Jordan Hand, Josh Malters, and Kevin Fong
CSE 415 Spring 2016

See README.txt for more information about this project such as data set info and file structure.

Title

Machine Learning Classifiers

Team Members

Jordan Hand, Joshua Malters, Kevin Fong

What is the program supposed to do?

This program allows users to explore 3 different data sets (see below for more usage info) and run multiple machine learning techniques on each data set. The specific techniques covered will be discussed in the following section. The primary focus is on classification algorithms in machine learning.

Our program is also designed to allow users to compare algorithms/techniques. The program is interactive so users can select options of what topics they would like to see in action. The program exits after the choose topic runs so the user should just run the program again to look into different techniques and topics.

Techniques Used

Note that we used these concepts in the context of classification of text data. These techniques work slightly differently when used for regression.

Algorithms

Naive Bayes Classifier:

> Naive Bayes classifiers rely on bayesian statistics and, in particular Bayes Rule to determine class labels for a given feature set. To train the classifier, a training set is used to build counts of certain features for each class label. When a data point is classified, probabilities are calculated for P(class|features) for each class. The record is classified based on which class label has the highest probability.

In our experience, Naive Bayes is incredibly fast and fairly accurate for classifying text data.

K Nearest Neighbors:

K Nearest Neighbors classifiers work by checking a record against each training record and finding the k (integer value) records in the training data which most closely match the test record. For classifying text data, we chose to compare records by how many words they have in common.

K Nearest Neighbors is quite a bit slower than naive bayes. This is because comparing long sets of words to find their intersection is a somewhat computationally intensive task.

Bagging:

Bagging, or Bootstrap Aggregating, is a meta algorithm that uses bootstrap sampling (covered later in this section) to create multiple training sets. These training sets are used to train multiple instances of different classifiers. Then when a record is classified, it is classified by each of the classifiers and then they each cast a vote for what class label they think the test record belongs to. The class label with the most votes is selected as the actual class label.

In our experience bagging is a bit more accurate for the most part. For example, in general, using the same training and test sets, bagging using 9 classifiers (both knn and naive bayes) is about 1-2% more accurate than naive bayes and 3-5% more accurate than k nearest neighbors. It takes a bit more time to run than the other 2 as it aggregates multiple instances of knn and naive bayes.

Sampling Techniques

Holdout:

The holdout method for sampling data is super simple basically some fraction of the data is held for testing a classifier and the rest is used for training. In our implementation of the holdout method we use 9/10ths for training and 1/10th for testing.

Bootstrap:

The bootstrap method is slightly more complicated but still pretty easy to understand. For a data set D of length n we will pick our training set by randomly selecting records from D with replacement to make a data set of size n' (usually

n’ <= n). This training set will contain some duplicates and on average the proportion of unique records from D found in this training set is 0.632 (Look up the .632 rule to learn more about the statistics behind this estimate). Any records from D that are not sampled to create the training set are then used for the test set.

Sample Session Transcript

Sample #1 (emails data model)

```
jhand:project(master) $ ./classify.py emails

Let's explore some machine learning algorithms.
Which topic would you like to cover?
Options are ['algorithms', 'bagging', 'comparison']
>> algorithms
Welcome to a data model of emails.

Please choose a classifier.
Options are: ['knn', 'naive_bayes']
>> naive

naive is not an available option.

Options are: ['knn', 'naive_bayes']
>> naive_bayes

Please choose a testing method.
Options are: ['bootstrap', 'holdout']
>> bootstrap

Running Test...

Elapsed Time: 0.981

Accuracy of naive_bayes on data model emails: 93.03%
```

Sample #2 (fruit data model)

```
jhand:project(master) $ ./classify.py fruit
Let's explore some machine learning algorithms.
Which topic would you like to cover?
Options are ['algorithms', 'bagging', 'comparison']
>> bagging
Running Test...

Classifying 9 data points.

Chosen label: banana
Votes: [('banana', 9)]
Actual Label: banana

Chosen label: lemon
Votes: [('lemon', 9)]
Actual Label: lemon

Chosen label: banana
Votes: [('banana', 9)]
Actual Label: banana

Chosen label: banana
Votes: [('lemon', 2), ('other', 3), ('banana', 4)]
Actual Label: banana

Chosen label: lemon
Votes: [('lemon', 9)]
Actual Label: lemon

Chosen label: lemon
Votes: [('lemon', 7), ('other', 2)]
Actual Label: other

Chosen label: banana
Votes: [('lemon', 2), ('other', 3), ('banana', 4)]
Actual Label: banana

Chosen label: banana
Votes: [('banana', 9)]
Actual Label: banana
```

```
Chosen label: lemon
Votes: [('lemon', 7), ('other', 2)]
Actual Label: other

Elapsed Time: 0.009

Accuracy of bagging on data model fruit: 77.78%
```

Sample #3 (emails data model)

```
jhand:project(master) $ ./classify.py emails
Let's explore some machine learning algorithms.
Which topic would you like to cover?
Options are ['algorithms', 'bagging', 'comparison']
>> comparison
Choose two algorithms to compare.
Options are: ['bagging', 'knn', 'naive_bayes']

First algorithm: naive_bayes
Second algorithm: bagging


The data is shuffled and the the holdout method is being used
to create training/test sets.
The same training and test sets are used for each algorithm.

Running Tests, this may time a minute or two...

Accuracy of naive_bayes on data model emails: 90.75%
Elapsed Time: 0.58

Accuracy of bagging on data model emails: 95.18%
Elapsed Time: 36.39
```

Demo Instructions

To run this program you should use the classify.py file. You will not need to run any other python files. We have only tested the program with python 3.5 so you should use that version of python to run it. It also requires numpy to be installed on the system.

You can run the program with any of our predefined data models. The options are emails, fruit, and chess. To run the program you can use the following command with a data model as the first and only argument:

python3 classify.py emails

Once the program starts, you can follow the instructions to explore different concepts and algorithms. Note that different algorithms may take a minute or two to run depending on the data set.

Interesting Code Excerpt

```
135     m = model
136     topic = intro()
137     classifier = None
138     if topic != "comparison":
139         if topic == "algorithms":
140             classifier = get_classifier(m)
141             test = get_test_method(m)
142             training_func = m.trainers[classifier]
143             test_func = m.tests[test]
144         elif topic == "bagging":
145             classifier = topic
146             training_func = m.bagging_trainer
147             test_func = m.bagging_test
148         acc = run_test(m, training_func, test_func) * 100
149
150         triple = (classifier, sys.argv[1], acc)
151         print("Accuracy of %s on data model %s: %.2f%%" % triple)
```

This code excerpt is interesting because of the way that it integrates a lot of the functionality from other files to actually run the code. I know there are a lot of method calls that make this code hard to understand but let me give you a broad overview. ‘m’ is the data model being used (emails/chess/fruit). Each model has a dictionary of functions that when called return classifiers (knn, naive bayes). There is a similar dictionary with test functions (bootstrap method, holdout, etc.) that returns the proportion of records that were correctly classified. This allows us to generalize and use the same code regardless of what algorithm, data model, or data sampling model is being used.

What we learned

Jordan

I learned tons about how machine learning classifiers work and how to actually implement them. It is one thing to learn about them in class but actually implementing them forces you to really understand how they work. I now have really good knowledge

of how k nearest neighbors, naive bayes, and bagging all work. Another thing I gained was experience is making an integrated multi file application in python which is a very valuable skill to have. Finally I learned how to use numpy which is great for doing tasks with a lot of data in python.

Joshua

I learned that even though an algorithm looks foreboding, with enough perseverance and outside education (blogs, youtube videos, open source code) it can be done. I primarily worked on knn, and this was the first algorithm I have implemented that was not discussed in a lecture. Although it is a simpler machine learning algorithm, to the inexperienced it can seem daunting. However when I just got to coding, I found it came along quite easily! Additionally this assignment made me think about “learning”. Initially I thought that updating probabilities in a Naive Bayes shouldn’t really be considered learning, its just updating data. But then I started think about how a human learns, and it opened up a whole philosophical realm. In the end I decided that human learning is kind of just “updating/inserting new data”.

Kevin

Before we started the project I didn’t know much about machine learning classification other than the bit we learned in class about decision trees. From having to actually implement the naive bayes classifier, I got a much better understanding about naive bayes. I also got to learn about the knn algorithm and not only what bagging is, but also how it works and the advantages from using it. Finally, I learned about the many different kinds of things machine learning classification can be used for when looking for potential datasets to use.

What we could add with more time

- More algorithms such as decision trees or decision forests
- Boosting
- Handling other data types besides text

Citations

Introduction to Data Mining by Tan, Steinbeach and Kumar:
http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

This reading was mostly useful for understanding the steps taken for building a classification model such as taking the training set > learning/training with the algorithm > applying it to deduce an appropriate class.

Machine Learning by Mitchell
https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

This reading helped supplement the material from lecture which was useful for understanding how naive bayes worked. The lecture itself was helpful with its examples (which was used as a test data set) and some specifics such as the need for phantom examples for zero values and smoothing.

How kNN algorithm works
https://www.youtube.com/watch?v=UqYde-LULfs

This video greatly helped introduce the general idea of the kNN algorithm. The lecturer (Youtuber?) provides many metaphors and visual elements to explain and showcase the ideology behind the algorithm and hints and some useful steps when implementing.