School of Computer Science
The University of Adelaide

Artificial Intelligence
Assignment 2

Semester 1 2020
Due 11:59pm Monday 4 May

> Wall Street is the only place that people ride to in a Rolls Royce to get advice from those who take the subway.
>
> Warren Buffett

1 Credit rating prediction

Financial corporations regularly need to assess the creditworthiness of potential borrowers or debtors. Credit rating agencies (e.g., Standard & Poor's, Moody's, Fitch Ratings) provide a service by assigning *credit ratings* to organisations (companies, sovereign governments, etc.) based on their creditworthiness. The ratings are typically designated using letters, for example, `AAA`, `AA`, `A`, `BBB`, `BB`, `B`, `C` (in decreasing creditworthiness), and each agency has its own set of symbols. Given the fundamental importance of credit, credit ratings play a major role in the financial system.

Credit ratings are assigned by analysing financial information on the borrower. In the case of corporate borrowers, information such as working capital, total assets, total debts etc., are used to assess creditworthiness. More specifically, a set of *financial ratios* of a company are used, for example,

- Working capital / Total Assets (`WC_TA`)
- Retained Earnings / Total Assets (`RE_TA`)
- Earnings Before Interests and Taxes / Total Assets (`EBIT_TA`)
- Market Value of Equity / Book Value of Total Debt (`MVE_BVTD`)
- Sales / Total Assets (`S_TA`)

The grading process is typically done based on expert analysis. Increasingly, however, the assignment of credit ratings is done by automated algorithms, especially for relatively small loans offered to a large market of potential borrowers (e.g., small enterprises). This assignment concerns building an automated credit rating prediction tool.

1.1 Decision tree learning

A sample set of training data is given in the file `previous.txt` (see MyUni). The file contains information acquired from 2,000 companies. The first line is a header containing the names of the columns. Each subsequent line contains values of five attributes (`WC_TA`, `RE_TA`, `EBIT_TA`, `MVE_BVTD`, `S_TA`) of a company, as well as a credit rating (one of `AAA`, `AA`, `A`, `BBB`, `BB`, `B`, `C`) assigned previously by a human expert.

From the given training data, our goal is to learn a function that can predict the credit rating of a new company, based on the financial ratios of the company. In this assignment, the predictor function will be constructed as a decision tree. Since the attributes (financial ratios) are continuous valued, we shall apply the DTL algorithm for continuous data, as outlined in Algorithms 1 and 2. Once the tree is constructed, Algorithm 3 can be used to assign credit rating to a new company.

Algorithm 1 DTL($data$, $minleaf$)

Require: $data$ in the form of N input-output pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$, $minleaf \geq 1$.

if ($N \leq minleaf$) or ($y_i = y_j$ for all i, j) or ($\mathbf{x}_i = \mathbf{x}_j$ for all i, j) **then**
  Create new leaf node n.
  if there is a unique mode (most frequent value) in $\{y_i\}_{i=1}^N$ **then**
    $n.label \leftarrow$ mode in $\{y_i\}_{i=1}^N$
  else
    $n.label \leftarrow$ `unknown`
  end if
  return n.
end if
[$attr$,$splitval$] $\leftarrow$ choose-split($data$)
Create new node n.
$n.attr \leftarrow attr$
$n.splitval \leftarrow splitval$
$n.left \leftarrow$ DTL($data$ with $\mathbf{x}_i[attr] \leq splitval$, $minleaf$)
$n.right \leftarrow$ DTL($data$ with $\mathbf{x}_i[attr] > splitval$, $minleaf$)
return n.

Algorithm 2 choose-split(*data*)

Require: *data* in the form of N input output pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$.

bestgain $\leftarrow 0$
for each *attr* in *data* **do**
  Sort the array $\mathbf{x}_1[attr], \mathbf{x}_2[attr], \ldots, \mathbf{x}_N[attr]$. // Can you speed this up?
  for $i = 1, 2, \ldots, N-1$ **do**
    splitval $\leftarrow 0.5(\mathbf{x}_i[attr] + \mathbf{x}_{i+1}[attr])$
    gain $\leftarrow$ Information gain of $(attr, splitval)$ // See lecture slides.
    if *gain* $>$ *bestgain* **then**
      bestattr $\leftarrow$ *attr* and *bestsplitval* $\leftarrow$ *splitval*
    end if
  end for
end for
return $(bestattr, bestsplitval)$.

Algorithm 3 predict(n, *data*)

Require: Decision tree with root node n, *data* in the form of attribute values $\mathbf{x}$.

while n is not a leaf node **do**
  if $\mathbf{x}[n.attr] \leq n.splitval$ **then**
    $n \leftarrow n.left$
  else
    $n \leftarrow n.right$
  end if
end while
return $n.label$.

1.2 Deliverables

Implement decision trees for credit rating prediction in C/C++, Java or Python. In the case of C/C++, you must supply a makefile (`Makefile`) with a rule called `creditrating` to compile your program into a Linux executable named `creditrating.bin`. Your program must be able to be compiled and run as follows:

```
$ make creditrating
$ ./creditrating.bin [train] [test] [minleaf]
```

In the case of Java, write your program in the file `creditrating.java`. Your program must be able to be compiled and run as follows:

```
$ javac creditrating.java
$ java creditrating [train] [test] [minleaf]
```

In the case of Python (see below on Python version), write you program in the file `creditrating.py`. Your program must be able to be run as follows:

```
$ python creditrating.py [train] [test] [minleaf]
```

The marking program will decide which of the above to invoke using the following structure:

if `Makefile` exists **then**
  Compile and run C/C++ submission.
else if `creditrating.java` exists **then**
  Compile and run Java submission.
else
  Run Python submission.
end if

On Python version At the moment, only an older version of Python (version 2.7.5) is supported on the school servers. If you are not familiar enough with this version of Python, PLEASE DO NOT USE PYTHON for the assignment.

Yes, the school should get on with the times and install the latest version of Python. Yes, Python is a popular programming language for AI, but really mainly for deep learning methods (note that this assignment is not about deep learning). When the opportunity arises feedback will be given to the school to update the Python version.

Explanation of the input parameters

- `[train]` specifies the path to a set of training data (input-output pairs), which is a text file formatted according to the sample `previous.txt` above.
- `[test]` specifies the path to a set of testing data, which is a text file where the first line is a header containing the names of the columns, and each subsequent line contains the five financial ratios of a company; see file `new.txt` (available on MyUni) for an example.
- `[minleaf]` is an integer greater than zero which specifies the second input parameter to the DTL algorithm (Algorithm 1).

Given the inputs, your program must learn a decision tree (following the prescribed algorithms) using the training data, then predict the credit rating of each of the company in the testing data. Your program must then print to standard output (i.e., the command prompt) the list of predicted credit ratings, vertically based on the order in which the testing cases appear in `[test]`.

As an example and sanity test for your program, executing DTL using only the first-100 entries in `previous.txt` as training data with $minleaf = 20$ yields the decision tree show in Figure 1. Note that it is not a requirement to write a function to print a decision tree in the manner in Figure 1 (though such a function may be useful for debugging). Thus you may store your decision tree in whichever internal format/data structure you find convenient, as long as it is time and memory efficient.

1.3 Expected run time

Your program must be able to terminate within 30 seconds on the sample data given.

1.4 Web submission

You must submit your program on the Computer Science Web Submission System. This means you must create the assignment under your own SVN repository to store the submission files. The SVN key for this submission is
`2020/s1/ai/Assignment2`
The link to the Web Submission System used for this assignment is
`https://cs.adelaide.edu.au/services/websubmission/`
For more details on the online submission procedures including SVN, visit the home page of the school and look under "Information for Current Students".

```
1. If MVE_BVTD <= 1.3820, goto 2, else goto 3.
2. If MVE_BVTD <= 0.5745, goto 4, else goto 5.
3. If MVE_BVTD <= 3.2610, goto 6, else goto 7.
4. If RE_TA <= -0.5550, goto 8, else goto 9.
5. If RE_TA <= 0.0645, goto 10, else goto 11.
6. If MVE_BVTD <= 2.1250, goto 12, else goto 13.
7. Return rating AAA.
8. Return rating CCC.
9. Return rating BB.
10. Return rating BB.
11. If EBIT_TA <= 0.0710, goto 14, else goto 15.
12. Return rating A.
13. Return rating AA.
14. If EBIT_TA <= 0.0480, goto 16, else goto 17.
15. Return rating BB.
16. Return rating BBB.
17. Return rating BBB.
```

Figure 1: Decision tree learnt from first-100 entries in `previous.txt` and $minleaf = 20$.

1.5 Due date and late submission policy

This assignment is due by **11:59pm Monday 4 May**. If your submission is late the maximum mark you can obtain will be reduced by 25% per day (or part thereof) past the due date or any extension you are granted.

1.6 Grading

I will compile and run your code on different tests. If it passes all tests you will get **15%** (undergrads) or **12%** (postgrads) of the overall course mark.

There will be no further manual inspection/grading of your program to award marks on the basis of coding style, commenting or "amount" of code written.

1.7 Using other source code

You may **not** use other source code for this assignment. All submitted code must be your own work written from scratch. Only by writing the solution yourself will you fully understand the concept.

Continues next page for postgraduate section.

2 Cross validation

For postgraduate students, completing this section will give you the remaining **3%** of the assignment marks.

A critical parameter in the decision tree learning algorithm is $minleaf$. As the name suggests, this parameter specifies the minimum number of data points "stored" in each leaf node. Effectively $minleaf$ controls tree growth: a larger $minleaf$ value stops the expansion of a branch of the tree earlier than a smaller $minleaf$ value. More fundamentally, $minleaf$ implements a form of tree pruning that controls the complexity of the decision function learned by the tree. To ensure good generalisation to unseen before data, it is vital to set an appropriate value for $minleaf$.

A standard framework to decide the value of such *hyperparameters* in learning algorithms is *cross validation* (also called *K-fold cross validation*). The basic idea is to partition the training data into K mutually exclusive subsets. One of the subsets is sequestered as the "testing data", and the learning algorithm (with a specific setting of the hyperparameters) is invoked on the other $K-1$ subsets to learn a decision function. The decision function is then applied on the unused subset, and the accuracy is recorded (note that since the unused subset is actually from the training data, target labels are available for comparison). The hyperparameter values are then changed and the process is repeated. Algorithm 4 outlines the cross validation method.

The goal in this section is to implement and conduct cross validation to select the appropriate value of $minleaf$ for credit rating prediction.

2.1 Deliverables

Write your program in Java, C/C++ or Python. In the case of C/C++, in the same makefile as in Sec. 1, you must create another rule called `kfoldcross` to compile your program into a Linux executable binary named `kfoldcross.bin`. Your program must be able to be compiled and run as follows:

```
$ make kfoldcross
$ ./kfoldcross.bin [train] [K] [minleaf1] [minleaf2] ... [minleafM]
```

In the case of Java, write your program in the file `kfoldcross.java`. Your program must be able to be compiled and run as follows:

```
$ javac kfoldcross.java
$ java kfoldcross [train] [K] [minleaf1] [minleaf2] ... [minleafM]
```

Algorithm 4 K-fold cross validation

Require: Training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, number of folds K, hyperparameter values to test $h_1, h_2, \ldots, h_M$.

$Z \leftarrow N/K$ // Assume N is divisible by K.
Separate $\mathcal{D}$ into K subsets $\{\mathcal{D}_k\}_{k=1}^{K}$, where each $\mathcal{D}_k = \{\mathbf{x}_i, y_i\}_{i=(k-1)Z+1}^{(k-1)Z+Z}$.
for $m = 1, 2, \ldots, M$ **do**
  $accerr_m \leftarrow 0$
  for $t = 1, 2, \ldots, K$ **do**
    Retain $\mathcal{D}_t$ as the testing data.
    On the other subsets $\{\mathcal{D}_k\}_{k \neq t}$, invoke the learning algorithm with hyperparameter h_m to produce a decision function.
    Test the decision function on $\mathcal{D}_t$ and record percent error as err_t.
    $accerr_m \leftarrow accerr_m + err_t$
  end for
  $avgerr_m \leftarrow accerr_m/K$
end for
return Return hyperparameter with the lowest average error.

In the case of Python (version 2.7.5 only), write you program in the file `kfoldcross.py`. Your program must be able to be run as follows:

```
$ python kfoldcross.py [train] [K] [minleaf1] [minleaf2] ... [minleafM]
```

Explanation of the input parameters

- `[train]` specifies the path to a set of training data (input-output pairs), which is a text file formatted according to the sample `previous.txt` above.
- `[K]` specifies the number of folds to use in cross validation.
- `[minleaf1]`, `[minleaf2]`, ..., `[minleafM]` are different choices of *minleaf* values for the DTL algorithm (Algorithm 1).

Given the inputs, your program must conduct cross validation (Algorithm 4) to select the best *minleaf* value among the choices given. The best *minleaf* value is then printed to standard output.

Besides writing the program, conduct your own experimentation to see what the appropriate values for *minleaf* are on the sample data given in Sec. 1. Upload a file named `crossval.pdf` in PDF format containing a plot of average error versus *minleaf*, for the range $minleaf \in [10, 200]$. Figure 2 shows what the graph should look like.

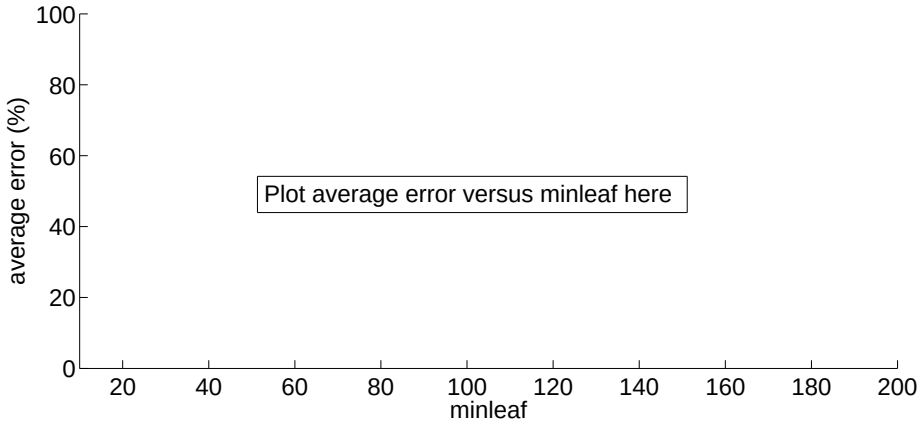

Figure 2: Upload a file named `crossval.pdf` that contains the above graph.

2.2 Submission policies

Submit your program and file in the same way as the previous section. The due date and code reuse policies are also the same.

2.3 Grading

I will test your program manually to see if it satisfies the required functionalities.

∼∼∼ The End ∼∼∼
by Tat-Jun Chin, 29 Mar 2020