Master program: ***Embedded Systems***

MACHINE LEARNING – Advanced group

Topic 1. Decision Trees Learning

Description:

Develop an application that implement the Decision Trees classifier algorithm for dataset attached at this subject on my web-page.
The dataset, first time need to be pre-processed in order to convert all attributes from the numeric domain into discrete domain. I recommend converting each attribute in only 3 different domains.

After pre-processed the dataset, split the dataset into two datasets (files). One set, called "training set", that contain approximately 70% samples from the initial set (randomly selected) and the remaining data put into the "testing set". Use the training set for training and building the decision tree.

After build the trees, implement the testing part, the part where we evaluate the performance of the algorithm using the testing set. In this part I recommend to use the accuracy as measure for evaluation. The accuracy can be computed as number of samples correct classified divided by total number of samples from the testing file.

Bibliography:

1. Tom Mitchell, Machine Learning, Publisher McGraw-Hill Science, **ISBN:** 0070428077, 1997
2. http://en.wikipedia.org/wiki/Decision_tree
3. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/mlbook/ch3.pdf

Topic 2. Text Document classification using Naïve Bayes

Description:

Develop an application that implement the Naïve Bayes classifier algorithm for text documents classification. For this subject the student will receive pre-processed text files (Reuter's dataset). The documents are represented as sparse words vectors (use the link from my web-space) that contain the word frequency from each document (the word is called attribute in the files). Also needs to implement the testing part for evaluate the performance of the algorithm (use measure like accuracy described before).
The dataset contain 4 text files, 2 file that use only 100 attributes for representing the documents (one file for training and one file for testing) and 2 files that use 1309 attributes for representing the documents.

The training/testing file structure

The file has a first part (lines that starts with "#" symbol) that contains information about number of documents (samples) from that file, number of attributes used to represent the samples and number of topics. The files continue whit part containing attributes, a part containing topics (classes) and a part containing data. In the attributes part there are specified all attributes (words, tokens) that characterize all the documents from dataset. The attributes are specified using the letterhead *"@attribute*" followed by the index of the attribute. In the topic part there are specified all the topics (classes) for this set according to Reuter's classification. The topics are specified using the letterhead "@topics" followed by the name of the topic and the number of samples that contain that topic. The data section is marked using "@data" and contains all large frequency vectors stored as sparse vectors.

In the frequency vector are stored only values that are greater than zero as frequency in the current document. Each pair of values, delimited by " " (space symbol) contain before ":" the attribute number, attributes been in the order presented in the attribute part of file and after ":" the number of occurrences of this attribute in the current document (for simplify the computation I recommend that for all value greater then 1, put the value 1 - we use binary representation) . After the "#" symbol at the end of the line introduces Reuter's classification topics. One document can have one or more than one class. For this application I recommend to take in consideration only the first class that occurs after the "#" symbol.

Bibliography:

1. Tom Mitchell, Machine Learning, Publisher McGraw-Hill Science, ISBN: 0070428077, 1997
2. http://en.wikipedia.org/wiki/Data_mining
4. http://en.wikipedia.org/wiki/Naive_Bayes_classifier

Topic 3. Automatic numbers recognition using the Hopfield network

Description:

Develop an application that allows you to draw into a grid with size of 10*10 different numbers. The application also will implement the Hopfield algorithm to train with different numbers drawn in the grid and after that automatically recognition the numbers, even if there are in testing part some mistakes in drawing numbers.

Bibliography:

1. http://en.wikipedia.org/wiki/Hopfield_net
2. http://www.eee.metu.edu.tr/~alatan/Courses/Demo/Hopfield.htm
3. http://www.cbu.edu/~pong/ai/hopfield/hopfieldapplet.html