Lecture 1
Introduction to Machine Learning for Biomedical Data Science

Mario Guarracino
University of Cassino and Southern Lazio

Version 2 2020 10 19 10:00

Data Analytics Winter School
Online, 20-22/11/2020

Aims

- These two lectures aim at promoting critical understanding of how machine learning works and how it can be used to solve different biomedical problems.
- This knowledge will help in:
  - creating new data-driven analytic strategies,
  - participate in conversations about machine learning and data analytics,
  - open a career path,
  - or improve students' understanding.

Jobs in data science

- Amazon: 9000 open positions in data science
  - https://www.amazon.jobs/it/search?base_query=Data+Science
- Glassdoor: 24000 positions:
  - https://www.glassdoor.it/Lavoro/data-scientist-lavori-SRCH_KO0,14.htm?countryRedirect=true
- Indeed 11000 positions:
  - https://www.indeed.com/q-data-scientist-jobs.html
- UK positions:
  - https://www.datascientistjobs.co.uk

What is machine learning?

- Machine learning refers to a vast set of tools for understanding data.
- Instead of writing a program to solve a task, we train a computer for this purpose.
- In training phase we can either use some prior knowledge about the problem or not.
- Therefore ML methods are broadly classified as supervised or unsupervised.

What type of problems can we solve with ML?

- Train a computer to answer questions
  - Supervised classification & regression
- Find patterns in data
  - Unsupervised learning (clustering)
- Determine important variables
  - Feature reduction / selection
- Outlier detection
  - Univariate / multivariate
- Behaviour detection
  - Association rules
- Concept drifts, causality relations, graph mining,...

Applications in biomedicine

- Medical imaging, signal processing and text analysis.
- Clinical expert systems for diagnosis and prognosis
- Interpreting genomic or metagenomic data
- Discovering regulatory or expression pathways
- Genome–disease & GWAS
- Modeling ecosystems or population dynamics
- Network biology/medicine
- Data integration,...

Supervised vs unsupervised

- Supervised learning builds a mathematical/statistical model to estimate or predict an output, given one or more inputs.
  - Any problem that can be posed in form of a question (with known possible answers) fits here!
- With unsupervised learning we characterize a phenomena, extracting interesting patterns.

Protein interactions

Predicting Protein-Protein Interactions with K-Nearest Neighbors Classification Algorithm

Mario R. Guarracino and Adriano Nebbia

High Performance Computing and Networking Institute (ICAR-CNR)
National Research Council, Italy
Via P. Castellino, 111 - 80131 Napoli (IT)
mario.guarracino@cnr.it, a_nebbia@hotmail.it

Abstract. In this work we address the problem of predicting protein-protein interactions. Its solution can give greater insight in the study of complex diseases, like cancer, and provides valuable information in the study of active small molecules for new drugs, limiting the number of molecules to be tested in laboratory. We model the problem as a binary classification task, using a suitable coding of the amino acid sequences. We apply *k*-Nearest Neighbors classification algorithm to the classes of interacting and noninteracting proteins. Results show that it is possible to achieve high prediction accuracy in cross validation. A case study is analyzed to show it is possible to reconstruct a real network of thousands interacting proteins with high accuracy on standard hardware.

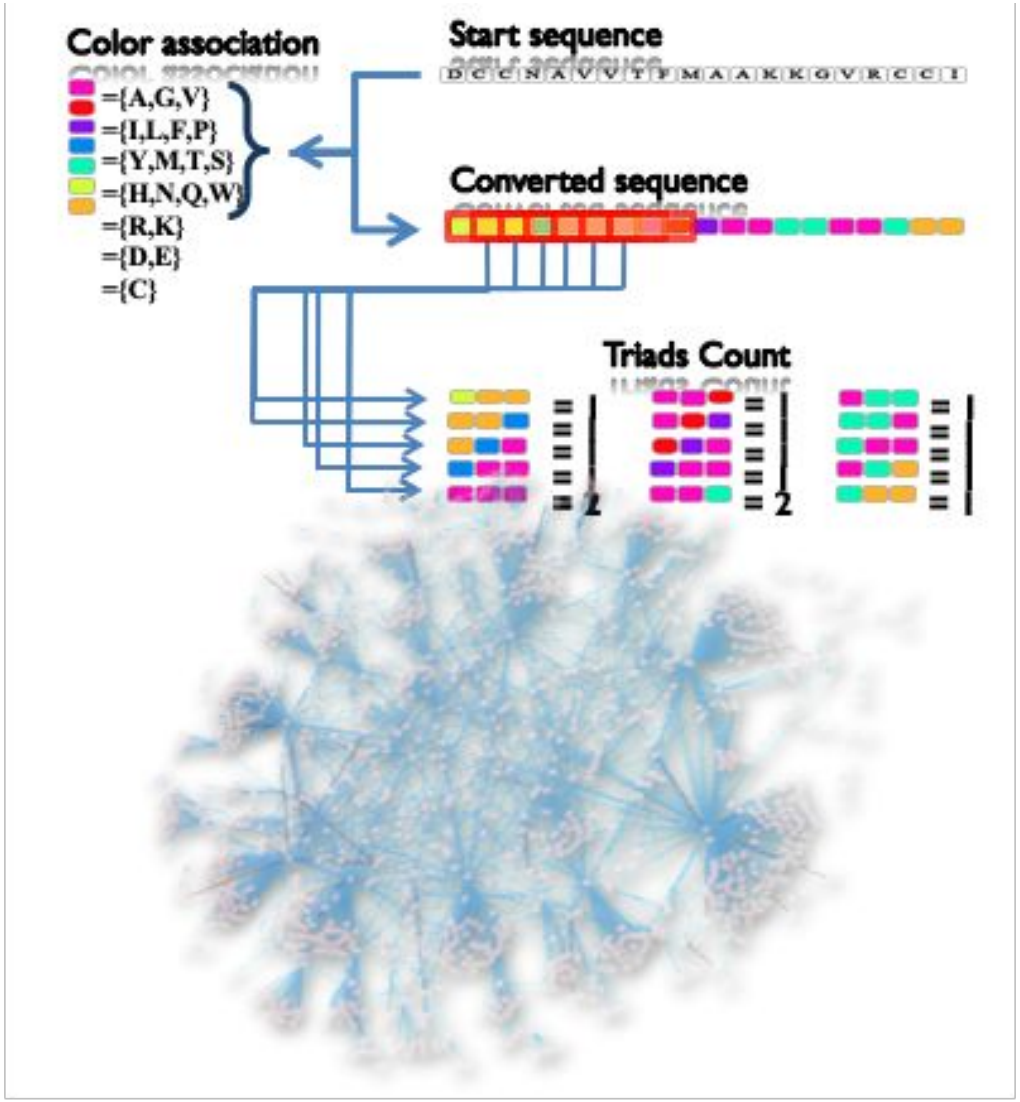
Keywords: Protein-protein interaction prediction, conjoint-triad method, *k*-Nearest Neighbors, binary classification.

1 Introduction

Proteins are the main components of living organisms and take part to every process within a cell. They are composed of amino acids arranged in a linear sequence of variable length. In general, protein sequences are composed of 20 different amino acids, except for some organisms having two more amino acids. The sequence of amino acids is defined by the nucleotide sequence of a gene and it is determined by its genetic code. The longest known human proteins are the titins, whose length is around 27,000 amino acids.

Protein−protein interactions (PPIs) take place in many biological processes and many diseases. The study of PPIs starts much earlier than the advent of proteomics. A notable example is the work of Ruhlmann and colleagues in the '70 and Amit and colleagues in '80 on antigenantibody and protease inhibitor complexes that has given insight into protein interfaces and their properties. At that time, although it was not clear to which extend and degree proteins would interact, it was clear that many biological processes relay on their interaction, and the protein functions may be regulated by other proteins. Therefore, the capability to predict interactions could be exploited to block these interactions with a drug. Such capability would give the chance to direct research towards highly probable predicted targets.

F. Masulli, L. Peterson, and R. Tagliaferri (Eds.): CIBB 2009, LNBI 6160, pp. 139–150, 2010.



Prediction of drug activity

Andreotti et al. Orphanet Journal of Rare Diseases 2010, 5:36
http://www.ojrd.com/content/5/1/36

ORPHANET JOURNAL OF RARE DISEASES

METHODOLOGY Open Access

Prediction of the responsiveness to pharmacological chaperones: lysosomal human alpha-galactosidase, a case of study

Giuseppina Andreotti[1], Mario R Guarracino[2], Marco Cammisa[3,4], Antonella Correra[3,4], Maria Vittoria Cubellis[3,4*]

Abstract

Background: The pharmacological chaperones therapy is a promising approach to cure genetic diseases. It relies on substrate competitors used at sub-inhibitory concentration which can be administered orally, reach difficult tissues and have low cost. Clinical trials are currently carried out for Fabry disease, a lysosomal storage disorder caused by inherited genetic mutations of alpha-galactosidase. Regrettably, not all genotypes respond to these drugs.

Results: We collected the experimental data available in literature on the enzymatic activity of ninety-six missense mutants of lysosomal alpha-galactosidase measured in the presence of pharmacological chaperones. We associated with each mutation seven features derived from the analysis of 3D-structure of the enzyme, two features associated with their thermo-dynamic stability and four features derived from sequence alone. Structural and thermodynamic analysis explains why some mutants of human lysosomal alpha-galactosidase cannot be rescued by pharmacological chaperones: approximately forty per cent of the non responsive cases examined can be correctly associated with a negative prognostic feature. They include mutations occurring in the active site pocket, mutations preventing disulphide bridge formation and severely destabilising mutations. Despite this finding, prediction of mutations responsive to pharmacological chaperones cannot be achieved with high accuracy relying on combinations of structure- and thermodynamic-derived features even with the aid of classical and state of the art statistical learning methods.

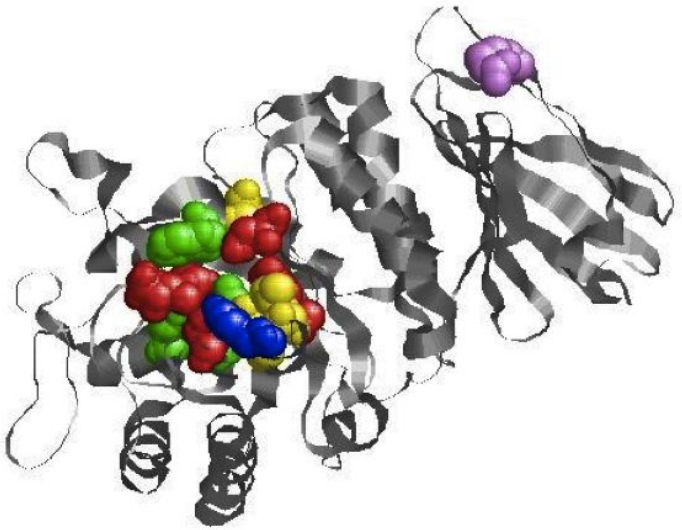
We developed a procedure to predict responsive mutations with an accuracy as high as 87%: the method scores the mutations by using a suitable position-specific substitution matrix. Our approach is of general applicability since it does not require the knowledge of 3D-structure but relies only on the sequence.

Conclusions: Responsiveness to pharmacological chaperones depends on the structural/functional features of the disease-associated protein, whose complex interplay is best reflected on sequence conservation by evolutionary pressure. We propose a predictive method which can be applied to screen novel mutations of alpha galactosidase. The same approach can be extended on a genomic scale to find candidates for therapy with pharmacological chaperones among proteins with unknown tertiary structures.

Background

Pharmacological chaperone (PC) therapy has been recently proposed as a promising strategy for the treatment of some genetic diseases. PC therapy exploits small molecules which can be administered orally, reach difficult tissues such as the brain and have low cost. The new approach relies on an unexpected finding: some molecules that at high dosage inhibit specific proteins, can, at low dosage, restore their activities in cells. They act as life jackets or chaperones for proteins that, although retaining the essential residues needed for activity, become unstable upon mutation and are degraded. These proteins are able to fulfill their duty if they are given the chance to survive long enough and get to the site where they are needed.

* Correspondence: cubellis@unina.it
[3]Dipartimento di Biologia Strutturale e Funzionale, Universita' Federico II, Napoli, Italy
Full list of author information is available at the end of the article



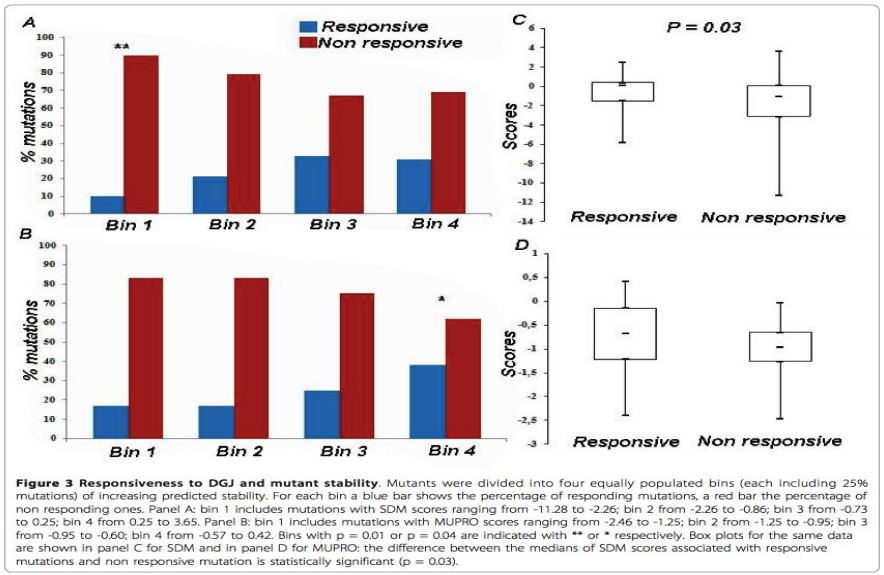


Figure 3 Responsiveness to DGJ and mutant stability. Mutants were divided into four equally populated bins (each including 25% mutations) of increasing predicted stability. For each bin a blue bar shows the percentage of responding mutations, a red bar the percentage of non responding ones. Panel A: bin 1 includes mutations with SDM scores ranging from -11.28 to -2.26; bin 2 from -2.26 to -0.86; bin 3 from -0.73 to 0.25; bin 4 from 0.25 to 3.65. Panel B: bin 1 includes mutations with MUPRO scores ranging from -2.46 to -1.25; bin 2 from -1.25 to -0.95; bin 3 from -0.95 to -0.60; bin 4 from -0.57 to 0.42. Bins with p = 0.01 or p = 0.04 are indicated with ** or * respectively. Box plots for the same data are shown in panel C for SDM and in panel D for MUPRO: the difference between the medians of SDM scores associated with responsive mutations and non responsive mutation is statistically significant (p = 0.03).

Classification of mRNAs

Contents lists available at ScienceDirect

Computational Toxicology

journal homepage: www.elsevier.com/locate/comtox

The sbv IMPROVER Systems Toxicology computational challenge: Identification of human and species-independent blood response markers as predictors of smoking exposure and cessation status

Vincenzo Belcastro [a,*,1], Carine Poussin [a,*,1], Yang Xiang [a], Maurizio Giordano [o], Kumar Parijat Tripathi [o], Akash Boda [a], Ali Tugrul Balci [l,3], Ismail Bilgen [l,3], Sandeep Kumar Dhanda [n,3], Zhongqu Duan [i,k,3], Xiaofeng Gong [i,k,3], Rahul Kumar [m,3], Roberto Romero [d,e,f,g,h,3], Omer Sinan Sarac [l,3], Adi L. Tarca [b,c,3], Peixuan Wang [i,k,3], Hao Yang [i,j,3], Wenxin Yang [i,j,3], Chenfang Zhang [i,k,3], Stéphanie Boué [a], Mario Rosario Guarracino [o], Florian Martin [a], Manuel C. Peitsch [a], Julia Hoeng [a]

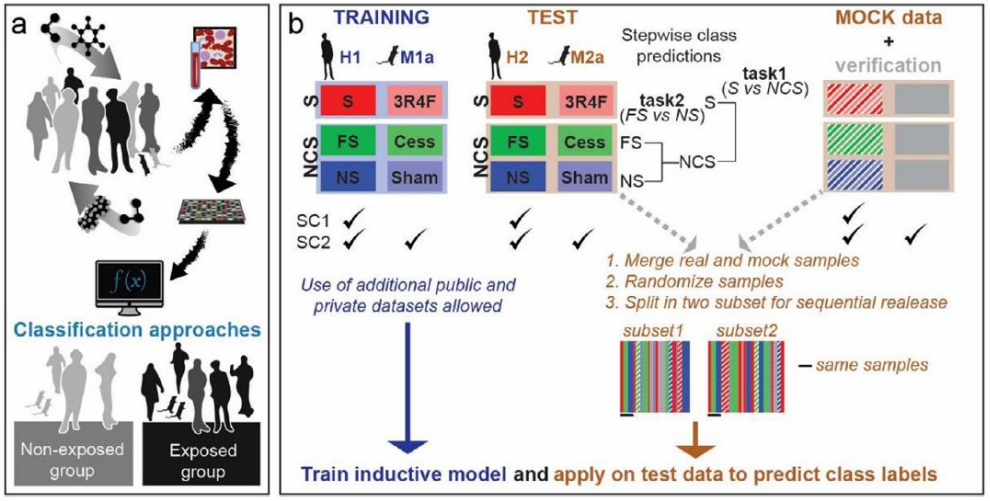


Fig. 1. Overview of the Systems Toxicology computational challenge. (a) Human and mouse blood samples were collected from smokers/3R4F-exposed (S/3R4F) and non-current smokers/not-exposed-to-3R4F (NCS) (mouse: exposed and non-exposed) and gene expression was measured. Classification approaches were developed by the participants to identify exposed and non-exposed subjects. (b) Human and mouse samples were divided into training (H1 and M1a) and test (H2 and M2a) datasets. Training datasets and class labels were released to allow participant to train their models. Test datasets (including mock samples) were released in two subsets. Participants were asked to provide their predictions on the first subset before the second subset was released. Participants had to apply their models to assess the class labels for the samples in the test set.

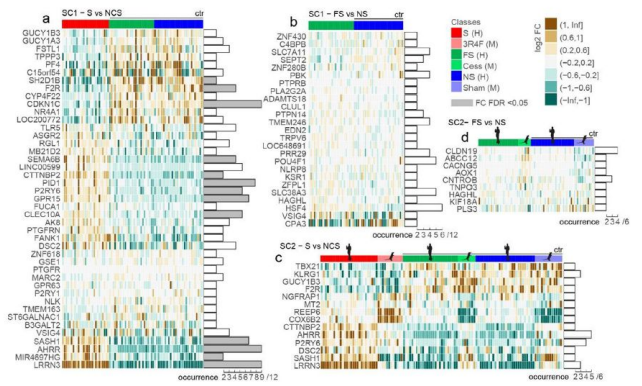


Fig. 5. Expression fold changes in the test dataset and co-occurrences of genes from consensus smoking exposure and cessation signatures. Differential gene expression heatmaps for the test datasets for (a, b) SC1 and (c, d) SC2. Subjects are in columns and grouped per class. Smokers (S) are in red (3R4F in light red), former smokers (FS) are in green (cessation (Cess) in light green), and never smokers (NS) are in blue (Sham in light blue). The respective control groups are annotated as ctr. (a) SC1: S vs NCS (ctr: FS + NS). (c) SC2: S vs NCS and 3R4F vs NCS (ctr: Cess + Sham). (b) SC1: FS vs NS. (d) SC2: FS vs NS and Cess vs Sham. Lengths of horizontal bars are proportional to the number of times a gene is selected as part of a signature. Gray bars denote genes for which the fold change (FC) is statistically significant (FDR < 0.05).

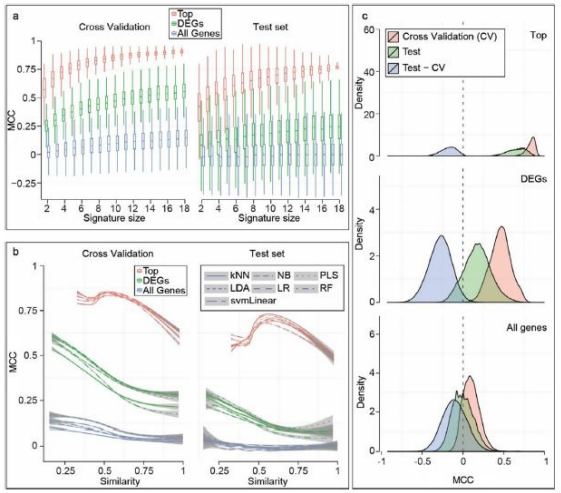
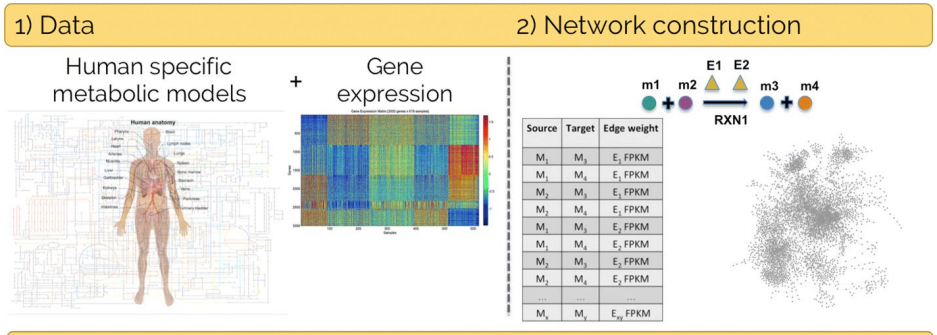


Fig. 6. Performance versus signature size and gene similarity. (a) Matthew correlation coefficient (MCC) score versus gene signature size for cross-validation and test dataset. Features were selected from the list of (i) "Top" genes (red), i.e., genes selected frequently by participants as part of the signature; (ii) "DEGs" (green), list of differentially expressed genes; (iii) "All Genes" (light blue), all measured genes. (b) MCC performance versus coefficient of similarity between genes in the signature. Seven different machine learning classifier were tested: (Random Forest (RF), support vector machine with linear kernel (svmLinear), partial least squares discriminant analysis (PLS), naive Bayes (NB), k-Nearest Neighbor (kNN), linear discriminant analysis (LDA), and logistic regression (LR)). (c) Distributions of MCC scores in CV (red) and test set (green) data, plus distribution of the differences (light blue), for "Top" (top), "DEGs" (middle), and "All genes" (bottom) selections.

Tumor metabolic networks



Manipur *et al. BMC Bioinformatics* 2020, **21**(Suppl 10):349
https://doi.org/10.1186/s12859-020-03564-9

BMC Bioinformatics

RESEARCH Open Access

Clustering analysis of tumor metabolic networks

Ichcha Manipur[1], Ilaria Granata[1], Lucia Maddalena[1] and Mario R. Guarracino[1,2*]

From 13th Bioinformatics and Computational Biology Conference - BBCC 2018
Naples, Italy. 19-21 November 2018

*Correspondence:
mario.guarracino@cnr.it
[1]National Research Council, Institute for High-Performance Computing and Networking, Via P. Castellino 111, 80131 Naples, Italy
[2]HSE - National Research University Higher School of Economics, LATNA Laboratory, 13 Rodionova Ulitsa, Nizhny Novgorod, Russia

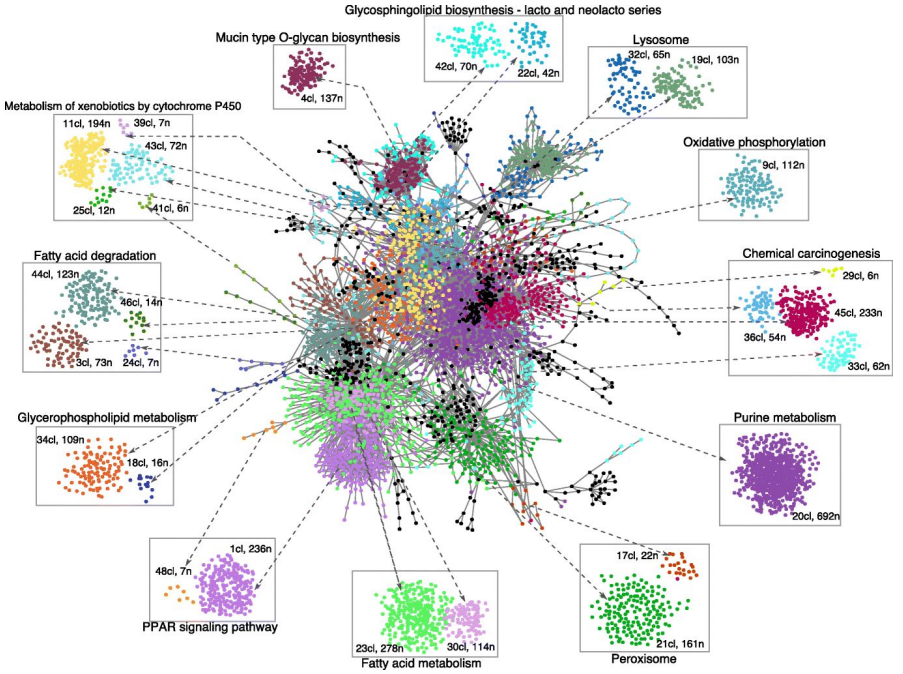
Abstract

Background: Biological networks are representative of the diverse molecular interactions that occur within cells. Some of the commonly studied biological networks are modeled through protein-protein interactions, gene regulatory, and metabolic pathways. Among these, metabolic networks are probably the most studied, as they directly influence all physiological processes. Exploration of biochemical pathways using multigraph representation is important in understanding complex regulatory mechanisms. Feature extraction and clustering of these networks enable grouping of samples obtained from different biological specimens. Clustering techniques separate networks depending on their mutual similarity.

Results: We present a clustering analysis on tissue-specific metabolic networks for single samples from three primary tumor sites: breast, lung, and kidney cancer. The metabolic networks were obtained by integrating genome scale metabolic models with gene expression data. We performed network simplification to reduce the computational time needed for the computation of network distances. We empirically proved that networks clustering can characterize groups of patients in multiple conditions.

Conclusions: We provide a computational methodology to explore and characterize the metabolic landscape of tumors, thus providing a general methodology to integrate analytic metabolic models with gene expression data. This method represents a first attempt in clustering large scale metabolic networks. Moreover, this approach gives the possibility to get valuable information on what are the effects of different conditions on the overall metabolism.

Keywords: Metabolic networks, Network simplification, Networks clustering



Cancer cell death

Artificial Intelligence in Medicine 53 (2011) 119–125

Contents lists available at ScienceDirect

Artificial Intelligence in Medicine

journal homepage: www.elsevier.com/locate/aiim

Classification of cancer cell death with spectral dimensionality reduction and generalized eigenvalues

Mario R. Guarracino[a], Petros Xanthopoulos[b], Georgios Pyrgiotakis[c], Vera Tomaino[b,d], Brij M. Moudgil[c], Panos M. Pardalos[b,*]

[a] High Performance Computing and Networking Institute - National Research Council of Italy (ICAR-CNR), Via P. Castellino, 111 - 80131 Naples, Italy
[b] Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611-6595, USA
[c] Particle Engineering Research Center, University of Florida, Gainesville, FL 32611, USA
[d] Department of Experimental Medicine and Clinic, University Magna Græcia of Catanzaro, Italy

ARTICLE INFO

Article history:
Received 26 April 2010
Received in revised form 22 February 2011
Accepted 18 July 2011

Keywords:
Spectral clustering
Dimensionality reduction
Generalized eigenvalue classification
Raman spectroscopy
Cancer treatment

ABSTRACT

Objective: Accurate cell death discrimination is a time consuming and expensive process that can only be performed in biological laboratories. Nevertheless, it is very useful and arises in many biological and medical applications.
Methods and material: Raman spectra are collected for 84 samples of A549 cell line (human lung cancer epithelia cells) that has been exposed to toxins to simulate the necrotic and apoptotic death. The proposed data mining approach for the multiclass cell death discrimination problem uses a multiclass regularized generalized eigenvalue algorithm for classification (multiReGEC), together with a dimensionality reduction algorithm based on spectral clustering.
Results: The proposed algorithmic scheme can classify A549 lung cancer cells from three different classes (apoptotic death, necrotic death and control cells) with $97.78\% \pm 0.047$ accuracy versus 92.22 ± 0.095 without the proposed feature selection preprocessing. The spectrum areas depicted by the algorithm corresponds to the)C= O bond from the lipids and the lipid bilayer. This chemical structure undergoes different change of state based on cell death type. Further evidence of the validity of the technique is obtained through the successful classification of 7 cell spectra that undergo hyperthermic treatment.
Conclusions: In this study we propose a fast and automated way of processing Raman spectra for cell death discrimination, using a feature selection algorithm that not only enhances the classification accuracy, but also gives more insight in the undergoing cell death process.

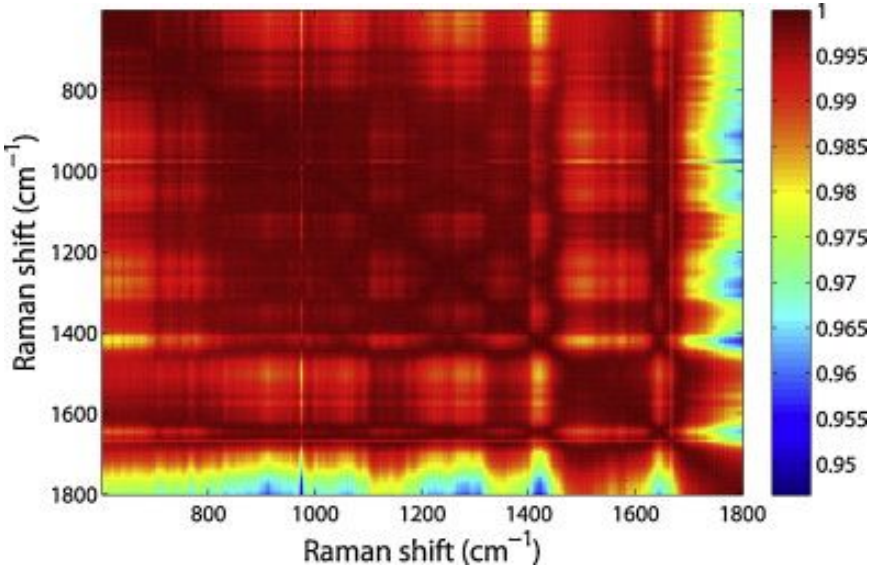
1. Introduction

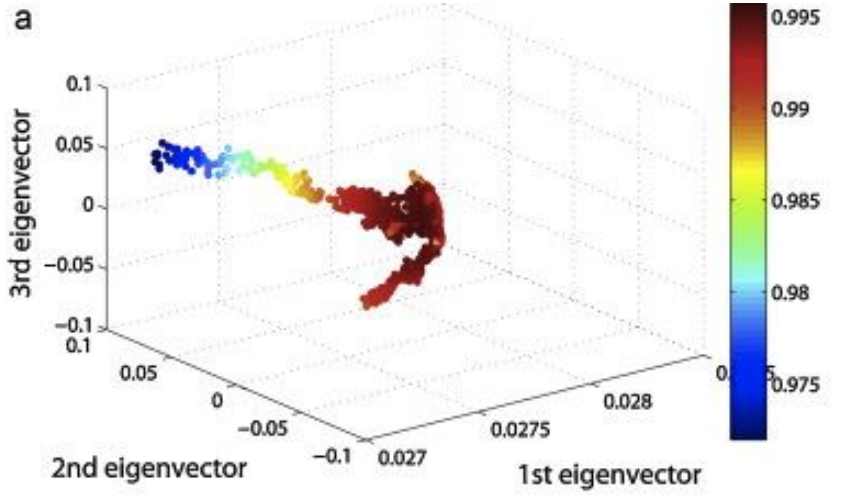
Cell classification has recently emerged as one of the most critical problems in modern biology. It has a wide range of applications: cell line purity [1], stem cells differentiation [2], cancer diagnosis and cell death [3] to mention a few. In case of cell death, the assessment of the cell pathway is critical when it comes to in vitro trials of chemotherapeutic drugs [4,5] and substance toxicity [6]. To this day, the most accurate way of discriminating among various cells is through genetic profiling (gene expression) [7] or identification with biomarkers, which can take long time, requires expensive equipment and trained personnel [8]. In cases where the cell type or cell itself needs to be recognized immediately, researchers rely on the morphological features of the cell. The latter strategy, however, is highly subjective and prone to errors.

One of the recently emerging techniques that has potential for cell discrimination is Raman spectroscopy [9]. The advantage of this method over Fourier transform infrared spectroscopy, or other vibrational techniques, is the very little interference of water, which is the basic constituent of the cells. In addition, it does not need any marker, or chemical. Furthermore, it can be done in vitro or in vivo and it is non-invasive. Finally, it is relatively rapid (10–30 s per acquisition) and it can be used outside laboratories [10]. Raman spectroscopy is based on the scattering of a laser light in the interaction with a sample. In the case of cells, the obtained spectrum is a set of frequencies changed by the interactions with DNA/RNA, proteins, lipids and amino acids [11].

Recently, there have been attempts to integrate statistical techniques such as principal component analysis and linear

* Corresponding author. Tel.: +1 352 392 9011; fax: +1 352 392 3537.
E-mail addresses: mario.guarracino@cnr.it (M.R. Guarracino), petrosx@ufl.edu





Clustering: hyrarchical & partition methods

Overview of Clustering

- Clustering is often used for exploratory analysis of the data.
- Clustering methods find the similarities between objects according to their attributes, and group them into clusters.
- Clustering groups similar observations, keeping them apart from those that are different.

Clustering super heros



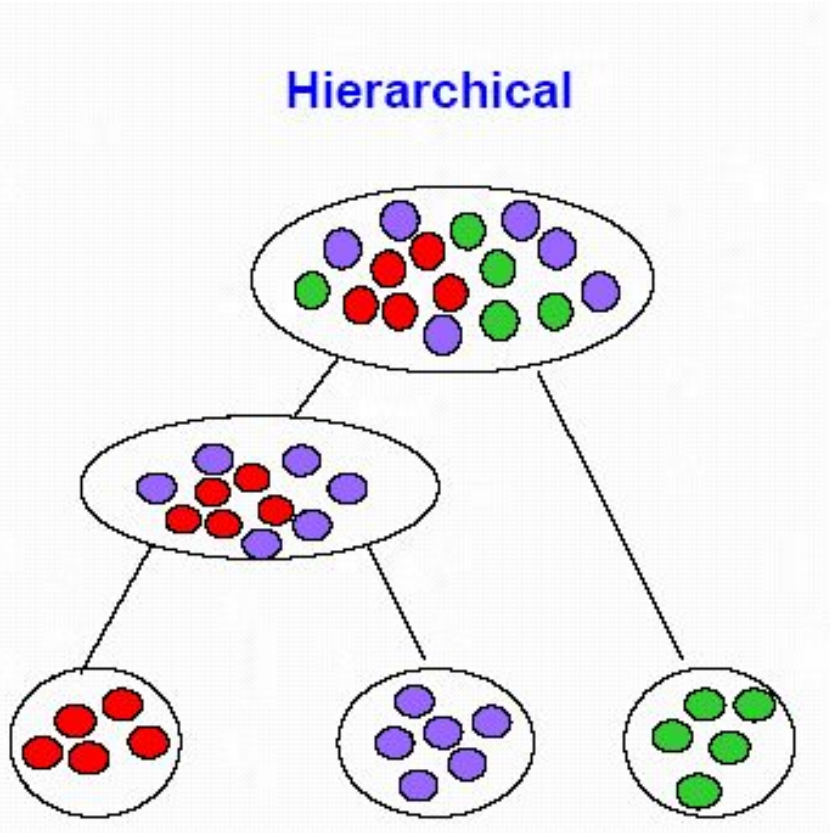


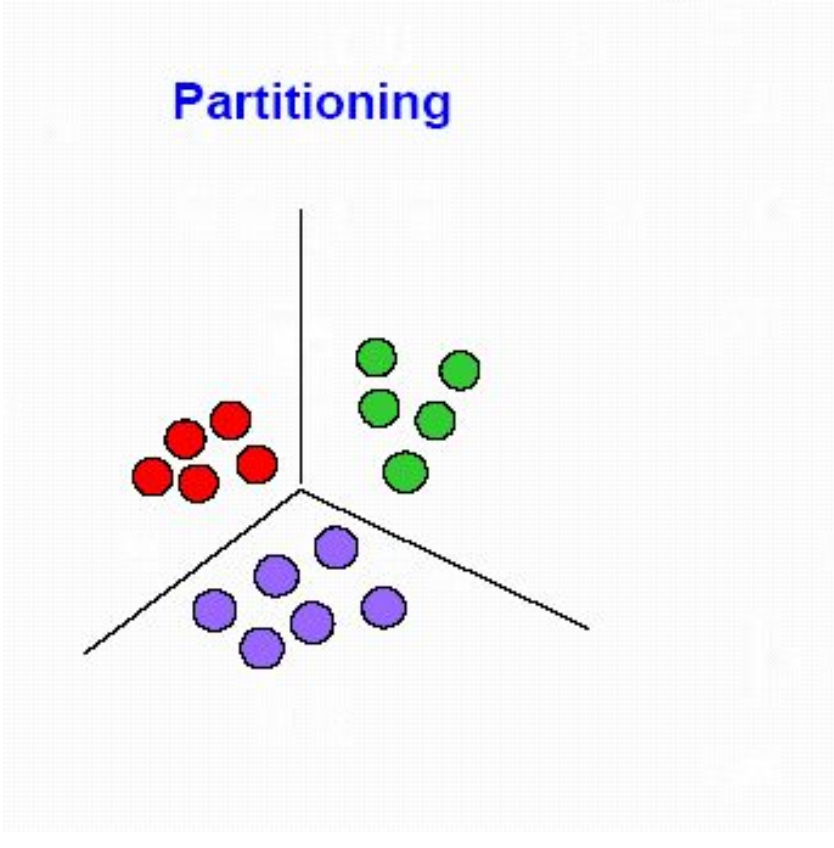





Clustering methods

Clustering algorithms come in two basic flavors





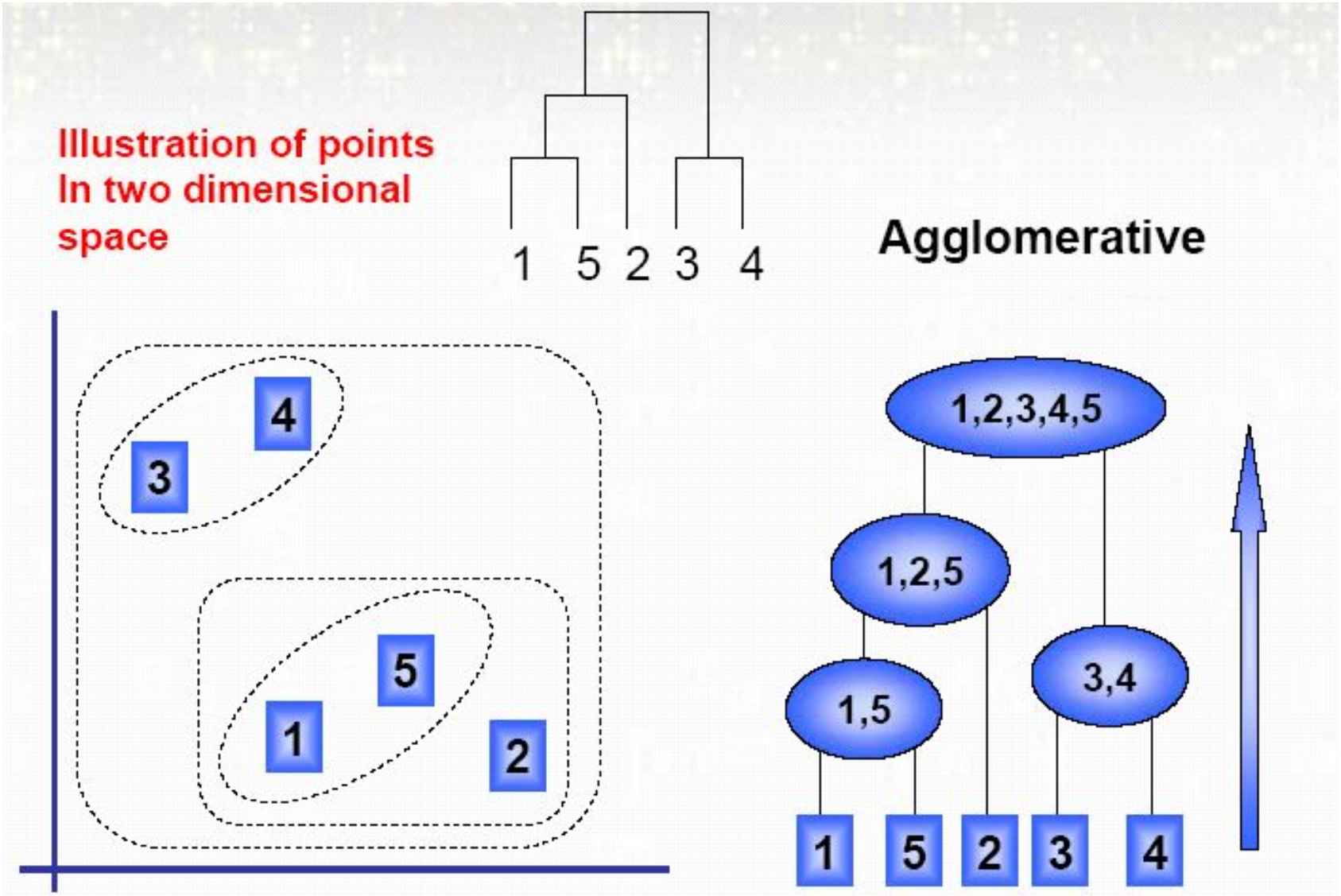
Hierarchical clustering

- Hierarchical clustering methods produce a tree or dendrogram.
- They avoid specifying how many clusters are appropriate by providing a partition for each k obtained from cutting the tree at some level.
- The tree can be built in two distinct ways
  - bottom-up: agglomerative clustering.
  - top-down: divisive clustering.

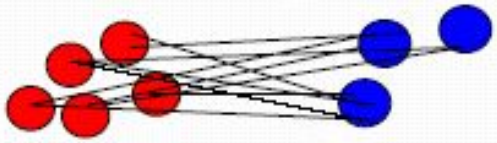
Hierarchical clustering

- Agglomerative methods are the most popular hierarchical clustering methods
- Start with k clusters of observations
  - At each step, merge the two closest clusters using a measure of between-cluster dissimilarity which reflects the shape of the clusters
  - The distance between clusters is defined by the method used
    - *e.g., if complete linkage, the distance is defined as the distance between furthest pair of points in the two clusters*

Hierarchical clustering



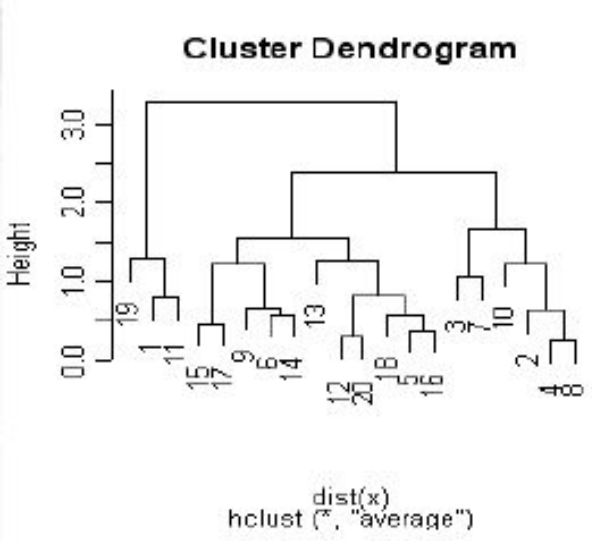
Distance between clusters

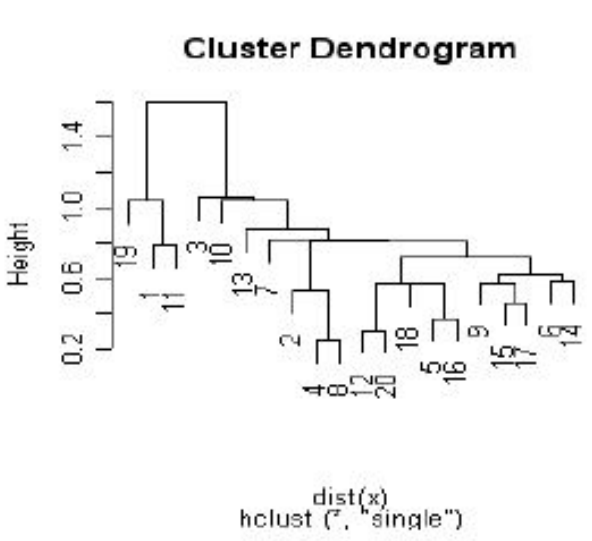


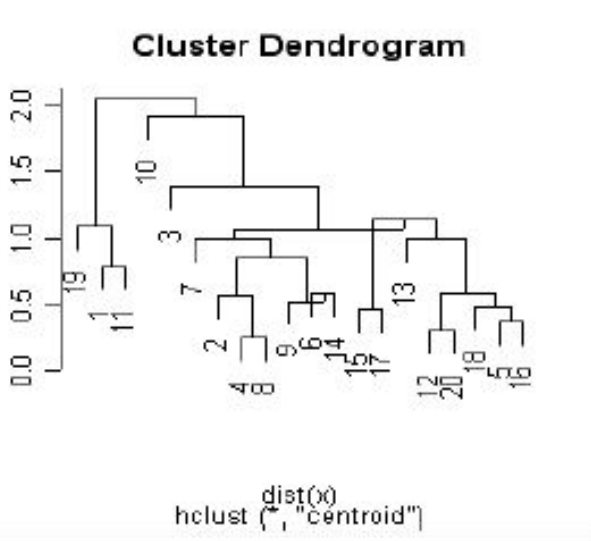
Average (Mean) linkage

Single (minimum)

Distance between centroids

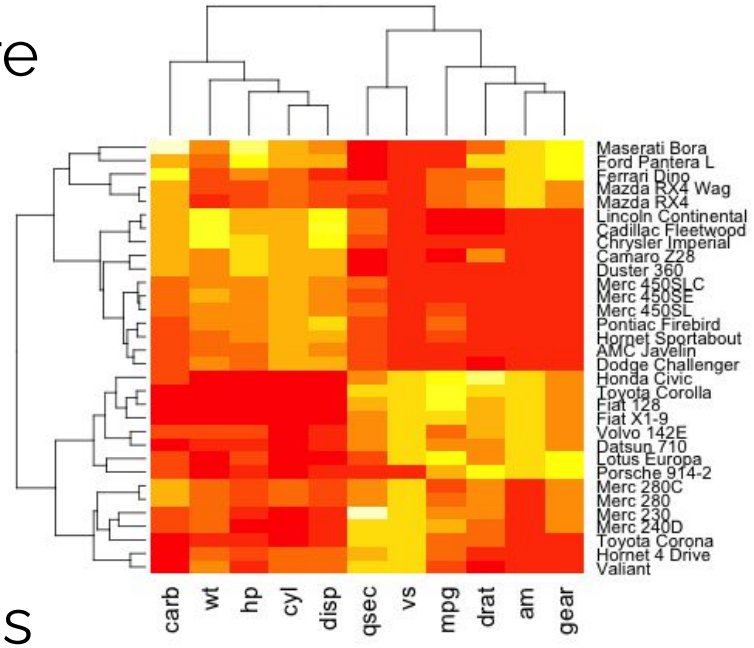






Hierarchical clustering

- Gives us pretty red-green picture with patterns (heatmaps)
- But the pretty picture tends to be unstable.
  - Tend to be sensitive to small changes in the data
- Provides clusters of every size: where to "cut" the dendrogram is user-determined.



https://matplotlib.org/3.1.1/gallery/images_contours_and_fields/image_annotated_heatmap.html

Partitioning methods

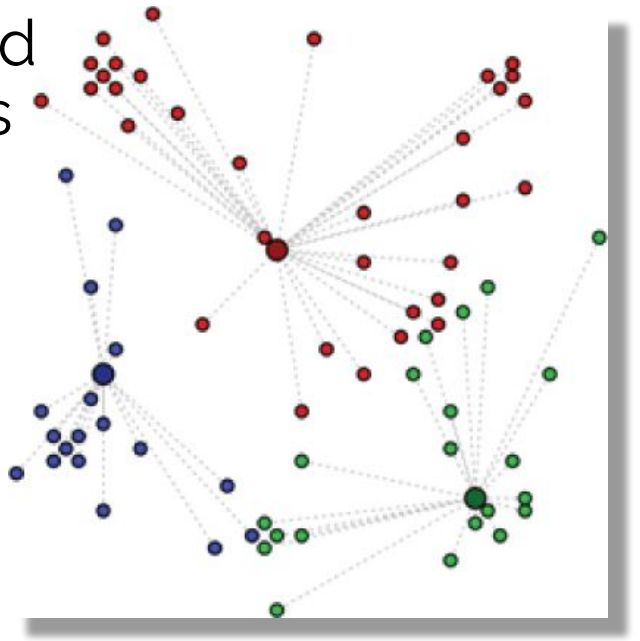
- K-means
- K-medoids
- Partition around Memoids (PAM)
- Spectral clustering
- Dynamic clustering
- Affinity propagation
- Self-Organizing maps
- Model-based clustering
- DBSCAN
- CLICK
- FUZZY
- Bayesian model-based clustering
- Tight clustering
- Penalized and weighted K-means

K-means

- Given a collection of objects each with n measurable attributes, for a chosen value of k, k-means identifies k clusters of objects based on the objects' proximity to the center of the k groups.
- The center is determined as the arithmetic average (mean) of each cluster's n-dimensional vector of attributes.
- Each object is assigned to the only one cluster whose center is closest.

K-means

- The figure illustrates three clusters of objects with two attributes.
- Each object in the dataset is depicted by a small dot, whose color indicates to which cluster it belongs.
- Each large dot represents the mean of the cluster.



Overview of the method

- Each object in the following examples has two attributes and corresponds to the point (x,y), where x and y denote the two attributes and $i = 1, 2, \ldots, M$.
- For a given cluster of m points ($m \leq M$), the point that corresponds to the cluster's mean is called a centroid.
- In mathematics, a centroid refers to a point that corresponds to the center of mass for an object.

K-means

Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$, where each observation is a *d*-dimensional real vector, *k*-means aims to partition the *n* observations into $k\ (\leq n)$ sets $\mathbf{S} = \{S_1, S_2, ..., S_k\}$ so as to minimize the within-cluster sum of squares (WCSS):

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

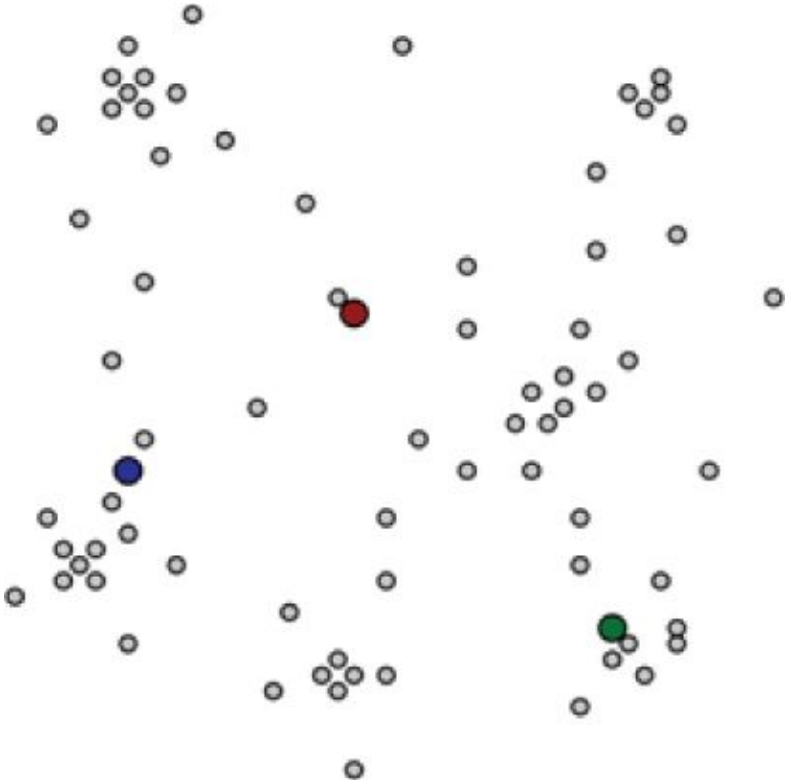
This is equivalent to:

minimizing the pairwise squared deviations of points in the same cluster

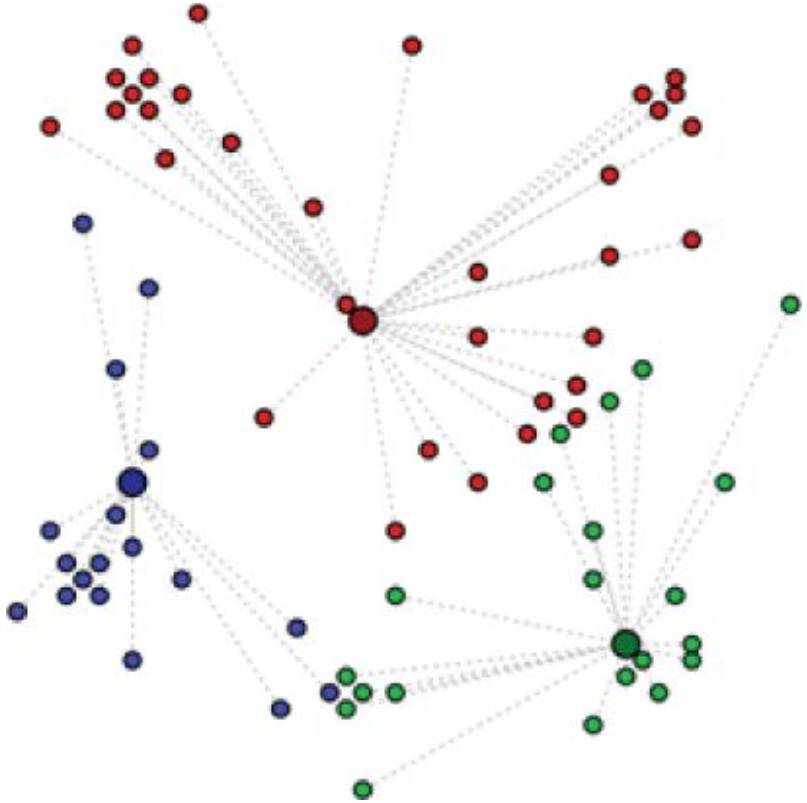
K-means algorithm

1. Choose K centroids at random.
2. Make initial partition of objects into k clusters by assigning objects to closest centroid.
3. M-step: Calculate the centroid (mean) of each of the k clusters.
4. E-step: Reassign objects to the closest centroids.
5. Repeat 3 and 4 until no reallocations occur.

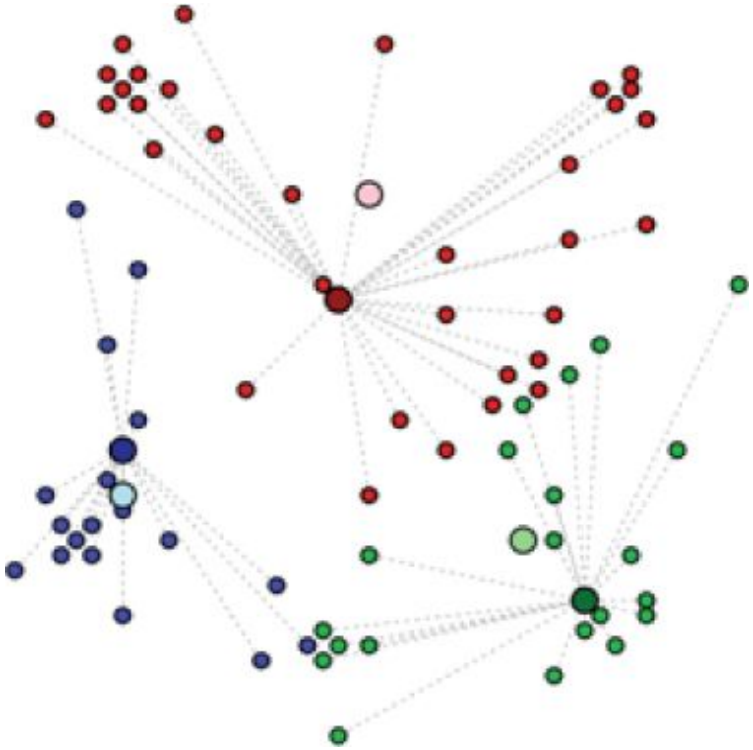
1. Choose K centroids at random

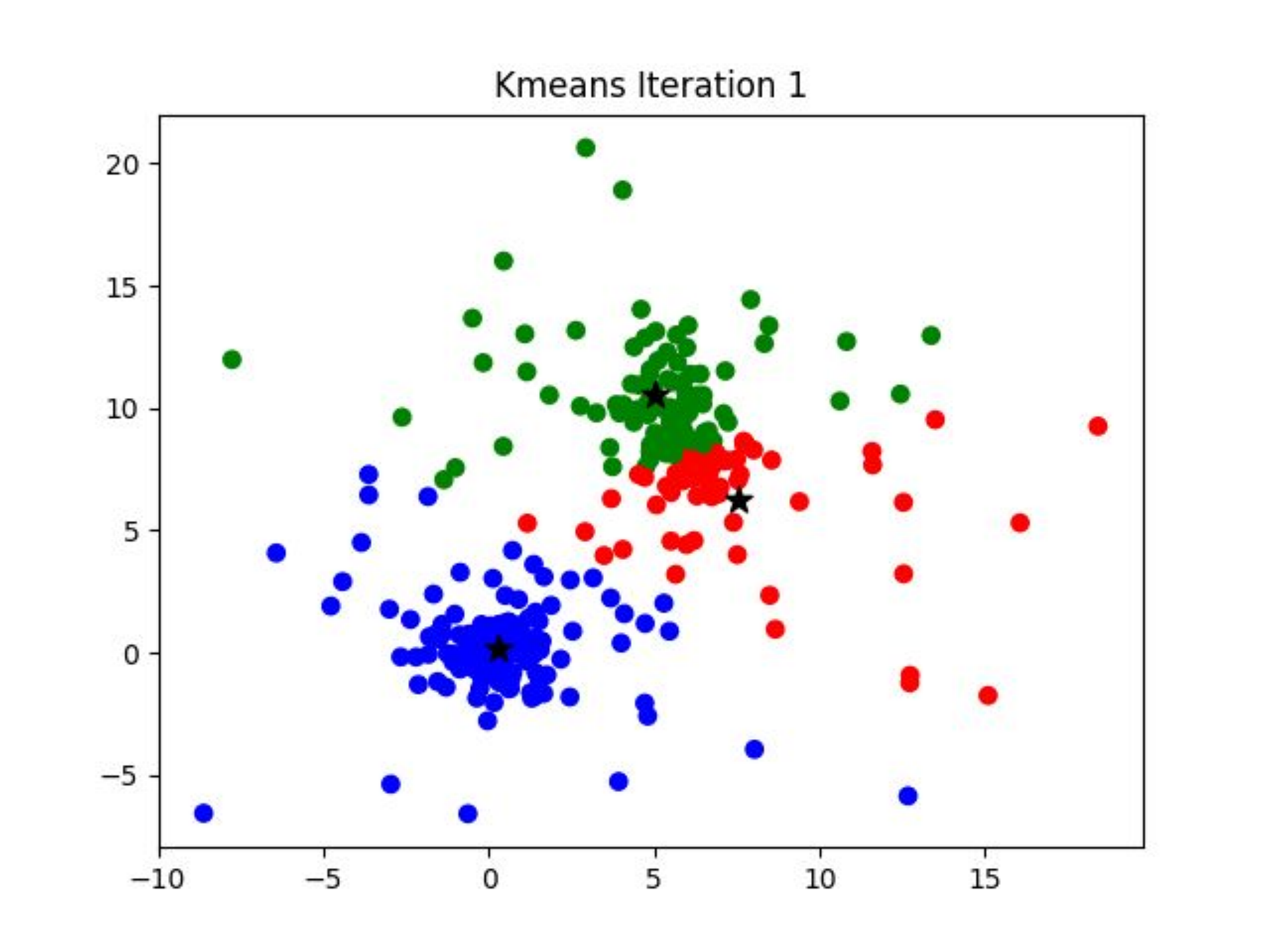


2. Partition objects into k clusters



3. Calculate the centroid (mean) of each of the k clusters.




Kmeans Iteration 1
20
15
10
5
0
−5
−10
−5
0
5
10
15

Determining the Number of Clusters

- With the preceding algorithm, k clusters can be identified in a given dataset, but which value of k should be selected?
- The value of k can be chosen based on a reasonable guess or some predefined requirement.
- However, even then, it would be good to know how much better or worse having k clusters versus k – 1 or k + 1 clusters would be in explaining the structure of the data.

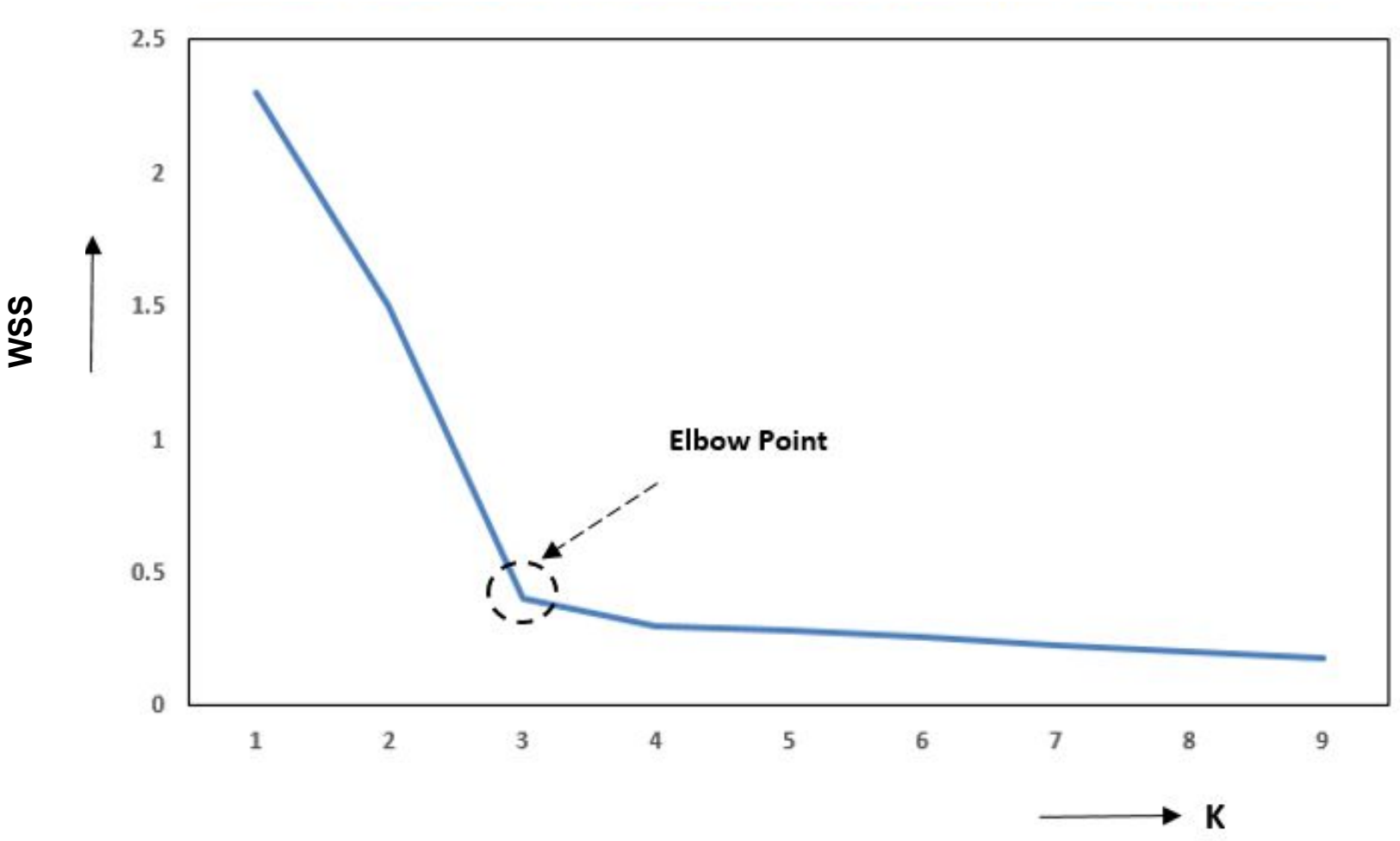
Determining the Number of Clusters

- We can use the Within Sum of Squares (WSS) metric, to determine a reasonably optimal value of k.

$$WSS = \sum_{i=1}^{M} d(p_i, q^{(i)})^2 = \sum_{i=1}^{M} \sum_{j=1}^{n} \left(p_{ij} - q_j^{(i)}\right)^2$$

- WSS is the sum of the squares of the distances between each data point and the closest centroid.
- The term $q^{(i)}$ indicates the closest centroid that is associated with the ith point.
  - If points are close to their respective centroids, WSS is small.
- If k + 1 clusters do not reduce the value of WSS with respect to k clusters, there is no benefit to adding another cluster.

Elbow method

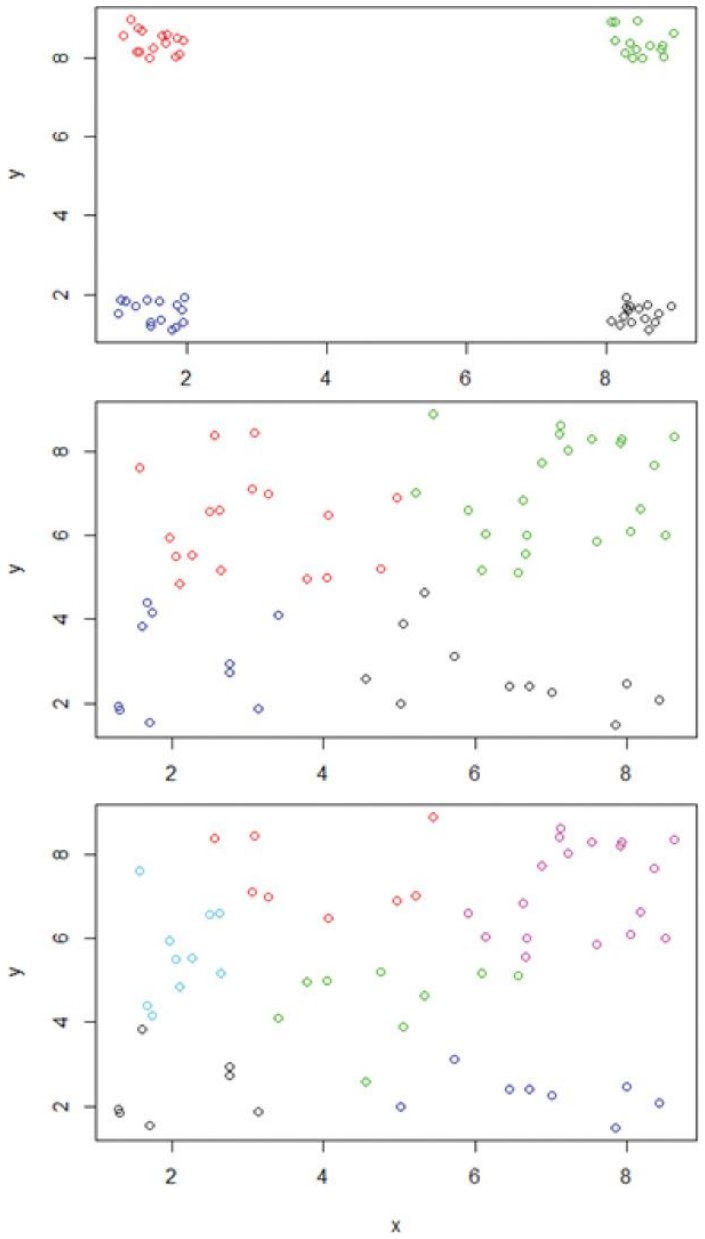


Diagnostics

- The WSS heuristic can provide several possible k values to consider.
- When the number of attributes is relatively small, a common approach to further refine the choice of k is to plot the data to determine how distinct the identified clusters are from each other.
- In general, the following questions should be considered.
  - Are the clusters well separated from each other?
  - Do any of the clusters have only a few points?
  - Do any of the centroids appear to be too close to each other?

Diagnostics

- In the first case, ideally the plot would look like the top one, for n = 4.
  - The clusters are well defined, with considerable space between the four identified clusters.
- However, in other cases, the clusters may be close to each other, and the distinction may not be so obvious.



Reasons to Choose and Cautions

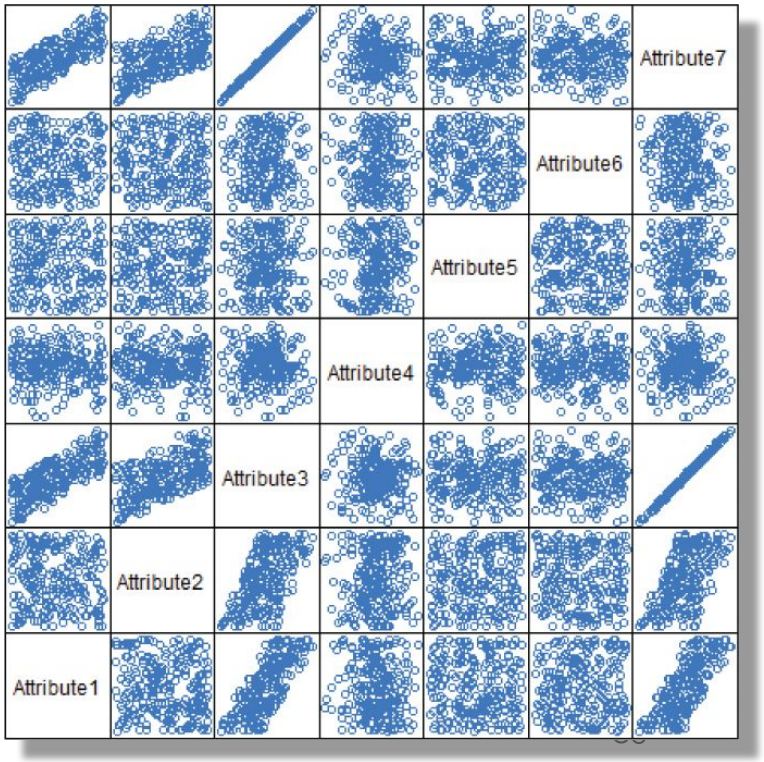
- Although k-means is quite useful, there are still several decisions that the practitioner must make:
  - What object attributes should be included in the analysis?
  - What unit of measure (for example, miles or kilometers) should be used for each attribute?
  - Do the attributes need rescaling so that one attribute does not have a disproportionate effect on the results?
  - What other considerations might apply?

Objects attribute

- Too many attributes can minimize the impact of the most important variables.
- The use of several similar attributes can place too much importance on one type of attribute.
  - For example, if five attributes related to personal wealth are included in a clustering analysis, these would dominate the analysis and mask the importance of other attributes.
- Detecting variables with low values and variability help is filtering out unecessary characteristics.

Objects attribute

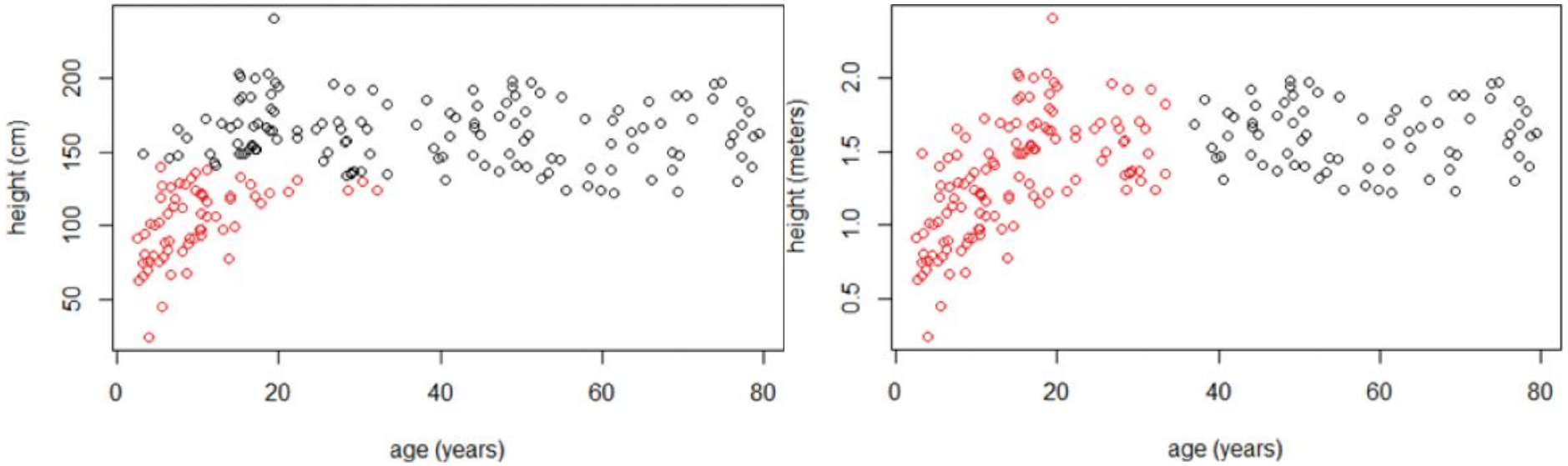
- When dealing with groups of highly correlated attributes, a useful approach is to use only some from each group.
- A scatterplot matrix is a useful tool to visualize the pairwise relationships between the attributes.
- When dealing with many variables ($>10^2$), better to use correlation heatmaps



Unit of measures

- k-means algorithm is indifferent to the units of measure for a given attribute.
  - Meters or centimeters for a patient's height.
- However, the algorithm will identify different clusters depending on the choice of the units of measure.
- For example, suppose that k-means is used to cluster patients based on age in years and height in centimeters.

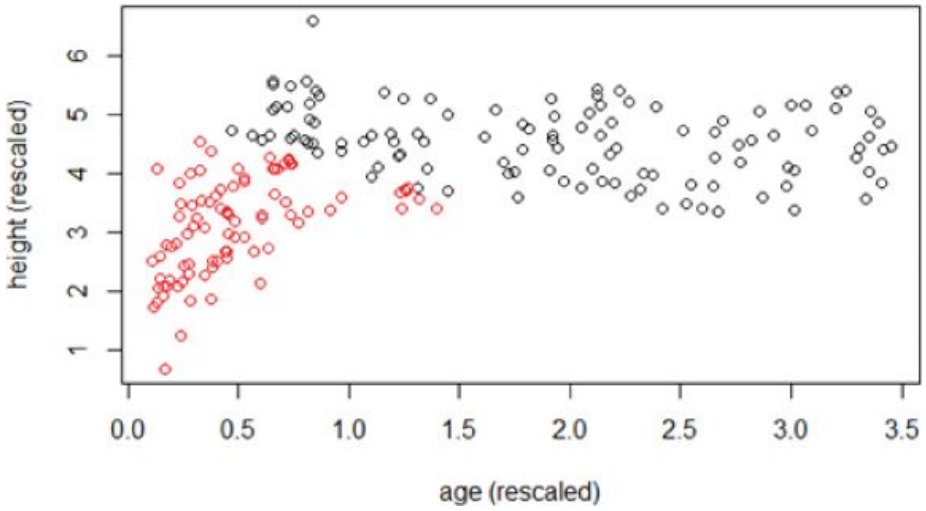
Unit of measures



- When the height is expressed in meters, the magnitude of the ages dominates the distance calculation between two points.

Rescaling

- Dividing each attribute value by its standard deviation changes the k-means analysis results.

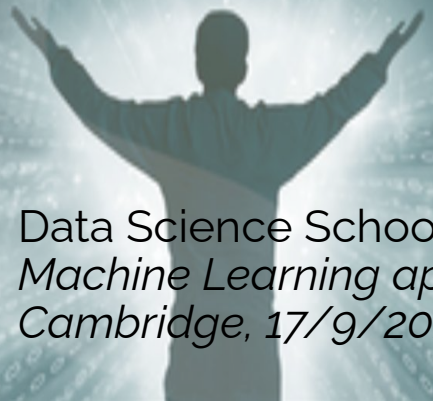


- Subtracting the mean before dividing by the std. dev. (z-score) is unnecessary, because the distance is sensitive to the scale of the attribute, not its location.

Lecture 2
Introduction to Classifiers

Mario Guarracino

University of Cassino and Southern Lazio



Data Science School

Machine Learning applications for life sciences

Cambridge, 17/9/2020

Version 2 2020 09 16 10:00

Introduction

- In addition to analytical methods such as clustering, association rule learning, and modeling techniques like regression, classification is another fundamental learning method.
- In classification, a classifier is presented with a set of examples that are already divided in classes and, starting from these, the classifier learns to assign unseen examples to the classes.
- In other words, the primary task performed by classifiers is to assign class labels to new observations.

Introduction

- Most classification methods are supervised: they start with a training set of prelabeled observations to learn how the attributes of these observations may contribute to the classification of future unlabeled observations.
- For example, existing marketing, sales, and customer demographic data can be used to develop a classifier to assign a “purchase” or “no purchase” label to potential future customers.

Introduction

- Classification is widely used for prediction purposes.
- For example, by building a classifier on the transcripts of United States Congressional floor debates, it can be determined whether the speeches represent support or opposition to proposed legislation.
- Classification can help health care professionals diagnose heart disease patients.
- Based on an e-mail's content, e-mail providers also use classification to decide whether the incoming e-mail messages are spam.

Supervised learning

- *Supervised learning* refers to the capability of a system to learn from examples (*training set*)
- The trained system is able to provide an answer (*output*) for each new question (*input*)
- *Supervised* means the desired output for the training set is provided by an external teacher
- *Binary classification* is among the most successful methods for supervised learning

Problem position

Data: Objects $\{X_i, Y_i\}(i=1,\ldots,n)$ i.i.d. from joint distribution $\{X, Y\}$.
Each object X_i is associated with a class label $Y_i \in \{1,\ldots,K\}$.

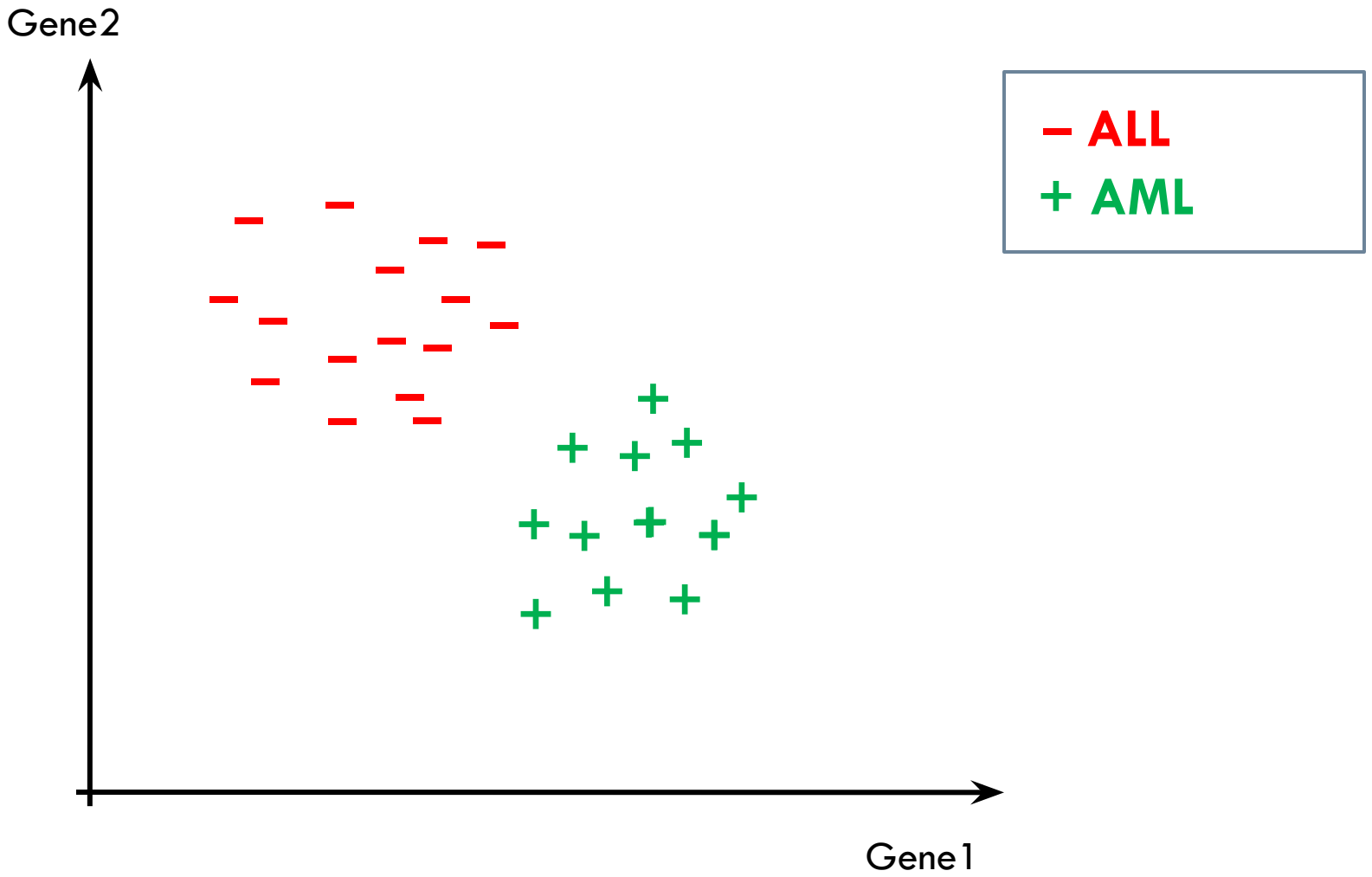
Method: Compute a classification function $C(X)$ that predicts the class label Y with low error rate. (error rate: $\#\{i: Y_i \neq C(X_i)\}$)

We want the classifier learned from the training data to generalize to (predict) the label of a new example.

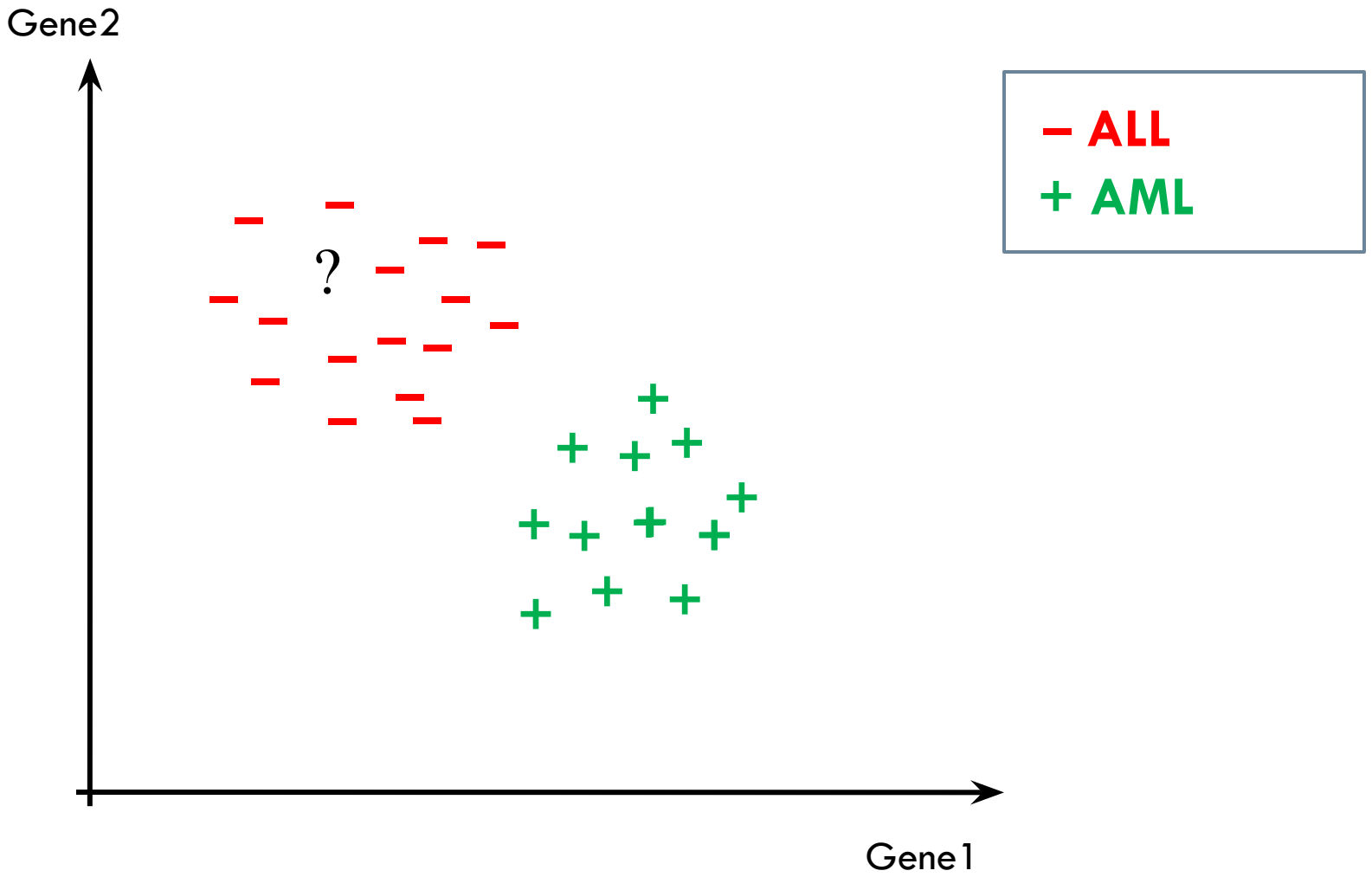
Goal: Find a classifier $C(X)$ with high generalization ability.

In the following discussion, only consider binary classification (K=2).

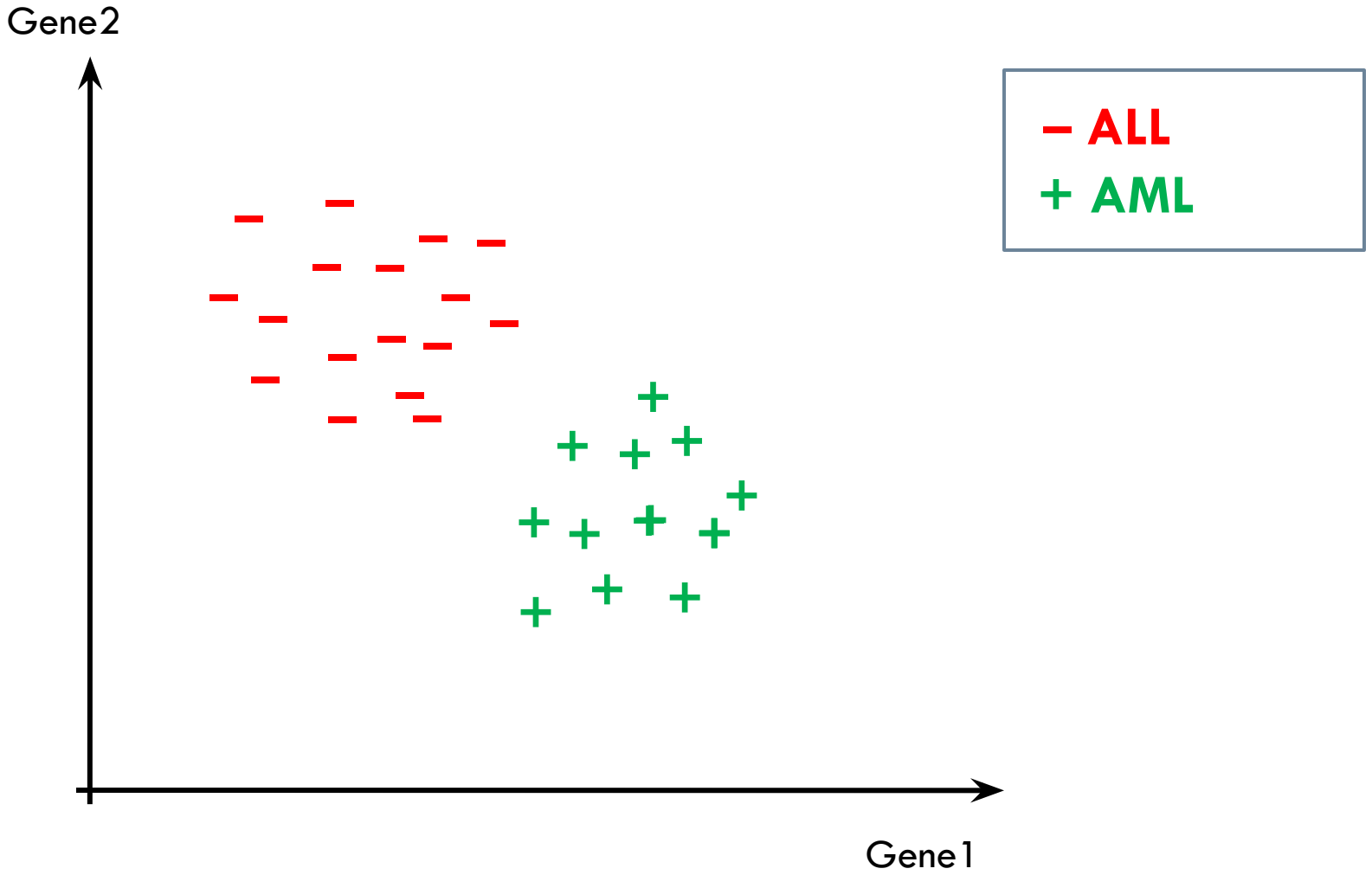
Disease subtypes classification



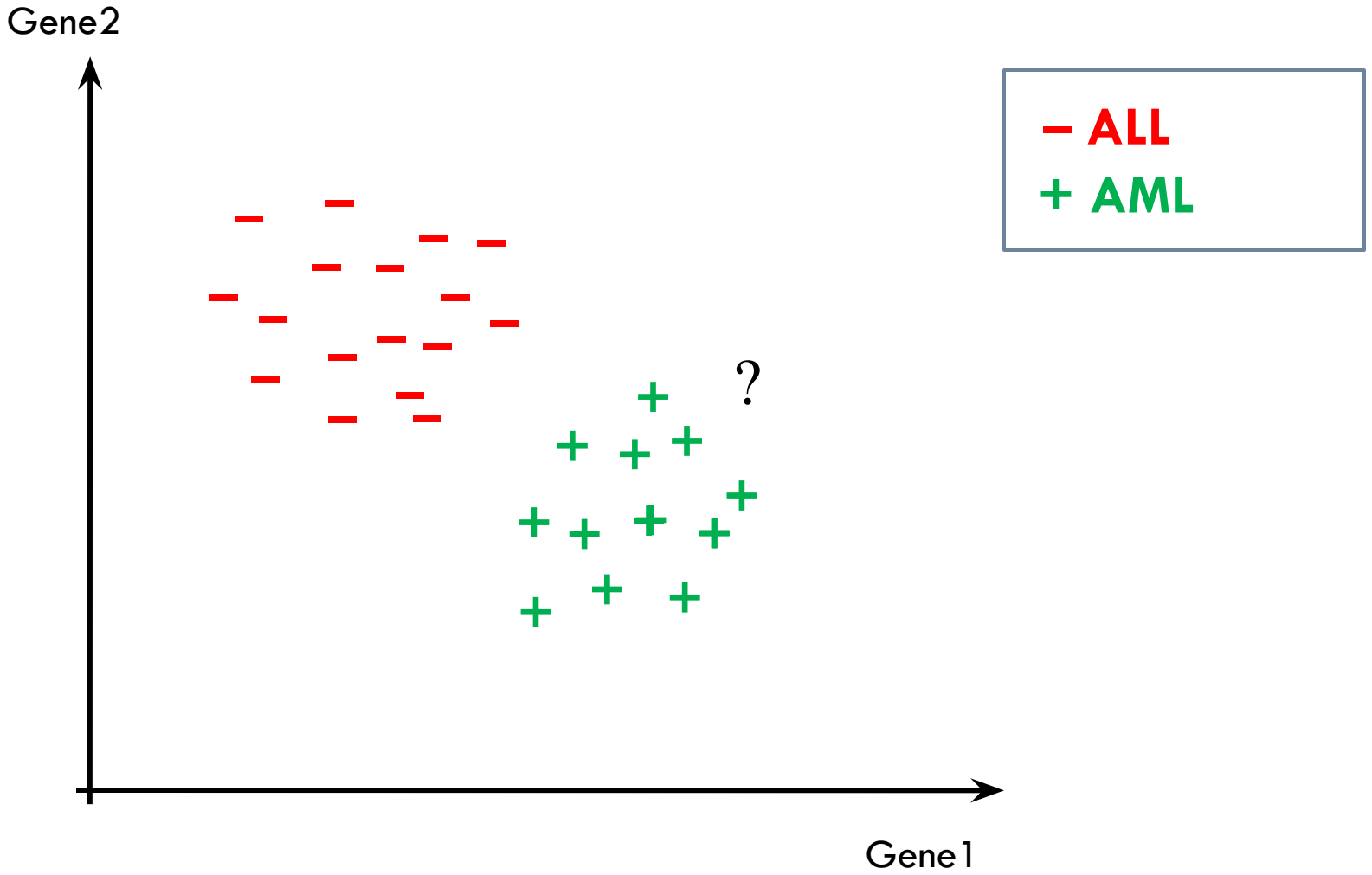
Disease subtypes classification



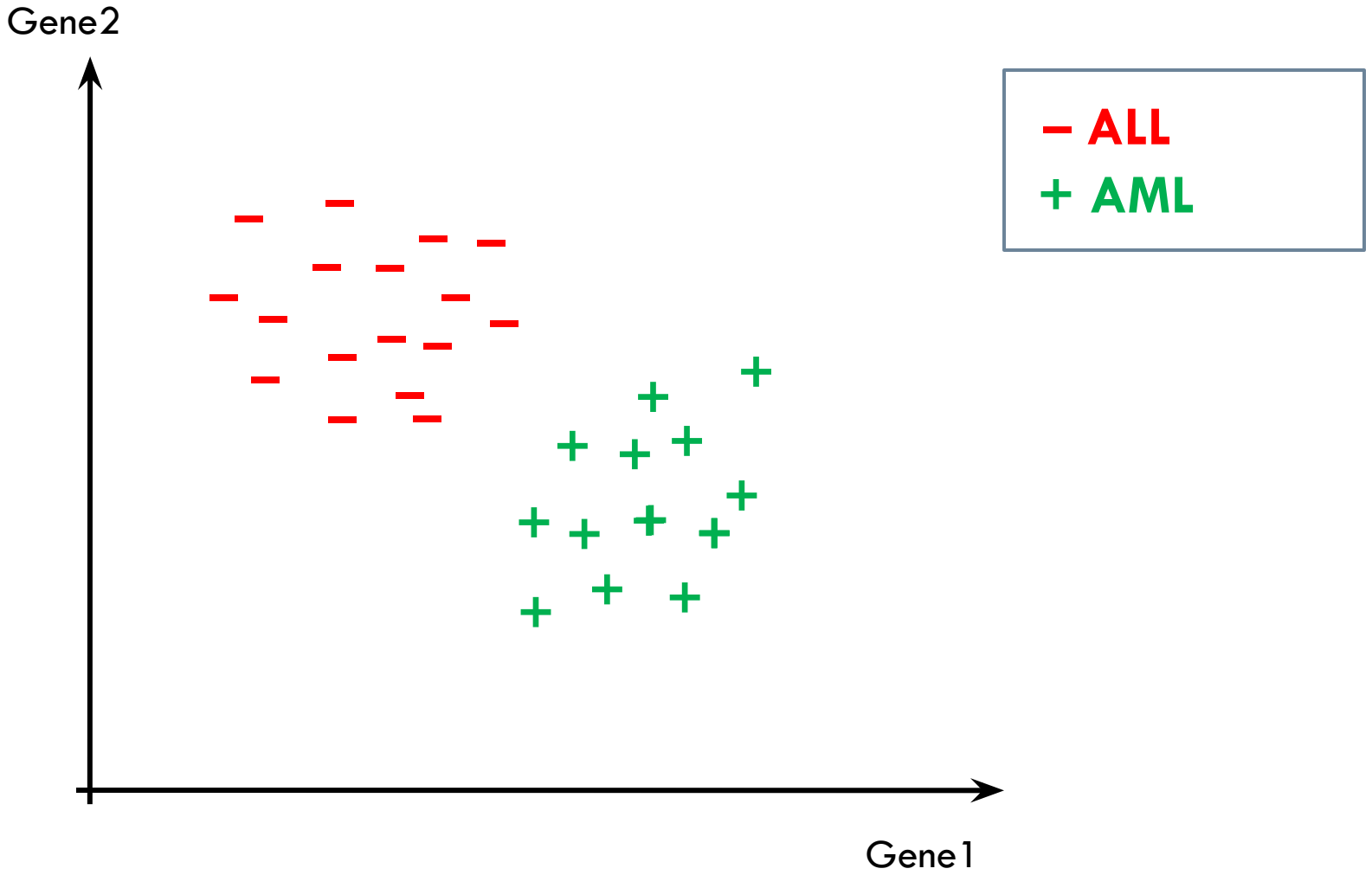
Disease subtypes classification



Disease subtypes classification



Disease subtypes classification



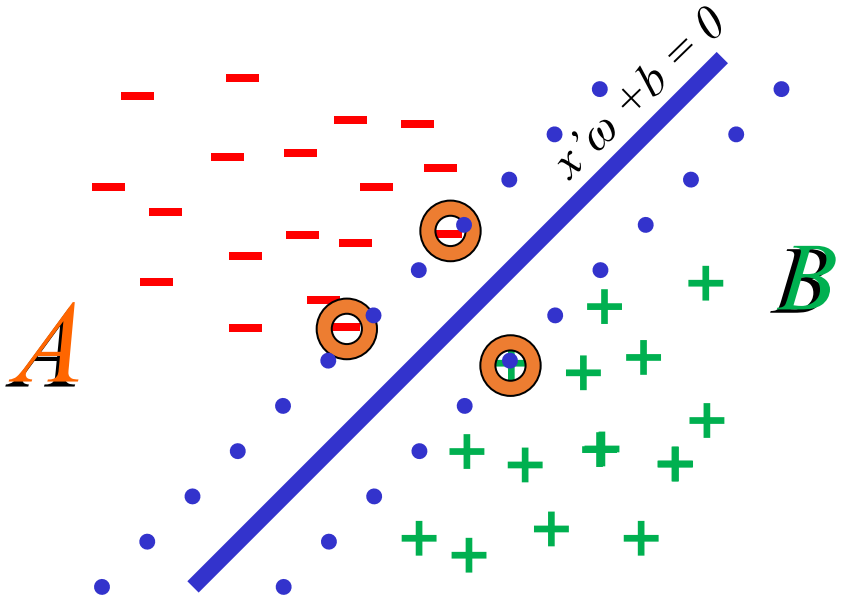
Nearest Neighbor Classification

- Based on a measure of distance between observations (e.g. Euclidean distance or one minus correlation)
- k-nearest neighbor rule (Fix and Hodges, 1951) classifies an observation X as follows:
  - finds the k observations in the learning set closest to X
  - predicts the class of X by majority vote, i.e., choose the class that is most common among those k observations.
- The number of neighbors k can be chosen by cross-validation (more on this later)

Generalized Eigenvalue Classifiers

Support Vector Machines

- Find the plane $x'\omega+b=0$ which maximizes the margin between the two classes



$$\min_{\omega \neq 0} \frac{\|\omega\|^2}{2}$$

$$s.t. \quad A\omega + b \geq e$$
$$B\omega + b < -e$$

- Only few points are needed to compute the plane (*support vectors*)

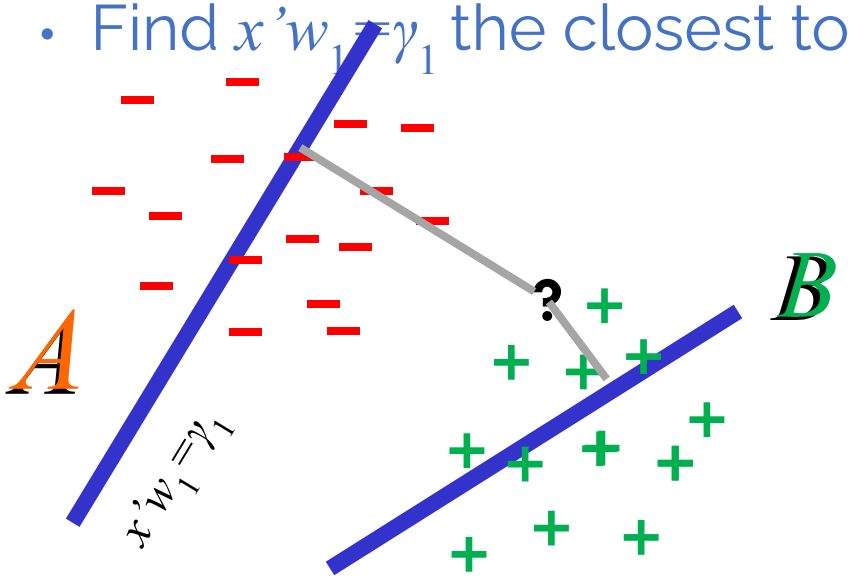
V. Vapnik, The Nature of Statistical Learning Theory. 1995

SVM classification

- The robustness of SVM relies in the strong fundamentals of statistical learning theory
- The training relies on optimization of a quadratic convex cost function, for which many methods are available
  - Available packages for Python, R, Matlab, Weka include SVM-Lite and LIBSVM
- These techniques can be extended to the nonlinear discrimination, embedding the data in a nonlinear space using *kernel functions*

A different approach: ReGEC

- A binary classification problem can be formulated as a generalized eigenvalue problem (ReGEC)
  - Find $x'w_1=\gamma_1$ the closest to A and the farthest from B:

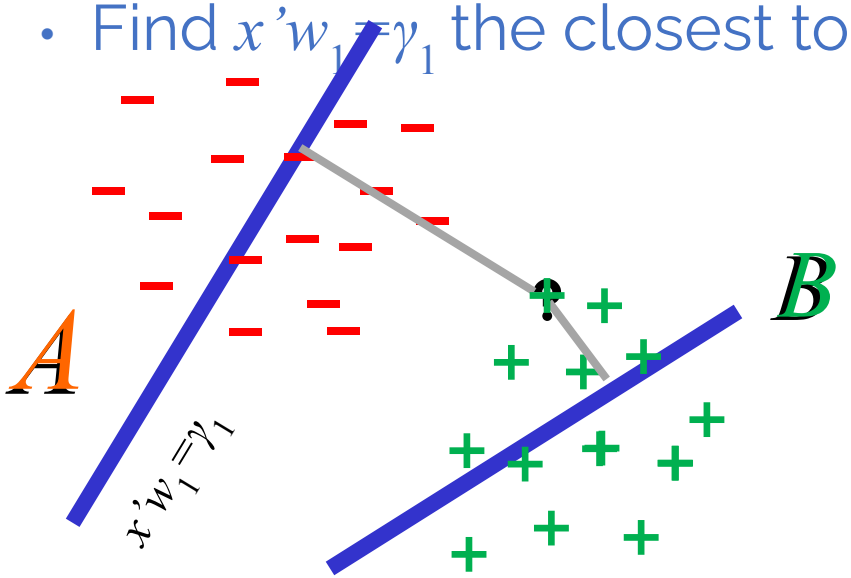


$$\min_{\omega,\gamma\neq 0} \frac{\| A\omega - e\gamma \|^2}{\| B\omega - e\gamma \|^2}$$

O. Mangasarian, E. Wild Multisurface Proximal Support Vector Classification via Generalized Eigenvalues. IEEE PAMI 2006.

A different approach: ReGEC

- A binary classification problem can be formulated as a generalized eigenvalue problem (ReGEC)
  - Find $x'w_1=\gamma_1$ the closest to A and the farthest from B:

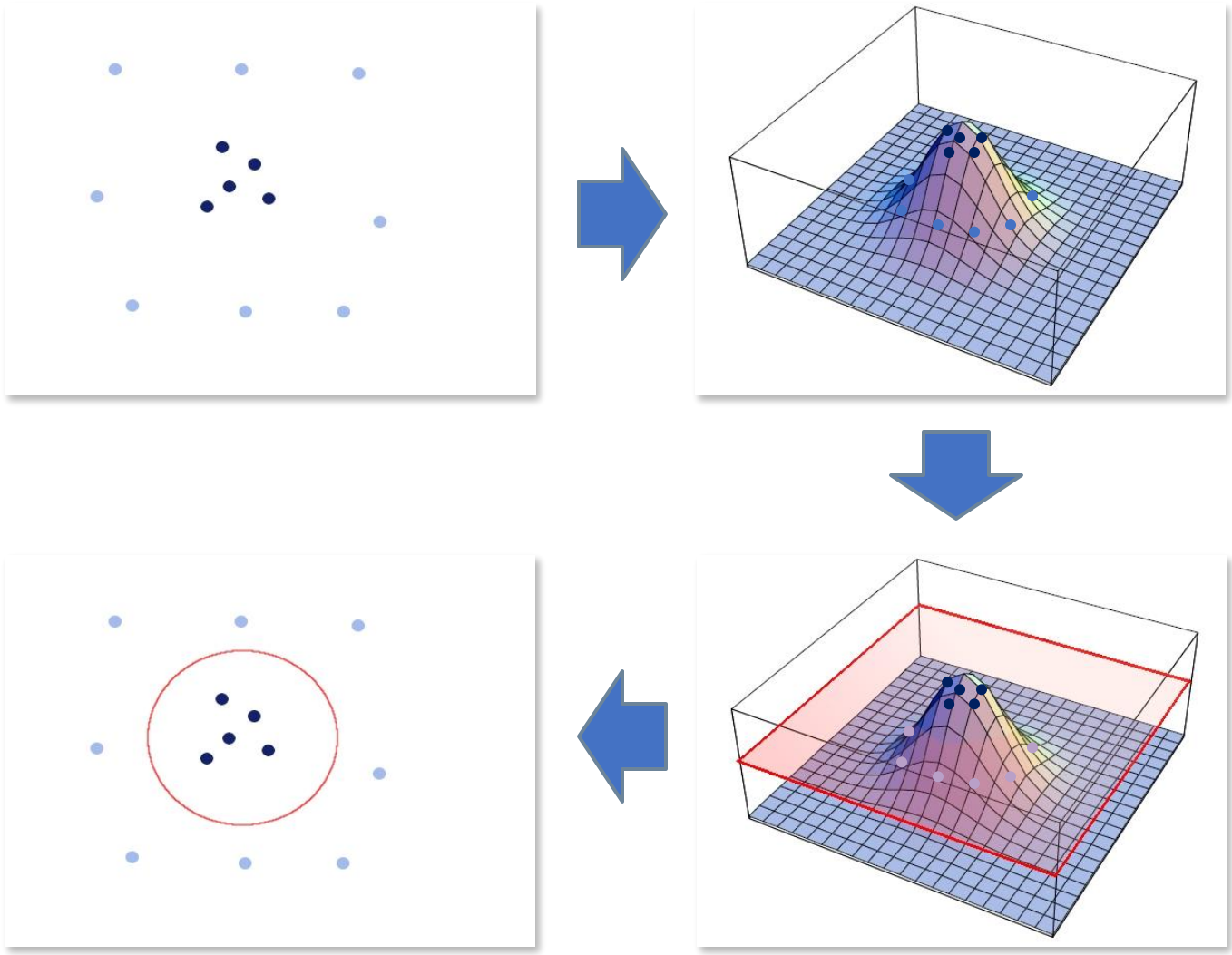


$$\min_{\omega,\gamma\neq 0} \frac{\| A\omega - e\gamma \|^2}{\| B\omega - e\gamma \|^2}$$

O. Mangasarian, E. Wild Multisurface Proximal Support Vector Classification via Generalized Eigenvalues. IEEE PAMI 2006.

The kernel trick

- To obtain greater separation between classes, nonlinearly embed points into a higher dimensional space



The kernel trick for ReGEC

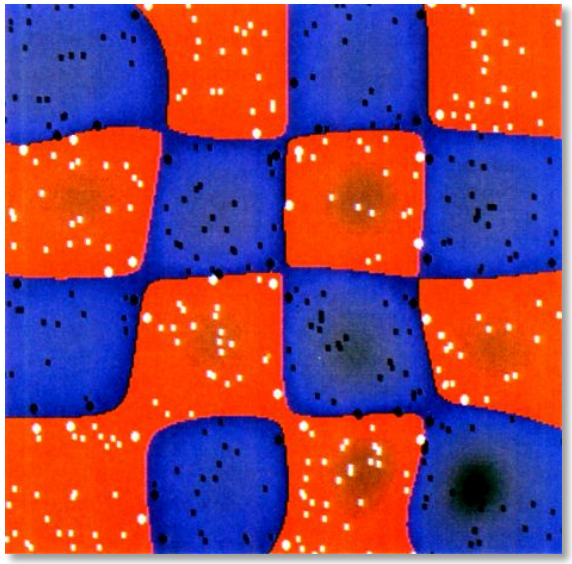
- The nonlinear embedding can be obtained with a *RBF kernel function*:

$$K(x_i, x_j) = e^{-\frac{|x_i - x_j|^2}{\sigma}}$$

- Each element of kernel matrix is:

$$K(A,\Gamma)_{ij} = e^{-\frac{\|A_i - \Gamma_j\|^2}{\sigma}}$$

$\Gamma = [A^T \; B^T]^T$



- And the model becomes:

$$\min_{u,\gamma \neq 0} \frac{\| K(A,\Gamma)u - e\gamma \|^2}{\| K(B,\Gamma)u - e\gamma \|^2}$$

ReGEC

$$\min_{u,\gamma\neq 0} \frac{\| K(A,\Gamma)u - e\gamma \|^2}{\| K(B,\Gamma)u - e\gamma \|^2} = \min_{u,\gamma\neq 0} \frac{\| [K(A,\Gamma) \quad -e]^T [u' \quad \gamma]' \|^2}{\| [K(B,\Gamma) \quad -e]^T [u' \quad \gamma]' \|^2}$$

- Let

$$G = [K(A,\Gamma) \quad -e]^T [K(A,\Gamma) \quad -e],$$
$$H = [K(B,\Gamma) \quad -e]^T [K(B,\Gamma) \quad -e],$$
$$z = [u' \quad \gamma]'.$$

- the equation becomes:

$$\min_{z\in R^{n+1}} \frac{z'Gz}{z'Hz}$$

- Rayleigh quotients of $Gz = \lambda Hz$.

Tikhonov Regularization

- Mangasarian et al. generate the two proximal surfaces:

$$K(x',\Gamma)u_1 - \gamma_1 = 0 \qquad K(x',\Gamma)u_2 - \gamma_2 = 0$$

- Solving the regularized problem:

$$\min_{u,\gamma \neq 0} \frac{\| K(A,\Gamma)u - e\gamma \|^2 + \delta \| [u^T \quad \gamma]^T \|^2}{\| K(B,\Gamma)u - e\gamma \|^2}$$

$$\min_{u,\gamma \neq 0} \frac{\| K(B,\Gamma)u - e\gamma \|^2 + \delta \| [u^T \quad \gamma]^T \|^2}{\| K(A,\Gamma)u - e\gamma \|^2}$$

Approximate Regularization

- A different regularization approach is:

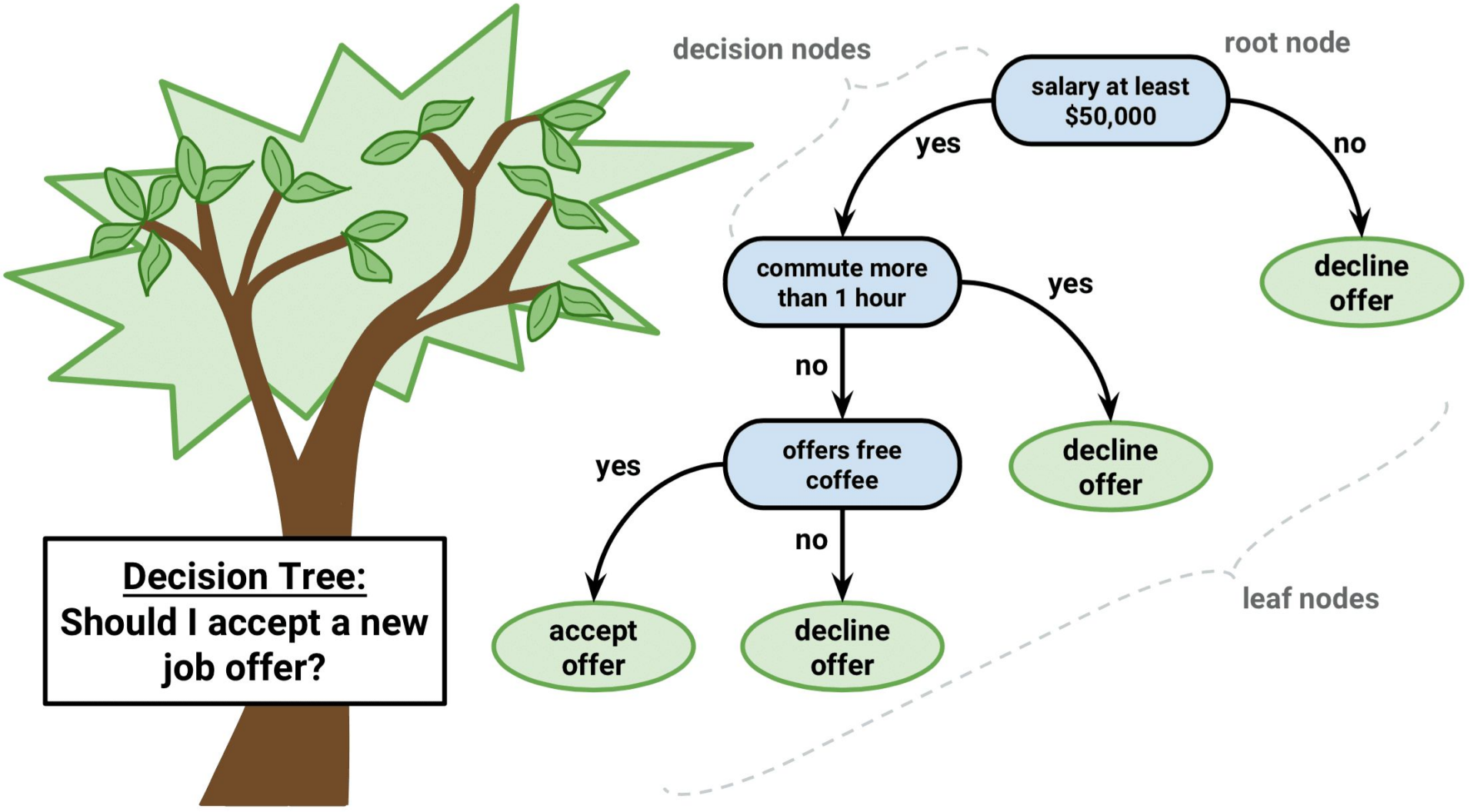
$$\min_{u,\gamma\neq 0} \frac{\| K(A,\Gamma)u - e\gamma \|^2 + \delta \| \tilde{K}_B u - e\gamma \|^2}{\| K(B,\Gamma)u - e\gamma \|^2 + \delta \| \tilde{K}_A u - e\gamma \|^2}$$

- $\tilde{K}_A$ and $\tilde{K}_B$ main diagonals of $K(A, \Gamma)$ and $K(B, \Gamma)$
- This approach halves the execution time, still providing similar accuracy results
- Regularization is a form of *robustification*

M.R. Guarracino, C. Cifarelli, O. Seref, P. Pardalos. A Classification Method Based on Generalized Eigenvalue Problems, OMS, 2007.

Decision trees & Random Forest

Decision tree



Decision tree

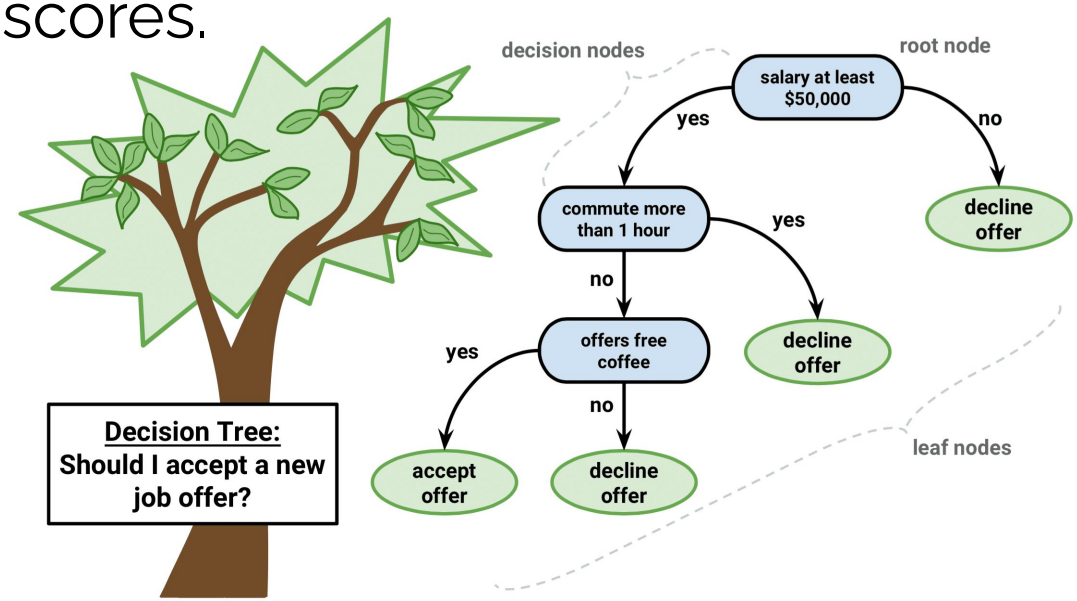
- A decision tree uses a tree structure to specify sequences of decisions and consequences.
- Given the set of input variables $X=\{x_1,x_2,\ldots, x_n\}$, the goal is to predict a response or output variable Y.

Decision tree

- The prediction can be achieved by constructing a decision tree with test points and branches.
- At each test point, a decision is made to pick a specific branch and traverse down the tree.
- Eventually, a final point is reached, and a prediction on the class can be made.
- Each test point in a decision tree involves testing a particular input variable (or attribute), and each branch represents the decision being made.
- Due to its flexibility and easy visualization, decision trees are commonly deployed for classification purposes.

Decision tree structure

- The input values can be categorical or continuous.
- A decision tree employs a structure of test points (***nodes***) and branches, i.e. the decision being made.
- A node without further branches is a ***leaf node***.
- The leaf nodes return class labels and, in some cases, the probability scores.



Decision tree structure

- A decision tree can be converted into a set of decision rules.
- In the following example rule, *income* and *mortgage_amount* are input variables, and the response is the output variable *default* with a probability score.

```
IF income < $50,000 AND mortgage_amount > $100K
THEN default = True WITH PROBABILITY 75%
```

Classification vs regression trees

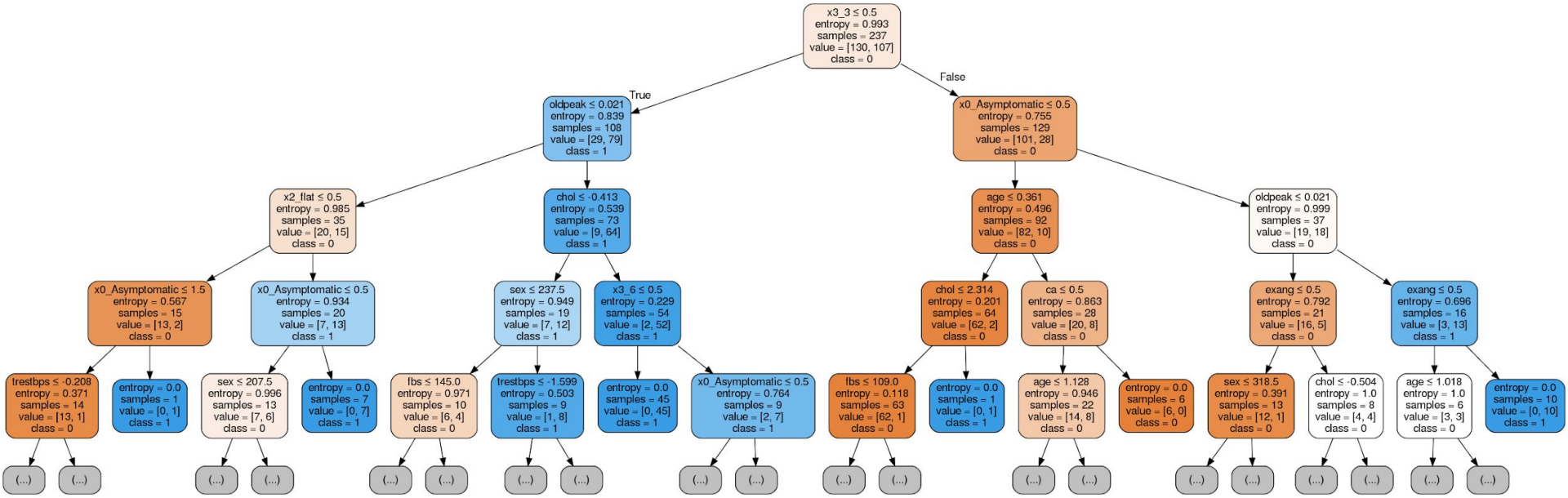
- Decision trees have two varieties: classification trees and regression trees.
- Classification trees usually apply to output variables that are categorical — often binary — in nature, such as yes or no, purchase or not purchase, and so on.
- Regression trees, on the other hand, can apply to output variables that are numeric or continuous, such as the predicted price of a consumer good or the likelihood a subscription will be purchased.

Visual representation

- Decision trees can be applied to a variety of situations.
- They can be easily represented in a visual way, and the corresponding decision rules are quite straightforward.
- Additionally, because the result is a series of logical if-then statements, there is no underlying assumption of a linear (or nonlinear) relationship between the input variables and the response variable.

Example of a Decision Tree

- An example of using a decision tree to predict whether customers will buy a product.



Decision trees structure

- *Internal nodes* are the decision or test points. Each internal node refers to an input variable or an attribute. The top internal node is called the ***root***.
- The decision tree shown is a binary tree: each internal node has two branches (splits).
- Sometimes decision trees may have more than two branches stemming from a node.
  - For example, a categorical variable with >2 modalities.

Decision trees structure

- The ***depth*** of a node is the minimum number of steps required to reach the node from the root.
- ***Leaf nodes*** are at the end of the last branches on the tree. They represent class labels—the outcome of all the prior decisions.
- The path from the root to a leaf node contains a series of decisions made at various internal nodes.

The UCI Cleveland dataset

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	Diagnosis_CHD
0	63.0	1.0	Typical	145.0	233.0	1.0	LVH	150.0	0.0	2.3	down	0.0	6.0	0
1	67.0	1.0	Asymptomatic	160.0	286.0	0.0	LVH	108.0	1.0	1.5	flat	3.0	3.0	1
2	67.0	1.0	Asymptomatic	120.0	229.0	0.0	LVH	129.0	1.0	2.6	flat	2.0	7.0	1
3	37.0	1.0	NonAnginal	130.0	250.0	0.0	Normal	187.0	0.0	3.5	down	0.0	3.0	0
4	41.0	0.0	Atypical	130.0	204.0	0.0	LVH	172.0	0.0	1.4	up	0.0	3.0	0
5	56.0	1.0	Atypical	120.0	236.0	0.0	Normal	178.0	0.0	0.8	up	0.0	3.0	0
6	62.0	0.0	Asymptomatic	140.0	268.0	0.0	LVH	160.0	0.0	3.6	down	2.0	3.0	1
7	57.0	0.0	Asymptomatic	120.0	354.0	0.0	Normal	163.0	1.0	0.6	up	0.0	3.0	0
8	63.0	1.0	Asymptomatic	130.0	254.0	0.0	LVH	147.0	0.0	1.4	flat	1.0	7.0	1
9	53.0	1.0	Asymptomatic	140.0	203.0	1.0	LVH	155.0	1.0	3.1	down	0.0	7.0	1
10	57.0	1.0	Asymptomatic	140.0	192.0	0.0	Normal	148.0	0.0	0.4	flat	0.0	6.0	0
11	56.0	0.0	Atypical	140.0	294.0	0.0	LVH	153.0	0.0	1.3	flat	0.0	3.0	0
12	56.0	1.0	NonAnginal	130.0	256.0	1.0	LVH	142.0	1.0	0.6	flat	1.0	6.0	1
13	44.0	1.0	Atypical	120.0	263.0	0.0	Normal	173.0	0.0	0.0	up	0.0	7.0	0
14	52.0	1.0	NonAnginal	172.0	199.0	1.0	Normal	162.0	0.0	0.5	up	0.0	7.0	0

The general algorithm

- The objective of a decision tree algorithm is to construct a tree T from a training set S.
- If all the records in S belong to some class C (Diagnosis_CHD = 1, for example), or if S is sufficiently pure (greater than a preset threshold), then that node is considered a leaf node and assigned the label C.
- The purity of a node is defined as its probability of the corresponding class.
- For example, the root has $P(\text{Diagnosis\_CHD} = 1) = 1 - 108/237 = 45.57\%$ therefore, the root is only 45.57% pure on the Diagnosis_CHD = 1 class.

The general algorithm

- In contrast, if not all the records in S belong to class C or if S is not sufficiently pure, the algorithm selects the next most informative attribute A (chol, fbs, and so on) and partitions S according to A's values.
- The algorithm constructs $T_1, T_2, \ldots$, for the subsets of S recursively until one of the following criteria is met:
  - All the leaf nodes in the tree satisfy the minimum purity threshold.
  - The tree cannot be further split with the preset minimum purity threshold.
  - Any other stopping criterion is satisfied (such as the maximum depth of the tree).

The general algorithm

- The first step in constructing a decision tree is to choose the most informative attribute.
- A common way to identify the most informative attribute is to use entropy-based methods.
  - Used by algorithms such as ID3 and C4.5.
- The entropy methods select the most informative attribute based on two basic measures:
  - *Entropy*, which measures the *impurity* of an attribute
  - *Information gain*, which measures the *purity* of an attribute

Definition of entropy

- Given a dataset divided in classes, each with label $x \in X$, let $P(x)$ be the frequentist probability of class x.
- The entropy H_X of the classes is defined as:

$$H_X = -\sum_{\forall x \in X} P(x) \log_2 P(x)$$

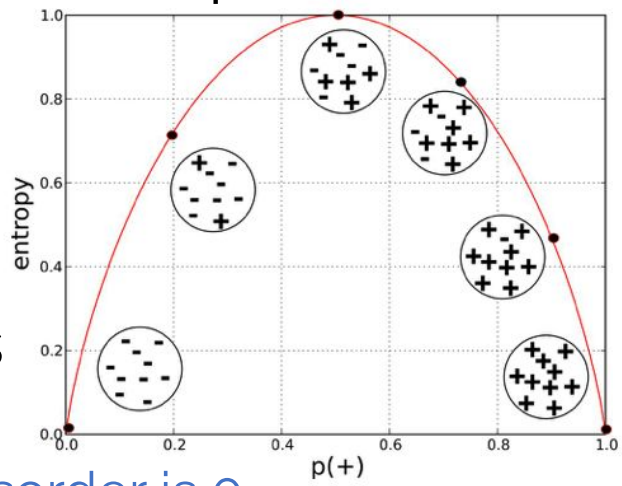
- Example: a dataset with 100 units, 30 in class x_1 and 70 in the other x_2, $P(x_1) = 30/100$ and $P(x_2) = 70/100$. The entropy is:

$$-\frac{30}{100} \times \log_2\left(\frac{30}{100}\right) - \frac{70}{100} \times \log_2\left(\frac{70}{100}\right) \approx .88$$

  - For two classes of equal size, $P(x_1) = P(x_2) = 0.5$, the entropy is 1.

Entropy of a random binary variable

- Consider tossing a coin with known, not necessarily fair, probabilities of coming up heads (+) or tails (-).
- The x-axis measures the proportion of data points belonging to the positive class and the y-axis axis measures their respective entropies.



- Entropy is lowest at the extremes, when there are no positive instances or only positive instances.
  - That is, when the mix is pure, and the disorder is 0.
- Entropy is highest in the middle: even split between positive and negative instances.
  - Extreme disorder , because there is no majority.

UCI Cleveland heart dataset

- Since P(Diagnosis_CHD = 1) = 0.4557 and P(Diagnosis_CHD = 0) = 0.5443, the entropy is:

- $H_{\text{Diagnosis\_CHD}}$ =
  $-0.4557 \times \log_2 0.4557 - 0.5443 \times \log_2 0.5443 \approx 0.994$

Information gain

- Now we know that entropy measures disorder.
- We need a metric to measure the reduction of this disorder in our target variable/class given additional information.
  - This metric is called Information Gain, and it can be written as:

$$\text{InfoGain}_Y = H_X - H_{(X|Y)}$$

- The reduction of uncertainty about X given an additional piece of information Y about X.
- The greater the reduction in this uncertainty, the more information is gained about X from Y.

Continous variables

- A brute-force method is to consider every value of the continuous variable in the training data as a candidate split position.
  - This brute-force method is computationally inefficient.
- To reduce the complexity, the training records can be sorted, and the candidate splits can be identified by taking the midpoints between two adjacent sorted values.
- For example, if the ages consists of sorted values {30, 40, 60, 70} and the candidate splits are 35, 50, and 65.

Decision tree algorithms

- Multiple algorithms exist to implement decision trees, and the methods of tree construction vary with different algorithms.
- ID3 (or Iterative Dichotomiser 3) is one of the first decision tree algorithms, and it was developed by John Ross Quinlan.
- The ID3 algorithm may construct a deep and complex tree, which would cause overfitting

Decision trees algorithms: C4.5

- The C4.5 algorithm can handle missing data: if the training records contain unknown attribute values, the C4.5 evaluates the gain for an attribute by considering only the records where the attribute is defined.
- Both categorical and continuous attributes are supported by C4.5.
  - Values of a continuous variable are sorted and partitioned.
  - For the corresponding records of each partition, the gain is calculated, and the partition that maximizes the gain is chosen for the next split.
- The C4.5 algorithm uses a bottom-up technique called pruning to simplify the tree by removing the least visited nodes and branches.

Decision trees algorithms: CART

- CART (or Classification And Regression Trees) is often used as a generic acronym for the decision tree, although it is a specific implementation.
- Similar to C4.5, CART can handle continuous attributes.
- Instead of entropy-based criteria to rank tests, CART uses the Gini diversity index:

$$Gini_X = 1 - \sum_{\forall x \in X} P(x)^2$$

- CART constructs a sequence of subtrees, uses cross-validation to estimate the misclassification cost of each subtree, and chooses the one with the lowest cost.

Evaluating decision trees

- Decision trees use greedy algorithms, in that they always choose the option that seems the best available at that moment.
- At each step, the algorithm selects which attribute to use for splitting the remaining records.
- This selection may not be the best overall, but it is guaranteed to be the best at that step.
- This characteristic reinforces the efficiency of decision trees.

Random Forest

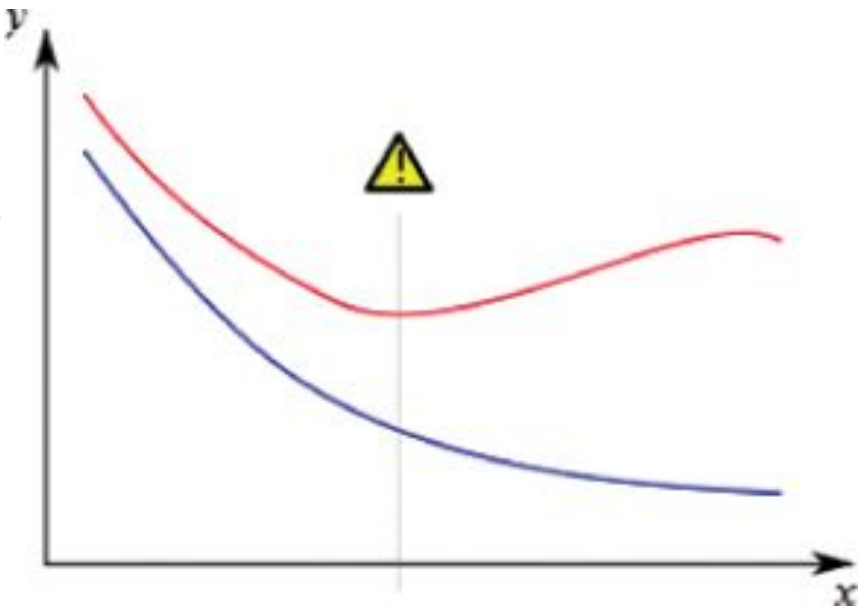
- However, once a bad split is taken, it is propagated through the rest of the tree.
- To address this problem, an ensemble technique (such as random forest) may randomize the splitting or even randomize data and come up with a multiple tree structure.
- These trees then vote for each class, and the class with the most votes is chosen as the predicted class.

Evaluating decision trees

- First, evaluate whether the splits of the tree make sense.
- Conduct sanity checks by validating the decision rules with domain experts, and determine if the decision rules are sound.
- Next, look at the depth and nodes of the tree.
- Having too many layers and obtaining nodes with few members might be signs of overfitting.
- In overfitting, the model fits the training set well, but it performs poorly on the new samples in the testing set.

Evaluating decision trees

- The figure illustrates the performance of an overfit model.
- The *x*-axis represents the amount of data, and the *y*-axis represents the errors.
- The blue curve is the training set, and the red curve is the testing set.
- The model predicts well on the testing set, but performs worse and worse on the testing set as more data is introduced.



Overfitting

- Overfitting can be caused by either the lack of training data or the biased data in the training set.
- Two approaches can help avoid overfitting in decision tree learning.
  - Stop growing the tree early before it reaches the point where all the training data is perfectly classified.
  - Grow the full tree, and then post-prune the tree with methods such as reduced-error pruning and rule-based post pruning.
- Last, many standard diagnostics tools that apply to classifiers can help evaluate overfitting.

Decision trees pros

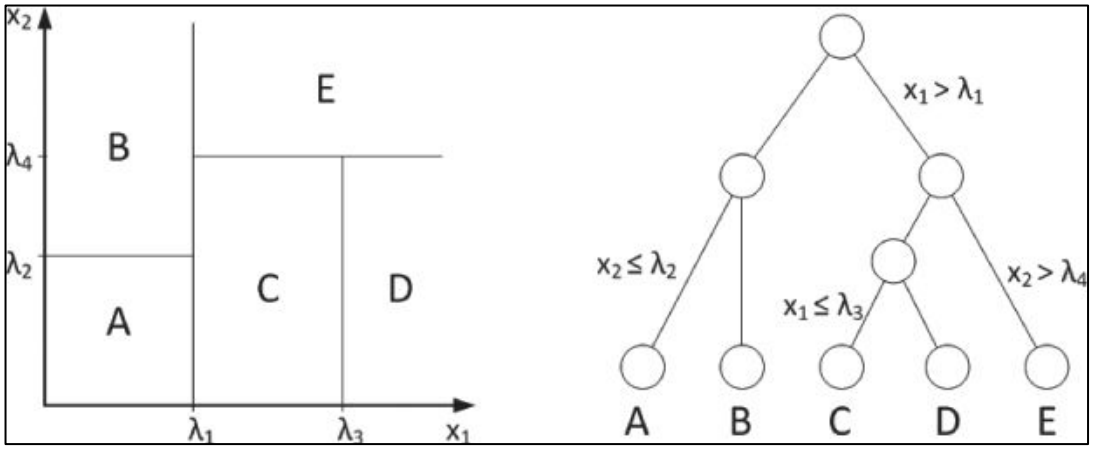
- Decision trees are computationally inexpensive, and it is easy to classify the data.
- The outputs are easy to interpret as a fixed sequence of simple tests.
- Many decision tree algorithms are able to show the importance of each input variable.
  - Basic measures, such as information gain, are provided by most statistical software packages.
- Decision trees are able to handle both numerical and categorical attributes and are robust with redundant or correlated variables.

Decision trees pros

- Decision trees can handle categorical attributes with many distinct values, such as country codes for telephone numbers.
- Decision trees can also handle variables that have a nonlinear effect on the outcome, so they work better than linear models (for example, linear regression and logistic regression) for highly nonlinear problems.
- Decision trees naturally handle variable interactions. Every node in the tree depends on the preceding nodes in the tree.

Decision trees cons

- In a decision tree, the decision regions are rectangular surfaces.



- A decision surface corresponds to a leaf node of the tree, and it can be reached by traversing from the root of the tree following by a series of decisions according to the value of an attribute.

Decision trees cons

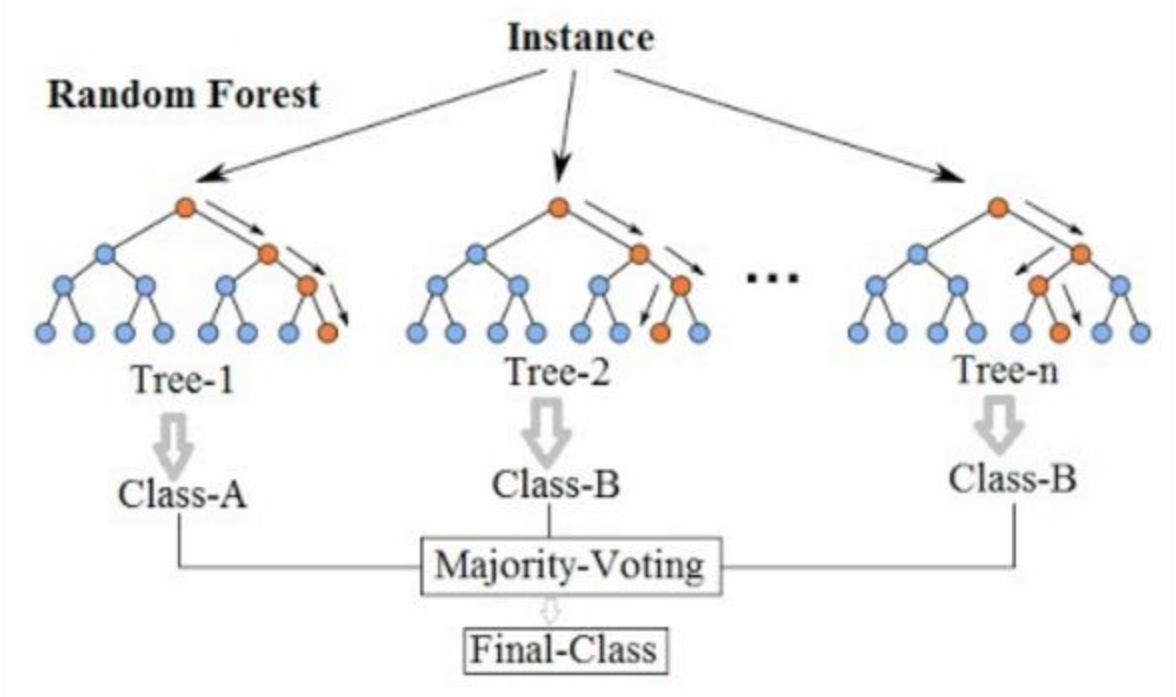
- The structure of a decision tree is sensitive to small variations in the training data.
- Although the dataset is the same, constructing two decision trees based on two different subsets may result in very different trees.
- If a tree is too deep, overfitting may occur, because each split reduces the training data for subsequent splits.
- Decision trees are not a good choice if the dataset contains many irrelevant variables.
  - This is different from being robust with redundant variables and correlated variables.

Decision trees cons

- If the dataset contains redundant variables, the resulting decision tree ignores all but one of these variables
  - The algorithm would not detect information gain by including more redundant variables.
- If the dataset contains irrelevant variables and if these variables are accidentally chosen as splits in the tree, the tree may grow too large.
  - It may end up with less data at each split, overfitting is likely to occur.
- To address this problem, feature selection can be introduced in the data preprocessing phase to eliminate the irrelevant variables.

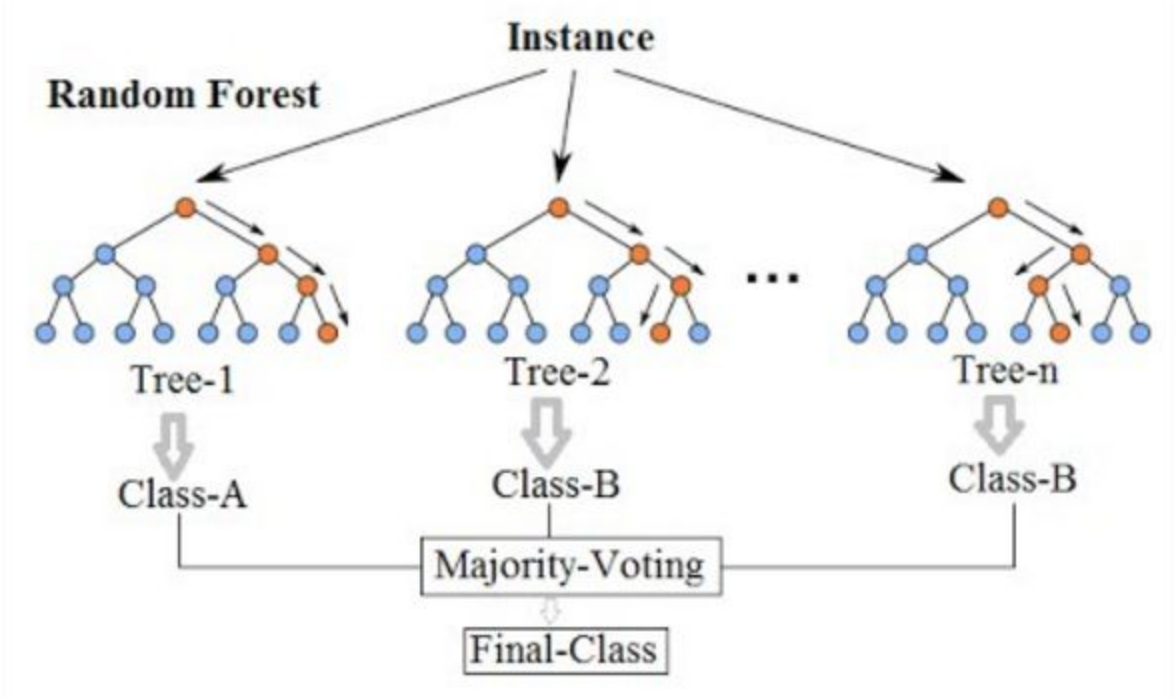
Random forest

- To overcome the issue of instability and potential overfitting of deep trees, one can combine the decisions of several randomized shallow decision trees.
  - These classifiers are called *random forest*.



Random forest

- To overcome the issue of instability and potential overfitting of deep trees, one can combine the decisions of several randomized shallow decision trees.
  - These classifiers are called *random forest*.



Diagnostics

Diagnostics of Classifiers

- So far, we talked about two classifiers: logistic regression and decision trees.
- These methods can be used to classify instances into distinct groups according to the similar characteristics they share.
- Each of these classifiers faces the same issue: how to evaluate whether they perform well.
- A few tools have been designed to evaluate the performance of a classifier.
- Such tools are not limited to the three classifiers in this book but rather serve the purpose of assessing classifiers in general.

Confusion matrix

- A table layout to visualize the performance of a classifier.

	Predicted class	
Actual class	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

  - *TP:* the number of instances correctly classified as positive.
  - *FP:* the number of negative instances classified as positive.
  - *TN:* the number of instances correctly classified as negative.
  - *FN:* the number of positive instances classified as negative.
- A good classifier should have large TP and TN and

Confusion matrix

- In the Cleveland example, the test set includes 60 instances.
- This table shows the confusion matrix of a K-NN classifier on 60 patients to predict whether they would have angina.

	Predicted class		
Actual class	Positive	Negative	**Total**
Positive	27	3	30
Negative	12	18	30
Total	39	21	60

Confusion matrix

- Of the 11 clients who subscribed to the deposit, the model predicted 3 subscribed and 8 not subscribed.
- Of the 89 clients who did not subscribe, the model predicted 2 subscribed and 87 not subscribed.
- All correct guesses are located from top left to bottom right of the table.
  - It's easy to visually inspect the table for errors, because they are represented by nonzero off diagonal values.

	Predicted class		
Actual class	Positive	Negative	**Total**
Positive	27	3	30
Negative	12	18	30
Total	39	21	60

Accuracy

- The ***accuracy*** (or the ***overall success rate***) is a metric defining the rate at which a model has classified the records correctly. It is defined as the sum of TP and TN divided by the total number of instances:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

- A good model should have a high accuracy score, but having a high accuracy score alone does not guarantee the model is well established.
  - For example, in case of unbalanced classes.

TPR, FPR & FNR

- ***True positive rate*** (TPR): percentage of positive instances correctly identified.

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

- ***False positive rate*** (FPR): percentage of negatives marked as positive.
  - The FPR is also called the ***type I error rate*** .

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

- ***False negative rate*** (FNR): percentage of positives marked as negatives.
  - It is also known as the ***miss rate*** or ***type II error rate***

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

- Note that the sum of TPR and FNR is 1.

Key parameters

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% = \frac{3 + 87}{3 + 87 + 2 + 8} \times 100\% = 90\%$$

$$TPR\ (or\ Recall) = \frac{TP}{TP + FN} = \frac{3}{3 + 8} \approx 0.273$$

$$FPR = \frac{FP}{FP + TN} = \frac{2}{2 + 87} \approx 0.022$$

$$FNR = \frac{FN}{TP + FN} = \frac{8}{3 + 8} \approx 0.727$$

$$Precision = \frac{TP}{TP + FP} = \frac{3}{3 + 2} = 0.6$$

Performance evaluation and comparison of classifiers

Classifier validation

- If the training data are representative of population, we can predict how good will be the classification model in predicting the class of an unlabeled instance.
- Many different techniques can be used to predict the generalization capability of a classifier:
  - Independent test set
  - Hold outs
  - Cross-validation
  - Leave one out

Cross-validation methods

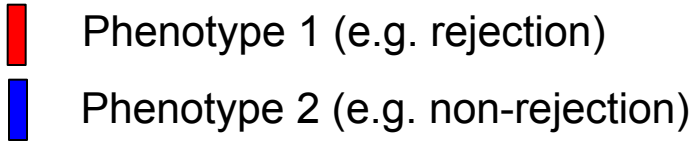
Single random data split

Multiple random splits

K-fold cross-validation

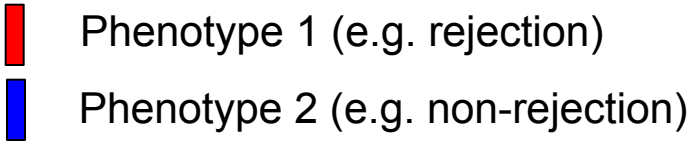
Leave-one-out cross-validation (LOOCV)

Training set

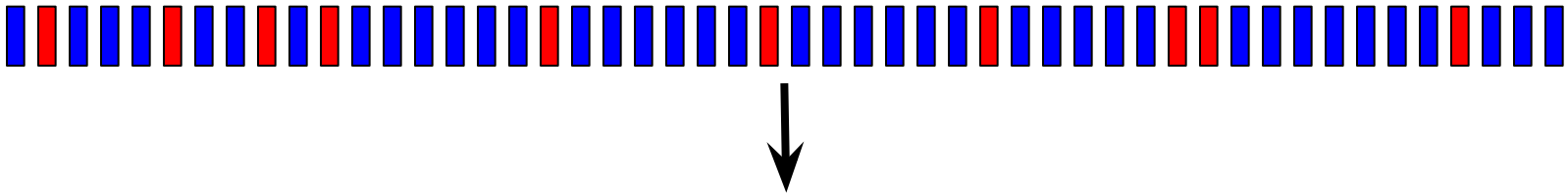


50 samples

Training set

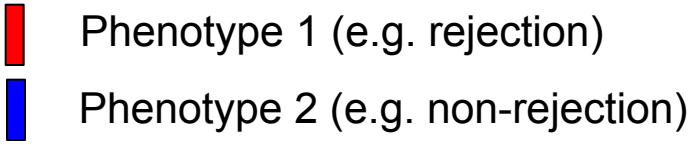


50 samples

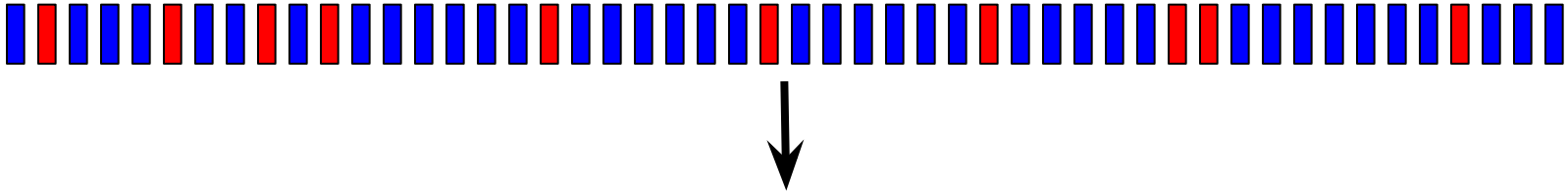


Build classifier:
Get a machine learning algorithm* to find patterns in the microarray data linking the phenotypes with the gene expression data

Training set

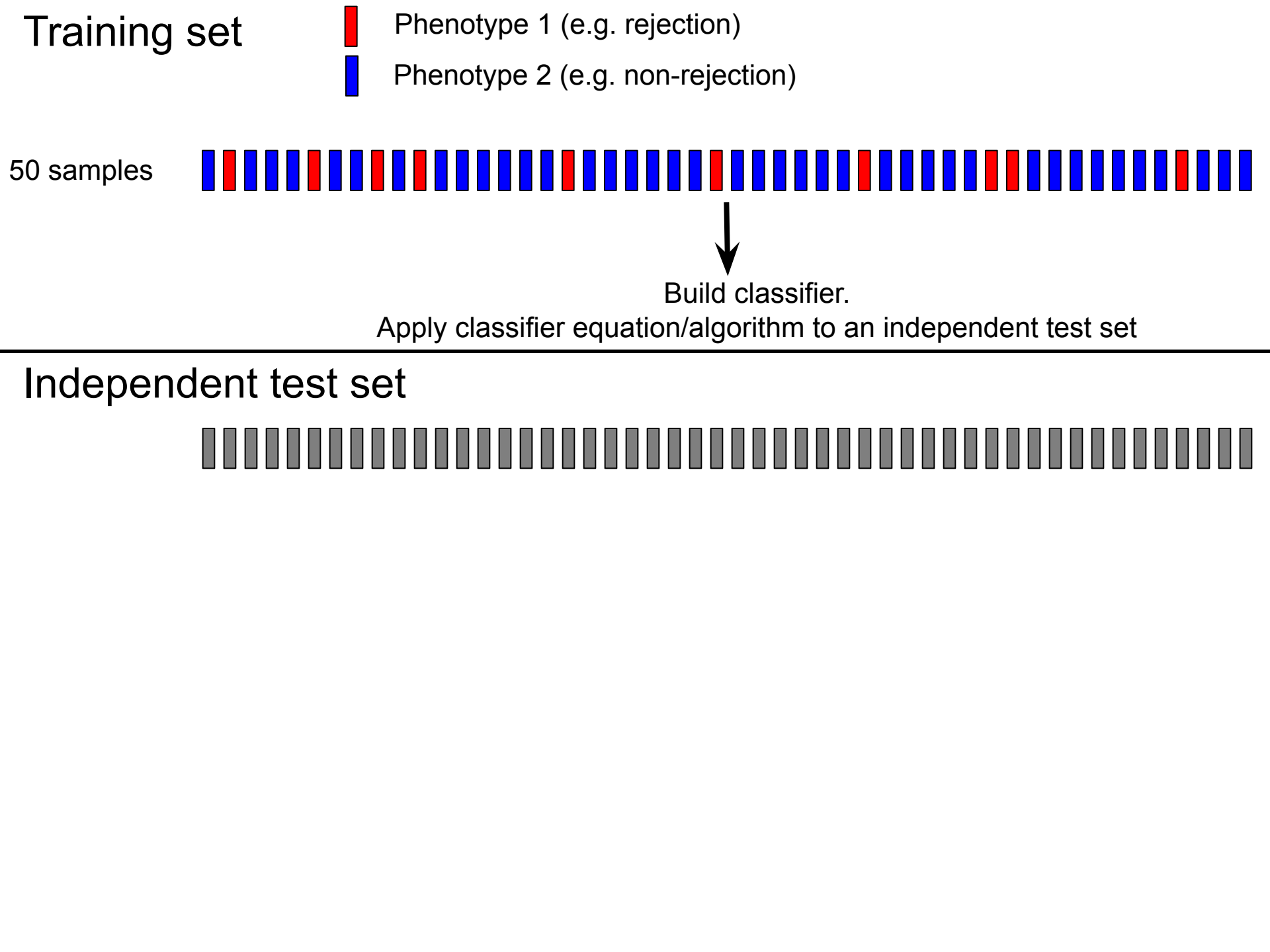


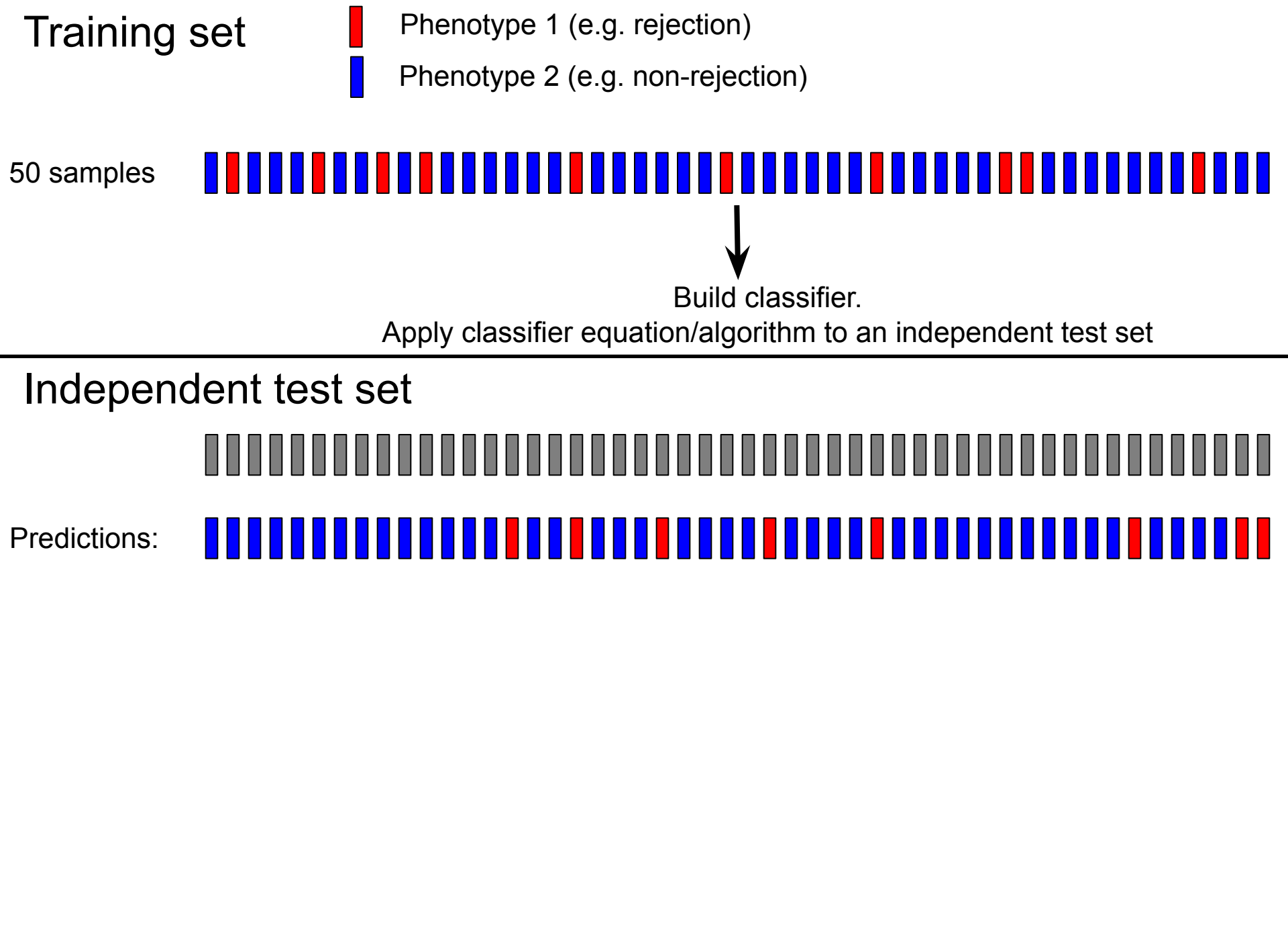
50 samples



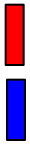
Build classifier:
Get a machine learning algorithm* to find patterns in the microarray data linking the phenotypes with the gene expression data

*e.g. support vector machines (SVM), linear discriminant analysis (LDA), neural networks (NN), predictive analysis of microarrays (PAM), *K*-nearest neighbours (*K*-NN), classification and regression trees (CART), random forests (RF), etc.


Training set
Phenotype 1 (e.g. rejection)
Phenotype 2 (e.g. non-rejection)
50 samples
Build classifier.
Apply classifier equation/algorithm to an independent test set
Independent test set


Training set
Phenotype 1 (e.g. rejection)
Phenotype 2 (e.g. non-rejection)
50 samples
Build classifier.
Apply classifier equation/algorithm to an independent test set
Independent test set
Predictions:

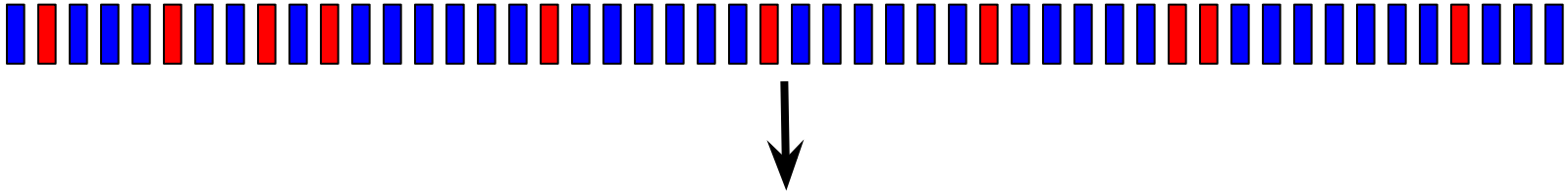
Training set



Phenotype 1 (e.g. rejection)

Phenotype 2 (e.g. non-rejection)

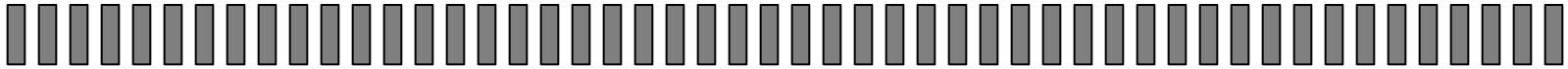
50 samples



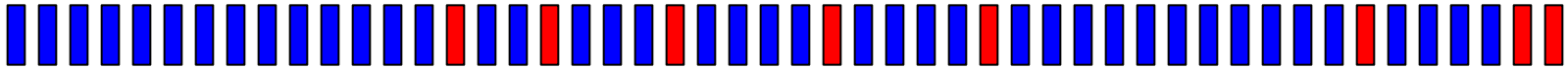
Build classifier.

Apply classifier equation/algorithm to an independent test set

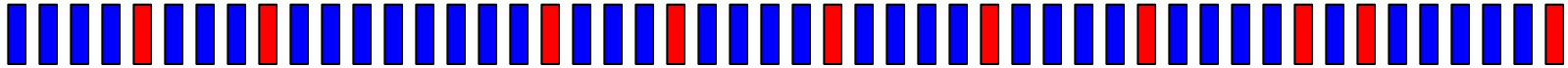
Independent test set



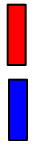
Predictions:



“Truth”:



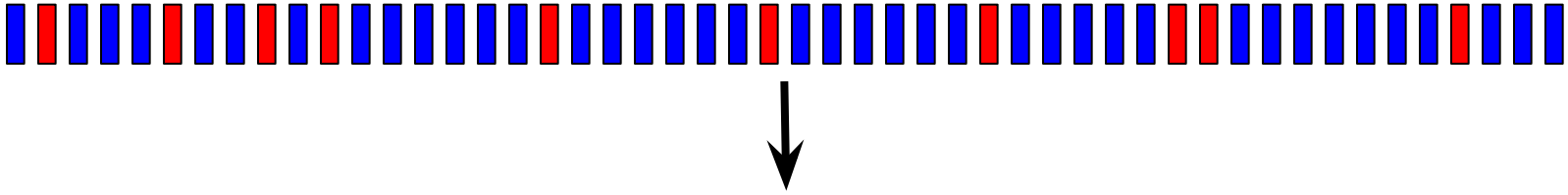
Training set



Phenotype 1 (e.g. rejection)

Phenotype 2 (e.g. non-rejection)

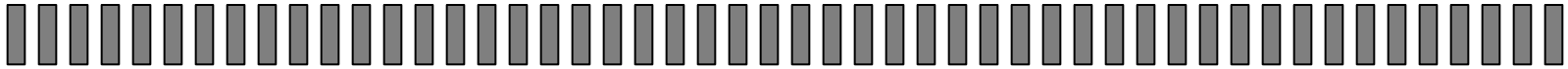
50 samples



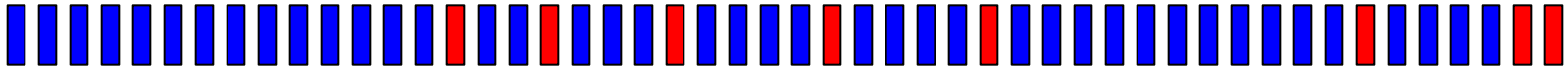
Build classifier.

Apply classifier equation/algorithm to an independent test set

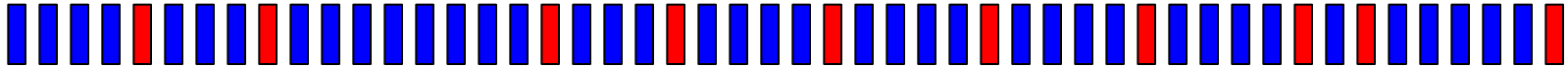
Independent test set



Predictions:



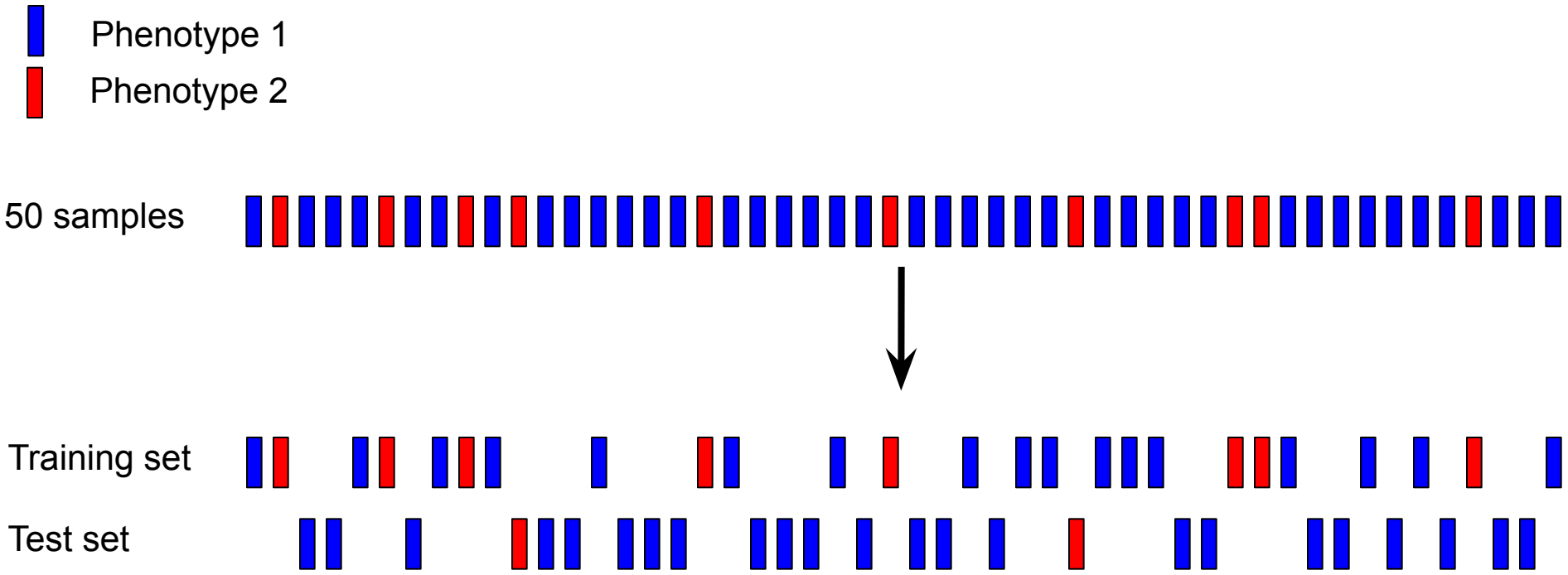
“Truth”:



Assess predictive value of the classifier:

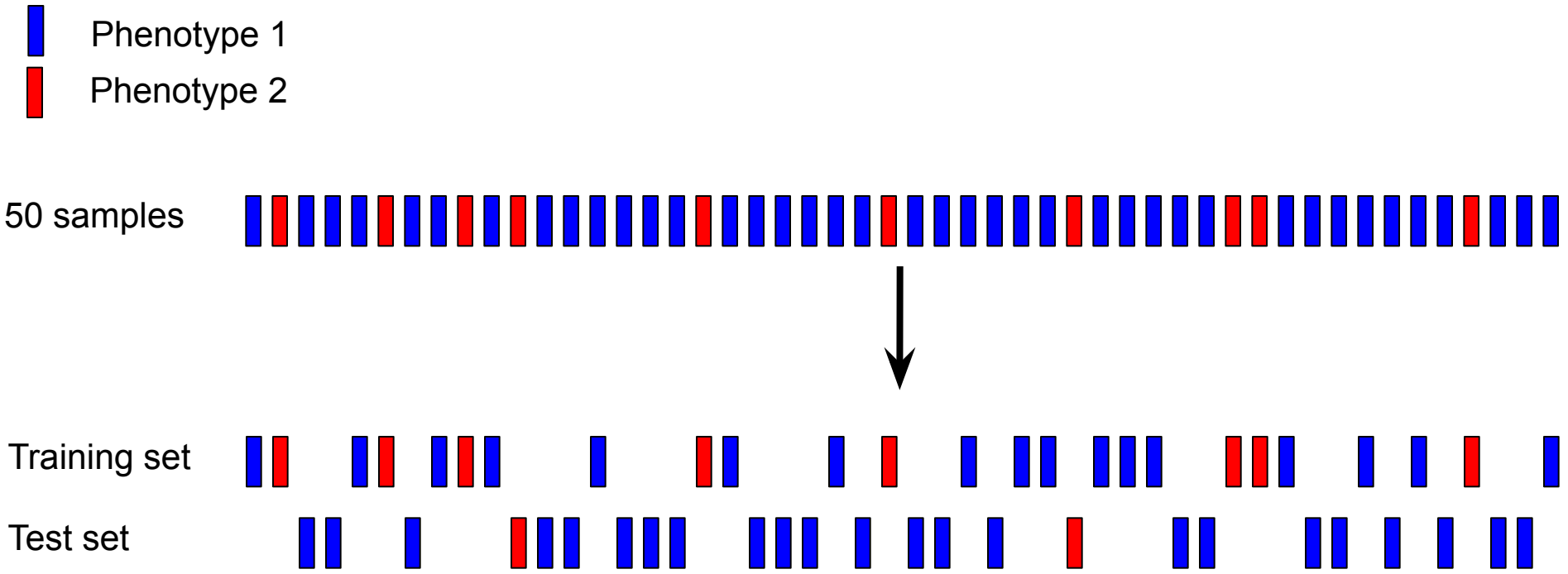
Accuracy = 88% Sensitivity = 60% Specificity = 95% PPV = 75% NPV = 90.5%

Single, random, training: test set split



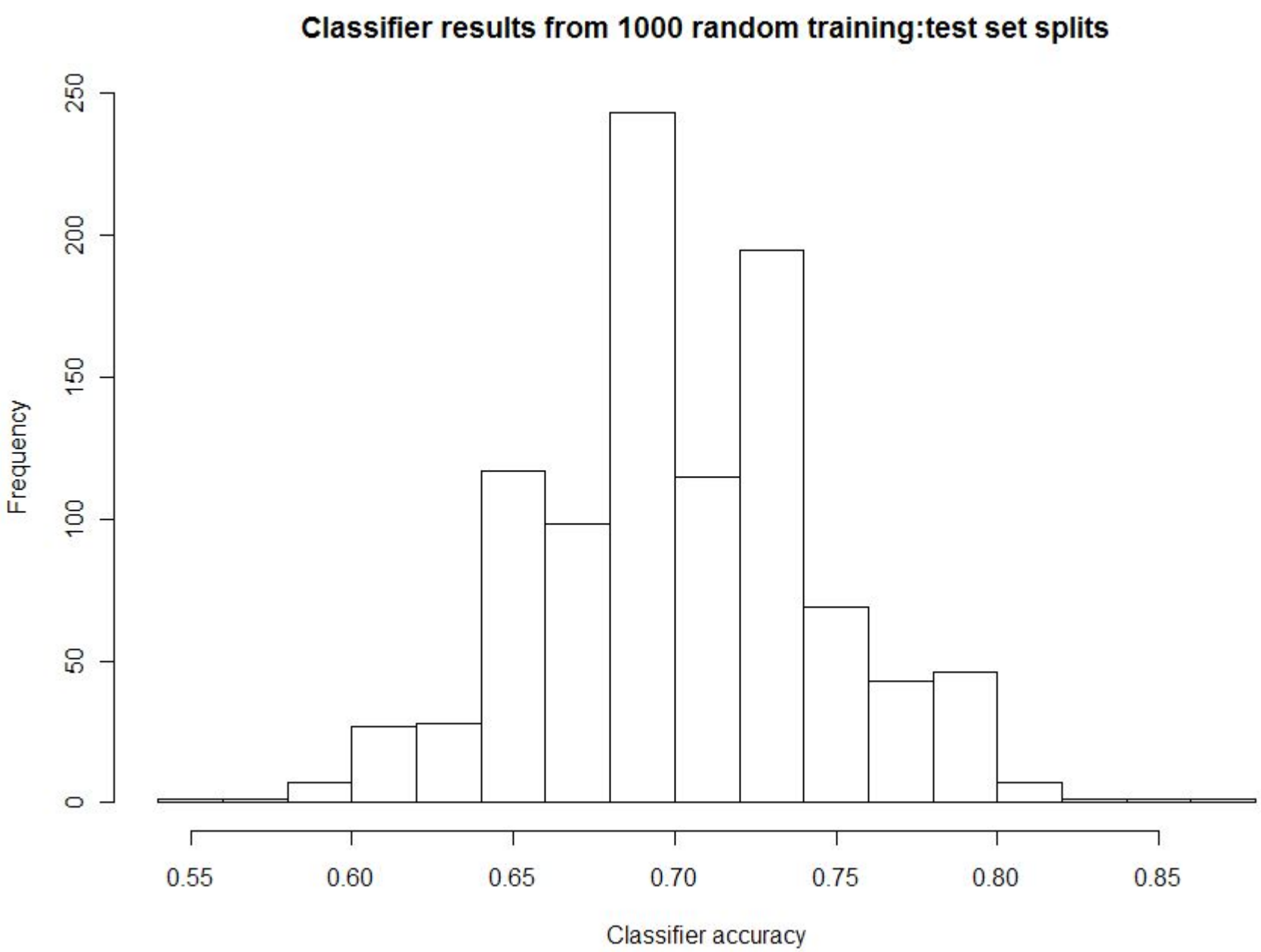
1) Build the classifier (equation to predict the phenotype) based on the training set
2) Apply this equation to the gene expression values in the test set to predict their phenotypes
3) Compare the predictions to the “true” phenotypes to assess accuracy, sensitivity, specificity etc. of the classifier

Single, random, training: test set split

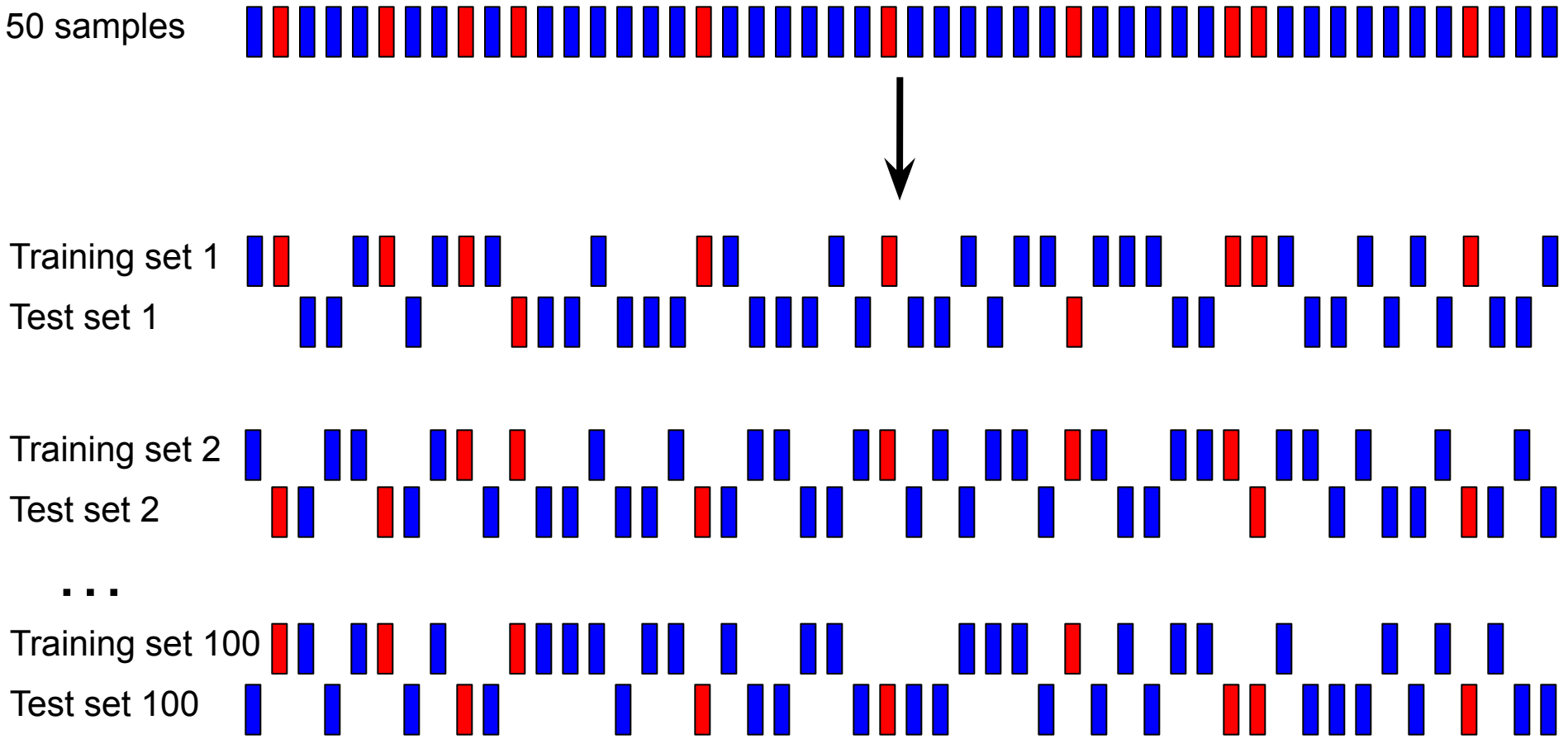


1) Build the classifier (equation to predict the phenotype) based on the training set
2) Apply this equation to the gene expression values in the test set to predict their phenotypes
3) Compare the predictions to the “true” phenotypes to assess accuracy, sensitivity, specificity etc. of the classifier

Problem: might get an “unlucky” split

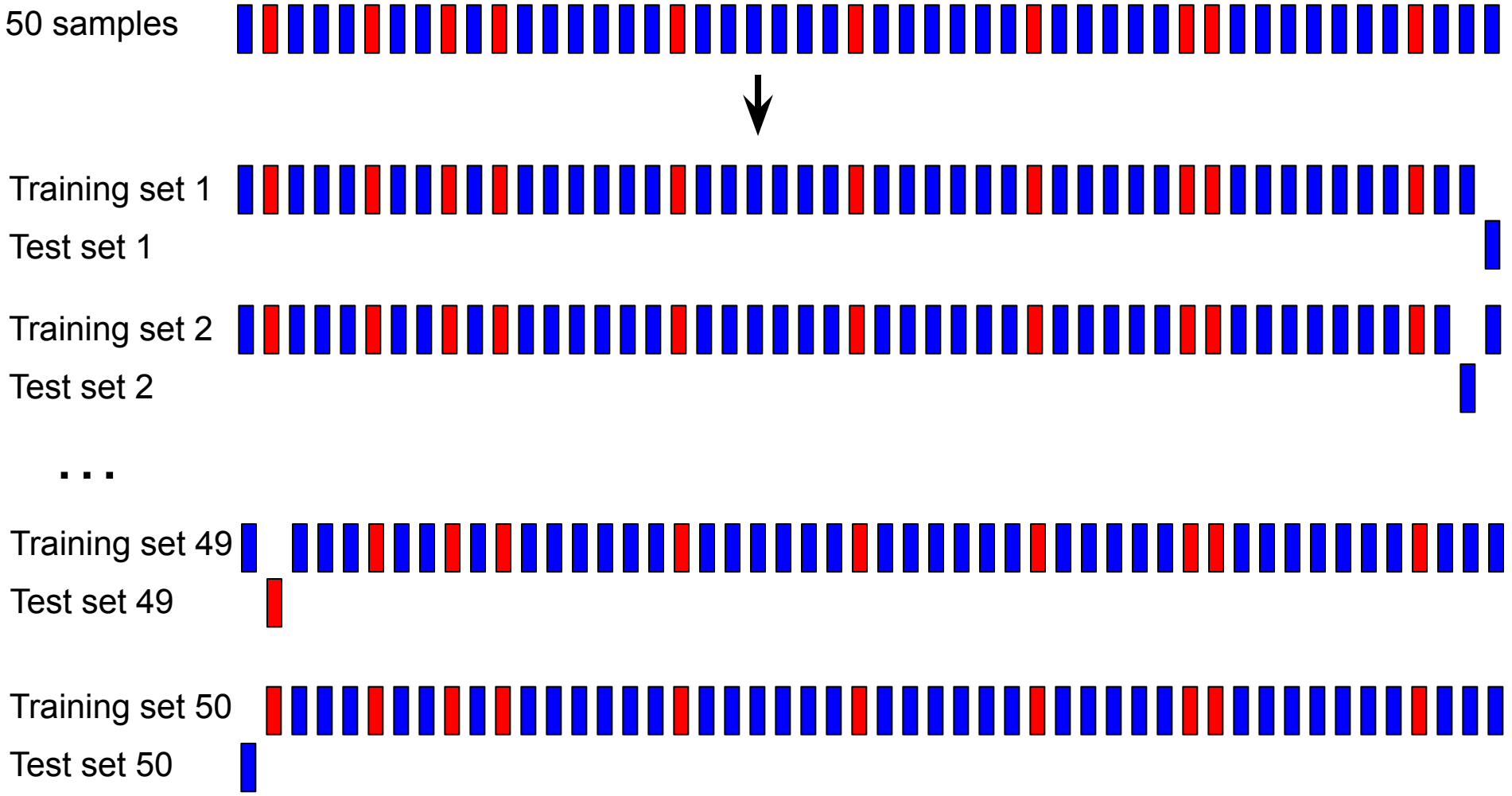

Classifier results from 1000 random training:test set splits
Frequency
0
50
100
150
200
250
0.55
0.60
0.65
0.70
0.75
0.80
0.85
Classifier accuracy

Multiple, random, training: test set splits



Store the predictions from each data split, and report the averages (and/or distributions)

Leave-out-one cross-validation



One prediction per sample. Large training set (n-1)