:: Decision Tree

Disadvantages of Decision Tree

- **They are prone to over-fitting.**

  Decision-tree learners can create over-complex trees that do not generalize the data well[1].

- **Decision Tree learner can create biased trees in case of unbalanced data**

  Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree. [2]

- **Instability**

  Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble[3]. Random Forests can limit this instability by averaging predictions over many trees.

- **Greedy approach used by Decision tree doesn't guarantee best solution**

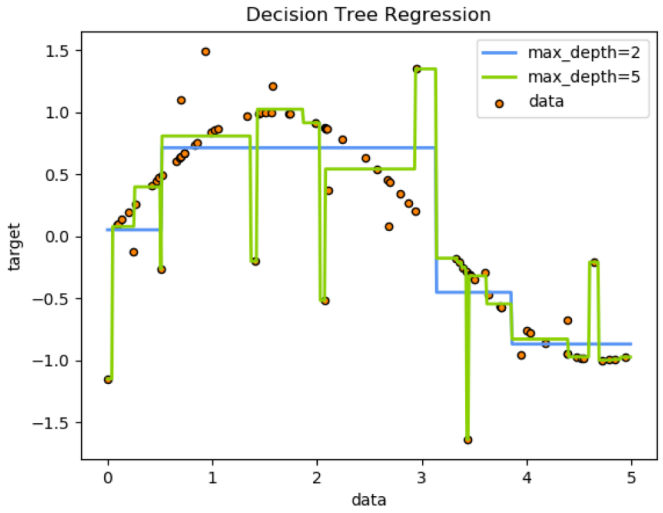
  Greedy algorithm where locally optimal decisions are made at each node cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement[4].

1, 2, 3 & 4 Source : http://scikit-learn.org/stable/modules/tree.html
Image: Designed by Asierromero / Freepik

:: Decision Tree

Disadvantages of Decision Tree - continued

- **Standard decision trees are restricted by hard, axis-aligned splits of the input space** [1]
  The splits are aligned with the axes of the feature space and this lead to rectilinear decision boundaries.



The deeper the tree, the more complex the decision rules and the fitter the model.

We can see that if the maximum depth of the tree is set too high, the decision trees learn too fine details of the training data and learn from the noise, i.e. they overfit.[3]

Another issue is particularly problematic in regression where we are typically aiming to model smooth functions, and yet the tree model produces piecewise-constant predictions with discontinuities at the split boundaries.[2]

1,2 Source: Christopher M. BishopPattern Recognition and Machine Learning
3 source: http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html
Image: Designed by Asierromero / Freepik

Naïve Bayes

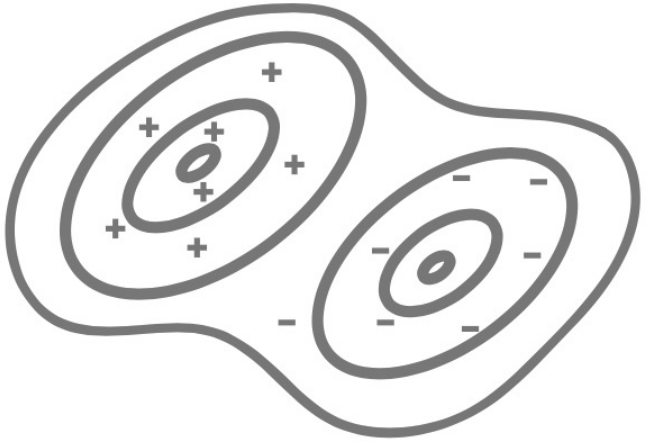
:: Naïve Bayes

Naïve Bayes is a simple but important probabilistic model

Naive Bayes is a simple multiclass classification algorithm based on applying Bayes' theorem with the **"naive" assumption** of independence between the features. It assumes that the conditional probabilities of the independent variables are statistically independent.

It computes the conditional probability distribution of each feature given label, and then it applies Bayes' theorem to compute the conditional probability distribution of label given an observation and use it for prediction[1].

It classifies new data based on the **highest probability** of its belonging to a particular class.



Naive Bayes is a simple classifier model that is [3]:

- Based on the Bayes theorem
- Supervised Learning
- Easy to build
- Faster to train, compared to other models
- Often used as a baseline classifier for benchmarking

1 Source: http://spark.apache.org/docs/2.2.0/mllib-naive-bayes.html
3 Source: https://www.slideshare.net/ananth/an-overview-of-nave-bayes-classifier

:: Naïve Bayes

Naïve Assumption

It assume that each input feature x_i is conditionally independent of every other feature x_j, given the class C.

$$P(X1, X2, ..., Xn \mid C) = P(X1 \mid C) \cdot P(X2 \mid C) \cdot ... \cdot P(Xn \mid C)$$

For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter.

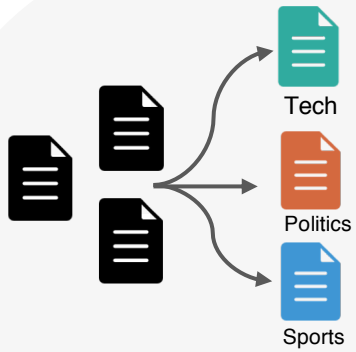
A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features[2].

2 Source: Predictive Analysis on Large Data for Actionable Knowledge: Emerging Research and Opportunities

:: Naïve Bayes

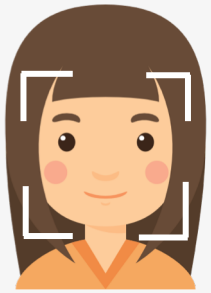
Application of Naïve **Bayes**

Although the assumption that the predictor (independent) variables are independent is not always accurate, it does simplify the classification task dramatically[1]. Naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters [2]. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods [3].



Document Classification

Spam Detection



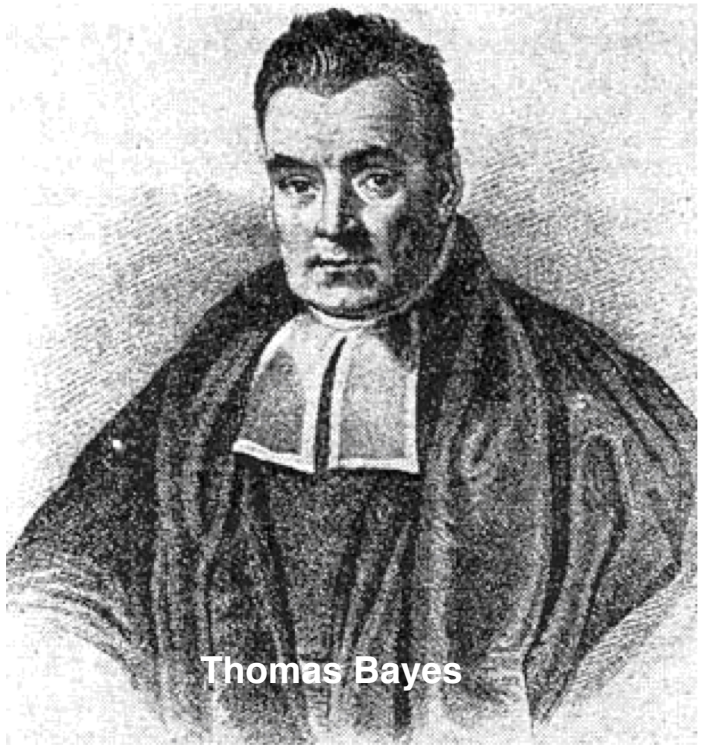
Face recognition

: http://scikit-learn.org/stable/modules/naive_bayes.html
Icon made by Freepik] from www.flaticon.com
1 Source: http://documentation.statsoft.com/STATISTICAHelp.aspx?path=MachineLearning/MachineLearning/Overviews/NaiveBayesClassifierIntroductoryOverview

:: Naïve Bayes

About Bayes Theorem (Bayes‘ rule)

Bayes theorem named after Rev. Thomas Bayes. It works on **conditional probability**. Conditional probability is the probability that something will happen, given that something else has already occurred. Using the conditional probability, we can calculate the probability of an event using its prior knowledge[1].


Thomas Bayes

It tells us how often A happens given that B happens

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

- **P (A I B)** is “Probability of A given B”, the probability of A given that B happens.
- **P (A)** is Probability of A
- **P (B I A)** is “Probability of B given A”, the probability of B given that A happen
- **P (B)** is Probability of B

For example:

- P(Fire I Smoke) means how often there is fire when we see smoke.
- P(Smoke I Fire) means how often we see smoke when there is fire.

Source: http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/

:: Naïve Bayes

Naïve Bayes classifier

Classification is to derive the maximum posteriori, i.e., the **maximal P (C_k| X)**

- Given training data X = (X_1, X_2, X_3, ... X_n), representing some *n* features (independent variables)
  - for example, there are 3 features : X = (Pants, long hair, red T-Shirt)
- Suppose C represents the class variable.
  - for example, there are 2 classes. C_1= Man , C_2= Woman
- Using Bayes' theorem, the conditional probability can be decomposed as

$$\underbrace{P(C_k \mid X_1,...,X_n)}_{\text{Posterior Probability}} = \frac{P(X_1,...,X_n \mid C_k)\, P(C_k)}{P(X_1,...,X_n)} = \frac{P(X_1 \mid C_k)\, P(X_2 \mid C_k) \ldots P(X_n \mid C_k)\, P(C_k)}{P(X_1,...,X_n)}$$

For example, there is a new observation :
a person who wears pants and red T-shirt and has long hair

- P (Man | Pants, long hair, red T-Shirt) = 0.7
- P (Woman | Pants, long hair, red T-Shirt) = 0.3

Classify *X* to the class which has bigger posterior P

- Predicts X belongs to C_i if $P(C_i \mid X_1,...,X_n)$ is the highest among all the $P(C_k \mid X)$ for all the *k* classes

  In this example, this person is predicted as Man because its posteriori probability 0.7 is bigger than 0.3

:: Naïve Bayes

Example : Can we play tennis today ?

Lets say we have a table that decided if we should play tennis under certain circumstances. These could be the **outlook** of the weather; the **temperature**; the **humidity** and the strength of the **wind:**

Day	Outlook	Temperature	Humidity	Wind	Play Tennis ?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

there are 5 cases of not being able to play a game, and 9 cases of being able to play a game.

If we were given a new instance :

X = (**Outlook** = Sunny, **Temperature** = Cool, **Humidity** = High, **Wind** = Strong)

We want to know if we can play a game or not ?

Source: https://computersciencesource.wordpress.com/2010/01/28/year-2-machine-learning-naive-bayes-classifier/

:: Naïve Bayes

Example : Can we play tennis today ?

Here we have 4 attributes. What we need to do is to create “look-up tables” for each of these attributes, and write in the probability that a game of tennis will be played based on this attribute.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis ?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Outlook	Play = Yes	Play = No	Total
Sunny	2/9	3/5	5/14
Overcast	4/9	0/5	4/14
Rain	3/9	2/5	5/14

Temperature	Play = Yes	Play = No	Total
Hot	2/9	2/5	4/14
Mild	4/9	2/5	6/14
Cool	3/9	1/5	4/14

Humidity	Play = Yes	Play = No	Total
High	3/9	4/5	7/14
Normal	6/9	1/5	7/14

Wind	Play = Yes	Play = No	Total
Strong	3/9	3/5	6/14
Weak	6/9	2/5	8/14

If we were given a new instance :

X = (**Outlook** = Sunny, **Temperature** = Cool, **Humidity** = High, **Wind** = Strong), can we play the game ?

Firstly we look at the probability that we can play the game

- P (Outlook=Sunny | Play=Yes) = 2/9
- P (Temperature=Cool | Play=Yes) = 3/9
- P (Humidity=High | Play=Yes) = 3/9
- P (Wind=Strong | Play=Yes) = 3/9
- P (Play=Yes) = 9/14

Next we consider the fact that we cannot play a game:

- P (Outlook=Sunny | Play=No) = 3/5
- P (Temperature=Cool | Play=No) = 1/5
- P (Humidity=High | Play=No) = 4/5
- P (Wind=Strong | Play=No) = 3/5
- P (Play=No) = 5/14

Source: https://computersciencesource.wordpress.com/2010/01/28/year-2-machine-learning-naive-bayes-classifier/

:: Naïve Bayes

Example : Can we play tennis today ?

If X = (Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong)， then

P (Play=Yes I X) = P (Play=Yes | Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong)

$$= \frac{P(\text{Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong I Play=Yes}) * P(\text{Play=Yes})}{P(\text{Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong})}$$

$$= \frac{P(\text{Outlook = Sunny I Play=Yes}) * P(\text{Temperature = Cool I Play=Yes}) * P(\text{Humidity = High I Play=Yes}) * P(\text{Wind = Strong I Play=Yes}) * P(\text{Play=Yes})}{P(\text{Outlook=Sunny}) * P(\text{Temperature=Cool}) * P(\text{Humidity=High}) * P(\text{Wind=Strong})}$$

$$= \frac{(2/9) * (3/9) * (3/9) * (3/9) * (9/14)}{(5/14) * (4/14) * (7/14) * (6/14)}$$

$$= \frac{0.0053}{0.02186} = \mathbf{0.2424}$$

P (Play= No I X) = P (Play= NO | Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong)

$$= \frac{(3/5) * (1/5) * (4/5) * (3/5) * (5/14)}{(5/14) * (4/14) * (7/14) * (6/14)} = \frac{0.0206}{0.02186} = \mathbf{0.9421}$$

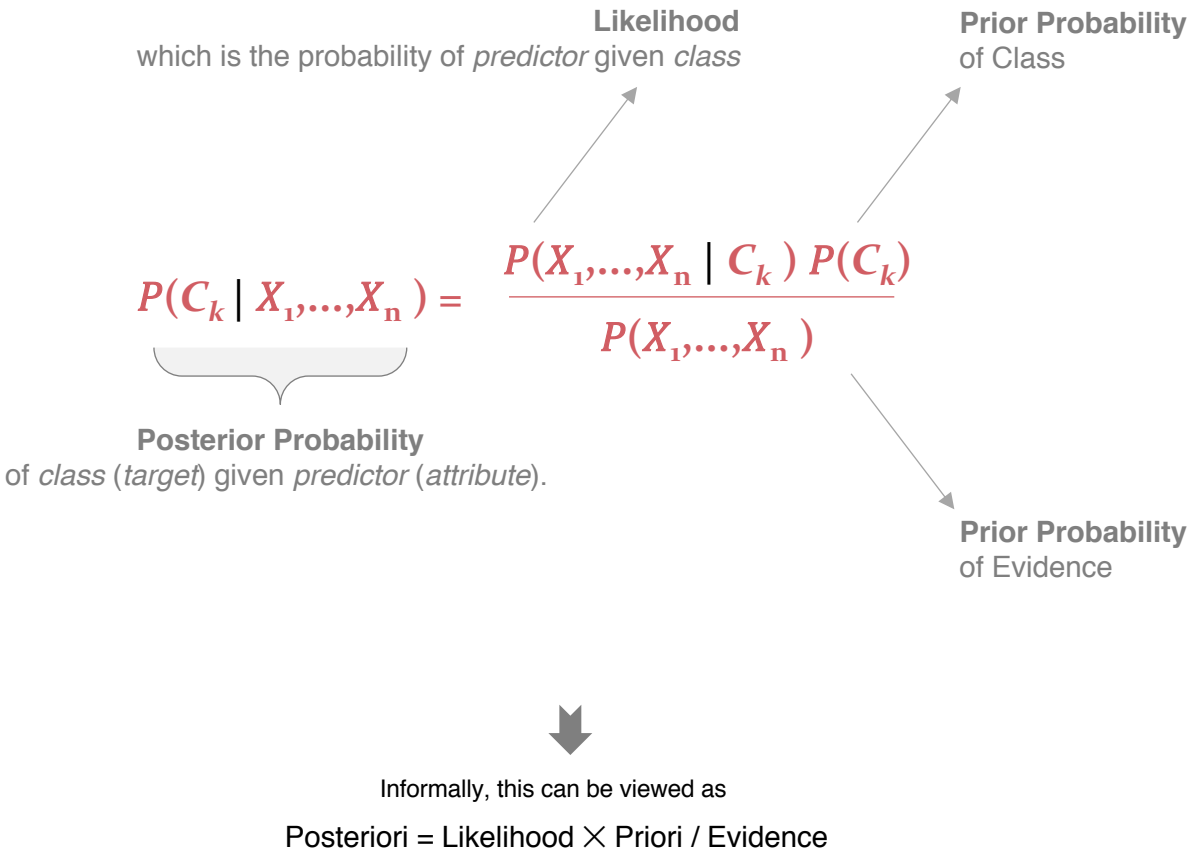
- P(Play=Yes I X) = 0.2424
- P(Play=No I X) = 0.9421

Since 0.9421 is greater than 0.2424 then the answer is ‘no’, we cannot play a game of tennis today.

Source: https://computersciencesource.wordpress.com/2010/01/28/year-2-machine-learning-naive-bayes-classifier/

:: Naïve Bayes

More details about Naive Bayes classifier



Estimating a-posteriori probabilities

In the Bayesian's interpretation for probability, probability measures a "degree of belief." To evaluate the probability of a hypothesis, the Bayesian probabilistic specifies some prior probability, which is then updated to a posterior probability in the light of new, relevant data (evidence)[1].

- Before observing the data, our prior beliefs can be expressed in a prior probability distribution that represents the knowledge we have about the unknown features[2].
- After observing the data our revised beliefs are captured by a posterior distribution over the unknown features[3].

1 Source: https://en.wikipedia.org/wiki/Bayesian_probability
2&3 Source: Gladys Castillo, Naive Bayes

:: Naïve Bayes

More details about Naive Bayes classifier

$$P(C_k \mid X_1,...,X_n) = \frac{P(X_1,...,X_n \mid C_k)\, P(C_k)}{P(X_1,...,X_n)}$$

P(x) can be ignored

- $P(X_1,...,X_n)$ can be ignored because it is the same for all the classes (a normalization constant).

Naïve Assumption: attribute independence

- It's difficult to learn the probability $P(X_1,...,X_n \mid C_k)$. It required initial knowledge of many probabilities, involving significant computational cost
- However, if we assume that the feature probabilities are conditionally independent (i.e., no dependence relationship between all input features), it can be simplified as follow.

$$P(X_1, X_2,..., X_n \mid C) = P(X_1 \mid C) \cdot P(X_2 \mid C) \cdot ... \cdot P(X_n \mid C)$$

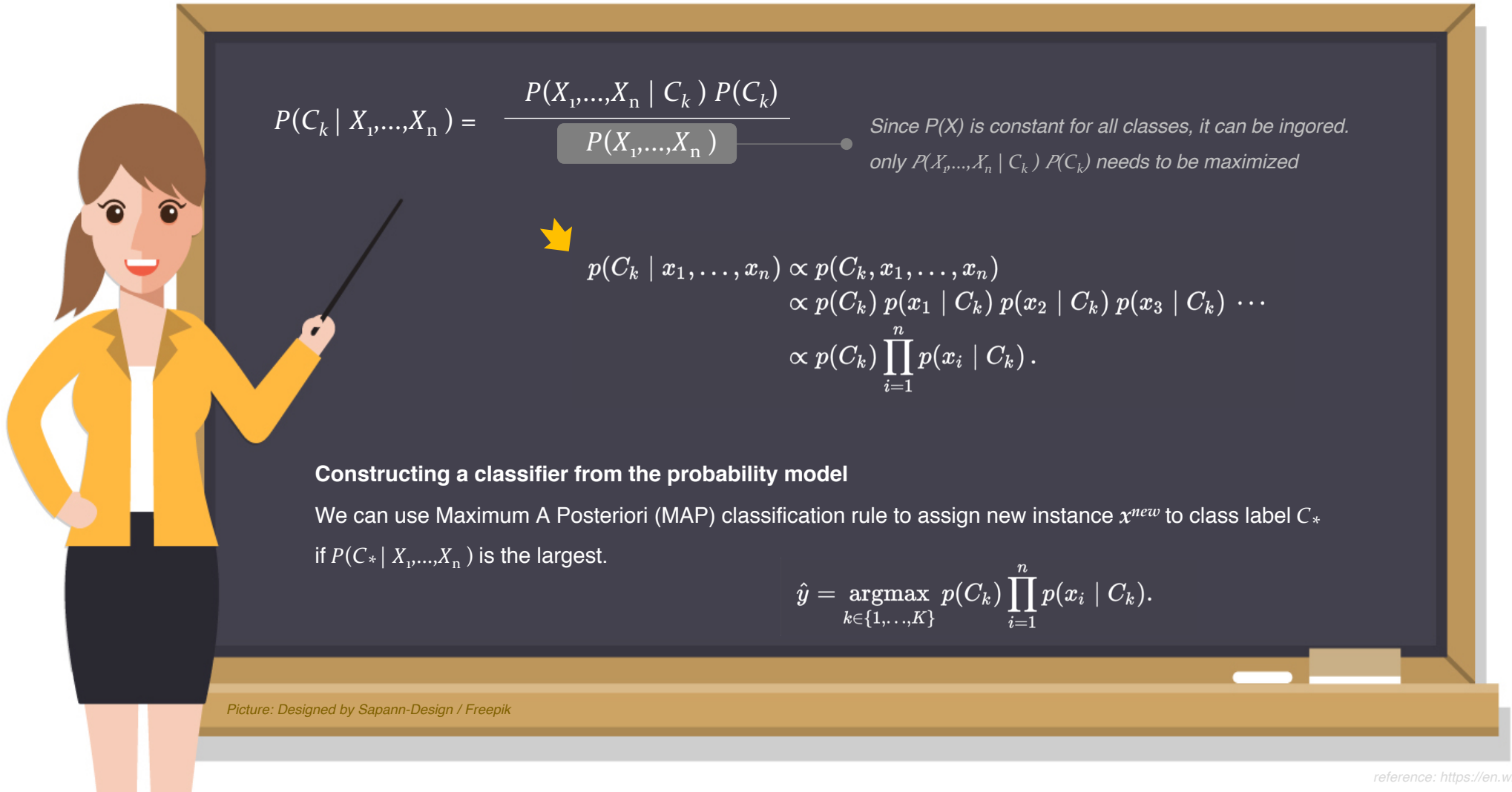
therefore,

$$P(C_k \mid X_1,...,X_n) = \frac{P(X_1,...,X_n \mid C_k)\, P(C_k)}{P(X_1,...,X_n)} = \frac{P(X_1 \mid C_k)\, P(X_2 \mid C_k) \ldots P(X_n \mid C_k)\, P(C_k)}{P(X_1,...,X_n)}$$

1 Source: https://en.wikipedia.org/wiki/Bayesian_probability

:: Naïve Bayes

More details about Naive Bayes classifier



$$P(C_k \mid X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n \mid C_k)\, P(C_k)}{P(X_1,\ldots,X_n)}$$

Since P(X) is constant for all classes, it can be ingored.

only $P(X_1,\ldots,X_n \mid C_k)\, P(C_k)$ needs to be maximized

$$\begin{aligned} p(C_k \mid x_1,\ldots,x_n) &\propto p(C_k, x_1,\ldots,x_n) \\ &\propto p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k) \cdots \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,. \end{aligned}$$

Constructing a classifier from the probability model

We can use Maximum A Posteriori (MAP) classification rule to assign new instance x^{new} to class label C_* if $P(C_* \mid X_1,\ldots,X_n)$ is the largest.

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

Picture: Designed by Sapann-Design / Freepik

Π

Uppercase Π is the product which is the result of multiplying, or an expression that identifies factors to be multiplied in mathematics

reference: https://en.wikipedia.org/wiki/Naive_Bayes_classifier

:: Naïve Bayes

If the features are continuous value

- One common technique for handling continuous values is to use binning to discretize the feature values, to obtain a new set of Bernoulli-distributed features
- **Another approach** : $P(X_j \mid C = c_i)$ is estimated assuming it has **Gaussian (normal) distribution**.

$$\hat{P}(X_j \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(X_j - \mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

μ_{ji} : mean (average) of feature values X_j of examples for which $C = c_i$

σ_{ji} : standard deviation of feature values X_j of examples for which $C = c_i$

Example

Example: Continuous-valued Features

- Temperature is naturally of continuous value.

Play Golf ?

Yes: 25.2, 19.3, 18.5, 21.7, 20.1, 24.3, 22.8, 23.1, 19.8

No: 27.3, 30.1, 17.4, 29.5, 15.1

Estimate mean and variance for each class

- $\mu_{Yes} = 21.64$, $\sigma_{Yes} = 2.35$
- $\mu_{No} = 23.88$, $\sigma_{No} = 7.09$

- **Learning Phase:** Output two Gaussian models for P(temperature | C)

$$\hat{P}(x \mid Yes) = \frac{1}{2.35\sqrt{2\pi}} \exp\left(-\frac{(x-21.64)^2}{2\times 2.35^2}\right) = \frac{1}{2.35\sqrt{2\pi}} \exp\left(-\frac{(x-21.64)^2}{11.09}\right)$$

$$\hat{P}(x \mid No) = \frac{1}{7.09\sqrt{2\pi}} \exp\left(-\frac{(x-23.88)^2}{2\times 7.09^2}\right) = \frac{1}{7.09\sqrt{2\pi}} \exp\left(-\frac{(x-23.88)^2}{50.25}\right)$$

Source of example: Chen Ke, Generative Models and Naïve Bayes

:: Naïve Bayes

Avoid the Zero frequency problem

- If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied [1]

What if you've never seen feature before?

For example , if we receive an email that contains a word that has never appeared in the training emails [2]

- P(**x** l y) will be 0 for all y values. This is bad !
- Just because an event has not happened before, does not mean that it won't ever happen [3]
- Naïve Bayesian prediction requires each conditional probability be non-zero. Otherwise, the predicted probability will be zero

Solution: Use Laplacian Correction [4]

For example, suppose a dataset with 1000 tuples, income=low (0), income= medium (990) and income = high (10)

Use **Laplacian correction** (or Laplacian estimator)

- Adding 1 to each case
  - Prob (income = low) = 1/1003
  - Prob (income = medium) = 991/1003
  - Prob(income = high) = 11/1003
- The "corrected" probablity estimates are close to their "uncorrected" counterparts

1 Source: https://en.wikipedia.org/wiki/Naive_Bayes_classifier
2 Source: http://classes.engr.oregonstate.edu/eecs/winter2011/cs434/notes/bayes-6.pdf
3 Source: https://classes.soe.ucsc.edu/cmps140/Winter17/slides/3.pdf
4 Source: Jiawei Han, Micheline Kamber, and Jian Pei, Data Mining: Concepts and Techniques

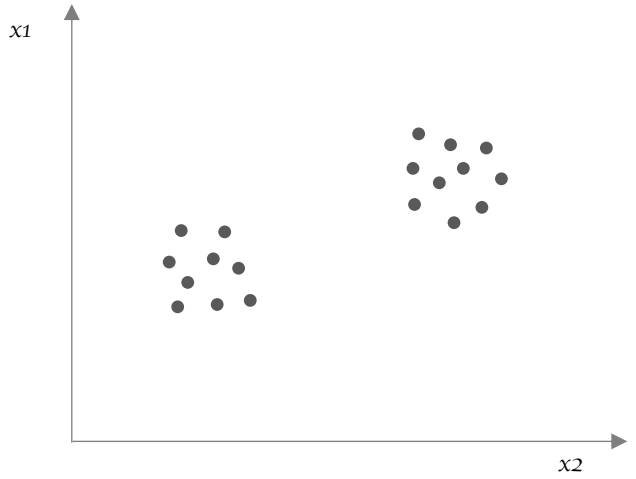
K-mean

Unsupervised Learning

Discover the structure within the un-labeled data

Unsupervised machine learning is the machine learning task of inferring a function to describe hidden structure from "**unlabeled**" data. Examples of unsupervised learning tasks include

- **clustering** (where we try to discover underlying groupings of examples)
- **anomaly detection** (where we try to infer if some examples in a dataset do not conform to some expected pattern)[1].

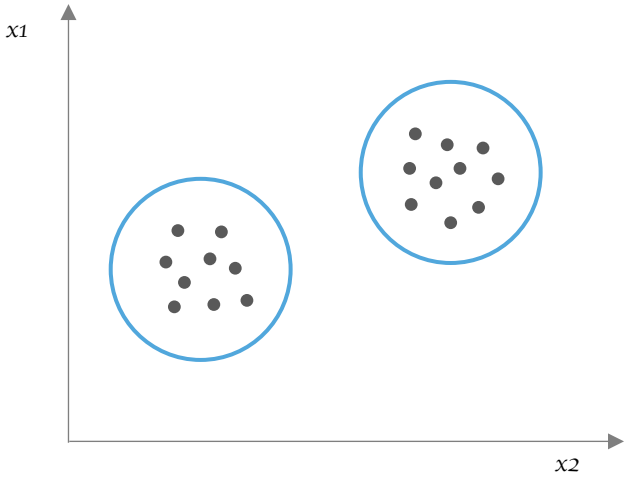


Training set : $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

Example: **Clustering**

A data set without labels is grouped into 2 clusters using K-means algorithm.

Note: There is no target variable to be predicted



Training set : $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

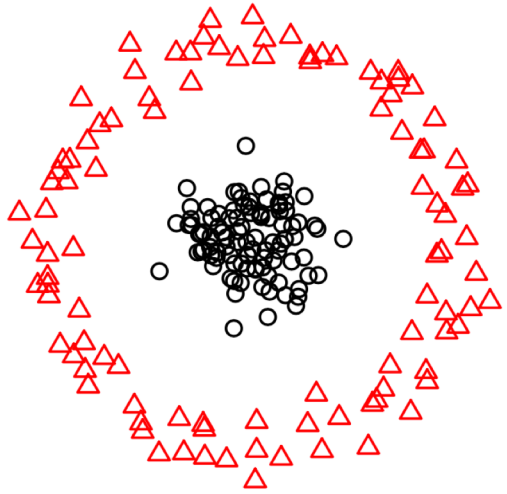
1Source: https://www.coursera.org/learn/machine-learning/lecture/czmip/unsupervised-learning-introduction

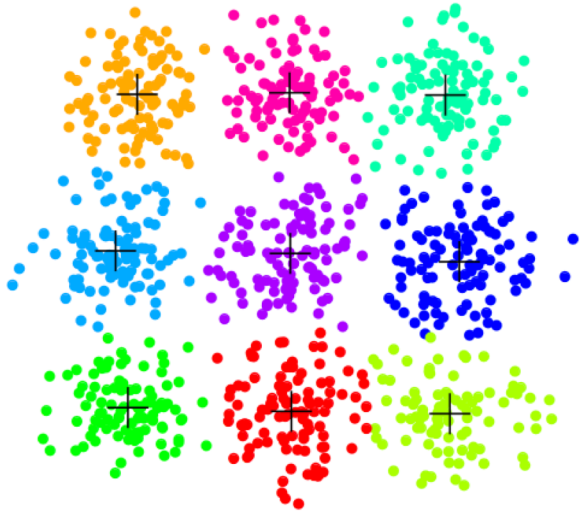
Cluster Analysis

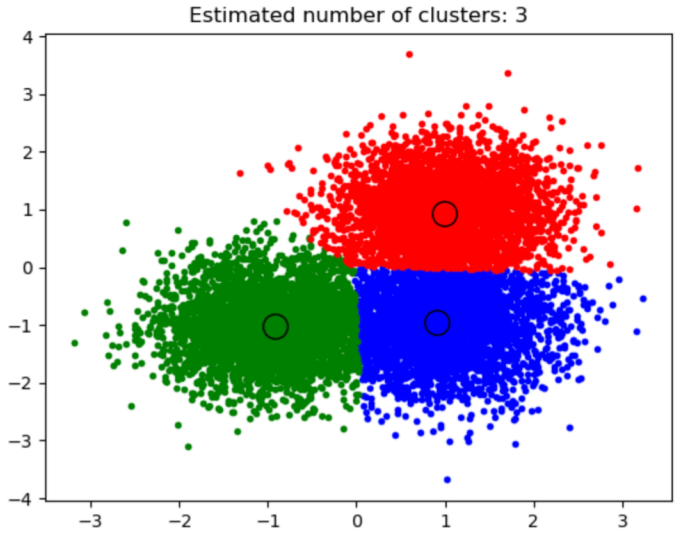
A cluster is a subset of data which are similar

Clustering is a technique for finding similarity groups in a data, called clusters. It attempts to group individuals in a population together by similarity, but not driven by a specific purpose. Clustering is often called an unsupervised learning, as you don't have prescribed labels in the data and no class values denoting a priori grouping of the data instances are given[1].

A cluster is a collection of data items which are "similar" between them, and "dissimilar" to data items in other clusters[2]





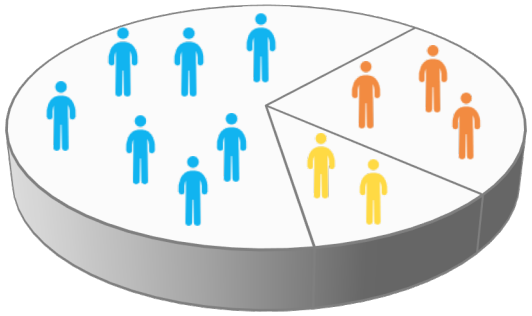


1 Source: http://bigdata-madesimple.com/possibly-the-simplest-way-to-explain-k-means-algorithm/
2 Source: http://www.mit.edu/~9.54/fall14/slides/Class13.pdf
picture: http://scikit-learn.org/stable/auto_examples/cluster/plot_mean_shift.html
Picture : http://www.stat.cmu.edu/~ryantibs/statml/lectures/clustering.pdf

Application of Clustering

Clustering is widely used in many applications

Clustering is hard to evaluate, but very useful in practice. Here are some examples of clustering applications

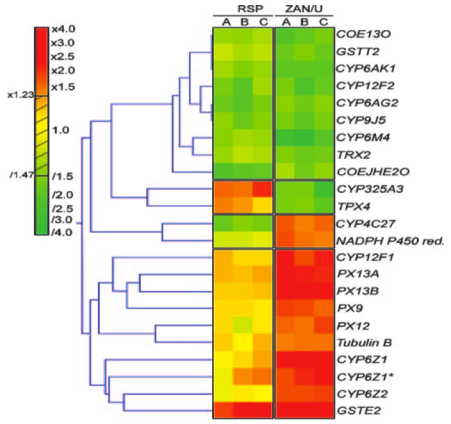


Market segmentation

Discover customer groups and use them for targeted marketing programs

Social network Analysis

Help you find a group of coherent friends on the social network



Genomics

Finding groups of gene with similar expressions

Source of picture: https://www.freepik.com/free-vector/social-media-network_716594.htm#term=social network&page=1&position=3
Source of picture: https://www.researchgate.net/figure/7981540_fig2_Fig-2-Clustering-analysis-of-gene-expression-patterns-between-the-permethrin-resistant

Clustering Methods

More than 100 Clustering algorithms

Clustering has a long history and still is in active research. There are possibly over 100 published clustering algorithms. Some of the methods of clustering include:

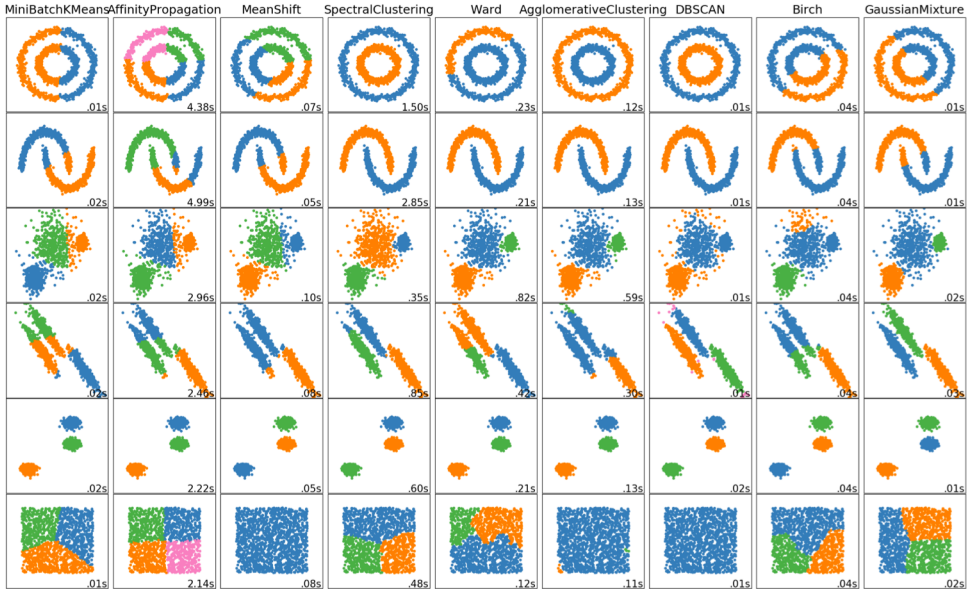
- **Partitioning methods (Centroid-based clustering)**
  - Find mutually exclusive clusters of spherical shape[1]
  - Distance-based [2]
  - **k-means** is most popular algorithm due to its simplicity and efficiency
- **Connectivity-based methods (Hierarchical clustering)**
  - Agglomerative hierarchical clustering (“bottom-up”)
  - Divisive hierarchical clustering (“top-down”)
- **Density-based methods**

  Popular examples of density models are DBSCAN and its variations (e.g. OPTICS)
- **Grid-based methods**

  Use a multi-resolution grid data structure[3]
- **Distribution-based clustering**

  A popular example of these models is Expectation-Maximization algorithm

Comparing different clustering algorithms on toy dataset



http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

1, 2,3 Source: http://users.math.yale.edu/users/gw289/CpSc-445_2016/Slides/CPSC445%20-%20Topic%2008%20-%20Clustering.pdf

Let’s start to learn k-means

k-Means

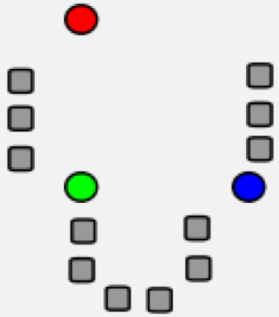
Partitions data into k number of mutually exclusive clusters. How well a point fits into a cluster is determined by the distance from that point to the cluster’s center[1].

1 Source : Mathworks, Applying Supervised Learning

:: k-Means

A graphical view of K-means algorithm

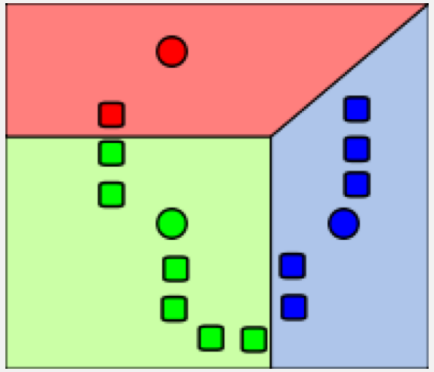
Chooses the initial centroids



1

k initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

Cluster assignment step

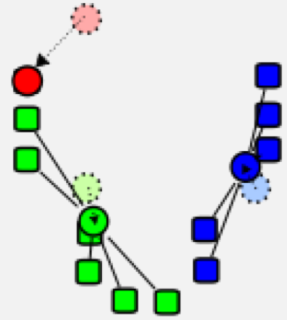


2

k clusters are created by associating every observation with the nearest mean.

In Cluster assignment step, the algorithm goes through each of the data points and depending on which cluster is closer, whether the red cluster centroid or the blue cluster centroid or the green; It assigns the data points to one of the three cluster centroids.

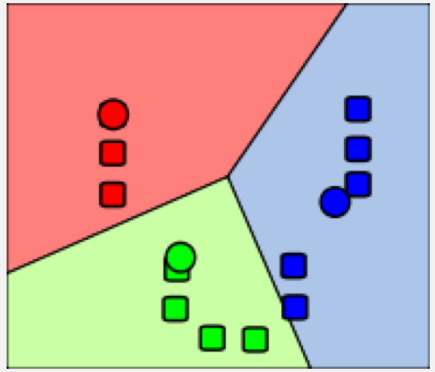
Move centroid step



3

The centroid of each of the *k* clusters becomes the new mean.

In move centroid step, K-means moves the centroids to the average of the points in a cluster. In other words, the algorithm calculates the average of all the points in a cluster and moves the centroid to that average location.



4

Steps 2 and 3 are repeated until convergence has been reached

In other words, it repeats until the centroids do not move significantly.

Source of diagram: https://en.wikipedia.org/wiki/K-means_clustering

:: k-Means

K-means (Llyod's) clustering algorithm

K-Means clustering intends to partition n objects into k clusters in which each object belongs to the cluster with the nearest mean

1. Randomly initialize *K* **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$

2. Repeat until convergence : {

    - For every example i, set

$$c^{(i)} := j \quad \text{that minimizes} \quad ||x^{(i)} - \mu_j||^2$$

Centroid for cluster :
μ_j = mean of the cluster

Euclidean distance

Cluster assignment step

Assign each data point to the nearest cluster whose mean has the least squared Euclidean distance.

    - For every k, set

μ_k := average (mean) of points assigned to the cluster k

$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x^{(i)}$$

Update centroid step (⇒ Move Centroid)

Calculate the new means to be the centroids of the observations in the new clusters.

Recommended video:

https://www.youtube.com/watch?v=LvgcfMOyREE&t=159s

}

Source of formular: http://stanford.edu/~cpiech/cs221/handouts/kmeans.html

Weakness of K-means

K-Means is simple and can be used for a wide variety of data type It's also quite computationally efficient , even though multiple runs are often performed[1].. However, similar to other algorithm, K-mean clustering has many weaknesses. For example,

- **The number of cluster "*k*" must be specified in advance.**
  The number of clusters k is an input parameter: an inappropriate choice of k may yield poor results[2]. Unfortunately there is no global theoretical method to find the optimal number of clusters. A practical approach is to compare the outcomes of multiple runs with different *k* and choose the best one based on a predefined criterion. In general, a large *k* probably decreases the error but increases the risk of overfitting[3]. For details, see next slide.

- **Sensitive to initial centroids selection, which leads to unwanted solution**
  One of the major drawback of K-means clustering is the random choice of initial cluster centers .For every different run of the algorithm on the same dataset, the initial centers might be different. This may lead to different clustering results. Therefore, it is very hard to repeat the clustering results.

  There are some methods for addressing this issue.
  - k-means is usually run many times, starting with different random centroids each time. The results can be compared by a numeric measure such as the clusters' distortion, which is the sum of the squared differences between each data point and its corresponding centroid [4]. Then, pick clustering that gave lowest cost function (distortion).
  - One of the other methods is K-means ++ algorithm.

For more details, ⇒ Video (7min) by Andrew Ng, https://www.youtube.com/watch?v=PpH_hv55GNQ

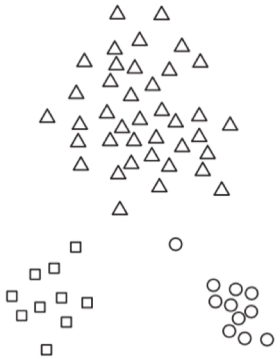
- **k-means can only handle numerical data**[5]
  Categorical data (i.e., category labels such as gender, country, browser type) needs to be encoded or separated in a way that can still work with the algorithm[6].

1 Source: https://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf 2 source: https://en.wikipedia.org/wiki/K-means_clustering
3Source: http://www.saedsayad.com/clustering_kmeans.htm 4 Source: Data Science for business
5 Source: https://www.inovex.de/blog/disadvantages-of-k-means-clustering 6Source: https://www.datascience.com/blog/k-means-clustering

Weakness of K-means *(continued)*

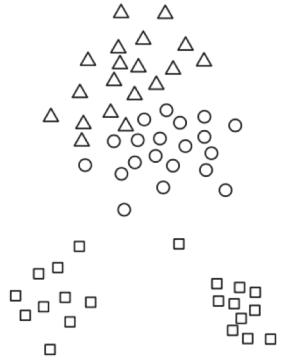
- **The algorithm may get stuck in the local optimum**

  Given enough time, K-means will always converge, however this may be to a local minimum (namely, a locally best clustering). This is highly dependent on the initialization of the centroids[1].



(a) Optimal Clustering

The figure shows a clustering solution that is the **global minimum**



(b) Suboptimal Clustering

The figure shows a suboptimal clustering that is only a **local minimum.**

For more details, check it out on this short video (7minutes) : https://www.youtube.com/watch?v=PpH_hv55GNQ

1 Source: http://scikit-learn.org/stable/modules/clustering.html

Weakness of K-means *(continued)*

- **Sensitive to outliers and noise, which results in an inaccurate partition**

  When the squared error criterion is used, outliers can unduly influence the clusters that are found[2]. Weakness of arithmetic mean is not robust to outliers. Very far data from the centroid may pull the centroid away from the real one[3].

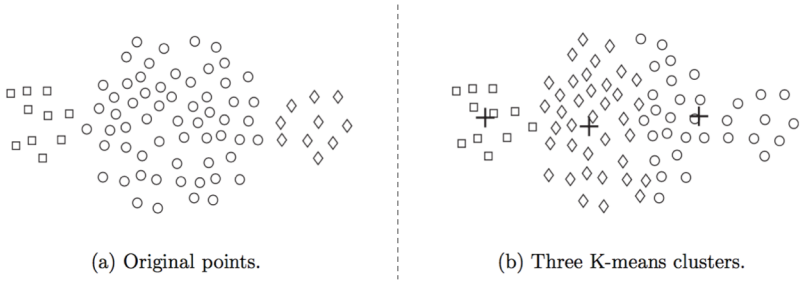
  - Because of this, it is often useful to discover outliers and eliminate them beforehand[4].
  - It is important, however, to appreciate that there are certain clustering applications for which outliers should not be eliminated[5].

2，4 & 5 Source: Introduction to Data Mining， https://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf
3 Source: http://people.revoledu.com/kardi/tutorial/kMean/Weakness.htm

Weakness of K-means *(continued)*

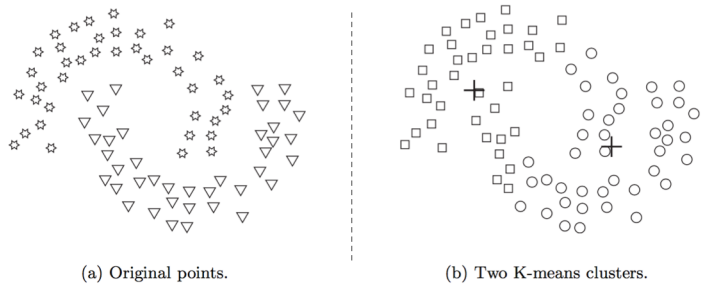
- **K-means cannot handle non-globular clusters or clusters of different sizes and densities**

  K-means has difficulty detecting the “natural” clusters, when clusters have non-spherical shapes or widely different sizes or densities.



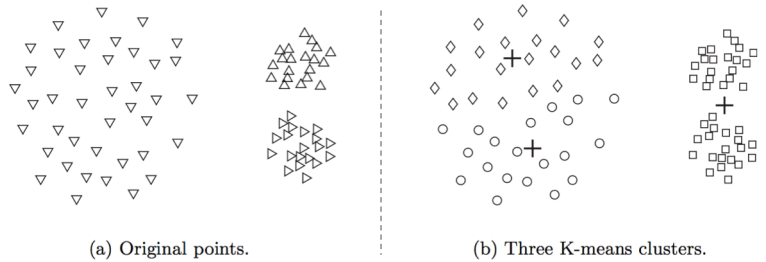
(a) Original points. (b) Three K-means clusters.

Figure 1: K-means with clusters of different size



(a) Original points. (b) Two K-means clusters.

Figure 3: K-means with non-globular clusters

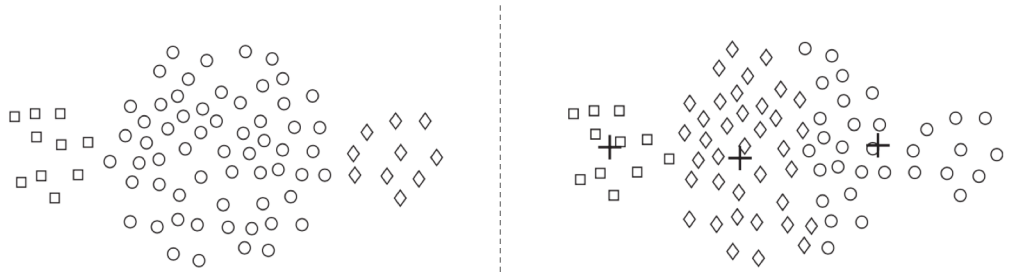


(a) Original points. (b) Three K-means clusters.

Figure 2: K-means with clusters of different density

Weakness of K-means *(continued)*

- **K-means cannot handle clusters of different sizes**



(a) Original points. (b) Three K-means clusters.

Figure 1: K-means with clusters of different size

In Figure 1, K-means cannot find the three natural clusters because one of the clusters is much larger than the other two, and hence, the larger cluster is broken, while one of the smaller clusters is combined with a portion of the larger cluster.

Weakness of K-means *(continued)*

- **K-means cannot handle clusters of densities**

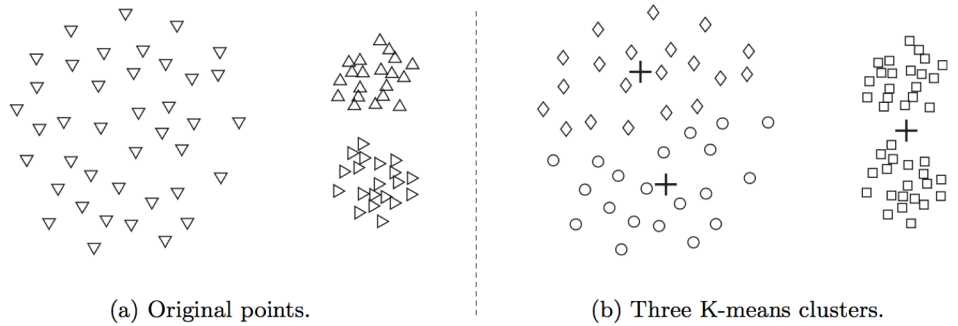
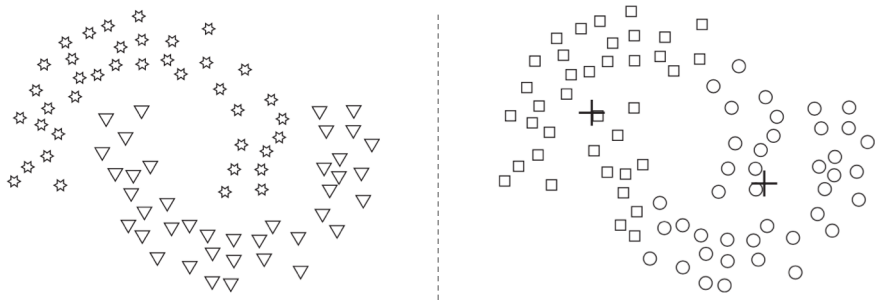


Figure 2: K-means with clusters of <u>different density</u>

In Figure 2, K-means fails to find the three natural clusters because the two smaller clusters are much denser than the larger cluster.

Weakness of K-means *(continued)*

- **K-means cannot handle non-globular clusters**



(a) Original points. (b) Two K-means clusters.

Figure 3: K-means with non-globular clusters

In Figure 3 , K-means finds two clusters that mix portions of the two natural clusters because the shape of the natural clusters is not globular.

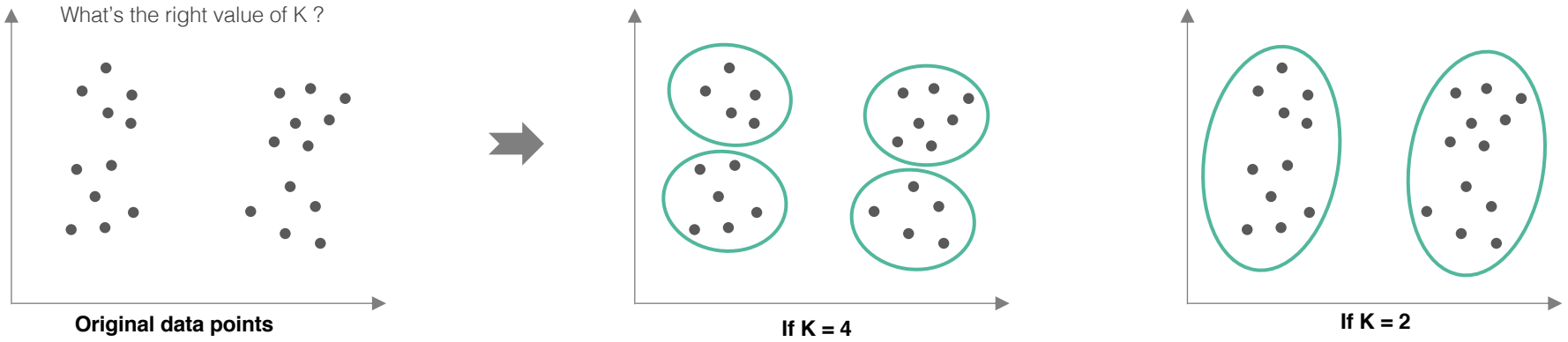
:: k-Means

How to choose the value of *k* ?

In k-means clustering we must choose a value for k. This is still an active area of research and there are no definitive answers[1]. By far the most common way of choosing the number of clusters, is still choosing it **manually** by looking at visualizations or by looking at the output of the clustering algorithm or something else[2]

Why is it not easy to determine the right value of K?

Because it is often generally ambiguous about how many clusters there are in the data. For instance, the data points in the following example can be viewed as 4 clusters or 2 clusters. There are no absolutely right answer for that



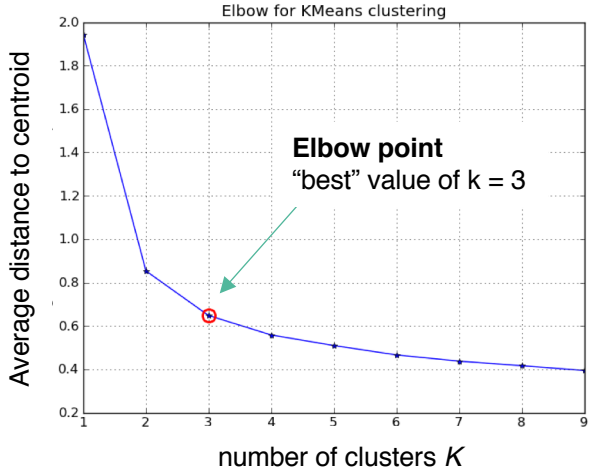
Source: CMU, http://www.stat.cmu.edu/~ryantibs/statml/lectures/clustering.pdf
2 Source:https://www.youtube.com/watch?v=lbR5br5yvrY

:: k-Means

How to choose the value of *k* ? *(continued)*

In general, there is no method for determining exact value of *K*, but an accurate estimate can be obtained using the following techniques[1].

- **Elbow Method**
  - Try different k, looking at the change in the average distance to the centroid, as k increases[2].
  - Average falls rapidly until right k, then falls much more slowly[3]
  - It might be hard to determine where the elbow of the curve actually is



Unfortunately, the elbow method often does not work well in practice because there may not be a well-defined elbow[4].

- **Silhouette method**

  The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). Silhouette values lies in the range of [-1, 1]. Silhouette coefficients near +1 indicate that the sample is far away from the neighboring clusters. More details ⇒ Video (3 min)

Video : Choosing The Number Of Clusters

Youtube Video (8 min)

https://www.youtube.com/watch?v=lbR5br5yvrY&t=24s

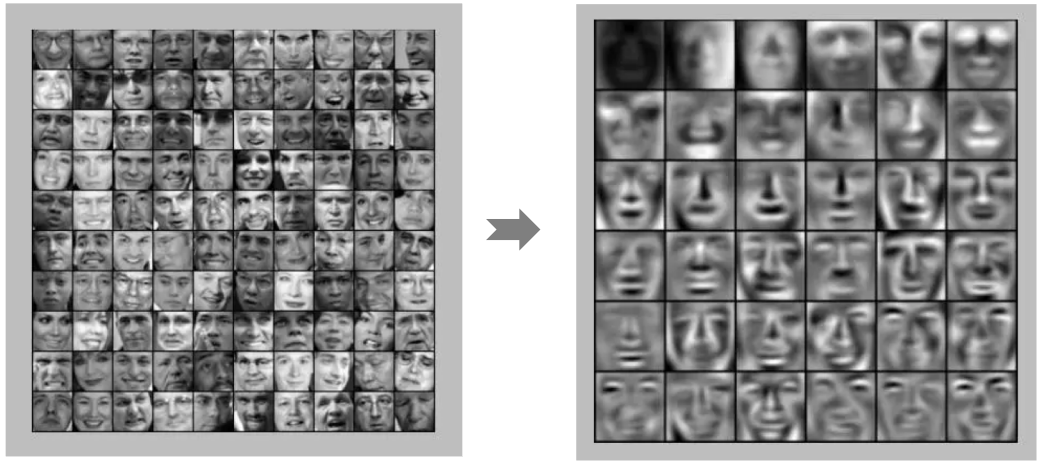
1 Source: https://www.datascience.com/blog/k-means-clustering
2&3 Source: https://www.youtube.com/watch?v=RD0nNK51Fp8&t=415s
4Source: https://www.coursera.org/learn/machine-learning/lecture/Ks0E9/choosing-the-number-of-clusters

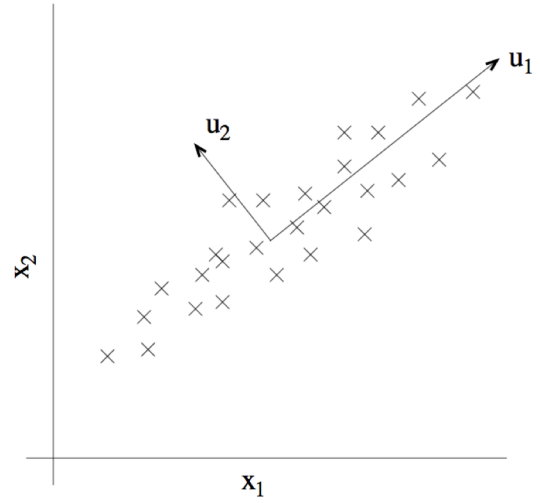
PCA

:: PCA

PCA is the most popular dimensionality reduction algorithm

- Principal Component Analysis (PCA) is by far the most popular dimensionality reduction algorithm.
- PCA is a technique to transform a dataset onto a lower dimensional subspace while preserving as much information as possible.
- It is widely used for applications such as dimensionality reduction, lossy data compression, feature extraction, and data visualization[1].
- It's a kind of unsupervised learning
- Before you implement PCA, first try running whatever you want to do with the original data/raw data. Only if that doesn't do what you want, then implement PCA[2]





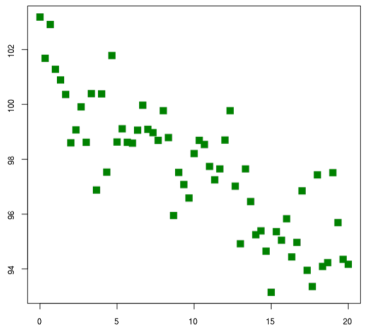
1 Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow
2 Source: https://www.coursera.org/learn/machine-learning/lecture/RBqQI/advice-for-applying-pca

:: The motivation for dimension reduction

Why reduce dimensions ?

For the sake of **Data Visualization**

- It's difficult to visualize high-dimensional data.
- So you can use dimensionality reduction, in order to reduce data from 50 dimensions or whatever, down to two dimensions, or maybe down to three dimensions, so that you can plot it and understand your data better.



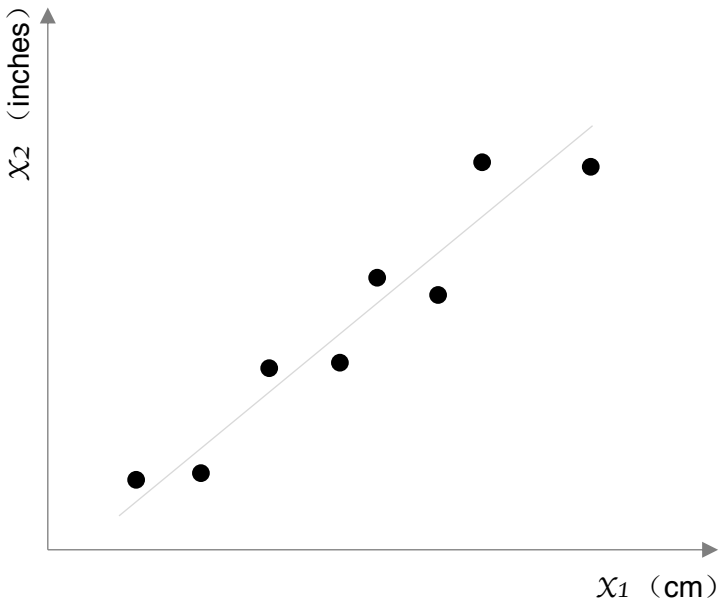
For the sake of **Data Compression**

- Remove redundant and noisy features
- Reduce memory/disk storage
- Speed up other learning algorithm because of few inputs

Source: https://www.coursera.org/learn/machine-learning/lecture/t6pYD/motivation-ii-visualization

:: The motivation for dimension reduction

Redundant information



“inch” = “cm”

In this example, there are two features:

- x1 is actually the length of something in centimeters
- x2 is the length of the same thing in inches.

The data points doesn’t perfectly lie on a straight line. It’s just due to measurement noise, number round-off , etc.

You can reduce redundancy

In this example, x1 is highly correlated with x2. Actually they measure the same thing just with different unit.

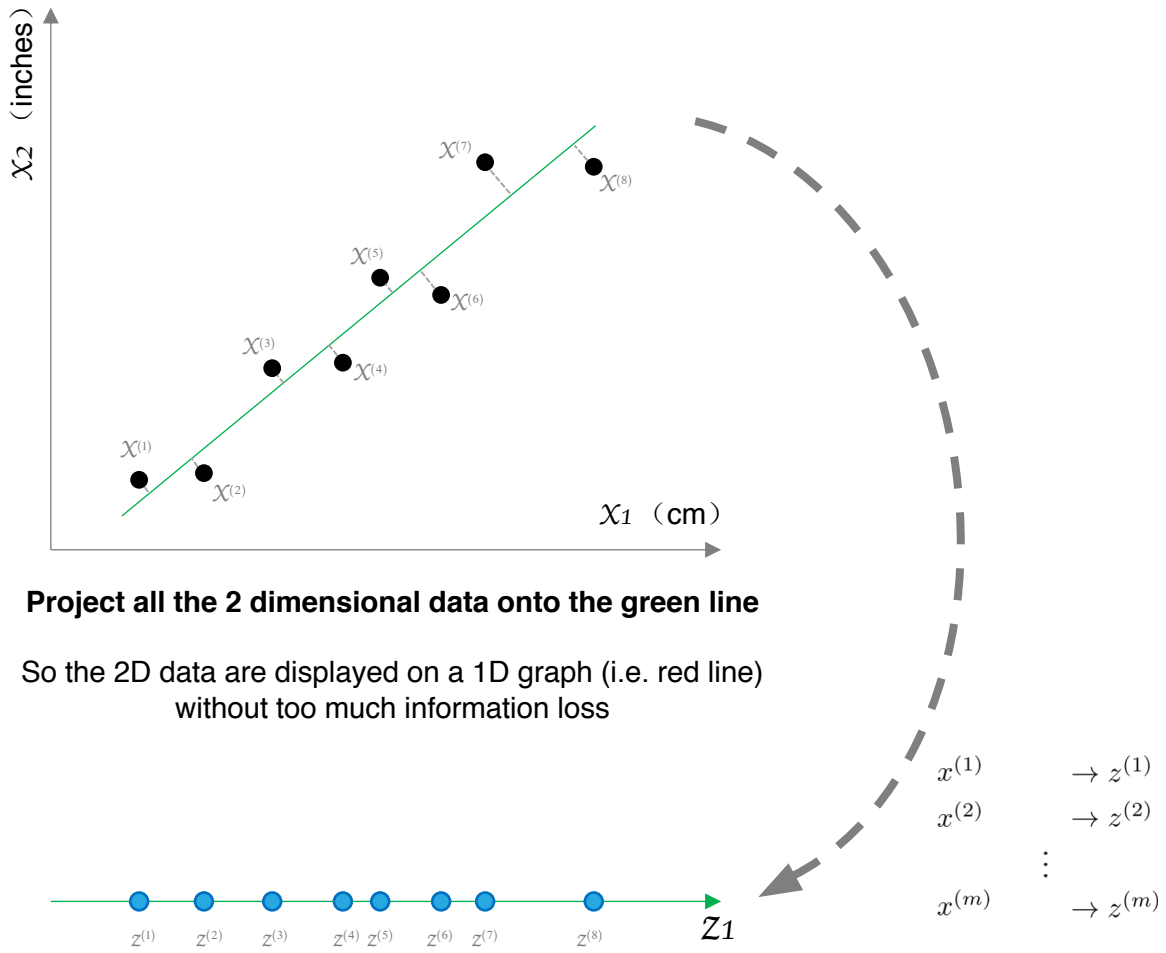
“…So, this gives us a highly redundant representation and maybe instead of having two separate features x1 then x2, both of which basically measure the length, maybe what we want to do is reduce the data to one-dimensional and just have one number measuring this length.

And if we can reduce the data to one dimension instead of two dimensions, that reduces the redundancy. [1] ”

Source: Andrew Ng , https://www.youtube.com/watch?v=Zbr5hyJNGCs

:: PCA

Data Compression example



Project all the 2 dimensional data onto the green line

So the 2D data are displayed on a 1D graph (i.e. red line) without too much information loss

come up with a new feature z1

Reduce data from 2D to 1D[1]

How to reduce the dimension of data from two-dimensional to one-dimensional?

Find a line onto which to project the data points. How to find such line will be dwelled on later.

The original data set by projecting all of my original examples onto this green line over here, then I need only one number, I need only real number to specify the position of a point on the line, and so what I can do is therefore use just one number to represent the location of each of my training examples after they've been projected onto that green line.

So this is an approximation to the original training self because I have projected all of my training examples onto a line.

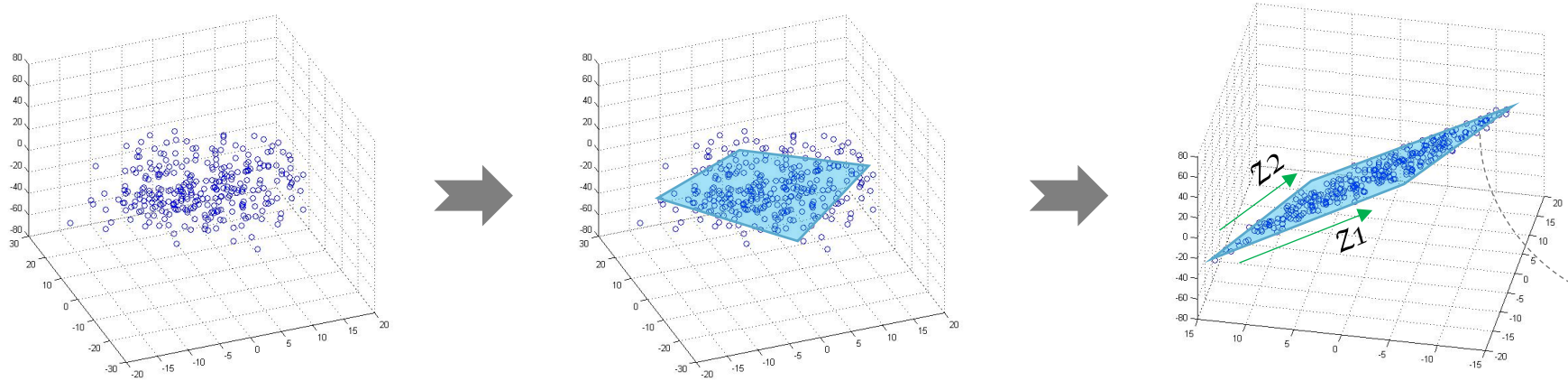
PCA algorithm aims to reduce the dimensionality of the data, while preserving its information (or minimizing the loss of information)

Source: https://www.youtube.com/watch?v=Zbr5hyJNGCs

:: PCA

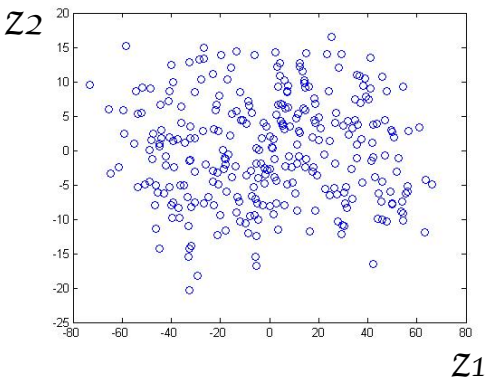
Data Compression example

Reduce data from 3D to 2D



project the data down onto a two dimensional plane

As you can see all the data lies on a plane, cause we've projected everything onto a plane, and so what this means is that now I need only two numbers, z1 and z2, to represent the location of point on the plane.



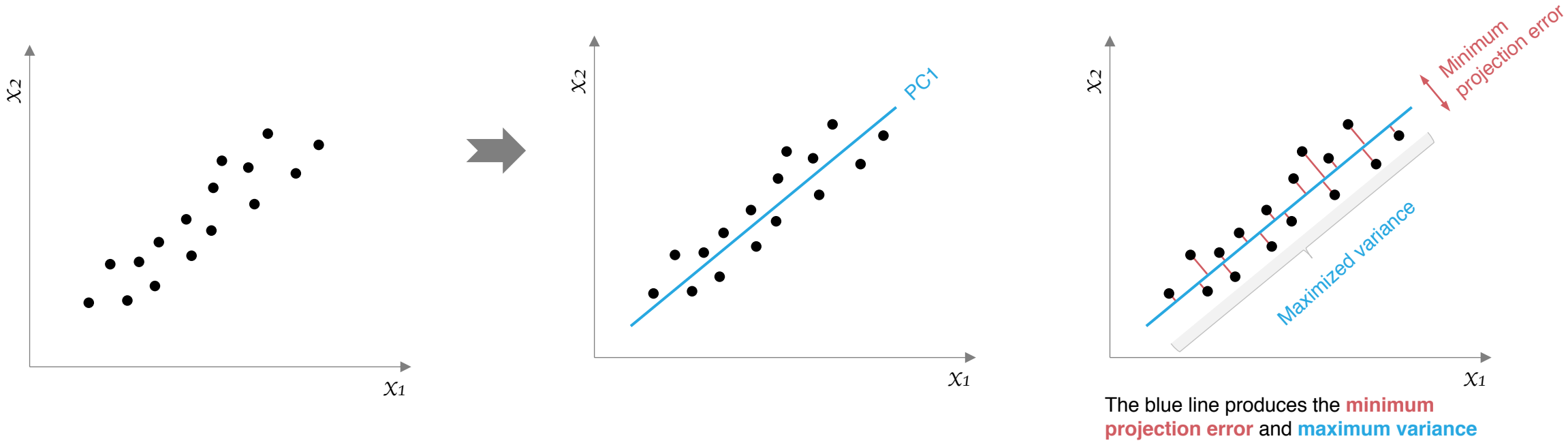
Recommended Video (10 minitues)

https://www.youtube.com/watch?v=Zbr5hyJNGCs

:: PCA

What's the criteria for finding the "best-fit" line ?

The goal of PCA is to project the data onto lower dimensional linear space, such that the variance of the projected data is maximized[1].



The blue line produces the **minimum projection error** and **maximum variance**

Select an axis or find a direction that:

- **The variance of projected data is maximized** : Pick a line along which the data is spread out the most
- **Or the projection error is minimized** : Linear projection should minimizes the average projection cost, defined as the mean squared distance between the data points and their projections

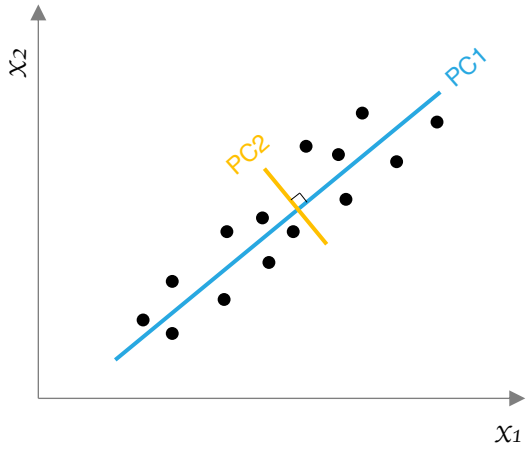
For more information, pleas watch Youtube Video (9minutes) : https://www.youtube.com/watch?v=T-B8muDvzu0&t=160s

1Source: Pattern Recognition and Machine Learning

:: PCA

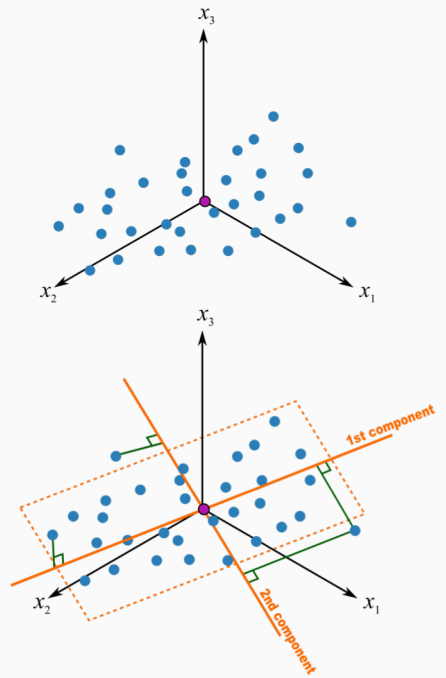
Identify the principal components

PCA uses orthogonal projection of highly correlated variables to a set of values of linearly uncorrelated variables called principal components. This linear transformation is defined in such a way that the first principal component has the largest possible variance[1].



- PC1 (the first principal component) is the direction in the original data that contains the most variance.
- PC2 captures the direction with 2rd most variation

- PCA identifies the axis that accounts for the largest amount of variance in the training set[2].
- It also finds a second axis, **orthogonal** to the first one, that accounts for the largest amount of remaining variance[3]. In this 2D example, it is the yellow line.
- If it were a higher-dimensional dataset, PCA would also find a third axis, orthogonal to both previous axes, and a fourth, a fifth, and so on—as many axes as the number of dimensions in the dataset[4].



Good article:
[Geometric explanation of PCA]

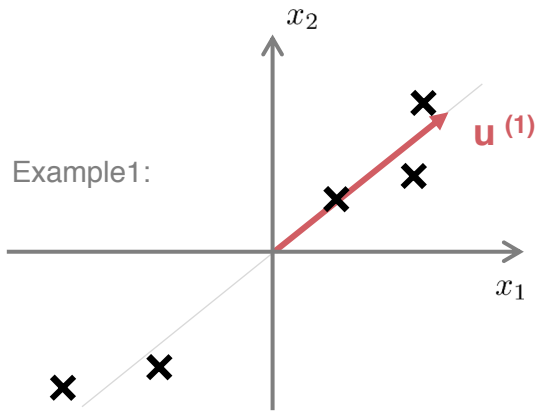
1 Source: https://www.linkedin.com/pulse/using-pca-guide-r-sas-sanchit-tiwari/
2,3,4 Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow

:: PCA

Problem formulation for PCA

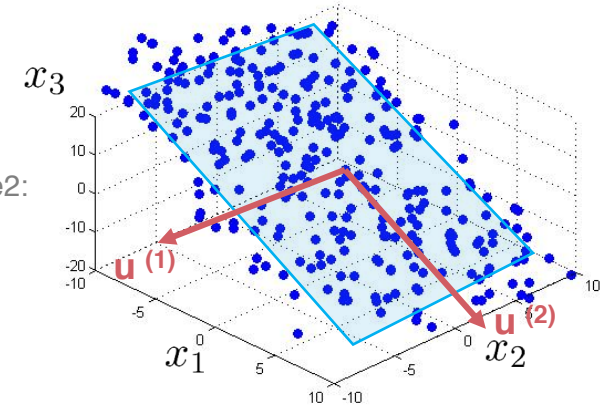
PCA is trying to find a lower dimensional surface onto which to project data so as to minimize the projection squared error, to minimize the squared distance between the each point and the location of where it gets projected[1].

Example1:



2D → 1D
(K= 1)

Example2:



3D → 2D
(K= 2)

Reduce from 2-dimension to 1-dimension: [2]

- Find a direction (a vector $u^{(1)}$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: [3]

- Find *k* vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

For more information ⇒ https://www.youtube.com/watch?v=T-B8muDvzu0&t=160s

1,2, 3 Source : Andrew Ng, https://www.youtube.com/watch?v=T-B8muDvzu0&t=160s

:: PCA

Some basic statistics knowledge

Variance

Variance measures how far a data set is spread out. The technical definition is “The average of the squared differences from the mean,”

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

where μ is the average value of X

Covariance

Covariance is the measure of how two different variables/dimensions change together.

- do x and y tend to increase together ?
- or does y decrease as x increase ?
- The covariance between two variables, X and Y, can be given by the following formula:

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

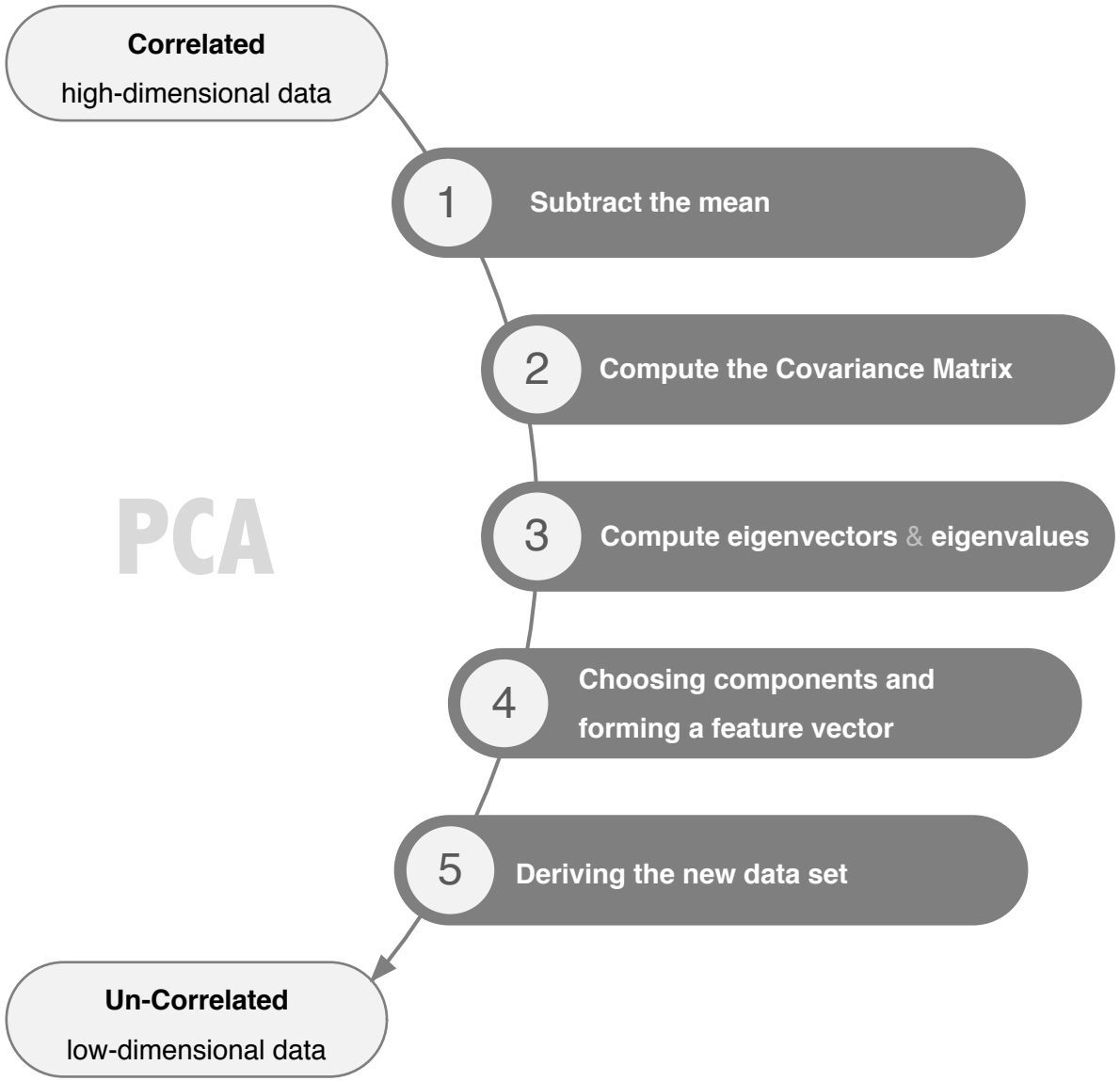
Covariance Matrix

A covariance matrix is a square matrix that contains the Variances and covariance associated with several variables. The diagonal elements of the matrix contain the variances of the variables and the off-diagonal elements contain the covariance between all possible pairs of variables[1].

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

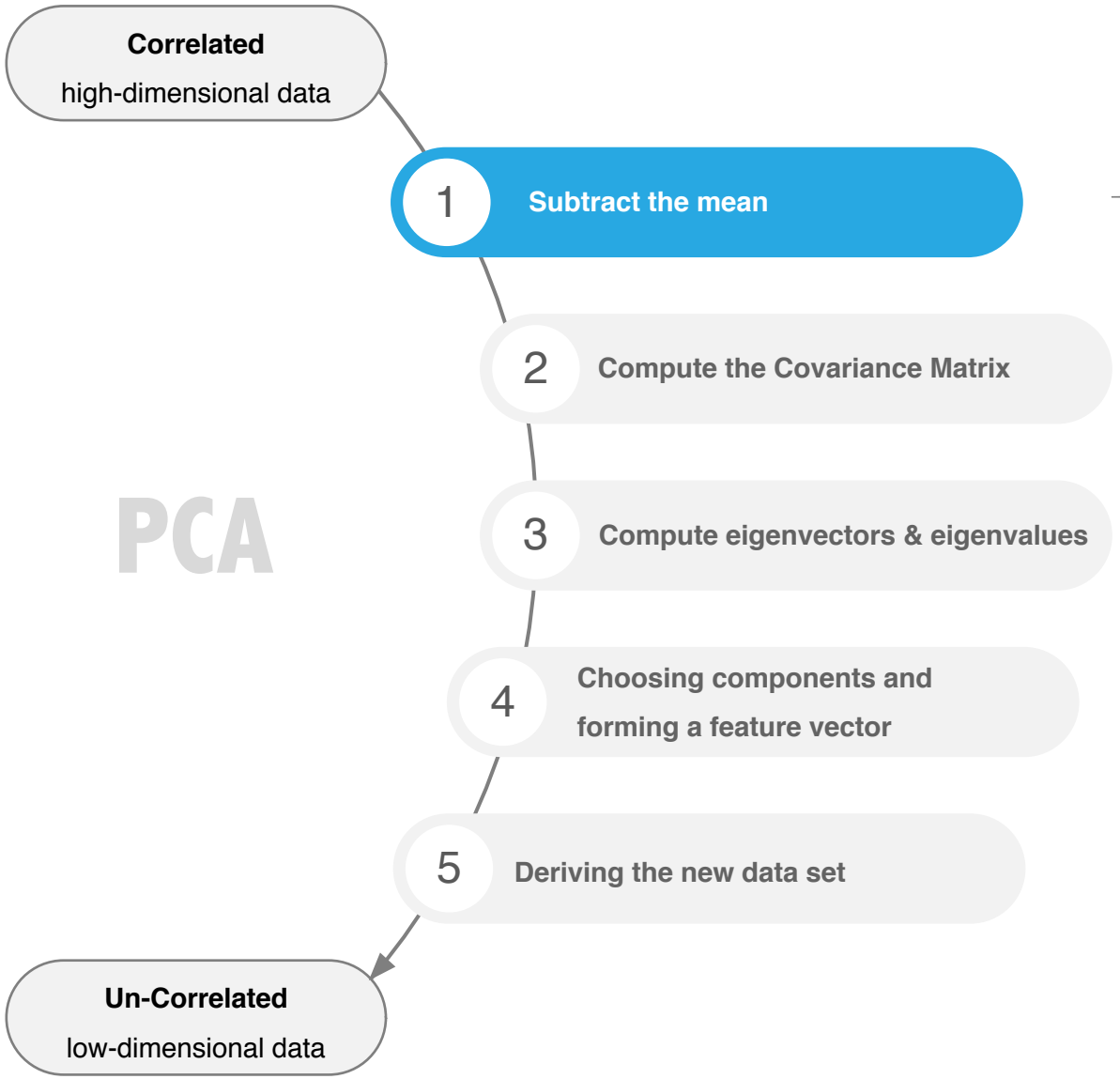
1 Source: https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/anova/supporting-topics/anova-statistics/what-is-the-variance-covariance-matrix/

:: PCA



1Source: Pattern Recognition and Machine Learning, Page 563
2 Source: unknown
Reference: Lindsay I Smith, A tutorial on Principal Components Analysis

:: PCA



Preprocessing Data - mean normalization

- Subtract the mean from each of the data dimensions. This produces a data set whose mean is zero [1].
- Subtracting the mean makes variance and covariance calculation easier by simplifying their equations. The variance and covariance value are not affected by the mean value [2].

Example:

Data =

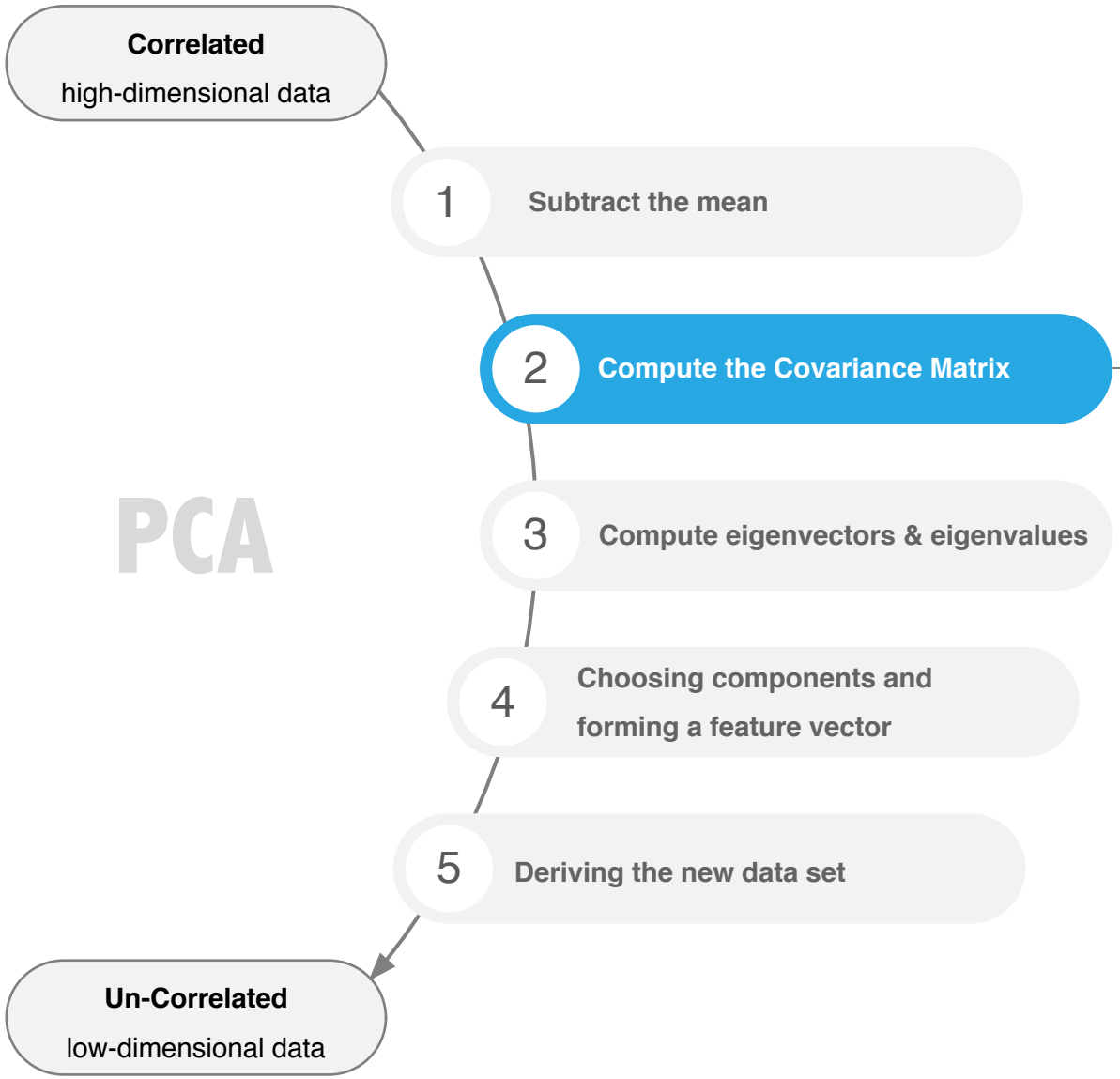
x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

1 Source: : Lindsay I Smith, A tutorial on Principal Components Analysis
2 Source: www.cs.cmu.edu/afs/cs/academic/class/15385-s06/lectures/ppts/lec-19.ppt

:: PCA



Compute the covariance matrix

- Given data $\{x_1, x_2, \ldots x_n\}$, compute the covariance matrix **S**

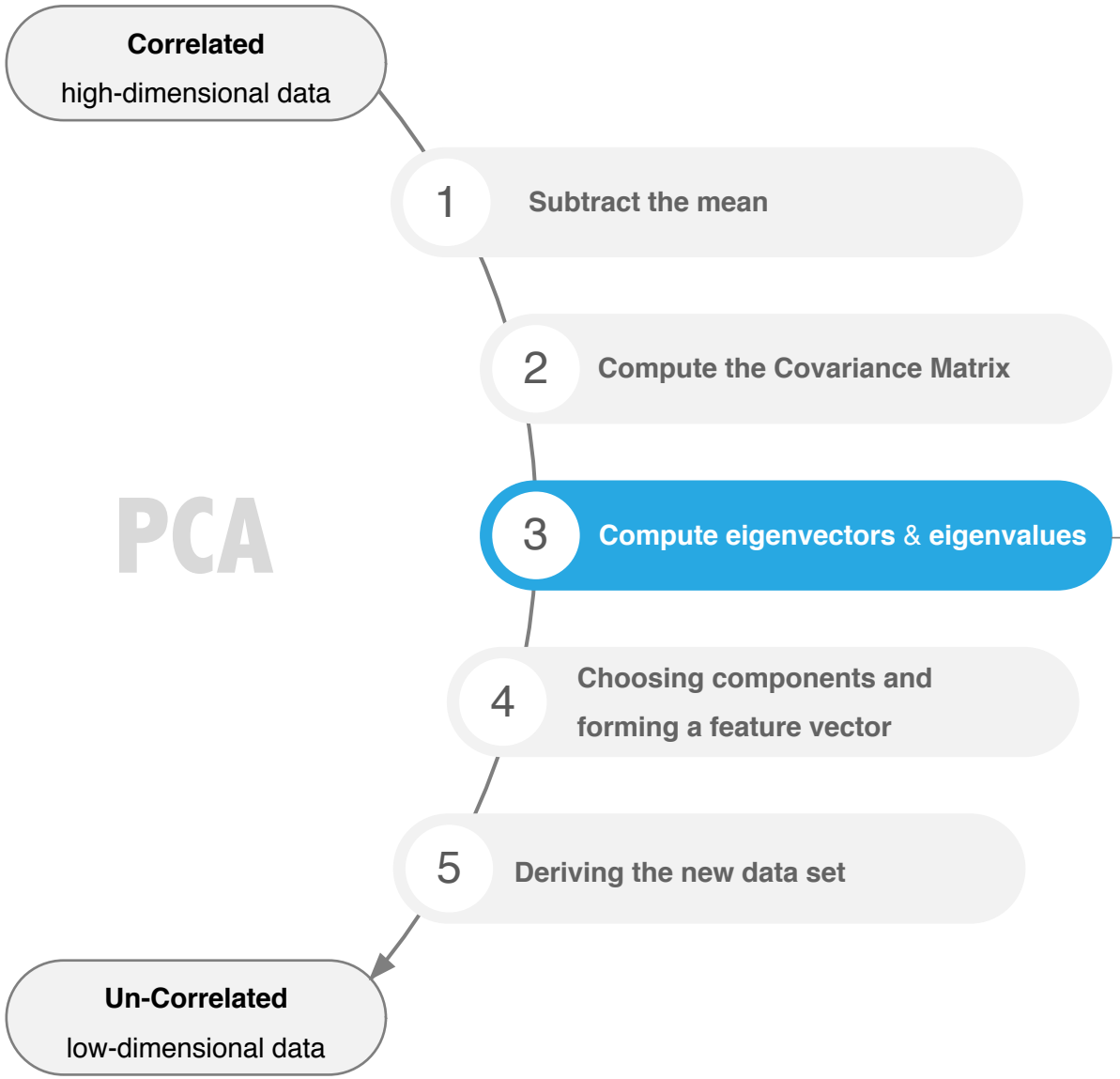
$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}$$

where mean vector $\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$

Example:

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

:: PCA



Compute eigenvectors and eigenvalue

- **What's Eigenvector and Eigenvalue ?**
  Each eigenvector (principal component) has an eigenvalue, which measure the amount of variance along that axis[1]
  - For details, ⇒ very good video (4mintues)

$$A \cdot \vec{v} = \lambda \cdot \vec{v}$$

Matrix **A** Eigenvector

λ is a scalar, known as eigenvalue associated with Eigenvector V

- **Calculate the eigenvectors and eigenvalues**
  Since the covariance matrix is square, we can calculate the eigenvectors and eigenvalues for the covariance matrix **S**

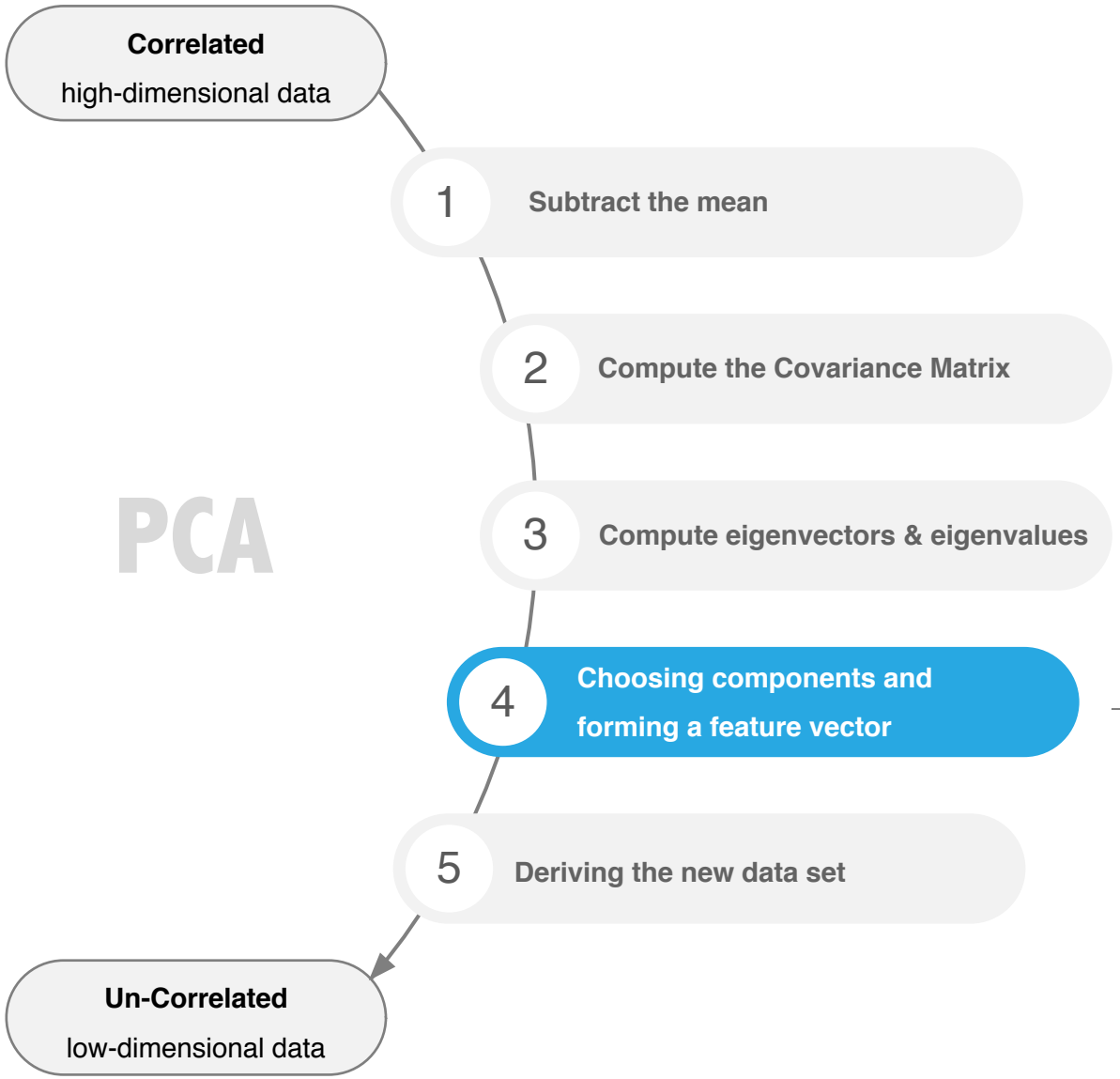
Example:

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

Source: https://www.youtube.com/watch?v=TSYL-oHx4T0

:: PCA



Pick top *k* principal components

1. Rank the eigenvectors by eigenvalue, highest to lowest

- Eigenvalues : $\lambda_1 > \lambda_2 > \ldots, \lambda_n$
- Eigenvector with largest eigenvalue captures the most variation among training vectors[2]
- Eigenvector with smallest eigenvalue has least variation[3]

2. Choose only the top *k* eigenvectors which explain most of variance (e.g 95% of variance)

- Select the top k eigenvectors $(u_1, \ldots, u_k)$ to form a matrix of vectors.

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n} \quad \Rightarrow \quad U_{reduce} = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}$$

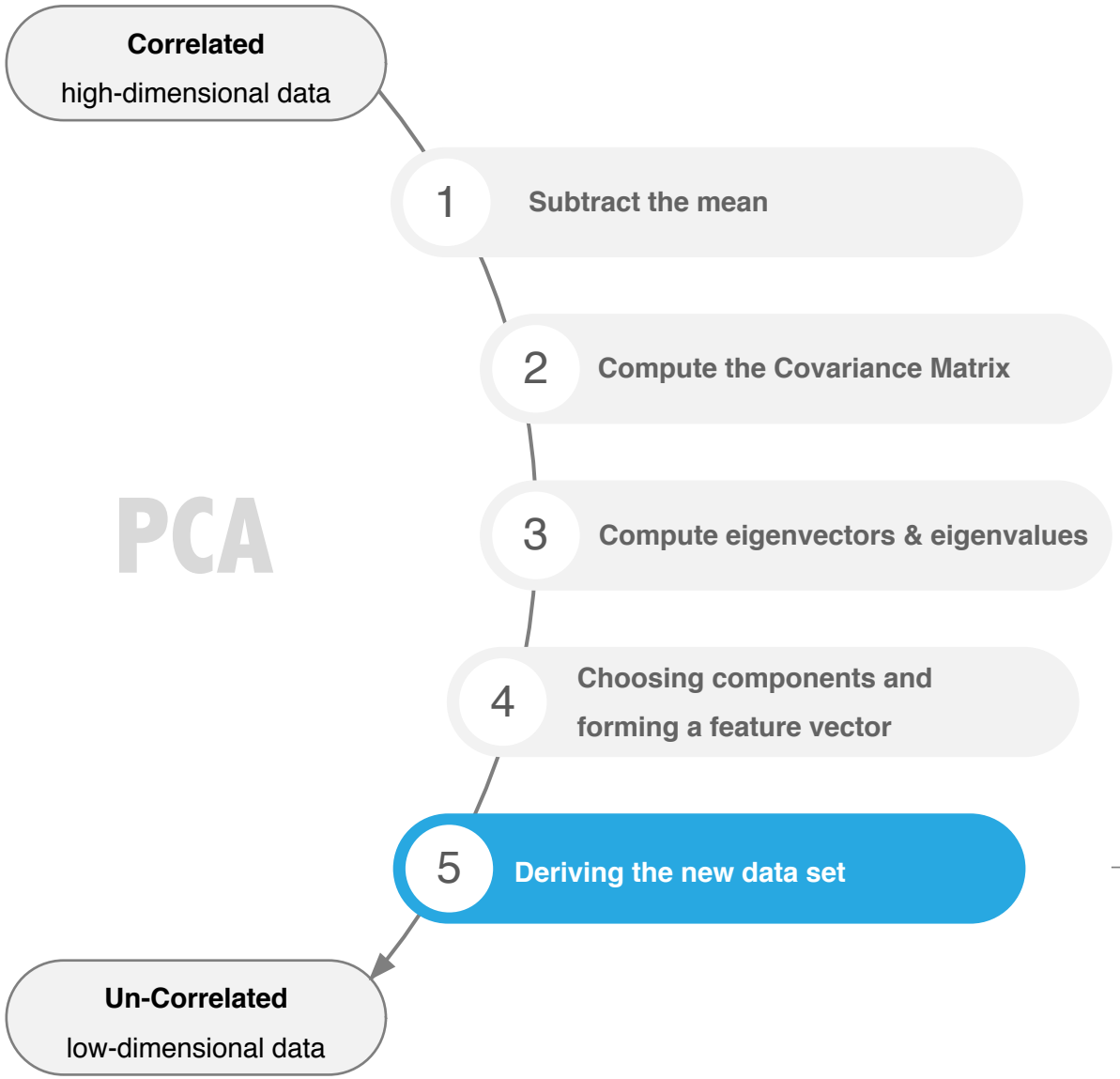
K eigenvectors

  - This corresponds to choosing a “linear subspace”. The eigenvectors will form the new axes. The eigenvectors with larger eigenvalues correspond to directions in which the data varies more.

- You can ignore the components of lesser significance. You do lose some information, but if the eigenvalues are small, you don’t lose much[4]

2 , 3 Source: http://www.cs.cmu.edu/afs/cs/academic/class/15385-s06/lectures/ppts/lec-19.ppt
4 Source: : Lindsay I Smith, A tutorial on Principal Components Analysis

:: PCA



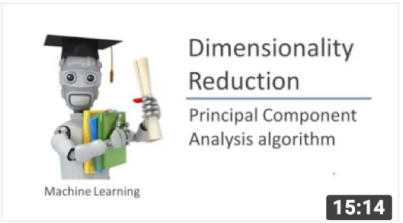
Deriving the new data set

Once you have chosen the principal components (eigenvectors) that you want to keep in the data and form a basis vectors, you can project data points onto the hyperplane (subspace) defined by the first k principal components.

Transform our data onto the new subspace via the equation:

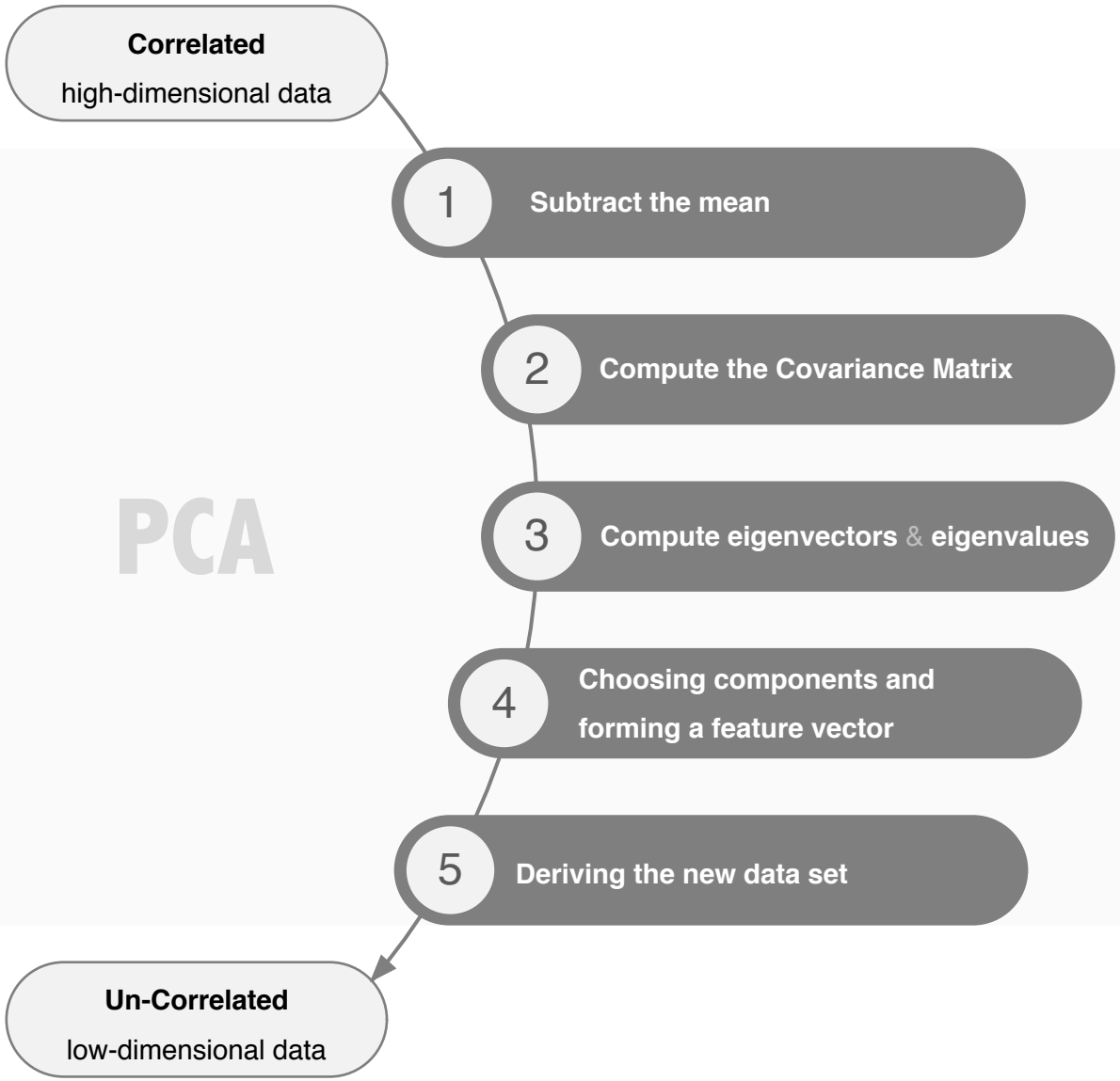
$$z = (U_{reduce})^T * x$$

- z is the FinalData (new, lower dimensionality feature vectors)
- x is the mean-adjusted data set
- U_{reduce} is a matrix of the top *k* eigenvectors.



Good Video (15 mintues)
https://www.youtube.com/watch?v=rng04VJxUt4

:: PCA



Summary

To summarize, principal component analysis involves evaluating the mean $\bar{x}$ and the covariance matrix **S** of the data set and then finding the *k* eigenvectors of **S** corresponding to the *k* largest eigenvalues[1].

Geometrical interpretation[2]

- PCA projects the data along the directions where the data varies the most.
- These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues.
- The magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions.

1Source: Pattern Recognition and Machine Learning, Page 563
2 Source: unknown
Reference: Lindsay I Smith, A tutorial on Principal Components Analysis

Choose parameter k for PCA

One important challenge in PCA is how to determine the number of components to retain.

How many principal components should be chosen ?

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n} \quad \Rightarrow \quad U_{reduce} = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}$$

K eigenvectors

Select the top k eigenvectors ($u_1, \dots, u_k$) to form a matrix of vectors.

But the question is how to determine the parameter k ?

How many principal components should be chosen ?

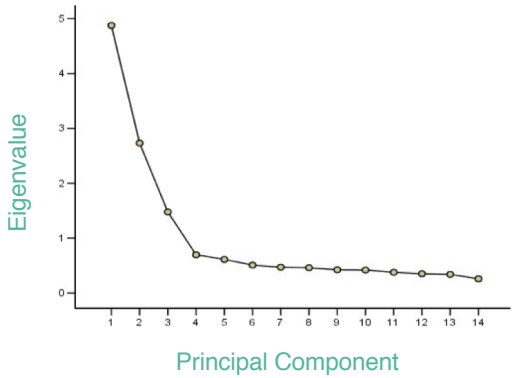
You can determine the number of principal components using several approaches. For example, Kaiser method, Scree test, Percentage of variation explained.

Scree test

Look for an “elbow” (a sharp change in the slopes of adjacent line segments) in the scree plot to determine an appropriate number of principal components.

The rule is to simply pick the number of components when your slope starts leveling off[1].

look for a “break” between the components with relatively large eigenvalues and those with small eigenvalues. The components that appearing after the break are assumed to be unimportant and are not retained.



However, scree test has its own weaknesses: very often, it’s difficult to determine exactly where in the scree plot a “elbow/break” exists

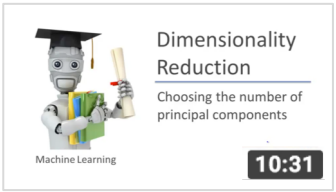
Percentage of Explained Variance

A useful measure is the so-called “explained variance,” which can be calculated from the eigenvalues. Retain components that cumulatively explain a certain percentage of variation. The acceptable level of explained variance depends on how you use Principal Components.

Use SVD , then pick smallest value of k for which:

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{m} S_{ii}} > Threshold \quad \text{(e.g., 0.9 or 0.95)}$$

- If the threshold is 0.99, then 99% variance is retained



Good Video (10 mintues)
https://www.youtube.com/watch?v=Ygekkjr98pg

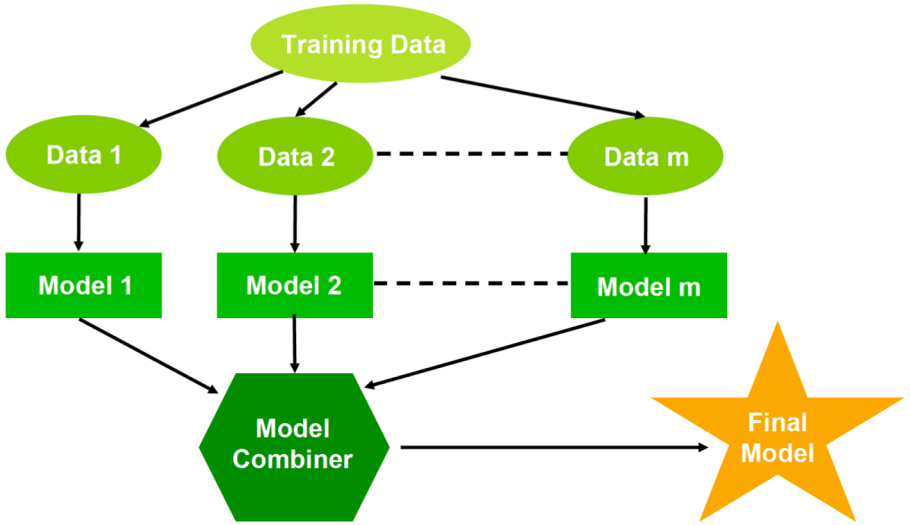
1 Source: https://www.quora.com/What-is-the-best-way-to-choose-the-number-of-components-in-PCA-during-dimensionality-reduction

AdaBoost

:: Ensemble Learning

About Ensemble Learning

Ensemble learning is a machine learning paradigm where multiple learners are trained and combined to solve the same problem. By using multiple learners, the generalization ability of an ensemble can be much better than that of a single learner[1].



Two heads are better than one

Suppose you ask a complex question to thousands of random people, then aggregate their answers. In many cases you will find that this aggregated answer is better than an expert's answer. This is called the wisdom of the crowd.

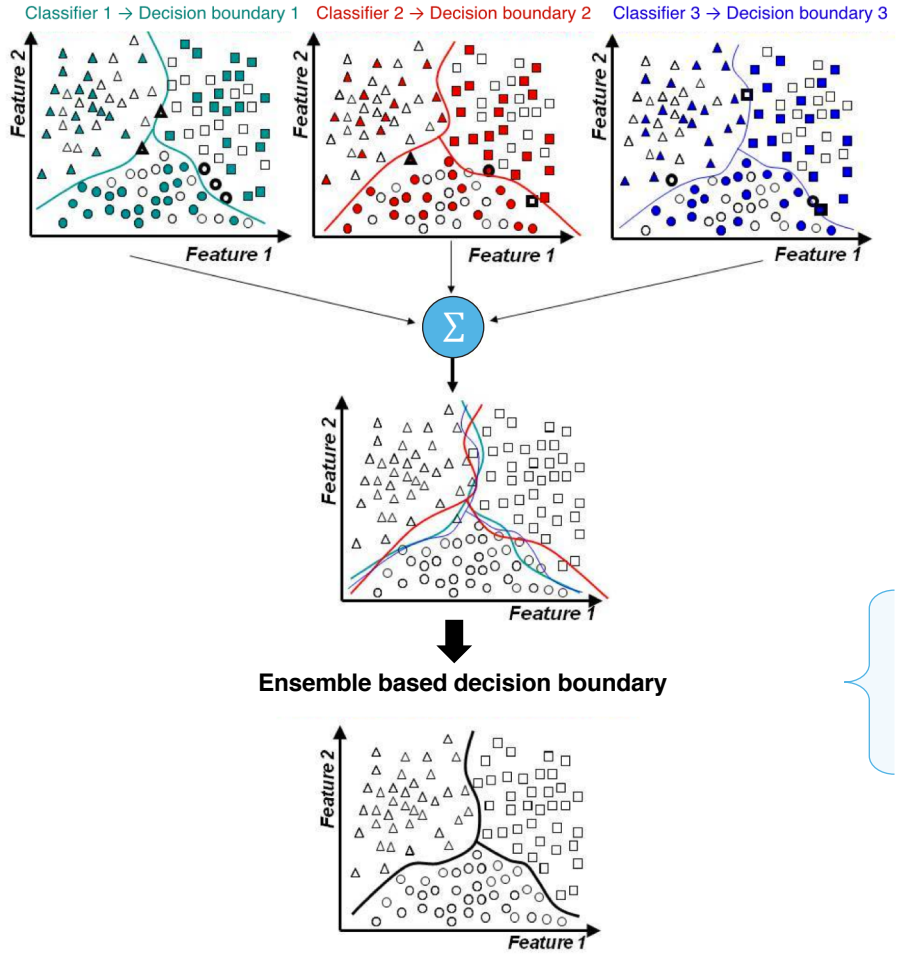
Similarly, if you aggregate the predictions of a group of predictors (such as classifiers or regressors), you will often get better predictions than with the best individual predictor. A group of predictors is called an ensemble; thus, this technique is called Ensemble Learning. [2]

1 Source : https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/springerEBR09.pdf
2 Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow
image Soure: https://open.sap.com/courses/ds1/items/4N4ywvRnKxx06RMhncvh6i

:: Ensemble Learning

Example of ensemble learning

Combining an ensemble of classifiers could reduce classification error and reduce the overall risk of making selection of a particularly poorly performing classifier



"… the diversity in the classifiers – typically achieved by using different training parameters for each classifier – allows individual classifiers to generate different decision boundaries. If proper diversity is achieved, a different error is made by each classifier, strategic combination of which can then reduce the total error." [2]

The figure graphically illustrates this concept, where each classifier - trained on a different subset of the available training data - makes different errors (shown as instances with dark borders), but the combination of the (three) classifiers provides the best decision boundary. [3]

2&3 Source: http://www.scholarpedia.org/article/Ensemble_learning

:: Ensemble Learning

Voting and averaging-based ensemble methods

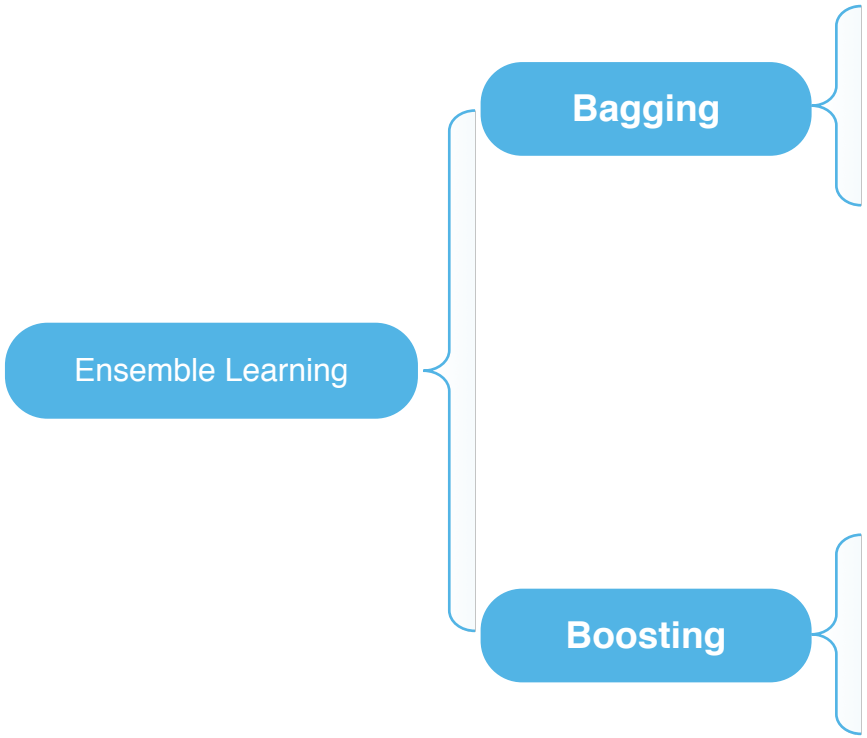
Actual (Target)						
Model 1						
Model 2						
Model 3						
Model 4						
Model 5						
Majority Vote						

example: Weather Forecast

- Voting and averaging are two of the easiest ensemble methods.
- Voting is used for classification and averaging is used for regression

:: Ensemble Learning

Common types of ensemble methods:



Bagging – e.g. Random Forest

The driving principle is to build several estimators/learners independently and then to average their predictions. On average, the combined learners is usually better than any of the single base learner because its variance is reduced [2].

Boosting – e.g. Gradient Boosting, AdaBoost

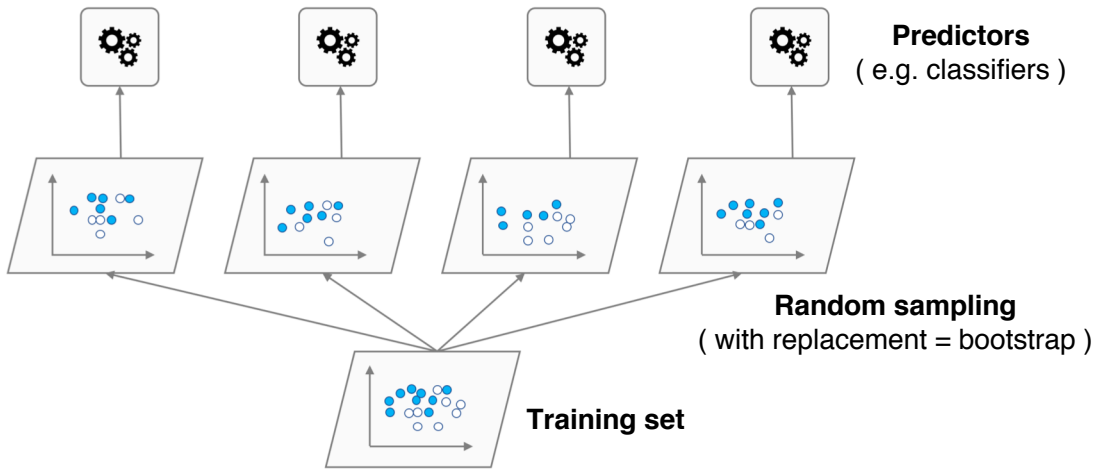
Base learners are built sequentially and one tries to reduce the bias of the combined learner. The motivation is to combine several weak models to produce a powerful ensemble[3].

2&3 Source: http://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting

:: Ensemble Learning

Bagging

Bagging (short for bootstrap aggregating) is a technique for reducing generalization error by combining several models. The idea is to train several different models separately, then have all of the models vote on the output for test examples. This is an example of a general strategy in machine learning called model averaging.[1]



- Bagging reduces **variance** by averaging model[4]
- Bagging has little effect on bias[5]

*One way to reduce the **variance** of an estimate is to average together multiple estimates. The reason that model averaging works is that different models will usually not make all the same errors on the test set.[2]*

- *One way to get a diverse set of classifiers is to use very different training algorithms, as just discussed.*
- *Another approach is to use the same training algorithm for every predictor, but to train them on different random subsets of the training set. When sampling is performed with replacement, this method is called **bagging** (short for bootstrap aggregating).[3]*

1,2 Source: Ian Goodfellow, et al. Deep Learning
3 & diagram Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow
4,5 source: David Sontag, Ensemble learning Lecture 13

:: Ensemble Learning

How Bagging works

Use bootstrapping to train multiple learners and then aggregate the predictions. Each individual predictor has a higher bias than if it were trained on the original training set, but aggregation reduces both bias and variance.

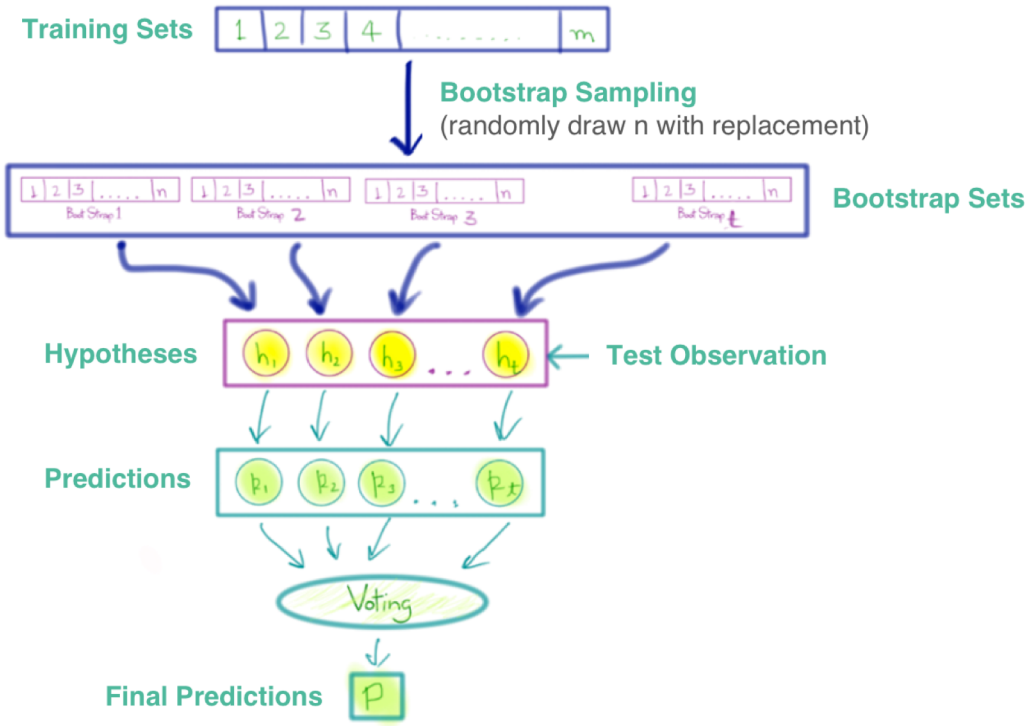
1. **Create random subsets of data by bootstrap sampling.**
   Different training data subsets are randomly drawn – with replacement – from the entire training dataset. By sampling with replacement some observations may be repeated in each new training data set.

2. **Train base learners on each bootstrap sample separately.**

3. **Average predictions from multiple base learners.**
   The final result is to aggregate the outputs of base learners by
   - Majority voting for classification
   - Averaging for regression

Notes:

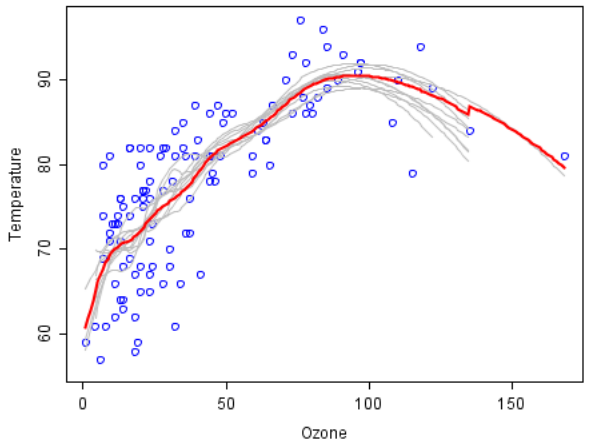
- Various learners/models are built in parallel.
- Bagging is more useful for nonlinear models
- Often used with tree – an extension is random forests.



Source of diagram: http://manish-m.com/?p=794

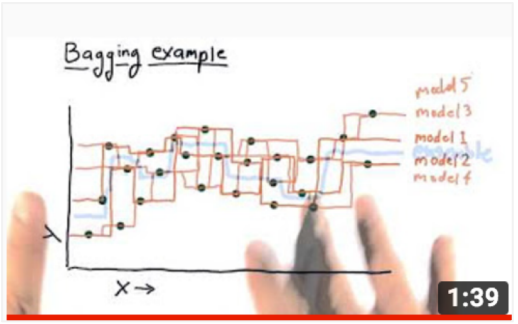
:: Ensemble Learning

Bagging example



An example of an ensemble of lines of best fit. Weak members are grey, the combined prediction is red.

YouTube Video



Bagging Example (1 min)
https://www.youtube.com/watch?v=sVriC_Ys2cw&pbjreload=10

Bootstrap Aggregating- bagging (3mins)
https://www.youtube.com/watch?v=2Mg8QD0F1dQ

Image: https://en.wikipedia.org/wiki/Bootstrap_aggregating

Boosting

Can a set of weak learners be combined to create a stronger learner ?

Yes ! Train the next learner based on the mistakes of the previous learners. After many iterations, the boosting algorithm combines these weak learner into a single strong prediction rule.

Boosting is a successful machine learning algorithm

Boosting is a popular machine learning method which has amazing impact.

- It's a simple approach
- Extremely useful in computer vision. For example, it's standard approach for face detection.
- Widely used in industry
- Used by most winners of machine learning competitions (Kaggle, KDD cup,…)[1]

There are a lot of boosting algorithms. For example,

- AdaBoost (Adaptive Boosting)
- Gradient Tree Boosting (GBRT)
- XGBoost
- …

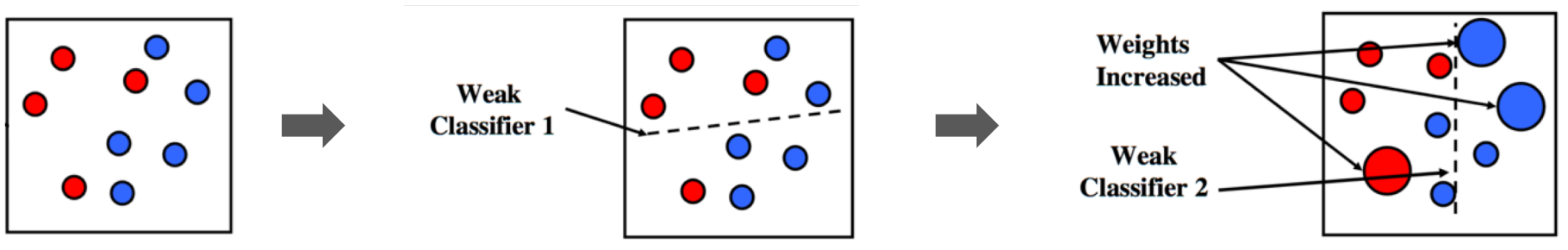
1 Sourec: https://www.coursera.org/learn/ml-classification

:: Ensemble Learning

How boosting works

- **Generate a sequence of base-learners.**

  Each learner is dependent on the previous one, and focus on previous one's error. Examples that are incorrectly predicted in previous learners are chosen more often or weighted more heavily.



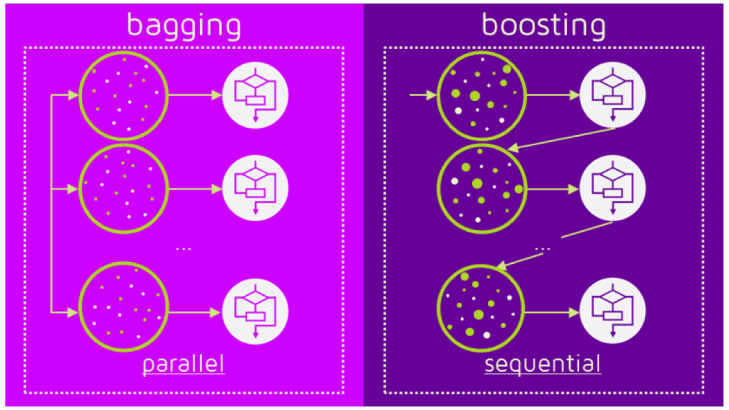
Each boosting round a new (simple) learner (e.g. classifier) is built on the weighted dataset.

- **Combine the base learners into a single powerful learner**

  Each base learner is "weak" but the final ensemble is "strong"

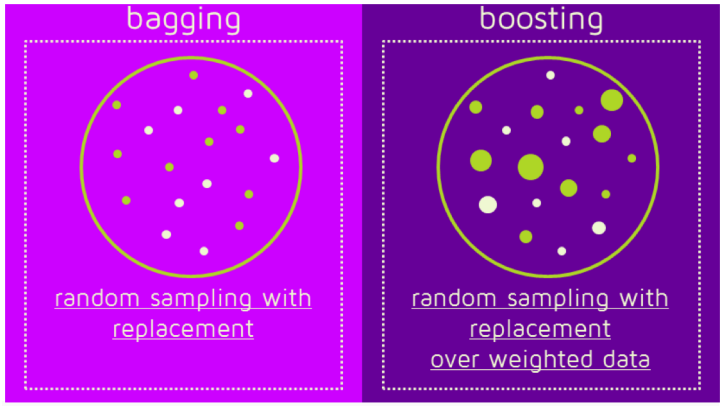
:: Ensemble Learning

The key differences between Bagging and Boosting



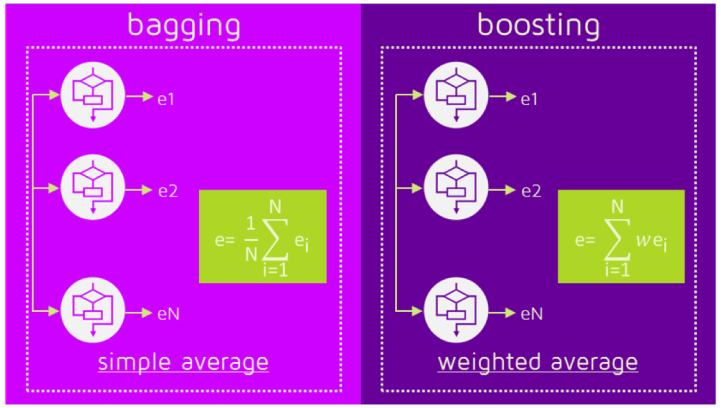
- **Boosting**
  Sequential ensemble methods where the base learners are generated sequentially (e.g. AdaBoost). Each learner is built on top of the previous one.

- **Bagging**
  Parallel ensemble methods where the base learners are generated in parallel.



- **Bagging**
  N new training data sets are produced by random sampling with replacement from the original set. By sampling with replacement some observations may be repeated in each new training data set.

- **Boosting**
  incorrectly predicted observations are weighted and therefore some of them will take part in the new sets more often



- **Bagging**
  The result is obtained by averaging the responses of the N learners (or majority vote).

- **Boosting**
  The final boosting ensemble uses a weighted average, more weight to those with better performance on training data.

Source: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

:: AdaBoost

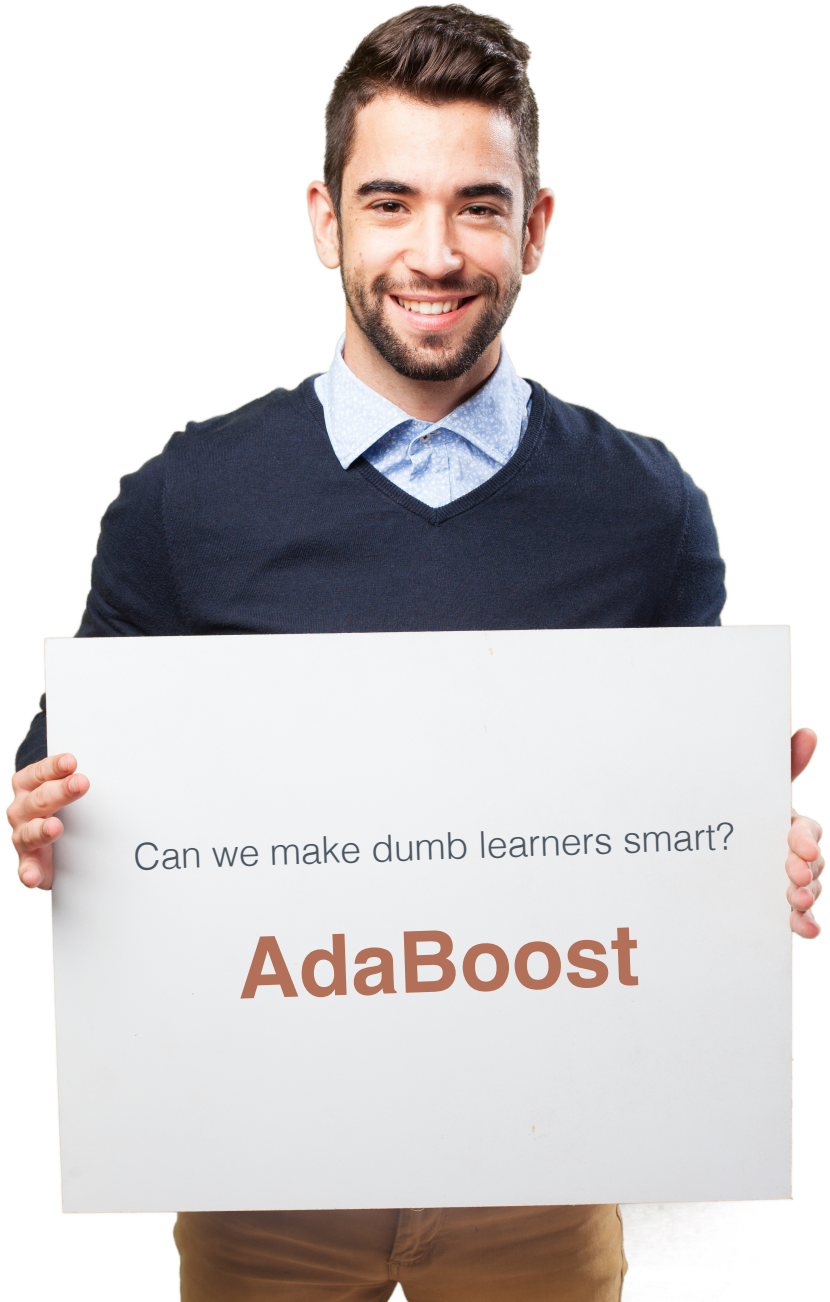


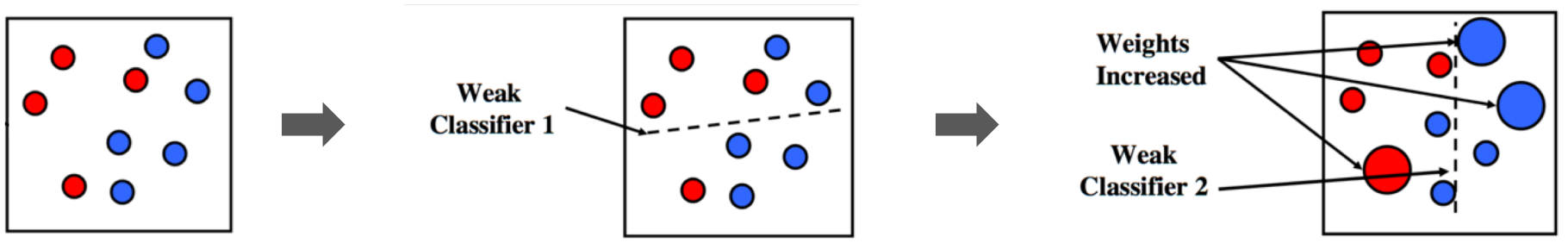
Image: Designed by Asierromero / Freepik

:: AdaBoost

Let's recall how boosting works

- **Generate a sequence of base-learners.**

  Each learner is dependent on the previous one, and focus on previous one's error. Examples that are incorrectly predicted in previous learners are chosen more often or weighted more heavily.



Each boosting round a new (simple) learner (e.g. classifier) is built on the weighted dataset.

- **Combine the base learners into a single powerful learner**

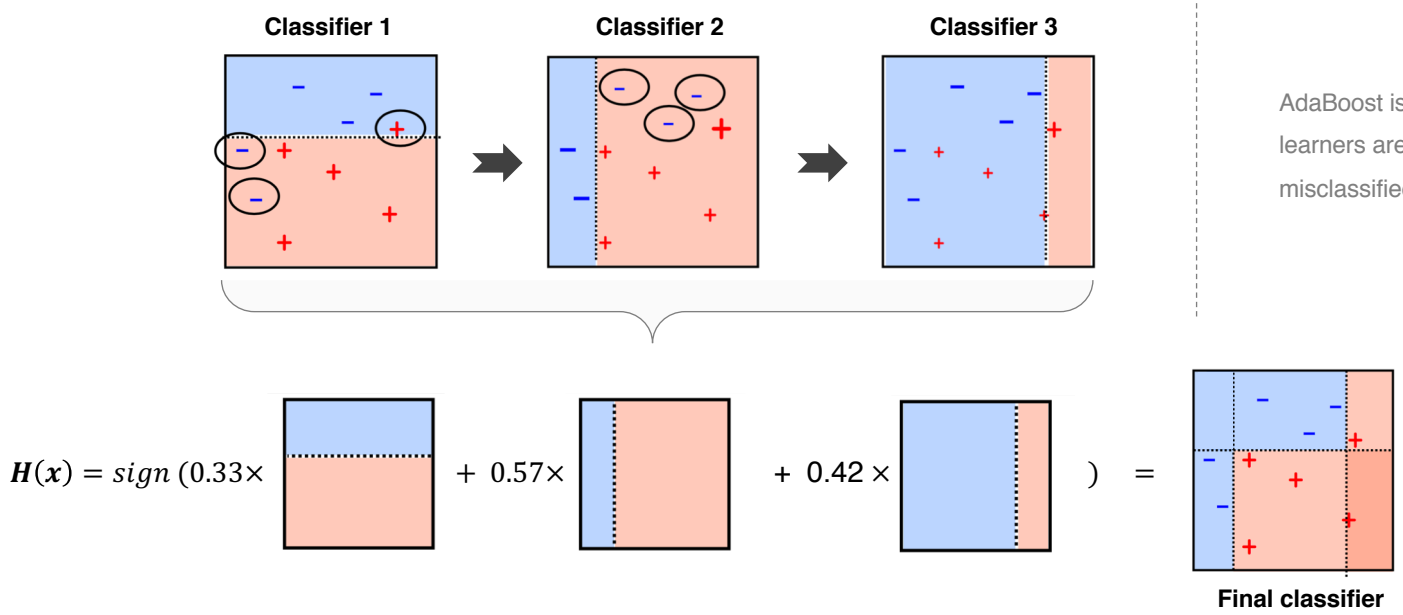
  Each base learner is "weak" but the final ensemble is "strong"

:: AdaBoost

AdaBoost aims to convert a set of weak classifiers into a strong one

AdaBoost, short for “Adaptive Boosting”, is the first practical boosting algorithm proposed by Freund and Schapire in 1996. It is the best starting point for understanding boosting. There are many boosting methods available, but by far the most popular are AdaBoost and Gradient Boosting.

AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. AdaBoost most commonly uses very simple decision stump (i.e. decision trees with one level) as the weak learners.



AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers.

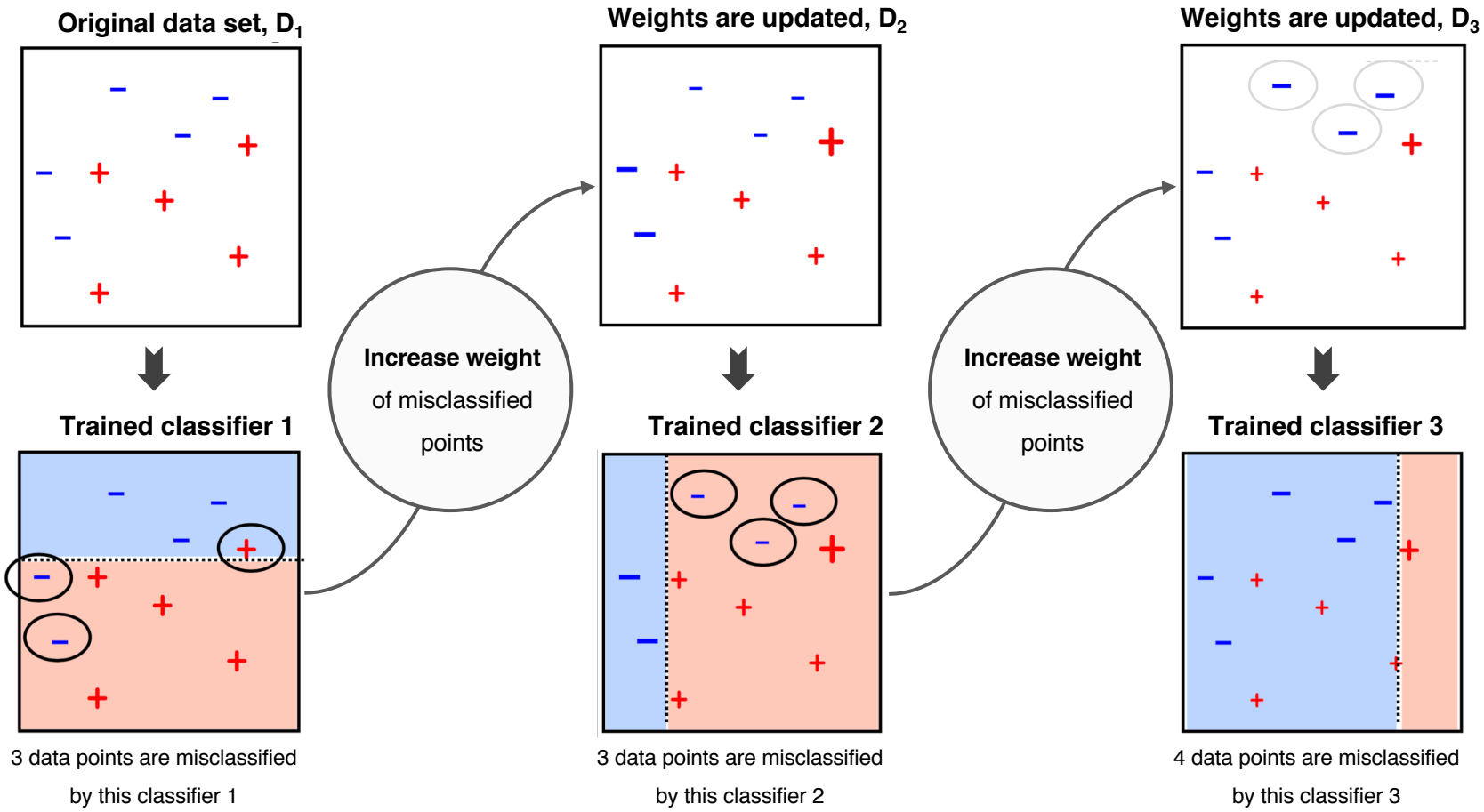
1 https://en.wikipedia.org/wiki/AdaBoost

:: AdaBoost

1. Train predictors (classifiers) sequentially, each trying to correct its predecessor

Initially, all N data points are assigned **equal weights**.

Then we apply very simple "decision stump" to classify them.

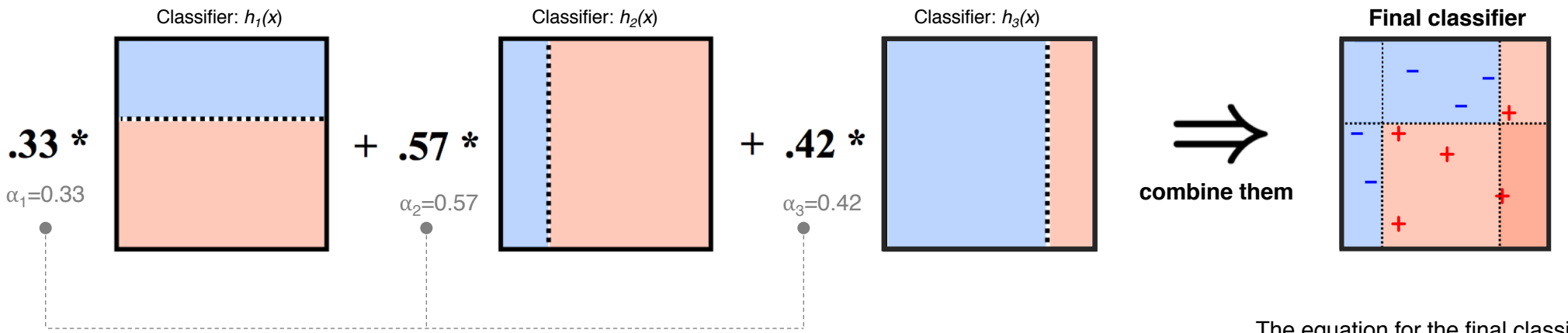


After the weights are increased, the size of three incorrectly predicted is bigger as compared to rest of the data points

- The weights of misclassified training examples are **increased** on each round in order to emphasize the most difficult cases,
- The weights **are decreased** for those that were predicted correctly. Subsequent learners will focus on these mistakes during their training.

:: AdaBoost

2. Combines the several weak classifiers into one strong learner



α_t : the weight for each classifier

After it's trained, we compute the output weight (alpha) for that classifier. After each classifier is trained, the classifier's weight is calculated based on its accuracy. More accurate classifiers are given more weight[1]. see next slide for more details.

$$\alpha_t = \frac{1}{2} ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

ϵ_t : It's based on the classifier's error rate. It is the weighted sum error for misclassified points.

ϵ_t > 0.5 means classifier is not better than random guessing

The equation for the final classifier

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

- $h_t(x)$ is the output of weak classifier 't'.
- α_t is the weight applied to classifier 't' as determined by AdaBoost

So the final output is just a linear combination of all of the weak classifiers $h_t(x)$, and then we make our final decision simply by looking at the sign of this sum[2].

The image is a modified version of diagram from https://www.youtube.com/watch?v=ix6IvwbVpw0&t=387s
1,2 Source: http://mccormickml.com/2013/12/13/adaboost-tutorial/

:: AdaBoost

AdaBoost Algorithm

1 **Initialize weight for each data points by setting** $w_i = \frac{1}{N}, \quad i = 1, 2, \ldots, N.$

2 **For iteration** $t = 1, 2, \ldots, T$:

(a) Fit a classifier $h_t(x)$ to the training data using weight w_i.

(b) Compute the weighted error rate of the t^{th} predictor:

$$err_t = \frac{\sum_{i=1}^{N} w_i \, I\,(yi \neq h_t(x_i))}{\sum_{i=1}^{N} w_i}$$

where $I\,(yi \neq \boldsymbol{h_t(x_i)})$ is the indicator function and equals 1 when $y_i \neq \boldsymbol{h_t(x_i)}$ and 0 otherwise. [1]

(c) Compute the predictor's weight α_t

$$\alpha_t = \frac{1}{2} ln\,(\frac{1 - err_t}{err_t})$$

$\boldsymbol{\alpha_t}$ **- a strength for this hypothesis** $\boldsymbol{h_t(x)}$

Here $\alpha_1, \alpha_2, \ldots, \alpha_t$ are computed by the boosting algorithm, and weight the contribution of each respective $h_t(x)$. The more accurate the predictor is, the higher its weight will be. If it is just guessing randomly, then its weight will be close to zero. However, if it is most often wrong (i.e., less accurate than random guessing), then its weight will be negative.[2]

(d) Update the weights on the training examples

$$w_i^{t+1} = w_i^t \cdot exp\,\{\alpha_t \, I\,(yi \neq h_t(x_i)\}$$

The individual weights of each of the observations(data points) are updated for the next iteration. Observations misclassified by $h_t(x)$ have their weights scaled by a factor $exp\,(\alpha_t)$, increasing their relative influence for inducing the next classifier $h_{t+1}(x)$ in the sequence.[3]

3 **Make predictions using the final model**, which is given by

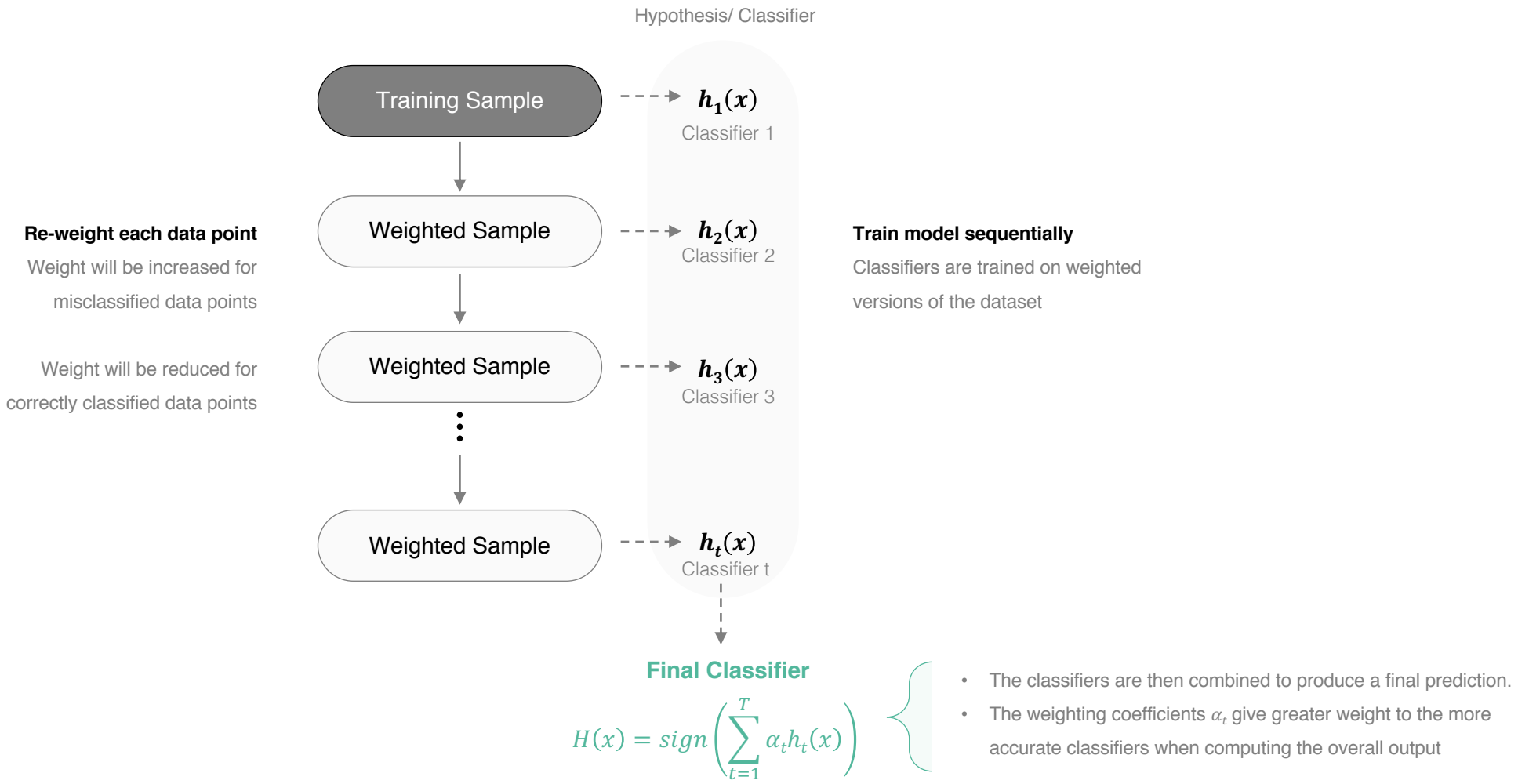
$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

The predictions from all of predictors are then combined through a weighted majority vote to produce the final prediction[4]

1 Christopher M. Bishop Pattern Recognition and Machine Learning
2 Hands-On Machine Learning with Scikit-Learn and TensorFlow
3,4 The Elements of Statistical Learning

AdaBoost

Schematic illustration of AdaBoost



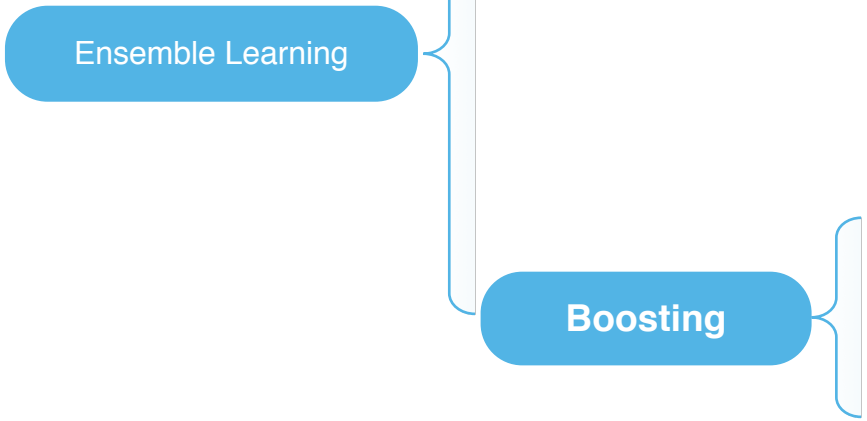
Gradient Boosting

To be added later

Random Forest

:: Ensemble Learning

Common types of ensemble methods:



Bagging – e.g. Random Forest

The driving principle is to build several estimators/learners independently and then to average their predictions. On average, the combined learners is usually better than any of the single base learner because its variance is reduced [2].

Boosting – e.g. Gradient Boosting

Base learners are built sequentially and one tries to reduce the bias of the combined learner. The motivation is to combine several weak models to produce a powerful ensemble[3].

2&3 Source: http://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting

:: Ensemble Learning

How Bagging works

Use bootstrapping to train multiple learners and then aggregate the predictions. Each individual predictor has a higher bias than if it were trained on the original training set, but aggregation reduces both bias and variance.

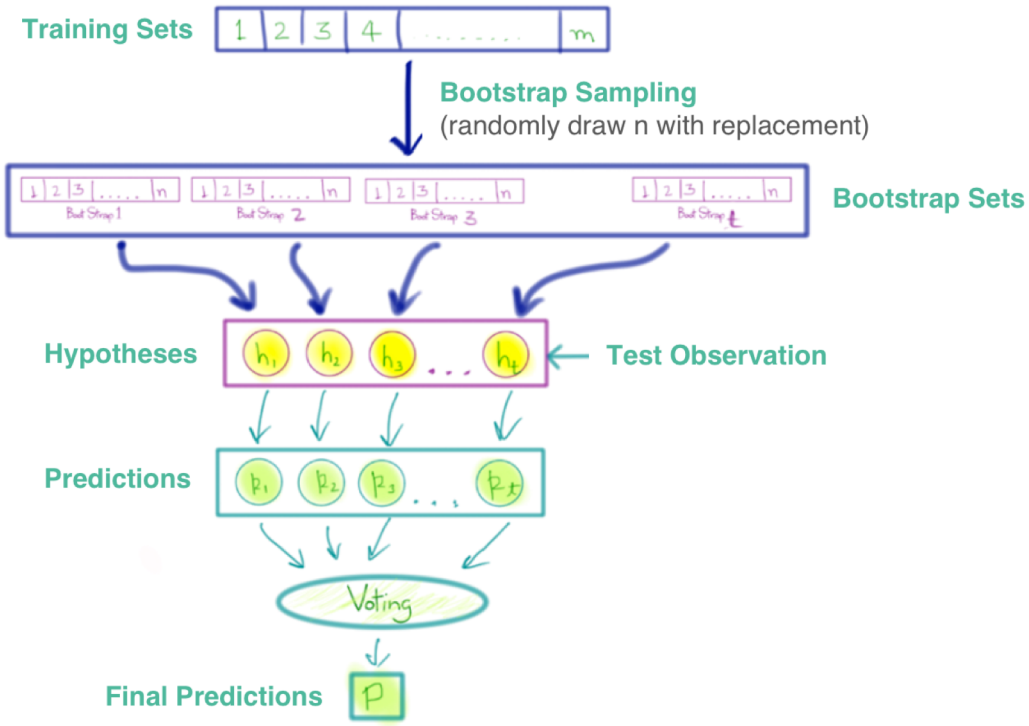
1. **Create random subsets of data by bootstrap sampling.**
   Different training data subsets are randomly drawn – with replacement – from the entire training dataset. By sampling with replacement some observations may be repeated in each new training data set.

2. **Train base learners on each bootstrap sample separately.**

3. **Average predictions from multiple base learners.**
   The final result is to aggregate the outputs of base learners by
   - Majority voting for classification
   - Averaging for regression

Notes:

- Various learners/models are built in parallel.
- Bagging is more useful for nonlinear models
- Often used with tree – an extension is random forests.



Source of diagram: http://manish-m.com/?p=794

:: Ensemble Learning

In a nut shell

(***bagging***)

The essential idea in bagging is to average many noisy but approximately unbiased models, and hence reduce the variance.[1]

1 soure: The Elements of Statistical Learning

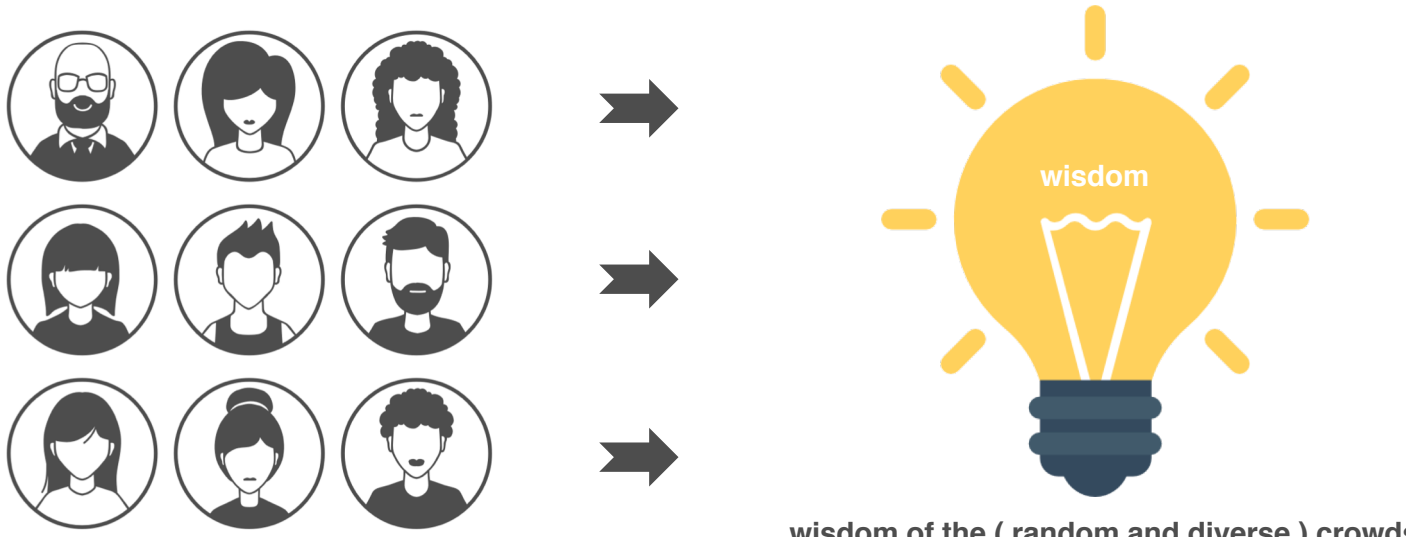
:: Random Forest

Image: Designed by Asierromero / Freepik

:: Random Forest

Random Forest is one of the most used algorithms

Random Forest is an ensemble of Decision Trees, generally trained via the bagging method[1]. On many problems the performance of random forests is very similar to boosting, and they are simpler to train and tune. As a consequence, random forests are popular, and are implemented in a variety of packages.[2]



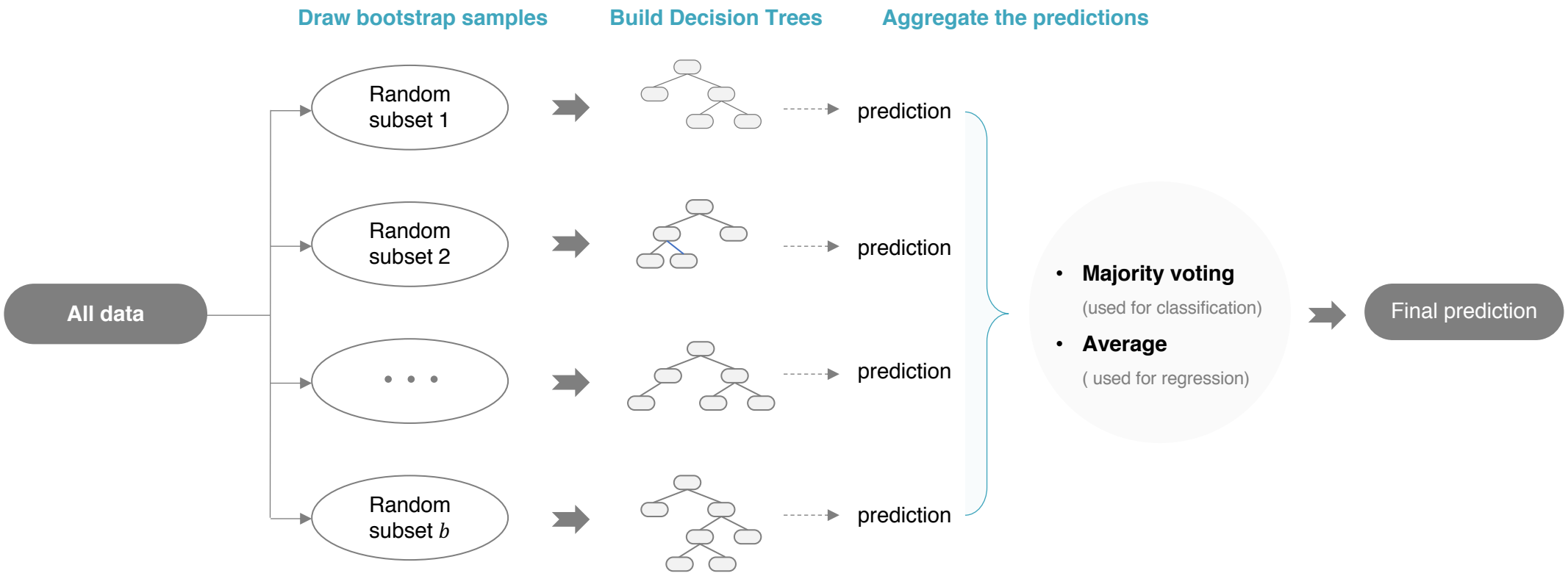
wisdom of the (random and diverse) crowds

:Image: Designed by Freepik.com
1 source: Hands-On Machine Learning with Scikit-Learn and TensorFlow
2 soure: The Elements of Statistical Learning

:: Random Forest

Random Forest = Bagging + Full-grown CART decision Tree

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. [1] It can be used for both classification and regression problems.

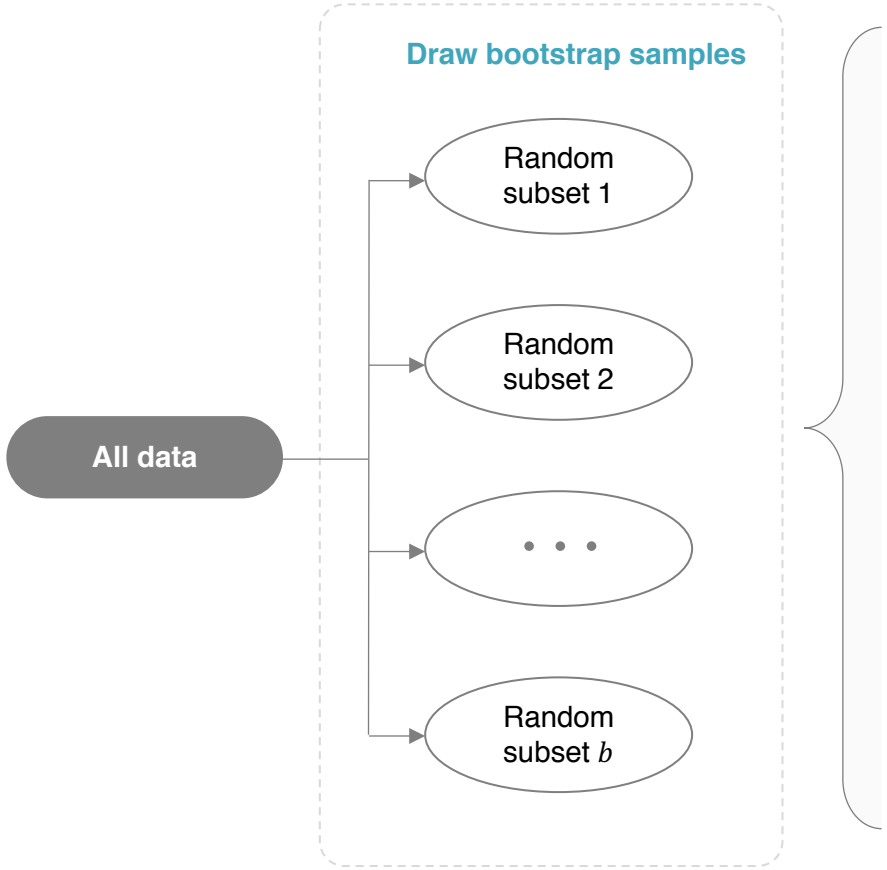


The Random Forest algorithm randomly selects **observations** and **features** to build several decision trees and then averages the results.

:: Random Forest

Step1 : Draw bootstrap samples

Use the bootstrap method to produce training subsets. Around 2/3 of all data will be drawn into the random subsets, the rest of data will be in “out of the bag”.



Randomly select samples by <u>sampling with replacement</u>

Use bootstrap sampling to create N subsets of the data !

- These sampled subset should be about 2/3 of the total data. They will be the training set for growing the tree.
- The remaining data are never drawn. These data are called “out of the bag” samples. Later they will be used in evaluation.

With bootstrap sampling (i.e. sampling with replacement), some instances may be sampled several times for any given predictor, while others may not be sampled at all[1]. After N draws with replacement, the probability of never have been selected is about 37 %.

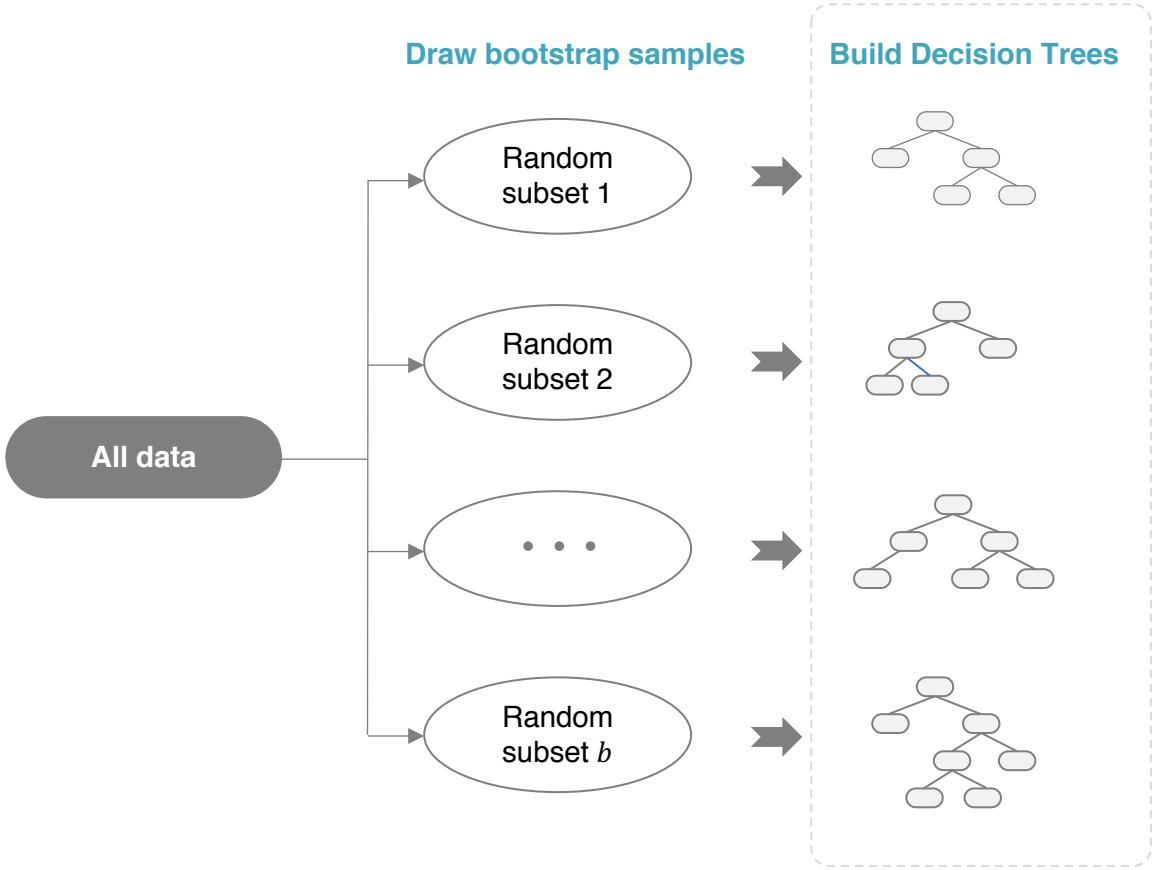
Since a predictor never sees the out-of-bag instances during training, it can be evaluated on these instances, without the need for a separate validation set or cross-validation.[2] Therefore, the traditional approach to reserve additional data for validation is no longer necessary

1 source: Hands-On Machine Learning with Scikit-Learn and TensorFlow

:: Random Forest

Step2 : Generate multiple decision trees based on bootstrap samples

Random forest builds each decision trees using the bootstrapped dataset, but only use a random subset of variables/features. This randomness increases diversity in the forest leading to more robust overall predictions.



Fit each random-forest tree T to different bootstrap samples

- Grow a decision tree, by recursively repeating the following steps for each terminal node of the tree
  - Select m variables at random from the p variables

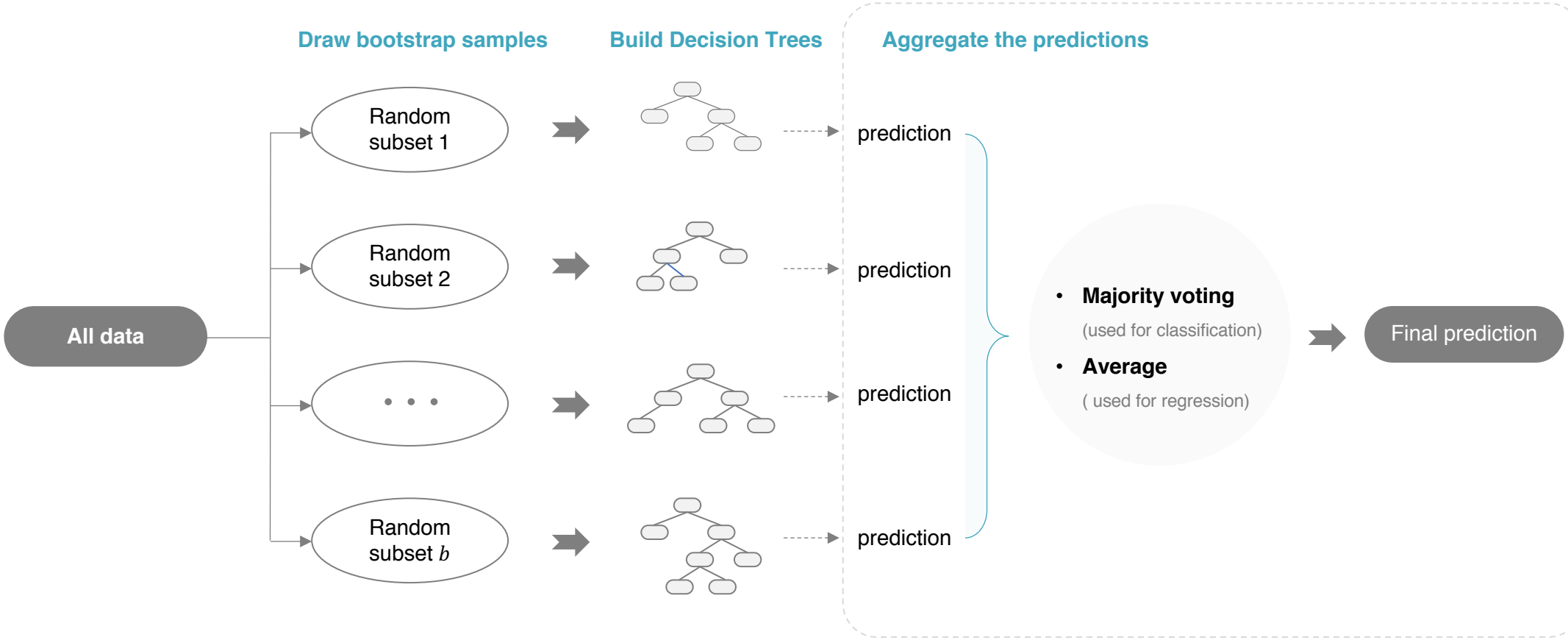
    If there are p input variables, a number $m < p$ is specified such that at each node, m variables are selected at random out of the p. The optimal m is usually around $\sqrt{p}$, but this can be used as a tuning parameter[1]
  - Calculate the best split based on these m variables in the training set. And split the node into two daughter nodes.
- Each tree is grown to the largest extent possible and there is no pruning.
- This will lead to very different (or "uncorrelated") trees from each sample.[2]

Source: Stanford, Lecture 20: Bagging, Random Forests, Boosting

:: Random Forest

Step3 : Predict new data by aggregating the predictions of the trees

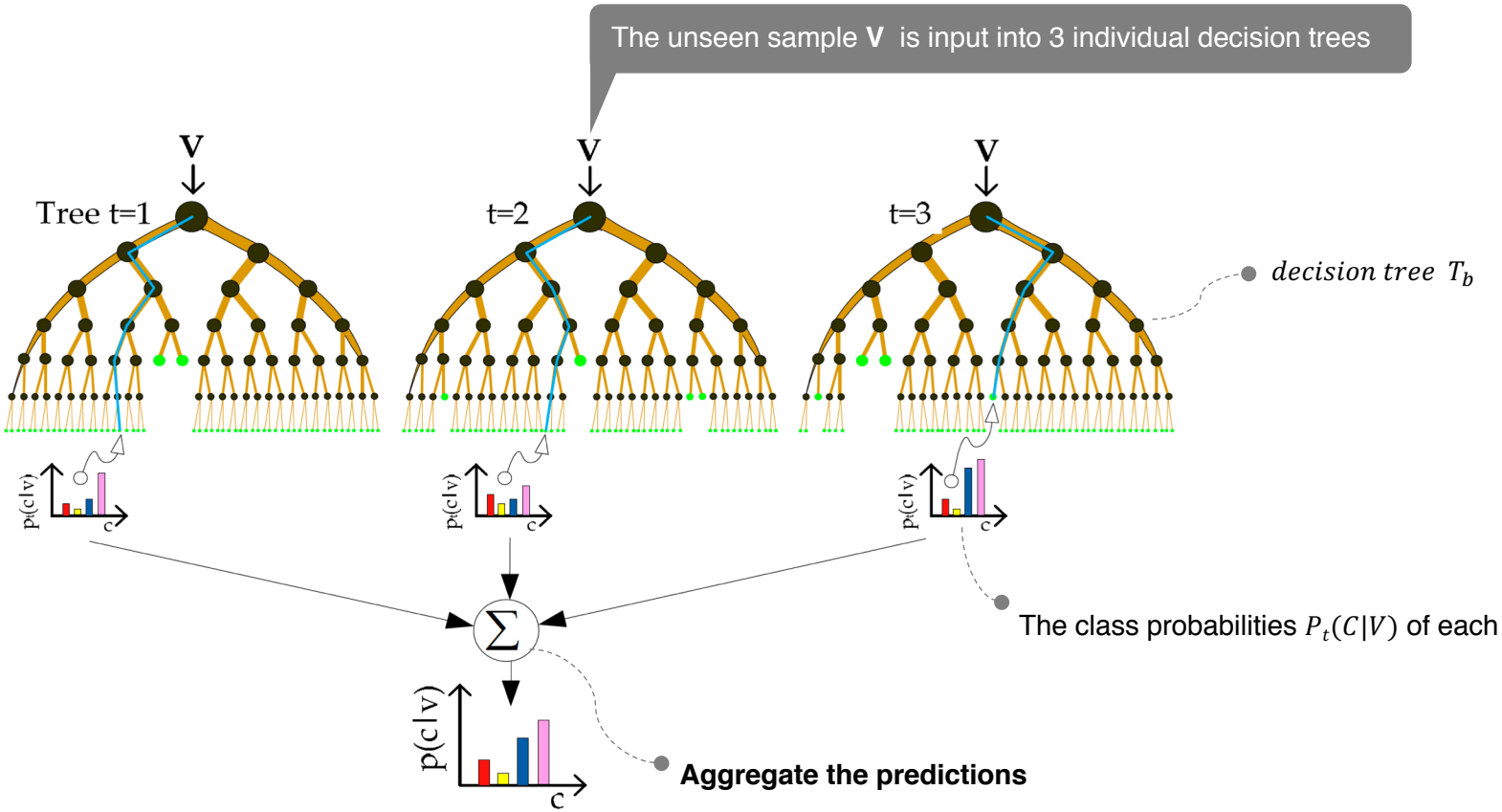
Use “majority votes” for classification, and “average” for regression problem



:: Random Forest

Classification example: use Random Forest to classify data

After training, a tree set $\{T\}$ can be obtained to predict the classes of the unseen samples by taking the majority vote from all individual classification trees.[1]



The class probabilities $P_t(C|V)$ of each tree are first computed

Aggregate the predictions

The final classification result $P\ (C|V)$ is obtained by averaging all 3 probability distributions

1 & image:Source: High-Resolution Remote Sensing Data Classification over Urban Areas Using Random Forest Ensemble and Fully Connected Conditional Random Field

:: Random Forest

Algorithm: Random Forest for Regression or Classification [1]

1. For b = 1 to B :

   (a) Draw a bootstrap sample $z^*$of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

   - i. Select m variables at random from the p variables.
   - ii. Pick the best variable/split-point among the m .
   - iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{\boldsymbol{T_b}\}_1^{\boldsymbol{B}}$

Note: [2]

- Beginners often assume that we select a random subset of predictors once at the start of the analysis and then grow the whole tree using this subset
- This is not how Random Forests work
- In Random Forests we select a new random subset of predictors in each node of a tree

To make a prediction at a new point x :

- Regression: $\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B}\boldsymbol{T_b}(x)$
- Classification: Let $\hat{C}_b(x)$be the class prediction of the b^{th} random-forest tree. Then

  $\hat{C}_{rf}^{B}(x) = majority\ vote\ \{\hat{C}_b(x)\}_1^B$

1 source of algorimthm: The Elements of Statistical Learning
2 Source: Salford Systems 2014, RANDOM FORESTS FOR BEGINNERS

:: Random Forest

Advantages of Using Random Forest

Overall, Random Forest is a fast, simple and flexible algorithm, although it has its limitations.

- **Random Forest is less likely to overfit**

  Unlike decision tree machine learning algorithms, Overfitting is less of an issue with Random Forests. No need to pruning trees

- **Random Forest algorithms can be grown in parallel.**

  Individual decision trees can be trained in parallel

- **Random Forest has higher classification accuracy**

  It's difficult to build a "bad" random forest because of its simplicity. One can easily build a decent model without much tuning.

- **Able to deal with the missing value and maintain accuracy in case of missing data**

  Random forest can handle the missing value. And it is also not very sensitive to outliers

- **Help data scientists save data preparation time**

  It does not requires the input preparation. It can handle thousands of input variables without variable deletion.[1]

Picture : Designed by Asierromero / Freepik
[1] source: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm