C247 HW1

Zhuohao Li

zhuohaol@ucla.edu

1. (a) $Q^TQ = I$
   i. $Q^T(Q^T)^T = Q^TQ^{-1T} = I$, $Q^{-1}(Q^{-1})^T = Q^{-1}Q = I$, so they're both orthogonal.
   ii. $|\lambda^2|x^Tx = (Qx)^TQx = x^TQ^TQx = x^Tx$, $|\lambda| = 1$
   iii. $QQ^T = I$, $|Q| = |Q^T| = +1, -1$
   iv. assume a vector x, Qx is the transformation of x. $||Qx|| = ||x||$ since $|Q|^2 = 1$

   (b) i. The SVD of $A = U\sum V^T$, eigenvectors of AA^T are the left singular vectors of A (columns of U) and eigenvectors of A^TA are the right singular vectors of A (columns of V).
   ii. Singular values of AA^T and A^TA are the same as the squares of eigenvalues of A.

   (c) i. False
   ii. False
   iii. True
   iv. False
   v. True

2. (a) i. $1 - p_B$
   ii. p_Ap_B
   iii. $(1 - p_A)^{n-1}(1 - p_B)^{n-1}(p_A + p_B - p_Ap_B)$
   iv. $(1 - p_B)^{n-1}p_A$
   v. $(1 - p_A)^{n-1}(1 - p_B)^{n-1}p_Ap_B$

   (b) i. An isolated faculty member is one who is seated between two faculty members from different departments. Since there are 6 faculty members from each department, we can calculate the probability that a given faculty member is isolated and then multiply by the total number of faculty members.
   For a specific faculty member, the probability that the person to their left is from a different department is $\frac{12}{17}$ (since there are 12 faculty members not from their department out of 17 others). Similarly, the probability that the person to their right is also from a different department (and different from the person to the left) is $\frac{11}{16}$.
   So, the probability that a given faculty member is isolated is $\frac{12}{17} \times \frac{11}{16}$. Since there are 18 faculty members in total, the expected number of isolated faculties $E(X) = 8.74$.
   ii. A semi-happy faculty member is one who is seated next to exactly one faculty member from the same department. The calculation is similar to the isolated case. but here we have two scenarios: one faculty member from the same department and

one from a different department, on either side.
The probability for this scenario can be calculated as follows: First, pick one person from the same department ($\frac{5}{17}$) and then one person from a different department ($\frac{12}{16}$). Since there are two ways this can happen (same department member can be either on the left or the right), we multiply this probability by 2. $E(Y) = 7.94$

iii. A joyous faculty member is surrounded by faculty members from the same department. The probability for this is the probability that both adjacent members are from the same department.
This probability is $\frac{5}{17} \times \frac{4}{16}$ for a given faculty member. $E(Z) = 1.32$

(c) Bayes' Theorem: $P(A|B) = \dfrac{(B|A) \times P(A)}{P(B)}$
Let A be 'the man has a dangerous type of lung cancer.'
Let B be 'the man has a positive test.'
$P(A) = 0.0005, P(B|A) = 0.9, P(B|\hat{A}) = 0.01$
$P(B) = P(B|A) \times P(A) + P(B|\hat{A}) \times P(\hat{A})$

$P(A|B) = 4.31\%$

ii. $P(A|\hat{B}) = 0.0051\%$

(d) $E(Ax + b) = AE(x) = b$

(e) $cov(Ax + b) = Acov(x)A'$

3. (a) $\nabla_x x^T Ay = Ay$

(b) $\nabla_y x^T Ay = (x^T A)^T$

(c) $\nabla_A x^T Ay = xy^T$

(d) $x^T(A + A^T) + b$

(e) B^T

(f) $B^T + B + (AB + BA)^T$

(g) $2(A + \lambda B)$

4. Assume $Y \in R^{n\times n}, X \in R^{n\times n}, M \in R^{n\times n}$

$argmin_W f \iff argmin_W \frac{1}{2}\sum_{i=1}^{n} ||y^{(i)} - Wx^{(i)}||^2 \iff argmin_W \frac{1}{2} tr((Y - WX)^T(Y - WX)) \iff argmin_W \frac{1}{2} tr(Y^T - Y^T WX - X^T W^T Y + X^T W^T WX)$

$\nabla_W = \frac{1}{2}(-2X^T Y + 2WX^T X) = 0$

$W = (X^T X)^{-1} X^T Y$

5. Assume $Y \in R^N, X \in R^{N\times M}, \theta \in R^M$, where each is $y^{(i)}, \hat{x}^{(i)}$ arrange by columns.

$argmin_\theta f \iff argmin_\theta \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \theta^T \hat{x}^{(i)})^2 + \frac{\lambda}{2}||\theta||_2^2 \iff argmin_\theta \frac{1}{2}(Y - X\theta)^T(Y - X\theta) + \frac{\lambda}{2}\theta^T\theta$

Get derivative: $\nabla_\theta f = (X^T X + \lambda I)\theta - X^T Y$

If $(X^T X + \lambda I)$ is invertible, $\theta* = (X^T X + \lambda I)^{-1} X^T Y$

Linear regression workbook

This workbook will walk you through a linear regression example. It will provide familiarity with Jupyter Notebook and Python. Please print (to pdf) a completed version of this workbook for submission with HW #1.

ECE C147/C247, Winter Quarter 2024, Prof. J.C. Kao, TAs: T.Monsoor, Y. Liu, S. Rajesh, L. Julakanti, K. Pang

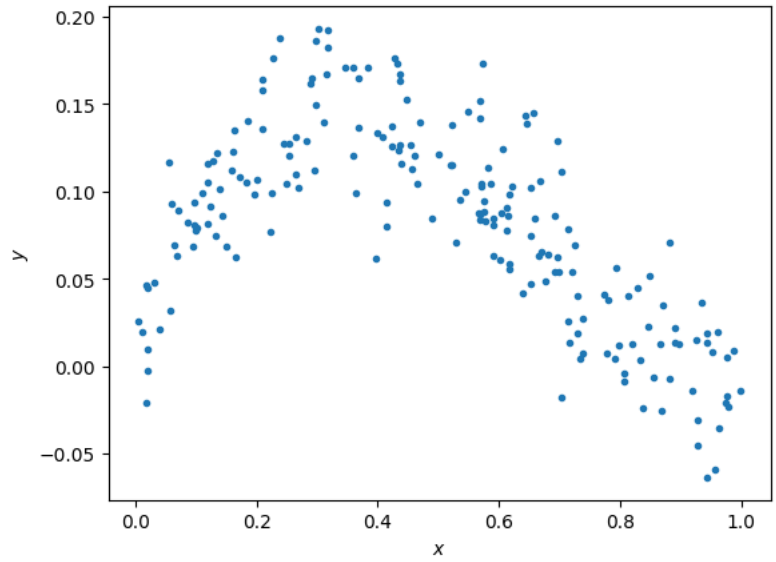
```
import numpy as np
import matplotlib.pyplot as plt

#allows matlab plots to be generated in line
%matplotlib inline
```

Data generation

For any example, we first have to generate some appropriate data to use. The following cell generates data according to the model:
$y = x - 2x^2 + x^3 + \epsilon$

```
np.random.seed(0)  # Sets the random seed.
num_train = 200     # Number of training data points

# Generate the training data
x = np.random.uniform(low=0, high=1, size=(num_train,))
y = x - 2*x**2 + x**3 + np.random.normal(loc=0, scale=0.03, size=(num_train,))
f = plt.figure()
ax = f.gca()
ax.plot(x, y, '.')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
```

```
Text(0, 0.5, '$y$')
```



QUESTIONS:

Write your answers in the markdown cell below this one:

(1) What is the generating distribution of x?

(2) What is the distribution of the additive noise ϵ?

ANSWERS:

(1) It is the uniform distribution.

(2) It is the normal distribution.

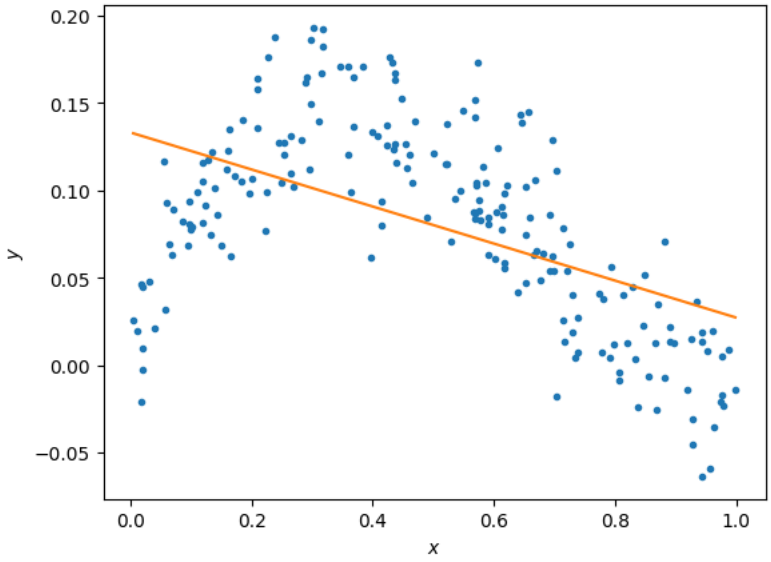
Fitting data to the model (5 points)

Here, we'll do linear regression to fit the parameters of a model $y = ax + b$.

```
# xhat = (x, 1)
xhat = np.vstack((x, np.ones_like(x)))

# ==================== #
# START YOUR CODE HERE #
# ==================== #
# GOAL: create a variable theta; theta is a numpy array whose elements are [a, b]

theta = np.linalg.inv(xhat.dot(xhat.T)).dot(xhat.dot(y))

# ================== #
# END YOUR CODE HERE #
# ================== #
```

```
# Plot the data and your model fit.
f = plt.figure()
ax = f.gca()
ax.plot(x, y, '.')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')

# Plot the regression line
xs = np.linspace(min(x), max(x),50)
xs = np.vstack((xs, np.ones_like(xs)))
plt.plot(xs[0,:], theta.dot(xs))
```

```
[<matplotlib.lines.Line2D at 0x7ce7553232e0>]
```



QUESTIONS

(1) Does the linear model under- or overfit the data?

(2) How to change the model to improve the fitting?

ANSWERS

(1) Under-fit

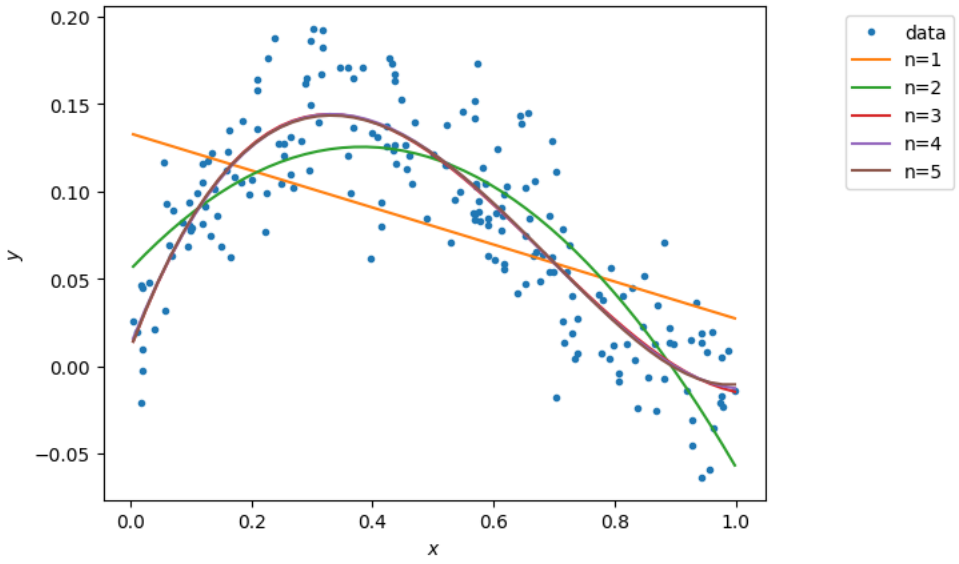
(2) The model needs to be more complex (increase the order of altitude)

Fitting data to the model (5 points)

Here, we'll now do regression to polynomial models of orders 1 to 5. Note, the order 1 model is the linear model you prior fit.

```
N = 5
xhats = []
thetas = []

# ==================== #
# START YOUR CODE HERE #
# ==================== #

# GOAL: create a variable thetas.
# thetas is a list, where theta[i] are the model parameters for the polynomial fit of order i+1.
#   i.e., thetas[0] is equivalent to theta above.
#   i.e., thetas[1] should be a length 3 np.array with the coefficients of the x^2, x, and 1 respectively.
#   ... etc.

# for i in range(1, N + 1):
#     theta = np.polyfit(x, y, i)
#     thetas.append(theta)

for i in np.arange(N):
    xhat = np.vstack((x, np.ones_like(x))) if i == 0 else np.vstack((x ** (i + 1), xhat))
    thetas.append(np.linalg.inv(xhat.dot(xhat.T)).dot(xhat.dot(y)))

# ================== #
# END YOUR CODE HERE #
# ================== #
```

```
# Plot the data
f = plt.figure()
ax = f.gca()
ax.plot(x, y, '.')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')

# Plot the regression lines
plot_xs = []
for i in np.arange(N):
    if i == 0:
        plot_x = np.vstack((np.linspace(min(x), max(x),50), np.ones(50)))
    else:
        plot_x = np.vstack((plot_x[-2]**(i+1), plot_x))
    plot_xs.append(plot_x)

for i in np.arange(N):
    ax.plot(plot_xs[i][-2,:], thetas[i].dot(plot_xs[i]))

labels = ['data']
[labels.append('n={}'.format(i+1)) for i in np.arange(N)]
bbox_to_anchor=(1.3, 1)
lgd = ax.legend(labels, bbox_to_anchor=bbox_to_anchor)
```



Calculating the training error (5 points)

Here, we'll now calculate the training error of polynomial models of orders 1 to 5.

```
training_errors = []

# ==================== #
# START YOUR CODE HERE #
# ==================== #

# GOAL: create a variable training_errors, a list of 5 elements,
# where training_errors[i] are the training loss for the polynomial fit of order i+1.
for theta in thetas:
    y_pred = np.polyval(theta, x)  # Calculate predicted y values
    mse = np.mean((y - y_pred) ** 2)  # Calculate mean squared error
    training_errors.append(mse)

# ================== #
# END YOUR CODE HERE #
# ================== #

print ('Training errors are: \n', training_errors)
```

```
Training errors are:
 [0.0023799610883627007, 0.001092492220926853, 0.0008169603801105368, 0.0008165353735296978, 0.0008161479195525294]
```

QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

ANSWERS

(1) 5

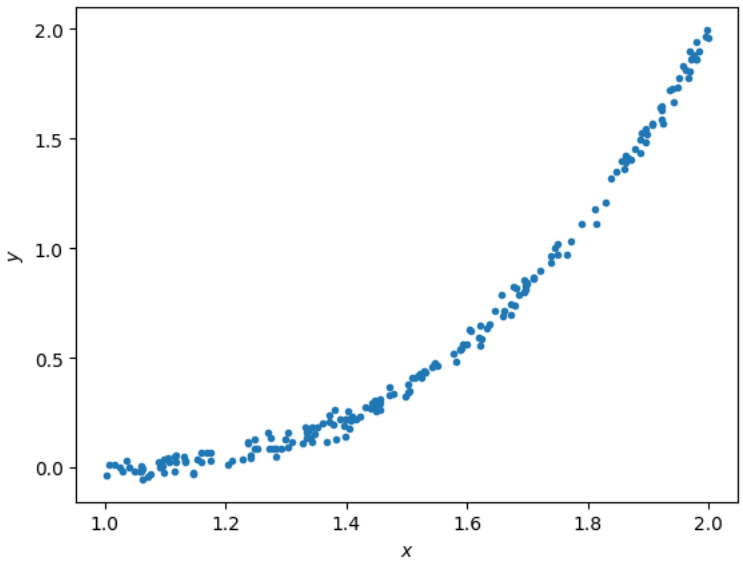
(2) The model is the most complex, it can derive more high-dimensional information

Generating new samples and testing error (5 points)

Here, we'll now generate new samples and calculate testing error of polynomial models of orders 1 to 5.

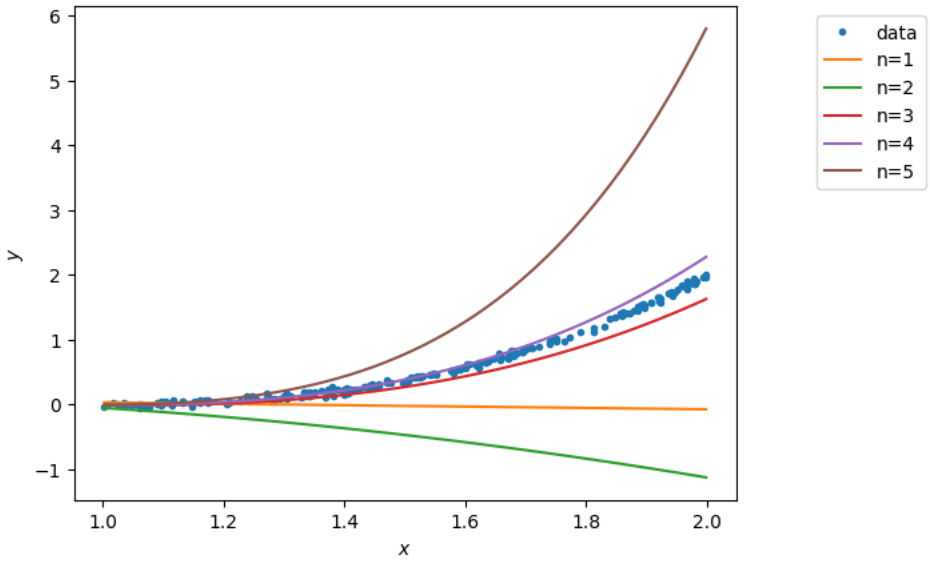
```
x = np.random.uniform(low=1, high=2, size=(num_train,))
y = x - 2*x**2 + x**3 + np.random.normal(loc=0, scale=0.03, size=(num_train,))
f = plt.figure()
ax = f.gca()
ax.plot(x, y, '.')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
```

```
Text(0, 0.5, '$y$')
```



```
xhats = []
for i in np.arange(N):
    if i == 0:
        xhat = np.vstack((x, np.ones_like(x)))
        plot_x = np.vstack((np.linspace(min(x), max(x),50), np.ones(50)))
    else:
        xhat = np.vstack((x**(i+1), xhat))
        plot_x = np.vstack((plot_x[-2]**(i+1), plot_x))

    xhats.append(xhat)
```

```
# Plot the data
f = plt.figure()
ax = f.gca()
ax.plot(x, y, '.')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')

# Plot the regression lines
plot_xs = []
for i in np.arange(N):
    if i == 0:
        plot_x = np.vstack((np.linspace(min(x), max(x),50), np.ones(50)))
    else:
        plot_x = np.vstack((plot_x[-2]**(i+1), plot_x))
    plot_xs.append(plot_x)

for i in np.arange(N):
    ax.plot(plot_xs[i][-2,:], thetas[i].dot(plot_xs[i]))

labels = ['data']
[labels.append('n={}'.format(i+1)) for i in np.arange(N)]
bbox_to_anchor=(1.3, 1)
lgd = ax.legend(labels, bbox_to_anchor=bbox_to_anchor)
```




```
testing_errors = []

# ==================== #
# START YOUR CODE HERE #
# ==================== #

# GOAL: create a variable testing_errors, a list of 5 elements,
# where testing_errors[i] are the testing loss for the polynomial fit of order i+1.
for theta in thetas:
    y_pred = np.polyval(theta, x)  # Calculate predicted y values
    mse = np.mean((y - y_pred) ** 2)  # Calculate mean squared error
    testing_errors.append(mse)


# for i in np.arange(N):
#     xhat = np.vstack((x, np.ones_like(x))) if i == 0 else np.vstack((x ** (i + 1), xhat))
#     testing_errors.append(np.sqrt(np.sum((y - thetas[i].dot(xhat)) ** 2) / len(y)))

# ================== #
# END YOUR CODE HERE #
# ================== #

print ('Testing errors are: \n', testing_errors)
```

```
Testing errors are:
 [0.8086165184550587, 2.1319192445057915, 0.031256971084083734, 0.011870765211495651, 2.1491021747199084]
```

QUESTIONS

(1) What polynomial has the best testing error?

(2) Why polynomial models of orders 5 does not generalize well?

ANSWERS

(1) 4

(2) Over-fitting the training data