HOMEWORK #1

ECBM E6040, Professor Aurel A. Lazar

Deadline: Noon 12:00PM, Feb 15, 2016

INSTRUCTIONS: This homework contains two parts - theory (I) and programming (II). Please submit your homework via your bitbucket repository. Your submission should consist of

1. a file called `hw1_writeup.pdf` providing the solution to the theory questions;
2. completed code in `hw1a.py` and `hw1b.py` ;
3. an output folder containing png files created by executing `hw1a.py` and `hw1b.py`.

Part I: Programming

For this homework, you will be using the JAFFE dataset.

Clone the repository created for you via bitbucket. Message the instructors via Piazza if a repository for Homework #1 was not created for you by end of Tuesday, Feb. 2, 2016. The repository contains a script `download.sh` that will download and unzip the dataset.

A public AMI `E6040_hw1_ami` has been created with all necessary packages and patches preinstalled for solving this assignment.

If you would like to use your own Ubuntu machine/AMI, the following instructions might be helpful

- Install libjpeg by executing `sudo apt-get install libjpeg-dev`
- Install libx11 by executing `sudo apt-get install libx11-dev`
- Install PIL in your theano environment by executing `pip install Pillow`
- Apply changes from this PR to your theano installation.

An alternative to the last step above is downgrading numpy by running the following commands in your conda environment

1. `conda uninstall numpy`
2. `conda uninstall scipy`
3. `conda uninstall matplotlib`
4. `conda install numpy==1.9.3`
5. `conda install scipy==0.15.1`
6. `conda install matplotlib==1.4.2`
7. `pip uninstall theano`
8. `pip install theano`

If you will be executing the code on a different OS, please search online for how to perform the above steps for your configuration.

Use single precision for both problems in this assignment.

PROBLEM a: **(35 points)**

In this problem, you will be dividing the images into blocks of sizes $(16, 16), (32, 32), (64, 64)$ and performing principal component analysis in each case. For each case you will visualize reconstructions using different number of principal components and also visualize the top components.

Skeleton code for this problem has been provided in `hw1a.py` in the repository.

Some useful links:-

1. PIL documentatiom
2. PIL convert documentation
3. theano.tensor.nnet.neighbours documentation
4. numpy.linalg.eigh documentation

PROBLEM b: **(35 points)**

In this problem, you will be essentially performing the same tasks as in PROBLEM a, but on the whole image instead of blocks. Since the images are of size 256×256, the matrix $\mathbf{X^T X}$ will be of size $65,536 \times 65,536$. You most likely won't be able to even load this matrix (unless you have enormous amount of RAM available), let alone

perform eigenanalysis on it. Hence, you will be solving this problem by using gradient descent.

Recall from class, that the top principal component can be extracted by solving the following optimization problem

$$\begin{aligned} \underset{\mathbf{d}}{\text{argmin}} \quad & -\mathbf{d}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{d}, \\ \text{subject to} \quad & \mathbf{d}^{\mathbf{T}}\mathbf{d} = 1. \end{aligned}$$

The above can be resolved by using gradient descent with the cost function $f(\mathbf{d}) = -\mathbf{d}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{d}$ while normalizing $\mathbf{d}$ after each update (descent).

Other principal components can be found by "taking out the contribution of the already determined components ". This can be done as in the following pseudocode.

Algorithm 1 Multiple principle components via gradient descent

Input: Data Matrix $\mathbf{X}$, number of components to extract N, learning rate η, Max steps T, Stopping condition

Returns: Principal components $\mathbf{d}_i$ for $i = 0, \cdots, N$

for $i = 0, \cdots, N$ **do**
  $\mathbf{A}_i \leftarrow \mathbf{X}^{\mathbf{T}}\mathbf{X} - \sum_{j=0}^{i-1} \lambda_j \mathbf{d}_j \mathbf{d}_j{}^{\mathbf{T}}$
  Initialize $\mathbf{d}_i$ randomly and let $t = 1$
  while ($t \leq T$ & Stopping condition is not True) **do**
    $\mathbf{y} \leftarrow \mathbf{d}_i - \eta \nabla_{\mathbf{d}_i} \left(-\mathbf{d}_i^{\mathbf{T}}\mathbf{A}_i\mathbf{d}_i\right)$
    $\mathbf{d}_i \leftarrow \dfrac{\mathbf{y}}{\|\mathbf{y}\|}$
    $t \leftarrow t + 1$
  $\lambda_i \leftarrow \mathbf{d}_i{}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{d}_i$

Gradient descent can be performed very easily in theano as it supports symbolic differentiation. Please note that there shouldn't be a need to compute large matrices of order $65,536 \times 65,536$ at any point. You should write your theano expressions and functions such that it avoids computing the large matrices. Choose an appropriate learning rate and an appropriate stopping criteria (for example, when the change in the cost is below some small ϵ or the change in $\|\mathbf{d}_i\|$ is below some small ϵ).

Skeleton code for this part is available in `hw1b.py`.

Some useful links:-

1. Theano Logistic Regression Example

Part II: Theory

PROBLEM c **(15 points)**

(i)

$$p_x(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$y = -\frac{1}{\lambda}\ln(x)$$

Find $p_y(y)$.

(ii)

$$p(\mathrm{x} = x, \mathrm{y} = y) = \begin{cases} 3(xy^2 + yx^2) & \text{for } x, y \in [0,1] \\ 0 & \text{otherwise} \end{cases}$$

Find $p(\mathrm{x} = x)$, $p(\mathrm{y} = y)$, $\mathbb{E}(\mathrm{x})$, $\mathbb{E}(\mathrm{y})$, $\mathbb{E}(\mathrm{xy})$. Are x and y independent?

PROBLEM d **(15 points)**

Let us assume that we model certain data $\mathbb{X} = \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}\}$, $\mathbf{x}^{(i)} \in \mathbb{R}^n$, to have been drawn (independently) from a multivariate gaussian distribution $\mathcal{N}(\mu, \mathbf{\Sigma})$.

(i) Find maximum likelihood estimators for $\mu, \mathbf{\Sigma}$;

(ii) Are the estimators biased or unbiased?

Note:- The multivariate normal distribution is given by

$$\mathcal{N}(\mu, \mathbf{\Sigma}) \sim \frac{1}{(2\pi)^{n/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^{\mathrm{T}} \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu)\right),$$

where μ is the n-dimensional mean vector and $\mathbf{\Sigma}$ is the $n \times n$ covariance matrix.

Some useful links:-

1. Matrix Cookbook

NEED HELP:

If you have any questions you are advised to use Piazza forum which is accessible through courseworks.

GOOD LUCK!