Midterm Practice Questions Notes

Question 1

Part (a)

Answer: E.

In a recurrent neural network, the input sequence length is variable, so batching is less straightforward. We pd shorter sequences in a batch so that all input sequences in the same batch have the same length.

In a word2vec model, the input sequence and output sequence length is the same size. The input is the one-hot encoding of a word, and the output is the one-hot encodings of a fixed number of words surrounding the input. Batching is therefore still straightforward.

Part (b)

Answer: C.

Everything else can helps reduce overfitting, but manipulating the test data does not affect the training of neural network models at all.

Part (c)

Answer: E

None of these cause an error. The function call `img.view(-1, 28*28)` reshapes the image to `[n, 28 * 28]` where `n` is variable. Tensors of all these shapes can be rehaped to the target dimension.

Part (d)

Answer: E

Transpose convolution is used in pixel-wise prediction and image generation tasks, where we need to increase the width/height of the hidden/output layer tensor dimension. Choice B is false because even though we never really talked about what an RNN that would take an image as input looks like, whenever we took images as input, we used standard convolution layers that consolidate information. We used standard convolution layers with some subsampling method.

Part (e)

Answer: A

The RNN output vector length is the same as the length of the hidden state vector. The RNN input vector length is not necessarily the same as the length of the hidden state vector. PyTorch `nn.LSTM` is more often used than `nn.RNN`, and has more parameters than `nn.RNN`.

Question 2

This was a training curve we saw in the first few weeks, on a binary classification problem where the probability of a data point belonging in one of the classes was 70%. Thus, the neural network learns very quickly that classifying all inputs into the larger class would result in a 70% accuracy. After several more iterations of training, the network begins to learn something "real" about the problem.

Question 3

Part (a)

The number of parameters in `NetworkA` is as follows:

- the layer `self.conv1` has `5 * (3 * 3 * 3 + 1)` parameters
- the layer `self.conv2` has `10 * (5 * 3 * 3 + 1)` parameters
- the layer `self.fc` has `(10 * 5 * 5) * 10 + 10` parameters

Part (b)

The number of parameters in `NetworkB` is as follows:

- the layer `self.fc1` has `(20 * 20) * 300 + 300` parameters
- the layer `self.fc2` has `300 * 100 + 100` parameters
- the layer `self.fc3` has `100 * 10 + 10` parameters

Part (c)

Even without a calculator it is easy to see that `NetworkB` has orders of magnitude more parameters than `NetworkA`, and is more likely to overfit.

Question 4

1. There is no activation function between `layer1` and `layer2`
2. The softmax activation should not be applied in the `forward` function
3. The output dimension of `layer1` does not match the input dimension of ‘layer2.

Question 5

Part (a)

The word2vec model uses an encoder-decoder architecture.

- The input of the encoder is the one-hot encoding of a word.
- The output of the encoder is the embedding of the word.
- The input of the decoder is the embedding of the word.
- The output of the decoder is the one-hot encoding of the words surrounding the input word.

The encoder and decoder both use fully-connected layers.

Part (b)

Any two of the below is fine:

- Max-pooling
- Convolution with a stride > 1
- Average-pooling

Part (c)

- Use pre-trained word embedding like GloVe embedding
- Use some kind of data augmentation: e.g. split / combine tweets, add spelling mistakes, taking care that the augmented tweets don’t alter any gender signals.