CPSC-340 Machine Learning and Data Mining 2009

Homework # 2

Due Wednesday, Oct 14 1pm.

NAME:_____

Signature:_____

STD. NUM: _____

General guidelines for homeworks:

You are encouraged to discuss the problems with others in the class, but all write-ups are to be done on your own.

Homework grades will be based not only on getting the "correct answer," but also on good writing style and clear presentation of your solution. It is your responsibility to make sure that the graders can easily follow your line of reasoning.

Try every problem. Even if you can't solve the problem, you will receive partial credit for explaining why you got stuck on a promising line of attack. More importantly, you will get valuable feedback that will help you learn the material.

Please acknowledge the people with whom you discussed the problems and what sources you used to help you solve the problem (e.g. books from the library). This won't affect your grade but is important as academic honesty.

When dealing with python exercises, please attach a printout with all your code and show your results clearly.

1. **(Image compression)**

   Convert a photo of yours to .png format. Then load it with python and compute its SVD compression as explained in the lecture notes and in class. Choose a number of eigenvalues (truncation threshold) and EXPLAIN your choice below. Hand in the explanation, the original image and the compressed image.

   > **Python hint:** (From the Matplotlib FAQ.) You can make Matplotlib write a figure directly to disk without a window popup by setting an image backend. This is particularly useful if you're generating figures using a script and you don't want to block the Python mainloop by calling `show()`.
   >
   > ```
   > import matplotlib
   > matplotlib.use('Agg')
   > import matplotlib.pyplot as plt
   > plt.plot([1,2,3])
   > plt.savefig('myfig')
   > ```

   (i) Explanation:

   (ii) What is the compression factor?

   (iii) What is the compression error using the definition of the norm of a matrix introduced in class?

2. **(Latent Semantic Indexing with the SVD)**

In this exercise, you will program a semantic search engine. The corpus of webpages is available at

`http://www.cs.ubc.ca/~nando/340-2009/assignments.php`

You should download the data in pickle format (a useful format for files in python). The corpus consists of 100 webpages (urls). Each webpage has a few tags and their respective frequency of occurrence. The following code snippet loads the data:

```
from pylab import *
from scipy import *
import cPickle as pickle
from math import acos, degrees

def loadData():
    """
    INPUT: loads the file lsiData.pkl
    OUTPUT: - data: dict[URL] -> list of tuples(tag,frequency)
            data.keys()[i] returns the i-th URL.
            data.values()[i] returns the tuples(tag,frequency) for the i-th URL.
    """
    data = pickle.load(open("lsiData.pkl"))
    return data
```

Note that all the data is stored as a dictionary. You should familiarize yourself with these powerful data structures. To get you started, the following function extracts the list of tags and websites from the dictionary:

```
def extractData(data):
    """
    INPUT: data
    OUTPUT: - urls: list of urls
            - tags: alphabetically sorted list of tags
    """
    urls = data.keys()   # List of urls.
    tags = set()         # Initialize set of tags.
    for tagfreq in data.values():
        tags.update(t for t,_ in tagfreq) # add new tags to set of tags.
    tags = sorted(list(tag))      # convert the set of tags to a list of tags,
                                  # sorted in alphabetical order.
    return urls, tags
```

This is one *possible* sketch of the functions involved. You don't have to follow this model, especially if you're familiar with Python's classes.

```
def generateCounts(data, tags, urls):
    """
    INPUT:  - data
            - tags: alphabetically sorted list of tags
    OUTPUT: - A = co-occurrence matrix.
    """
    ???
    return A


def truncatedSVD(A,k):
    """
    INPUT: - A = co-occurrence matrix.
           - k = truncation point. Number of components.
    OUTPUT: truncated SVD U_k, Sigma_k and V_k.
    """
    ???
    return Uk, Sk, Vk


def query(q, Uk, Sk, Vtk, tags, urls):
    """
    e.g. r = query(array(['ai','cs']), Uk, Sk, Vtk, tags, urls)
    """
    ???
    return result


def buildEngine(k):
    """
    e.g. Uk, Sk, Vtk, tag, url = buildEngine(2)
    """
    data = loadData()
    ???
    return Uk, Sk, Vk, tags, urls
```

(i) Generate the term-frequency matrix for this dataset. As done in question 1, plot this matrix as an image. Remember to normalize the co-occurrence matrix.

A term-frequency matrix is a matrix where the columns correspond to documents and the rows correspond to terms. The $(i, j)^{\text{th}}$ entry is the number of occurrences of term j in document i.

(ii) Building the search engine: Compute the truncated SVD of the term-frequency matrix and truncate to an acceptable level. Here, again, you need to choose a reasonable truncation level and explain your choice. Plot the compressed matrix as an image. You must hand in both images. Explanation:

(iii) Querying the search engine: Implement a program that takes as input one or more tags and returns the closest URLs to the query. Do this by using the truncated SVD of the previous part to project the query and compute the angle between the query and all the compressed webpages. Hand in all your code.

> **Python hint:** You might want to look into some or all of the following Python/NumPy/SciPy functions: `enumerate`, `resize`, `reshape`, `degrees`, `norm`, `eye`, `acos`. Just type `help(`*func*`)` for details.

(iv) What are the top 5 webpages retrieved by the query "australia"?

(v) What are the top 5 pages retrieved by the composite query "python animals"?

3. **(Probability revision)**

   (i) Let $P(HIV = 1) = 1/500$ be the probability of contracting HIV by having unprotected sex. If one has unprotected sex twice, what is the probability of contracting HIV? What if one has unprotected sex 500 times?

(ii) Let $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$ be given matrices, and $X \in R^n$ be a random variable with mean $\mathbb{E}(X) = \mu_x \in R^n$ and covariance $cov(X) = \Sigma_X \in \mathbb{R}^{n \times n}$. We define a new random variable

$$Y = AX + b$$

If $X \sim N(\mu_x, \Sigma_x)$, show that $Y \sim N(\mu_y, \Sigma_y)$ by deriving expressions for μ_y and Σ_y in terms of the sufficient statistics (μ_x, Σ_x) of X and the parameters of the linear transformation (A, b).

(iii) Let $x_1 \sim N(\mu_1, \sigma_1^2)$ and $x_2 \sim N(\mu_2, \sigma_2^2)$ be two **independent** random variables, derive an expression for $p(x_1, x_2)$. Explain the zeros in the covariance matrix of the joint distribution of x_1 and x_2.

(iv) A random variable X with a uniform distribution between 0 to 1 is written as $X \sim \mathcal{U}_{[0,1]}(x)$. Draw pictures of the pdf and cdf of this random variable in the one-dimensional case. Draw a picture of the pdf of X when it is two-dimensional.

(v) Let $X \sim N(\mu, \Sigma)$ and let the eigenvalue decomposition of the covariance be $\Sigma = U\Lambda U^T$. Prove that $X \sim \mu + U\Lambda^{1/2}N(0, I)$. Provide a geometric interpretation of the effect of U and Λ in 2D. How does this relate to PCA?

(vi) For the set indicator variable $\mathbb{I}_{\mathbb{A}}(\omega)$, complete the following:

$$\mathbb{E}[\mathbb{I}_{\mathbb{A}}(\omega)] =$$

4. **(PCA)** As explained in class in the context of digit images (the images of 2's), you'll have to implement a PCA projection of, say, 10 images to a single 2D display. Use any images you might like to display on the PCA layout.

Python hint: Make sure you understand how slices and reshaping work on two-dimensional arrays. For example, if the following is unclear, look at it step-by-step until it makes sense (`zeros()` and `arange()` are NumPy functions to create and initialize arrays with predefined values):

```
X = zeros((4,4))
X[1:3,1:3] = (arange(4)*10+5).reshape(2,-1)
```