***Table of content***

1. Introduction to Back propagation neural network
   1. Back propagation
   2. Gradient-based learning method
   3. Optimization methodologies
2. Optimization 1 - Feature transformation
   1. Background and literature review
   2. Our assumptions
   3. Expertments for different scalling methods
   4. Conclusion
3. Optimization 2 – Choose of activiation function
   1. Background
   2. Literature review
   3. Our assumptions
   4. Expertments for different scalling methods
   5. Conclusion
4. Optimization 3 – Gradient descent
   1. Background
   2. Literature review
   3. Expertments for different scalling methods
   4. Conclusion
5. Recommondation
6. Refernce

***Abstract***

Neural network is an import techinial in machine learning. But to apply network and make it works for us need lots of assumptions such as feature transformations, chose of activiation funtions and the methods of gradient descent. In this report, we introduce Yann’s idea to optimize artificial neural network for above perspectives and try to evaluate each optimization methods. In feature transformation part, we compare the efficiency of standardization and normalization, based on that, we use PCA methods to drop some features and evalueate each methods again. In gradient descent, we try three methodologies of gradient descent and test the performance of each methods on sitimulated neural network. In activation function part, we deeply discussed difference between sigmoid and tanh methods.

1. **Introduction**

**1.1 BP neural network**

**Back propagation**

Backpropagation (BP) is a popular algorithm to converge the loss in a neural network. It is conceptually easy to understand and simple to realize (Yann, 1988). However, to make a BP neural network work, even efficient, need to make lots of assumptions, such as type, number of neurons, number of hidden layers, activation function for each layer, learning rate and methods of initializing features etc.

**1.2 Leraning method**

There are lots of machine learning methods, most of learning methods are gradient-based. The concept graph of gradient-based learning is presented as follow.

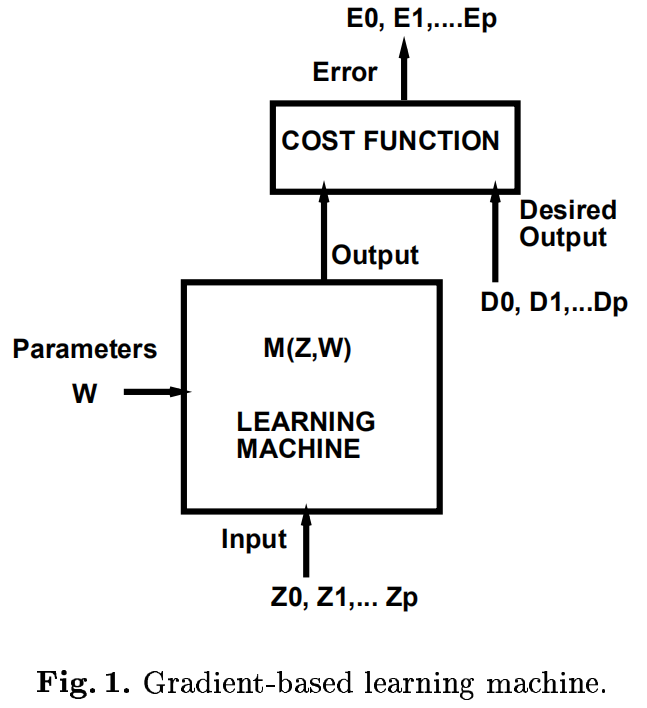


Fig.1. foraward propagation

The fact of simple neural network is a function that learns the from the input to the output, which is the dot product between weight parameters and input , denoted by , and calculate the average loss between . This process is named as forward propagation. The cost functions (sum and average for loss function) are denoted as

Above functions are one of the most cared things in machine learning problem, which is to find a minimum value for loss. One another is the time costing of doing that process. Back propagation is the algorithm to partial derivative the weights in outcome layer, back to the input layer in accordance with the activation function(s), and update the value of weight, and do forward propagation again. The network will loop the process until the loss is converged, or stopping criterial is satisfied. In this process, the network will calculate the gradient and update the weight by the gradient after multiply learning rate. However, the speed of above process is highly correlated with the magnitude and dimensions of input values, the accuracy of final performance of neural network is also affected. Thus, one important stage to optimize our network is to scale the values in a range by keeping the covariances among features at 1. Based on the work from previous scholar, the direactions of optimization seems already been limited into a specific range. In our report, we will try 3 directions to optimizing a self – established network by randomly generate dataset. These 3 direaactions are feature transformation, choose of activation function, and methodologies of gradient descent.

1. **Feature transformation**
   1. **Background and Literature review**

In Yann’s paper, he mentioned that when the mean of input variables closes to 0, the speed of convergence will be faster. This can be approved by the distribution of sigmoid activation or tanh activation function.

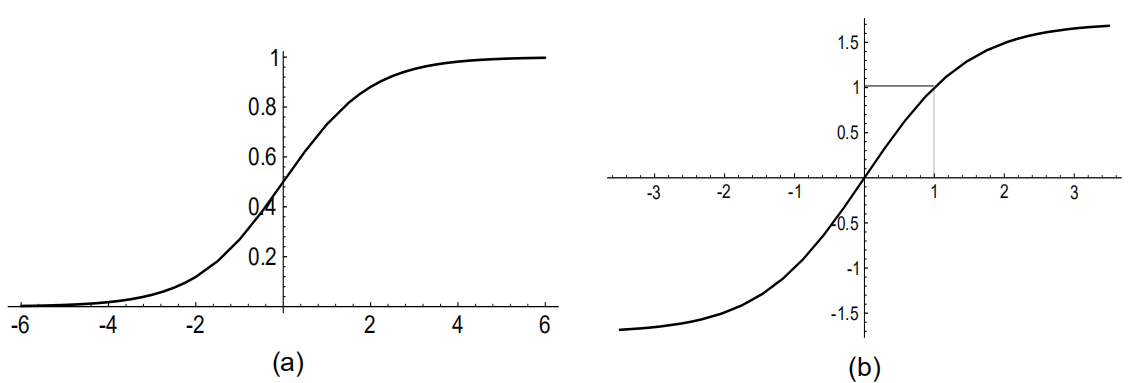


Fig2. The sigmoid and tanh

As above, when value in x-axis between -2 and 2 for sigmoid, and -1 and 1 for tanh, the gradients are the largest, but the slope outside above range begin to flatter and close to 0. That means, when we input an extremely large or small value into the activation function, the gradient would come to an extremely small value, even close to 0, thus cause slow optimization speed. This issue is usually cause by the variance of dimension between different features. For instance, if 2 features, one values between 1000 and 10000, and anther range between 1 and 10, the input will be easy to become large or small, and cause the result of activation function large or small. Therefore, normalization solutions for input are desired, to scale the value into a desirable range. This can also solute another issue for input, which is data dimension variance.

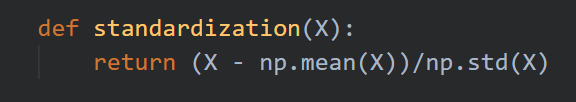
Yann mentions 3 methodologies for normalization, which are mean cancellation, decorrelation and covariance equalization, which can accelerate the speed of convergence. The next, we will try to code this method, and compare the performance of using normalization and do not use normalization, and give a short conclusion in the end. But before we begin the work, some assuptions has been made.

* 1. **Assumptions**

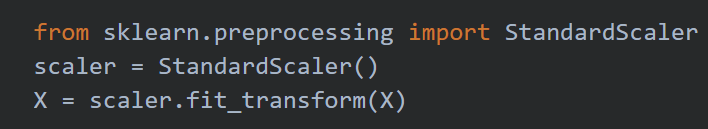
**2.2.1 Standardization method**

In nowdays machine learning field, we have multiplus methodology to cancal the mean for our dataset. One of the most popular one is standardlization. Shanker, Hu and Hung in 1996’s work summaries the advantages and also the disadvantages of data standardlization on neural network training (Shanker, Hu and Hung, 1996). The work stated, standardlization method can efficiently prevent the neural network from being large, however, the utility of standardlization woulbe be affected if the sample size goes large. To prove this, we will try to establish a simple network with 2 layers of sigmoid activiation function, and test the performance of data standardlization method. First of all, the function of standardlization is written as following

In python, we can realize the function by following code, or direactly import sklearn package:



Or by sklearn package:



**2.2.2 Normalization method**

Comparing with remove the mean from the smaple, normalization will scare the input into a stable range, such (0,1) or (-1,1). The function of normalization is expressed as

The functions of normalization are quite same with standardization, but these 2 methods gerenally will cause various performance on neural work. Jayalakshmi and Dr.Santhakumaran in 2011’s project compared various transformation methodologies for the input of artificial neural network (Jayalakshmi and Santhakumaran, 2011 ). The results indicated that MinMax scaler has a better perforamce than standardization methods.

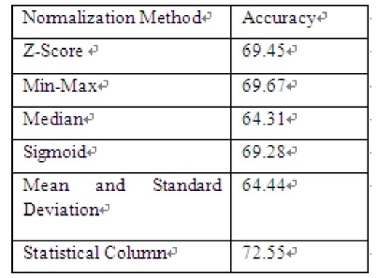


Fig3. Performance of various normalization

**2.2.3 Principal Component Analysis (PCA)**

PCA is a more complex method to process our input data. The principle of PCA is a algorithm to reduce the dimension of data by mapping the n-dimensional features to k-dimensions. The k-dimension is the new orthogonal feature, also called principal component. The job that PCA can do is sequentially calculate a set of mutally orthogonal coordinate axes, which is highly correlated with input data. Of those axes, the axes with low variance will be dropped. In the end, the lefted axes contains the most of variance for the dataset, which also contains most of features of dataset, and eventually realise the reduction of dimensions. Patra, Das and Pradhan listed the detail process of PCA, as following graph:

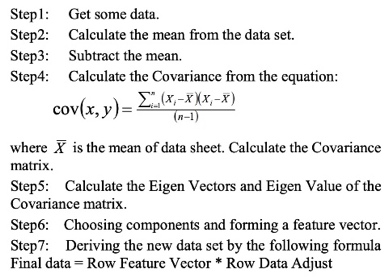
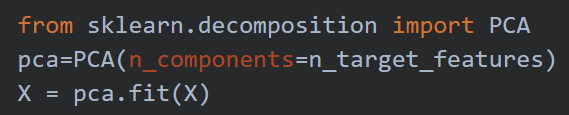


Fig 4. PCA Calculation

In the experiment stage, we will simply use sklearn package to apply the PCA to the input data, in order to reduce the dimension of data.



Patra, Das and Pradhan’s experiment already proved that, by introducing PCA to deal the dataset, the model might receive a lower loss and faster running time for model.

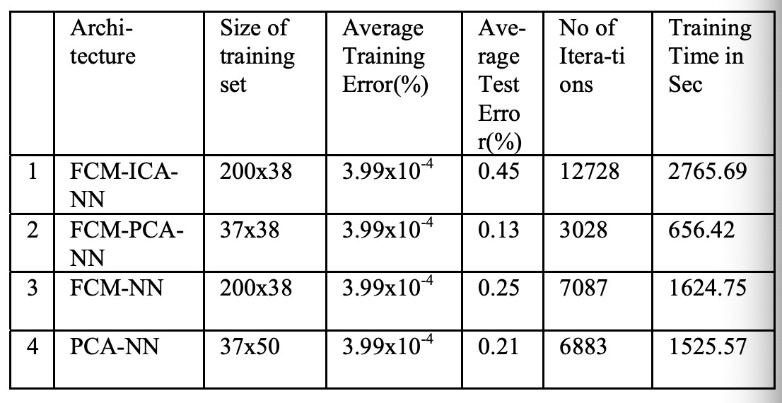
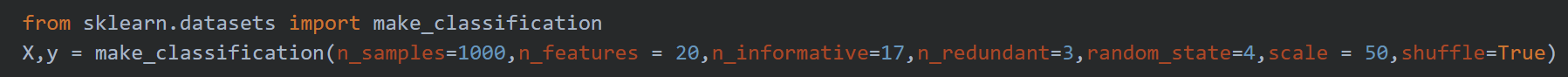


Fig 5. Copmare of PCA performance of FCM

**2.2.4 The dataset**

The sitimulate dataset we will used in this part is automatically generated by sklearn.datasets, the parameters are shown by following:



By import the package, we generate a dataset shape with (1000,20), that is 1000 samples with 20 features. Of the 20 features, 17 are informatives features, and 3 are redundant features. And all features will be expanded by 50 times, in order to keep the original dataset to a large number, so we can oberserve the benefits taken by the standardlization.

**2.2.5 The network**

To visualize the comparsion between various methodology of feature transformation, I, in this part established a simple neural network with 2 layers of sigmoid function. The idea of the code of the net is referenced from [CSDN (2016)](https://blog.csdn.net/zm714981790/article/details/51251759). The weight will be initialized as gaussian distribution, centraled with 0 and 1 standard deviation. The distribution of one column of the weight is distributed as follow.

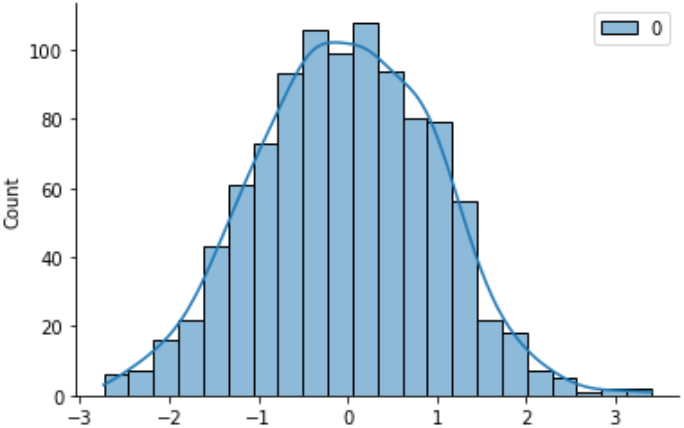


Fig 6. Weight distribution

The other required parameters in the net are learing rate, max\_iterations, stop criterial and random state. To control the parameter in same, and find the influence of feature transformation, we will keep all parameters the same for all trails. And the choosing of parameters is in following table

|  |  |  |  |
| --- | --- | --- | --- |
| Learning rate | Max\_iterations | Stop criterial | Random state |
| 2 | 50000 | 0.000001 | 6850 |

Table 1: parameters

* 1. **Experiment**

**Orginal dataset**

The distribution of all 20 features in the dataset is presented in fig 7. In order to make the influence and noise of dataset as smaller as possible, all features are normally distributed.

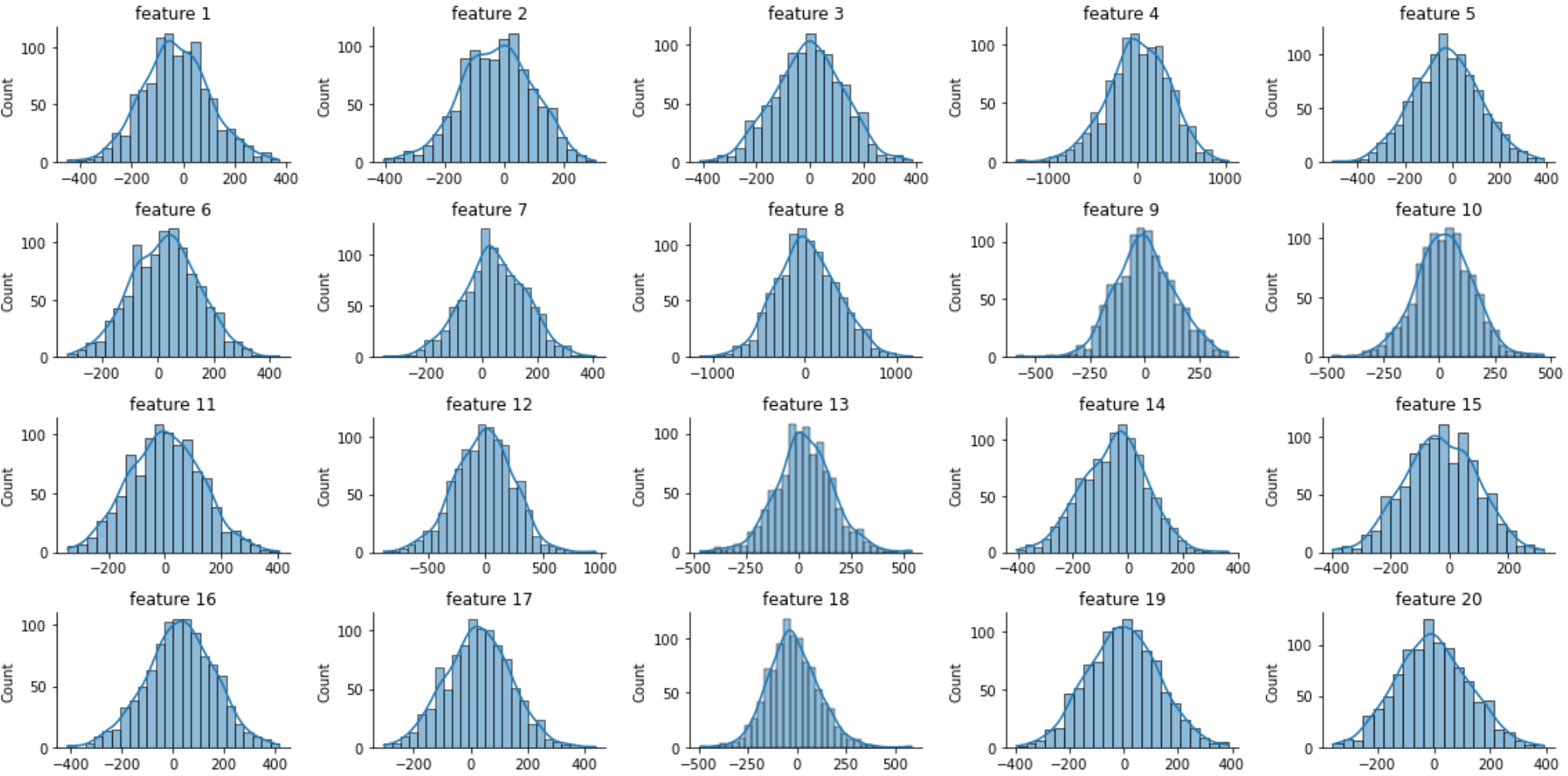


Fig 7. Distribution of all features

By not process the inputs, the performance of our net get a final cost of 0.44. the Covergence of cost is presented in fig 8.

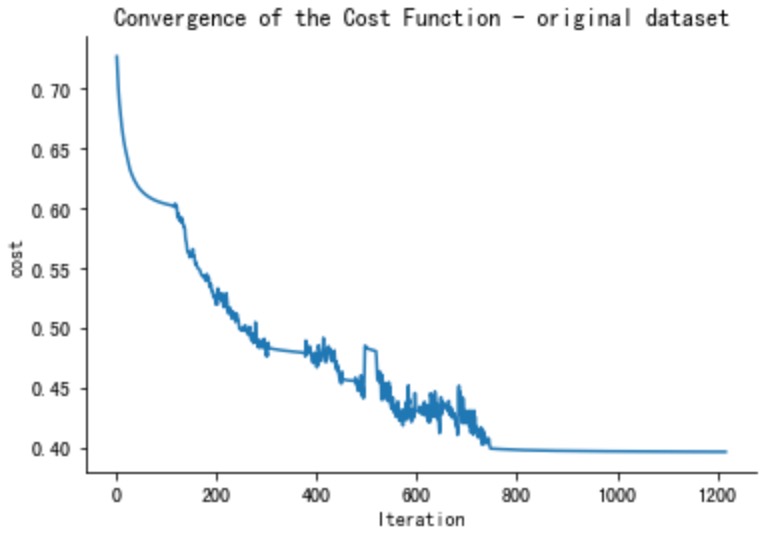
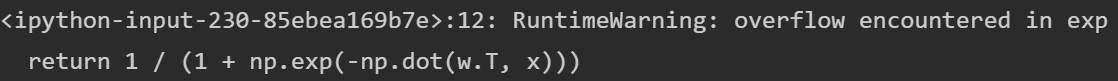


Fig 8. Convergence of cost – not process input

In the first trial, the gradient of convergence is little bit unstable, and the gradient stopped around 1200 iterations, cost 0.46 seconds to run the model and receive a final cost of 03968. In this trial, our code reminds following error. This error occurs because of the value inputed into the sigmoid function is too small, and cause too large, which cause vanishing gradient. This is the issue that we mentioned previously.



**Standardize the input**

Thus, to prevent gradient vanishing happened, we introduce Yann’s method to cancel the mean for out input data. We first apply standardization method, which will not affect the correlation between features and rescale all features into smaller range. The performance of the net now become better.

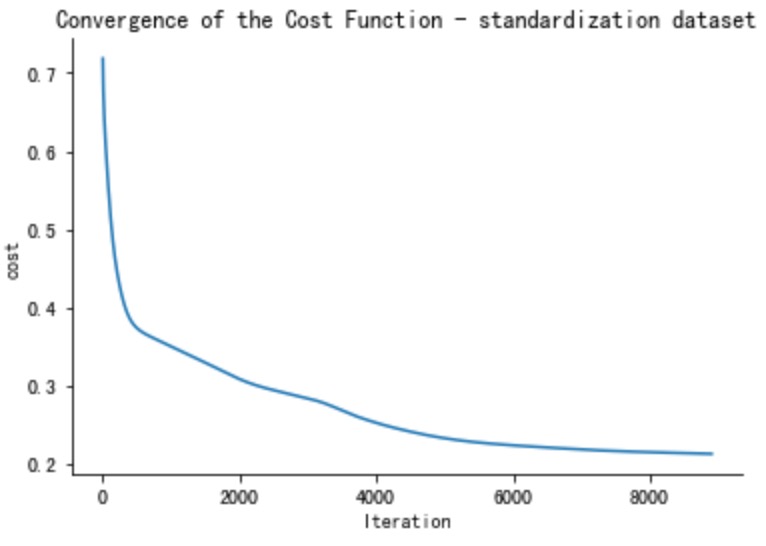
****

Fig 9. Convergence of cost – standardizsed input

Since the inputs are standardizsed, the convergence of cost is much falter, and can reach a much lower cost when stop cretiera was met. In summary, the standardlization method has a lower cost, and iterates approx. 8500 times to converge. The final cost is 0.2133 and cost 2.71 seconds to the train the input.

**Normlize input**

This methology is more time costing than standardizating, but a lower cost can be reached.

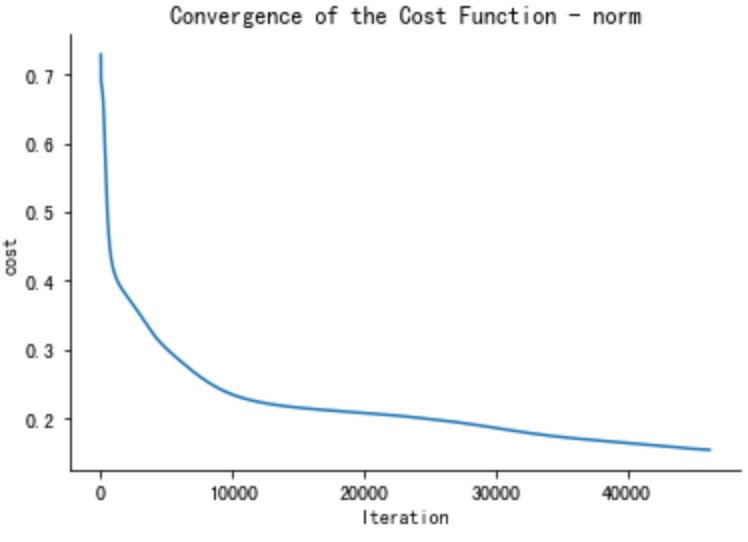
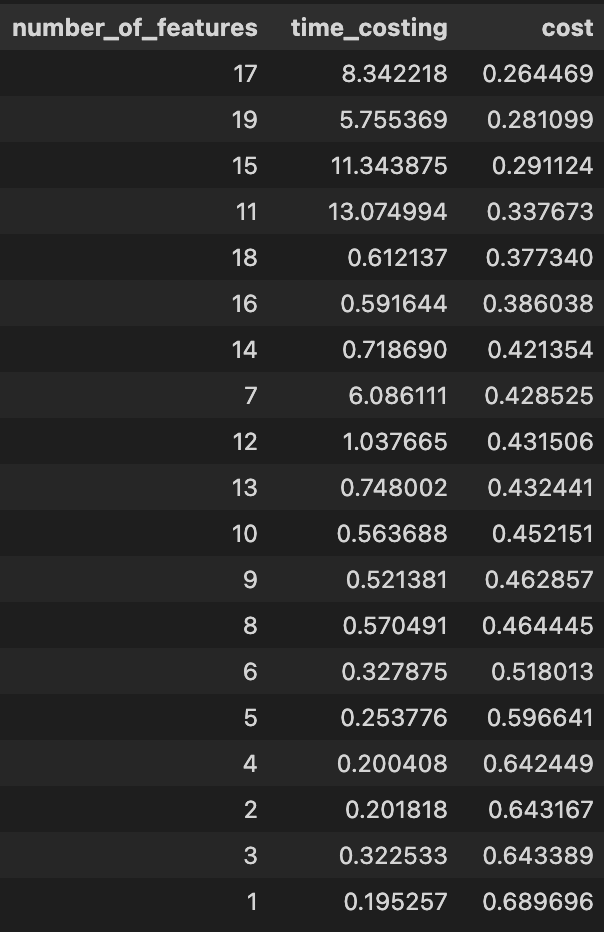


Fig 10. Convergence of cost – normlize input

By applying normlization, we eventually cost 14.75 secnonds to train the model, and receive a cost of 0.1544. However, since this method iterates too many times, which might overfit the data, thus, we will try to implement PCA to the input, to seek a much efficient to receive a better loss.

**PCA the norm**

Import PCA from decomposition in sklearn, we can receive a target dimension of input in array type. Since we have 20 features, the best fitted dimension should be 17 (since we set 17 informative features and 3 redundant features). But we still we try to loop from dimension 1 to 19, to try to find something more surprising. The summary of performance is presented in table 2

(Table 2: pca – norm performance)

From table 2, we found that 17 features are the best parameter to reach a lowest cost, the training process cost 8.3 seconds, which is much lower than 14.75 seconds. We actually can have a faster training speed, if 0.02 more punish on cost can be accepted.

**PCA the standardization**

As the same we did in normalization, the performance of pca in standardized data is shown in table 3.

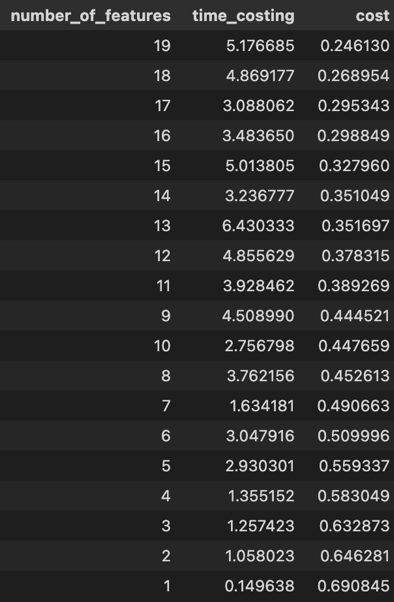


Table 3: pca – standardization performance

From the table, we find after pca, the performance of convergence speed and accuracy between standardization and normalization get closer, but standardization method still more efficient and more accuracy than normalization method. The cost, however gets higher, but, in overall, the PCA method reduces the risk of overfitting, and reduces the time costing of training model.

* 1. **Concluison**

In this part, we explore the performance of various feature transformations on neural network. These methods are popular, and widely used in field of machine learning to pre process the data. The net in this part, by introducing the feature transformation, can fit the dataset much better, by not only cancelling the gradient vananishing problem of dataset, also improve the cost and time consuming of training the model. The best transformation method in this part is the combination of standardization and PCA dimensionality reduction, and limit the number of features to 19. The final cost (mean loss) is 0.246 and will cost 5.17 seconds to train the model. However, the limitations for this party is also obverious, we listed some known limits for this part’s experiments, as following:

1. The dataset is sitimulated dataset, we did not assume too much noise.
2. The dataset is almost perfence normally distributed.
3. Parameters, such as initializing of weight, learning rate, max\_iterations etc are not selected in a strict way.
4. **Optimization 2- Choose of Activation function**
   1. **Background**

The typical artificial neural networks (ANN) are composed of different functions. In neural networks, every neuron will do two computations. Firstly, linear summation of inputs. For instance, if there are three inputs x1, x2, x3 and bias, and we have the wights of each layer w1, w2, w3. Then the linear result of summation will be y = x1w1 + x2w2 +x3w3 +bias. Subsequently, the activation function will perform diverse computations between the hidden layers and output layers. Through computing the weighted sum and further adding bias, activation computation decides which neuron should be activated (Nwankpa, Ijomah, Gachagan & Marshall, 2016).

* 1. **Literature review**

The primary purpose of the activation function is to convert linear inputs into non-linear outputs (LeCun et al., 2000). The reason why we need a non-linear activation function is that the non-linear transformation enables the neural network to be capable of learning non-linear relationships in the data. In other words, without an activation function, a neural network is essentially just a linear regression model, regardless of how many layers the network has. However, in the real world, problems are non-linear (Nwankpa, Ijomah, Gachagan & Marshall, 2016). Therefore, we use the activation function to introduce non-linearity into a neuron (LeCun et al., 2000). The non-linear outputs after application of the activation function(α) will be: y= α (x1w1 + x2w2 +x3w3 +bias).To learn the relationships and patterns in data and accordingly make effective decisions, choosing the most appropriate activation functions is essential to improve the performance of learning logarithms. There are several different types of activations functions. This report will mainly focus on sigmoid function and Tanh function.

* 1. **Experiment**

**Sigmoid Function**

The sigmoid function:

The Derivative:f′(x)=f(x)(1−f(x))

Sigmoid which is also called as logistic function and Tanh function are two common activation functions in neural networks.

Chart, line chart

Description automatically generated

Fig. 11 plot for sigmoid and tahn

According to the graph, it shows the sigmoid function range between (0,1). Also, the output is not Zero-Centered. Based on the derivative curve, the derivative is asymmetric about the y-axis and the maximum value is 0.25. Thus, the range of the sigmoid derivative is between 0 to 0.25. This means when progressing backwards each layer in backpropagation, the size of “error" will be reduced by at least 75% at each layer. The small range of the sigmoid derivative has limited the ability to change the weights in layers and leads to ineffective learning (Szandała, 2020).

Based on the graph, at the top and bottom level of sigmoid functions, the curve changes slowly, and the slope(gradients) is extremely small (close to 0), it has been shown in the derivative curve(yellow) below the sigmoid curve(blue). When the gradient = 0, there is no change in weights and no learning. The main vanishing gradient issue of sigmoid function leads to slow convergence (LeCun et al., 2000). Due to this, when the x value is small or big, the slope is zero, then the zero derivative means there is no learning. Consequently, the sigmoid activation function has fallen out of in real-world neural networks because of the vanishing gradient problem (LeCun et al., 2000).

**Tanh Function**

Function:

Derivative:

Tanh is one of the most popular activation functions in Machine Learning since it is a continuous and differential function (Szandała, 2020).

Chart, line chart

Description automatically generated

Fig 12. S-Shaped curves

As the above graph shows, both sigmoid and tanh are S-Shaped curves, sigmoid lies between 0 and 1 but tanh lies between 1 and -1. Tanh It is a wide variety of sigmoid functions (Szandała, 2020). Mean of sigmoid, tanh and their derivatives for the set of integers [-5,5] in this case can vary if the input varies. However, compared to the sigmoid function, the mean of the tanh function is always closer to zero. Furthermore, the mean of the input data is around zero for tanh because the data is centered around zero. Consequently, this zero-centered property enables the gradients are not restricted to move in certain directions (Szandała, 2020). This is an essential reason why tanh is preferred and performs better than sigmoid (LeCun et al., 2000).

Chart, line chart

Description automatically generated

Fig 13. Tanh and derivate distribution

* 1. **Conclusion**

Tanh activation function has a higher range compared to the sigmoid activation function. Meanwhile, the tanh function is symmetric about the origin, where the inputs are normalised and are more likely to create outputs (which are inputs to the following layer) and are also close to zero on average. These properties enable the Tanh function to converge faster than the sigmoid function. Therefore, Tanh activation function can effectively improve the learning of the patterns in data.

1. **Optimization 3 – Gradient descent** 
   1. **Background**

Gradient descent optimization algorithms, while increasingly popular, are often used as black-box optimizers because practical explanations of their advantages and disadvantages are difficult to obtain. (Ruder, 2016) This section aims to practice and discuss the effectiveness and performance of random, batch, and small-batch learning in gradient descent, and summarize the actual performance of each method. The only thing worth noting is that, depending on the amount of data, we need to pay attention to the accuracy of parameter updates and the time required for each run, and make trade-offs.

* 1. **Literature review**

In Yann’s paper, he only mentions two methods of gradient descent among the three mentioned above, they are stochastic gradient descent and batch gradient descent.

He listed the advantages of the two approaches and compared them. He believes that stochastic gradient descent is usually faster than batch gradient descent, which means that due to random optimization of the loss function on a certain training data rather than all training data in each iteration, stochastic gradient descent greatly speeds up the updating speed of parameters in each iteration. Second, sL's performance will be better than batch gradient descent. The stochastic gradient descent will go directly over the local optimize point to the global optimize point. But batch gradient descent will often stay at the local optimize point. As shown in the figure 14

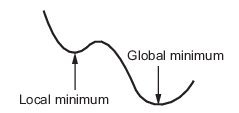


Fig 14. Gradient descent

Third, stochastic gradient descent uses one data in the whole data set to run each time, so it is easier to track the changes of changes. However, batch gradient descent uses the whole data set to run each time. It's smooth, so it's not easy to see the changes. But batch gradient descent also has its own advantages. Batch gradient descent puts the whole data set into the process of solving each time, so the Gradient update obtained is smooth, which can be seen from the loss curve quickly and conveniently.

It is worth adding that there is a third method besides batch gradient descent and stochastic dradient gescent. That is mini-batch gradient descent. Mini-batch gradient descent is a compromise between batch gradient descent and stochastic gradient descent. The idea is that each iteration uses 'batch\_size 'samples to update the parameters. It overcomes the disadvantages of the above two methods while simultaneously taking into account the advantages of both methods. Mini-batch gradient descent through matrix operations, optimizing neural network parameters on one batch at a time is not much slower than individual data. Using batch each time can greatly reduce the number of iterations required for convergence, and make the convergence result closer to the effect of gradient descent. As shown in the figure:

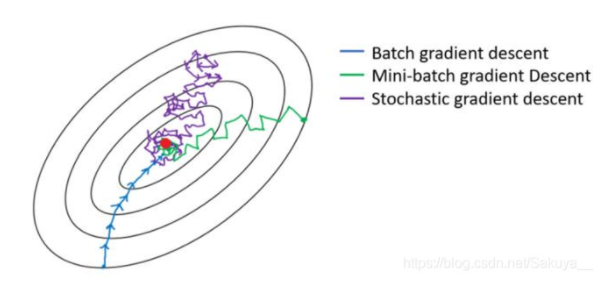
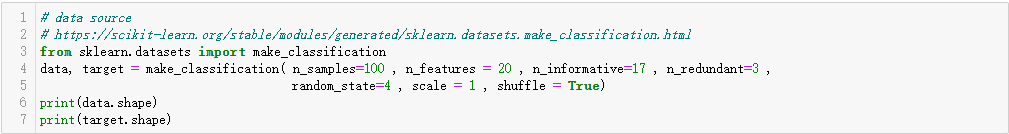


Fig 15. Three methodologies of gradient descent

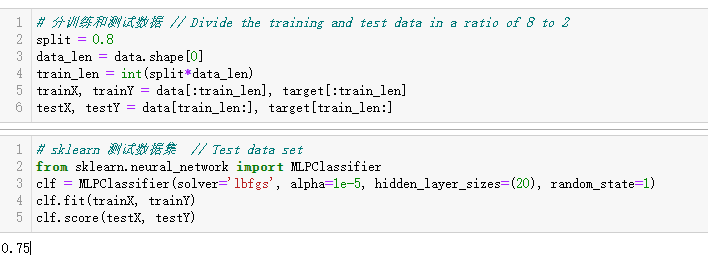
* 1. **Experiments for different scalling methods**

We will use aset of data generated by Sklearn.datasets to investigate the effectiveness and performance of random, batch, and small-batch learning.

By import the package, we generate a dataset shape with (100,20), that is 100 samples with 20 features. Of the 20 features, 17 are informatives features, and 3 are redundant features.



At the same time, in order to train and test the results, I divided the data set into a training set and test set in the ratio of eight to two, and tested the data set, as shown in the figure below:



In order to visualize the results of various gradient descent, I established a simple neural network in this part. Other parameters required in the network are learning rate, max\_iterations, and hidden. In order to control parameters to make comparisons under the same conditions and observe the performance of each gradient descent, I will keep the parameters of all gradient descent the same. For parameter Settings, see the following code blocl



The results of batch gradient descent after training and test are shown in the figure below:

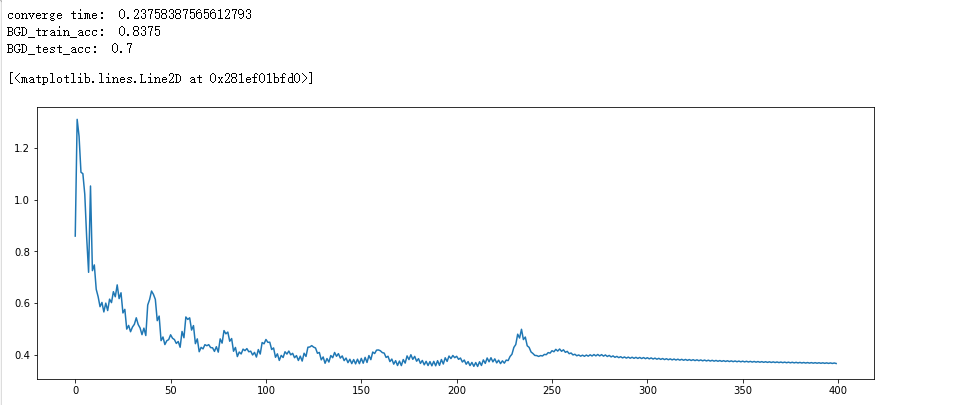


Fig 16. Batch gradient descent

The results of stochastic gradient descent after training and test are shown in the figure below:

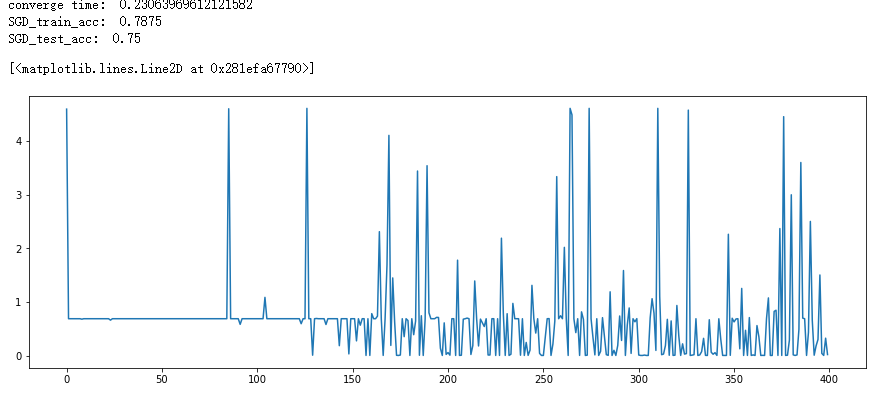


Fig 17. Stochastic gradient descent

The results of mini-batch gradient descent after training and test are shown in the figure below:

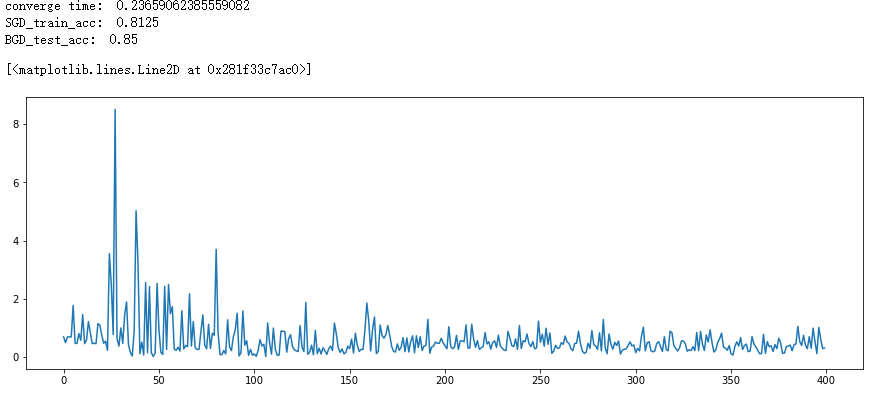


Fig 18. Mini-batch gradient descent

It is obvious that the mini-batch gradient descent has the best performance and is not much slower than the stochastic gradient descent while ensuring accuracy. However, it is worth mentioning that the stochastic gradient descent and mini-batch gradient descent randomly capture one data and part of the data for testing respectively, so the final accuracy will fluctuate due to different data, which needs to be noted. It should also be noted that the learning rate was set manually at 0.001 in this experiment. If the learning rate is too large, the gradient descent algorithm will oscillate back and forth between the upslope and downslope of the graph, and serious results may fail to converge. If the learning rate is too small, it will take a long time to train the gradient descent algorithm in the whole process of graph iteration to the optimal point, resulting in slow training, but the optimal selection can be obtained. So in the choice of learning rate or can consider.

* 1. **Conclusion**

In this part, we first verified the advantages and disadvantages of stochastic gradient descent and batch gradient descent in Y's paper, and supplemented mini-batch gradient descent to verify and obtain the results.  Mini-batch gradient descent is a good choice.

1. **Recommondations**

In this report, we try difference methods to optimiz our network. For feature transformation methods, we find standardization performs much better than normlizaiton, even the cost of standardization is a little higher than normalization. Because standardization methods is almost 2 times efficiency than normalization, and even thougn normalization has a lower cost, but there might be risk of overfitting, since its iterates more than 30k+ times. The PCA method can reduce the various between standardlization and normalization, and also can make the model much efficient.

In activiation function part, we found tanh can perform better than sigmoid, since the tanh is a sysmetric function based on 0 points and range between [-1,1]. In this area, we can get a relatively clearer gradients, thus the model can converge faster and flatte, and avoid vanishing gradient.

In gradient descent method, we found mini-batch gradient descent performs better than other 2 methods. By comparison, Using mini-batch each time can greatly reduce the number of iterations required for convergence, and make the convergence result closer to the effect of gradient descent.

1. **Reference**

Jayalakshmi, T., & Santhakumaran, A. (2011). Statistical normalization and back propagation for classification. International Journal of Computer Theory and Engineering, 3(1), 1793-8201.

LeCun, Bottou, L., Orr, G. B., & Müller, K.-R. (1988). Efficient Backprop. In Neural Networks: Tricks of the Trade (pp. 9–48). Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-642-35289-8_3>

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2016). Activation Functions: Comparison of trends in Practice and Research for Deep Learning. Retrieved 12 November 2021, from https://arxiv.org/abs/1811.03378

Patra, D., Das, M. K., & Pradhan, S. (2009). Integration of FCM, PCA and neural networks for classification of ECG arrhythmias. IAENG International Journal of Computer Science, 36(3).

Ruder, S. (2016) An overview of gradient descent optimization algorithms. <https://arxiv.org/pdf/1609.04747.pdf>

Shanker, Hu, M. ., & Hung, M. . (1996). Effect of data standardization on neural network training. Omega (Oxford), 24(4), 385–397. [https://doi.org/10.1016/0305- 0483(96)00010-2](https://doi.org/10.1016/0305-%090483(96)00010-2)

Szandała, T. (2020). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. In Bio-inspired Neurocomputing (pp. 203–224). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-15-5495-7\_11