DS-GA 3001: Assignment 1

Jiali Zhou

September 28, 2016

Activation and dropout

First comes with the experiments of changing activation and adding dropout.

evaluation	*dropout*(0.5)	*dropout*(0.3)	*dropout*(0.7)
loss	1.45	1.52	1.27
accuracy	0.74	0.73	0.73

I chose dropout with probability 0.5 as the default.
After adding dropout, here comes with the experiments of different activation methods.

evaluation	*tanh*	*relu*	*relu*6	*sigmoid*
loss	1.24	1.10	1.26	1.41
accuracy	0.71	0.74	0.72	0.72

From this I chose relu as the activation as it has the lowest loss and highest accuracy.
Considering the learning rate, I chose different epochs and decaying learning rate. I found that if the epochs is chosen to be larger than 30 without decaying learning rate, then the loss will become larger and accuracy lower from some point. This is due to overfitting of the training set.

epochs	*learning rate*	*loss*	*accuracy*
50	0.001	5.31	0.68
100	initial 0.001, decay to 0.96 after 3000 steps	4.01	0.71
30	0.001	1.31	0.732
15	0.001	1.14	0.745

Regularization

From above we can find that at epochs 15, the training process is already converged. Another way to prevent overfitting is adding regularization. As I add in the regularization, I tune the epochs to be slightly larger than the last experiment(15).

epochs	*regularizationL*	*loss*	*accuracy*
30	0.5	0.97	0.734
30	0.2	0.86	0.742

We can see from above that although it does little improvement to the accuracy, regularization greatly reduces the evaluation loss. Listing 1 shows activation and regularization codes.

Adding layer

Considering the depth, I added one more layer to see if there is improvement.

epochs	*feature length fromfirst pooling layer*	*regularizatione*	*loss*	*accuracy*
15	10	0.2	1.18	0.701
15	5	0.5	0.89	0.729

Feature fine tuning

Then comes with different combination of the parameters.

epochs	*batch size*	*embedding size*	*filter size*	*loss*	*accuracy*
15	64	64	3, 6, 9	1.19	0.733
15	64	64	3, 4, 5	1.13	0.741
15	32	64	3, 4, 5	1.14	0.745

Here comes the figure to show accuracy and loss of different parameters in dev and test set. From the figures, we can clearly see that accuracy starts to converge after 2500 steps. Models start to overfit after 500 steps. So there should be a compromise between accuracy and loss.

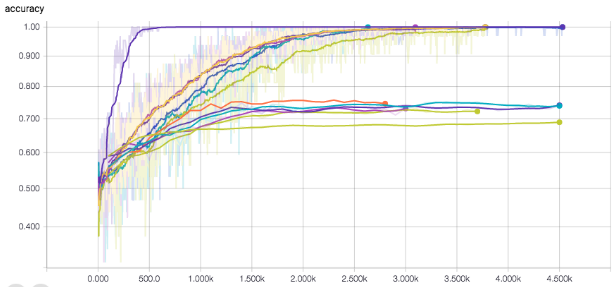

Figure 1: Accuracy in different settings of features in dev and test set.

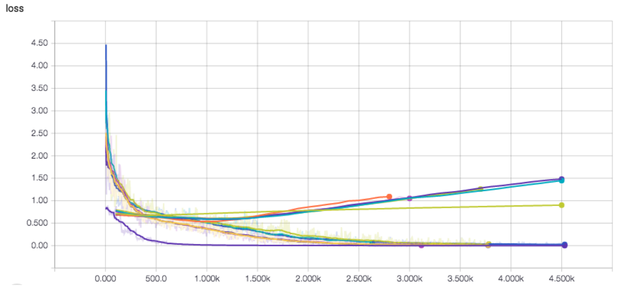

Figure 2: Loss in different settings of features in dev and test set.

Listing 1: Codes

```
for i, filter_size in enumerate(filter_sizes):
    with tf.name_scope("conv-maxpool-%s" % filter_size):
        # Convolution Layer
        filter_shape = [filter_size, embedding_size, 1, num_filters]
        W = tf.Variable(tf.truncated_normal(filter_shape, stddev=0.1), name="W")
        b = tf.Variable(tf.constant(0.1, shape=[num_filters]), name="b")
        conv = tf.nn.conv2d(
            self.embedded_chars_expanded,
            W,
            strides=[1, 1, 1, 1],
            padding="VALID",
            name="conv")
        # Apply nonlinearity
        h = tf.nn.relu(tf.nn.bias_add(conv, b), name="relu")
        # Maxpooling over the outputs
        pooled = tf.nn.max_pool(
            h,
            ksize=[1, sequence_length - filter_size + 1, 1, 1],
            ksize=[1, 10, 1, 1]
            strides = [1, 1, 1, 1],
            padding='VALID',
            name="pool")
        pooled_outputs.append(pooled)

# Combine all the pooled features
num_filters_total = num_filters * len(filter_sizes)
self.h_pool = tf.concat(3, pooled_outputs)
self.h_pool_flat = tf.reshape(self.h_pool, [-1, num_filters_total])

# Add dropout
###############add your code here################
#hint: you need to add dropout on self.h_pool_flat with tf.nn.dropout()
with tf.name_scope("dropout"):
    self.h_drop = tf.nn.dropout(self.h_pool_flat,self.dropout_keep_prob)
```