50.021 -AI

Alex

Week 04: Tensorflow

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

In this class - convolutional neural network on mnist

- the same neural net we had, now in tensorflow (before we move to convolutions)
- adding summaries that monitor train and validation error even those are not available in tensors

in class coding:

Consider `nnet2e_studentversion.py`. It has the typical structure as described before.

For a real deep network tensorflow suggests tfrecord for training. see `https://kwotsin.github.io/tech/2017/02/11/transfer-learning.html`

There is one task in it:

1. Your task: complete the neural network definition. Prefilled is the first layer in the method `somenetclass.predict`. The definition of weight and biases in it are given in: `somenetclass.paramsdict`.

   (a) in short:
   $(n, 28, 28, 1) \rightarrow$
   $conv(numkernels = 20, ksize = (5,5), stride = 1, pad = same)$
   $-relu - maxpool(ksize = 2, stride = 2, pad = same) -$
   $conv(numkernels = 40, ksize = (5,5), stride = 1, pad = valid)$
   $-relu - maxpool(ksize = 2, stride = 2, pad = same) - reshape$
   $-fullyconn(1000) - relu - dropout$
   $-fullyconn(1000) - relu - dropout$
   $-fullyconn(10)$

   (b) the input after reshaping are images in shape $(batchsize, height = 28, width = 28, c)$. where $c = 1$ because the images are grey.

(c) first layer: A convolution layer with 20 kernels, stride 1, kernel size 5, Padding SAME.

check the doc `tf.nn.conv2d`.

For determining the weight in the current layer, and the shape of the output (which will be input in the next layer) you will have to understand how all these parameters affect them.

For the weight shape:

Note that the weight in `somenetclass.paramsdict` is required to have shape $[filter\_height, filter\_width, in\_channels, out\_channels]$. kernel size here is $filter\_height, filter\_width$.

How many $in\_channels, out\_channels$ are present given ... the input shape, and given the fact that we use a convolution layer with 20 kernels?

For the output shape:

- without padding (tensorflow padding parameter 'VALID'), one would have as resulting size of $output\_spatial\_shape[i] = ceil((input\_spatial\_shape[i] - kernel\_size + 1)/strides[i])$
- with padding (tensorflow padding parameter 'SAME'): $output\_spatial\_shape[i] = ceil(input\_spatial\_shape[i]/strides[i])$

This applies to spatial shape being any dimension, in particular height and width.

The 2 padding parameters of tensorflow are explained in `https://www.tensorflow.org/api_docs/python/tf/nn/convolution`.

For the first layer, we set Padding to be SAME.

(d) This conv layer is followed by a relu, and a maxpool with 2×2 and stride 2. we set Padding to be SAME. check the doc `tf.nn.max_pool`.

To understand the effect of this: pooling is similar to a convolution when it comes to the output shapes (see above). So 2×2 is the kernel size. the stride needs to be applied to both height and width. What will be the output shape ?

(e) the next layer is a convolution: 40 kernels, stride 1, kernel size 5, now padding is VALID. what will be weight shape and output shape?

(f) This conv layer is followed again by a relu, and a maxpool with 2×2 and stride 2. we set Padding to be SAME. check the doc `tf.nn.max_pool`.

(g) now comes an already coded reshape layer (which flattens the result)

(h) now comes a fully connected layer with 1000 neurons as outputs

(i) now comes a relu and a dropout layer

(j) now comes again a fully connected layer with 1000 neurons as outputs

(k) now comes a relu and a dropout layer

(l) now comes again a fully connected layer with 10 neurons as outputs

2. evaluate the effect of different values for the`keep_prob` parameter in the dropout layers during training $0.2, 0.4, 0.6, 0.8, 1.0$.

3. question to think: what are weight and output shapes for
$(n, 28, 28, 1) \rightarrow$
$conv(numkernels = 20, ksize = (4, 4), stride = 1, pad = same)$
$-relu - maxpool(ksize = 2, stride = 2, pad = same) -$
$conv(numkernels = 40, ksize = (6, 6), stride = 1, pad = valid)$
$-relu - maxpool(ksize = 2, stride = 2, pad = same) - reshape$
$-fullyconn(1000) - relu - dropout$
$-fullyconn(1000) - relu - dropout$
$-fullyconn(10)$?

4. compare this networks performance to the above for `keep_prob`$=$ 0.5