15-381 Problem Set 5: Machine Learning

Out: Tuesday, November 7, 2000
Due: Tuesday, November 21, 2000

In this problem set you will be classifying text and mushroom data using Decision Trees and the k-Nearest-Neighbor algorithm. We have generated random groups for the assignment. Please check the class web page to find your new partner. Send email to Rune (runej@cs) if you need to be reassigned to a new group, or if you for some reason were not assigned a partner.

1 Datasets

Text Data

The text database consists of news messages from Reuters. The dataset is referred to as Reuters 21450 and can be found at `http://moscow.mt.cs.cmu.edu:8081/reuters_21450/parc/`. From this set we only consider two classes of news messages: *trade* and *money-foreign-exchange*. An example from the trade class is shown below:

```
SWEDISH PRIME MINISTER'S CHINA VISIT BOOSTS TRADE.

    PEKING, April 3 - A big expansion in bilateral trade is
expected as a result of Swedish Prime Minister Ingvar
Carlsson's visit to China, an official of China's Foreign
Economic Relations and Trade Ministry official said.
    He told the China Daily that petroleum, coal, cotton,
non-ferrous metals and electric engines could help balance the
trade running at over three-to-one in Sweden's favour.
    Total trade reached 290 mln dlrs last year, up 32 pct on
1985, according to Chinese statistics. Swedish sources said
major paper mill projects and an aviation agreement would be
discussed during Carlsson's one week visit.
```

As discussed in class, we use a vector to represent the text information. Each entry in this vector corresponds to a word in a vocabulary. The value of the entry is the number of times the word occurs in the text. The text is preprocessed to reduce the number of entries and increase the quality of the information in the vector . In our case we remove punctuation, commas and numbers. We also get rid of stop-words like *the*, *on* and *of* and normalize words to their base form. After the preprocessing the text is a single list of words. After preprocessing the above example consists of the following list of words:

```
april balance expect mln pct result trade trade trade trade
trade week year agreement economic expansion major minister
minister official official prime prime visit visit visit
daily dlrs foreign ministry petroleum reach statistics total
big china china china china project relation metal sweden
run source electric chinese paper swedish swedish swedish
boost favour discuss accord mill cotton engine bilateral
peking ferrous coal aviation carlsson carlsson ingvar
```

The goal of the classifiers is to detect text examples of the trade class. In the following, we will regard examples from this class as positive and examples from the money-fx class as negative.

The text data files provided for this homework are found in `/afs/cs/academic /class/15381-f00/public/hw5/data/reuters`. There are 5 files:

```
reuters.data.training
reuters.data.holdout
reuters.data.testing
reuters.data.handin
reuters.data.error
```

All of the files contain 50 text examples in preprocessed form. A text example is described by two lines. The first line is the text class: `trade`, `money-fx` or `?`. A question mark denotes that the class is unknown (unlabeled data). The second line is the word list of the preprocessed text.

The `training` and `holdout` dataset contains labeled examples that the learning algorithms can use for classifying new examples. The `testing` dataset contains new examples to be classified. Since you are classifying this dataset, it should be unlabeled. We kept it labeled, though, because we want you to be able to calculate how accurate the classifiers are. The `handin` dataset also contains examples to be classified, but these examples are unlabeled. Finally, the `error` set is a noisy version of the training set where 10% of the labels have been changed.

Mushroom Data

The mushroom database can be found at `http://grial.uc3m.es/~ftp/uci _database/mushroom/`. Each mushroom in the dataset is represented by a tuple. The attributes in the tuple describe the following nominal features of a mushroom:

Attr. #	Attr. Name	Possible Values
0.	toxic?	edible=e, poisonous=p
1.	cap-shape	bell=b,conical=c,convex=x, flat=f,knobbed=k,sunken=s
2.	cap-surface	fibrous=f,grooves=g,scaly=y,smooth=s
3.	cap-color	brown=n,buff=b,cinnamon=c,gray=g, green=r,pink=p,purple=u,red=e, white=w,yellow=y
4.	bruises?	bruises=t,no=f
5.	odor	almond=a,anise=l,creosote=c,fishy=y, foul=f,musty=m,none=n,pungent=p, spicy=s
6.	gill-attachment	attached=a,descending=d,free=f, notched=n
7.	gill-spacing	close=c,crowded=w,distant=d
8.	gill-size	broad=b,narrow=n
9.	gill-color	black=k,brown=n,buff=b,chocolate=h, gray=g,green=r,orange=o,pink=p, purple=u,red=e,white=w,yellow=y
10.	stalk-shape	enlarging=e,tapering=t
11.	stalk-root	bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r
12.	stalk-surface-above-ring	ibrous=f,scaly=y,silky=k,smooth=s
13.	stalk-surface-below-ring	ibrous=f,scaly=y,silky=k,smooth=s
14.	stalk-color-above-ring	brown=n,buff=b,cinnamon=c,gray=g, orange=o,pink=p,red=e,white=w, yellow=y
15.	stalk-color-below-ring	brown=n,buff=b,cinnamon=c,gray=g, orange=o,pink=p,red=e,white=w, yellow=y
16.	veil-type	partial=p,universal=u
17.	veil-color	brown=n,orange=o,white=w,yellow=y
18.	ring-number	none=n,one=o,two=t
19.	ring-type	cobwebby=c,evanescent=e,flaring=f, large=l,none=n,pendant=p,sheathing=s, zone=z
20.	spore-print-color	black=k,brown=n,buff=b,chocolate=h, green=r,orange=o,purple=u,white=w, yellow=y
21.	population	abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
22.	habitat	grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

The goal is to detect edible mushrooms, thus *toxic?* is our target attribute. In the following we define edible examples to be positive and poisonous examples to be negative.

The mushroom data files are found in `/afs/cs/academic/class/15381-f00/public/hw5/data/mushroom`. A naming convention similar to the text data files is used. Each data file has 1000 mushroom examples. Each line in a data file contains a comma-separated mushroom tuple. Attributes have letter values corresponding to the mapping given in the table above. An example of a mushroom tuple is:

```
e,b,s,n,f,n,f,c,b,w,e,b,y,y,n,n,p,w,t,p,w,y,p
```

Part I: Implementation

We ask you to implement a Decision Tree (DTree) and a k-Nearest-Neighbor (kNN) classifier. To help you out we provide C++/STL source code containing data structures for representing datasets and functions for reading input files. Further, a main function is given with a parser for relevant command line options of your program. The files are found in `/afs/cs/academic/class/15381-f00/public/hw5/src/`.

1.1 The DTree Classifier

Download the available source code. Extend it with a recursive function for generating DTrees. You can choose between one of the following three approaches:

1. Grow the tree using both the training set and the holdout set until the leaves are pure. That is, grow the tree until the leaves only contain positive or negative examples. An algorithm for doing this is shown in Figure 18.7 on page 537 of the text book. Notice that we also stop growing the tree in two other cases: 1) if there are no attributes left to test on, or 2) if the set of examples associated with the node is empty. In the former case, we return the majority value of the examples, in the latter, we return some default value.

2. Grow the tree using the training set until the classification accuracy on the holdout set starts decreasing. This approach tries to avoid the DTree from overfitting the training set.

3. Another way to do this is to grow the tree using the training set as described in the first approach. Then for each internal node in the resulting tree calculate how much the accuracy on the holdout set increases by pruning the node from the tree. Prune the node with highest accuracy gain from the tree (when pruning a node it becomes a leaf node with a value equal to the majority of the examples of the node). Continue pruning nodes until no node can be found that increases the accuracy on the holdout set.

Generating a DTree from the mushroom data is straightforward. For the text data, though, we suggest that you change the vector format such that an entry is **0** if the corresponding word doesn't occur in the text, and otherwise **1**. What is obtained by doing this ? What is the price ? (Please hand in your answer for these two questions together with your answers for the questions in part II)

1.2 The kNN Classifier

Extend your program with a kNN classifier. The algorithm is described on slide 9 Nov. 2.

- For the text data you should use the similarity function discussed in class (slide 6 Oct. 24):
$$\mathrm{sim}(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1||v_2|}.$$
For the mushroom data this similarity function doesn't work since the attributes are nominal. Instead we define the similarity between two mushroom tuples to equal the number attributes with the same value, thus:
$$\mathrm{sim}(t_1, t_2) = \sum_{i=1}^{22} \mathrm{equal}(t_{1,i}, t_{2,i}).$$
Note that the target attribute is left out of this sum, since it is assumed to be unknown.

- Your voting function should use similarity-weighted voting with $\beta = 1$.

Part II: Analysis

1. Use your DTree and kNN implementations to classify the testing dataset of the mushroom and text data. As training set you are only allowed to use examples in the training and holdout data files. For kNN, experiment with different k-values to obtain the best accuracy. You don't have to describe these experiments just state which k-values you used to obtain your results.

2. Print out the two DTrees used to classify the mushroom and text data. What attribute is tested in the root node of the mushroom DTree ? What word is tested in the root node of the text DTree ? Why is it interesting to know the root attribute ?

3. Fill out two tables showing the CPU time and accuracy (and the k-value used by kNN). CPU time is given in seconds and the accuracy is given as the percentage of correctly classified examples in the testing set. The tables should have the form:

Accuracy	Mushr.	Text
DTree		
kNN	$k =$	$k =$

CPU time	Mushr.	Text
DTree		
kNN	$k =$	$k =$

4. Which classifier is better for the two data sets in terms of 1) accuracy 2) CPU time. What do you think causes the differences between the classifiers, if any.

5. Explain any difference between the mushroom and text dataset in the overall classification accuracy.

6. Now, classify the unlabeled handin datasets. Save your results in two files: `reuters.handin` and `mushroom.handin`. These files should contain a single string of zeroes and ones terminated by a newline. *One* denotes a positive example, while *zero* denotes a negative example. These files will be automatically graded, so it's important that you use the right format. Thus, `reuters.handin` must contain a string with exactly 50 digits, and `mushroom.handin` must contain a string with exactly 1000 digits.

Extra Credit

1. Repeat 1. and 3. above but this time you are only allowed to use the error data files as training examples. In these sets 10% of the labels have been changed.

2. Does your result support the claim that both classifiers are robust to noise in the training set ?

3. Why ? / Why not ?.

4. Explain any difference between the mushroom and text dataset in the relative degeneration of the accuracy.

Submission Instructions

After completing the assignment, place one copy of your written answers, source code and classification result files in:

`/afs/andrew/scs/cs/15-381/handin/`*userid1_userid2*

Submission directories for each group are available now, and will be closed at 11:59pm on Tuesday November 21. Please contact either Laurie (hyaku@cs) or Rune (runej@cs) if you have any questions.