CS577: Assignment 1

Jane Downer
Department of Computer Science
Illinois Institute of Technology

September 20, 2022

1. The training set is the data the model considers as it makes updates to its weights and biases. It is *not* referenced when tuning hyperparameters. The validation set is a set of separate data that is used to see how the model will generalize to unseen data, and is also used inform design decisions. The testing set is data that is not involved at all in the training process -- it is unseen data that we use to test the final model after training.

2.
Data: define and load training datatset and labels.
Model: Use Keras' Sequential or Functional API to build a neural network model. Choose layers, nodes, activation functions, learning rate, and various other hyperparameters.
Learning Process: Define optimizer, loss function, and metrics. Cal the 'compile' method to configure the learning process.
Fitting: Provide training data and labels as input, along with batch size and number of epochs. Train the model.

3. Number of units indicates the size of the output from the dense layer. Number of units can be interpreted as the number of features that the current layer is considering. If the neural network is a classifier, the number of units in the final layer will match the number of categories of data; if the neural network is a regressor, there will only be one unit in the final layer, giving a single estimate. Activation refers to a nonlinear function that is applied after multiplying input by weights and adding bias -- the introduction of nonlinearity by the activation function allows the model to learn complex patterns.

4. An optimizer is the method used to backpropagate through the model and apply updates. Both the loss and metrics are methods of defining distance between the ground truth and the output of the model. The difference is this: the loss function is what the optimizer tries to minimize as it makes updates to the model weights, whereas the metric is used to simply observe the performance of the model.

5. Input refers to the dimensions of the data being fed into a neural network. Output refers to the dimensions of data after leaving the neural network. Batch size refers to the number of examples that are fed through the neural network between each update. Epochs refers to the number of times the model sees the entire dataset. The validation data parameter is the corresponding sets of labels and values that we will use as our validation set, as described in my response to question 1.

6. One option is to perform one-hot encoding on the string, instead representing the string as a binary vector indicating whether each word is present or not.

7. Training accuracy is expected to grow higher than validation accuracy as the model trains. However, if validation accuracy starts to decrease, the model is overfitting to the training data -- it is learning the peculiarities of the training set rather than characteristics that are also representative of unseen data. When this happens, the wider the gap between training accuracy (higher) and validation accuracy (lower), the worse the overfitting is. If both training accuracy and validation accuracy are still increasing, the model has not yet reached its best performance, indicating that it is underfitting to the training data. Normally during training, the model will underfit up to a certain epoch, after which it overfits. The ideal epoch to stop at is when validation accuracy equals training accuracy.

8. The number of layers indicates the depth of the neural network. Generally, deeper neural networks are capable of learning more and perform better than their shallow counterparts. However, adding layers incurs a higher computational cost, can lead to overfitting, or can cause vanishing or exploding gradients. Units per layer indicates the number of features of the data that are being considered in a given layer. Like increasing the number of layers,

increasing the number of units increases the network's capacity to learn, but is computationally expensive and can lead to overfitting. An activation function is the nonlinear function being applied at the end of a given layer -- activation functions differ in the range and distribution of output values, and they are not equally suited to every task. Loss functions determine the way we measure distance between the ground truth and the output, and different loss functions are suited for different tasks -- for example, mean squared error is typically used for regression tasks, and cross entropy loss functions are used for classification tasks.

9. In a binary classification problem with a logistic function, the prediction vector will have two values -- each corresponding to one of the two possible classes. Each value can be interpreted as the probability of an observation belonging to its corresponding class.

10. One-hot encoding converts categorical labels into numerical labels. Prior to one-hot encoding, label data may be represented as a vector of categories -- one per observation -- with no numerical significance. The domain of this vector would be equal to the set of categories. With one-hot encoding, the label data is represented as a matrix -- with each column corresponding to a unique category, and each entry of that column specifying whether an observation belongs to that category (1) or not (0).

11. The softmax activation will result in output values between 0 and 1, the sum of which is exactly 1. When used in the output layer of a neural network, it can be interpreted as the probability of an observation belonging to each class.

12. Categorical cross-entropy can be used when label data is one-hot encoded. If the label data is represented as a vector of integers, with each distinct integer responding to a distinct category, then sparse cross entropy should be used instead.

13. The accuracy should be 20% -- i.e., 100% divided by 5, the number of classes.

14. For each feature vector, subtract the mean of *all* feature vectors, and divide by the standard deviation of *all* feature vectors. This prevents exploding gradients (i.e., large gradient updates), which would prevent the model from converging.

15. MAE, or mean absolute error, is the absolute value of the difference between ground truth and outputs. MSE, or mean squared error, is the squared difference between ground truth and outputs. MAE is easier to interpret, because it preserves the units of the input data.

16. With k-fold cross validation, the training data is split into k equal-sized subsets, or 'folds'. For each of the k folds, we train the model using that given fold as the validation set and the other k-1 folds as the training set -- i.e., k rounds of training, with each of the k folds getting a turn as the validation set. K-fold validation is particularly useful when we have small datasets. After we take the average of the validation accuracies from all k rounds and use this as our validation score.

17. When performing k-fold cross validation, we can keep logs of the validation MAE for each epoch. At the end of training, we can take the average of the k MAE values at each epoch and use this as our validation error. This helps ensure that the way we tune our hyperparameters is not as sensitive to the way we partition the dataset into training/validation -- we change our hyperparameters on the average behavior of k different data partitions, which reduces bias. This, in turn, helps us prevent overfitting.