CS440 Project 2

Raheel Ozair
ro173

Yanbo Jiang
yj274

8/5/2020

- In terms of implementation: perceptron starts by making a guess- if it is correct, it does nothing and moves to the next datum point; otherwise, it guessed wrong, and it will increment the correct label's weight and decrement the incorrect label's weight.
  On the other hand, for NaiveBayes, we make the assumption with our features vector that any feature is conditionally independent of any other feature, and uses Bayes' Theorem to calculate the label most probable given the feature values of each pixel.

- When comparing Bayes and Perceptron (with only 1 iteration) on a training set size of 450, we found that Bayes was consistently faster by a large margin. When decreasing the training set size to around 100, both programs performed around the same in terms of time, with Bayes being slightly faster. Here's some data, using Faces:

  - Bayes testing averages (5 repetitions for each percentage of training data):
    * 10% of training data: 49%, Std Dev: 4.0
    * 20% of training data: 54%, Std Dev: 2.0
    * 30% of training data: 61%, Std Dev: 5.0
    * 40% of training data: 61%, Std Dev: 7.5
    * 50% of training data: 69%, Std Dev: 4.1
    * 60% of training data: 72%, Std Dev: 3.8
    * 70% of training data: 69%, Std Dev: 1.9
    * 80% of training data: 72%, Std Dev: 1.9
    * 90% of training data: 72%, Std Dev: 1.0
    * 100% of training data: 73%, Std Dev: 0.0
  - Perceptron testing averages (5 repetitions for each percentage of training data, on 5 iterations of Perceptron):
    * 10% of training data: 77.8%, Std Dev: 12.4
    * 20% of training data: 78.8%, Std Dev: 10.4
    * 30% of training data: 81.0%, StdDev: 6.0
    * 40% of training data: 82.8%, StdDev: 2.4
    * 50% of training data: 83.8%, Std Dev: 0.4
    * 60% of training data: 83.4%, Std Dev: 1.2
    * 70% of training data: 83.4%, Std Dev: 1.2
    * 80% of training data: 84.0%, Std Dev: 0.0

    * 90% of training data: 84.0%, Std Dev: 0.0
    * 100% of training data: 84.0%, Std Dev: 0.0

  Thus, it's clear that while Bayes is much quicker per iteration, Perceptron is much more accurate on average.
  It's also pertinent to note that Perceptron becomes more accurate in less iterations than Bayes takes to become accurate. On the first repetition of Perceptron, it reached an accuracy of 70% after the first iteration and did not decrease below, whereas Bayes did not reach 70% accuracy until having about 60% of the training data, on average.
  Additionally, it seems that while Perceptron has a larger standard deviation initially than Bayes, this approaches 0 more quickly than Bayes does.

- We may also compare Bayes and Perceptron on Digits:
  - Bayes Testing averages (5 repetitions for each percentage of training data)
    * 10% of training data: 76.8%, Standard Dev: 3.6
    * 20% of training data: 76.4%, Standard Dev: 1.3
    * 30% of training data: 79.0%, Standard Dev: 2.1
    * 40% of training data: 79.2%, Standard Dev: 1.3
    * 50% of training data: 77.8%, Standard Dev: 1.7
    * 60% of training data: 79.0%, Standard Dev: 0.6
    * 70% of training data: 79.4%, Standard Dev: 1.6
    * 80% of training data: 79.4%, Standard Dev: 0.5
    * 90% of training data: 79.0%, Standard Dev: 0.0
    * 100% of training data: 79.0%, Standard Dev: 0.0
  - Attempting to run Perceptron on all of Digits would have taken extremely long, so instead, we'll run it on 500 data points from digits with 5 iterations, repeated 5 times:
    * 10% of training data: 75.0%, Standard Dev: 14.0
    * 20% of training data: 80.4%, Standard Dev: 3.2
    * 30% of training data: 79.2%, Standard Dev: 5.6
    * 40% of training data: 80.8%, Standard Dev: 2.4
    * 50% of training data: 80.0%, Standard Dev: 4.0
    * 60% of training data: 79.2%, Standard Dev: 5.6
    * 70% of training data: 81.4%, Standard Dev: 1.2

* 80% of training data: 82.0%, Standard Dev: 0.0
* 90% of training data: 82.0%, Standard Dev: 0.0
* 100% of training data: 82.0%, Standard Dev: 0.0

As you can see, even though we only ran Perceptron on 500 data points, it still maintains a higher degree of accuracy (especially with training data amounts over 70%) than Bayes in most cases, with more-quicky decreasing standard deviation than Bayes.