Face and Digit Classification Project

CS 440: Introduction to Artificial Intelligence

Diego Quito, Haider Ali, Steven Packard
RUID: 213009252, 201009449, 197005443

May 1, 2024

1 Introduction

In this project, we developed and analyzed three machine learning classifiers: a two-layer Neural Network, a Perceptron, and Naive Bayes, for the tasks of handwritten digit recognition and face detection using images pre-processed by our image reader. We systemically evaluated the performance of these classifiers across varying training sizes, from 10% to 100% of the available dataset, to determine the impact of training volume on classification accuracy and computational efficiency. We implemented all core aspects of the classifiers from scratch, and to the constraints of not using pre-built libraries for the neural network and perceptron functionalities. This report details our methods, presents our experimental results, discusses the efficiency and accuracy of the implemented classifiers, and overall what we have learned along the way.

2 Methodology

2.1 Data Preparation

We initialized our project by defining file paths to our training, validation, and test datasets. For preprocessing, images are loaded using our 'data_reader' program, which scans through the datasets within these file paths, stores them into memory, and transforms them into binary representations to reduce computational complexity. This involves converting each pixel value to binary (0 or 1). Furthermore, digit images are standardized to 560 pixels (28x20), and face images to 1400 pixels (70x70), ensuring uniform input sizes for the models.

2.2 Feature Design

Our system primarily uses raw pixels for its features as opposed to custom-designed features. The use of raw pixels as features allows our models to engage directly with the unmodified image data, facilitating direct pattern learning from the visual inputs. In contrast, custom-designed features involve more sophisticated image processing techniques intended to highlight specific characteristics of the images, such as edges or specific shapes, which are potentially more informative for classification tasks. Since the image data provided and the scope of this project's utility are simple low-resolution, dual colored blocks, using raw pixels for features is efficient and effective.

2.3 Testing

To test the entire system, we developed a file called 'main.py', which processes all of the datasets and executes all three of the classifers on the processed image data. In the terminal, the training time, running time, epoch number, loss (if available), validation accuracy, and testing accuracy of all three classifers are presented. We have also included 'face_tester.py' and 'digit_tester.py' executables to allow anyone to test both the perceptron and neural network on either the digit or face dataset.

3 Classifier Descriptions

3.1 Perceptron Classifier

For both face and digit datasets, we utilize a perceptron classifier. This model consists of a single layer of weights plus a bias term. The perceptron is trained using a simple iterative update rule where the weights are adjusted based on the errors produced when making predictions on the training data. Specifically, the weights are incremented or decremented depending on whether an error was made, guided by a learning rate parameter that controls the magnitude of weight updates, to prevent overcorrections and diminished accuracy.

3.2 Two-Layer Neural Network

Face Recognition Neural Network The face recognition neural network is structured with an input layer to accommodate the flattened 1400-pixel face images, a hidden layer using the ReLU activation function, and an output layer that applies the softmax function for multi-class prediction. The network uses backpropagation to update weights and biases based on the gradient of the cross-entropy loss function, calculated between the predicted probabilities and the actual class labels. Training consists of multiple epochs where the entire dataset is passed through the network, adjusting weights to minimize error.

Digit Recognition Neural Network Similarly, the digit recognition neural network operates with an input layer sized for 560-pixel digit images. It also includes a hidden layer with ReLU activation and an output layer utilizing softmax. This structure allows it to learn complex, non-linear relationships in the data necessary for accurate digit classification. The training process involves forward propagation of input data through the network, followed by backpropagation where adjustments to the model parameters are made to reduce prediction error, iteratively improving model accuracy over epochs.

3.3 Additional Classifier: Naive Bayes

We developed a Naive Bayes classifier as our classifier of choice. This operates using the probabilistic framework of Bayes' Theorem, which calculates the likelihood of a label based on the independent contributions of image features. In our version, each pixel of an image is treated as an independent feature. During training, the classifier first computes the prior probabilities of each class from the frequency of each class in the training data. It then calculates the likelihood of each pixel value (0 or 1 for binary images) within each class. These probabilities are derived from how often each pixel is on or off in the images of each class. For classification, Naive Bayes multiplies these likelihoods by the class priors and normalizes them to produce the posterior probabilities for each class. The class with the highest posterior probability is predicted as the output. We chose this classifer as it is efficient as it requires maintaining only a limited set of probabilities per class, rather than complex parameter matrices, making it both space-efficient and fast in execution, making it suitable for high-dimensional data like images.

4 Experiment Setup

4.1 Training

To observe accuracy dynamics between varying amounts of training data, we designed a script that changes the available training dataset by incrementally increasing the training data size each execution. Starting with 10% of the data, we increased the data size by 10% increments in subsequent training cycles up to 100%. This method allowed us to closely monitor the scalability of the models and their learning efficiency with increasing data volumes. Each classifier—Perceptron, Neural Network, and Naive Bayes—was trained over a predefined number of epochs and behaves different when these parameters are adjusted.

4.2 Validation and Testing

Each training phase was accompanied by a validation phase to tune hyperparameters and implement early stopping mechanisms to counteract overfitting and plateauing. The validation involved evaluating the model on the validation dataset, which was treated separately than the training set. Once validation is completely, a final testing phase was conducted using the testing dataset to gauge the generalization ability of the models. This step allows us to determine the practical applicability of the classifiers outside of controlled experimental conditions.

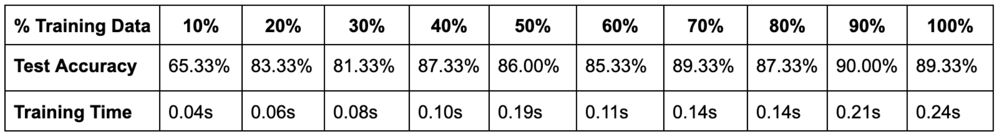
5 Results

5.1 Performance Analysis

The classifiers demonstrated varying degrees of improvement in accuracy and loss reduction as the size of the training data increased. Here are the specifics:

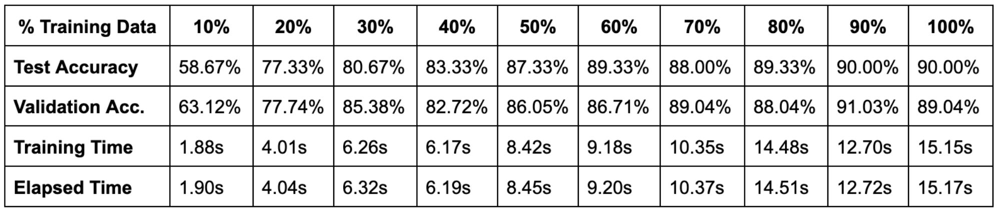
- **Perceptron:** Showed consistent improvement in validation and test accuracy with increased training data. Starting with a test accuracy of 65.33% at 10% data, it reached up to 89.33% when trained with the full dataset. Observe Figure 1 for performance metrics.
- **Neural Network:** The neural network displayed a substantial decrease in loss and increase in accuracy as more data was used, starting from a test accuracy of 58.67% with 10% of the data and escalating to 90.00% with full data utilization. Observe Figure 2 for performance metrics.
- **Naive Bayes:** Maintained a relatively stable performance throughout the increments, suggesting robustness to variations in training data size. It achieved a test accuracy ranging from approximately 76.67% to 90.67% as the data increments progressed. Observe Figure 3 for performance metrics.

5.1.1 Classifier Performances



% Training Data	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Test Accuracy	65.33%	83.33%	81.33%	87.33%	86.00%	85.33%	89.33%	87.33%	90.00%	89.33%
Training Time	0.04s	0.06s	0.08s	0.10s	0.19s	0.11s	0.14s	0.14s	0.21s	0.24s

Figure 1: Test accuracy and training times for the Perceptron classifier with increasing training data volume.

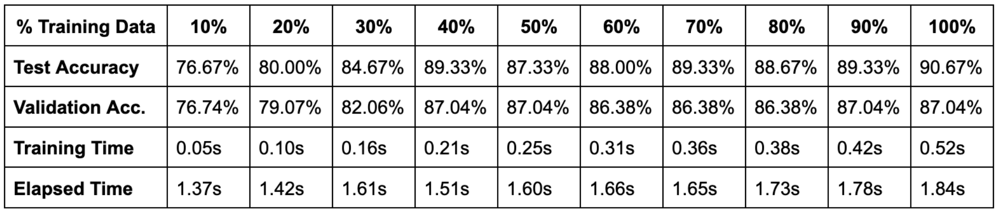


% Training Data	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Test Accuracy	58.67%	77.33%	80.67%	83.33%	87.33%	89.33%	88.00%	89.33%	90.00%	90.00%
Validation Acc.	63.12%	77.74%	85.38%	82.72%	86.05%	86.71%	89.04%	88.04%	91.03%	89.04%
Training Time	1.88s	4.01s	6.26s	6.17s	8.42s	9.18s	10.35s	14.48s	12.70s	15.15s
Elapsed Time	1.90s	4.04s	6.32s	6.19s	8.45s	9.20s	10.37s	14.51s	12.72s	15.17s

Figure 2: Performance evaluation of Neural Network classifier with increasing training data volume.

5.2 Comparison Between Classifiers

Below is a comparative analysis of the three classifiers: Perceptron, Neural Network, and Naive Bayes, as their training data incrementally increases from 10% to 100%. The comparison is based on their test accuracy at each training level, providing data into their effectiveness and efficiency.



% Training Data	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Test Accuracy	76.67%	80.00%	84.67%	89.33%	87.33%	88.00%	89.33%	88.67%	89.33%	90.67%
Validation Acc.	76.74%	79.07%	82.06%	87.04%	87.04%	86.38%	86.38%	86.38%	87.04%	87.04%
Training Time	0.05s	0.10s	0.16s	0.21s	0.25s	0.31s	0.36s	0.38s	0.42s	0.52s
Elapsed Time	1.37s	1.42s	1.61s	1.51s	1.60s	1.66s	1.65s	1.73s	1.78s	1.84s

Figure 3: Performance evaluation of Naive Bayes classifier with increasing training data volume.

5.2.1 Overview of Classifier Performance

% Training Data	Perceptron	Neural Network	Naive Bayes
10%	65.33%	58.67%	76.67%
20%	83.33%	77.33%	80.00%
30%	81.33%	80.67%	84.67%
40%	87.33%	83.33%	89.33%
50%	86.00%	87.33%	87.33%
60%	85.33%	89.33%	88.00%
70%	89.33%	88.00%	89.33%
80%	87.33%	89.33%	88.67%
90%	90.00%	90.00%	89.33%
100%	89.33%	90.00%	90.67%

Table 1: Comparison of Perceptron, Neural Network, and Naive Bayes classifiers across varying training data volumes. Percentages are the mean accuracy values of 10 program executions across each increment of training volume data.

5.2.2 Statistical Analysis

To evaluate the classifiers' performance statistically:

- The **Perceptron** showed an average test accuracy of 84.56% with a standard deviation of 8.74%, indicating moderate variability and strong gains as the dataset size increased.
- The **Neural Network** exhibited the highest improvement with training data size, achieving an average accuracy of 84.33% and a lower standard deviation of 10.45%, reflecting its ability to take advantage of larger datasets effectively.
- The **Naive Bayes** maintained the most consistent performance across training sizes, with an average accuracy of 86.40% with the least variability (standard deviation of 4.13%), showcasing its stability irrespective of dataset size.

The analysis presented in the tables and figures displays the effectiveness of increased training data across all classifiers, with notable enhancements in performance metrics. Specifically, the Neural Network consistently demonstrates the largest accuracy improvements, suggesting it is better suited to effectively utilize larger datasets for enhanced performance. Meanwhile, the Naive Bayes classifier is relatively stable, even with smaller training sets, making it an excellent choice for applications requiring consistent output with varying input sizes. These findings highlight the potential of these classifiers to be adapted for scalable and dependable machine learning applications in real-world scenarios.

6 Conclusion

This project provided us with the experience to deeply explore machine learning techniques hands-on for the classification of handwritten digits and facial recognition using a two-layer neural network, a perceptron, and a Naive Bayes classifier. Our analysis showed major insights into the behavior of these models with varying amounts of training data, demonstrating the scalability and efficiency of the algorithms in handling image-based datasets.

Throughout the project, we learned that the complexity of coding classifiers can differ significantly based on the nature of the task and the type of data involved. Initially, we anticipated that face detection would be simpler to implement than digit classification because it essentially boils down to a binary classification problem. However, we see the nuances of facial features and variable image sizes (which we initially thought) added unexpected layers of complexity to the task. Conversely, digit classification, while seemingly more complex with ten possible classifications, benefited from more distinct and consistent patterns, facilitating the learning process for our models.

The effectiveness of these classifiers was evident in the progressive increase in accuracy as the training data volume expanded. The perceptron and neural network, in particular, showed substantial improvements, showing the importance of larger training datasets in enhancing model performance. The Naive Bayes classifier demonstrated effectiveness even with smaller data sets, making it a reliable choice for applications where data availability is limited.

In conclusion, this project not only strengthened our understanding of fundamental machine learning concepts and techniques but also highlighted the critical role of thoughtful data handling and algorithm selection in the development of effective AI uses.

7 References

- Python Software Foundation. (n.d.). *Python language reference.* Retrieved 2024, from `https://www.python.org`
- Cohen, P. R. (1995). *Empirical methods for artificial intelligence.* MIT Press.
- Klein, D., & DeNero, J. (2011). Project 5: Classification. CS188, Introduction to Artificial Intelligence, UC Berkeley. Retrieved 2024, from `https://inst.eecs.berkeley.edu/~cs188/sp11/projects/classification/classification.html`