Data Mining

Lab 3: Data classification

Outline:

- The classification problem
- Decision trees
- Rule-based classifiers (OneR, JRip)
- Instance based classifiers (k- NearestNeighbor)
- Naïve Bayes
- Neural Networks - Multilayer Perceptron
- Support Vector Machines

1. Classification

Quick reminder of main concepts:

- *Aim of a classifier*: establish to which class the input data belongs
- *Constructing a classifier*: extract a classification model using the data from a training set; the construction process is called supervised training/learning
- *Performance evaluation*: confusion matrix, accuracy, sensitivity, specificity, recall, precision, F-measure, Kappa statistics, ROC curve, AUC – area under ROC curve

Confusion matrix (in the context of binary classification: positive vs. negative class):

TP = True Positive = number of data correctly classified in the positive class
TN = True Negative = number of data correctly classified in the negative class
FP = False Positive = number of data incorrectly classified in the positive class
FN = False Negative = number of data incorrectly classified in the negative class

Accuracy = (TP+TN)/(TP+TN+FP+FN) = nr of correctly classified data/total nr of data

Sensitivity = TP/(TP+FN) (TP rate or recall)

Specificity= TN/ (TN+FP) (TN rate), 1-specificity=FP/(TN+FP) = FP rate

Precision = TP/(TP+FP) (nr of true positive cases/ nr of cases classified as positives)

F-measure = 2*precision*recall/(precision+recall)

Kappa = (Accuracy – ExpectedAcc)/(1-ExpectedAcc) - adjustment of the accuracy measure which takes into account the rate of generated right outputs by chance – appropriate especially in the case of unbalanced datasets.

ExpectedAcc= rate(positive observed and true)+rate(negative observed and true) = $(TP+FP)(TP+FN)/N^2 + (TN+FN)(TN+FP)/N^2$ (N=total nr of data)

ROC (Receiver Operating Characteristic) Curve: graphical representation of TP rate (sensitivity) vs. FP rate (1-specificity). It is constructed by computing pairs like (FP rate, TP rate) for each fold of the data set (in the context of cross-validation) and the points (0,0), P_1, …, P_n, (1,1) (where P_i is the point corresponding to the i-th value in increasing ordet of the FP rate). Ideally would that all points have the coordinates (0,1), and the area under the curve equals the area of the square defined by (0,0) and (1,1), i.e. 1.

For the unbalanced datasets (the number of examples in one class is significantly smaller than the number corresponding to other classes) one use the Matthew correlation coefficient (MCC):

MCC=(TP*TN-FP*FN)/sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Exercise 1. Get familiar with "caret" package – Classification And Regression Training (it offers an unified interface to a large number of classification and regression algorithms – around 200 algorithms - implemented in various R packages). Main facilities:

- partitioning of datasets in training and testing subsets s.t. the class distribution from the original dataset is preserved (function createDataPartition)
- training a model (function train) and defining a procedure controlling the training process (function trainControl); it allows to specify the strategy of controlling and tuning the hyper-parameters (parameters of the classification models): bootstrapping (default), cross-validation etc.
- possibility to include a pre-processing step in the training phase (e.g. function preProcess or parameter preProc the of train function); types of pre-processing options/methods: centering ("center"), scaling ("scale"), PCA transformation ("pca"), imputation ("knnImpute"), identification of correlated attributes ("findCorrelation")

(Starting point: templateUsingCaret.r)
https://cran.r-project.org/web/packages/caret/vignettes/caret.pdf)

Decision (classification) trees

A decision tree contains:

- *Internal nodes*: each internal node has an attribute associated and some splitting conditions (each splitting condition corresponds to a branch in the tree)
- *Leaf nodes*: each leaf node has a class associated

Top-down construction (recursive partitioning):

- For the root node one identifies the splitting attribute and the splitting conditions which leads to the highest informational gain (or another criterion, e.g. Gini index) based on the available dataset
- For each branch, recursively apply the same procedure until a stopping condition is identified, e.g. the dataset corresponding to the current node is pure (all data belong to the same class) or it is too small (it contains less than the minimal number of instances corresponding to a splitting node).

Remark: one of the most important aspects when constructing decision trees is the complexity level of the inferred tree. Simpler trees (i.e. fewer nodes) are always preferred. The simplification of the tree (pruning) can be done at different stages:

- During the tree construction (e.g. the branching process is stopped when the number of instances in the current dataset is smaller than a given threshold)

- After the tree construction some subtrees can be replaced with leaf nodes (as long as the classifier performance is not significantly decreased)

Decision tree usage: for a given instance identify the tree branch which matches the corresponding attribute values; the label of the leaf node in that branch will be the predicted class.

R packages: rpart, party

Weka implementation: ID3, J48

Exercise 2/ R: Construct, visualize and evaluate the performance of classification trees for: (a) iris dataset; (b) Titanic dataset; (c) Breast Cancer dataset. Usage of: party, rpart and caret packages

Starting point: DecisionTrees_PartyRpart.R, DecisionTrees_caret.R

Exercise 2/ Weka: Comparing algorithms for decision trees construction

a) Open in Weka the file "weather_nominal.arff". Using Weka Explorer compare the performance of the following classifiers: Id3, J48 (in this cased both the "pruned" and "unpruned" variants are analyzed – this option is set in the list of parameters of J48)
b) Open in Weka the file "weather_numeric.arff" and try to apply the same processing steps as at (a).
c) Open in Weka the file "glass.arff" and analyze the impact on the J48 performance of the minimal number of objects which correspond to a leaf node (the value of the parameter minNumObj)

2. Rule based classifiers

The classification rules have the following structure: IF <antecedent> THEN <class label>, where the antecedent part contains conditions concerning the attribute values (the individual conditions are combined using conjunction and/or disjunction operators)

Construction of the classification rules set. There are two main variants:

- Starting from a *decision tree*: from each branch one construct a classification rule (the conditions in the antecedent are obtained by using the conjunction operator)
- Directly from the data: by using the so-called *covering algorithms* (e.g. RIPPER algorithm)

R packages: caret includes some rule based classifiers through RWeka (JRip and OneR)

Weka implementation:

- Starting from decision trees: M5rules, PART
- Covering algorithms: OneR, JRip

Exercise 3/ R: Construct, visualize and evaluate the performance of rule-based classifiers for: (a) iris dataset; (b) Breast Cancer dataset (c) Titanic dataset. Usage of: caret and RWeka packages

Starting point: ClassificationRules.R

Exercise 3/ Weka: Analysis of algorithms for extracting classification rules

a) Open the file "weather.nominal.arff" and apply successively (using Weka Explorer) the following algorithms: OneR, JRip, M5Rules, PART
b) Identify the best rule based classifier for the dataset "weather.nominal.arff"

3. Instance Based Learning Classifiers

The simplest instance based classifier is the kNN (k nearest neighbours).

Construction:

- The classification model consists of the training set (there is no specific training step – this is why these classifiers are considered lazy classifiers)

Usage:

- For an input data the class is predicted based on the following steps:
  - Step 1: Find the most similar k instances from the training set
  - Step 2: Identify the dominant class among the classes corresponding to the k instances selected at the previous step

R packages: package caret/ method knn, package class / methods knn, knn1

Weka implementation: IB1 (only 1 neighbour is used), IBk (k neighbours are used)

Exercise 4/ R: Construct and evaluate the performance of a nearest neighbor classifier for: (a) Breast Cancer dataset; (b) Titanic dataset.

(i) Analyze the influence of the attributes types and of pre-processing (e.g. standardization) on the classification performance and on the learned classifier parameter (k)
(ii) Visualize the ROC curve and analyze the value of AUC.

Starting point: kNN.R

Exercise 4/ Weka: Choose the "right" number of neighbors for kNN

a) Open in Weka the file "breast-w.arff". For IBk identify the value(s) of k (out of {1,2,3,4,5,6,7,8,9,10}) which lead(s) to the best classifier performance (Hint: modify the value of k from IBk).
b) Open in Weka the file "glass.arff" and apply the same processing steps.

4. Probabilistic models

They are based on the estimation of the probability that a data belongs to a class ($P(C_k|(a_1,a_2,\ldots,a_n)$) knowing the probabilities of observing values of the attributes in different classes, $P(a_i| C_k)$ – these probabilities are computed during the training process based on the training set. The class having the maximal probability will be considered the class to which the data belongs. The probability of each class is computed using the Bayes formula: $P(C_k| a_1,a_2,\ldots,a_n)=P(a_1,a_2,\ldots,a_n | C_k)P(C_k)/ P(a_1,a_2,\ldots,a_n)$.

Examples:

- Naïve Bayes: it is based on the simplifying assumption that all attributes are independent (consequently the probability to observe a data instance is the product of the probabilities to observe each of the attributes, i.e. $P(a_1,a_2,\ldots,a_n)=P(a_1)P(a_2)\ldots P(a_n)$).
- Bayesian Network (Belief network): it allows to describe the relationships between attributes by using a directed acyclic graph.

Exercise 5/ R: Construct and evaluate the performance of a naïve Bayes classifier for: (a) Breast Cancer dataset; (b) Titanic dataset.

Starting point: NaiveBayes.R

Exercise 5/ Weka:

a) Open the file "weather.nominal.arff". Apply the classifiers Naïve Bayes and Bayes Net
b) Use the Bayes Network Editor (from Weka GUI Choser -> Tools) to construct a Bayesian network. Main steps:
    - Select the dataset: Tools->Set Data
    - Place the nodes (each attribute, including that corresponding to the class will have an associated node) by using Edit->Add node
    - Specify the relationships by using Edit->Add arc (for instance one can specify an arc from the class node to the nodes corresponding to the attributes)
    - Network training (Tools->Learn network (this corresponds to the learning of the structure), Tools->Learn CPT (this corresponds to the computation of the tables with probabilities))
    - Usage of the Bayes network classifier: select the values which correspond to the instance to be classified (by right clicking on each node associated to an attribute and by using Set evidence); the decision is based on the values of the probabilities associated to the class.

Remark: details on using the Weka implementation of Bayesian networks can be find at http://www.cs.waikato.ac.nz/~remco/weka_bn/

5. Neural networks

The main steps in designing a neural network for classification are:

- *The choice of the architecture.* In the case of multi-layer perceptrons one have to choose:
    - The number of input units = the number of attributes
    - The number of hidden layers and units – it depends on the complexity of the problem (there is no unique solution/ best recommendation) – simplest approach: use only one hidden layer
    - Number of output units:
        - Binary classification: one unit (the result is interpreted by using a threshold value, e.g. if the output value is larger than 0.5 then class 1 otherwise class 2) or two units (the unit which produces the larger value will indicate the class – this variant can be easily extended to multiple classification)
        - Multiple classification: The number of output units is equal to the number of classes (the unit producing the highest value indicates the predicted class).

- *The choice of the activation functions*. For the neural networks to be trained using the backpropagation algorithm the activation functions should be differentiable, thus the typical choices are:
    - The hidden units have sigmoidal activation functions (logistic or tanh function).
    - The output functions have sigmoidal (logistic function is the default variant) or linear functions or softmax (output of unit i: $y_i=\exp(z_i)/(\exp(z_1)+...+\exp(z_K))$ – in the case of K classes, i.e. K output units)

- *The choice of the training algorithm and of its parameters*. In the case of the backpropagation algorithm the typical parameters (also used in Weka and in nnet package of R – included in caret) are:
    - Number of training epochs
    - Learning rate
    - Coefficient of the momentum term

R implementation: nnet package (included in caret) – one hidden -layer neural network (with various activation functions for the output layer – logistic (default), linear, softmax) trained using a Backpropagation algorithm aiming to minimize MSE (mean squared error – default) or the cross-entropy (when softmax activation function is used). Included also in Rattle.

Weka implementation: MultilayerPerceptron = feedforward neural network trained using Backpropagation (the variant with momentum term); the network architecture has by default one hidden layer but the number of layers (and of units per each layer) can be changed.

Exercise 6/ R: Construct and evaluate the performance of a neural network based classifier for: (a) Breast Cancer dataset; (b) Titanic dataset.

Starting point: NeuralNetworks.R

Exercise 6/ Weka: Analysis of the network architecture on the classification performance

a) Open the file "breast-w.arff" \
b) Train a neural network using the default settings (one hidden layer with K=(nr attributes+nr classes)/2 units)
a) Compare the performance of the classification (accuracy values) for the following values of the Hidden Layers (H) parameter:
    a. 'a' (K=(nr attributes+nr classes)/2)
    b. 'i' (K=nr attributes)
    c. 'o' (K=nr classes)
    d. 't' (K = nr attributes+nr classes)
    e. 4,2 (two hidden layers with 4 and 2 units, respectively)
b) Analyze the influence of the learning rate and of the coefficient of the "momentum" term.
c) Compare the results obtained by using the MultilayerPerceptron with that obtained by using a RBF network – radial basis function network (functions->RBF Network)

Remark. By activating the option GUI from Multilayer Perceptron one can visualize the network architecture.

6. Classification using Support Vector Machines (SVM)

The Weka implementation of SVM uses a fast algorithm for solving the optimization problem involved in the estimation of the weights corresponding to support vectors (Sequential Minimal Optimization). The standard SVM is designed for binary classification. In the case of the classification in M classes the classification problem is transformed in several binary classification problems.

Exercise 7/ R: Construct and evaluate the performance of a naïve Bayes classifier for: (a) Breast Cancer dataset; (b) Titanic dataset.

Starting point: SVM.R

Exercise 7/Weka:

a) Open in Weka the file "breast-w.arff" and use SVM (functions->SMO); analyze the influence of the kernel function on the classification accuracy (variants: PolyKernel, NormalizedPolyKernel, RBFKernel)
b) Open in Weka the file "arrhythmia.arff" and apply the SVM classifier; try to solve the same problem using a neural network (multilayer perceptron or RBF network).

Homework:

1. Use Rattle or R (at your choice) to construct and compare the performance of classifiers based on (i) decision tree; (ii) support vector machine; (iii) neural network.
   Datasets: (i) Glass (from mlbench); (ii) LetterRecognition (from mlbench); German credit data [http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29]