Classification Report

Apurva Narde, Sai Kalagotla

August 17, 2021

Implementation

Naive Bayes

We first calculate the prior distribution $\hat{P}(y) = \frac{c(y)}{n}$, where $c(y)$ represents the count of the training instances that have the label y. Here, $y \in \{0, 1, ..., 9\}$ (for digits) and $y \in \{0, 1\}$ (for faces). We then created a dictionary to keep track of the counts of the feature values per feature (784 for digits and 4440 for faces) per label (10 for digits and 2 for faces). We then proceeded on towards smoothing (specifically Laplace smoothing) which we calculate as follows,

$$\hat{P}(F_i = f_i|Y = y) = \frac{c(f_i, y) + k}{\sum_{f_i' \in \{0,1,2\}} c(f_i', y) + k}.$$

Here we computed the probability table for each value of k in $kgrid$. We chose the k value that demonstrated the highest accuracy in the validation data. One component of naive bayes was to calculate the log join probability which was calculated using the prior distribution $\hat{P}(y)$ and the conditional probabilities $\hat{P}(F_i = f_i|Y = y)$ as follows,

$$\arg\max_y \log P(y|f_1, ..., f_m) = \arg\max_y \left\{ \log \hat{P}(y) + \sum_{i=1}^{m} \log \hat{P}(f_i|y) \right\}.$$

Note: The log probabilities have the same arg max.

Perceptron

We implemented the Perceptron Classifier by first making a dictionary of dictionaries to hold all the weights for each feature. We made 10 weight dictionaries when classifying digits, and 2 for faces, one for each label. Here, $y \in \{0, 1, ..., 9\}$ (for digits) and $y \in \{0, 1\}$ (for faces). The initial value for the weights was set to be 1. We then calculated the score for each label and took the arg max to get the guess label. To calculate the guess label we used

$$y' = \arg\max_{y''} score(f, y'')$$

where y' is the guessed label. After the algorithm made a guess if the guess was wrong we updated the weights by using

$$w^y = w^y + f$$
$$w^{y'} = w^{y'} - f$$

where w^y is the true label's dictionary of weights and $w^{y'}$ is the guess label's dictionary of weights and f is the datum's feature values. Using the equations we decreased the weights of the guessed label and increased the weights of the true label with the feature values of the training data. Otherwise, if the guess was correct we moved on to the next training datum without updating any weights.

MIRA

The MIRA classifier is similar to our Perceptron classifier where we maintain a weight vector w^y for each label y and find the label with the highest score, $y' = \arg\max_{y''} score(f, y'')$. However, the only difference is in the way we update the weights,

$$w^y = w^y + \tau f$$
$$w^{y'} = w^{y'} - \tau f$$

where we compute τ as

$$\tau = \min \left\{ C, \frac{(w^{y'} - w^y) f + 1}{2\|f\|_2^2} \right\}$$

where $C \in \mathbb{R}^+$. Similar to the naive bayes approach we iterate through each value of C in $Cgrid$ and compute the weights which are then tested using the validation data for the C value of the highest validation accuracy.

Results

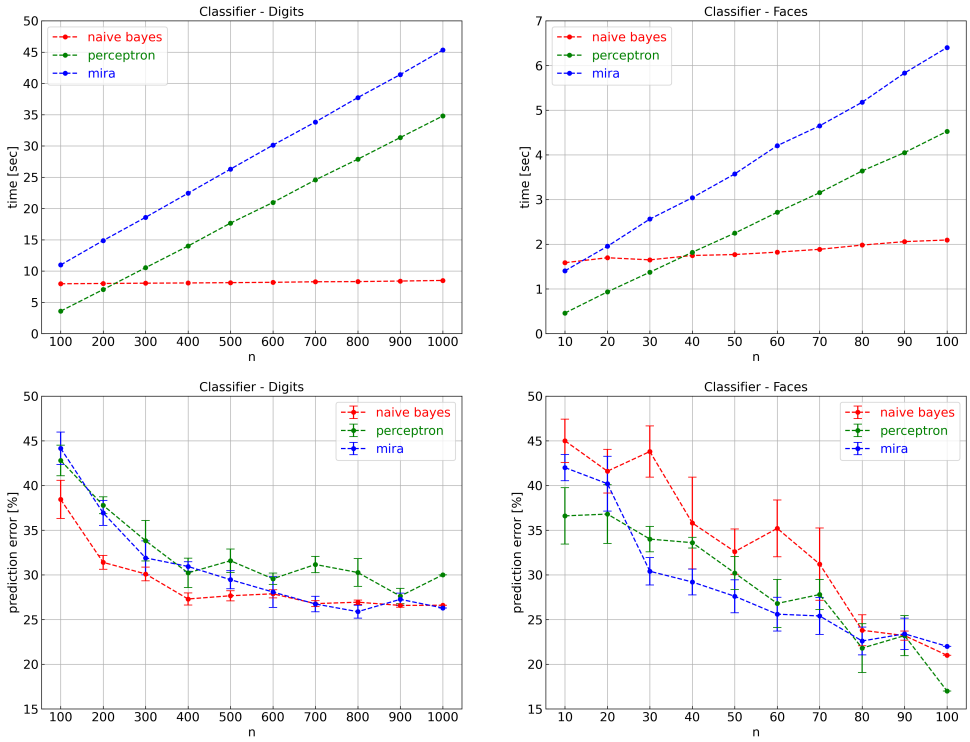

Figure 1: Comparing the performance of: Naive Bayes, Perceptron, and MIRA

In Figure 1 we graphed the time each classifier took on the top row and the prediction error for each classifier in the bottom row. Throughout the top row of graphs we see a common trend where the time it takes to train increases based on the size of the training data. However, the Naive Bayes algorithm had a minimal increase in computation time compared to the other 2 algorithms used. This is because in Perceptron and MIRA, we have to iterate any number of times (we chose to use 5) for each datum to get the weights to the optimal value. Thus, as the size of the training data increases by n, the computation time increases by $5n$. However, with Naive Bayes we only have to calculate the probability once per feature value per feature per label. Another trend we can observe for all the algorithms is that the mean prediction error percentage decreases as we increase the training size as shown in the bottom row of Figure 1. This occurs because as we increase the training size each algorithm receives more information to fine tune its parameters. Perceptron and MIRA have more opportunities to update its weights while Naive Bayes is able to get better probabilities to make guesses off of. Another interesting feature we can look at is the standard error of the mean prediction error percentage for each algorithm. From the bottom row of Figure 1 we can see that the standard error when classifying digits is lower that the standard error when classifying faces. A possible reason could be because of the amount of data used for training. We used up to 1000 data points for digit and up to 100 for faces. However, recall that digits had 784 features while faces had 4440 features. Therefore, there is a lot more error when classifying faces because we trained large number of feature with a small amount of data as opposed to training small number of features with a large amount of data we observe less error in digits.

References

[1] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2009.

[2] Berkeley. *Project 5: Classification*. https://inst.eecs.berkeley.edu//∼cs188/sp11/projects/classification/classification.html