1) Linear Algebra Review

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

$$AA^T = I \Rightarrow \begin{bmatrix} a^2+b^2 & ac+bd \\ ac+bd & c^2+d^2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$a^2+b^2 = 1$ $\quad ac = -bd$

$c^2+d^2 = 1$

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

Eigenvalues & Eigen Vectors

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

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

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

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

$$\lambda^2 = 1$$

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

Eigen Vectors

for $\lambda_1 = 1$

$$(A - \lambda_1 I)x_1 = 0$$

$$(A - I)x_1 = 0$$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} - 1 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} - 1 \end{bmatrix} x_1 = 0$$

$$\therefore x_1 = \begin{bmatrix} 1 \\ \sqrt{2} - 1 \end{bmatrix}$$

for $\lambda_1 = -1$

$$(A + I)x_2 = 0$$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} + 1 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} + 1 \end{bmatrix} x_2 = 0$$

$$x_2 = \begin{bmatrix} 1 - \sqrt{2} \\ 1 \end{bmatrix}$$

we notice the following:

$|\lambda| = 1$ & ~~norm~~ of eigenvector

$|\det A| = 1$. Eigen vectors are orthogonal to each other & can be normalized.

3

i)

ii) $Av = \lambda v$

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

$$\|Av\|^2 = v^T A^T A v = v^T v = \|v\|^2$$

$$\therefore \|\lambda\|^2 = 1 \implies \boxed{|\lambda| = 1}$$

iii) if $AA^T = I \implies A^TA = I$

in this case, $\lambda = \pm 1$.

$Ax_1 = \lambda_1 x_1$ $\qquad$ $AX_2 = \lambda_2 X_2$

$AX_1 = X_1$ $\qquad$ $AX_2 = -X_2$

$X_1^T = X_1^T A^T$ $\qquad$ $X_2 = -AX_2$

$$X_1^T X_2 = (X_1^T A^T)(-AX_2)$$

$$= -X_1^T A^T A X_2$$

$$\boxed{X_1^T X_2 = -X_1^T X_2}$$

$$\therefore X_1^T X_2 = 0$$

$\Rightarrow$ $\underline{\underline{X_1 \perp^r X_2}}$ $\quad\therefore$ eigenvectors corresponding to distinct eigenvalues are orthogonal.

iv) Any vector X is subject to a rotation or reflection under the transformation AX.

In this current example, $\det A = 1$

$\therefore$ it is a rotation by $45°$ in the counter-clockwise direction.

The orientation & the length of X are preserved as is.

---

b) A is a matrix

A can be written as

$$\boxed{A = UDV^T}$$

columns of $U \rightarrow$ left singular vectors

columns of $V \rightarrow$ right singular vectors

elements along diagonal of $D \rightarrow$ singular values.

i) The left singular vectors of A are the eigenvectors of AA^T.

The Right singular vectors of A are the eigenvectors of A^TA.

ii) The non zero singular values of 'A' are the square roots of eigen values of A^TA.

The same is true for AA^T.

---

c) True / false

i) False : There are linear operators with no eigenvalues

ii) False

iii) True

iv) False. If $A = UDV^T \Rightarrow$ rank(A) = rank(D) = # of non zero eigen values.

v) True.

Q2) Probability Refresher

a)

i) $$P(H50|T) = \frac{P(T|H50) \cdot P(H50)}{P(T)} \quad -①$$

$$P(T) = P(T|H50) \cdot H50 + P(T|H60) \cdot P(H60)$$

$$= \frac{1}{2} \cdot \frac{1}{2} + \frac{2}{5} \cdot \frac{1}{2}$$

$$P(T) = \frac{9}{20}$$

substituting in ①

$$\therefore P(H|50|T) = \frac{1/4}{9/20} = \frac{5}{9} = 0.5555$$

$\therefore$ The posterior probability that the tail is an H50 coin is $\underline{0.5555}$.

ii) $$P(H50|THHH) = \frac{P(THHH|H50) \cdot P(H50)}{P(THHH)}$$

$$P(THHH) = P(THHH|H50)\,P(H50) + P(THHH|H60)\,P(H60)$$

$$= \left(\frac{1}{2}\times\frac{1}{2}\times\frac{1}{2}\times\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{2}{5}\times\frac{3}{5}\times\frac{3}{5}\times\frac{3}{5}\right)\left(\frac{1}{2}\right)$$

$$= \frac{1}{16} + \frac{9}{125}$$

$\therefore P(H50 / THHH) = \dfrac{1/16}{1/16 + \frac{54}{1250}}$

$$\boxed{P(H50 / THHH) = \cancel{0.4646}} \; 0.4197$$

iii)

$P(H50 / 9 \text{ Heads}, 1 \text{ Tail}) = \dfrac{P(9 \text{ heads}, 1 \text{ Tail} / H50) \cdot P(H50)}{P(9 \text{ heads}, 1 \text{ tail})}$

$\Rightarrow P(9 \text{ heads}, 1 \text{ tail}) = P(9 \text{ heads } 1 \text{ Tail} / H50)\, P(H50)$
$+ P(9 \text{ heads } 1 \text{ Tail} / H55)\, P(H55)$
$+ P(9 \text{ heads } 1 \text{ Tail} / H60)\, P(H60)$

$= \dfrac{1}{2^9} \cdot \dfrac{1}{2} \cdot \dfrac{1}{3} + (0.55)^9 (0.45) \cdot \dfrac{1}{3}$

$+ (0.6)^9 (0.4) \cdot \dfrac{1}{3}$

$= 2.36 \times 10^{-3}$

$\therefore P(H50 / 9 \text{ heads } 1 \text{ tail}) = \dfrac{3.25 \times 10^{-4}}{2.36 \times 10^{-3}}$

$$\boxed{P(H50 / 9 \text{ heads } 1 \text{ tail}) = 0.1379}$$

$$P(H55 \mid 9H1T) = \frac{P(9H1T \mid H55)\ P(H55)}{P(9H1T)}$$

$$P(9H1T) = P(9H1T \mid H50)\, P(H50) + P(9H1T \mid H55)\, P(H55) + P(9H1T \mid H60)\, P(H60)$$

$$= 2.36 \times 10^{-3}$$

$$P(H55 \mid 9H1T) = \frac{6.90 \times 10^{-4}}{2.36 \times 10^{-3}}$$

$$\boxed{P(H55 \mid 9H1T) = 0.2927}$$

similary,

$$P(H60 \mid 9H1T) = \frac{P(9H1T \mid H60)\ P(H60)}{P(9H1T)}$$

$$= \frac{1.3436 \times 10^{-3}}{2.36 \times 10^{-3}}$$

$$\boxed{P(H60 \mid 9H1T) = 0.5694}$$

2) b)

	Positive	Negative
Pregnant	0.99	0.01
Not Pregnant	0.10	0.90

$$P(\text{Pregnant} \mid \text{Positive}) = \frac{P(\text{Positive} \mid \text{Pregnant})\, P(\text{Pregnant})}{P(\text{Positive})}$$

$$P(\text{Positive}) = P(\text{Positive}/\text{Preg.}) * P(\text{Preg}) + P(\text{Positive} \mid \text{Not Preg.}) * P(\text{Not Preg.})$$

$$= (0.99 \times 0.01) + (0.10 \times 0.99)$$

$$= 0.1089$$

$$\therefore P(\text{Preg.} \mid \text{Positive}) = \frac{0.99 \times 0.01}{0.1089}$$

$$= \frac{9.9 \times 10^{-3}}{0.1089}$$

$$\boxed{\therefore P(\text{Preg} \mid \text{Positive}) = 0.09091}$$

Intiutively, the answer says that the pregnancy test results are <u>not</u> <u>reliable</u> since a woman is pregnant only 9% of the times when the test results are positive.

2) c)

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

$$E(X) = \sum_x x\,p(x)$$

$$\therefore E(AX+b) = \sum_x (AX+b)\,p(x)$$

$$= \sum_x AX\,p(x) + b\sum_x p(x)$$

$$= A\sum_x x\,p(x) + b(1)$$

$$\boxed{\therefore E(Ax+b) = A\,E(X) + b}$$

This proves linearity of expectation with respect to one variable.

2) d)

$$Cov(x) = E\left((X-E(X))(X-E(X))^T\right)$$

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

$$= E\left[(Ax+\cancel{b}-AE(X)-\cancel{b})\,(Ax+b-AE(X)-b)^T\right]$$

{from part c}

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

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

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

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

Q) Problem 3: Multivariate Derivative

a) $x \in R^n,\ y \in R^m,\ A \in R^{n \times m}$

Let,

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

$$f = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}\, x_i\, y_j$$

a) $\nabla_x x^T A y$

$$\frac{\partial f}{\partial x_i} = \sum_{j=1}^{m} a_{ij}\, y_j = Ay$$

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

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

$$\nabla_y (x^T A y) = \frac{\partial f}{\partial y_j} = \sum_{i=1}^{n} a_{ij} x_i = A^T x$$

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

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

$\nabla_A x^T A y$

$$= \frac{\partial f}{\partial A} = \begin{bmatrix} \frac{\partial f}{\partial a_{11}} & \cdots & \frac{\partial f}{\partial a_{1m}} \\ \\ \frac{\partial f}{\partial a_{n1}} & \cdots & \frac{\partial f}{\partial a_{nm}} \end{bmatrix}$$

$$= \begin{bmatrix} x_1 y_1 & \cdots & x_1 y_m \\ \\ x_n y_1 & \cdots & x_n y_m \end{bmatrix}$$

$$= x y^T$$

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

Q7 $f = x^T A x + b^T x$

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

$$\nabla_x f = \nabla_x (x^T A x) + \nabla_x (b^T x) \quad \text{--- (1)}$$

Consider $y = x^T A x$

$$y = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j$$

$$\frac{\partial y}{\partial x_1} = 2a_{11} x_1 + \sum_{j \neq 1}^{n} a_{1j} x_j + \sum_{i \neq 1}^{n} a_{i1} x_i$$

$$= \sum_{j=1}^{n} a_{1j} x_j + \sum_{i=1}^{n} a_{i1} x_i$$

$$\frac{\partial y}{\partial x_i} = (AX)_i + (A^T X)_i$$

$$\therefore \nabla_x (x^T A x) = (A + A^T) X \quad \text{--- (2)}$$

$$\nabla_x (b^T x) = b^T \quad \text{--- (3)}$$

substituting (2), (3) in (1)

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

3) e)
$$A = n \times m$$
$$B = m \times n$$

Let $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ \vdots & & & \\ a_{n1} & \cdots & \cdots & a_{nm} \end{bmatrix}$ & $B = \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ b_{m1} & & b_{mn} \end{bmatrix}$

$$AB = \begin{bmatrix} (a_{11}b_{11} + a_{12}b_{21} + \cdots + a_{1m}b_{m1}) & \cdots & \cdots \\ \vdots & (a_{21}b_{12} + a_{22}b_{22} + \cdots + a_{2m}b_{m2}) & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & & (a_{n1}b_{1n} + a_{n2}b_{2n} + \cdots a_{nm}b_{mn}) \end{bmatrix}$$

$$\therefore tr(AB)_i = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} b_{ji}$$

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

$$\frac{\partial y}{\partial A} = \begin{bmatrix} \frac{\partial y}{\partial a_{11}} & \cdots & \frac{\partial y}{\partial a_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial a_{n1}} & \cdots & \frac{\partial y}{\partial a_{nm}} \end{bmatrix}$$

$$= \begin{bmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nm} \end{bmatrix} = B^T$$

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

4) Deriving Least Squares

In the given problem, we have to minimize the exp^{r}

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

$\therefore$ our cost fun^{c} is

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

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

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

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

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

to find optimum W,

$\nabla_W L = 0$. using the given hints, we have,

$$\frac{1}{2}\left(0 - 2Y^TX + W(X^TX + X^TX)\right) = 0$$

$$Y^TX = WX^TX$$

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

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

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

$$\nabla_x f = \nabla_x (x^T A x) + \nabla_x (b^T x) \quad \text{—— (1)}$$

Consider $y = x^T A x$

$$y = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j$$

$$\frac{\partial y}{\partial x_1} = 2a_{11} x_1 + \sum_{j \neq 1}^{n} a_{1j} x_j + \sum_{i \neq 1}^{n} a_{i1} x_i$$

$$= \sum_{j=1}^{n} a_{1j} x_j + \sum_{i=1}^{n} a_{i1} x_i$$

$$\frac{\partial y}{\partial x_i} = (AX)_i + (A^T X)_i$$

$$\therefore \nabla_x (x^T A x) = (A + A^T) X \quad \text{—— (2)}$$

$$\nabla_x (b^T x) = b \quad \text{—— (3)}$$

substituting (2), (3) in (1)

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

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

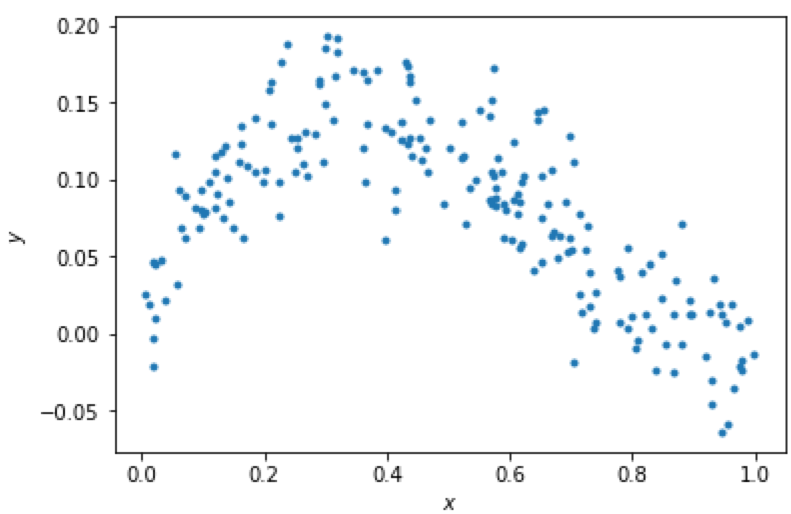
In [134]:
```
import numpy as np
import matplotlib.pyplot as plt

#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 [135]:
```
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[135]: <matplotlib.text.Text at 0x1126bf850>



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

(2) The distribution of the additive noise ϵ is Normal

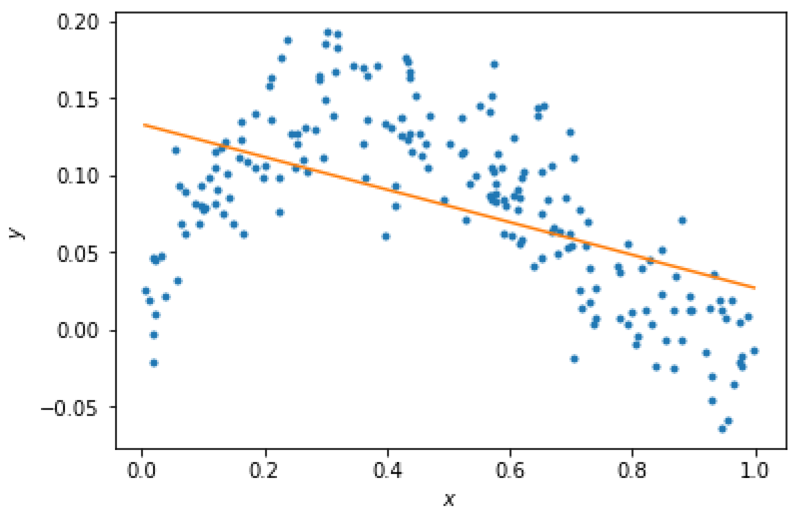
Fitting data to the model (5 points)

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

In [136]:
```
# 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 [137]:
```
# 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[137]: [<matplotlib.lines.Line2D at 0x112866fd0>]



QUESTIONS

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

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

ANSWERS

(1) The model is underfitting.

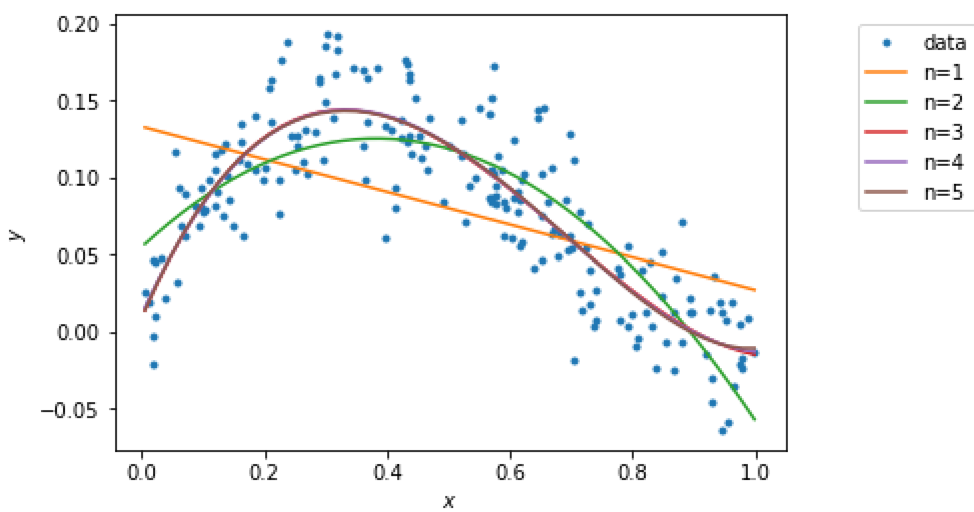
(2) We can improve the fitting by using an equation of higher order or degree i.e., to increase the dimensions of theta and X correspondingly.

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 [138]:
```
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):
    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].d
ot(y)))



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

In [139]:
```
# 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 [141]:
```
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.
for i in np.arange(N):

    training_errors.append((1/float(len(y))) * np.linalg.norm(y - thetas[i].do
t(xhats[i])))


pass

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

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

```
('Training errors are: \n', [0.0034496094622164849, 0.0023371908575540567, 0.
0020210892856459082, 0.0020205635025033215, 0.0020200840571032307])
```

QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

ANSWERS

(1) The polynomial of degree 5 has the best training error (i.e., the least training error.)

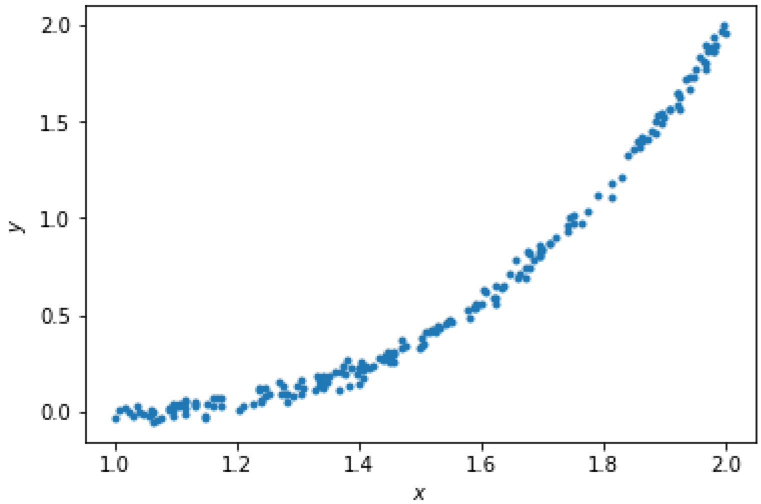
(2) This was expected because a polynomial of higher degree covers all the points in the training data thus decreasing the training error to its least value.

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 [144]:
```
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('$y$')
```

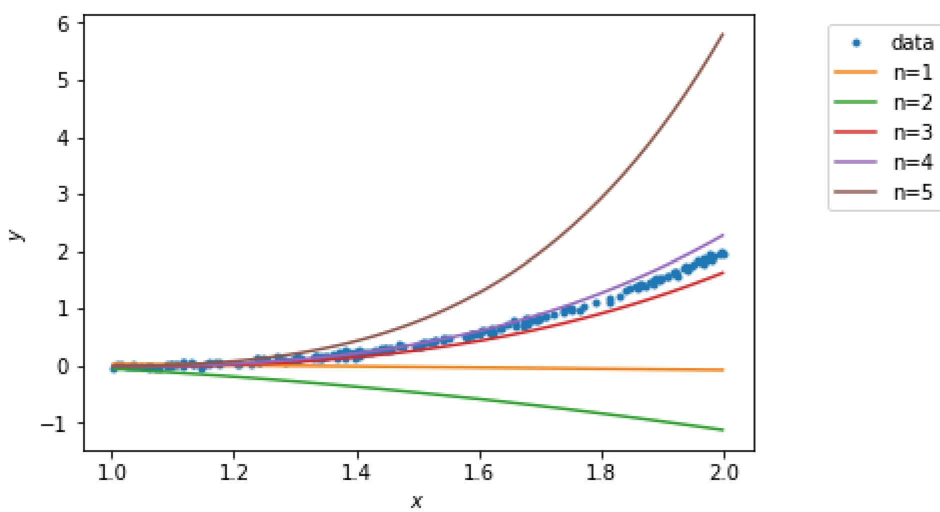
Out[144]: <matplotlib.text.Text at 0x112a28e50>



In [145]:
```
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 [146]:

```
# 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 [147]:
```
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/float(len(y))) * np.linalg.norm(y - thetas[i].dot
(xhats[i])))


pass

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

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

```
('Testing errors are: \n', [0.063585238792311649, 0.10324532058417471, 0.0125
01394138930177, 0.0077041434297982813, 0.10366055616645076])
```

QUESTIONS

(1) What polynomial has the best testing error?

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

ANSWERS

(1) Polynomial of degree 4 has the best Testing Error

(2) The polynomial of degree 5 is over fitting the training data. Although the training error is the least, the 5th degree polynomial does not generalize well as it tries to cover maximum number points in the training data.