Adam Jaworski

Machine Learning HW 2 Experiment Report

In order to compare the performance of the Winnow and Perceptron learning algorithms, I used a 10-fold cross validation method. Each algorithm was exposed to two data sets: one generated artificially, and one taken from real-world congressional voting records. The algorithms’ performance on each data set was compared independently (so I might have found that Winnow is better for one data set, while Perceptron is better for the other).

Each data set was split up into 10 equal partitions. However, for the voting records data set (where the algorithms are meant to predict whether a voter is a democrat or a republican), I noticed that all of the democrats were listed first, followed by all of the republicans. To truly make the sets equal, it wouldn’t be enough to simply have them all contain the same number of instances. Rather, they should also all contain a similar ratio of positive and negative instances.

For this reason, I had the first partition filled with n/10 randomly-selected instances (where n is the total number of instances). The second partition was then filled with n/10 instances randomly-selected from the instances that had not yet been assigned to a partition, and this process was repeated for all 10 partitions.

With the data sets separated into 10 equal partitions, I could then evaluate the algorithms. For each data set, for each algorithm, the evaluation consisted of 10 trials. For trial k, the instances in partition k were used as the test set, while the instances in all other partitions were used as the training set. The size of the test set was therefore n/10 (the size of one partition), which worked out to be 100 for the artificial data set and 23 for the voting data set. For each trial, an algorithm’s accuracy was determined by its number of correct predictions during the test set, divided by the size of the test set. These results are summarized in the tables below: (the difference is Perceptron – Winnow)

VotesData.txt

|  |  |  |  |
| --- | --- | --- | --- |
| Trial | Winnow Accuracy | Perceptron Accuracy | Difference |
| 1 | .26 | .83 | .57 |
| 2 | .57 | .78 | .21 |
| 3 | .39 | .96 | .57 |
| 4 | .52 | .83 | .31 |
| 5 | .57 | .83 | .26 |
| 6 | .48 | .96 | .48 |
| 7 | .22 | .83 | .61 |
| 8 | .91 | .74 | -.17 |
| 9 | .48 | .78 | .30 |
| 10 | .43 | .87 | .44 |

ArtificialData.txt

|  |  |  |  |
| --- | --- | --- | --- |
| Trial | Winnow Accuracy | Perceptron Accuracy | Difference |
| 1 | 1.00 | 1.00 | 0 |
| 2 | 1.00 | 1.00 | 0 |
| 3 | 1.00 | 1.00 | 0 |
| 4 | 1.00 | 1.00 | 0 |
| 5 | 1.00 | 1.00 | 0 |
| 6 | 1.00 | .97 | -.03 |
| 7 | 1.00 | 1.00 | 0 |
| 8 | 1.00 | 1.00 | 0 |
| 9 | 1.00 | 1.00 | 0 |
| 10 | 1.00 | 1.00 | 0 |

With this information, I could compare the algorithms’ performance on the data sets using a paired t-test. The end result of a paired t-test will be a range for the real difference between the two algorithms’ accuracy (to some confidence level). But first I needed to compute the sample mean and sample variance for the difference (for each data set).

VotesData.txt:

Sample mean = m = (.57 + .21 + .57 + .31 + .26 + .48 + .61 - .17 + .30 + .44)/10 = .358

Sample variance = s2 = ((.57 - .358)2 + (.21 - .358)2 + (.57 - .358)2 + (.31 - .358)2 + (.26 - .358)2 + (.48 - .358)2 + (.61 - .358)2 + (-.17 - .358)2 + (.30 - .358)2 + (.44 - .358)2)/9 = .05455

s = sqrt(s2) = .2336

ArtificialData.txt

Sample mean = m = (0 + 0 + 0 + 0 + 0 - .03 + 0 + 0 + 0 + 0)/10 = -.003

Sample variance = s2 = ((0 + .003)2 + (0 + .003)2 + (0 + .003)2 + (0 + .003)2 + (0 + .003)2 + (-.03 + .003)2 + (0 + .003)2 + (0 + .003)2 + (0 + .003)2 + (0 + .003)2)/9 = .00009

s = sqrt(s2) = .00949

The last thing I needed was a t-value. For this I consulted http://en.wikipedia.org/wiki/Student’s\_t-distribution. With 10 trials and a 95% confidence level, the appropriate t-value is 2.262. So t = 2.262, and I can say with 95% confidence that the true difference between the algorithms will be within the range I found. The formula for the range is as follows:

[m – (t\*(s/sqrt(k))), m + (t\*(s/sqrt(k)))], where k = 10 (the number of trials)

Plugging in the other numbers, we have:

VotesData.txt

[.358 – (2.262\*(.2336/sqrt(10))), .358 + (2.262\*(.2336/sqrt(10)))]

or

[.191, .525]

ArtificialData.txt

[-.003 – (2.262\*(.00949/sqrt(10))), -.003 + (2.262\*(.00949/sqrt(10)))]

or

[-.0098, .0038]

What these results tell me is that, for the VotesData data set, there is a 95% chance that the true difference between the algorithms’ accuracy is between .191 and .525. Since the difference was defined as Perceptron’s accuracy – Winnow’s Accuracy, a positive value for the difference means that Perceptron is more accurate. All values within the range are positive (the range does not contain 0), and so I can say with 95% confidence that the Perceptron algorithm is superior to Winnow for the voting data set.

For the ArtificialData data set, there is a 95% chance that the true difference between the algorithms’ accuracy is between -.0098 and .0038. The difference is still defined in the same way, but now the range contains both positive and negative values. Therefore I cannot say that either algorithm is superior for the artificial data set.

These results are not really surprising to me. The Winnow algorithm only works when the target function is a disjunction. With the real-world data set of congressional voting, I wouldn’t have expected the examples to follow a disjunctive function. I imagine that for every vote (feature), at least one democrat voted no (false), at least one democrat voted yes (true), and at least one republican voted each way as well. In the long run this would ultimately lead to the Winnow algorithm thinking that none of the features are important, and so it will always predict that a voter is a republican (where republican corresponds to a false prediction; the voter is NOT a democrat). About half of the instances were republican voters, and the Winnow algorithm looks to have around a 50% accuracy on average for the voting data set, which is what I would expect.

Meanwhile, the Perceptron algorithm is not limited to finding disjunctive functions, and so it performs significantly better on the voting data set, as confirmed by the paired t-test. For the artificial data set, both algorithms performed extremely well. Considering the success of Winnow, the target function must have been some sort of disjunction of the 16 features. After 900 training examples and only 16 features, it probably had the target function figured out exactly, and that’s why it didn’t make a single mistake. If anything surprised me it was that the Perceptron algorithm performed almost as well, because it is a more general algorithm and not specifically designed for disjunctions. According to the paired t-test, the difference between Winnow and Perceptron wasn’t even statistically significant for the artificial data set. But on the other hand, 900 training examples is still a lot when there are only 16 features, especially when you’re not dealing with noisy real-world data. So even this result was not enormously surprising to me.

Because the results basically matched my expectations, I don’t feel like there were any issues with the experiment. Also, I don’t think that FIND-S would have worked well for either data set. FIND-S is intended to find conjunctive target functions. Certainly the real-world voting data wouldn’t fit a conjunctive function, in much the same way that it didn’t fit a disjunctive function. Additionally, Winnow’s success on the artificial data set suggests that it had a disjunctive target function, and not a conjunctive one. So FIND-S wouldn’t work well there, either.

References

“Student’s t-distribution”. *Wikipedia: The Free Encyclopedia*. Wikimedia Foundation, Inc., 9 February 2014. Web. 10 February 2014.