Ivan Miller - Problem Set 8 for Adversarial AI. Tuesday, December 6, 2022

1. Suppose there are two different settings of the weights and biases for a neural net that lead to the same overall performance on a training set. What’s a criterion you might use for choosing which model to use as a classifier? Explain your reasoning.

Even if the model performance on the training set is the same between two sets of weights, I would want to test it on the validation set, to check if the model would achieve similar performance on the unseen data. Since difference in performance between train and validation sets could be dependent on multiple factors. For example, if we're working with a training set that has an imbalance between classes. It would be possible for a model to learn the features of the majority class and show high overall accuracy even if it misclassified most of the data from the minority class.

One of the issues that may occur during training of the model is overfitting. It happens when the model learns well the features in the training set but it is not capable of generalizing to the unseen data. When that occurs there's usually a big gap between the accuracy of classification on the training and validation sets (with validation accuracy being significantly lower).

In addition to accuracy of the model on both validation and training sets, I would also calculate precision (calculated by dividing the true positives by anything that was predicted as a positive), recall (divide the true positives by anything that should have been predicted as positive) an F1 score (harmonic mean of precision and recall) on the validation set to select the set of weights that shows better performance.

2. Besides the sigmoid function 1/(1 + *ee*−*xx*), can you come up with another mathematical function that maps very large negative numbers to values near 0 and very large positive numbers to values near 1? Do you think your function would be a good replacement for the sigmoid function in neural nets? Discuss why or why not.

I would pick a Softmax function which takes a vector of real numbers as an input and outputs a probability distribution which sums up to 1, see the formula below:

$$S(z) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}}$$

where K is the number of classes in the multi-class classifier.

The main difference with sigmoid function is that in the denominator we sum all values together, which makes sure that each of the potential output classes will be in the range between 0 and 1 and all elements will add up to 1. Such a quality makes softmax function a great solution for multiclass classification problems without any extra work.
Even though technically we could still use sigmoid function for a multiclass classification problem but, since it produces a binary result, we would need to apply an additional technique, for example one-vs-all and create as many classifiers as the number of class instances we need to work with.