Redmond Xiq

due 1/20/20

1. $AA^T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} a & c \\ b & d \end{bmatrix}$ $\quad a = -\frac{\sqrt{2}}{2} \quad b = \sqrt{1-\frac{1}{2}} = \frac{\sqrt{2}}{2}$

$= \begin{bmatrix} a^2+b^2=1 & ac+bd=0 \\ ac+bd=0 & c^2+d^2=1 \end{bmatrix}$ $\quad -\frac{\sqrt{2}}{2}c + \frac{\sqrt{2}}{2}d = 0 \quad c=d$

let $c=d=\frac{\sqrt{2}}{2}$

$\det(A - \lambda I)\vec{x} = 0$ to find eigen values

$A - \lambda I = \begin{bmatrix} -\frac{\sqrt{2}}{2}-\lambda & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}-\lambda \end{bmatrix} \Rightarrow \left(-\frac{\sqrt{2}}{2}-\lambda\right)\left(\frac{\sqrt{2}}{2}-\lambda\right) - \frac{1}{2}$

$= -\frac{1}{2} + \frac{\lambda\sqrt{2}}{2} - \frac{\sqrt{2}}{2}\lambda + \lambda^2 - \frac{1}{2} \Rightarrow \lambda^2 = 1 \quad \lambda = \pm 1$

let $\lambda = 1$ $\quad (A - \lambda I)\vec{x} = 0 \Rightarrow \begin{bmatrix} -\frac{\sqrt{2}}{2}-1 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}-1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\left(-\frac{\sqrt{2}}{2}-1\right)x_1 + \frac{\sqrt{2}}{2}x_2 = 0$

$\left(\frac{\sqrt{2}}{2}\right)x_1 + \left(\frac{\sqrt{2}}{2}-1\right)x_2 = 0$

$\left(+\frac{\sqrt{2}}{2}+1\right)x_1 = \frac{\sqrt{2}}{2}x_2$ $\quad \left(1 + \frac{\sqrt{2}\sqrt{2}}{\sqrt{2}\sqrt{2}}\right)x_1 = x_2$

$(1+\sqrt{2})x_1 = x_2$

$x_1 = \left(\frac{\sqrt{2}}{\frac{\sqrt{2}}{2}}+1\right)x_2 \Rightarrow x_1 = (-1+\sqrt{2})x_2$

$(1+\sqrt{2}) = x_2$ $\quad$ let $x_1 = 1$ $\quad$ we normalize

$k\sqrt{1^2 + (1+\sqrt{2})^2} = 1$ $\quad k\sqrt{4+2\sqrt{2}+2} = 1$

$\vec{x}_1 = \begin{bmatrix} \frac{1}{\sqrt{4+2\sqrt{2}}} \\ \frac{1+\sqrt{2}}{\sqrt{4+2\sqrt{2}}} \end{bmatrix}$ $\quad k = \frac{1}{\sqrt{4+2\sqrt{2}}}$

$\approx \begin{bmatrix} 0.38268 \\ 0.92388 \end{bmatrix}$

for $\lambda = -1$

$\begin{bmatrix} -\frac{\sqrt{2}}{2}+1 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}+1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\quad \left(-\frac{\sqrt{2}}{2}+1\right)x_1 + \frac{\sqrt{2}}{2}x_2 = 0$

$\frac{\sqrt{2}}{2}x_1 + \left(\frac{\sqrt{2}}{2}+1\right)x_2 = 0$

$x_2 = (-1+\sqrt{2})$ $\quad$ set $x_1 = 1$ $\quad (1-\sqrt{2})x_1 = x_2$

$k\sqrt{1^2 + (1-\sqrt{2})^2} = 1$ $\quad k\sqrt{4-2\sqrt{2}} = 1$

$k = \frac{1}{\sqrt{4-2\sqrt{2}}}$

$\lambda = -1$ corresponds with eigenvector

$$\vec{x}_2 = \begin{bmatrix} \frac{-1}{\sqrt{4-2\sqrt{2}}} \\ \left(\frac{-1+\sqrt{2}}{\sqrt{4-2\sqrt{2}}}\right) \end{bmatrix} \approx \begin{bmatrix} -0.9239 \\ 0.38268 \end{bmatrix}$$

- eigenvalues magnitudes are the same

and $\vec{x}_1^T \vec{x}_2 = 0$

(b)

$A\vec{x} = \lambda\vec{x}$ know $AA^T = I$

$(A\vec{x})^T A\vec{x} = (\lambda\vec{x})^T(\lambda\vec{x}) = x^T\lambda^T\lambda\vec{x}$

$= |\lambda|^2\vec{x}^T\vec{x} = \vec{x}^T\underbrace{A^TA}_{I}\vec{x} = \vec{x}^T\vec{x} = 1$

$|\lambda|^2 = 1$ so norm eigenvalue is 1,

iii) let λ_1, λ_2 be eigen values of $\vec{x}_1, \vec{x}_2$.

$A\vec{x}_1 = \lambda_1 \vec{x}_1 \qquad A\vec{x}_2 = \lambda_2 \vec{x}_2$

$(A\vec{x}_1)^T (A\vec{x}_2) = (\lambda_1 \vec{x}_1)^T (\lambda_2 \vec{x}_2)$

$\vec{x}_1^T A^T A \vec{x}_2 = \lambda_1 \lambda_2 \vec{x}_1^T x_2$

$\vec{x}_1^T \vec{x}_2 = \lambda_1 \lambda_2 x_1^T x_2 \Rightarrow (\lambda_1 \lambda_2 - 1) x_1^T x_2 = 0$

$(\lambda_1 \lambda_2 = 1)$ so $\vec{x}_1^T \vec{x}_2$ must $= 0$ to be true

iv) the length or magnitude of $\vec{x}$ will not change, it can be reflected or ~~rotated~~ rotated though.

1) b) Let A be a matrix. know by SVD

$$A = U\Sigma V^T \qquad V^TV = I$$

So $AA^T = U\Sigma^T V^T V \Sigma U^T$

$= U\Sigma^T\Sigma U^T$ so the left singular vectors of A is the eigenvectors of AA^T

$$A^TA = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T\Sigma V^T$$

right singular vetors of A is the eigenvectors of A^TA

ii) singular vectors of A is the ~~eigenvectors~~ square roots of eigenvalues of AA^T and A^TA

(c) i) False

I_2 diagonal matrix, means its columns are the eigenvalues

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ only has 1 distinct. eigenvalue 1.

ii) False. in example 1)a)

$$\begin{bmatrix} 0.38268 \\ 0.9239 \end{bmatrix} + \begin{bmatrix} -0.9239 \\ 0.38268 \end{bmatrix} = \begin{bmatrix} -0.54122 \\ 1.30658 \end{bmatrix}$$

which is not an eigen vector of it in example 1.

1.d) True let $\vec{x}$ be an eigen vector of A.
Then $A\vec{x} = \lambda\vec{x}$ multiple $\vec{x}^T$ each
side $\vec{x}^T A\vec{x} = \lambda\vec{x}^T\vec{x}$ we assume A semi definite
is $\vec{x}^T A\vec{x} \geq 0$ thus $\lambda\vec{x}^T\vec{x} \geq 0 \Rightarrow \lambda \sum_{i=1}^{n} x_i^2 \geq 0 \Rightarrow \lambda \geq 0$

iv) True. in general eigenvalues have multiplicity, and the a matrix could have full rank. $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ full rank but only 1 distinct eigenvalues

v) True

2) a) i) $P(HSO) = \frac{1}{2}$ $P(HGO) = \frac{1}{2}$ $P(H|HSO) = \frac{1}{2}$

$P(H|HGO) = \frac{3}{5}$

$P(HSO|T) = ?$

$$P(HSO|T) = \frac{P(T \cap HSO)}{P(T)} = \frac{P(T|HSO)\,P(HSO)}{P(T|HSO)P(HSO) + P(T|HGO)P(HGO)}$$

$$= \frac{\frac{1}{2}\frac{1}{2}}{\left(\frac{1}{2}\frac{1}{2} + \frac{2}{5}\frac{1}{2}\right)} = \frac{\frac{1}{4}}{\frac{1}{4} + \frac{2}{10}} = \frac{\frac{1}{4}}{\frac{18}{40}} = \frac{40}{4}\frac{1}{18} = \frac{10}{18}$$

$$= \frac{5}{9}$$

(8) ii) $P(H50 \mid THHH) = \dfrac{P(THHHH \cap H50)}{P(THHH)}$

$= \dfrac{P(THHHH \mid H50)\,P(H50)}{P(THHH \mid H50)\,P(H50) + P(THHH \mid H60)\,P(H60)}$

$= \dfrac{\frac{1}{2^5}}{\frac{1}{2^5} + (.6)^3(.4)\frac{1}{2}}$ using a calculator $\dfrac{.03125}{.03125 + .0432}$

$\approx .419745$ or $\dfrac{625}{1489}$

iii) denote $\star = P(HHHHHHHHHT \mid \text{any type}, \text{any})$

$P(A \mid \text{any}) = P(A) = P(H^9T)$

$A = H^9T$

$P(H50) = P(H55) = P(H60) = \frac{1}{3}$

$$P(H50 \mid A) = \frac{P(A \mid H50)P(H50)}{P(A)} = \frac{P(A \mid H50)P(H50)}{P(A \mid H50)P(H50) + P(A \mid H55)P(H55) + P(A \mid H60)P(H60)}$$

$$= \frac{\frac{1}{3}\frac{1}{2^{10}}\frac{1}{3}}{\frac{1}{2^{10}}\frac{1}{3} + \frac{1}{3}(.45)(.55)^9 + \frac{1}{3}(.4)(.6)^9} = \frac{0.000325208}{0.00236}$$

≈ 0.137914 note denominator is the same

$$P(H55 \mid A) = \frac{P(A \mid H55)P(H55)}{0.00236} = \frac{\frac{1}{3}(.45)(.55)^9}{0.00236}$$

≈ 0.242716

$$P(H60|A) = \frac{P(A|H60)P(H60)}{0.00236019} = \frac{\frac{1}{3} \cdot .6^9 \cdot .4}{.00236}$$

$$\approx 0.5693569$$

b) denote P_0 for pregnancy and P for positive

$P(P|P_1) = 99\%$

$P(P|P_0^c) = 10\%$

$P(P_1^c) = 99\%$

$$P(P_1|P) = \frac{P(P \cap P_1)}{P(P)}$$

$$= \frac{P(P_1)P(P|P_1)}{P(P|P_1)P(P_1) + P(P|P_1^c)P(P_1^c)}$$

$$= \frac{.99(.01)}{.99(.01) + .01(.99)} = .5$$

This means there are more false detections than these are pregnant women since the test fails on 10% of all women.

c) Since A, b is deterministic we have

$$E[A\vec{x} + b] = E[Ax] + b = E\left[\sum_{j=1}^{n} A_{ij} x_j\right] + b$$

$$= \sum_{j=1}^{n} A_{ij} E(x)_j + b = AE[x] + b$$

d $\quad Cov(X) = E[(X-EX)(X-EX)^T]$ what is $Cov(Ax+b)$

$Cov(Ax+b) = E[Ax+b - (AE[X]+b)\,(Ax+b - (AE[X]+b))^T]$

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

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

$A\,Cov(X)\,A^T$

3) a) $\nabla_x x^T A y$ $\nabla_x \cdot Ax = A^T$ know this

$\nabla_x (A^T x) y = \nabla_x A^T x^T y = A y$ this

6)

$$\nabla_y \vec{x} A y = \nabla_y \left(y^T A^T \vec{x}\right)^{\textcircled{1}} = \nabla_y (Ay)\vec{x}$$

$$= A^T x$$

c.) $\nabla_A \vec{x}^T A \vec{y} = \begin{pmatrix} \frac{\partial \vec{x}^T A \vec{y}}{\partial a_{11}} & \frac{\partial \vec{x}^T A \vec{y}}{\partial a_{12}} & \cdots & \frac{\partial \vec{x}^T A \vec{y}}{\partial a_{1m}} \\ \frac{\partial \vec{x}^T A \vec{y}}{\partial a_{21}} & & & \\ \vdots & & & \frac{\partial \vec{x}^T A \vec{y}}{\partial a_{nm}} \end{pmatrix}$

$= \vec{x}\vec{y}^T$

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

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

e) $\nabla_A tr(AB) =$ From the hint in 4

$= \nabla_A tr\left(\begin{bmatrix} a_1^T b_1 & a_1^T b_2 & \cdots & a_1^T b_n \\ \vdots & \ddots & & \\ a_m^T b_1 & \cdots & & a_m^T b_n \end{bmatrix}\right) = B^T$

4)

$$\min_W \frac{1}{2}\sum_{i=1}^{n} \|y_i - W x_i\|^2 = \min_W \frac{1}{2}\sum_{i=1}^{n} (y_i - Wx_i)^T (y_i - Wx_i)$$

$$= \min_W \frac{1}{2}\sum_{i=1}^{n} \left(y_i^T y_i - 2 y_i^T W x_i + x_i^T W^T W x_i \right)$$

$$= \min_W \frac{1}{2}\sum_{i=1}^{n} \left(-2 y_i^T W x_i + x_i^T W^T W x_i \right)$$

$$= \min_W \frac{1}{2}\sum_{i=1}^{n} \left(-2 \operatorname{tr}(W x_i y_i^T) + \operatorname{tr}(W x_i x_i^T W^T) \right)$$

$$= \min_W -\mathrm{tr}\left(W \sum x_i y_i^T\right) + \frac{1}{2}\mathrm{tr}\left(W \sum (x_i x_i^T) W^T\right)$$

$$= \min_W -\mathrm{tr}(WXY^T) + \frac{1}{2}\mathrm{tr}(WXX^TW^T)$$

$$\Rightarrow -\frac{\partial}{\partial W}\mathrm{tr}(WXY^T) + \frac{1}{2}\frac{\partial}{\partial W}\mathrm{tr}(WXX^TW^T)$$

$$= -YX^T + \frac{1}{2}(WXX^T + WXX^T) = -YX^T + WXX^T = 0$$

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

linear_regression

January 19, 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 [1]:
```
import numpy as np
import matplotlib.pyplot as plt

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

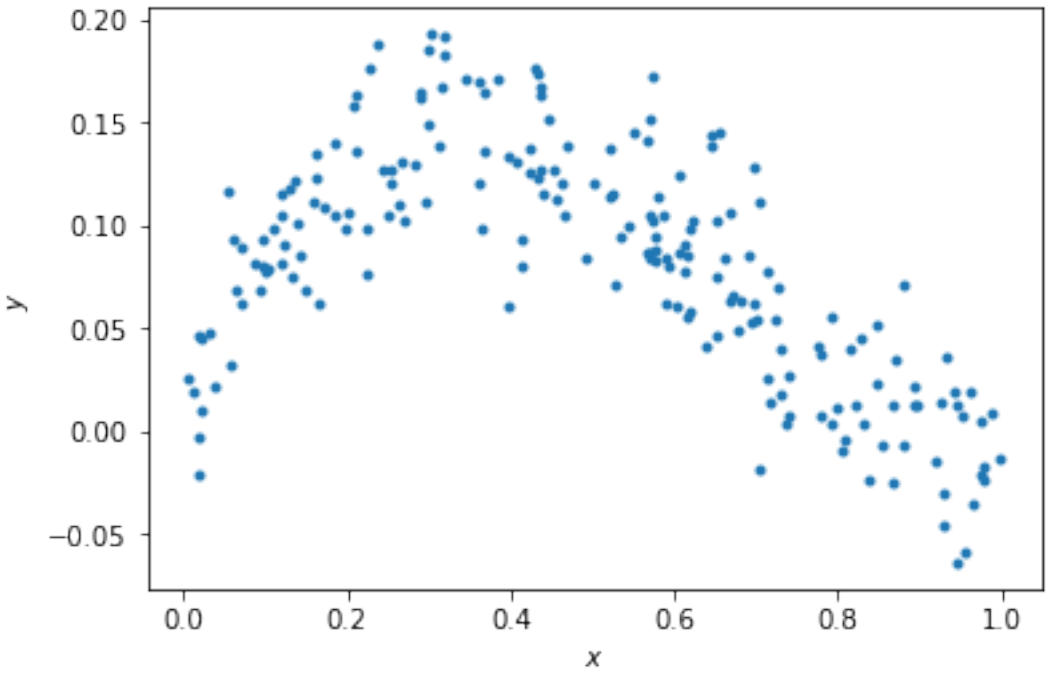
0.1.1 Data generation

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 [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, '$y$')
```



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) The generating distribution of x is a uniform(0,1) distribution

(2) The distribution of the additive noise of epsilon is normal(0,.03) .03 being stddev

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 [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]
```

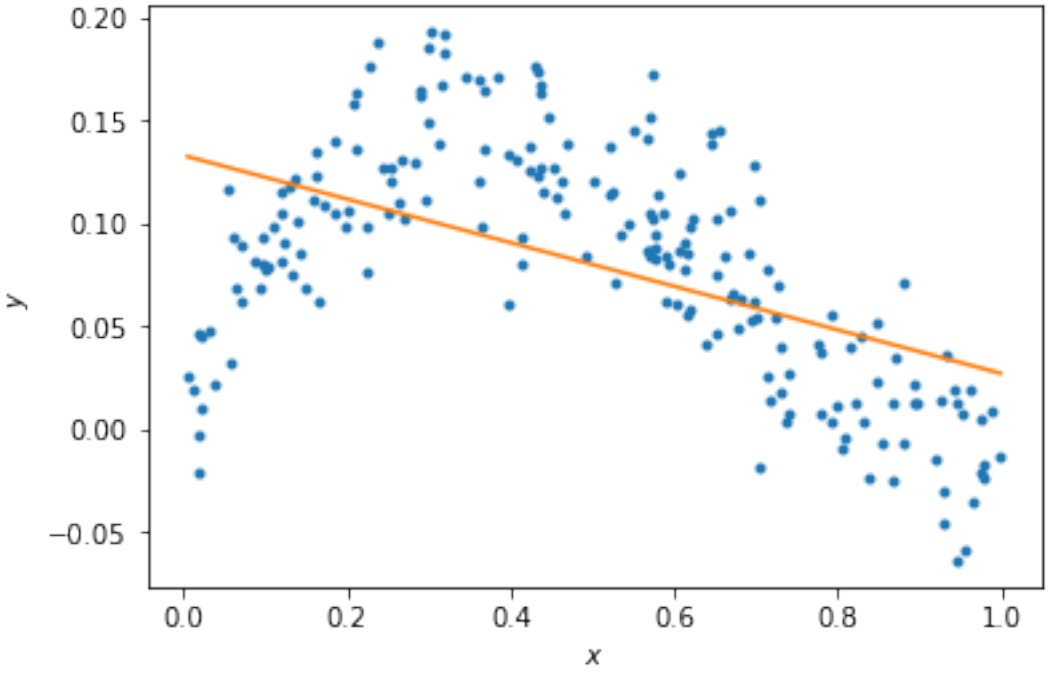
```
theta = np.linalg.inv(xhat.dot(xhat.T)).dot(xhat.dot(y))

# ================== #
# 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 0x21ddc2316d8>]



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 the data because it doesn't accuractly represent the function

(2) We would use an X^3 model to fit better, aka a higher order polynomial.

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 [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
#   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 range(0,N):
    if i == 0:
        xhat = np.vstack((x,np.ones_like(x)))
    else:
        xhat = np.vstack((x**(i+1),xhats[i-1]))
    xhats.append(xhat)

for i in range(0,N):
    thetas.append(np.linalg.inv(xhats[i].dot(xhats[i].T)).dot(xhats[i].dot(y)))

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

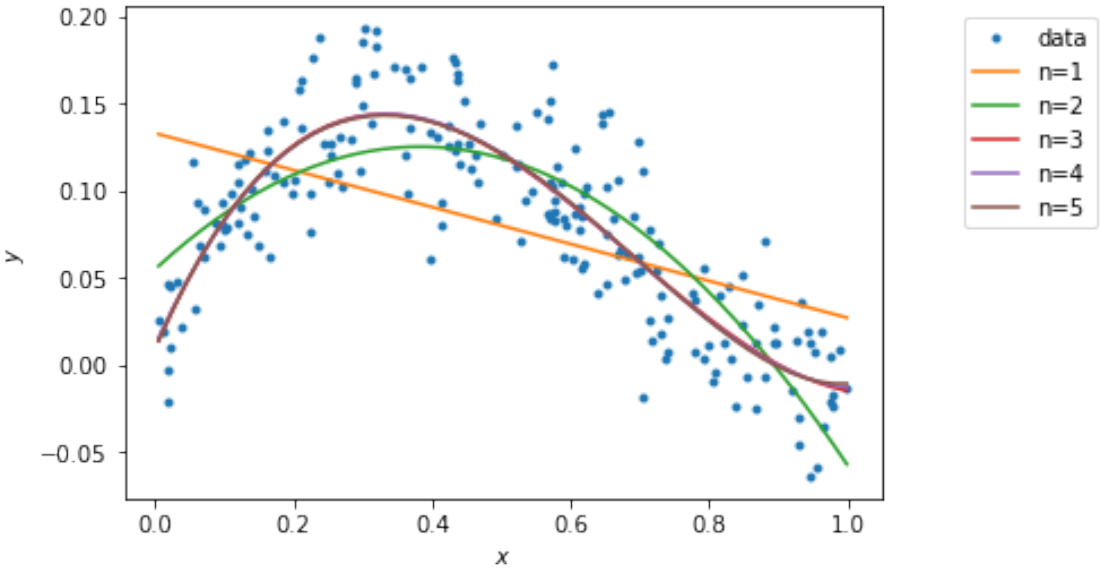
In [6]:
```
# Plot the data
f = plt.figure()
ax = f.gca()
ax.plot(x, y, '.')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')

# 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)
```



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 [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 range(0,N):
            yhats = thetas[i].dot(xhats[i])
            mse = (y - yhats)**2
            training_errors.append((1/2)*sum(mse))

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

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

```
Training errors are:
 [0.2379961088362701, 0.10924922209268527, 0.08169603801105373, 0.08165353735296979, 0.0816147
```

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 order 5 has the best training error

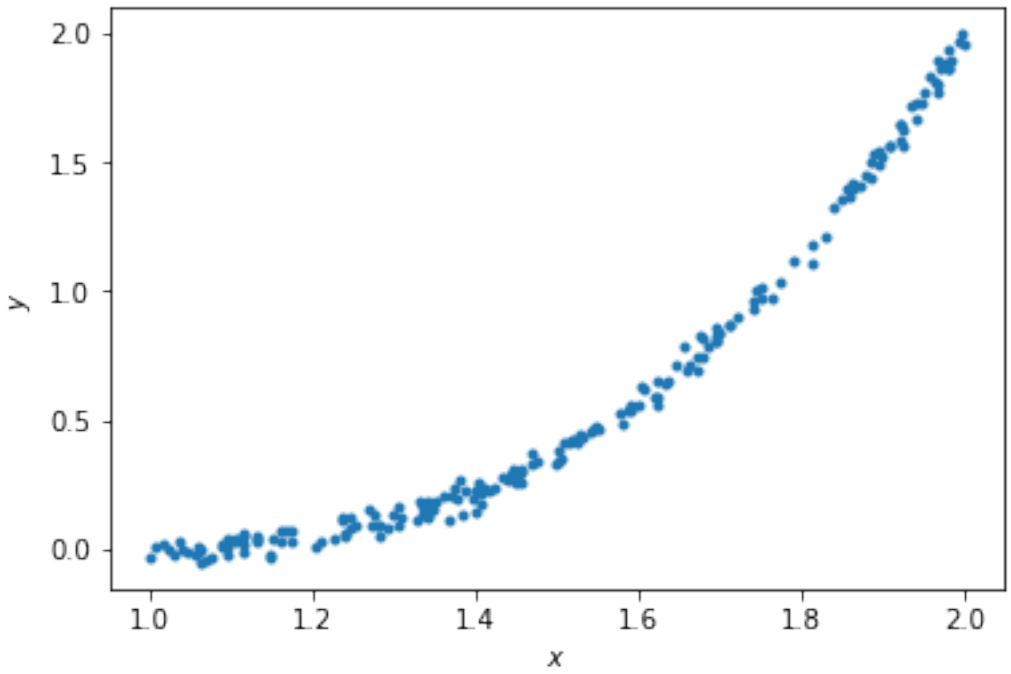
(2) This is to be expected because a higher training order can mimick the lower order by setting the higher order coefficient to zero. It has more freedom and it over-fits the data

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 [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)
```

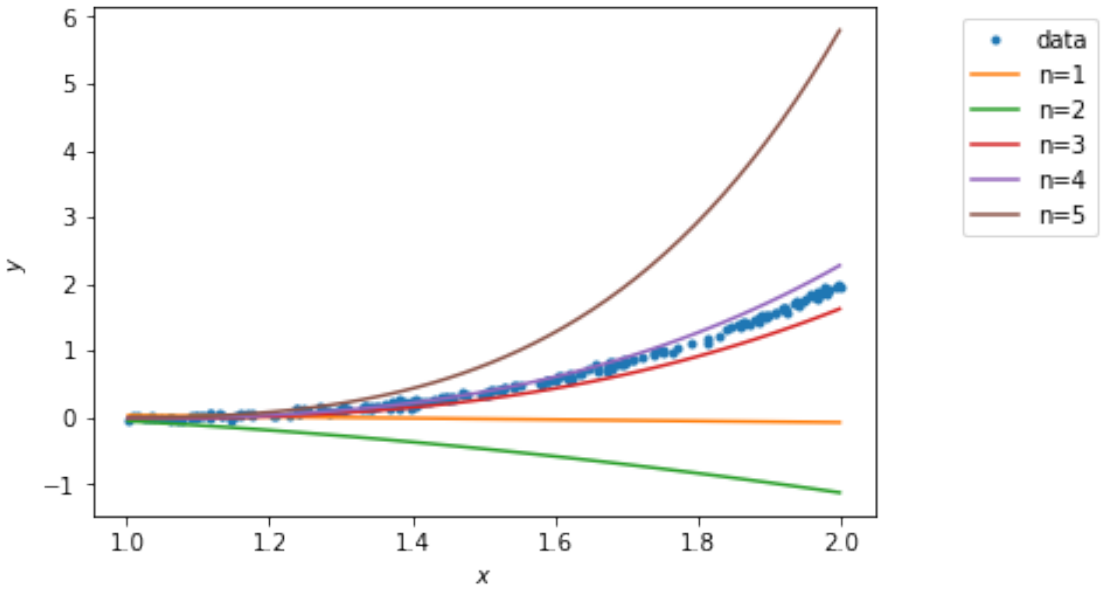
```
In [10]: # Plot the data
         f = plt.figure()
         ax = f.gca()
         ax.plot(x, y, '.')
         ax.set_xlabel('$x$')
         ax.set_ylabel('$y$')

         # 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 [14]:
```
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):
    yhats = thetas[i].dot(xhats[i])
    mse = (yhats - y)**2
    testing_errors.append((1/2)*sum(mse))

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

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

```
Testing errors are:
 [80.86165184550582, 213.19192445057888, 3.1256971082763934, 1.1870765189474701, 214.910218176
```

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 of the 4 order has a better testing error now.

(2) Polynomial of order 5 does not generalize well because they tend to overfit the data, which allows the variance in mse term to grow greater than the bias. In other words, when over fitting you reduce bias but you get more error in terms of noise.