EE239AS
Xiaohan Wang
405033965

1. Solutions:
(a) i. example:

$$A=\begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}$$

eigenvalues and eigenvectors:

$$\det(A-\lambda I)=\det\begin{pmatrix} \frac{\sqrt{3}}{2}-\lambda & \frac{1}{2} \\ \frac{1}{2} & -\frac{\sqrt{3}}{2}-\lambda \end{pmatrix}=\lambda^2=1$$

$\Rightarrow \lambda_1=1, \lambda_2=-1$

for $\lambda_1=1$, we have

$$(A-\lambda_1 I)V_1=\begin{bmatrix} \frac{\sqrt{3}}{2}-1 & \frac{1}{2} \\ \frac{1}{2} & -\frac{\sqrt{3}}{2}-1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}=0 \Rightarrow y=(2-\sqrt{3})x$$

let $x^2+y^2=1 \Rightarrow x^2+(2-\sqrt{3})^2x^2=1 \Rightarrow x=\pm\sqrt{\frac{2+\sqrt{3}}{4}}, y=\pm\sqrt{\frac{2-\sqrt{3}}{4}}$

for $\lambda_2=-1$, we have

$$(A-\lambda_2 I)V_2=\begin{bmatrix} \frac{\sqrt{3}}{2}+1 & \frac{1}{2} \\ \frac{1}{2} & -\frac{\sqrt{3}}{2}+1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}=0 \Rightarrow x=(\sqrt{3}-2)y$$

let $x^2+y^2=1 \Rightarrow x=\pm\sqrt{\frac{2-\sqrt{3}}{4}}, y=\pm\sqrt{\frac{2+\sqrt{3}}{4}}$

$\therefore$ for eigenvalue $\lambda_1=1$, the eigenvector is $\begin{bmatrix} \pm\sqrt{\frac{2+\sqrt{3}}{4}} \\ \pm\sqrt{\frac{2-\sqrt{3}}{4}} \end{bmatrix}$,

for eigenvalue $\lambda_2=-1$, the eigenvector is $\begin{bmatrix} \pm\sqrt{\frac{2-\sqrt{3}}{4}} \\ \pm\sqrt{\frac{2+\sqrt{3}}{4}} \end{bmatrix}$.

let $V_1=\begin{bmatrix} \sqrt{\frac{2+\sqrt{3}}{4}} \\ -\sqrt{\frac{2-\sqrt{3}}{4}} \end{bmatrix}, V_2=\begin{bmatrix} \sqrt{\frac{2-\sqrt{3}}{4}} \\ -\sqrt{\frac{2+\sqrt{3}}{4}} \end{bmatrix}$

$\therefore V_1 \cdot V_2^T=0$.

$\therefore$ we can find that the eigenvalues satisfy $\|\lambda\|_2=1$.
eigenvectors corresponding to distinct eigenvalues are orthogonal.

ii. from the definition of eigenvalue, we have

$Av = \lambda v$.

$\Rightarrow ||Av|| = ||\lambda v|| = ||\lambda|| ||v||$

$\Rightarrow \sqrt{(Av)^T(Av)} = ||\lambda|| ||v||$

$\Rightarrow \sqrt{v^T v} = ||\lambda|| ||v|| \Rightarrow ||v|| = ||\lambda|| \cdot ||v||$

$\Rightarrow ||\lambda|| = 1$.

$\therefore$ generally A has eigenvalues with norm 1.

iii. let λ_1, λ_2 as two arbitrary eigenvalues of A. and v_1, v_2 are corresponding eigenvectors. then we have

$Av_1 = \lambda_1 v_1, \ Av_2 = \lambda_2 v_2$.

$\Rightarrow v_1^T v_2 = v_1^T I v_2 = v_1^T A^T A v_2 = (Av_1)^T (Av_2)$

$= (\lambda_1 v_1)^T (\lambda_2 v_2) = \lambda_1 \lambda_2 v_1^T v_2$.

$\because ||\lambda|| = 1 \Rightarrow \lambda_1 \lambda_2 \neq 0 \ldots$

$\because ||v_1|| \neq 0, ||v_2|| \neq 0$.

$\therefore v_1^T v_2 = 0 \Rightarrow v_1 \perp v_2$.

$\therefore$ generally the eigenvectors of A corresponding to distinct eigenvalues are orthogonal.

iv. under transformation Ax, the vector x may be rotated or reflected, but the length of the vector x will not change.

(b) i. the relationships are as below:

- $\circ$ left-singular vectors of A are eigenvectors of AA^T.
- $\triangle$ right-singular vectors of A are eigenvectors of A^TA.

ii. the relationships are as below:

the nonzero singular values of A are the square root of eigenvalues of AA^T, also the square root of eigenvalues of A^TA.

(c). i. false. Correct: every linear operator in an n-dimensional vector space has at most n distinct eigenvalues.
e.g. $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, it only has 1 distinct eigenvalue in 2-D vector space.

ii. false. eigenvectors corresponding to different eigenvalues are linearly irrelavance. which means, for $\forall \lambda_1, \lambda_2$; we have their eigenvectors v_1, v_2. then there's no such an eigenvectors $v_x = a \cdot v_1 + b v_2$, (a. b are some constants)

iii. true.

iv. false. $\text{rank}(A) = \text{nullity}(A) = n$. if $\text{nullity}(A) = x \geq 0$, then there are $n-x$ nonzero eigenvalues.
$\Rightarrow \text{rank}(A) = n - \text{nullity}(A) = n - x =$ number of nonzero eigenv

v. true.

2. Solutions:

(a) i. from the question, we have

$P(H_{50}) = 0.5$. $P(H_{50}, \text{heads}) = 0.5 \times 0.5 = 0.25$. $P(H_{50}, \text{tails}) = 0.5 - 0.25 = 0.$

$P(H_{60}) = 0.5$. $P(H_{60}, \text{heads}) = 0.5 \times 0.6 = 0.3$ $P(H_{60}, \text{tails}) = 0.5 - 0.3 = 0.2$

$\therefore P(H_{50}, \text{tails}) = P(H_{50}) \cdot P(\text{tails} | H_{50})$
$= P(\text{tails}) \cdot P(H_{50} | \text{tails})$

$$\Rightarrow P(H_{50} | \text{tails}) = \frac{P(H_{50}) \cdot P(\text{tails} | H_{50})}{P(\text{tails})} = \frac{0.5 \times 0.5}{0.25 + 0.2} = \frac{5}{9}.$$

ii. let event "lands T, H, H, H" as E. then we have

$$P(H_{50} | E) = \frac{P(H_{50}) \cdot P(E | H_{50})}{P(E)} = \frac{0.5 \times 0.5^4}{0.5 \times 0.5^4 + 0.5 \times 0.4 \times 0.6^3} \simeq 0.41974$$

iii. let event "lands heads 9 times" as E. then we have

$$P(E) = (\tfrac{1}{3} \times 0.5^{10} + \tfrac{1}{3} \times 0.55^9 \times 0.45 + \tfrac{1}{3} \times 0.6^9 \times 0.4) \times \tfrac{1}{10} \times 10$$

$$\therefore P(H_{50} | E) = \frac{P(H_{50}) \cdot P(E | H_{50})}{P(E)} = \frac{\frac{1}{3} \times 0.5^{10} \times \frac{1}{10} \times 10}{P(E)} = 0.01379$$

$$P(H_{55} | E) = \frac{P(H_{55}) \cdot P(E | H_{55})}{P(E)} = \frac{\frac{1}{3} \times 0.55^9 \times 0.45 \times \frac{1}{10} \times 10}{P(E)} = 0.2927$$

$$P(H_{60} | E) = \frac{P(H_{60}) \cdot P(E | H_{60})}{P(E)} = \frac{\frac{1}{3} \times 0.6^9 \times 0.4 \times \frac{1}{10} \times 10}{P(E)} = 0.5694.$$

(b) from the question, we have

$P(\text{pos}|\text{pregnant}) = 0.99$.

$P(\text{pos}|\text{not pregnant}) = 0.1$.

$P(\text{not pregnant}) = 0.99$.

$$\therefore P(\text{pregnant}|\text{pos}) = \frac{P(\text{pregnant}) \cdot P(\text{pos}|\text{pregnant})}{P(\text{pos})}$$

$$= \frac{0.01 \times 0.99}{0.99 \times 0.1 + 0.01 \times 0.99} \approx 0.0909.$$

we can see that the probability is very small. It said that at any given point in time, 99% of the female population is not pregnant, which means most female are not pregnant. So we may get many false positive results.

(c) $E(Ax+b) = E(Ax) + E(b) = E(Ax) + b = AE(x) + b$.

(d) $\text{cov}(Ax+b) = E((Ax+b-E(Ax+b))(Ax+b-E(Ax+b))^T)$

$= E((Ax+b-AE(x)-b)(Ax+b-AE(x)-b)^T)$

$= E(A(x-E(x)) \cdot (A(x-E(x)))^T)$

$= E(A(x-E(x)) \cdot (x-E(x))^T \cdot A^T)$

$= AE((x-E(x))(x-E(x))^T)A^T$

$= A \cdot \text{cov}(x) \cdot A^T$.

3. Solutions:

(a) $\nabla_x x^TAy = \frac{\partial x^TAy}{\partial x} = Ay$

(b) $\nabla_y x^TAy = \frac{\partial x^TAy}{\partial y} = (x^TA)^T = A^Tx$.

(c) $\nabla_A x^TAy = \frac{\partial x^TAy}{\partial A} \Rightarrow x^TAy = \sum_{i=1}\sum_{j=1} x_i y_j a_{ij} \Rightarrow \nabla_{a_{ij}} \sum_{i=1}\sum_{j=1} x_i y_j a_{ij} = xy^T$.

(d) let $p = x^TAx \Rightarrow p = \sum_{i=1}\sum_{j=1} a_{ij} x_i y_j. \Rightarrow \frac{\partial p}{\partial x_k} = \sum_{i=1} a_{ki} x_i + \sum_{j=1} a_{jk} x_j$

$\Rightarrow \frac{\partial p}{\partial x} = (A+A^T)x$.

$\frac{\partial b^Tx}{\partial x} = b. \Rightarrow \nabla_x f = (A+A^T)x + b$.

(e) $f = tr(AB) = \sum_{i=1} (AB)_{ii} = \sum_{i=1} \sum_{k=1} A_{ik} \cdot B_{ki}$.

$$\Rightarrow \frac{\partial tr(AB)}{\partial A_{ik}} = B_{ki} = B^T_{ik}.$$

$$\Rightarrow \nabla_A f = B^T.$$

4. Solutions:

let $R(W) = \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - Wx^{(i)})^2$

$$= \sum_{i=1}^{n} (y^{(i)} - (x_{i1} w_1 + x_{i2} w_2 + \cdots + x_{ip} w_p))^2$$

$$\therefore \frac{\partial R}{\partial w_j} = -\sum_{i=1}^{n} (y^{(i)} - x^{(i)T} W) x_{ij}$$

$$\Rightarrow \frac{\partial R}{\partial W} = \begin{pmatrix} \frac{\partial R}{\partial w_1} \\ \vdots \\ \frac{\partial R}{\partial w_p} \end{pmatrix} = \sum_{i=1}^{n} \left[(y^{(i)} - x^{(i)T} W) \begin{pmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix} \right]$$

$$= \sum_{i=1}^{n} (y^{(i)} - x^{(i)T} W) X^{(i)} = X^T (Y - XW)$$

let $\frac{\partial R}{\partial W} = 0 \Rightarrow \frac{\partial R}{\partial W} = \sum_{i=1}^{n} (y^{(i)} x^{(i)} - x^{(i)T} W x^{(i)}) = 0.$

$$\Rightarrow \sum_{i=1}^{n} y^{(i)} x^{(i)} = (\sum_{i=1}^{n} x^{(i)} x^{(i)T}) W$$

$$\Rightarrow W = (\sum_{i=1}^{n} x^{(i)} x^{(i)T})^{-1} (\sum_{i=1}^{n} x^{(i)} y^{(i)})$$

$$= \left[[x^{(1)} \cdots x^{(n)}] \begin{bmatrix} x^{(1)T} \\ \vdots \\ x^{(n)T} \end{bmatrix} \right]^{-1} \cdot \left[[x^{(1)} \cdots x^{(n)}] \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \right]$$

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

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 239AS, Winter Quarter 2019, Prof. J.C. Kao, TAs M. Kleinman and A. Wickstrom and K. Liang and W. Chuang

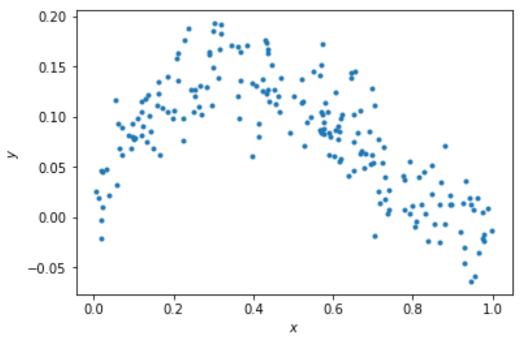
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

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

In [2]:
```
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('$y$')
```

Out[2]:
```
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 generating distribution of x is uniform distribution. The parameters are a = 0, b = 1.

(2) The distribution of the additive noise ϵ is Gaussian distribution. The mean is 0 and standard deviation is 0.03.

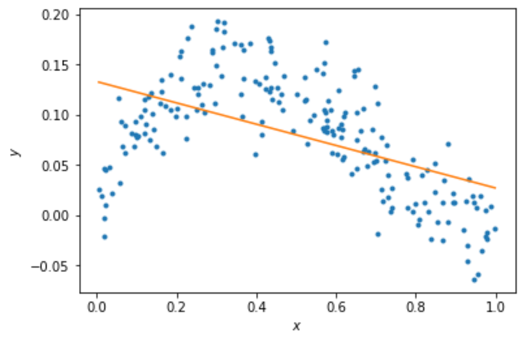
Fitting data to the model (5 points)

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

In [3]:
```
# 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]

x_trans = xhat.T
theta = np.linalg.inv(x_trans.T.dot(x_trans)).dot(x_trans.T.dot(y)) # please modify this line

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

In [4]:
```
# 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))
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x117edfc50>]
```



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.

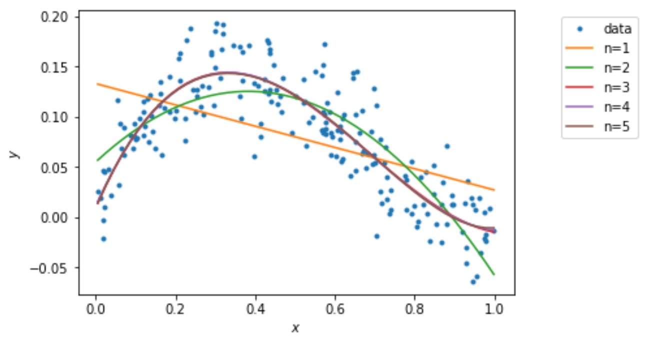
(2) We can change to a polynomial model with higher orders, like a quadratic model.

Fitting data to the model (10 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.

In [5]:
```python

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):
    xhat = np.ones_like(x)
    for j in range(1, i+1):
        xhat = np.vstack((x**j, xhat))

    x_trans = xhat.T
    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 #
# ================== #
```

In [6]:
```python
# 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 (10 points)

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

In [7]:
```python
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):
    pred = y - xhats[i].dot(thetas[i])
    training_errors.append(pred.T.dot(pred) / num_train)

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

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

```
Training errors are: 
 [0.002379961088362701, 0.0010924922209268528, 0.0008169603801105373, 0.0008165353735296978, 0.0008161479195525294]
```

QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

ANSWERS

(1) The polynomial with order 5 has the best (lowest) training error.

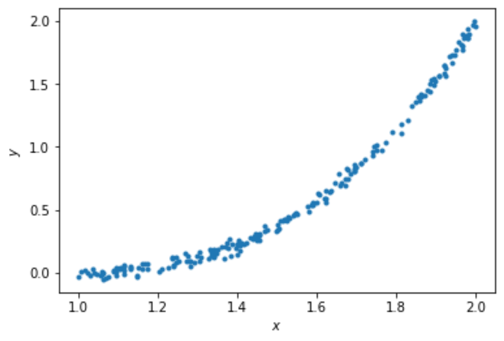
(2) Intuitively, higher order polynomial function gives more dimensions to fit the data. We can adjust more parameters to control the shape of the function, and make it more fitted.

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.

In [8]:
```python
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('$y$')
```

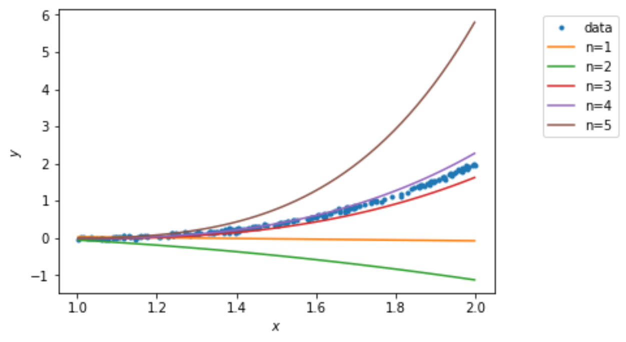
Out[8]:
```
Text(0, 0.5, '$y$')
```



In [9]:
```python
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)
```

In [10]:

```python
# 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)
```



In [11]:

```python
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):
    test = y - xhats[i].T.dot(thetas[i])
    testing_errors.append(test.T.dot(test) / num_train)

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

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

```
Testing errors are:
 [0.8086165184550584, 2.131919244505791, 0.03125697108408374, 0.011870765211496222, 2.14910217470128]
```

QUESTIONS

(1) What polynomial has the best testing error?

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

ANSWERS

(1) The polynomial with order 4 has the best (lowest) testing error.

(2) The polynomial model with order 5 has poor performance because it overfits the data. The model learns the information from both the input data (x, y) and the noise. For different data, we will get different order-5 model with exact information. But these models are not general. So if we change the input data but still with this model, then the testing error would be large.

In [ ]: