Jonathan Hurwitz
804258351
EE239 - HW1

(1) $AA^T = I$

a.i) $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$A^T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$

$AA^T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} a & c \\ b & d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$a^2 + b^2 = 1$ $\quad a = \cos\Theta$

$ac + bd = 0$ $\quad b = \sin\Theta$

$c^2 + d^2 = 1$ $\quad c = \ldots\sin\Theta$

$d = -\cos\Theta$

$bd = -ac$

} only need 1 negative term

$A = \begin{bmatrix} \cos\Theta & \sin\Theta \\ \sin\Theta & -\cos\Theta \end{bmatrix}$

$Av = \lambda v$

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

$\begin{bmatrix} \cos\Theta - \lambda & \sin\Theta \\ \sin\Theta & -\cos\Theta - \lambda \end{bmatrix} = A - \lambda I$

$|A - \lambda I| = (\cos\Theta - \lambda)(-\cos\Theta - \lambda) - \sin^2\Theta$

$0 = -\cos^2\Theta + \lambda^2 + \lambda\cos\Theta - \lambda\cos\Theta - \sin^2\Theta$

$\lambda^2 = \sin^2\Theta + \cos^2\Theta$

$\lambda = \pm 1$

$\lambda_1 = 1$

$\lambda_2 = -1$

λ_1: $\begin{bmatrix} \cos\Theta - 1 & \sin\Theta \\ \sin\Theta & -\cos\Theta - 1 \end{bmatrix}\begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = 0$

$(\cos\Theta - 1)v_{11} + \sin\Theta\, v_{12} = 0$

$\sin\Theta\, v_{11} - (\cos\Theta + 1)v_{12} = 0$

$v_{12} = 1$

$(\cos\Theta - 1)v_{11} + \sin\Theta = 0$

$(\sin\Theta)v_{11} - \cos\Theta - 1 = 0$

$v_{11} = \frac{\sin\Theta}{\cos\Theta - 1}$

$v_{11} = \frac{\cos\Theta}{\sin\Theta} + \frac{1}{\sin\Theta}$

$v_{11} = \cot(\Theta) + \csc(\Theta)$

$(\cos\Theta - 1)\left(\frac{\cos\Theta}{\sin\Theta} + \frac{1}{\sin\Theta}\right) + \sin\Theta$

$\frac{\cos^2\Theta}{\sin\Theta} - \frac{1}{\sin\Theta} - \frac{\cos\Theta}{\sin\Theta} + \frac{\cos\Theta}{\sin\Theta} + \sin\Theta$

$\frac{\cos^2\Theta - 1}{\sin\Theta} + \sin\Theta = \frac{-\sin^2\Theta}{\sin\Theta} + \sin\Theta$

$= 0$ ✓

$v_1 = \begin{bmatrix} \cot(\Theta) + \csc(\Theta) \\ 1 \end{bmatrix}$

λ_2: $\begin{bmatrix} \cos\theta + 1 & \sin\theta \\ \sin\theta & -\cos\theta + 1 \end{bmatrix} \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = 0$

$(\cos\theta + 1)\, v_{21} + (\sin\theta)\, v_{22} = 0$

$(\sin\theta)\, v_{21} + (-\cos\theta + 1)\, v_{22} = 0$ for $v_{22} = 1$

$$v_{21} = \frac{\cos\theta}{\sin\theta} - \frac{1}{\sin\theta}$$

$$v_{21} = \cot(\theta) - \csc(\theta)$$

$$v_2 = \begin{bmatrix} \cot(\theta) - \csc(\theta) \\ 1 \end{bmatrix}$$

Eigenvalues are $\lambda_1 = 1$, $\lambda_2 = -1$ and eigenvectors are:

$$v_1 = \begin{bmatrix} \cot(\theta) + \csc(\theta) \\ 1 \end{bmatrix}, \quad v_2 = \begin{bmatrix} \cot(\theta) - \csc(\theta) \\ 1 \end{bmatrix}$$

$$\vec{v}_1 \cdot \vec{v}_2 = (\cot(\theta) + \csc(\theta))(\cot(\theta) - \csc(\theta)) + 1$$

$$= \cot^2(\theta) + \csc(\theta)\cot(\theta) - \csc^2(\theta) - \csc(\theta)\cot(\theta) + 1$$

$$= \frac{\cos^2\theta}{\sin^2\theta} - \frac{1}{\sin^2\theta} + 1$$

$$= \frac{\cos^2 - 1}{\sin^2\theta} + 1$$

$$= \frac{(-\sin^2\theta)}{\sin^2\theta} + 1$$

$= -1$

$$\boxed{\vec{v}_1 \cdot \vec{v}_2 = 0}$$

↳ the dot product of the eigenvectors is 0, so they are orthogonal.

aii) Let A be an orthogonal matrix s.t. $A^TA = I$ is satisfied.
Let λ_i be some eigenvalue associated with eigenvector v_i

$$Av_i = \lambda_i v_i$$

$$\|Av_i\|^2 = \|\lambda_i v_i\|^2 \quad \text{since } \lambda_i \text{ is a scalar}$$

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

$$v_i^T A^T A v_i = |\lambda_i|^2 \|v_i\|^2 \quad \text{since } A^TA = I$$

$$v_i^T I v_i = |\lambda_i|^2 \|v_i\|^2$$

$$\|v_i\|^2 = |\lambda_i|^2 \|v_i\|^2 \quad \text{since } v_i^T v_i = \|v_i\|^2$$

$A, I \in \mathbb{R}^{m \times m}$
$v_i \in \mathbb{R}^{m \times n}$

$(n \times m)(m \times m)(m \times n) \to n \times n$

$|\lambda_i|^2 = 1$

$|\lambda_i| = \pm 1$ → length cannot be negative, so

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

aiii) $A \in \mathbb{R}^{n \times n}$

$A v_1 = \lambda_1 v_1$

$A v_2 = \lambda_2 v_2$ } $\lambda_1 \neq \lambda_2$ (distinct eigenvalues)

$v_1^T v_2 = v_1^T I v_2$ → $A^T A = I$

$= v_1^T A^T A v_2$

$= (A v_1)^T (A v_2)$

$= \lambda_1 v_1^T \lambda_2 v_2$

$= \lambda_1 \lambda_2 v_1^T v_2$

Since $\lambda_i \in \{-1, 1\}$, if $\lambda_1 \neq \lambda_2$ then $\lambda_1 \lambda_2 \neq 1$ and $v_1^T v_2 = 0$, meaning $v_1 \perp v_2$

aiv) The transformation Ax maps some coefficient $\wedge$ $a_{ij} \in A$ to $x_i \in \vec{x}$ $\forall i, j$. The coefficients in A are associated with some variable in x. This forms a system of equations when a b is given s.t. $Ax = b$.

bi) The eigenvectors of AA^T are the left-singular vectors of A.
The eigenvectors of A^TA are the right-singular vectors of A.

bii) The square roots of the eigenvalues of A^TA and AA^T are the singular (non-zero) values of A.

c i) F — Can have an operator w/ no eigenvalues

c ii) F

c iii) T — - IF A is PSD, $x^TAx \geq 0 \; \forall x$, then its eigenvalues are all positive or zero-valued

c iv) F — By rank-nullity theorem, rank(A) + nullity(A) = n for $A \in \mathbb{R}^{n \times n}$, thus, rank(A) cannot exceed # of non-zero eigenvalues — it is in fact equal to the # of non-zero eigenvalues.

c v) T

$Av_1 = \lambda v_1$

$Av_2 = \lambda v_2$ $\Rightarrow \quad u = v_1 + v_2$ Prove: $Au = \lambda u$

In general, let $v_1, \ldots, v_N \in \mathbb{R}^m$ be the eigenvectors of A sharing eigenvalue λ.

$Av_i = \lambda v_i \quad \forall \; v_i \in \{v_1, \ldots, v_N\}$

$$Au = A\sum_i^m v_i \quad \Big\} \text{ where } u = \sum_i^m v_i$$

$$= \sum_i^m Av_i$$

$$= \sum_i^m \lambda v_i$$

$$= \lambda \sum_i^m v_i$$

$Au = \lambda u$ ∎

$\longrightarrow$ for $m = 2$ case:

$Au = A(v_1 + v_2)$

$= Av_1 + Av_2$

$= \lambda v_1 + \lambda v_2$

$= \lambda(v_1 + v_2)$

$Au = \lambda u$ ∎

(2)

a i) $P(c = H50 \mid x = tails) = \dfrac{p(x = tails \mid c = H50)\, p(c = H50)}{p(x = tails)}$

$= \dfrac{(0.5)(0.5)}{p(x = tails)}$ → $p(x = tails) = p(x = tails \mid c = H50)\, p(c = H50) + p(x = tails \mid c = H60)\, p(c = H60)$

$= (0.5)(0.5) + (0.4)(0.5)$

$= 0.25 + 0.2$

$p(x = tails) = 0.45$

$= \dfrac{(0.5)(0.5)}{0.45}$

$\boxed{p(c = H50 \mid x = tails) = 0.5\overline{5}}$

a ii) $p(c = H50 \mid x_1 = T, x_2 = H, x_3 = H, x_4 = H) \equiv p(c = H50 \mid T, H, H, H)$

$p(c = H50 \mid T, H, H, H) = p(c = H50)\, p(x = T \mid c = H50)\, p(x = H \mid c = H50)\, p(x = H \mid c = H50)\, p(x = H \mid c = H50)$

$= (0.5)(0.5)^4$

$\boxed{p(c = H50 \mid T, H, H, H) = 0.03125}$

a iii)

$H50 = 0.5$
$H55 = 0.55$
$H60 = 0.6$
} p of heads

$p(c = H50 \mid 9H, 1T) = p(c = H50)\, p(x = H \mid c = H50)^9\, p(x = T \mid c = H50)$

$= \left(\frac{1}{3}\right)\left(\frac{1}{2}\right)^9\left(\frac{1}{2}\right)$

$= \left(\frac{1}{3}\right)\left(\frac{1}{2^{10}}\right)$

$\boxed{p(c = H50 \mid 9H, 1T) = 0.00033}$

$p(c = H55 \mid 9H, 1T) = p(c = H55)\, p(x = H \mid c = H55)^9\, p(x = T \mid c = H55)$

$= \left(\frac{1}{3}\right)(0.55)^9(0.45)$

$\boxed{p(c = H55 \mid 9H, 1T) = 0.00069}$

$p(c = H60 \mid 9H, 1T) = p(c = H60)\, p(x = H \mid c = H60)^9\, p(x = T \mid c = H60)$

$= \left(\frac{1}{3}\right)(0.6)^9(0.4)$

$\boxed{p(c = H60 \mid 9H, 1T) = 0.00134}$

b) $p(t=P \mid w=preg) = 0.99$

$p(t=p \mid w=\neg preg) = 0.1$

$\Rightarrow$

$p(w=preg) = 0.01$

$p(w=\neg preg) = 0.99$

$$p(t=p) = p(t=P \mid w=preg)\,p(w=preg) + p(t=P \mid w=\neg preg)\,p(w=\neg preg)$$

$$= (0.99)(0.01) + (0.1)(0.99)$$

$$p(t=p) = 0.1089$$

$$p(w=preg \mid t=P) = \frac{p(t=P \mid w=preg)\,p(w=preg)}{p(t=p)}$$

$$= \frac{(0.99)(0.01)}{0.1089}$$

$$\boxed{p(w=preg \mid t=p) = 0.091}$$

{ Makes sense because most of the female population is not pregnant yet test returns positive for 10% of this majority population.

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

$$E[\alpha f(x) + \beta g(x)] = \alpha E[f(x)] + \beta E[g(x)]$$

$$E(Ax+b) = E(Ax) + E(b)$$

$$\boxed{E(Ax+b) = AE[x] + b}$$

d) $$cov(x) = E((x - Ex)(x - Ex)^T)$$

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

$$= E[(Ax+b - AE[x]-b)(Ax+b - AE[x]-b)^T]$$

$$= E[(Ax - AE[x])(Ax - AE[x])^T]$$

$$= E[A^2((x - E[x])(x - E[x])^T)]$$

$$= A^2 E[(x - E[x])(x - E[x])^T]$$

$$\boxed{cov(Ax+b) = A^2 cov(x)}$$

(3)

a) $x \in \mathbb{R}^{n\times 1}$, $y \in \mathbb{R}^{m\times 1}$, $A \in \mathbb{R}^{n\times m}$ $\longrightarrow$ $\nabla_x \underset{(1\times n)(n\times m)(m\times 1)}{x^T A y} \rightarrow (1\times 1)$

$$\boxed{\nabla_x x^T A y = (Ay)^T}$$

$$\begin{bmatrix} a_1 \end{bmatrix} y_1 + \cdots + \begin{bmatrix} a_m \end{bmatrix} y_m$$

$$\frac{\partial(x^T a)}{\partial x} = \frac{\partial(a^T x)}{\partial x} = a^T \qquad \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix} = s_1 x_1 + \cdots + s_n x_n$$

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

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

$x^T A y$
$(1\times n)(m\times m)(m\times 1)$

$[\longleftrightarrow]_{1\times m} [y]$

c) $\nabla_A x^T A y$

$x^T \underbrace{A y}$
$(1\times n) \quad (m\times 1)$

$$\nabla_A x^T A y = \nabla_A x^T \sum_{i=1}^{n} \sum_{j=1}^{m} A_{ij} y_j$$

$$= \nabla_A x^T \sum_{j=1}^{m} \sum_{i=1}^{n} A_{ij} y_j$$

$$= x^T \sum_{j=1}^{m} \sum_{i=1}^{n} \nabla_A A_{ij} y_j$$

$$= x^T \sum_{j=1}^{m} \sum_{i=1}^{n} y_j$$

$$= \sum_{j=1}^{m} \sum_{i=1}^{n} x_j y_i$$

$$\boxed{\nabla_A x^T A y = x y^T}$$

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

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

$$= (Ax)^T + x^T A + b^T$$

$$= A^T x^T + x^T A + b^T$$

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

$\frac{\partial (x^T a)}{\partial x} = \frac{\partial (a^T x)}{\partial x} = a^T$

$\frac{\partial ((a x^T)^T)}{\partial x} = a^T$

e) $f = tr(AB)$ Assume $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times m}$

$\nabla_A f = \nabla_A tr(AB)$

$$tr(AB) = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} A_{ij} B_{ji} \right)$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ji} = \sum_{j=1}^{n} \sum_{i=1}^{m} B_{ji} A_{ij}$$

$$\nabla_A \left(\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ji} \right) = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{\partial (A_{ij} B_{ji})}{\partial A_{ij}}$$

$\frac{\partial}{\partial A_{ij}} A_{ij} B_{ji} = B_{ji}$

$$\boxed{\nabla_A tr(AB) = B^T}$$

(4)

$\hat{y} = wx$

$$\min_w \frac{1}{2} \sum_{i=1}^{n} \| y^{(i)} - w x^{(i)} \|^2 = \min_w \frac{1}{2} \| \vec{y} - Xw \|^2$$) drop $\frac{1}{2}$

$$\nabla_w \frac{1}{2} \sum_{i=1}^{n} \| y^{(i)} - w x^{(i)} \|^2 = \nabla_w \frac{1}{2} \| \vec{y} - Xw \|^2$$

n

$y \in \mathbb{R}^{n \times 1}$

$X \in \mathbb{R}^{n \times m}$

$w \in \mathbb{R}^{m \times 1}$

$$\| y - Xw \|^2 = (y - Xw)^T (y - Xw)$$

$$= (y^T - x^T w^T)(y - Xw)$$

$$= y^T y + w^T X^T X w - w^T X^T y - y^T X w$$

$(1 \times n)(n \times 1) \quad (1 \times m)(m \times n)(n \times m)(m \times 1) \quad (1 \times m)(m \times n)(n \times 1)$ → $(1 \times n)(n \times m)(m \times 1)$

$y^T \in \mathbb{R}^{1 \times n}$

$x^T \in \mathbb{R}^{m \times n}$

$w^T \in \mathbb{R}^{1 \times m}$

$$\| y - Xw \|^2 = y^T y + w^T X^T X w - 2 w^T X^T y$$

$$\nabla_w \| y - Xw \|^2 = \nabla_w (y^T y) + \nabla_w (w^T X^T X w) - \nabla_w (2 w^T X^T y)$$

$$\nabla_w \| y - Xw \|^2 = 0 + (X^T X) w - 2 X^T y$$) set to 0 to find min

$\nabla_w = 0 \rightarrow 2(X^T X) w - 2 X^T y = 0$

$(X^T X) w = X^T y$

$$\boxed{w = (X^T X)^{-1} X^T y}$$

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 239AS, Winter Quarter 2018, Prof. J.C. Kao, TAs C. Zhang and T. Xing

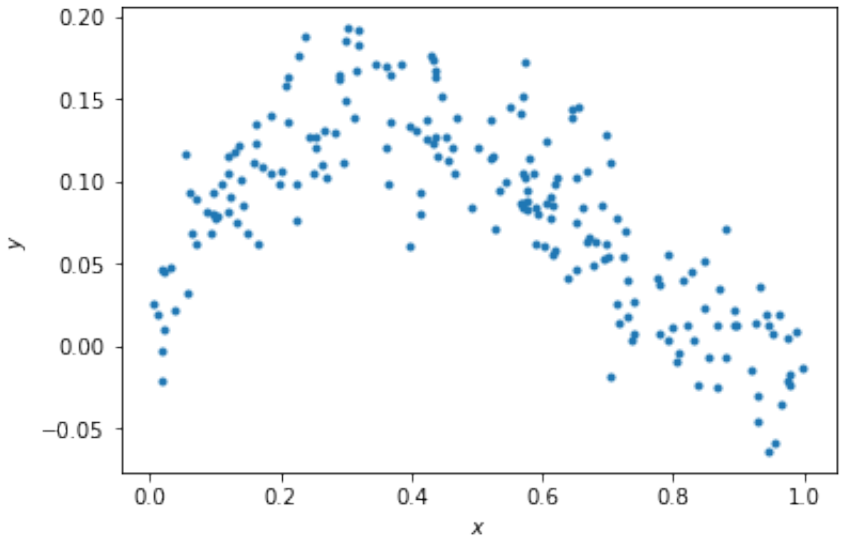
In [478]:
```
import numpy as np
import matplotlib.pyplot as plt

#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 [479]:
```
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[479]: 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 uniform, since we're creating it via np.random.uniform.

(2) The distribution of the additive noise is Gaussian, since we're creating it via np.random.normal.

Fitting data to the model (5 points)

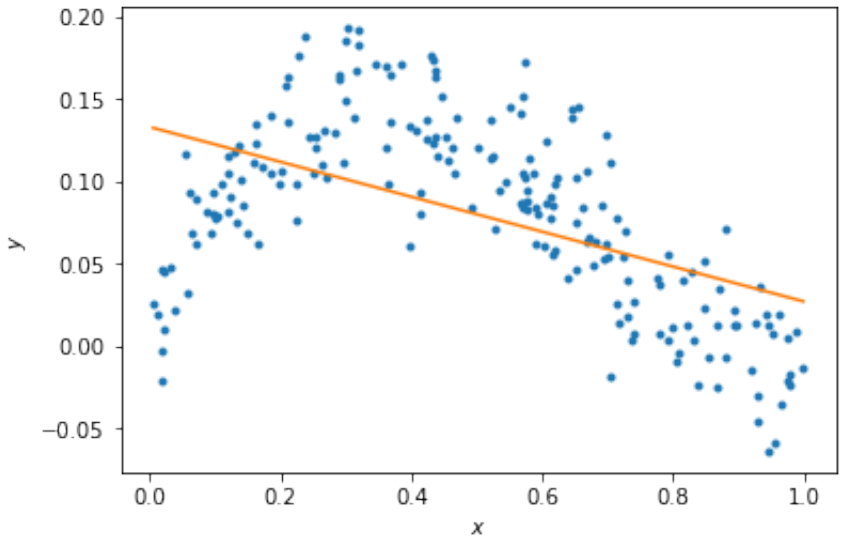
Here, we'll do linear regression to fit the parameters of a model $y = ax + b$.

In [480]:
```
# xhat = (x, 1)
xhat = np.vstack((x, np.ones_like(x)))
#print(xhat)

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

#using normal eq
theta = np.linalg.inv(xhat.dot(xhat.T)).dot(xhat.dot(y))

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

In [481]:
```
# Plot the data and your model fit.
print(theta)
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))
```

```
[-0.10599633  0.13315817]
```

Out[481]:
```
[<matplotlib.lines.Line2D at 0x114e09ac8>]
```



QUESTIONS

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

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

ANSWERS

(1) The linear model underfits the data, since it is too simple. This data appears to resemble some higher order polynomial function (such as a parabola), but our linear model has no higher order terms.

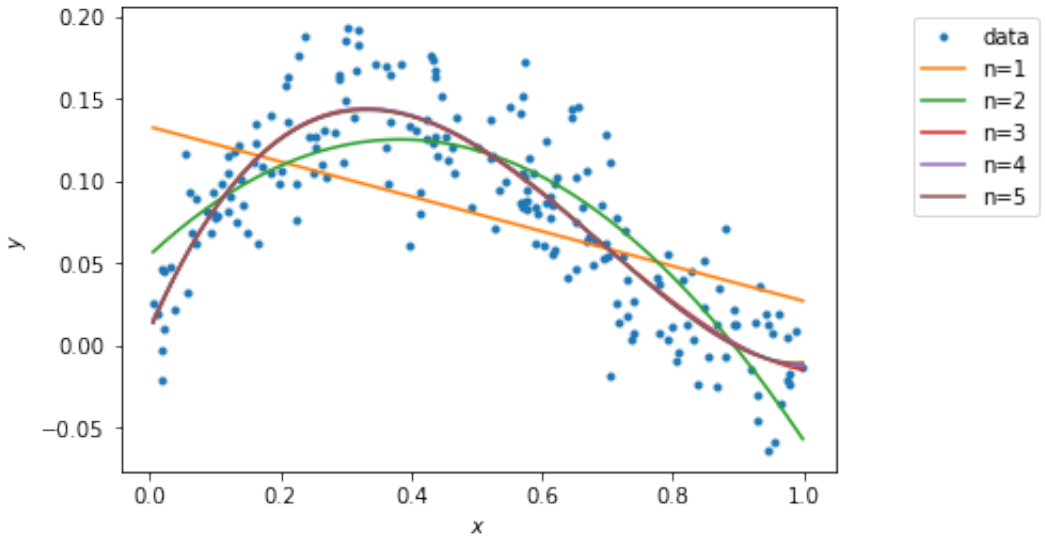
(2) We can introduce higher order terms in order to improve the fit.

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 [482]:
```
N = 5
xhats = []
thetas = []

# ==================== #
# START YOUR CODE HERE #
# ==================== #
#create xhats
xhats.append(xhat)
#print(xhats[0])
for i in range(1,N):
   arr = np.vstack((np.array(xhat[0]**(i+1)), xhats[i-1]) )
   xhats.append(arr)


#print(xhats[3])

# GOAL: create a variable thetas.
# thetas is a list, where theta[i] are the model parameters for the po
lynomial 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 coefficient
s of the x^2, x, and 1 respectively.
#   ... etc.
#theta = np.linalg.inv(xhat.dot(xhat.T)).dot(xhat.dot(y))
#print(theta)

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


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

In [483]:
```
# 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),200), np.ones(2
00)))
    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 [484]:
```
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 fi
t of order i+1.

for i in range(0, N):
    mse = ((y - thetas[i].dot(xhats[i]))**2).mean(axis=None)
    training_errors.append(mse)



min_error_index = np.argmin(training_errors) + 1
print("Best polynomial is order: " + str(min_error_index))

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

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

```
Best polynomial is order: 5
Training errors are:
 [0.0023799610883627007, 0.001092492220926853, 0.0008169603801105369
4, 0.00081653537352969791, 0.00081614791955252942]
```

QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

ANSWERS

(1) The 5th order polynomial has best fit.

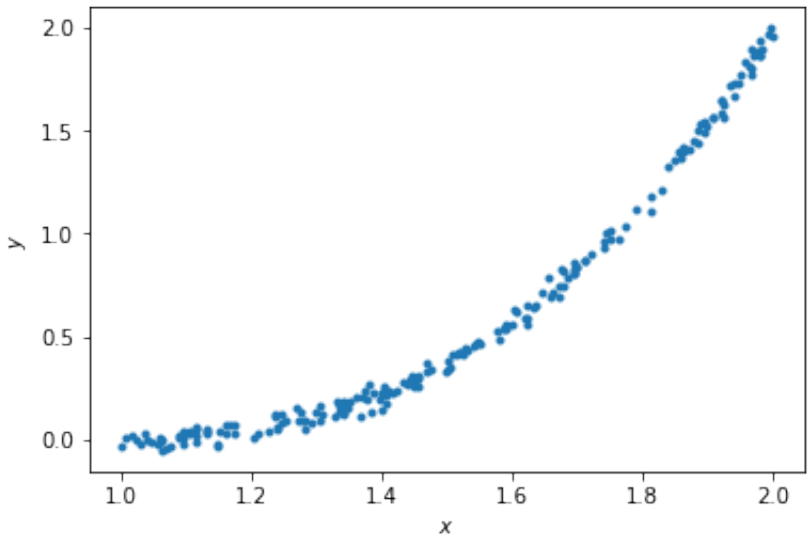
(2) This is expected since it is our most complicated model (it has the highest possible degree term). Since the goal is to minimize the cost function, weights will be adjusted such that the distribution can be best approximated. Given enough complexity, models will overfit and "memorize" the data, as shown here. It's probable that while the 5th order polynomial performed best in terms of training error, it will do poorly in terms of test error for a newly generated set. Models that have memorized the training set do not generalize well.

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.

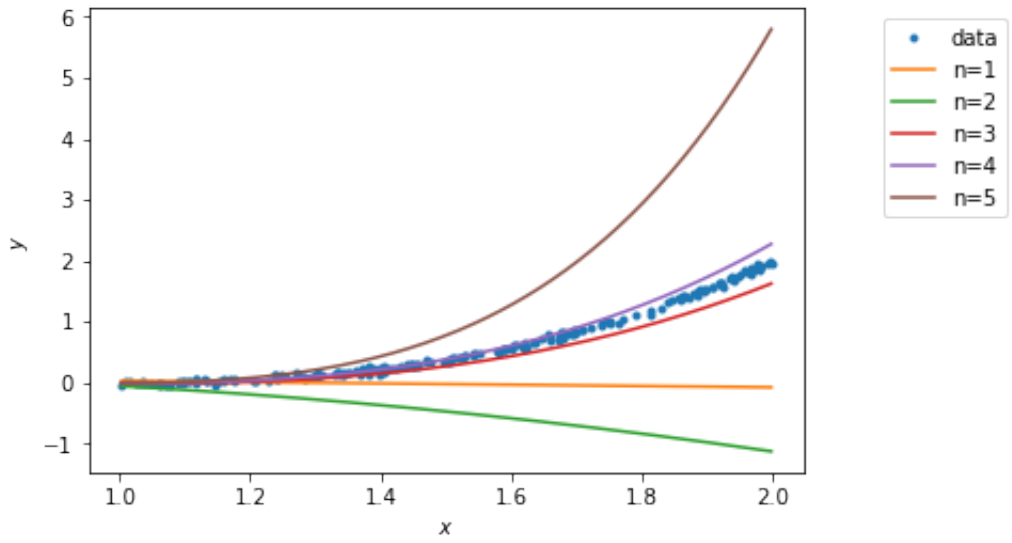
In [485]:
```
x = np.random.uniform(low=1, high=2, size=(num_train,))
y = x - 2*x**2 + x**3 + np.random.normal(loc=0, scale=0.03, size=(num_
train,))
f = plt.figure()
ax = f.gca()
ax.plot(x, y, '.')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
```

Out[485]:
```
Text(0,0.5,'$y$')
```



In [486]:
```
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 [487]:
```
# 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),200), np.ones(2
00)))
    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 [488]:
```
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):
    mse = ((y - thetas[i].dot(xhats[i]))**2).mean(axis=None)
    testing_errors.append(mse)

index_min_error = np.argmin(testing_errors)
print("Lowest test error for polynomial: " + str(index_min_error + 1))
# ================== #
# END YOUR CODE HERE #
# ================== #

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

```
Lowest test error for polynomial: 4
Testing errors are:
 [0.80861651845505866, 2.1319192445057897, 0.031256971083289675, 0.0
11870765196608337, 2.1491021807712438]
```

QUESTIONS

(1) What polynomial has the best testing error?

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

ANSWERS

(1) The 4th order polynomial has the best test error.

(2) Order 5 polynomial models do not generalize well because they are overfitting. This means that the model is too complex and is able to effectively memorize the test data. Since linear regression "learning" is based on adjust weights the minimize a cost function, a complex higher order function that can memorize the test data will do very well in training (since the criteria is loss w/ respect to training data) but will do poorly on data that has not been incorporated into the generalization. Order 4 seems to be sufficiently complex such that it can reasonbly output values from the target distribution without overfitting.