ECE C147/C247: Neural Networks & Deep Learning, Winter 2023 Homework #1

Linear Algebra Refresher

a) Let Q be a real orthogonal matrix

1. Q transpose and Q inverse are also orthonormal
   - Proof: Q is orthonormal, which means that $Q^TQ = QQ^T = I$. Therefore, $Q^T = (Q^TQ)Q^T = IQ^T = Q^{-1}$. Additionally, since $QQ^T = I, Q^{-1}Q = I$ as well, which means Q^{-1} is also orthonormal.
2. Q has eigenvalues with norm 1
   - Proof: Since Q is orthonormal, it preserves the Euclidean norm of vectors, meaning that the length of a vector is unchanged after being transformed by Q. Therefore, all of the eigenvalues of Q must have a magnitude of 1, since otherwise the eigenvectors would change length.
3. The determinant of Q is either ± 1
   - Proof: Q is orthonormal, which means that $Q^TQ = QQ^T = I$. Therefore, the determinant of Q is the product of the eigenvalues, which all have magnitude 1, so the determinant must be either ± 1.
4. Q defines a length preserving transformation.
   - Proof: Since Q is orthonormal, it preserves the Euclidean norm of vectors. Therefore, it preserves the length of any vector, which means it defines a length-preserving transformation.

b) Let A be a matrix

1. What is the relationship between the singular vectors of A and the eigenvectors of AA^T ? What about A^TA ?
   - The relationship between the singular vectors of A and the eigenvectors of AA^T and A^TA is that the singular vectors of A are the eigenvectors of either AA^T or A^TA .
2. What is the relationship between the singular values of A and the eigen-values of AA^T ? What about A^TA ?
   - The relationship between the singular values of A and the eigenvalues of AA^T and A^TA is that the square of the singular values of A are the eigenvalues of either AA^T or A^TA .

c) True or False

1. Every linear operator in an n-dimensional vector space has n distinct eigenvalues
   - False. Not every linear operator in an n-dimensional vector space has n distinct eigenvalues. An operator may have multiple eigenvectors corresponding to the same eigenvalue, in which case the number of distinct eigenvalues will be less than n.
2. A non-zero sum of two eigenvectors of a matrix A is an eigenvector
   - False. A non-zero sum of two eigenvectors of a matrix A is not necessarily an eigenvector. The sum of two eigenvectors of a matrix A is an eigenvector if and only if the sum of the corresponding eigenvalues is 0.
3. 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. 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. This is because the eigenvalues of a matrix are the roots of the characteristic polynomial, which is a polynomial of degree n. The roots of a polynomial are always real, and the roots of a polynomial of degree n are always non-negative if the polynomial is positive semidefinite.
4. The rank of a matrix can exceed the number of distinct non-zero eigenvalues.
   - True. The rank of a matrix can exceed the number of distinct non-zero eigenvalues. This is because the rank of a matrix is the number of linearly independent columns, which may be less than the number of distinct non-zero eigenvalues.
5. A non-zero sum of two eigenvectors of a matrix A corresponding to the same eigenvalue λ is always an eigenvector.
   - False. A non-zero sum of two eigenvectors of a matrix A corresponding to the same eigenvalue λ is not always an eigenvector. The sum of two eigenvectors of a matrix A corresponding to the same eigenvalue λ is an eigenvector if and only if the sum of the corresponding eigenvalues is 0.

Probability Refresher

a) A jar of coins is equally populated with two types of coins. One is type "H50" and comes up heads with probability 0.5. Another is type "H60" and comes up heads with probability 0.6

1. You take one coin from the jar and flip it. It lands tails. What is the posterior probability that this is an $H50$ coin?
   - Use Bayes Theorem to calculate the posterior probability:
   - $P(H50|T) = \frac{P(T|H50)*P(H50)}{P(T)} = \frac{0.5*0.5}{(0.5*0.5+0.4*0.5)} = 0.56$
2. You put the coin back, take another, and flip it 4 times. It lands T, H, H, H . How likely is the coin to be type $H50$?
   - The likelihood of getting T, H, H, H in 4 flips given that the coin is type $H50 : P(T,H,H,H|H50) = (0.5)^3 * (0.5)^1 = 0.0625$.
   - Use Bayes Theorem to calculate the posterior probability:
   - $P(H50|T,H,H,H) = \frac{P(T,H,H,H|H50)*P(H50)}{P(T,H,H,H)} = \frac{(0.5)^3*(0.5)^1*0.5}{P(T,H,H,H)}$
     - $P(T,H,H,H) = P(T|H50) * P(H,H,H|T,H50) * P(H50) + P(T|H60) * P(H,H,H|T,H60) * P(H60) = (1-0.5) * (0.5)^3 + (1-0.6) * (0.6)^3 = 0.1489$
   - $P(H50|T,H,H,H) = \frac{(0.5)^3*(0.5)^1*0.5}{0.1489} \approx 0.21$
3. A new jar is now equally populated with coins of type $H50, H55$, and $H60$ (with probabilities of coming up heads $0.5, 0.55$, and 0.6 respectively. You take one coin and flip it 10 times. It lands heads 9 times. How likely is the coin to be of each possible type?
   - Each coin is equally likely to be chosen, so the prior probability of each coin is $\frac{1}{3}$.
   - Use Bayes Theorem and Binomial Theorem, $\frac{n!}{k!(n-k!)}$, to calculate the posterior probability for each coin flipping $9H\ in\ 10F$:
     - $P(H50|9H\ in\ 10F) = \frac{P(9H\ in\ 10F|H50)*P(H50)}{P(9H\ in\ 10F)}$
       - $P(H50|9H\ in\ 10F) = \frac{\binom{10}{9}*(0.5)^9*(0.5)^1*\frac{1}{3}}{P(9H\ in\ 10F)}$
     - $P(H55|9H\ in\ 10F) = \frac{P(9H\ in\ 10F|H55)*P(H55)}{P(9H\ in\ 10F)}$
       - $P(H55|9H\ in\ 10F) = \frac{\binom{10}{9}*(0.55)^9*(0.45)^1*\frac{1}{3}}{P(9H\ in\ 10F)}$
     - $P(H60|9H\ in\ 10F) = \frac{P(9H\ in\ 10F|H60)*P(H60)}{P(9H\ in\ 10F)}$
       - $P(H60|9H\ in\ 10F) = \frac{\binom{10}{9}*(0.6)^9*(0.4)^1*\frac{1}{3}}{P(9H\ in\ 10F)}$
     - $P(9H\ in\ 10F) = \binom{10}{9}(0.5)^9 * (0.5)^1 + \binom{10}{9}(0.55)^9 * (0.45)^1 + \binom{10}{9}(0.6)^9 * (0.4)^1$
     - $P(H50|9H\ in\ 10F) \approx 0.046$
     - $P(H55|9H\ in\ 10F) \approx 0.098$
     - $P(H60|9H\ in\ 10F) \approx 0.19$

b) Students at UCLA are from these disciplines: 15% Science, 21% Healthcare, 24% Liberal Arts, and 40% Engineering. (Each student belongs to a unique discipline.) The students attend a lecture and give feedback. Suppose 90% of the Science students liked the lecture, 18% of the Healthcare students liked it, none of the Liberal Arts students liked it, and 10% of the Engineering students liked it. If a student is randomly chosen, and the student liked the lecture, what is the conditional probability that the student is from Science?

- Make S the event that the student is from Science, L the event that the student liked the lecture, then P(S|L) is the conditional probability that the student is from Science given that the student liked the lecture.
- $P(S) = 0.15$
  - The probability that a student is from Science
- $P(L|S) = 0.9$
  - The probability that a student liked the lecture given that the student is from Science
- Also make H, A, E are the events that the student is from Healthcare, Liberal Arts and Engineering respectively
- $P(H) = 0.21$
- $P(B|H) = 0.18$
- $P(A) = 0.24$
- $P(B|A) = 0$
- $P(E) = 0.4$
- $P(B|E) = 0.1$
- The total Probability to calculate the probability that a student liked the lecture:
  - $P(L) = P(L|S)P(S) + P(L|H)P(H) + P(L|A)P(A) + P(L|E)P(E) = 0.9 * 0.15 + 0.18 * 0.21 + 0 * 0.24 + 0.1 * 0.4 = 0.153$
- So, the conditional probability that the student is from Science given that the student liked the lecture is:
  - $\frac{P(S|L)=P(L|S)*P(S)}{P(L)} = \frac{0.9*0.15}{0.153} = 0.59$

c) Consider a pregnancy test with the following statistics

"If the woman is pregnant, the test returns "positive" (or 1, indicating the woman is pregnant) 99% of the time. If the woman is not pregnant, the test returns "positive" 10% of the time. At any given point in time, 99% of the female population is not pregnant."

What is the probability that a woman is pregnant given she received a positive test? The answer should make intuitive sense; give an explanation of the result that you find

- Given that a woman received a positive test, the probability that she is pregnant is:
- $P(Preg|T) = \frac{P(T|Preg)*P(Preg)}{P(T)}$
  - $Preg$ is the event that the woman is pregnant
  - T is the event that the test returns positive
- $P(Preg)$ is the probability that a woman is pregnant (1% or 0.01)
- $P(T|preg)$ is the probability that the test returns positive given that the woman is pregnant (99% or 0.99)
- $P(T)$ is the overall probability of a positive test result and can be calculated as:
  - $P(T) = P(T|A) * P(A) + P(T|Preg') * P(Preg')$
    - $P(T|Preg')$ is the probability that the test returns positive given that the woman is not pregnant (10% or 0.1)
    - $P(Preg')$ is the event that the woman is not pregnant (99% or 0.99)
  - Thus, $P(T) = 0.99 * 0.01 + 0.1 * 0.99 = 0.108$
- Therefore $P(Preg|T) = \frac{0.99*0.01}{0.108} = 0.0917$ or 9.17%
- This makes sense since a positive test result doesn't necessarily mean that the woman is pregnant, even though the test returns positive 99% of the time when the woman is pregnant. The test also returns positive 10% of the time when the woman is not pregnant. So, that combined with the fact the majority of the female population is not pregnant, a positive test result by itself is not a strong indicator of a pregnancy.

d) Let $x_1, x_2, ..., x_n$ be identically distributed random variables. A random vector, x, is defined as:

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

What is $\mathbb{E}(Ax + b)$ in terms of $\mathbb{E}(x)$, given that A and b are deterministic?

Expected value is the weighted average of possible values of a random variable, with weights given by their respective theoretical probabilities. The formula for expected value is:

$$\mathbb{E}(x) = \sum_{i=1}^{n} x_i P(x_i)$$

First since A and b are deterministic, they can be pulled out of the Expected value equation:

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

Additionally, since $x_1, x_2, \ldots, x_n$ are identically distributed, they have the same expected value $\mathbb{E}(x)$, and since A is a deterministic matrix, it doesn't change the expected value of the vector.

e) Let

$$\text{cov}(x) = \mathbb{E}((x - \mathbb{E}x)(x - \mathbb{E}x)^T)$$

What is $\text{cov}(Ax + b)$ in terms of $\text{cov}(x)$, given that A and b are deterministic?

Covariance is a measure of how two random variables change together. The formula for covariance is:

$$\text{cov}(X) = \mathbb{E}[(X - \mathbb{E}(X))(X - \mathbb{E}(X))^T]$$

So given $\text{cov}(Ax + b)$ in terms of $\text{cov}(x)$, we can write:

$\text{cov}(Ax + b) = \mathbb{E}[((Ax + b) - \mathbb{E}(Ax + b))((Ax + b) - \mathbb{E}(Ax + b))^T]$

A and b are deterministic, so they can be pulled out of the Expected values:

$\text{cov}(Ax + b) = \mathbb{E}[A(x - \mathbb{E}(x)) + (b - \mathbb{E}(b))(A(x - \mathbb{E}(x)) + (b - \mathbb{E}(b)))^T]$

We can simplify the above expression using the properties of the Expected value:

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

After substituting the above expression, we get:

$\text{cov}(Ax + b) = A\mathbb{E}((x - \mathbb{E}x)(x - \mathbb{E}x)^T)A^T$

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

So $\text{cov}(Ax + b) = A\text{cov}(x)A^T$ maening the covariance matrix of $Ax + b$ is equal to the covariance matrix of x

Multivariate Derivatives

a) Let $x \in \mathbb{R}^n, y \in \mathbb{R}^m$, and $A \in \mathbb{R}^{n \times m}$. What is $\nabla_x x^TAy$?

$x^TAy = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}x_iy_j$

$\nabla_x x^TAy = \frac{\partial}{\partial x_i}(\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}x_iy_j) = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}y_j = A^Ty$

b)Let $x \in \mathbb{R}^n, y \in \mathbb{R}^m$, and $A \in \mathbb{R}^{n \times m}$. What is $\nabla_y x^TAy$?

$\nabla_y x^TAy = \frac{\partial}{\partial y_j}(\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}x_iy_j) = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}x_i = x^TA$

c)Let $x \in \mathbb{R}^n, y \in \mathbb{R}^m$, and $A \in \mathbb{R}^{n \times m}$. What is $\nabla_A x^TAy$?

$\nabla_A x^TAy = \frac{\partial}{\partial A_{ij}}(\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}x_iy_j) = \sum_{i=1}^{n}\sum_{j=1}^{m} x_iy_j = x^Ty$

d) Let $x \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}$, and let $f(x) = x^TAx + b^Tx$. What is $\nabla_x f$?

$\frac{\partial}{\partial x_i}(x^TAx + b^Tx) = \frac{\partial}{\partial x_i}(x^TAx) + \frac{\partial}{\partial x_i}(b^Tx)$

$\frac{\partial}{\partial x_i}(x^TAx) = \frac{\partial}{\partial x_i}(\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j) = 2Ax \quad \because \text{A is symetric}$

$\frac{\partial}{\partial x_i}(b^Tx) = \frac{\partial}{\partial x_i}(\sum_{j=1}^{n} b_jx_j) = b$

Therefore,

$\nabla_x f = 2Ax + b$

e) Let $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times n}$. and $f = tr(AB)$. What is $\nabla_A f$?

$tr(AB) = \sum_{i=1}^{n} a_{ii}b_{ii}$

$\nabla_A f = \frac{\partial}{\partial A_{ii}}(\sum_{i=1}^{n} a_{ii}b_{ii}) = \sum_{i=1}^{n} b_{ii} = B$

f) Let $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times n}$. and $f = tr(BA + A^TB + A^2B)$. What is $\nabla_A f$?

Since the derivative of trace of a matrix is the transpose of the matrix, we can breakdown the expression as:

$\frac{\partial}{\partial A_{i,j}} tr(BA) = B^T$

$\frac{\partial}{\partial A_{i,j}} tr(A^TB) = B$

$\frac{\partial}{\partial A_{i,j}} tr(A^2B) = 2AB$

So the gradient of this function with respect to A is the sum of these three derivatives is $B^T + B + 2AB$ or $\nabla_A f = B^T + B + 2AB$.

g) Let $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times n}$. and $f = \|A + \lambda B\|_F^2$. What is $\nabla_A f$?

$\nabla_A f = 2(A + \lambda B)$

The Frobenius norm of a matrix is defined as the square root of the sum of the squares of its elements:

$$\|C\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{i,j}^2$$

Using the chain rule, we can derive the derivative of the Frobenius norm with respect to a matrix:

$\frac{\partial}{\partial A_{i,j}}\|A + \lambda B\|_F^2 = \frac{\partial}{\partial A_{ij}}(\sum_{i=1}^{n}\sum_{j=1}^{n}(A_{ij} + \lambda B_{ij})^2)$

$= 2(A_{ij} + \lambda B_{ij})$

Therefore,

$\nabla_A f = 2(A + \lambda B)$

Deriving Least-Squares With Matrix Derivatives

In least-squares, we seek to estimate some multivariate output y via the model

$$\hat{y} = Wx$$

In the training set we're given paired data examples $(x^{(i)}, y^{(i)})$ from $i = 1, ..., n$. Least-squares is the following quadratic optimization problem:

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

Derive the optimal W

Where W is a matrix, and for each example in the training set, both $x^{(i)}$ and $y^{(i)}$ $\forall\, i = 1, ..., n$ are vectors

SOLUTION:

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

$\frac{\partial}{\partial W_{ij}}\frac{1}{2}tr((y^{(i)} - Wx^{(i)})^T(y^{(i)} - Wx^{(i)}))$

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

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

$= \sum_{i=1}^{n}(x^{(i)}(y^{(i)} - Wx^{(i)}))$

To minimize the above expression, we set the gradient to zero adn solve for W:

$\sum_{i=1}^{n}(x^{(i)}(y^{(i)} - Wx^{(i)})) = 0$

$\sum_{i=1}^{n} x^{(i)}y^{(i)} - W(\sum_{i=1}^{n} x^{(i)}x^{(i)^T}) = 0$

$x^Ty = W(x^Tx)$

$W = (x^Tx)^-1(x^Ty)$

Regularized Least Squares

In lecture, we worked through the following least squares problem

$$\arg\min_\theta \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \theta^T\hat{x}^{(i)})^2$$

However, the least squares has a tendency to overfit the training data. One common technique used to address the overfitting problem is regularization. In this problem, we work through one of the regularization techniques namely ridge regularization which is also known as the regularized least squares problem. In the regularized least squares we solve the following optimization problem

$$\arg\min_\theta \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \theta^T\hat{x}^{(i)})^2 + \frac{\lambda}{2}\|\theta\|_2^2$$

where λ is a tunable regularization parameter. From the above cost function it can be observed that we are seeking least squares solution with a smaller 2-norm. Derive the solution to the regularized least squares problem, i.e Find θ^*.

SOLUTION:

The optimazation problem becomes:

$$\arg\min_\theta \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \theta^T\hat{x}^{(i)})^2 + \frac{\lambda}{2}\|\theta\|_2^2 = \arg\min_\theta \frac{1}{2}(y^{(i)} - \hat{x}^{(i)}\theta)^T(y^{(i)} - \hat{x}^{(i)}\theta) + \frac{\lambda}{2}\theta^T\theta$$

Simplifing the cost function:

$$\mathcal{L}(\theta) = \frac{1}{2}[y^{(i)^T}y^{(i)} - 2y^{(i)^T}\hat{x}^{(i)}\theta + \theta^T\hat{x}^{(i)^T}\hat{x}^{(i)}\theta] + \frac{\lambda}{2}\theta^T\theta$$

The Cost function is convex, so we can solve for θ^* by setting the dervative equal to zero:

$\nabla_\theta\mathcal{L}(\theta) = -\nabla_\theta[y^{(i)^T}\hat{x}^{(i)}\theta] + \frac{1}{2}\nabla_\theta[\theta^T\hat{x}^{(i)^T}\hat{x}^{(i)}\theta] + \frac{\lambda}{2}\nabla_\theta[\theta^T\theta]$

$= \hat{x}^{(i)^T}y^{(i)} + (\hat{x}^{(i)^T}\hat{x}^{(i)} + \lambda I)\theta$

Setting the derivative to zero and soving for θ^*:

$\theta^* = (\hat{x}^{(i)^T}\hat{x}^{(i)} + \lambda I)^{-1}\hat{x}^{(i)^T}y^{(i)}$

Linear Regression

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.

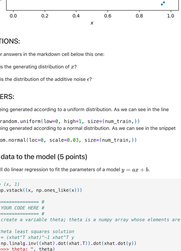
ECE C147/C247, Winter Quarter 2023, Prof. J.C. Kao, TAs: T.M, P.L, R.G, K.K, N.V, S.R, S.P, M.E

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 [22]:
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[22]:
Text(0, 0.5, '$y$')
```



QUESTIONS:

Write your answers in the markdown cell below this one:

(1) What is the generating distribution of ϵ?

(2) What is the distribution of the additive noise ϵ?

ANSWERS:

(1) x is being generated according to a uniform distribution. As we can see in the line

```
x = np.random.uniform(low=0, high=1, size=(num_train,))
```

(2) ϵ is being generated according to a normal distribution. As we can see in the snippet

```
np.random.normal(loc=0, scale=0.03, size=(num_train,))
```

Fitting data to the model (5 points)

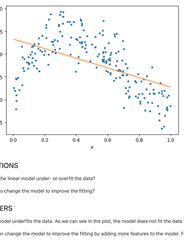
Here, we'll do linear regression to fit the parameters of a model $y = ax + b$.

```
In [23]:
# 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]

# Make theta least squares solution
# theta = (xhat^T xhat)^-1 xhat^T y
theta = np.linalg.inv(xhat.dot(xhat.T)).dot(xhat.dot(y))
print("===> theta:", theta)
print("===> theta.shape:", theta.shape)

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

```
===> theta:  [-0.10199633  0.13315817]
===> theta.shape:  (2,)
```

```
In [24]:
# 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[24]:
[<matplotlib.lines.Line2D at 0x117824910>]
```



QUESTIONS

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

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

ANSWERS

(1) This model underfits the data. As we can see in the plot, the model does not fit the data well. The model is not complex enough to fit the data well.

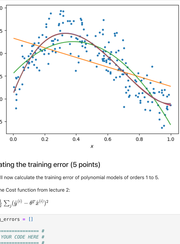
(2) We can change the model to improve the fitting by adding more features to the model. For example, we can add a quadratic term to the model. This will allow the model to fit the data better.

Fitting data to the model (5 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 [25]:
N = 5
xhats = []
thetas = []

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

# GOAL: create a variable thetas,
# thetas is a list, where thetas[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 2 np.array with the coefficients of the x^2, x, and 1 respectively.
#   ... etc.

for i in range(N):
    # Index is i+1 since iteration
    if i == 0:
        # Append the model parameters for linear regression to the list
        thetas.append(theta)
        # Append the design matrix for linear regression to the list
        xhats.append(xhat)
    else:
        # Create a new design matrix with additional polynomial features
        xhat = np.vstack((x**(i+1), xhat))
        # Append the new design matrix to the list
        xhats.append(xhat)
        # Create new model parameters with additional polynomial features
        theta = np.linalg.inv(xhats[i].dot(xhats[i].T)).dot(xhats[i].dot(y))
        # Append the model parameters for the new design matrix to the list
        thetas.append(theta)

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

```
In [26]:
# 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 range(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 range(N):
    ax.plot(plot_xs[i][-2,:], thetas[i].dot(plot_xs[i]))

labels = ['data']
[labels.append('n={}'.format(i+1)) for i in range(N)]
bbox_to_anchor=(1.3, 1)
lgd = ax.legend(labels, bbox_to_anchor=bbox_to_anchor)
```



Calculating the training error (5 points)

Here, we'll now calculate the training error of polynomial models of orders 1 to 5.

Implent the Cost function from lecture 2:

$L(\theta) = \frac{1}{N}\sum_i (y^{(i)} - \theta^T \tilde{x}^{(i)})^2$

```
In [27]:
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.

# Implement the L(theta) I added above over range of N
for i in range(N):
    training_errors.append((1/2) * (np.linalg.norm(y - thetas[i].dot(xhats[i]))**2))

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

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

```
Training errors are:
 [0.2379961880382627, 0.1892492220926527, 0.0016968308108534, 0.0016553737596939, 0.0016147919552595]
```

QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

ANSWERS

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

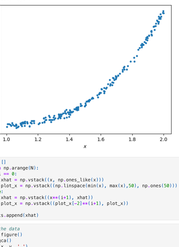
(2) This is expected because the polynomial of order 5 is the most complex model. It is able to fit the data better than the other models.

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.

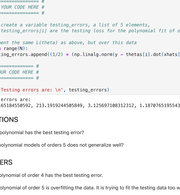
```
In [28]:
x = np.random.uniform(low=1, high=2, size=(num_train,))
y = x - 2*x**2 + x**3 + np.random.normal(loc=0, scale=0.03, size=(num_train,))
f = plt.figure()
ax = f.gca()
ax.plot(x, y, '.')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
```

```
Out[28]:
Text(0, 0.5, '$y$')
```




```
In [29]:
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 [30]:
# 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 [31]:
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.

# Implement the same L(theta)as above, but over this data
for i in range(N):
    testing_errors.append((1/2) * (np.linalg.norm(y - thetas[i].dot(xhats[i]))**2))

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

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

```
Testing errors are:
 [80.08165184558592, 213.19192404585849, 3.1250973803123121, 1.1878765159554373, 214.91821831759682]
```

QUESTIONS

(1) What polynomial has the best testing error?

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

ANSWERS

(1) The polynomial of order 4 has the best testing error.

(2) The polynomial of order 5 is overfitting the data. It is trying to fit the testing data too well. This is causing the model to not generalize well, giving a high testing error.