CS 5600/6600: F24: Intelligent Systems
Assignment 4

Vladimir Kulyukin
Department of Computer Science
Utah State University

September 21, 2024

Learning Objectives

1. Reflections on Deep Learning (DL)
2. Building, Training, Testing, Validating Nets with PyTorch
3. Synthetic Data Generation

Introduction

We're coming to the end of the data-driven AI part of our course. We're not there yet, but I want you to start reflecting on what we've learned about NNs. The programming component of this assignment (Problems 2 and 3) will give you an opportunity to build, train, and test multi-layer ANNs and CNNs with PyTorch.

1 Coding Lab (0 points)

Complete the coding lab in `CS5600_6600_F24_HW04_CodingLab.pdf`.

Paper Analysis (2 pts)

This week's paper is "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images" by Nguyen, Yosinski, and Clune. The PDF is in the zip. This paper was published in 2015 and sparked a lot of controversy, because its findings poured some cold water on the deep learning hype that was beginning to spin out of control at the time. These researchers showed that it may be possible to automatically generate images that human observers cannot recognize at all, but deep NNs classify, with a very high degree of certainty, as objects on which they were trained.

I learned about this paper from a student who was at the time (a few years ago) in my scientific computing class. He sent me a short Hey-Dr-K-Did-You-See-This? email with a link to this paper. This is what university is all about – open knowledge sharing and open discussion. I learn as much, and often more, from my students, as, I hope, they learn from me. I was grateful to this student, because I found this article to be very thought provoking. In my humble opinion, any serious student of deep learning should carefully study this article and take heed of its findings and possible implications on the limitations of DL.

I know, I know — hopium is a powerful drug, because we're neurolinguistically conditioned to believe in perpetual linear scientific progress (although History and Anthropology whisper to us many cautionary tales – cf., e.g. "The Collapse of Complex Societies" by Dr. J. Tainter). Be it as it may, we cannot brush aside as nonsense these researchers's arguments and findings. So, read this paper with an open mind and write a one-page analysis of it. Save your analysis in `deep_nets.pdf` and include it in your submission in Canvas.

You don't have to agree with the authors' views or my views. I welcome alternative views (so long as they're well-articulated and well-argued). I always look forward to your analyses and enjoy reading and commenting on them.

Problem 2 (1 pt)

I wrote the stubs of six ANN building functions in `mlp_hw.py` for you to complete. Each function asks you to build a specific ANN. These should be straightforward after you complete the coding lab in Problem 0. Three functions ask you to build ANNs with the ReLU activation functions and the other three – with Sigmoids.

Now implement the function `train_mlp` and `test_mlp` in `train_eval_mlp_hw.py`. The first function takes an ANN model, an optimizer, a loss function, and number of epochs and trains the model with the specified optimizer and the loss function for the specified number of epochs. Again, these should be straightforward after the coding lab. On the random dataset I generated for you in `train_eval_mlp_hw.py`, i.e.,

```
### prepare 2000 samples with 10 features and 5
### centers with the cluster STD = 1.75.
(X, y) = make_blobs(n_samples=2000,
                    n_features=10,
                    centers=5,
                    cluster_std=1.75,
                    random_state=13)
```

one of my sample runs looked like this.

```
>>> ReLU_MLP1 = mlp_hw.build_10_20_5_mlp_relu_model().to(DEVICE)
### LR abbreviates learning rate (eta).
>>> ReLU_OPT1 = SGD(ReLU_MLP1.parameters(), lr=LR)
>>> lossFunc = nn.CrossEntropyLoss()
>>> train_mlp(ReLU_MLP1, ReLU_OPT1, lossFunc, 5)
------------ TRAINING ANN ----------------
<DBG>: epoch: 1...
epoch: 1 train loss: 1.418 train accuracy: 0.786
<DBG>: epoch: 2...
epoch: 2 train loss: 1.194 train accuracy: 0.768
<DBG>: epoch: 3...
epoch: 3 train loss: 0.979 train accuracy: 0.696
<DBG>: epoch: 4...
epoch: 4 train loss: 0.796 train accuracy: 0.821
<DBG>: epoch: 5...
epoch: 5 train loss: 0.640 train accuracy: 0.893
>>> test_mlp(ReLU_MLP3, lossFunc, 5)
------------ TESTING ANN ----------------
epoch: 1 test loss: 0.508 test accuracy: 0.927
epoch: 2 test loss: 0.508 test accuracy: 0.927
epoch: 3 test loss: 0.508 test accuracy: 0.927
epoch: 4 test loss: 0.508 test accuracy: 0.927
epoch: 5 test loss: 0.508 test accuracy: 0.927
```

I have written 6 unit tests for you in `mpl_uts.py`. You can use these to test your solutions for this problem.

Problem 3 (2 points)

This is a learn-by-experiment problem. I'd like you to improve the performance of my LeNet on KMNIST in the coding lab. Recall it was 95%. After you complete the lab, you'll know how to train and test LeNet. You don't have to change any source code in `lenet.py` or `predict_lenet.py`. All you need to do is to experiment with the following hyperparameters in `train_lenet.py`.

```
## iinitial learning rate
INIT_LR = 1e-3
## batch size
BATCH_SIZE = 64
## number of epochs
EPOCHS = 10
```

```
## train/test split
TRAIN_SPLIT = 0.75
VAL_SPLIT = 1 - TRAIN_SPLIT
```

In preparation for Project 1 (and Project 2, actually, if you choose to do DL for it), I'd like you to write a 2-page report documenting what parameters you played with and the accuracy/loss you got during training and validation. Experiment with at least 2 values of each hyperparameter. You don't have to be verbose in your report. You can include the generated plots (a picture is worth a 1K words!) and the classification reports that the code in `train_lenet.py` print, i.e.,

```
print(classification_report(testData.targets.cpu().numpy(),
                            np.array(preds),
                            target_names=testData.classes))
```

What to Submit

1. Your 1-page paper analysis saved in `deep_nets.pdf`.
2. `mlp_hw.py` and `train_eval_mlp_hw.py` with your code for Problem 2.
3. Your 2-page `LeNetKMNIST.pdf` report for Problem 3. You don't have to submit any source code for Problem 3.

Happy Reading, Writing, and Hacking!