Lab Course: Distributed Data Analytics
Exercise Sheet 7

Mofassir ul Islam Arif
Information Systems and Machine Learning Lab
University of Hildesheim
Submission deadline: Monday June 24, 23:59PM (on LearnWeb, course code: 3116)

Instructions

Please following these instructions for solving and submitting the exercise sheet.

1. You should submit a zip or a tar file containing two things a) python scripts and b) a pdf document.
2. In the pdf document you will explain your approach (i.e. how you solved a given problem), and present your results in the form of graphs and tables.
3. The submission should be made before the deadline, only through learnweb.

Complex Data: Image Classification with TensorFlow

Image classification is a task of assigning an input image $I \in \mathbb{R}^{H \times W \times C}$ a class l out of a fixed set of given classes $\mathcal{L} \in \{1 \ldots L\}$, where H is the height, W is the weight and C is the number of channels, L are the number of classes. For example an input image $[32 \times 32 \times 3]$ has height 32, weight 32 and three channels (a color image has RGB values). In this task we will look at CIFAR-10 dataset available at `https://www.cs.utoronto.ca/~kriz/cifar.html`. CIFAR-10 dataset consists of 60000 color images (each image is a $32 \times 32 \times 3$. The data is already divided into five batches and test batch. You can download the data and load it into python dict object as explained on the said URL i.e.

- download `https://www.cs.utoronto.ca/~kriz/cifar-10-python.tar.gz`
- for python3 you have to use the following code snippet.

```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

```
cifar = unpickle('data_batch_1')
```

The cifar holds a dict object returned by unpickle. (read more about the dict object returned by the function at `https://www.cs.utoronto.ca/~kriz/cifar.html`).

Note: Performance analysis with respect to the number of workers is not required in this exercise sheet.

Exercise 1: Convolutional Neural Networks for Image Classification (CNN)(10 points)

Convolutional Neural Networks (CNNs) are very powerful models capable of extracting and learning structural features from the image [2]. In this task you have to build a first CNN model. You can have a look at tutorial [1] to make yourself familiarized with implementing CNNs in Tensorflow. You can re-run the code and understand different parts that are useful in Tensorflow tf.nn and tf.train packages i.e. tf.nn.conv2d, tf.nn.selu, tf.nn.relu, tf.nn.max_pool, softmax_cross_entropy_with_logits, tf.nn.batch_normalization, tf.train.GradientDescentOptimizer, tf.train.AdamOptimizer, tf.train, tf.nn.softmax .

First task here is to build a complete pipeline with a simple CNN architecture. You will build a network with the following components (see [6, 7] for more details about components)

1. conv1: convolution and rectified linear activation (RELU)
2. pool1: max pooling
3. FC2: fully connected layer with rectified linear activation (RELU)
4. softmax_layer: final output predictions i.e. classify into one of the ten classes.

Build a training pipeline

1. To train the CNN model you might have to perform some standard transformation on the images to increase the size of the data. This is usually done by randomly scaling, rotating and translating the images. This step is called data augmentation. You are required to implement this and for 1 example image, show the effect of the augmentation. Meaning you should show the original image and the 3 variants with the agumentation techniques applied to it.
2. Develop a mini-batch optimization loop that picks up a small batch of images (5 to 10) and perform an iteration of optimization (see [7]).
3. Explain how NNs are regularized (one paragraph). Implement the reguarlization.
4. Record some information for Tensorboard visualization. i.e. as scalars, graphs, distributions and histograms. You might check your Tensorboard at `localhost:6006`.
5. Report and plot accuracy on both training set and test set. **Hint:** If you write a smart function that can make CONV layers and interleave them with pooling layers, than you wont ever have to handcraft an entire CNN ever again. Think about it.

Exercise 2: Convolutional Neural Networks for Image Classification (CNN) (10 points)

In this task you will build your own CNN architecture with following components (see [6, 7] for more details).

1. conv1: convolution and self normalization activation (selu)
2. pool1: max pooling
3. norm: batch normalization
4. pool2: max pooling
5. FC1: fully connected layer with self normalization activation (selu)
6. FC2: fully connected layer with self normalization activation (selu)
7. softmax_layer: final output predictions i.e. classify into one of the ten classes.

Once you have build your model, its time to train it. [Note:] This network might become already complex and will take longer to train on a CPU machine.

1. Use data augmentation you did in previous task
2. Train your model using mini-batch optimization.
3. Create a Tensorboard that presents basic information such as scalars, graphs, distributions and histograms. You might check your Tensorboard at `localhost:6006`.

Annex

1. TensorFlow Tutorial CNNs: https://www.tensorflow.org/tutorials/deep_cnn
2. CNNs: http://cs231n.github.io/convolutional-networks/
3. CNNs: http://cs231n.github.io/convolutional-networks-1/
4. Data preprocess http://cs231n.github.io/neural-networks-2/
5. http://cs231n.github.io/neural-networks-3/
6. CNN Lecture 1 https://www.youtube.com/watch?v=bNb2fEVKeEo&t=833s&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRI
   index=6
7. CNN Lecture 2 mini-batch http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture6.
   pdf