CS 291K – Advanced Data Mining, Spring 2016
Machine Problem 2
(100 points)
Due *Wednesday, May 4, 2016, 11:59pm*

Notes:

- Make sure to re-read the "Policy on Academic Integrity" on the course syllabus.
- Updates or corrections will be posted on the Announcements page in web page of the course, so check there occasionally.
- You have to work individually for this assignment.
- Each student must turn in their report and code electronically.
- Responsible TA for Machine Problem 2: Honglei Liu honglei@cs.ucsb.edu

---

1. Problem Definition

For this assignment, you will be given the **CIFAR-100** dataset. This dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 50000 training images and 10000 testing images in total. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. You need to implement a classifier, which is a convolutional neural network (ConvNet), **for the 20 superclasses**. The output of this ConvNet is a 20 dimensional vector representing the predicted probabilities of an image belonging to each superclass. You can either implement it from scratch or using TensorFlow. If you **have to** use other tools such as Caffe, Torch and Lasagne, please contact me to make other arrangements. The architecture of the ConvNet is not given, you can use any architecture you want but you have to train it properly using the training data and report its accuracy on the test data.

2. What you are given

1) The CIFAR-100 dataset
This dataset is divided into a training set with 50000 images and a test set with 10000 images. Each image comes with a "fine" label (the class to which it belongs, 100 in total) and a "coarse" label (the superclass to which it belongs, 20 in total). **For this assignment, you only need to consider the "coarse" label (the superclass).** So, when you load the data, pay attention to the label you use.

You can check the descriptions of this dataset on https://www.cs.toronto.edu/~kriz/cifar.html. Depending on the programming language you are using, you could also find links to download Python, MATLAB or binary version of the dataset on the website. Again, use the CIFAR-100 dataset, **NOT** the CIFAR-10 dataset.

2) A python program to load the dataset
You will be given a python file named "data_utils.py" within which there is function called "load_CIFAR100" to load all the training and test data from the dataset directory. The input of this function is the python version dataset directory. The outputs of this function are 4 numpy arrays: Xtr, Ytr, Xte, Yte. Xtr contains the training images, which is a 50000x32x32x3 dimensional array because there are 50000 training images and each image has a size of 32x32 with 3 color channels. Ytr is a 50000x1 array containing the training labels in the range of 0 - 19. Similarly, Xte and Yte contain the test images and test labels.

3. What you need to do

1) Implement the ConvNet
You need to implement the ConvNet either from scratch or using TensorFlow. Depending on the programing language you are using, you should name the code file as "conv_net.py" / "conv_net.m" / "conv_net.cpp" / "conv_net.java". There's no requirement on the structure of this file, but you should try to make your implementation easily generalizable to different ConvNet architectures. In addition to native libraries, only TensorFlow is supported in our test environment. If you are using other tools, please contact me for other arrangements.

2) Train the ConvNet and report your best accuracies
You need to train the ConvNet using the training data and report the accuracies for the training data, validation data and test data in the assignment report. You should report the top-1 accuracy, which only considers the label of an image is correctly predicted if it's the one with the highest probability in the output. **Please do not train your model or tune your hyperparameters on the test data. This will be treated as cheating!** A good practice would be leaving out a small set of training data as validation set or using *k*-fold cross-validation to tune your hyperparameters. Since convolutional neural nets usually take very long time to train well, you can get full credit for this part as long as you can get above **30%** accuracy (takes about half an hour to train using CPU) for the validation and test set.

3) Make choices about the hyperparameters
Since the architecture of the ConvNet is not given, you need to make decisions on what architecture you want to use. You also need to make decisions about several other hyperparameters. For example, you need to decide on the number of filters, the filter size, the regularization strength, learning rate, number of iterations, batch size and the number of hidden neurons for the fully connected layer. It's ok that you happen to find a set of parameters that works very well, but for the following hyperparameters we would expect some empirical results in the report to support you decisions.

• **architecture of the ConvNet**: Results using different number of layers.

• **number of filters**: A figure of number of filters v.s. accuracy for training and validation set.
• **filter size in the convolution layer**: A figure of filter size v.s. accuracy for training and validation set.
• **regularization strength**: A figure of regularization strength v.s. accuracy for training and validation set.
• **learning rate**: A figure of training loss v.s. training steps.

These are just some suggestions. As long as you can justify your decisions, we are good.

4) Write a program to reproduce the training and testing
You need to write a program to automatically redo your training process with the best hyperparameters you find and report the accuracies. You should name your source code file as "redo.py" / "redo.m" / "redo.cpp" / "redo.java". This program takes one command-line argument which is the path of the data directory. For example, if the data directory is *"dataset"*, the python version CIFAR-100 data files are located in "*dataset/cifar-100-python"*, the matlab version CIFAR-100 data files are located in "*dataset/cifar-100-matlab"* and the binary version CIFAR-100 data files are located in "*dataset/cifar-100-binary".*
Your code should run as
• For Python
```
% python redo.py dataset
```
• For MATLAB
```
% matlab -r "runsort('dataset')" -nodisplay
```
• For C/C++
```
% make
% ./redo dataset
```
• For Java
```
% make
% java –jar redo.jar dataset
```

The program should print the accuracies for the training, validation and testing dataset to the console after some outputs for the training process. For example, the outputs should look like

```
…
Some outputs for the training process
…
Training accuracy: 50%
Validation accuracy: 47%
Testing accuracy: 45%
```

Your program should finish within 3 hours using CPU only. We would expect that results generated by this program do not differ a lot from the ones in your report.

5) Write a report

As for the report, it should be between 1 and 4 pages in length (no more than 4 pages) with at least 11pt font size and should consist of at least the following sections:

- Implementation: Brief explanations of your code (briefly describe the key points of your implementation)
- Architecture: Specify the architecture of ConvNet you used.
- Model Building: How you train the ConvNet and choose the hyperparameters.
- Results: Your results on the provided datasets (training time, accuracy).
- Extra Credits: Things you did to earn extra credits. (optional)
- Challenges: The challenges you faced and how you solved them
- Possible Improvements: Things you could do but haven't done to make your model better.

There should be only one pdf report which includes all the above (no additional readme file).

4. Tip for installing TensorFlow

1) **Install TensorFlow with Anaconda distribution of python**

Sometimes you may get errors like "Cannot remove entries from nonexistent file…". Try the following:

```
pip install --ignore-installed --upgrade
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-0.7.1-cp27-none-linux_x86_64.whl
```

2) **Get errors while importing TensorFlow**

If you are getting errors like "ImportError: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.17' not found…."

Try the following

```
### After tensorflow is installed
mkdir ~/my_libc_env
cd ~/my_libc_env
wget http://launchpadlibrarian.net/137699828/libc6_2.17-0ubuntu5_amd64.deb
wget http://launchpadlibrarian.net/137699829/libc6-dev_2.17-
0ubuntu5_amd64.deb
wget ftp://ftp.icm.edu.pl/vol/rzm5/linux-
slc/centos/7.0.1406/cr/x86_64/Packages/libstdc++-4.8.3-9.el7.x86_64.rpm
ar p libc6_2.17-0ubuntu5_amd64.deb data.tar.gz | tar zx
ar p libc6-dev_2.17-0ubuntu5_amd64.deb data.tar.gz | tar zx
rpm2cpio libstdc++-4.8.3-9.el7.x86_64.rpm| cpio -idmv

### run python program with tensorflow
LD_LIBRARY_PATH="$HOME/my_libc_env/lib/x86_64-linux-
gnu/:$HOME/my_libc_env/usr/lib64/" $HOME/my_libc_env/lib/x86_64-linux-
gnu/ld-2.17.so `which python` your_program.py
```

5. Instructions on What and How to Submit

Use the CSIL **turnin** command to submit the necessary files and directories. Note that all directory and file names are case sensitive. For this assignment you need to issue the following command:
% turnin mp2@cs291k mp2

For exchange students that do not have a CSIL account:
A. Obtain a *Proof of Registration* to the class, usually with a *Receipt from Extension*.
B. Go to *HFH1140E* with the *Proof of Registration* and someone at the help desk will get the account process started.

Once you have finished developing your code, copy all necessary files (your source code files and the pdf report) into a directory named “mp2”.

If you use C/C++, or Java, please prepare the necessary Makefile and ensure that your code can compile and execute as follows:

• For C/C++
```
% make
% ./redo dataset
```
• For Java
```
% make
% java –jar redo.jar dataset
```

Note: Put all the necessary files in a directory named *mp2* and submit the directory as a whole.

6. Grading

Grade Breakdown:
• Correctness of the implementation (60 points)
• Completeness of the report (20 points)
• Accuracy on validation set (10 points for greater than 30%)
• Accuracy on test set (10 points for greater than 30%)

7. Extra Credits

• Testing accuracy ranked top 1 (20 points)
• Testing accuracy ranked top 5 to top 2 (10 points)
• Try different optimization method and show its effects on the results (5 points)
• Use *dropout* and show its effects on the results (5 points)
• Try other initialization method and show its effects on the results (5 points)
• Try different filter sizes of the convolution layer and show its effects on the results (5 points)
• Try different filter sizes of the pooling layer and show its effects on the results (5 points)
• Try different number of hidden neurons for the fully connected layer and show its effects on the results (5 points)

Plagiarism Warning: We are going to use software to check plagiarism. You are expected to finish the project by yourself without using any code from others.