|  |
| --- |
| **Assignment 2** |

**Tutors: Maoying Qiao, Jue Wang**

**Group members: Wenjing Deng(470349281), Hefei Chen(470488188), (Li Shihao, 480106740, shli4656)**

**Abstract**

Machine learning has been an important role in image classification which is one of the most promising technologies in many practical applications. There are many different algorithms for the image classification which have different performance. Based on the dataset of MNIST fashion, this paper provides three available methods to realize the image classification, which includes Random Forest, K-Nearest Neighbors, and Convolutional Neural Network. The theory and design choice of each method will be introduced. To detail each method, it will also mention the main methods which include data pre-processing and the classification approaches. In order to find an optimal classification models among them, the performance of three different methods are compared in terms of accuracy, precision, recall and so on. In addition, a meaningful discussion and relevant personal reflection will be presented. Finally, a convincible conclusion will be evaluated according to the performance of three methods.

Keywords: Machine learning; Image classification; Random Forest; K-Nearest Neighbors; Convolutional Neural Network

1 Introduction

The core issue of the project is to use a variety of different machine learning methods to classify or regress data in different data sets to achieve a better understanding and application of machine learning algorithms. In this assignment, after a group discussion, we finally decided to use the Fashion-MNIST dataset as our training and testing dataset. Fashion-MNIST consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. The 10 labels are T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. In this report, we applied three different classifiers, including K-Nearest Neighbors, Random Forest and Convolution Neural Network, to train and predict the labels to which different image data belong.

The problem of image classification is very important because it is popular and can be applied in various industries. Through this study, we can know more about the difference and characteristics among these methods. Moreover, a comprehensive understand of the whole process on how an algorithm can be implemented and how the performance of these methods will be study. This knowledge will contribute to a deeper and further study in the future and help us a lot in the field of machine learning.

2 Previous work

Fashion-MNIST is a famous dataset of Zalando's article images. It is one of the most common and basic data sets. At the same time, Fashion-MNIST also is one of the most worthwhile data sets for junior developers and researchers in the field of data mining and mechanical learning. Due to the moderate amount of dataset capacity and simple data structure, Fashion-MNIST can be applied to test a variety of different classification algorithms, such as SVM, CNN, KNN, FR.

Until 2018, many researchers have applied classification algorithms to achieve high accuracy in the Fashion-MNIST data set. In 2017, Li, Chen and Rudin design their CNN classifier based on a unique architecture and test their algorithm on Fashion-MNIST which get classification accuracy of 89.95%. Currently, Rolinek and Martius optimized the stochastic gradient descent method, adopted an adaptive scheme of stochastic gradient descent (SGD), and finally demonstrated its function by applying the scheme to improve the performance of the Adam and Momentum optimizer. The average accuracy of their SGD method is 93.25%. In addition, Agarap Combines Convolutional Neural Network (CNN) and Support Vector Machine (SVM) into an architecture for classification and get accuracy of 91.29% based on Fashion-MNIST in 2017.

**3 Methods**

3.1 Theory on different techniques compared

There are two main categories of machine learning algorithm for image classifier: one is the parametric classifiers, the other is non-parametric classifiers. Parametric classifiers are summarizes data with a set of parameters of fixed size. Simple Neural Networks are very famous representations which are easier to understand and interpret results. In terms of the speed, parametric models are very fast to learn from data. In addition, they can work well even if the fit to data is not perfect. Nonparametric methods seek to best fit the training data in constructing the mapping function. They can fit a large number of functional forms. For example, k-nearest neighbors algorithm is the most common method. It makes predictions based on the k most similar training patterns for a new data instance.

It has been more than half a century since the KNN rule was first proposed[2]. In this half century, KNN has been recognized as one of the top ten algorithms in the field of mechanical learning and data mining due to the simplicity, validity and usability of KNN-based classifiers[5]. KNN is an instance-based learning, or a lazy learning that defers all calculations to classification. The core idea of the classical KNN is that if a majority of the k most neighboring samples in a feature space belong to a certain category, the sample also belongs to this category and has the characteristics of the samples on this category. This means that each sample can be represented by its nearest k neighbors.

Convolutional Neural Network (CNN) has attracted considerable attention in recent years which has a remarkable and excellent performance in various visual recognition problems, especially the problem of image classification. In general, a convolutional neural network is consisted of an input layer and an output layer which are also multiple hidden layers. The hidden layers of a CNN include convolutional layers, pooling layers, fully connected layers and normalization layers. This structure enables CNNs to take advantage of the two-dimensional structure of input data. Compared with other deep learning structures, CNNs can give better results in terms of image recognition. This model can also be trained by using the backpropagation algorithm which can consider less parameters than other deep and feed-forward neural networks. There are three main distinguishing features of CNNs. The first one is 3D volumes of neurons. The layer of CNNs has three dimensions of neurons: width, height and depth. The second one is local connectivity. CNNs utilize the spatial locality to enforce join local pattern between adjacent neurons. The last one is shared weight. It reduces the complexity of the network, especially that the multi-dimensional input vector images can be directly input to the network. This feature avoids the complexity of feature extraction and data reconstruction in the process of image classification. According to three features, CNNs have a better performance in dealing with multi-dimensional data and extraction features.

3.2 Pre-processing

Data preprocessing plays an important role in many algorithms of data mining because the quality of data improve the efficiency and ease of the process of data mining. There are many pre-processing techniques like clustering, regression, and normalization. Due to the data set we choose is the fashion-MNIST, we apply appropriate methods to preprocessing according to the kind of data which will be classified. In this part, we will introduce the preprocessing of three methods separately.

3.2.1 Random forest

In Random Forest, we tried two pre-processing methods: denoising and compression. Firstly, we test Average Color Method (point A means itself and its surrounding numbers). However, this method did not increase the accuracy obviously but increase the runtime. So we delete relevant code. Secondly, we compress color level. The provided pictures are of 256-level gray color and we compress them to 16-level pictures. Then the compress process can decrease the calculation to decrease runtime. In the meantime, the similar color, such as 150 and 159, can be regard as one color (15). This also can contribute to denoising.

3.2.2 K-Nearest neighbors

In the process of implementing KNN and Random Forest, we tried three different pre-processing methods, including denoising, normalization, and compression grayscale. After implementing and comparing these three normalization methods, based on the accuracy and time complexity, we decided to use the compression grayscale (slightly reduce the accuracy, significantly reduce the time complexity) as a data pre-processing method. Specifically, we divide each pixel of each graph in the data set by 16, thus converting the grayscale image of [0, 255] into a grayscale image of [0, 15], thereby reducing the amount of calculation and lightly denoising (eliminate irrelevant differences between data).

3.2.3 Convolutional neural network

There are three preprocessing methods will be used in CNN.

1. Normalization

Normalization is the first step in the data preprocessing which is proportional to scale into a small specific range. Considering the data is the natural image, a common way is to acquire the value of data vector between [0, 1]. So we adjust the value of every dimension of the data through dividing pixel value by 255 to scale them in [0, 1]. In this way, it is easier to deal with data processing when the features are on the similar scale.

1. One-Hot Encoding

In the algorithms for classification, computing distances between features or similarities between features is very important. Many definitions of distances and similarities are defined over features in Euclidean space. So we use one-hot encoding to realize that binarize the categorical input to embed the corresponding vector in the Euclidean space. It solves the problem that classifier do not categorical data well.

1. K-fold cross validation(k=10)

In this algorithm, we use 10-fold validation which means use 1/10 of the training data for validation. Because it is impossible to access test data in some situations, using part of the available data for test is more feasible. So we split our data into 10 parts, which 9 of them are training sets and 1 is validation set. Then training model on training set and adjust parameter, objective function and model structure according to the performance tested on the validation set.

3.3 Design choices

3.3.1 Random forest

RF is an algorithm that derives from Decision Tree algorithm and the Bagging strategy. Decision Tree is an algorithm that obtain the final classification and prediction by continuously setting branch conditions while Bagging strategy is a simple algorithm that decides the final model(s) by vote. RF contains a set of simple Decision Trees, which each produce a predictor that can classify the dataset and predict the result. The RF then rank the presented features of each tree by Bagging strategy and use the highest score of features of classification model to predict the result. Given an input dataset

D\_n={（X\_i,Y\_i）}\_(i=1)^n

the number of iterations is T. With t = 1, 2..., T, the algorithm randomly samples for m times in total, creating m times of sample: Dm. The algorithm trains Dm in order to obtain the Desion Tree Gm(x). During this process, the RF compute the importance of j-th feathure, which is average of the difference in out-of-bag errors. Afterwards, the T iterations vote the final model by the j-th features.



3.3.2 K-Nearest neighbors

After pre-processing, designing and implementing KNN can be divided into the following steps. First, set the value of K. Since k is the only variable parameter in the KNN algorithm, the classification is easily affected by the sensitivity of k, which affects the classification accuracy. Second, traverse the test set. For each sample in the test set, calculate the Euclidean distance:



from the sample (test set) to each sample in the training set; then, extract the category labels of the k samples with the smallest distance from the training set to the sample (test set). Finally, the category labels are counted, and the category labels are most often the category labels for that sample (test set).

* + 1. Convolutional neural network

In our algorithm, the structure of the network is: convolutional layer – polling layer – convolutional layer – polling layer – fully connected layer – fully connected layer. It is shown as Figure 1.

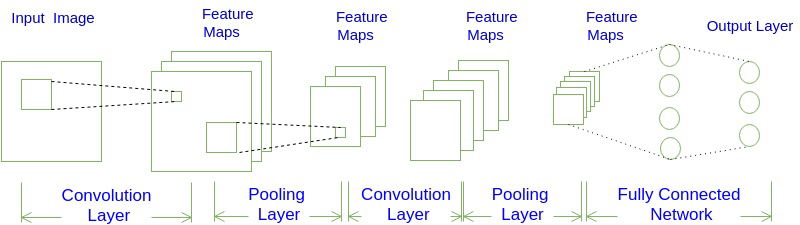


Figure 1 The structure of network

1. Convolutional layer

Convolutional layer consists of several convolution neurons. Each convolutional neuron processes data only for its receptive. Considering our input data is image which has a large number of pixels, we input an image through the convolutional layer to extract the features of the image. So we choose 32 convolution kernels with the size of 3\*3 in the first convolutional layer and 64 convolution kernels with the size of 3\*3 in the second convolutional layer. In this way, only 25 learnable parameters will be required. Each of them with the same shared weights. It reduces the number of free parameters which help the network to be deeper with fewer parameters.

1. Pooling layer

The polling layer combines the outputs of neuron clusters at one layer into a single neuron in the next layer. In out algorithm, we choose the max pooling to get the maximum value from each of a cluster of neurons at the prior layer. Two main benefits can be achieved. One is it can keep invariance which includes translation, rotation, and scale. The other is it can retain the main features meanwhile reducing the parameters and the amount of calculation to prevent over-fitting and improve the generalization ability of the model. The pooling layers use a filter of size 2\*2 applied with a stride of 2 downsamples at every depth slice in the input by 2 along both width and height which will be discarding 75% of the activations according to the corresponding result of convolutional layer.

1. Fully connected layer

The neurons in the fully connected layer are connected to all the activations in the preceding layer and it will be computed with a matrix multiplication followed by a bias offset to implement the classification by softmax.

1. Activation function

(1)ReLU(Rectified linear units) activation function

Using ReLU activation function will be used after convolutional layer which can decrease the calculation of the whole process. It will output 0 if the input is less than 0. If the input is greater than 0, it will output the result which is equal to the input. The mathematical formula is



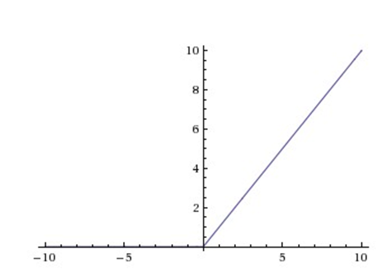
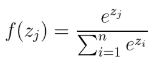


Figure 2 ReLU activation function

(2)Softmax activation function

The softmax activation function will be used in output layer which maps the output to interval values (0, 1), and normalizes the sum of all elements to be equal to 1. It can be treated directly as probability. The highest probability classification is chosen as the target of prediction. Because the using of exponent in softmax, it can increase the contrast between larger values and smaller values which can improve learning efficiency. It is also continuously derivable which eliminating inflexions. This feature is very necessary for gradient descent in machine learning.

The formula of softmax is:



4 Experiments and Discussion

4.1 Experiments of each methods

4.1.1 Random forest

Table 2: Random Forest Confusion Matrix

Predicted    0    1    2    3    4    5    6    7    8    9   All

True

0          622    5   16   22    3    0  138    0    4    0   810

1            2  641    0    4    1    0    3    0    1    0   652

2           14    0  546    7   77    0   87    0    3    0   734

3           28   25   10  620   25    0   20    0    3    0   731

4            1    3   78   15  544    0   68    0    5    0   714

5            1    0    0    0    2  674    0   20    3   13   713

6           70    2   54   17   37    0  382    0    4    0   566

7            0    0    0    0    0   28    0  671    4   21   724

8            8    0    3    0    2    2    3    0  631    0   649

9            0    0    0    0    0    8    0   26    1  672   707

All           746  676  707  685  691  712  701  717  659  706  7000

Table 5: Random Forest Performance

       precision    recall    f1-score   support

0       0.77      0.83      0.80       746

1       0.98      0.95      0.97       676

2       0.74      0.77      0.76       707

3       0.85      0.91      0.88       685

4       0.76      0.79      0.77       691

5       0.95      0.95      0.95       712

6       0.67      0.54      0.60       701

7       0.93      0.94      0.93       717

8       0.97      0.96      0.96       659

9       0.95      0.95      0.95       706

Average   0.86      0.86      0.86       7000

4.1.2 K-Nearest neighbors

Regarding the KNN classifier, we mainly focus on the pre-processing method and the choice of the parameter K.

In pre-processing part, two pre-processing methods are been tested, one is compression grayscale, and another is binarization. About K parameter, we selected 9 different K (1, 3, 5, 10, 11, 13, 15, 21, 23) for testing. Figure 1 and Figure 2 shows the accuracy of KNN classifier for different K parameter and pre-processing methods. When the parameter K is taken to 3, the KNN based on the binary method achieves the highest accuracy of 83.50%; when the parameter K is taken to 5, the KNN based on the compressed gray scale reaches the highest accuracy of 85.53%.

Figure1: Accuracy of Different K under Binarization

Figure 2: Accuracy of Different K under Compression Garyscale

In the evaluation of time complexity, the average cost of the binarized KNN classifier is 1031s, and the average cost of the KNN classifier based on the compressed gray scale is 722s. Therefore, by weighing the algorithm accuracy and time complexity, it is clearly to find when K=5 and using compression grayscale method, the KNN classifier has the best performance.

In order to better test the performance of the above KNN classifier(K=5), we applied 10 cross-validation to test its performance in more detail and accurately. In this part, the external library sklearn is been used to provide 10 cross-validation function. By applying 10 cross verification, the performance of this 5-NN classifier can be better expressed as:

Table 1: Average Confusion Matrix

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 514.7 | 1.6 | 8.1 | 23.4 | 2.1 | 0.4 | 104.1 | 0 | 2 | 0 |
| 0.6 | 580.9 | 0.1 | 3.3 | 0.5 | 0 | 0.5 | 0 | 0.1 | 0 |
| 9.6 | 3.3 | 478.5 | 5.8 | 59.1 | 0.1 | 73.3 | 0 | 8.8 | 0 |
| 11.3 | 10.1 | 5 | 521.0 | 16.8 | 0.1 | 10.3 | 0 | 3 | 0.1 |
| 2.6 | 0.6 | 57.9 | 27.7 | 461.4 | 0.1 | 42.2 | 0 | 3.2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 490.2 | 0 | 1.6 | 0.6 | 1.3 |
| 56.6 | 3.4 | 49.4 | 17.9 | 59.1 | 3.5 | 365.4 | 0.2 | 7 | 0.3 |
| 0.3 | 0 | 0.2 | 0 | 0 | 60.6 | 0 | 568.8 | 3 | 18.2 |
| 4.2 | 0 | 0.6 | 0.9 | 1 | 1.9 | 4.1 | 0.1 | 570.6 | 0.1 |
| 0.1 | 0.1 | 0.2 | 0 | 0 | 42.9 | 0.1 | 29.3 | 1.7 | 580.0 |

4.1.3 Convolutional neural network

1. The size of convolutional kernel

In general, the size of convolutional filters in CNNs always can be 3\*3, 5\*5, or even greater. So we just choose three numbers as the size of convolutional filters. There are 1\*1, 3\*3, and 5\*5 which is shown in figure 3. The abscissa of the figure is the step of training, the ordinate of the figure is the accuracy of each step. The results are shown in figure 3.

Figure 3 Training accuracy with different size of convolutional kernel

According to the Figure 3, the size of 3\*3 and 5\*5 are have similar accuracy after 1000 steps. Because the smaller the size of convolutional kernel, the less parameters we need and the less calculation. Therefore, we choose 3\*3 as the size of convolutional kernel.

1. The number of convolutional kernel

The number of convolutional kernel is always 32 or 64, when the neural network is used to deal with image classification. Because we have two convolutional layers, we try both of the number to each layer to identify which one has better performance. The results are shown as figure 4.

Figure 4 Training accuracy with different number of convolutional kernel

According to figure 4, when the first layer and second layer are 32, the accuracy is lower than the others. So we will choose between the others. Considering the training time of these, when the first layer and second layer are 64, when running 5000 steps, it will spend twice times to achieve a similar accuracy with the first layer is 32 and the second layer is 64. Finally, comparing the accuracy and running time, we decide to choose the first layer is 32 and the second layer is 64.

1. Learning rate

Learning rate plays a decisive role in the convergence of networks. Considering the gradient descent: 

The is the learning rate. If the rate is too small, it will lead to a very slow decrease of loss. But if the rate is too large, the loss will be increased. The selection strategy of learning rate is very important in the process of training, so choose some learning rate as is shown in figure 5.

Figure 5 Training accuracy with different learning rate

As is shown in figure 5, the abscissa of the figure is the step of training, the ordinate of the figure is the accuracy of each step. When the learning rate is 0.0001, the accuracy is always the highest. Therefore, the learning rate of our algorithm is 0.0001, which is also as 1e-4.

1. Other elements which represent the performance of our algorithm

* Accuracy

The calculation of accuracy is (TP+TN)/(TP+TN+FP+FN). After classified all training data by 10-fold cross validation, we calculate the average of the 10 results. The average accuracy is 89.58%. In addition, when running over 20000 steps, the average accuracy can over 91.50%

* Precision

The calculation of precision is TP/(TP+FP), which is the proportion of correct part among the particular predicted class. The average precision of our algorithm is 89.90%.

* Recall

The calculation of recall is TP/(TP+NP),which is the percentage of the correct portion over the actual class. The average recall of our algorithm is 89.60%.

* F1 Score

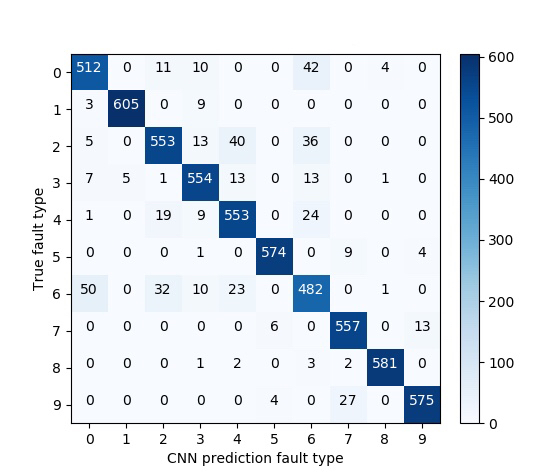
F1 score is a measure of a test’s accuracy. It is the 2\*((precision\*recall)/(precision+recall)). In our algorithm, the average f1 score is 89.40%.

* Support

The support is the number of samples of the true response that lie in that class. The average of support is 600.

* Confusion matrix

Confusion matrix describes the performance of a classification model on a set of test data for which the true vales are known. The average of confusion matrix is as shown in the table.



* Run-Time

The whole algorithm will need to run 3550s to get the results of 10-fold cross validation. The average of each cross validation is 355s.

* 1. Comparisons and evaluation

4.2.1 Software and hardware

For all three classifiers, the hardware includes CPU: 2.5 GHz Intel Core i7, memory: 16 GB 1600 MHz DDR3; the software is window 10 platform.

The average runtimes and accuracies in 10-fold cross validation are presented in Table 000\_\_. The CNN is the most accurate method among the 3 methods (91.5%). The RF and k-NN have similar accuracy. From perspective of run speed, the RF is overwhelmingly quicker than other two classifications (1.94s).

* + 1. Comparison and discussion

Table 1: Runtimes and accuracies

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Accuracy** | **Runtime** |
| Random Forest | 86.0% | 1.94s |
| k-NN | 85.53% | 722s |
| CNN | 91.5% | 355s |

In this section, from the above information, we found that, in one algorithm, each of fold (in 10-fold Cross Validation) has the similar performance and runtime. This means that these algorithms we chose works in a stable method.

Apart from that, when we choose test set by 10-fold Cross Validation, we take different strategies: sequential data; and totally muddled statistics. The choose of strategies did not impact the accuracies in all 3 algorithms. This means that the training data is even and muddled.

RF perform excellent in time at less than 1.94 seconds, but normal in accuracy, at 86.0%. The accuracy and runtime are majorly impacted by parameters of estimators and jobs, which means the

5 Conclusions and future work

In conclusion, in the chosen task, all 3 machine learning algorithms can contribute accuracies more than 85%. Among them, CNN can obtain the highest accuracy, at 91.5%. By contrast, the Random Forest, with 86.0% accuracy, is advantageous in runtime, at only 1.94, 1 in 150 of CNN. The k-NN is not recommended because it does terrible in both time and accuracy.

We also found that these algorithms did not get the utmost out of CPU.

Therefore, in the future, we would use other technologies, such as the multithreading, to make full use of CUP ability in order to develop time performance. It is predicted that k-NN algorithm written will have a better performance by this method.

References

1. Agarap, A. F. (2017). An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification. *arXiv preprint arXiv:1712.03541.*
2. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification.*IEEE Transactions on Information Theory,*13(1), 21-27.
3. Li, O., Liu, H., Chen, C., & Rudin, C. (2017). Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. *arXiv preprint arXiv:1710.04806*.
4. Rolinek, M., & Martius, G. (2018). L4: Practical loss-based stepsize adaptation for deep learning.
5. Wu, X., & Kumar, V. (2009). *The top ten algorithms in data mining*. Boca Raton: CRC Press.

Appendix

1. Contributions

Contributions shows in the below table.

Table 1: Contribution Table

|  |  |  |
| --- | --- | --- |
| Name | Code | Report |
| Shihao Li  (480106740) | Random Forest Code | Random Forest-related information in Methods and Experiment  Conclusion  Appendix |
| Hefei Chen  (470488188) | k-NN Code | Introduction  Previous work  KNN related part of report |
| Wenjing Deng  (470349281) | CNN Code | Abstract  CNN related part of report  Specification article format |

1. instructions of code

Our assignment is based on MNIST-fashion datasets. The download website address is:

https://github.com/zalandoresearch/fashion-mnist

We use 3 classification: Random Forest, CNN, and k-NN. Each method is written in one independent python file. These files are written by Python 3.

In the python files, we use several open-source libraries: numpy (v1.14.3) scikit-learn (v0.20.0), pandas (v0.23.0).