Winnow2 and Naïve Bayes Algorithms

Tarney, Brandon
Johns Hopkins University Engineering Professionals CS 605.649

Table of Contents

Abstract

The purpose of this assignment was to implement Winnow2 and Naïve Bayes algorithms and use them on real-world data. Additionally, the algorithms are compared and each is tweaked for performance. Pre-processing data for analysis is time intensive and typically requires a combination of binning input features and then creating binary vectors from those bins and additionally creating binary representations of the classes (required for Winnow2). As expected, both algorithms perform very well on data sets with binary classifications and ample learning data. Furthermore, each algorithm is linear by nature and therefore fits best with data which is linear by nature. All code and example data can be found here: https://github.com/1amBulletproof/machine_learning_p1_winnow_and_bayes.

Keywords: Winnow2, Naïve Bayes, Machine Learning

Winnow2 and Naïve Bayes Algorithms

Experiment Summary

Problem Statement

An introduction to machine learning by implementing Winnow2 and Naïve Bayes algorithms and testing them against several real-world data sets. Preprocessing the

Hypothesis

Significant pre-processing of real-world data will be required for these simple algorithms, especially Winnow2. Both Winnow2 and Naïve Bayes will be able to generalize solutions to *linear* problems but will struggle to classify non-linear problem spaces. The accuracy and consistency of both algorithms will increase as more training data is provided. Naïve Bayes will perform better than Winnow2 in general and on non-binary classification problems.

Background

Winnow2

Winnow2 is a machine learning algorithm more theoretical than practical. It works by calculating the summation of the products of every feature in an input vector by a given weight and then comparing the result with a threshold value. If the summation is greater than the threshold, the classification is 1, otherwise it is 0. If the result is not the expected result, promotion is applied when the expected result is 1 but the calculated result was 0, otherwise demotion is applied. Promotion will increase the weight of an input feature by alpha*weight if the input feature was 1. Similarly, demotion will decrease the weight of an input feature by dividing the weight by alpha if the input feature was 1. Winnow2 can only make binary classifications, but multiple models can be used to classify non-binary classifications. (Littlestone, 1988)

Naïve Bayes

Naïve Bayes is a machine learning algorithm which seeks to calculate the probability of every input value {0,1} for every possible classification. This can be accomplished simply by counting the occurrences of every input variable for every classification. The algorithm classifies a given input vector as the class which has the highest probability. This algorithm is dubbed naïve because it assumes conditional independence, i.e. all input features are independent of each other. (Alpaydin, 2010)

Experiment Details

Experimental Approach

This experiment covers the entire process of creating and applying a machine learning algorithm, not simply implementing the algorithm in isolation. Overall, this experiment can be broken down into several key steps:

Real World Data

Five real world data sources were provided for this experiment all of which can be found in the “data” folder. These real-world data sets covered the areas of classifying breast-cancer tumors, glass, an iris (flower), a republican or democrat, and a soybean. All the data sets required some form of pre-processing before Winnow2 or Naïve Bayes could learn them. For this experiment, breast-cancer, voting, and iris data sets were chosen.

Pre-processing Data

In this experiment data must be preprocessed for typically three reasons. First of all, there is a need for a standard data format the program expects. One of the first parts of pre-processing in this experiment was to ensure the following input data format: [X0, X1, …Xn, Fn]. This typically meant some useless data (like "id") was stripped-away and the Fn or classification was moved to be the last column. Secondly, the data often has real-world, non-discrete values for feature inputs. In order to create only binary feature inputs, any non-binary inputs are binned or sorted into equal-size groups and then each bin is given an input vector. Finally, Winnow2 can only learn on a binary classification set, therefore, when three or more classifications exist, multiple data sets must be created where the classification is class A or *not* class A instead of class A or class B or class C. The pre-processing was performed by a combination of Unix tools and the following python programs: "src/preprocess_*.py" which uses other utilities such as "src/file_manager.py" and "src/data_manipulator.py" for reading and manipulating data. All of the pre-processed data can be found in "data/pre_processed_data". Note that the value given for each classification is implied by the name of the pre-processed data file e.g. "vote_data_r0_d1.csv" implies republicans are a "0" and democrats are a "1".

Splitting Data

Data must be split into a learning set and a test set. Additionally, the distribution of vectors into either learning or test sets is randomized. This functionality is available through "src/data_manipulator.py" and is used by the pre-processing modules mentioned previously.

Developing the Winnow2 Algorithm

The code which implements Winnow2 is "src/winnow_2.py". Simply running it with python3 ("python3 winnow_2.py") will input dummy data into the Winnow2 algorithm and effectively unit test all the functionality. Winnow2 design was more difficult than Naïve Bayes since the capability to make non-binary classifications based on multiple Winnow2 binary models was incorporated. The output of the learned Winnow2 algorithm is a vector of weights which can be seen in figure 2.

Developing the Naïve Bayes algorithm

The code which implements Naïve Bayes algorithm is "src/naïve_bayes.py". Simply running it with python3 ("python3 naïve_bayes.py") will input dummy data into the internal algorithm and effectively unit test all the functionality. The output of the learned Naïve Bayes algorithm is a percentage table for each input feature given a class, shown in figure 3.

Running and Validation

To actually learn, test, and output the results of this experiment, "src/test_model_framework.py" and "src/test_multiple_model_framework.py" were developed. The only difference between the two is the latter is designed for processing multiple models of the same data where the classification is non-binary (i.e. required for Winnow2). The output of these programs can be seen in figures 1-4. An example run command would look like this: "python3 test_model_framework.py data/pre_processed_data/breast_cancer_wisconsin_benign0_malignant1.csv". The output of multiple tests on each data set can be found in "results/" folder.

Modifications and Tweaks

At first Winnow2 performed much worse than the results you see below. Changing the threshold from 0.5 to half the number of inputs instead improved the performance of the model considerably. Altering alpha seemed to changed how quickly the model learned: a larger alpha resulted in better results with very little training data but worse results with much training data.

Naïve Bayes required smoothing to work effectively so that no 0 probabilities cancelled out other feature input probabilities. Finally, by varying the amount of training data vs. test data it was observed that more training data produces more consistent and generally better results, however, given an inverse relationship of training data to test data (i.e. when increasing training data you decrease test data due to a finite sample size) there is a minimum point reached where the performance of the model given a small test sample is less predictable.

Results, Analysis and Conclusions

Winnow2 and Naïve Bayes performed much better than I expected overall. This was at least due in part to the data sets I chose, 2 of which were binary classifications. The output of my experiences can be seen in figures 1-4. Figure 1 simply shows the input from a given run as well as the learning vs. testing sample sizes. The larger the learning sample size, the better the model seemed to generalize, however, at some point the amount of test data becomes so small that there is more unpredictability and it skews results. The average performance per data set for each model is shown in table 1. To summarize, both algorithms performed extremely well on all three data sets but performed noticeably worse on the iris data set, which was *not* a binary classification problem.

Summary

In conclusion, Naïve Bayes and Winnow2 algorithm are fairly trivial to develop but will only work well on a limited subset of problem spaces. Binary classifications are typically better learned than multiple classifications. Furthermore, if the classifications do not exhibit a linear relationship, then neither algorithm will perform well, as explained by representation bias. Naïve Bayes performed considerably better with smoothing and Winnow2 performed considerably better with a threshold value dependent on the number of input features. Both algorithms performed better than expected for binary classification data sets at an average rate of 85-95% accuracy but performed worse when more classifications were present. All code can be found here: https://github.com/1amBulletproof/machine_learning_p1_winnow_and_bayes.

References

Alpaydin, E. (2010). Introduction to machine learning (3rd ed.). Cambridge, MA: The MIT Press. ISBN-10: 0262028182; ISBN-13: 978-0262028189.

Fisher, R.A. (1988). *UCI Machine Learning Repository: Iris Data Set*. [online] Archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/Iris [Accessed 10 Sep. 2018].

Littlestone, N. (1988). *Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm*. Machine Learning 285-318(2).

Schlimmer, J. (1987). *UCI Machine Learning Repository: Congressional Voting Records Data Set*. [online] Archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records [Accessed 10 Sep 2018].

Wolberg, W. H. and Mangasarian, O. L. (1990). *Cancer diagnosis via linear programming*. Siam News, Volume 23, Number 5, September 1990, pp 1 & 18.

Wolber, W.H. and Mangasarian, O.L. (1992). *UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set.* [online] Available at: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic) [Accessed 10 Sep. 2018].

Tables

Table 1: Performance

		Algorithms		
		Winnow2 (% avg success)	Naïve Bayes (% avg success)	Combined (% avg success)
Data Sets	Vote	92	90	91
	Breast Cancer	95	96	95.5
	Iris	85	85	85
	Total Avg	90.66666667	90.33333333	90.5

Figures

Figure 1: Inputs

```
INPUTS
overall_model_file_path: ../data/pre_processed_data/iris_data_setosa0_versicolor1_virginica2.csv
class0_filepath: ../data/pre_processed_data/iris_data_setosa1_else0.csv
class1_filepath: ../data/pre_processed_data/iris_data_versicolor1_else0.csv
class2_filepath: ../data/pre_processed_data/iris_data_virginica1_else0.csv
fraction of data to train:  0.66
number of classes:  3
number of input vectors:  150

learning data size:  91
test data size:  59
```

Figure 2: Winnow Weights Output

```
Winnow2 learned weights for class 0:
[4.0, 1.0, 1.0, 1.0, 1.0, 1.0, 4.0, 1.0, 4.0, 1.0, 1.0, 1.0, 4.0, 1.0, 1.0, 1.0]

Winnow2 learned weights for class 1:
[2.0, 4.0, 0.5, 1.0, 0.5, 2.0, 4.0, 1.0, 1.0, 4.0, 2.0, 0.5, 1.0, 4.0, 2.0, 0.5]

Winnow2 learned weights for class 2:
[1.0, 0.5, 2.0, 2.0, 4.0, 1.0, 0.5, 1.0, 0.5, 1.0, 1.0, 4.0, 0.5, 1.0, 0.5, 8.0]
```

Figure 3: Naïve Bayes Percentage Table

```
Naive Bayes learned percentages as input[ class[ (prob0, prob1) ] ]
[[(0.16670000000000001, 0.8000333333333334), (0.8823823529411765, 0.08826470588235294), (0.9667, 3.3333333333333335e-05)], [(0.8000333333333334, 0.166
70000000000001), (0.32355882352941173, 0.6470882352941176), (0.8000333333333334, 0.16670000000000001)], [(0.9667, 3.3333333333333335e-05), (0.73532352
94117647, 0.23532352941176468), (0.4000333333333333, 0.5667000000000001)], [(0.9667, 3.3333333333333335e-05), (0.9706176470588235, 2.9411764705882354e
-05), (0.7333666666666667, 0.2333666666666667)], [(0.9333666666666667, 0.03336666666666666), (0.5588529411764707, 0.4117941176470588), (0.83336666666
66667, 0.1333666666666667)], [(0.6333666666666667, 0.33336666666666664), (0.44120588235294117, 0.5294411764705883), (0.3000333333333333, 0.66670000000
00001)], [(0.5000333333333333, 0.4667), (0.9412058823529411, 0.029441176470588235), (0.8000333333333334, 0.16670000000000001)], [(0.8333666666666667, 0
.1333666666666667), (0.9706176470588235, 2.9411764705882354e-05), (0.9667, 3.3333333333333335e-05)], [(3.3333333333333335e-05, 0.9667), (0.97061764705
88235, 2.9411764705882354e-05), (0.9667, 3.3333333333333335e-05)], [(0.9667, 3.3333333333333335e-05), (0.7059117647058823, 0.26473529411764707), (0.96
67, 3.3333333333333335e-05)], [(0.9667, 3.3333333333333335e-05), (0.26473529411764707, 0.7059117647058823), (0.5333666666666667, 0.4333666666666666)],
 [(0.9667, 3.3333333333333335e-05), (0.9706176470588235, 2.9411764705882354e-05), (0.4333666666666666, 0.5333666666666667)], [(3.3333333333333335e-05,
 0.9667), (0.9706176470588235, 2.9411764705882354e-05), (0.9667, 3.3333333333333335e-05)], [(0.9667, 3.3333333333333335e-05), (0.6176764705882354, 0.3
529705882352941), (0.9667, 3.3333333333333335e-05)], [(0.9667, 3.3333333333333335e-05), (0.3529705882352941, 0.6176764705882354), (0.5333666666666667,
 0.4333666666666666)], [(0.9667, 3.3333333333333335e-05), (0.9706176470588235, 2.9411764705882354e-05), (0.4333666666666666, 0.5333666666666667)]]
```

Figure 4: Classification results for given test data

```
Testing Winnow2 model
classification attempts( 59 ), #fails( 14 ), #success( 45 )

Testing Naive Bayes model
#classification attempts( 59 ), #fails( 12 ), #success( 47 )
```