Amirali Omidfar UID: 204869241

1

HW 1 Amirali Omidfar

1 a)

i) $A = \begin{bmatrix} .6 & .8 \\ .8 & -.6 \end{bmatrix}$ $AA^T = \begin{bmatrix} .6 & .8 \\ .8 & -.6 \end{bmatrix}\begin{bmatrix} .6 & .8 \\ .8 & -.6 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ ✓✓

$AX = \lambda X$

$AX - \lambda X = 0$

$(A - I\lambda)X = 0$

$\begin{bmatrix} .6-\lambda & .8 \\ .8 & -0.6-\lambda \end{bmatrix} X = 0$

$\det|A - \lambda I| = (.6-\lambda)(-.6-\lambda) - .8^2$

$\Rightarrow \lambda^2 - 1 = 0 \Rightarrow \lambda = \begin{cases} 1 \\ -1 \end{cases}$

if $\lambda_1 = 1 \Rightarrow \left[\begin{array}{cc|c} -.4 & .8 & 0 \\ .8 & -1.6 & 0 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} -.4 & .8 & 0 \\ 0 & 0 & 0 \end{array}\right]$

Eigenvector $V_1 = \begin{bmatrix} x_1 \\ x_1/2 \end{bmatrix}$

if $\lambda = -1 \Rightarrow \left[\begin{array}{cc|c} 1.6 & .8 & 0 \\ .8 & .4 & 0 \end{array}\right] \Rightarrow \left[\begin{array}{cc|c} 1.6 & .8 & 0 \\ 0 & 0 & 0 \end{array}\right]$

Eigenvector $V_2 = \begin{bmatrix} x_1 \\ -2x_1 \end{bmatrix}$

$\Rightarrow |\det A| = 1$ & eigenvectors are orthogonal to each other

$|\lambda| = 1$

ii) $Av_i = \lambda_i v_i$

$\Rightarrow \|Av_i\|_2^2 = \|\lambda_i v_i\|_2^2$

$\Rightarrow v_i^T \underbrace{A^T A}_{I} v_i = \lambda_i^2 \|v_i\|_2^2$

$\Rightarrow v_i^T v_i = \underbrace{\lambda_i^2}_{1} \|v_i\|_2^2$

$\Rightarrow \|v_i\|_2^2 = \|v_i\|_2^2$ //

iii) Given v_1 & v_2 being the eigenvectors for λ_1 & λ_2

$\Rightarrow v_1 . v_2 = v_2 . v_1$

$\Rightarrow A(v_1 . v_2) = A(v_2 . v_1)$

$\Rightarrow (Av_1) . v_2 = (Av_2) . v_1$

$\Rightarrow (\lambda_1 v_1) . v_2 = (\lambda_2 v_2) . v_1$

$\Rightarrow \lambda_1 (v_1 . v_2) - \lambda_2 (v_2 . v_1) = 0$

$\Rightarrow (\lambda_1 - \lambda_2)(v_1 . v_2) = 0$

since $\lambda_1 \neq \lambda_2 \Rightarrow \lambda_1 - \lambda_2 \neq 0$

So $v_1 . v_2 = 0 \Rightarrow \boxed{v_1 \perp v_2 \text{ //}}$

iv) Any x is subject to rotation or reflection under Ax when $|\det A| = 1$ in this case its reflection as $\det A = -1$

The length is preserved.

b)

i) $A = U \Sigma V^T$

$AA^T = U \Sigma \underbrace{V^T V}_{I} \Sigma^T U^T =$ | $A^T A = V \Sigma^T \underbrace{U^T U}_{I} \Sigma V^T$

$AA^T = U \underbrace{\Sigma \Sigma^T}_{\text{diagonal matrix}} U^T$ | $A^T A = V \underbrace{\Sigma^T \Sigma}_{\text{diagonal matrix}} V^T$

According to Eigendecomposition $\Rightarrow$ singular vectors of A are eigenvectors of AA^T & $A^T A$.

Also λ's for $A^T A$ & AA^T are σ's for A (singular values)

ii) As shown ~~A above~~ : $AA^T = \underbrace{U}_{X} \underbrace{\Sigma \Sigma^T}_{\Lambda} U^T \Rightarrow$ so

λ's for AA^T & AA^T are singular values of A squared!!

c)

i) False $\quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ has one only

ii) False $\quad$ if eigenvalues are different then it doesn't hold

$$v_3 = v_1 + v_2 \Rightarrow A[v_3] = Av_1 + Av_2 = \lambda_1 v_1 + \lambda_2 v_2 \neq \lambda(v_1 + v_2)$$

unless $\lambda_1 = \lambda_2$

iii) True. If λ is an eigenvalue for $A \Rightarrow Ax = \lambda x$

$$\Rightarrow \begin{cases} x^T A x \geq 0 \\ x^T A x = \lambda x^T x \end{cases} \Rightarrow \begin{cases} \lambda x^T x \geq 0 \\ x^T x \geq 0 \text{ is always greater than zero} \end{cases}$$

$\Rightarrow \lambda \geq 0$ So the λ's are non negative.

iv) False, according to the theorem: $A \in R^{m \times n} \Rightarrow n = \text{Rank}A + \text{nullity of }A$

$\Rightarrow$ if v is in the nullity of $A \Rightarrow Av = 0$ so v is eigenvector for 0 eigenvalue. Since there exists an eigenspace of degree at least one for each eigenvalue, the number of non-zero eigenvalues must be less than or equal to the rank.

v) True, Assume: $A \in \mathbb{R}^{n \times n}$ & $\begin{matrix} Av = \lambda v \\ Av' = \lambda v' \end{matrix}$

$v, v' \in \mathbb{R}^{n \times 1}$

$\Rightarrow C = A(v + v') \Rightarrow C_{i1} = \sum_{k=1}^{n} a_{ik}(v_{k1} + v'_{k1}) = \sum a_{ik} v_{k1} + \sum a_{ik} v'_{k1}$

$\Rightarrow C = Av + Av' \Rightarrow \lambda v + \lambda v' = \lambda(v + v') \Rightarrow A(v + v') = \lambda(v + v')$ //

2. $P(H|H50) = 0.5$, $P(H|H60) = 0.6$

i) $P(H50|\overset{\text{tail}}{T}) = \dfrac{P(T|H50)P(H50)}{P(T|H50)P(H50) + P(T|H60)P(H60)}$

$$= \frac{(0.5)(0.5)}{(0.5)(0.5) + (0.4)(0.5)} = \boxed{5/9}$$

ii)

$S = THHH$

$$P(H50|S) = \frac{P(S|H50)P(H50)}{P(S|H50)P(H50) + P(S|H60)P(H60)}$$

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

iii) S = 9 heads in ten attempts

$P(S|H50) = \binom{10}{9}(0.5)^9(0.5) = .0097$

$P(S|H55) = \binom{10}{9}(0.55)^9(0.45) = 0.0207$

$P(S|H60) = \binom{10}{9}(0.6)^9(0.4) = 0.0403$

$$P(Hi|S) = \frac{P(S|Hi)P(Hi)}{\sum_{Hi=50,55,60} P(S|Hi)P(Hi)}$$

$\Rightarrow P(H50|S) = \dfrac{(0.0097)(1/3)}{0.0707\,(1/3)} = \boxed{0.1372}$

$P(H55|S) = \dfrac{(0.0207)(1/3)}{0.0707\,(1/3)} = \boxed{0.2928}$

$P(H60|S) = \dfrac{(0.0403)(1/3)}{0.0707\,(1/3)} = \boxed{0.5700}$

b) $P(+|P_r) = 0.99$

$P(+|NP_r) = 0.1$

$P(NP_r) = 0.99$

$$P(P_r|+) = \frac{P(+|P_r)P(P_r)}{P(+|P_r)P(P_r) + P(+|NP_r)P(NP_r)}$$

$$= \frac{(0.99)(0.01)}{(0.99)(0.01) + (0.1)(0.99)}$$

$$= \frac{1}{11} = 0.\overline{09}$$

It means the test is not reliable when only in 9% of the time the women are pregnant given the test being positive.

c) $E(X) = \sum_x x P(X)$

$$\Rightarrow E(AX+b) = \sum_x (AX+b)P(x)$$

$$= A\sum x P(x) + b\underbrace{\sum P(x)}_{1}$$

$$= A\sum x P(x) + b$$

$$= \boxed{AE(X)+b}$$

Using scaling & additivity properties

$\Rightarrow$ linearity of expection with respect to one variable

7

2.d) $Cov(x) = E((x-E(x))(x-E(x))^T)$

$$Cov(Ax+b) = E\left((Ax+b-E(Ax+b))(Ax+b-E(Ax+b))^T\right)$$

$$= E\left((Ax+\cancel{b}-AE(x)-\cancel{b})(Ax+b-AE(x)-b)^T\right)$$

$$= E\left((Ax-AE(x))(Ax-AE(x))^T\right)$$

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

$$= E\left(A\underbrace{(x-E(x))(x-E(x))^T}_{Cov(x)}A^T\right)$$

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

6

3, a) $x \in R^n, y \in R^m, A \in R^{n \times m} \Rightarrow \nabla_x x^T A y$?

$f = \nabla_x x^T A y \Rightarrow \in R^{n \times 1} \mid f_i = \frac{\partial}{\partial x_i} x^T A y$

$$f = x^T A y = [x_1 \cdots x_n] \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ & \ddots & \\ a_{n1} & & a_{nm} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

$$f = \sum_{j=1}^{m} \left(\sum_{i=1}^{n} x_i a_{ij} \right) y_j$$

$$\frac{\partial f}{\partial x_k} = \frac{\partial \sum_{j=1}^{m} (x_k a_{kj}) y_j}{\partial x_k} = \frac{\partial}{\partial x_k} x_k \sum_{j=1}^{m} (a_{kj} y_j) = \sum_{j=1}^{m} a_{kj} y_j$$

6

$$\Rightarrow \nabla_x f = \begin{bmatrix} \sum_{j=1}^{m} a_{1j} y_j \\ \vdots \\ \sum_{j=1}^{m} a_{nj} y_j \end{bmatrix} = \boxed{Ay}$$

1 x n

b) $\nabla_y x^T A y$?

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

$$\nabla_y f = \begin{bmatrix} \sum_{i=1}^{n} x_i a_{i1} \\ \vdots \\ \vdots \\ \sum_{i=1}^{n} x_i a_{in} \end{bmatrix} = \boxed{A^T X}$$

c) $\nabla_A x^T A y$?

$$f = \sum_{j=1}^{m} \left(\sum_{i=1}^{n} x_i a_{ij} \right) y_j$$

$$\nabla_A f = \begin{bmatrix} \frac{\partial f}{\partial a_{11}} & \cdots & \frac{\partial f}{\partial a_{1m}} \\ & \ddots & \\ \frac{\partial f}{\partial a_{n1}} & & \frac{\partial f}{\partial a_{nxm}} \end{bmatrix} \Rightarrow \frac{\partial f}{\partial a_{kl}} = x_k y_l$$

$$= \begin{bmatrix} x_1 y_1 & \cdots & x_1 y_m \\ \vdots & & \vdots \\ x_n y_1 & & x_n y_m \end{bmatrix} = \boxed{x y^T}$$

d) $f = x^T A x + b^T x$

$$\nabla_x f = \nabla_x x^T A x + \nabla_x b^T x$$

$$\nabla_x b^T x = \begin{bmatrix} \frac{\partial \sum b_i x_i}{\partial x_1} \\ \vdots \\ \frac{\partial \sum b_i x_i}{\partial x_n} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \boxed{b}$$

$$x^T A x = \sum_{j=1}^{n} x_j \left(\sum_{i=1}^{n} x_i a_{ij} \right)$$

$\Rightarrow$ linearity of expecation with respect to one variable

$$\frac{\partial x^TAx}{\partial x_k} = \frac{\sum x_j(\sum x_i a_{ij})}{\partial x_k} = \sum x_i(a_{ik}) + \sum x_i(a_{kj})$$

$$= A^TX + AX$$

$$\Rightarrow \nabla_x(x^TAx + b^TX) = \boxed{A^TX + AX + b}$$

e) $f = tr(AB) \Rightarrow \nabla_A f$?

$A \in R^{m\times n}$, $B \in R^{n\times m} \Rightarrow C = AB$, $C \in R^{m\times m}$

$$c_{kl} = \sum_{i=1}^{n} a_{ki} b_{il}$$

$$tr(AB) = tr(C) = \sum_{j=1}^{m} c_{jj} = \sum_{j=1}^{m}\left(\sum_{i=1}^{n} a_{ji}\, b_{ij}\right)$$

$$\nabla_A f = \begin{bmatrix} \frac{\partial(AB)}{\partial a_{11}} & & \frac{\partial(AB)}{\partial a_{1n}} \\ \vdots & & \\ \frac{\partial(AB)}{\partial a_{mn}} & & \frac{\partial(AB)}{\partial a_{mn}} \end{bmatrix}$$

$$\Rightarrow \frac{\partial f}{\partial a_{ij}} = b_{ji} \qquad \Rightarrow \boxed{\nabla_A f = B^T}$$

4 $$\min_W \frac{1}{2}\sum_{i=1}^{n} \| y^i - Wx^i \|^2$$

$$\min_W \frac{1}{2}\sum_{i=1}^{n} (y^i - Wx^i)^T (y^i - Wx^i)$$

$$\min_W \frac{1}{2}(Y - XW^T)^T (Y - XW^T)$$

$$\min_W \frac{1}{2}(Y^T - WX^T)(Y - XW^T)$$

$$\min_W \underbrace{\frac{1}{2}(Y^TY - Y^TXW^T - WX^TY + WX^TXW^T)}_{L}$$

$$\nabla_W L = \frac{1}{2}(0 - 2Y^TX + 2WX^TX) = 0 \qquad WX^TX = X^TY$$

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

linear_regression

January 20, 2020

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 2020, Prof. J.C. Kao, TAs W. Feng, J. Lee, K. Liang, M. Kleinman, C. Zheng

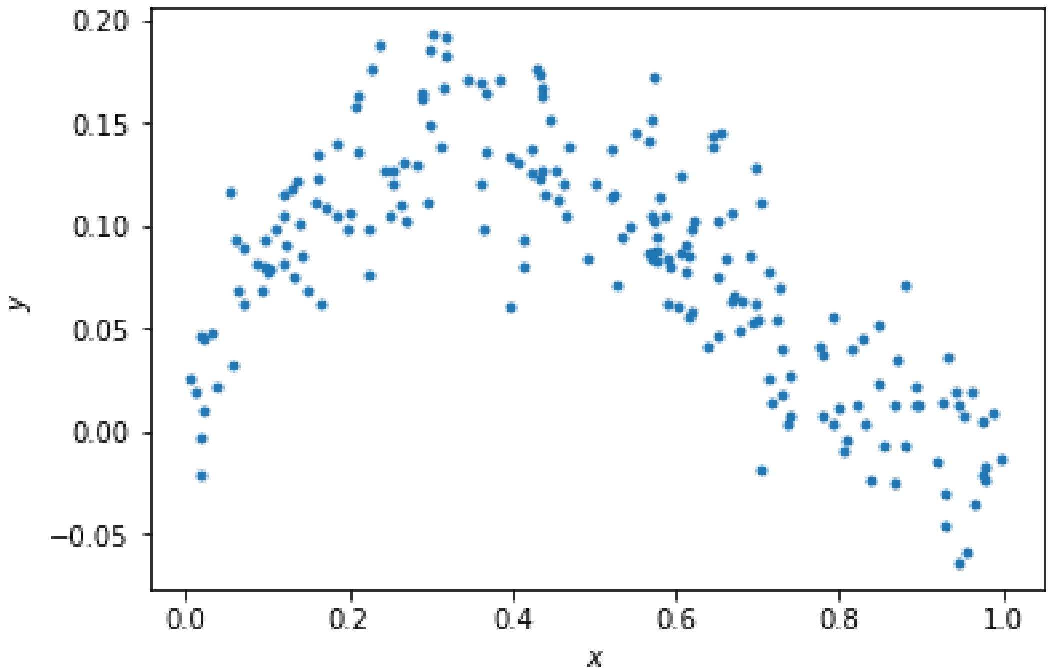
In [58]:
```
import numpy as np
import matplotlib.pyplot as plt

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

In [59]:
```
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[59]:
```
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) It is a uniform distribution from (including)0 to (excluding) 1. (200 data points)

(2) It is a normal (gussain) distribution with mean=0 and standard deviation = 0.33. (200 samples)

0.1.4 Fitting data to the model (5 points)

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

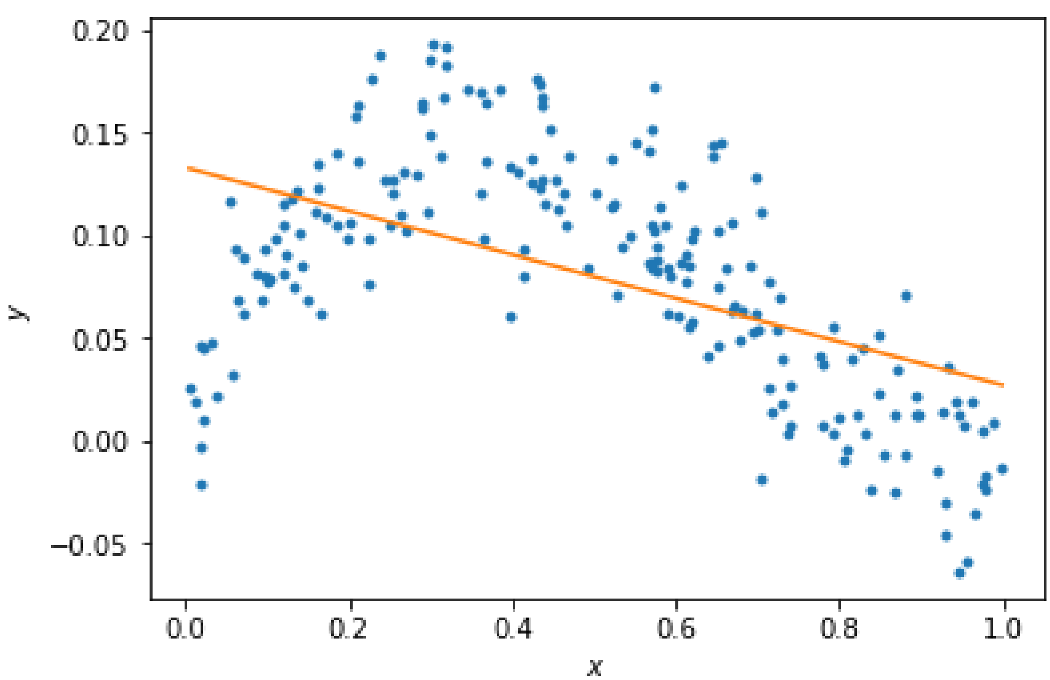
```
In [60]: # xhat = (x, 1)
         xhat = np.vstack((x, np.ones_like(x)))

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

```
    #theta = np.zeros(2) # please modify this line
    theta = np.linalg.inv((xhat).dot(xhat.T)).dot(xhat.dot(y))
    print(theta)
    print(theta.shape)

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

```
[-0.10599633  0.13315817]
(2,)
```

In [33]:
```
# 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[33]: [<matplotlib.lines.Line2D at 0x10dfa1ac8>]



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) The linear model underfits

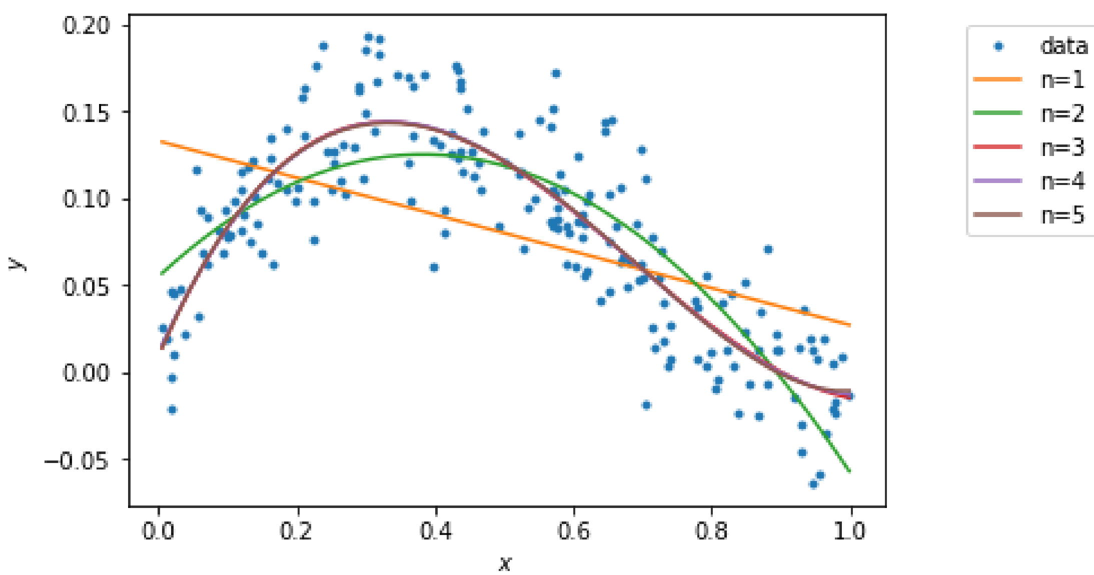
(2) To improve the model, we can add more parameters to theta to better fit the curve. This way we'll an equation of higher order and increase the dimensions of theta and X correspondingly.

0.1.7 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 [61]:
```
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
#   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
#   ... etc.

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



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

In [62]:
```
# 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 (10 points)

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

$L(\theta) = \frac{1}{2}\sum_j(\hat{y}_j - y_j)^2$

```
In [73]: 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 np.arange(N):
             training_errors.append( (1/2) * (np.linalg.norm(y - thetas[i].dot(xhats[i]))**2))

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

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

```
Training errors are:
 [80.861651845505861, 213.19192445058013, 3.1256971083304963, 1.187076519711066, 214.910218105
```

0.1.9 QUESTIONS

(1) Which polynomial model has the best training error?

(2) Why is this expected?

0.1.10 ANSWERS

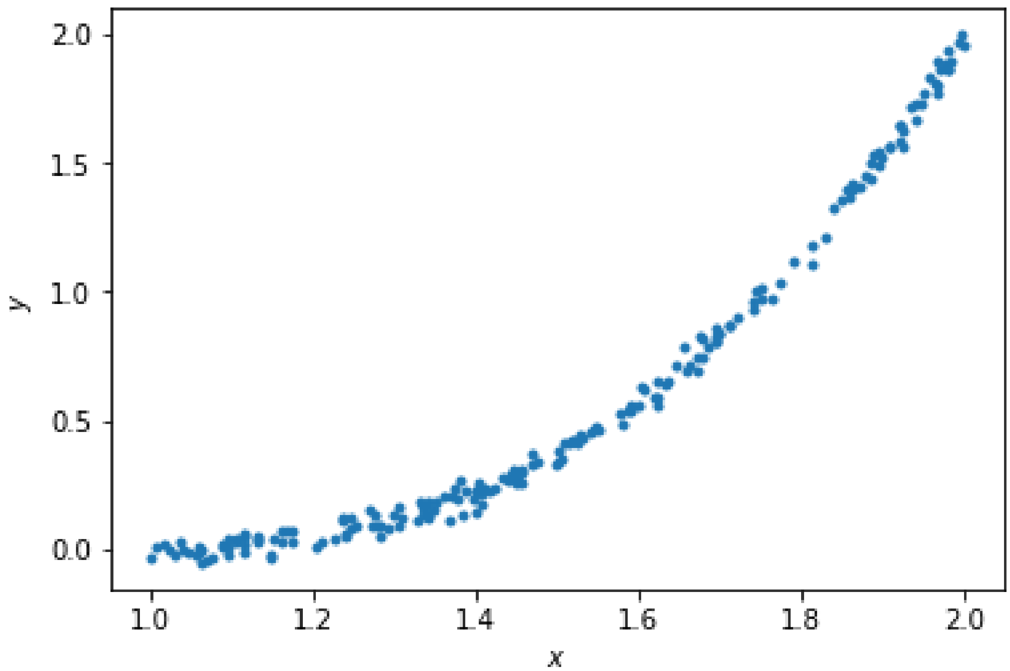
(1) The polynomial with 5 (4+1) degree has the best error. (The least training error)

(2) The higher the degree goes the model picks more parameters to fit the curve and according to the theorem it would at least perform as good as the one degree below itself if not better. Therefore it is expected to have lower training error as the degree goes up.

0.1.11 Generating new samples and testing error (5 points)

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

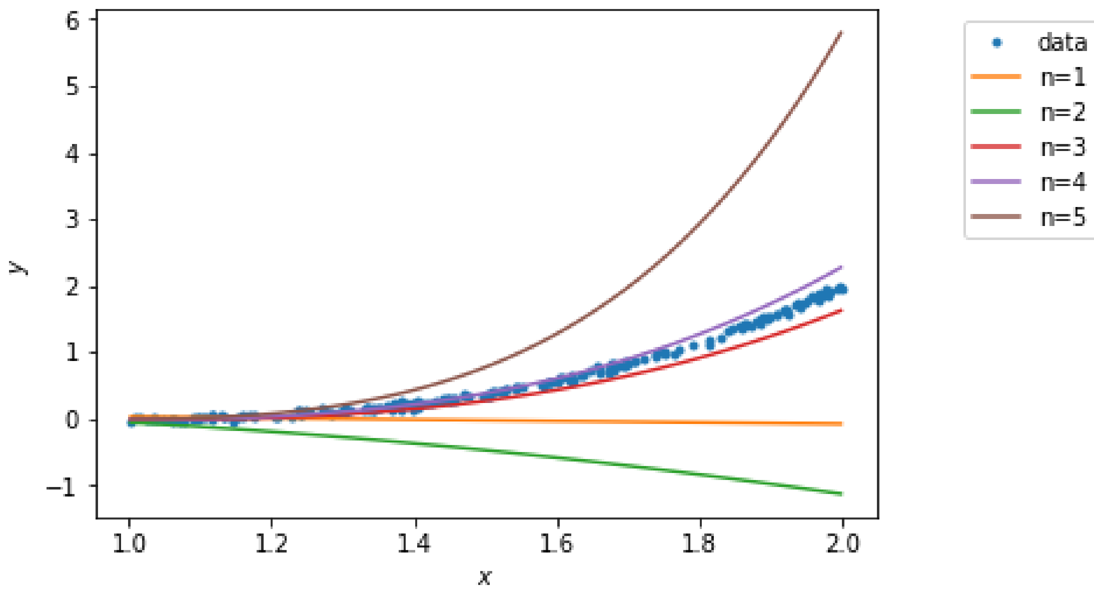
```
In [66]: x = np.random.uniform(low=1, high=2, size=(num_train,))
         y = x - 2*x**2 + x**3 + np.random.normal(loc=0, scale=0.03, size=(num_train,))
         f = plt.figure()
         ax = f.gca()
         ax.plot(x, y, '.')
         ax.set_xlabel('$x$')
         ax.set_ylabel('$y$')

Out[66]: Text(0,0.5,'$y$')
```



In [67]:
```
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 [68]:
```
# 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 [75]:
```
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 np.arange(N):
    testing_errors.append((1/2)* (np.linalg.norm(y - thetas[i].dot(xhats[i]))**2))

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

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

```
Testing errors are:
 [80.861651845505861, 213.19192445058013, 3.1256971083304963, 1.187076519711066, 214.910218105
```

0.1.12 QUESTIONS

(1) Which polynomial model has the best testing error?

(2) Why does the order-5 polynomial model not generalize well?

0.1.13 ANSWERS

(1) The polynomial with degree 4 does the best (the lowest) testing error

(2) This is a very good example of overfitting. The training error with n=5 tries to cover the maximum number of that training set which does not necessarily generalize to the best error rate for all test cases.