E0 270: Assignment 1

Due date: March 31, 2024

Instructions

1. Download and extract the assignment template. Download your test images after providing your 5 digit SR number from http://10.192.30.80:8000/download and move it inside the data folder of the assignment template.
2. Setting up the environment, running into technical problems, and figuring out their solutions is a part of the learning process. Use Google, Stack overflow, and discuss with each other. Unfortunately, we will not be able to help you.
3. The report must be typeset in LaTeX, Ensure that everything is in a single pdf file and all pages are in correct order.
4. You must use PyTorch, however we are not imposing any version mandates. It should be fine as long as you use the latest stable version.
5. If you feel any particular problem in underspecified, you can safely assume that it has been done intentionally. You are free to make assumptions, but please specify the assumptions in your report. No further clarifications on the problem will be provided. However feel free to contact us if you believe that there is an error in the problem specification.
6. The grading will be relative based on the final performance achieved by your model. You are encouraged to discuss the problems amongst yourselves and engage in a healthy competition. However copying codes from others verbatim, along with the use of LLM assisted code generators are strictly forbidden, and will be penalized heavily if detected.
7. Use the documentation to read about the common type of layers that are offered by the standard frameworks like PyTorch. This will help you in understanding various deep learning concepts that will in turn allow you to improve the performance of your models.

Data

Consider the CIFAR-10 dataset https://www.cs.toronto.edu/ kriz/cifar.html. Each data point is an $3 \times 32 \times 32$ image, whose features are the individual pixel values when a linear model (multi-class logistic regression) is being considered, otherwise its the RGB channels when considering a Convolutional Neural Network (CNN).

Problem 1: Softmax Regression

1. Browse through the template code that you have been provided. Specifically, look at files `run.py`, `utils.py`, and all the files under the folder `LogisticRegression`. Fill up the required sections of the code (marked with TODO/NotImplementedError) to execute the command `python run.py <5 digit SR#>`
   This will let you know that binary classification (with logistic regression) is working properly. You should get $> 80\%$ accuracy on the validation set.

2. Change all the remaining missing sections of the code to implement multi-class logistic regression (softmax regression).
   Execute: `python run.py <5 digit SR#> --mode softmax <other hyper-params here>`
   Specify the cost function used by you in the report along with a breif write-up on the training strategy, i.e. things like: How was the batch size chosen? How were the images sampled? What was the criteria for stopping the training? And so on. Plot the loss and accuracy as a function of the number of iterations and save them as "1.1a.png" and "1.1b.png" respectively.

Problem 2: Contrastive Representation Learning

1. Now look at the files under the folder `ContrastiveRepresentation`. Set up a neural network (Encoder in `ContrastiveRepresentation/model.py`) that maps an image to a vector. Let's call this network as A. You are free to chose **(i)** the dimension d_{out} of the output vector, and **(ii)** the overall architecture of the network. It is up to you to make $Network\ A$ fully connected or Convolutional or any other variant. Specify the architecture you've used in the report (like the value of d_{out}, architecture type, number of layers, number of units in each layer, activation function used, and any other design choice).

2. Fill in the functions in the template files as and when required. Let I_a and I_b be two randomly sampled images from the training set having the same label (these are often referred to as Anchor and Positive samples respectively). Sample an image I_c (also called a Negative sample) such that I_c has a different label. Train $Network\ A$ to maximize similarity between v_a and v_b while simultaneously minimizing similarity between v_a and v_c, where v_a, v_b, and v_c are the $d_{out}-$dimensional output vectors obtained from $Network\ A$ corresponding to the inputs I_a, I_b, and I_c respectively. Specify the cost function used by you in the report along with a brief write-up on the training strategy, i.e. things like: How was the batch size chosen? How were the images sampled? What was the criteria for stopping the training? And so on.
   Execute: `python run.py <5 digit SR#> --mode cont_rep <other hyper-params here>`
   Plot the cost as a function of the number of iterations and save it as "2.2.png".

3. Once $Network\ A$ has been trained, iterate over the entire training set and obtain the corresponding trained vectors for all images. Plot these vectors on a two dimensional t-SNE plot and save it as "1.3.png". The points in the plot should be colored based on the ground truth labels. Have a legend specifying which color corresponds to which digit. Write as briefly as possible about your observations from this t-SNE plot.

4. Freeze the learned vectors and use them as features to train a multi-class logistic regression classifier (use Softmax Regression from Problem 1). Report the accuracy on the training set. Now take all examples from the validation set of CIFAR10, obtain the vectors corresponding to them by using the already trained logistic regression classifier (again, please don't train the classifier using examples from the validation set) and report the accuracy on the validation set. Specify the values of all hyperparameters that were used by you.
   Execute: `python run.py <5 digit SR#> --mode fine_tune_linear <other hyper-params here>`

5. A multi-class logistic regression classifier is almost equivalent to a single layer neural network. Now take the pretrained $Network\ A$ and add one more layer to it having 10 units (one for each of the digits). Let us call this network as $Network\ B$. Note that a portion of $Network\ B$ has to be initialized using the trained weights of $Network\ A$, while the last layer has to

be randomly initialized. Train *Network B* using the negative log likelihood cost function on examples from the training set. As before, plot the value of the cost function of training iterations in a file called "2.5.png". and report the classification accuracy of *Network B* on the training and validation sets. Also specify the values of various hyperparameters used by you for the training. This process is called finetuning. Did you obtain better results in Problem 2.4 or in Problem 2.5? Why?
Execute: `python run.py <5 digit SR#> --mode fine_tune_nn <other hyper-params here>`

6. You may add more layers to the classifier to improve the performance of your model. Submit the predicitions generated by your best model on http://10.192.30.80:8000/submit to check your standings amongst your peers on the learderboard (http://10.192.30.80:8000).

Reference: Supervised Contrastive Learning

Deliverables

- Completed code for implementing the Softmax regression algorithm without using any libraries except `numpy` in the given incomplete code snippets.
- Completed code for the Contrastive Representation Learning and associated fine-tuning. In this case however, you must use pytorch library.
- A report containing the complete details of the algorithm and your implementation, along with justification for any extra assumptions made if necessary. Since the dataset is perfectly balanced among all classes, reporting just the losses and accuracies should be sufficient.
- Training plots corresponding to loss and accuracies for both train and validation set. Also mention your best test accuracy obtained on the leaderboard.

Submission

Attach **a single zip file** named in the format `Asst1_FirstName_LastName_5DigitsOfSRNo.zip` to the assignment in Teams, before the due date. The zip file should contain your code that you filled up, your report (in a pdf file named `Report_<5digitSR>.pdf`), and any plots that you put in the report. Ensure to upload the zip file on Teams submission portal on or before March 31, 2024, 11:59 pm. Ensure that you do **not** include the data files and pycache files in the code that you submit.