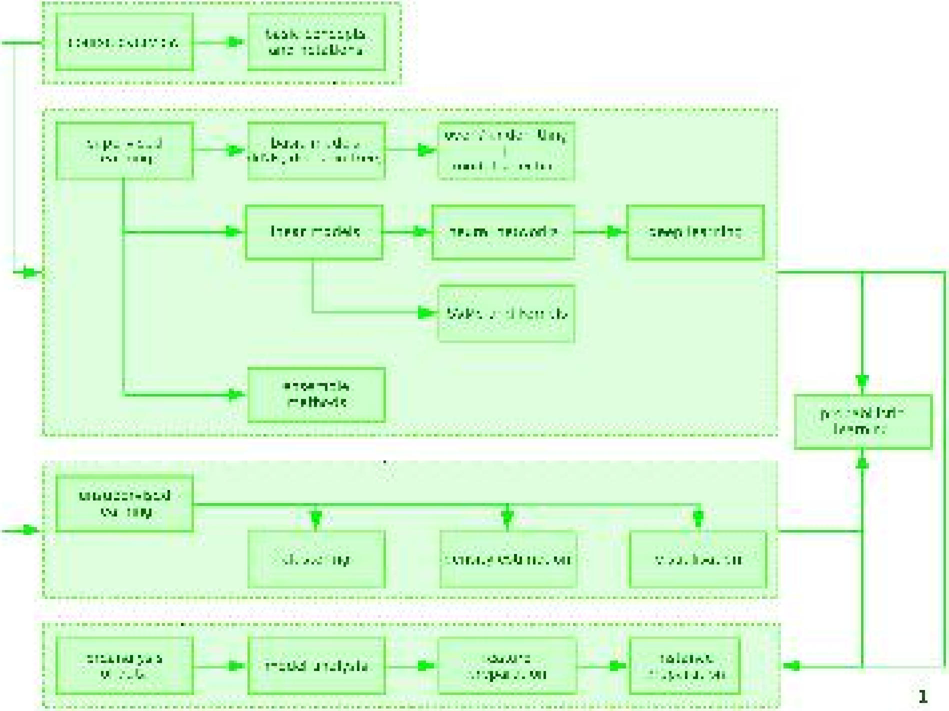
Machine Learning: Lesson 4

Basic Models for Supervised Learning

Benoît Frénay - Faculty of Computer Science

UNIVERSITÉ DE NAMUR


linear models
ensemble methods

Outline of this Lesson

- k-nearest neighbours
- decision trees

k-Nearest Neighbours

k-Nearest Neighbours for Classification

Training of a kNN classifier

Input: dataset $D = \{(\mathbf{x}_i, t_i)\}$
Output: kNN classifier

store the dataset for future predictions

Prediction with a kNN classifier

Input: new instance x
Output: predicted class y

find the k nearest neighbours of x in the training set D: $x_{i_1}, \ldots, x_{i_k}$
return the majority class y amongst the corresponding labels $t_{i_1}, \ldots, t_{i_k}$

k-Nearest Neighbours for Classification

Training of a kNN classifier

Input: dataset $D = \{(\mathbf{x}_i, t_i)\}$
Output: kNN classifier

store the dataset for future predictions

Prediction with a kNN classifier

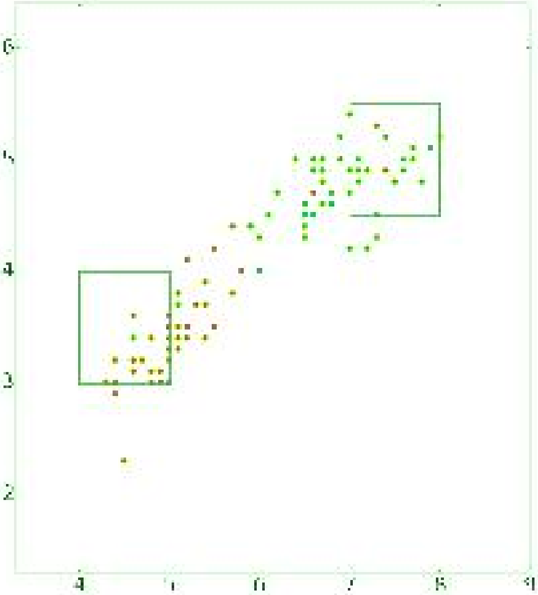
Input: new instance x
Output: predicted class y

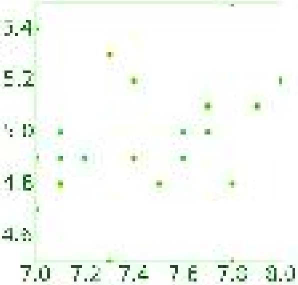
find the k nearest neighbours of x in the training set D: $x_{i_1}, \dots, x_{i_k}$
return the majority class y amongst the corresponding labels $t_{i_1}, \dots, t_{i_k}$

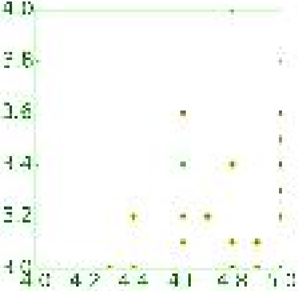
Learning bias

classification of an instance is close to the classification of nearby instances

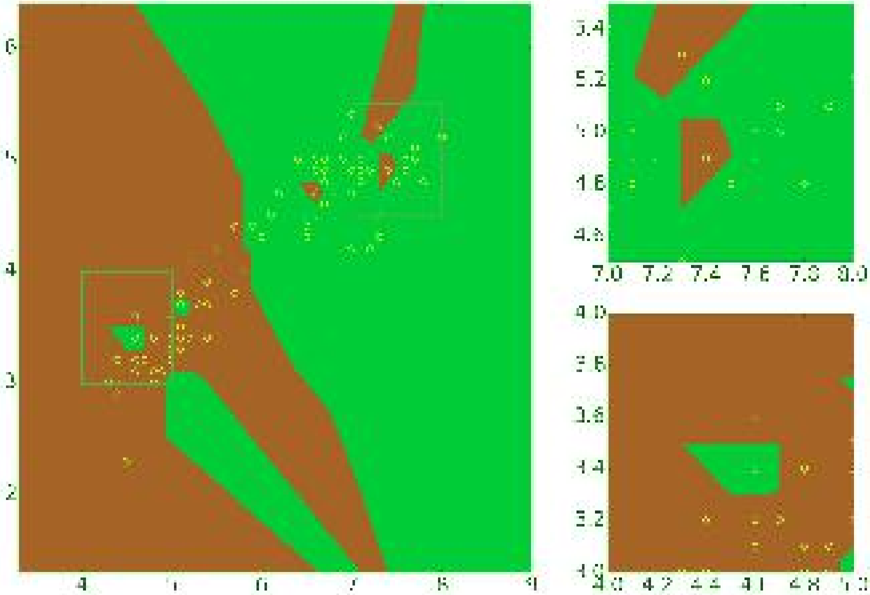
Example of k-Nearest Neighbours Binary Classifier (data)



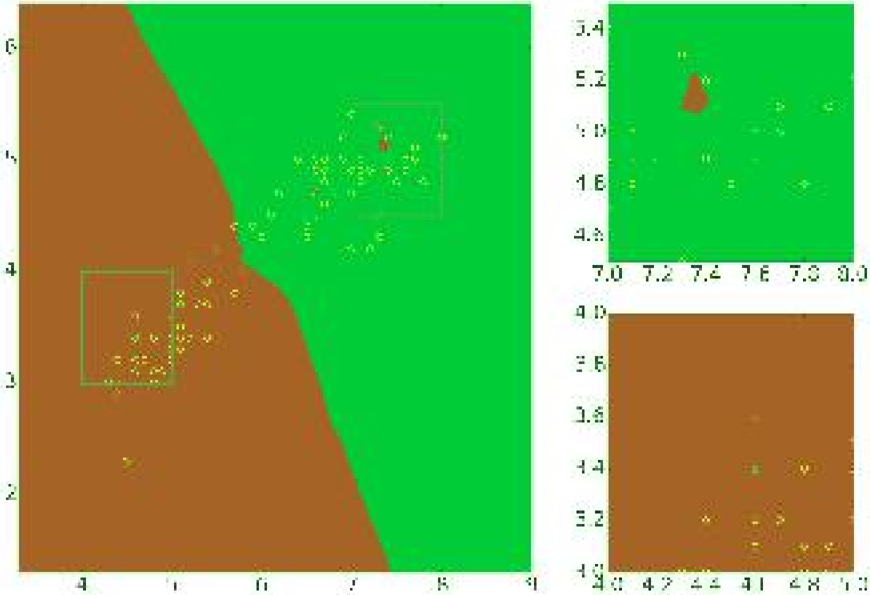




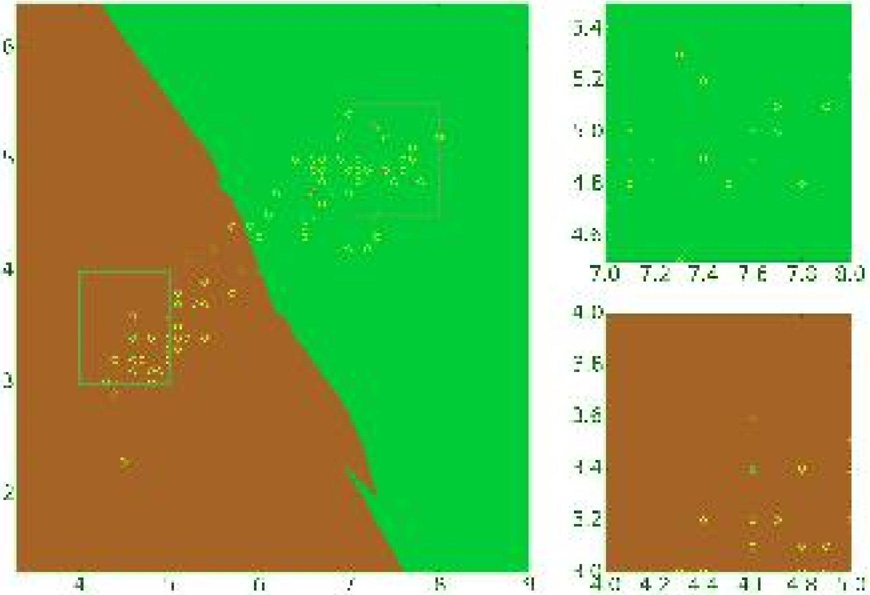
Example of k-Nearest Neighbours Binary Classifier ($k = 1$)



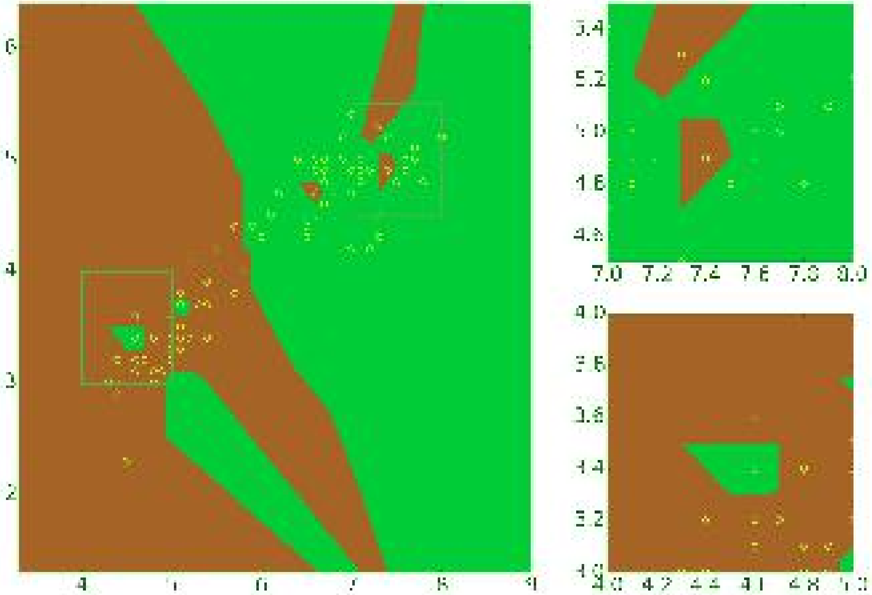
Example of k-Nearest Neighbours Binary Classifier ($k = 3$)



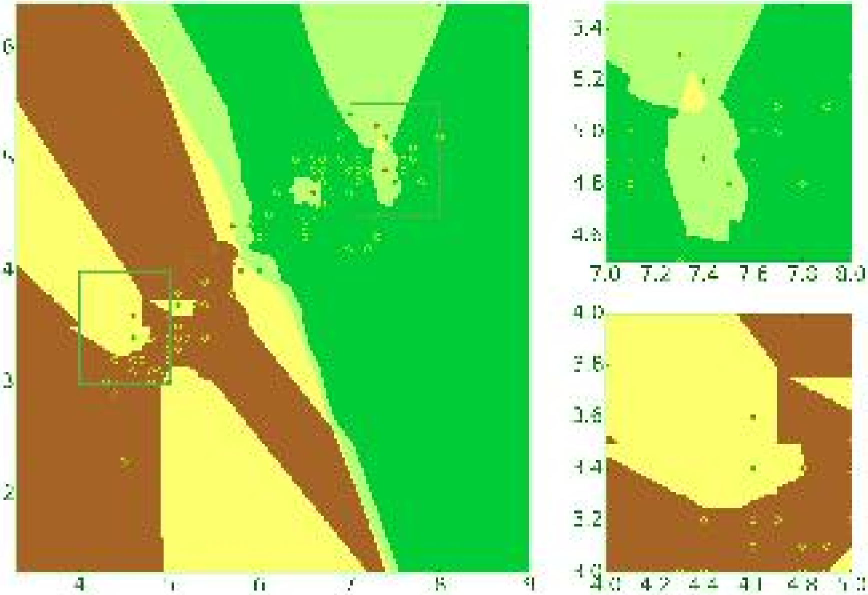
Example of k-Nearest Neighbours Binary Classifier ($k = 10$)



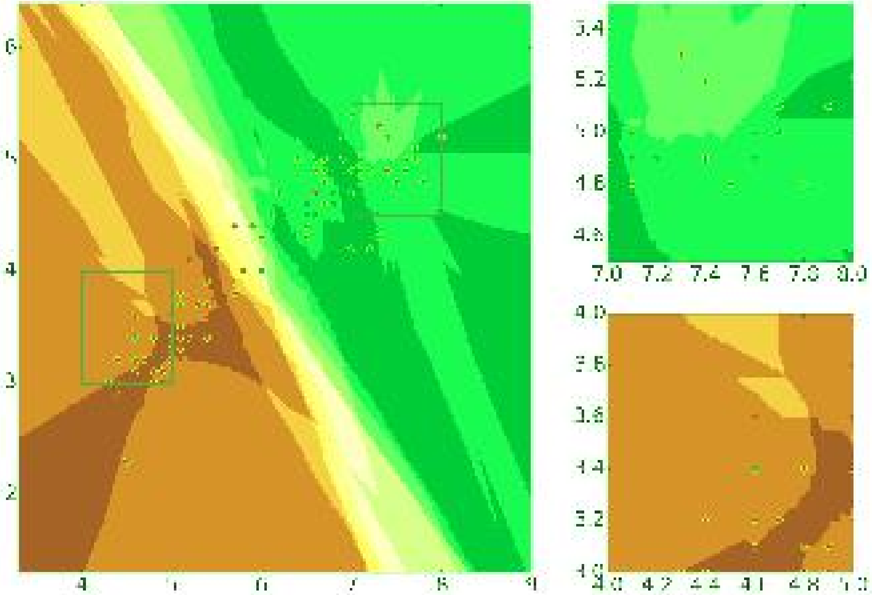
Example of k-Nearest Neighbours Binary Classifier ($k = 1$)



Example of k-Nearest Neighbours Binary Classifier ($k = 3$)



Example of k-Nearest Neighbours Binary Classifier ($k = 10$)



Asymptotic Behaviour of the kNN Classifier

Definition of asymptotic behaviour

properties of the model when the number of instances $n \to \infty$

- typical question: is the model optimal when $n \to \infty$?
- beware: in practice, every dataset is finite (even big data)

Asymptotic misclassification rate with $k = 1$

bounds on the misclassification rate:

$$P_e^* \leq P_e \leq P_e^* \left(2 - P_e^* \frac{C}{C-1} \right)$$

where P_e^* is the Bayes probability of error and C is the number of classes

the nearest neighbour error rate is bounded by twice the Bayes error rate

Pros and Cons of the kNN Classifier

Advantages

- ✓ easy to understand, simple to implement
- ✓ no time-consuming learning procedure (*lazy learning*)
- ✓ gives (surprisingly) good results and is rather robust
- ✓ can be generalised to non-Euclidian distances
- ✓ probabilistic version $p(y|\mathbf{x}) = \#(\text{neighbours of } \mathbf{x} \text{ with label } y)/k$

Potential issues

- × computational cost of prediction: $\mathcal{O}(n)$
- × memory usage for data storage: $\mathcal{O}(n)$
- × not suitable for descriptive modelling
- × what if one of the features is more important ?

Extension of kNN Models

Distance-based weighting schemes

$$p(y|x) = \frac{1}{k}\sum_{s=1}^{k} \mathbf{I}\left[t_{i_s} = y\right] \quad \Rightarrow \quad p(y|x) = \frac{\sum_{s=1}^{k} w_{i_s}\mathbf{I}\left[t_{i_s} = y\right]}{\sum_{s=1}^{k} w_{i_s}}$$

where e.g.

$$w_{i_s} = \frac{d(x, x_{i_k}) - d(x, x_{i_s})}{d(x, x_{i_k}) - d(x, x_{i_1})} \quad \text{or} \quad w_{i_s} = \frac{1}{d(x, x_{i_s})^2}$$

Choice of the distance metric

Manhattan/Mahalanobis distance, metrics for non-vectorial data, etc

Efficient implementations

kd-trees with cost $\mathcal{O}(\log n)$ for $d \leq 10$, ball-trees for high-dimensional data

k-Nearest Neighbours for Regression

Training of a kNN for regression

Input: dataset $\mathcal{D} = \{(\mathbf{x}_i, t_i)\}$
Output: kNN classifier

store the dataset for future predictions

Prediction with a kNN for regression

Input: new instance $\mathbf{x}$
Output: predicted target value y

find the k nearest neighbours of $\mathbf{x}$ in the training set $\mathcal{D}$: $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}$
return the average target value for the neighbours $y = \frac{1}{k} \sum_{s=1}^{k} t_{i_s}$

k-Nearest Neighbours for Regression

Training of a kNN for regression

Input: dataset $\mathcal{D} = \{(\mathbf{x}_i, t_i)\}$
Output: kNN classifier

store the dataset for future predictions

Prediction with a kNN for regression

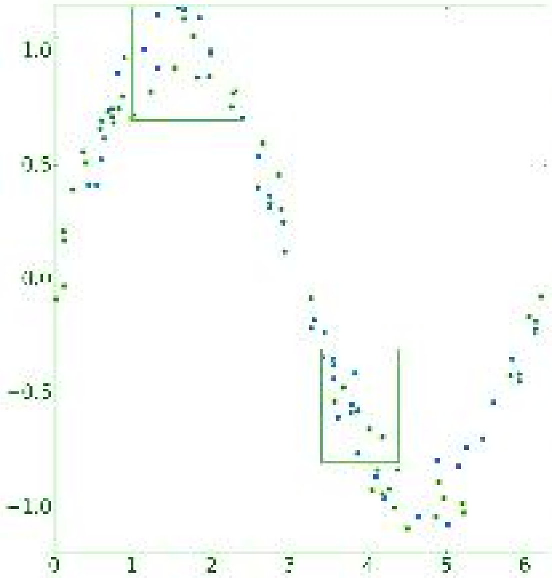
Input: new instance $\mathbf{x}$
Output: predicted target value y

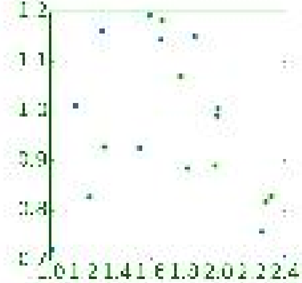
find the k nearest neighbours of $\mathbf{x}$ in the training set $\mathcal{D}$: $\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k}$
return the average target value for the neighbours $y = \frac{1}{k} \sum_{s=1}^{k} t_{i_s}$

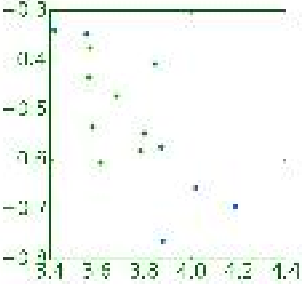
Learning bias

target value of an instance is close to the target value of nearby instances

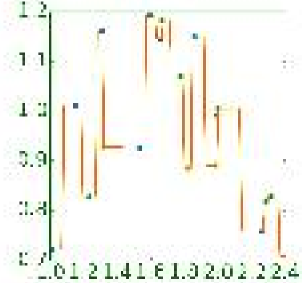
Example of k-Nearest Neighbours Regression (data)

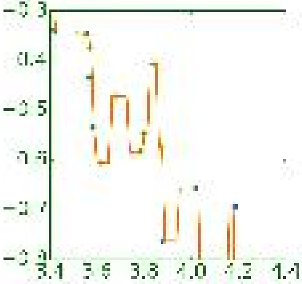




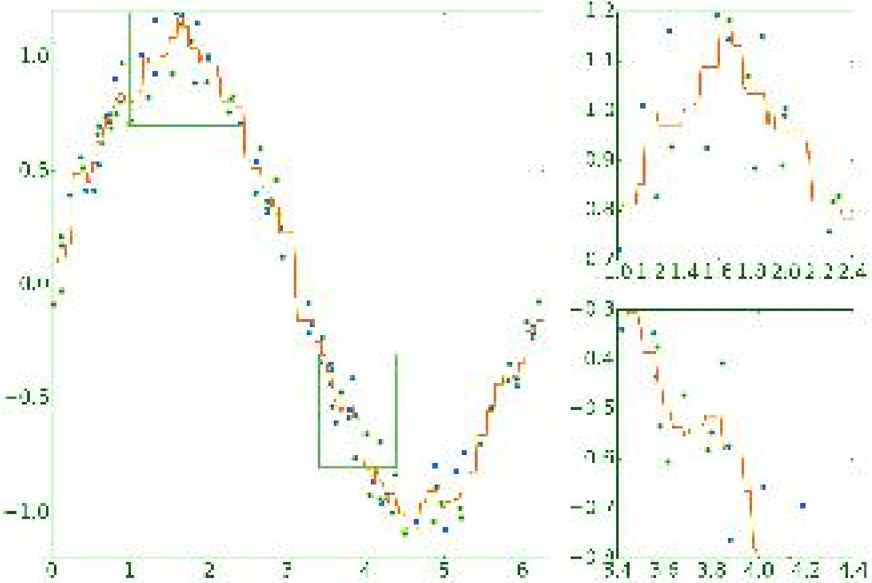


Example of k-Nearest Neighbours Regression ($k = 1$)

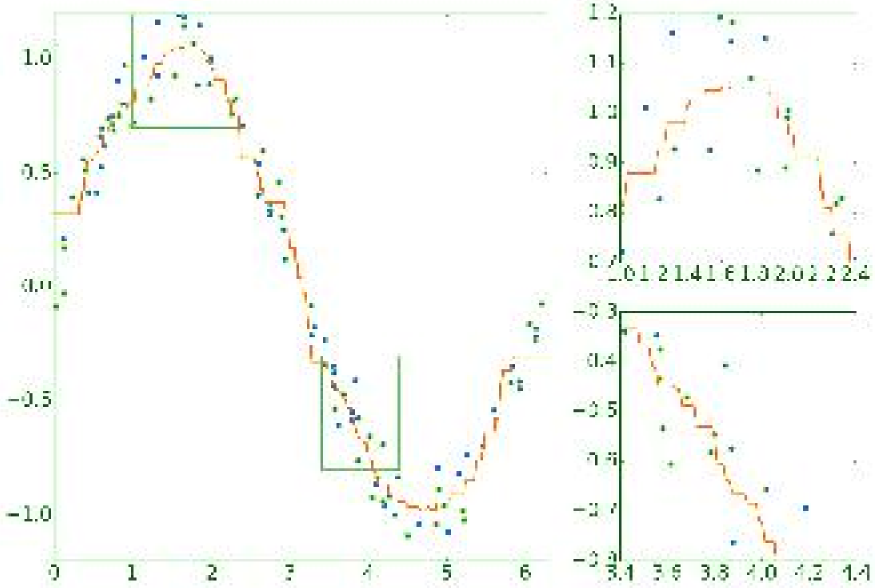




Example of k-Nearest Neighbours Regression ($k = 3$)



Example of k-Nearest Neighbours Regression ($k = 10$)



Decision Trees

Automation of Rule-based Reasoning

How is classification performed by humans ?

- human experts often think in terms of rules (e.g. in medicine)
- powerful way to express expert knowledge ⇒ descriptive model

Examples of rules

- **if** (client_age < 23) ∧ (has_car = false) **then** product = voice_3G
- **if** (client_age > 65) ∧ (has_car = true) **then** product = voice_only
- **if** (client_age < 15) ∧ (prepaid = true) **then** product = text_only

Issues with rules

- not easy to read (imagine a large-scale real-world diagnostic system)
- rules are hard to obtain ⇒ what if we can obtain them from data ?

Simple Example of Decision Tree

Set of rules

- if (PC starting) then (use PC)
- if (PC not starting) ∧ (PC not plugged in) then (plug PC in)
- if (PC not starting) ∧ (PC plugged in) then (call technical service)

Definition of Decision Trees

Types of nodes

- root node: at the top of the tree, no incoming edges
- internal node: one incoming edge and at least two outgoing edges
- leaf/terminal nodes: one incoming edge, but no outgoing edges

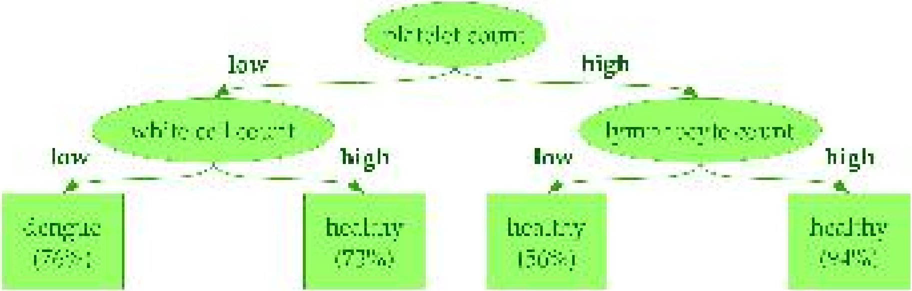
How to read a decision tree (top-bottom)

- a decision always starts in root node
- the root and each internal node corresponds to one feature
- each outgoing edge corresponds to a feature value
- each leaf corresponds to one of the possible decision

Simplified Decision Tree for Dengue Fever

Dengue fever diagnosis

- target concept: does the patient have dengue fever ?
- available features: count of lymphocytes, platelets and white cells
- possible value for each feature: low or high (binary tree)



ID3($\mathcal{D}$, $\mathcal{F}$)

Input: dataset $\mathcal{D} = \{(\mathbf{x}_i, t_i)\}$ and set $\mathcal{F}$ of features
Output: recursive decision tree classifying $\mathcal{D}$ with features in $\mathcal{F}$

if all instances have the same label t **then**
  return a node with label t
else if the set of features $\mathcal{F}$ is empty **then**
  return a node with label t = majority label t in $\mathcal{D}$
else
  create a node where decisions will use the best feature X_j in $\mathcal{F}$ w.r.t. $\mathcal{D}$
  for each feature value v of X_j **do**
    if $\mathcal{D}_v = \{\mathbf{x}_i \in \mathcal{D} | x_{ij} = v\} \neq \emptyset$ **then**
      add child ID3($\mathcal{D}_v$, $\mathcal{F} \setminus \{X_j\}$) to the current node
    else
      add child to the current node with label t = majority label t in $\mathcal{D}$
    end if
  end for
  return current node
end if

Prediction with a Decision Tree

decision_tree_classify(r, x)

Input: root of decision tree r, new instance x
Output: predicted class y

if r is a leaf (single-node tree) with label t **then**
  return class $y = t$
else
  let X_j be the decision feature associated with r
  let c be the child of r on the branch $X_j = x_j$

  return class y = decision_tree_classify(c, x)
end if

important: the prediction algorithm is independent of the learning algorithm

Splitting Criteria for Decision Trees

How do we choose the "best feature X_j in $\mathcal{F}$ w.r.t. $\mathcal{D}$" ?

- the ID3 algorithm does not explain how to choose decision features
- however, this choice determines the quality of the decision tree

Information gain

- measures how well a given feature separates training instances
- information gain = reduction in impurity when a given feature is used
- question: how can we measure "impurity" ?

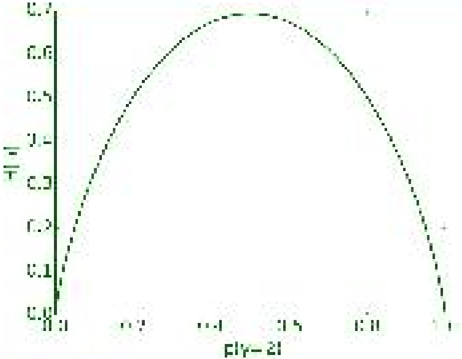
Splitting Criteria for Decision Trees

Definition of the entropy

the Shanon entropy of the probability distribution $p(Y)$ on class Y is

$$H[p] = -\sum_{y \in Y} p(y) \log p(y)$$

notation: $H[p]$ is a *functional* that returns a scalar for any function p(y)



Splitting Criteria for Decision Trees

Information gain

expected reduction in entropy if instances are partitioned using feature X_j

$$\text{gain}(\mathcal{D}, X_j) = H[p] - \sum_{v \in X_j} \frac{|\mathcal{D}_v|}{|\mathcal{D}|} H[p_v]$$

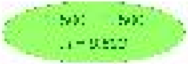
where p is the class distribution in $\mathcal{D}$ and for each value v of feature X_j

$$p_v(t|\mathbf{x}) = \text{percentage of instances of class } t \text{ in } \mathcal{D} \text{ with } X_j = v$$

Other solutions

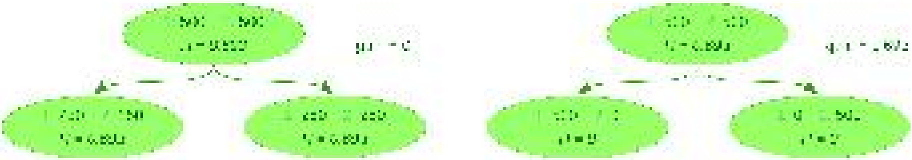
- there exist many other definitions to measure the impurity
- information gain can be extended for non-uniform classification costs

Example of Splitting Evaluation with Information Gain

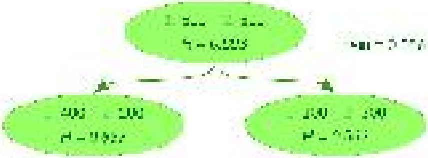


Example of Splitting Evaluation with Information Gain

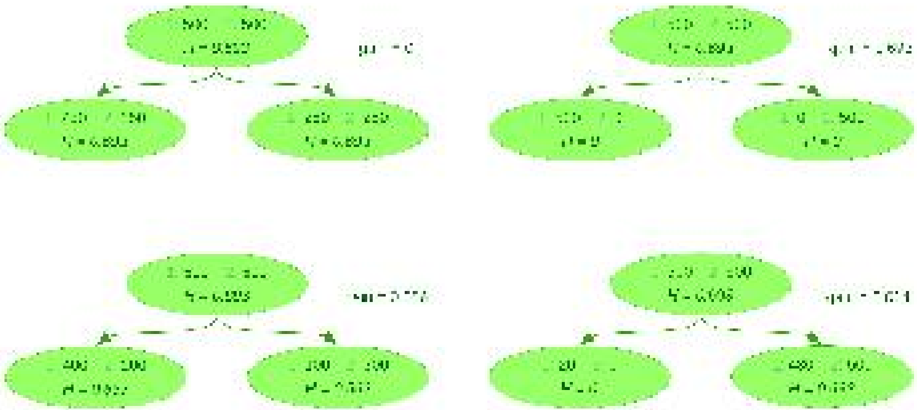
Example of Splitting Evaluation with Information Gain



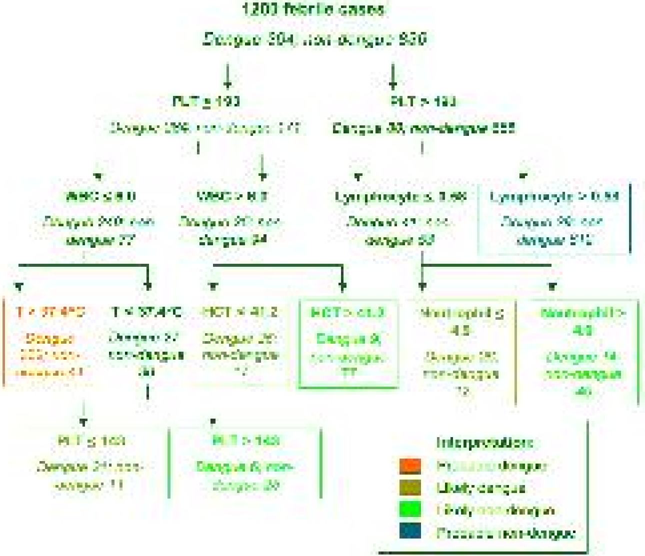
Example of Splitting Evaluation with Information Gain



Example of Splitting Evaluation with Information Gain



Real-World Decision Tree: Dengue Fever (P_e = 15%)



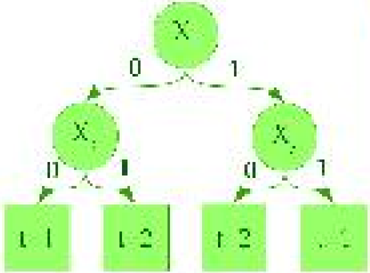
Decision Node Feature

Tanner L, Schreiber M, Low JGH, Ong A, Tolfvenstam T, et al. (2008) Decision Tree Algorithms Predict the Diagnosis and Outcome of Dengue Fever in the Early Phase of Illness. PLoS Negl Trop Dis 2(3): e196.

Rule Extraction from Decision Trees

Automatic Extraction

- each path in the decision tree is a conjunction (internal nodes)
- each conjunction term is a test on the value of a particular feature
- each conjunction is associated with a decision (if-then rule)
- a set of rules can be extracted by considering all possible paths



Extracted rules

- if $(X_1 = 0) \wedge (X_2 = 0)$ then $(t = 1)$
- if $(X_1 = 0) \wedge (X_2 = 1)$ then $(t = 2)$
- if $(X_1 = 1) \wedge (X_3 = 0)$ then $(t = 2)$
- if $(X_1 = 1) \wedge (X_3 = 1)$ then $(t = 1)$

+ probabilities if leaves are not "pure"

Pros and Cons of Decision Trees

Advantages

- easy to understand, simple to implement
- efficient learning procedure, can be performed online
- can be used for predictive/descriptive modelling
- easy to explain to non-experts in machine learning
- can be extended to real variables (e.g. binary split $x > v$)

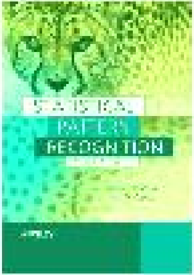
Potential issues

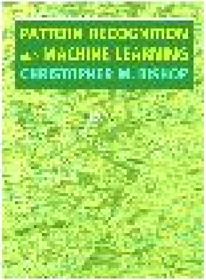
- number of nodes can increase very quickly for large datasets
- finding the smallest tree is NP-complete (ID3 is a greedy heuristic)
- limited expressiveness (only one variable at a time)

Outline of this Lesson

- k-nearest neighbours
- decision trees

References


STATISTICAL
PATTERN
RECOGNITION


PATTERN RECOGNITION
MACHINE LEARNING
CHRISTOPHER M. BISHOP