2020-08-31
David Dashti
Filip Söderquist

Lab 1 - Report

Task 1) *Run the code and interpret the results. Please note the output of the model is the predicted of class labels of each of the four points. If you run the code several times, will you observe the same results? Why? Keep the parameter "n_unit=1" and increase the number of iterations starting from 10, 50, 100, 500, 2000, and compare the loss values. What can you conclude from increasing the number of iterations? Now, with a fixed value of "iterations = 1000", increase the parameter "n_unit" to 2, 5, 10 and interpret the results.*

Answer) No, you will not observe the same results every time. This is because the weights are initialized randomly. The predicted values will also improve with an increased number of iterations. Our conclusion is thus that increasing the number of iterations improves the performance of the network. Lastly, the performance of predicted values improves, with an increasing number of neurons.

Task 2) *Repeat task1 for XOR logic operator. For fixed values of parameters (iterations=2000, and n_unit=1), which of the AND or XOR operators has lower loss values? Why? Increase the number of neurons in the hidden layer (n_unit) to 2, 5, 10, 50. Does increasing the number of neurons improve the results? Why?*

Answer) The AND example has lower loss than the XOR example. This is because the classes for the XOR cannot be separated by a single line. You require at least two neurons to be able to perform the classification properly. Thus, in accordance with this argument, the performance improves when increasing the number of neurons in the hidden layer. This is again, is because you need at least two lines to be able to perform the classification task.

Task 3) *In the above code, change the parameter "n_unit" as 1, 2, 4, 16, 25, 50, and interpret the observed results.*

Answer) The performance of the network does not improve with different amounts of neurons when the learning rate is set to: lr = 0.01 and the epochs is set to 2000. However, the values do improve slightly when increasing the amounts of epochs. This led us to try an increased learning rate of: lr = 0.1 and this drastically improved the results. Especially when changing the amounts of neurons from 1 to 10.

Task 4) *Develop a 4-layers MLP. If you call the number of neurons in the first fully-connected layer as "base_dense", this 4-layers MLP should contain "base_dense", "base_dense//2", and "base_dense//4" as the number of neurons in the first 3 layers respectively. The activation function of all those neurons should set as "Relu". However, in the last layer (4th layer), choose a proper number of neurons as well as activation function(s) that fit the binary classification task. How do you interpret the observed values of loss and accuracy values? Is the number of epochs enough to make a good decision about model performance? For the same number of epochs, reduce the learning rate parameter to 0.1 and interpret the results.*
Now increase the number of epochs to 150 with LR=0.0001. Does this model have enough capacity to yield acceptable results? Increase the "base_dense" parameter to 256 and compare the results with the case of "base_dense=64". Is increasing the model capacity helpful to improve the model performance? Why?

Answer) The way to interpret the loss and accuracy values is that the closer the loss is to zero (cross-entropy loss) the better the score. Intuitively, the higher the accuracy is, the better the model. There is however a small probability that a lower loss will lead to a lower accuracy, but this is not the general case.

When choosing the setting of 50 epochs, the results are poor. This is regardless if the learning rate is: lr = 0.1 or lr = 0.0001. The binary accuracy stays around 50 % with learning rate of: lr = 0.1, with both 50 and 150 epochs. Which means that the model is basically guessing at random. Thus, number of epochs is not enough to make a good decision of model performance.

2020-08-31
David Dashti
Filip Söderquist

Increasing base_dense from 64 to 256, has no major effect on the network with number of epochs = 150 and learning rate: lr = 0.0001. At base_dense = 64 we get the values: val_loss = 0.6167 and val_accuracy = 0.715. At base dense = 256 we get the values: val_loss = 0.6242 and val_accuracy = 0.72. We deem the model itself to be inaccurate, nonetheless. Thus, changing the complexity of the architecture does not improve the performance of the network. Loss remains high and the accuracy low.

Task 5 A) *Set the following parameters: # of epochs = 20; batch size = 8; number of feature maps at the first convolutional layer = 32 and learning rate = 0.00001 and then run the experiment. What are the value training and validation accuracies? What can you infer from the learning curves? Is it reasonable to decide based on this set-up of the parameters?*

Answer) We get a training_accuracy = 0.711 and a validation_accuracy = 0.76. The learning curve is still going downwards and has not reached its point of convergence. Yes, it is reasonable to conclude since learning rate and the number of epochs is very low.

Task 5 B) *Leave all the parameters from the previous task unchanged except for the n_epoch = 200. Compare the results with task 5A.*

Answer) With the above settings we get a training_accuracy = 0.865 and a validation_accuracy = 0.815. This is an obvious improvement from the results in Task 5 A).

Task 5 C) *Keep all parameters from the last step except for the LR = 0.0001. Run the experiment with new LR and interpret the results. How do you evaluate the generalization power of the model? What are the values of training and validation accuracies?*

Answer) With the above settings we get a training_accuracy = 1 and a validation_accuracy = 0.765 after 200 epochs. The model is overfitted which can be seen in the graph produced. The training accuracy is decreasing whist the validation accuracy is increasing, thus overfitting. With a validation_accuracy of 0.765, we deem the generalization power of the model to be poor. The best results were achieved at 29 epochs with a validation_accuracy = 0.805.

Task 5 D) *What is the role of the first two convolutional layers?*

Answer) The role of the first two convolutional layers is to do feature extraction of the skin images. Generally, this is the setup of Convolutional Neural Networks

Task 5 E) *What is the role of the last two dense layers?*

Answer) The role of the last two fully connected layers is to do classification based on the features that are extracted from the first two convolutional layers. The last layer in specific is a one neuron layer with a sigmoid function that has the purpose of generating a probability of a binary yes/no answer to the classification task

Task 5 F) *What is the major difference between the LeNet and MLP?*

Answer) The major difference between the two is that LeNet is a CNN that utilizes convolution and convolution layers along with fully connected layers do perform classification whilst MLP is a straight up Artificial Neural Network which only uses fully connected layers and weight summations.

Task 5 G) *Look at the last layer of the network. How should we choose the number of neurons and the activation function of the last layer?*

Answer) Like mentioned in task 5E, the last layer should have only one neuron and a sigmoid function as activation function to be able to generate a binary output to the classification task.