Exercise 2

AMTH/CPSC 445a/545a - Fall Semester 2016

September 27, 2016

Your solutions must be in a single zip file titled `assignment02.zip`. The zip file should include a single PDF title `assignment02.pdf` and MATLAB or Python scripts as specified. Your homework should be submitted to your Classes*v2 dropbox before Thursday, October 6, 2016 at 1:00 PM.

Problem 1

Sketch a "unit circle" around $(0, 0)$ in each of the following metrics:

1. Euclidean
2. Manhattan
3. Supremum
4. $L^{2/3}$
5. Mahalanobis with covariance $\Sigma = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 1 \end{bmatrix}$

Include the five sketches in `assignment02.pdf`.

Problem 2

Show that the following are distance metrics:

1. $1 - J(x, y)$, where J is the Jaccard coefficient
2. Minimal angle between vectors (i.e., arccos(cosine_similarity(x, y)))
3. Length of shortest path (i.e., Geodesic distance) on a weighted undirected graph $G(V, E, w)$, where V are the set vertices, E is the set of edges and $w : E \times E \to (0, \infty)$ are the weights.
4. Mutual set difference $d(A, B) = |A\backslash B| + |B\backslash A|$, where A and B are arbitrary sets. Show that this distance is equivalent to an L^1 distance.

Include the four proofs in `assignment02.pdf`.

Problem 3

Show that given N data points in $\mathbb{R}^n$ centered around their mean (i.e., the data is a $N \times n$ matrix where the mean of the rows is 0), the k-dimensional coordinates (for $k < n$) given by PCA and by MDS are identical. Include this proof in `assignment02.pdf`.

Problem 4

Using the template `script4.py` or `script4.m` complete the following problem.

- The template will load a list of cities `c` and distance matrix D. In MATLAB notation the distance `D(i,j)` corresponds to the distance between city `c{i}` and city `c{j}` (In Python notation `D[i,j]` is the distance between city `c[i]` and city `c[j]`).
- Using the (classical) Multidimensional Scaling (MDS) algorithm, assign each city coordinates in $\mathbb{R}^2$.
- Plot a point at the MDS coordinates of each city and label the point with the name of the corresponding city

Include the produced plot in `assignment02.pdf`.

Notice

Use the template `script4.m` or `script4.py` depending on if you are using MATLAB or Python.

If you're using MATLAB, complete the exercise only using the core MATLAB language (arithmetic operations, loops, etc.) and the following functions as needed: **`size, eye, ones, svd, diag, figure, plot, text, title, hold on`**

If you're using Python, complete the exercise only using the core Python language and the following imported functions as needed

```
from scipy.io import loadmat as load
from numpy import power as power
from numpy import eye as eye
from numpy import ones as ones
from numpy.linalg import svd as svd
from numpy import sqrt as sqrt
from numpy import diag as diag
from numpy import mat as mat
from matplotlib.pyplot import figure as figure
from matplotlib.pyplot import plot as plot
from matplotlib.pyplot import text as text
from matplotlib.pyplot import title as title
from matplotlib.pyplot import show as show
from matplotlib.pyplot import savefig as savefig
```

Problem 5

Using the template `script5.py` or `script5.m` complete the following problem. Generate a toy example to test Mahalanobis distances & PCA and visualize them using scatter and quiver plots, based on the following steps:

Part A: data generation

- Sample 1000 points uniformly from the interval $[0, 10]$ on the horizontal axis in $\mathbb{R}^2$(i.e., x-coordinate in $[0, 100]$ and y-coordinate being 0);
- Choose an angle θ and rotate your points about the origin using a rotation matrix R_θ formed using this angle;
- Add 2-dimensional Gaussian noise to generate the toy example data.

Part B: Mahalanobis and principal components

- Find mean and covariance of the data;
- Compute the Mahalanobis distance of each data point from the mean of the data;
- Compute the two principal components ϕ_1 & ϕ_2 (i.e., covariance eigenvectors) and associated eigenvalues λ_1 & λ_2;

Part C: Visualization

- Produce a 2D scatter plot of the generated data, where each data point is colored by its Mahalanobis distance from the data mean. For this plot use filled markers (using the flag 'filled') of size of 9 (using the parameter 'S');
- Use the quiver command to add to this plot the two vectors $\sqrt{\lambda_1}\phi_1$ & $\sqrt{\lambda_2}\phi_2$ whose directions is determined by the principal components and their length is determined by the corresponding eigenvalues. For this plot set the color of the vectors to be black (using the flag 'k') and the line width to 4 (using the 'LineWidth' option).
- Save the produced plot as a PNG file with the name `mahal_pca_illustration.png`.

Include the produced plot in `assignment02.pdf`.

Notice: Use the template `script4.m` or `script4.py` depending on if you are using MATLAB or Python.

If you're using MATLAB, complete the exercise only using core MATLAB language (arithmetic operations, loops, arrays, etc.) and the following functions as needed: `rng, cos, sin, rand, randn, mean, ones, repmat, inv, sqrt, svd, figure, scatter, title, colormap, hold on, axis equal, quiver`

If you're using Python, complete the exercise only using the core Python language as the following imported functions included in the template.

```
from numpy.random import seed as seed
from numpy.random import rand as rand
from numpy.random import randn as randn
from numpy import pi as pi
from numpy import cos as cos
from numpy import sin as sin
from numpy import zeros as zeros
from numpy import ones as ones
from numpy import mean as mean
from numpy import mat as mat
from numpy import array as array
from numpy import transpose as transpose
from numpy import sqrt as sqrt
from numpy import diag as diag
from numpy.linalg import inv as inv
from numpy.linalg import svd as svd
from numpy.matlib import repmat as repmat
from matplotlib.pyplot import figure as figure
from matplotlib.pyplot import title as title
from matplotlib.pyplot import plot as plot
from matplotlib.pyplot import scatter as scatter
from matplotlib.pyplot import axis as axis
from matplotlib.pyplot import quiver as quiver
from matplotlib.pyplot import savefig as savefig
from matplotlib.pyplot import show as show
```

Problem 6

Using the template `script6.py` or `script6.m` complete the following. In this problem you will write an implementation of PCA that is more efficient for data with many features. This implementation skips the computation of the covariance matrix and uses the following steps:

1. The template will load a data matrix `X` whose rows correspond to images of a spinning bunny, `theta` which corresponds to the angle of the bunny, and a vector `sz` which describes the dimensions of the bunny images. An example of how to reshape the rows of `X` using `sz` to visulize the bunny images is included in the template.
2. Average over the columns to `X` to compute the mean or average row for `X`. This mean row corresponds to the mean bunny image.
3. Plot an image of the mean bunny.
4. Save the image as `mean_bunny.png`
5. Subtract the mean bunny from each row of `X` to center the data.
6. Compute the eigenvectors and eigenvalues of the covariance matrix (indirectly) using the Singular Value Decomposition (SVD) of `X` [Hint: to avoid memory problems, the SVD needs to be run on `'econ'` mode in MATLAB or with `full_matrices=False` in Python.]
7. Project `X` on to the first three principal components
8. Create a 3-dimensional scatter plot of these coordinates colored by the angle `theta`
9. Add a title and save your PCA scatter plot as `pca_coordinates.png`
10. Using (classical) Multidimensional Scaling (MDS), compute coordinates for the bunny images in $\mathbb{R}^3$.
11. Plot a 3-dimensional scatter plot of the MDS coordinates colored by the angle `theta` [Hint: How should your two scatter plots compare in light of problem 3?]
12. Add a title and save your MDS scatter plot as `mds_coordinates.png`

Include the three produced plot in `assignment02.pdf`.

Notice

Use the template `script6.m` or `script6.py` depending on if you are using MATLAB or Python.

If you're using MATLAB, complete the exercise only using the core MATLAB language (arithmetic operations, loops, etc.) and the following functions: `size, mean, svd, figure, imshow, reshape, title, scatter3, pdist2, eye, ones`.

If you're using Python, complete the exercise only using the core Python language and the following imported functions as needed

```
from scipy.io import loadmat as load
from matplotlib.pyplot import figure as figure
from matplotlib.pyplot import imshow as imshow
from scipy.spatial.distance import pdist as pdist
from scipy.spatial.distance import squareform as squareform
from numpy import reshape as reshape
from numpy import array as array
from numpy import mat as mat
from numpy import diag as diag
from numpy import power as power
from numpy import ones as ones
from numpy import transpose as transpose
from numpy import eye as eye
from matplotlib.pyplot import show as show
from numpy import mean as mean
from numpy.linalg import svd as svd
from matplotlib.pyplot import title as title
from mpl_toolkits.mplot3d import Axes3D as Axes3D
from matplotlib.pyplot import scatter as scatter
from matplotlib.pyplot import savefig as savefig
from numpy import array as array
```