Sean Deery HW 6 & 7

# Introduction

Computer image recognition is a powerful and important field of machine learning and artificial intelligence. It allows computers to process pictures and videos in a similar way to the human eye. This can enable people and organizations to automate time-consuming tasks that were previously only able to be done by hand. Today, computer image recognition is used in self-driving cars, x-ray analysis, manufacturing defect inspection, reading text and barcodes, and many more applications.

One of the largest examples of a use case is the United States Postal Office. According to https://facts.usps.com/one-day, the USPS processed 421.4 Million mail pieces daily on average in 2022. This is a massive operation and using image recognition is a way for them to quickly scan a package for the address information as well as other packaging needs and get it to its destination as fast as possible.

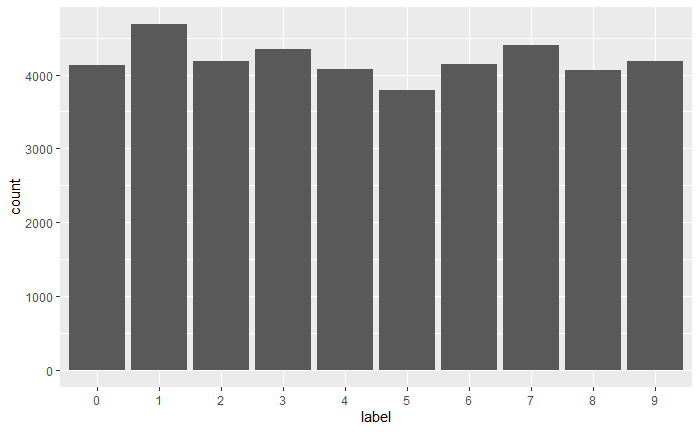
Pattern recognition is instinctual for humans, whereas computers need to be trained on labeled data. This analysis seeks to train a model on labeled pictures of handwritten digits to predict what digit unlabeled pictures of handwritten digits are.

# Analysis and Models

## About the Data

The MNIST dataset comes from the Kaggle Digit Recognizer competition. It contains 42,000 rows. Each row of the training data contains a label and 784 pixels representing a 28-pixel by 28-pixel image. Each pixel value is an integer between 0 and 255, inclusive. The images are handwritten digits from 0 to 9, and the goal is to create a model that can accurately label new handwritten digits.

The bar plot below shows the training data has a relatively uniform distribution, where each digit has roughly the same amount of examples. This means that when evaluating models, accuracy should be a good metric to compare models.



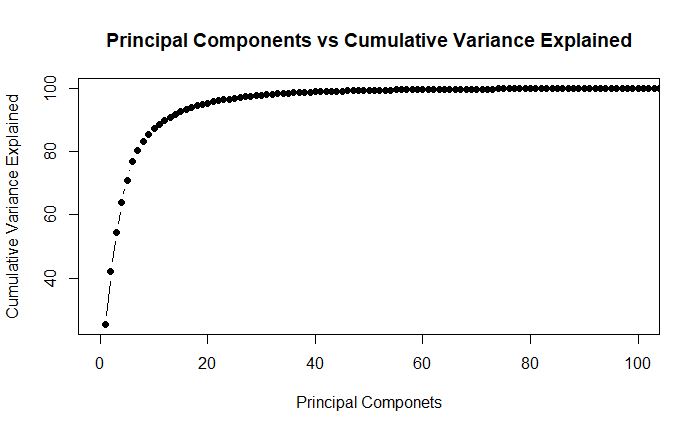
### Principle Component Analysis

Data sets that have a high number of attributes, like the MNIST dataset, can experience a phenomenon called the curse of dimensionality. For classification, the curse of dimensionality can mean that there are not enough data objects to allow the creation of a model that reliably assigns a class to all possible objects. To avoid the issues that come with the curse of dimensionality, there are several methods to reduce the number of attributes.

There are a variety of benefits of dimensionality reduction. Many data mining algorithms work better if the number of attributes is lower because it can eliminate irrelevant features and reduce noise. Reducing the number of attributes can also lead to a more understandable model and may allow for the data to be more easily visualized. Finally, reducing the dimensions reduces the time and memory required by the data mining algorithm.

Principle Component Analysis (PCA) is a method of dimensionality reduction based on linear algebra. It extracts important information from the data to express the information as a set of summary indices called principle components. The principle components are linear combinations of the original attributes, are orthogonal to each other, and capture the maximum amount of variation in the data.

To find the best number of principle components to use for the MNIST dataset, the percent of variance explained is calculated for each number of principle components possible. The plot and table below show the cumulative variance explained for 1 principle component up to 100 principal components. It shows that the increase in variability explained seems to flatten out between 20 and 40 principle components. Therefore, 25 principal components were chosen to complete the analysis.



A screenshot of a computer

Description automatically generated with medium confidence

## Models

### Decision Tree

Decision Trees are a classification technique that classifies new data by asking a series of questions about the attributes of the new data. Each answer leads to a follow-up question until a conclusion is reached about the class label of the record. The questions can be organized into a decision tree containing a hierarchical structure of nodes and edges. A root node has no incoming edges and zero or more outgoing edges. Internal nodes have exactly one incoming edge and two or more outgoing edges. A leaf node has exactly one incoming edge and no outgoing edges. The leaf node is where the classification is completed.

The order of the questions, or the order of attributes to split on, is decided based on the information gain and gain ratio. Information gain is a statistical measure that measures how well a given attribute separates the training examples according to their target classification. The gain ratio is the information gain divided by “split info”, which refers to a penalty for a large number of splits.

The strengths of the decision tree are that they are fast in prediction, they have interpretable patterns, and they are robust to noise in the data. The weaknesses of a decision tree are that they tend to overfit, they are error-prone when there are too many classes, and they are computationally expensive in training. The problem with overfitting arises because there may not be enough training data to fully represent all possible cases or the decision tree may be too detailed a fit to the training data. To account for overfitting, the decision tree model’s complexity can be controlled by pruning. Prepruning is done by increasing the minimum information gain threshold, at which point the nodes do not split any further. Postpruning is removing branches from a decision tree that has already been trained by increasing the minimum number of instances threshold.

### Naïve Bayes

The Naïve Bayes classifier works by computing the posterior probability for each digit given a set of pixels, and then choosing the digit with the highest probability. This model is naïve in that it assumes each of the pixel values are independent of each other. This is a practical method to use as it does not require computing the class-conditional probability for every combination of pixels.

A problem that can arise with Naïve Bayes is when one of the conditional probabilities is zero, then the entire product becomes zero as well. The solution to a zero probability is called smoothing, which replaces the zero with a very small estimate, meaning that it still occurs in the real world but is so rare that the training data did not include any of them. Since all probabilities need to add up to one, the other nonzero probabilities are reduced.

Another problem with Naïve Bayes is that the conditional probabilities need a way to handle continuous values. One solution is to discretize the attribute into bins and the other solution is to use probability density estimation. Probability density estimation assumes the attribute follows a normal distribution and uses data to estimate the parameters of the distribution. Once the probability distribution is known, the probability density function can estimate the conditional probability.

### kNN

K Nearest Neighbors (kNN) is an instance-based learning method that stores the training examples without doing any calculations during the training process, and classification and prediction are delayed until new examples are given. The classification process involves taking a test sample and comparing the similarity between the test example and all the training examples and categorizing it based on the nearest training examples. The number of training examples to include is set by the parameter k. If k is too small, the model will be more sensitive to noise points. If k is too large, the neighborhood may include too many points from other classes. The benefit of kNN is that it makes no assumptions about the data, like how Naïve Bayes assumes the variables are independent of each other, and it works well when the decision function to be learned is very complex. A disadvantage of using kNN is that it is sensitive to noisy trained data and has a high computational cost for predicting.

### SVM

Support vector machines (SVM) are classifiers that find a linear hyperplane that separates the data into classes. The algorithm to find the best hyperplane maximizes the margin between the two classes by choosing specific points at the decision boundary as support vectors. If there are no perfect linear boundaries between two classes, slack variables can be introduced to pay a cost for each misclassified example. A problem that can arise with SVM is overfitting. To avoid overfitting, regularization can be used to tune the model with a ‘cost for misclassification’ parameter. When the cost is high, the algorithm, tries to build a model with the fewest training errors, resulting in a narrow margin and a high chance of overfitting. When the cost is small, the margin is wider and more robust. If the cost is too small, it will not respect the data at all and will not be an accurate predictor.

Some data may not be linearly separable, but they are separable in a higher dimensional space. SVM can map the data to a higher dimensional space to accurately classify this type of data using a kernel parameter. Typical kernel functions include linear, polynomial, Gaussian radial basis, and sigmoid. When multiple classes are introduced, SVM creates multiple classifiers and combines them for predictions. In this scenario, the algorithm uses either a class-vs-class or a class-vs-all classes strategy and picks the most confident prediction for the result. The strengths of the SVM include having a high tolerance for noisy data, being flexible in data representation (continuous and discrete values), having a probabilistic prediction result, being successful on large problems, and being successful on a variety of real-world data. Weaknesses of the SVM include the need for more parameters and interpretability when the model is generated by a nonlinear kernel.

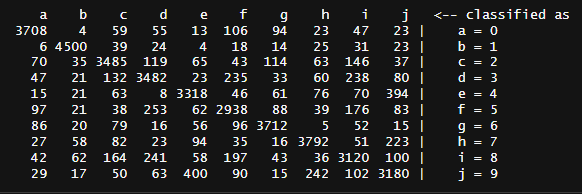
### Random Forest

Random forests are a class of ensemble methods specifically designed for decision tree classifiers. The original training data are randomly selected into multiple subsets. Decision trees are trained on each of these subsets, and then the results are combined to form the prediction model. The random selections of data are generated from a fixed probability distribution. This combination of multiple classifiers can make better predictions and achieve better performance than any single model. One problem Random Forest can run into is overfitting if there are too many decision trees. The way to avoid this problem is to plot the error rate as the number of trees gets larger to identify when the error rate seems to flatten out as more trees are added.

# Results

### Decision Tree

The decision tree model was trained with different pruning confidence and minimum instance thresholds and was then evaluated with 3-fold cross-validation. Running the algorithm with a minimum instance threshold of 1, the highest accuracy of 83.81% was achieved by setting the confidence threshold to 0.1. Running the algorithm again with a minimum instance threshold of 3, the highest accuracy of 83.97% was achieved by setting the confidence threshold to 0.1. Running the algorithm again with a minimum instance threshold of 4, the highest accuracy of 83.89% was achieved by setting the confidence threshold to 0.1. Since the accuracy of the model starts to go down when the minimum instance threshold is increased to 4, the final decision tree model that is used has a minimum instance threshold of 3, a confidence threshold of 0.1, and a 3-fold cross-validation accuracy of 83.97%. The confusion matrix below shows the results of the final decision tree model. It shows the algorithm had a particularly hard time correctly classifying fours, fives, and eights.



### Naïve Bayes

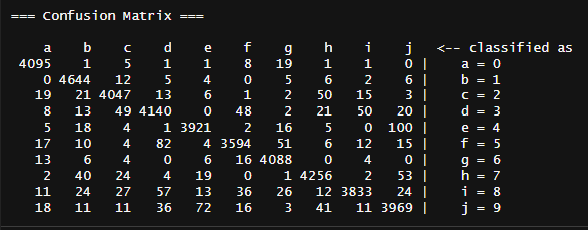
The Naïve Bayes model was trained with different parameters for how to handle the numeric attributes and was then evaluated with 3-fold cross-validation. Running the algorithm using a kernel density estimator rather than a normal distribution for numeric attributes resulted in an accuracy of 82.36%. Running the algorithm using supervised discretization to process numeric attributes resulted in an accuracy of 85.14%. Running the algorithm with the default parameters using a normal distribution for numeric attributes resulted in an accuracy of 85.39%. This is a huge improvement in accuracy from the best decision tree, and it took about half of the time to compute. The confusion matrix below shows the algorithm still had a harder time correctly classifying fours, fives, and eights, but the accuracy is much better across all the digits.

A screenshot of a computer screen

Description automatically generated with medium confidence

### kNN

The kNN model was trained with different k values to find the most accurate model and was then evaluated with 3-fold cross-validation. The k value of 3 resulted in the highest accuracy of 96.64%. Increasing the k value to 4 dropped the accuracy to 96.61% and decreasing it to 1 resulted in an accuracy of 96.53%.



### SVM

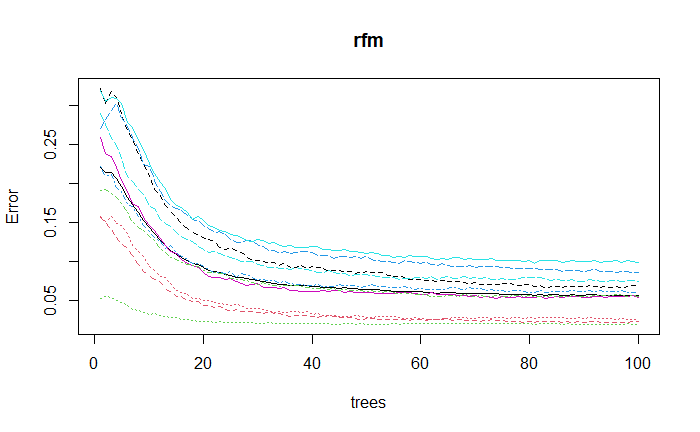
The SVM model was trained with different kernels and ‘cost for misclassification’ parameters and was then evaluated with 3-fold cross-validation. The model with the polynomial kernel and a cost parameter set to 1 had the highest accuracy of 90.96%. The worst result was when the cost was set to 0.1, which had an accuracy of 89.73%.

A screenshot of a computer

Description automatically generated

### Random Forest

The Random Forest was trained using different numbers of trees. The line plot below shows that the error rate seems to flatten out at around 30 trees. Running the algorithm with 30 trees resulted in an error rate of 7.65% and an accuracy of 92.35%.



A screenshot of a computer screen

Description automatically generated with low confidence

# Conclusions

The MNIST dataset contains images of handwritten digits along with labels to train machine learning models with. The goal is to create an image recognition model based on the pixels in each image and then be able to accurately predict the digit in unlabeled images. First, the pixels were reduced to 25 principal components which explained 96.82% of the variation in the data. Then, in training and tuning the models, the accuracy of each model was measured using 3-fold cross-validation. The kNN model resulted in the highest accuracy of 96.53%, followed by Random Forest, SVM, Naïve Bayes, and then the Decision Tree at the tail end. Submitting the prediction to Kaggle resulted in similar scores with kNN achieving an accuracy of 97.03%. One thing to note is that the test accuracy is above the training accuracy, suggesting the models were tuned in a way that did not overfit the data.

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy (3-fold CV)** | **Kaggle Accuracy** |
| kNN | 96.53% | 97.03% |
| Random Forest | 92.35% | 94.10% |
| SVM | 90.96% | 91.08% |
| Naïve Bayes | 85.39% | 85.59% |
| Decision Tree | 83.97% | N/A |

The kNN model likely did well because of how the pixel values were changed from 0-255 to 0-1 and then combined into 25 principal components during preprocessing. Once the data was in a format where all the data points were proportionally spaced out, the kNN was able to use the 3 data points closest to each of the test values to predict the correct label more than any of the other models. Taking a closer look at the confusion matrix for the kNN cross-validation, it seemed to have the hardest time correctly distinguishing between 3, 5, and 8. It also seemed to confuse 4 and 9 frequently. These are numbers that share some patterns in how they are handwritten, so it makes sense that these digits would be mixed up the most.

The Random Forest outperformed the Decision Tree and was second in terms of accuracy. This supports the idea that breaking the data up into multiple decision trees and aggregating the result improved the model’s performance. This indicates that there are some images in the data that are not good representations of the population and that the Random Forest was able to omit their influence on the predictions.

The Support Vector Machine came in third in terms of accuracy. The kernel chosen was the polynomial kernel, but perhaps there was a better option out there or preprocessing steps that would have been better to use for the SVM. SVMs are known to be good models to use for image classification, but surprisingly, it was not able to predict the labels more accurately for the handwritten digits.

Lastly, the Naïve Bayes model may not have done so well because it assumes that each of the variables are independent of each other. This is not necessarily the case with handwriting, as the lines move through the pixels in a connected way. This could be why the Naïve Bayes predictor is not the best model to choose for this type of image recognition.