CS 294A/W, Winter 2010
Programming Assignment: Sparse Autoencoder

This is the only homework in this class, which all students taking CS294A/W are required to successfully complete by **5:30pm on Wednesday, January 13**. Submit your solution via email to cs294a-qa@cs.stanford.edu.

Collaboration policy: This assignment should be done individually. It is okay to discuss sparse autoencoders and neural networks (e.g., the material in the lecture notes) with others. But please do not discuss anything specific to this problem set or to your implementation with anyone else. Please also do not look at anyone else's code (including source code found on the internet), or show your code to anyone else. If you have questions about the assignment or would like help, please email us at cs294a-qa@cs.stanford.edu.

The collaboration policy stated above applies only to this programming assignment. Once you've submitted your solution, for the rest of the class you're welcome (and encouraged) to talk to anyone about your work and use any open-source/etc. code you find on the internet (with attribution).

1 Sparse autoencoder implementation

In this problem set, you will implement the sparse autoencoder algorithm, and show how it "invents" edge detection. The sparse autoencoder algorithm is described in the online lecture video, and in the lecture notes found on the course website.

In the file `asgn1.tar.gz`, we have provided some starter code (`main.cpp`). This code reads in 10 images (stored in `olsh.dat`).[1] You should write your code at the place indicated in the file ("`YOUR CODE HERE`").

Specifically, you should implement a sparse autoencoder, trained with 8x8 image patches using stochastic gradient descent.

You should implement your algorithm as follows:

- To get a training example, randomly pick one of the 10 images, then randomly sample an 8x8 image patch from the selected image, and convert the image patch (either in row-major order or column-major order; it doesn't matter) into a 64-dimensional vector to get a training example $x \in \mathbb{R}^{64}$.
- Use an autoencoder with 64 input units, 30 hidden units, and 64 output units.
- Feel free to play with different settings of the parameters. Some values that worked for us are $\beta = 5$, $\rho = -0.996$, $\lambda = 0.002$. But sometimes, little implementational changes can cause the best value of the parameters to change, so don't take any of the values we're suggesting here as cast in stone. Try a range of values of α and find one that works. The value of α that we ended up using was between 0.001 and 1. (You should consider searching for α on an logarithmic scale, and try 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1.)

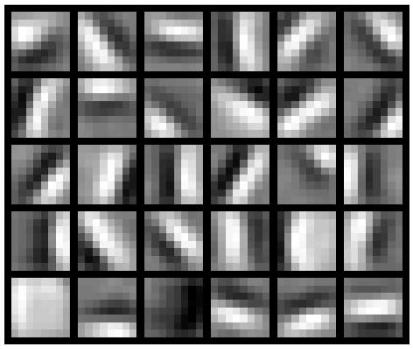
[1] Images courtesy of Bruno Olshausen.

- In our implementation, we initialized the biases $b_i^{(l)}$ to zero, and the weights $W_{ij}^{(l)}$ to random numbers drawn from $[0, \frac{1}{\sqrt{q}}]$, where q is the fan-in (the number of inputs feeding into a node).
- In our implementation, we ran stochastic gradient descent for 4 million iterations.
- Save your weight matrix $W^{(1)}$ (specifying the weights connecting the input units to the hidden layer) to disk and run `view_bases.m` in MATLAB to visualize your learned weights. The `view_bases.m` (see the code for details) assumes that the weights are saved in an ASCII file. Also, in the file, the number of rows is the number of hidden units (so, you should have 30 rows), and number of columns is the dimension of the input (should be 64).

The code uses Eigen as a matrix library. To get started with Eigen, please see the tutorial at: `http://eigen.tuxfamily.org/dox/TutorialCore.html`. Eigen is a very easy-to-use, but powerful, C++ template library that you may find useful for future C++ machine learning projects as well. For this assignment, you'll only need to read page 1 of 3 of the tutorial. Eigen is already in the tar file we provide, in the `eigen/` subdirectory.

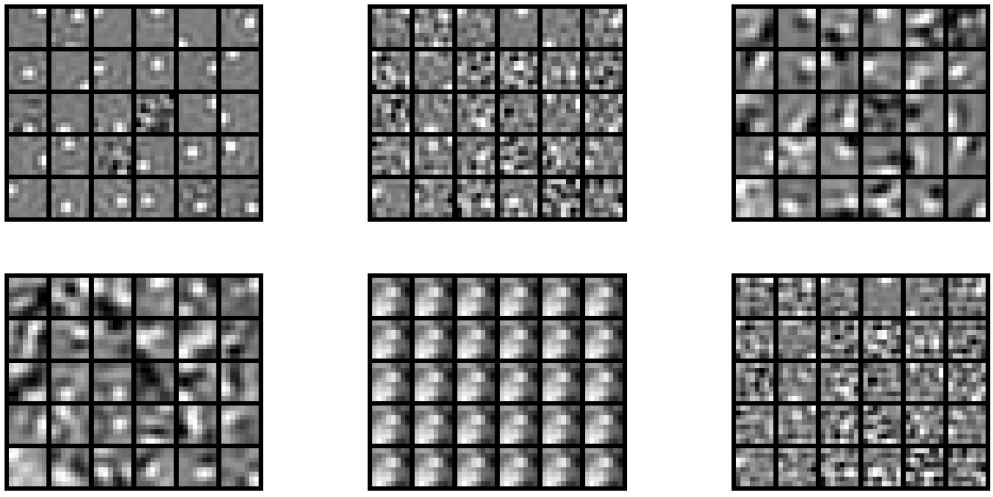
2 Results

To successfully complete this assignment, you should demonstrate your sparse autoencoder algorithm learning a set of edge detectors. For example, this was the visualization we obtained (after 4 million iterations of stochastic gradient descent):



Our implementation took about 100 minutes to run on a fast computer. Given that you may have to try out different parameter values, be sure to budget enough time to run the experiments you'll need.

Also, by way of comparison, here are some visualizations from implementations that we do not consider successful (either a buggy implementation, or where the parameters were poorly tuned):



3 What to submit

Please submit your solution by emailing a tar file to cs294a-qa@cs.stanford.edu.

Your submission should include `main.cpp`, `weights1.dat`, `weights1.jpg` (and, if you wish, other files as well). You can also include a `README` if there're notes that you'd like us to look at. Here, `main.cpp` is your implementation of the sparse autoencoder algorithm, `weights1.dat` is a weight matrix in ASCII format produced by the code and `weights1.jpg` is the visualization result of the weight matrix.

A good solution should have code that is compilable using the provided `Makefile`. Also, the code should run and produce weights that look like "edge detectors" (as in the image on the previous page).

4 Contact

This programming assignment is (by design) more open-ended than most assignments you might have seen in other classes (including CS221 and CS229). If you have questions, don't understand parts of it, find parts of it ambiguous, or need help with C++ or Matlab, don't hesitate to email us at cs294a-qa@cs.stanford.edu to ask for help.