Homework 5
M1522.001000 Computer Vision (2016 Spring)
Due: Monday May 30, 11:59PM

In this assignment you will be implementing Convolution and Pooling for CNN(Convolutional Neural Network). Your code will be able using for feature extraction and classification for images. By running your code on the test dataset you will learn about how the CNN works.

Many of the algorithms are already implemented. In this assignment you don't have to implement many functions or algorithms. You should complete just 4 functions. This assginment mainly focus on running and understanding CNN.

1 Submitting Your Assignment

Your submission for this assignment consists of answers to two theory questions, the code for your MATLAB implementation and a short writeup describing any thresholds and parameters used, interesting observations you made or things you did differently while implementing the assignment. You need to zip the following items, name it as `M1522001000_HW5_(your student ID)#.zip`, and send it to TA's email.

Your submitted zip file should include the files arranged in this layout:

- `theory.txt` or `theory.pdf`
- `experiments.txt` (or `.pdf`): answers to the theory questions in Section 2
- Folder `matlab`
  - `given codes`: not to include the `.mat` data files.

Your zip file should be sent by due date(Monday May 30, 11:59PM). Later than that, you will use one late day.

2 Theory Questions

Question 1: Convolutional Neural Network (20 points)

AlexNet(One of CNN) CNN code is 4096 dimension vector. Explain how to get 4096 dimension vector. Your answer should include explanation of Convolution and Pooling layer with dimension calculation.

Question 2: Pooling (10 points)

What is difference between Min, Mean and Max pooling.

3 Programming

We have included a wrapper script named `cnnExercise.m` that will use the features you learned on 8x8 patches sampled from images from the STL-10 dataset for classifying images. The reduced STL-10 dataset comprises 64x64 images from 4 classes (airplane, car, cat, dog). You should write your code at the places indicated "YOUR CODE HERE" in the cnnConvolve.m, cnnPool.m, softmaxPredict.m and softmaxCost.m files.

Part 1 Implement convolution (30 points)

First, we want to compute $\sigma(Wx(r,c)+b)$ for all valid (r,c) (valid meaning that the entire 8x8 patch is contained within the image; this is as opposed to a full convolution, which allows the patch to extend outside the image, with the area outside the image assumed to be 0), where W and b are the learned weights and biases from the input layer to the hidden layer, and x(r,c) is the 8x8 patch with the upper left corner at (r,c). To accomplish this, one naive method is to loop over all such patches and compute $\sigma(Wx(r,c)+b)$ for each of them; while this is fine in theory, it can very slow. Hence, we usually use Matlab's built in convolution functions, which are well optimized.

Observe that the convolution above can be broken down into the following three small steps. First, compute Wx(r,c) for all (r,c). Next, add b to all the computed values. Finally, apply the sigmoid function to the resulting values. You can implement the loop in the first step with one of MATLAB's optimized convolution functions, conv2, speeding up the process significantly.

First, conv2 performs a 2-D convolution, but you have 5 `dimensions` - image number, feature number, row of image, column of image, and (color) channel of image - that you want to convolve over. Because of this, you will have to convolve each feature and image channel separately for each image, using the row and column of the image as the 2 dimensions you convolve over. This means that you will need three outer loops over the image number imageNum, feature number featureNum, and the channel number of the image channel. Inside the three nested for-loops, you will perform a conv2 2-D convolution, using the weight matrix for the featureNum-th feature and channel-th channel, and the image matrix for the imageNum-th image.

Second, because of the mathematical definition of convolution, the feature matrix must be `flipped` before passing it to conv2. The following implementation tip explains the `flipping` of feature matrices when using MATLAB's convolution functions.
`Implementation tip: Using conv2`

Next, to each of the convolvedFeatures, you should then add b, the corresponding bias for the featureNum-th feature.
However, there is one additional complication. If we had not done any preprocessing of the input patches, you could just follow the procedure as described above,

and apply the sigmoid function to obtain the convolved features, and we'd be done. However, because you preprocessed the patches before learning features on them, you must also apply the same preprocessing steps to the convolved patches to get the correct feature activations.

In particular, we give the following to the patches:
1. subtract the mean patch, meanPatch to zero the mean of the patches

Given code (bt = b - wt * meanPatch)
2. ZCA whiten using the whitening matrix ZCAWhite.

Given code (wt = W * ZCAWhite)

Taking the preprocessing steps into account, the feature activations that you should compute is $\sigma(W(T(x-\bar{x}))+b)$, where T is the whitening matrix and $\bar{x}$ is the mean patch. Expanding this, you obtain $\sigma(WTx - WT\bar{x} + b)$, which suggests that you should convolve the images with WT rather than W as earlier, and you should add $(b - WT\bar{x})$, rather than just b to convolvedFeatures, before finally applying the sigmoid function.

Test script automatically check your Convolution code in cnnExercise.m.

Part 2 Pooling (20 points)

Implement pooling in the function cnnPool in cnnPool.m. You should implement mean pooling (i.e., averaging over feature responses) for this part.

Test script automatically check your Pooling code in cnnExercise.m.

Part 3 Prediction (5 points)

Implement prediction in the function softmaxPredict in softmaxPredict.m. You should implement softmax prediction by using theta for this part.

Part 4 Cost (10 points)

Implement cost function in the softmaxCost in softmaxCost.m. You should implement softmax cost by given data, theta and parameters.

Note that you need to compute both gradient and cost in this part.

4 Experiments

(30 points)

Use the script included to run your CNN on the data set. Explain about result and what does it means. Include a `experiments.txt` or `experiments.pdf` file with your assignment submission describing what effect the parameters had and any improvements you made to your CNN to make it work better. Also draw multiple graphs (eg. training time - loss, parameter setting - accuracy) on your report.

Tip: Focus on Accuracy, dataset, and CNN classification.