MNIST – digits data analysis

김병헌 (Automotive Engineering 2018015878)

Department of Automotive Engineering, Hanyang Universit), Seoul 110-04763, Korea

Abstract :

This analysis focused on examining the MNIST digits dataset to understand its characteristics. The dataset consists of 1797 handwritten digit images, each composed of a total of 64 pixels in an 8x8 grid. The images are grayscale, and the pixel intensity ranges from 0 to 16.

To ensure randomness in model training, the order of the dataset, initially sorted from 0 to 9, was shuffled. Due to the minimal variance in the number of data points across each class, there was no significant issue of class imbalance. The problem was approached as a multi-class classification task, and the algorithms covered during artificial intelligence classes, namely Logistic Regression, SVM, FLDA, and MLP, were employed. After comparing and analyzing the performance of each model, the SVM utilizing the rbf kernel trick demonstrated the highest accuracy. However, considering the marginal differences in performance, it is important to note that model rankings could change with parameter adjustments.

If a single model have to be selected for problem-solving with this data, I would choose the SVM(poly)model due to its relatively faster training and simpler structure compared to the others, despite similar performance. But may consider to use MLP or CNN with larger data later.

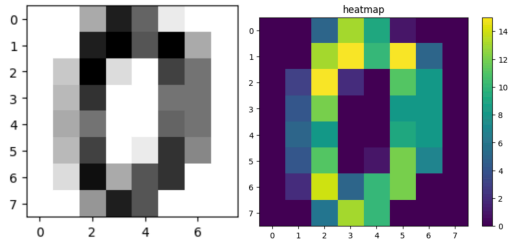
Key words : MNIST, digits, Logistic regression, FLDA, SVM, MLP

1. Problem setting

After loading the data, I proceeded to inspect the dataset.

```
Number of samples: 1797, Number of features: 64
Number of classes: 10
```

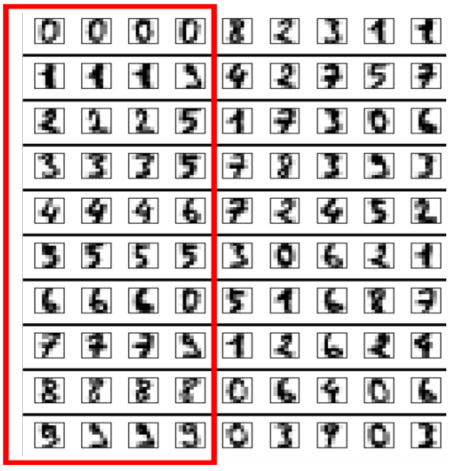
The loaded dataset consists of 1797 image data. Each image is structured in an 8x8 format, totaling 64 pixels. There are 10 classes in the dataset, representing digits from 0 to 9.



The above figure visualizes the first data from the loaded dataset in the form of a heatmap. Each data point represents handwritten digit data, and the presence of shading indicates the variations in the handwriting.

```
[ 0.,  0.,  5., 11.,  4.,  1.,  0.,  0.],
[ 0.,  0., 15., 16., 16., 11.,  0.,  0.],
[ 0.,  2., 16.,  9.,  2., 12.,  4.,  0.],
[ 0.,  6., 13.,  0.,  0.,  6.,  6.,  0.],
[ 0.,  3., 13.,  0.,  0.,  5.,  9.,  0.],
[ 0.,  3., 16.,  0.,  0.,  6.,  8.,  0.],
[ 0.,  0., 13., 12.,  8., 16.,  7.,  0.],
[ 0.,  0.,  4., 13., 12., 10.,  0.,  0.]]
```

The intensity levels range from 0 to 16, where 0 represents white and closer to 16 indicates black in color.



This is a subset of the entire dataset, consisting of the first 90 entries. It is arranged in a column-wise order, starting from the 1st column and proceeding downward.

It is evident that the data at the beginning is systematically ordered from 0 to 9. Therefore, for future modeling, the plan is to shuffle the data. Additionally, this observation indicates that data of the same class differ from each other since the digits are handwritten. Consequently, it can be observed that MNIST digits data is nonlinear, given the variations in shapes, curves, shadows, etc., for each digit.

```
0 : 178
1 : 182
2 : 177
3 : 183
4 : 181
5 : 182
6 : 181
7 : 179
8 : 174
9 : 180
```

This figure represents the number of data points for each class among the total 1797 samples. It is evident that the data is evenly distributed across classes, indicating that there is no need to separately consider class imbalance issues.

The results of the data analysis indicate that to create a predictive model using this dataset, a classification model should be employed. With a total of 10 categories ranging from 0 to 9, it corresponds to a multi-class classification problem. Based on the algorithms learned during the class, modeling experiments were conducted.

2. Experiments

Based on the algorithms learned during the class, modeling experiments were conducted.

2.1 Linear Regression

Initially, I considered the first model we learned, the linear regression technique. However, linear regression is suitable for regression models, so it was excluded from this experiment.

2.2 Logistic Regression

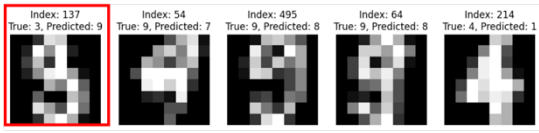
Unlike linear regression, which is used for regression problems, logistic regression can be applied to classification problems. Logistic regression is primarily used for binary classification problems because it inherently applies the sigmoid function to produce probabilities between 0 and 1. Since the problem at hand is a multi-class classification problem, it takes a different

approach than determining positives/negatives based on a threshold. In the case of Logistic Regression provided by scikit-learn, it typically employs the 'One vs Rest' method to solve classification problems. This method considers one class as positive and treats all other classes as negative, classifying based on the class with the highest probability among all binary classifiers.

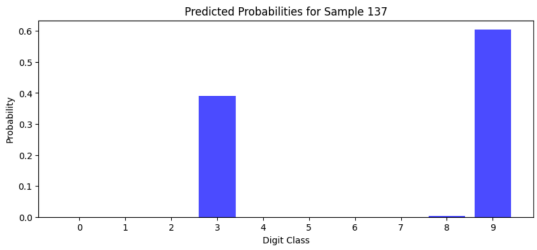
In the Logistic Regression applied in this experiment, the multi_class parameter was set to 'multinomial' to use the softmax activation function suitable for multi-class classification. This is because softmax considers interactions between various classes to normalize probabilities. Therefore, this model considers all possible class pairs using the 'One vs One' method, deciding the most positively classified class through a voting mechanism.

```
Training Accuracy: 0.9992044550517104
Test Accuracy: 0.9648148148148148
```

The training results of the model show a high accuracy on the test data, approximately 96%.



The figure above visualizes samples that the model predicted incorrectly on the test data when Logistic Regression was applied. With an accuracy of 96%, it's evident that the digits predicted by the model and the actual digits share similar shapes.



The figure above represents the model's probabilities for classifying 137th sample data point into different classes. As depicted in the graph, even though the actual value is 3, the model predominantly predicted it as 9, considering the probability of being 9 as the highest.

2.3 Fisher's linear discriminant analysis

The FLDA (Fisher's Linear Discriminant Analysis) algorithm is a statistical analysis method designed to maximize the between-class variance and minimize the within-class variance. To achieve this, the algorithm calculates matrices and computes eigenvalues and eigenvectors to maximize the ratio of between-class variance to within-class variance. The calculated eigenvalues and eigenvectors are then sorted in descending order based on their magnitudes. Subsequently, to map the data into a lower-dimensional space than the original feature space, the algorithm selects the top k eigenvectors with the largest eigenvalues and maps the data into a k-dimensional space. This process retains the directions with high variance from the original high-dimensional space while reducing the dimensionality, thus saving computational and storage space while preserving important information.

This algorithm is based on the assumption that the data follows a normal distribution, and the covariance matrices of each class are identical. If the actual data does not conform to these assumptions, there is a significant risk of degraded classification performance. Additionally, FLDA relies on statistical properties during the process of finding the direction that optimizes variance, making it less effective in datasets with many outliers. In other words, it is sensitive to outliers. Furthermore, in the matrix calculation process, if the number of variables exceeds the number of data samples, the computation of the inverse of the covariance matrix can become challenging or unstable. Since FLDA ultimately creates a linear decision boundary, it may exhibit lower performance in cases where the data has a nonlinear structure. Here are the performance results of the FLDA model.

```
Training Accuracy: 0.9665871121718377
Test Accuracy: 0.9592592592592593
```

As mentioned earlier, considering the diverse and complex shapes, curves, and shadows for each digit in the MNIST digits dataset, indicating that the dataset is nonlinear, we proceeded to analyze it using models better suited for

Nonlinear Discriminant Analysis. Specifically, we utilized models with kernel tricks, such as SVM, and MLP, to demonstrate their suitability and excellent performance.

2.4 SVM

Support Vector Machine (SVM) is a machine learning algorithm primarily applied to classification problems. One of the strengths of this algorithm lies in its excellent generalization performance. This is because SVM aims to maximize the margin, the distance between the decision boundary and the closest data points from different classes. SVM can also take advantage of the kernel trick in high-dimensional spaces, allowing it to effectively separate data. Through the kernel trick, SVM can map data into higher-dimensional spaces, enabling the discovery of complex nonlinear decision boundaries. In this analysis, we compared linear, polynomial, and RBF kernels. SVM can operate effectively on relatively small datasets and often yields superior results in high-dimensional data compared to other algorithms.

However, SVM has its drawbacks. The learning speed can significantly decrease when dealing with very large datasets or a high number of features. Additionally, when using the kernel trick to map data into high-dimensional spaces, interpreting the model can become challenging.

The accuracy results when applying SVM with a linear kernel (c=0.001) are presented.

```
Training Accuracy: 0.9880668257756563
Test Accuracy: 0.9796296296296296
```

Because the regularization parameter C was set very low, the model was trained to prefer a hard margin, making it less sensitive to outliers. The accuracy is approximately 98%, indicating that the model found features well for classifying classes even without mapping the data into high dimensions due to the relatively small dataset size.

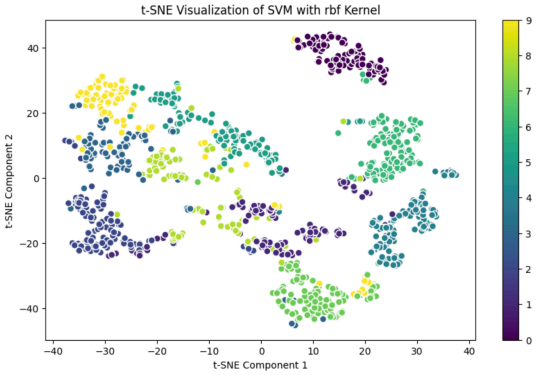
Next, the accuracy results when applying SVM with a polynomial kernel (c=0.00001, gamma=auto) are presented.

```
Training Accuracy: 0.9976133651551312
Test Accuracy: 0.9833333333333333
```

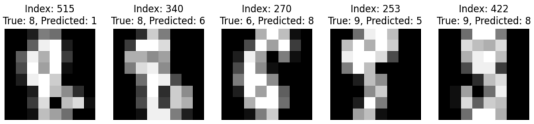
From the polynomial kernel onward, gamma parameters are introduced as the data is mapped into a high-dimensional feature space. Since the data is mapped into higher dimensions, it can better capture the non-linearity in the data, resulting in more accurate class classification.

Finally, the accuracy results when applying SVM with an RBF kernel (c=0.6, gamma=0.001) are presented.

```
Training Accuracy: 0.9976133651551312
Test Accuracy: 0.9907407407407407
```



The figure above visualizes the training dataset in a 2-dimensional space by applying t-SNE to the decision function values of the SVM model trained with the RBF kernel trick. This visualization reveals clusters formed by each class in the 2-dimensional coordinates.



As seen in the figure, there is a considerable overlap between clusters, especially for number pairs like 8 and 1, 8 and 6, or 9 and 5, 3 and 5. This is due to the structural characteristics of these digits.

2.5 Multilayer perceptron

MLP, or Multilayer Perceptron, is a type of artificial neural network with a structure composed of multiple layers. It consists of an input layer, hidden layers, and an output layer. Each layer is composed of neurons (perceptrons), and each neuron has weights and an activation function.

MLP is suitable for describing complex non-linearities in data through multiple hidden layers. It can be applied to multi-class classification and regression problems, depending on the number of neurons in the output layer. Additionally, MLP offers flexibility by allowing the adjustment of the number of neurons not only in the output layer but also in the hidden layers, allowing control over the model's complexity. However, it has the drawback of having more parameters compared to the models introduced earlier, leading to longer training times and higher memory requirements. Moreover, with many hyperparameters influencing model performance, effectively tuning them can be a challenging process.

The accuracy achieved through training with MLP is presented.

```
Training Accuracy: 0.9992044550517104
Test Accuracy: 0.9777777777777777
```

Despite MLP demonstrating good performance, when compared to the relatively simple model like Logistic Regression with an accuracy of approximately 96%, the training time for MLP is around 168 times longer. This significant time difference can become more pronounced when dealing with more complex and larger datasets in the future.

```
'Logistic Regresstion' Elapsed Time: 0.02909 seconds
```

```
'MLP' Elapsed Time: 4.86812 seconds
```

Conclusion

This analysis holds significance as it applies the algorithms learned during artificial intelligence classes to the MNIST digits dataset, comparing and analyzing the algorithms, strengths and weaknesses, and performance of each model. Various algorithms were applied to classify nonlinear data in a multi-class setting, and, in general, the performance of the models was observed to be favorable. If a situation arises where only one model needs to be selected, the polynomial SVM model (98.33%) is preferred due to its suitability for classifying nonlinear data with a moderate level of model complexity. However, considering the drawback of SVM becoming significantly slower as the data size increases and the uncertainty about the amount of future data, one might consider MLP (97.78%) or CNN, which are more optimized for large-scale data, for future model updates if the training time becomes a critical factor.