Assignment 3  
CS 6375: Machine Learning

**1.Report for Perceptron and Neural Networks**

Name: Saraswathi Shanmugamoorthy

Course: CS 6375.003

**Accuracy Tables:**

**Accuracy for Perceptron algorithm:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | Accuracy | | Number of iterations | Learning constant |
|  | Accuracy before filtering stop words | Accuracy after filtering stop words |  |  |
| 1 | 93.0962% | 93.9331% | 500 | .01 |
| 2 | 93.3054% | 93.9331% | 1000 | .01 |
| 3 | 93.3054% | 93.9331% | 1500 | .01 |
| 4 | 93.3054% | 93.9331% | 2000 | .01 |
| 5 | 92.4686% | 92.0502% | 500 | .05 |
| 6 | 92.2594% | 92.2594% | 1000 | .05 |
| 7 | 92.2594% | 92.2594% | 1500 | .05 |
| 8 | 92.2594% | 92.2594% | 2000 | .05 |
| 9 | 91.2134% | 92.2594% | 500 | .006 |
| 10 | 91.8410% | 93.5146% | 1000 | .006 |
| 11 | 91.8410% | 93.5146% | 500 | .008 |
| 12 | 91.8410% | 93.7238% | 1000 | .008 |
| 13 | 92.8870% | 93.7238% | 500 | .009 |
| 14 | 93.9331% | 93.7238% | 1000 | .009 |
| 15 | 91.6318% | 92.0502% | 500 | .019 |
| 16 | 93.3054% | 92.4686% | 1000 | .019 |
| 17 | 92.2594% | 91.2134% | 500 | .02 |
| 18 | 93.7238% | 93.7238% | 1000 | .02 |
| 19 | 92.0502% | 92.2594% | 500 | .03 |
| 20 | 93.0962% | 92.4686% | 1000 | .03 |

**Explanation for the above results:**

The Accuracy generally tends to increase as the learning constant value is reduced. For higher values of learning constant the accuracy is lower. The accuracy also tends to decline again if the learning constant value is too small. As the number of iterations increase there is a slight increase in the accuracy and sometimes there is no increase. The general trend is that the accuracy increases after filtering out or removing stop words.

Stop words are, by definition, words that do not contain information for our classification task. So removing stop words should increase the accuracy. But there is no universal set of stop words that will improve all text classification tasks. Since we are just using an off-the-shelf stop words dictionary, we could be throwing away potentially valuable information.

For example lets consider the word ‘me’, the conditional probability of me is higher than average in ham, which proves that it’s a good word to classify. But since ‘me’ is included in the list of stop words we remove it, this might be one of the reasons for the accuracy to decrease.

**Sample Output for Perceptron algorithm: Number of iterations = 1000; Learning Constant = 0.01:**

|  |
| --- |
| Sachus-MacBook-Pro:Perceptron-Algorithm-for-Text-Classification-master Sachuleonid$ python Perceptron.py train test 1000 .01  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    Learning constant: 0.0100  Number of iteration: 1000    Correct Predictions before filtering out stop words: 446/478  Accuracy before filtering out stop words: 93.3054%    Correct predictions after filtering out stop words: 449/478  Accuracy after filtering out stop words: 93.9331% |

**Accuracy tables:**

|  |  |  |
| --- | --- | --- |
| Accuracy | Unfiltered data set (before removing stop words) | Filtered Data Set (After removing stop words) |
|  | 94.7699% | 94.3515% |

**Accuracy for Naïve Bayes:**

**Accuracy for Logistic Regression:**

|  |  |  |
| --- | --- | --- |
| **λ values** | **Accuracy on the unfiltered data set (before removing the stop words)** | **Accuracy on the filtered Data Set (After removing the stop words)** |
| λ = 0.1 | 92.8870% | 95.6067% |
| λ = 0.2 | 92.8870% | 95.6067% |
| λ = 1.0 | 93.3054% | 95.8159% |
| λ = 2.0 | 93.3054% | 95.6067% |
| λ = 3.0 | 93.5146% | 95.3975% |
| λ = 4.0 | 93.9331% | 95.1883% |
| λ = 5.0 | 93.5146% | 94.9791% |

The hard limit on the numbers of iterations is **50**.

**Comparing Accuracy of Perceptron algorithm with Accuracy of Naïve Bayes and Logistic Regression:**

Comparing the accuracies in all the above tables, we can observe that the accuracy of Perceptron is slightly lower than Naïve Bayes and Logistic Regression. Logistic Regression has the highest accuracy followed by Naïve Bayes and finally Perceptron.

Generally in the case of Logistic Regression we assume that there is some function which accurately fits our dataset. And this function will have parameters. And our job is to find the best possible values for these parameters using the gradient descent method. Naïve Bayes algorithm is even simpler, here we don’t have to optimize functions or whatsoever, we just have to calculate the Bayesian Conditional Probabilities, from the training dataset provided. This can be done

The third method is using Perceptrons. It is a lightweight algorithm. It can be used only on linearly separable binary datasets, that is datasets with only two classes. Perceptron doesn’t guarantee to find the optimal separating hyperplane, whereas SVM gives only the optimum separating hyperplane.

**Neural networks implementation in WEKA (called MultiLayered Perceptron) :**

To convert the datasets in text format to ARFF format, open Weka and click on Simple CLI and type the following command:

Java weka.core.converters.TextDirectoryLoader -dir “path-of-dataset-folder” > “path-of-destination-to-save-converted-ARFF-file”

**Accuracy Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Accuracy | Learning Rate | Momentum | No of Hidden Layers | No of units in each hidden layer | Iterations |
| 91.6047 | 0.1 | 0.2 | 1 | 1 | 500 |
| 97.0244 | 0.09 | 0.2 | 1 | 1 | 500 |
| 96.7056 | 0.09 | 0.1 | 1 | 1 | 500 |
| 96.5994 | 0.09 | 0.2 | 2 | 1,2 | 500 |
| 96.9182 | 0.09 | 0.2 | 1 | 1 | 1000 |
| 92.0298 | 0.09 | 0.2 | 2 | 1,2 | 1000 |
| 81.7216 | 0.09 | 0.1 | 3 | 2,4,2 | 500 |
| 87.1413 | 0.09 | 0.1 | 3 | 2,4,2 | 1000 |
| 73.1137 | 0.09 | 0.1 | 5 | 1,2,4,2,1 | 500 |
| 73.1137 | 0.09 | 0.1 | 5 | 1,2,4,2,1 | 1000 |

The Accuracy is observed to increase as the number of iterations increases. As we iterate more the model gets more and more accurately fitted to the training data. As the momentum reduces the accuracy increases. As the learning rate reduces the accuracy increases.

Using many hidden layers causes the model to overfit, hence it perfectly fits the training dataset, but is not able to generalize and hence performs poorly on the testing dataset.

**Screenshot of run of Weka Run for one of the cases**:

where Learning Rate = 0.09; Momentum = 0.2, No of Hidden Layers =1, No of units in each hidden layer = 1, Number of iterations =500, Here accuracy is 97.0244%

A screenshot of a computer

Description automatically generated

**Comparing the perceptron algorithm with WEKA Multi -Layered Perceptron Algorithm:**

The accuracy of the perceptron algorithm is in the range of 93% to 94%, and WEKA Multi -Layered Perceptron Algorithm accuracy is in the range of 73% to 97%, this wide range is due to the changes in the number of hidden layers used and hidden units used in each layer. The accuracy of the perceptron algorithm which I have developed is almost as accurate as WEKA, where the highest accuracy obtained using the perceptron algorithm which I had developed is 94% and WEKA’s Multi -Layered Perceptron Algorithm has the highest accuracy of 97%.