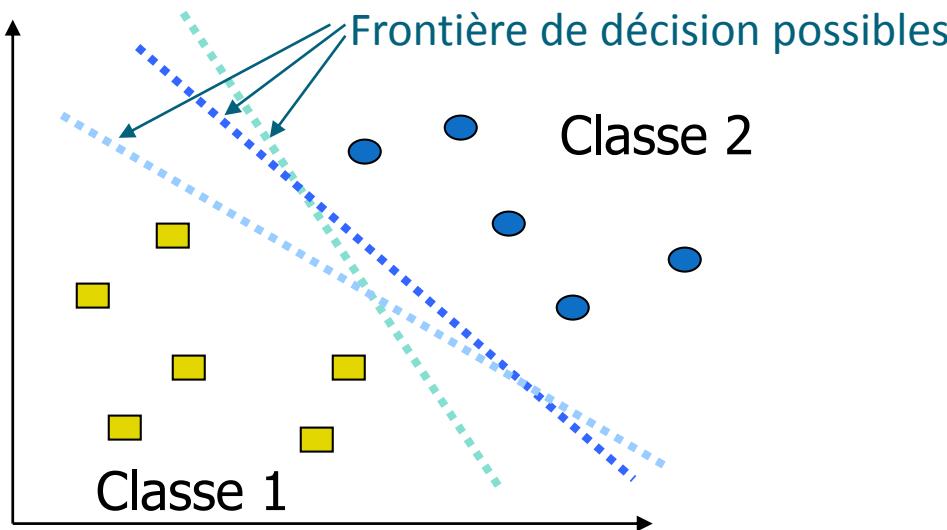
344

- Classifieur devenu populaire depuis que, partant d'images formées de pixels, il a permis des performances égales aux RNA pour reconnaître l'écriture manuscrite.
- Proche de :
 - Séparateurs à vastes marges
 - Méthodes à fonctions noyau
 - Réseaux de neurones à bases radiales

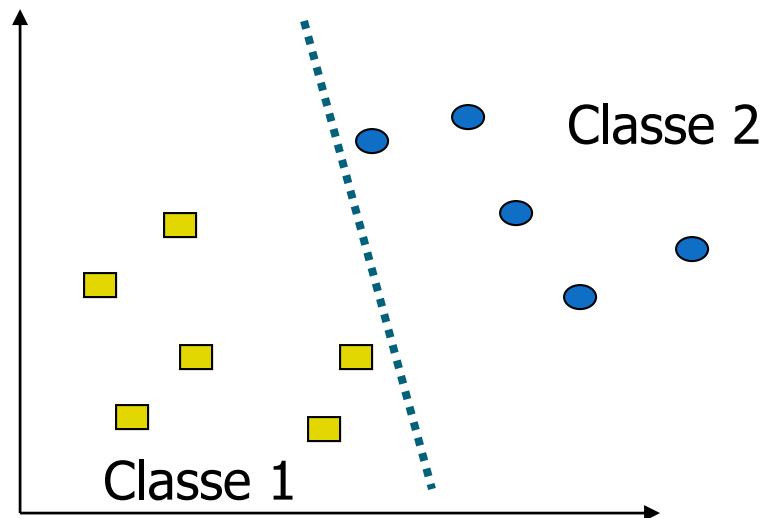
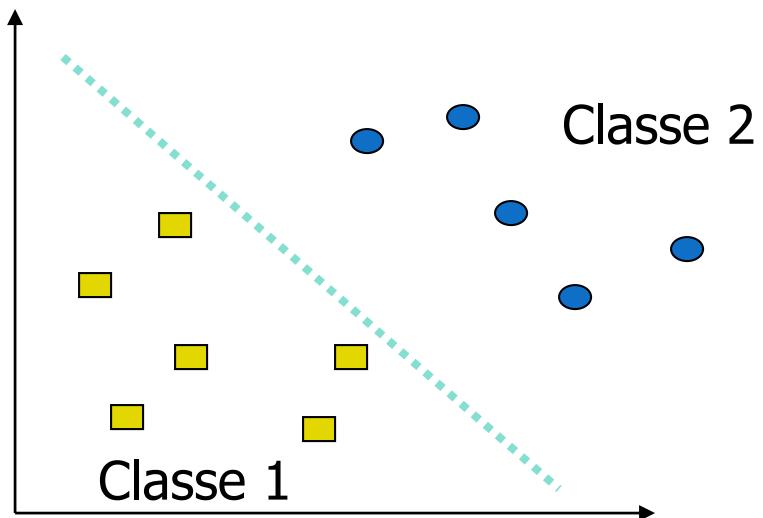
Problème à deux classes linéairement séparables

345

- Plusieurs surfaces de décision existent pour séparer les classes ; laquelle choisir ?

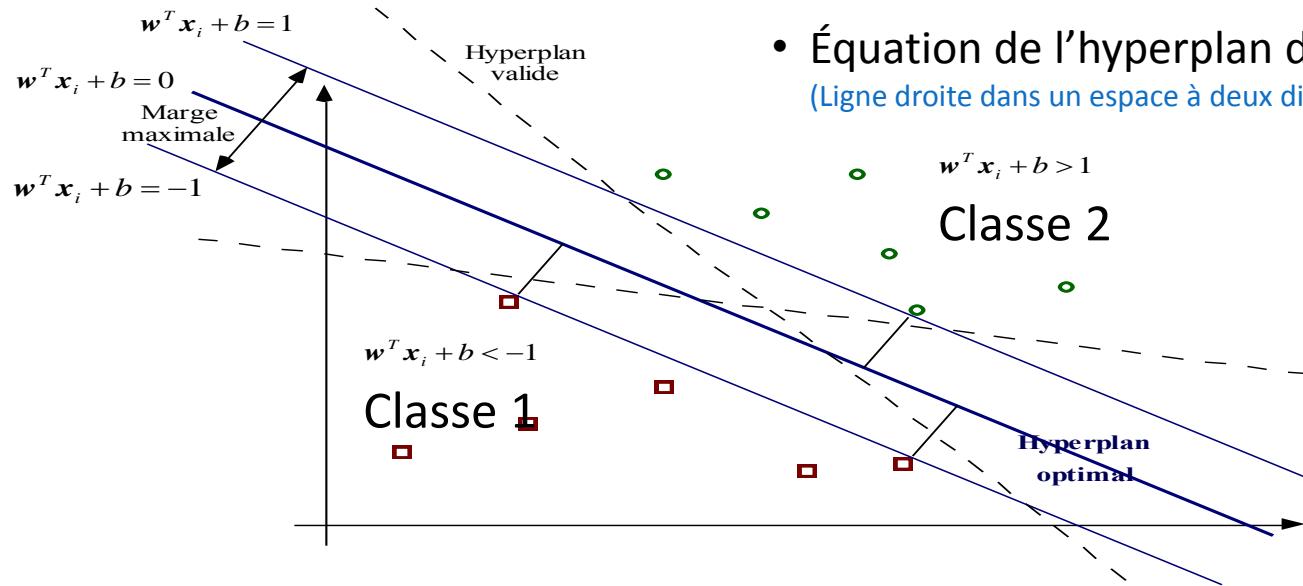


Exemples de choix mal avisés



- ▶ Pour minimiser la sensibilité au bruit, la surface de décision doit être aussi éloignée que possible des données les proches de chaque classe

Hyperplan de plus vaste marge



- Équation de l'hyperplan de séparation : $y = w^T x + b$
(Ligne droite dans un espace à deux dimensions)

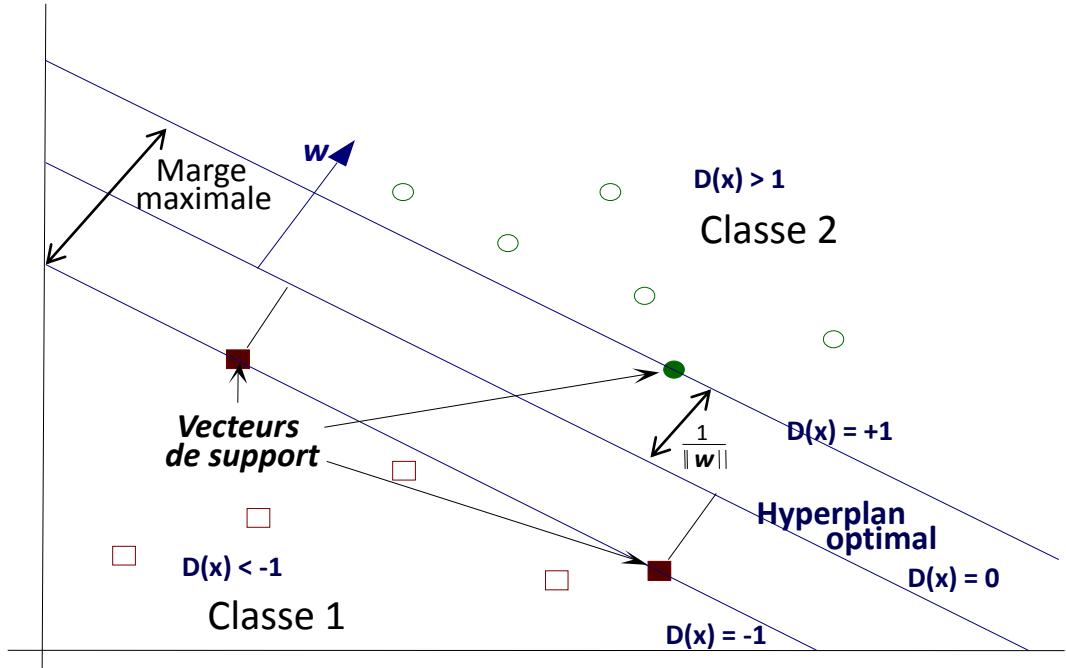
- ▶ Si $\{x_i\} = \{x_1, \dots, x_n\}$ est l'ensemble des données et $y_i \in \{1, -1\}$ est la classe de chacun $y_i(w^T x_i + b) \geq 1, \quad \forall i$

tout en ayant une distance optimale entre x_i et le plan de séparation

Optimisation de la marge

$$|\mathbf{w}^T \mathbf{x} + b| = 1$$

- ▶ Distance d'un point à l'hyperplan :
$$D(\mathbf{x}) = \frac{|\mathbf{w}^T \mathbf{x} + b|}{\|\mathbf{w}\|}$$
- ▶ Marge max. avant d'atteindre $m = \frac{2}{\|\mathbf{w}\|}$ unitières des deux classes :
- ▶ Maximiser m revient à minimiser $\|\mathbf{w}\|$ tout en préservant le pouvoir de classification :
$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 \text{ sous la contrainte } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i$$



Problème d'optimisation quadratique

- ▶ Maximiser le pouvoir de généralisation du classeur revient donc à trouver w et b tels que :

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

est minimum

et

$$y_i(w^T x_i + b) \geq 1, \quad i=1, \dots, n$$

- ▶ Si d est la dimension des x_i (nombre d'entrées), cela revient à régler $d+1$ paramètres (les éléments de w , plus b)
 - ▶ Possible par des méthodes d'optimisation classiques ([optimisation quadratique](#)) seulement si d pas trop grand (< qqs 10^3)

Les multiplicateurs de Lagrange en 30 s

- Problème : maximiser ou minimiser $f(x)$ sous la contrainte $g(x)=0$
- Solutions possibles :
 - Résoudre $g(x)=0$ et substituer la/les racines trouvées dans $f(x)$; résoudre alors $f'(x) = 0$: pas toujours facile !
 - Considérer $f(x)$ et $g(x)=0$ évoluent pareillement au voisinage de l'extrémum recherché. Leurs tangentes sont alors colinéaires et on a :
$$f'(x) = \alpha g'(x) \quad (\text{ou } f'(x) - \alpha g'(x) = 0)$$
 α étant à déterminer
 - La méthode des multiplicateurs de Lagrange regroupe la fonction à optimiser et la contrainte en une seule fonction $\Lambda(x, \alpha) = f(x) - \alpha g(x)$
 - La solution de $d\Lambda(x, \alpha)/dx = 0$ donne le point où les tangentes sont colinéaires; en même temps, $d\Lambda(x, \alpha)/d\alpha$ répond à la contrainte.

Cas à plusieurs dimensions

- On veut minimiser (ou maximiser) une fonction $f(\mathbf{x})$ en respectant des contraintes $g_i(\mathbf{x})=0, i=1,\dots,n$

- On peut monter qu'à l'extréumum recherché :
(égalité des tangentes)

$$\nabla f(\mathbf{x}) = \sum_{i=1}^n \alpha_i \nabla g_i(\mathbf{x})$$

ou encore : $\nabla f(\mathbf{x}) - \sum_{i=1}^n \alpha_i \nabla g_i(\mathbf{x}) = 0$ où les coefficients α_i sont à déterminer

- Si on forme la fonction (lagrangien) :
$$\Lambda(\mathbf{x}, \boldsymbol{\alpha}) = f(\mathbf{x}) - \sum_{i=1}^l \alpha_i g_i(\mathbf{x})$$

Alors : $\nabla_{\mathbf{x}} \Lambda(\mathbf{x}, \boldsymbol{\alpha}) = \nabla f(\mathbf{x}) - \sum_{i=1}^l \lambda_i \nabla g_i(\mathbf{x})$
et $\nabla_{\alpha_i} \Lambda(\mathbf{x}, \boldsymbol{\alpha}) = g_i(\mathbf{x})$

=> la solution de $\nabla_{\mathbf{x}} \Lambda(\mathbf{x}, \boldsymbol{\alpha}) = 0$ mène à un extréumum qui respecte les contraintes.

Forme duale du problème d'optimisation

- ▶ Dans notre cas, la fonction à optimiser sous contrainte est celle donnant la marge maximale, ce qui revient à trouver les paramètres (w, b) correspondants.
 - ▶ Donc, partant d'un ensemble de données $\{(x_i, y_i)\}$, de l'ensemble de contraintes (w, b) , on a:
- $$\begin{cases} \Lambda(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i \{(x_i^T w + b) y_i - 1\} \\ \forall i \quad \alpha_i \geq 0 \end{cases}$$
- $$\nabla_{w,b} \Lambda(w, b, \alpha) = 0$$

Le minimum recherché est donné par la solution de $\min_{\alpha} \max_x (\max_{\alpha} \Lambda(x, \alpha)) = \max_x (\min_{\alpha} \Lambda(x, \alpha))$

- ▶ Il existe un problème dual plus facile à résoudre :
 $\nabla_{w,b} \Lambda(w, b, \alpha) = 0$
 - Théorème de Kuhn-Tucker :

=> on peut aussi trouver w et b en solvant

sujet aux

Forme duale du problème d'optimisation

353

□ Avantage de résoudre $\nabla_{\alpha} \Lambda(\mathbf{w}, b, \boldsymbol{\alpha}) = 0$ au lieu de $\nabla_{\mathbf{w}, b} \Lambda(\mathbf{w}, b, \boldsymbol{\alpha}) = 0$

- La complexité du problème d'optimisation devient proportionnelle à n (nombre de paires d'apprentissage (\mathbf{x}_i, y_i)) et non d (dimension de chaque \mathbf{x}_i)
- Possible d'obtenir des solutions pour des problèmes impliquant $\approx 10^5$ exemples
- C'est aussi un problème pour lequel le maximum global des α_i peut toujours être trouvé

Formulation du problème dual

▶ Partant de $\begin{cases} \Lambda(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i \{ (\mathbf{x}_i^T \mathbf{w} + b) y_i - 1 \} \\ \forall i \quad \alpha_i \geq 0 \end{cases}$

$$\nabla_{\mathbf{w}, b} \Lambda(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \quad \text{donne} \quad \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

et $\Lambda(\mathbf{w}, b, \boldsymbol{\alpha})$

▶ On $\boxed{\begin{cases} \Lambda(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) \\ \forall i \quad \alpha_i \geq 0 \\ \sum_{i=1}^n \alpha_i y_i = 0 \end{cases}}$

$$\Lambda(\mathbf{w}, b, \boldsymbol{\alpha})$$

$$\nabla_{\boldsymbol{\alpha}} \Lambda(\mathbf{w}, b, \boldsymbol{\alpha}) = 0$$

Solution du problème d'optimisation

$$\left\{ \begin{array}{l} \hat{\mathbf{w}} = \sum_{i=1}^{n_s} \hat{\alpha}_i y_i \mathbf{x}_i \\ \hat{b} = y_s - \sum_{i=1}^{n_s} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}_s) \\ D(\mathbf{x}) = (\hat{\mathbf{w}}^T \mathbf{x} + \hat{b}) \end{array} \right.$$

- $\hat{\cdot}$: estimé
- n_s : nombre de vecteurs de support (\mathbf{x}_i avec $\alpha \neq 0$)
- (\mathbf{x}_s, y_s) : vecteur de support arbitraire (pour trouver $\hat{\mathbf{w}}$ et \hat{b})

- ▶ Les données \mathbf{x}_i avec $\alpha \neq 0$ sont appelées vecteurs de support. Ils correspondent aux points les plus proches de la surface de séparation $\hat{\mathbf{w}}$ et \hat{b}
- Dans l'expression du Lagrangien pour déterminer $\hat{\mathbf{w}}$ et \hat{b} , seuls interviennent les produits scalaires entre les données \mathbf{x}_i et \mathbf{x}_j

$$L(\mathbf{w}, b, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

Caractéristiques de la solution

- Puisque plusieurs α_i sont nuls, $\hat{\mathbf{w}}$ est une combinaison linéaire d'un petit nombre de données
- La surface de décision est uniquement déterminée par les n_s vecteurs de support trouvés:

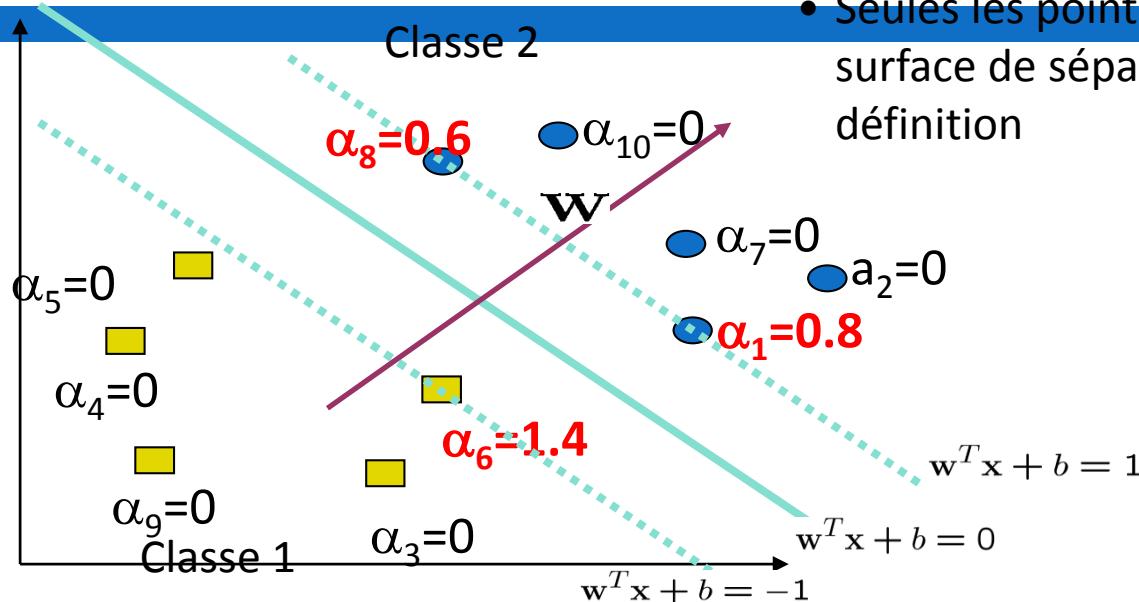
$$\hat{\mathbf{w}} = \sum_{i=1}^{n_s} \hat{\alpha}_i y_i \mathbf{x}_i$$

$$\hat{b} = y_s - \sum_{i=1}^{n_s} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}_s)$$

- Pour classer une nouvelle donnée \mathbf{z}
- Calculer $\hat{\mathbf{w}}^T \mathbf{z} + \hat{b} = \sum_{i=1}^{n_s} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{z}) + \hat{b}$ et classer \mathbf{z} dans la classe 1 si le résultat est positif, la classe 2 s'il est négatif

Interprétation géométrique

357



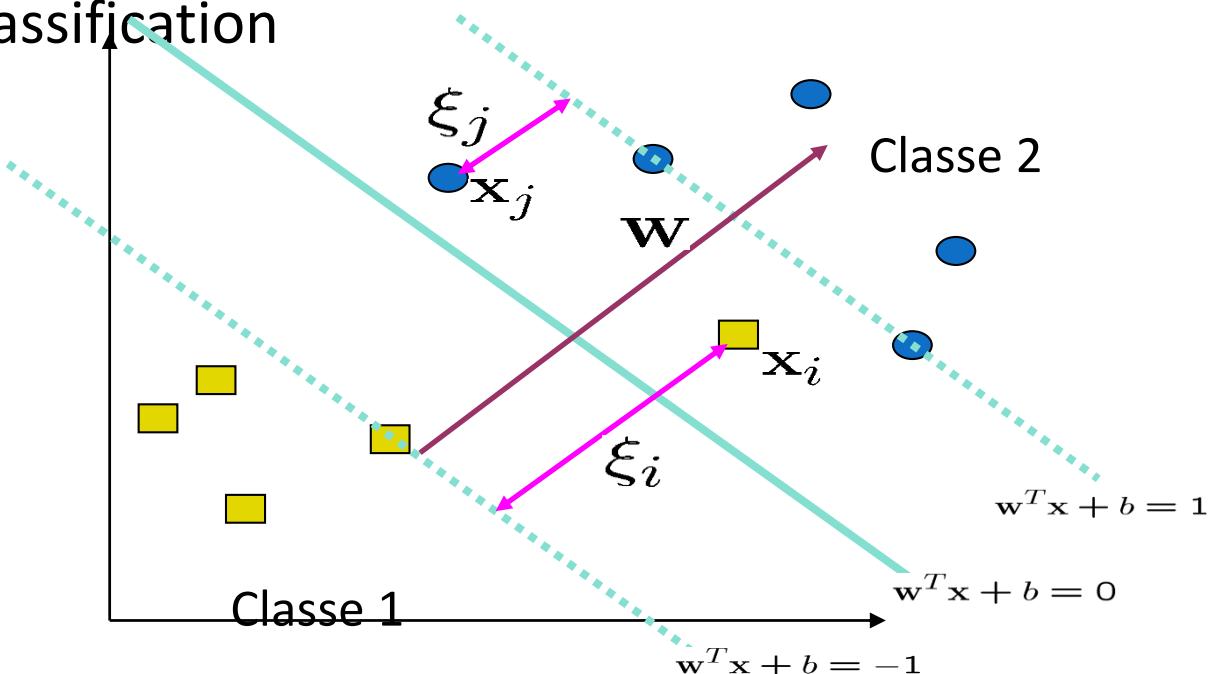
- Seules les points les plus proches de la surface de séparation influent sur sa définition

- Il existe des limites théoriques pour l'erreur de classification de données nouvelles
 - Plus grande la marge, plus petite la limite
 - Plus petit le nombre de SV, plus petite la limite

Et pour un cas non linéairement séparable ?

358

- On peut introduire une marge d'erreur ξ_i pour la classification



Hyperplan à marges douces

$\xi_i = 0$ s'il n'existe pas d'erreur pour x_i

- ξ_i sont des variables qui donnent du "mou" aux marges optimales

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$$

- Nous voulons minimiser

- C : paramètre de régularisation entre l'erreur et la marge
- Le problème d'optimisation devient

Détermination de l'hyperplan de séparation

360

- La forme duale du problème est

$$\max. \quad W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

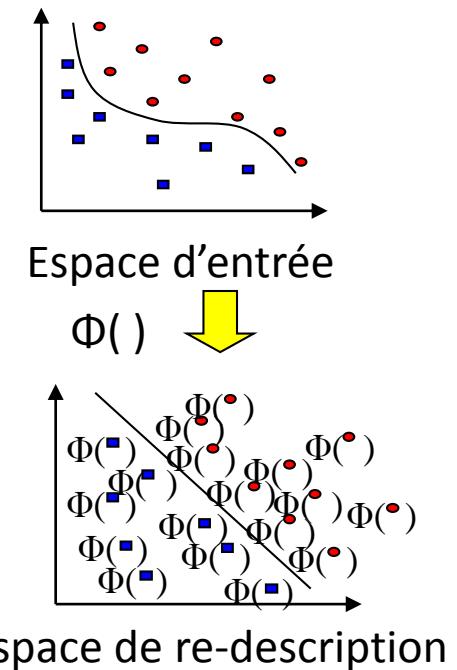
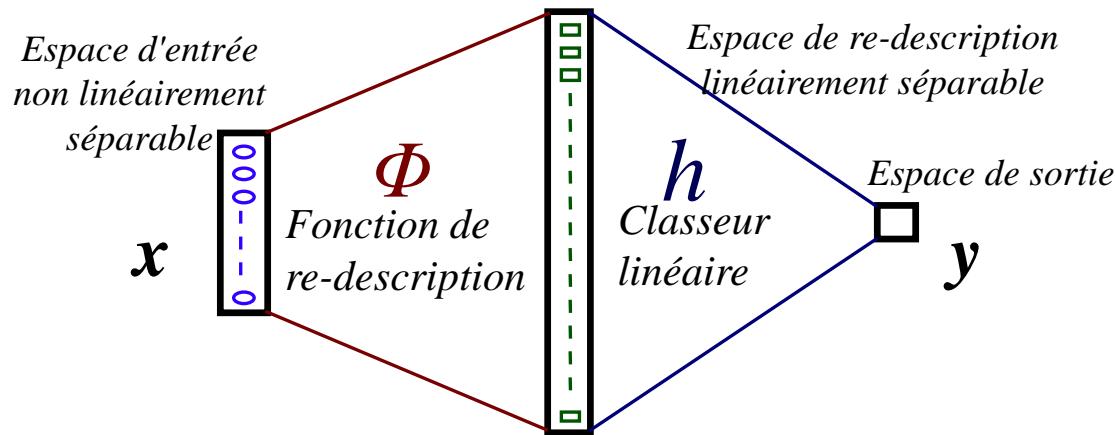
$$\text{subject to } C \geq \alpha_i \geq 0, \quad \sum_{i=1}^n \alpha_i y_i = 0$$

$$\mathbf{w} = \sum_{j=1}^s \alpha_j y_j \mathbf{x}_j$$

- \mathbf{w} est aussi donné par
- La seule différence avec le cas linéairement séparable est qu'il existe une limite supérieure C aux α_i

Extension à une surface de séparation non-linéaire

- « Simplifier les choses » en projetant les x_i dans un nouvel espace où ils sont linéairement séparables



Modification due à la transformation

- Substituer les arguments transformés dans les produits scalaires lors de la phase d'apprentissage,

Problème

original :

$$\max. \quad W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to $C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$

Après

xformation :

$$\max. \quad W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \boldsymbol{\Phi}(x_i)^T \boldsymbol{\Phi}(x_i)$$

subject to $C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$

- Mais trouver $\Phi()$ pas évident !

Modification due à la transformation

- Les nouvelles données \mathbf{z} sont toujours classées dans la classe 1 si $f \geq 0$, la classe 2 sinon :

Original :

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

Après
xformation :

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \Phi(\mathbf{x}_{t_j})^T \Phi(\mathbf{z}) + b$$

et la surface de séparation dans le nouvel espace est :

$$D(\mathbf{x}) = \sum_{j=1}^s \hat{\alpha}_j y_j K(\mathbf{x}_j, \mathbf{x}) + \hat{b}$$

Extension à une surface de séparation non-linéaire

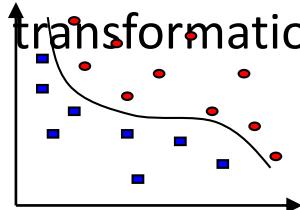
□ Problèmes cependant :

- ▶ $\Phi = ?$
- ▶ Grand effort de calcul potentiel (*d* explose !)

□ SVM à fonctions noyaux résout les deux problèmes

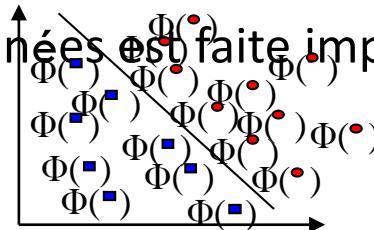
- Efficacité computationnelle

- La transformation désirée des données est faite implicitement !



$\Phi()$

Espace d'entrée



Espace de re-description

L'astuce des fonctions noyau

- Définition d'une fonction noyau :

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

- ⇒ La connaissance de $K(\cdot)$ permet de calculer indirectement un produit scalaire où intervient $\Phi(\cdot)$, sans connaître l'expression de $\Phi(\cdot)$
- Or, seuls des produits scalaires interviennent dans la solution du problème d'optimisation
 - Un autre avantage d'utiliser $K()$ est qu'il représente intuitivement la similarité entre les \mathbf{x} et \mathbf{y} , obtenue de nos connaissances a priori
 - Cependant, $K(\mathbf{x}, \mathbf{y})$ doit satisfaire certaines conditions (conditions de Mercer) pour que le $\Phi(\cdot)$ correspondant existe

Les conditions de Mercer

- Pour une fonction K symétrique, il existe une fonction Φ telle que :

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') = \sum_{i=1}^m g_i(\mathbf{x}) \cdot g_i(\mathbf{x}')$$

$\int f(\mathbf{x})^2 d\mathbf{x}$ est fini

ssi, pour toute fonction f telle que $d\mathbf{x} d\mathbf{x}' \geq 0$

l'on a :

- Si cette condition est vérifiée, on peut appliquer la fonction noyaux dans le SVM

Exemple de d'utilisation

- Définissons la fonction noyau $K(\mathbf{x}, \mathbf{y})$ telle que, pour toute paire de vecteurs $\mathbf{x}=(x_1, x_2)$ et $\mathbf{y}=(y_1, y_2)$:

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1 y_1 + x_2 y_2)^2$$

- Considérons maintenant une transformation Φ qui prend un vecteur de dimension 2 et le projette dans un espace de dimension 6 :

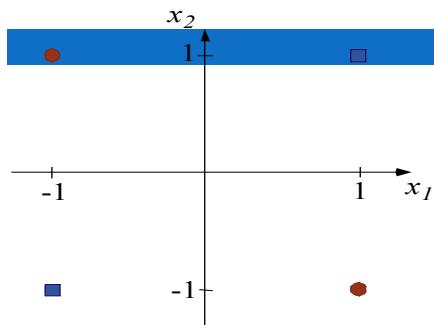
$$\Phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$

On peut voir en effectuant le calcul que

$$\begin{aligned}\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle &= (1 + x_1 y_1 + x_2 y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y})\end{aligned}$$

On peut donc obtenir le résultat sans avoir à passer par l'espace transformé

Illustration : le cas du XOR



□ Il faut résoudre :

$$\begin{cases} \max_{\alpha} \left(\sum_{i=1}^4 \alpha_i - \frac{1}{2} \sum_{i=1}^4 \sum_{j=1}^4 \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) \\ \forall i \quad 0 \leq \alpha_i \leq C \\ \sum_{i=1}^4 \alpha_i y_i = 0 \end{cases}$$

Index i	\mathbf{x}_i	y
1	(1,1)	1
2	(1,-1)	-1
3	(-1,-1)	1
4	(-1,1)	-1

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1 y_1 + x_2 y_2)^2$$

□ $Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$
 Si on reprend la fonction noyau
 on obtient les équations suivantes pour le Lagrangien
 :
$$+ 9\alpha_2^2 - 2\alpha_2\alpha_3 + 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2$$

$$\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 = 0$$

Illustration : le cas du XOR

369

- Le maximum de $Q(\alpha)$ est obtenu en prenant ses dérivées par rapport aux α_i et en trouvant les valeurs de α_i qui les annulent :

$$\begin{cases} 1 - 9\alpha_1 + \alpha_2 - \alpha_3 + \alpha_4 = 0 \\ 1 + \alpha_1 - 9\alpha_2 + \alpha_3 - \alpha_4 = 0 \\ 1 - \alpha_1 + \alpha_2 - 9\alpha_3 - \alpha_4 = 0 \\ 1 + \alpha_1 - \alpha_2 + \alpha_3 - 9\alpha_4 = 0 \end{cases}$$

- La valeur optimale des multiplicateurs de Lagrange est :

$$\hat{\alpha}_1 = \hat{\alpha}_2 = \hat{\alpha}_3 = \hat{\alpha}_4 = \frac{1}{8}$$

- Les 4 données du où exclusif sont donc des vecteurs de support, puisque aucune valeur trouvée de α n'est nulle

Illustration : le cas du XOR

- Dans l'espace de Espace de re-description :

$$\begin{cases} \hat{\mathbf{w}} = \sum_{i=1}^{n_s} \hat{\alpha}_i y_i \Phi(\mathbf{x}_i) \\ \hat{b} = y_s - \sum_{i=1}^{n_s} \hat{\alpha}_i y_i K(\mathbf{x}_i^T \mathbf{x}_s) \\ D(\mathbf{x}) = \sum_{j=1}^s \hat{\alpha}_j y_j K(\mathbf{x}_j, \mathbf{x}) + \hat{b} \end{cases}$$

- Donc :

$$\begin{aligned} \hat{\mathbf{w}} &= \frac{1}{8} [-\Phi(\mathbf{x}_1) + \Phi(\mathbf{x}_2) + \Phi(\mathbf{x}_3) - \Phi(\mathbf{x}_4)] \\ &= \frac{1}{8} \left\{ -\begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{pmatrix} \right\} = \begin{pmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{aligned}$$

(on connaît $\Phi()$ dans cet exemple, mais il n'est pas requis en général, car l'équation de la marge dépend seulement de $K()$)

$$\hat{b} = 1 - \frac{1}{8} \sum_{j=1}^4 y_j K(\mathbf{x}_j, \mathbf{x}_1) = 1 + \frac{1}{8} \sum_{j=1}^4 (-1)^j K(\mathbf{x}_j, \mathbf{x}_1) = 0$$

et $D(\mathbf{x}) = \frac{1}{8} \sum_{j=1}^4 y_j K(\mathbf{x}_j, \mathbf{x}) = -\frac{1}{8} \sum_{j=1}^4 (-1)^j K(\mathbf{x}_j, \mathbf{x}) = -x_1 x_2$

(on aurait obtenu le même résultat en utilisant $\Phi()$) :

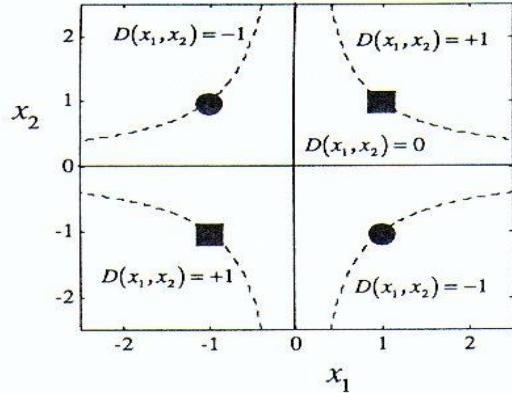
$$\hat{\mathbf{w}}^T \Phi(\mathbf{x}) = \left(0, 0, \frac{-1}{\sqrt{2}}, 0, 0, 0 \right) \begin{pmatrix} 1 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \\ \sqrt{2} x_1 \\ \sqrt{2} x_2 \end{pmatrix} = -x_1 x_2$$

$\frac{1}{2} \|\hat{\mathbf{w}}\|^2 = \frac{1}{2} \hat{\mathbf{w}}^T \hat{\mathbf{w}} = \frac{1}{2} \left(\sum_{i=1}^4 \hat{\alpha}_i y_i \Phi(\mathbf{x}_i) \right)^T \left(\sum_{j=1}^4 \hat{\alpha}_j y_j \Phi(\mathbf{x}_j) \right)$

► La marge optimale est : $\frac{1}{2} \cdot \frac{4}{8^2} \sum_{i=1}^4 \sum_{j=1}^4 y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{4} \Rightarrow \|\hat{\mathbf{w}}\| = \frac{1}{\sqrt{2}}$

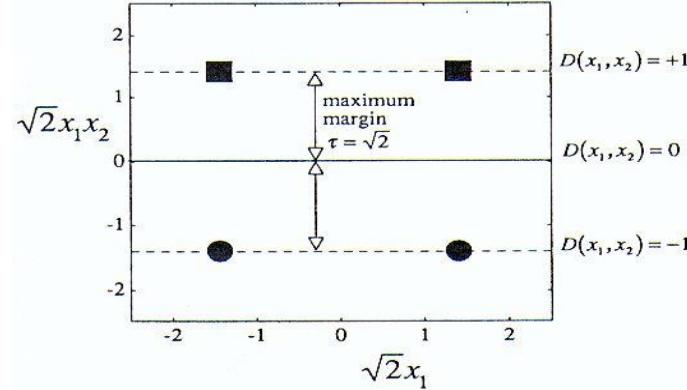
Illustration : le cas du XOR

371



Séparatrice dans l'espace
d'entrée

$$D(x) = -x_1 x_2$$



Séparatrice dans l'espace

$$\sqrt{2} \Phi(x) = 0$$

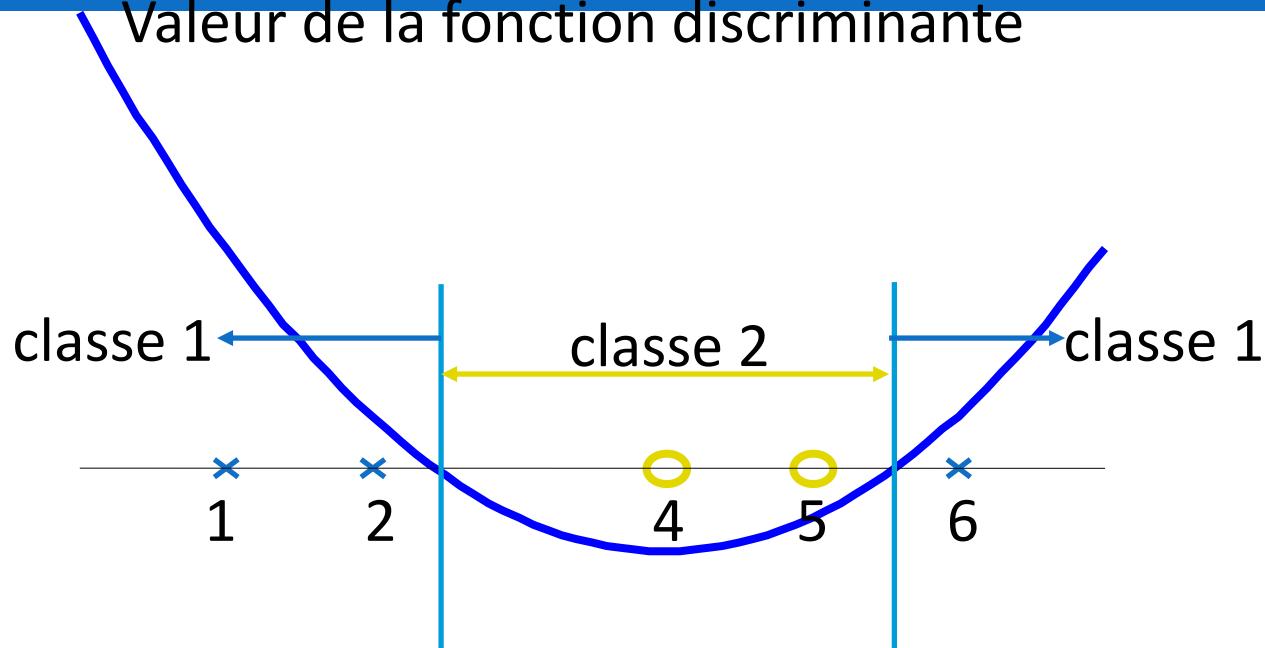
Autre Exemple

- Supposons 5 nombres $x_1=1, x_2=2, x_3=4, x_4=5, x_5=6$, avec
 - ▣ $1, 2, 6 \in$ classe 1 ($y=1$)
 - ▣ $4, 5 \in$ classe 2 ($y=-1$)
 - ▣ Donc: $\{(x_i, y_i)\}_{i=1, \dots, 5} = \{(1, 1), (2, 1), (4, -1), (5, -1), (5, 1)\}$
- Utilisons à nouveau le noyau polynomial de degré 2
 - ▣ $K(x, y) = (1 + x^T y)^2$
 - ▣ $\max_{\alpha_i} \sum_{i=1}^5 \alpha_i - \frac{1}{2} \sum_{i=1}^5 \sum_{j=1}^5 \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$
- Trouver α_i subject to $100 \geq \alpha_i \geq 0, \sum_{i=1}^5 \alpha_i y_i = 0$

Exemple

373

Valeur de la fonction discriminante



Exemples de fonctions noyaux

374

- Noyau polynomial de degré d

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- Noyau à fonction à base radiale de dispersion σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2 / (2\sigma^2))$$

- Très proche des RN avec fonctions à base radiale

- Sigmoïde avec paramètres κ et θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x}^T \mathbf{y} + \theta)$$

- Ne satisfait pas la condition de Mercer pour tous κ et θ

- La recherche d'autres fonctions noyau pour diverses applications est très active !

Classification multi-classes

375

- SVM est à la base un classifieur binaire
- On peut changer la formulation pour permettre la classification multi-classe
 - ▣ L'ensemble des données est divisé en deux parts de multiples façons, et classé ensuite
 - Un contre tous ou un contre chaque alternative
 - Un SVM séparé est formé pour chaque division
 - ▣ La classification multi-classes est accomplie en combinant la sortie de tous les SVM

Sommaire: étapes de la classification

376

- Préparer la matrice des patrons
- Choisir la fonction noyau à utiliser
- Choisir les paramètres de la fonction noyau et la valeur de C (valeurs suggérées par le logiciel SVM ou essai-erreur).
- Exécuter l'algorithme d'apprentissage pour trouver α_i

- Les données nouvelles peuvent être classées en fonctions des α_i et des vecteurs supports trouvés

Effet des paramètres de contrôle.

377

- Apprentissage de données en damier
 - Apprentissage de deux classes
 - SVM à fonction noyau gaussienne

$$K(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}}$$

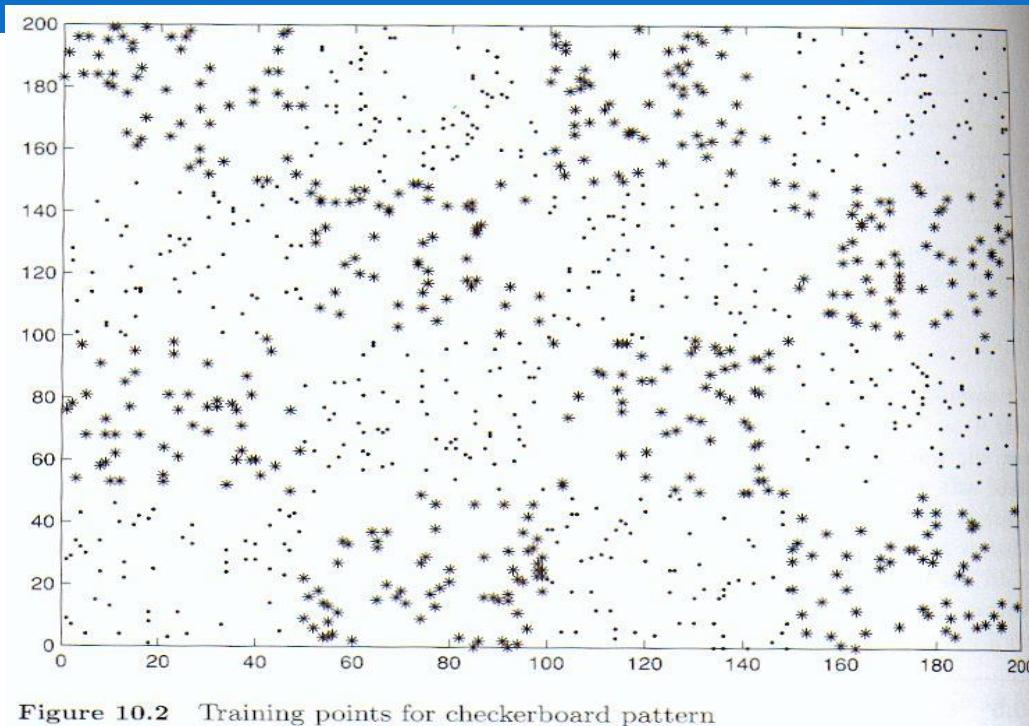


Figure 10.2 Training points for checkerboard pattern

Effet des paramètres de contrôle

378

- Apprentissage de deux classes

- exemples tirés uniformément sur l'échiquier

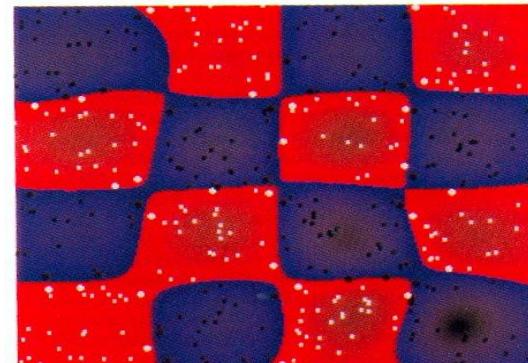
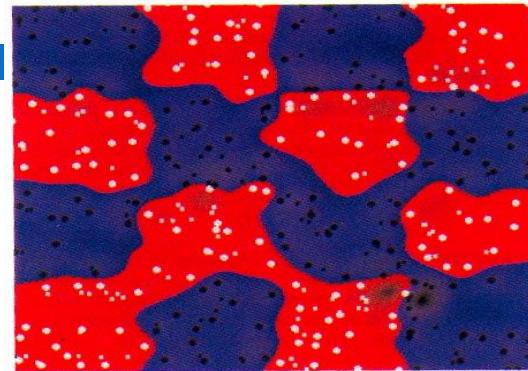
- SVM à fonctions noyau gaussienne

$$K(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}}$$

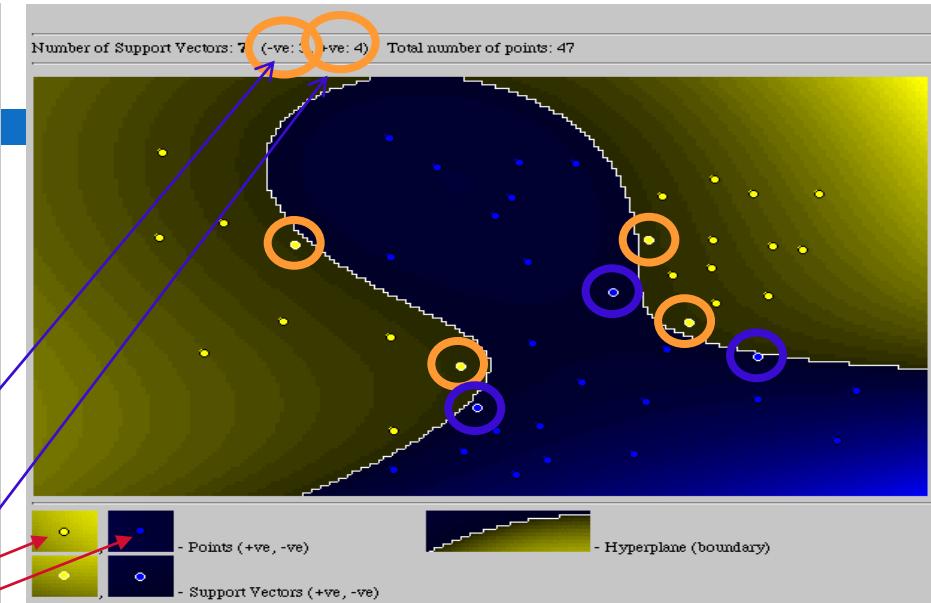
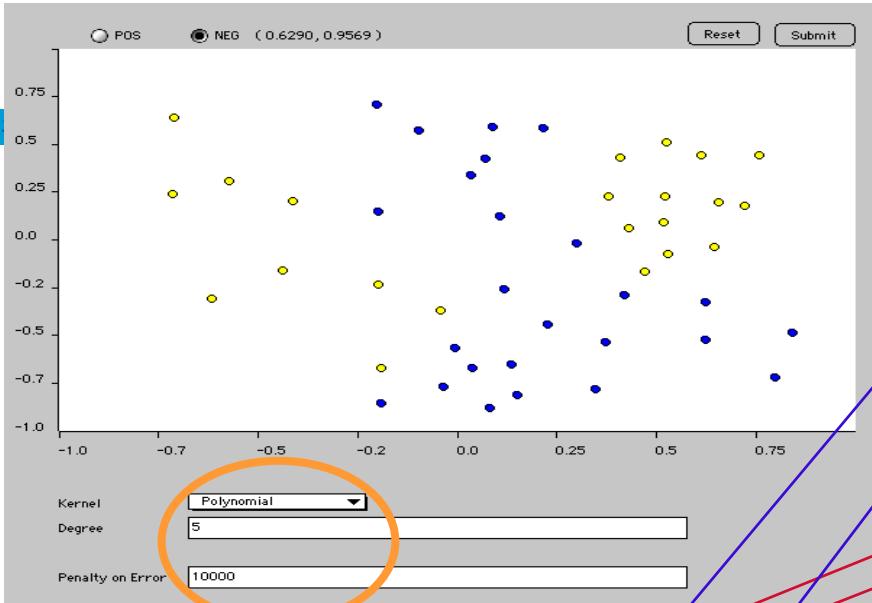
- Ici deux valeurs de σ

- En haut : petite valeur
 - En bas : grande valeur

- Les gros points sont des exemples critiques



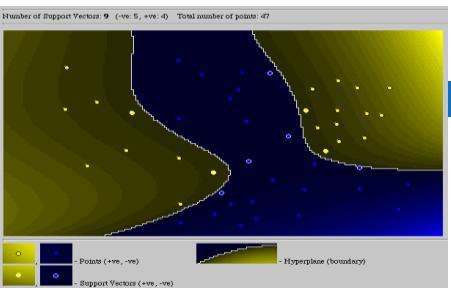
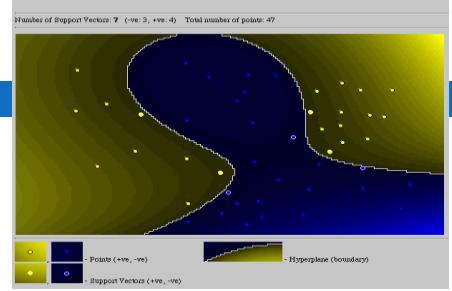
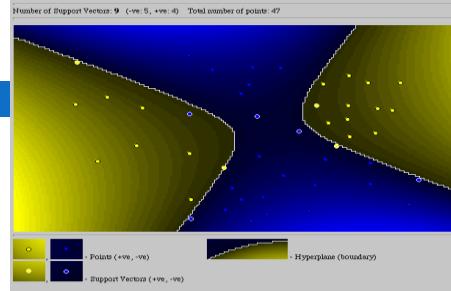
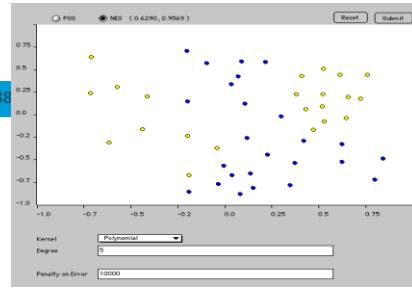
Une applette de démonstration



- <http://svm.cs.rhul.ac.uk/pagesnew/GPat.shtml>
- 47 exemples (22 +, 25 -)
- Exemples critiques : 4 + et 3 -
- Ici *fondction polynomiale* de degré 5 et $C = 10000$

Paramètres de contrôle : les fonctions noyau

38



(5-, 4+)

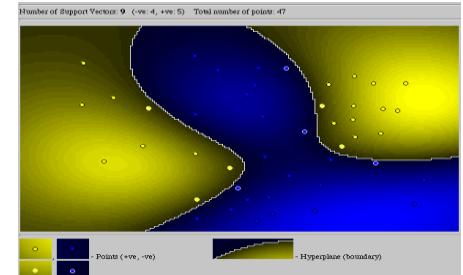
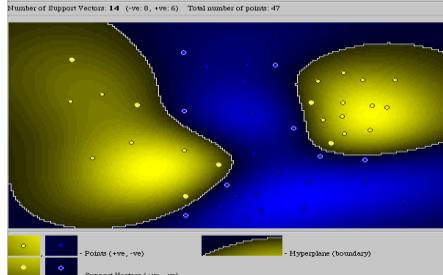
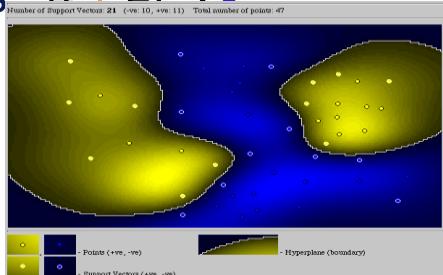
(3-, 4+)

(5-, 4+)

Ici *fonction polynomiale* de degré 2, 5, 8 et $C = 10000$

- 47 exemples (22 +, 25 -)

- *Exemples critiques* · 1 + et 3 -



(10-, 11+)

(8-, 6+)

(4-, 5+)

Ici *fonction Gaussienne* de $\sigma = 2, 5, 10, 20$ et $C = 10000$

Domaines d'application des SVMs

□ Traitement d'images

- Reconnaissance de caractères manuscrits
 - Reconnaissance de scènes naturelles
 - Reconnaissance de visages
-
- ▣ *Entrées :* image bidimensionnelle en couleur ou en tons de gris
 - ▣ *Sortie :* classe (chiffre / personne)

Domaines d'application des SVMs

□ Catégorisation de textes

- Classification d'e-mails
- Classification de pages web
- *Entrées* : document (texte ou html)
 - Approche « sac de mots »
 - Document = vecteur de mots (lemmatisés pondérés par tf-idf)
- *Sortie* : catégorie (thème, spam/non-spam)
- **Noyau** :
 - Produit scalaire des vecteurs
 - $C = \infty$ (marge dure)

Domaines d'application des SVMs

□ Diagnostic médical

- Évaluation du risque de cancer
- Détection d'arythmie cardiaque
- Évaluation du risque d'accidents cardio-vasculaires à moins de 6 ans
- *Entrées* : état du patient (sexe, age, bilan sanguin, ...)
- *Sortie* :
 - Classe : à risque ou non
 - Probabilité d'accident à échéance donnée

Extensions

□ Leçon à retenir des SVM:

- *Un algorithme linéaire dans l'espace de re-description peut remplacer un algorithme non-linéaire dans l'espace d'entrée*
- Les algorithmes linéaires classiques peuvent être généralisés en des versions non-linéaires en allant vers l'espace de re-description
 - ACP à noyaux, k-moyennes à noyaux, etc.
 - Régression
 - Détection de « nouveautés »

SVM et régression

□ Fonction de perte : $|y - f(\mathbf{x})|_\varepsilon = \max\{0, |y - f(\mathbf{x})| - \varepsilon\}$

■ Régression linéaire :

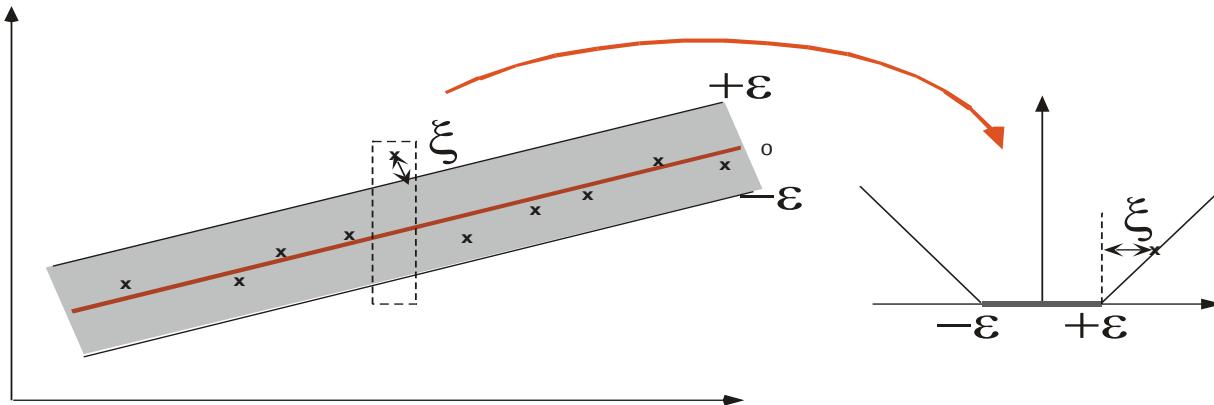
$$f(\mathbf{x}) := (\mathbf{w} \cdot \mathbf{x}) + w_0$$

■ Soit à minimiser :

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m |y_i - f(\mathbf{x}_i)|_\varepsilon$$

■ Généralisation :

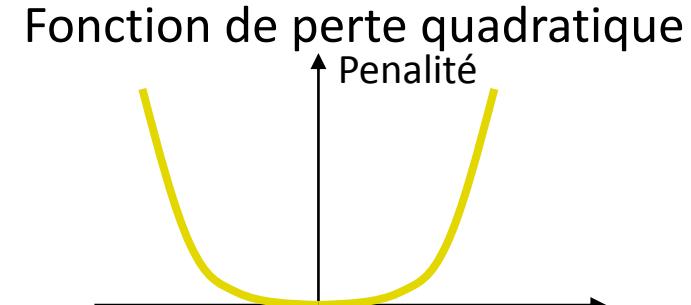
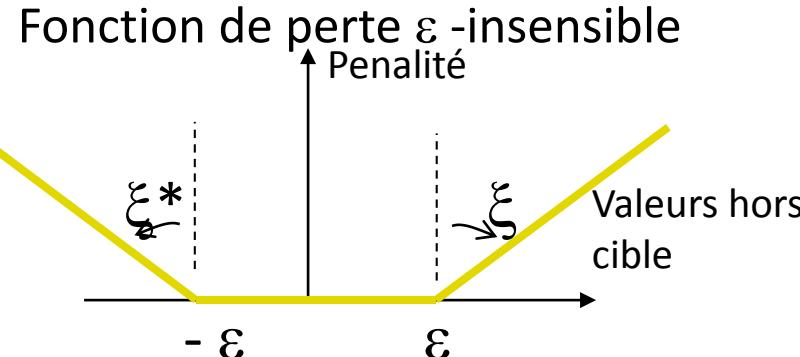
$$f(\mathbf{x}) = \sum_{i=1}^m (\alpha_i^\star - \alpha_i) K(\mathbf{x}_i, \mathbf{x}) + w_0$$



Régression vectorielle à support ε (ε -SVR)

386

- Régression linéaire dans l'espace de redescription
- À l'encontre de la régression par moindres carrés, la fonction d'erreur est une fonction de perte ε -insensible
 - Intuitivement, une erreur inférieure à ε est ignorée
 - Cela mène à des points de marge terse similaires à SVM



Régression vectorielle à support ε (ε -SVR)

387

- Soit un ensemble de données $\{x_1, \dots, x_n\}$ avec valeurs cibles $\{u_1, \dots, u_n\}$, on veut réaliser ξ -SVR
- Le problème d'optimisation est

$$\text{Min } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*)$$

subject to
$$\begin{cases} u_i - \mathbf{w}^T \mathbf{x}_i - b \leq \epsilon + \xi_i \\ \mathbf{w}^T \mathbf{x}_i + b - u_i \leq \epsilon + \xi_i^* \\ \xi_i \geq 0, \xi_i^* \geq 0 \end{cases}$$

- Formulation similaire à SVM, donc peut être résolu en tant que problème QP

Régression vectorielle à support ε (ε -SVR)

388

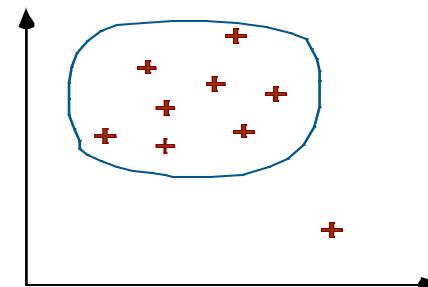
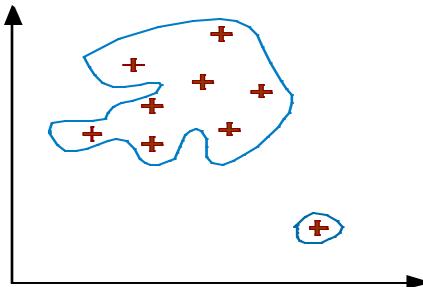
- C permet de contrôler l'influence de l'erreur
- Le terme $\frac{1}{2} \|w\|^2$ sert à contrôler la complexité de la fonction de régression
- Après l'apprentissage (solution du problème QP), on trouve les valeurs α_i and α_i^* , qui sont toutes deux zéros si x_i ne contribue pas au support.
- Pour ce faire, on a :

$$f(z) = \sum_{j=1}^s (\alpha_{t_j} - \alpha_{t_j}^*) K(x_{t_j}, z) + b$$

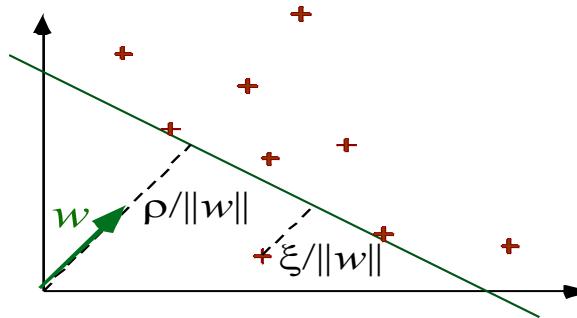
SVM et apprentissage non supervisé

□ Détection de « nouveautés »

389



On cherche à séparer au maximum le nuage de points de l'origine



Pourquoi ça marche ?

390

La marge est liée à la capacité en généralisation

- Normalement, la classe des hyperplans de \mathbb{R}^d est de $d_H = d + 1$
- Mais la classe des hyperplans de marge
est bornée par :
$$\frac{1}{\|\omega\|} \text{ tq. } \|\omega\|^2 \leq c$$
$$d_H \leq \text{Min}(R^2 c, d) + 1$$
- où R est le rayon de la plus petite sphère englobant l'échantillon
d'apprentissage S
- ↳ Peut être beaucoup plus petit que la dimension d de l'espace d'entrée X

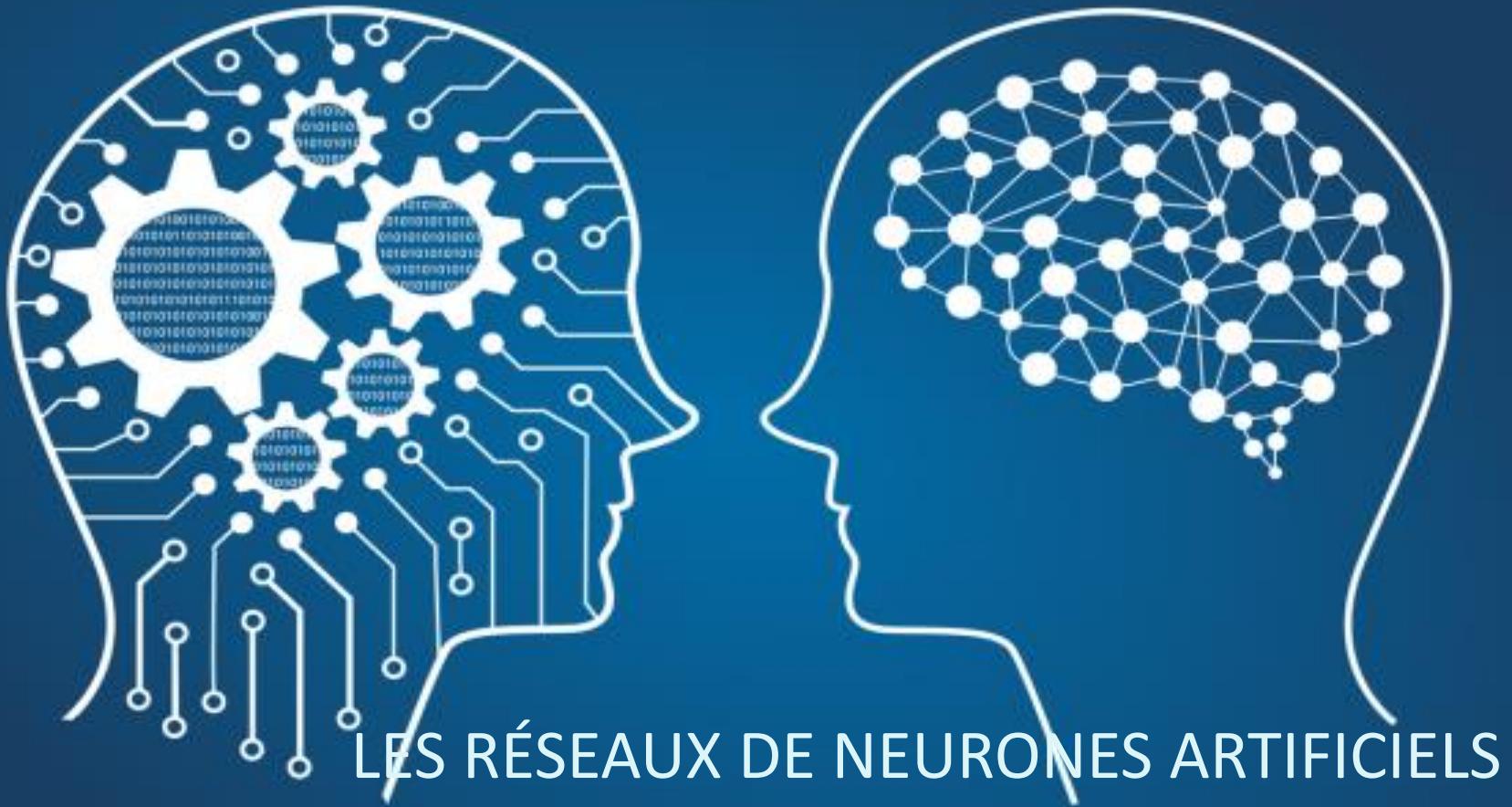
Forces et faiblesses des SVM

□ Forces

- L'apprentissage est relativement facile
 - Pas de minima locaux, comme pour les RNA
- L'algorithme est robuste face aux changements d'échelle
- Le compromis entre la complexité du classifieur et l'erreur de classification peut être gérée explicitement
- Méthode générale
 - Des données non conventionnelles, telles des chaînes et des arbres peuvent servir d'entrées au SVM, à la place des vecteurs de traits
- Résultats en général **équivalents et souvent meilleurs**

□ Faiblesses

- Il faut trouver la “bonne” fonction noyau
- Problèmes i.i.d. (données indépendantes et identiquement distribuées)
- Deux classes à la fois



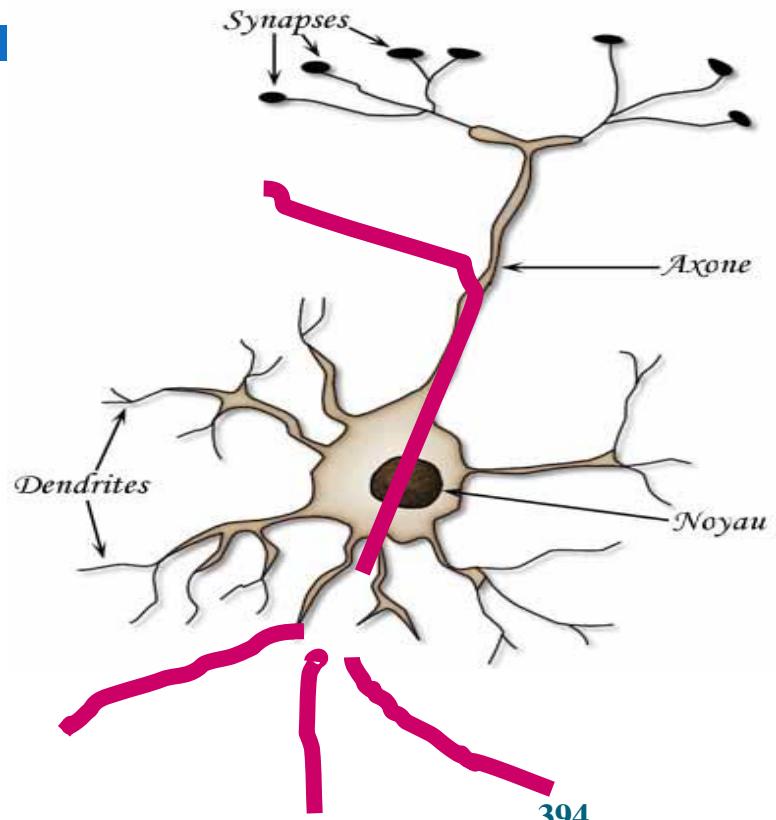
LES RÉSEAUX DE NEURONES ARTIFICIELS

Réseaux de neurones

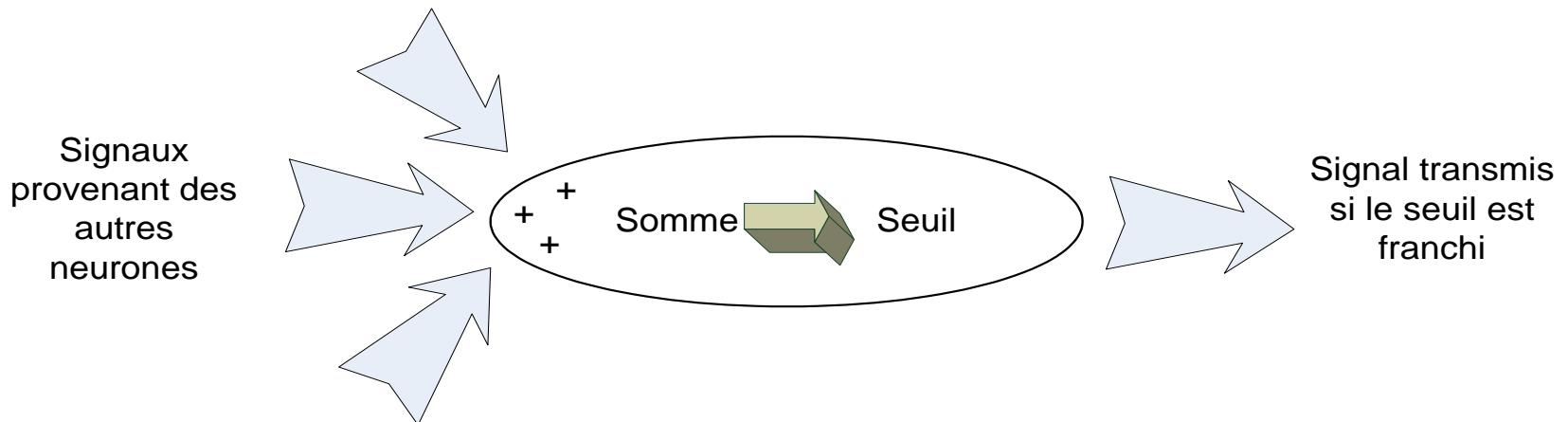
- Tentative de reproduction des structures du cerveau afin de raisonner
- Ensemble d'unités transformant des entrées en sorties (neurones) connectées, où chaque connexion à un poids associé
- La phase d'apprentissage permet d'ajuster les poids pour produire la bonne sortie (la classe en classification)

Analogie avec le cerveau

- Le cerveau humain contient environ 100 milliards de neurones, et chacun est connecté à environ 10.000 autres
- Un neurone reçoit des impulsions électriques de ses voisins via les dendrites. Si la somme des signaux dépasse un certain seuil, il se produit une décharge électrique de type tout ou rien appelée potentiel d'action. Le potentiel d'action se propage le long de l'axone, qui se ramifie en une multitude de dendrites.
- La terminaison d'une dendrite est une petite usine de production chimique. Elle diffuse des neurotransmetteurs chimiques dans un espace appelé synapse, qui rejoint un autre neurone.

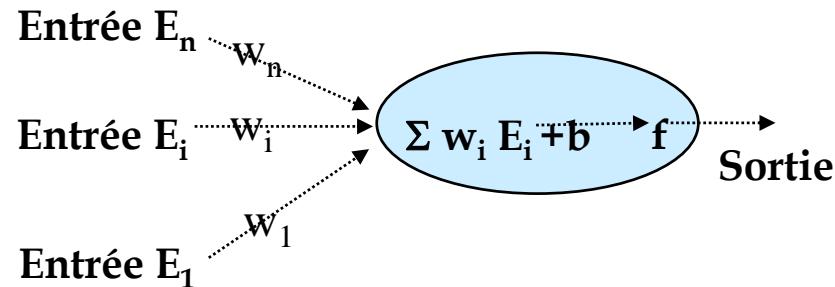


Modélisation du neurone

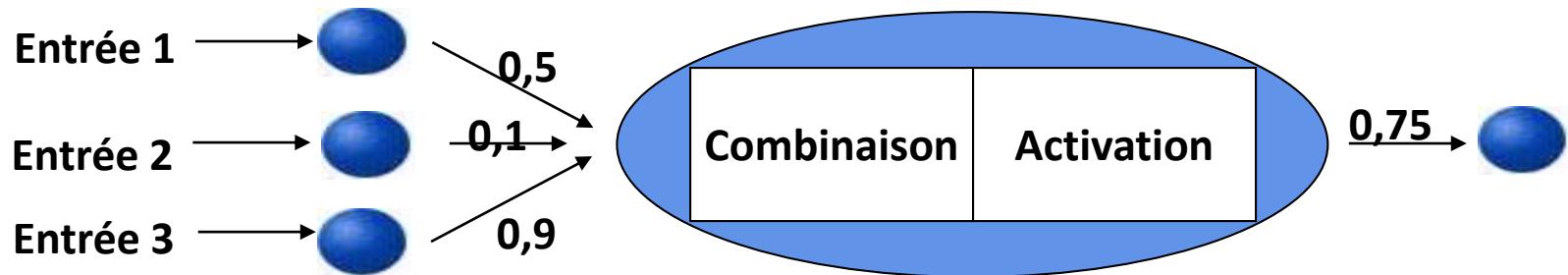


Plus précisément ...

- Induit une valeur en sortie à partir d'un ensemble de valeurs en entrée
- Les liens sont pondérés par des poids
- Réalise une combinaison linéaire des entrées suivie d'une fonction de transfert (fonction à seuil)
 - Fonction Sigma ($\sum w_i E_i$)
 - Biais optionnel b
 - Fonction Sigmoïde $f(\sum) = 1/(1+e^{-\sum})$



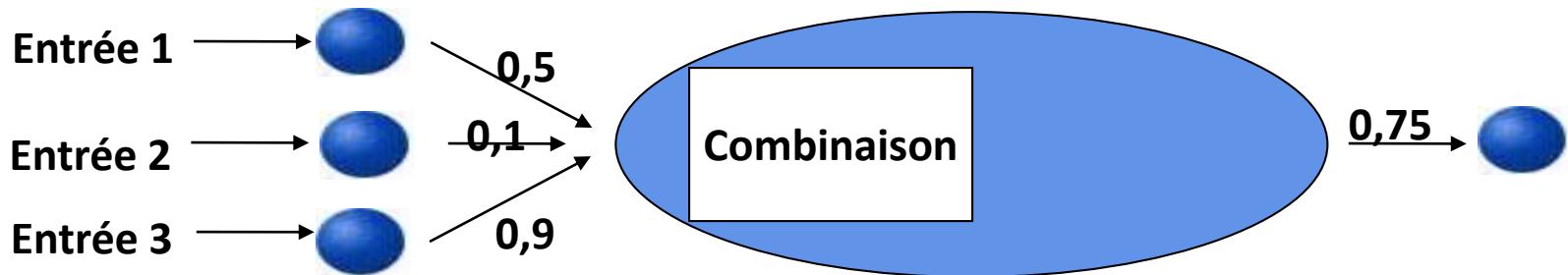
Combinaison/Activation



Phase de combinaison : combine les entrées et produit une valeur en sortie

Phase d'activation : prend en entrée la sortie de la fonction de combinaison et déduit la valeur de sortie

Combinaison



Fonctions de combinaison :

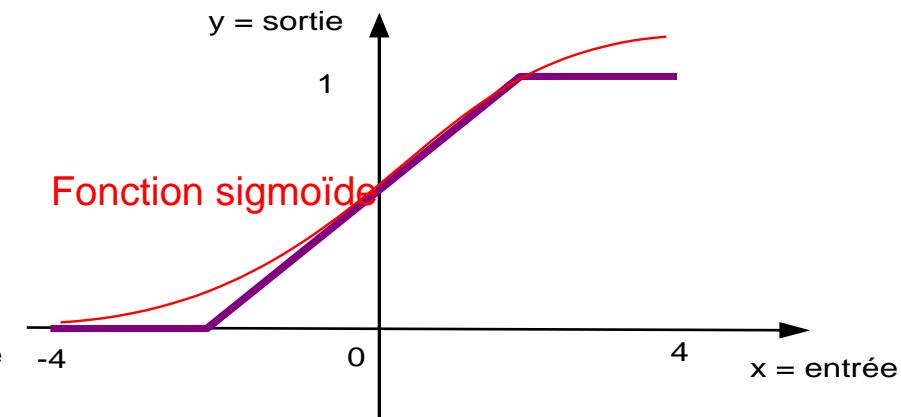
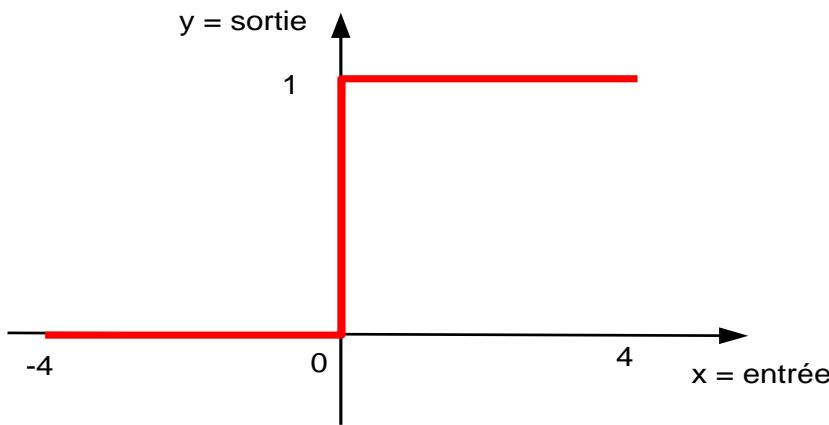
- Produit scalaire
- Norme euclidienne

$$\begin{array}{c} \xrightarrow{\hspace{1cm}} \\ \left(\begin{array}{c} E1 \\ E2 \\ E3 \end{array} \right) \end{array} \bullet \begin{array}{c} \xrightarrow{\hspace{1cm}} \\ \left(\begin{array}{c} 0,5 \\ 0,1 \\ 0,9 \end{array} \right) \end{array} \quad \left\| \left(\begin{array}{c} E1 \\ E2 \\ E3 \end{array} \right) \right\|$$

Activation

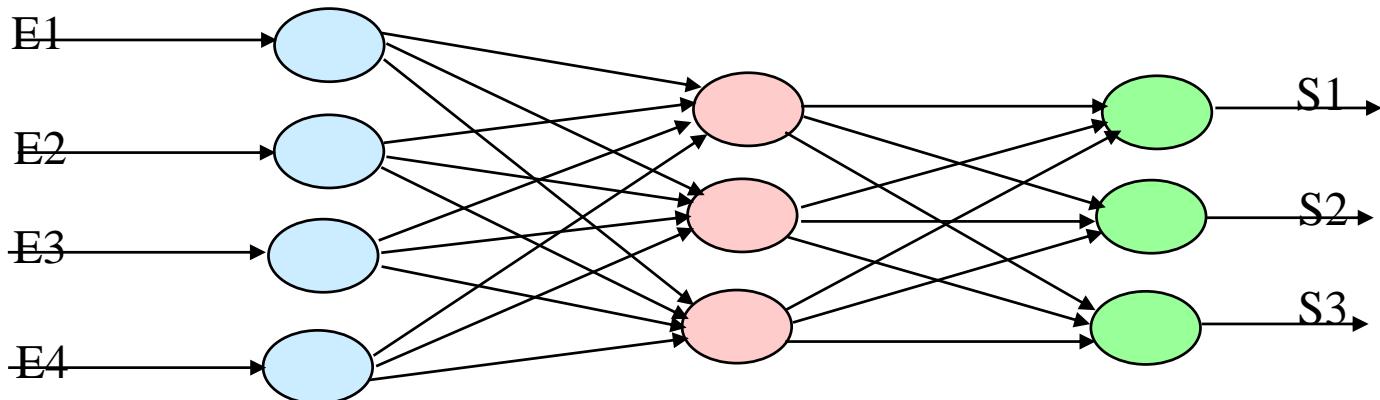
Trois intervalles :

- en dessous du seuil : neurone non actif
- aux alentours du seuil : phase de transition
- au dessus du seuil : neurone actif



Organisation en réseau

- Réseau multi-couches totalement connecté
- Entrées, Calculs (cachés), Sorties

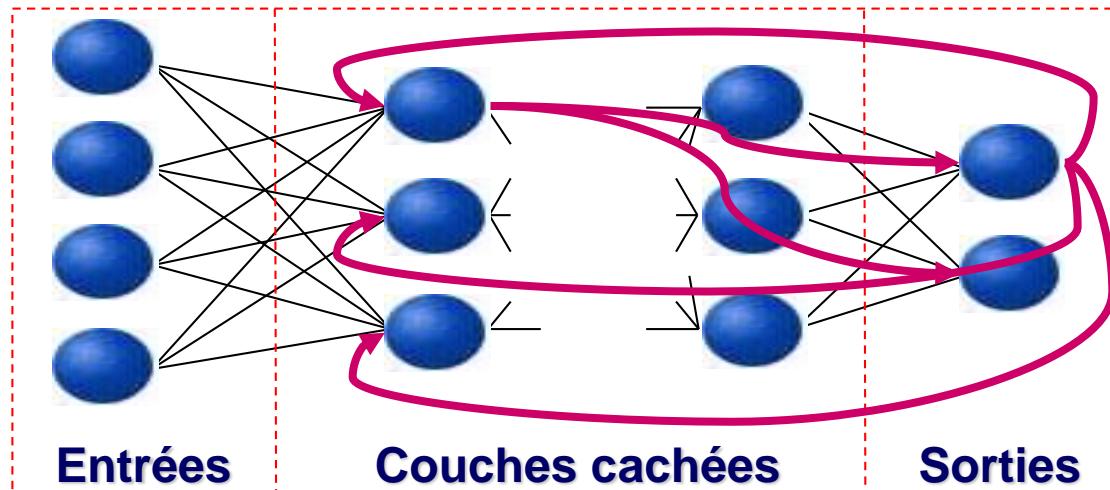


Topologie

□ Choix du nombre de couches

- entrées, 1 ou 2 couches cachées, sorties
- Choix du nombre de neurones par couche
 - dépend des entrées et sorties
 - couches cachées intermédiaires
- Normalisation des variables d'entrées
 - Variable continue centrée réduite $[-1,+1]$
 - Variable discrète codée ou valeurs attribuées aux entrées
 - Sorties booléenne codant les classes

Perceptron multicouche



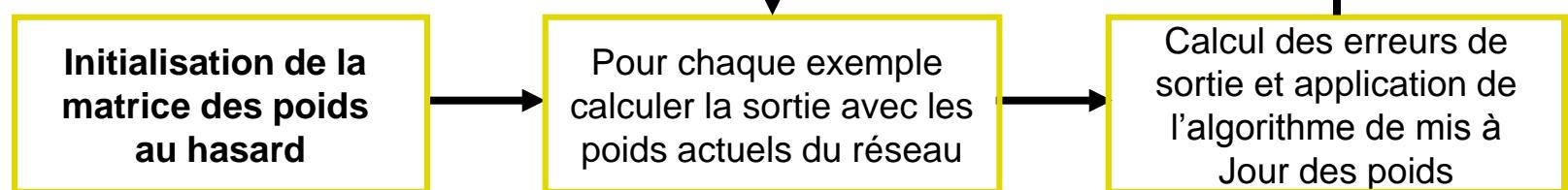
Apprentissage

Découverte de modèles complexes avec affinage progressif

- Le réseau s'adapte lors de la phase d'apprentissage
- Plusieurs algorithmes possibles
 - le plus utilisé = rétropropagation
 - modification des poids w_i par rétropropagation

Principe

- Off-Line ou Batch : après tous les exemples
- On-Line ou Stochastique : après chaque exemple



Rétropropagation

Initialiser les poids et les biais

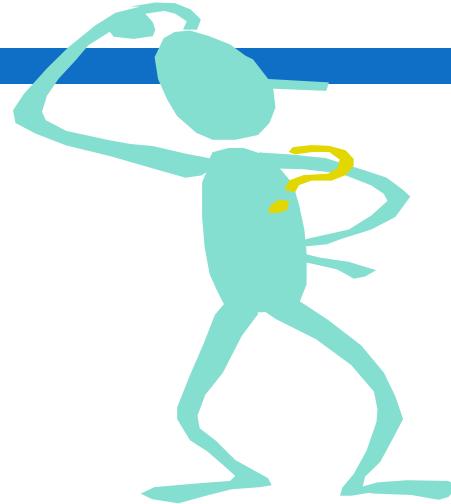
- tirage aléatoire sur [-1,+1]
- Propager les entrées en avant
 - Un exemple est appliqué aux entrées
 - Le réseau calcule les sorties
- Propager les erreurs en arrière
 - Sortie devant délivrer T : $\text{Err} = O(1-O)(T-O)$
 - Cellule cachée : $\text{Err} = O(1-O) \sum_k w_k * \text{Err}_k$
- Corriger poids et biais de sorte à réduire les erreurs
 - $\Delta w_{ij} = \lambda * \text{Err}_j * O_i ; \Delta b_j = \lambda * \text{Err}_j$

Forces et Faiblesses

- Permet d'approcher toute sorte de fonction
- Coûteux en apprentissage
 - ▣ calculs complexes
 - ▣ possibilité d'élaguer le réseau en connexions
 - ▣ peu applicable sur de larges BD
- Effet boite noire
 - ▣ comportement difficile à expliquer
- Autres applications possibles
 - ▣ prédiction, décodage, reconnaissance de formes, etc.

Bilan Classification

- De nombreuses techniques dérivées de l'IA et des statistiques
 - Autres techniques
 - règles associatives, raisonnement par cas, ensembles flous, ...
 - Problème de passage à l'échelle
 - arbre de décisions, réseaux
 - Tester plusieurs techniques pour résoudre un problème
-
- Y-a-t-il une technique dominante ?





LÉ CLUSTERING

Qu'est ce qu'un bon regroupement ?

- Une bonne méthode de regroupement permet de garantir
 - Une grande similarité intra-groupe
 - Une faible similarité inter-groupe
- La qualité d'un regroupement dépend donc de la mesure de similarité utilisée par la méthode et de son implémentation

Structures de données

Matrice de données

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{if} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nf} & \dots & x_{np} \end{bmatrix}$$

Matrice de similarité

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

Mesurer la qualité d'un clustering

- Métrique pour la similarité: La similarité est exprimée par le biais d'une mesure de distance
- Une autre fonction est utilisée pour la mesure de la qualité
- Les définitions de distance sont très différentes que les variables soient des intervalles (continues), catégories, booléennes ou ordinaires
- En pratique, on utilise souvent une pondération des variables

Types des variables

- Intervalles:
- Binaires:
- catégories, ordinale, ratio:
- Différents types:

Intervalle (discrètes)

□ Standardiser les données

- Calculer l'écart absolu moyen:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|)$$

$$m_f = \frac{1}{n}(x_{1f} + x_{2f} + \dots + x_{nf})$$

où

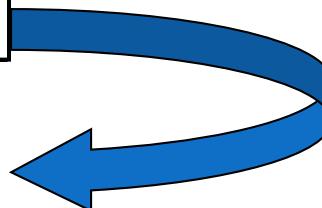
- Calculer la mesure standardisée (z-score)

Exemple

	Age	Salaire
Personne1	50	11000
Personne2	70	11100
Personne3	60	11122
Personne4	60	11074

$$M_{Age} = 60 \quad S_{Age} = 5$$

$$M_{salaire} = 11074 \quad S_{salaire} = 148$$



	Age	Salaire
Personne1	-2	-0,5
Personne2	2	0,175
Personne3	0	0,324
Personne4	0	2

Similarité entre objets

- Les distances expriment une similarité

- Ex: la *distance de Minkowski* :

$$d(i, j) = \sqrt[q]{(|x_{i_1} - x_{j_1}|^q + |x_{i_2} - x_{j_2}|^q + \dots + |x_{i_p} - x_{j_p}|^q)}$$

où $i = (x_{i_1}, x_{i_2}, \dots, x_{i_p})$ et $j = (x_{j_1}, x_{j_2}, \dots, x_{j_p})$ sont deux objets p -dimensionnels et q un entier positif

- Si $q = 1$, d est la distance de Manhattan : $d(i, j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \dots + |x_{i_p} - x_{j_p}|$

Similarité entre objets(I)

- Si $q = 2$, d est la distance Euclidienne :

$$d(i, j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \dots + |x_{i_p} - x_{j_p}|^2)}$$

- Propriétés

- $d(i, j) \geq 0$
- $d(i, i) = 0$
- $d(i, j) = d(j, i)$
- $d(i, j) \leq d(i, k) + d(k, j)$

Exemple: distance de Manhattan

	Age	Salaire
Personne1	50	11000
Personne2	70	11100
Personne3	60	11122
Personne4	60	11074

$$d(p1, p2) = 120$$

$$d(p1, p3) = 132$$

Conclusion: p1 ressemble plus à p2 qu'à p3 ☹

	Age	Salaire
Personne1	-2	-0,5
Personne2	2	0,175
Personne3	0	0,324
Personne4	0	0

$$d(p1, p2) = 4,675$$

$$d(p1, p3) = 2,324$$

Conclusion: p1 ressemble plus à p3 qu'à p2 ☺

Variables binaires

□ Une table de contingence pour données binaires

		Objet j		sum
		1	0	
$Objet i$	1	a	b	$a+b$
	0	c	d	$c+d$
sum		$a+c$	$b+d$	p

$a = \text{nombre de positions où } i$
 $\text{a } 1 \text{ et } j \text{ a } 1$

□ Exemple $o_i=(1,1,0,1,0)$ et $o_j=(1,0,0,0,1)$

$$a=1, b=2, c=1, d=2$$

Mesures de distances

- Coefficient d'appariement (matching) simple (invariant pour variables symétriques):

$$d(i, j) = \frac{b + c}{a + b + c + d}$$

Exemple $\mathbf{o}_i = (1, 1, 0, 1, 0)$ et $\mathbf{o}_j = (1, 0, 0, 0, 1)$

$$d(\mathbf{o}_i, \mathbf{o}_j) = 3/5$$

- Coefficient de Jaccard

$$d(\mathbf{o}_i, \mathbf{o}_j) = 3/4$$

$$d(i, j) = \frac{b + c}{a + b + c}$$

Variables binaires (I)

- Variable symétrique: Ex. le sexe d'une personne, i.e coder masculin par 1 et féminin par 0 c'est pareil que le codage inverse
- Variable asymétrique: Ex. Test HIV. Le test peut être positif ou négatif (0 ou 1) mais il y a une valeur qui sera plus présente que l'autre. Généralement, on code par 1 la modalité la moins fréquente
 - 2 personnes ayant la valeur 1 pour le test sont *plus similaires* que 2 personnes ayant 0 pour le test

Variables binaires(II)

□ Exemple

Nom	Sexe	Fièvre	Toux	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

- Sexe est un attribut symétrique
- $d(jack, mary) = \frac{0 + 1}{2 + 0 + 1} = 0.33$
- Les autres attributs sont asymétriques
- Y et P sont ~~pas~~ pas $\neq 0$, la distance n'est mesurée que sur les asymétriques

$$d(jim, mary) = \frac{1 + 2}{1 + 1 + 2} = 0.75$$

Les plus similaires sont Jack et Mary \Rightarrow atteints du même mal

Variables Nominales

- Une généralisation des variables binaires, ex: rouge, vert et bleu
- Méthode 1: Matching simple
 - m : # d'appariements, p : # total de variables
 - $$d(i, j) = \frac{P - m}{p}$$
- Méthode 2: utiliser un grand nombre de variables binaires
 - Créer une variable binaire pour chaque modalité (ex: variable rouge qui prend les valeurs vrai ou faux)

Variables Ordinales

■ Une variable ordinaire peut être discrète ou continue

- L'ordre peut être important, ex: classement
- Peuvent être traitées comme les variables intervalles
 - remplacer x_{if} par son rang $r_{if} \in \{1, \dots, M_f\}$
 - Remplacer le rang de chaque variable par une valeur dans $[0, 1]$ en remplaçant la variable f dans l'objet I par

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

- Utiliser une distance pour calculer la similarité

En Présence de Variables de différents Types

- Pour chaque type de variables utiliser une mesure adéquate.
Problèmes: les clusters obtenus peuvent être différents
- On utilise une formule pondérée pour faire la combinaison

$$d(i, j) = \frac{\sum_{f=1}^P \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^P \delta_{ij}^{(f)}}$$

- f est binaire ou nominale:

$$d_{ij}^{(f)} = 0 \text{ si } x_{if} = x_{jf}, \text{ sinon } d_{ij}^{(f)} = 1$$

- f est de type intervalle: utiliser une distance normalisée

- f est ordinaire

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

- calculer les rangs r_{if} et

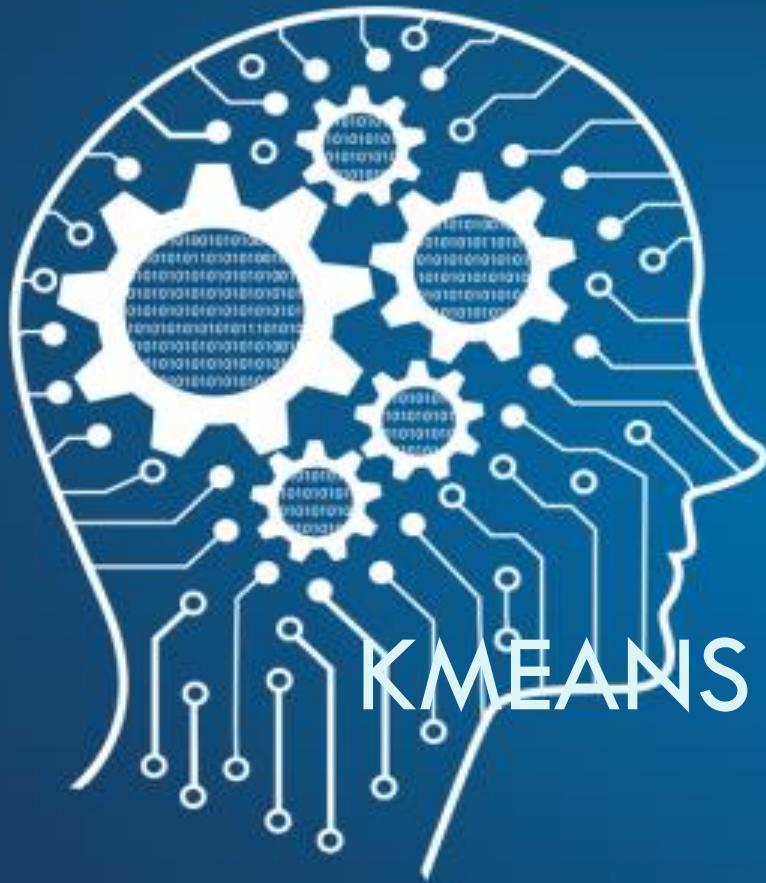
- Ensuite traiter z_{if} comme une variable de type intervalle

Approches de Clustering

- Algorithmes de Partitionnement: Construire plusieurs partitions puis les évaluer selon certains critères
- Algorithmes hiérarchiques: Créer une décomposition hiérarchique des objets selon certains critères
- Algorithmes basés sur la densité: basés sur des notions de connectivité et de densité
- Algorithmes de grille: basés sur un structure à multi-niveaux de granularité
- Algorithmes à modèles: Un modèle est supposé pour chaque cluster ensuite vérifier chaque modèle sur chaque groupe pour choisir le meilleur

Algorithmes à partitionnement

- Construire une partition à k clusters d'une base D de n objets
- Les k clusters doivent optimiser le critère choisi
 - ▣ Global optimal: Considérer toutes les k -partitions
 - ▣ Heuristic methods: Algorithmes k -means et k -medoids
 - ▣ k -means (MacQueen'67): Chaque cluster est représenté par son centre
 - ▣ k -medoids or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Chaque cluster est représenté par un de ses objets



KMEANS



La méthode des k-moyennes (K-Means)

□ L'algorithme *k-means* est en 4 étapes :

1. Choisir k objets formant ainsi k clusters
2. (Ré)affecter chaque objet O au cluster C_i de centre M_i tel que $\text{dist}(O, M_i)$ est minimal
3. Recalculer M_i de chaque cluster (le barycentre)
4. Aller à l'étape 2 si on vient de faire une affectation

K-Means :Exemple

- $A = \{1, 2, 3, 6, 7, 8, 13, 15, 17\}$. Créer 3 clusters à partir de A
- On prend 3 objets au hasard. Supposons que c'est 1, 2 et 3. Ca donne $C_1 = \{1\}$, $M_1 = 1$, $C_2 = \{2\}$, $M_2 = 2$, $C_3 = \{3\}$ et $M_3 = 3$
- Chaque objet O est affecté au cluster au milieu duquel, O est le plus proche. 6 est affecté à C_3 car $\text{dist}(M_3, 6) < \text{dist}(M_2, 6)$ et $\text{dist}(M_3, 6) < \text{dist}(M_1, 6)$
On a $C_1 = \{1\}$, $M_1 = 1$,
 $C_2 = \{2\}$, $M_2 = 2$
 $C_3 = \{3, 6, 7, 8, 13, 15, 17\}$, $M_3 = 69/7 = 9.86$

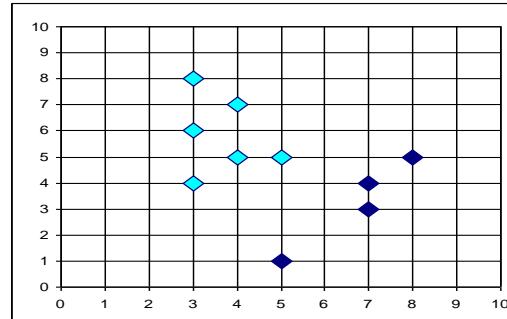
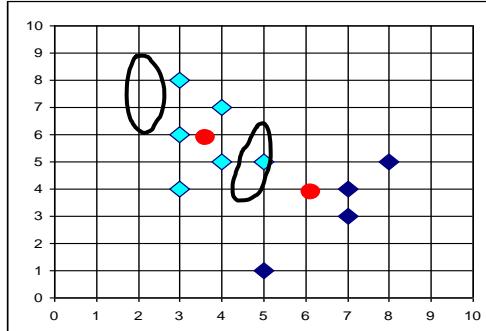
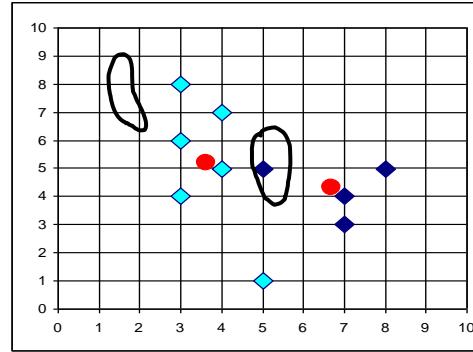
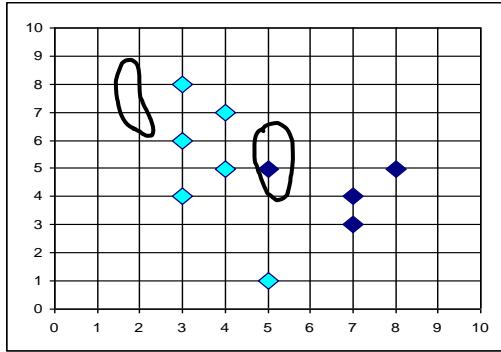
K-Means: Exemple (suite)

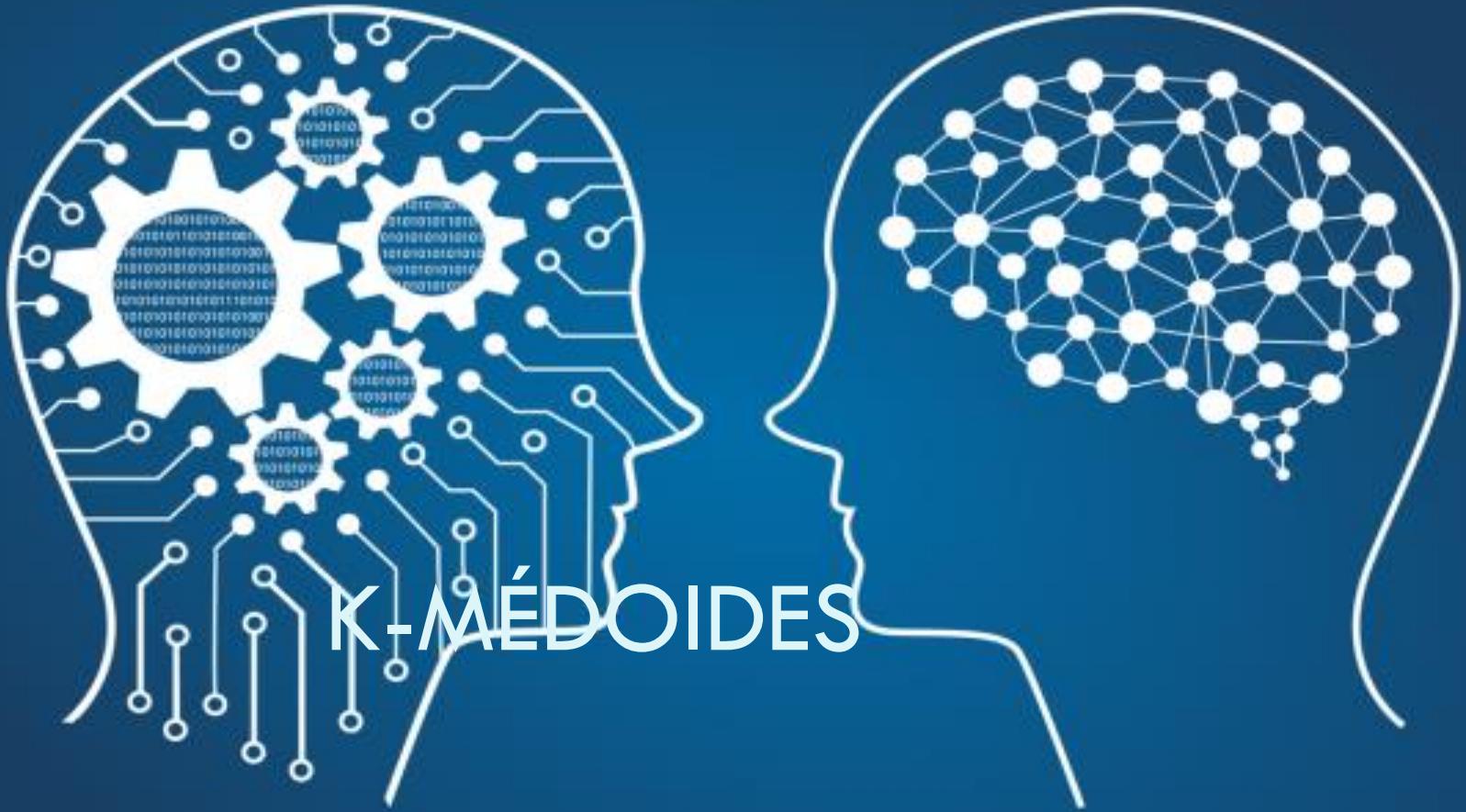
- $\text{dist}(3, M_2) < \text{dist}(3, M_3) \rightarrow 3$ passe dans C_2 . Tous les autres objets ne bougent pas. $C_1 = \{1\}$, $M_1 = 1$, $C_2 = \{2, 3\}$, $M_2 = 2.5$, $C_3 = \{6, 7, 8, 13, 15, 17\}$ et $M_3 = 66/6 = 11$
- $\text{dist}(6, M_2) < \text{dist}(6, M_3) \rightarrow 6$ passe dans C_2 . Tous les autres objets ne bougent pas. $C_1 = \{1\}$, $M_1 = 1$, $C_2 = \{2, 3, 6\}$, $M_2 = 11/3 = 3.67$, $C_3 = \{7, 8, 13, 15, 17\}$, $M_3 = 12$
- $\text{dist}(2, M_1) < \text{dist}(2, M_2) \rightarrow 2$ passe en C_1 . $\text{dist}(7, M_2) < \text{dist}(7, M_3) \rightarrow 7$ passe en C_2 . Les autres ne bougent pas. $C_1 = \{1, 2\}$, $M_1 = 1.5$, $C_2 = \{3, 6, 7\}$, $M_2 = 5.34$, $C_3 = \{8, 13, 15, 17\}$, $M_3 = 13.25$
- $\text{dist}(3, M_1) < \text{dist}(3, M_2) \rightarrow 3$ passe en 1. $\text{dist}(8, M_2) < \text{dist}(8, M_3) \rightarrow 8$ passe en 2
 $C_1 = \{1, 2, 3\}$, $M_1 = 2$, $C_2 = \{6, 7, 8\}$, $M_2 = 7$, $C_3 = \{13, 15, 17\}$, $M_3 = 15$

Plus rien ne bouge

Algorithme K-Means

Exemple





K-MÉDOIDES

La méthode des K-Medoids (PAM)

- Trouver des objets représentatifs (medoïdes) dans les clusters (au lieu de la moyenne)
- Principe
 - Commencer avec un ensemble de medoïdes puis itérativement remplacer un par un autre si ça permet de réduire la distance globale
 - Efficace pour des données de petite taille

Algorithme des k-Medoides

Choisir arbitrairement k medoides

Répéter

affecter chaque objet restant au medoide le plus proche

Choisir aléatoirement un non-medoide O_r

Pour chaque medoide O_i

 Calculer le coût TC du remplacement de O_i par O_r

 Si $TC < 0$ alors

 Remplacer O_i par O_r

 Calculer les nouveaux clusters

 Finsi

FinPour

Jusqu'à ce ce qu'il n'y ait plus de changement

PAM (Partitioning Around Medoids) (1987)

Choisir arbitrairement k objets représentatifs

- Pour toute paire (h,i) d'objets t.q h est choisi et j non,
calculer le coût TC_{ih} du remplacement de j par h
 - Si $TC_{ih} < 0$, j est remplacé par h
 - Puis affecter chaque objet non sélectionné au
medoïde qui lui est le plus similaire
- Répéter jusqu'à ne plus avoir de changements

La méthode des K-Medoids

TC_{jh} représente le gain en distance globale que l'on va avoir en remplaçant h par j

- Si TC_{jh} est négatif alors on va perdre en distance. Ca veut dire que les clusters seront plus compacts.
- $TC_{jh} = \sum_i dist(j,h) - dist(j,i) = \sum_i C_{ijh}$

La méthode des K-Medoids: Exemple

Soit $A = \{1, 3, 4, 5, 8, 9\}$, $k=2$ et $M = \{1, 8\}$ ensemble des medoides

$\rightarrow C_1 = \{1, 3, 4\}$ et $C_2 = \{5, 8, 9\}$

$$E_{\{1,8\}} = \text{dist}(3,1)^2 + \text{dist}(4,1)^2 + \text{dist}(5,8)^2 + \text{dist}(5,9)^2 + \text{dist}(9,8)^2 = 39$$

Comparons 1 et 3 $\rightarrow M = \{3, 8\} \rightarrow C_1 = \{1, 3, 4, 5\}$ et $C_2 = \{8, 9\}$

$$E_{\{3,8\}} = \text{dist}(1,3)^2 + \text{dist}(4,3)^2 + \text{dist}(5,3)^2 + \text{dist}(9,8)^2 = 10$$

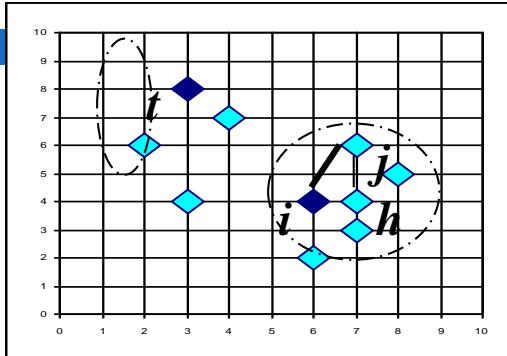
$E_{\{3,8\}} - E_{\{1,8\}} = -29 < 0$ donc le remplacement est fait.

Comparons 3 et 4 $\rightarrow M = \{4, 8\} \rightarrow C_1$ et C_2 inchangés et

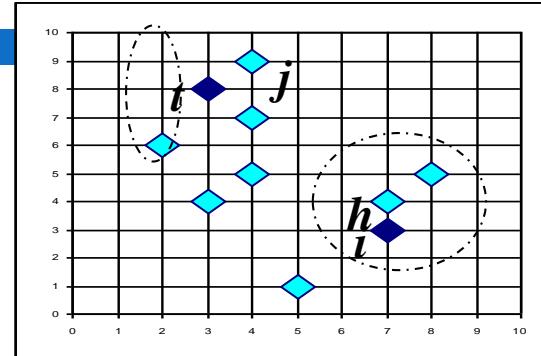
$$E_{\{4,8\}} = \text{dist}(1,4)^2 + \text{dist}(3,4)^2 + \text{dist}(5,4)^2 + \text{dist}(8,9)^2 = 12 \rightarrow 3$$
 n'est pas remplacé par 4

Comparons 3 et 5 $\rightarrow M = \{5, 8\} \rightarrow C_1$ et C_2 inchangés et $E_{\{5,8\}} > E_{\{3,8\}}$

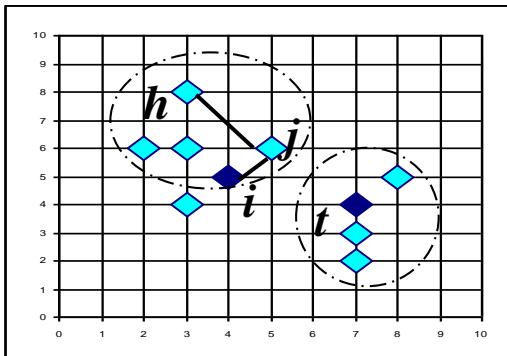
PAM Clustering: $TC_{ih} = \sum_i C_{jih}$



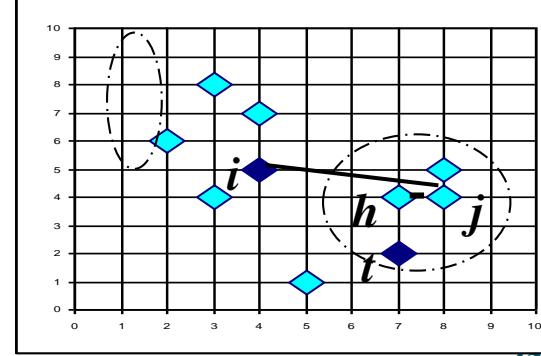
$$C_{jih} = d(j, h) - d(j, i)$$



$$C_{jih} = 0$$



$$C_{jih} = d(j, t) - d(j, i)$$



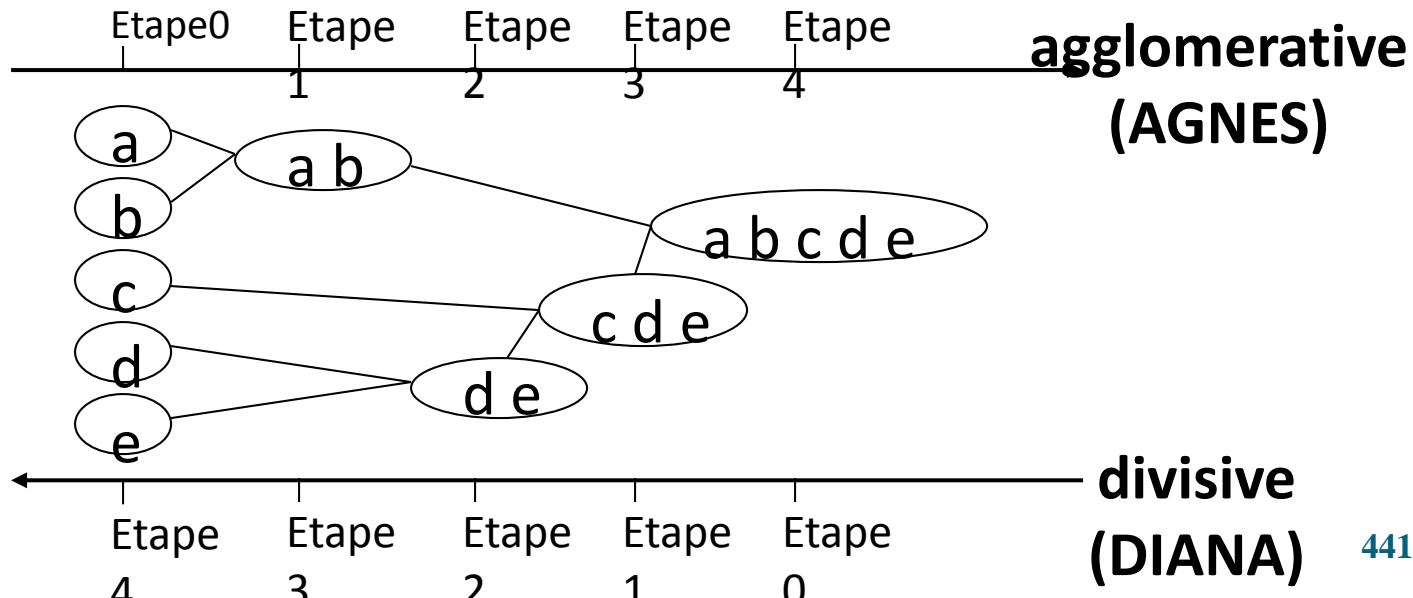
$$C_{jih} = d(j, h) - d(j, t)$$
⁴³⁹



CLASSIFICATION HIÉRARCHIQUE

Clustering Hiérarchique

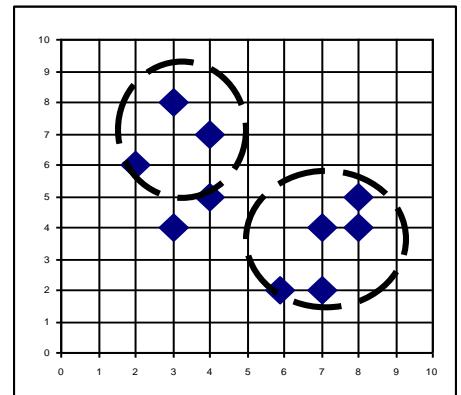
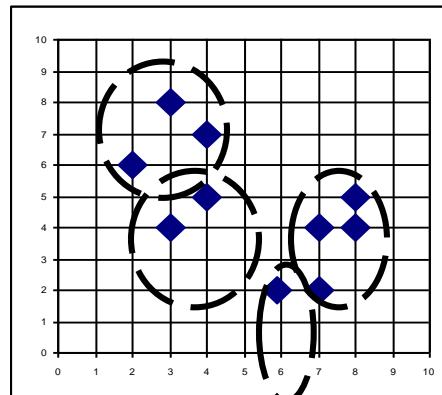
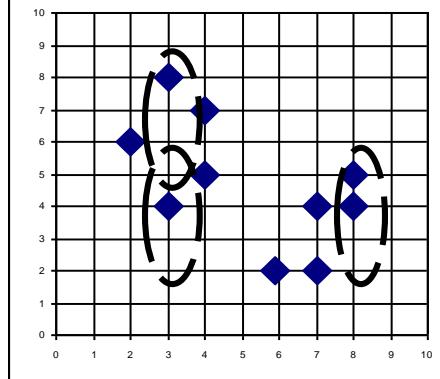
- Utiliser la matrice de distances comme critère de regroupement.
 k n'a pas à être précisé, mais a besoin d'une condition d'arrêt



AGNES (Agglomerative Nesting)

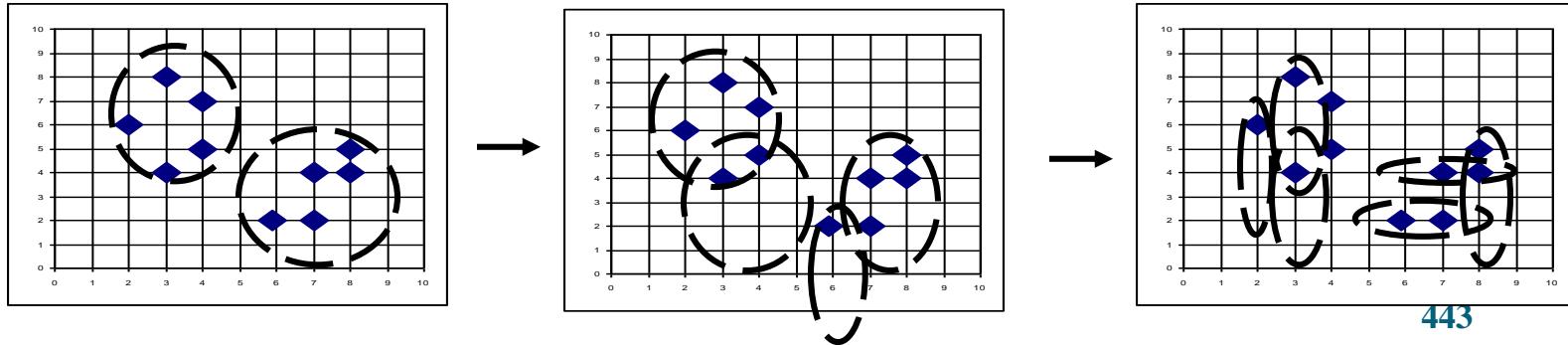
- Utilise la matrice de dissimilarité.
- Fusionne les nœuds qui ont la plus faible dissimilarité
- On peut se retrouver dans la situation où tous les nœuds sont

↑
coupe



DIANA (Divisive Analysis)

- L'ordre inverse de celui d'AGNES
- Il se peut que chaque objet forme à lui seul un groupe



Critères de fusion-éclatement

- Exemple: pour les méthodes agglomératives, C1 et C2
- sont fusionnés si
 - il existe $o1 \in C1$ et $o2 \in C2$ tels que $\text{dist}(o1, o2) \leq \text{seuil}$, ou
 - il n'existe pas $o1 \in C1$ et $o2 \in C2$ tels que $\text{dist}(o1, o2) \geq \text{seuil}$, ou
 - distance entre $C1$ et $C2 \leq \text{seuil}$ avec unique

Lien →

$$\text{dist} (C_1, C_2) = \frac{1}{n_1 * n_2} \sum_{o1 \in C1, o2 \in C2} \text{dist}(o1, o2)$$

et $n_1 = |C1|$.

- Ces techniques peuvent être adaptées pour les méthodes divisives

BIRCH (1996)

Birch: Balanced Iterative Reducing and Clustering using Hierarchies

- Construit incrémentalement un arbre (CF-tree : Clustering Feature), une structure hiérarchique où chaque niveau représente une phase de clustering
 - Phase 1: scanner la base pour construire le CF-tree dans la mémoire
 - Phase 2: utiliser n'importe quel algorithme de clustering sur les feuilles du CF-tree
- Avantage: trouve les clusters en une seule passe sur la BD
- Inconvénient: ne considère que les données numériques et est sensible à l'ordre des enregistrements

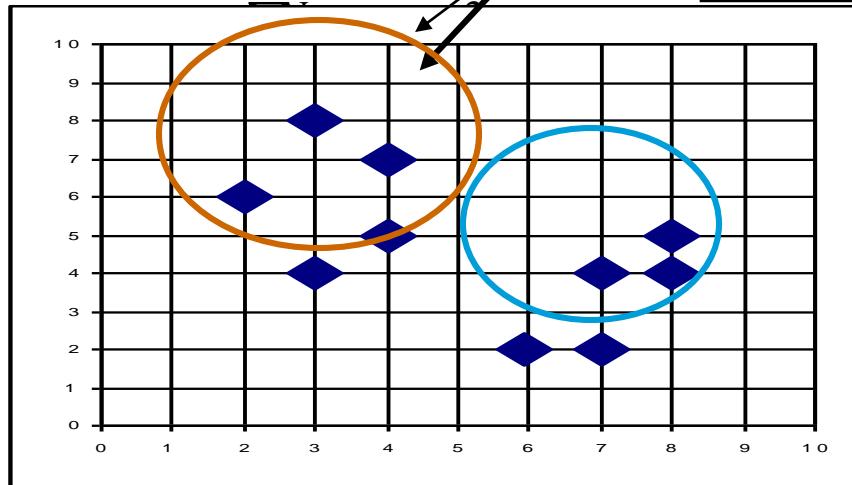
Clustering Feature Vector

Clustering Feature: $CF = (\overrightarrow{N}, LS, SS)$

N: Number of data points

$$LS: \sum_{i=1}^N = \vec{X_i}$$

$$CF = (5, (16,30), (54,190))$$



(3,4)

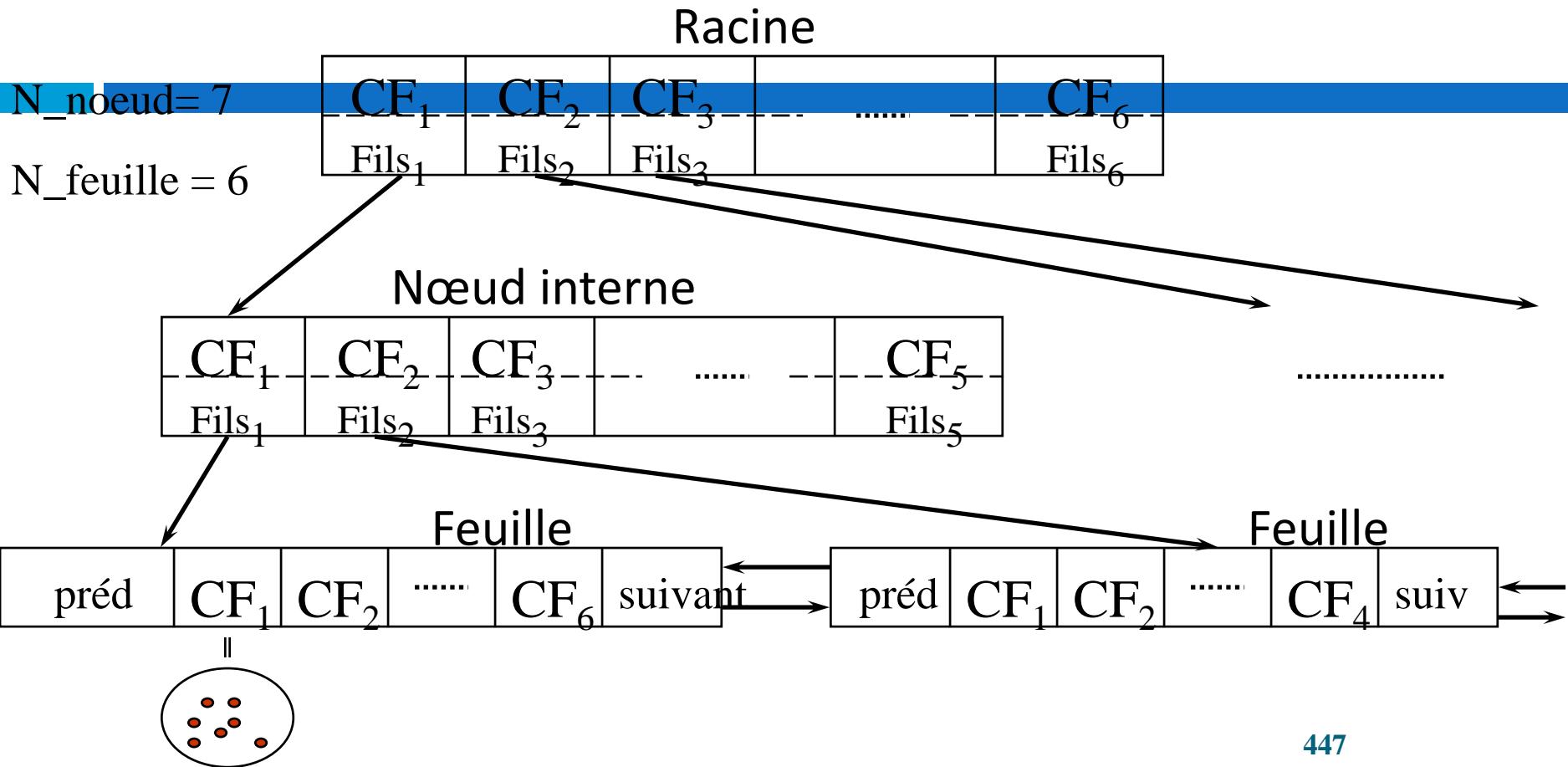
(2,6)

(4,5)

(4,7)

(3,8) 446

CF Tree



CURE (Clustering Using REpresentatives)



- Les méthodes précédentes donnent les groupes (b)
- CURE: (1998)
 - Arrête la création de clusters dès qu'on en a k
 - Utilise plusieurs points représentatifs clusters

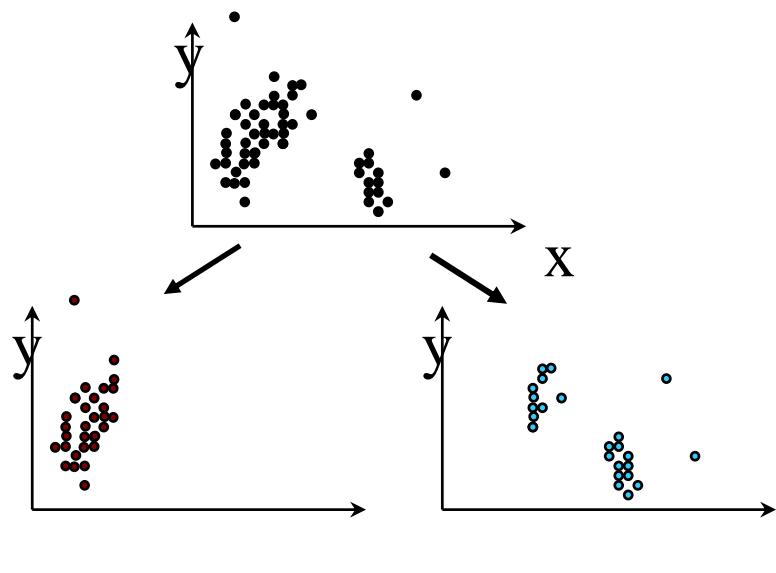
Cure: l'algorithme

- Prendre un sous-ensemble s
- Partitionner s en p partitions de taille s/p
- Dans chaque partition, créer s/pq clusters
- Eliminer les exceptions (points aberrants)
- Regrouper les clusters partiels

Partitionnement et Clustering

$s = 50$

- $p = 2$
- $s/p = 25$



■ $s/pq = 5$

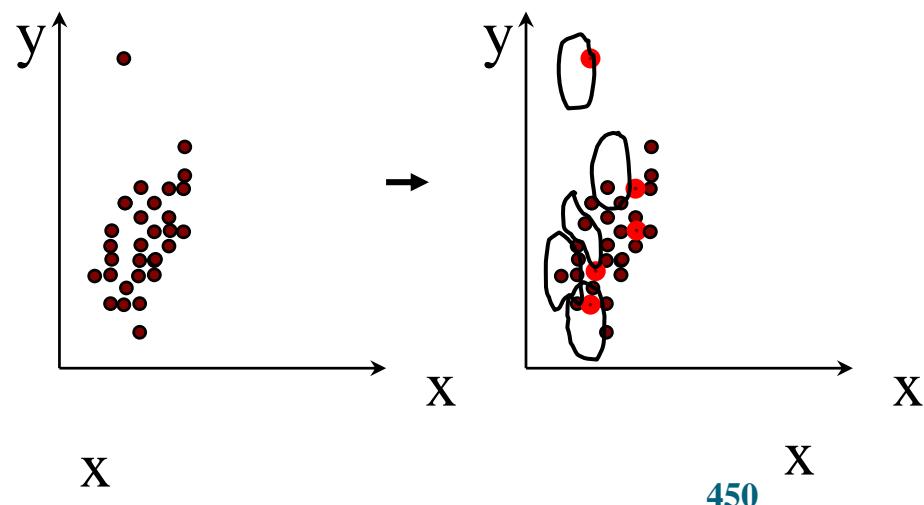
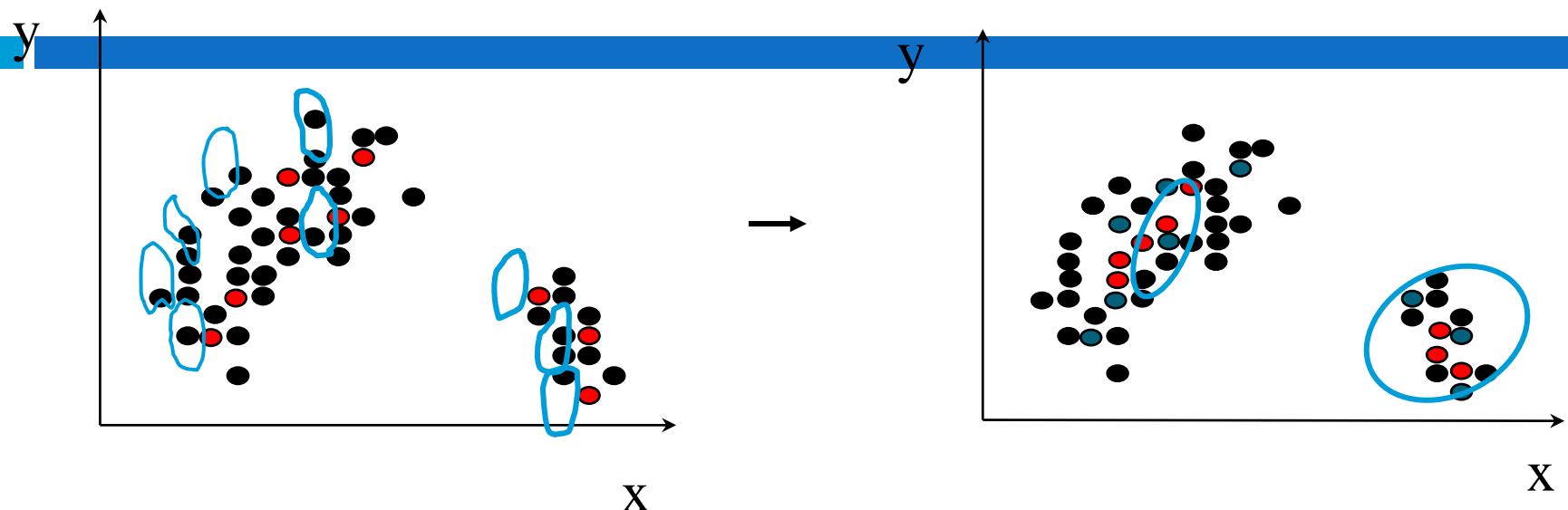


Figure: Rapprochement des points représentatifs



- Rapprocher les points représentatifs vers le centre de gravité par un facteur α .
- Plusieurs points représentatifs permettent de figurer la forme du cluster

Clustering de données Catégorielles : ROCK

- ROCK: Robust Clustering using links
 - Utilise les liens pour mesurer la similarité/proximité
 - N'est pas basé sur la notion de distance
- Idée :
 - Fonction de similarité et voisins:

Let $T_1 = \{1, 2, 3\}$, $T_2 = \{3, 4, 5\}$

$$Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|} = \frac{1}{5}$$

$$Sim(T_1, T_2) = \frac{|\{3\}|}{|\{1, 2, 3, 4, 5\}|} = \frac{1}{5} = 0.2$$

Rock

- Considérons 4 transactions et 6 produits t.q

$$T1=\{1,2,3,5\} \quad T2=\{2,3,4,5\}$$

$$T3=\{1,4\} \text{ et } T4=\{6\}$$

- $T1$ peut être représentée par $\{1,1,1,0,1,0\}$

$\text{dist}(T1, T2)=2$ qui est la plus petite distance entre 2 transactions $\rightarrow T1$ et $T2$ dans même cluster. La moyenne de $C1=(0.5,1,1,0.5,1,0)$.

$C2=\{T3, T4\}$ car $\text{dist}(T3, T4)=3$. Or $T3$ et $T4$ n'ont aucun produit en commun !

Idée : se baser sur le nombre d'éléments en commun

Ce n'est pas suffisant $\{1,2\}$ est plus proche de $\{1,2,3\}$ que de $\{1,2,3,4,5,6\}$

Rock: l'algorithme

- Liens: Le nombre de voisins communs de 2 points

$\{1,2,3\}, \{1,2,4\}, \{1,2,5\}, \{1,3,4\}, \{1,3,5\}$
 $\{1,4,5\}, \{2,3,4\}, \{2,3,5\}, \{2,4,5\}, \{3,4,5\}$

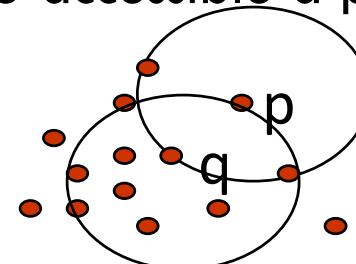
$\{1,2,3\} \xleftarrow[3]{\quad} \{1,2,4\}$

- Algorithme

- Prendre un sous ensemble
 - Regrouper avec les liens

Clustering basé sur la densité

- Voit les clusters comme des régions denses séparées par des régions qui le sont moins (bruit)
- Deux paramètres:
 - **Eps:** Rayon maximum du voisinage
 - **MinPts:** Nombre minimum de points dans le voisinage-Eps d'un point
- **Voisinage :** $V_{Eps}(p) : \{q \in D \mid dist(p,q) \leq Eps\}$
- Un point p est directement densité-accessible à partir de q resp. à Eps , $MinPts$ si
 - 1) $p \in V_{Eps}(q)$
 - 2) $|V_{Eps}(q)| \geq MinPts$



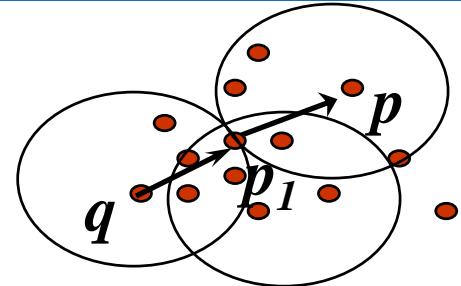
MinPts = 5

Eps = 1 cm

Clustering basé sur la densité

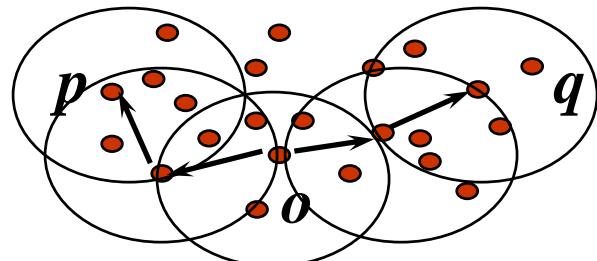
□ Accessibilité:

- p est accessible à partir de q resp. à $Eps, MinPts$ si il existe p_1, \dots, p_n , $p_1 = q$, $p_n = p$ t.q p_{i+1} est directement densité accessible à partir de p_i



□ Connexité

- p est connecté à q resp. à $Eps, MinPts$ si il existe un point o t.q p et q accessibles à partir de o resp. à Eps et $MinPts$.



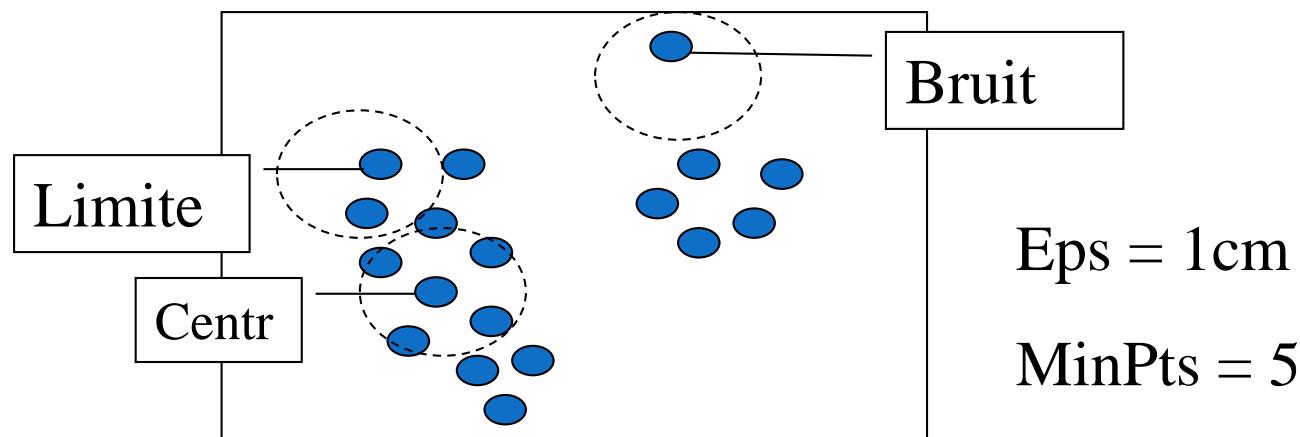


DBSCAN



DBSCAN: Density Based Spatial Clustering of Applications with Noise

- Un *cluster* est l'ensemble maximal de points connectés
- Découvre des clusters non nécessairement convexes



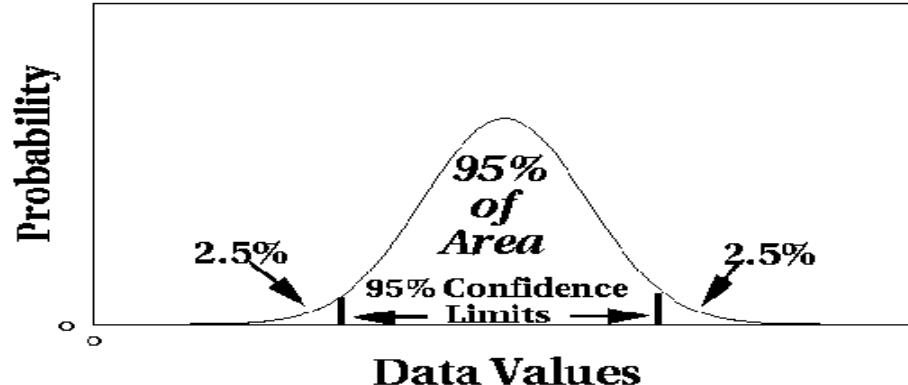
DBSCAN: l'algorithme

- Choisir p
- Récupérer tous les points accessibles à partir de p resp. Eps et $MinPts$.
- Si p est un centre, un cluster est formé.
- si p est une limite, alors il n'y a pas de points accessibles de p : passer à un autre point
- Répéter le processus jusqu'à épuiser tous les points.

Découverte d'exceptions

- Ce sont les objets qui sont considérablement différents du reste, exemple: ornithorynque, kiwi
- Problème
 - Trouver n objets qui sont les plus éloignés du reste
- Applications:
 - fraude
 - Analyse médicale
 - ...

Approche statistique



- On suppose que les données suivent une loi de distribution statistique (ex: loi normale)
- Utiliser les tests de discordance
 - $\text{Proba}(X_i=\text{val}) < \beta$ alors X est une exception
- Problèmes
 - La plupart des tests sont sur un attribut
 - Dans beaucoup de cas, la loi de distribution est inconnue

Approche Basée sur la Distance

- Une (α, β) -exception est un object O dans T tel qu'il y a au moins α objets O' de T avec $\text{dist}(O, O') > \beta$



Thank you

Pr. BENLAHMAR EL Habib