HW 1

(1) (a)(i) $A\,A^T = I$, where A is a 2×2 square matrix

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

Then $A\,A^T = \begin{bmatrix} a^2+b^2 & ac+bd \\ ac+bd & c^2+d^2 \end{bmatrix}$, which must be I

→ So we get the following criteria:

$ac + bd = 0$ $\qquad a^2 + b^2 = c^2 + d^2 = 1$

Then a suitable choice for A would be:

$A = \begin{bmatrix} \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$ Now finding eigenvalues & eigenvectors:

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

$\det\left(\begin{bmatrix} -\frac{1}{\sqrt{2}}-\lambda & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}-\lambda \end{bmatrix}\right) = \left(-\frac{1}{\sqrt{2}}-\lambda\right)\left(\frac{1}{\sqrt{2}}-\lambda\right) - \left(\frac{1}{\sqrt{2}}\right)^2$

$= \lambda^2 - 1 = 0 \rightarrow \boxed{\lambda_1 = 1,\ \lambda_2 = -1}$

Finding eigenvectors $V_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$ for λ_1:

$(A - \lambda_1 I)\vec{V}_1 = 0$

$\begin{bmatrix} -\frac{1}{\sqrt{2}} - \lambda_1 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} - \lambda_1 \end{bmatrix}\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} \frac{-1-\sqrt{2}}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1-\sqrt{2}}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$-\left(\frac{1+\sqrt{2}}{\sqrt{2}}\right)x_1 + \frac{1}{\sqrt{2}}y_1 = 0$ $\qquad \frac{1}{\sqrt{2}}x_1 + \left(\frac{1-\sqrt{2}}{\sqrt{2}}\right)y_1 = 0$

$y_1 = (1+\sqrt{2})x_1$ $\qquad x_1 = (\sqrt{2}-1)\,y_1$

So $\vec{V}_1$ is of the form $k_1\begin{bmatrix} 1 \\ 1+\sqrt{2} \end{bmatrix}$, where k_1 is a const.

(1) (a) (i) (continued)

$(A - \lambda_2 I)\vec{v}_2 = 0$

$$\begin{bmatrix} -\frac{1}{\sqrt{2}} - \lambda_2 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} - \lambda_2 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -\frac{1}{\sqrt{2}} + 1 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} + 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \begin{matrix} (1) \\ (2) \end{matrix}$$

$$\frac{(\sqrt{2}-1)}{\sqrt{2}} x_2 + \frac{1}{\sqrt{2}} y_2 = 0$$

$$\rightarrow y_2 + (\sqrt{2}-1) x_2 = 0 \qquad (1)$$

$$y_2 = (1-\sqrt{2}) x_2$$

$$\frac{1}{\sqrt{2}} x_2 + \frac{1+\sqrt{2}}{\sqrt{2}} y_2 = 0$$

$$\rightarrow x_2 + (1+\sqrt{2}) y_2 = 0$$

$$x_2 = -(1+\sqrt{2}) y_2$$

$$y_2 = -\frac{1}{1+\sqrt{2}} x_2 \cdot \frac{1-\sqrt{2}}{1-\sqrt{2}} \qquad (2)$$

$$= -\frac{(1-\sqrt{2})}{-1} x_2$$

$$= (1-\sqrt{2}) x_2$$

So we get $\vec{v}_2 = k_2 \begin{bmatrix} 1 \\ 1-\sqrt{2} \end{bmatrix}$

The magnitudes of the eigenvalues are 1 and the eigenvectors are orthogonal to each other

① (a)(iii) $A v_i = \lambda_i v_i$

$\|A v_i\|^2 = \|\lambda_i v_i\|^2$

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

$v_i^T A^T A v_i = \|\lambda_i\|^2 \|v_i\|^2$ ($A^T A \to I$)

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

$\boxed{\|\lambda_i\|^2 = 1}$

(iii) By commutativity of dot product, $v_1 \cdot v_2 = v_2 \cdot v_1$

Multiply both sides by A:

$A\, v_1 \cdot v_2 = A v_2 \cdot v_1$ (eq 1)

Since $A\vec{v}_1 = \lambda_1 \vec{v}_1$ and $A\vec{v}_2 = \lambda_2 \vec{v}_2$, eq 1 becomes

$\lambda_1 v_1 \cdot v_2 = \lambda_2 v_2 \cdot v_1$

$\lambda_1 v_1 \cdot v_2 - \lambda_2 v_2 \cdot v_1 = 0$

$\lambda_1 v_1 \cdot v_2 - \lambda_2 v_1 \cdot v_2 = 0$

$v_1 \cdot v_2 (\lambda_1 - \lambda_2) = 0$

because $\lambda_1 \neq \lambda_2$, $v_1 \cdot v_2$ must be zero;

therefore, $\boxed{v_1 \perp v_2}$

(iv) If $\vec{v}$ is an eigenvector of A, then applying A to v will have the same result as multiplication by the corresponding eigenvalue.

If $\vec{V}$ is some linear combo of eigenvectors of A, then because they're orthogonal, the respective components of each eigenvector will be scaled by the corresponding eigenvalues.

(1) (b) (i) Singular value decomposition of A:

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

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

$= U\Sigma\Sigma^T U^T$, which is a square matrix

Which means the eigenvalues and eigenvectors of AA^T would be the singular values and singular vectors of A.

$A^TA = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T\Sigma V^T$, so the same would apply to A^TA.

(ii) The singular values are the eigenvalues of A^TA and AA^T, arranged in descending order.

(1) (c) (i) Two eigenvalues could be the same, False

(ii) Using $A\vec{v} = \lambda\vec{v}$: ←

$A(v_1 + v_2) = Av_1 + Av_2$

$Av_1 + Av_2 = \lambda_1 v_1 + \lambda_2 v_2$ because of

but since $\lambda_1 \neq \lambda_2$, we can't factor, so False

(iii) True

(iv) True

(v) $Av_1 + Av_2 = \lambda v_1 + \lambda v_2$ (assume $A \in \mathbb{R}^{2\times 2}$)

$A(v_1 + v_2) = \lambda(v_1 + v_2)$

by definition, $(v_1 + v_2)$ corresponds to an eigenvalue of A, True

(2) @ (i) Select coin from jar $\rightarrow P(H50) = \frac{1}{2} = P(H60)$

Find $P(H50 \mid T)$.

We know $P(T \mid H50) = 0.5$. By Bayes' Rule,

$$P(H50|T) = \frac{P(T|H50) \cdot P(H50)}{P(T)} = \frac{0.5 \cdot 0.5}{P(T)}$$

By law of Total probability,

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

So,

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

$$= \frac{(0.5)(0.5)}{(0.5)(0.5)+(0.4)(0.5)} = \frac{0.25}{0.45} \approx \boxed{0.556}$$

(ii) Replaced coin, so next coin picked still has $P(H50) = P(H60) = \frac{1}{2}$

Define sequence $\{T, H, H, H\}$ as the event A.

Assuming we don't care about order,

$$P[A|H50] = (0.5)^1 \cdot (0.5)^3 = 0.5^4 = \frac{1}{16} \approx 0.0625$$

$$P[A|H60] = (0.4)^1 \cdot (0.6)^3 = \frac{2 \cdot 3^3}{5^4} = \frac{54}{625} \approx 0.0864$$

Find $P(H50|A)$:

$$P(H50|A) = \frac{P(A|H50) \cdot P(H50)}{P(A)} = \frac{(\frac{1}{16} \cdot \frac{1}{2})}{P(A)}$$

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

$$\approx \frac{1}{16} \cdot \frac{1}{2} + \frac{54}{625} \cdot \frac{1}{2} \approx 0.07445$$

$$\rightarrow P(H50|A) = \boxed{0.419745}$$

(This result makes sense because the sequence has more heads. Therefore you'd be inclined to say the coin was weighted towards H).

(2) (a) (iii) Now we have $P(H50) = P(H55) = P(H60) = \frac{1}{3}$

10 flips → ~~Binomial(n, p)~~ Geom ($k=9$)

Let this sequence be known as B.

$P(B|H50) = (0.5)(0.5)^9$

$P(B|H55) = (0.45)(0.55)^9$

$P(B|H60) = (0.4)(0.6)^9$

$P(B) = P(B|H50)P(H50) + P(B|H55)P(H55) + P(B|H60) \cdot P(H60)$

$= (0.5)^{10} \cdot \frac{1}{3} + (0.45)(0.55)^9 \cdot \frac{1}{3} + (0.4)(0.6)^9 \cdot \frac{1}{3}$

≈ 0.00236

$P(H50|B) = \frac{P(B|H50) \cdot P(H50)}{P(B)} \approx 0.137931$

$P(H55|B) = \frac{P(B|H55) \cdot P(H55)}{P(B)} \approx 0.2927$

$P(H60|B) = \frac{P(B|H60) \cdot P(H60)}{P(B)} \approx 0.5694$

These add up to 1 so solution makes sense

(2) (b) $P(1 \mid \text{pregnant}) = 0.99$ $\quad P(\text{not preg}) = 0.99$

$P(1 \mid \text{not preg}) = 0.10$ $\quad \rightarrow P(\text{preg}) = 0.01$

Find $P(\text{preg} \mid 1)$: By Bayes Rule,

$$P(\text{preg} \mid 1) = \frac{P(1 \mid \text{preg}) \cdot P(\text{preg})}{P(1)}$$

By Total Probability,

$$P(1) = P(1 \mid \text{preg})\,P(\text{preg}) + P(1 \mid \text{not preg})\,P(\text{not preg})$$

$$= (0.99)\cdot(0.01) + (0.10)\cdot(0.99) = 0.1089$$

$$\rightarrow P(\text{preg} \mid 1) = \frac{0.99 \cdot 0.01}{0.1089} \simeq \boxed{0.09091}$$

This makes sense because the overall likelihood of being pregnant is quite low, so absent any other information (is the woman trying to get pregnant or is she, for example, not sexually active?) the test has low reliability in practice when used on "random" women. If you select from a population of women who are trying to get pregnant, $P(\text{preg} \mid \text{positive})$ is much higher for the same test, with fewer false positives.

(2) (c)

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad x_i\text{'s} \sim \text{iid}, \quad A \text{ \& } b \text{ deterministic}$$

$$E(A\vec{x} + b) = E(A\vec{x}) + E(b), \text{ by linearity of expectation}$$

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

$$= \sum_{j=1}^{k} A \cdot x_j p_j + b$$

$$= A \sum_{j=1}^{k} x_j p_j + b$$

$$= A\,E[\vec{x}] + b$$

(2) (d) $COV(Ax+b) = E[((Ax+b) - E(Ax+b))((Ax+b) - E(Ax+b))^T]$

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

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

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

$= A\,E((x - E(x))(x - E(x))^T)\,A^T$

$= A\,COV(X)\,A^T$

(3) @ $\nabla_x \underset{(1\times n)}{x^T} \underset{(n\times m)}{A} \underset{(m\times 1)}{y}$ First multiply Ay:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & & & \vdots \\ a_{n1} & \cdots & & a_{nm} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{m} a_{1j} y_j \\ \sum_{j=1}^{m} a_{2j} y_j \\ \vdots \\ \sum_{j=1}^{m} a_{nj} y_j \end{bmatrix}$$

So $x^T(Ay) = [x_1 \; x_2 \; x_3 \cdots x_n] \begin{bmatrix} \sum_{j=1}^{m} a_{1j} y_j \\ \sum_{j=1}^{m} a_{2j} y_j \\ \vdots \\ \sum_{j=1}^{m} a_{nj} y_j \end{bmatrix}$

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

$$\frac{\partial}{\partial x}\left(\sum_{i=1}^{n} \sum_{j=1}^{m} x_i a_{ij} y_j\right) = \left(\frac{\partial}{\partial x} \sum_{i=1}^{n} x_i\right)\left(\sum_{j=1}^{m} a_{ij} y_j\right)$$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial x_i} x_i \left(\sum_{j=1}^{m} a_{ij} y_j\right)$$

$$= \begin{bmatrix} \sum_{j=1}^{m} a_{1j} y_j \\ \sum_{j=1}^{m} a_{2j} y_j \\ \vdots \\ \sum_{j=1}^{m} a_{nj} y_j \end{bmatrix} = \boxed{Ay}$$

(3)(b) Find $\nabla_y x^T A y$

from (a), we have

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

$$\nabla_y f = \begin{bmatrix} \sum_{i=1}^{n} x_i a_{i1} \\ \sum_{i=1}^{n} x_i a_{i2} \\ \vdots \\ \sum_{i=1}^{n} x_i a_{im} \end{bmatrix} = \boxed{A^T x}, \text{ which is } (m \times 1)$$

$$= \begin{bmatrix} (x_1 a_{11} + x_2 a_{21} + \cdots + x_n a_{n1}) \\ (x_1 a_{12} + x_2 a_{22} + \cdots + x_n a_{n2}) \\ \vdots \\ (x_1 a_{1m} + x_2 a_{2m} + \cdots + x_n a_{nm}) \end{bmatrix}$$

$$= \underset{(m \times n)}{\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & & & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{bmatrix}} \underset{(n \times 1)}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}$$

(3) (c) Find $\nabla_A x^T A y$. From (a), we know that:

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

$$\nabla_A f = \begin{bmatrix} \frac{\partial f}{\partial a_{11}} & \frac{\partial f}{\partial a_{12}} & \cdots & \frac{\partial f}{\partial a_{1m}} \\ \frac{\partial f}{\partial a_{21}} & \frac{\partial f}{\partial a_{22}} & \cdots & \frac{\partial f}{\partial a_{2m}} \\ \vdots & & & \\ \frac{\partial f}{\partial a_{n1}} & \frac{\partial f}{\partial a_{n2}} & \cdots & \frac{\partial f}{\partial a_{nm}} \end{bmatrix}$$

$$= \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_m \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_m \\ \vdots & & & \\ x_n y_1 & x_n y_2 & \cdots & x_n y_m \end{bmatrix}$$ (which is $n \times m$)

$$= \boxed{x y^T} \quad (n \times m) \checkmark$$

(3)(d) Find $\nabla_x f$, where $f = x^T A x + b^T x$

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

$\nabla_x (x^T A x)$: $f_1(x) = x^T A x = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} x_i x_j$

- for $i=1, j=1$, the term is $a_{11} x_1^2$

$$\frac{\partial}{\partial x_1}(a_{11} x_1^2) = 2 a_{11} x_1$$

- for $i=1, j \neq 1$, the term is $a_{1j} x_1 x_j \rightarrow \frac{\partial}{\partial x_1}(a_{1j} x_1 x_j) = a_{1j} x_j$
- and $i \neq 1, j=1$, the term is $a_{i1} x_i x_1 \rightarrow \frac{\partial}{\partial x_1}(a_{i1} x_i x_1) = a_{i1} x_i$

$$\rightarrow \text{So } \frac{\partial f(x)}{\partial x_1} = 2 a_{11} x_1 + \sum_{j=2}^{m} a_{1j} x_j + \sum_{i=2}^{n} a_{i1} x_i$$

$$= \sum_{j=1}^{m} a_{1j} x_j + \sum_{i=1}^{n} a_{i1} x_i = (A\bar{x})_1 + (A^T \bar{x})_1$$

$$\rightarrow \frac{\partial f_1(x)}{\partial x} = (A + A^T) x$$

$$\nabla_x (b^T x) = \begin{bmatrix} \frac{\partial \sum_{i=1}^{n} b_i x_i}{\partial x_1} \\ \frac{\partial \sum_{i=1}^{n} b_i x_i}{\partial x_2} \\ \vdots \\ \frac{\partial \sum_{i=1}^{n} b_i x_i}{\partial x_n} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = b$$

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

(3) (e) $f = tr(AB)$, find $\nabla_A f$

A is $n \times m$, so to make AB square, B must be $m \times n$

$$AB = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ a_{21} & & \vdots \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ b_{m1} & \cdots & b_{mn} \end{bmatrix} = \begin{bmatrix} \sum_i a_{1i} b_{i1} & \cdots & \sum_i a_{1i} b_{in} \\ \vdots & & \vdots \\ \sum_i a_{ni} b_{i1} & \cdots & \sum_i a_{ni} b_{in} \end{bmatrix} \quad (n \times n)$$

call this matrix C

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

$$\nabla_A Tr(AB) = \begin{bmatrix} \frac{\partial}{\partial a_{11}} \sum\sum a_{ji} b_{ij} & \cdots & \frac{\partial}{\partial a_{1m}} \sum\sum a_{ji} b_{ij} \\ \vdots & & \\ \frac{\partial}{\partial a_{n1}} \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ji} b_{ij} & \cdots & \frac{\partial}{\partial a_{nm}} \sum\sum a_{ji} b_{ij} \end{bmatrix}$$

$$= \begin{bmatrix} b_{11} & b_{21} & \cdots & b_{m1} \\ \vdots & & & \vdots \\ b_{1n} & \cdots & & b_{mn} \end{bmatrix} = \boxed{B^T}$$

(4)

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

$$= \min_W \frac{1}{2} \| Y - WX \|_F^2, \text{ where } Y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix} \text{ and } X = \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(n)} \end{pmatrix}$$

$$= \min_W \frac{1}{2} \mathrm{Tr}\left[(Y - WX)^T (Y - WX)\right]$$

$$= \min_W \frac{1}{2} \mathrm{Tr}\left[Y^T Y - Y^T WX - X^T W^T Y + X^T W^T WX\right]$$

$$= \min_W \frac{1}{2}\left[\mathrm{Tr}(Y^T Y) - \mathrm{Tr}(Y^T WX) - \mathrm{Tr}(X^T W^T Y) + \mathrm{Tr}(X^T W^T WX)\right]$$

$$= \min_W \frac{1}{2}\left[\mathrm{Tr}(Y^T Y) - 2\mathrm{Tr}(WXY^T) + \mathrm{Tr}(X^T W^T WX)\right]$$

Now, take $\frac{\partial}{\partial W}$ and set $= 0$:

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

$$-YX^T + WXX^T = 0$$

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

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

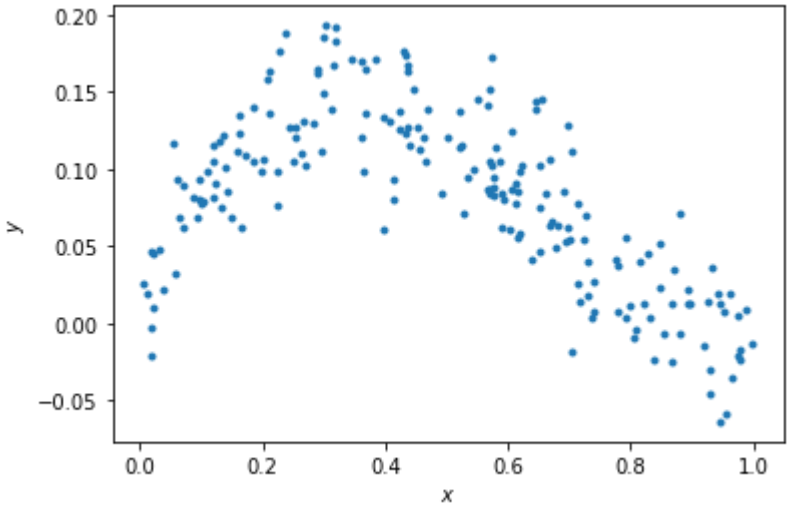
In [2]:
```
import numpy as np
import matplotlib.pyplot as plt

#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 [3]:
```
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[3]: 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) X is uniformly distributed in (0, 1).

(2) ϵ is normally distributed with μ = 0 and σ = 0.03.

Fitting data to the model (5 points)

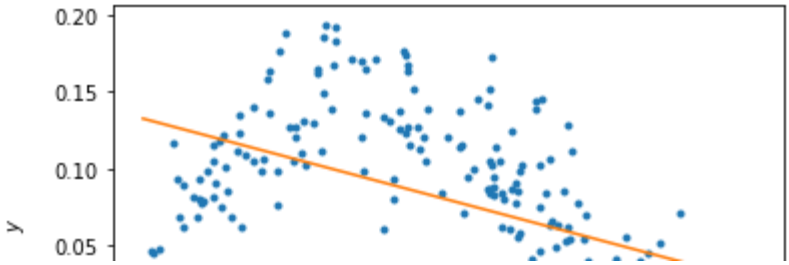
Here, we'll do linear regression to fit the parameters of a model $y = ax + b$.

In [26]:
```
# xhat = (x, 1)
xhat = np.vstack((x, np.ones_like(x)))


# GOAL: create a variable theta; theta is a numpy array whose elements are [a

print(xhat.shape)
print(xhat.T.shape)

#use normal equation to calculate theta
th = np.linalg.inv((xhat).dot(xhat.T)).dot(xhat.dot(y))
print(th)
print(th.shape)
```

```
(2, 200)
(200, 2)
[-0.10599633  0.13315817]
(2,)
```

In [32]:
```
# 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,:], th.dot(xs))
```

Out[32]:
```
[<matplotlib.lines.Line2D at 0x7f8186b53a00>]
```



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 under-fits the data.

(2) Include higher-order terms in the model by increasing the size of theta.

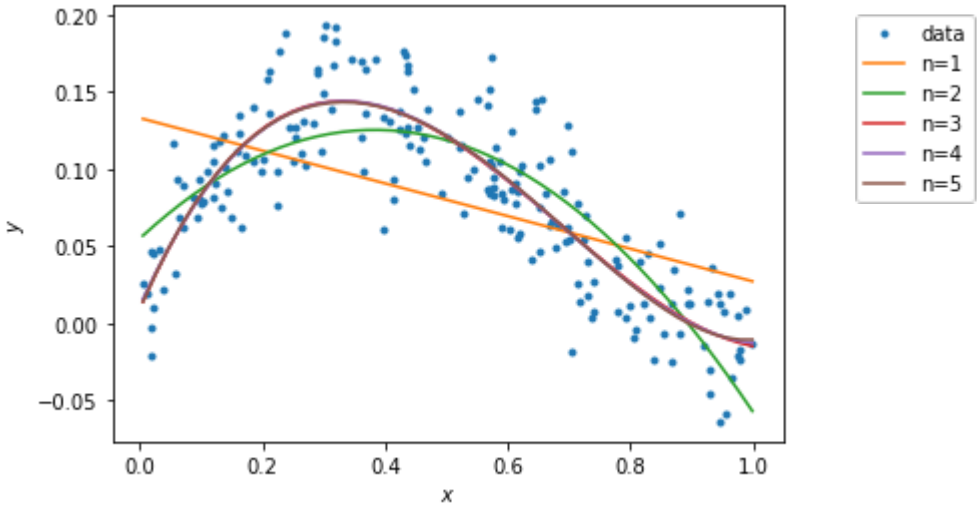
Fitting data to the model (10 points)

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 [35]:
```
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 polynomia
#   i.e., thetas[0] is equivalent to theta above.
#   i.e., thetas[1] should be a length 3 np.array with the coefficients of th
#   ... etc.

xhat = np.vstack((x, np.ones_like(x)))
theta = np.linalg.inv((xhat).dot(xhat.T)).dot(xhat.dot(y))

xhats.append(xhat)
thetas.append(theta)

for i in range(1, N):
    x_row = x**(i+1)
    xhat = np.vstack((x_row, xhat))
    xhats.append(xhat)
    print("xhat shape: {}".format((xhat.shape)))
    theta = np.linalg.inv((xhat).dot(xhat.T)).dot(xhat.dot(y))
    thetas.append(theta)
    print("theta shape: {}".format((theta.shape)))


#pass

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

```
xhat shape: (3, 200)
theta shape: (3,)
xhat shape: (4, 200)
theta shape: (4,)
xhat shape: (5, 200)
theta shape: (5,)
xhat shape: (6, 200)
theta shape: (6,)
```

In [36]:
```
# 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 [38]:
```
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 or

for i in range(0, N):
    training_loss = ((np.linalg.norm(y - thetas[i].dot(xhats[i])))**2) * 0.5
    training_errors.append(training_loss)

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

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

```
Training errors are:
 [0.2379961088362701, 0.1092492220926853, 0.08169603801105373, 0.081653537352
96978, 0.08161479195525295]
```

QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

ANSWERS

(1) The highest-order polynomial has the best training error.

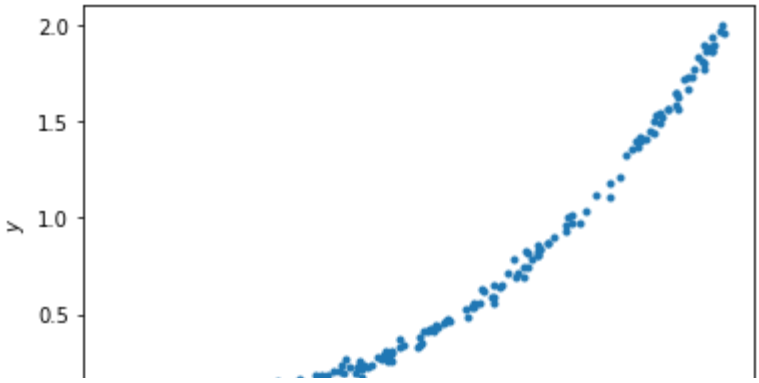
(2) It has the most degrees of freedom to fit each individual point in the training dataset, whether or not that reflects the noise in the dataset or the actual trends in the data.

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.

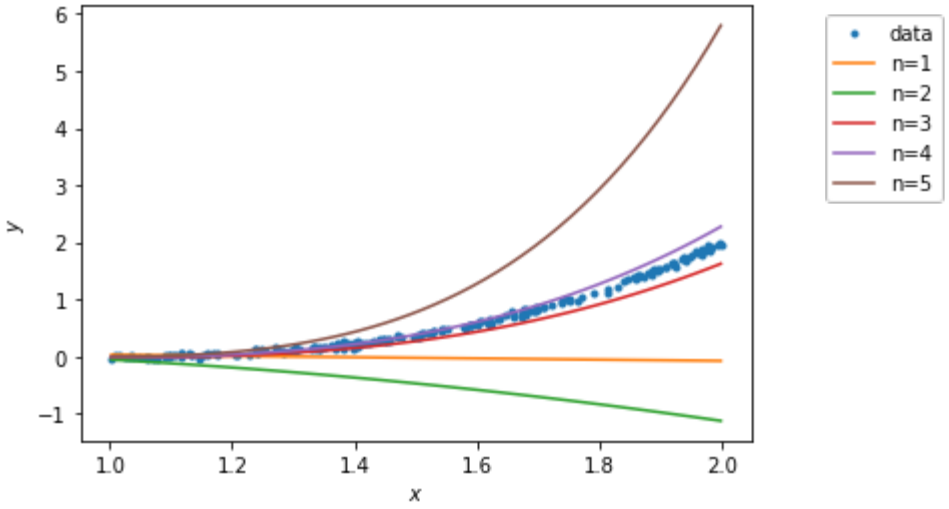
In [39]:
```
x = np.random.uniform(low=1, high=2, size=(num_train,))
y = x - 2*x**2 + x**3 + np.random.normal(loc=0, scale=0.03, size=(num_train,)
f = plt.figure()
ax = f.gca()
ax.plot(x, y, '.')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
```

Out[39]:
```
Text(0, 0.5, '$y$')
```



In [40]:
```
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 [41]:
```
# 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 [44]:
```
testing_errors = []

# print(xhats[0].shape)

# ==================== #
# 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 orde


for i in range(0, N):
    test_error = ((np.linalg.norm(y - thetas[i].dot(xhats[i])))**2) * 0.5
    testing_errors.append(test_error)

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

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

```
Testing errors are:
 [80.86165184550585, 213.1919244505791, 3.125697108408374, 1.187076521149622
6, 214.91021747012792]
```

QUESTIONS

(1) What polynomial has the best testing error?

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

ANSWERS

(1) The 4th degree polynomial has the lowest testing error.

(2) The 5th degree polynomial doesn't generalize well because it overfits - it finds patterns in the training set which do not correspond to patterns in the unseen test data with different noise or range.

In [ ]: