Pezhman Lamei

Assignment # 1

Abstract

Doing this project shows us how we can handle overfitting in neural networks. We can do that by four methods. we use three of them here.

- Get more training data.
- Reduce the capacity of the network. We use this method in this project by reducing hidden layers or units.
- Add weight regularization. We use this method in this project
- Add dropout. We use this method in this project

The conclusion is that we can manage the model's capacity by reducing or increasing the number of hidden layers or units. A model with more capacity can model more different types of functions and may be able to learn a function to sufficiently map inputs to outputs in the training dataset. Whereas a model with too much capacity may memorize the training dataset and fail to generalize or get lost or stuck in the search for a suitable mapping function. we can think of model capacity as control over whether the model is likely to underfit or overfit a training dataset. In this project, a smaller number of layers and units enhance the model regarding overfitting and accuracy which shows that the model's complexity and data fit more with lower capacity. As well as previous literature, regularization and dropout prevent overfitting problem and enhances accuracy.

The below table presents the results of different methods of handling overfitting by providing validation accuracy.

Model	Accuracy	Conclusion
Two hidden layers with 16 units	0.88015	Accuracy for one layer is better than for two or three models and increasing the number of layers does not associate with accuracy. Having one layer, we see that validation loss increases after 4 epochs, whereas this happens for 2-3 layers after 3 epochs. This shows that for two- or three-layers overfitting happens sooner than for one layer.
1 layer with 16 units	0.88208	
3 layers with 16 units	0.87643	
2 layers with 8 units	0.8885	Increasing the units does mitigate the model's accuracy and validation loss (accuracy drops to 87%, and we have
2 layers with 32/64 units	0.8768	

		overfitting after two epochs). Additionally, we can observe that using fewer units (8 units) is a little bit better regarding accuracy and validation. This means that we can use 8 or 16 units as the required units of layers.
Using MSE	0.87440	Cross entropy fits better than MSE which is more compatible with linear regression problems. The theory behind this is that in a classification problem like this project, we have a specific set of output and we need a loss function that penalizes the error. The Cross entropy function does that rather than MSE.
Using Tanh	0.8632	Because Tanh activation is more compatible with recurrent networks like time-series or sequential data, while Relu is appropriate for multilayer perceptions. Thus, in this project, the accuracy goes down by using Tanh function.
2 layers with regularization and dropout	0.88756	The improved model is useful to prevent overfitting from 4 epochs to 7 epochs. As well as validation, the accuracy increases a little bit by including regularization and dropout.

1 & 2 | Running with a few or more layers/ Adding or decreasing units

A deep learning network has two dimensions depth (the number of layers) and width (the number of units). These dimensions control the capacity of the model. A model with less capacity may not be able to sufficiently learn the training dataset. A model with more capacity can model more different types of functions and may be able to learn a function to sufficiently map inputs to outputs in the training dataset. Whereas a model with too much capacity may memorize the training dataset and fail to generalize or get lost or stuck in the search for a suitable mapping function. we can think of model capacity as control over whether the model is likely to underfit or overfit a training dataset.

<u>Adding or decreasing layers (Depth)</u>

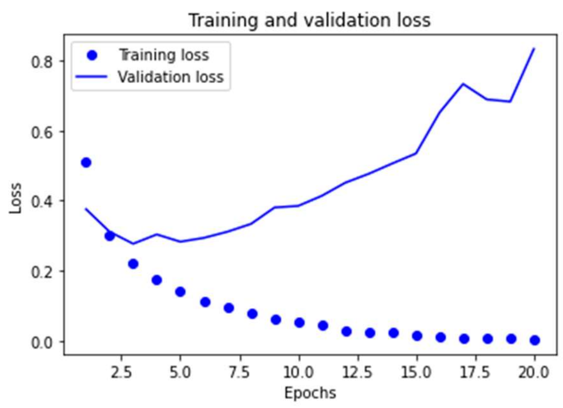
Comparing the results of adding or decreasing the number of hidden layers shows that the accuracy for one layer is better than for two or three models and increasing the number of layers does not associate with accuracy. Having one layer, we see that validation loss increases after 4 epochs, whereas this happens for 2-3 layers after 3 epochs. This shows that for two- or three-layers overfitting happens sooner than for one layer.

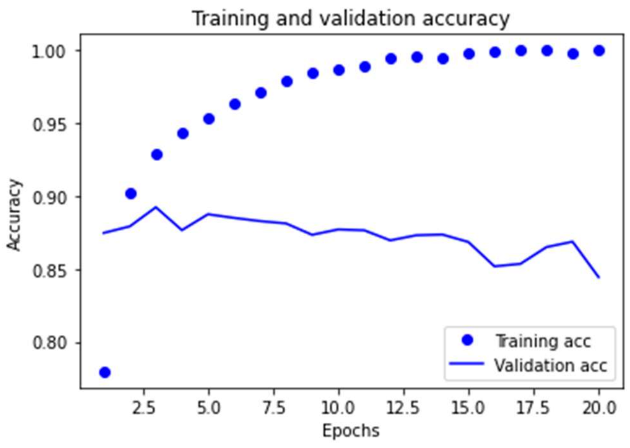
Increasing the number of hidden layers may or may not enhance accuracy, depending on how complicated the issue you are trying to solve is. Accuracy in the test set will decline when the number of hidden layers is increased above a critical threshold. Our network will learn the training data but overfit the training set. As a result, making it is unable to generalize to new, untried data.

The bottom line is that the appropriate number of layers depends on the complexity of the problem and the structure of the data (how much training data we have). For example, if the appropriate number of layers is two layers, using one layer gives us the underfitting problem, and using more hidden layers than 2 layers brings an overfitting problem.

<u>Results</u>

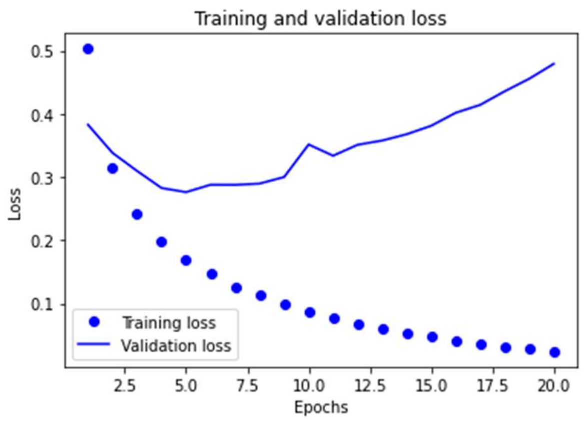
<u>For two hidden layers:</u>

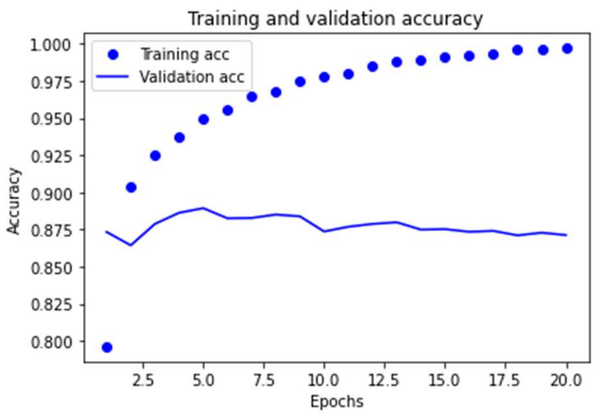




[Loss, accuracy] = `[0.30282720923423767, 0.8801599740982056]`

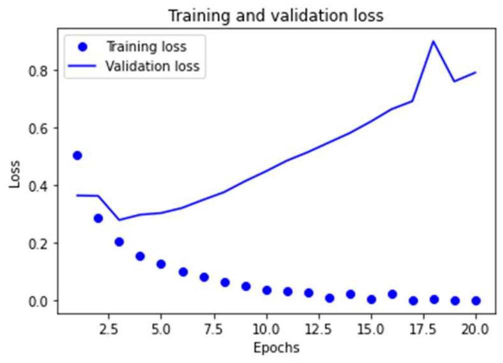
<u>For one hidden layer:</u>

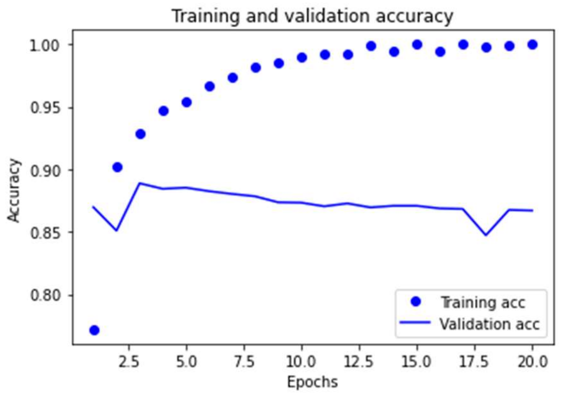




[Loss, Accuracy] = `[0.2894096374511719, 0.8820800185203552]`

<u>For three hidden layers:</u>



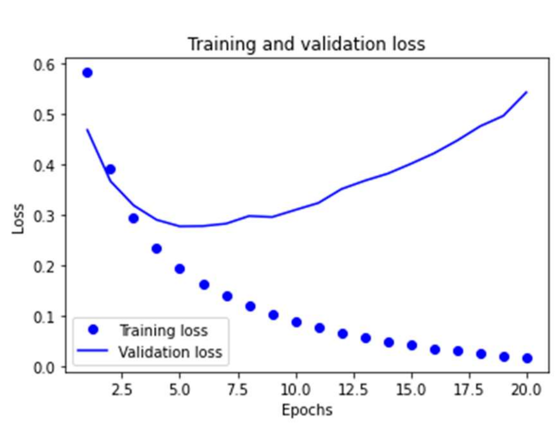


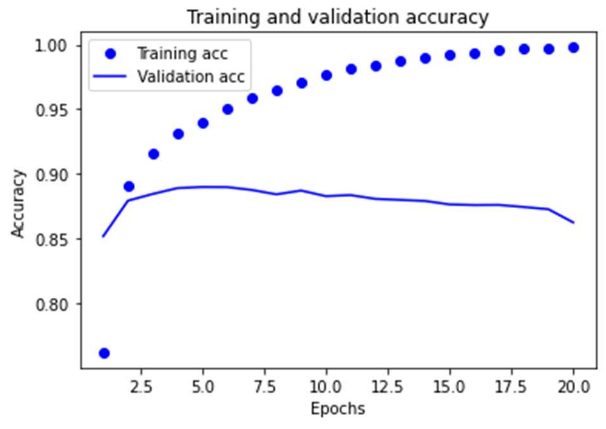
[Loss, accuracy] = [0.3108311593532562, 0.8764399886131287]

<u>Adding or decreasing Units (Width)</u>

We test the number of units for the model with two hidden layers. As we discussed previously, increasing the units of layers enhances the capacity which captures the input-output relationship. So, the required size of units depends on the input-output relationship and the model complexity. In this case, we see that increasing the units does mitigate the model's accuracy and validation loss (accuracy drops to 87% and we have overfitting after two epochs). Additionally, we can observe that using fewer units (8 units) is a little bit better regarding accuracy and validation. This means that we can use 8 or 16 units as the required units of layers.

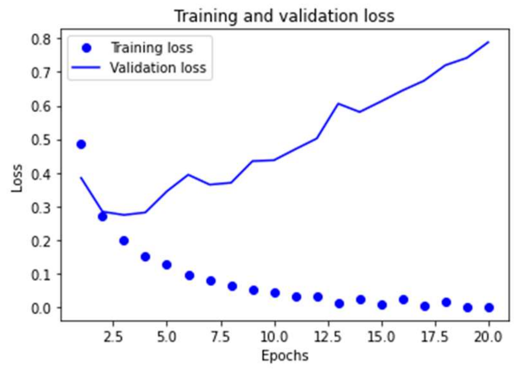
For 8 units:

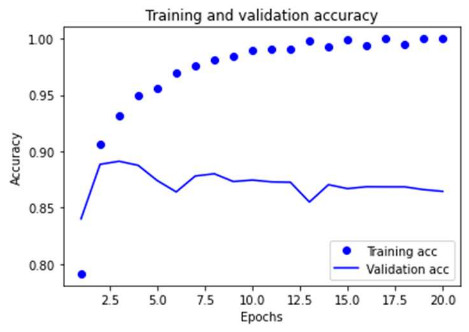




[Loss, accuracy] = `[0.27887991070747375, 0.8885200023651123]`

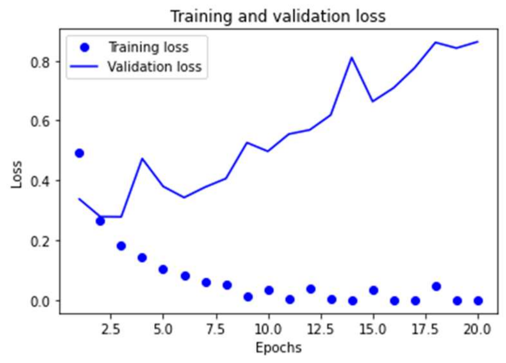
For 32 units:

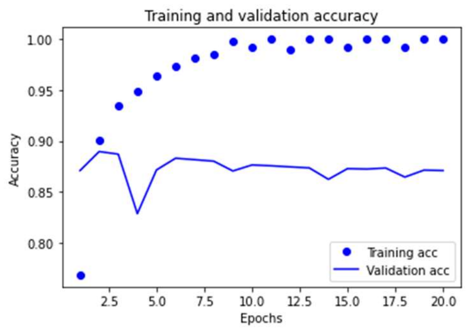




[Loss, accuracy] = `[0.3196236789226532, 0.8768799901008606]`

For 64 Units:



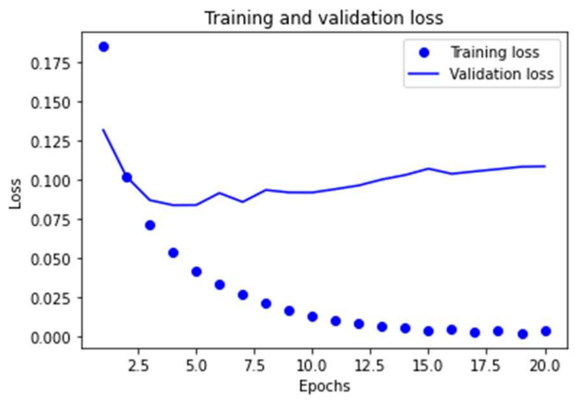


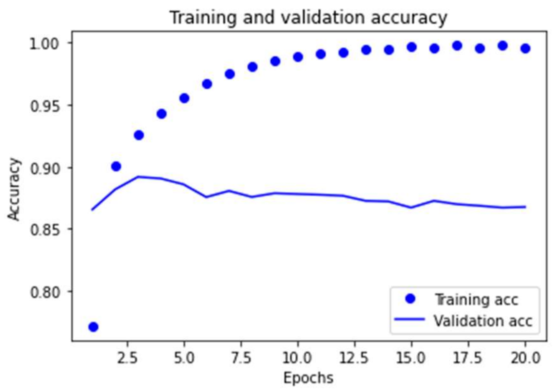
[Loss, accuracy] = `[0.3362124562263489, 0.8766400218009949]`

3 | Using the MSE loss function instead of Cross entropy

We replaced the MSE loss function instead of the Cross entropy function for the model with two hidden layers. The results show that the Cross entropy loss function outperforms than MSE loss function. When we do logistic regression (We use the sigmoid function to estimate the probability), the cross entropy is the loss function and gradient descent to minimize it. Cross entropy measures the “distance” between probability distributions. Since in this example with measure probabilities, Cross entropy fits better than MSE which is more compatible with linear regression problems. The theory behind this is that in a classification problem like this project, we have a specific set of output and we need a loss function that penalizes the error. The Cross entropy function does that rather than MSE.

[Loss, accuracy] = `[0.09375255554914474, 0.8744000196456909]`





4 | Using Tanh instead of Relu activation function

An activation function which is called the transfer function, in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network. The activation function calculates the multiplication of input to the model’s weights.

Tanh activation function as a traditional function results in output in a range of -1 to 1 with the below formula:

$$g(x) = (e^{-x} - e^{-x})/(e^{-x} + e^{-x})$$

The Relu function which is more widespread nowadays results in a range of 0 to infinity. This is represented with the formula $g(x) = \max(0 , x)$

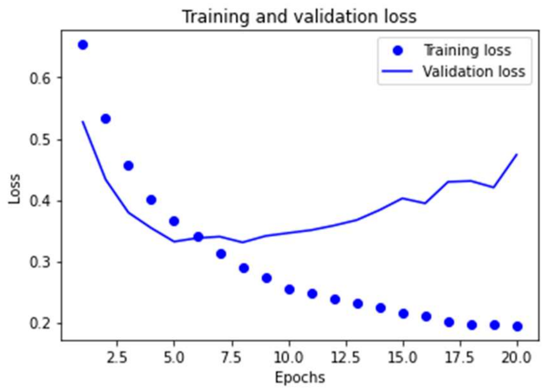
Using Tanh activation function instead of Relu for this example shows decreased model performance accuracy. Tanh function suffers from the vanishing gradient problem which causes a small change of output by the large change of input. A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training session. Because Tanh activation is more compatible with recurrent networks like time-series or sequential data, while Relu is appropriate for multilayer perceptions. Thus, in this project, the accuracy goes down by using Tanh function.

[Loss, accuracy] = `[0.36644941568374634, 0.8632000088691711]`

5 | Using Regularization and drop-out to prevent overfitting

In previous questions, we saw how decreasing capacity improves the model's accuracy. Additionally, two techniques of Regularization and Dropout are used to prevent the overfitting problem and consequently add accuracy to the model. Regularization is a model which adds to the loss function of the network a cost associated with having large. In code, there is L2 regularization which means every coefficient in the weight matrix of the layer will add (0.001 o any other value) * weight_coefficient_value to the total loss of the network.

Also, we included dropout (0.5) in this model to handle the overfitting problem. This technique randomly removes a different subset of neurons on each example which prevents conspiracies and thus reduces overfitting. 0.5 means that the model will drop 50% of neurons in the output. As we can see, the validation curves overfit later than the baseline model which means more training data is used by the regularization model. So the improved model is useful to prevent overfitting from 4 epochs to 7 epochs. As well as validation, the accuracy increases a little bit by including regularization and dropout.



[Loss, accuracy] = `[0.32776084542274475, 0.8875600099563599]`