NATIONAL CHIAO TUNG UNIVERSITY

Institute of Computer Science and Engineering

Deep Learning

Homework 1

AUTHOR: ALESSANDRO SAVIOLO

STUDENT NUMBER: 0845086

2nd November 2019

Summary

1. Introduction

The purpose of this document is to present and analyze the neural network models built from scratch for Homework 1. Furthermore, the results produced by such networks on different datasets are illustrated.

The first model analyzed is the neural network for regression. The model architecture consists of four layers. The two hidden layers are used to obtain better performances, at the cost of more computational time. It uses the ReLU activation function [3] for all layers except for the output layer (no activation function there). The neural network uses Gradient Descent [5] in order to reduce the loss function which is the Mean Squared Error.

The second model analyzed is the neural network for classification. The model architecture consists of 3 layers. This choice is related to the complexity of the dataset. It uses the ReLU activation function for all layers except for the output layer which uses the Sigmoid [2]. The neural network uses Gradient Descent in order to reduce the loss function which is the Cross-Entropy.

In Chapter 2, the neural network for regression is presented and a feature selection technique is used to achieve better performances. In Chapter 3, the neural network for classification is presented. Furthermore, the resulting learning curves and the distribution of latent features at different training stages are illustrated.

The code of the project has been uploaded to Github at the repository [6].

2. Regression

2.1 Data Preprocessing

The energy efficiency dataset is a collection of 768 instances, where 576 are used for training and 192 for testing. Each instance is defined by 2 simulation energy loads and 8 different features, namely Heating load, Cooling load, Relative compactness, Surface area, Wall area, Roof area, Overall height, Orientation (with values north, east, south, west), Glazing area, Glazing area distribution (with values uniform, north, east, south, west).

The data has been standardized to improve the numerical condition of the optimization problem and to make the training process well behaved.

2.2 Network Architecture

The network architecture is presented in Table 2.1. The choices for the input and output neurons are fixed by the task: the input layer has 16 neurons, equal to the number of features in the training set, and the output layer has 1 neuron, since we are dealing with a regression task.

The first crucial choice for designing the network architecture is to choose the number of hidden layers. One hidden layer can approximate any function that contains a continuous mapping from one finite space to another. But, in order to learn more complex representations (i.e., automatic feature selection), it is common to add one or more hidden layers to the model. Since we are dealing with a regression task and by considering the complexity of the dataset, I decided to use two hidden layers.

The second crucial choice is to choose the number of neurons for the hidden layers. Using too few neurons in the hidden layers will result in underfitting (i.e., too few neurons in the hidden layers to adequately fit the dataset and learn the task). On the other hand, using too many neurons in the hidden layers can result in several problems, such as overfitting and a substantial increase in the time it takes to train the neural network. It is necessary to find a compromise between the two. By considering the complexity of the dataset, which is rather simple, I chose to use 10 neurons for the first hidden layer and 5 for the second one.

The last crucial choice for designing the network architecture is to choose the activation function for the different layers. The output layer activation function is fixed by the regression task (i.e., linear activation function). The activation function used in the two hidden layers is ReLU, since it provides faster training process and convergence. Furthermore, it overcomes many problems introduced by the other well-known activation functions such as Sigmoid (e.g., the problem of vanishing gradient).

Layer	# Neurons	Activation
Input	16	
Hidden_1	10	ReLU
Hidden_2	5	ReLU
Output	1	

Table 2.1: Neural network architecture.

2.3 Feature Selection

I use the Pearson correlation coefficient [4] to find which input features influence the energy load the most. Such coefficient is used in statistics to measure how strong a relationship is between two variables. The Pearson correlation coefficient between two variables is obtained by the covariance of the two variables divided by the product of their standard deviations.

In descending order, the sorted features using the Pearson correlation coefficient are:

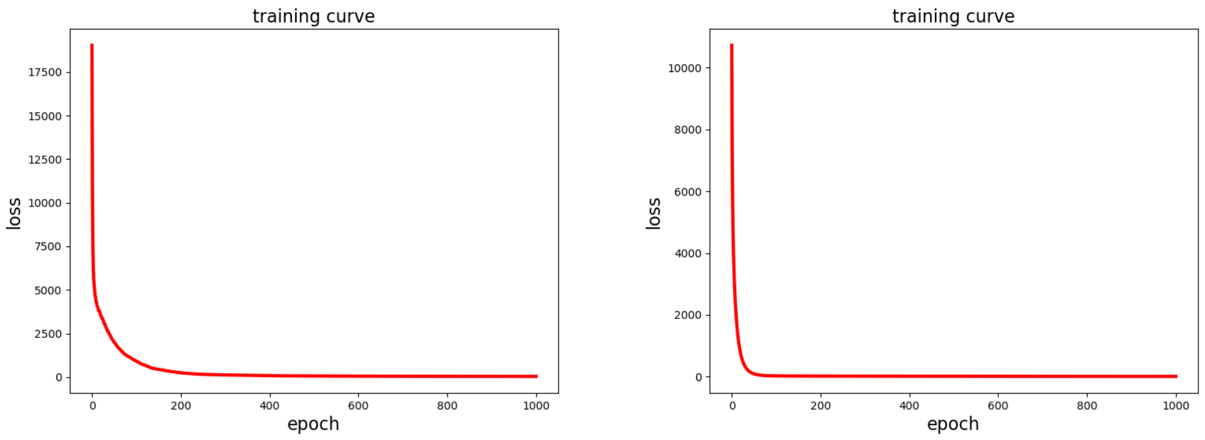
1. Cooling Load
2. Overall Height
3. Roof Area
4. Surface Area
5. Relative Compactness
6. Wall Area
7. Glazing Area
8. Glazing Area Distribution uniform
9. Glazing Area Distribution north
10. Glazing Area Distribution south
11. Glazing Area Distribution east
12. Glazing Area Distribution west
13. Orientation east
14. Orientation south
15. Orientation west
16. Orientation north

As expected, the Orientation, the Glazing Area Distribution and the Wall Area are the features that less influence the energy load. By dropping these features, the results obtained by the model (Section 2.2) can be further improved (see and compare the results in Table 2.2).

2.4 Results

Error Type	Feature Selection	Error Value
Training MSE	All features	37.383
Training MSE	Top 5 features	9.254
Training RMS	All features	0.251
Training RMS	Top 5 features	0.144
Test RMS	All features	0.126
Test RMS	Top 5 features	0.264

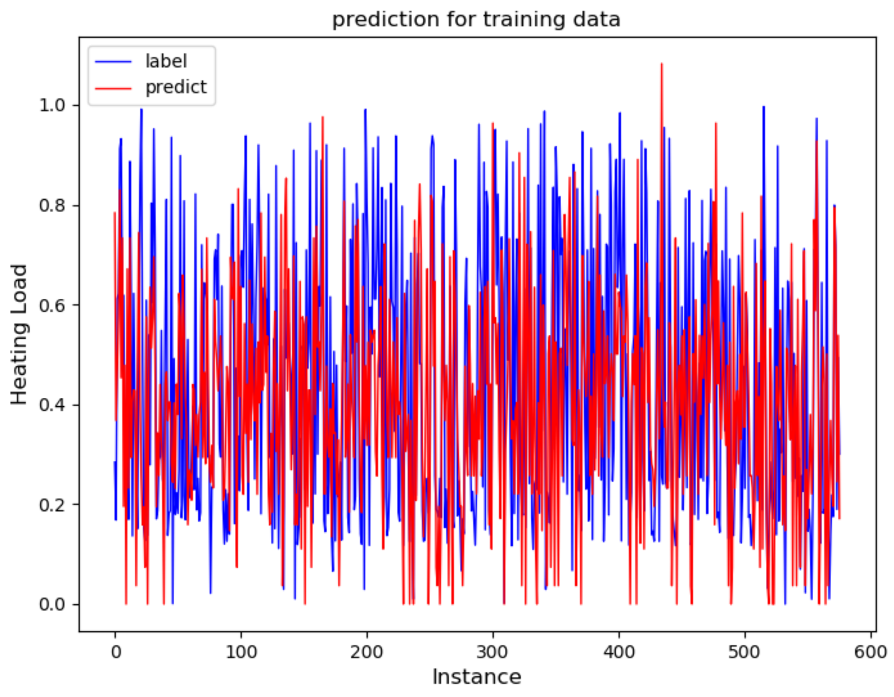
Table 2.2: Results produced by training the neural network for 1000 epochs and using 0.000001 as learning rate. The results show that the training and the test errors are significantly decreased when using feature selection. This fact is due to the features in the dataset which influence the target feature differently.



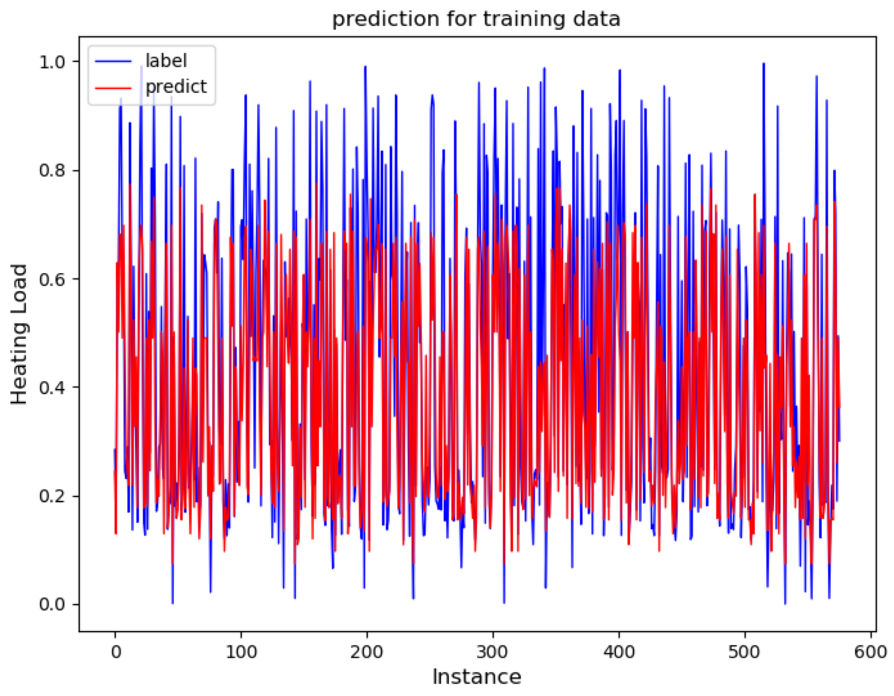
(a) All features

(b) Top 5 features

Figure 2.1: Loss curve produced by the model after training for 1000 epochs and using 0.000001 as learning rate. Feature selection makes the model converge faster (i.e., less epochs).

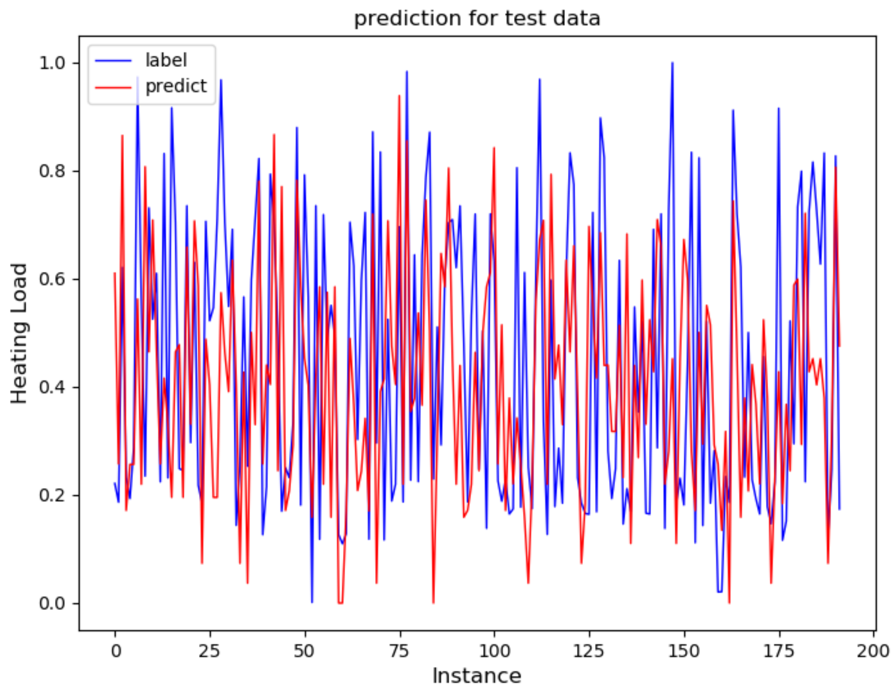


(a) All features

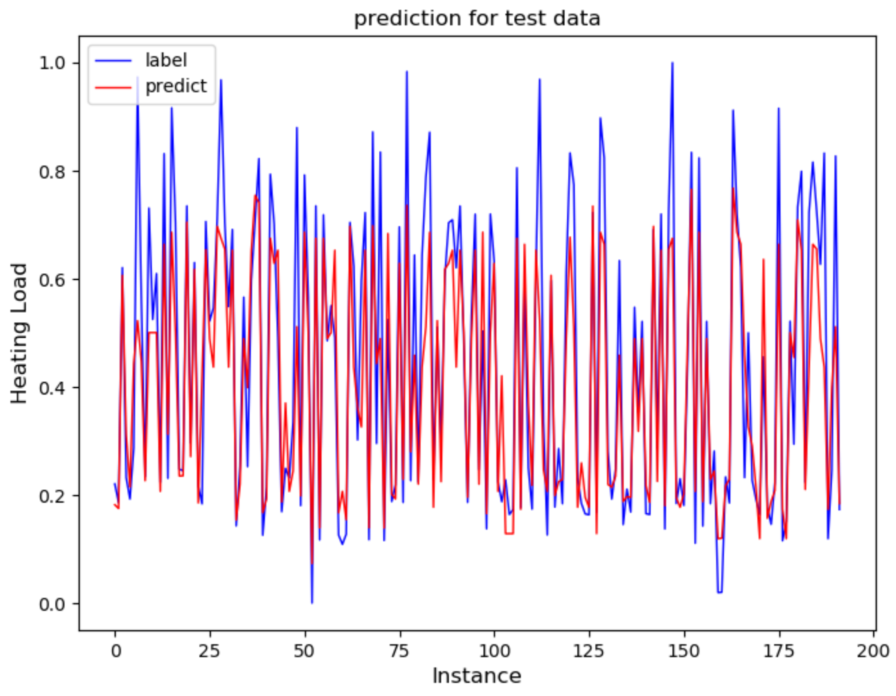


(b) Top 5 features

Figure 2.2: Prediction for training data after training for 1000 epochs and using 0.000001 as learning rate.



(a) All features



(b) Top 5 features

Figure 2.3: Prediction for test data after training for 1000 epochs and using 0.000001 as learning rate. Feature selection makes the model more robust and capable of better generalizing. As a matter of fact, the predictions on the test data are more accurate.

3. Classification

3.1 Data Preprocessing

The ionosphere dataset is a collection of 351 instances, where 281 are used for training and 70 for testing. Each instance belongs to a class (i.e., good, bad) and is defined by 34 different features.

The data has been standardized to improve the numerical condition of the optimization problem and to make the training process well behaved.

3.2 Network Architecture

The network architecture is presented in Table 3.1. The choices for the input and output neurons are fixed by the task: the input layer has 16 neurons, equal to the number of features in the training set, and the output layer has 1 neuron, since we are dealing with a binary classification task.

The number of hiddens layers is one, due to the low complexity of the dataset and to speed up the computation (i.e., using 4000 epochs for training instead of the 1000 used for the neural network for regression - Section 2.2). The number of nodes in the hidden layer is 15.

Layer	# Neurons	Activation
Input	34	
Hidden	15	ReLU
Output	1	Sigmoid

Table 3.1: Neural network architecture.

3.3 Results

Cross-Entropy Error	72.072
Training Accuracy	91.428
Test Accuracy	90.281
Training Error	0.857
Test Error	0.972

Table 3.2: Results produced by training the neural network for 4000 epochs and using 0.0005 as learning rate. The training and test accuracies are expressed in percentual.

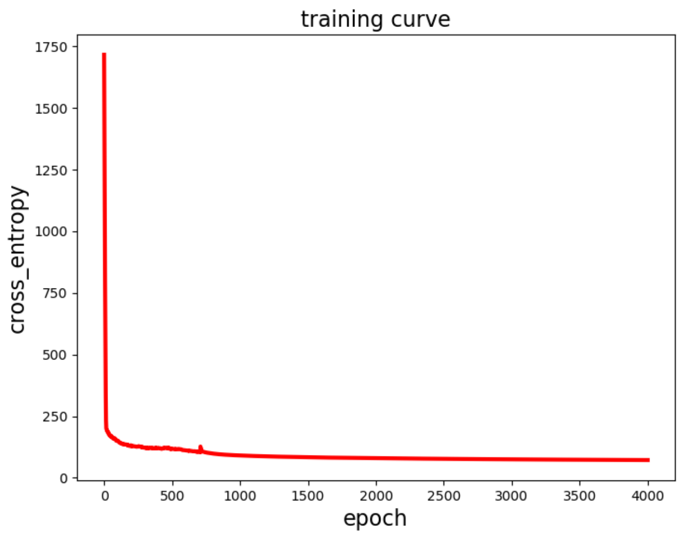


Figure 3.1: Loss curve produced by the model trained for 4000 epochs using 0.0005 as learning rate.

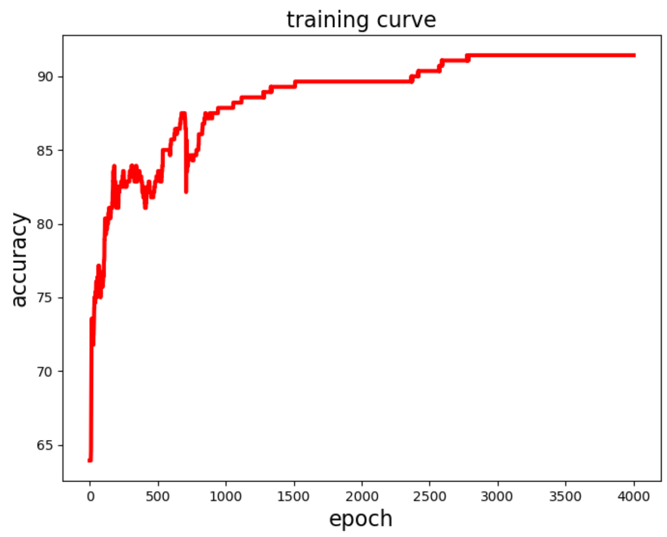
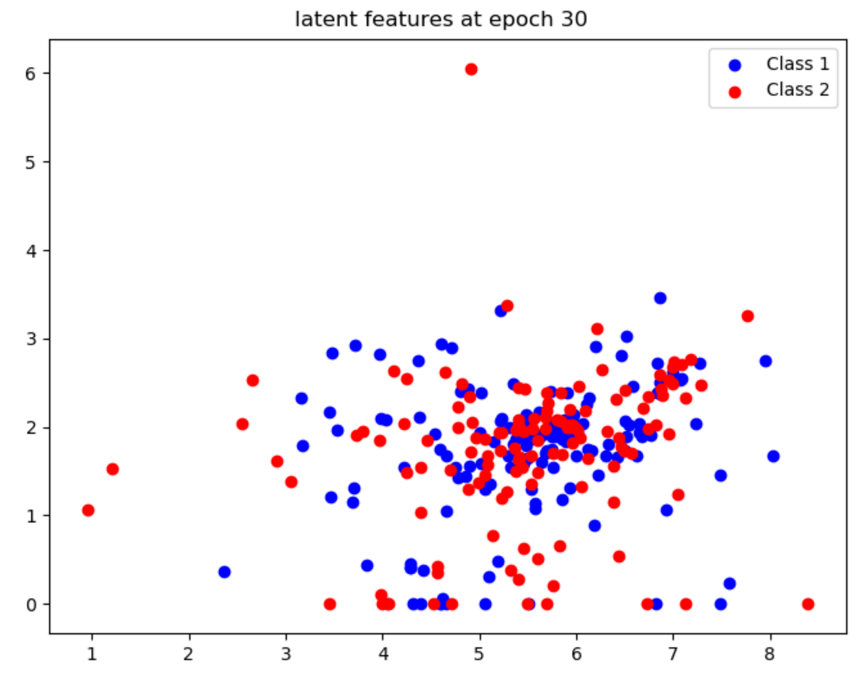
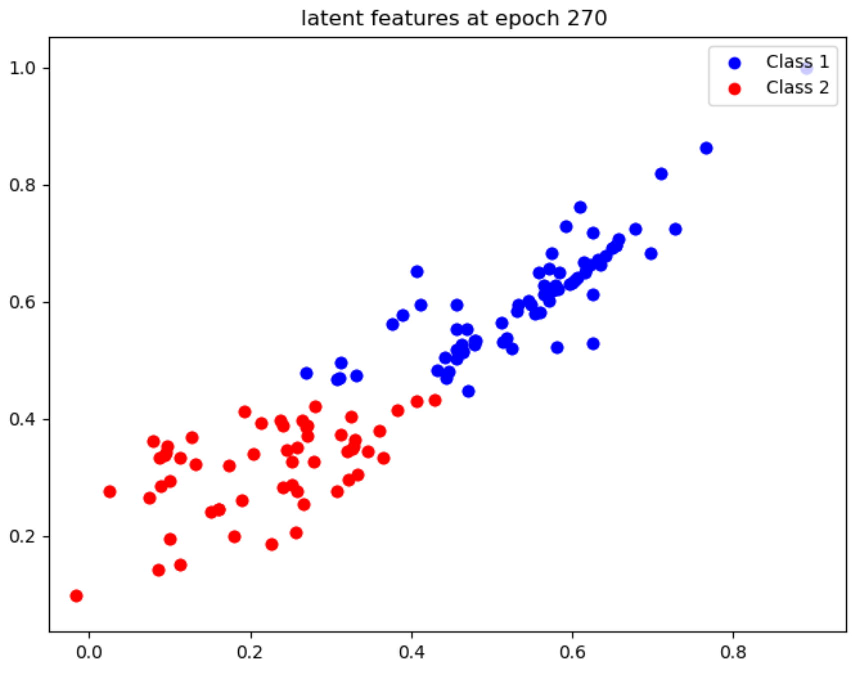


Figure 3.2: Accuracy curve produced by the model trained for 4000 epochs using 0.0005 as learning rate.

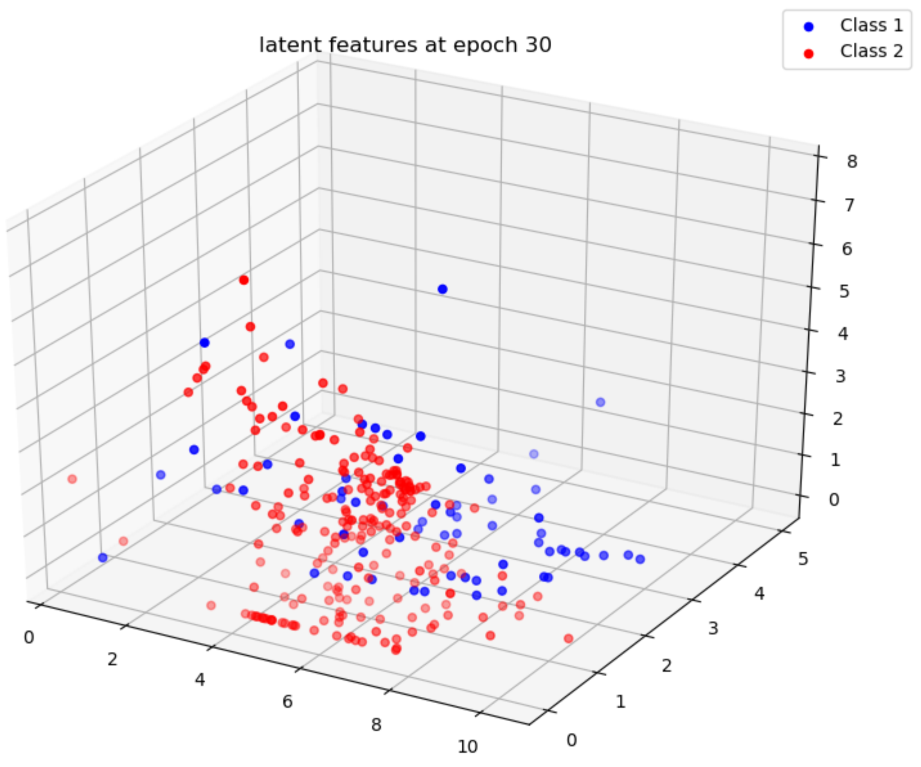


(a)

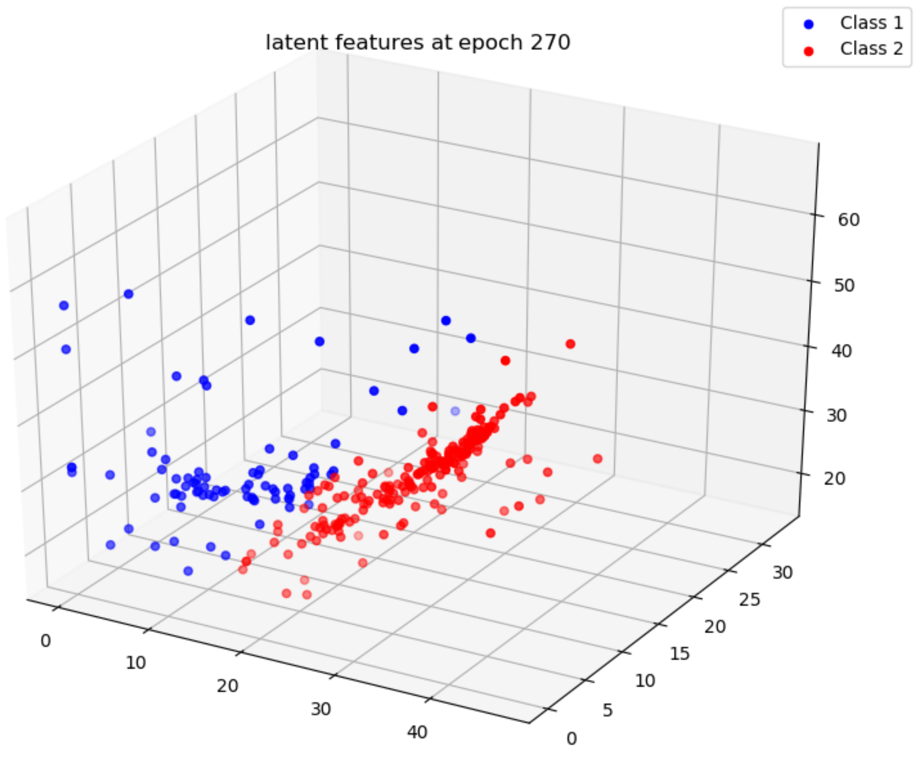


(b)

Figure 3.3: Distribution of latent features after 30 (*a*) and 270 (*b*) epochs of training using 0.00001 as learning rate. The neural network architecture has 3 hidden layers with, resp., 10, 10 and 2 neurons. After 30 epochs of training the model has not learned the classification task yet, as a matter of fact the two classes are spread randomly in the feature space. After 270 epochs of training the model is now capable of separating the two classes, mapping them in two separate groups. The choice of adding two hidden layers to the model presented in Section 3.2 is due to the fact that for plotting the 2-dimensional distribution of latent features the last hidden layer must have 2 nodes only. Then, a single hidden layer with 2 neurons would have made the neural network incapable of learning the classification task (i.e., underfitting).



(a)



(b)

Figure 3.4: Distribution of latent features after 30 (*a*) and 270 (*b*) epochs of training using 0.00005 as learning rate. The neural network architecture has 3 hidden layers with, resp., 10, 10 and 3 neurons. After 30 epochs of training the model has not learned the classification task yet, as a matter of fact the two classes are spread randomly in the feature space. After 270 epochs of training the model is now capable of separating the two classes, mapping them in two separate groups. The choice of adding two hidden layers to the model presented in Section 3.2 is due to the fact that for plotting the 3-dimensional distribution of latent features the last hidden layer must have 3 nodes only. Then, a single hidden layer with 3 neurons would have made the neural network incapable of learning the classification task (i.e., underfitting).

APPENDIX

A.1 GitHub Repository

The code of the project is available on GitHub, in the repository [6].

Bibliography

[1] Bottou L., and Bousquet O. *The Tradeoffs of Large Scale Learning.* Optimization for Machine Learning. MIT Press, pp. 351–368, Cambridge, 2012.

[2] Han J., and Morag C. *The influence of the sigmoid function parameters on the speed of backpropagation learning.* From Natural to Artificial Neural Computation. Lecture Notes in Computer Science, pp. 195–201, 1995.

[3] Nair V., and Hinton G. *Rectified Linear Units Improve Restricted Boltzmann Machines.* ICML. 2010.

[4] Stigler S. *Francis Galton's Account of the Invention of Correlation.* Statistical Science. pp. 73–79, 1989.

[5] Taddy S. *Stochastic Gradient Descent.* Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions. McGraw-Hill, pp. 303–307, New York, 2019.

Sitography

[6] Github Repository, last access 2 Nov. 2019, [Online].
https://github.com/AlessandroSaviolo/DeepLearning_0845086_HW1