Homework 4

10-405/10-605: Machine Learning with Large Datasets

Due Friday, April 10th at 9:00 AM

.

1 Overview

The goal if this assignment is to gain familiarity with deep learning in TensorFlow, and to understand the basics of automatic differentiation—a key building block in TensorFlow and other deep learning frameworks. There are two parts to the homework.

In Part A, you will use an automatic differentiation system to implement a Multilayer Perceptron (MLP) for character level entity classification, using `python` and `numpy`. [Note: you **can't** use TensorFlow or any other deep learning frameworks in this part.]

In Part B, you will implement *neural style transfer* in TensorFlow. Neural style transfer will involve building a system that can take in two images (one content image and one style image), and output a third image whose content is closest to the content of the content image while the style matches the style of the style image.

1.1 Preparing for submission

We provide several public tests via `assert` in the notebook. You may want to pass all those tests before submitting your homework.

In order to enable auto-grading, please do not change any function signatures (e.g., function name, parameters, etc) or delete any cells. If you do delete any of the provided cells (even if you re-add them), the autograder will fail to grade your homework. If you do this, you will need to re-download the empty 'assignment_notebook.ipynb' file and fill in your answers again and resubmit.

1.2 Notebook editing and Submission

To download the notebooks of this assignment, go to the following links and open the notebook in **playground** mode so that you can edit and complete the notebook on colab.

1. Autodiff notebook
2. Tensorflow notebook

Make sure that you login to your CMU account when accessing to these notebooks.

Then go to `File -> Download .ipynb` to download the notebook to your local computer, and upload the 2 notebooks to the corresponding Gradescope grader.

Please make sure you name the notebook **assignment_notebook.ipynb** before you submit.

2 Background / Resources

Part A. In Part A, you will use an automatic differentiation system to implement a feedforward neural network or multilayer perceptron (MLP). To understand the details of this system, we encourage you to read the starter code provided in the notebook, as well as these notes: http://www.cs.cmu.edu/~wcohen/10-605/notes/autodiff.pdf.

Part B. In Part B, you will implement a style transfer system using TensorFlow. Recall that in the lecture on ML frameworks + TensorFlow we introduced an Iris dataset classification example where we used Keras Sequential API to design layers of the model. In this part you will use another TensorFlow API (Functional API) to build your model.

The Keras functional API is a way to create models that are more flexible than the tf.keras.Sequential API. The functional API can handle models with non-linear topology, models with shared layers, and models with multiple inputs or outputs. The main idea that a deep learning model is usually a directed acyclic graph (DAG) of layers. So the functional API is a way to build graphs of layers.

You can refer to the workflow introduced in the slides and design your model using Functional API in Part B. Detailed instructions are provided in the notebook.

3 Part A: Autodiff & MLPs

In this section you will go through the steps for building an MLP to classify a DBPedia dataset. You need to predict the category label of a DBPedia entity based on its title. The data format is different from the earlier assignments. The data contains two columns separated by tab, with the title in the first column and label in second. Detailed instructions are included in the assignment notebook.

We provide you the following starter codes:

- xman.py - classes for expression manager, registers and operations.
- utils.py - classes for data preprocessing and forming minibatches (more on this below). registers and operations.
- functions.py - function definitions and their gradients are declared here.

- autograd.py - class for forward and backward propagation over a Wengert list.

You should read the starter codes to get a thorough understanding of how to implement the MLP classifier and perform training. The instructions in the notebook will provide you templates to complete each subtask.

4 Part B: Neural Style Transfer in TensorFlow

In this part you will implement style transfer based on the paper: "A Neural Algorithm of Artistic Style": https://arxiv.org/pdf/1508.06576.pdf.

Basically, we will build a system that can take in two images (one content image and one style image), and output another image whose content is closest to the content of the content image while style is closest to the style of the style image.

There are 5 parts of the assignment:

- **Visualize data**
- **Process the data**
- **Set up loss functions**
- **Create Model**
- **Optimize for loss function**

The first part (visualizing the data) is written for you.

The second part has following functions:

1. load_and_process_img()
2. deprocess_img()
3. Define content and style representations
   we need to define the content layers and style layers
4. Build the Model
   Use tf.keras.applications.vgg19.VGG19 get style_outputs and content_outputs

The third part is to Define and create the Loss functions

1. compute_cotent_loss()
   Use tf.reduce_mean and implement the formula

$$L^l_{content}(p, x) = \sum_{i,j}(F^l_{ij}(x) - P^l_{ij}(p))^2$$

2. gram_matrix()

   The gram matrix, G^l_{ij} is the inner product between the vectorized feature map i and j in layer l. Remember to divide the gram_matrix by input_tensor.shape[0]

3. compute_style_loss()

   We need to implement the formula

$$E_l = \sum_{i,j} (G^l_{ij} - A^l_{ij})^2$$

Now that we have defined the loss functions, we need to develop the fourth part which consists of following functions:

1. compute_features()

   This function has following arguments:

   model: The model that we are using.

   content_path: The path to the content image.

   style_path: The path to the style image

   The function returns:

   the style features and the content features.

2. calculate_loss()

   Returns:

   total_loss, style_loss, content_loss

   We will give more emphasis to deep layers. For example, w for 'block1_conv1' can be 1, then weight for 'block2_conv1' can be 2, and so on weight_per_style_layer = [1.0,2.0,3.0,4.0,5.0]

The final part is to optimize the loop so that we can get the best stylized image.