Introduction to Deep Learning for Computer Vision
Assignment 3: Multi-layer Perceptrons

Due: March 2, 2018

Abstract

This assignment is designed to provide you with first-hand experience on how `TensorFlow` works. We will continue on from assignment 2, and this time use `TensorFlow` to implement a classifier using the Multi-layer Perceptron.

Instructions are provided first for a reason.
Please read the instructions carefully.

1 Instructions

1.1 Submission package

Within the homework assignment package, there should be a `submission-package.zip`, which contains the directory structure and empty files for you to get started. Please edit the contents within the file for code, and then create a `zip` archive with the file name `submission-package.zip`, and submit it. **Do not use other archive formats such as `rar` or `tar.gz`.** Also, submit the report in `pdf`, **as a separate file alongside** the archive with the codes.

All assignments should be submitted electronically. Hand written reports are **not** accepted. You can, however, include scanned pages in your report. For example, if you are not comfortable with writing equations, you can include a scanned copy.

1.2 Assignment Report

Even for programming assignments, all results should be summarised in a assignment report. Reports should be in `pdf` format. Though not mandatory, students are encouraged to submit their reports written in LaTeX. In the assignment package, you should have been given an empty skeleton file for you to get started.

However, it is **not required** for you to explain your code in the reports. Reports are for discussing results. You **should** however, provide comments in your code, well enough to be understood by directly reading the code.

1.3 Code

All assignments should be in `Python` 3. Codes that fail to run on `Python` 3 will receive 20% deduction on the final score. In other words, do **not** use `Python` 2.7.

For this assignment, you should **not** need to create additional files. Fill in the skeleton files in the submission package. Do **not** change the name of these scripts.

It is **strongly encouraged** to follow `PEP8`. It makes your code much more readable, and less room for mistakes. There are many open source tools available to automatically do this for you.

1.4 Delayed submission

In case you think you will not meet the deadline due to network speed or any other reasons, you can send an email with the `SHA-256` hash of your `.zip` archive first, and then submit your assignment through email later on. This will **not** be considered as a delay.

Delayed submissions are subject to 20% degradation per day. For example, an assignment submitted 1 minute after the deadline will receive 80% of the entire mark, even if it was perfect. Likewise, an assignment that was submitted one day and 1 minute after the deadline will receive 60%.

1.5 Use of open source code

Any library under any type of open source license is allowed for use, given full attribution. This attribution should include the name of the original author, the source from which the code was obtained, and indicate terms of the license. Note that using copyrighted material without an appropriate license is not permitted. Short snippets of code on public websites such as StackOverflow may be used without an explicit license, but proper attribution should be given even in such case. This means that if you embed a snippet into your own code, you should properly cite it through the comments, and also embed the full citation in a LICENSES file. However, if you include a full, unmodified source, which already contains the license within the source file, this is unnecessary. Please note that without proper attribution, *it will be considered plagiarism.*

In addition, as the assignments are intended for you to learn, (1) if the external code implements the core objective of the task, no points will be given; (2) code from other CSC486B/CSC586B students will count as plagiarism. To be more clear on (1), with the assignment, we will release a `requirements.txt` file, that you can use with `pip` to setup your environment. On top, you are also allowed to use `OpenCV3.X`, which we will not include in `requirements.txt` as `OpenCV` installation depends on your own framework.

2 Continuation from assignment 2

This assignment is a continuation from the second assignment. However, do not worry as the parts that were necessary for you to implement in the second assignment is already done for you in the assignment package.

However, in this assignment, we will not implement cross-validation, but rather will rely on simply a random split of the training set to 80% being the training set, and 20% being the validation set. Again you will need to test on the test set to report your final performance.

Another thing we will be doing is that we will implement our function as a `Python` class. This makes it much easier to re-use the model for both training and testing. This is because we construct the computational graph when the class is initialized, and use it for both training and testing.

3 Implementing `MyNetwork` (80 points)

In this assignment, we are going to implement our network as a `Python` Class object. The reason we want to do this is because of how `TensorFlow` operates. Remember that in `TensorFlow` we first declare our computation graph first, set everything up, and then run it. If you think about it, it nicely fits into the concept of object oriented programming. You initialize your object, then use class attributes to store things (e.g. your computation graph, placeholders to be used later, etc.), and execute things through class methods. This is what we will do.

Functions that could be used to implement your assignments are embedded in the comments! They may not be the complete set, but should definitely help!

3.1 `_build_placeholder` (0 points)

A function that creates the placeholders for data input. This part is already done for you, but do take a look.

3.2 `_build_preprocessing` (5 points)

Implement a function that creates the placeholders for reading and storing the training dataset's mean and range, and then create an `TensorFlow` operation that executes that assignment.

3.3 `_build_model` (15 points)

Implement the function to build your multi-layer perceptron. Allow the network generation to take configuration parameters such as the number of neurons in a layer, and the number of hidden layers. Also allow the configuration to decide which activation function you want to apply.

Optional: Try using batch normalization before the activation. This is purely optional, and there will not be additional points for this. However, it's just fun to play around with deep nets.

3.4 `_build_loss` (5 points)

Implement the loss function. We will stick to cross entropy this time.

3.5 `_build_optim` (10 points)

Implement the optimizer using Adam.

Optional: If you are using batch normalization, make sure that you are using control dependencies. See this link for more details.

3.6 `_build_eval` (5 points)

Implement the evaluation related `TensorFlow` operations. We will also create a summary at this point to see this through `TensorBoard`.

3.7 `_build_summary` (0 points)

This part has been done for you. We will simply gather all summary ops and create a single `self.summary_op` that we use later.

3.8 `_build_writer` (5 points)

Implement a function that creates summary writers and `TensorFlow` saver instances.

3.9 `train` (25 points)

We will reuse a lot from our previous assignment. Implement according to the comments in the assignment package.

3.10 `test` (10 points)

Implement the test function. You will need to load the best model you obtained through validation. Implement according to the comments.

4 Reporting (20 points)

As our final goal is to analyze the results we get, reporting is also a critical element. For this assignment, you will have to report the following mandatory items.

Note that we are now going to use `TensorBoard` for graph illustrations. You can draw multiple lines in the same graph by designating sub directories in a proper way. Do this to draw multiple experiments on the same graph when you are comparing different configurations.

4.1 Abstract (5 points)

Write a brief abstract summarizing the assignment, as well as your discoveries.

4.2 Activation functions (10 points)

Discuss your findings about how the convergence changes when you modify your activation function. Also discuss how the accuracy changes. When discussing, use the graphs you obtained through `TensorBoard`. You are required to compare `tanh` and `relu` for a single hyperparameter setting as minimum. You are free to do more if you want.

4.3 Network Architectures (5 points)

Also discuss how your network behaves according the different architectures that you have tried. If implemented correctly, you can easily alter the architecture with configurations. You are required to try minimum of two architectural choices with the same hyperparameter setting. You are free to do more if you want.