**Chapter 3 (Part 2)**

Getting started with neural networks

**Questions**

1. Why do you have to turn your lists into tensors for inputting into a neural network?
2. Explain how we turned lists into tensors in 3.4.2.
3. What is the output for the following code?

x = [5, 1, 3, 2, 4]

results = np.zeros((len(x), 10))

for i, sequence in enumerate(x):

results[i, sequence] = 1

print(results)

1. What is the output for the following code?

x = [[5, 1, 3, 2, 4], [5, 1, 3, 2, 4], [1, 2, 3, 4, 9]]

results = np.zeros((len(x), 10))

for i, sequence in enumerate(x):

results[i, sequence] = 1

print(results)

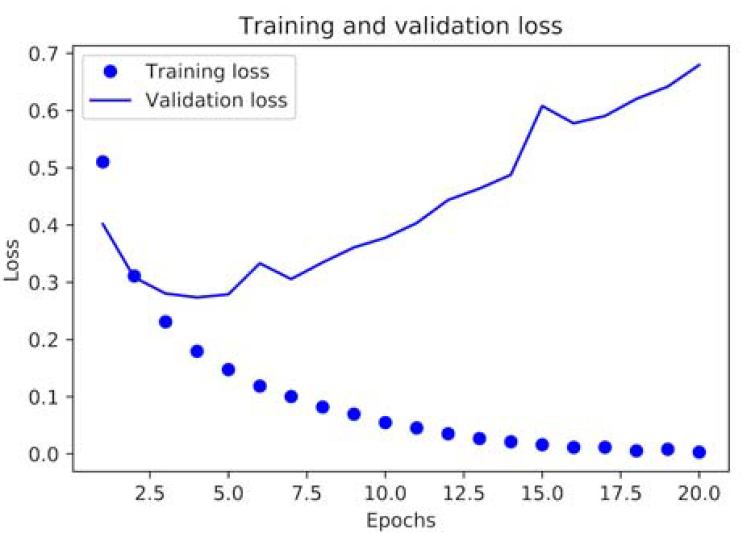
1. What does *relu* stand for and what does the function do?
2. What is a *hidden unit*? What are the pros and cons of having many *hidden units*?
3. What features of *Dense* layers does one have to adjust?
4. Why does it not really matter how many layers there are if using linear transformations? How does relu help with this?
5. The *crossentropy* loss function is the best choice when dealing with what kind of models? What does it do in simple terms?
6. Fill in the blank code.

model.compile(optimizer='rmsprop',

loss='binary\_crossentropy',

metrics=['accuracy'])

1. What does “plt.clf()” do?
2. Typically, what needs to be done to raw data before inputting into a neural network?
3. In a binary classification problem, what two things should your network end with?
4. In a binary classification problem that outputs a scalar between 0 and 1, what loss function should you use?
5. Why does the validation loss start and continue rising?



1. What is the different between single-label, multiclass classification and multilabel, multiclass classification?
2. Name the two ways labels can be vectorized.
3. Explain the categorical\_crossentropy function.
4. What is the difference between the loss functions categorical\_crossentropy and sparse\_categorical\_crossentropy?
5. If you are classifying data into 10 classes, how many Dense layers does the last layer in the neural network need to have?
6. Trying to predict a continuous value is what type of machine-learning problem?
7. If each feature in the input data has a different scale, what should you do before inputting the data? Explain how to do this in general terms.
8. When is it okay to use quantities computed on test data in your workflow?
9. If you have few samples, it is important to use a deeper neural network. (True or False)
10. In a regression machine-learning problem, it is optional to use a sigmoid activation function. (True or False)
11. What do *mse* and *mae* stand for?
12. In what situation is K-fold validation used? Why?
13. Write the code for feature-normalization given “train\_data” and “test\_data”.

**Answers**

1. (69) You can’t feed lists of integers into a neural network
2. (69) One-hot encode lists to convert into vectors of 0s and 1s. Use top layers as *Dense* layers for floating-point vector data.

“[[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]

[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]

[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]]”

“[[0. 1. 1. 1. 1. 1. 0. 0. 0. 0.]

[0. 1. 1. 1. 1. 1. 0. 0. 0. 0.]

[0. 1. 1. 1. 1. 0. 0. 0. 0. 1.]]”

1. (71) Rectified linear unit. Zeroes out negative values.
2. (70) a dimension in the representation space of the layer; how many dimensions a layer has in a neural network. Pros: learn more complex representations. Cons: computationally expensive and overfitting.
3. (70) # of layers and # of hidden units for each layer.
4. (72) Linear transformations use linear operation which would not extend the hypothesis even with added layers; limits scope of model. Relu is an activation function that is non-linear.
5. (73) Models that output probabilities. Measures the distance between probability distributions.
6. (73)

model.compile(optimizer='rmsprop',

loss='binary\_crossentropy',

metrics=['accuracy'])

1. (75) Clears the figure
2. (77) Preprocessing
3. (77) A dense layer with one unit and a sigmoid activation.
4. (77) binary\_crossentropy
5. (77) overfitting
6. (78) Grouping multiple (>2) data points into one label versus labeling data points in 1+ labels.
7. (79) one-hot encoding and categorical encoding. One-hot encoding involves creating a function to create label indexes while the other uses the to\_categorical method from keras.utils.np\_utils.
8. (80) “It measures the distance between two probability distributions: here, between the probability distribution output by the network and the true distribution of the labels. By minimizing the distance between these two distributions, you train the network to output something as close as possible to the true labels.”
9. Mathematically identical except categorical\_crossentropy needs categorically encoded labels. sparse\_categorical\_crossentropy can use integer tensors.
10. (84) 10
11. (85) regression
12. (86) feature-wise normalization. Subtract the mean of the feature and divide by the standard deviation to make sure the feature is centered around 0 with unit standard deviation.
13. (86) NEVER
14. (87) mean squared error. Mean absolute error.
15. (86) False. One must use a small network to mitigate overfitting issues related with smaller samples sizes.
16. (86) False. This would constrain output to a range [0:1].
17. (87) When there is a small data set. This can cause high variance depending on the validation split. To mitigate this risk, split the data into ‘K’ partitions and average the validation scores obtained from each fold.
18. (86)

mean = train\_data.mean(axis=0)

train\_data -= mean

std = train\_data.std(axis=0)

train\_data /= std

test\_data -= mean

test\_data /= std

1. –