Ege Cetintas ECE C247

HOMEWORK #1

Question 1

(a) A is a square matrix, $AA^T = I$

i.)

$$A = \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} \\ 2/\sqrt{5} & -1/\sqrt{5} \end{bmatrix} \qquad A^T = \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} \\ 2/\sqrt{5} & -1/\sqrt{5} \end{bmatrix}$$

$$A.A^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Let's find eigenvalues and eigenvectors!

$$Av_i = \lambda_i v_i \qquad i = 1,2$$

since, $Av = \lambda v \Rightarrow \det(A - \lambda I) = 0$

$$\det\left(\begin{bmatrix} \frac{1}{\sqrt{5}} - \lambda & \frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} - \lambda \end{bmatrix} \right) = 0$$

$$\hookrightarrow \left(1/\sqrt{5} - \lambda \right)\left(-\frac{1}{\sqrt{5}} - \lambda \right) - \frac{2}{\sqrt{5}} \cdot \frac{2}{\sqrt{5}} = 0$$

$$\lambda^2 - \frac{1}{5} - \frac{4}{5} = 0 \Rightarrow \lambda^2 = 1 \qquad \boxed{\begin{matrix} \lambda_1 = 1 \\ \lambda_2 = -1 \end{matrix}} \rightarrow \text{eigenvalues}$$

* Let's find the eigenvectors:

Eigenvectors should satisfy the following:

$$(A - \lambda_1 I) v_1 = 0$$
$$(A - \lambda_2 I) v_2 = 0$$

For v_1:

$$\begin{bmatrix} 1/\sqrt{5} - 1 & \frac{2}{\sqrt{5}} \\ 2/\sqrt{5} & -\frac{1}{\sqrt{5}} - 1 \end{bmatrix} \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\left(1/\sqrt{5} - 1\right) v_{11} + \frac{2}{\sqrt{5}} v_{12} = 0 \rightarrow v_{12} = \left(\frac{1-\sqrt{5}}{\sqrt{5}}\right)\left(-\frac{\sqrt{5}}{2}\right) v_{11}$$

$$\frac{2}{\sqrt{5}} v_{11} - \left(\frac{1}{\sqrt{5}} + 1\right) v_{12} = 0 \qquad = \frac{\sqrt{5}-1}{2} v_{11}$$

$$v_{12} = \frac{2}{\sqrt{5}} \cdot \frac{\sqrt{5}}{(1+\sqrt{5})} v_{11}$$

$$= \frac{2}{1+\sqrt{5}} v_{11}$$

→ These are basically the same. So, we can pick one:

$$\boxed{v_{12} = \frac{\sqrt{5}-1}{2} v_{11}}$$

$$v_1 = \begin{bmatrix} v_{11} \\ \frac{\sqrt{5}-1}{2} v_{11} \end{bmatrix}$$

⇒ We can further normalize v_1:
and we would find that:

$$v_{11} = \begin{bmatrix} 0.8507 \\ 0.5257 \end{bmatrix}$$

↳ for the other eigenvector:

$$(A-\lambda_2 I)v_2 = 0$$

$$\begin{bmatrix} 1/\sqrt{5}+1 & 2/\sqrt{5} \\ 2/\sqrt{5} & -1/\sqrt{5}+1 \end{bmatrix}\begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\left(\frac{1}{\sqrt{5}}+1\right)v_{21} + \frac{2}{\sqrt{5}}v_{22} = 0 \qquad v_{22} = \left(\frac{-\sqrt{5}}{2}\right)\left(\frac{1+\sqrt{5}}{\sqrt{5}}\right)v_{21}$$

$$v_{22} = \left(\frac{-1-\sqrt{5}}{2}\right)v_{21}$$

$$v_2 = \begin{bmatrix} v_{21} \\ \left(\frac{-1-\sqrt{5}}{2}\right)v_{21} \end{bmatrix}$$

. We can further normalize v_2 and find not

$$v_2 = \begin{bmatrix} 0.5257 \\ -0.8507 \end{bmatrix}$$

* I observe that eigenvectors are <u>orthogonal</u> to each other and the eigenvalues has a <u>norm</u> of ± 1.

<u>ii.</u> Show that A has eigenvalues with norm ± 1.

$$Av = \lambda v \qquad \text{where } v \text{ is the eigenvector}$$

we can write

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

$$\|Av\|^2 = (Av)^T(Av)$$

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

since $A^T A = I$

So, the expression becomes

$$\|Vx\|^2 = |\lambda|^2 \|y\|^2$$

hence, $|\lambda|^2 = 1$ so $|\lambda| = 1$.

This states that the norm of the eigenvalue should be 1.

iii) Show that eigenvectors of A are orthogonal:

$Av_1 = \lambda_1 v_1$ v_1, v_2 are eigenvectors

$Av_2 = \lambda_2 v_2$ λ_1, λ_2 are corresponding eigenvalues.

We're trying to prove that $v_1 \cdot v_2 = 0$, dot product to vector multip.

$$\lambda_1 (v_1 \cdot v_2) = (\lambda_1 v_1) \cdot v_2 = (Av_1) \cdot v_2 = (Av_1)^T v_2$$

$$= v_1^T A^T v_2 \quad \text{(A is symmetric)}$$

$$= v_1^T A v_2 \quad \text{(From eigenvalue.)}$$

$$= v_1^T \lambda_2 v_2 = \lambda_2 v_1^T v_2 = \lambda_2 (v_1 \cdot v_2)$$

to dot product

So, $\lambda_1 (v_1 \cdot v_2) = \lambda_2 (v_1 \cdot v_2) \Rightarrow (\lambda_1 \lambda_2$

Since λ_1 and λ_2 are distinct and $\lambda_1 \neq \lambda_2$

$v_1 \cdot v_2 = 0$. Thus, v_1 and v_2 are orthogonal.

or $v_1^T v_2$

Q1

Q. iv | In words, describe what may happen to vector x under the transformation Ax.

(5)

In the previous part, we've shown that the eigenvectors (v_1, v_2) are orthogonal. Thus any input vector x can be written in terms of these orthogonal vectors:

$$x = \alpha_1 v_1 + \alpha_2 v_2$$

When the input vector (x) is multiplied by A:

$$Ax = \alpha_1 A v_1 + \alpha_2 A v_2 = \alpha_1 \lambda_1 v_1 + \alpha_2 \lambda_2 v_2$$

where λ_1, λ_2 are eigenvalues.

We can see that x vector are mapped to the $v_1 \perp v_2$ space. In other word, it will be either rotated or reflected.

Q1. b. i | Let A be a matrix.

SVD of A is:

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

The colums of U are called the left singular vectors of A.
The colums of V " " " right " " "

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

* The right singular vectors of A are the eigenvectors of A^TA.

Q1.b.ii]

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

Q1.c.i] "Every linear operator in an n-dimensional vector space has n-distinct eigenvalues."

False → we can come up with linear operators with no eigenvalues.

c.ii]: "A non-zero sum of two eigenvectors of a matrix A is an eigenvector"

False.

c.iii]: "If a matrix A has the positive semidefinite property, i.e, $x^TAx \geq 0$ for all x, then its eigenvalues must be non-negative"

True. all eigenvalues of A are nonnegative.

c.iv]: "The rank of a matrix can exceed the number of distinct non-zero eigenvalues":

True: However, the number of non-zero eigenvalues is at most the rank of matrix.

Q1

C.V1: "A non-zero sum of two eigenvectors of a matrix A corresponding to the same eigenvalue λ is always an eigenvector"

TRUE: Let's have a look at a general example.

$A v_i = \lambda v_i$ Let's say we have another vector u which is $u = \sum v_i$

Let's look at the eigenvectors of A.

$$Au = A \sum v_i = \sum_i A v_i = \sum_i \lambda v_i$$

$$= \lambda \sum_i v_i$$

$$= \lambda u$$ → which states that the summation is also an eigenvector

Q2

(a)

i) $P(H50 \mid T) = ?$

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

Let's find $P(T)$:

$$P(T) = P(T \mid H50)P(H50) + P(T \mid H60)P(H60)$$
$$= \frac{1}{2} \cdot \frac{1}{2} + \frac{4}{10} \cdot \frac{1}{2} = \frac{9}{20}$$

$$P(H50 \mid T) = \frac{\frac{1}{2} \cdot \frac{1}{2}}{\frac{9}{20}} = \frac{1}{4} \cdot \frac{20}{9} = \frac{5}{9} \approx 0.556$$

So, the posterior probability is $\boxed{0.556}$ (appx.)

ii) $P(H50 \mid THHH) = ?$

We can rewrite:

$$P(H50 \mid THHH) = \frac{P(THHH \mid H50)\,P(H50)}{P(THHH)}$$

$$P(THHH) = P(THHH \mid H50)P(H50) + P(THHH \mid H60)P(H60)$$
$$= \left(\frac{1}{2}\right)^4\left(\frac{1}{2}\right) + \left(\frac{2}{5}\right)\left(\frac{3}{5}\right)^3\left(\frac{1}{2}\right) = \frac{1}{32} + \frac{27}{625}$$

$$P(H50 \mid THHH) = \frac{(1/2)^4 \cdot 1/2}{1/32 + \frac{27}{625}} = \frac{1}{1 + \frac{27 \times 32}{625}} = \frac{625}{625 + 864}$$

$$\approx 0.4197$$

(iii) $P(H50 \mid 9H, 1T) = ?$

$$P(H50 \mid 9H, 1T) = \frac{P(9H, 1T \mid H50)\, P(H50)}{P(9H, 1T)}$$

$$P(9H, 1T) = P(9H, 1T \mid H50) P(H50) + P(9H, 1T \mid H55) P(H55) + P(9H, 1T \mid H60) P(H60)$$

$$= (0.5)^9 (0.5) \cdot (1/3) + (0.55)^9 (0.45) (1/3) + (0.6)^9 (0.4)^1 (1/3)$$

$$\approx 0.0024$$

$$P(H50 \mid 9H, 1T) = \frac{(0.5)^9 (0.5) (1/3)}{(0.0024)} \approx \boxed{0.1379}$$

$$P(H55 \mid 9H, 1T) = \frac{(0.55)^9 (0.45) (1/3)}{(0.0024)} \approx \boxed{0.2927}$$

$$P(H60 \mid 9H, 1T) = \frac{(0.6)^9 (0.4) (1/3)}{(0.0024)} \approx \boxed{0.5694}$$

2.6

$P(T = \text{Positive} \mid W = \text{Pregnant}) = 0.99$

$P(T = \text{Positive} \mid W = \text{NOT Pregnant}) = 0.1$

$P(W = \text{Pregnant}) = 0.01$

What is $P(W = \text{Pregnant} \mid T = \text{Positive}) = ?$

$$P(W = \text{Pregnant} \mid T = \text{Positive}) = \frac{P(T = \text{Positive} \mid W = \text{Pregnant}) P(W = \text{Preg})}{P(T = \text{Positive})}$$

$$P(T = \text{Positive}) = P(T = \text{Positive} \mid W = \text{Pregnant}) P(W = \text{Pregnant}) + P(T = \text{Positive} \mid W = \text{NOT pregnant}) P(W = \text{NOT Pregnant})$$

$$= 0.99 \cdot 0.01 + 0.1 \cdot 0.99$$

$$= 0.1089$$

$$P(W = \text{Pregnant} \mid T = \text{Positive}) = \frac{0.99 \cdot 0.01}{0.1089} = \boxed{0.0909}$$

From this, we conclude that only 9.1% of the times the women are pregnant when the test results in positive. This makes sense because $P(\text{Positive} \mid \text{NOT pregnant})$ is 0.1 which is quite high considering most of the time women are not pregnant.

2.c): $x = \begin{bmatrix} x_1 \\ x_2 \\ \\ \\ x_n \end{bmatrix}$

$E(Ax+b) = ?$

$E[x] = \sum_n x P(x)$

$E[Ax+b] = \sum_n (Ax+b)P(x) = \sum_n Ax P(x) + b\sum_n P(x)$

($\sum_n P(x) = 1$)

$= A\sum_n x P(x) + b$

$$E[Ax+b] = \boxed{A E[x] + b}$$

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

$cov(Ax+b) = ?$

$E[Ax+b] = AE[x]+b$

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

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

$$= \boxed{A\, cov(x)\, A^T}$$

QUESTION 3

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

$\nabla_x x^T A y$?

$x^T A y$

$1 \times n \quad n \times m \quad m \times 1$

$$x^T A y = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} a_{11}y_1 & a_{12}y_2 & \cdots & a_{1m}y_m \\ \vdots & & & \\ a_{n1}y_1 & a_{n2}y_2 & \cdots & a_{nm}y_m \end{bmatrix}$$

$$= x_1 [a_{11}y_1 + \cdots + a_{1m}y_m] + \cdots + x_n [a_{n1}y_1 + \cdots + a_{nm}y_m]$$

$$\nabla_x \overbrace{x^T A y}^{f} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} a_{11}y_1 + \cdots + a_{1m}y_m \\ \vdots \\ a_{n1}y_1 + \cdots + a_{nm}y_m \end{bmatrix} = Ay$$

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

Q3.b : $\nabla_y x^T A y$

From the derivations of the previous part

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

$$\begin{bmatrix} (x_1 a_{11} + \cdots + x_n a_{n1}) & \cdots & (x_1 a_{1m} + \cdots + x_n a_{nm}) \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

$$\nabla_y \overbrace{x^T A y}^{f} = \begin{bmatrix} \frac{\partial f}{\partial y_1} \\ \vdots \\ \frac{\partial f}{\partial y_m} \end{bmatrix} =$$

$$x^T A y = y_1 \left[x_1 a_{11} + \cdots + x_n a_{n1} \right] + \cdots + y_m \left[x_1 a_{1m} + \cdots + x_n a_{nm} \right]$$

$$\nabla_y x^T A y = \begin{bmatrix} x_1 a_{11} + \cdots + x_n a_{n1} \\ \vdots \\ x_1 a_{1m} + \cdots + x_n a_{nm} \end{bmatrix} = (x^T A)^T = \boxed{A^T x}$$

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

Q3.c : $\nabla_A x^T A y$? $A \in \mathbb{R}^{n \times m}$ $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$

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

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

$$= \underbrace{\begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}}_{1 \times n} \begin{bmatrix} a_{11}y_1 + a_{12}y_2 + \cdots + a_{1m}y_m \\ \\ a_{n1}y_1 + a_{n2}y_2 + \cdots + a_{nm}y_m \end{bmatrix}$$

$n \times 1$

$$= x_1[a_{11}y_1 + a_{12}y_2 + \cdots + a_{1m}] + \cdots + x_n[a_{n1}y_1 + a_{n2}y_2 + \cdots + a_{nm}y_m]$$

$$= x_1 y_1 a_{11} + x_1 y_2 a_{12} + \cdots + x_n y_1 a_{n1} + \cdots + x_n y_m a_{nm}$$

$$\nabla_A x^T A y = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_m \\ \vdots & & & \\ x_n y_1 & & & x_n y_m \end{bmatrix} = X Y^T$$

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

Q3.d: $A \in \mathbb{R}^{n \times n}$

$f = x^T A x + b^T x$ what is $\nabla_x f$?

$$f = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \underbrace{\begin{bmatrix} a_{11} & \cdots & a_{n1} \\ a_{n1} & & a_{nn} \end{bmatrix}}_{n \times n} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 & \cdots & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$f = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \begin{bmatrix} a_{11}x_1 + \cdots + a_{n1}x_n \\ \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n \end{bmatrix} + \left(b_1 x_1 + \cdots + b_n x_n\right)$$

$$f = x_1\left(a_{11}x_1 + \cdots + a_{n1}x_n\right) + \cdots + x_n\left(a_{n1}x_1 + \cdots + a_{nn}x_n\right) + \left(b_1 x_1 + \cdots + b_n x_n\right)$$

$$= a_{11}x_1x_1 + a_{12}x_1x_2 + \cdots + a_{nn}x_nx_n + \left(b_1x_1 + \cdots + b_nx_n\right)$$

$$\nabla_x f = \begin{bmatrix} 2a_{11}x_1 + (a_{21} + a_{12})x_2 + \cdots + (a_{1n} + a_{n1})x_n + b_1 \\ \vdots \\ (a_{1n} + a_{n1})x_1 + \cdots + 2a_{nn}x_n + b_n \end{bmatrix}$$

$$\boxed{\nabla_x f = Ax + A^T x + b}$$ Provided that $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$

Q3.e: $f = tr(AB)$ what is $\nabla_A f$?

let's assume $A \in R^{m \times n}$, $B \in R^{n \times m}$

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

$AB \in R^{m \times m}$ $\rightarrow$ AB should be a square matrix in order to have a $tr(AB)$!

$$tr(AB) = AB_{11} + AB_{22} + \cdots + AB_{mm}$$

$$AB_{11} = a_{11}b_{11} + a_{12}b_{21} + \cdots + a_{1n}b_{n1}$$

$$AB_{mm} = a_{m1}b_{1m} + a_{m2}b_{2m} + \cdots + a_{mn}b_{nm}$$

$$\nabla_A f = \begin{bmatrix} \frac{\partial f}{\partial a_{11}} & \cdots & \\ \vdots & & \\ & & \frac{\partial f}{\partial a_{nm}} \end{bmatrix} = \underbrace{\begin{bmatrix} b_{11} & b_{21} & \cdots & b_{n1} \\ \vdots & & & \\ b_{1m} & & \cdots & b_{nm} \end{bmatrix}}_{m \times n} = B^T$$

So,

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

Question 4: $\hat{y} = Wx$

$(x^{(i)}, y^{(i)})$ from $i = 1, \dots, n$

$$\min_W \frac{1}{2} \sum_{i=1}^{n} \| y^{(i)} - Wx^{(i)} \|^2$$ Desire the optimal W.

Hint: $\frac{\partial tr(WA)}{\partial W} = A^T$ and $\frac{\partial tr(WAW^T)}{\partial W} = WA^T + WA$

* $x \in R^k$, $W \in R^{m \times k}$ $(m \geq 1)$ and $\hat{y} \in R^m$.

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

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

$$(y^{(i)T} - x^{(i)T} W^T)(y^{(i)} - Wx^{(i)})$$

$$y^{(i)T} y^{(i)} - y^{(i)T} Wx^{(i)} - x^{(i)T} W^T y^{(i)} + x^{(i)T} W^T W x^{(i)}$$

$y^{(i)T} Wx^{(i)} = (y^{(i)T} Wx^{(i)})^T = x^{(i)T} W^T y^{(i)}$ as it's scalar.

→ We only care about terms that include 'W'. So

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

$$Y = \begin{bmatrix} | & & | \\ y^{(1)} & \cdots & y^{(n)} \\ | & & | \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} | & & | \\ x^{(1)} & \cdots & x^{(n)} \\ | & & | \end{bmatrix}$$

With these matrices (X and Y), we can write summation traces of matrix multiplication:

$$\rightarrow f = \sum_{i=1}^{n} \left[-\mathrm{tr}\left(y^{(i)T} W x^{(i)}\right) + \frac{1}{2}\mathrm{tr}\left(x^{(i)T} W^T W x^{(i)}\right) \right]$$

$$= \sum_{i=1}^{n} \left[-\mathrm{tr}\left(W x^{(i)} y^{(i)T}\right) + \frac{1}{2}\mathrm{tr}\left(W x^{(i)} x^{(i)T} W^T\right) \right]$$

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

$$= -\mathrm{tr}\left(W X Y^T\right) + \frac{1}{2}\mathrm{tr}\left(W X X^T W^T\right)$$

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

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

$$\boxed{W = YX^T(XX^T)^{-1}} \longrightarrow \text{Optimal Solution.}$$

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
```

Data generation

For any example, we first have to generate some appropriate data to use. The following cell generates data according to the model: $y = x - 2x^2 + x^3 + \epsilon$

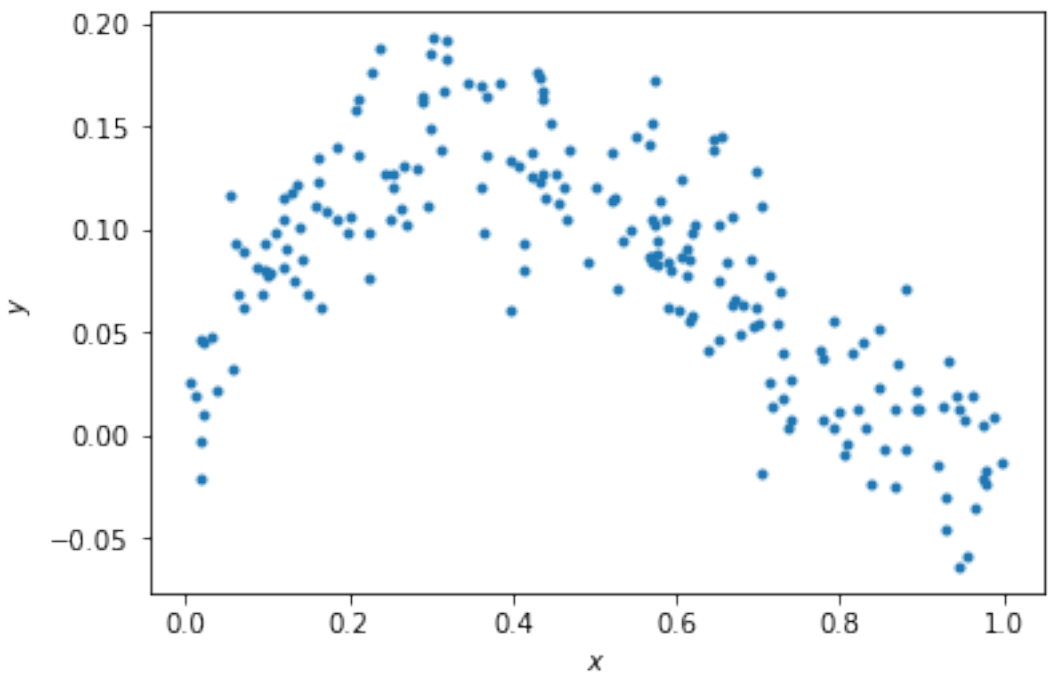
In [2]:

```
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) X is generated from a uniform distribution between 0 and 1. In total, 200 Points are generated.

(2) ϵ is generated from a normal (Gaussian) distribution with mean 0 and standard deviation 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]

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

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

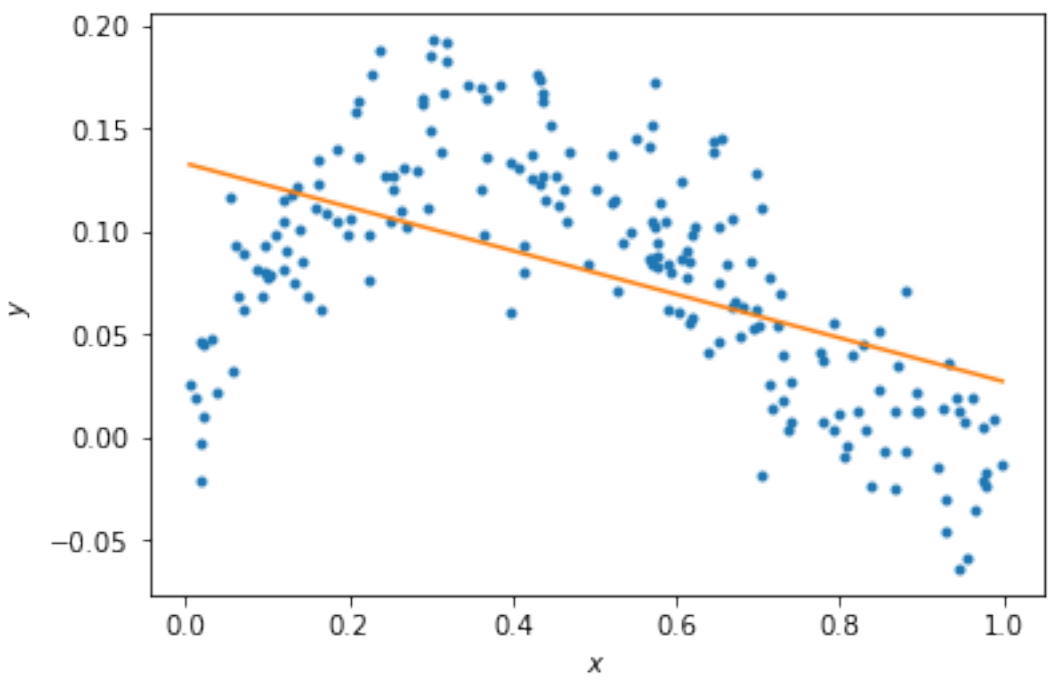
In [4]:

```
# 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 0x10f09f358>]
```



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.

(2) In order to improve the fitting, we need to increase the order of the polynomial that we are using to fit the data.

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.

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

for i in range(N-1):
    xhats.append(np.vstack((np.power(x,i+2), xhats[i])))
    thetas.append(np.dot(np.dot(np.linalg.inv(np.dot(xhats[i+1],xhats[i+1].tra
nspose())),xhats[i+1]),y))

pass

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

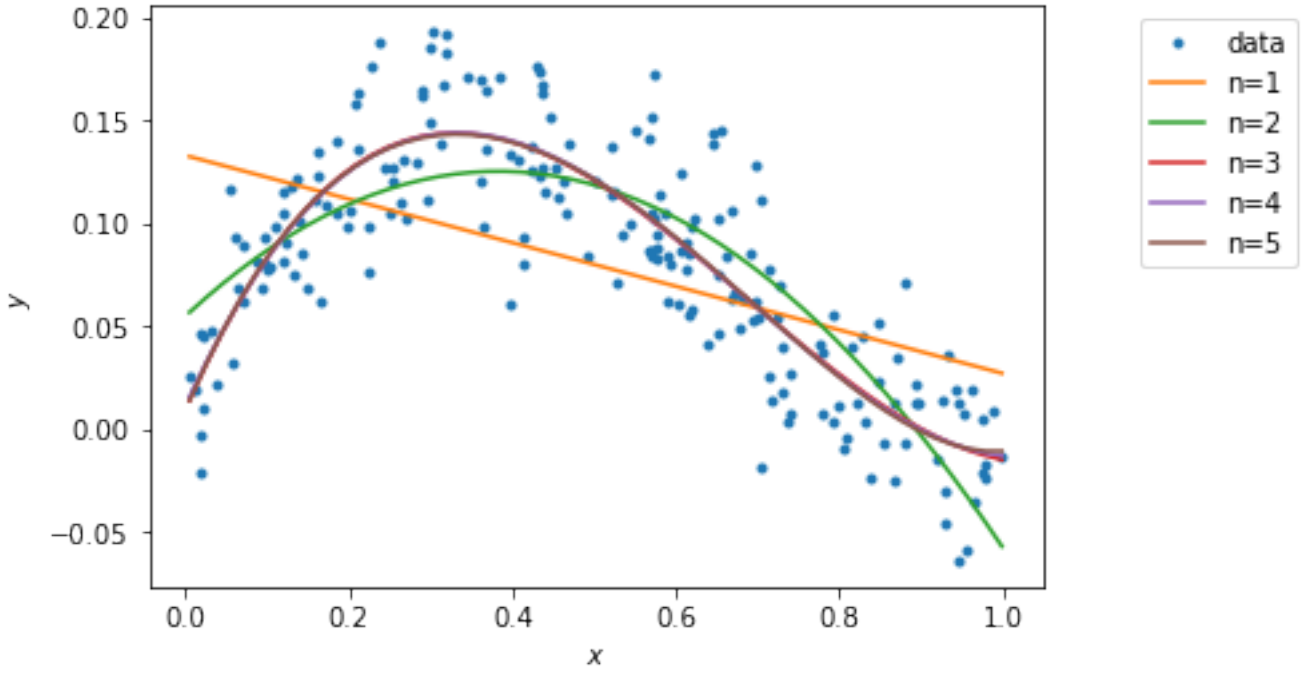
In [6]:

```
# 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:

$$L(\theta) = \frac{1}{2} \sum_j (\hat{y_j} - y_j)^2$$

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 ord
er i+1.

for i in range(N):
    diffY = y-thetas[i].dot(xhats[i])
    #print(diffY.T.dot(diffY))
    training_errors.append((diffY.T.dot(diffY))/2)
pass




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


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

```
Training errors are:
 [0.23799610883627009, 0.10924922209268531, 0.08169603801105374, 0
.08165353737352969804, 0.08161479195525295 2]
```

QUESTIONS

(1) Which polynomial model has the best training error?

(2) Why is this expected?

ANSWERS

(1) The polynomial with the highest order (5th order) has the best training error (meaning, it has the lowest error) as expected

(2) This is expected because when we are using an higher-order polynomial, it can overfit the training data thus minimizing the training error. Another way to look at it is that higher-order polynomials still includes the lower-order polynomials as a part of the solution. It means that one-order higher polynomial will, for sure, have a better training error.

Generating new samples and testing error (5 points)

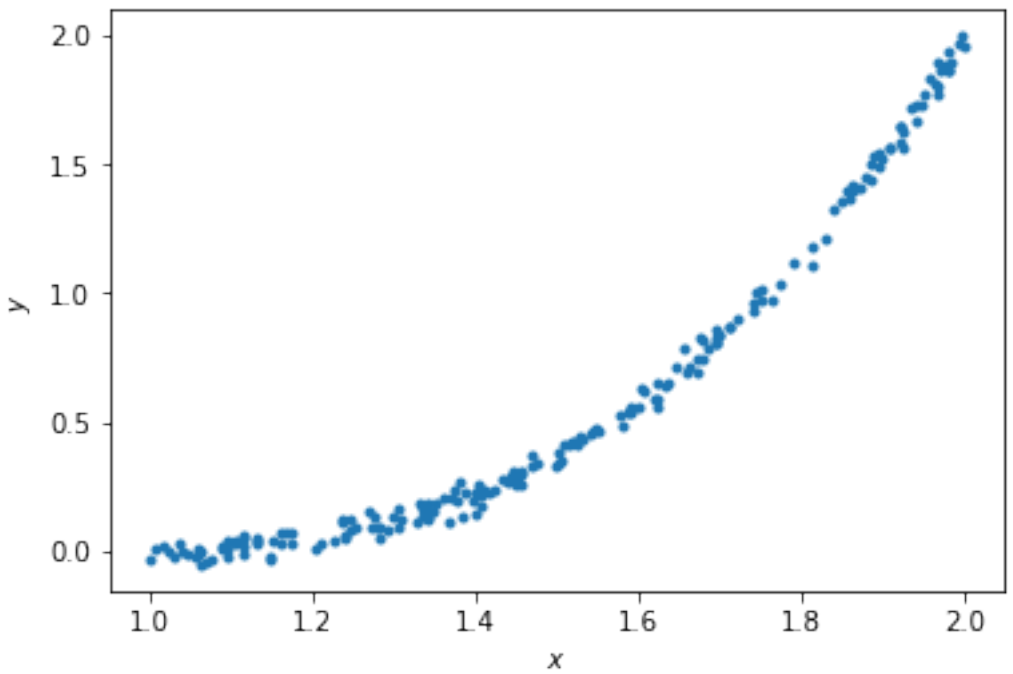
Here, we'll now generate new samples and calculate the testing error of polynomial models of orders 1 to 5.

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)
```

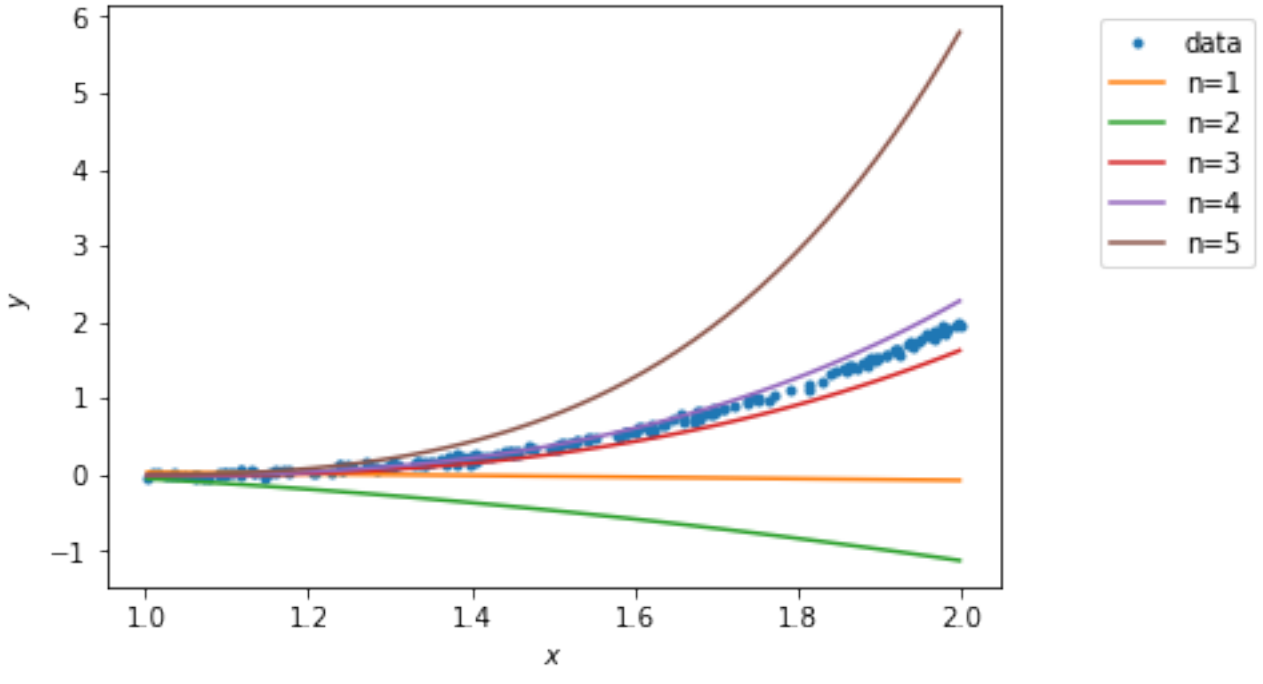
In [10]:

```
# 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 [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.
for i in range(N):
    diffY = y-thetas[i].dot(xhats[i])
    #print(diffY.T.dot(diffY))
    testing_errors.append((diffY.T.dot(diffY))/2)
pass

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

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

```
Testing errors are:
 [80.861651845505847, 213.1919244505786, 3.1256971083280094, 1.187
0765196620086, 214.91021807641894]
```

QUESTIONS

(1) Which polynomial model has the best testing error?

(2) Why does the order-5 polynomial model not generalize well?

ANSWERS

(1) The order-4-polynomial model (n=4) has the best testing error.

(2) The order-5-polynomial model overfits the training data. Thus, it is not able to generalize well. We also know that the actual data (y) is a order-3-polynomial with a normal noise and we observe that both order 3 and 4 polynomials perform smoothly on the data. However, order-5 polynomial overfits the training data thus it does not generalize for a larger test data.