1. (25 points) **Linear algebra refresher.**

   (a) (12 points) Let $\mathbf{Q}$ be a real orthogonal matrix.

   i. (3 points) Show that $\mathbf{Q^T}$ and $\mathbf{Q^{-1}}$ are also orthogonal.

   ii. (3 points) Show that $\mathbf{Q}$ has eigenvalues with norm 1.

   iii. (3 points) Show that the determinant of $\mathbf{Q}$ is either +1 or -1.

   iv. (3 points) Show that $\mathbf{Q}$ defines a length preserving transformation.

i. If Q is a real orthogonal matrix,

$$Q^TQ = I$$

$$\Rightarrow (Q^T)^TQ^T = QQ^T = (Q^TQ)^T = I^T = I.$$

so Q^T is orthogonal.

$$QQ^T = I = QQ^{-1} \Rightarrow Q^TQQ^T = Q^TQQ^{-1} \Rightarrow Q^{-1} = Q^T.$$

so Q^{-1} is also orthogonal.

ii. Let λ_i be an eigenvalue of A with corresponding eigenvector v_i,

$$Av_i = \lambda_i v_i \Rightarrow A^TAv_i = \lambda_i A^Tv_i \Rightarrow v_i = \lambda_i A^Tv_i.$$

Taking the 2-norm of both sides: $\|v_i\|_2^2 = |\lambda_i|^2 \|A^Tv_i\|_2^2$

$$\Rightarrow \|A^Tv_i\|_2^2 = (A^Tv_i)^T(A^Tv_i) = v_i^TAA^Tv_i = \|v_i\|_2^2$$

Hence, we have $\|v_i\|_2^2 = |\lambda_i|^2 \|v_i\|_2^2$

$\therefore |\lambda_i| = 1$, Hence, all eigenvalues of A have norm 1.

iii. $\det(I) = \det(Q^TQ) = \det(Q^T)\det(Q) = [\det(Q)]^2 = 1$

so $\det(Q) = \pm 1$

iv. For $y = Qx$ orthogonal transformation,

$$|y| = |Qx| = \sqrt{(Qx)^2} = \sqrt{(Qx)^TQx} = \sqrt{x^TQ^TQx} = |x|$$

so Q defines a length preserving transformation.

(b) (8 points) Let $\mathbf{A}$ be a matrix.

i. (4 points) What is the relationship between the singular vectors of $\mathbf{A}$ and the eigenvectors of $\mathbf{A}\mathbf{A}^T$? What about $\mathbf{A}^T\mathbf{A}$?

ii. (4 points) What is the relationship between the singular values of $\mathbf{A}$ and the eigenvalues of $\mathbf{A}\mathbf{A}^T$? What about $\mathbf{A}^T\mathbf{A}$?

i. Let u, v be singular vectors of A. σ is singular value.

A can be expressed as $A = U\Sigma V^T$

$$AA^T = U\Sigma V^T \cdot (U\Sigma V^T)^T = U\Sigma V^T V \Sigma U^T$$

Because $V^T V = I_{s\times s}$, V is an orthogonal matrix

$$AA^T = U\Sigma^2 U^T$$

so U is the eigenvector of AA^T.

$$A^T A = (U\Sigma V^T)^T \cdot U\Sigma V^T = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T$$

so V is the eigenvector of $A^T A$.

ii. Σ^2 is the eigenvalue of AA^T and $A^T A$.

and singular values of A make up Σ.

So singular values of A are square roots of eigenvalues from AA^T to $A^T A$.

i.e. $\lambda_i(AA^T) = \lambda_i(A^T A) = \sigma_i^2(A)$

(c) (5 points) True or False. Partial credit on an incorrect solution may be awarded if you justify your answer.

i. Every linear operator in an n-dimensional vector space has n distinct eigenvalues. (n×n matrix)

ii. A non-zero sum of two eigenvectors of a matrix $\mathbf{A}$ is an eigenvector.

iii. If a matrix $\mathbf{A}$ has the positive semidefinite property, i.e., $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ for all $\mathbf{x}$, then its eigenvalues must be non-negative.

iv. The rank of a matrix can exceed the number of distinct non-zero eigenvalues.

v. A non-zero sum of two eigenvectors of a matrix $\mathbf{A}$ corresponding to the same eigenvalue λ is always an eigenvector.

i. False.

ii. False.

Let x_1, x_2 are two eigenvectors of A with corresponding λ_1, λ_2.

$Ax_1 = \lambda_1 x_1$, $Ax_2 = \lambda_2 x_2$

$A(x_1+x_2) = Ax_1 + Ax_2 = \lambda_1 x_1 + \lambda_2 x_2$

because $\lambda_1 \neq \lambda_2$, $A(x_1+x_2) = \lambda_1 x_1 + \lambda_2 x_2 \neq \lambda(x_1+x_2)$.

iii. True.

Let λ_i be an eigenvalue of A with corresponding v_i.

$Av_i = \lambda_i v_i$

since $x^TAx \geq 0$ for all x, $v_i^TAv_i = v_i^T\lambda_i v_i \geq 0$.

$v_i^T\lambda_i v_i = \lambda_i \|v_i\|_2^2 \geq 0$

since $\|v_i\|_2^2 > 0$, so $\lambda_i \geq 0$

∴ All eigenvalues of A are non-negative.

iv. True.

Define $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, rank$(A) = 2$, eigenvalue $= 1$.

v. True.

Define x_1, x_2 are two eigenvectors of A with corresponding same λ

$Ax_1 = \lambda x_1$, $Ax_2 = \lambda x_2$

$\Rightarrow A(x_1+x_2) = Ax_1 + Ax_2 = \lambda x_1 + \lambda x_2 = \lambda(x_1+x_2)$.

2. (25 points) **Probability refresher.**

(a) (5 points) A jar of coins is equally populated with two types of coins. One is type "H50" and comes up heads with probability 0.5. Another is type "H60" and comes up heads with probability 0.6.

i. (1 points) You take one coin from the jar and flip it. It lands tails. What is the posterior probability that this is an H50 coin?

ii. (2 points) You put the coin back, take another, and flip it 4 times. It lands T, H, H, H. How likely is the coin to be type H50?

iii. (2 points) A new jar is now equally populated with coins of type H50, H55, and H60 (with probabilities of coming up heads 0.5, 0.55, and 0.6 respectively. You take one coin and flip it 10 times. It lands heads 9 times. How likely is the coin to be of each possible type?

i. Let A = tails, B = this is an H50 coin.

$\Pr(A) = \frac{1}{2} \times 0.5 + \frac{1}{2} \times (1-0.6) = 0.25 + 0.2 = 0.45$

$\Pr(B) = \frac{1}{2}$. $\Pr(A|B) = 0.5$.

$\Pr(B|A) = \frac{\Pr(B)\Pr(A|B)}{\Pr(A)} = \frac{\frac{1}{2} \times 0.5}{0.45} = \frac{5}{9} \approx 0.5556$

ii. Let C = Lands T,H,H,H. B = this is an H50 coin.

$\Pr(C) = \frac{1}{2} \times (\frac{1}{2})^4 + \frac{1}{2} \times \frac{2}{5} \times (\frac{3}{5})^3 = \frac{1}{32} + \frac{3^3}{5^4} = \frac{1}{32} + \frac{27}{625} = \frac{1489}{20000} \approx 0.0744$

$\Pr(B) = \frac{\frac{1}{2} \times (\frac{1}{2})^4}{\Pr(C)} = \frac{\frac{1}{32}}{\frac{1489}{20000}} = \frac{625}{1489} \approx 0.4197$

iii. Let A = 10 times flip results,

B = type H50, C = type H55, D = type H60.

$\Pr(A) = \frac{1}{3} \times (\frac{1}{2})^{10} + \frac{1}{3} \times 0.55^9 \times 0.45 + \frac{1}{3} \times 0.6^9 \times 0.4 = 0.0024$

$\Pr(B) = \frac{\frac{1}{3} \times (\frac{1}{2})^{10}}{\Pr(A)} = 0.1379$

$\Pr(C) = \frac{\frac{1}{3} \times 0.55^9 \times 0.45}{\Pr(A)} = 0.2927$

$\Pr(D) = \frac{\frac{1}{3} \times 0.6^9 \times 0.4}{\Pr(A)} = 0.5694$

(b) (5 points) Students at UCLA are from these disciplines: 15% Science, 21% Healthcare, 24% Liberal Arts, and 40% Engineering. (Each student belongs to a unique discipline.) The students attend a lecture and give feedback. Suppose 90% of the Science students liked the lecture, 18% of the Healthcare students liked it, none of the Liberal Arts students liked it, and 10% of the Engineering students liked it. If a student is randomly chosen, and the student liked the lecture, what is the conditional probability that the student is from Science?

Let A = this random student liked this lecture,

B = the student is from science.

Thus $Pr(A) = 0.15 \times 0.9 + 0.21 \times 0.18 + 0.4 \times 0.1 = 0.2128$.

$Pr(A|B) = 0.9$. $Pr(B) = 0.15$.

so $Pr(B|A) = \frac{Pr(B)Pr(A|B)}{Pr(A)} = \frac{0.15 \times 0.9}{0.2128} \approx 0.634$

(c) (5 points) Consider a pregnancy test with the following statistics.

- If the woman is pregnant, the test returns "positive" (or 1, indicating the woman is pregnant) 99% of the time.
- If the woman is not pregnant, the test returns "positive" 10% of the time.
- At any given point in time, 99% of the female population is not pregnant.

What is the probability that a woman is pregnant given she received a positive test? The answer should make intuitive sense; give an explanation of the result that you find.

The probability that a woman received a positive test:

$Pr(A) = 0.99 \times 0.1 + 0.01 \times 0.99 = 0.1089$

The probability that the woman is truly pregnant given above:

$$Pr(B|A) = \frac{Pr(B)Pr(A|B)}{Pr(A)} = \frac{0.01 \times 0.99}{0.1089} = 0.0909 = 9.09\%.$$

The pregancy test accuray is not high for following reasons:

① the truly pregnant women are few enough, only around 1%.

② the wrong test rate for the non-pregant women is really high, around 10%.

(d) (5 points) Let $x_1, x_2, \ldots, x_n$ be identically distributed random variables. A random vector, $\mathbf{x}$, is defined as

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

What is $\mathbb{E}(\mathbf{A}\mathbf{x} + \mathbf{b})$ in terms of $\mathbb{E}(\mathbf{x})$, given that $\mathbf{A}$ and $\mathbf{b}$ are deterministic?

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

(e) (5 points) Let

$$\mathbf{cov}(\mathbf{x}) = \mathbb{E}\left((\mathbf{x} - \mathbb{E}\mathbf{x})(\mathbf{x} - \mathbb{E}\mathbf{x})^T\right)$$

What is $\mathbf{cov}(\mathbf{A}\mathbf{x} + \mathbf{b})$ in terms of $\mathbf{cov}(\mathbf{x})$, given that $\mathbf{A}$ and $\mathbf{b}$ are deterministic?

$$\begin{aligned} Cov(Ax+b) &= E\left((Ax+b-AE(x)-b)(Ax+b-AE(x)-b)^T\right) \\ &= E\left[A(x-E(x))\left[A(x-E(x))\right]^T\right] \\ &= E\left[A(x-E(x))(x-E(x))^T A^T\right] \\ &= A\,cov(x)\,A^T. \end{aligned}$$

3. (10 points) **Multivariate derivatives.**

(a) (1 points) Let $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{n \times m}$. What is $\nabla_{\mathbf{x}} \mathbf{x}^T \mathbf{A} \mathbf{y}$?

(b) (1 points) Let $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{n \times m}$. What is $\nabla_{\mathbf{y}} \mathbf{x}^T \mathbf{A} \mathbf{y}$?

(c) (1 points) Let $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{n \times m}$. What is $\nabla_{\mathbf{A}} \mathbf{x}^T \mathbf{A} \mathbf{y}$?

(d) (1 points) Let $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{n \times n}$, and let $f = \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x}$. What is $\nabla_{\mathbf{x}} f$?

(e) (1 points) Let $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times n}$ and $f = \mathrm{tr}(\mathbf{A}\mathbf{B})$. What is $\nabla_{\mathbf{A}} f$?

(f) (2 points) Let $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times n}$ and $f = \mathrm{tr}(\mathbf{B}\mathbf{A} + \mathbf{A}^T \mathbf{B} + \mathbf{A}^2 \mathbf{B})$. What is $\nabla_{\mathbf{A}} f$?

(g) (3 points) Let $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times n}$ and $f = \|\mathbf{A} + \lambda \mathbf{B}\|_F^2$. What is $\nabla_{\mathbf{A}} f$?

(a) $\nabla_x x^T A y = Ay$

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

(c) $\nabla_A x^T A y = \nabla_A A^T y^T x = y^T x.$

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

(e) $\nabla_A f = B^T$

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

(g) $\nabla_A f = Tr\left[(A+\lambda B)(A+\lambda B)^T\right]$
$= Tr\left[AA^T + A\lambda B^T + \lambda BA^T + \lambda^2 BB^T\right]$
$= 2A + 2\lambda B$

4. (10 points) **Deriving least-squares with matrix derivatives.**
In least-squares, we seek to estimate some multivariate output $\mathbf{y}$ via the model

$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{x}$$

In the training set we're given paired data examples $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ from $i = 1, \ldots, n$. Least-squares is the following quadratic optimization problem:

$$\min_{\mathbf{W}} \quad \frac{1}{2}\sum_{i=1}^{n} \left\| \mathbf{y}^{(i)} - \mathbf{W}\mathbf{x}^{(i)} \right\|^2$$

Derive the optimal $\mathbf{W}$.

Where $\mathbf{W}$ is a matrix, and for each example in the training set, both $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ $\forall i = 1, \ldots, n$ are vectors.
Hint: you may find the following derivatives useful:

$$\begin{aligned} \frac{\partial \text{tr}(\mathbf{WA})}{\partial \mathbf{W}} &= \mathbf{A}^T \\ \frac{\partial \text{tr}(\mathbf{WAW}^T)}{\partial \mathbf{W}} &= \mathbf{WA}^T + \mathbf{WA} \end{aligned}$$

Define loss function of W as below:

$$L(W) = \frac{1}{2}\sum_{i=1}^{n} \left\| y^{(i)} - Wx^{(i)} \right\|^2$$

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

$$= \frac{1}{2}(Y^TY - Y^TXW - W^TX^TY + W^TX^TXW)$$

$$= \frac{1}{2}(Y^TY - 2W^TX^TY + W^TX^TXW)$$

$$\frac{\partial L(W)}{\partial W} = \frac{1}{2}\left[-2X^TY + (XX^T + XX^T)W\right] = 0$$

$\therefore$ the optimal W is as follows:

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

5. (10 points) **Regularized least squares**

In lecture, we worked through the following least squares problem

$$\arg\min_{\theta} \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \theta^T \hat{\mathbf{x}}^{(i)})^2$$

However, the least squares has a tendency to overfit the training data. One common technique used to address the overfitting problem is regularization. In this problem, we work through one of the regularization techniques namely ridge regularization which is also known as the regularized least squares problem. In the regularized least squares we solve the following optimization problem

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

where λ is a tunable regularization parameter. From the above cost function it can be observed that we are seeking least squares solution with a smaller 2-norm. Derive the solution to the regularized least squares problem, i.e Find θ^*.

Let's define loss function as follows,

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

$$= \frac{1}{2}(Y - X\theta)^T(Y - X\theta) + \frac{\lambda}{2}\|\theta\|_2^2$$

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

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

$$\nabla_\theta L(\theta) = -X^TY + (X^TX + \lambda I)\theta$$

$$\therefore \theta^* = (X^TX + \lambda I)^{-1}X^TY.$$

6. (20 points) **Linear regression.**
Complete the Jupyter notebook `linear_regression.ipynb`. Print out the Jupyter notebook as a PDF and submit it to Gradescope.

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

In [13]:

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

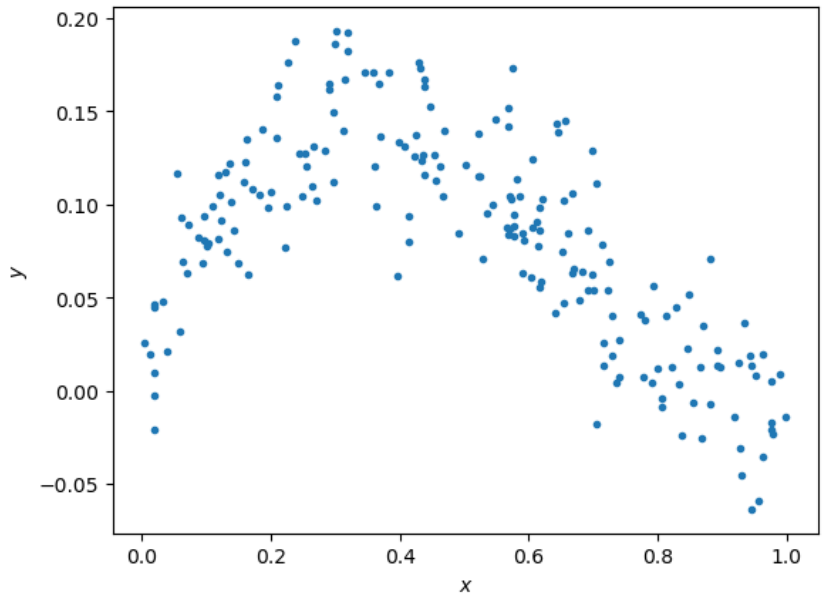
In [2]:

```
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) With the function of numpy.random.uniform, it generates the unifrom distribution of x with lower bound = 0, upper bound = 1 and size of 200.

(2) With the function of numpy.random.normal, it generates the normal distribution of ϵ with mean = 0, standard deviation = 0.03 and size of 200.

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]


#theta = np.zeros(2) # please modify this line
theta = np.linalg.inv(xhat.dot(xhat.T)).dot(xhat.dot(y))
# ================= #
# END YOUR CODE HERE #
# ================= #
```

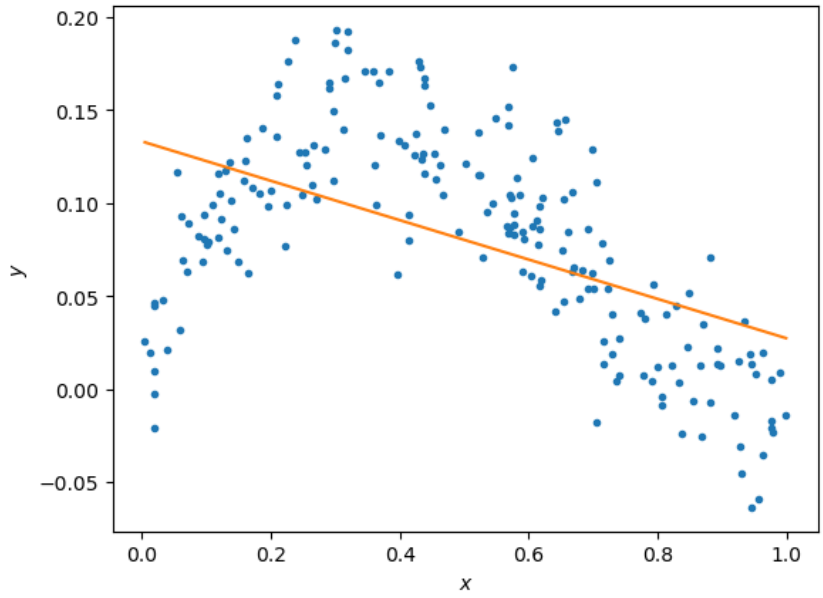
In [4]:

```
# 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 0x2114818e490>]
```



QUESTIONS

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

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

ANSWERS

(1) This linear model underfits the data.

(2) We can introduce polynomial models because the fitting will be better along with the order increases.

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.

In [5]:

```
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 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)))
pass

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

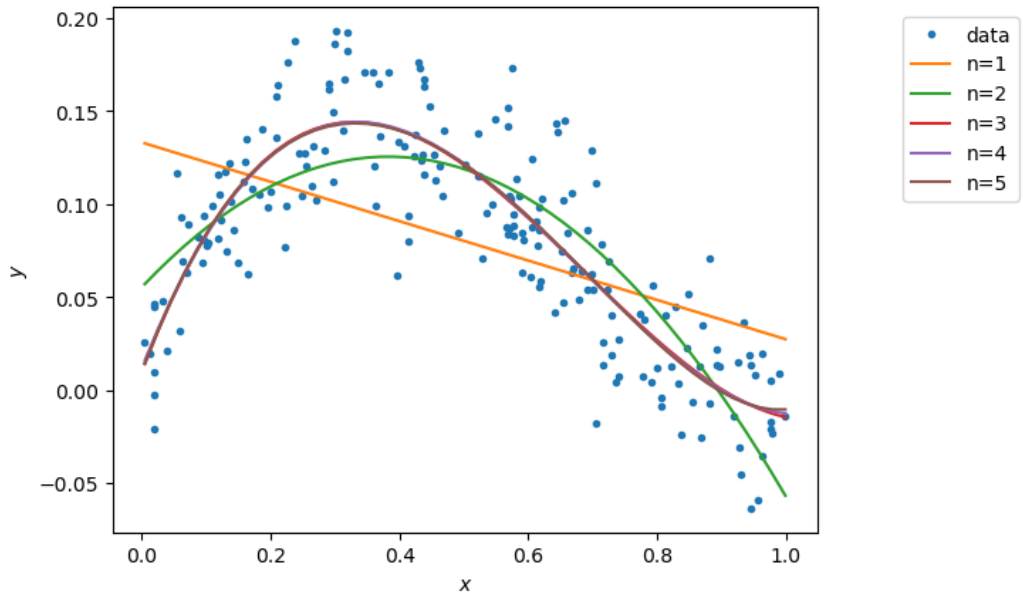
In [6]:

```
# 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.

In [7]:

```
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):
    xhat = np.vstack((x,np.ones_like(x))) if i == 0 else np.vstack((x**(i+1), xhat))
    training_errors.append(np.sqrt(np.sum((y - thetas[i].dot(xhat))**2) / len(y)))
pass

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

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

```
Training errors are:
 [0.04878484486357111, 0.03305287008607351, 0.02858251878527393, 0.028575083088762804, 0.02856830270689054]
```

QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

ANSWERS

(1) For the order of 5, the polynomial model fits best since the training error is the minimum among five models.

(2) Because with higher order, the polynomial models can fit more data points from the training data sets.

Generating new samples and testing error (5 points)

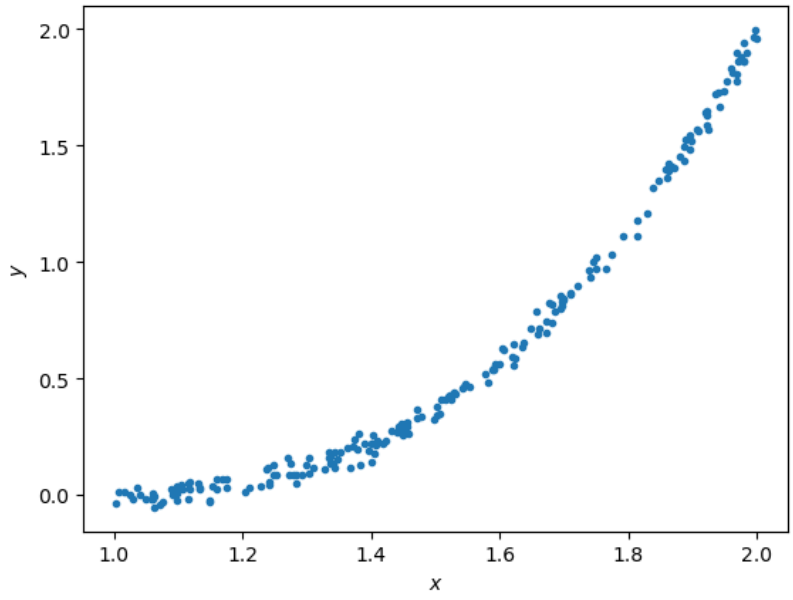
Here, we'll now generate new samples and calculate testing error of polynomial models of orders 1 to 5.

In [8]:

```
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]:

```
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)
```

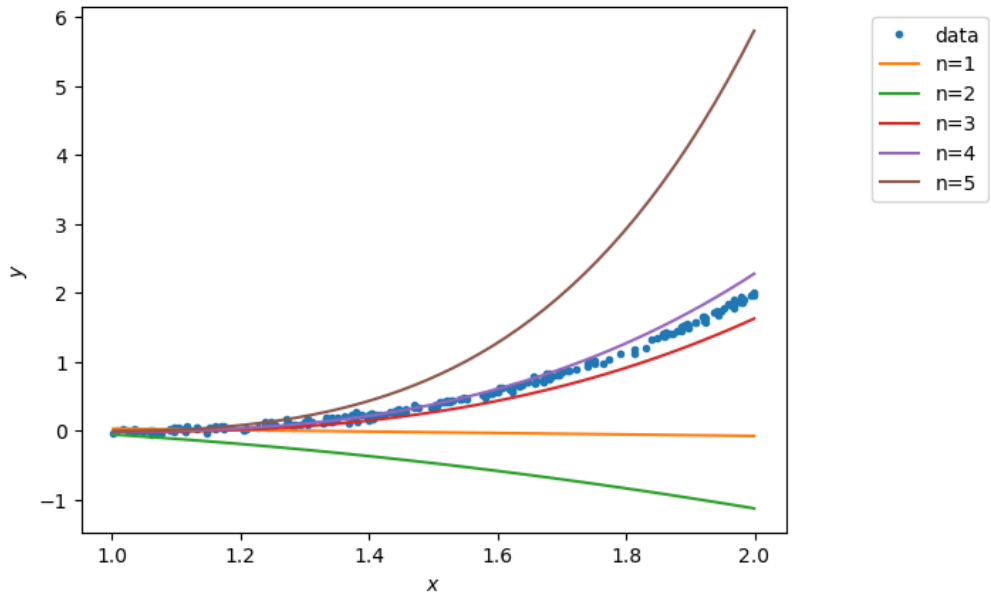
In [10]:

```
# 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]:

```
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):
    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)))
pass

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

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

```
Testing errors are:
 [0.8992310706681899, 1.4601093262169795, 0.17679641139668179, 0.10895304121239116, 1.4659816449941137]
```

QUESTIONS

(1) What polynomial has the best testing error?

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

ANSWERS

(1) For the order of 4, the polynomial model has the best testing error.

(2) Becuase the polynomial model of order 5 overfits the training data sets, which results in a high testing error when generalizing.