ECE C247, Winter 2022
Neural Networks and Deep Learning, UCLA

Homework #1

Sidarth Srinivasan, 005629203

Question 1 : Linear Algebra refresher

1 a) Given that A is a square matrix and that $A.A^T = I$.

let $A = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$, so $A.A^T = \begin{bmatrix} x^2+y^2 & xz+yw \\ zx+wy & z^2+w^2 \end{bmatrix}$

Since $A.A^T = I$, we can infer that $x = -1/\sqrt{2}$

$y = z = w = 1/\sqrt{2}$ (Better to construct a matrix that is symmetric)

Hence $A = \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$

We can now solve for its Eigenvalues & Eigenvectors :-

$$A - \lambda I = \begin{bmatrix} -1/\sqrt{2} - \lambda & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} - \lambda \end{bmatrix}$$

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

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

$$\Rightarrow \quad 1/2 - \lambda^2 + 1/2 = 0$$

$$\Rightarrow \quad \lambda^2 = 1 \qquad \Rightarrow \quad \lambda = \pm 1$$

Now, lets solve for the eigenvector corresponding to $\lambda = 1$.

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

$$\Rightarrow \begin{pmatrix} -1/\sqrt{2} - 1 & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} - 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

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

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

We can set $x_2 = 1$, and solve for x_1

Hence $x_1 = \sqrt{2} - 1$

Hence the eigenvector corresponding to $\lambda = 1$ is

$v_1 = \begin{bmatrix} \sqrt{2} - 1 \\ 1 \end{bmatrix}$ and we can normalize this vector to obtain

$v_1 = \begin{pmatrix} 0.383 \\ 0.924 \end{pmatrix}$ is the eigenvector corresponding to the Eigenvalue $\lambda = 1$.

Now, lets compute the eigenvector corresponding to $\lambda = -1$

$(A + I)x = 0$

$$\Rightarrow \begin{pmatrix} -1/\sqrt{2} + 1 & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} + 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

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

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

let's set $x_2 = 1$, hence $x_1 = -(\sqrt{2}+1)$

Hence the Eigenvector $x = \begin{bmatrix} -(\sqrt{2}+1) \\ 1 \end{bmatrix}$

Normalizing this vector to obtain :-

$$v_2 = \begin{bmatrix} -0.924 \\ 0.383 \end{bmatrix}$$

Hence the Eigenvectors corresponding to values $\lambda = 1$ and -1 are:

$$v_1 = \begin{bmatrix} 0.383 \\ 0.924 \end{bmatrix} \quad \text{and} \quad v_2 = \begin{bmatrix} -0.924 \\ 0.383 \end{bmatrix}$$

* We notice that the Eigenvalues are 1
* We also notice that the Eigenvectors are Orthonormal / Orthogonal.

1 a) (ii) Let's consider the eigenvalues to be λ.

So, $Ax = \lambda x$.

$$(\lambda x)^T (\lambda x) = (Ax)^T (Ax)$$

$$|\lambda|^2 x^T x = x^T A^T A x$$

$$|\lambda|^2 = 1$$

1 a) (iii) Let's consider λ_1, λ_2 be the eigenvalues of A and let v_1, v_2 be its corresponding eigenvectors, such that $\lambda_1 \neq \lambda$

So, $Av_1 = \lambda_1 v_1$ and $Av_2 = \lambda_2 v_2$

$$\Rightarrow (Av_1)^T (Av_2) = (\lambda_1 v_1)^T (\lambda_2 v_2)$$

$$\Rightarrow \quad v_1^T . A^T . A . v_2 = \lambda_1 \lambda_2 v_1^T v_2$$

$$\Rightarrow \quad (\lambda_1 \lambda_2 - 1) v_1^T . v_2 = 0$$

Since we know that $\lambda_1 \neq \lambda_2$ and $|\lambda_{1,2}| = 1$, hence $(\lambda_1 \lambda_2 - 1) \neq 0$

and $v_1^T . v_2 = 0$

Hence Eigenvectors corresponding to distinct eigenvalues are Orthogonal.

1a)(iv) under the Transformation, the length of vector will remain unchanged as the vector will either to subjected to a rotation / reflection.

For example in our case from 1a)(i) we know that :-

$$A = \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \quad , \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Hence under transformation becomes $Ax = \begin{bmatrix} -1/\sqrt{2}\, x_1 + 1/\sqrt{2}\, x_2 \\ 1/\sqrt{2}\, x_1 + 1/\sqrt{2}\, x_2 \end{bmatrix}$

$$||Ax|| = ||x||$$

1 b)(i) We can represent the matrix A as $A = U\Sigma V^T$

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

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

Hence the left singular vectors of A are the eigenvectors of $A.A^T$

Similarly let's compute $A^T.A$

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

Hence the Right singular vectors of A are the Eigenvectors of A^TA.

1) b) ii From the previous part, $A = U\Sigma V^T$

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

Hence the singular value of A is the square root of the Eigenvalues of AA^T and A^TA.

1) c) (i) FALSE

Reason: Any $n \times n$ identity matrix has all identical eigenvalues 1.

(ii) FALSE

Reason: Let's consider a 2×2 matrix $A = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$ where $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 3 \end{bmatrix}$ are eigenvectors, but their sum is not an eigenvector of A.

(iii) TRUE

Reason: $A\bar{v} = \lambda\bar{v}$, For any matrix to be positive semi definite, $\vec{x}^T A\vec{x} \geq 0$ for all $\vec{x}$. But if $\bar{v}$ is an eigen vector of A, then $\vec{v}^T A\vec{v} = \vec{v}^T\lambda\bar{v} = \bar{v}^T v\lambda$

Since $\bar{v}^T\vec{v}$ is a positive number, in order for $\bar{v}^T A\bar{v} \geq 0$, λ must be greater than or equal to zero (non negative).

(iv) TRUE

Reason: This statement is true for any $n \times n$ identity matrix.

(v) TRUE

Reason: Let's consider $A\bar{v}_1 = \lambda\bar{v}_1$ and $A\bar{v}_2 = \lambda\bar{v}_2$

$$A\bar{v}_1 + A\bar{v}_2 = \lambda\bar{v}_1 + \lambda\bar{v}_2 \Rightarrow A(\bar{v}_1+\bar{v}_2) = \lambda(\bar{v}_1+\bar{v}_2)$$

Hence, from above equation, we could infer that the statement is true.

Question 2 : Probability Refresher

Q2): $P(H|H50) = P(T|H50) = 0.5$

$P(H|H60) = 0.6$, $P(T|H60) = 0.4$

$P(H50) = P(H60) = 0.5$

a) (i) $P(H50|T) = ?$

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} = \frac{P(A \cap B)}{P(B)} \rightarrow (1)$$

From (1), we can say that:

$$P(H50|T) = P(T|H50) \cdot P(H50) \Big/ P(T)$$

$P(T|H50) = 0.5$, $P(H50) = 0.5$

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

$= 0.5 \times 0.5 + 0.4 \times 0.5 = 0.5 \times 0.9 = 0.45$

$$P(T) = 0.45$$

$$P(H50|T) = \frac{0.5 \times 0.5}{0.45} = 25/45 = 5/9$$

Hence $P(H50|T) = 5/9$

---

a) (ii) The Sample size $S = \{T, H, H, H\}$

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

where $P(S) = P(S|H50) \cdot P(H50) + P(S|H60) \cdot P(H60)$

$$P(S) \Rightarrow (0.5)^4 \cdot (0.5) + (0.4)(0.6)^3(0.5)$$

$$P(H50|S) = \frac{(0.5)^4 \cdot (0.5)}{((0.5)^4 \cdot (0.5) + (0.4)(0.6)^3(0.5))}$$

$$P(H50|S) = 0.42$$

---

a) (iii) $P(H50) = P(H55) = P(H60) = 1/3$

Let S be the sample size describing the event of 9 Heads & 1 Tail, the order of the same does not matter.

We need to calculate the following:

$P(H50|S)$, $P(H55|S)$, $P(H60|S)$

$$P(H50|S) = P(S|H50) \cdot P(H50) \,/\, P(S)$$

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

$$P(S|H50) = (0.5 \times 0.5^9)$$

$$P(S|H55) = (0.45 \times 0.55^9)$$

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

$$P(H60|S) = \frac{1/3 \cdot (0.4) \cdot (0.6)^9}{1/3\,((0.5 \cdot (0.5)^9) + (0.45) \cdot (0.55)^9 + (0.4) \cdot (0.6)^9)}$$

$$P(H60|S) = 0.596$$

$$P(H55|8) = \frac{1/3\left((0.45)\cdot(0.55)^9\right)}{1/3\left((0.5\cdot(0.5)^9) + (0.45)\cdot(0.55)^9 + (0.4)\cdot(0.6)^9\right)}$$

$$P(H55|8) = 0.293$$

$$P(H50|8) = \frac{1/3\left((0.5)\cdot(0.5)^9\right)}{1/3\left((0.5\cdot(0.5)^9) + (0.45)\cdot(0.55)^9 + (0.4)\cdot(0.6)^9\right)}$$

$$P(H50|8) = 0.148$$

Hence,

$P(H50|8) \simeq 0.148$, $P(H55|8) \simeq 0.293$, $P(H60|8) \simeq 0.596$

---

2) b) $P(\text{Positive} \mid \text{Pregnant}) = 0.99$, $P(\text{Negative} \mid \text{Pregnant}) = 0.01$

$P(\text{Positive} \mid \text{Not Pregnant}) = 0.10$, $P(\text{Negative} \mid \text{Not Pregnant}) = 0.9$

$P(\text{Pregnant}) = 0.01$, $P(\text{Not Pregnant}) = 0.99$.

We consider X to denote if the women is pregnant (1) or not (0).

and Y be the test result i.e Positive (1) and Negative (0).

$$P(X=1 \mid Y=1) = \frac{P(Y \mid X) \cdot P(X)}{P(Y)}$$

$$= \frac{(0.99 \times 0.01)}{(0.99)(0.01) + (0.01)(0.99)} = 0.09$$

$$P(X=1 \mid Y=1) = 0.09$$

Given that the 99 % of the female population is not pregnant and the model false detects (10 %) of the total population. So, the result that we obtain makes sense as the number of false detections are many more than actual pregnant women. Hence the model's confidence is less and is a bad test.

---

2) c) $E[\alpha f(n) + \beta g(n)] = \alpha \cdot E[f(n)] + \beta \cdot E[g(n)]$

$E[An + b] = E[An] + E[b]$

$$E[An] = E\left(\sum_{j=1}^{n} A_{i,j} X_j\right) = \left(\sum_{j=1}^{n} A_{i,j} E(X_j)\right)$$

$$E(Ax) = \left(\sum_{j=1}^{n} A_{i,j} \cdot E(x)_j \right) = A \cdot E(x)$$

Hence $E(Ax) = A.E(x)$

So, $E(Ax + b) = A.E(x) + b$

2) d) We know that :

$$cov(x) = E\left((x - Ex)(x - Ex)^T\right)$$

$$cov(Ax+b) = E\left((Ax+b - A.E(x) - b)(Ax+b - A.E[x] - b)^T\right)$$

$$Cov(Ax+b) = E\left(\left(A(x - E(x))\right)\left(A(x - E(x))^T\right)\right)$$

$$\Rightarrow E\left(A(x - E(x))\left((x - E(x))^T \cdot A^T\right)\right)$$

$$\Rightarrow A.E\left((x - E(x))\left((x - E(x))\right)^T\right).A^T$$

$$\Rightarrow A.cov(x).A^T$$

Hence $cov(Ax+b) = A.cov(x).A^T$.

Question 3 : Multivariate Derivatives

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

$x^T A y$

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

$$\nabla_x x^T A y = \frac{\partial x^T A y}{\partial x}$$

$$\Rightarrow \underset{1 \times N}{\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}} \underset{N \times M}{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & \\ \vdots & & & a_{nm} \end{bmatrix}} \underset{M \times 1}{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}}$$

$$\Rightarrow \underset{1 \times N}{\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}} \underset{N \times 1}{\begin{bmatrix} a_{11} y_1 + a_{12} \cdot y_2 + \dots a_{1m} \cdot y_m \\ \vdots \\ a_{n1} y_1 + \dots \dots + a_{nm} \cdot y_m \end{bmatrix}}$$

Hence $x^T A y = x_1 (a_{11} y_1 + a_{12} \cdot y_2 + \dots a_{1m} \cdot y_m) + \dots + x_n (a_{n1} y_1 + \dots a_{nm} \cdot y_m)$

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

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

$$\nabla_x x^T A y = Ay$$

Hence $\nabla_x x^T A y = Ay$

---

3) b) From the previous question, we know that:-

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

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

Converting into a matrix form we obtain:

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

Hence $\nabla_y x^T A y = A^T x.$

3c) $\nabla_A x^T A y = \begin{bmatrix} \frac{\partial x^T A y}{\partial a_{1,1}} & \frac{\partial x^T A y}{\partial a_{1,2}} & \cdots & \frac{\partial x^T A y}{\partial a_{1,m}} \\ \vdots & & & \\ \frac{\partial x^T A y}{\partial a_{n,1}} & & \cdots & \frac{\partial x^T A y}{\partial a_{n,m}} \end{bmatrix}$

where $\partial \left(\sum_{i=1}^{n} \sum_{j=1}^{m} (x_i a_{i,j}) \cdot y_j \right) / \partial a$

Hence $\nabla_A x^T A y = x y^T$.

3) d) Given $A \in \mathbb{R}^{n \times n}$ and $f = x^T A x + b^T x$

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

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

We clearly know that: $\partial (b^T x) / \partial x = b^T \rightarrow (1)$

We could solve $\partial (x^T A x) / \partial x$

$$x^T A x = \sum_{j=1}^{n} \sum_{i=1}^{n} a_{ij} \cdot x_i x_j$$

$$\partial (x^T A x) / \partial x_k = \sum_{j=1}^{n} a_{kj} \cdot x_j + \sum_{i=1}^{n} a_{ik} \cdot x_i$$

$$\forall \; k = 1, 2 \ldots n$$

Hence $\partial (x^T A x) / \partial x_k = (A^T + A) x$

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

---

3) e) Given: $f = tr(AB)$

$A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times m}$ where $AB \in \mathbb{R}^{m \times m}$

$$\nabla_A f = \partial f / \partial A = \partial (tr(AB)) / \partial A$$

$$AB = \underset{(M \times N)}{\begin{bmatrix} a_{11} \ldots a_{1n} \\ \vdots \\ a_{m1} \ldots a_{mn} \end{bmatrix}} \underset{(N \times M)}{\begin{bmatrix} b_{11} \ldots b_{1m} \\ \vdots \\ b_{n1} \ldots b_{nm} \end{bmatrix}}$$

$$= \begin{bmatrix} - a_1 - \\ - a_2 - \\ \vdots \\ - a_m - \end{bmatrix} \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \ldots & b_y \\ | & | & & | \end{bmatrix}$$

$$= tr\left(\begin{bmatrix} a_1^T b_1 & a_1^T b_2 & \dots & a_1^T b_r \\ a_2^T b_1 & a_2^T b_2 & \dots & a_2^T b_r \\ \vdots & & & \\ a_M^T b_1 & a_M^T b_2 & \dots & a_M^T b_r \end{bmatrix}\right)$$

$$= \sum_{i=1}^{N} a_{1i} b_{i1} + \sum_{i=1}^{N} a_{2i} \cdot b_{i2} + \dots$$

$$\partial(tr(AB))/\partial a_{ij} = b_{ji}$$

Hence $\nabla_A tr(AB) = B^T$

Question 4 : Deriving Least Squares with matrix derivatives

We shall first differentiate the objective function wrt to W

$$1/2 \sum_{i=1}^{n} \|y_i - W x_i\|^2 = 1/2 \sum_{i=1}^{N} (y_i - W x_i)^T (y_i - W x_i)$$

$$\Rightarrow 1/2 \sum_{i=1}^{N} \left(y_i^T y_i - y_i^T W x_i - x_i^T . W^T y_i + x_i^T W^T W x_i \right)$$

$$= 1/2 \sum_{i=1}^{N} \left(y_i^T y_i - 2 y_i^T W x_i + x_i^T . W^T W x_i \right)$$

We know that $\langle y_i^T, W x_i \rangle = tr(y_i^T W x_i)$

$$= \sum_{i=1}^{N} \left(1/2 y_i^T y_i - tr(y_i^T W x_i) + 1/2 tr(x_i^T W^T W x_i) \right)$$

$$= \sum_{i=1}^{N} \left(1/2 y_i^T y_i - tr(W x_i y_i^T) + 1/2 tr(W x_i x_i^T W^T) \right)$$

$$= \sum_{i=1}^{N} 1/2 y_i^T y_i - tr(W \sum_{i=1}^{N} x_i y_i^T) + 1/2 tr(W \Sigma (x_i x_i^T) W^T)$$

$$= \sum_{i=1}^{N} 1/2 y_i^T y_i - tr(W x y^T) + 1/2 tr(W x x^T W^T)$$

$$\nabla_W h = 0 - y x^T + 1/2 (W x x^T + W x x^T) = 0$$

$$\Rightarrow \quad -yx^T + wxx^T = 0$$

Solving for w, we get:

$$w = yx^T(xx^T)^{-1}$$

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 2022, Prof. J.C. Kao, TAs Y. Li, P. Lu, T. Monsoor, T. wang

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt

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

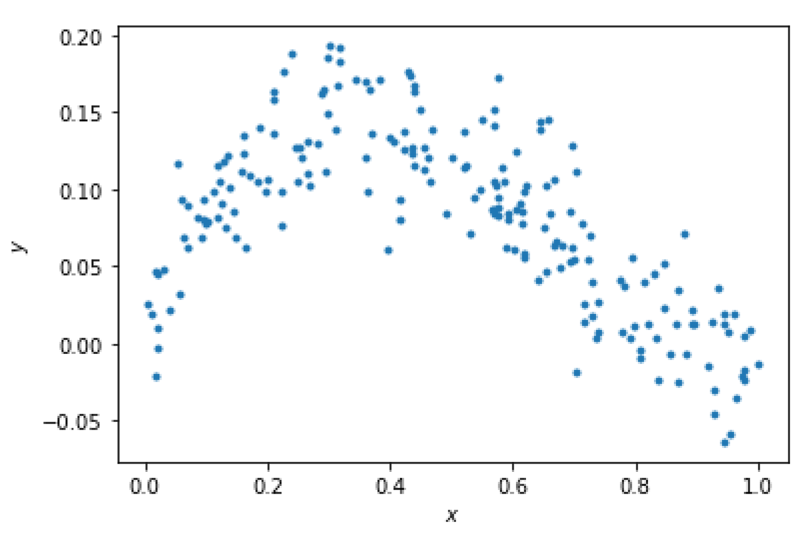
Data generation

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) The generating distribution of x is uniform with parameters are a = 0, b = 1. We have used np.random.uniform.

(2) The distribution of the additive noise is Gaussian with mean 0 and stdev 0.03. We have used np.random.normal to generate the same.

Fitting data to the model (5 points)

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

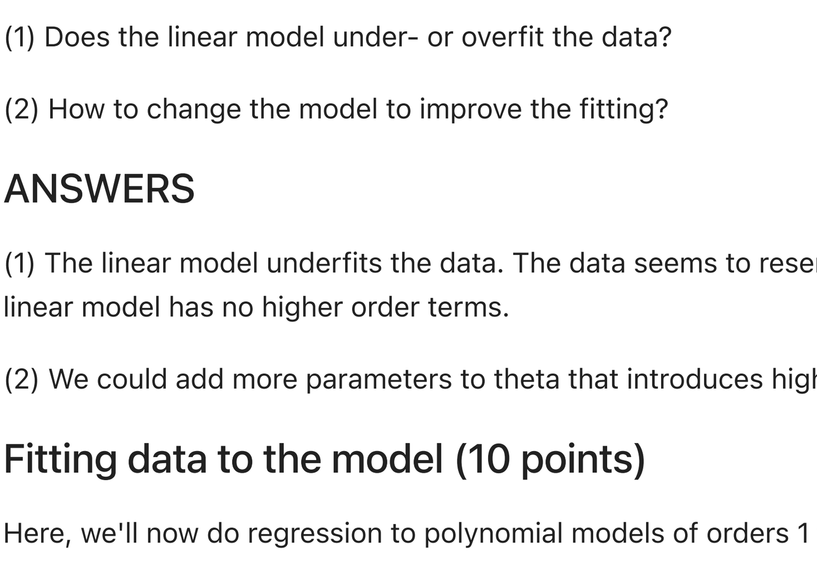
In [4]:

```
# 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)@(xhat.T))@(xhat@(y))

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

In [5]:

```
# 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[5]: `[<matplotlib.lines.Line2D at 0x7f929154a7c0>]`



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. The data seems to resemble higher order polynomial function (such as a parabola), but our linear model has no higher order terms.

(2) We could add more parameters to theta that introduces higher order terms to obtain a better fit.

Fitting data to the model (10 points)

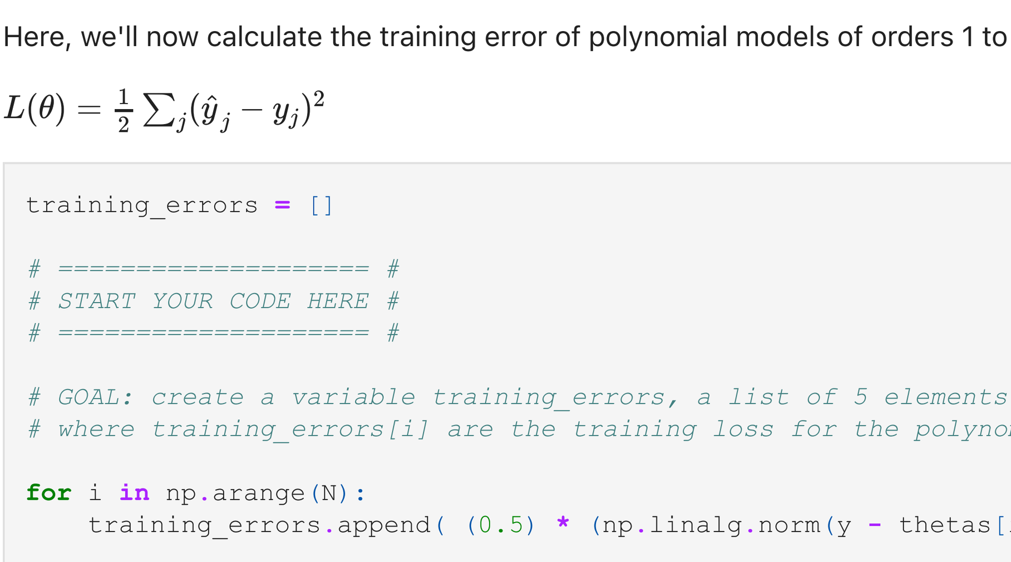
Here, we'll now do regression to polynomial models of orders 1 to 5. Note, the order 1 model is the linear model you prior fit.

In [6]:

```
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.

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

for i in np.arange(1,N):
    xhat = np.vstack((x**(i+1), xhat))
    xhats.append(xhat)
    thetas.append(np.linalg.inv((xhats[i])@(xhats[i].T))@(xhats[i]@(y)))

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

In [7]:

```
# 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 [8]:

```
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 np.arange(N):
    training_errors.append( (0.5) * (np.linalg.norm(y - thetas[i]@(xhats[i]))**2))


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

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

```
Training errors are: 
 [0.2379961088362701, 0.1092492220926853, 0.08169603801105374, 0.08165353735296982, 0.08161479195525295]
```

QUESTIONS

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

(2) Why is this expected?

ANSWERS

(1) The polynomial with order 5 has the best (lowest) training error.

(2) Higher order polynomial functions contains more dimensions to fit the data. Hence, we can adjust the parameters and control the shape of the function and thus make it more fitted.

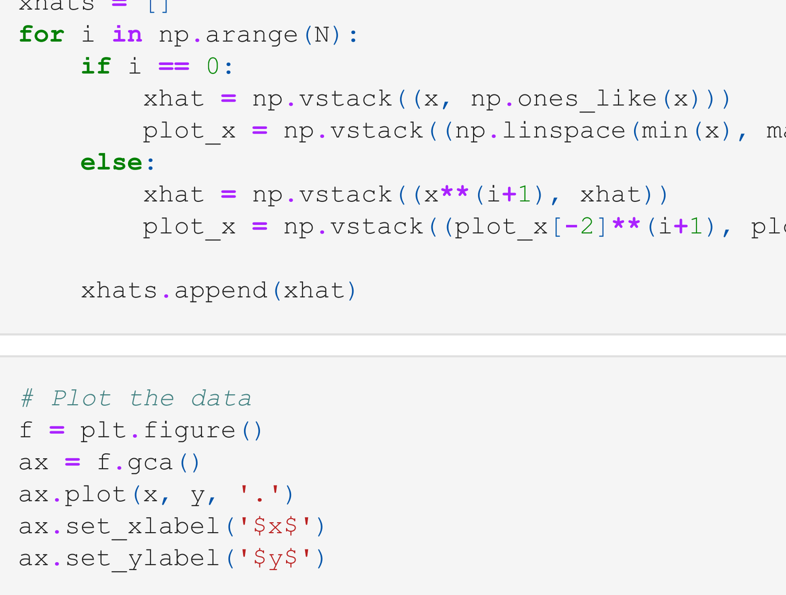
Generating new samples and validation error (5 points)

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

In [9]:

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

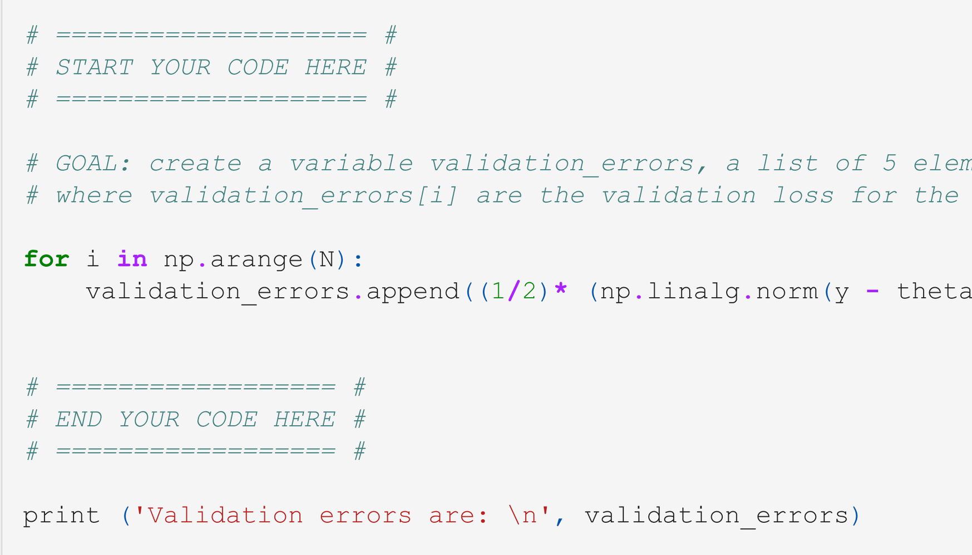


In [10]:

```
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 [11]:

```
# 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 [12]:

```
validation_errors = []

# ==================== #
# START YOUR CODE HERE #
# ==================== #

# GOAL: create a variable validation_errors, a list of 5 elements,
# where validation_errors[i] are the validation loss for the polynomial fit of order i+1.

for i in np.arange(N):
    validation_errors.append((1/2)* (np.linalg.norm(y - thetas[i]@(xhats[i]))**2))


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

print ('Validation errors are: \n', validation_errors)
```

```
Validation errors are: 
 [80.86165184550586, 213.19192445057962, 3.12569710827847, 1.1870765196576454, 214.91021814405832]
```

QUESTIONS

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

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

ANSWERS

(1) The polynomial with order 4 has the best (lowest) validation error.

(2) Order 5 polynomial models do not generalize well because they overfit the data. It means, the model is too complex and only memorizes the data. Since linear regression "learning" is based on adjust weights the minimize a cost function, a complex higher order function that can memorize the 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.

In [ ]: