# CS 440: Introduction to Artificial Intelligence

## Final Project Digit/Face Recognition

May 1, 2023

*Professor Abdelsam Boularias*

**Jay Patwardhan** 208001851
**Alan Wu** 208000574
**Neel Shejwalkar** 207004853

# Problem 0

## Constructing Features and Algorithms

Our features were each pixel: they have a status of 1 or 0 depending on whether they are black or white. The 3 algorithms are the 3 on the Berkeley website: Naive Bayes, Perceptron, and MIRA.
Naive Bayes uses Bayes formula under the independence of feature assumption to compute probabilities, it's a straightforward algorithm, but the bookkeeping is a little tedious. We estimate parameters using the training data, and apply a smoothing constant k to reduce an overfitting concern. We find the best model over a grid of potential k values by testing each one. Finally, we compute log probabilities since multiplying probabilities results in underflow, but the monotonicity and injectivity of the logarithm function grants us that our conversion remains faithful and doesn't mess up the probabilities.

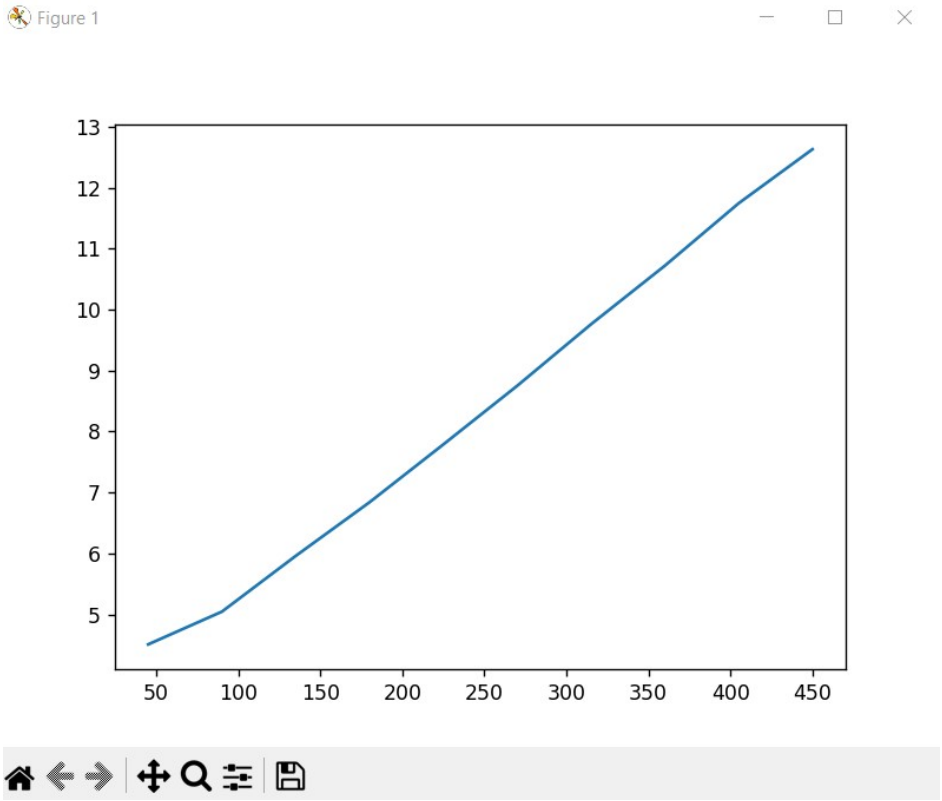
Perceptron is a single layered neural network which utilizes online learning, so it takes in each sample at a time, and computes a guess, compares it to the actual label, and tweaks the weights for each label accordingly to make the correct label more likely to be the output of the prediction function. We run multiple (3) iterations of this to make sure that everything is properly tuned and trained, but we don't run it too many times to make sure that we don't overtrain on our training data.

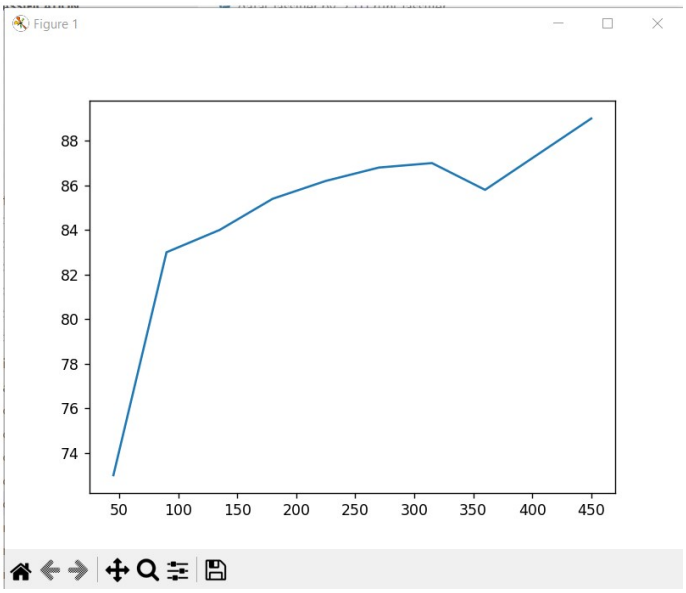
MIRA is an extension of the perceptron, where instead of updating weights only by the vector of features, we attach a nonnegative scalar constant to the features and update by this multiplication of $\tau * f$. This scalar is chosen such that it minimizes the sum of dot products of each estimated label minus the actual label, attaching a sense of Euclidean distance metric to the space in order to find the closest possible reflection of the actual outcome. Without dealing with the technical aspects of actually minimizing this function, this comes down to selecting a minimum of some constant C, chosen to attach a maximum value of our scalar, and another value determined by taking the vector distance of the labels, multiplying by the feature function, adding 1, and dividing by the dot product of the features to keep our values from getting too large if our feature vector contains big values, in an almost normalization measure (except we miss the square root here). When implementing MIRA, we test multiple values of our constants C to determine which gives us the best model for our training data, and of course we keep the multiple iteration rounds to compute and tweak our labels from the perceptron model.

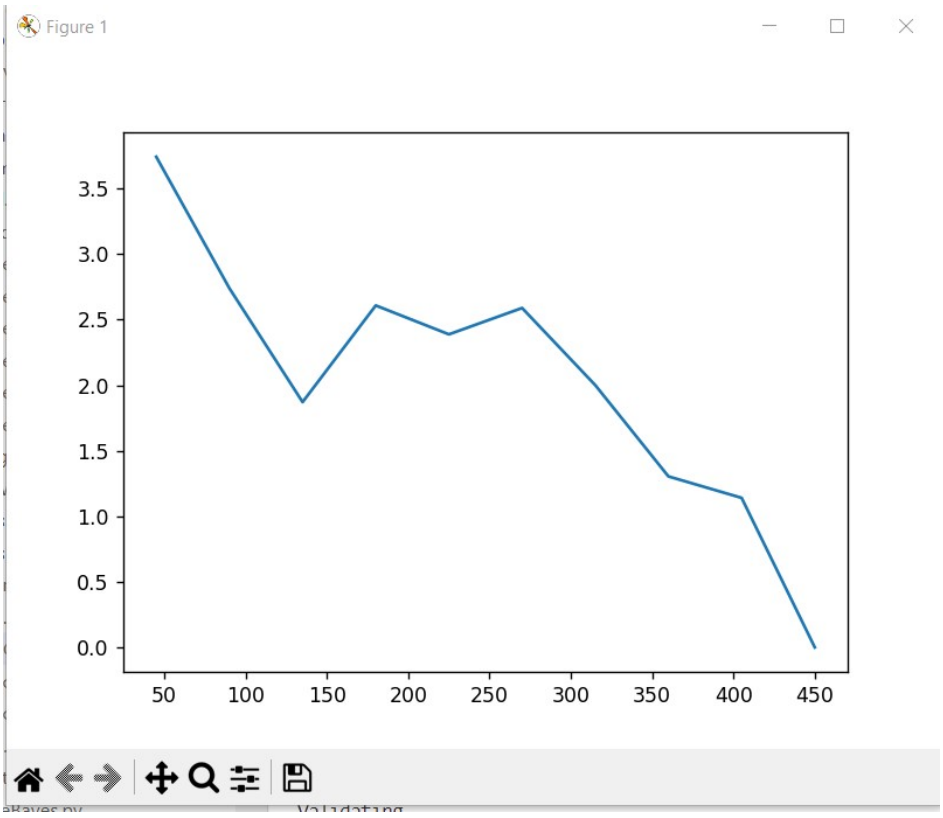
# Problem 1

## Naive Bayes

We show plots and results for Naive Bayes on faces. The digits case is close to identical, so in order to avoid redundancy we attach only the faces plots to show our results. One thing to note about the digits plots are that since the number of samples are so large, they become uniform very quickly, even from the beginning for certain tests like accuracy; even though their performance does get better with more samples, the distinction isn't as striking as the faces plots, since we only work with a maximum of 450 faces versus 5000 digits. Attached are plots of the time it took to train Naive Bayes against the number of samples used to train, the mean of each iteration (10 percent, 20 percent,...,100 percent samples) against the number of samples, and finally the standard deviation of each iteration against the number of samples. Note that all samples were taken randomly from the whole.





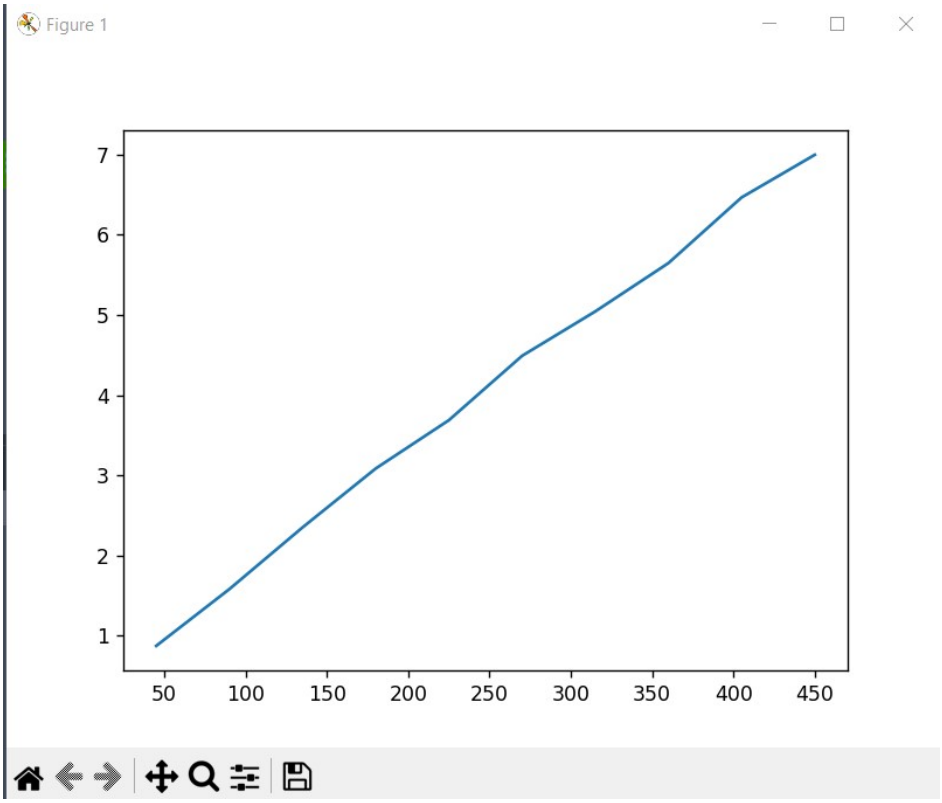




It is clear to see as well that the mean accuracy increases with more samples and the standard deviation decreases dramatically, so the estimates get increasingly precise.

# Problem 2

## Perceptron

Same as the Naive Bayes case, we show our results for faces, the digits plots have an identical discussion as the one we did for Naive Bayes. In the same order as above, we have the plots: time vs samples, mean vs samples, and std vs samples:



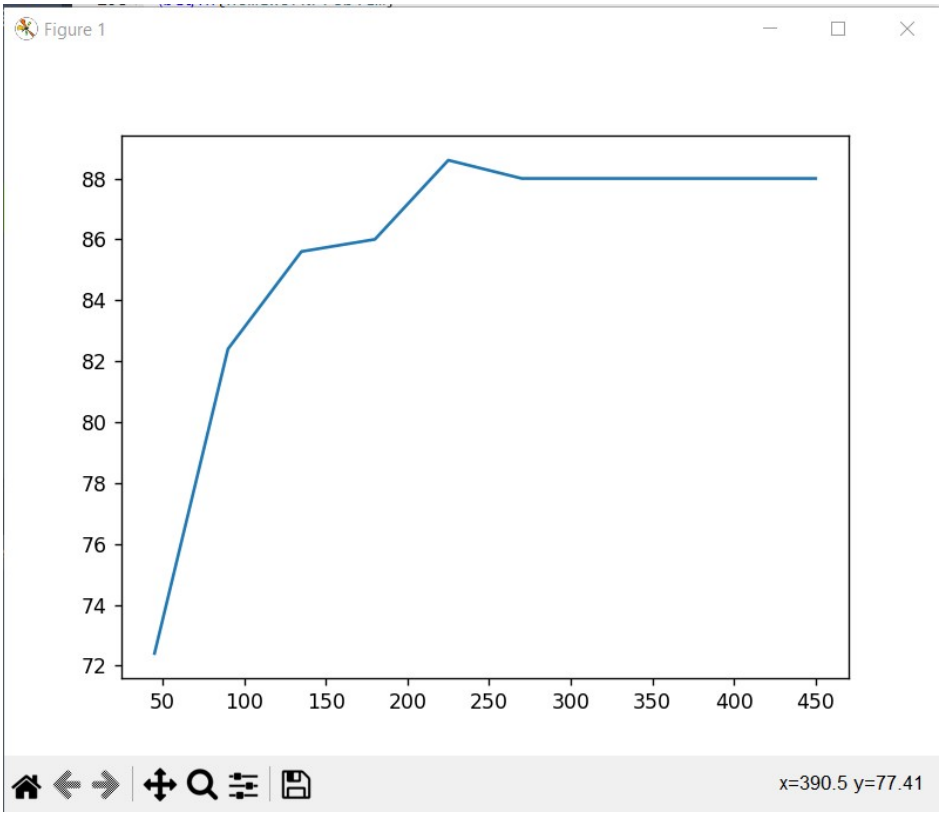


Figure 1
88
86
84
82
80
78
76
74
72
50
100
150
200
250
300
350
400
450
x=390.5 y=77.41

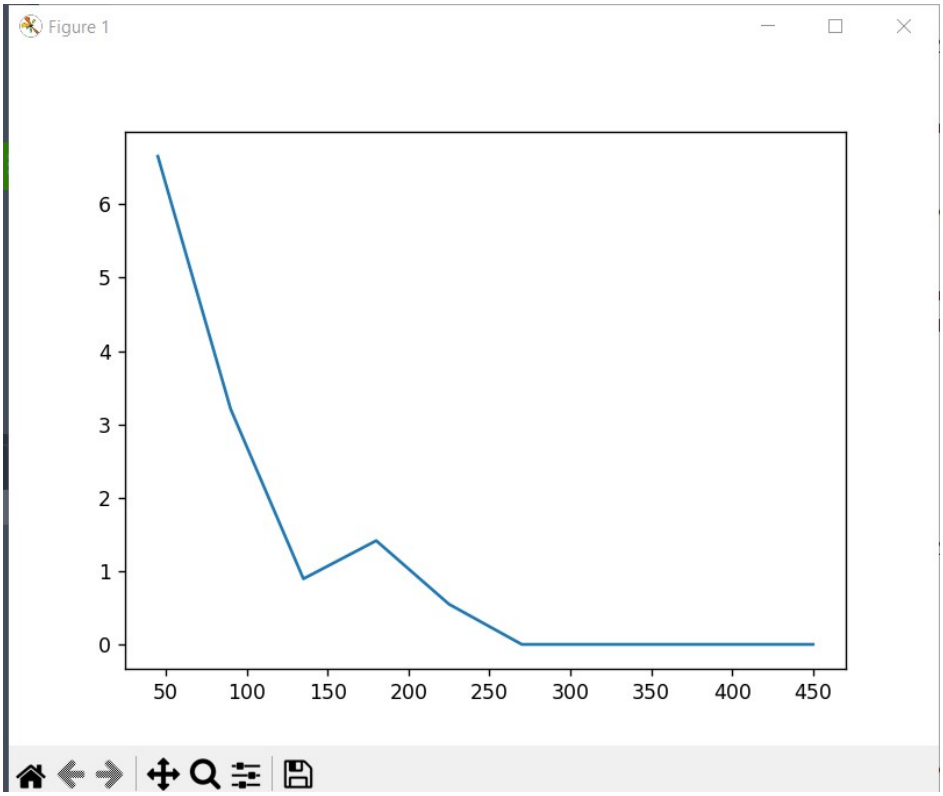
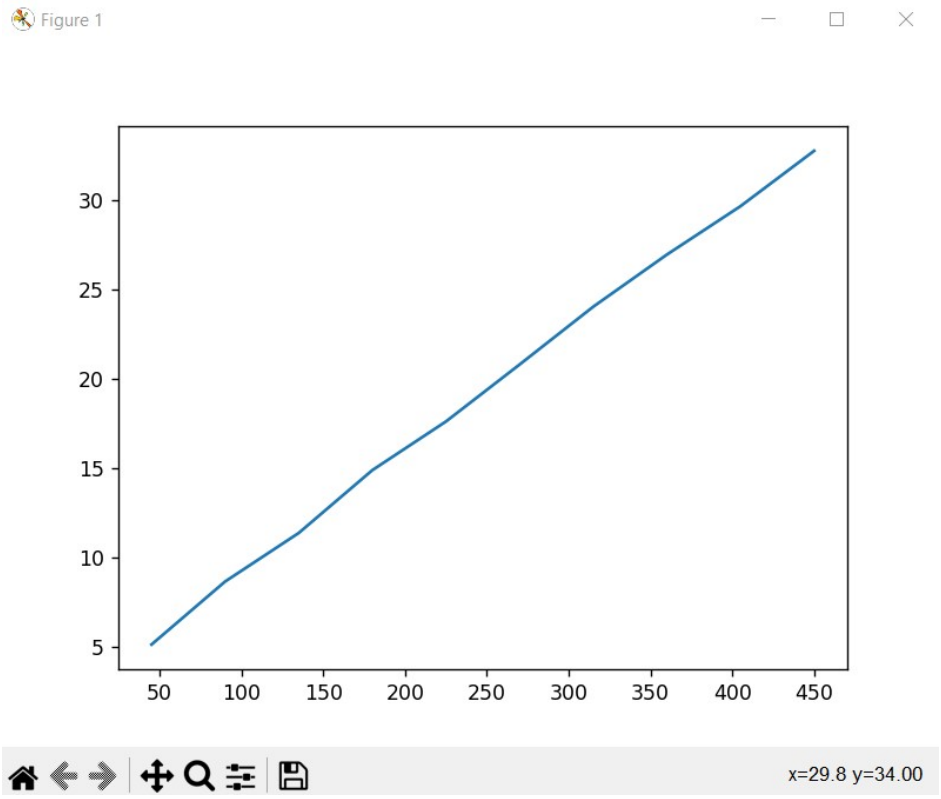


Figure 1
6
5
4
3
2
1
0
50
100
150
200
250
300
350
400
450

# Problem 3

## MIRA

Same as the previous two cases. We plot faces. The plots are: time vs samples, mean vs samples, and std vs samples.



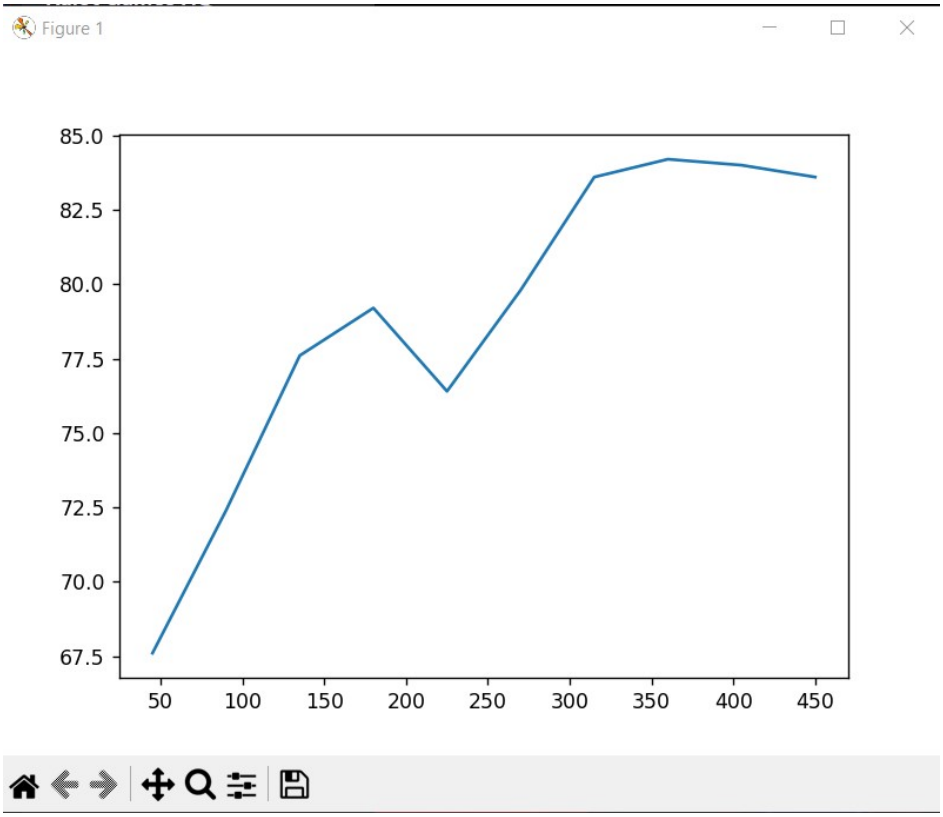


Figure 1
85.0
82.5
80.0
77.5
75.0
72.5
70.0
67.5
50
100
150
200
250
300
350
400
450

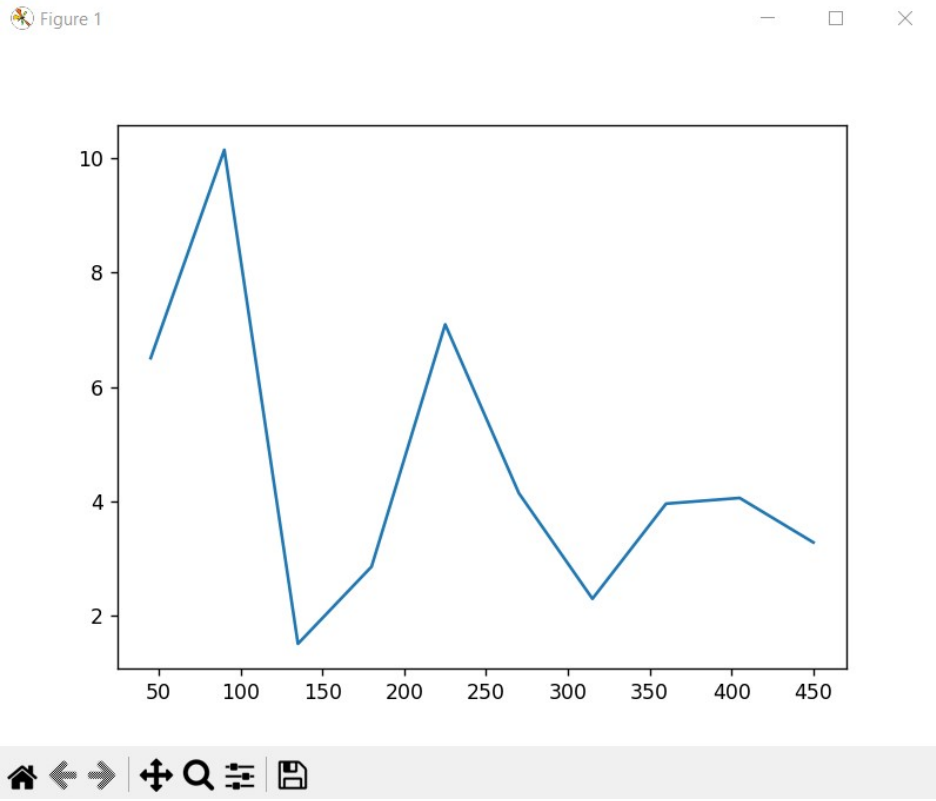


Figure 1
10
8
6
4
2
50
100
150
200
250
300
350
400
450

# Problem 4

## Comparative Analysis

For the full sample size, the average time for Naive Bayes was 13 seconds, the average time for perceptron was 7 seconds, and the average time for MIRA was 30 seconds. Our test set for all cases was 100 samples to construct these plots. The average accuracy for Naive bayes was 88 percent, the average accuracy for the perceptron was also 88 percent, and the average accuracy for MIRA was 85 percent. All standard deviations generally converge to zero, although MIRA has a slightly higher standard deviation of 2. In general, the first two algorithms perform quite well, and perceptron is the fastest and most accurate. The increased complexity of MIRA made it cost extra time. The learned lessons is that basic features, even raw pixel mapping, can be very effective and lead to quite good estimations, and of course that a greater size training set generally leads to best results. We also note that smoothing is quite effective in the Naive Bayes estimation and helps to prevent taking the logarithm of 0 in case a feature was 0.