NATIONAL SUPERCOMPUTING MISSION

INFRASTRUCTURE | APPLICATIONS | R&D | HRD

Introduction to Machine Learning Algorithms

Pabitra Mitra

Indian Institute of Technology Kharagpur

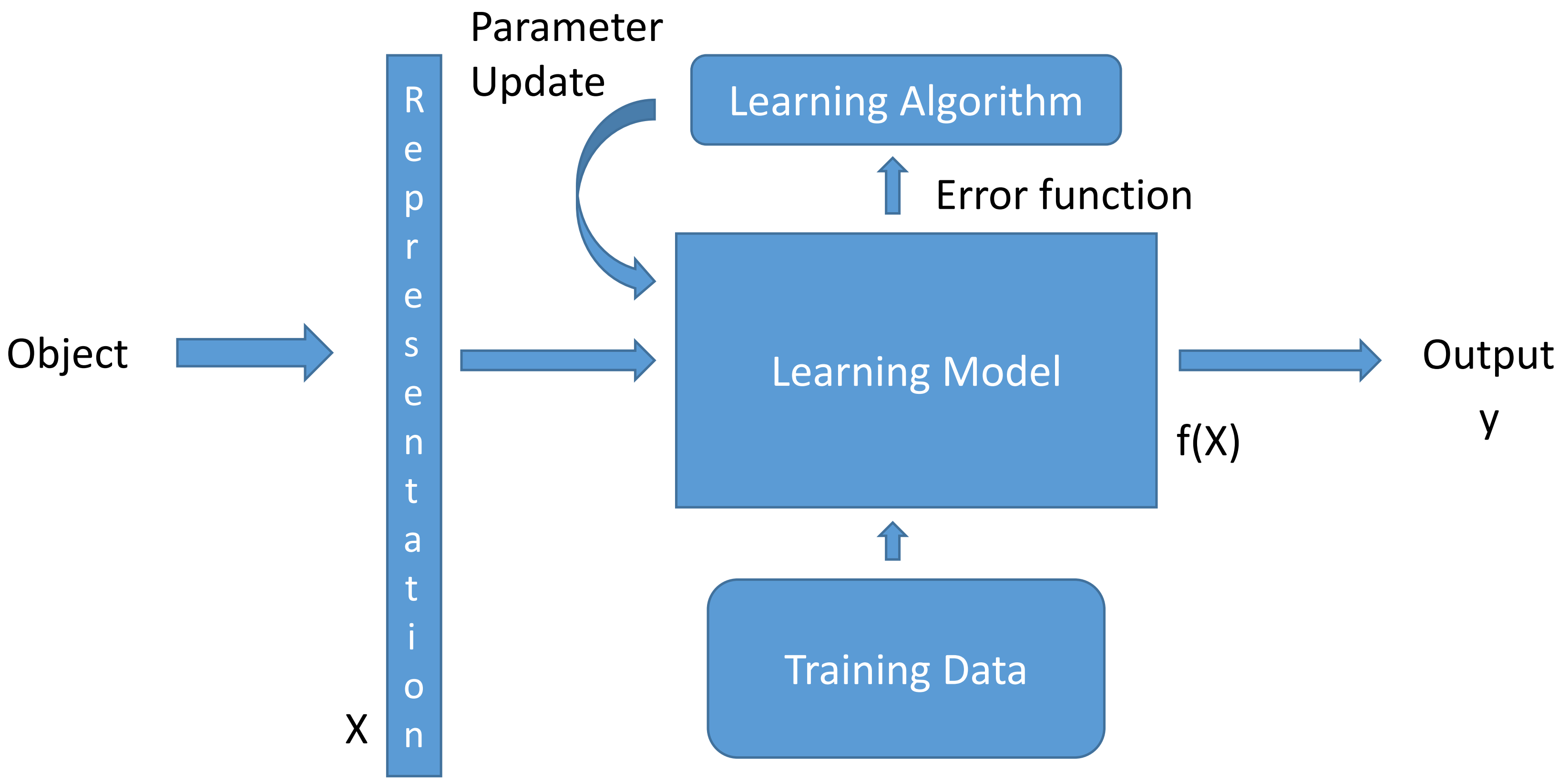
pabitra@cse.iitkgp.ac.in

NSM Workshop on Accelerated Data Science

Machine Learning

- Learning Algorithms/Systems: Performance improvement with experience, generalize to unseen input
- Example:
    - Face recognition
    - Email spam detection
    - Market segmentation
    - Rainfall forecasting
- Inductive inference – Data to Model

Machine Learning



Machine Learning Models

- Classification
  - Predicts category of input objects – predefined classes
  - Object recognition in images, email spam detection
- Regression
  - Predicts real valued output for a given input
  - Predicting value of a stock, predicting number of clicks in an advertisement
- Clustering
  - Groups objects into homogeneous clusters – clusters not predefined
  - Market segmentation, anomaly detection in industrial plants

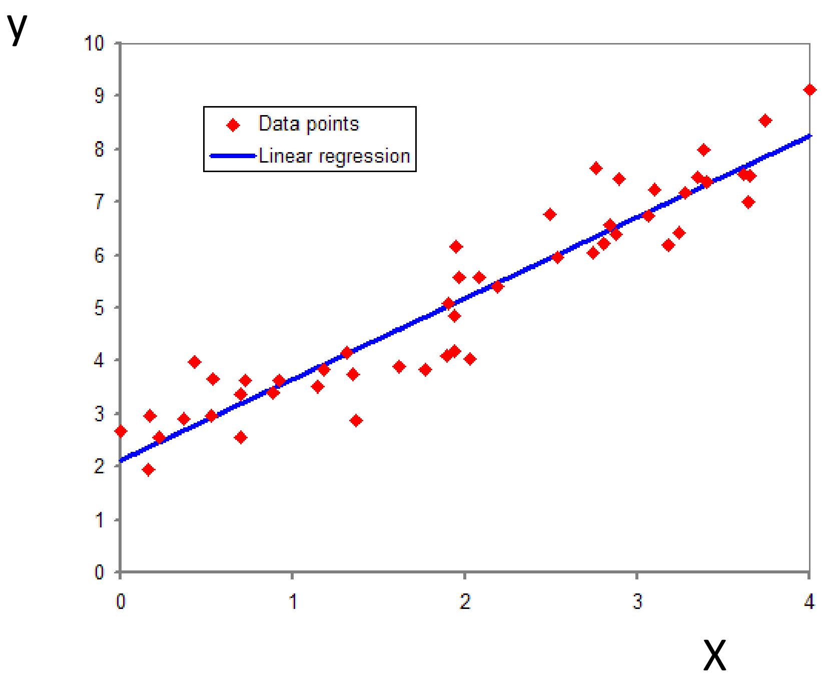
Learning Algorithms

- Supervised (predictive data analysis)
  - For each input in the training data the desired output is known
  - Previous history, ground truth, annotations, labels
- Unsupervised (explorative data analysis)
  - Output is not specified
  - Natural groups are to be determined
- Semi-supervised
  - Supervisory output available for few data points
  - Output not available for most data points

Examples of Machine Learning Models

- Classification and Regression
  - Logistic Regression
  - Bayesian learning
  - K-Nearest neighbor
  - Decision Tree
  - Support Vector Machine
  - Boosting – Random Forests, Xgboost
  - Neural Networks and Deep Learning
- Clustering
  - K-means clustering
  - Hierarchical clustering
  - DBSCAN

Linear Regression



Prediction Model: $y = f(X, \beta) + \varepsilon$

Linear Regression: $f(X, \beta) = \beta_0 + \beta_1 X + \varepsilon$

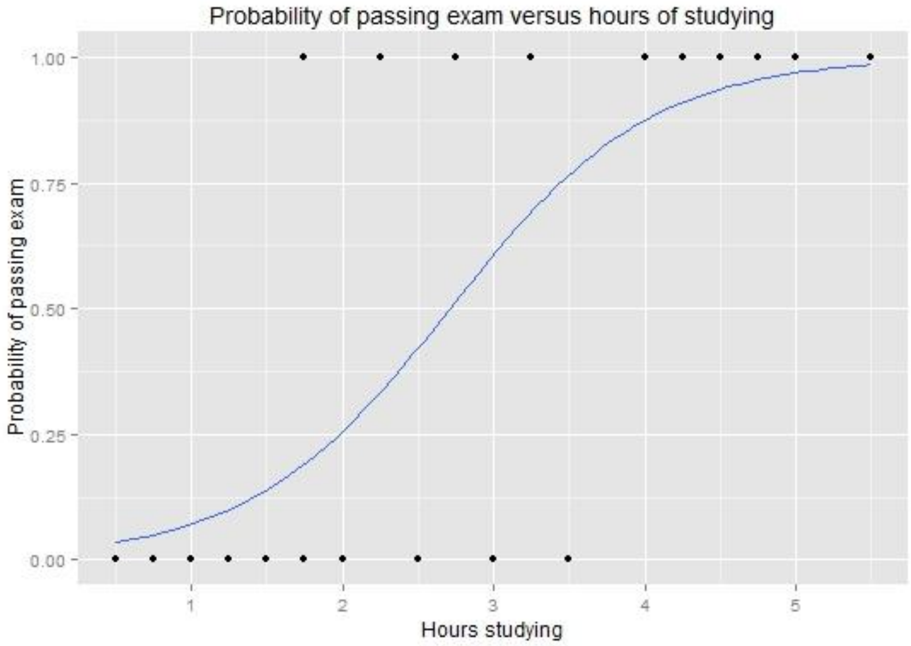
Find β that minimises the sum squared error

Logistic Regression: Binary Classification

Predict if a student will pass an exam depending on how many hours she has studied

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

Instead of modeling *y*, model $P(y = 1 \mid X) = p_X$


Probability of passing exam versus hours of studying

$$logit(p_X) = \log\left(\frac{p_X}{1-p_X}\right) = \beta_0 + \beta_1 X$$

$\log\left(\frac{p_X}{1-p_X}\right)$ is called the **logit** function

$p_X = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$ (Logistic function: inverse of logit)

$$\lim_{x\to-\infty} \frac{e^x}{1+e^x} = 0 \text{ and } \lim_{x\to\infty} \frac{e^x}{1+e^x} = 1, \text{ so } 0 \leq p_x \leq 1.$$

Predict class = 1, if $p_X > 1 - p_X$

Computing Parameters of Logistic Regression

β_0 :: *b*, β_1 :: *w*, *σ( )* :: sigmoid

$$P(y=1) = \sigma(w\cdot x+b)$$
$$= \frac{1}{1+e^{-(w\cdot x+b)}}$$

$$P(y=0) = 1-\sigma(w\cdot x+b)$$
$$= 1-\frac{1}{1+e^{-(w\cdot x+b)}}$$
$$= \frac{e^{-(w\cdot x+b)}}{1+e^{-(w\cdot x+b)}}$$

Find values of *w*, *b* that minimizes the cross entropy loss:

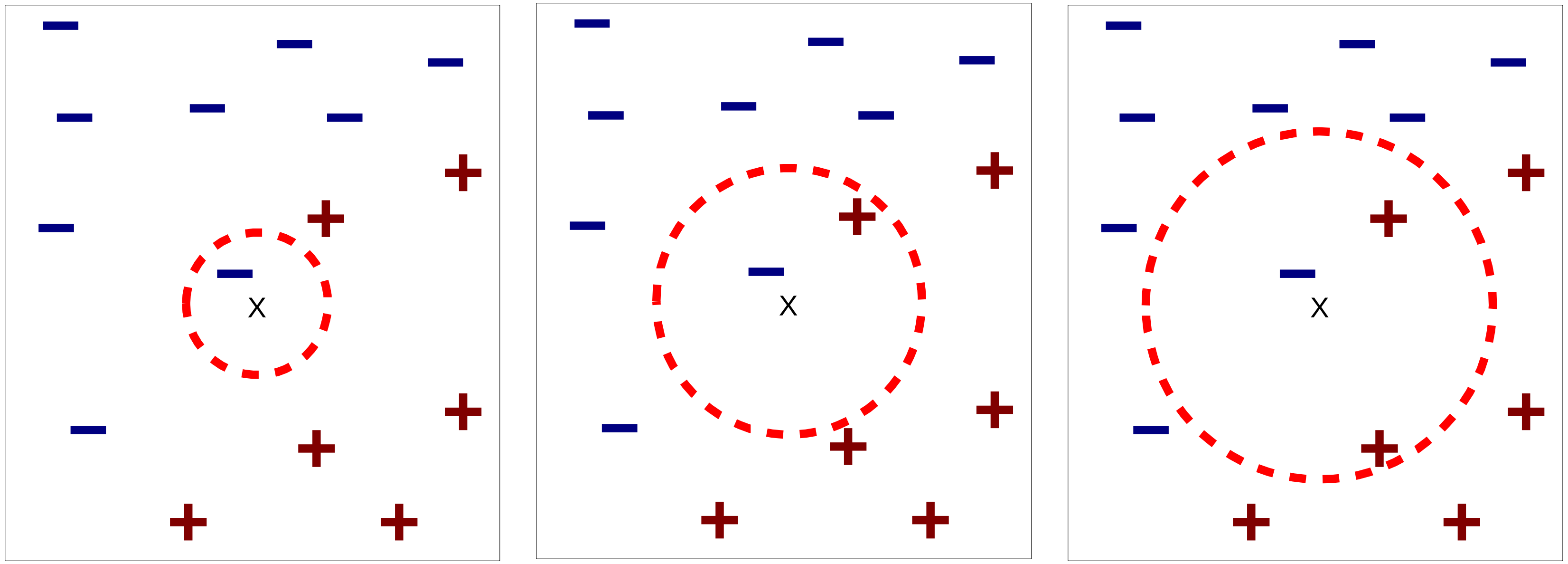
$$L_{CE}(w,b) = -[y\log\sigma(w\cdot x+b)+(1-y)\log(1-\sigma(w\cdot x+b))]$$

$$\frac{\partial L_{CE}(w,b)}{\partial w_j} = [\sigma(w\cdot x+b)-y]x_j$$

Difference between the model prediction and the correct answer y

Feature value for dimension j

K Nearest Neighbors

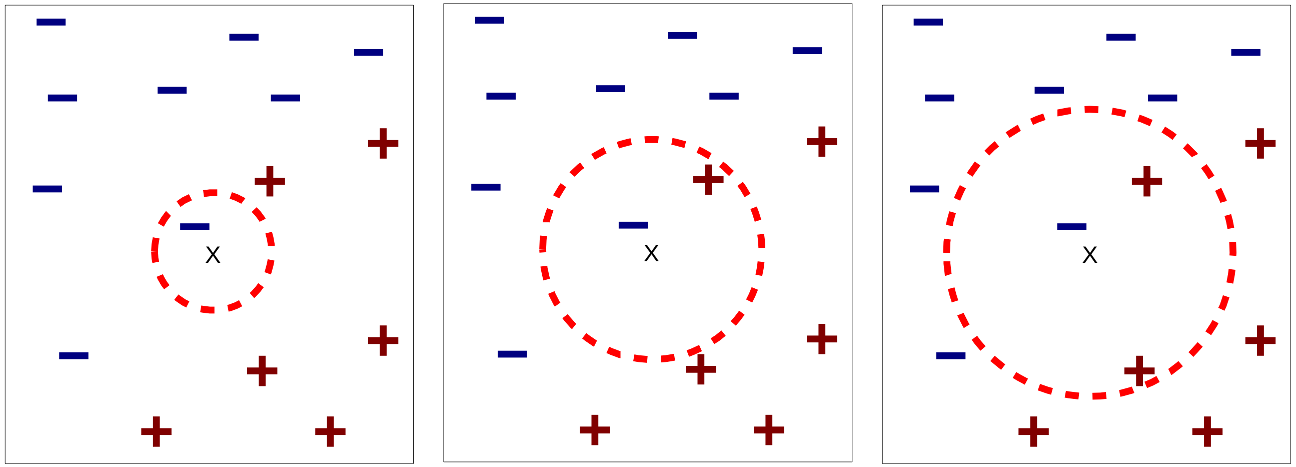


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of an input x are training data points that have the K smallest distance to x

K-Nearest Neighbor Classifier

- Find K-nearest neighbors of an input data
- Count class membership of the neighbors and find the majority class
- The majority class is the predicted class for the input



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Predicted class for x according to 3-NN rule is +

For K-NN regression predict the average value of the neighbors

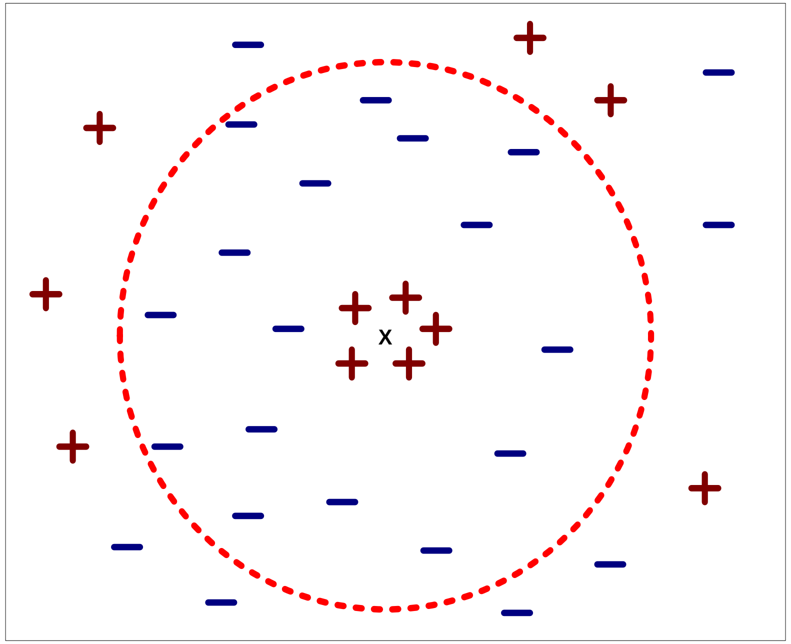
Nearest-Neighbor Classifiers: Design Choices

- The value of *k*, the number of nearest neighbors to retrieve
- Distance Metric to compute distance between data points

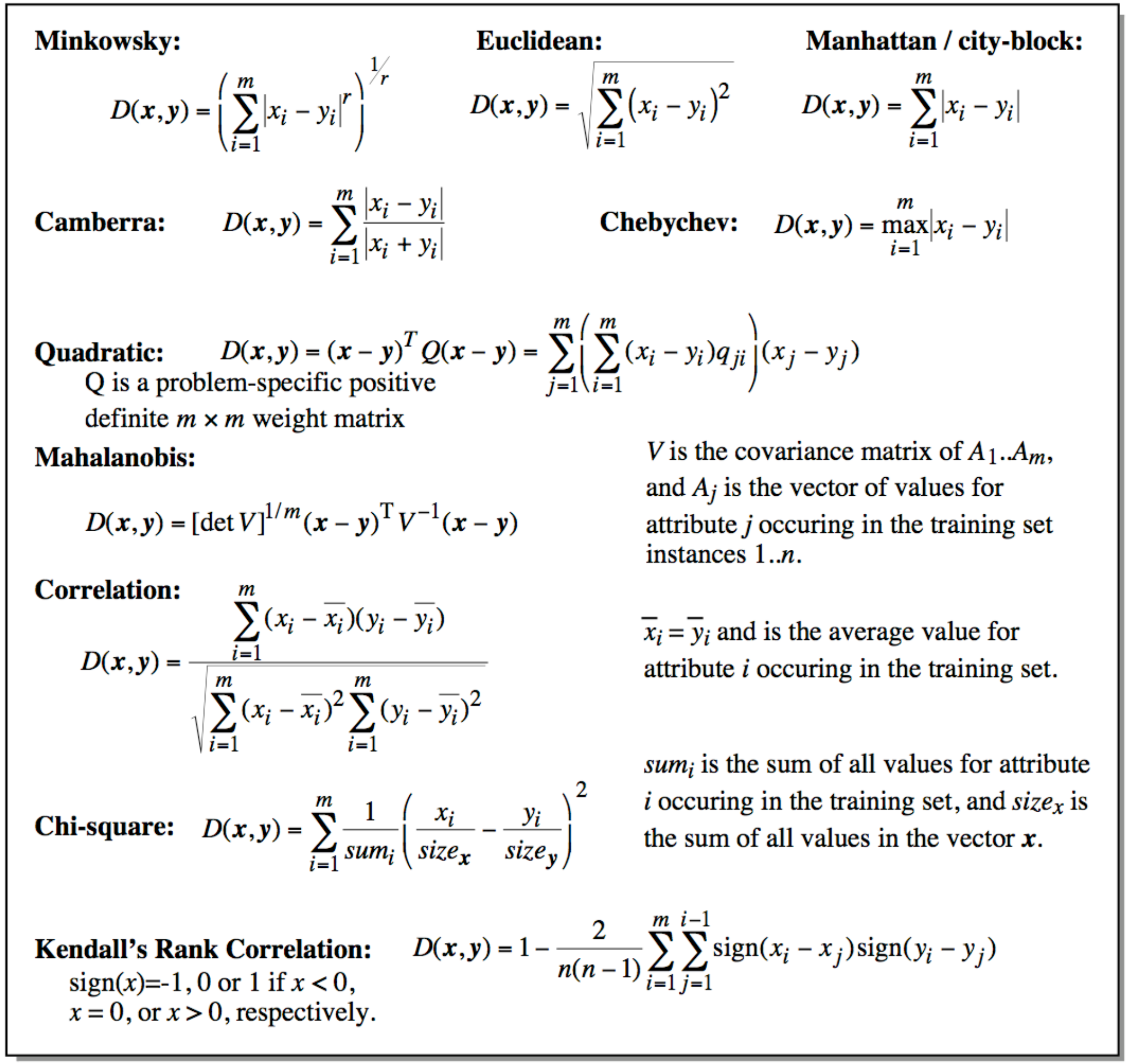
Value of K

- Choosing the value of K:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

Rule of thumb:
K = sqrt(N)
N: number of training points



Distance Metrics



Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} \left|x_i - y_i\right|^r\right)^{1/r}$$

Euclidean:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} \left(x_i - y_i\right)^2}$$

Manhattan / city-block:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \left|x_i - y_i\right|$$

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{\left|x_i - y_i\right|}{\left|x_i + y_i\right|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} \left|x_i - y_i\right|$$

Quadratic:

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^T Q(\boldsymbol{x}-\boldsymbol{y}) = \sum_{j=1}^{m}\left(\sum_{i=1}^{m}(x_i - y_i)q_{ji}\right)(x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m}(\boldsymbol{x}-\boldsymbol{y})^{\mathrm{T}} V^{-1}(\boldsymbol{x}-\boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m}(x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m}(x_i - \overline{x_i})^2 \sum_{i=1}^{m}(y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i}\left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}}\right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_x$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m}\sum_{j=1}^{i-1} \operatorname{sign}(x_i - x_j)\operatorname{sign}(y_i - y_j)$$

sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

Figure 1. Equations of selected distance functions. ($\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of m attribute values).

Distance Measure: Scale Effects

- Different features may have different measurement scales
  - E.g., patient weight in kg (range [50,200]) vs. blood protein values in ng/L ([-3,3])
- Consequences
  - Patient weight will have a greater influence on the distance between samples
  - May bias the performance of the classifier
- Transform raw feature values into z-scores $z_{ij} = \frac{x_{ij} - m_j}{s_j}$
  - x_{ij} is the value for the *i*th sample and *j*th feature
  - m_j is the average of all inputs or feature *j*
  - s_j is the standard deviation of all inputs over all input samples
- Range and scale of z-scores should be similar (providing distributions of raw feature values are alike)

Nearest Neighbor : Dimensionality

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results
  - Shrinking density – sparsification effect

1 1 1 1 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 1 1 1 1

d = 1.4142

vs

1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1

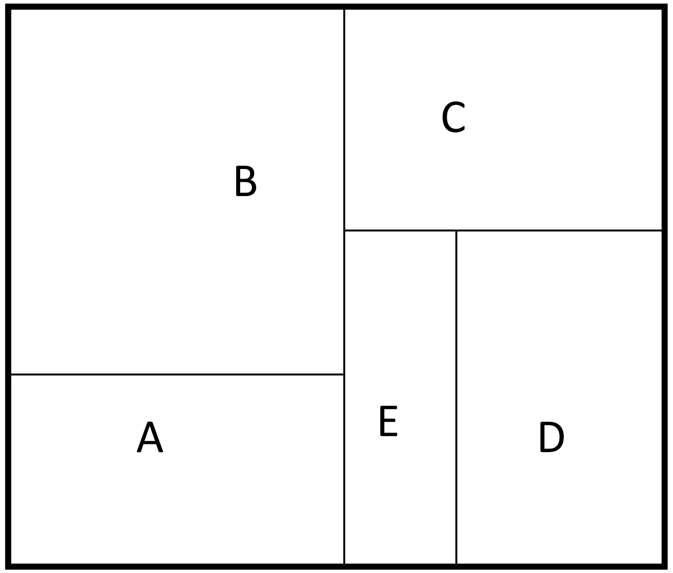
d = 1.4142

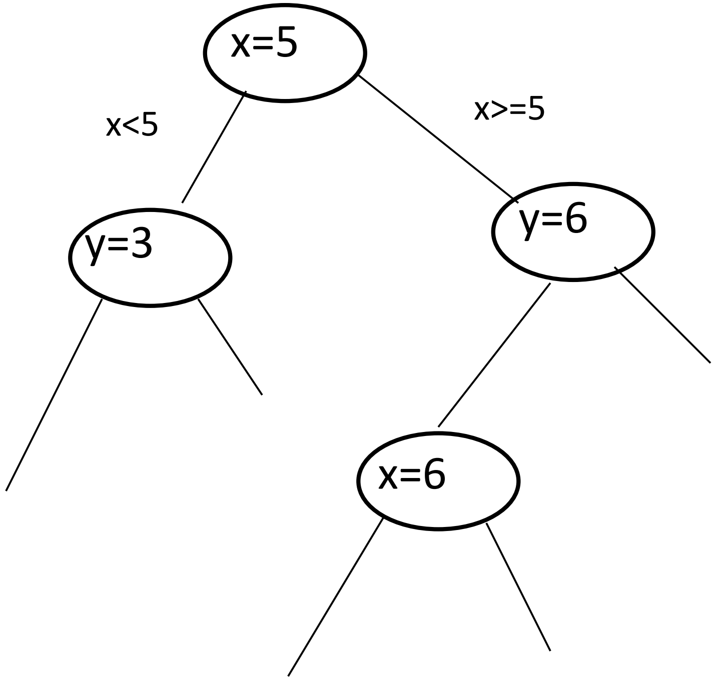
Nearest Neighbour : Computational Complexity

- Expensive
  - To determine the nearest neighbour of a query point *q*, must compute the distance to all *N* training examples
    - + Pre-sort training examples into fast data structures (kd-trees)
    - + Compute only an approximate distance (LSH)
    - + Remove redundant data (condensing)
- Storage Requirements
  - Must store all training data **P**
    - + Remove redundant data (condensing)
    - - Pre-sorting often increases the storage requirements
- High Dimensional Data
  - "Curse of Dimensionality"
    - Required amount of training data increases exponentially with dimension
    - Computational cost also increases dramatically
    - Partitioning techniques degrade to linear search in high dimension

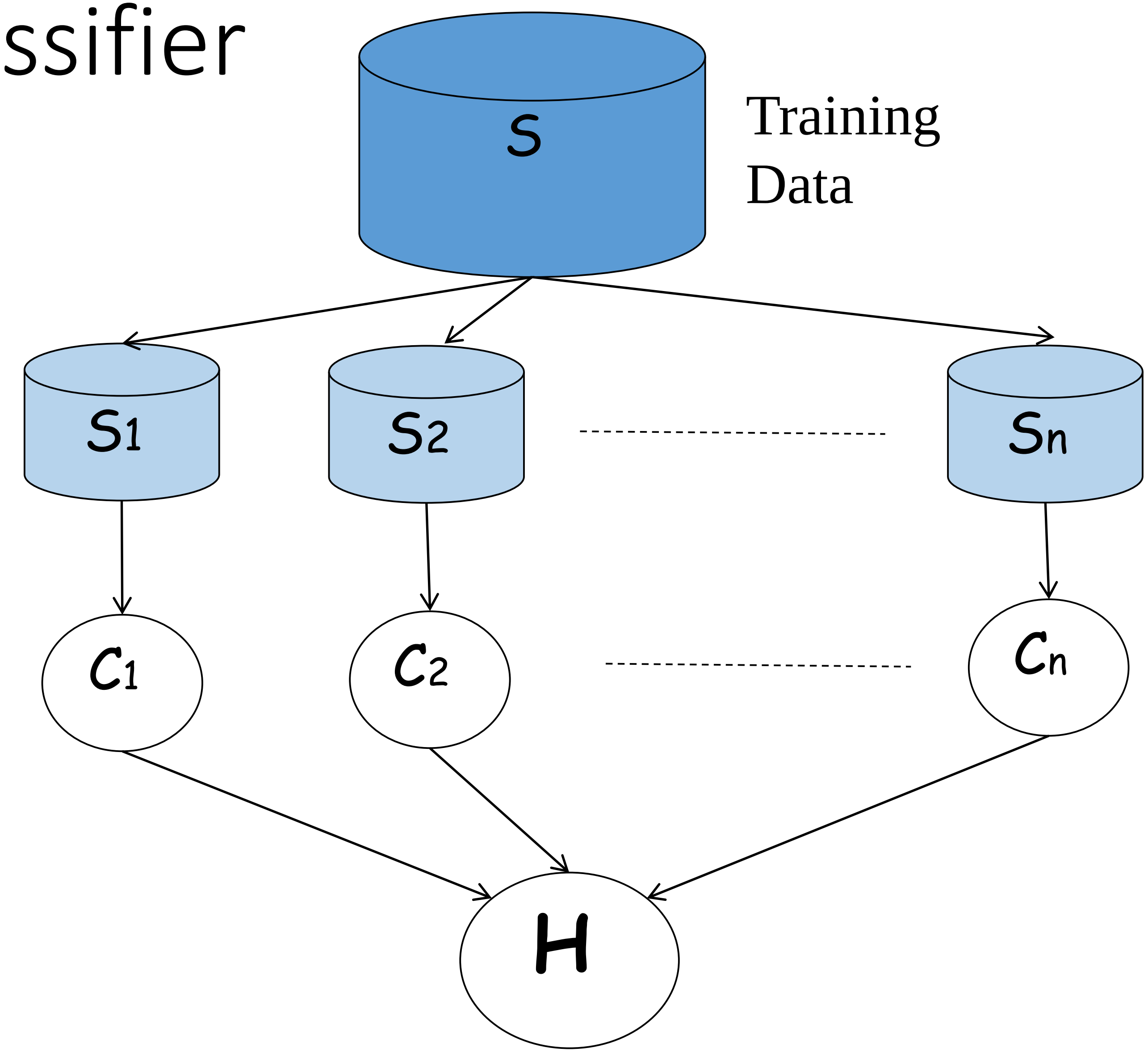
kd-tree: Data structure for fast range search

- Index data into a tree
- Search on the tree
- Tree construction: At each level we use a different dimension to split





Ensemble Classifier



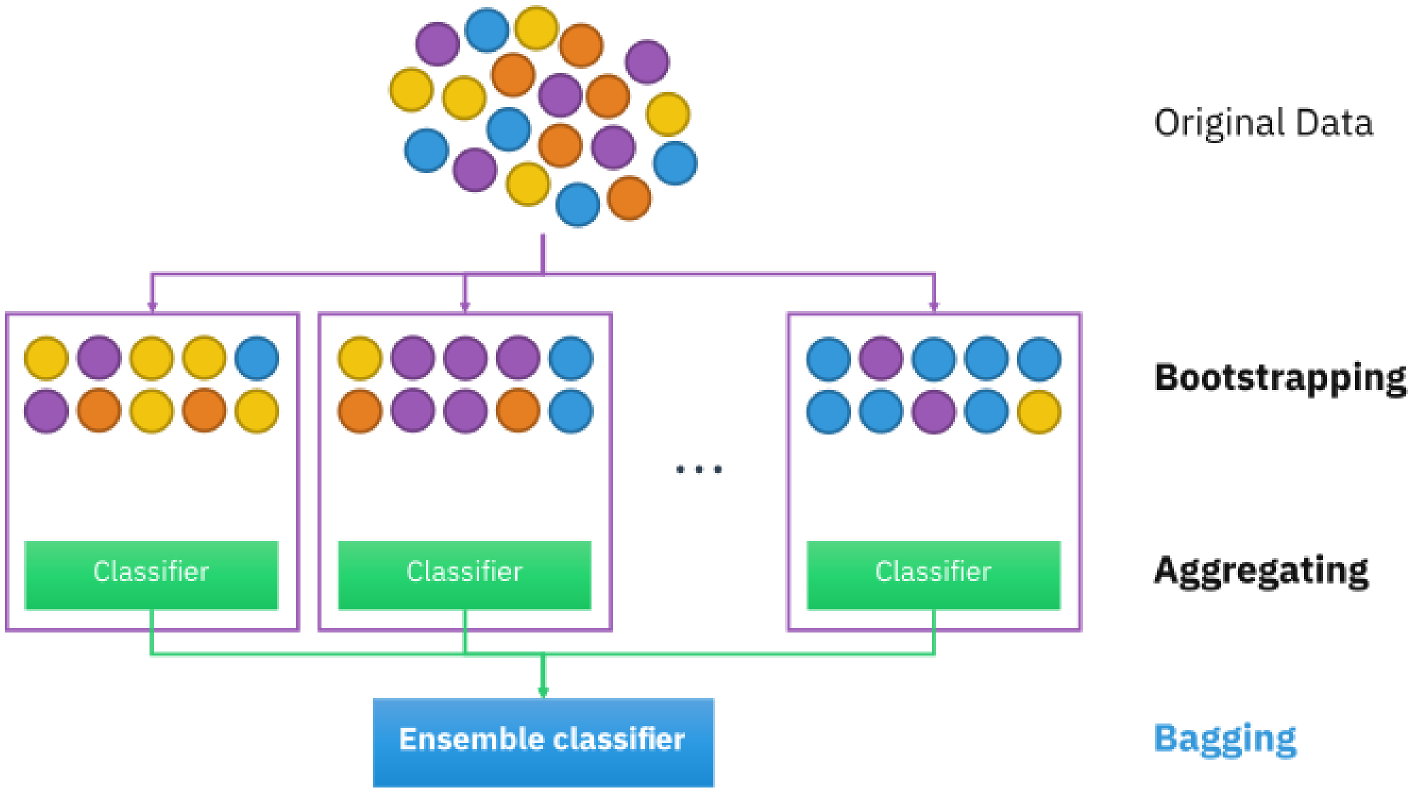
Training Data

Multiple Data Sets

Multiple Classifiers

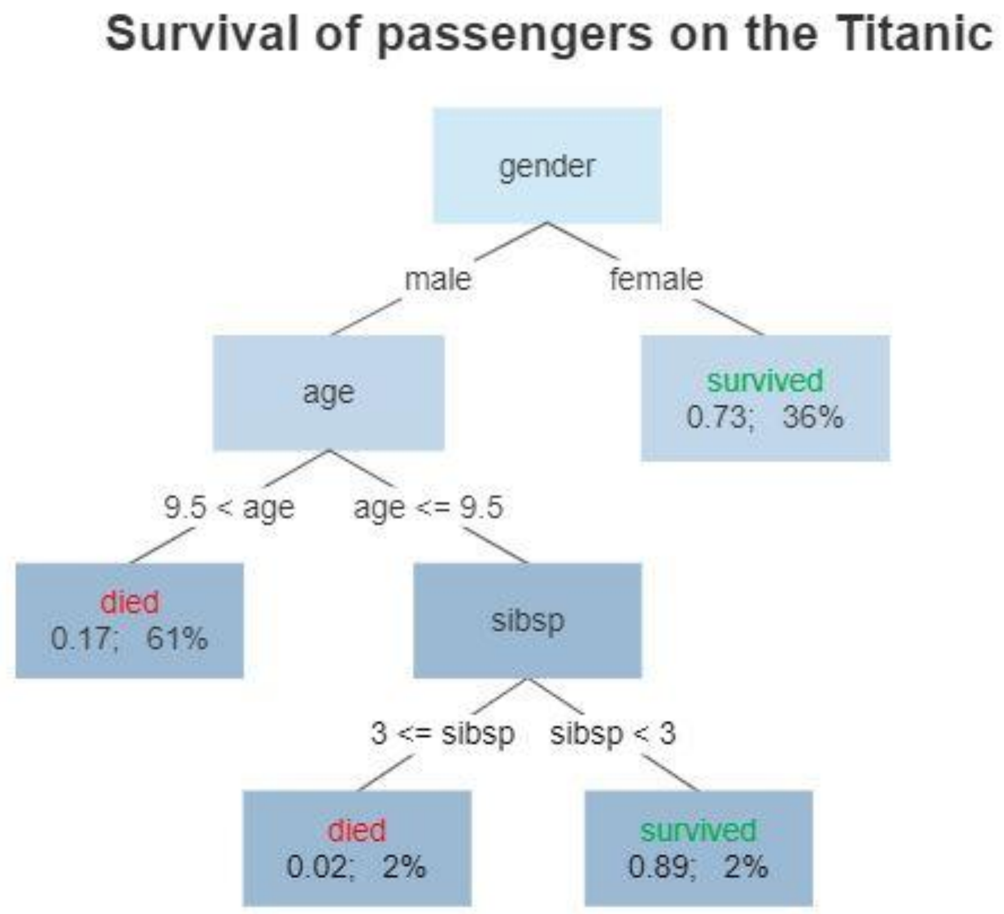
Combined Classifier

Bagging (Bootstrapped Aggregation)



Bootstrapping: Sampling with replacement from the original data set

Decision Trees


Survival of passengers on the Titanic

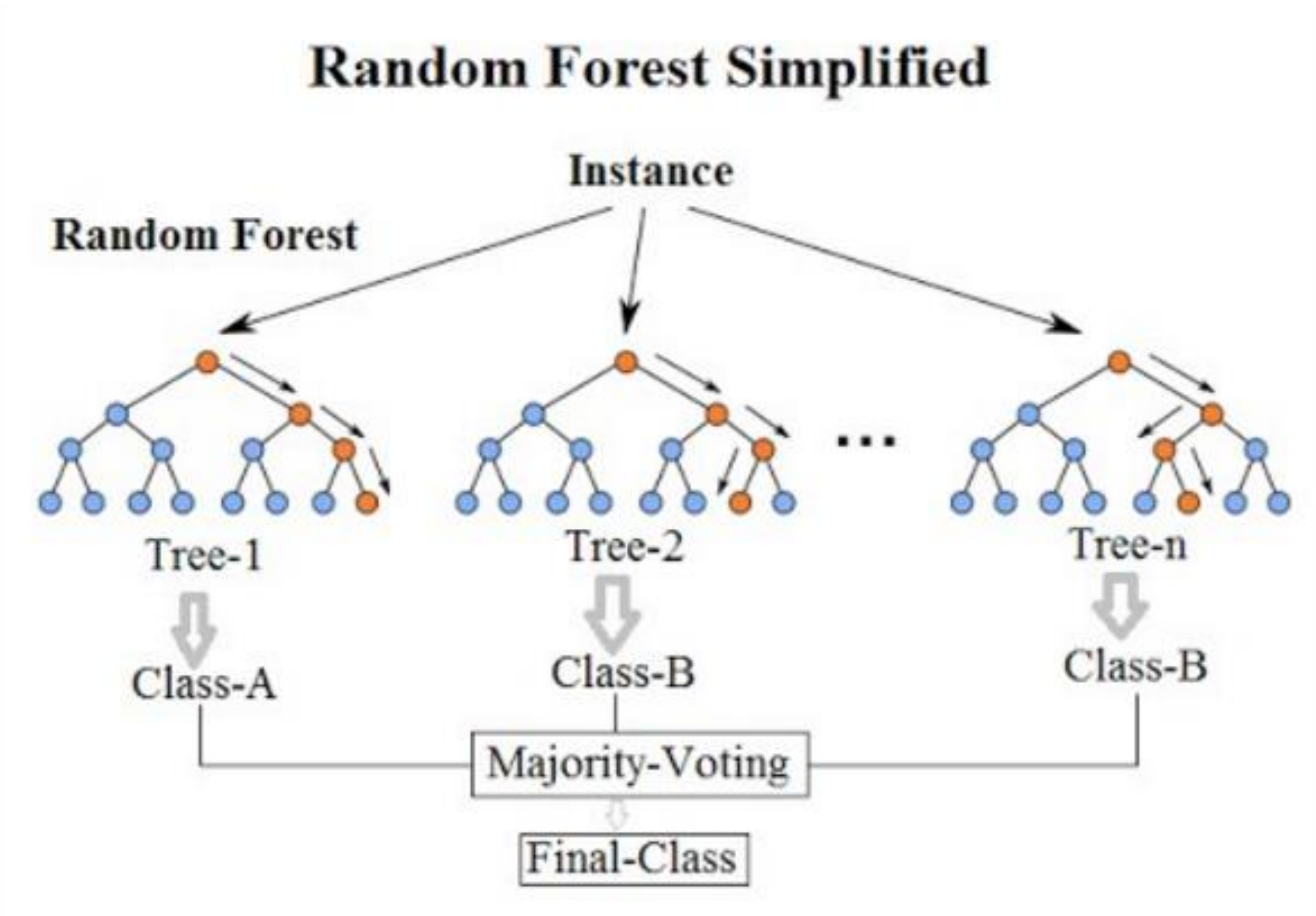
Leaves denote class decisions, other nodes denote attributes of data points

Decision Tree Construction

- Repeat:
  1. Split the “best” decision attribute (*A)* for next node
  2. For each value of *A*, create new descendant of node
  4. Sort training examples to leaf nodes
  5. If training examples perfectly classified, STOP,

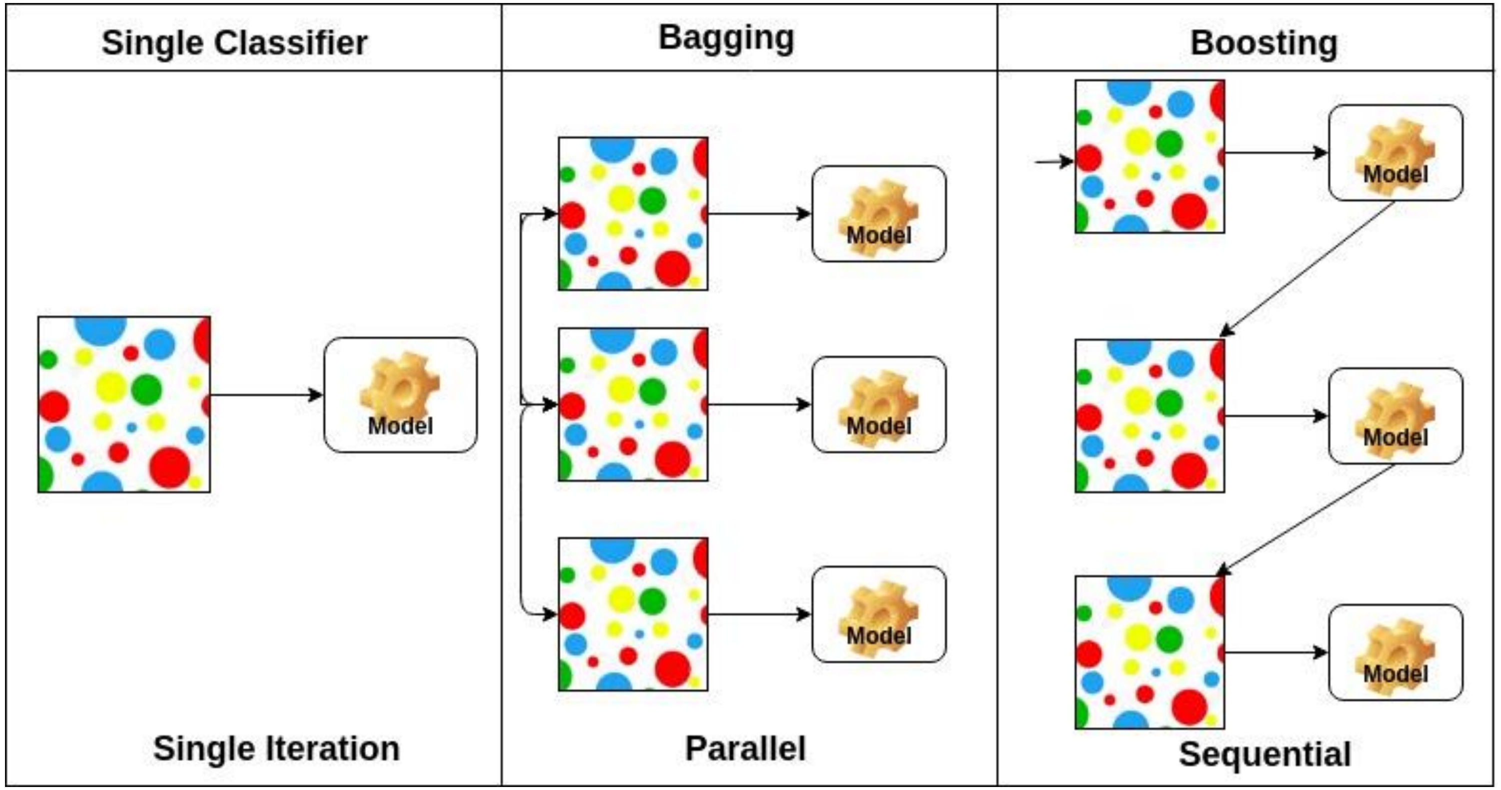
  Else iterate over new leaf nodes
- Grow tree just deep enough for perfect classification
  - –If possible (or can approximate at chosen depth)
- Which attribute is best? (Information Gain Maximization)

- Simplified tree construction: At each level use only a small random subset of attributes to create descendants



Randomization on attributes + Randomization on training data points

Boosting



Data points are adaptively weighted. Misclassified points are emphasised such that the next classifier Compensates for error of earlier classifier

Adaboost

- Data Point Weight Updates

$$\alpha_m = \log\left(\frac{1 - \text{err}_m}{\text{err}_m}\right)$$

$$w_i^{(\text{m})} = w_i^{(\text{m-1})} \cdot \exp(\alpha_m \cdot \mathbb{I}(y_i \neq g_m(x_i)))$$

Hence:

$$w_i^{(\text{m})} = \begin{cases} w_i^{(\text{m-1})} & \text{if } g_m \text{ classifies } x_i \text{ correctly} \\ w_i^{(\text{m-1})} \cdot \frac{1-\text{err}_m}{\text{err}_m} & \text{if } g_m \text{ misclassifies } x_i \end{cases}$$

- Weighted Classifier Combination $f(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m g_m(x)\right)$

Forward Stagewise Additive Modelling

FSAM: For $m = 1, \ldots, M$, find model f_m by minimizing the empirical risk

$$f_m = \underset{h \in \Phi}{argmin} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f^{(m-1)}(x_i) + h(x_i))$$

- $f^{(m-1)}(x) = \sum_{j=1}^{m-1} f_j(x)$, with $\mathrm{f}^{(0)}(x) = 0$
- For some model class Φ
- A very general procedure, but hard to do for general loss function

Adaboost FSAM for exponential loss.

Gradient Boosting

- Gradient Descent + Boosting

$y_i = M_1(x_i) + \varepsilon_{1i}$ Error term indicate inadequacy of the model

$e_{i1} = y_i - M_1(x_i)$ Residual

Model residual with another classifier M2. And append it to M1.

$e_{i1} = M_2(x_i) + \varepsilon_{2i}$ Continue over iterations

$\hat{y}_i = M_1(x_i) + M_2(x_i)$ M1 additively compensates inadequacy of M1

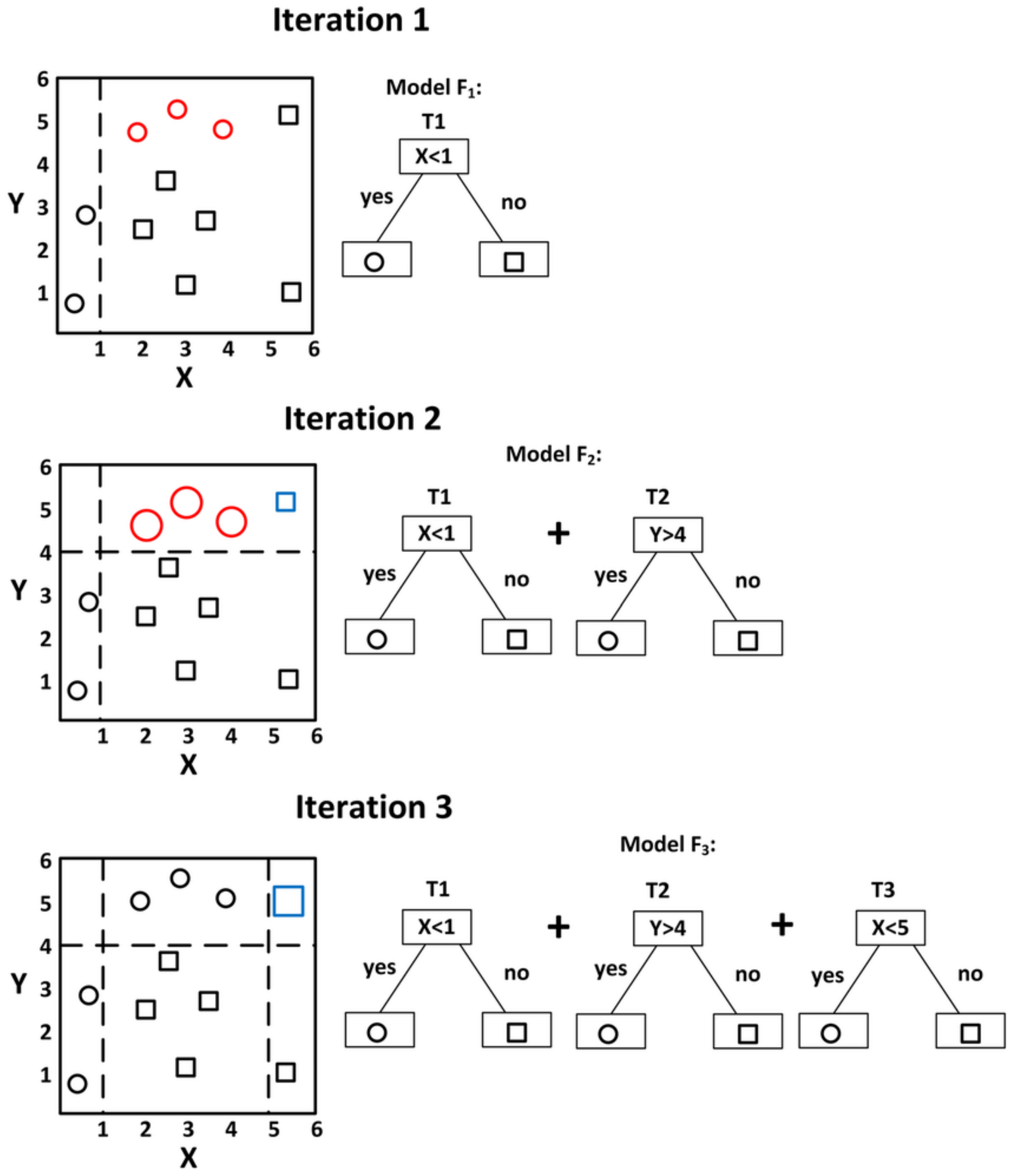
$$\frac{\partial j(y_i, f(x_i))}{\partial f(x_i)} = \frac{\partial\left[\frac{1}{2}\left(y_i - f(x_i)\right)^2\right]}{\partial f(x_i)} = f(x_i) - y_i$$

Residual = negative gradient

$$\begin{aligned} f_b(x_i) &= f_{b-1}(x_i) + M_b(x_i) \\ &= f_{b-1}(x_i) + \left(y_i - f_{b-1}(x_i)\right) \\ &= f_{b-1}(x_i) - 1 \times \frac{\partial j(y_i, f(x_i))}{\partial f(x_i)} \\ &= f_{b-1}(x_i) - \eta \times \nabla j(y_i, f(x_i)) \end{aligned}$$

Iterative process a gradient descent

Gradient Boosting



Gradient Boosting for Regression

- We have a set of variables vectors x1 , x2 and x3. You need to predict y which is a continuous variable.
- **Steps of Gradient Boost algorithm**

  Step 1 : Assume mean is the prediction of all variables.

  Step 2 : Calculate errors of each observation from the mean (latest prediction).

  Step 3 : Find the variable that can split the errors perfectly and find the value for the split. This is assumed to be the latest prediction.

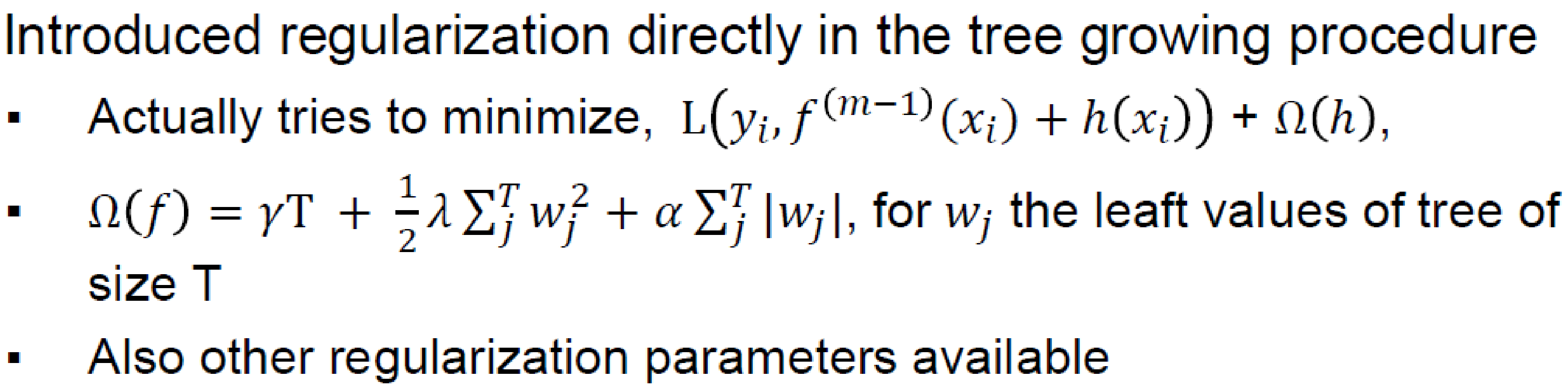
  Step 4 : Calculate errors of each observation from the mean of both the sides of split (latest prediction).

  Step 5 : Repeat the step 3 and 4 till the objective function maximizes/minimizes.

  Step 6 : Take a weighted mean of all the classifiers to come up with the final model.

XGBoost: eXtreme Gradient Boosting

- Gradient Boosting + Regularization

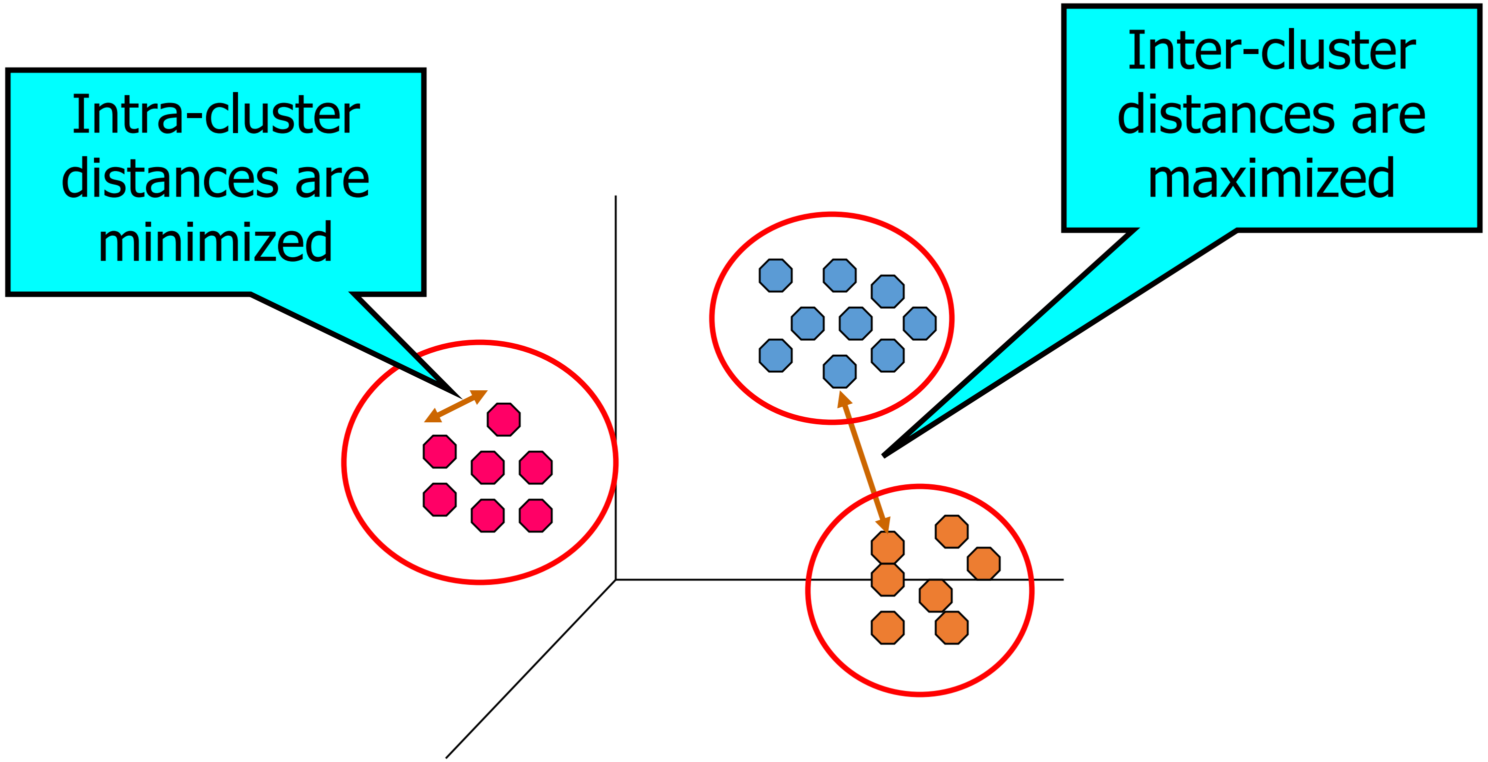


Introduced regularization directly in the tree growing procedure

- Actually tries to minimize, $\mathrm{L}\big(y_i, f^{(m-1)}(x_i) + h(x_i)\big)$ + $\Omega(h)$,
- $\Omega(f) = \gamma\mathrm{T} \; + \frac{1}{2}\lambda \sum_j^T w_j^2 + \alpha \sum_j^T |w_j|$, for w_j the leaft values of tree of size T
- Also other regularization parameters available

Clustering

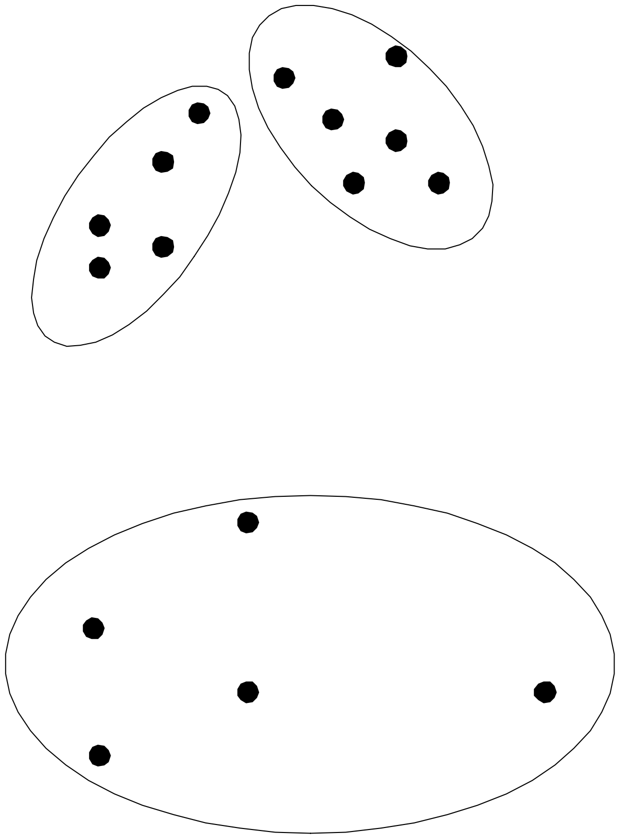
- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



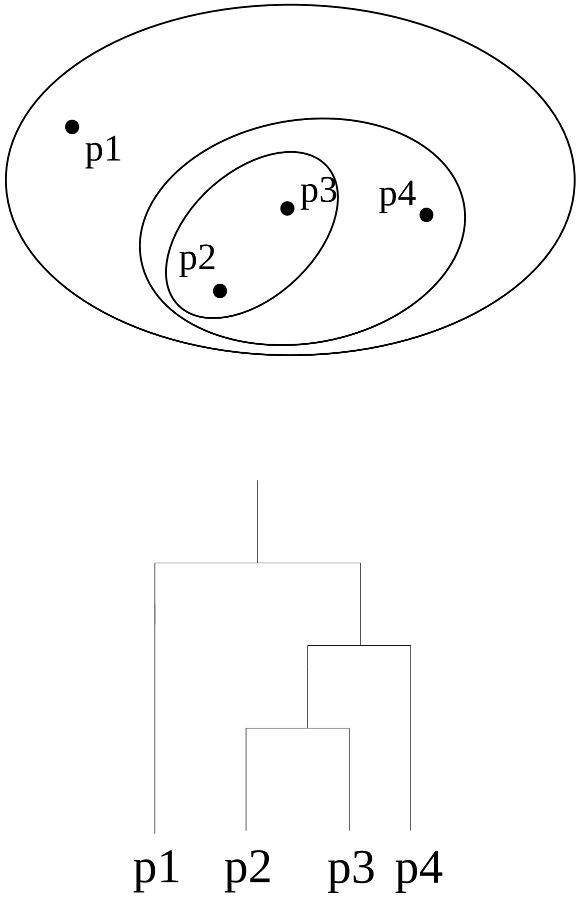
Intra-cluster distances are minimized

Inter-cluster distances are maximized

Clustering Algorithms



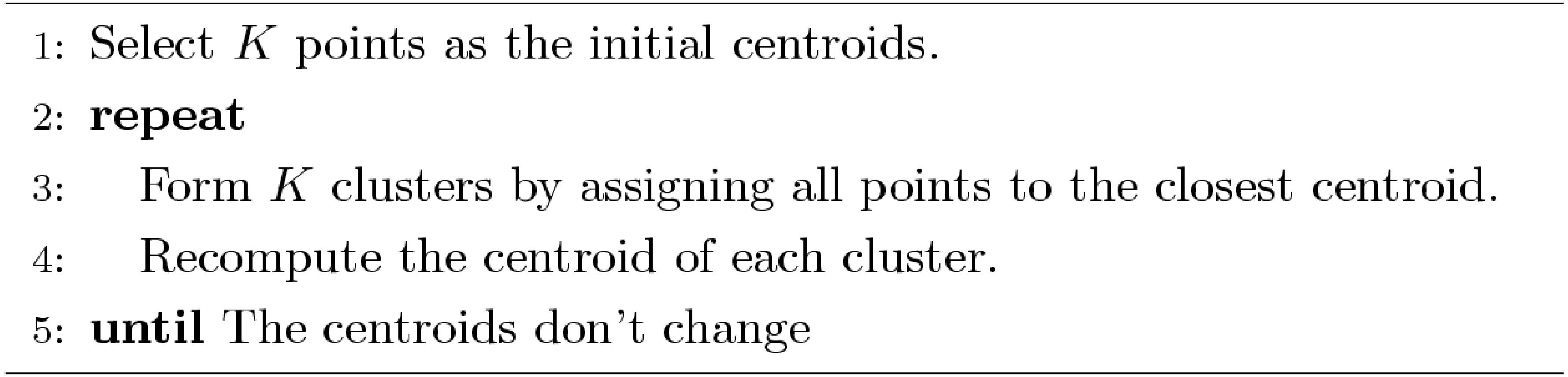
Partitional Clustering



Hierarchical Clustering

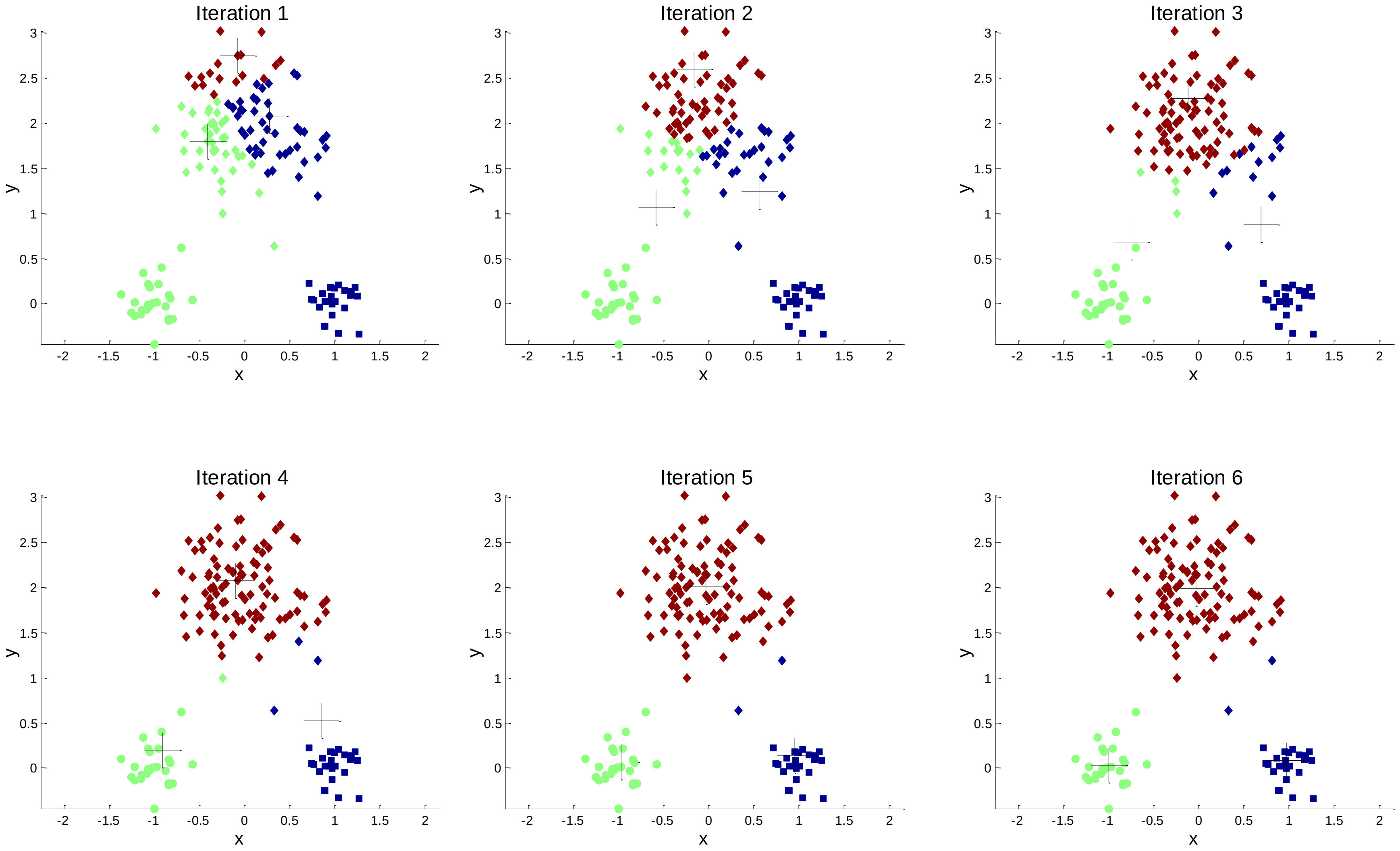
K-Means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified

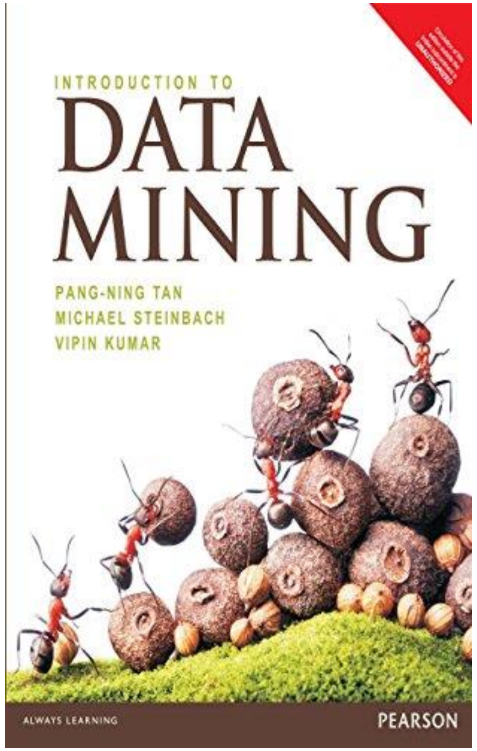


```
Select $K$ points as the initial centroids.
**repeat**
    Form $K$ clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
**until** The centroids don't change
```

K-Means Iterations



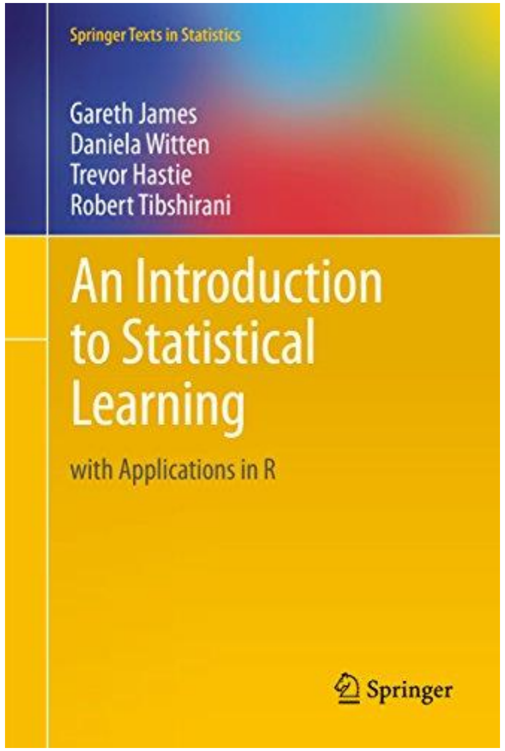
References



INTRODUCTION TO DATA MINING

PANG-NING TAN
MICHAEL STEINBACH
VIPIN KUMAR

ALWAYS LEARNING PEARSON



Springer Texts in Statistics

Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani

An Introduction to Statistical Learning

with Applications in R

Springer