

Classification using ML methods and Neural Networks

by Xueyang Hu

Submission date: 26-Mar-2023 04:53PM (UTC+0100)

Submission ID: 200553160

File name:

41181_Xueyang_Hu_Classification_using_ML_methods_and_Neural_Networks_495066_1831940357.docx
(98.14K)

Word count: 5718

Character count: 33065

Xueyang Hu
Software Engineering and Artificial
Intelligence
University of Wales Trinity Saint David
Swansea, Unit Kingdom
812914168@qq.com

However, there is a huge defect in supervised learning method, that is, it can't learn the data without prior knowledge which is provided by human. Therefore, unsupervised learning was proposed. The main purpose of unsupervised learning is to find the pattern of data in a pure dataset with noise, which can provide assistance for some data analysis work that is hard to find the features of data or cost a lot to find them[6]. The unsupervised learning usually be used to solve clustering and dimensionality reduction problem, and algorithms such as principle component analysis(PCA) and Kmeans etc belongs to this category. It's worth mentioning that there is a semi-supervised learning methodology which between supervised learning and unsupervised learning. This

kind of algorithm will be provided with a small data sets with prior knowledge and a large number of unlabeled data sets, and learn data patterns from them, and finally train a model that can use prior knowledge to interpret data. This is usually applicable when there is less manually labelled data but the aim is to obtain the a result which likes supervised learning.

Self-supervised learning is an extension of unsupervised learning. The biggest difference between them is that it will use inherent cooccurrence relationships to achieve self-supervision, which means the self-supervised learning can automatically construct label to data without prior knowledge through pretext, so as to achieve the same training as supervised learning. For tasks such as cloze test, the trained language model can predict the words to be filled according to the similar context appeared in training dataset[7]. Generative Adversarial Nets(GAN) is one of the popular neural network algorithms with self-supervised learning as the core concept. It was first proposed by Ian Goodfellow et al. in 2014 [8]. This algorithm includes two neural network models, in which the generation model G is responsible for generating data, and the discrimination model D is responsible for determining whether the data source is G or groundtruth. The judgment result of D will be fed back to G through G's loss function, so that G will continue to approach the real data model, and D will continue to improve its judgment ability through its own loss function. Through this framework of minimax two-player game, user can finally get a model G that can generate data close to real data, which is very useful in some fields such as image generation. Another neural network model that adopts the concept of self-supervised learning is autoencoder [9]. Its input and expected output are groundtruth. It learns by minimizing the input reconstruction error, and its intermediate products can be used as the compressed representation of the input. Its output model part can also be used in some reconstruction work.

In the above learning process, the data input is passive, and the learning machine cannot learn actively and obtain results. In order to solve this problem, the concept of reinforcement learning was proposed. It learns the current problems through trial-and-error interaction with the environment [10], and its core is a set of rewards and punishment mechanism for the learning machine. This kind of algorithm is very useful in solving some problems that require the learning machine to dynamically change the strategy according to the environment change. For example, the framework Goal-Oriented Action Planning (GOAL) commonly used in the game AI design process is designed based on this idea, which can change its task priority automatically according to the external environment changes and its current state. A kind of neural network associated with this concept is the deep Q-network (DQN) proposed by Mnih V et al. It applies the output of the neural network to the environment, and designs the loss function based on the feedback of the environment, so as to continuously improve the neural network [11].

III. METHODOLOGY

A. Random forest

Before introducing the random forest algorithm, we need to know what the decision tree is. The decision tree is a tree-structured decision process model, in which each intermediate

node represents a decision process, and each leaf node represents a decision result. The data is input from the root node, judged by intermediate nodes, and finally output by the leaf nodes. The training process, or the process of constructing a decision tree, is a process of using greedy algorithm to select the best segmentation location to make the segmented subsets as pure as possible, including CART, ID3, C4.5 and other construction methods. It is considered to be one of the most effective data mining algorithms, and has been widely used in various disciplines because of its ease of use and robustness in the presence of missing values[12].

However, the decision tree also has its disadvantages. When the amount of training data is large or the training data's dimension is high, its construction process will be very slow, and the final structure of the decision tree will be too complex, resulting in a reduction of computing speed. At the same time, it gives a local optimal decision in the calculation process of each intermediate node, which makes it hard to get a global optimal result. In order to solve these problems, random forest algorithm is proposed. Inspired by the Bagging aggregation scheme, it divides the training dataset into multiple subsets, uses each subset to generate a decision tree, and finally forms a random forest [13]. In the process of application, the same input data will be input into each decision tree, and the output of each decision tree will be integrated in some way (vote best or average value), and finally output a result. Random forest not only has the advantages of decision tree, but also shows excellent processing ability for large or high-dimension data sets. According to the experimental results of Ali J et al., when faced with the same large data set, the accuracy of random forest is higher than that of decision tree, while the decision tree performs better on small data sets [14].

Today, random forest has been successfully applied to solve various practical problems, including prediction of air quality data, ecology, bioinformatics, 3D object recognition, etc. [13]. Belgiu M et al. verified its excellent performance in the field of remote sensing [15]. Compared with other classifiers, random forest classifier is more suitable for classifying hyperspectral data which have high dimension and high data correlation features. At the same time, it also shows excellent results in the selection of optimal variables computed from sensors or combination of remote sensing and geographic data.

B. Neural network

According to the research, the concept of neural network originated from William James, an American psychologist, who explained the basic principles of relational learning and associative memory in the brain. This biological concept inspired later research, and McCulloch and Pitts proposed the first complete neural network model theory [14]. Feedforward neural network is the most basic neural network model. It can be seen as a nonlinear mathematical function that converts a set of input variables into a set of output variables, and the conversion process is controlled by weight parameters. and the process of adjusting weight is called training [15]. In summary, the neural network model is a mathematical function model based on the neural structure of the brain. Its training process is the process of constantly approaching the ideal function by adjusting the weights in the function.

Neuron is the basic unit in the neural network. It is a linear function that multiply the data from several neurons from the previous layer with different weight values, sum them up and transfers the result to the neurons in the next layer. In order to facilitate calculation, neurons are usually presented in the form of weight matrix in the neural network, and the process of data conversion is the process of matrix multiplication.

The neural network is a hierarchical model. Each layer of neural network is composed of several neurons, which are responsible for receiving input data or data from the previous layer and outputting or transmitting them to the next layer after transformation. The earliest neural network is Perceptron, which only contains one input layer and one output layer. Later, the Multiple Layer Perceptron(MLP) introduced the concept of hidden layer to simulate more complex functions. Generally speaking, the more hidden layers a neural network has, the more complex the functions that can be simulated, but the higher the storage required. Therefore, some solution like Sparse Connection and Weight Sharing are proposed to solve this problem. In addition to these three layers, other layers such as the convolution layer for feature extraction and the pooling layer for compressed data have also been proposed in subsequent studies, but these three layers are essential. The whole process of obtaining results is also called forward promotion.

Back Propagation is the most basic training process of neural network. It is used to adjust the weight parameters of neurons, that is, the weight matrix. It is based on the loss function, which is used to calculate the error between the output result of the neural network and the groundtruth (in supervised learning, it is the label of the training data). Through a reverse chain derivative starting from the loss function, each weight obtains its current gradient value, so as to know which direction it should change if the result wants to approach the ideal value. The influence range of each neuron's reverse derivation is determined by neurons it connects to in the upper layer. Through continuous forward and back propagation training of the neural network, it will continue to approach the ideal function.

If only the weight matrix plays a conversion role in data processing, the neural network model can only be used to fit linear function. To solve this problem, the activation function is proposed. It is a nonlinear function, which is usually attached to some layers of the neural network for nonlinear transformation of the results. In this way, the neural network can also be used to fit the nonlinear transformation process. At present, the commonly used activation function is ReLU function, which can solve the problem of gradient

disappearance in the back propagation process while providing the nonlinear transformation process.

So far, neural networks have evolved different architectures to solve different problems, such as CNN that can extract data features[16], GNN that can process data in a graph architecture[17], and Resnet that is used to eliminate the effect of poor middle layer[18]. But their core is to fit the problem solving process by adjusting the weight parameters. Because it can fit arbitrary functions in theory, it is widely used in many fields such as image recognition, speech processing, 3D reconstruction, etc. However, due to the unknowability of its training process and the difficulty of interpretation to model, the effect of it is not always stable. Therefore, many applications of it are still in the experimental stage.

IV. RESULT AND DISCUSSION

A. Preparation

The training and testing of the ML algorithms are based on two datasets: Fashion MNIST data set and CIFAR10 data set. Fashion MNIST dataset is a 28 * 28 size clothing grayscale image dataset with 10 types of data, including pullover, bag, shirt, trouser, coat, T-shirt, dress, sandal, ankleboot, and snake [19]. Compared with the original MNIST dataset, its classification task is more challenging. CIFAR10 data set is a 32 * 32 color image dataset with 10 types of data, including airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck [20].

The implementation of the algorithm is based on two python libraries: scikit-learn and tensorflow. Scikit-learn is a free machine learning library on python, which includes the implementation of various classification, regression and other kinds of algorithms. It is mainly used to achieve random forests. Tensorflow is an open source deep learning platform based on dataflow programming, which is used to achieve neural networks.

The IDE of the experiment is Pycharm Community 2022.

B. Result of random forest

There are two main random forest parameters in the following test: `n_estimators` to determine the number of decision trees in the random forest, and `max_depth` to determine the maximum depth of each tree.

When `n_estimators = 50`, `max_depth = 20`, the confusion matrix of classification result is as Table I.

TABLE I. CONFUSION MATRIX OF RANDOM FOREST

True predict \	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle
T-shirt	853	0	15	31	4	1	84	0	12	0
Trouser	4	953	4	27	3	0	8	0	1	0
Pullover	12	0	790	9	127	0	57	0	5	0
Dress	21	2	9	896	32	0	37	0	2	1
Coat	0	0	100	34	802	0	61	0	3	0
Sandal	0	0	0	0	0	955	0	33	1	11
Shirt	155	1	120	33	92	0	580	0	19	0
Sneaker	0	0	0	0	0	12	0	947	0	41
Bag	1	2	5	2	5	2	8	5	970	0
Ankle	0	0	0	0	0	8	0	42	1	949

From the classification results of the confusion matrix, Sneaker, Bag, Ankle, Sandal and Trouser are less likely to be confused with other categories, which might because these three categories of items have more distinct shape and color characteristics; The classification accuracy of shirt, coat and pullover is low, which may be due to their similar shapes. Especially the shirt class, which is easy to be confused with the T-shirt class. This data shows that the higher the similarity, the easier it is to confuse the two types of items in the classification process.

Fixing max_depth = 20, adjust parameter n_estimators, and make statistics on the F1 score corresponding to the classification of various items. and overall accuracy rate. The results are as Table II.

TABLE II. F1 SCORE AND ACCURACY UNDER DIFFERENT N_ESTIMATORS

n_estimators type	20	50	100	200
T-shirt	0.827	0.830	0.839	0.837
Trouser	0.974	0.976	0.974	0.976
Pullover	0.770	0.782	0.777	0.788
Dress	0.878	0.887	0.889	0.886
Coat	0.782	0.787	0.789	0.791
Sandal	0.960	0.965	0.967	0.968
Shirt	0.636	0.637	0.652	0.655
Sneaker	0.930	0.938	0.936	0.936
Bag	0.960	0.965	0.966	0.965
T-shirt	0.944	0.947	0.950	0.949
Accuracy	0.868	0.873	0.875	0.876

It can be seen that with the increase of n_estimators, the overall accuracy shows an upward trend. This shows that the increase of the number of trees in the random forest within a certain range can improve the performance of the random forest. Specific to F1 score of each category, it can be seen that with the increase of n_estimators, the F1 score of Coat, Shirt and Pullover that were previously difficult to classify has shown an obvious upward trend, while the F1 score of Bag, Trouser and Sandal that had higher classification accuracy has not changed much. This shows that the over-fitting effect caused by the increase of the number of trees in the random forest is not very obvious due to the existence of its voting mechanism, otherwise the F1 score of several types of objects with higher accuracy before should be significantly reduced.

Fixing n_estimators = 20, adjust parameter max_depth, and make statistics on the F1 score corresponding to the classification of various items. and overall accuracy rate. The results are as Table III.

TABLE III. F1 SCORE AND ACCURACY UNDER DIFFERENT MAX_DEPTH

max_depth type	10	20	40	80
T-shirt	0.812	0.841	0.843	0.837
Trouser	0.967	0.978	0.976	0.977
Pullover	0.719	0.785	0.779	0.784

Dress	0.859	0.887	0.895	0.889
Coat	0.733	0.793	0.796	0.798
Sandal	0.954	0.966	0.970	0.967
Shirt	0.569	0.642	0.646	0.654
Sneaker	0.905	0.939	0.938	0.940
Bag	0.956	0.964	0.962	0.967
T-shirt	0.924	0.950	0.949	0.950
Accuracy	0.842	0.876	0.877	0.877

It can be seen that with the increase of max_depth, the overall accuracy also presents an upward trend, and this trend follows the trend gradually stabilized with the increase of max_depth. The trend of F1 score of various image categories is also similar. This indicates that as the max_depth growing in a limited range, the accuracy of classification will also be improved.

Setting n_estimators = 100, and max_depth = 50, testing the random forest model on the Cifar10 dataset, the results are as Table IV.

TABLE IV. RESULT ON CIFAR10 WITH RANDOM FOREST

Indicator type	Precision	Recall	F1-score
Airplane	0.553	0.568	0.560
Automobile	0.541	0.568	0.554
Bird	0.374	0.336	0.354
Cat	0.343	0.283	0.310
Deer	0.383	0.383	0.383
Dog	0.422	0.391	0.406
Frog	0.475	0.566	0.516
Horse	0.517	0.451	0.482
Ship	0.566	0.605	0.585
Truck	0.482	0.550	0.514

The overall accuracy is 0.470.

It can be seen that when the setting parameters are similar, the classification result of random forest on Cifar10 is far worse than that of Fashion MNIST. This is because the size of each picture in Cifar10 is 32 * 32. What's more, it is a color picture data set, which means each pixel in every image contains three dimensions of data. These factors make a picture have 3072 dimensions of data after flattening, which is much larger than 784 dimensions in Fashion MNIST. Therefore, the parameters applicable to Fashion MNIST are far from meeting the requirements of fitting for Cifar10. In other words, the random forest formed under this parameter is under-fitted for Cifar10 dataset.

C. Result of neural network

Set the neural network as four layers: the input layer is responsible for flattening the input image; The number of neurons in the first hidden layer is 300, and the activation function is relu; The number of neurons in the second hidden layer is 100, and the activation function is relu; The number of output neurons in the output layer is 10, corresponding to

ten types of input images. The activation function of this layer is softmax, which is used to limit the output results to a range of 0 to 1. The loss function is sparse categorical crossover, which can make the model learn faster when the ground truth and training results differ greatly, so as to achieve rapid convergence; The optimizer is Stochastic Gradient Descent (SGD), which randomly extracts a batch of samples from a training dataset for learning, calculates gradients and errors, and updates parameters of neurons.

Take Fashion MNIST as the training dataset, and extract a part from the training set as the verification set. The epoch of training is set as 30. During the training process, the trend of loss value, accuracy of the training set and the loss value, accuracy of the verification set are shown in the Figure 1.

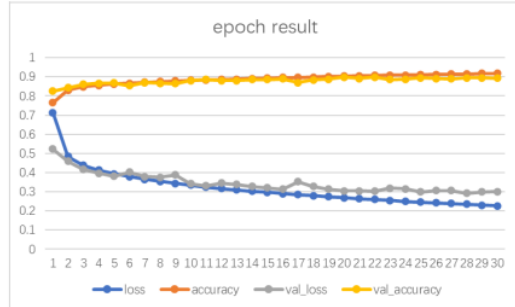


Figure 1. Loss value and accuracy change with increasing epoch on Fashion MNIST dataset

As can be seen from the Figure 1, the change of loss value of both datasets show a downward trend, while the accuracy shows an upward trend, indicating that the neural network model is indeed gradually optimized as the number of iterations increases. It can be seen that the loss value and accuracy of the training dataset show a strict upward and downward trend respectively, while the change trend of the verification set fluctuates slightly. This shows that the neural network adjusts and optimizes itself according to the loss function of the training set, so that the classification results can continuously approach the groundtruth of the training set. Once the iterative training is too much, it may lead to overfitting of the training set.

Modify the number of neurons in each layer, and the F1 score and accuracy results of classification are as Table V.

TABLE V. F1 SCORE AND ACCURACY UNDER DIFFERENT NEURON NUMBER

(L1,L2) type	(50,25)	(100,50)	(200,100)	(300,100)	(400,200)
T-shirt	0.788	0.809	0.832	0.796	0.842
Trouser	0.972	0.977	0.978	0.978	0.981
Pullover	0.773	0.783	0.791	0.79	0.807
Dress	0.862	0.843	0.883	0.884	0.877
Coat	0.765	0.797	0.81	0.805	0.816
Sandal	0.95	0.955	0.96	0.96	0.962
Shirt	0.676	0.684	0.679	0.7	0.702
Sneaker	0.932	0.938	0.937	0.941	0.944
Bag	0.949	0.963	0.969	0.968	0.969

Ankle	0.946	0.953	0.95	0.957	0.958
Accuracy	0.861	0.871	0.879	0.877	0.886

It can be seen that with the increase of the number of neurons on each layers, the F1 score and overall accuracy of each classification show an upward trend, but ultimately tend to be flat. This indicates that increasing the number of neurons per layer within a limited range can improve the feature dimensions extracted from last layer, thereby contain more informations and better fitting the original dataset. However, there are certain limitations to this improvement, especially when the epoch is fixed. Unlimited increases in the number of neurons per layer may lead to some neurons not being fully trained, which may not play a significant role in classification. This not only slows down the calculation speed, but also increases the risk of neural network overfitting.

Modify the number of layers of the neural network (add one hidden layer at back each time, and the final number of neurons in each of the five layers respectively is 300,200,100,50,25). The F1 score and overall accuracy results of the final classification results are as Table VI.

TABLE VI. F1 SCORE AND ACCURACY UNDER DIFFERENT LAYER NUMBER

Layer type	1	3	4	5
T-shirt	0.827	0.824	0.845	0.836
Trouser	0.976	0.98	0.979	0.978
Pullover	0.78	0.797	0.808	0.798
Dress	0.872	0.867	0.894	0.885
Coat	0.797	0.795	0.81	0.758
Sandal	0.958	0.967	0.967	0.963
Shirt	0.675	0.71	0.702	0.691
Sneaker	0.937	0.948	0.95	0.946
Bag	0.96	0.97	0.97	0.97
Ankle	0.954	0.957	0.959	0.96
Accuracy	0.875	0.881	0.889	0.87

It can be seen that as the number of layers increases, the ability of neural networks to fit nonlinear functions gradually increases, while the results of feature extraction also become more refined, resulting in an upward trend in overall accuracy and F1 scores for various categories' classification. However, when the number of layers is 5, the accuracy and F1 score of each layer decrease, which may be due to the poor training result of some layers, or the vanishing gradients problem.

Set the activation functions of the last layer to sigmoid and softmax respectively, and compare the F1 score and overall accuracy of the final classification results as Table VII.

TABLE VII. F1 SCORE AND ACCURACY UNDER DIFFERENT ACTIVATION FUNCTION

Function type	sigmoid	softmax
T-shirt	0.834	0.845
Trouser	0.983	0.979

Pullover	0.799	0.808
Dress	0.891	0.894
Coat	0.807	0.81
Sandal	0.965	0.967
Shirt	0.714	0.702
Sneaker	0.946	0.95
Bag	0.969	0.97
Ankle	0.953	0.959
Accuracy	0.886	0.889

It can be seen that compared to sigmoid, the F1 score and accuracy of each classification obtained by using softmax as the final activation function are relatively high. This may be due to the fact that the sigmoid function does not standardize the final result, that is, the probabilities of each classification do not add up to 1, which makes it often used in non mutually exclusive classification problems, while softmax is often used in mutually exclusive situations; At the same time, both sides of the sigmoid function tend to be flat, which easily leads to the problem of vanishing gradient problems. Finally, the softmax function can well strengthen the differences between classification probabilities through exponential mapping. These factors make softmax performs better than sigmoid in this problem's solution.

Set the hidden layer of the neural network as four layers (L1=1000, L2=700, L3=300, L4=200), and test its training results on the Cifar10 dataset. The data is as Table VIII.

TABLE VIII. F1 SCORE TESTING ON CIFAR10

Type	F1 score
Airplane	0.582
Automobile	0.657
Bird	0.425
Cat	0.318
Deer	0.438
Dog	0.441
Frog	0.574
Horse	0.600
Ship	0.636
Truck	0.578
Overall accuracy	0.529

It can be seen that the training effect is not ideal, and only about 50% of the data can be correctly classified. The preliminary assumption is that the number of neurons and layers of the neural network still do not meet the requirements for fitting, but after checking the iterative process, it is found that this is not the case. The changes in loss value and accuracy in both vaild dataset and training dataset during the iteration process are shown in the Figure 2.

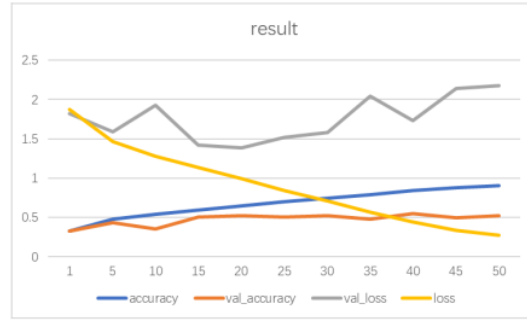


Figure 2. Loss value and accuracy change with increasing epoch on Cifar10 dataset

As can be seen from the Figure 2, when the accuracy of the training dataset constantly approaches 1, the accuracy of the vaild dataset always fluctuates around 0.5; At the same time, while the loss value of the training set continues to decline, the loss value of the vaild dataset continues to fluctuate and does not shows a significant overall trend of change. This indicates that when the neural network continuously approximates the results of the training dataset, its effect on the vaild dataset is not good. By viewing the data in the Cifar10 dataset, The problem can be found. Unlike Fashion MNIST, the images in the Cifar 10 dataset have not been cut, that is, there is a certain amount of unrelated data in one image and the location and size of the unrelated data cannot be determined. This makes it difficult for neural networks to learn the characteristics of targets and classify them. Therefore, it may be necessary to use specialized convolution layers to extract target features.

V. CONCLUSION

This article discusses two machine learning models, random forest and neural network, and tests their classification ability on the Fashion MNIST dataset and the Cifar10 dataset respectively. On the random forest model, the effects of different maximum depths and the number of trees on the classification results are tested, and found that within a certain range, the higher the maximum depth, the more trees, and the better the classification result. On the neural network model, the effects of different neuron numbers, hidden layer numbers, and activation functions on classification results are tested. It shows that within a limited range, the greater the number of neurons, the deeper the number of hidden layers, and the better the classification result. In the case of mutually exclusive classification, the softmax function is more effective than the sigmoid function. On the Cifar10 dataset, due to data characteristics, neither model can achieve good results, which may require other preprocessing methods to extract features.

VI. REFERENCES

- [1] Russell S J. Artificial intelligence a modern approach[M]. Pearson Education, Inc., 2010.
- [2] Janiesch C, Zschech P, Heinrich K. Machine learning and deep learning[J]. Electronic Markets, 2021, 31(3): 685-695.
- [3] Bishop C M, Nasrabadi N M. Pattern recognition and machine learning[M]. New York: springer, 2006.
- [4] Shinde P P, Shah S. A review of machine learning and deep learning applications[C]//2018 Fourth international conference on computing communication control and automation (ICCUBEA). IEEE, 2018: 1-6.
- [5] Learned-Miller E G. Introduction to supervised learning[J]. I: Department of Computer Science, University of Massachusetts, 2014: 3.
- [6] Ghahramani Z. Unsupervised learning[J]. Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures, 2004: 72-112.
- [7] Liu X, Zhang F, Hou Z, et al. Self-supervised learning: Generative or contrastive[J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 35(1): 857-876.
- [8] Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative Adversarial Nets[J]. stat, 2014, 1050: 10.
- [9] Wang W, Huang Y, Wang Y, et al. Generalized autoencoder: A neural network framework for dimensionality reduction[C]//Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2014: 490-497.
- [10] Kaelbling L P, Littman M L, Moore A W. Reinforcement learning: A survey[J]. Journal of artificial intelligence research, 1996, 4: 237-285.
- [11] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. nature, 2015, 518(7540): 529-533.
- [12] Song Y Y, Ying L U. Decision tree methods: applications for classification and prediction[J]. Shanghai archives of psychiatry, 2015, 27(2): 130.
- [13] Biau G, Scornet E. A random forest guided tour[J]. Test, 2016, 25: 197-227.
- [14] Ali J, Khan R, Ahmad N, et al. Random forests and decision trees[J]. International Journal of Computer Science Issues (IJCSI), 2012, 9(5): 272.
- [15] Belgiu M, Drăguț L. Random forest in remote sensing: A review of applications and future directions[J]. ISPRS journal of photogrammetry and remote sensing, 2016, 114: 24-31.
- [16] O'Shea K, Nash R. An introduction to convolutional neural networks[J]. arXiv preprint arXiv:1511.08458, 2015.
- [17] Scarselli F, Gori M, Tsoi A C, et al. The graph neural network model[J]. IEEE transactions on neural networks, 2008, 20(1): 61-80.
- [18] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
- [19] Xiao H, Rasul K, Vollgraf R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms[J]. arXiv preprint arXiv:1708.07747, 2017.
- [20] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[J]. 2009.

VI. APPENDIX

A. Codes(python)

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics

#load datasets
#fasion mnist

(fm_x_train_full, fm_y_train_full), (fm_x_test, fm_y_test)
=keras.datasets.fashion_mnist.load_data()
```

```
fm_x_valid = fm_x_train_full[:5000]
fm_y_valid = fm_y_train_full[:5000]
fm_x_train = fm_x_train_full[5000:]
fm_y_train = fm_y_train_full[5000:]
#Cifar10
(ci_x_train_full, ci_y_train_full), (ci_x_test, ci_y_test)
=keras.datasets.cifar10.load_data()
ci_x_valid = ci_x_train_full[:5000]
ci_y_valid = ci_y_train_full[:5000]
ci_x_train = ci_x_train_full[5000:]
ci_y_train = ci_y_train_full[5000:]

#randomForest
#train and test Fashion MNIST

fm_x_train_flatted = np.array(fm_x_train_full)
fm_x_train_flatted
fm_x_train_flatted.reshape(fm_x_train_flatted.shape[0],
,784)
rf_clf
=RandomForestClassifier(n_estimators=80,max_depth
=80,n_jobs=-1)
rf_clf.fit(fm_x_train_flatted, fm_y_train_full)
fm_x_test_flatted = np.array(fm_x_test)
fm_x_test_flatted
fm_x_test_flatted.reshape(fm_x_test_flatted.shape[0],7
84)
rf_pred_y = rf_clf.predict(fm_x_test_flatted)
#train and test Cifar10
ci_x_train_flatted = np.array(ci_x_train_full)
ci_x_train_flatted
ci_x_train_flatted.reshape(ci_x_train_flatted.shape[0],
3072)
ci_y_train_flatted = np.array(ci_y_train_full)
ci_y_train_flatted
ci_y_train_flatted.reshape(ci_y_train_flatted.shape[0])
rf_clf2
=RandomForestClassifier(n_estimators=100,max_depth
=50,n_jobs=-1)
rf_clf2.fit(ci_x_train_flatted, ci_y_train_flatted)
ci_x_test_flatted = np.array(ci_x_test)
ci_x_test_flatted
ci_x_test_flatted.reshape(ci_x_test_flatted.shape[0],30
72)
ci_y_test_flatted = np.array(ci_y_test)
ci_y_test_flatted
ci_y_test_flatted.reshape(ci_y_test_flatted.shape[0])
rf_pred_y2 = rf_clf2.predict(ci_x_test_flatted)
#result fasion MNIST

Confuse = metrics.confusion_matrix(fm_y_test, rf_pred_y)
report
metrics.classification_report(fm_y_test, rf_pred_y, digits
=3);
```

```

print(Confuse)
print(report)
print(metrics.accuracy_score(fm_y_test,rf_pred_y))
#output
output = open('confuse_res.xls', 'w', encoding='gbk')
output.write('confuse_res\n')
for i in range(len(Confuse)):
    for j in range(len(Confuse[i])):
        output.write(str(Confuse[i][j]))
        output.write('\t')
    output.write('\n')
output.close()
#result Cifar10
Confuse2 = metrics.confusion_matrix(ci_y_test_flatted,rf_pred_y2)
report2 = metrics.classification_report(ci_y_test_flatted,rf_pred_y2,digits=3);
print(Confuse2)
print(report2)
print(metrics.accuracy_score(ci_y_test_flatted,rf_pred_y2))

#nn
#Fashion MNIST
#buildModel

model = tf.keras.Sequential();
model.add(keras.layers.Flatten(input_shape=(28,28)))
model.add(keras.layers.Dense(300,activation='relu'))
model.add(keras.layers.Dense(200,activation='relu'))
model.add(keras.layers.Dense(100,activation='relu'))
model.add(keras.layers.Dense(50,activation='relu'))
model.add(keras.layers.Dense(10,activation='sigmoid'))#limit output to 0~1 as a probability

fm_x_train = fm_x_train/255.0
fm_x_valid = fm_x_valid/255.0
fm_x_test = fm_x_test/255.0
model.compile(loss='sparse_categorical_crossentropy',optimizer='sgd',metrics=["accuracy"])
history=model.fit(fm_x_train,fm_y_train,epochs=30,validation_data=(fm_x_valid,fm_y_valid))

```

```

score = model.evaluate(fm_x_test,fm_y_test)
print(score)
nn_y_res = model.predict(fm_x_test)
nn_y_res = np.argmax(nn_y_res,1)
#result
Confuse = metrics.confusion_matrix(fm_y_test,nn_y_res)
report = metrics.classification_report(fm_y_test,nn_y_res,digits=3);
print(Confuse)
print(report)
print(metrics.accuracy_score(fm_y_test,nn_y_res))
#Cifar10
#buildModel
model = tf.keras.Sequential();
model.add(keras.layers.Flatten(input_shape=(32,32,3)))
model.add(keras.layers.Dense(1000,activation='relu'))
model.add(keras.layers.Dense(700,activation='relu'))
model.add(keras.layers.Dense(300,activation='relu'))
model.add(keras.layers.Dense(200,activation='relu'))
model.add(keras.layers.Dense(10,activation='softmax'))#limit output to 0~1 as a probability
ci_x_train = ci_x_train/255.0
ci_x_valid = ci_x_valid/255.0
ci_x_test = ci_x_test/255.0

model.compile(loss='sparse_categorical_crossentropy',optimizer='sgd',metrics=["accuracy"])
history=model.fit(ci_x_train,ci_y_train,epochs=50,validation_data=(ci_x_valid,ci_y_valid))
score = model.evaluate(ci_x_test, ci_y_test)
print(score)
nn_y_res = model.predict(ci_x_test)
nn_y_res = np.argmax(nn_y_res,1)
#result
Confuse = metrics.confusion_matrix(ci_y_test,nn_y_res)
report = metrics.classification_report(ci_y_test,nn_y_res,digits=3);
print(Confuse)
print(report)
print(metrics.accuracy_score(ci_y_test,nn_y_res))

```

Classification using ML methods and Neural Networks

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

85 / 100

Instructor

A well structured report, generally well written too. References appropriate and well used. Related work is wide ranging and identifies relevant research and experiments, covers underlying concepts too. What about dimensionality reduction? Also a little more, please, on the datasets used and what sort of bench marks to expect based on other's attempts using ML models.

Thank you for placing confusion matrices in a readable table. Very good.

Try not to split tables at end of pages.

Results are as expected and your discussions and arguments are quite thorough. Well done.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

XUEYANG HU

0

RESEARCH-INFORMED LITERATURE

KNOWLEDGE AND UNDERSTANDING

ANALYSIS

PRACTICAL APPLICATION AND DEP.

SKILLS FOR PROFESSIONAL PRAC.