Naïve Bayes With Laplace Smoothing: Multivariate Bernoulli and Multinomial Event Models

Joe Shymanski

Department of Computer Science, University of Tulsa
CS-5333: Machine Learning, Dr. Sandip Sen, January 27, 2022

Abstract. This paper seeks to report on the implementation of the Naïve Bayes algorithm with Laplace smoothing. The learner will use the multivariate Bernoulli and multinomial event models to test against a text dataset for spam detection and an image dataset for handwritten digit recognition.

1 Introduction

The Naïve Bayes assumption is incredibly useful for classifying large, discrete-valued datasets. There are a plethora of domains with can be predicted well using a Naïve Bayes model, from categorizing annoying emails to forecasting Alzheimer's years before its symptoms using genome data [4]. Its implementation for different problems can vary, or it can be used in a generic form to apply to almost any domain. Due to complications from the kinds of probabilities utilized within the algorithm, some smoothing technique needs to be employed to ensure accurate results. There are several event models to consider when making the calculations as well.

2 Models

It is certainly possible to build a generic Naïve Bayes learner to fit any number of supervised learning problems. While such an algorithm would have its obvious benefits, I anticipate certain domains would not be classified as efficiently or accurately as a slightly more personalized Naïve Bayes algorithm would allow. The latter could also be implemented much more intuitively for the different problem sets. Thus, the learner created in this project has two slightly different modes depending on the type of data it is trying to classify.

2.1 Digit Recognition

One domain the algorithm can tackle is that of handwritten digit recognition. When working with this data, the learner follows a much more generic approach to classification. It utilizes the multinomial event model.

When training on a dataset, the algorithm seeks to create two important data tables for easy lookup during prediction: a table of label frequencies and a table of feature frequencies within each label. In this domain, the features are each pixel of a 28×28 image of a handwritten digit. The original dataset from MNIST contained intensity levels from 0 to 255 for each pixel [2]. For this Naïve Bayes learner, the pixels with intensity 128 or higher were considered "on" and given a value of 1; the rest were considered "off" and given a value of 0.

The algorithm needs two key inputs to begin training a model: a training dataset and a value of k for Laplace smoothing. Using the training data, the learner will immediately create the label

frequency table by dividing the number of instances of each unique label found by the total number of instances. Then it calculates the number of times each feature (pixel) has a value of 1 ("on") given a label, and it does this for all labels. This is the numerator for determining the probability of the feature given the label. The algorithm then adds k to every count; this is the Laplace smoothing in action. For the denominator, two different methods were investigated. The first was the suggested approach of adding the total number of instances for the given label to k times the number of features ($k * |F|$). The second was a variation that I had initially incorrectly implemented, which instead just added $k + 1$ to the label's total. I had to add the 1 to avoid getting any probabilities of 1, which would turn into zeroes when calculating the probability that the pixel is "off". This second approach seemed to perform much better than the first, which will be discussed later.

After all these calculations, the label and feature probabilities are returned to form the model. Now the model can be used to predict unseen data. The prediction function accepts the Naïve Bayes model created from the training dataset as well as a feature vector for an unseen sample. It first extracts the two probability tables from the model. Then it loops over all the labels and computes the following for each label:

$$\ln P(l) + \sum_i \ln P(F_i|l)$$

where $P(l)$ represents the probability of the current label and $P(F_i|l)$ represents the probability of the ith feature of the feature vector given the label. The natural log is used to avoid any underflow errors in these calculations. For digit recognition, if the new pixel is "on", the given probability is used; $1 - P(F_i|l)$ is used otherwise. Once a value has been calculated for each label, the label with the maximum value is returned as the predicted class.

2.2 Spam Filter

Spam classification is the other domain that the algorithm handles. When working with this textual data, which comes from UCI [3], the learner follows a slightly more personalized approach. It also has an option to use either the multinomial event model or the multivariate Bernoulli event model. Compared to the algorithm for digit recognition, the majority of the learner is the same. However, there are three main differences.

The first occurs within the training function when trying to count all the features. Instead of having a vocabulary vector of all English words, the learner simply counts the words it encounters. This reduces the size of the problem immensely and allows the learner to loop over many more text samples in training. It also allows the learner to consider words that are not in the dictionary or the English language at all, which are common within text messages. The algorithm does not clean punctuation or capitalization at all, which could potentially lead to better performance or generalization. If the Bernoulli model is being used, the words are only counted once per text, even if it occurs multiple times within the text. If the multinomial model is in effect, the words are counted as many times as they appear.

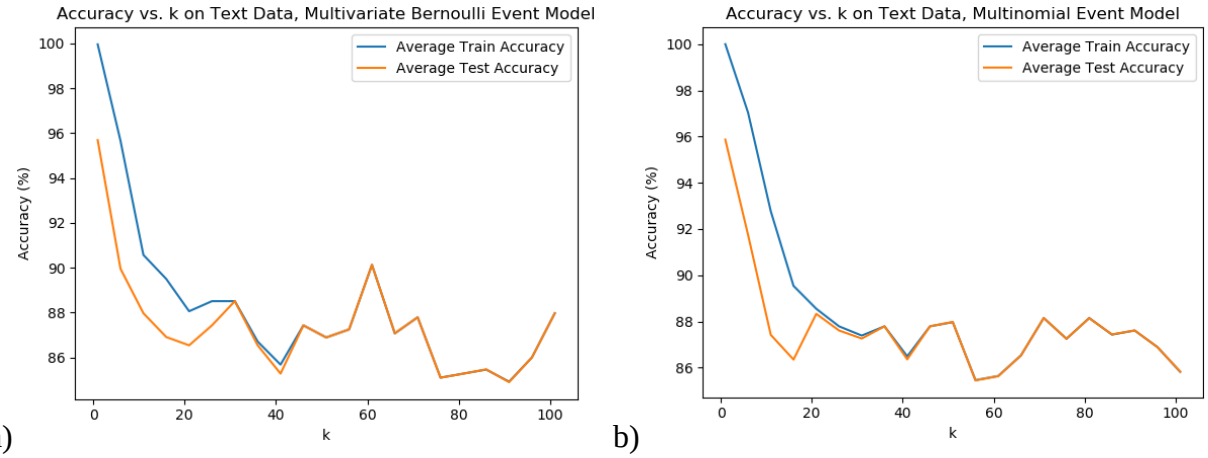
The next difference occurs when dividing the word counts to obtain the feature probabilities. The denominator of each fraction is simply the total number of words within the given label. This occurs after Laplace smoothing on the word counts, so these extra counts are already factored into the numerator and denominator.

The final difference takes place in the prediction function. Unlike the digit recognition problem, the algorithm does not calculate $1 - P(F_i|l)$ for words that do not occur in the unseen

sample. If it does not recognize a word, it simply does not consider it when calculating the probabilities.

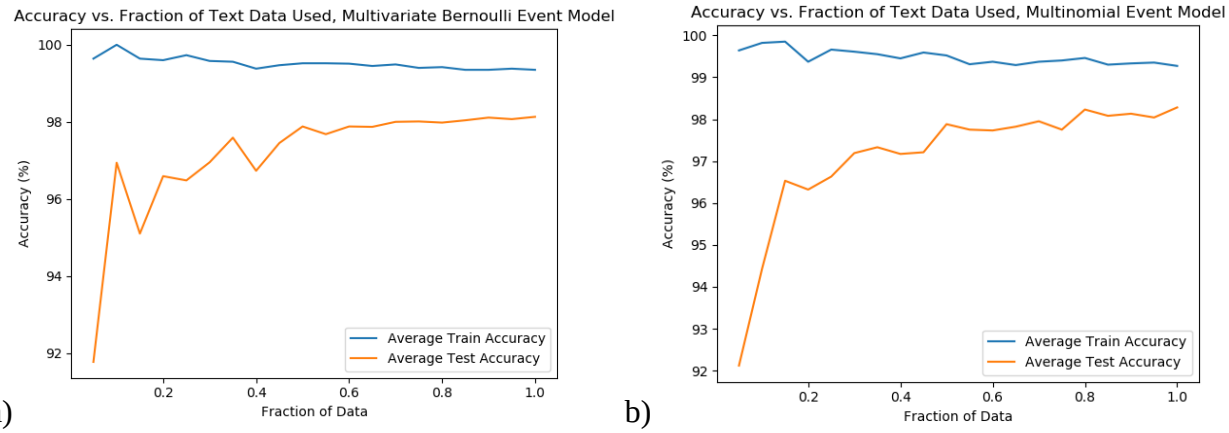
3 Experimental Results

3.1 Hyperparameters



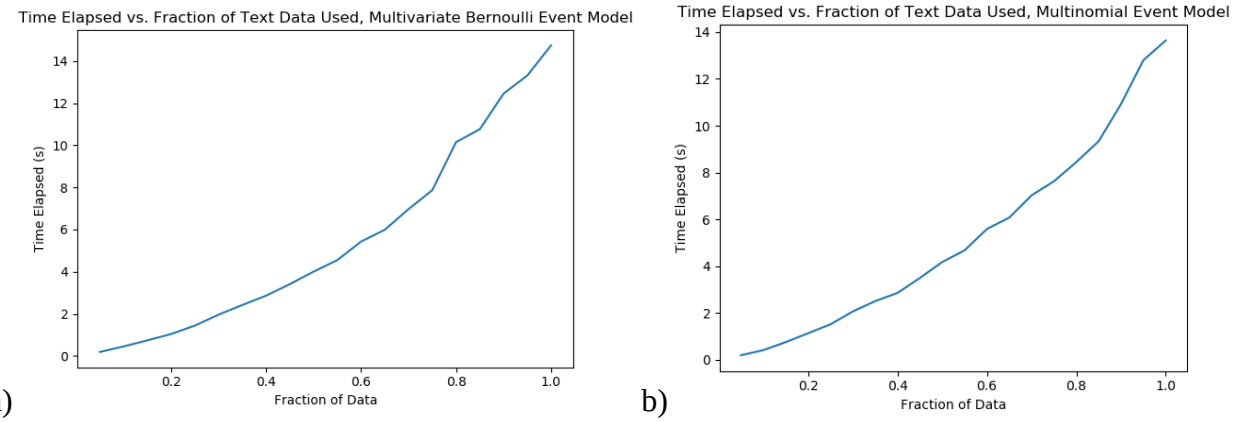
a) b)

Figure 1. Average training and testing accuracies against the value of k on the text dataset using: a) the multivariate Bernoulli event model; b) the multinomial event model. Both used 10% of the dataset (about 557 samples).



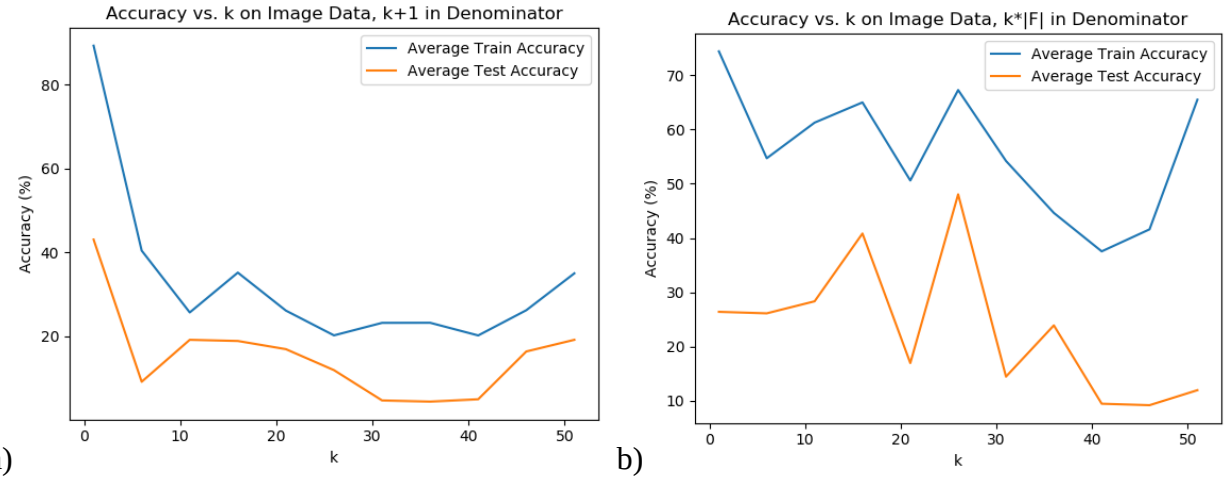
a) b)

Figure 2. Average training and testing accuracies against the fraction of the text dataset used with: a) the multivariate Bernoulli event model; b) the multinomial event model. Both used $k = 1$.



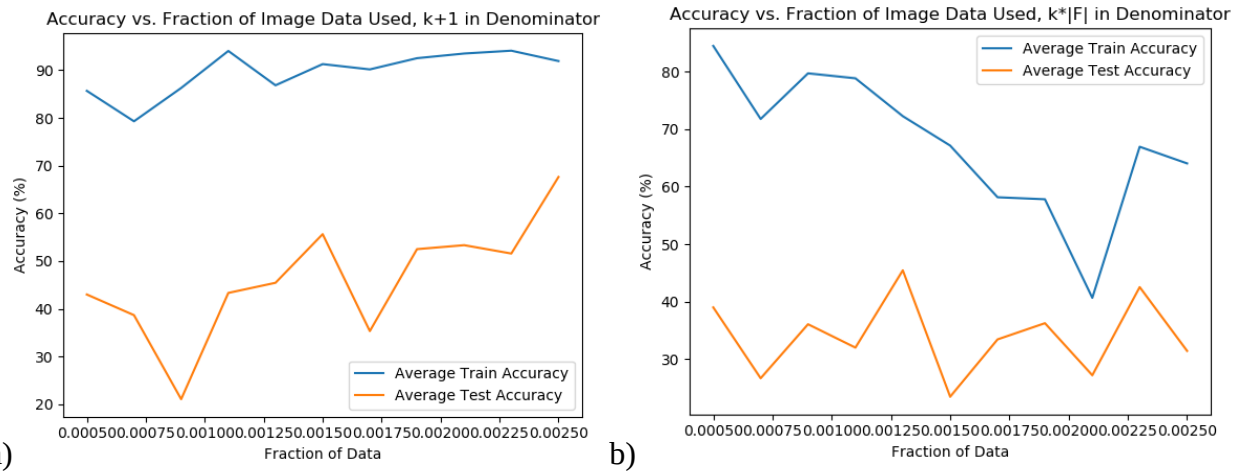
a) b)

Figure 3. The average time elapsed against the fraction of the text dataset used with: a) the multivariate Bernoulli event model; b) the multinomial event model. Both used $k = 1$.



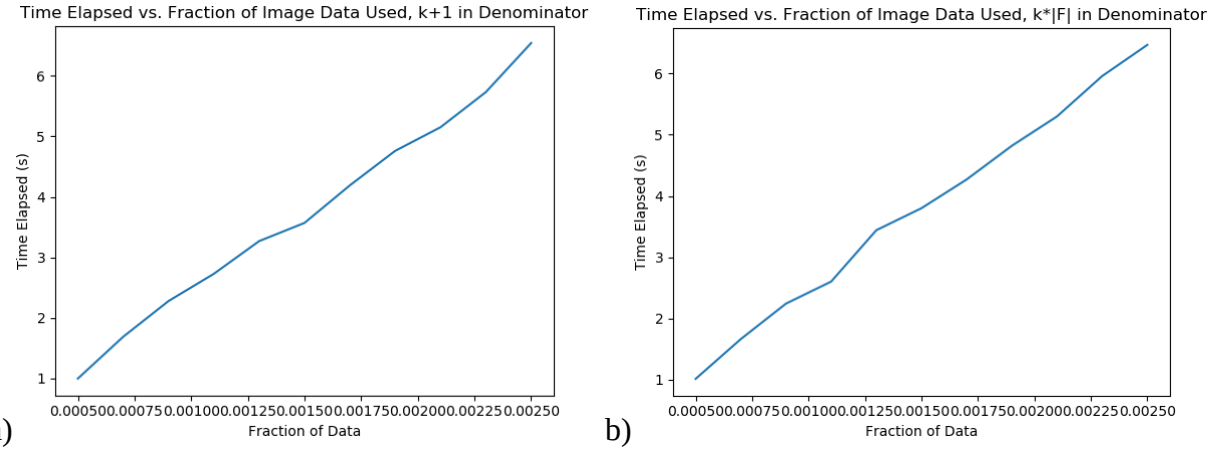
a) b)

Figure 4. Average training and testing accuracies against the value of k on the image dataset using: a) $k + 1$ in the denominator; b) $k * |F|$ in the denominator. Both used 0.1% of the dataset (42 samples).



a) b)

Figure 5. Average training and testing accuracies against the fraction of the image dataset used with: a) $k + 1$ in the denominator; b) $k * |F|$ in the denominator. Both used $k = 1$.



a) b)

Figure 6. The average time elapsed against the fraction of the image dataset used with: a) $k + 1$ in the denominator; b) $k * |F|$ in the denominator. Both used $k = 1$.

3.2 Tuned Results

Dataset, Model	Training Set Accuracy (%)	Testing Set Accuracy (%)	Time Elapsed (s)
Text, Bernoulli	99.34	98.29	11.97
Text, Multinomial	99.30	98.24	12.10
Image, $k + 1$	85.77	76.19	25.73
Image, $k * \|F\|$	62.62	51.19	26.05

Table 1. Fine-tuned average results using the optimal hyperparameters as observed from above. The runs on the text dataset used $k = 1$ and all of the data. The runs on the image dataset used $k = 1$ and 1% of the data (420 samples).

4 Discussion

As is evident in Figure 1 and Figure 4, the overall accuracy of the model diminishes as k increases. It appears that $k = 1$ is the optimal selection, considering that the value of k does not affect the time the algorithm takes to train or predict. Figure 1 shows that the relationship may resemble exponential decay, where both the training and testing cases become perfectly fit, albeit less accurate than small values of k. Figure 4a corroborates this shape, but Figure 4b suggests a simple negative linear correlation.

Figures 1, 2, and 3 seem to suggest that the multivariate Bernoulli model and the multinomial model were equally as accurate and efficient in classifying the data, no matter the hyperparameters. This is substantiated by the fine-tuned results in Table 1, where the accuracies and time elapsed were almost identical. They are slightly overfitted, but not to a worrying degree.

Table 1 also seems to suggest that the $k + 1$ method is much more accurate and just as quick as the suggested $k * |F|$ approach for the image dataset. Both techniques overfit the same fraction of data by about 10%, but the former solution is about 25% more accurate. Thus, I believe it to be the better method given this specific dataset for data recognition.

Figure 5 seems to support this claim. As the fraction of data used increases, the $k * |F|$ accuracy seems to stay stagnant or even decrease. On the contrary, the $k + 1$ accuracy seems to

slowly increase and perhaps become less overfit. This further solidifies the confidence in the performance of $k + 1$.

The time the algorithm takes on the text dataset appears to follow a slight exponential growth as the fraction of data used increases, as seen in Figure 3. Figure 6 seems to suggest a linear relationship, but the range of fractions used in Figure 6 is much smaller than that of Figure 3, so this could be misleading. However, the times shown in Table 1 would align quite closely with the suggested linear model.

5 Related Work

As mentioned above, Naïve Bayes models are used throughout many industries and across many applications. The aforementioned Alzheimer's predictions are quite fascinating and suggest that Naïve Bayes may be a viable algorithm for genome-wide datasets [4]. This paper also demonstrates that simple accuracy is not the only way to measure a learner's performance. The area under the ROC curve (AUC) is great for tuning one's true positive and false positive rates.

While Naïve Bayes is typically used for discrete classification, it is fascinating to think about how it might perform with regression. This has been investigated by Frank et al. [1], but the results were just as dissatisfying as one would imagine. The underlying issue that Naïve Bayes has with continuous values is its assumption, which only works due to the nature of discrete values.

6 Conclusion

The implementation of the Naïve Bayes learner with Laplace smoothing in this paper offered some interesting conclusions. The accidental $k + 1$ technique seemed to prove much more effective than the preferred $k * |F|$ method. Also, the multivariate Bernoulli and multinomial event models did not have substantial differences in performance, but this could easily be the result of a lack of tested datasets. Maybe the text dataset did not feature many texts with duplicate words. This could also arise from the absence of any text cleaning. For the hyperparameters, more data is always preferable to less, just at the expense of time, and smaller values of k always seem preferable to larger values.

References

[1] Frank, E., Trigg, L., Holmes, G., & Witten, I. H. (2000). Naive Bayes for regression. *Machine Learning*, *41*(1), 5-25.

[2] MNIST. (2022, January 22). *Digit recognizer* [Learn computer vision fundamentals with the famous MNIST data]. Kaggle. https://www.kaggle.com/c/digit-recognizer

[3] UCI Machine Learning. (2016, December 2). *SMS spam collection dataset* (Version 1) [Collection of SMS messages tagged as spam or legitimate]. Kaggle. https://www.kaggle.com/uciml/sms-spam-collection-dataset

[4] Wei, W., Visweswaran, S., & Cooper, G. F. (2011). The application of naive Bayes model averaging to predict Alzheimer's disease from genome-wide data. *Journal of the American Medical Informatics Association*, *18*(4), 370-375.