ECE421: Assignment #4

Won-Jin Kim
(10036114 24)

1. (1 point) Your data has been split into training, validation and test set. Examine the ratio of the split and number of examples in each set. Suppose you were to train on batches of 32 examples each. That is, in each step of gradient descent, you randomly select 32 examples from the training set, compute your average loss on these examples, and then compute the gradient of this average loss with respect to the model parameters.

   How many iterations will it take to go through the entire training set given the number of training examples yielded by the data split? How many iterations are there in 30 epochs? Recall that one epoch is the number of iterations needed to train over the entire dataset.

out of 60000 dataset

80% training data : 48000
10% validation data 6000
10% test data 6000

batch : 32 samples each

How many iteration to go through training set

$\therefore \frac{48000}{32} =$ 1500 iterations

How many iterations in 30 epoch?

1500 × 30 = 45000 iterations

2. (2 points) Fill in the code for your custom convolution filter and show that it returns the same output as Objax's own convolution routine.

in the notebook file (.ipynb)

3. (1 point) Fill in the code for your linear layer, and show that it returns the same output as passing through Objax's own linear layer.

in the notebook file (.ipynb)

4. (1 point) Explain in a short paragraph what is the difference between the training and validation set.

We use training set to train the model & find optimal weight & bias in the case of NN. Validation set is use to evaluate the model fitted on the training set while tuning hyperparameters. Hence model sees the data but does not actually learn from them. We can not tune hyperparameter in training set because we want hyperparameters that generalize.

Part 2. Training and Tuning a CNN. For this part, you have been given a starter code. Navigate to the portion where you define your optimizer.

1. (1 point) Complete the optimizer by using the definition of (stochastic) gradient descent: $w_{k+1} = w_k - \epsilon \nabla L(w_k)$. Note that you need to update `params.value`, which are the values of the trainable variables of your model.

in the notebook file (.ipynb)

2. (1 point) Complete the batch sampling code in the train function by specifying a batch of examples. You should make use the lists `train_indices` and `val_indices`.

in the notebook file (.ipynb)

3. (1 point) Train the model for a few epochs, and observe the training/validation loss and training/validation accuracy plots. Include these plots within the PDF you hand in.

   You should observe that the validation accuracy is low and stagnates after a few epochs. Next we will go through a rudimentary way of adjusting the hyperparameters of the model which we created.

in the notebook file (.ipynb)

4. (1 point) In one sentence, define the meaning of a "hyperparameter". Explain in a short paragraph why it is important not to evaluate the accuracy on the test set until all hyperparameters have been tuned.

Hyperparameter is configuration that is external to the model & whose value can not be estimated from data.

Tuning of the hyper parameters are done in the validation set. However evaluation using this set become more biased as still on the validation dataset is incorporated into the model configuration. Thus we use test set to evaluate the model as it provides unbiased evaluation of a final model fit on training data.

5. (2 points) Select 4 hyperparameters associated with your network, one of the hyperparameter must involve your CNN architecture, and come up with two different sets of hyperparameters.

   For example, suppose my set of hyperparameters are defined as (yours might be different) $H = \{$batch size, learning rate, number of outputs of conv layer 1, number of conv layers$\}$, where conv layer is defined as a composition between filter, activation and pooling, then two sets of hyperparameters may be,

$$H_1 = \{32,\ 0.001,\ 16,\ 2\} \qquad H_2 = \{64,\ 0.0001,\ 32,\ 3\}$$

   The hyperparameters that you tune does not need to be specified as a numerical value. For instance, you can specify the optimizer you are using to train the network, or the activation function you are using. You may wish to consult: `https://objax.readthedocs.io/en/latest/notebooks/Custom_Networks.html`

in the notebook (.ipynb)

↓ every other questions written in Jupyter Notebook