Homework 5

Matrix Diagonalization and PCA

This notebook is arranged in cells. Texts are usually written in the markdown cells, and here you can use html tags (make it bold, italic, colored, etc). You can double click on this cell to see the formatting.

The ellipsis (...) are provided where you are expected to write your solution but feel free to change the template (not over much) in case this style is not to your taste.

Hit "Shift-Enter" on a code cell to evaluate it. Double click a Markdown cell to edit.

Link Okpy

In [ ]:

```
from client.api.notebook import Notebook
ok = Notebook('hw5.ok')
_ = ok.auth(inline = True)
```

Imports

In [ ]:

```
import numpy as np
from scipy.integrate import quad
#For plotting
import matplotlib.pyplot as plt
%matplotlib inline
```

Problem 1 - Asymmetric Quantum Well

Quantum mechanics can be formulated as a matrix problem and solved on a computer using linear algebra methods. Suppose, for example, we have a particle of mass M in a one-dimensional quantum well of width L , but not a square well like the examples you've probably seen before. Suppose instead that the potential $V(x)$ varies somehow inside the well:

We cannot solve such problems analytically in general, but we can solve them on the computer. In a pure state of energy E, the spatial part of the wavefunction obeys the time-independent Schrodinger equation $\hat{H}\psi(x) = E\psi(x)$, where the Hamiltonian operator $\hat{H}$ is given by

$$\hat{H} = -\frac{\hbar^2}{2M}\frac{d^2}{dx^2} + V(x)$$

For simplicity, let's assume that the walls of the well are infinitely high, so that the wavefunction is zero outside the well, which means it must go to zero at $x = 0$ and $x = L$. In that case, the wavefunction can be expressed as a Fourier sine series thus:

$$\psi(x) = \sum_{n=1}^{\infty} \psi_n \sin(\frac{\pi n x}{L})$$

where $\psi_1, \psi_2, \ldots$ are the Fourier coefficients.

Using the orthogonality relationships of the sine functions, we find that $\hat{H}\psi(x) = E\psi(x)$ implies that

$$\sum_{n=1}^{\infty} \int_0^L \sin(\frac{\pi m x}{L})\hat{H}\sin(\frac{\pi n x}{L})dx = \frac{L}{2}E\psi_m.$$

Hence, defining a Hamiltonian matrix $\mathbf{H}$ with elements

$$H_{mn} = \frac{2}{L}\int_0^L \sin(\frac{\pi m x}{L})\hat{H}\sin(\frac{\pi n x}{L})dx.$$

Then, the Schrodinger's equation can be written in matrix form as $\mathbf{H}\boldsymbol{\psi}$ = $E\,\boldsymbol{\psi}$, where $\boldsymbol{\psi}$ is an eigenvector of the Hamiltonian matrix with eigenvalue E. If we can calculate the eigenvalues of this matrix, then we know the allowed energies of the particle in the well.

Let $V(x) = ax/L$, and then we can evaluate the integral in H_{mn} analytically. Here is a general expression for the matrix element H_{mn}:

$$H_{mn} = \left\{ \frac{1}{2M}(\frac{\hbar\pi n}{L})^2 + \frac{a}{2} \quad m = n \quad -\frac{8a}{\pi^2}\frac{mn}{(m^2-n^2)^2} \quad m \neq n \text{ and one is even, one is odd} \right.$$

1. Is the matrix $\mathbf{H}$ *real and symmetric?*

Answer:

...

2. Write a Python program to evaluate your expression for H_{mn} for arbitrary m and n when the particle in the well is an electron, the well has width 5 Angstrom, and a = 10 eV. (The mass and charge of an electron are 9.1094×10^{-31} kg and 1.6022×10^{-19} C respectively.) Evaluate H_{22}, H_{23}, and H_{35}.

In [ ]:

```
L = 5e-10
hbar = 1.0546e-34
M = 9.1094e-31
a = 10*1.6022e-19


def H_element(m,n):

    ...

    return ...
```

In [ ]:

```
print('H22 =', H_element(2,2), ', H23 =', H_element(2,3), ', H35 =', H_element(3
,5))
```

3. The matrix $\mathbf{H}$ is in theory infinitely large, so we cannot calculate all its eigenvalues. But we can get a pretty accurate solution for the first few of them by cutting off the matrix after the first few elements. Use the program you wrote for part 2 to create a 10 × 10 array of the elements of $\mathbf{H}$ up to m, n = 10.

In [ ]:

```
...
H = ...
...
```

In [ ]:

```
# Show the matrix H
print(H)
```

4. Calculate the eigenvalues of this matrix using the appropriate function from numpy.linalg and hence print out, in units of electron volts, the first ten energy levels of the quantum well, within this approximation.

In [ ]:

```
# Suggestion - See https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/n
umpy.linalg.eigh.html
from numpy.linalg import eigh
...
```

5. What is the ground-state energy of the system? (in eV).

In [ ]:

```
...
```

6. Use a 100 × 100 array instead and again calculate the first ten energy eigenvalues.

In [ ]:

```
...
```

7. Comparing with the values you calculated in part 4, what do you conclude about the accuracy of the calculation?

Answer:

...

8. Now modify your program once more to calculate the wavefunction $\psi(x)$ for the ground state and the first two excited states of the well. Use your results to make a graph with three curves showing the probability density $|\psi(x)|^2$ as a function of x in each of these three states.

In [ ]:

```
...
```

9. In the setup, the eigenvector $\boldsymbol{\psi}$ of the Hamiltonian is normalized. Then, is $\int_0^L |\psi(x)|^2 = 1$? (Hint: $\int_0^L \sin(\frac{\pi n x}{L})\sin(\frac{\pi m x}{L}) = \frac{L}{2}$ if $m = n$ and 0 otherwise). Using "quad," integrate the wavefunction for the ground state from $x = 0$ to L.

Answer:

We have $\int_0^L |\psi(x)|^2 = (\sum_n |\psi_n|^2) \int_0^L \sin(\frac{\pi n x}{L})^2 = 1 \cdot \frac{L}{2} \approx 2.5 \times 10^{-10}$

In [ ]:

```
from scipy.integrate import quad
...
```

Problem 2 - Applying the PCA Method on Quasar Spectra

The following analysis is based on https://arxiv.org/pdf/1208.4122.pdf (https://arxiv.org/pdf/1208.4122.pdf).

"Principal Component Analysis (PCA) is a powerful and widely used technique to analyze data by forming a custom set of “principal component” eigenvectors that are optimized to describe the most data variance with the fewest number of components. With the full set of eigenvectors the data may be reproduced exactly, i.e., PCA is a transformation which can lend insight by identifying which variations in a complex dataset are most significant and how they are correlated. Alternately, since the eigenvectors are optimized and sorted by their ability to describe variance in the data, PCA may be used to simplify a complex dataset into a few eigenvectors plus coefficients, under the approximation that higher-order eigenvectors are predominantly describing fine tuned noise or otherwise less important features of the data." (S. Bailey, arxiv: 1208.4122)

In this problem, we take the quasar (QSO) spectra from the Sloan Digital Sky Survey (SDSS) and apply PCA to them. Filtering for high S/N in order to apply the standard PCA, we select 18 high-S/N spectra of QSOs with redshift 2.0 < z < 2.1, trimmed to $1340 < \lambda < 1620\ \mathring{A}$.

In [ ]:

```
# Load data
wavelength = np.loadtxt("HW5_Problem2_wavelength.txt")
flux = np.loadtxt("HW5_Problem2_QSOspectra.txt")
```

In [ ]:

```
# Data dimension
print( np.shape(wavelength) )
print( np.shape(flux) )
```

In the above cell, we load the following data: wavelength in Angstroms ("wavelength") and 2D array of spectra x fluxes ("flux").

We have 824 wavelength bins, so "flux" is 18 $\times$ 824 matrix, each row containing fluxes of different QSO spectra.

1. Plot any three QSO spectra flux as a function of wavelength. (In order to better see the features of QSO spectra, you may plot them with some offsets.)

In [ ]:

```
...
```

"Flux" is the data matrix of order 18 × 824. Call this matrix $\mathbf{X}$.

We can construct the covariance matrix $\mathbf{C}$ *using the mean-centered data matrix. First, calculate the mean of each column and subtracts this from the column. Let* $\mathbf{X_c}$ *denote the mean-centered data matrix.*

$$\mathbf{X_c} = \begin{bmatrix} x_{(1,1)} - \bar{x}_1 & x_{(1,2)} - \bar{x}_2 & \cdots & x_{(1,824)} - \bar{x}_{824} \\ x_{(2,1)} - \bar{x}_1 & x_{(2,2)} - \bar{x}_2 & \cdots & x_{(2,824)} - \bar{x}_{824} \\ \vdots & \vdots & \vdots & \vdots \\ x_{(18,1)} - \bar{x}_1 & x_{(18,2)} - \bar{x}_2 & \cdots & x_{(18,824)} - \bar{x}_{824} \end{bmatrix}$$

where $x_{m,n}$ *denote the flux of* m*th QSO in* n*th wavelength bin, and* $\bar{x}_k$ *is the mean flux in* k*th wavelength bin.*

Then, the covariance matrix is: $\mathbf{C} = \frac{1}{N-1}\ \mathbf{X_c^T X_c}$. *(*$N$ *is the number of QSOs.)*

2. Find the covariance matrix C using the data matrix flux.

In [ ]:

```
C =
...
```

3. Using numpy.linalg, find eigenvalues and eigenvectors of the covariance matrix. Order the eigenvalues from largest to smallest and then plot them as a function of the number of eigenvalues. (Remember that the eigenvector with the highest eigenvalue is the principle component of the data set.) In this case, we find that our covariance matrix is rank-17 matrix, so we only select the first 17 highest eigenvalues and corresponding eigenvectors (other eigenvalues are close to zero).

In [ ]:

```
np.linalg.matrix_rank(C)
```

In [ ]:

```
from numpy.linalg import eig
...
```

In [ ]:

```
# Make plot
...
```

4. Plot the first three eigenvectors. These eigenvectors represent the principal variations of the spectra with respect to that mean spectrum.

In [ ]:

```
...
```

The eigenvectors indicate the direction of the principal components, so we can re-orient the data onto the new zes by multiplying the original mean-centered data by the eigenvectors. We call the re-oriented data "PC scores." (Call the PC score matrix $\mathbf{Z}$) Suppose that we have k eigenvectors. Construct the matrix of eigenvectors $\mathbf{V} = [\mathbf{v_1}\,\mathbf{v_2} \ldots \mathbf{v_k}]$, with $\mathbf{v_i}$ the ith highest eigenvector. Then, we can get 18 × k PC score matrix by multiplying the 18 × 824 data matrix with the 824 × k eigenvector matrix:

$$\mathbf{Z} = \mathbf{X_c V}$$

Then, we can reconstruct the data by mapping it back to 824 dimensions with $\mathbf{V^T}$:

$$\mathbf{\hat{X}} = \boldsymbol{\mu} + \mathbf{ZV^T}$$

where $\boldsymbol{\mu}$ is the vector of mean QSO flux.

Now, comparing the original data with the reconstructed data, we can calculate the residuals. Let $\mathbf{X_{(i)}}, \mathbf{\hat{X}_{(i)}}$ denote the rows of $\mathbf{X}, \mathbf{\hat{X}}$ respectively. Remember that the data matrix has the dimension 18 × 824, so each row $\mathbf{X_{(i)}}$ corresponding the spectra of one particular QSO. (For example, if you wish to see the QSO spectra in row 7, you can plot $\mathbf{X_{(7)}}$ as a function of wavelength.). Then, we can simply calculate the residual as $\frac{1}{N}\sum_{i=1}^{N} |\mathbf{\hat{X}_{(i)}} - \mathbf{X_{(i)}}|^2$ where N is the total number of QSOs (NOTE: $|\mathbf{\hat{X}_{(i)}} - \mathbf{X_{(i)}}|$ is the magnitude of the difference between two vectors $\mathbf{\hat{X}_{(i)}}$ and $\mathbf{X_{(i)}}$.)

5. First, start with only mean flux value $\boldsymbol{\mu}$ (in this case $\mathbf{\hat{X}} = \boldsymbol{\mu}, \mathbf{V} = \mathbf{0}$) and calculate the residual. Then, do the reconstruction using the first two principal eigenvectors $\mathbf{V} = [\mathbf{v_1}\,\mathbf{v_2}]$ and calculate the residual. Finally, let $\mathbf{V} = [\mathbf{v_1}\,\mathbf{v_2} \ldots \mathbf{v_6}]$ (the first six principal eigenvectors) and compute the residual.

In [ ]:

```
...
```

6. For any two QSO spectra, plot the original and reconstructed spectra using the first six principal eigenvectors.

In [ ]:

```
...
```

7. Plot the residual as a function of the number of included eigenvectors.

In [ ]:

```
...
```

In this problem, we only have 18 QSO spectra, so the idea of using PCA may seem silly. We can also use SVD to find eigenvalues and eigenvectors. With SVD, we get $\mathbf{X_c} = \mathbf{USV^T}$. Then, the covariance matrix is $\mathbf{C} = \frac{1}{N-1}\mathbf{X_c^T X_c} = \frac{1}{N-1}\mathbf{VS^2V^T}$. Then, the eigenvalues are the squared singular values scaled by the factor $\frac{1}{N-1}$ and the eigenvectors are the columns of $\mathbf{V}$.

8. Find the eigenvalues applying SVD to the mean-centered data matrix $\mathbf{X_c}$.

In [ ]:

```
from scipy.linalg import svd

...

# Print Eigenvalues
...
```

To Submit

Execute the following cell to submit. If you make changes, execute the cell again to resubmit the final copy of the notebook, they do not get updated automatically.
We recommend that all the above cells should be executed (their output visible) in the notebook at the time of submission.
Only the final submission before the deadline will be graded.

In [ ]:

```
_ = ok.submit()
```