CS 577 - F22 - Assignment 1 Questions Portion

Due date: 09/20/2022

Anas Puthawala - A20416308

Professor Gady Agam

Questions

(1) The training set is solely used to train the model to learn the parameters required for it to perform as intended and make the correct predictions when it's given new unseen data. The validation and testing set perform somewhat similar functions in that they use the model to make predictions, but they're different in the sense that the validation set is used throughout model training to help optimize and tune the model and its hyperparameters whereas the test set is ideally used at the end of model training / model development phase and is used to evaluate the model to gauge one last time the models' performance.

(2) Four steps involved in model development:
   (1) The first step is to ensure that the model can read the data in, so if it's text for example then vectorize it and convert it to a vector / numbers in a tensor format so the model can read and understand it.
   (2) Next you need to configure the actual neural network model, how many layers do you want, etc.
   (3) You also need to configure the learning process like choosing the loss function, the metrics for evaluation etc.
   (4) Finally call the .fit() module and fit the model with the parameters and arguments configured in steps 2, 3

(3) The basic parameters are:
   (1) The number of units you want in a dense layer (this could be 1 single unit or 100 units). The number of units is a positive integer.
   (2) The activation function you want, there are a lot of choices and sometimes it's best to understand the output that is desired before choosing an activation function

(4) Optimizer: An optimizer is an algorithm that serves to minimize the loss function, the one we learned so far in class is called gradient descent which calculates the gradient and minimize it by subtracting the gradient off to move towards the minima.
Loss: The loss is a quantity that the model wants to minimize during the optimization / learning process. There are also different types of loss functions used for different cases.
Metrics: Metrics are used to evaluate the performance of the model as it trains. In other words, metrics are used to gauge the model's performance as the optimizer minimizes the loss and tweaks the models decision boundary.
Difference between loss and metrics: The loss function is used by the optimizer to actually tune and tweak your model whereas the metric is used to gauge the 'tuning' and 'tweaking' process to see how well the model is performing after adjustments are being made.

(5) 5 arguments passed in to the .fit() module:
   (1) The input, which is the data that's passed in to the network (must be in tensor form, can't be text for example)
   (2) The output, which is the labels that the model wants to be able to match and learn a decision boundary / function from.
   (3) Batch size, which is the amount of training examples that should be considered at once before the model gets tuned and tweaked to reflect changes in the models decision function that it's learning
   (4) Epochs, which is the number of passes that should be done over the entire dataset in order to best learn the decision function / boundary for the model.
   (5) Validation data, which is used to optimize the models hyper-parameters during the model training / development phase.

(6) You can use one-hot-encoding to encode the index and wether or not each word is present in a given variable length string.

(7) The model underfits when there is high bias and the model isn't able to learn the relationship between the input and output data. A way you can tell the model is underfitting by looking at the loss graphs over epochs is that the loss graph is decreasing but very slowly. In this case, increasing the epochs might help the process of underfitting (i.e. let it train some more and the model can learn the parameters its needs and the loss will therefore continue decreasing).
When the model overfits, it has high variance and it's learning the parameters of the training data too well. This is often occurring because the model is either too complex, or there isn't richness in the training data. One way to tell that a model is overfitting when looking at the loss curves is that the loss diverges over time (goes to infinity) on the validation set whereas the loss may be decreasing simultaneously on the training set. The model is curve fitting the training data but failing to be able to generalize on new fresh data.

(8) Some possible hyper-parameters that can be tuned are:
   (1) The layers in a model (i.e. do we want multiple layers making a deep neural net, or do we want a shallow network with only around 2-3 layers?)
   (2) The number of units per layers. If we're going to make the network shallow (for example), how many units do we want per layer? The units can also help to determine what the size of the output will be, so it's usually a good idea to pay closer attention to Dense layers towards the end of your model.
   (3) Activation functions can also be classified as a hyper-parameters. Some activation functions can't run specific optimizers to minimize, such as for example the gradient descent optimizer will not work on piecewise constant loss functions.
   (4) Loss functions are also a hyper-parameter because if we have a binary classifier that we want to train, the optimal loss function for that case is going to be binary cross-entropy.

(9) If the logistic function is in the final layer then it'll output two numbers (which are probabilities) for the two classes. The two numbers will sum up to 1 and the number that is higher will be the chosen class for that training example. So basically you can use np.argmax to get index / class of the predicted output.

(10) In a multi-class classification problem, the class labels first need to get encoded to integers, so for example, Dog, Cat, Horse can be encoded as 1, 2, 3. Next, the integer value is encoded as a vector with all 0's excluding the index of the actual integer which will be a 1 (note that this 1 is different from the 1 that is encoding the labels Dog, Cat, and Horse — this 1 is solely used for one-hot-encoding which only contains 0s and 1s).

(11) The softmax function is used as an activation function in the output layer of a neural network that predicts a multinomial probability distribution. So for example, if you had a multi-class classification with 10 classes, you can have a softmax layer at the end with layer size of 10 which will contain the probabilities of that the input data belongs to either of these classes. If you sum the outputs from the 10 classes, they should sum to 1 and the class with the highest probability is the chosen class for that specific data.

(12) Sparse categorical cross entropy maintains that the labels are integers whereas categorical cross entropy maintains that the labels be one hot encoded vectors.

(13) The accuracy of a random classifier can be modeled as 1/k where k is the number of classes. In this case, with 5 classes the random classifier's accuracy should be 1/5 or

~20%.

(14) You subtract the mean and divide by the standard deviation to standardize the feature vector to have equal mean and standard deviation. The purpose of standardization and normalization is to help the optimization algorithm converge because sometimes the larger values may cause an exploding gradient problem or simply cause the learning algorithm to diverge. Normalization & standardization also helps by curbing the effects that larger values may have, for example if there are 10 people and you're tracking the wealth and 1 of those 10 is Elon Musk then the distribution will be skewed heavily, but if we normalize and standardize we can curb that skew.

(15) MSE measures the average squared distance between the estimated and actual values whereas MAE measures the the average absolute magnitude of errors (i.e. distances between estimated and actual values). The MAE is easier to interpret because the units are more intuitive to understand.

(16) K-fold cross validation is a technique in which the entire training / validation sample set is combined and there are 'k' partitions created in the set. One of the k partitions are used as a holdout / validation set and the remaining partitions are used as the training set. The model is exposed to the entire training sample set because the partition that gets held out for validation alternates through the entire set. This entire process is repeated k times and the final result of this method is that the results are averaged to produce a single estimate for the models performance. K-fold cross validation is used when there's a lack of sufficient data to train an algorithm and sometimes can be used for cases when there is sufficient data but the model is overfitting.

(17) The validation error is decided using the validation set which is being held out respective to the k'th partition of the data of k-fold cross validation. To report the validation error you can take the average of the model results for each of the 'k' folds