Performance Comparison of Naive Bayes and Logistic Regression on MNIST Dataset

Naive Bayes

We implemented the Naive Bayes algorithm for classification on the MNIST dataset. The algorithm was trained and tested using the provided images and labels. We applied some preproccessing steps to make the data convenient for the algorithm to use.

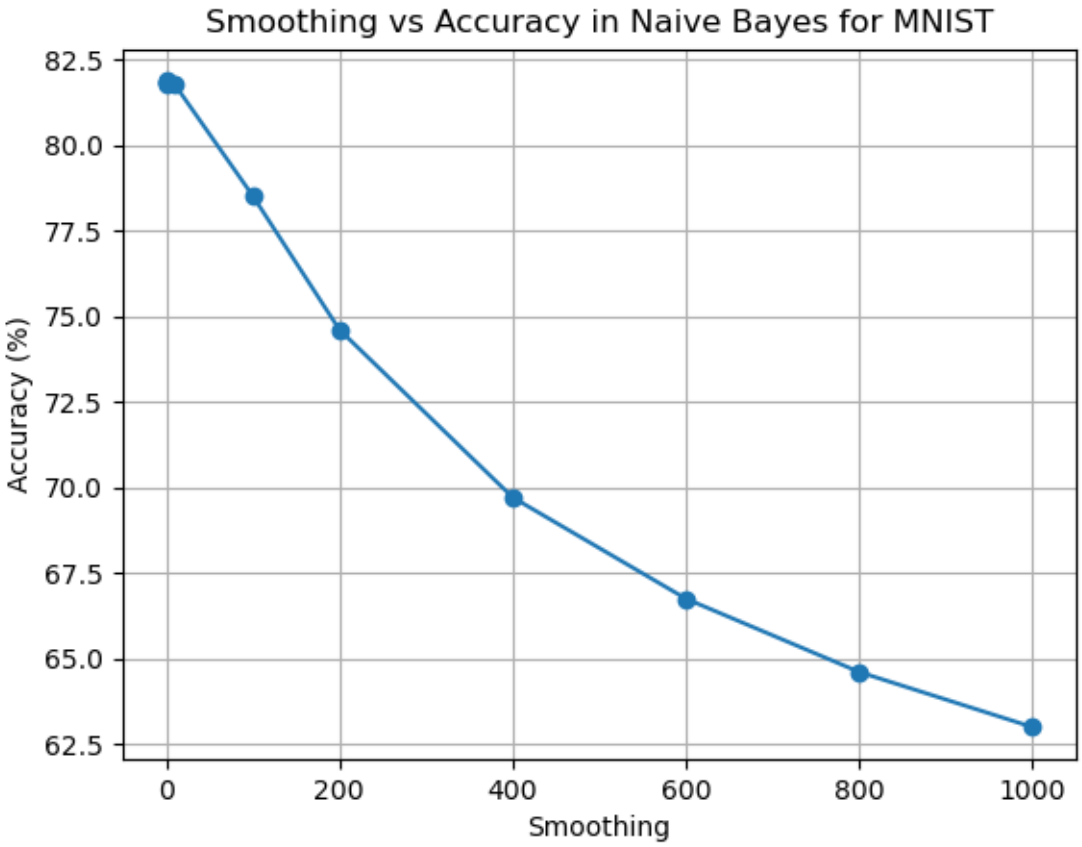
1. Implementation Details

The `NaiveBayes` class in `naive_bayes.py` contains a `predict` method, which takes the images, class probabilities, and pixel probabilities as input. It predicts the labels for the given images based on these probabilities.

For each image, we calculate the probability of each label by summing the logarithm of the pixel probabilities multiplied by the corresponding pixel values in the image, and adding the logarithm of the class probability. We select the label with the highest probability as the predicted label for the image.

2. Accuracy

After applying the Naive Bayes algorithm, we achieved an accuracy of 81.8% on the MNIST dataset. This indicates that the algorithm performs well in classifying the handwritten digits. Although, we observed that changing the laplace smoothing values can change the accuracy dramatically. Specifically, increasing the lapalace smoothing will decrease the accuracy as is shown in the following graph.



Logistic Regression

We also implemented logistic regression for classification on the same MNIST dataset. Logistic regression is a popular algorithm for binary classification, but it can be extended to handle multi-class classification as well.

1. Implementation Details

The `LogisticRegression` class in `logistic_regression.py` contains methods for initialization, softmax activation, and training the model.

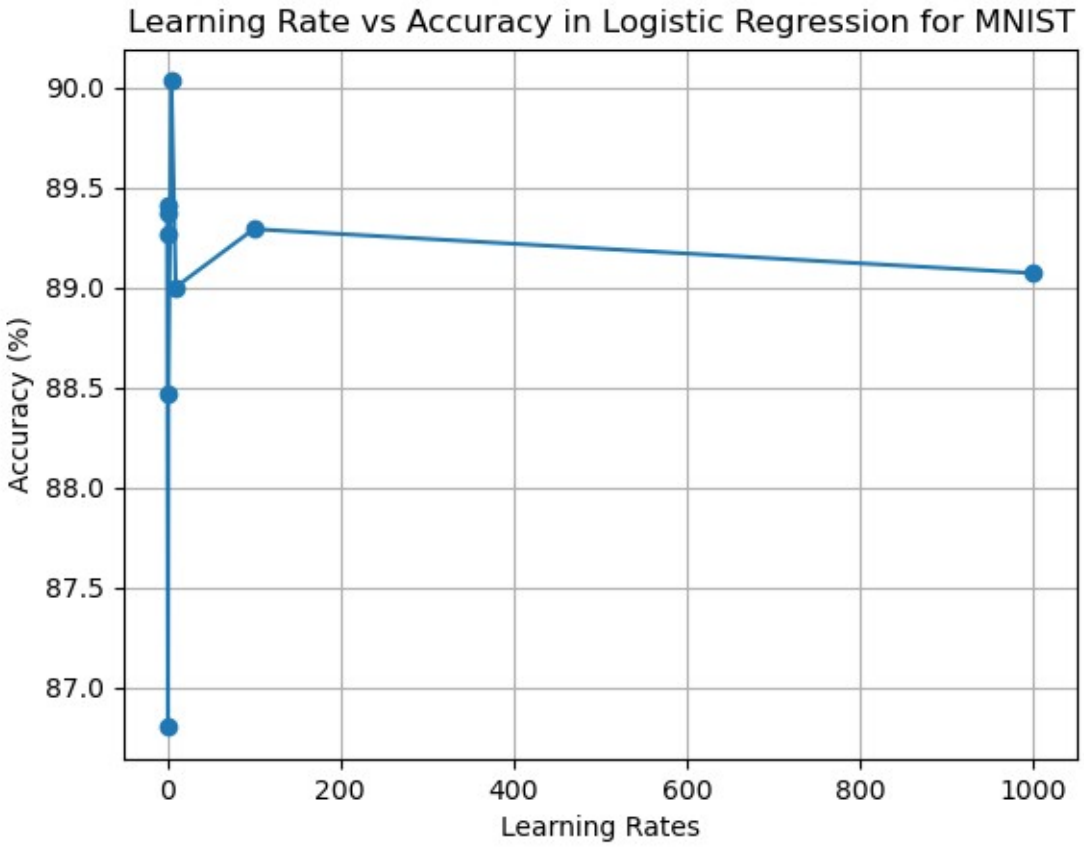
During initialization, the weights (`w`) and biases (`b`) are randomly initialized using a Gaussian distribution with mean 0 and standard deviation inversely proportional to the square root of the product of the number of inputs and classes.

The `softmax` function calculates the probabilities for each class using the softmax activation, ensuring that the probabilities sum up to 1.

The `train` method trains the logistic regression model using gradient descent. It iterates over multiple epochs and mini-batches of the training data. The gradients of the weights and biases are calculated using the cross-entropy loss and updated using the momentum-based gradient descent with a learning rate.

2. Accuracy

After training the logistic regression model, we achieved an accuracy of 90.7% on the MNIST dataset. The model demonstrates a significant improvement over Naive Bayes, indicating its effectiveness in handling the classification task.



Observations and Comparison

When comparing the performance of Naive Bayes and Logistic Regression on the MNIST dataset, we observe the following:

- Naive Bayes achieved an accuracy of 81.8%, while Logistic Regression achieved an accuracy of 90.7%. This indicates that Logistic Regression is more effective in classifying the MNIST images.
- The accuracy of both algorithms can be further changed and compared by tuning hyperparameters such as smoothing in Naive Bayes and learning rate in Logistic Regression.
- The Naive Bayes algorithm shows a smoother decrease in accuracy as the smoothing parameter increases, as observed in the graph generated by the `try_naive_bayes` function in `results.py`.
- Logistic Regression demonstrates a more stable accuracy trend with different learning rates, as shown in the graph generated by the `try_logistic` function in `results.py`.

In conclusion, the Logistic Regression algorithm outperforms Naive Bayes on the MNIST dataset, achieving a higher accuracy of 90.7% compared to 81.8%.