ECE 147 HW 1 Claire Li

1. Q is a real orthoganol matrix → $QQ^T = Q^TQ = I$

a i. Q^T is orthoganol

$Q^T(Q^T)^T = Q^TQ = I$, so $\boxed{Q^T \text{ is orthoganol}}$

Q^{-1} is orthoganol

Since Q is an orthoganol matrix, $Q^{-1} = Q^T$.

Q^T is orthoganol (shown above) so $Q^{-1} = \boxed{Q^T \text{ is orthoganol.}}$

ii. Q has eigenvalues w/ norm (magnitude) 1

eigenvalues + vectors: $Ax - \lambda x = 0$, let λ = eigen values + x = eigen vector

$Qx - \lambda x = 0$

$Qx = \lambda x$

$x^T \overset{I}{\cancel{Q^TQ}} x = x^T \lambda^T \lambda x$

$x^T x = \|\lambda\|^2 x^T x$

$\|\lambda^2\| = 1$

$\boxed{\lambda = 1}$

iii. determinant of Q is 1 or −1

$\det(Q^TQ) = \det(I) = 1$

$\det(Q^T) \cdot \det(Q) = 1$, $\det(Q^T) = \det(Q)$

$(\det(Q))^2 = 1$ → $\boxed{\det(Q) = \pm 1}$

iv. Q defines a length preserving transformation

↳ dist b/w points is same after transformation

show $\|Qv\| = \|v\|$

$\|Qv\|^2 = (Qv)^T(Qv) = v^T(Q^TQ)v = v^Tv = \|v\|^2$

so $\|Qv\| = \|v\|$

∴ Q defines a length preserving transformation

b i. A + AA^T, A + A^TA

singular value decomposition (SVD) of A: $A = U\Sigma V^T$

$AA^T = U\Sigma^T V^T V \Sigma U^T$

$= U\Sigma\Sigma^T U^T$

$A^TA = V\Sigma^T U^T U \Sigma V^T$

$= V\Sigma^T\Sigma V^T$

→ the left singular values of A are the eigenvectors of AA^T (columns of U)

→ the right singular values of A are the eigenvectors of A^TA (columns of V)

b ii. diagonal entries Σ^2 are the eigenvalues of A^TA → AA^T

→ the singular values of A are the square roots of the eigenvalues of AA^T + A^TA

c i. false — a linear operator (matrix) can have less than n distinct eigenvalues

e.g. $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ → only 1 distinct eigenvalue, not 2

ii. false — let $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$, then $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is an eigenvector w/ eigenval 1 + $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ is an eigenvector w/ eigen val 2, but their sum is not an eigenvector of A

iii. true if $x^T A x \geq 0$ + let v be eigen vector, λ be an eigen value

$v^T \lambda v \geq 0$ → $v > 0$, $\lambda \geq 0$

iv. true let $A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ rank $= 2 >$ # of eigenvalues $= 1$

v. true let v_1, v_2 corr. to the same eigenvalue λ

$A(v_1 + v_2) = Av_1 + Av_2 = \lambda v_1 + \lambda v_2 = \lambda(v_1 + v_2)$

2. Probability Refresher

a i. P(A is not hit) $p_A(1-p_B)[(1-p_B)(1-p_A)]^{n-1} = p_A(1-p_B)\sum_{n=1}^{\infty}(1-p_B)^{n-1}(1-p_A)^{n-1} = \boxed{\frac{(1-p_B)p_A}{1-[(1-p_A)(1-p_B)]}}$

ii. P(both duelists are hit) $p_A \cdot p_B[(1-p_B)(1-p_A)]^{n-1} = p_A p_B \sum_{n=1}^{\infty}[(1-p_B)(1-p_A)]^{n-1} = \boxed{\frac{p_A p_B}{1-[(1-p_A)(1-p_B)]}}$

iii. P(duel ends after n^{th} round)

$= P(n, A\text{ hit}) + P(n, B\text{ hit}) + P(n, \text{both hit})$

$= \boxed{\frac{(1-p_A)p_B}{(1-p_A)^{n-1}(1-p_B)^{n-1}} + \frac{(1-p_B)p_A}{(1-p_A)^{n-1}(1-p_B)^{n-1}} + \frac{p_A p_B}{(1-p_A)^{n-1}(1-p_B)^{n-1}}}$

iv. P(ends after n^{th} round given A is not hit)

E = event duel ends after n^{th} round

A = event A not hit

B = event both duelists hit

$P(E|A) = \frac{P(A \cup E)}{P(A)} = \boxed{\frac{((1-p_A)(1-p_B))^{n-1}(1-p_B)p_A}{p_A \cdot \frac{1-p_B}{1-(1-p_A)(1-p_B)}}}$

v. P(ends after n^{th} round given both duelists hit)

$P(E|B)$

$= \frac{P(B \cap E)}{P(B)} = \boxed{\frac{((1-p_A)(1-p_B))^{n-1}p_A p_B}{p_A p_B \cdot \frac{1}{1-(1-p_A)(1-p_B)}}}$

b i. conditional prob. density functions of y

if $X=1$ → $y = 1 + N$, so $y \sim (1, \sigma^2)$

$\boxed{f_{y|x=1}(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-1)^2}{2\sigma^2}}}$

if $X=1$ → $y = -1 + N$, so $y \sim (-1, \sigma^2)$

$\boxed{f_{y|x=-1}(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y+1)^2}{2\sigma^2}}}$

ii. case 1: $X=1$

$P(\text{error} \mid X=1) = P(y < \varphi \mid X=1)$

$y = 1 \sim N(1, \sigma^2)$, so to standardize y:

$Z = \frac{y-1}{\sigma}$, $Z \sim N(0,1)$

so, $P(\text{error} \mid X=1) = P(Z < \frac{\varphi-1}{\sigma}) = \Phi(\frac{\varphi-1}{\sigma})$

case 2: $X=-1$

$P(\text{error} \mid X=-1) = P(y < \varphi \mid X=-1)$

$y = 1 \sim N(1, \sigma^2)$, so to standardize y:

$Z = \frac{y+1}{\sigma}$, $Z \sim N(0,1)$

so, $P(\text{error} \mid X=-1) = P(Z < \frac{\varphi+1}{\sigma}) = 1 - \Phi(\frac{\varphi+1}{\sigma})$

$= \boxed{0.5(\Phi(\frac{\varphi-1}{\sigma})) + 0.5(1-\Phi(\frac{\varphi+1}{\sigma}))}$

iii. optimal φ

to minimize error for $X=1$ or $X=-1$: $\Phi(\frac{\varphi-1}{\sigma}) = 1 - \Phi(\frac{\varphi+1}{\sigma})$

$\frac{d}{d\varphi} P(\text{error}) = 0.5\frac{1}{\sigma}\Phi(\frac{\varphi-1}{\sigma}) - 0.5\frac{1}{\sigma}\Phi(\frac{\varphi+1}{\sigma}) = 0$

symmetry of $\Phi(z)$ $\quad \frac{\varphi-1}{\sigma} = -(\frac{\varphi+1}{\sigma})$

$2\varphi = 0$

$\underline{\varphi = 0}$

$P(\text{error}) = 0.5\,\Phi(\frac{\varphi-1}{\sigma}) + 0.5(1-\Phi(\frac{\varphi+1}{\sigma}))$

$= \boxed{0.5\,\Phi(\frac{-1}{\sigma}) + 0.5(1-\Phi(\frac{1}{\sigma}))} = 0.5[\Phi(\frac{-1}{\sigma}) + 1 - \Phi(\frac{1}{\sigma})] = 0.5[1-\Phi(\frac{1}{\sigma}) + 1 - \Phi(\frac{1}{\sigma})] = 1 - \Phi(\frac{1}{\sigma})$

ci. D: event that man has a dangerous disease $= 0.0005$

T: man has a positive test

$P(D) = 0.0005$, $P(T \mid D) = 0.9$, $P(T \mid D^c) = 0.01$

$P(D \mid T) = \frac{P(T \mid D)}{P(T)} = \frac{P(T \mid D)(P(D))}{P(T \mid D)P(D) + P(T \mid D^c)P(D^c)} = \frac{0.9(0.0005)}{0.9(0.0005) + 0.01(0.9995)} = \boxed{0.043}$

ii. $P(D \mid T^c) = \frac{P(T^c \mid D)}{P(T^c)} = \frac{P(T^c \mid D)(P(D))}{P(T^c \mid D)P(D) + P(T^c \mid D^c)P(D^c)} = \frac{0.1(0.0005)}{0.1(0.0005) + (0.99)(0.9995)} = \boxed{0.000051}$

di. 3 daughters, 3 sons

X = # of daughters accompanying dad $\quad 1 \le i \le 3$

$X_i = \begin{cases} 1, & i^{th} \text{ child is a girl} \\ 0, & i^{th} \text{ child is a boy} \end{cases}$

$E[X_i] = P(X_i = 1) + 0 \cdot P(X_i = 0)$

$= \frac{3}{6} = \frac{1}{2}$

$X_i^2 = X_i \; (1^2 = 1, 0^2 = 0)$

$Cov(X_i, X_i) = E(X_i^2) - E(X_i)E(X_i)$

$= \frac{1}{2} - \frac{1}{4} = \boxed{\frac{1}{4}}$

ii. $Cov(X_i, X_j) = E(X_i X_j) - E(X_i)E(X_j)$

$E(X_i) = E(X_j) = \frac{1}{2}$

$E(X_i X_j) = P(X_i = 1 + X_j = 1) = \frac{3}{6} \cdot \frac{2}{5} = \frac{1}{5}$

$Cov(X_i, X_j) = \frac{1}{5} - \frac{1}{4}$

$= \boxed{-\frac{1}{20}}$

iii. $\mathrm{Var}(X) = \mathrm{Cov}(X,X)$

$$= \sum_{i=1}^{3} \mathrm{var}(X_i) + \sum_{i=1}^{n} \sum_{i \neq j} \mathrm{cov}(X_i, X_j)$$

$$= 3\left(\frac{1}{4}\right) + 3(2)\left(-\frac{1}{20}\right)$$

$$= \frac{3}{4} - \frac{3}{10}$$

$$= \boxed{\frac{9}{20}}$$

3a. $\nabla_x x^T A y \qquad x \in \mathbb{R}^n,\ y \in \mathbb{R}^m,\ A \in \mathbb{R}^{n \times m}$

$$\underset{1\times n}{[\ \ ]}\ \underset{n\times m}{[\ \ ]}\ \underset{m\times 1}{[\ ]} = 1\times 1 \quad \text{scalar}$$

$$\nabla_x x^T A y = \boxed{Ay} \qquad \text{from matrix cookbook}$$

b. $\nabla_y x^T A y$

$$= \nabla_y y^T A^T x = \boxed{A^T x}$$

c. $\nabla_A x^T A y = \boxed{xy^T}$ from matrix cookbook

d. $f = x^T A x + b^T x$, what is $\nabla_x f$?

$$\nabla_x x^T A x + b^T x = \underbrace{2Ax}_{\substack{(A+A^T)x \\ = 2Ax \text{ when } A \text{ is } n\times n}} + \nabla_x x^T b = \boxed{2Ax + b}$$

e. $f = \mathrm{tr}(AB)$, $\nabla_A f$?

$$\nabla_A \mathrm{tr}(AB) = \boxed{B^T}$$

f. $f = \mathrm{tr}(BA + A^T B + A^2 B)$, $\nabla_A f$?

$$= \mathrm{tr}(BA) + \mathrm{tr}(A^T B) + \mathrm{tr}(A^2 B)$$

$$\nabla_A f = \boxed{B^T + B + (AB + BA)^T}$$

g. $f = \| A + \lambda B \|^2_F$, $\nabla_A f$?

Frobenius Norm: $\|M\|_F^2 = \mathrm{tr}(MM^T)$

$$f = \mathrm{tr}\left((A + \lambda B)(A + \lambda B)^T \right)$$

$$= \mathrm{tr}(A^T A + 2A^T \lambda B + \lambda^2 B^T B)$$

$$\nabla_A f = \boxed{2A + 2\lambda B}$$

4. $\min \frac{1}{2} \sum_{i=1}^{n} \| y^{(i)} - W x^{(i)} \|^2$

$$\mathcal{L} = \min \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - W x^{(i)})^T (y^{(i)} - W x^{(i)})$$

$$= \min \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)T} y^{(i)} - 2 y^{(i)T} W x^{(i)} + x^{(i)T} W W^T x^{(i)} \right) \qquad \mathrm{tr}(WX) = \mathrm{tr}(X^T W^T), \quad \mathrm{tr}(Y^T) = \mathrm{tr}(Y)$$

$$= \mathrm{tr}\left(Y^T Y - W X Y^T + \frac{1}{2} W X X^T W \right)$$

$$\nabla_W \mathcal{L} = (XY^T)^T + \frac{1}{2} W (X X^T + X^T X)$$

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

5. $\arg_\theta \min \frac{1}{2} \sum_{i=1}^{\infty} (y^{(i)} - \theta^T x^{(i)})^2 + \frac{\lambda}{2} \|\theta\|_2^2$

$$L = \frac{1}{2} \sum_{i=1}^{\infty} (y^{(i)} - \theta^T x^{(i)})^T (y^{(i)} - \theta^T x^{(i)}) + \frac{\lambda}{2} \theta^T \theta \qquad \hat{x}^{(i)T} \theta \text{ is scalar}$$

$$= \frac{1}{2} (Y - X\theta)^T (Y - X\theta) + \frac{\lambda}{2} \theta^T \theta$$

$$= \frac{1}{2} \left(Y^T Y - \underbrace{Y^T X\theta - \theta^T X^T Y}_{-2Y^T X\theta} + \theta^T X^T X \theta \right) + \frac{\lambda}{2} \theta^T \theta \Rightarrow -Y^T X \theta + \frac{1}{2} \theta^T (X^T X + \lambda I) \theta + \frac{1}{2} Y^T Y$$

$$\nabla_\theta \mathcal{L} = -X^T Y + \frac{1}{2} (X^T X + \lambda I) \theta = 0$$

$$\boxed{\theta = (X^T X + \lambda I)^{-1} X^T Y}$$

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 2025, Prof. J.C. Kao, TAs: B. Qu, K. Pang, S. Dong, S. Rajesh, T. Monsoor, X. Yan

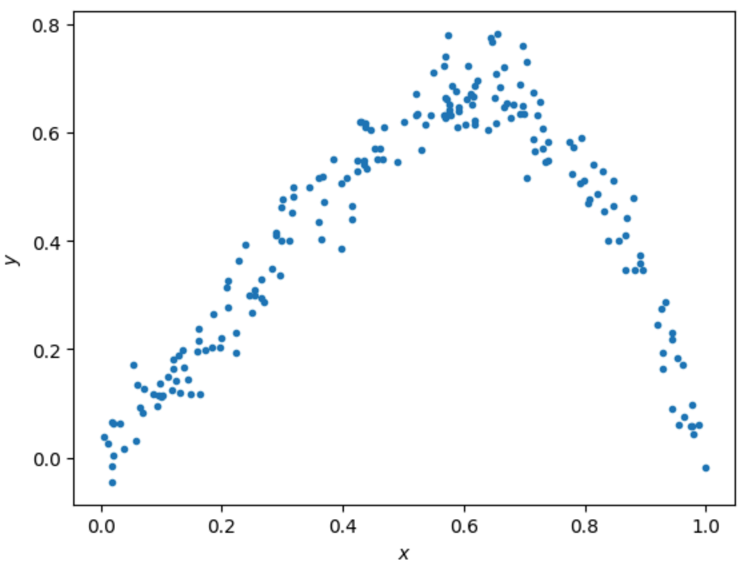
```
import numpy as np
import matplotlib.pyplot as plt

#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: $y = x + 2x^2 - 3x^3 + \epsilon$

```
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 - 3*x**3 + np.random.normal(loc=0, scale=0.05, 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) The distribution of x is a uniform distribution with paramers 0 and 1 so the values of the distribution are between 0 and 1.

(2) The distribution of the additive noise is a normal distribution with a mean 0 and standard deviation 0.05.

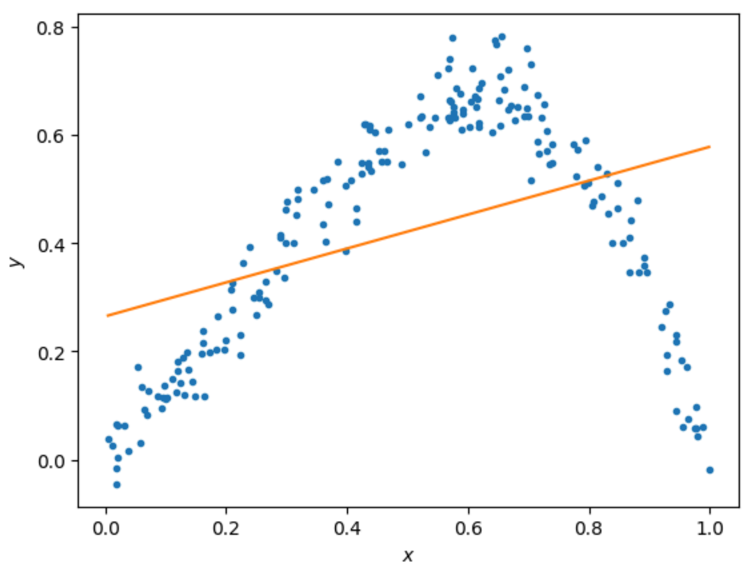
Fitting data to the model (5 points)

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)))
# adds column of ones to original x data

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

x_trans = xhat.T
theta = np.linalg.inv(x_trans.T.dot(x_trans)).dot(x_trans.T.dot(y))
# θ=argmin||y-Xθ||^2
# θ = (X^T.X)^(-1)X&^TY

# ================== #
# 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 0x7bbc205f4090>]
```



QUESTIONS

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

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

ANSWERS

(1) The linear model underfits the data, as the data has the appearance of a polynomial function but the linear model takes on the shape of a line.

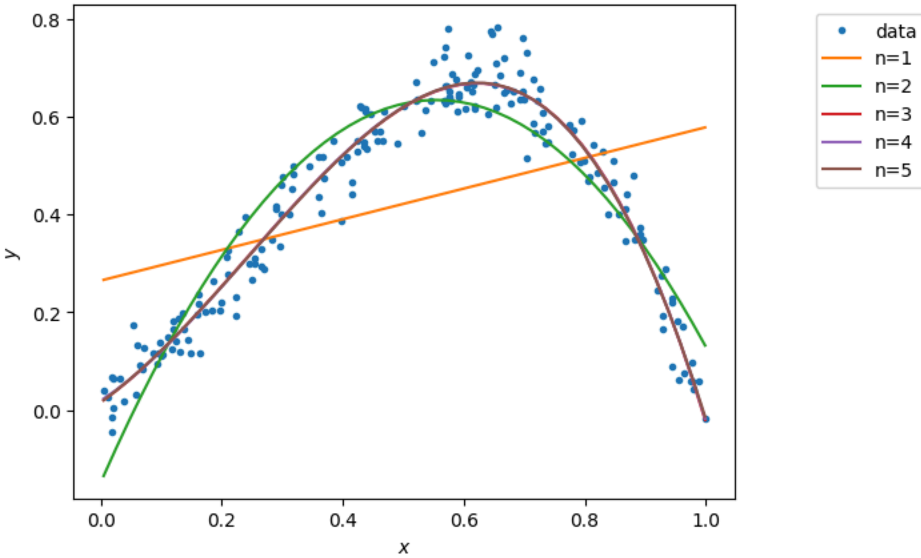
(2) To improve fitting, we can increase the complexity of the model, using a function with more polynomial terms such as a quadratic model.

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.

# y = θ0 + θ1x + θ2x^2 + ... + θix^i

for i in range(1, N+1):
  xhat = np.ones_like(x)
  for j in range(1, i+1):
    # X = [1 x x^2 ... x^(i+1)]
    xhat = np.vstack((x**j, xhat))

  # x_trans = X^T
  x_trans = xhat.T
  # θ = (X^T.X)^(-1)X&^TY
  theta = np.linalg.inv(x_trans.T.dot(x_trans)).dot(x_trans.T.dot(y))

  xhats.append(x_trans)
  thetas.append(theta)

# ================== #
# 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 i in range(N):
  diff = y - xhats[i].T.dot(thetas[i])
  training_errors.append(diff.dot(diff)/num_train)

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

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

```
Training errors are:
 [0.041899148752618985, 0.005860281754804516, 0.002269334389195936, 0.0022681538153602712, 0.0022670775543125817]
```

QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

ANSWERS

(1) The polynomial with the best training data was the 5th-order polynomial as it had the lowest training error.

(2) This is expected as higher-order models are expected to do at least as good as models with a lower order function. Higher-order models have more demnsions to fit to the exact shape of the data.

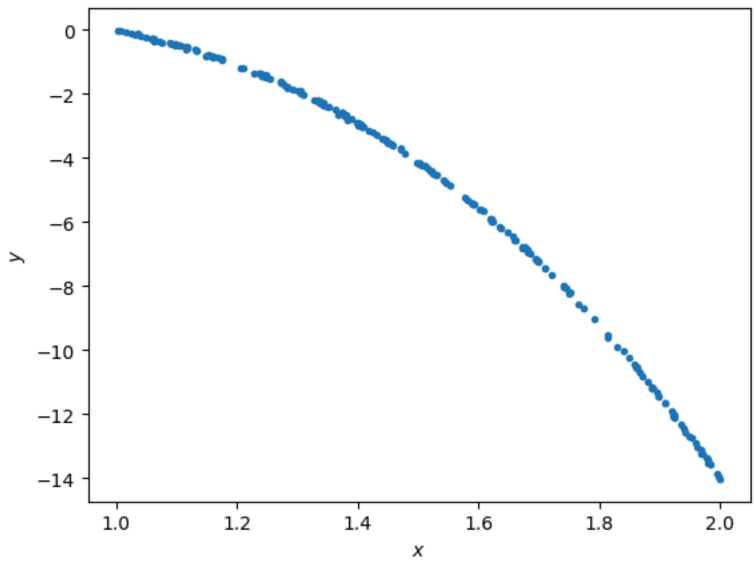
Generating new samples and testing error (5 points)

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 - 3*x**3 + np.random.normal(loc=0, scale=0.03, size=(num_train,))
```

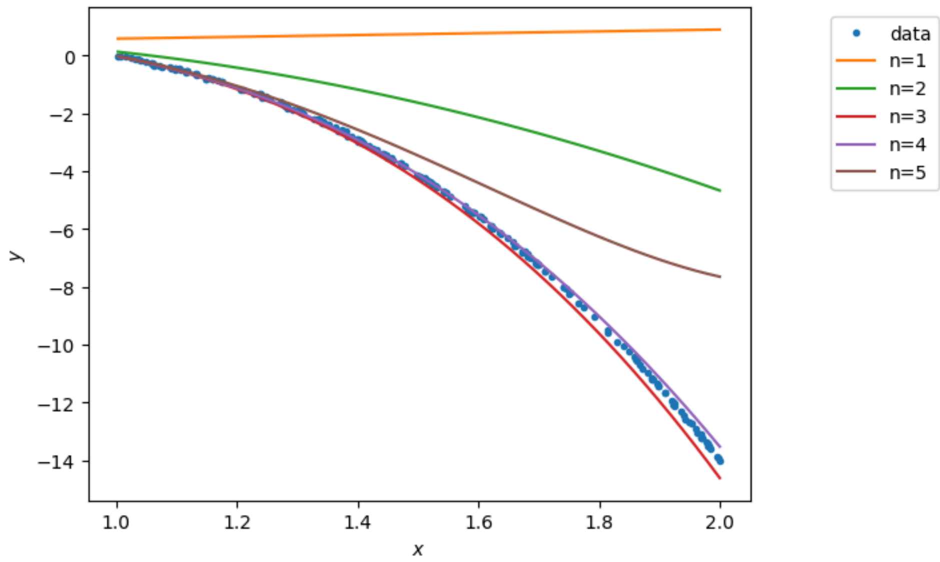
```
f = plt.figure()
ax = f.gca()
ax.plot(x, y, '.')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
```

```
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 i in range(N):
  diff = y - xhats[i].T.dot(thetas[i])
  testing_errors.append(diff.T.dot(diff)/num_train)

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

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

```
Testing errors are:
 [54.24631771601227, 18.91115921753081, 0.08693862570381264, 0.03042966867000048, 5.954356338935129]
```

QUESTIONS

(1) What polynomial has the best testing error?

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

ANSWERS

(1) The model with the best testing error was the 4th-order model as it has the lowest testing error.

(2) Models of order 5 may not generalize well because they are overfit to the training data. As a result, the model may pick up patterns that are common to the training set but are not actually common features of general cases.

```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc

from google.colab import drive
drive.mount('/content/drive')

!cp drive/My Drive/Colab Notebooks/ECE147HW1.ipynb ./

!jupyter nbconvert --to PDF "ECE147HW1.ipynb"
```

...