MACHINE LEARNING IN BIOINFORMATICS

Part 4: Classification

(Adapted slides by Junming Yin)

František Mráz
KSVI MFF UK

Classification

- **Problem:**
  - Given sample objects together with labeling to which class they belong
  - For a new object x predict its class label y
- **Examples:**
  - Is this transaction a fraud?
  - Will this customer buy this product?
  - Is this protein an enzyme?
  - Is this DNA sequence a gene?
  - Is this site on RNA a splicing site?

Setting – A Supervised Learning

- **A training dataset:** a set of pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is an object and y_i is its class label
  - Usually $x_i \in \mathbb{R}^d$ is called input vector and $y_i \in \Theta$ is called a target variable
- A **test dataset**: a set of objects $x_1', x_2', \ldots$ with unknown class labels
- The **task**: predict class labels $y_1', y_2', \ldots$ of the objects $x_1', x_2', \ldots$
- Domain of y:
  - $\Theta = \{0,1\}$: a **binary classification problem**
  - $\Theta = \{1, 2, \ldots, n\}$: a **multiclass classification problem**
  - $\Theta = \mathbb{R}$: a **regression problem**

Setting – A Supervised Learning

- **A training dataset:** a set of pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is an object and y_i is its class label
  - Usually $x_i \in \mathbb{R}^d$ is called input vector and $y_i \in \Theta$ is called a target variable
- A **test dataset**: a set of objects $x_1', x_2', \ldots$ with unknown class labels
- The **task**: predict class labels $y_1', y_2', \ldots$ of the objects $x_1', x_2', \ldots$
- **In bioinformatics:**
  - x are called explanatory variables, they describe the causes – often refer to genotypic data
  - y are called predictive variables, they describe observed phenotypic data
  - task: find causes (model) to interpret the observed phenotypic data; i.e. model is a mapping: explanatory variables → predictive variables

Classification

- We assume that $\mathcal{D}$ is randomly sampled from a space $(\mathbb{R}^d \times \Theta)$ satisfying an unknown function

$$f(x) \longmapsto y$$

- The number of possible datapoints in $\mathbb{R}^d \times \Theta$ satisfying $f(x) \longmapsto y$ is infinite, but the size of $\mathcal{D}$ is finite

Overview of Classifiers

- **Nearest Neighbour**
  - Key idea: if x' is most similar to x_i, then $y_i = y'$
  - Classification by looking at the 'Nearest Neighbour'
- **Naïve Bayes**
  - A simple probabilistic classifier based on applying Bayes' theorem with strong (naïve) independence assumptions
- **Decision trees**
  - A series of decisions has to be taken to classify an object, based on its attributes
  - The hierarchy of these decisions is ordered as a tree, a 'decision tree'.

Overview of Classifiers

- **Support Vector Machine**
  - Key idea: Draw a line (plane, hyperplane) that separates two classes of data
  - Maximize the distance between the hyperplane and the points closest to it (margin)
  - Test point is predicted to belong to the class whose half-space it is located in
- **Criteria for a good classifier**
  - Accuracy
  - Runtime and scalability
  - Interpretability
  - Flexibility

Binary Classifier Evaluation

		Predicted class	
		True	False
True class	True		
	False		

- Specifity $Spe = \frac{TN}{TN+FP}$
- Sensitivity $Sen = \frac{TP}{TP+FN}$
- Total prediction accuracy $Tot = \frac{TN+TP}{TN+FP+TP+FN}$
- *Which of the measures is the most important?*

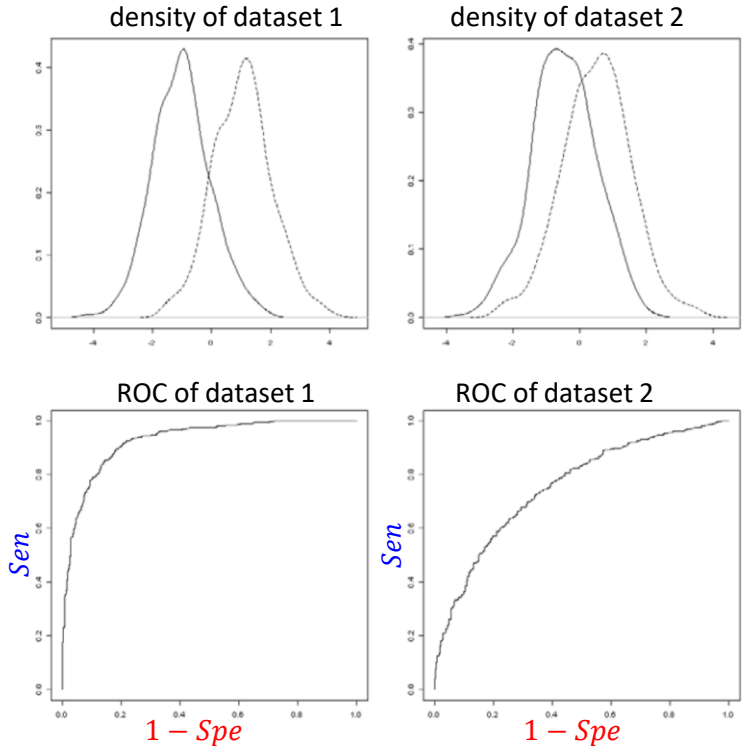
		Predicted class	
		True	False
True class	True	TP	FN
	False	FP	TN

The Receiver Operating Characteristics (ROC)

- Only in two-class classification, and only if we can parametrize the classifier
- Plot $Sen = \frac{TP}{TP+FN}$ (vertical axis) against $(1 - Spe) = 1 - \frac{TN}{TN+FP} = \frac{FP}{TN+FP}$ (horizontal axis)

		Predicted class	
		True	False
True class	True		
	False		

- **Criterion: A**rea **U**nder the ROC **C**urve (AUC)



Nearest Neighbour

- Given x', we predict its label y' by

$$\text{if } x_i = \arg\min_{x \in D} \|x - x'\|^2 \text{ then } y' = y_i$$

- Label predicted for x' is that of the point closest to it, that is its 'nearest neighbour'
- Runtime
  - Naïvely, one has to compute the distance to all N neighbours in the dataset for each point:
    - $O(N)$ for one point
    - $O(N^2)$ for the entire dataset
  - improving the performance and speed of a nearest neighbour classification
    - pre-sort the training sets in some way (such as kd-trees or Voronoi cells).
    - choose a subset of the training data such that classification by the 1-NN rule (using the subset) approximates the Bayes classifier (LVQ)

Naïve Bayes

- **Bayes' Rule**

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)}$$

- **Naïve Bayes Classification**
  - Classify x' into one of K classes $C_1, \dots, C_K$

$$\arg\max_{C_k} P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$

Naïve Bayes

$$\arg\max_{C_i} P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$$

- **Simplifications**
  - $P(x)$ is the same for all classes, ignore this term.
  - If x is multidimensional, that is if x contains d features $x = (x_1, \dots, x_d)$, we further assume that

$$P(x|C_k) = P(C_k) \prod_{j=1}^{d} P(x_j|C_k)$$

- **The actual classification**
  - We compute

"Proportional to"

$$P(x|C_k) \propto P(C_k) \prod_{j=1}^{d} P(x_j|C_k)$$

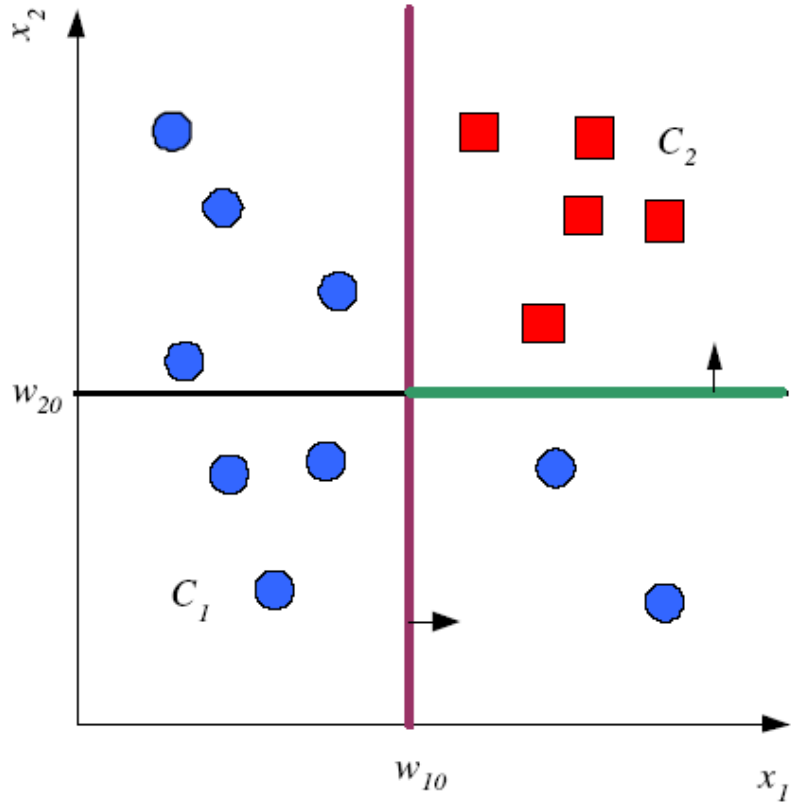
    for each class

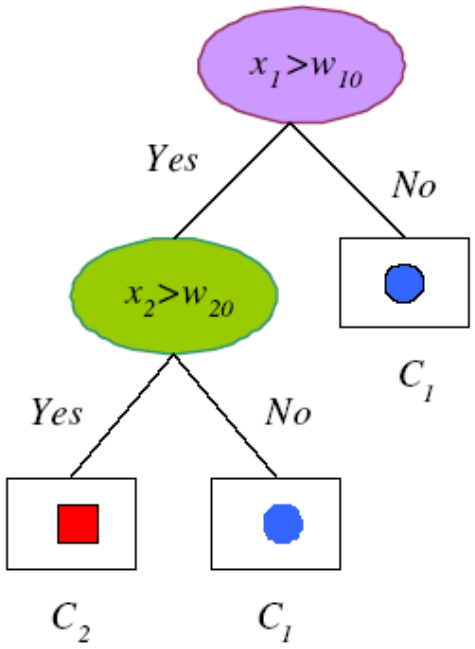
Time complexity $O(NKd)$

- Further simplification
  - assume $P(C_k)$ is the same for all classes $1 \le k \le K$, ignore this term as well.
  - That means $P(x|C_k) \propto \prod_{j=1}^{d} P(x_j|C_k)$ Simple Bayes Classifier

Decision Trees

- Recursively split the data space into regions that contain a single class only





Decision Tree

- **Concept**
  - A decision tree is a flowchart like tree structure with
  - a root: this is the uppermost node
  - internal nodes: these represents tests on an attribute
  - branches: these represent outcomes of a test
  - leaf nodes: these hold a class label
- **Classification**
  - given a test point x
  - perform test on the attributes of x at the root
  - follow the branch that corresponds to the outcome of this test
  - repeat this procedure, until you reach a leaf node
  - predict the label of x to be the label of that leaf node

Decision Tree

- **Popularity**
  - requires no domain knowledge
  - easy to interpret
  - construction and prediction is fast
- **Construction**
  - requires to determine a splitting criterion at each internal node
  - this splitting criterion tells us which attribute to test at node v
  - we would like to use the attribute that best separates the classes on the training dataset

Decision Tree

- **Information gain**
  - ID3 uses information gain as attribute selection measure
  - The information content is defined as:

$$\text{Info}(D) = -\sum_{k=1}^{K} p(C_k) \log_2 p(C_k)$$

where $p(C_k)$ is the probability that an arbitrary tuple in D belongs to class C_k and is estimated by $|C_{k,D}|/|D|$

  - This is also known as the Shannon entropy of D

Classification Trees (ID3, CART, C4.5)

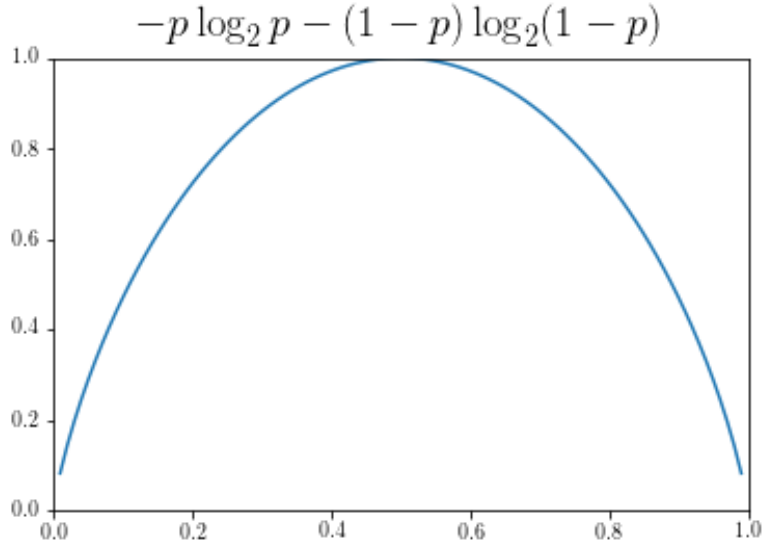
- For a node m, N_m instances reach m, $N_m^{(k)}$ out of them belong to C_k

$$\hat{P}(C_k|x,m) = p_m^{(k)} = \frac{N_m^{(k)}}{N_m}$$

The probability that an instance x which reaches the node m is from class C_k

- Node m is pure if $p_m^{(k)}$ is 0 or 1
- Measure of impurity is entropy

$$\mathcal{I}_m = -\sum_{k=1}^{K} p_m^{(k)} \log_2 p_m^{(k)}$$



Best Split

- If node m is pure (or almost pure $\mathcal{I}_m < \Theta$), generate a leaf and stop, otherwise split and continue recursively
- Impurity after split – weighted entropy: N_{mj} instances from N_m take branch j, $1 \leq j \leq n$, $N_{m_j}^{(k)}$ of them belong to C_k

$$\hat{P}(C_k|x, m, j) = p_{mj}^{(k)} = \frac{N_{mj}^{(k)}}{N_{mj}}$$

$$\mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{k=1}^{K} p_{mj}^{(k)} \log_2 p_{mj}^{(k)}$$

The probability that an instance x in branch j under the node m is from class C_k

- Find the variable and split that minimizes impurity (among all variables – and split positions for numeric variables)

GenerateTree(X)

If NodeEntropy(X)< θ_I

Create leaf labelled by majority class in X

Return

$i \leftarrow$ SplitAttribute(X)

For each branch of x_i

Find X_i *falling in branch*

GenerateTree(X_i)

SplitAttribute(X)

$MinEnt \leftarrow MAX$

For all attributes $i = 1, \ldots, d$

If x_i is discrete with n values

Split X into $X_1, \ldots, X_n$ by x_i

$e \leftarrow$ SplitEntropy($X_1, \ldots, X_n$)

If $e < MinEnt$

$MinEnt \leftarrow e$;

$bestf \leftarrow i$

Else /* x_i *is numeric* */

For all possible splits

Split X into X_1, X_2 on x_i

$e \leftarrow$ SplitEntropy(X_1, X_2)

If $e < MinEnt$

$MinEnt \leftarrow e$;

$bestf \leftarrow i$

Return $bestf$

Regression Trees

- Error at node m:

$$b_m(x) = \begin{cases} 1 & \text{if } x \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$

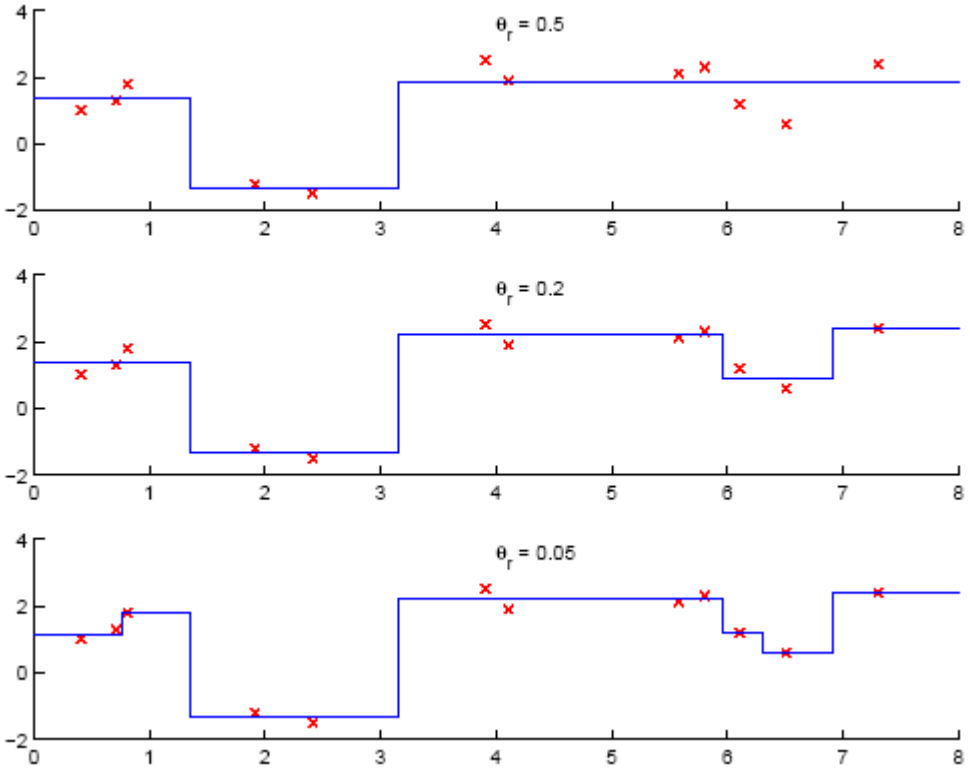
$$E_m = \frac{1}{N_m} \sum_t (y_t - g_m)^2 b_m(x_t) \qquad g_m = \frac{\sum_t b_m(x_t) y_t}{\sum_t b_m(x_t)}$$

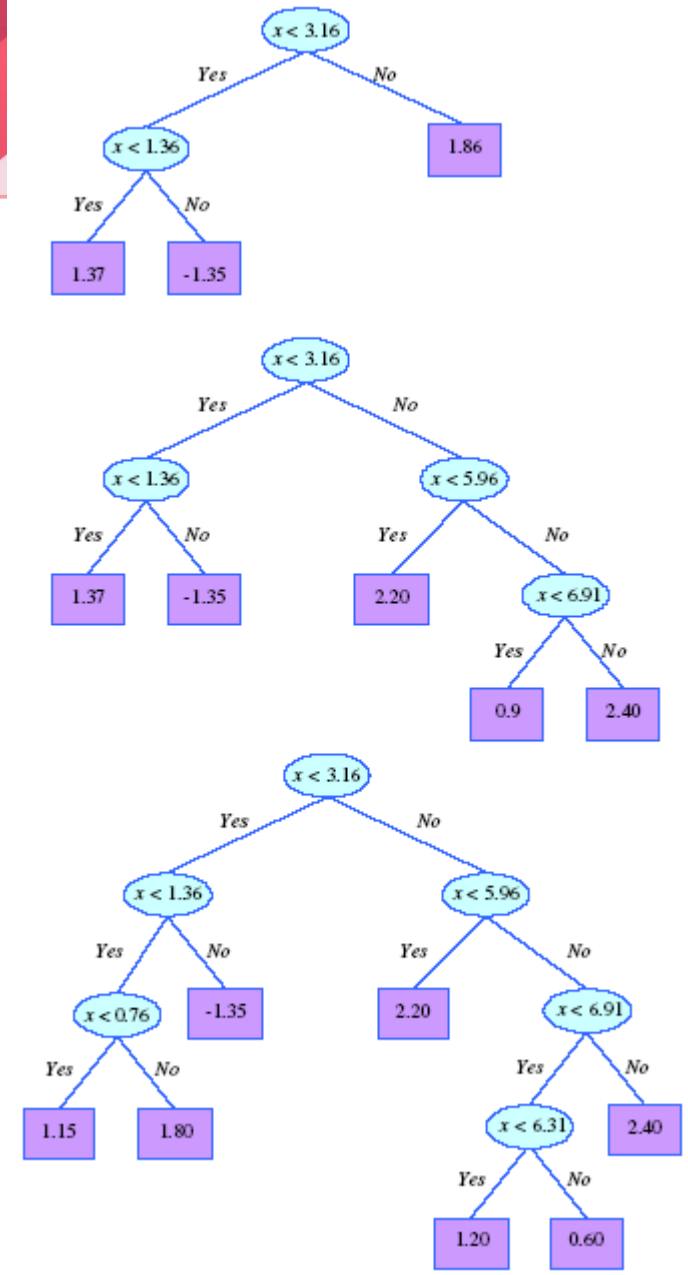
- After splitting:

$$b_{mj}(x) = \begin{cases} 1 & \text{if } x \text{ reaches node } m \text{ and branch } j \\ 0 & \text{otherwise} \end{cases}$$

$$E'_m = \frac{1}{N_m} \sum_j \sum_t \left(y_t - g_{mj}\right)^2 b_{mj}(x_t) \qquad g_{mj} = \frac{\sum_t b_{mj}(x_t) y_t}{\sum_t b_{mj}(x_t)}$$

Model Selection in Trees





Controlling Size of the Tree

- Grow-then-prune strategy (CART algorithm – Classification And Regression Tree):
  - create very large tree
  - prune back according to some criterion
- Pruning criteria:
  - $(impurity + \alpha \cdot |tree|)$
  - α determined by cross-validation

Categorical Variables, Missing Values

- **Categorical variables:**
  - **Problem**: with d possible unordered variables, e.g., color (blue, white, red), there are $2^{d-1}-1$ possible partitions
  - **Solution** (when only two possible outcomes $0/1$): sort variables according to the number of occurrences in each, e.g., white 0.9, red 0.45, blue 0.3 . Split predictor as with ordered variables.

- **Missing values:**
  - **Problem**: points x with missing values y, due to:
    - the proper measurement not taken
    - a source causing the absence of labels
  - **Solution**:
    - categorical case: create new category missing
    - use surrogate variables: use only those variables that are available for a split

Instability

- **Problem:** high variance
  - small changes in the data may lead to very different splits,
  - price to pay for the hierarchical nature of decision trees,
  - more stable criteria could be used.

Decision Tree Tools

- Most commonly used tools for learning decision trees:
  - CART (**C**lassification **A**nd **R**egression **T**ree) (Breiman et al., 1984)
  - C4.5 (Quinlan, 1986, 1993) and C5.0 (RuleQuest Research) a commercial system.
- Differences: minor between latest versions.

Summary

- Straightforward to train.
- Easily interpretable (modulo instability).
- Often not best results in practice

  → boosting decision trees in a later lecture