Evaluation

Evaluate a fully trained neural network using the model accuracy as the evaluation metric.

Chapter Goals:

- Evaluate model performance on a test set

A. Evaluating using accuracy

After training a model, it is a good idea to evaluate its performance. We do this by using a *test set* (i.e. data points not used in model training) and observe the model's prediction accuracy on the test set.

The code for this chapter makes use of the `accuracy` metric defined in Chapter 4. The accuracy represents the classification accuracy of an already trained model, i.e. the proportion of correct predictions the model makes on a test set.

B. Different amounts of data

Since we used `None` when defining our placeholder shapes, it allows us to run training or evaluation on any number of data points, which is super helpful since we normally want to evaluate on many more data points than the training batch size.

It is good practice to split up a dataset into a three sets:

- *Training set (~80% of dataset)*: Used for model training and optimization
- *Validation set (~10% of dataset)*: Used to evaluate the model in between training runs, e.g. when tweaking model parameters like batch size
- *Test set (~10% of dataset)*: Used to evaluate the final model, usually through some accuracy metric

Time to Code!

The coding exercise for this chapter uses the `accuracy` metric from Chapter 4 (which is initialized in the backend). We also provide the test set's data and labels (`test_data` and `test_labels`) as NumPy arrays initialized in the backend, as well as the `inputs` and `labels` placeholders.

We've taken the liberty of loading a pretrained model in the backend using a `tf.Session` object called `sess` . You'll be evaluating the accuracy of the pretrained model on the test data and labels.

Set `feed_dict` equal to a python dictionary with key-value pairs `inputs: test_data` and `labels: test_labels` .

Set `eval_acc` equal to the output of `sess.run` , with first argument `accuracy` and a keyword argument `feed_dict=feed_dict` .

```
1  # test_data, test_labels, inputs, labels, accuracy
2  # are all predefined in the backend
3  # CODE HERE
```

Solution

```
1  feed_dict = { inputs: test_data, labels: test_labels }
2  eval_acc = sess.run(accuracy, feed_dict=feed_dict)
3
```

Back

Training

Next

Linear Limitations

Mark as Completed

85% completed, meet the criteria and claim your course certificate!

Buy Certificate