Assignment 2

CS5370: Deep Learning for Vision
IIT-Hyderabad
Jan-Apr 2017

Max Marks: 40
Due: 23rd Mar 2017 11:59 pm

Instructions

- Please use Google Classroom to upload your submission by the deadline mentioned above. Your submission should comprise of a single ZIP file, named `<Your_Roll_No>_Assign2`, with all your solutions, including code.
- For late submissions, 10% is deducted for each day (including weekend) late after an assignment is due. Note that each student begins the course with 6 grace days for late submission of assignments. Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the CS5370 Marks and Grace Days document under the course Google drive.
- Please read the department plagiarism policy. Do not engage in any form of cheating - strict penalties will be imposed for both givers and takers. Please talk to instructor or TA if you have concerns.

1 Programming (40 marks)

The code for the assignment is uploaded alongside this document on the Classroom portal (`assign2code.zip`) [1]. To start with, do the following:

[1] Thanks to Stanford's CS231n course - Fei-Fei Li, Andrej Karpathy and Justin Johnson - for this code and assignment design

- Download the CIFAR-10 dataset. In order to do this, run the following from the `utils` directory:

  ```
  cd utils/datasets
  ./get_datasets.sh
  ```

- Convolutional Neural Networks require a very efficient implementation. We will use the functionality implemented using [Cython](http://cython.org/); you will need to compile the Cython extension before you can run the code. From the `utils` directory, run the following command:

  ```
  python setup.py build_ext --inplace
  ```

Now, work on the following tasks for this assignment (if this is the first time you are using iPython, please go through this brief iPython tutorial: `http://cs231n.github.io/ipython-tutorial/`).

1. **Multi-layer Perceptrons (10 marks):** The IPython notebook `FullyConnectedNets.ipyn` will introduce you to our modular layer design, and then use those layers to implement fully-connected networks of arbitrary depth. To optimize these models you need to implement several popular update rules.

2. **Batch Normalization (10 marks):** In the IPython notebook `BatchNormalization.ipynb`, you will implement batch normalization, and use it to train deep fully-connected networks.

3. **Dropout (5 marks):** The IPython notebook `Dropout.ipynb` will help you implement Dropout and explore its effects on model generalization.

4. **ConvNet on CIFAR-10 (10 marks):** In the IPython Notebook `ConvolutionalNetworks.ipynb`,‘ you will implement several new layers that are commonly used in convolutional networks. You will train a (shallow) convolutional network on CIFAR-10, and it will then be up to you to train the best network that you can.

5. **Do something extra (5 marks):** In the process of training your network, you should feel free to implement anything that you want to get better performance. You can modify the solver, implement additional layers, use different types of regularization, use an ensemble of models, or anything else that comes to mind.

2 Practice Exercises

NO SUBMISSION REQUIRED; PLEASE USE THIS FOR PRACTICE AND LEARNING.

1. **Steerable Filters:** Let $G^0(x, y)$ be some 2-D filter, a function of the Cartesian coordinates x and y. Let $G^\theta(x, y)$ be a rotation of $G^0(x, y)$ by q radians about the origin in the counterclockwise direction, i.e.:

$$G^\theta(x, y) = G^\theta(r\cos\phi, r\sin\phi) = G^0(r\cos\phi - \theta, r\sin\phi - \theta)$$

where $r = \sqrt{(x^2 + y^2)}$ and $\tan\theta = y/x$. In this problem, you may give your answers in Cartesian or polar coordinates, whichever is more convenient.

   (a) Suppose $G^0(x, y) = -2xe^{-(x^2+y^2)}$. Find $G^\theta(x, y)$.

   (b) Show that:

$$G^\theta(x, y) = \cos\theta G^0(x, y) + \sin\theta G^{\pi/2}(x, y)$$

   and that the output image $F(x, y) = I(x, y) * G^\theta(x, y)$ is equal to:

$$\cos\theta\{I(x, y) * G^0(x, y)\} + \sin\theta\{I(x, y) * G^{\pi/2}(x, y)\}$$

   where $*$ denotes convolution.

   (c) Find the direction and magnitude of maximum response at a point (x, y) of the image $I(x, y)$ to the steerable filter $G^\theta(x, y)$. The direction of maximum response is the q, such that $F^\theta(x, y)$ has the biggest magnitude. Give your answer in terms of the image $I(x, y)$ and the responses $F^0(x, y)$, $F^{\pi/2}(x, y)$ to the two steerable basis filters $G^0(x, y)$, $G^{\pi/2}(x, y)$. Note that other steerable filters could require more basis filters than two.

2. **SIFT:**

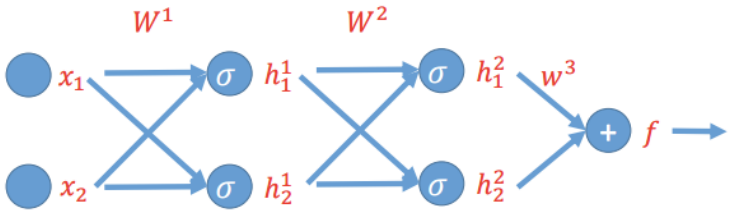
   (a) Given that the dimensions of the SIFT keypoint descriptor is 128, what exactly does the value recorded in a single dimension of a SIFT keypoint descriptor signify? (Dont just describe how it is obtained, think and explain what aspect of the keypoint it is capturing.)

(b) Give any one idea to extend SIFT to find spatio-temporal keypoints in videos (3-dimensions, considering space and time).

3. **Backpropagation:** Consider a 3-layer network:

$$h^1 = \sigma(W^1 x), h^2 = \sigma(W^2 h^1), f(x) =< w^3, h^2 >$$

. Compute $\frac{\partial f}{\partial W^1_{i,j}}$.



4. L_2**-Weight decay:** Show that L_2-weight decay is equivalent to adding Gaussian noise with zero mean and unit variance to the data.

5. **Cross-entropy Loss:** How does the cross-entropy loss function avoid the saturation issues of the sigmoid neuron? Derive your answer mathematically.

6. **The Bias:** We generally initialize the bias to random numbers larger than 0. Why? What happens if we initialize it to a value below zero? Does this affect our ability to train?

7. For a neural network with one hidden layer and `tanh(s)` as the activation functions on all neurons (including the output neuron), prove that for the backprop algorithm (with gradient descent), when all the initial weights w^l_{ij} are set to 1 (for all layers l), then $w^{(1)}_{ij} = w^{(1)}_{i(j+1)}$ for all i and $1 \leq j \leq d^{(1)}$.

8. **Understanding the Hessian:** The points $(0, 0)$, $(\pi/2, \pi/2)$, and $(\pi/2, -\pi/2)$ are critical points of $f(x, y) = \sin x \sin y$.

   (a) Calculate the local quadratic approximation to f at each of the three given critical points.

(b) Graph the surface $z = f(x, y)$ together with the local quadratic approximations at the three given critical points. Feel free to use any tool of your choice for this purpose.

(c) What does the concavity of the local quadratic approximation at a given point have to do with the concavity of the surface at that point?

(d) Calculate the Hessian matrix $H_f(x, y)$.

(e) Fill in the following table. (This will require you to evaluate H_f and find its eigenvalues at each of the three given critical points. You can figure out the concavity—whether up, down, or inconsistent—from the work you did for Part (c).)

Critical Point (x_0,y_0)	$H_f(x_0, y_0)$	Eigenvalues of $H_f(x_0, y_0)$	Concavity at (x_0,y_0)
$(0, 0)$			
$(\pi/2, \pi/2)$			
$(\pi/2, -\pi/2)$			

(f) Examine the table and the graph you made in Part (b). How can the eigenvalues help you classify the concavity of the surface at each critical point?