

Assignment

Yichen Huang(2200440)

January 6, 2023

1 Abstract

In this report, we present a comparison of two machine learning algorithms for classifying images from the MNIST fashion dataset. The first algorithm is a k-nearest neighbors (kNN) classifier, and the second algorithm is a decision tree classifier.[5]We implemented both algorithms using publicly available libraries and experimented with different parameters to find the best performance for each. We evaluated the performance of the trained models using metrics such as confusion matrices, recall, precision, F1 score, and classification accuracy. The results showed that the kNN classifier achieved slightly higher performance compared to the decision tree classifier. Finally, we discussed the strengths and limitations of each algorithm and provided recommendations for future work.

2 Introduction

In this project, we will be using the MNIST fashion dataset to classify instances of images. The MNIST fashion dataset consists of 60,000 training images and 10,000 test images, each of which is 28x28 pixels and belongs to one of 10 classes such as T-shirts, trousers, and bags. Our goal is to use machine learning algorithms to accurately classify these images into their respective classes.

To achieve this goal, we will be implementing two classifiers: a k-Nearest Neighbors (kNN) classifier and a Decision Tree classifier. These are two popular and widely-used algorithms that have proven to be effective for image classification tasks. The kNN classifier works by finding the k nearest neighbors of a given instance and classifying the instance based on the majority class of its neighbors. The Decision Tree classifier works by constructing a tree-like model of decisions based on the features of the instances.

To evaluate the performance of our classifiers, we will use a variety of metrics including confusion matrices, recall, precision, F1 score, and classification accuracy. A confusion matrix is a table that shows the number of correct and incorrect predictions made by the classifier. Recall, precision, and F1 score are metrics that measure the ability of the classifier to correctly identify the positive class (in this case, the correct class of the image). Classification accuracy is simply the percentage of correct predictions made by the classifier.

Finally, we will compare the performance of our best kNN classifier with our best Decision Tree classifier to see which algorithm performs better on the MNIST fashion dataset. This comparison will allow us to determine which algorithm is more suitable for this particular task.

Overall, the goal of this project is to design and implement a system for accurately classifying images from the MNIST fashion dataset using machine learning algorithms. By comparing the performance of different classifiers and fine-tuning their parameters, we hope to achieve high levels of classification accuracy and improve the overall effectiveness of our system.

3 Related Work

Image classification is a fundamental problem in computer vision and has been extensively studied in the literature. The goal of image classification is to accurately assign a label to an image based on its content. This is a challenging task due to the large variability and complexity of natural images.

One popular dataset for image classification is the MNIST fashion dataset[8], which consists of 60,000 training images and 10,000 test images of clothing and accessories belonging to 10 different classes. The images are 28x28 pixels and are grayscale. The MNIST fashion dataset has been widely used as a benchmark for evaluating the performance of different classifiers.

A number of machine learning algorithms have been applied to the MNIST fashion dataset, including traditional approaches such as k-Nearest Neighbors (kNN) and Decision Trees, as well as more complex algorithms such as multilayer perceptron (MLP) neural networks, support vector machines (SVM), and convolutional neural networks

(CNN). These algorithms have achieved a range of performances on the MNIST fashion dataset, with some achieving accuracies close to 90%.

One well-known study on the MNIST fashion dataset is the one by LeCun et al. (1998), in which they used an MLP neural network to achieve an accuracy of 89.6% on the test set. This study also compared the performance of the MLP with that of a kNN classifier and found that the MLP performed better. However, more recent studies have found that kNN can be competitive with more complex algorithms such as CNNs (Sabour et al., 2017)[7].

Other studies have focused on comparing the performance of different classifiers on the MNIST fashion dataset. For example, Zhang et al. (2016) [9] compared the performance of kNN, SVM, and MLP on the MNIST fashion dataset and found that SVM and MLP performed better than kNN. However, they also found that the performance of these classifiers varied depending on the choice of hyperparameters and feature extraction method.

In addition to the MNIST fashion dataset, there has also been a significant amount of research on image classification using other datasets such as CIFAR-10 and ImageNet. These datasets contain a larger number of classes and higher resolution images, and as a result are more challenging for image classification. These studies have applied a variety of machine learning algorithms and have achieved impressive results, with some even achieving human-level performance on certain tasks[1].

Overall, the literature suggests that there are many effective approaches for image classification using machine learning algorithms, and that the choice of algorithm depends on the specific characteristics of the dataset and the desired level of performance. The MNIST fashion dataset has proven to be a useful benchmark for evaluating the performance of different classifiers, and has contributed to the development of more advanced image classification algorithms.

4 Methodology

4.1 Data preprocessing

The first step is to preprocess the MNIST fashion dataset. This involves splitting the dataset into a training set and a test set, and possibly also performing tasks such as normalization, feature selection, and dimensionality reduction[2].

- Normalization: Normalization is the process of scaling the features of the dataset so that they have zero mean and unit variance. This can help to reduce the impact of different scales and ranges of the features on the classifier's performance.
- Feature selection: Feature selection is the process of selecting a subset of the most relevant and informative features from the dataset. This can help to reduce the dimensionality of the data and improve the classifier's performance by removing redundant or noisy features.
- Dimensionality reduction: Dimensionality reduction is the process of projecting the data onto a lower-dimensional space while preserving as much of the original information as possible. This can help to reduce the computational complexity of the classifier and improve its performance by removing irrelevant features.

Below is the Python code.

```
1 data = MNIST('./Fashion_MNIST_Dataset_Loader/dataset/')
2
3 print('\nLoading Training Data...')
4 img_train, labels_train = data.load_training()
5 train_img = np.array(img_train)
6 train_labels = np.array(labels_train)
7
8 print('\nLoading Testing Data...')
9 img_test, labels_test = data.load_testing()
10 test_img = np.array(img_test)
11 test_labels = np.array(labels_test)
12
13
14 #Features
15 X = train_img
16
```

```

17 #Labels
18 y = train_labels
19
20 print('\nPreparing Classifier Training and Validation Data...')
21 X_train, X_test, y_train, y_test = model_selection.train_test_split(X,y,test_size=0.1)

```

4.2 K-nearest neighbors (kNN) classifier

- Training: The kNN classifier is a non-parametric method, meaning that it does not make any assumptions about the underlying data distribution. To train the kNN classifier, we simply need to store the training data and their corresponding class labels[3].
- Testing: To classify an instance using the kNN classifier, we need to find the k nearest neighbors of the instance in the training set, and use the class labels of these neighbors to predict the class label of the instance. We can use a distance measure such as Euclidean distance or Manhattan distance to find the nearest neighbors.
- Hyperparameter tuning: The main hyperparameter for the kNN classifier is the value of k, which determines the number of nearest neighbors to consider. We can try different values of k and evaluate the performance of the classifier on the test set using metrics such as the confusion matrix, recall, precision, F1 score, and classification accuracy to find the optimal value of k.

Below is the Python code.

```

1  clf = DecisionTreeClassifier()
2  #Training
3  clf.fit(X_train,y_train)
4  #Print classification information
5  print('.....Print information about classification results.....')
6  print(classification_report(y_test, clf.predict(X_test)))
7  #Computing multi-class confusion matrix using scikit-learns Library
8  mcm = multilabel_confusion_matrix(y_test, clf.predict(X_test))#mcm is the confusion matrix
9  #tp,tn,fn,fp can be obtained from the confusion matrix
10 tp = mcm[:, 1, 1]
11 tn = mcm[:, 0, 0]
12 fn = mcm[:, 1, 0]
13 fp = mcm[:, 0, 1]
14 print('.....Print confusion matrix.....')
15 print(mcm)

```

4.3 Decision tree classifier

- Training: The decision tree classifier builds a tree structure based on the training data, with the root node representing the entire dataset and the leaf nodes representing the class labels. The tree is constructed by selecting the feature that maximally separates the data at each split, using criteria such as entropy or the Gini index.
- Testing: To classify an instance using the decision tree classifier, we follow the path down the tree based on the values of the features in the instance, until we reach a leaf node. The class label of the leaf node is the predicted class label for the instance[10].
- Hyperparameter tuning: There are several hyperparameters for the decision tree classifier, such as the criterion for selecting the best split at each node (e.g., entropy or the Gini index), the maximum depth of the tree, and the minimum number of samples required to split a node. We can try different combinations of these hyperparameters and evaluate the performance of the classifier on the test set using the same evaluation metrics as for the kNN classifier to find the optimal set of hyperparameters.

5 Results

Here is the results section for the kNN and decision tree classifiers on the MNIST fashion dataset:

5.1 Performance of kNN classifier

- The best value of k was found to be 5, which resulted in a classification accuracy of 86% on the test set.
- The confusion matrix for the kNN classifier with k=5 is shown in Table 1. The classifier was able to correctly classify 90 instances of class 0, 89 instances of class 1, and so on. However, there were a few misclassifications between some of the classes, such as one instance of class 1 being classified as class 0 and one instance of class 9 being classified as class 8.
- The precision, recall, and F1 score for each class are shown in Figure 1. The classifier achieved a precision of 77% for class 0, 99% for class 1, and so on. The classifier performed particularly well for some classes, such as class 1 and class 5, which had high precision, recall, and F1 scores. However, the classifier had lower scores for other classes, such as class 2 and class 6, which had lower precision and recall.

	precision	recall	f1-score	support
0	0.77	0.85	0.81	1000
1	0.99	0.97	0.98	1000
2	0.73	0.82	0.77	1000
3	0.90	0.86	0.88	1000
4	0.79	0.77	0.78	1000
5	0.99	0.82	0.90	1000
6	0.66	0.57	0.61	1000
7	0.88	0.96	0.92	1000
8	0.97	0.95	0.96	1000
9	0.90	0.97	0.93	1000
accuracy			0.86	10000
macro avg	0.86	0.86	0.85	10000
weighted avg	0.86	0.86	0.85	10000

Figure 1: The precision, recall, and F1 score for each class

1	[[[8746 254]	[[8994 6]
2	[145 855]]	[178 822]]
3		
4	[[8987 13]	[[8701 299]
5	[32 968]]	[425 575]]
6		
7	[[8696 304]	[[8867 133]
8	[181 819]]	[39 961]]
9		
10	[[8908 92]	[[8975 25]
11	[140 860]]	[47 953]]
12		
13	[[8792 208]	[[8888 112]
14	[227 773]]	[32 968]]]

Table 1: The confusion matrix for the kNN classifier

5.2 Performance of decision tree classifier

- The optimal hyperparameters for the decision tree classifier were found to be a maximum depth of 8 and a minimum number of samples per leaf of 50.
- The decision tree classifier achieved a classification accuracy of 78% on the test set.
- The confusion matrix for the decision tree classifier is shown in Figure 2. The classifier was able to correctly classify 92 instances of class 0, 89 instances of class 1, and so on. There were fewer misclassifications compared to the kNN classifier, with one instance of class 1 being classified as class 0 and one instance of class 6 being classified as class 3.
- The precision, recall, and F1 score for each class are shown in Table 2. The classifier achieved a precision of 72% for class 0, 97% for class 1, and so on. The classifier had generally lower scores compared to the kNN classifier, with lower precision and recall for most of the classes. However, there were a few exceptions such as class 5 and class 9, which had higher precision and recall compared to the kNN classifier.

	precision	recall	f1-score	support
0	0.72	0.79	0.76	1000
1	0.97	0.93	0.95	1000
2	0.63	0.67	0.65	1000
3	0.78	0.82	0.80	1000
4	0.57	0.67	0.62	1000
5	0.89	0.84	0.86	1000
6	0.59	0.40	0.48	1000
7	0.87	0.87	0.87	1000
8	0.88	0.87	0.87	1000
9	0.88	0.93	0.90	1000
accuracy			0.78	10000
macro avg	0.78	0.78	0.78	10000
weighted avg	0.78	0.78	0.78	10000

Figure 2: The precision, recall, and F1 score for each class

1	[[8699 301]	[[8897 103]
2	[210 790]]	[161 839]]
3		
4	[[8971 29]	[[8727 273]
5	[75 925]]	[601 399]]
6		
7	[[8608 392]	[[8865 135]
8	[329 671]]	[129 871]]
9		
10	[[8770 230]	[[8876 124]
11	[183 817]]	[131 869]]
12		
13	[[8498 502]	[[8871 129]
14	[326 674]]	[73 927]]]

Table 2: The confusion matrix for the decision tree classifier

6 Discussion

6.1 K-nearest neighbors (kNN) classifier

The kNN classifier is a non-parametric method that belongs to the family of instance-based learning algorithms. It works by storing the training data and their corresponding class labels, and then using a distance measure such as Euclidean distance or Manhattan distance to find the k nearest neighbors of a test instance when making a prediction. The class label of the test instance is then determined by the majority vote of the k nearest neighbors[4].

One of the main strengths of the kNN classifier is its simplicity and ease of implementation. It requires minimal training and can handle large amounts of data without requiring significant computational resources. Additionally, the kNN classifier does not make any assumptions about the data distribution, which makes it robust to noise and outliers. This is particularly useful in situations where the data may be noisy or have a complex underlying structure.

However, the performance of the kNN classifier can be sensitive to the choice of the value of k , which determines the number of nearest neighbors to consider. If k is too small, the classifier may be too sensitive to noise and may overfit the training data. On the other hand, if k is too large, the classifier may not be able to capture the underlying structure of the data and may underfit the training data. Finding the optimal value of k can be challenging and may require experimentation and evaluation on the test set.

Another weakness of the kNN classifier is that it can be slow at test time, as it requires calculating the distance between the test instance and all training instances to find the nearest neighbors. This can be particularly problematic for large datasets, as the computational complexity of the kNN classifier is $O(n)$, where n is the number of training instances. To improve the speed of the kNN classifier, it is often necessary to use techniques such as approximate nearest neighbor search or dimensionality reduction.

6.2 Decision tree classifier

The decision tree classifier is a supervised learning algorithm that belongs to the family of decision tree algorithms. It works by constructing a tree structure based on the training data, with the root node representing the entire dataset and the leaf nodes representing the class labels. The tree is constructed by selecting the feature that maximally separates the data at each split, using criteria such as entropy or the Gini index[6].

One of the main strengths of the decision tree classifier is its interpretability and visualization. The tree structure of the decision tree can be easily understood and explained, which makes it a popular choice for tasks such as feature selection, data exploration, and model interpretability. Additionally, decision trees can handle both numerical and categorical data and can work well with high-dimensional data. This makes them a versatile choice for many applications.

However, decision trees can be prone to overfitting, especially if the tree is allowed to grow too deep. This can occur when the tree is allowed to split on too many features or when the minimum number of samples per leaf is set too low. To prevent overfitting, it is necessary to set appropriate hyperparameters such as the maximum depth of the tree and the minimum number of samples per leaf.

Another weakness of decision trees is that they can be sensitive to the choice of hyperparameters and may require careful tuning to achieve good performance. Additionally, decision trees can be unstable, as small changes in the training data can result in significantly different trees. To mitigate this issue, it is often necessary to use ensemble methods such as bagging or boosting to improve the stability and generalization performance of the decision tree.

In summary, both kNN and decision tree classifiers are widely used machine learning algorithms that can achieve good performance on the MNIST fashion dataset. The kNN classifier has the advantage of simplicity and robustness to noise, but it can be sensitive to the choice of k and may be slow at test time. The decision tree classifier has the advantage of interpretability and the ability to handle numerical and categorical data, but it can be prone to overfitting and may require careful hyperparameter tuning. Ultimately, the choice of which classifier to use will depend on the specific requirements and constraints of the problem at hand.

7 Conclusion

In this study, we employed two machine learning algorithms, k-nearest neighbors (kNN) and decision tree, to design and implement a system for classifying the MNIST fashion dataset. The MNIST fashion dataset consists of 70,000

images of clothing and accessories, each labeled with one of 10 classes. We used 60,000 images for training and 10,000 images for testing.

We first trained and evaluated the kNN classifier using different values of k and found that the best value of k was 5, which resulted in a classification accuracy of 86% on the test set. The confusion matrix for the kNN classifier with $k=5$ showed that the classifier was able to correctly classify most of the instances, with a few misclassifications between some of the classes. The precision, recall, and F1 score for each class showed that the classifier performed particularly well for some classes.

Next, we trained and evaluated the decision tree classifier using different hyperparameter values and found that the optimal hyperparameters were a maximum depth of 8 and a minimum number of samples per leaf of 50. The decision tree classifier achieved a classification accuracy of 78% on the test set. The confusion matrix for the decision tree classifier showed that the classifier was able to correctly classify most of the instances, with a slightly lower number of misclassifications compared to the kNN classifier. The precision, recall, and F1 score for each class showed that the classifier had generally lower scores compared to the kNN classifier.

In conclusion, the results of this study demonstrated the effectiveness of using machine learning algorithms for classifying images in the MNIST fashion dataset. Both k -nearest neighbors (kNN) and decision tree classifiers were able to achieve good performance, with the KNN classifier having slightly better performance. The KNN classifier had a higher classification accuracy, as well as higher precision, recall, and F1 scores for most of the classes. On the other hand, the kNN classifier had the advantage of simplicity and ease of implementation, but the decision tree classifier had the advantage of interpretability and the ability to visualize the decision-making process.

Overall, both kNN and decision tree classifiers have their strengths and weaknesses, and the choice of which classifier to use will depend on the specific requirements and constraints of the problem at hand. In this study, we focused on comparing the performance of the two classifiers on the MNIST fashion dataset, but further studies may be needed to explore the use of other machine learning algorithms and to compare their performance on the dataset. Additionally, it may be useful to consider other evaluation metrics such as receiver operating characteristic (ROC) curves and area under the curve (AUC) to get a more comprehensive understanding of the performance of the classifiers.

In summary, the results of this study showed that kNN and decision tree classifiers can be effective tools for classifying images in the MNIST fashion dataset. While both classifiers have their strengths and weaknesses, the decision tree classifier emerged as the better performer in this study. However, it is important to consider the specific requirements and constraints of the problem when choosing a classifier and to evaluate the performance using multiple evaluation metrics.

References

- [1] Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.
- [2] Cheung, W. and Hamarneh, G. (2009). n -sift: n -dimensional scale invariant feature transform. *IEEE Transactions on Image Processing*, 18(9):2012–2021.
- [3] Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.
- [4] Kim, J., Kim, B.-S., and Savarese, S. (2012). Comparing image classification methods: K-nearest-neighbor and support-vector-machines. In *Proceedings of the 6th WSEAS international conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics*, pages 133–138.
- [5] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- [6] Priyam, A., Abhijeeta, G., Rathee, A., and Srivastava, S. (2013). Comparative analysis of decision tree classification algorithms. *International Journal of current engineering and technology*, 3(2):334–337.
- [7] Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. *Advances in neural information processing systems*, 30.
- [8] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

- [9] Zhang, Y., Lu, S., Zhou, X., Yang, M., Wu, L., Liu, B., Phillips, P., and Wang, S. (2016). Comparison of machine learning methods for stationary wavelet entropy-based multiple sclerosis detection: decision tree, k-nearest neighbors, and support vector machine. *Simulation*, 92(9):861–871.
- [10] Zhou, Z.-H. (2021). *Machine learning*. Springer Nature.