Deep Learning Frameworks - Tensorflow and Pytorch

Introduction

In this exercise we learn a more efficient and reasonable way of implementing deep learning algorithms. This exercise is based on some of the tutorials of Tensorflow and Pytorch, and it is highly recommended for you to use these frameworks APIs. Please make sure to implement the requested parts by hand and use external examples wisely. You are requested to use **Tensorflow version 1.4 or above**, and **pytorch 0.4.1 or below**. links: {tensorflow, pytorch}. [1]

Exercise

Part I: Tensorflow

1. **Tensorflow basics** Read the tensorflow guide and go over the low level APIs, then answer the following questions:

   (a) In order to run a tensorflow program one should define a graph (i.e. tf.Graph), describe shortly the procedure of running a program using a graph and the purpose of graphs in tensorflow.

   (b) Tensorflow has introduced a feature that enables the users to execute a program without compiling a graph, describe this option shortly.

   (c) Tensorflow graphs that contain dependent operations (for example: $a = some\_input, b = 2 * a$, where the value of b depends on the assignment of a before), does not automatically ensure the correct order of the operations execution. Describe the tensorflow graph operation used to ensure the correct order of operation execution.

   In the next tensorflow sections you should use low level tensorflow operations, and not using 'keras'. Please define a Graph and run it using Session, do not use 'eager' operations.

[1] On the implementation tasks you can use the examples and descriptions by Peter Neilsen in {this link}, chapters 1,3,6.

2. **Tensorflow feed forward network** Implement a 2-layer feed forward network for MNIST classification, each layer should be sized 512 and with unlinear activation function using tensorflow. Use softmax layer for classification and the fitted cost function. Make sure to use dropout and consider more regularization methods to avoid overfitting. Use the entire training set and evaluate it over the $10,000$ test samples. Add a table of results with the training and test accuracy results for each 10 epochs.

3. **Tensorflow CNN** Implement a 2-layers convolutional neural network with a single fully connected layer and softmax layer for classification on top of it, using Tensorlfow. Use 20 filters at each layer, the convolution filter kernels should be of size 5x5. Max pooling and RELU activation for the convolutional layers are recommended. Make sure to use dropout and consider more regularization methods to avoid overfitting. Use the entire training set and evaluate it over the $10,000$ test samples. Add a table of results with the training and test accuracy results for each 10 epochs.

Part II: Pytorch

1. Learning Pytorch basics

   (a) **Pytorch tutorial** Read through the {pytorch tutorial} and go over the examples.

   (b) **Pytorch feed forward network** Based on the examples from the previous tutorial, implement a 2-layer feed forward network for MNIST classification, each layer should be sized 512 and with unlinear activation function. Use softmax layer for classification and the fitted cost function. Make sure to use dropout and consider more regularization methods to avoid overfitting. Use the entire training set and evaluate it over the $10,000$ test samples. Add a table of results with the training and test accuracy results for each 10 epochs.

   (c) **Pytorch CNN** Implement a 2-layers convolutional neural network with a single fully connected layer and softmax layer for classification on top of it, using Pytorch. Use 20 filters at each layer, the convolution filter kernels should be of size 5x5. Max pooling and RELU activation for the convolutional layers are recommended. Make sure to use dropout and consider more regularization methods to avoid overfitting. Use the entire training set and evaluate it over the $10,000$ test samples. Add a table of results with the training and test accuracy results for each 10 epochs.