Homework Assignment 3

CSE 253: Neural Networks

Winter 2016

Instructions

Due on Feb 9^{th} 11:59pm PST

1. You are required to have a group of 2-3 people to finish this homework assignment.
2. Please use a NIPS style template for your LaTeX report.
3. Please submit the homework electronically to **all** TAs' Emails (liam.fedus@gmail.com, lil121@eng.ucsd.edu, akdave@eng.ucsd.edu, mabiswas@eng.ucsd.edu).
4. You may use a language of your choice (Python and MATLAB are recommended) but all source code used in the assignment must be attached in the appendix. Please keep the code clean with explanatory comments, as they may be reused in the future.
5. Late policy: You will get deduction from the total score for: a) 0-1 day:10%; b) 1-2 days: 20%; c) 2-3 days: 50%; d) more than 3 days: 100%;
6. Any form of cheating, lying, or plagiarism will not be tolerated. Discussions on course materials and homework solutions are allowed, but you should write the final solutions alone. Books, notes, and Internet resources can be consulted, but not copied from. Working with people outside your team on homeworks must follow the (spirit of) Gilligan's Island Rule (Dymond, 1986): No notes can be made during a discussion, and you must watch one hour of Gilligan's Island or equally mindless TV before writing anything down. Suspected cheating will be reported to the Dean.

Convolutional Neural Networks

In this assignment, you will design and implement a Convolutional Neural Network (CNN) in one of the modern deep learning frameworks, Caffe. We will use the CIFAR-100 Dataset for this project. Please complete the following tasks:

1. **Amazon Web Services (AWS).** You are going to run all experiments remotely using Caffe on Amazon Web Services (AWS). This is a great opportunity to become more familiar with Cloud Computing if you are not already!
   (a) **Logon to AWS**. Go to https://cse253.signin.aws.amazon.com/console and login with the User Name you provided us. Your password on initial login will also be your User Name.
   (b) **Set up Account**. Click on *EC2*, then click on *AMIs* under the Images tab. Note that you will be running on one of the three US images: N.Virigina, N. California, and Oregon. We have only been allocated 10 g2.2xlarge instances for each region, but we will expand it as necessary. If you see a region already has many instances running, please switch to one of the other two!
   (c) **Launch EC2 Instance**. Launch *Caffe and CuDNN AMI*. An Amazon Machine Image (AMI) is a template necessary to launch a virtual server in the cloud, an EC2 Instance. You will need to choose an Instance Type (this is the hardware you are requesting in the cloud). Please select *g2.2xlarge* then click *Review and Launch*. Everything should be fine, then click *Launch*.

Layer	Type	Input Size	Kernel Size	# Filters	Nonlinearity	Pooling	Stride	Size	Output Size	Parameters
1	Conv	32*32*3	3*3	48	ReLU	Average	2	2*2	16*16*48	1344
2	FC	16*16*48	1*1		ReLU				100*1	1228900
...										

Table 1: Example of network structure

(d) **Create Key Pair**. If this is your first time launching an instance, please select *Create a new key pair*, generate a *Key pair name* and save the private key file (*.pem) to a local place on your computer. This file will be needed on your each of your team members' computers in order to access this Image. Once you've done this, finally click *Launch Instances*. Your instance will appear as initiated with some Instance ID. You can rename it as you like.

(e) **Remote into EC2 Instance**. Change the Permissions on your .pem file. Go to that directory and execute,

```
$ chmod 400 your-key-name.pem
```

Now, refer to the running instance on EC2. You can use the Public DNS to access this remote server. To ssh in, use the following,

```
$ ssh -i your-key-name.pem ubuntu@ec2...amazonaws.com
```

2. **Load the data**. Download the CIFAR-100 dataset and put it in your /home/ubuntu/caffe/data directory. To train a network using Caffe, it is advisable to transform your data into a LMDB file or simply .txt file with the path and label of an image for each line (See here for how data layer is constructed). It is your job to transform the images into the acceptable format for Caffe.

3. **Build your network**. You can refer to the network in Caffe's CIFAR-10 tutorial. Your network should have at least 3 convolutional layers and 1 fully connected layer. Describe your network using a train_val.prototxt file, and show the table in Table 1's format for each layer of your network.

4. **Train your network**. Use Caffe to train your network based on your network structure. Report your training parameters in solver.prototxt. Describe your training procedure. Plot the following:

   (a) Training and test loss (not classification accuracy!) vs. training iterations.

   (b) Classification accuracy on the test set vs. training iterations.

5. **Experiment with preprocessing the input data**. Start from the network in the previous part. Subtract the mean and perform global contrast normalization of the images. Try augmenting your dataset by randomly cropping and mirroring your images. Train your network.

   (a) Plot classification accuracy on the test set vs. training iterations. Comment on the change of performance. Does the network train faster? Generalize better? Discuss your result.

   (b) Change the number of training images per category to 100 and 300. Plot classification accuracy on the test set vs. training iterations. Comment on change of performance.

6. **Experiment with optimization methods**. Devise experiments to test the performance of various optimization methods on your model.

   (a) Stochastic Gradient Descent (type: "SGD")

   (b) Adaptive Gradient (type: "AdaGrad")

   (c) Nesterov's Accelerated Gradient (type: "Nesterov")

   (d) RMSprop (type: "RMSProp")

   At a minimum, show training loss curves and timing information. Which what was the most computationally efficient? Did all models learned generalize well to new test data?

7. **Experiment with network structure**. Some people argue that deep convolutional networks are successful because they simply have many more parameters than a shallow network. Here we want to test whether depth really matters if the networks have similar amount of parameters. Calculate the total amount of parameters in your network in Part 3 and show your calculation. Then, try the following:

   (a) Create a network that has at least one more hidden layer than your network in Part 3. Decrease the number of feature maps and/or the size of kernels to have the amount of trained parameters be similar to your network in Part 3.

   (b) Train this modified network and plot classification accuracy on the test set vs. the number of training iterations. Comment on the change of performance. Is depth important?

8. **Experiment with network fine-tuning**. Read the Caffe fine-tuning tutorial `http://caffe.berkeleyvision.org/gathered/examples/finetune_flickr_style.html`. Now we want to see whether the features of our network can generalize to other datasets. Here we use CIFAR-10 as the new dataset to be fine-tuned. Make the weights and bias of your network (except the last layer) unchanged, and train a new softmax classifier on top using the training images from the new dataset. Choose your own stopping criteria. Please complete the following tasks:

   (a) Plot training and test loss vs. training iterations.

   (b) Plot classification accuracy on the test set vs. training iterations.

   (c) Does fine-tuning improve generalization?

9. **Feature visualization.** Follow the tutorial of feature visualization, visualize the layer 1 and layer 2 filters of your network. Do they look similar? (Note: you may have to compile Caffe's python interface to run the ipython notebook.)