CS 615 - Deep Learning

Assignment 3 - Convolutional Neural Networks
Spring 2020

Introduction

In this assignment we'll implement a simple convolutional neural network. We'll ease our way into it by first working on some synthetic data, then expanding to a real dataset.

Programming Language/Environment

While you may work in a language of your choosing, if you work in any languages other than Matlab, you **must** make sure you code can compile (if applicable) and run on tux. If you need a package installed on tux reach out to the professor so that he can initialize the request.

Allowable Libraries/Functions

In addition, you **cannot** use any libraries to do the training or evaluation for you. Using basic statistical and linear algebra function like *mean*, *std*, *cov* etc.. is fine, but using ones like *train*, *confusion*, etc.. is not. Using any ML-related functions, may result in a **zero** for the programming component. In general, use the "spirit of the assignment" (where we're implementing things from scratch) as your guide, but if you want clarification on if can use a particular function, DM the professor on slack.

Grading

Part 1 (Theory Questions)	10pts
Part 2 (CNN for LSE Classification)	20pts
Part 3 (CNN for MLE Classification)	20pts
Part 4 (CNN for LCE Classification)	20pts
Part 5 (CNN with Multiple Filters)	10pts
Part 6 (CNN for Yale Faces)	10pts
Report and Simplicity to Run	10pts
TOTAL	100pts

Datasets

Synthetic Data To get started we're going to play around with using simple CNNs on synthetic data to see if we can actually learn useful filters. In Parts 2-5 we'll create synthetic data and train CNNs with different architectures.

For each of these parts, first create two 40×40 "images". One is all back (zeros) except a white vertical stripe at a location of your choosing and the other is all black except a horizontal wite stripe at a location of your choosing. You can consider the first one to be from class 0, and the second one to be from class 1. Do this programmatically as part of your script.

Yale Faces Datasaet This dataset consists of 154 images (each of which is 243x320 pixels) taken from 14 people at 11 different viewing conditions (for our purposes, the first person was removed from the official dataset so person ID=2 is the first person).

The filename of each images encode class information:

$$\text{subject} < ID > . < condition >$$

Data obtained from: http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html

1 Theory

1. (2pts) Apply kernel K to data X. In other words, what is $X * K$?:

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \; K = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

2. Given the feature map filter output, $F = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 & 3 \\ 4 & 5 & 6 & 0 & 0 & 12 \\ 7 & 8 & 9 & 1 & 0 & 4 \\ -100 & -100 & -100 & -100 & -100 & -100 \end{bmatrix}$, what is the output from a pooling layer with width of 2 and stride of 2 if we are using:

   - (3pts) Max-Pooling?
   - (3pts) Mean-Pooling?

3. (2pts) Given an image X, what would the kernel K be that can reproduce the image when convoluted with it? That is, what is K such that $X * K = X$?

2 CNN for LSE Classification

Let's start attempting to differentiate between your two synthetic images using a simple CNN architecture. The CNN architecture is as follows. All hyperparameter choice are up to you:

1. A **single** 40×40 convolutional kernel.
2. A max-pool layer with $width = 1$ and $stride = 1$.
3. A flattened layer
4. A linear activation function
5. A squared error objective function.

What you will need for your report

1. Image representations of your initial and final kernels.
2. A plot of the RMSE as a function of the number of iterations.
3. Your hyperparameter choices.

3 CNN for MLE Classification

Next we'll change our CNN architecture to use a sigmoid activation function and log likelihood objective function. Therefore, the CNN architecture is as follows. Once again, all hyperparameter choice are up to you:

1. A single 40×40 convolutional kernel.
2. A max-pool layer with $width = 1$ and $stride = 1$.
3. A flattened layer
4. A sigmoid activation function
5. A log likelihood objective function.

What you will need for your report

1. Image representations of your initial and final kernels.
2. A plot of the mean log likelihood as a function of the number of iterations.
3. Your hyperparameter choices.

4 CNN for LCE Classification

One more basic architecture.... Change our CNN architecture to use a softmax activation function and cross-entropy objective function. Therefore, the CNN architecture is as follows. Once again, all hyperparameter choice are up to you:

1. A single 40×40 convolutional kernel.
2. A max-pool layer with $width = 1$ and $stride = 1$.
3. A flattened layer
4. A softmax activation function.
5. A cross-entropy objective function.

What you will need for your report

1. Image representations of your initial and final kernels.
2. A plot of the mean cross-entropy loss as a function of the number of iterations.
3. Your hyperparameter choices.

5 CNN With Multiple Kernels

Let's play around with the convolutional and pooling layers now. Here's the architecture:

1. Four 5×5 convolutional kernels.
2. A max-pool layer with $width = 2$ and $stride = 2$.
3. A flattened layer
4. Your choice of activation and objective function!

What you will need for your report

1. Image representations of your final kernels.
2. A plot of the your objective function as a function of the number of iterations.
3. Your hyperparameter choices.

6 Multi-Kernel CNN For Image Classification

Finally let's open this up to our Yale Faces problem.

At this point *all design decisions are up to you*. The only constraints are that you must have at least one convolutional layer and that you try at least two different architectures.

What you will need for your report For each architecture.

1. The architecture.
2. The values of any additional hyperparameters that you chose (outside of the architecture decisions).
3. Plot of iteration vs objective function evaluation.
4. Final training and testing accuracies.

Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF Writeup
2. Source Code
3. readme.txt file

The readme.txt file should contain information on how to run your code to reproduce results for each part of the assignment.

The PDF document should contain the following:

1. Part 1:
   (a) Your solutions to the theory question
2. Part 2:
   (a) Image representations of your initial and final kernels.
   (b) A plot of the RMSE as a function of the number of iterations.
   (c) Your hyperparameter choices.
3. Part 3:
   (a) Image representations of your initial and final kernels.
   (b) A plot of the mean log likelihood as a function of the number of iterations.
   (c) Your hyperparameter choices.
4. Part 4:
   (a) Image representations of your initial and final kernels.
   (b) A plot of the mean cross-entropy loss as a function of the number of iterations.
   (c) Your hyperparameter choices.
5. Part 5:
   (a) Image representations of your final kernels.
   (b) A plot of the your objective function as a function of the number of iterations.
   (c) Your hyperparameter choices.
6. Part 6:
   For each architecture.
   (a) The architecture.
   (b) The values of any additional hyperparameters that you chose (outside of the architecture decisions).
   (c) Plot of iteration vs objective function evaluation.
   (d) Final training and testing accuracies.