9.03 Deep Learning Optimization

Deciding on Batch Size

- The batch size defines the number of samples that will be propagated through the network.

- For instance, let's say you have 1000 training samples and you want to set up a batch_size equal to 100. The algorithm takes the first 100 samples (from 1st to 100th) from the training dataset and trains the network. Next, it takes the second 100 samples (from 101st to 200th) and trains the network again. We can keep doing this procedure until we have propagated all samples through the network.

Deciding on Batch Size

- Advantages of using a batch size < number of all samples:
  - It requires less memory. Since you train the network using fewer samples, the overall training procedure requires less memory. That’s especially important if you are not able to fit the whole dataset in your machine’s memory.
  - Typically networks train faster with mini-batches. That’s because we update the weights after each propagation.

Deciding on Batch Size

- Disadvantages of using a batch size < number of all samples:
    - The smaller the batch the less accurate the estimate of the gradient will be.

Deciding on Number of Epochs

- The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset.

- One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches.

- There are no hard and fast rules for selecting batch sizes or the number of epochs, and there is no guarantee that increasing the number of epochs provides a better result than a lesser number.

Tuning Hyperparameters using GridSearch

- To do cross-validation with keras we will use the wrappers for the Scikit-Learn API. They provide a way to use Sequential Keras models (single-input only) as part of your Scikit-Learn workflow.

- There are two wrappers available:
  - keras.wrappers.scikit_learn.KerasClassifier(build_fn=None, **sk_params), which implements the Scikit-Learn classifier interface,
  - keras.wrappers.scikit_learn.KerasRegressor(build_fn=None, **sk_params), which implements the Scikit-Learn regressor interface.

Tuning Hyperparameters using GridSearch

- Usually, we are not interested in looking at how just one parameter changes, but how multiple parameter changes can affect our results. We can do cross-validation with more than one parameters simultaneously, effectively trying out combinations of them.

- **Note: Cross-validation in neural networks is computationally expensive**. Think before you experiment! Multiply number of features you are validating on to see how many combinations there are.

What do we do if num of params and list of values in GridSearchCV is very large?

- This can be a particularly troublesome problem — imagine a situation where there are 5 parameters being selected for and 10 potential values that we have selected for each parameter.

- The number of unique combinations of this is 10^5, which means we would have to train a ridiculously large number of networks.

- Clearly, it would be impractical to actually do it this way, so it is common to use RandomizedCV as an alternative.

What do we do if num of params and list of values in GridSearchCV is very large?

- RandomizedCV allows us to specify all of our potential parameters, and then for each fold in the cross-validation, it selects a random subset of parameters to use for the current model.

- In the end, the user can select the optimal set of parameters and use these as an approximate solution.