1.

a.

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

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

$\left| \frac{1}{\sqrt{2}}\begin{bmatrix} \lambda & 1 \\ 1 & -\lambda \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right| = 0$

$\left| \begin{bmatrix} \frac{1}{\sqrt{2}}\lambda - 1 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-\lambda}{\sqrt{2}} - 1 \end{bmatrix} \right| = 0$

$\left(\frac{\lambda}{\sqrt{2}} - 1\right)\left(\frac{-\lambda}{\sqrt{2}} - 1\right) - \frac{1}{2}$

$\frac{-\lambda^2}{2} + \frac{\lambda}{\sqrt{2}} - \frac{\lambda}{\sqrt{2}} + 1 - \frac{1}{2} = 0$

$\frac{-\lambda^2}{2} + \frac{1}{2} = 0$

$\lambda^2 = 1$

$\lambda_1 = 1$

$\lambda_2 = -1$

$A v = \lambda v$

$Av - \lambda v = 0$

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

$\lambda = 1$ $\quad \begin{array}{c} \begin{bmatrix} \frac{1}{\sqrt{2}} - 1 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} - 1 \end{bmatrix} \\ a \qquad\quad b \end{array} \begin{bmatrix} a \\ b \end{bmatrix} = 0$

$\left[\begin{array}{cc|c} \frac{1-\sqrt{2}}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{-1-\sqrt{2}}{\sqrt{2}} & 0 \end{array}\right]$

$\left[\begin{array}{cc|c} \frac{1-\sqrt{2}}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{(1-\sqrt{2})}{2} & -\frac{1}{\sqrt{2}} & 0 \end{array}\right]$ $\quad -(1-\sqrt{2})(1+\sqrt{2})$ $\; -(1-2)$ $\; +1$

$\left[\begin{array}{cc|c} \frac{1-\sqrt{2}}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 \end{array}\right]$

$b = K_1$

$\frac{1-\sqrt{2}}{\sqrt{2}} a + \frac{1}{\sqrt{2}} K_1 = 0$

$a = \frac{-1}{\sqrt{2}} K_1 \cdot \frac{\sqrt{2}}{1-\sqrt{2}}$

$a = \frac{-K_1}{1-\sqrt{2}}$

$e_1 = K_1 \begin{bmatrix} 1+\sqrt{2} \\ 1 \end{bmatrix}$

$\lambda = -1$

$$\left[\begin{array}{cc|c} \frac{-1-\sqrt{2}}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1-\sqrt{2}}{\sqrt{2}} & 0 \end{array}\right]$$

$$\left[\begin{array}{cc|c} \frac{-1-\sqrt{2}}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{+1+\sqrt{2}}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \end{array}\right] \qquad (1+\sqrt{2})(1-\sqrt{2}) = 1-2=-1$$

$$\left[\begin{array}{cc|c} \frac{-1-\sqrt{2}}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 \end{array}\right]$$

$b = k_2$

$$\frac{-(1+\sqrt{2})}{\sqrt{2}} a + \frac{1}{\sqrt{2}} k_2 = 0$$

$$\frac{+(1+\sqrt{2})}{\sqrt{2}} a = \frac{1}{\sqrt{2}} k_2$$

$$a = \frac{k_2}{1+\sqrt{2}} = -k_2 \frac{(1-\sqrt{2})}{1}$$

$$e_2 = k_2 \begin{bmatrix} 1-\sqrt{2} \\ 1 \end{bmatrix}$$

$\lambda_1 = 1$ $\qquad e_1 = \begin{bmatrix} 1+\sqrt{2} \\ 1 \end{bmatrix} \cdot \frac{1}{\sqrt{1+(1+\sqrt{2})^2}}$

$\lambda_2 = -1$ $\qquad e_2 = \begin{bmatrix} 1-\sqrt{2} \\ 1 \end{bmatrix} \cdot \frac{1}{\sqrt{1+(1-\sqrt{2})^2}}$

The eigenvalues are same magnitude (magnitude of 1) and the eigenvectors are orthogonal

$(1+\sqrt{2})(1-\sqrt{2}) + (1)(1) = (1-2)+1 = 0$

ii.

$\lambda_1 = 1 \quad |\lambda_1| = 1$

$\lambda_2 = -1 \quad |\lambda_2| = 1$

$Av = \lambda v$

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

$||Av||^2 = ||\lambda v||^2$

$(Av)^T(Av) = (\lambda v)^T(\lambda v)$

$v^T A^T A v = v^T \lambda^T \lambda v$

$v^T I v = v^T ||\lambda||^2 v$

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

$||v||^2 = ||\lambda||^2 ||v||^2$

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

$\boxed{||\lambda|| = 1}$

iii. $e_1^T e_2 = 0$ if orthogonal and distinct eig value

$$[1+\sqrt{2} \quad 1]\begin{bmatrix} 1-\sqrt{2} \\ 1 \end{bmatrix} = (1+\sqrt{2})(1-\sqrt{2}) + 1 = -1 + 1 = 0$$

$AA^T = I$

$$U = \begin{bmatrix} \uparrow & \uparrow \\ \frac{e_1}{||e_1||} & \frac{e_2}{||e_2||} \\ \downarrow & \downarrow \end{bmatrix} \quad \Sigma = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

$(U\Sigma U^T)(U\Sigma U^T)^T = I$

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

$UIU^T = I$

$UU^T = I$

$$\begin{bmatrix} \frac{e_1^T e_1}{||e_1||^2} & \frac{e_2^T e_1}{||e_2|| ||e_1||} \\ \frac{e_1^T e_2}{||e_1|| ||e_2||} & \frac{e_2^T e_1}{||e_2|| ||e_1||} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\frac{e_2^T e_1}{||e_2|| ||e_1||} = 0$$

$\boxed{e_2^T e_1 = 0}$

iv. A vector x under the transformation A is only rotated and reflected as the magnitude of x will not change due to the norm of the eigenvalues being 1.

b. $A = U\Sigma V^T$

i. The columns of U are called the left singular vectors of A (and are orthonormal eigenvectors of AA^T).

The colums of V are called the right singular vectors of A (and are orthonormal eigenvectors of A^TA).

proof:

$A \in R^{m\times n}$ $\quad U \in R^{m\times 1}$ $\quad V \in R^{n\times 1}$ $\quad eig(\underbrace{A^TA}_{\Sigma^T \Sigma}) = eig(\underbrace{AA^T}_{\Sigma\Sigma})$ $\quad \Sigma^T = \Sigma$

$A^TAv_i = \sigma_i^2 v_i$

assume v_i is a unit norm eigenvector of A^TA

$v_i^T A^T A v_i = \sigma_i^2 v_i^T v_i$

$(Av_i)^T (Av_i) = \sigma_i^2 ||v_i||^2$

$||Av_i||^2 = \sigma_i^2$

$||Av_i|| = \sigma_i$

$A^TAv_i = \sigma_i^2 v_i$

$A\underbrace{A^TAv_i} = \sigma_i^2 \underbrace{Av_i}$

Av_i is an eigenvector of AA^T

$u_i = \frac{Av_i}{||Av_i||} = \frac{Av_i}{\sigma_i}$

$Av_i = \sigma_i u_i$

$AA^Tu_i = \sigma_i^2 u_i$

assume u_i is unit norm eigenvector of AA^T,

$u_i^T AA^T u_i = \sigma_i^2 u_i^T u$

$(A^Tu_i)(A^Tu_i) = \sigma_i^2 ||u_i||^2$

$||A^Tu_i||^2 = \sigma_i^2$

$AA^Tu_i = \sigma_i^2 u_i$

$A^T \underbrace{A A^T u_i} = \underbrace{A^T \sigma_i^2 u_i}$

A^Tu_i is eigenvector of A^TA

$v_i = \frac{A^Tu_i}{||A^Tu_i||} = \frac{A^Tu_i}{\sigma_i}$

$v_i\sigma_i = A^Tu_i$

$u_i^T A = \sigma_i v_i^T$

$u_i^T A v_i = \sigma_i v_i^T v_i$

$u_i^T A v_i = \sigma_i ||v_i||^2 = \sigma_i$

$u_i^T A v_i = \sigma_i \rightarrow U^TAV = \Sigma \rightarrow \boxed{A = \underbrace{U}_{m\times m} \underbrace{\Sigma}_{m\times n} \underbrace{V^T}_{n\times n}}$

$U^TU = I$

$VV^T = I$

$\Sigma_{ii} = \sigma_i$

$\Sigma_{ij} = 0$ for $i \neq j$

ii.

$$A^TA = (U\Sigma V^T)^T(U\Sigma V^T) \qquad AA^T = (U\Sigma V^T)(U\Sigma V^T)^T$$
$$= V\Sigma^T U^T U \Sigma V^T \qquad = U\Sigma V^T V \Sigma^T U$$
$$= V\Sigma^T\Sigma V^T \qquad = U\Sigma\Sigma^T U$$

$$\Sigma^T\Sigma = \Sigma^T\Sigma = \begin{bmatrix} \sigma_1^2 & & & & \\ & \sigma_2^2 & & & \\ & & \ddots & & \\ & & & \sigma_r^2 & \\ & & & & \ddots\ 0 \end{bmatrix}$$

hence $\boxed{\lambda_i(AA^T) = \lambda_i(A^TA) = \sigma_i^2(A)}$

c.

i. False (it has at most n-distinct eigenvalue)

ii. False

$$A(\underbrace{c_1e_1 + c_2e_2}_{v}) = c_1Ae_1 + c_2Ae_2$$
$$= c_1\lambda_1 e_1 + c_2\lambda_2 e_2$$

iii. True.

iv. False

v. True

$$A(\underline{c_1e_1 + c_2e_2}) = c_1\lambda e_1 + c_2\lambda e_2$$
$$= \lambda(\underline{c_1e_1 + c_2e_2})$$

2.

a.

$P(H50) = .5 = P(H60)$

$P(H \mid H50) = .5 \quad P(H \mid H60) = .6$

i.

$$P(H50 \mid T) = \frac{P(T \mid H50)\,P(H50)}{P(T)} = \frac{P(T \mid H50)P(H50)}{P(T \mid H50)P(H50) + P(T \mid H60)P(H60)}$$

$$= \frac{(.5)(.5)}{(.5)(.5) + (.4)(.5)} = \boxed{0.5556}$$

ii.

$P(\{T,H,H,H\} \mid H50) \rightarrow$ conditionally ind $= (.5)^4 \quad P((T,H,H,H) \mid H60) = (.4)(.6)^3$

$$P(H50 \mid (T,H,H,H)) = \frac{P((T,H,H,H) \mid H50)\,P(H50)}{P((T,H,H,H) \mid H50)\,P(H50) + P((T,H,H,H) \mid H60)\,P(H60)}$$

$$= \frac{(.5)^4(.5)}{(.5)^4(.5) + (.4)(.6)^3(.5)}$$

$$\boxed{= 0.4197}$$

iii.

$$P(H50 \mid \text{the flips}) = \frac{(.5^9)(.5)(\frac{1}{3})}{(.5^9)(.5)(\frac{1}{3}) + (.6)^9(.4)(\frac{1}{3}) + (.55)^9(.45)(\frac{1}{3})} = \boxed{0.1379 = P(H50 \mid \text{the flips})}$$

$$P(H55 \mid \text{the flips}) = \frac{(.55^9)(.45)(\frac{1}{3})}{\text{same den}} = \boxed{0.2927 = P(H55 \mid \text{the flips})}$$

$$\boxed{P(H60 \mid \text{the flips}) = 0.5694}$$

b.

$p(+|\text{preg}) = .99$

$p(+|\text{not preg}) = .10$

$p(\text{not preg}) = .99$

$$p(\text{preg}|+) = \frac{p(+|\text{preg})p(\text{preg})}{p(+|\text{preg})p(\text{preg}) + p(+|\text{not preg})p(\text{not preg})}$$

$$= \frac{(.99)(.01)}{(.99)(.01) + (.10)(.99)} = \boxed{0.0909}$$

This makes sense because the prior for a woman being pregnant any time is very low (.01), but once the woman gets a positive test we receive new information and we may update the prior ~~with~~ to a posterior (shes pregnant given we know she tested positive). Now its shown ~~the~~ shes much more likely to be pregnant than she was before the test.

c. $\boxed{E(Ax+b) = \int (Ax+b)\,dX = A\int x\,dX + \underbrace{\int b\,dX}_{\text{integ of constant}} = A\,E[X] + b = A\mu_x + b}$

d. $\text{cov}(Ax+b) = E[(Ax+b-A\mu_x-b)(Ax+b-A\mu_x-b)^T]$

$E[(Ax-A\mu_x)(Ax-A\mu_x)^T]$

$E[A(x-\mu_x)(A(x-\mu_x))^T]$

$A\,E[(x-\mu_x)(x-\mu_x)^T]A^T$

$\boxed{\text{cov}(Ax+b) = A\,\text{cov}(x)A^T}$

3. $A = \begin{bmatrix} - a_1 - \\ - a_2 - \\ \vdots \\ - a_n - \end{bmatrix}$

a.

$$x^T \begin{bmatrix} a_1 y \\ a_2 y \\ \vdots \\ a_n y \end{bmatrix} = \sum_{i=1}^{n} x_i a_i y$$

$$\boxed{\nabla_x x^T A y = \begin{bmatrix} a_1 y \\ a_2 y \\ \vdots \\ a_n y \end{bmatrix} = Ay}$$

b. $A = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_m \\ | & | & & | \end{bmatrix}$

$$[x a_1 \;\; x a_2 \cdots x a_m] y = \sum_{i=1}^{m} (x a_i) y_i \qquad \boxed{\nabla_y x^T A y = A^T x}$$

c. $\sum_{i=1} \sum_{j=1} x_i y_j a_{ij}$ $\qquad \boxed{\nabla_A x^T A y = x y^T}$

d. $\boxed{\nabla_x f = Ax + A^T x + b = (A + A^T) x + b}$

use notes in class to get gradient of $x^T A x$

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

e. $f = tr(\underset{n \times m}{A} \underset{m \times n}{B})$

$$tr\left(\begin{bmatrix} - a_1 - \\ - a_2 - \\ \vdots \\ - a_n - \end{bmatrix} \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_n \\ | & | & & | \end{bmatrix} \right) = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$$

$$\sum_{i=1}^{n} a_i b_i = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} b_{ji}$$

$$\boxed{\nabla_A f = B^T}$$

4.

$$Z = y - Wx$$ vector

$$f = z^T z$$ scalar

Tr(scalar) = scalar

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

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

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

$$\frac{1}{2} \sum_{i=1}^n (y^{(i)T} y^{(i)} - 2 \mathrm{Tr}(y^{(i)T} W x^{(i)}) + \mathrm{Tr}(x^{(i)T} W^T W x^{(i)}))$$

$$f = \frac{1}{2} \sum_{i=1}^n (y^{(i)T} y^{(i)} - 2\mathrm{Tr}(W x^{(i)} y^{(i)T}) + \mathrm{Tr}(W^T x^{(i)} x^{(i)T} W))$$

$$\frac{d}{dW} f = 0$$

$$\frac{1}{2} \sum_{i=1}^n (-2 y^{(i)} x^{(i)T} + W(x^{(i)} x^{(i)T}) + W(x^{(i)} x^{(i)T})) = 0$$

$$\frac{1}{2} \sum_{i=1}^n (-2y^{(i)} x^{(i)T} + 2W(x^{(i)} x^{(i)T})) = 0$$

$$\sum_{i=1}^n \left[W(x^{(i)} x^{(i)T}) - y^{(i)} x^{(i)T} \right] = 0$$

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

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

can make as matrices as well ... $\boxed{W = (Y^T X)(X^T X)^{-1}}$ $X = \begin{bmatrix} - x_1^T - \\ - x_2^T - \\ \vdots \\ - x_n^T - \end{bmatrix}$

$$Y = \begin{bmatrix} - y_1^T - \\ - y_2^T - \\ \vdots \\ - y_n^T - \end{bmatrix}$$

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 2021, Prof. J.C. Kao, TAs: N. Evirgen, A. Ghosh, S. Mathur, T. Monsoor, G. Zhao

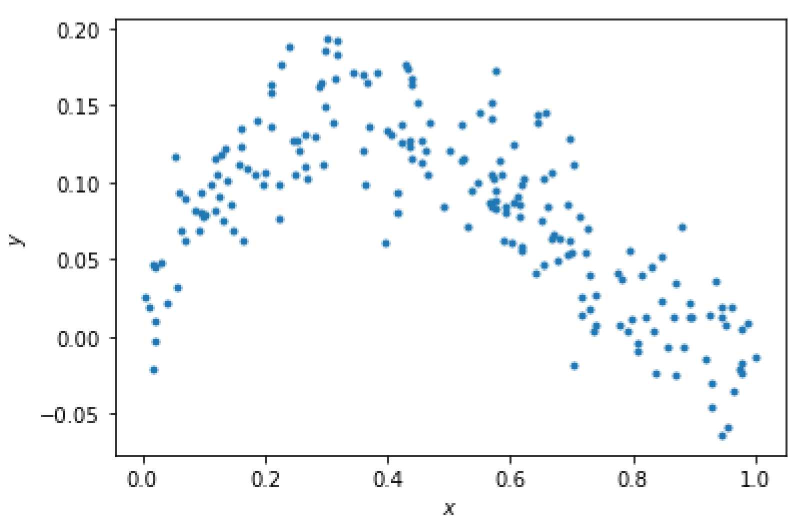
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

#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 a uniform distribution U(a=0,b=1)

(2) The distribution of the additive noise ϵ is a normal distribution $N(\mu = 0, \sigma = 0.03)$

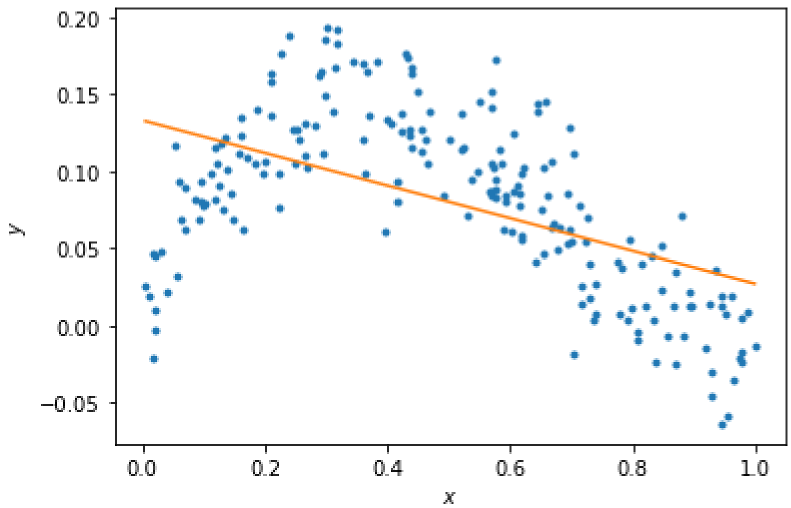
Fitting data to the model (5 points)

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]

theta = np.matmul(np.linalg.inv(np.matmul(xhat,xhat.T)),np.matmul(xhat,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 0x171a449ad30>]



QUESTIONS

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

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

ANSWERS

(1) The line underfits the data.

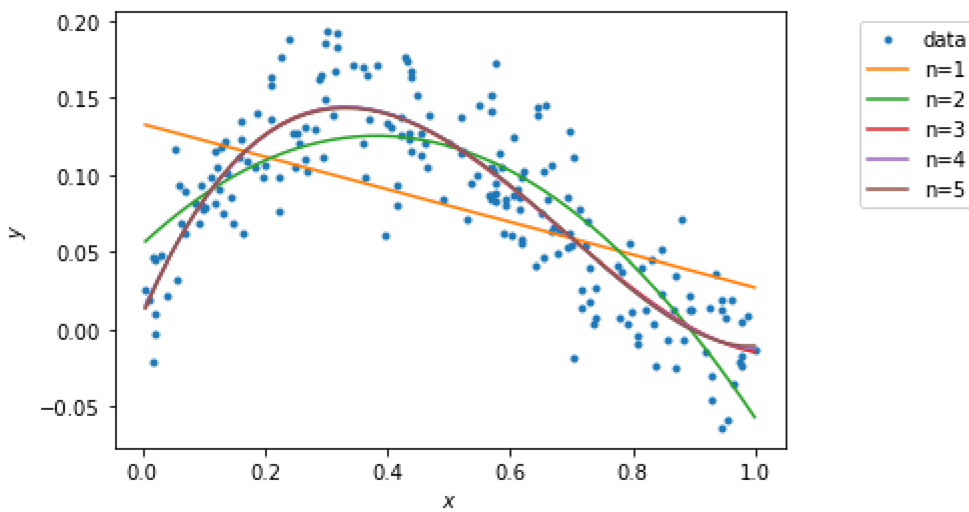
(2) To improve the model fit, more complexity to the model should be added, i.e. more model parameters.

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]:
```
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.

xhat = np.vstack((x**5,x**4,x**3,x**2,x, np.ones_like(x)))
xhats = [xhat[i:,:] for i in range(N-1,-1,-1)]

theta_derivation = lambda inp: np.matmul(np.linalg.inv(np.matmul(inp,inp.T)),n
p.matmul(inp,y))

thetas = [theta_derivation(xhats[i]) for i in range(0,N)]

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 (10 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 ord
er i+1.
f = lambda theta,x,y: np.mean((np.matmul(x.T,theta)-y)**2)

training_errors = [f(thetas[i],xhats[i],y) for i in range(0,N)]
pass

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

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

```
Training errors are:
 [0.0023799610883627007, 0.001092492220926853, 0.0008169603801105374, 0.00081
65353735296982, 0.0008161479195525297]
```

QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

ANSWERS

(1) The polynomial with the best training error is polynomial of order 5.

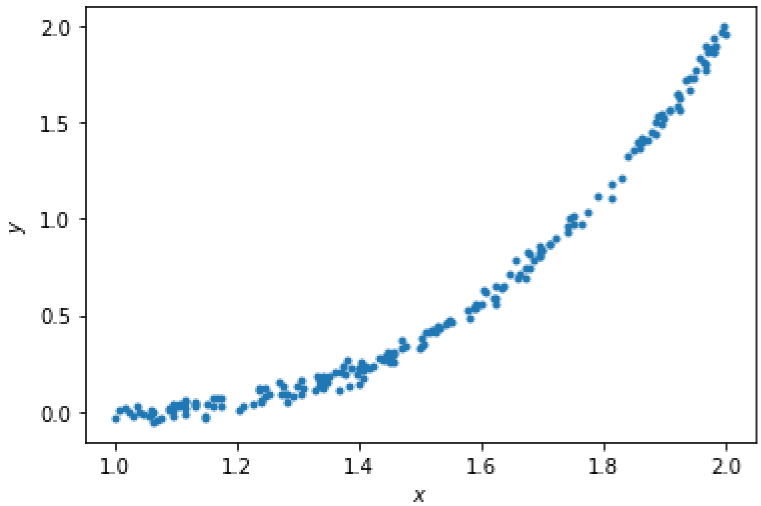
(2) This is expected because as the model complexity increases, the error should be reduced on the training data. If the polynomial order were to approach the number of data points trained on, the training error would approach zero.

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('$y$')
```

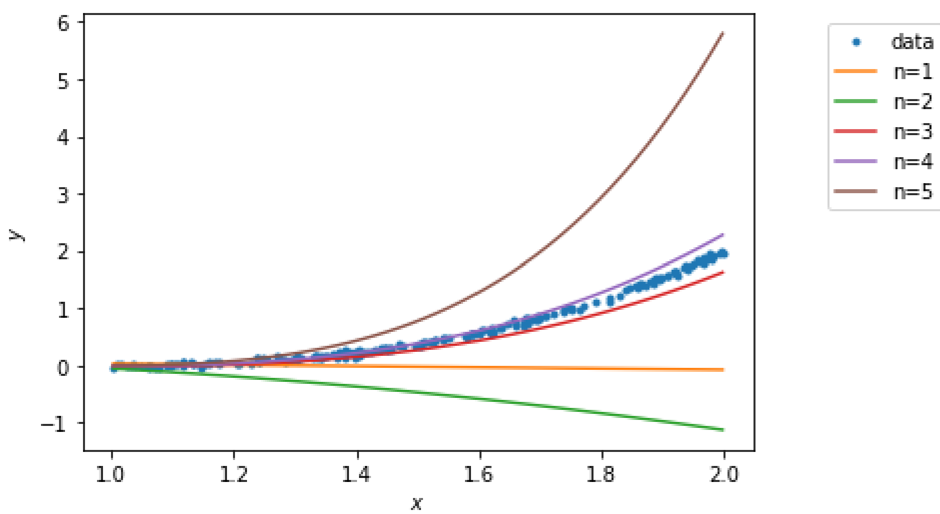
Out[8]:
```
Text(0, 0.5, '$y$')
```



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.
f = lambda theta,x,y: np.mean( (np.matmul(x.T,theta)-y)**2)

testing_errors = [f(thetas[i],xhats[i],y) for i in range(0,N)]
pass

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

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

```
Testing errors are:
 [0.8086165184550587, 2.1319192445057893, 0.03125697108276392, 0.011870765189
474703, 2.1491021817652625]
```

QUESTIONS

(1) What polynomial has the best testing error?

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

ANSWERS

(1) A polynomial of order 4.

(2) Polynomial of order 5 does not generalize well because we begin to overfit the training data. Therefore, the model variance is extremely high, even though the model bias is low causing a not desired mean squared error.