AI Project

Fulton III Wilcox, Daniel Teytel, Long Tran

May 2023

1 Introduction

In the focus of classifying digits and faces, we implemented Naive Bayes, Perceptron and [3rd algo] methods.

2 Observations

Method: by letting the classifier train randomly parts of the data

$$[10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%, 100\%]$$

with number of n times - we call it n iteration for each part of the data, we record the time and and accuracy rate, standard deviation for each algorithm.

Comparison between digit classifying:

Listing 1: digit_naiveBayes.txt

```
Doing classification
--------------------
data:           digits
classifier:             naiveBayes
using enhanced features?:       False
training set size:      5000
using smoothing parameter k=2.000000 for naivebayes
Extracting features...
Sampling 5 iterations on 10.0%...
numBatch is 500
Mean Accuracy of 10.0%: 0.77
Standard Deviation of 10.0%: 0.0289827534924



Sampling 5 iterations on 20.0%...
numBatch is 1000
Mean Accuracy of 20.0%: 0.776
Standard Deviation of 20.0%: 0.0149666295471
```

```
Sampling 5 iterations on 30.0%...
numBatch is 1500
Mean Accuracy of 30.0%: 0.798
Standard Deviation of 30.0%: 0.0146969384567


Sampling 5 iterations on 40.0%...
numBatch is 2000
Mean Accuracy of 40.0%: 0.79
Standard Deviation of 40.0%: 0.0141421356237


Sampling 5 iterations on 50.0%...
numBatch is 2500
Mean Accuracy of 50.0%: 0.792
Standard Deviation of 50.0%: 0.00979795897113


Sampling 5 iterations on 60.0%...
numBatch is 3000
Mean Accuracy of 60.0%: 0.79
Standard Deviation of 60.0%: 0.00894427191


Sampling 5 iterations on 70.0%...
numBatch is 3500
Mean Accuracy of 70.0%: 0.792
Standard Deviation of 70.0%: 0.00979795897113


Sampling 5 iterations on 80.0%...
numBatch is 4000
Mean Accuracy of 80.0%: 0.788
Standard Deviation of 80.0%: 0.0116619037897


Sampling 5 iterations on 90.0%...
numBatch is 4500
Mean Accuracy of 90.0%: 0.794
Standard Deviation of 90.0%: 0.008


Sampling 5 iterations on 100%...
numBatch is 5000
Mean Accuracy of 100%: 0.79
Standard Deviation of 100%: 0.0
```

```
Training time 139.978657007 | Number of iterations 5
```

Listing 2: digit_perceptron.txt

```
Doing classification
--------------------
data:           digits
classifier:             perceptron
using enhanced features?:       False
training set size:      5000
Extracting features...
Sampling 5 iterations on 10.0%...
numBatch is 500
Mean Accuracy of 10.0%: 0.792
Standard Deviation of 10.0%: 0.0116619037897



Sampling 5 iterations on 20.0%...
numBatch is 1000
Mean Accuracy of 20.0%: 0.816
Standard Deviation of 20.0%: 0.0300665927567



Sampling 5 iterations on 30.0%...
numBatch is 1500
Mean Accuracy of 30.0%: 0.83
Standard Deviation of 30.0%: 0.0236643191324



Sampling 5 iterations on 40.0%...
numBatch is 2000
Mean Accuracy of 40.0%: 0.844
Standard Deviation of 40.0%: 0.012



Sampling 5 iterations on 50.0%...
numBatch is 2500
Mean Accuracy of 50.0%: 0.852
Standard Deviation of 50.0%: 0.0132664991614



Sampling 5 iterations on 60.0%...
numBatch is 3000
Mean Accuracy of 60.0%: 0.856
Standard Deviation of 60.0%: 0.0233238075794
```

```
Sampling 5 iterations on 70.0%...
numBatch is 3500
Mean Accuracy of 70.0%: 0.866
Standard Deviation of 70.0%: 0.0101980390272



Sampling 5 iterations on 80.0%...
numBatch is 4000
Mean Accuracy of 80.0%: 0.86
Standard Deviation of 80.0%: 0.0141421356237



Sampling 5 iterations on 90.0%...
numBatch is 4500
Mean Accuracy of 90.0%: 0.856
Standard Deviation of 90.0%: 0.0174355957742



Sampling 5 iterations on 100%...
numBatch is 5000
Mean Accuracy of 100%: 0.86
Standard Deviation of 100%: 0.0



Training time 3220.738235 | Number of iterations 5
```

Listing 3: faces_naiveBayes.txt

```
Doing classification
--------------------
data:            faces
classifier:              naiveBayes
using enhanced features?:        False
training set size:       150
using smoothing parameter k=2.000000 for naivebayes
Extracting features...
Sampling 5 iterations on 10.0%...
numBatch is 15
Mean Accuracy of 10.0%: 0.532
Standard Deviation of 10.0%: 0.0636867333124



Sampling 5 iterations on 20.0%...
numBatch is 30
Mean Accuracy of 20.0%: 0.574
Standard Deviation of 20.0%: 0.0504380808517



Sampling 5 iterations on 30.0%...
```

```
numBatch is 45
Mean Accuracy of 30.0%: 0.594
Standard Deviation of 30.0%: 0.0392937654088



Sampling 5 iterations on 40.0%...
numBatch is 60
Mean Accuracy of 40.0%: 0.762
Standard Deviation of 40.0%: 0.00748331477355



Sampling 5 iterations on 50.0%...
numBatch is 75
Mean Accuracy of 50.0%: 0.762
Standard Deviation of 50.0%: 0.0426145515053



Sampling 5 iterations on 60.0%...
numBatch is 90
Mean Accuracy of 60.0%: 0.81
Standard Deviation of 60.0%: 0.0414728827067



Sampling 5 iterations on 70.0%...
numBatch is 105
Mean Accuracy of 70.0%: 0.778
Standard Deviation of 70.0%: 0.0299332590942



Sampling 5 iterations on 80.0%...
numBatch is 120
Mean Accuracy of 80.0%: 0.804
Standard Deviation of 80.0%: 0.0387814388593



Sampling 5 iterations on 90.0%...
numBatch is 135
Mean Accuracy of 90.0%: 0.814
Standard Deviation of 90.0%: 0.00489897948557



Sampling 5 iterations on 100%...
numBatch is 150
Mean Accuracy of 100%: 0.84
Standard Deviation of 100%: 1.11022302463e-16
```

```
Training time 80.3584620953 | Number of iterations 5
```

Listing 4: faces_perceptron.txt

```
Doing classification
--------------------
data:           faces
classifier:             perceptron
using enhanced features?:       False
training set size:      150
Extracting features...
Sampling 5 iterations on 10.0%...
numBatch is 15
Mean Accuracy of 10.0%: 0.748
Standard Deviation of 10.0%: 0.0934665715644



Sampling 5 iterations on 20.0%...
numBatch is 30
Mean Accuracy of 20.0%: 0.824
Standard Deviation of 20.0%: 0.020591260282



Sampling 5 iterations on 30.0%...
numBatch is 45
Mean Accuracy of 30.0%: 0.798
Standard Deviation of 30.0%: 0.0430813184571



Sampling 5 iterations on 40.0%...
numBatch is 60
Mean Accuracy of 40.0%: 0.85
Standard Deviation of 40.0%: 0.0



Sampling 5 iterations on 50.0%...
numBatch is 75
Mean Accuracy of 50.0%: 0.85
Standard Deviation of 50.0%: 0.0



Sampling 5 iterations on 60.0%...
numBatch is 90
Mean Accuracy of 60.0%: 0.85
Standard Deviation of 60.0%: 0.0



Sampling 5 iterations on 70.0%...
```

```
numBatch is 105
Mean Accuracy of 70.0%: 0.85
Standard Deviation of 70.0%: 0.0



Sampling 5 iterations on 80.0%...
numBatch is 120
Mean Accuracy of 80.0%: 0.85
Standard Deviation of 80.0%: 0.0



Sampling 5 iterations on 90.0%...
numBatch is 135
Mean Accuracy of 90.0%: 0.85
Standard Deviation of 90.0%: 0.0



Sampling 5 iterations on 100%...
numBatch is 150
Mean Accuracy of 100%: 0.85
Standard Deviation of 100%: 0.0
```

```
Training time 499.263456106 | Number of iterations 5
```

Observation #1: Digit classifying The running time of Perceptron toke nearly 23 times slower than Naive Bayes!!! In terms of accuracy and standard deviation, Naive Bayes showed that there was a steady increase trend in accuracy. However, the increases were small the accuracy started to converge as the standard deviation converges to 0.0 at the final sampling with 100% of the data.

In the mean time, Perceptron showed higher accuracy with 86% and also showed small improvements on bigger training data. However, the changes were not significant enough to form a conclusion, and the standard deviation converged to 0.0 as the size of training data increases but need more samplings to jump to conclusion if standard deviation converges.

Observation #2: Faces classifying The running time of Perceptron took more than 6 times slower than Naive Bayes method. However, Perceptron had slightly better results than Naive Bayes - 85% vs 84%. Both methods obtain convergence in standard deviation close to 0.0

3 Explanation

Through the above observations, we can conclude that Perceptron takes more time to train the model (some times excessively large) but gives higher results. The reason why Perceptron takes longer time to train the model is that for each datum trained by the model, it needs iterating through the datum multiple times to self-adjust their weights for better result. Therefore, the time-accuracy constraints of the method is proportional and gives better rewards if we give it enough time to train.