ECE C147/C247 Winter 2023 Homework #1

Neural Networks & Deep Learning

UCLA ECE

Yaman Yucel, UID = 605704529

1) Linear algebra Refresher

a) Q real orthogonal matrix. $\Rightarrow$ $QQ^T = QQ^{-1} = I$, $Q^TQ = Q^{-1}Q = I$

i) $Q^T = A \Rightarrow A^TA \overset{?}{=} I$

$(Q^T)^T Q^T = QQ^T \overset{\checkmark}{=} I$, Q^T is an orthogonal matrix

Since $Q^T = Q^{-1} \Rightarrow Q^{-1}$ is an orthogonal matrix.

ii) $Qx = \lambda x$

$(Qx)^T Qx = |\lambda|^2 x^Tx$

$x^TQ^TQx = \downarrow x^Tx$

Then, $x^Tx = \lambda^2 x^Tx$

$\lambda^2 = 1 \Rightarrow \boxed{|\lambda| = 1}$

iii)

$QQ^T = I$

$\downarrow$

$\det(I) = \det(QQ^T)$

$\det(I) = 1$

$\Rightarrow \det(QQ^T) = 1$

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

$\Rightarrow \det(Q)^2 = 1 \rightarrow \boxed{\det(Q) = \mp 1}$

iv) $\boxed{\|Qx\|^2} = (Qx)^T(Qx) = x^T Q^T Q x = x^T x = \boxed{\|x\|^2}$

Therefore, Q is a length-preserving transformation

6) $A = U \Sigma V^T$ → compact form

$AA^T = U\Sigma V^T V \Sigma U^T = U \Sigma^2 U^T$, $A^TA = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T$

i) ⇒ eigenvectors of AA^T are left singular vectors of AA^T.

ii) ⇒ eigenvectors of A^TA are right singular vectors of A^TA.

⇒ eigenvalues of A^TA and AA^T are squared singular values of A.

$\boxed{\begin{aligned} eig(AA^T) &= \sigma(A)^2 \\ eig(A^TA) &= \sigma(A)^2 \end{aligned}}$

c)

i) $A = I_{n \times n} \Rightarrow \lambda_i = 1$ for all $1 \le i \le n$

⇒ eigenvalues are not distinct, $\boxed{\text{FALSE}}$

ii) $Ax_1 = \lambda_1 x_1$

$Ax_2 = \lambda_2 x_2$

$A(x_1 + x_2) = \lambda_1 x_1 + \lambda_2 x_2 \stackrel{?}{=} \lambda(x_1 + x_2)$

This is only true if $\lambda_1 = \lambda_2 = \lambda$, therefore in general it is $\boxed{\text{FALSE}}$

Ex $\lambda_1 = 2, \lambda_2 = 3$

$x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow 2\begin{bmatrix} 1 \\ 0 \end{bmatrix} + 3\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$

$\begin{bmatrix} 2 \\ 3 \end{bmatrix} \neq \lambda \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow \lambda$ can not be found.

iii) $x^T A x \geq 0 \Rightarrow (x^T x)\lambda \geq 0$

$x^T x = \|x\|^2 \geq 0$, then $\lambda \geq 0$

λ must be non-negative

TRUE

iv) $A = I_{n \times n}$

$rank(A) = n$, $eig(A) = 1$

the rank of A is n, but number of distinct non-zero eigenvalues is 1. Therefore, rank can exceed the number of distinct non-zero eigenvalues.

TRUE, $rank(A) = r$, then there are at most r distinct non-zero eigenvalues

v) $Ax = \lambda x$

$Ay = \lambda y$

$x + y \neq 0$

$A(x+y) = \lambda x + \lambda y = \lambda(x+y)$

$Az = \lambda z$, z is an eigenvector, **TRUE**

2) Probability refresher

a) $Pr(X = HSO) = 0.5 = p(HSO)$

$Pr(X = HO) = 0.5 = p(HO)$

$Pr(Y = H \mid X = HSO) = 0.5 = p(H \mid HSO)$

$Pr(Y = H \mid X = HO) = 0.6 = p(H \mid HO)$

$Pr(Y = T \mid X = HSO) = 0.5 = p(T \mid HSO)$

$Pr(Y = T \mid X = HO) = 0.4 = p(T \mid HO)$

i) $P(HSO \mid T) = \dfrac{P(T \mid HSO) \cdot P(HSO)}{P(T \mid HSO) \cdot P(HSO) + P(T \mid HO) P(HO)}$

$= \dfrac{0.5 \times 0.5}{0.5 \times 0.5 + 0.4 \times 0.5} = \dfrac{0.5}{0.9} = \dfrac{5}{9} = 0.56$

ii) $P(T,H,H,H \mid HSO) = (0.5)^4$ (i) flipping is an independent event

$P(T,H,H,H \mid HO) = 0.4\,(0.5)^3$

$P(HSO \mid T,H,H,H) = \dfrac{P(HSO)(0.5)^4}{P(HSO)(0.5)^4 + P(HO)(0.4)(0.6)^3}$

$= \dfrac{(0.5)^4}{(0.5)^4 + (0.4)(0.6)^3} = 0.4197$

iii) $Pr(X = HSO) = 1/3 = p(HSO)$

$Pr(X = HGO) = 1/3 = p(HGO)$

$Pr(X = HSS) = 1/3 = p(HSS)$

$Pr(Y = H \mid X = HSO) = 0.5 = p(H \mid HSO)$

$Pr(Y = H \mid X = HGO) = 0.6 = p(H \mid HGO)$

$Pr(Y = H \mid X = HSS) = 0.55 = p(H \mid HSS)$

$Pr(Y = T \mid X = HSO) = 0.5 = p(T \mid HSO)$

$Pr(Y = T \mid X = HGO) = 0.4 = p(T \mid HGO)$

$Pr(Y = T \mid X = HSS) = 0.45 = p(T \mid HSS)$

$P(9H, 1T) = \frac{1}{3}(0.5)^{10} + \frac{1}{3}(0.55)^9(0.45) + \frac{1}{3}(0.60)^9(0.4)$

$(3.255 \times 10^{-4}) + 6.908 \times 10^{-4} + 0.001343$

$= 0.00236$

$P(HSO \mid 9H, 1T) = \frac{3.255 \times 10^{-4}}{0.00236} = 0.13793 \cong 0.14$

$P(HSS \mid 9H, 1T) = \frac{6.908 \times 10^{-4}}{0.00236} = 0.29271 \cong 0.30$

$P(HGO \mid 9H, 1T) = \frac{0.001343}{0.00236} = 0.56935 \cong 0.56$

which makes sense since we got a lot of heads than tails

b) $Pr(X = Science) = 0.15 \Rightarrow P(S) = 0.15$

$Pr(X = Healthcare) = 0.21 \Rightarrow P(H) = 0.21$

$Pr(X = LA) = 0.24 \Rightarrow P(L) = 0.24$

$Pr(X = Engi) = 0.4 \Rightarrow P(E) = 0.4$

$P(Y = Liked \mid X = Science) = 0.9$

$P(Y = Liked \mid X = Hea..) = 0.18$

$P(Y = Liked \mid X = LA) = 0$

$P(Y = Liked \mid X = Engi) = 0.10$

$$P(S \mid L) = \frac{P(L \mid S)\,P(S)}{P(S)P(L \mid S) + P(H)\,P(L \mid H) + \underbrace{\dots}_{\text{2 more terms}}}$$

$$= \frac{0.15 \times 0.9}{0.15 \times 0.9 + 0.21 \times 0.18 + 0 \times 0.24 + 0.10 \times 0.4}$$

$$\simeq \frac{0.135}{0.2128} = 0.63439 \simeq \boxed{0.63}$$

c) $P(P \mid preg) = 0.99 \Rightarrow P(N \mid preg) = 0.01$

$P(P \mid notpreg) = 0.10$

$P(notpreg) = 0.99$ $\qquad P(notpreg) = 0.01$

$$P(preg \mid P) = \frac{0.99 \times 0.01}{0.99 \times 0.01 + 0.1 \times 0.99} = \frac{0.099}{0.1089} = \boxed{0.091}$$

We are more likely to choose someone who is not pregnant since pregnant population is low.

d) $$Ax+b = \begin{bmatrix} a_1^T x + b_1 \\ \vdots \\ a_n^T x + b_n \end{bmatrix}$$

$$E[Ax+b] = \begin{bmatrix} E[a_1^T x + b_1] \\ \vdots \\ E[a_n^T x + b_n] \end{bmatrix}$$

$$E[a_i^T x + b_i] = E[a_i^T x] + b_i$$

$$E[a_i^T x] = E\left[\sum_{j=1}^{n} (a_i)_j x_j\right] = \sum_{j=1}^{n} (a_i)_j E[x_j]$$

$$= \sum_{j=1}^{n} (a_i)_j E[x_j] + b_i = a_i^T E[x] + b_i$$

$$E[Ax+b] = \begin{bmatrix} a_1^T E[x] + b_1 \\ \vdots \end{bmatrix}$$

$$\boxed{= A E[x] + b} \quad \text{where } E[x] = \begin{bmatrix} E[x_1] \\ \vdots \\ E[x_n] \end{bmatrix}$$

e) $$cov(Ax+b) = E[(Ax+b - E[Ax+b])(\quad)^T]$$

$$= E[(Ax+b - A E[x] - b)(\quad)^T]$$

$$= E[(A(x - E[x]))(A(x - E[x]))^T]$$

$$= E[(A(x - E[x]))((x - E[x])^T A^T)]$$

$$= A E[(x - E[x])(x - E[x])^T] A^T$$

$$\boxed{= A\, cov(x)\, A^T}$$

3) $\nabla_x x^T A y$

a) $x^T A y = x^T b = b^T x \Rightarrow$

$\nabla_x b^T x = b = \boxed{Ay}$

b) $\nabla_y x^T A y$

$x^T A y = b^T y$

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

c) $x^T A y = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} x_i y_j$

$\frac{\partial \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} x_i y_j}{\partial a_{ij}} = x_i y_j \Rightarrow \begin{bmatrix} x_1 y_1 & \cdots & x_1 y_1 \\ x_2 y_1 & & \\ \vdots & & \\ x_n y_1 & & x_n x_1 \end{bmatrix}$

$= \boxed{x y^T}$

d) $\nabla_x x^T A x = A^T x + A x$

$\nabla_x b^T x = b$

$\nabla_x (x^T A x + b^T x) = A^T x + A x + b$

$\boxed{= (A^T + A) x + b}$

e) $\nabla_A tr(AB)$

$tr(AB) = \sum_i \sum_j a_{ij} b_{ji}$

$\frac{\partial f}{\partial a_{ij}} = b_{ji} \Rightarrow \nabla_A tr(AB) = \boxed{B^T}$

f) $tr(BA + A^T B + A^2 B) =$

$tr(BA) + tr(A^T B) + tr(A^2 B)$

$\nabla_A tr(BA) = \nabla_A tr(AB) = B^T \rightarrow$ cook book (100)

$\nabla_A tr(A^T B) \underset{tr(C) = tr(C^T)}{=} \nabla_A tr(B^T A) \underset{= tr(BA)}{tr(AB)} = \nabla_A tr(AB^T) = B$ (cookbook (103))

$\nabla_A tr(A^2 B) = (AB + BA)^T \rightarrow$ cook book (107)

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

$= B^T + B + B^T A^T + A^T B^T$

g) $\|A + \lambda B\|_F^2 = Tr((A + \lambda B)(A + \lambda B)^T)$

$(A + \lambda B)(A^T + \lambda B^T)$

$(AA^T + \lambda AB^T + \lambda BA^T + \lambda^2 BB^T) = g(A)$

$\nabla_A Tr(g(A)) = \nabla_A Tr(AA^T) + \nabla_A Tr(\lambda AB^T)$

$+ \nabla_A Tr(\lambda BA^T) + \nabla_A Tr(\lambda^2 BB^T)$

$$\Rightarrow \nabla_A Tr(AA^T) = 2A \quad (115)$$

$$\Rightarrow \nabla_A Tr(\lambda AB^T) = \lambda \nabla_A Tr(AB^T) = \lambda B$$

$$= \nabla_A Tr(\lambda BA^T) = \lambda \nabla_A Tr(BA^T) =$$

$$\lambda \nabla_A Tr(AB^T) = \lambda B$$

$$\Rightarrow 2A + 2\lambda B \Rightarrow \boxed{2(A + \lambda B)}$$

4) $\hat{y} = wx$

$$X = \begin{bmatrix} x^{(1)T} \\ \vdots \\ x^{(n)T} \end{bmatrix} \quad , Y = \begin{bmatrix} y^{(1)T} \\ \vdots \\ y^{(n)T} \end{bmatrix}$$

$$\hat{Y} = Xw^T$$

$$J(w) = \frac{1}{2} \sum_{i=1}^{n} \| y^{(i)} - wx^{(i)} \|^2$$

$$= \frac{1}{2} \| Y - \hat{Y} \|_F^2$$

$$= \frac{1}{2} Tr\left[(Y - Xw^T)(Y^T - wX^T) \right]$$

$$= \frac{1}{2} \Big(Tr(YY^T) - Tr(Xw^TY^T) - Tr(Ywx^T) +$$

$$Tr(Xw^Twx^T) \Big)$$

$$\nabla_W Tr(YY^T) = 0$$

$$\nabla_W Tr(XW^TY^T) \overset{Tr(A)=Tr(A^T)}{=} \nabla_W Tr(YWX^T) = Y^TX \quad \begin{pmatrix} \text{cookbook} \\ 101 \end{pmatrix}$$

$$\nabla_W Tr(\underset{A}{X}\underset{B}{W^T}\underset{C}{W}\underset{D}{X^T}) = \nabla_W Tr(\underset{C}{W}\underset{D}{X^T}\underset{A}{X}\underset{B}{W^T}) = WX^TX + WX^TX$$

$Tr(ABCD) = Tr(CDAB)$ ~ Cyclic permutation

cookbook (111)

↗ set derivative to zero

Total

$$\Rightarrow \frac{1}{2}(-2Y^TX + 2WX^TX) = 0$$

$$\Rightarrow -Y^TX + WX^TX = 0$$

$$WX^TX = Y^TX$$

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

5)

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

$$Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}, \quad X = \begin{bmatrix} \hat{x}^{(1)} \\ \vdots \\ \hat{x}^{(n)} \end{bmatrix}$$

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

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

$$\frac{\partial J}{\partial \theta} = \frac{1}{2} \left[0 - 2X^T y + 2X^T X\theta + 2\lambda I\theta \right]$$

set derivative to 0

$$= -X^T y + (X^T X + \lambda I)\theta \overset{!}{=} 0$$

$$= (X^T X + \lambda I)\theta = X^T y$$

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

perturbation

6) Linear Regression

linear_regression

January 22, 2023

0.1 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 2023, Prof. J.C. Kao, TAs: T.M, P.L, R.G, K.K, N.V, S.R, S.P, M.E

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
!pip install -r '/content/drive/My Drive/DL_247/HW1/requirements.txt';
```

```
Mounted at /content/drive
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting appnope==0.1.0
  Downloading appnope-0.1.0-py2.py3-none-any.whl (4.0 kB)
Collecting bleach==2.1.2
  Downloading bleach-2.1.2-py2.py3-none-any.whl (27 kB)
Collecting decorator==4.1.2
  Downloading decorator-4.1.2-py2.py3-none-any.whl (9.1 kB)
Collecting entrypoints==0.2.3
  Downloading entrypoints-0.2.3-py2.py3-none-any.whl (9.4 kB)
Collecting fancycompleter==0.8
  Downloading fancycompleter-0.8.tar.gz (514 kB)
                           515.0/515.0 KB
12.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: html5lib==1.0.1 in /usr/local/lib/python3.8/dist-
packages (from -r /content/drive/My Drive/DL_247/HW1/requirements.txt (line 6))
(1.0.1)
Collecting ipykernel==4.7.0
  Downloading ipykernel-4.7.0-py3-none-any.whl (106 kB)
                           106.9/106.9 KB
12.1 MB/s eta 0:00:00
Collecting ipython==6.2.1
  Downloading ipython-6.2.1-py3-none-any.whl (745 kB)
                           745.9/745.9 KB
35.6 MB/s eta 0:00:00
```

```
Requirement already satisfied: ipython-genutils==0.2.0 in
/usr/local/lib/python3.8/dist-packages (from -r /content/drive/My
Drive/DL_247/HW1/requirements.txt (line 9)) (0.2.0)
Collecting ipywidgets==7.1.0
  Downloading ipywidgets-7.1.0-py2.py3-none-any.whl (68 kB)
                           68.4/68.4 KB
7.7 MB/s eta 0:00:00
Collecting jedi==0.11.1
  Downloading jedi-0.11.1-py2.py3-none-any.whl (250 kB)
                          250.4/250.4 KB
26.3 MB/s eta 0:00:00
Collecting Jinja2==2.10
  Downloading Jinja2-2.10-py2.py3-none-any.whl (126 kB)
                           126.4/126.4

KB 9.7 MB/s eta 0:00:00
Collecting jsonschema==2.6.0
  Downloading jsonschema-2.6.0-py2.py3-none-any.whl (39 kB)
Collecting jupyter==1.0.0
  Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Collecting jupyter-client==5.2.1
  Downloading jupyter_client-5.2.1-py2.py3-none-any.whl (88 kB)
                           88.4/88.4 KB
7.2 MB/s eta 0:00:00
Collecting jupyter-console==5.2.0
  Downloading jupyter_console-5.2.0-py2.py3-none-any.whl (20 kB)
Collecting jupyter-core==4.4.0
  Downloading jupyter_core-4.4.0-py2.py3-none-any.whl (126 kB)
                          126.8/126.8 KB
13.4 MB/s eta 0:00:00
Collecting MarkupSafe==1.0
  Downloading MarkupSafe-1.0.tar.gz (14 kB)
  error: subprocess-exited-with-error

  × python setup.py egg_info did not run successfully.
   exit code: 1
   > See above for output.

  note: This error originates from a subprocess, and is likely not a
problem with pip.
  Preparing metadata (setup.py) … error
error: metadata-generation-failed

× Encountered error while generating package metadata.
 > See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

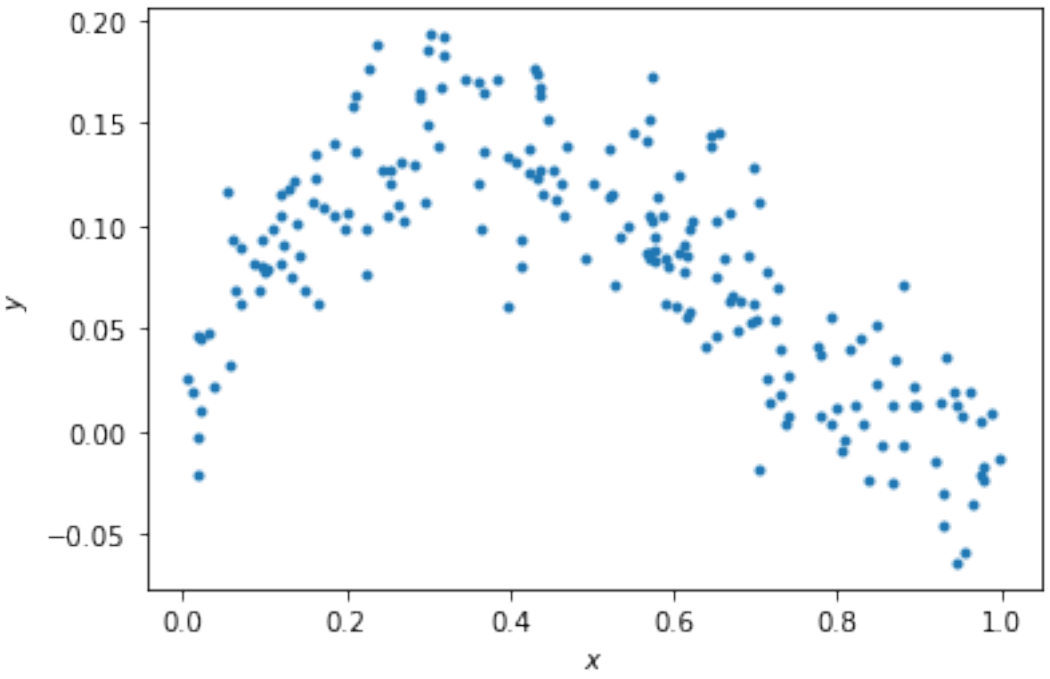
[2]:
```
import numpy as np
import matplotlib.pyplot as plt

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

0.1.1 Data generation

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

[3]:
```
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('$y$')
```

[3]:
```
Text(0, 0.5, '$y$')
```



0.1.2 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?

0.1.3 ANSWERS:

(1) x is uniformly distributed between 0 and 1 (1 excluded).

(2) ϵ is normally distributed with $\mu = 0$ and $\sigma = 0.03$.

0.1.4 Fitting data to the model (5 points)

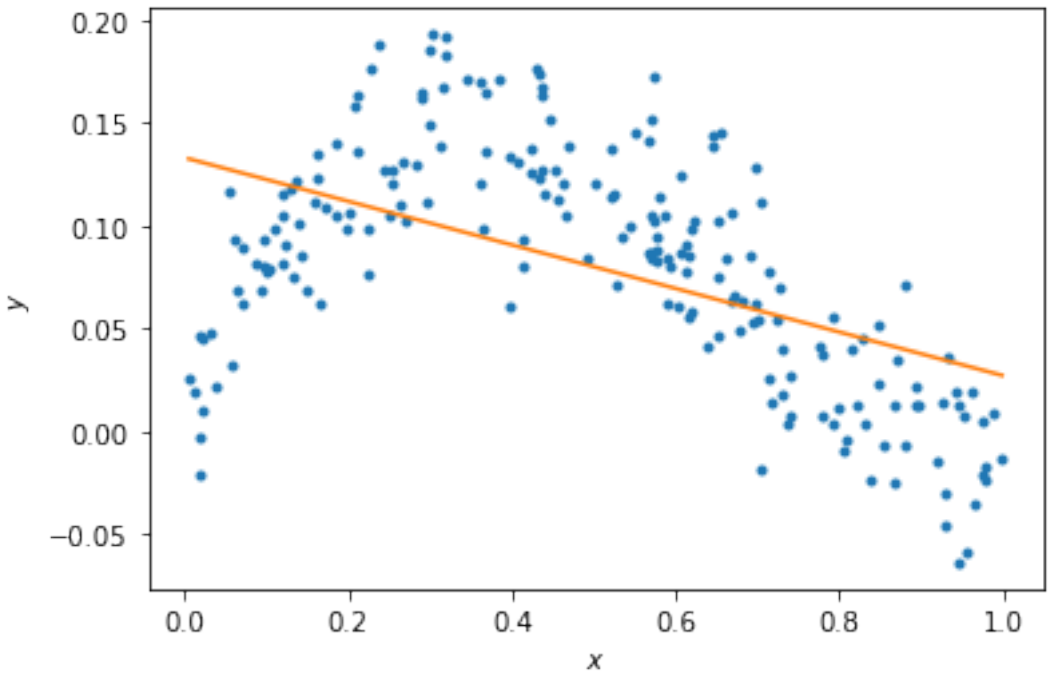
Here, we'll do linear regression to fit the parameters of a model $y = ax + b$.

[4]:
```
# 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]
xhat = xhat.T # xhat Nx2
x_p = np.matmul(np.linalg.inv(np.matmul(np.transpose(xhat), xhat)), np.
 ↪transpose(xhat)) #(X.TX)^-1X.T
theta = np.matmul(x_p, y) # #(X.TX)^-1X.Ty
print(theta)
# ================== #
# END YOUR CODE HERE #
# ================== #
```

```
[-0.10599633  0.13315817]
```

[5]:
```
# 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))
```

[5]:
```
[<matplotlib.lines.Line2D at 0x7f944b86eeb0>]
```



0.1.5 QUESTIONS

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

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

0.1.6 ANSWERS

(1) Underfit the data since data seems to be more complex than our fit. (3 dimensional)

(2) Polynomial of higher degree can be used to improve fitting.

0.1.7 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.

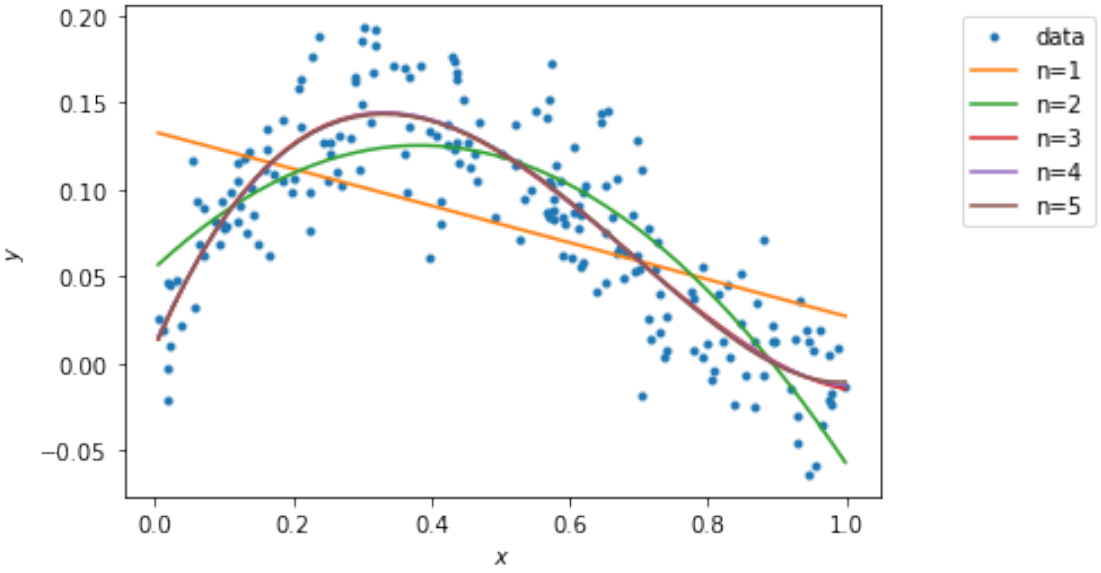
[6]:
```
N = 5
xhats = []
thetas = []

# ==================== #
# 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.
xhats.append(xhat)
for i in range(0,N):
  if i != 0:
    xhats.append(np.hstack((x.reshape(-1,1)**(i+1), x_hat)))

  x_hat = xhats[i]
  x_p = np.matmul(np.linalg.inv(np.matmul(np.transpose(x_hat),x_hat)), np.
 ↪transpose(x_hat))
  thetas.append(np.matmul(x_p, y))



print(thetas)
# ================== #
# END YOUR CODE HERE #
# ================== #
```

```
[array([-0.10599633,  0.13315817]), array([-0.48023061,  0.36743967,
0.05521084]), array([ 0.8843808 , -1.82077417,  0.91178032,  0.00979068]),
array([ 0.14080037,  0.60466289, -1.64250929,  0.87250485,  0.01175321]),
array([ 0.52432591, -1.164568  ,  1.76052438, -2.07430275,  0.93373916,
        0.009716  ])]
```

[7]:
```
# 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)
```



0.1.8 Calculating the training error (5 points)

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

[8]:
```
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(0,N):
  theta = thetas[i]
  xhat = xhats[i]
  yhat = np.matmul(xhat,theta)
  error = np.sum((yhat - y)**2)/2
  training_errors.append(error)

print("Polynomial with the best training error: ", np.argmin(training_errors) +␣
 ↪1,"th degree polynomial.")
```

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

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

```
Polynomial with the best training error:  5 th degree polynomial.
Training errors are:
 [0.2379961088362701, 0.10924922209268528, 0.08169603801105368,
0.08165353735296982, 0.08161479195525292]
```

0.1.9 QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

0.1.10 ANSWERS

(1) Polynomial with the best training error is 5th degree polynomial

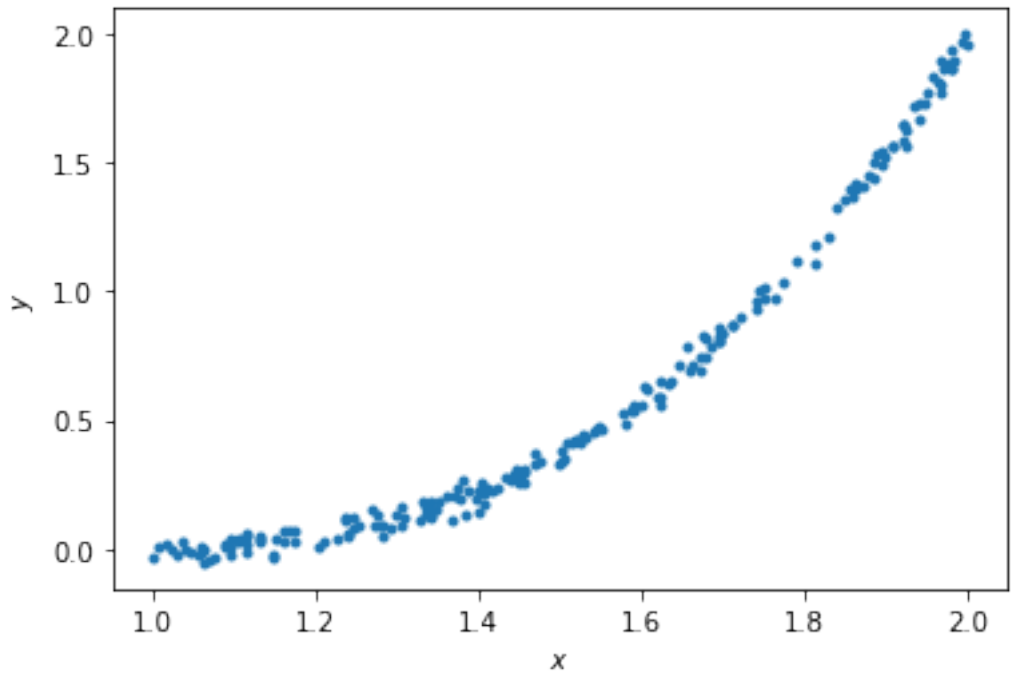
(2) Higher degree polynomials have higher flexibility to fit the training data since they have more parameters to be trained.

0.1.11 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.

[9]:
```
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('$y$')
```

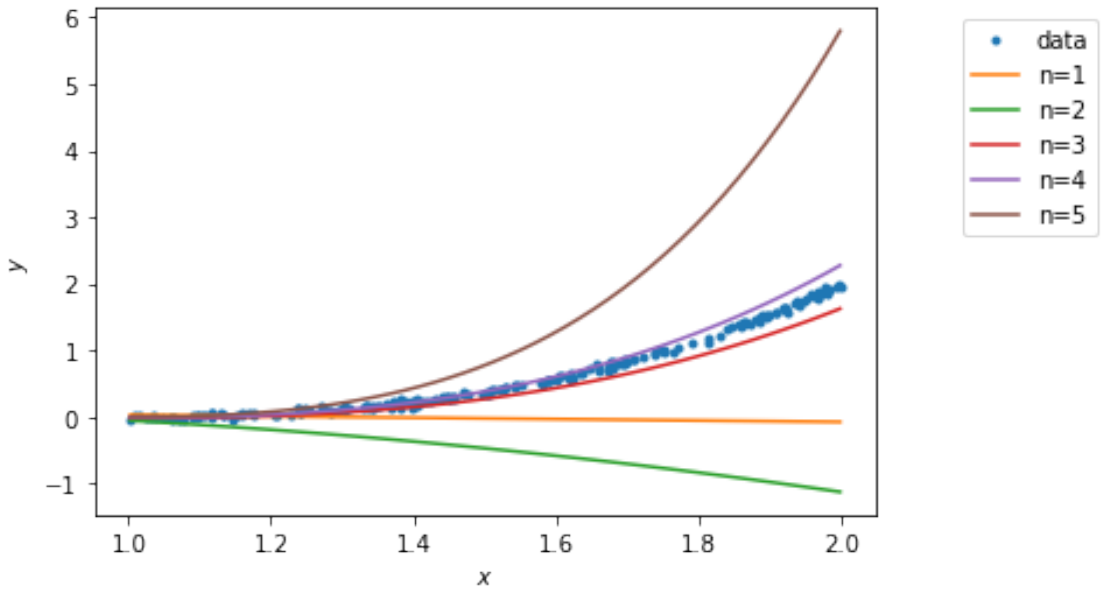
[9]:
```
Text(0, 0.5, '$y$')
```



[10]:
```
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)
```

[11]:
```
# 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)
```



[12]:
```
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(0,N):
  theta = thetas[i]
  xhat = (xhats[i]).T
  yhat = np.matmul(xhat,theta)
  error = sum((yhat - y)**2)/2
  testing_errors.append(error)
```

```
print("Polynomial with the best test error: ", np.argmin(testing_errors) +␣
 ↪1,"th degree polynomial.")
# ================== #
# END YOUR CODE HERE #
# ================== #

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

```
Polynomial with the best test error:  4 th degree polynomial.
Testing errors are:
 [80.86165184550579, 213.19192445057894, 3.1256971084092977, 1.1870765211189493,
214.9102174865315]
```

0.1.12 QUESTIONS

(1) What polynomial has the best testing error?

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

0.1.13 ANSWERS

(1) Polynomial with the best test error is 4th degree polynomial.

(2) Polynomial models with orders 5 and higher tend to fit the error introduced in the first section which is called also overfitting. Therefore, these polynomials can not perform well to unseen test data since error samples are different.

[12]: