Last updated on Monday, May 9, 2016

2016 CAB320 - Assignment Two (Machine Learning)

Assessment information

- Code and report submission at the end of **Week 13 (Monday 6th June, 08.30am)**
- Use **Blackboard** to submit your work
- Group size: 1 or 2 people per submission

Overview

- You are provided with a data set of anonymized credit applications.
- The aim of this assignment is to build some classifiers and evaluate their performance.
- The classifiers that you will implement are
    - a *naive Bayes classifier*
    - a simple *decision tree classifier*
- Decision trees are not a great choice for a feature space with complex relationships between numerical variables, but are often a good option for data with a simpler mix of numerical and categorical inputs.

Dataset

The records are stored in a text file named “records.txt”. This file concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.

This dataset is interesting because there is a good mix of attributes; continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values (marked with the character '**?**').

Attribute Information:

There are 16 attributes. In the following, we call them A1,A2,...,A16. The class attribute is A16. This variable can take the value '**+**' or '**-**'.

A1:	b, a.
A2:	continuous.
A3:	continuous.
A4:	u, y, l, t.
A5:	g, p, gg.
A6:	c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.
A7:	v, h, bb, j, n, z, dd, ff, o.
A8:	continuous.
A9:	t, f.
A10:	t, f.
A11:	continuous.
A12:	t, f.
A13:	g, p, s.
A14:	continuous.
A15:	continuous.
A16:	+,- (class attribute)

Your tasks

Preprocessing (5 marks)

Write a Python script (with the file named ***task_1.py***), that

- loads the data records from the text file, and converts the information to a format suitable for numpy arrays. Nominal (discrete) attribute values should be converted to integers. Replace any missing value with the most likely value for the corresponding attribute.
- creates two numpy arrays; one called *data* and another called *attribute_type.* The numpy array *data* is two dimensional (2D) and has 16 columns and as many rows as there are records in the text file "records.txt". The 1D numpy array *attribute_type* should code the type of the attributes as follows. If the ith attribute takes *n* different discrete/nominal values, then *attribute_type[i]* should be equal to *n*. For example, *attribute_type[15]* should be 2 as the attribute with index 15 is the binary class label ('-' or '+'). If the ith attribute is a real valued attribute, then *attribute_type[i]* should be equal to *-1* by convention.
- shuffles the data, and create two numpy arrays *train_data* and *test_data* which contain respectively 80% and 20% of the shuffled data.

Naive Bayes Classifier (5 marks)

Write a Python script (with the file namedOne of the simplest forms of pruning is reduced error pruning. Starting at the leaves, each node is replaced with its most popular class. If the prediction accuracy is not affected then the change is kept. While somewhat naive, reduced error pruning has the advantage of simplicity and speed. ***task_2.py***), that

- builds a naïve Bayes classifier based on the data contained in the numpy array *train_data* created by the script task_1.py
- evaluates the prediction errors on *train_data* as well as on *test_data.*
- reports these errors by printing the proportion of misclassified inputs.

Create decision trees using these features (10 marks)

Write a Python script (with the file named ***task_3.py***), that

- builds a decision tree classifier based on the data contained in the numpy array *train_data* created by the script task_1.py
- evaluates the prediction errors on *train_data* as well as on *test_data.*
- reports these errors by printing the proportion of misclassified inputs.
- Build a decision classifier and report its generalization error.

The decision tree should be built recursively using an information gain criterion. The growing of the tree should stop when the training set becomes too small. Pruning can reduce the size of a learning tree without reducing the predictive accuracy as measured by a cross-validation set. One of the simplest forms of pruning is reduced error pruning. Starting at the leaves, each node is replaced with its most popular class. If the prediction accuracy is not affected then the change is kept. While somewhat naive, reduced error pruning has the advantage of simplicity and speed.

Deliverables

You should submit via Blackboard a zip file containing

1. A README.TXT file containing a *statement of completeness* (concise description of what you have implemented, what works, what doesn't).
2. All Python files with proper commenting. Good code documentation is critical as no report is needed for this assignment.

Draft Marking Scheme

- Code quality (readability, simplicity, structure, genericity, in line documentation):   15 marks
- Tasks as indicated

Final Remarks

- Do not underestimate the workload. Start early. You are strongly encouraged to ask questions during the practical sessions.
- Email questions to f.maire@qut.edu.au

Have fun while learning!