CS 224N: Assignment 2

RYAN MCMAHON TUESDAY 7TH FEBRUARY, 2017

Contents

Problem 1: Tensorflow Softmax (25 pts)

In this question, we will implement a linear classifier with loss function

$$J(\boldsymbol{W}) = CE(\boldsymbol{y}, softmax(\boldsymbol{x}\boldsymbol{W})) \tag{1.1}$$

Where $\boldsymbol{x}$ is a row vector of features and $\boldsymbol{W}$ is the weight matrix for the model. We will use TensorFlow's automatic differentiation capability to fit this model to provided data.

1.1 (a) Softmax in Tensorflow (5 pts)

Implement the softmax function using TensorFlow in `q1_softmax.py`. *Remember that*

$$softmax(\boldsymbol{x})_i = \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}} \tag{1.2}$$

Note that you may not use `tf.nn.softmax` *or related built-in functions. You can run basic (nonexhaustive tests) by running* `python q1_softmax.py`.

Answer:

See code: `~/code/q1_softmax.py`.

1.2 (b) Cross-Entropy Loss in Tensorflow (5 pts)

Implement the cross-entropy loss using TensorFlow in `q1_softmax.py`. *Remember that*

$$CE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{N_c} y_i log(\hat{y}_i) \tag{1.3}$$

*where $\boldsymbol{y} \in \mathbb{R}^{N_c}$ is a one-hot label vector and N_c is the number of classes. This loss is summed over all examples (rows) of a minibatch. Note that you may **not** use TensorFlows built-in cross-entropy functions for this question. You can run basic (non-exhaustive tests) by running* `python q1_softmax.py`.

Answer:

See code: `~/code/q1_softmax.py`.

1.3 (c) Placeholders and Feed Dictionaries (5 pts)

Carefully study the Model class in `model.py`. *Briefly explain the purpose of placeholder variables and feed dictionaries in TensorFlow computations. Fill in the implementations for* `add_placeholders` *and* `create_feed_dict` *in* `q1_classifier.py`.

Hint: *Note that configuration variables are stored in the* `Config` *class. You will need to use these configurations variables in the code.*

Answer:

The purpose of placeholder variables and feed dictionaries is, in short, to improve computational efficiency. They allow us to specify the graph structure without supplying the actual data. By doing this, our model can be flexible to different training approaches (e.g., different mini-batch sizes). Specifically, the placeholders stand in for whatever data/values we want to use and the feed dictionaries tell the placeholders what data/values to use.

Also, see code: `~/code/q1_classifier.py`.

1.4 (d) Add Softmax and Add CE Loss (5 pts)

Implement the transformation for a softmax classifier in the function `add_prediction_op` *in* `q1_classifier.py`. *Add cross-entropy loss in the function* `add_loss_op` *in the same file. Use the implementations from the earlier parts of the problem,* ***not*** *TensorFlow built-ins.*

Answer:

See code: `~/code/q1_classifier.py`.

1.5 (e) Add Training Optimizer – Gradient Descent (5 pts)

Fill in the implementation for `add_training_op` *in* `q1_classifier.py`. *Explain how TensorFlows automatic differentiation removes the need for us to define gradients explicitly. Verify that your model is able to fit to synthetic data by running* `q1_classifier.py` *and making sure that the tests pass.*

Hint: *Make sure to use the learning rate specified in* `Config`.

Answer: