TensorFlow Dropout

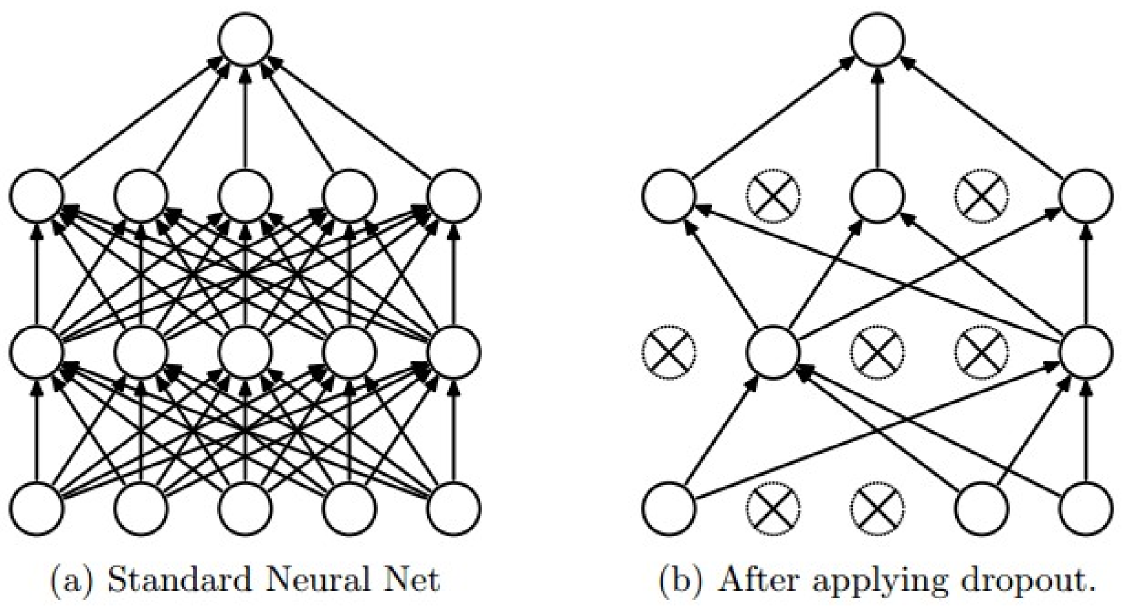

Figure 1: Taken from the paper "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" (https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf)

Dropout is a regularization technique for reducing overfitting. The technique temporarily drops units (artificial neurons) from the network, along with all of those units' incoming and outgoing connections. Figure 1 illustrates how dropout works.

TensorFlow provides the `tf.nn.dropout()` function, which you can use to implement dropout.

Let's look at an example of how to use `tf.nn.dropout()`.

```
keep_prob = tf.placeholder(tf.float32) # probability to keep units

hidden_layer = tf.add(tf.matmul(features, weights[0]), biases[0])
hidden_layer = tf.nn.relu(hidden_layer)
hidden_layer = tf.nn.dropout(hidden_layer, keep_prob)

logits = tf.add(tf.matmul(hidden_layer, weights[1]), biases[1])
```

The code above illustrates how to apply dropout to a neural network.

The `tf.nn.dropout()` function takes in two parameters:

1. `hidden_layer`: the tensor to which you would like to apply dropout
2. `keep_prob`: the probability of keeping (i.e. *not* dropping) any given unit

`keep_prob` allows you to adjust the number of units to drop. In order to compensate for dropped units, `tf.nn.dropout()` multiplies all units that are kept (i.e. *not* dropped) by `1/keep_prob`.

During training, a good starting value for `keep_prob` is `0.5`.

During testing, use a `keep_prob` value of `1.0` to keep all units and maximize the power of the model.

Quiz 1

Take a look at the code snippet below. Do you see what's wrong?

There's nothing wrong with the syntax, however the test accuracy is extremely low.

```
...

keep_prob = tf.placeholder(tf.float32) # probability to keep units

hidden_layer = tf.add(tf.matmul(features, weights[0]), biases[0])
hidden_layer = tf.nn.relu(hidden_layer)
hidden_layer = tf.nn.dropout(hidden_layer, keep_prob)

logits = tf.add(tf.matmul(hidden_layer, weights[1]), biases[1])
```

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for epoch_i in range(epochs):
        for batch_i in range(batches):
            ....

            sess.run(optimizer, feed_dict={
                features: batch_features,
                labels: batch_labels,
                keep_prob: 0.5})

    validation_accuracy = sess.run(accuracy, feed_dict={
        features: test_features,
        labels: test_labels,
        keep_prob: 0.5})
```

QUESTION 1 OF 2

What's wrong with the above code?

Dropout doesn't work with batching.

The keep_prob value of 0.5 is too low.

There shouldn't be a value passed to keep_prob when testing for accuracy.

✓ keep_prob should be set to 1.0 when evaluating validation accuracy.

SUBMIT

Quiz 2

This quiz will be starting with the code from the ReLU Quiz and applying a dropout layer. Build a model with a ReLU layer and dropout layer using the `keep_prob` placeholder to pass in a probability of `0.5`. Print the logits from the model.

Note: Output will be different every time the code is run. This is caused by dropout randomizing the units it drops.

quiz.py | solution.py

```python
# Solution is available in the other "solution.py" tab
import tensorflow as tf

hidden_layer_weights = [
    [0.1, 0.2, 0.4],
    [0.4, 0.6, 0.6],
    [0.5, 0.9, 0.1],
    [0.8, 0.2, 0.8]]
out_weights = [
    [0.1, 0.6],
    [0.2, 0.1],
    [0.7, 0.9]]

# Weights and biases
weights = [
    tf.Variable(hidden_layer_weights),
    tf.Variable(out_weights)]
biases = [
    tf.Variable(tf.zeros(3)),
    tf.Variable(tf.zeros(2))]

# Input
features = tf.Variable([[0.0, 2.0, 3.0, 4.0], [0.1, 0.2, 0.3, 0.4], [11.0, 12.0, 13.0, 14.0]])

```

```
hidden_layer = tf.add(tf.matmul(features, weights[0]),biases[0])
hidden_layer = tf.nn.relu(hidden_layer)
hidden_layer = tf.nn.dropout(hidden_layer, keep_prob)

```

RESET QUIZ TEST RUN SUBMIT ANSWER

NEXT