CS4710 - Artificial Intelligence: Homework 4

Fall 2016

Due: Wednesday, November 30, 2016

1 Introduction

In this last homework assignment we will apply machine learning and try to do prediction to unseen data. In particular we will try to predict the *cuisine* of various recipes that are given to us.

2 Training Set

Our training set consists of 1794 recipes from 20 different cuisines. As an example, one recipe from our training set is shown below.

```
{
    "id" : 2,
    "cuisine" : "greek",
    "ingredients" : [
        "minced garlic",
        "dried oregano",
        "red wine vinegar",
        "olive oil",
        "boneless chop pork",
        "lemon juice"
    ]
}
```

Note that the above recipe is a *greek* recipe with six *ingredients*. Thus, the ingredients here play the role of *features* and in every recipe each ingredient is either present or not. There are 2398 different features (ingredients) that appear in the recipes of our training set and for convenience we are given a file that lists all of them. Table 1 shows the distribution of the recipes in our training set.

2.1 Files and Format

We are given the files below.

training.json and training.csv which contain our training data. Each line in the json file has a different recipe in json format. Each line in the csv file has a different recipe in csv format. You can use either of the two files so that you can provide your algorithm with input. Both files contain the same piece of information, so it is a matter of which file you prefer to read with your program.

ingredients.json and ingredients.txt which contain a list of the ingredients that appear in the recipes of the training set. The file ingredients.json is one big json with all the ingredients. The file ingredients.txt contains the same information; its format is different. Thus, the file ingredients.txt has the name of one ingredient per line. We are free to use either file.

Table 1: Breakdown of the 1794 recipes that form our training set.

cuisine	recipes
brazilian	20
british	35
cajun_creole	73
chinese	124
filipino	38
french	122
greek	57
indian	124
irish	34
italian	324

cuisine	recipes
jamaican	30
japanese	66
korean	47
mexican	275
moroccan	38
russian	27
southern_us	192
spanish	49
thai	74
vietnamese	45

2.1.1 Format of training.json

Each line has a different recipe in json format. Thus, the file has the format shown below.

```
{"id":0,"cuisine":"greek","ingredients":["romaine lettuce","black olives","grape tomatoes","garlic","pepper","purple onion","seasoning","garbanzo beans","feta cheese crumbles"]}
{"id":1,"cuisine":"greek","ingredients":["ground pork","finely chopped fresh parsley","onions","salt","vinegar","caul fat"]}
{"id":2,"cuisine":"greek","ingredients":["minced garlic","dried oregano","red wine vinegar","olive oil","boneless chop pork","lemon juice"]}
...
...
...
```

2.1.2 Format of training.csv

Each line has a different recipe in csv format (comma separated values). The first column is the id of the recipe (integer), The second column indicates the cuisine of the recipe. All subsequent columns in the line contain the ingredients of the particular recipe. Thus, the file has the format shown below.

```
0,"greek","romaine lettuce","black olives","grape tomatoes","garlic","pepper","purple onion","seasoning","garbanzo beans","feta cheese crumbles"
1,"greek","ground pork","finely chopped fresh parsley","onions","salt","vinegar","caul fat"
2,"greek","minced garlic","dried oregano","red wine vinegar","olive oil","boneless chop pork","lemon juice"
...
...
...
```

2.1.3 Format of ingredients.json

We have a single json object. The file has the format shown below.

```
{"ingredients":["boneless chop pork","dried oregano","lemon juice","minced garlic","olive oil","red wine vinegar",...]}
```

2.1.4 Format of ingredients.txt

In case the above file with the one big json is not convenient for us, the ingredients are also given to us in a text file, where in every line of the text file we have a single ingredient surrounded by double quotes. The file has the format shown below.

```
"boneless chop pork"
"dried oregano"
"lemon juice"
"minced garlic"
"olive oil"
"red wine vinegar"
...
...
...
```

3 Implementation

You need to implement one of the machine learning algorithms that we saw, or are about to see, in class (decision trees, regression, naive bayes, neural networks). You may use additional libraries that help you compute or read information; e.g. for algebraic manipulation of matrices, reading json files, or something along these lines. However, you need to impement by yourselves the core of the algorithm that you will be using.

Further, you will need to apply 6-fold cross validation (see Appendix A) and report on the generalization error that you observed each time as well as the average error using these six values that you have reported.

4 Turn In

You need to return one zip file with all of your code and the write-up. In case your program requires a makefile for the compilation, the makefile should be included too. You are free to choose any language that you prefer.

4.1 Write-Up

Return a document of at most 1-2 pages, briefly describing the method that you used for training as well as a brief summary of the 6-fold cross-validation results that you obtained from the training set that was given to you. Essentially one or two small paragraphs should be enough to describe your method. Further, one small paragraph with a simple table should suffice for the generalization error that you obtained in each one of the 6 runs of the 6-fold cross validation method that you applied. Do not forget to clearly report the minimum and the maximum generalization error that you obtained after all 6 runs, e.g., write down these two entries with boldface letters so that it is clear. Also, do not forget to clearly report the average error that you observed in these 6 runs.

In order to be safe in case one wants to elaborate on the technique that she/he used, it is ok if you want to turn in more text. Note though, that it is not required. Only a brief summary of what you have done is needed and a brief summary of the generalization error that you obtained using a 6-fold cross validation method. Finally, also comment on the time required for training[1] and the specs of the machine where training was performed (i.e. cpu and memory).

A Cross Validation

Cross validation is a method that allows us to estimate the performance of the hypotheses that we obtain on *unseen data*. The typical approach when we are given a data set for training, is to keep a subset of the training set on the side and not use it as part of the input of our machine learning algorithm. Then, when we derive a hypothesis using this restricted training set, we now examin how well our hypothesis predicts unseen data using the set that we kept initially on the side.

The holdout method. The holdout method is the simplest kind of cross validation. The data set is separated into two sets, called the *training set* and the *test set*. Our machine learning algorithm comes up with a hypothesis *using the training set only*. Then, the derived hypothesis is asked to predict the output values for the data in the test set (note that these are unseen examples). The mistakes our hypothesis makes are accumulated and in the end we report the percent of error (incorrect predictions for the classification) our hypothesis makes in the test set. In other words we use the formula,

$$\frac{\text{number of mistakes}}{\text{number of examples in the test set}}.$$

[1] A rough estimate within the nearest second or minute is enough. We just need to know what we should expect when we will test your code

The main drawback of this approach is that the reported error may fluctuate a lot (in other words, it has high *variance*) depending on how one divides the original training set into the two smaller sets (the one actually used for training and the other one used for testing).

k-Fold Cross Validation. k-fold cross validation is the natural improvement over the holdout method and is the method that you need to use in this assignment with $k = 6$.

The original data set is divided into k disjoint sets and the holdout method is repeated k times. The idea is now that each one of the k times, one of the sets is going to be used for testing (predicting the error) and the rest $k - 1$ sets will be used for training. In other words, every data point gets to be in a test set exactly once, and gets to be in a training set $k - 1$ times. The advantage of this method is that it matters less how the data gets divided; i.e., the variance of our estimated error on unseen data (also called *generalization error*) decreases as k increases. The drawback of the method is that it takes longer to run so that we can come up with a good estimate on the generalization error of the method.

Typically, we do k-fold cross validation and report an estimate on the generalization error of the method that we have used. Once this part is over, then we use all the original data for training (i.e., no test set) so that we can come up with a hypothesis to be used on *new* unseen data.

Leave-One-Out Cross Validation. Leave-one-out cross validation is k-fold cross validation when $k = N$, where N is the number of examples in the given data set. In other words, we come up with a hypothesis N separate times and we use each hypothesis for predicting the label of just one example. Again, we report the average error among all those runs.