Problem 1

Part (a)

Let $\mathbf{Q}$ be a real orthogonal matrix.

i. The definition of orthogonality says that a matrix $\mathbf{A}$ is orthogonal if and only if $\mathbf{A}\mathbf{A}^T = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. We are given that $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$. Therefore, $\mathbf{Q}^T$ is orthogonal:

$$\begin{aligned} \mathbf{A}\mathbf{A}^T &= \mathbf{Q}^T(\mathbf{Q}^T)^T \\ &= \mathbf{Q}^T\mathbf{Q} \\ &= \mathbf{Q}\mathbf{Q}^T \\ &= \mathbf{I}. \end{aligned}$$

It also follows that $\mathbf{Q}^T = \mathbf{Q}^{-1}$:

$$\begin{aligned} \mathbf{Q}\mathbf{Q}^T &= \mathbf{I} \\ \mathbf{Q}^{-1}\mathbf{Q}\mathbf{Q}^T &= \mathbf{Q}^{-1}\mathbf{I} \\ \mathbf{I}\mathbf{Q}^T &= \mathbf{Q}^{-1}\mathbf{I} \\ \mathbf{Q}^T &= \mathbf{Q}^{-1}. \end{aligned}$$

Thus, $\mathbf{Q}^T$ is orthogonal, and since $\mathbf{Q}^T = \mathbf{Q}^{-1}$, it follows that $\mathbf{Q}^{-1}$ is also orthogonal.

ii. Using the definition of eigenvalues and eigenvectors,

$$\begin{aligned} \mathbf{Q}\mathbf{x} &= \lambda\mathbf{x} \\ (\mathbf{Q}\mathbf{x})^T(\mathbf{Q}\mathbf{x}) &= (\lambda\mathbf{x})^T(\lambda\mathbf{x}) \\ \mathbf{x}^T\mathbf{Q}^T\mathbf{Q}\mathbf{x} &= \lambda^2\mathbf{x}^T\mathbf{x} \\ \mathbf{x}^T\mathbf{I}\mathbf{x} &= \lambda^2\mathbf{x}^T\mathbf{x} \\ \mathbf{x}^T\mathbf{x} &= \lambda^2\mathbf{x}^T\mathbf{x} \\ 1 &= \lambda^2. \end{aligned}$$

Thus, $|\lambda| = 1$.

iii. According to Equation 18 in *The Matrix Cookbook*, the determinant of any nxn matrix $\mathbf{A}$ is given by $det(\mathbf{A}) = \prod_i \lambda_i$, i.e. the product of its eigenvalues. As shown in (ii), the eigenvalues of $\mathbf{Q}$ must be +1 and/or -1. Thus, the product of the eigenvalues must also be +1 or -1, i.e. the determinant of $\mathbf{Q}$ is also +1 or -1.

iv. We show that $\mathbf{Q}$ defines a length-preserving transformation:

$$\begin{aligned} ||\mathbf{Q}\mathbf{x}||^2 &= (\mathbf{Q}\mathbf{x})^T\mathbf{Q}\mathbf{x} \\ &= \mathbf{x}^T\mathbf{Q}^T\mathbf{Q}\mathbf{x} \\ &= \mathbf{x}^T(\mathbf{Q}^T\mathbf{Q})\mathbf{x} \\ &= \mathbf{x}^T\mathbf{I}\mathbf{x} \\ &= \mathbf{x}^T\mathbf{x} \\ &= ||\mathbf{x}||^2 \end{aligned}$$

Thus, $||\mathbf{Qx}|| = ||\mathbf{x}||$.

Part (b)

Let $\mathbf{A}$ be a matrix.

i. The singular value decomposition of $\mathbf{A}$ is given by $\mathbf{A} = \mathbf{U\Sigma V}^T$, where the columns of $\mathbf{U}$ are its left singular vectors, and the columns of $\mathbf{V}$ are its right singular vectors.

It follows that

$$\begin{aligned}
\mathbf{AA}^T &= \mathbf{U\Sigma V}^T(\mathbf{U\Sigma V}^T)^T \\
&= \mathbf{U\Sigma V}^T\mathbf{V\Sigma}^T\mathbf{U}^T \\
&= \mathbf{U\Sigma V}^T\mathbf{V\Sigma}^T\mathbf{U}^T \\
&= \mathbf{U\Sigma\Sigma}^T\mathbf{U}^T.
\end{aligned}$$

Thus, the left singular vectors of $\mathbf{A}$ are the eigenvectors of $\mathbf{AA}^T$.

Using similar logic, it follows that $\mathbf{A}^T\mathbf{A} = \mathbf{V\Sigma\Sigma}^T\mathbf{V}^T$. Thus, the right singular vectors of $\mathbf{A}$ are the eigenvectors of $\mathbf{A}^T\mathbf{A}$.

ii. As shown in (ii), the singular values of $\mathbf{A}$ are the square roots of the eigenvalues of $\mathbf{AA}^T$ and $\mathbf{A}^T\mathbf{A}$.

Part (c)

i. False. Every linear operator in an n-dimensional vector space has **up to** n distinct eigenvalues.

ii. False. As a counterexample, the eigenvectors for $\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ are $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$. However, $\begin{bmatrix} 0 \\ 2 \end{bmatrix}$ is not an eigenvector.

iii. True. Suppose $\mathbf{A}$ is PSD. Then, for eigenvector $\mathbf{v}$, $\mathbf{v}^T\mathbf{Av} = \mathbf{v}^T\lambda\mathbf{v} = \lambda||\mathbf{v}||^2 \geq 0$. Since $||\mathbf{v}||^2$ is non-negative, it follows that λ must also be non-negative.

iv. True. For example, the matrix $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ has no non-zero eigenvalues but has a rank of one.

v. True. Suppose for matrix $\mathbf{A}$ that $\mathbf{Ax} = \lambda\mathbf{x}$ and $\mathbf{Ay} = \lambda\mathbf{y}$. If we let $\mathbf{z} = \mathbf{x} + \mathbf{y}$, then $\mathbf{Az} = \mathbf{A}(\mathbf{x} + \mathbf{y}) = \mathbf{Ax} + \mathbf{Ay} = \lambda\mathbf{x} + \lambda\mathbf{y} = \lambda(\mathbf{x} + \mathbf{y}) = \lambda\mathbf{z}$.

Problem 2

Part (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.

i. Let $P(\text{H50}|T)$ be the probability that the coin type is "H50" given that the coin landed tails. Then,

$$\begin{aligned}
P(\text{H50}|T) &= \frac{P(\text{H50} \cap T)}{P(T)} \\
&= \frac{P(T|H50) * P(\text{H50})}{P(T|\text{H50}) * P(\text{H50}) + P(T|\text{H60}) * P(\text{H60})} \\
&= \frac{0.5 * 0.5}{(0.5 * 0.5) + (0.6 * 0.5)} \\
&= 0.455.
\end{aligned}$$

ii. Let $P(\text{H50}|THHH)$ be the probability that the coin type is "H50" given that the coin lands THHH. Then,

$$\begin{aligned}
P(\text{H50}|THHH) &= \frac{P(\text{H50} \cap THHH)}{P(THHH)} \\
&= \frac{P(THHH|H50) * P(\text{H50})}{P(THHH|\text{H50}) * P(\text{H50}) + P(THHH|\text{H60}) * P(\text{H60})} \\
&= \frac{0.5^4 * 0.5}{(0.5^4 * 0.5) + (0.4 * 0.6^3 * 0.5)} \\
&= 0.42.
\end{aligned}$$

iii. Let $P(\text{H50}|9H)$ be the probability that the coin type is "H50" given that the coin lands heads 9 out of 10 flips. Then,

$$\begin{aligned}
P(\text{H50}|9H) &= \frac{P(\text{H50} \cap 9H)}{P(9H)} \\
&= \frac{P(9H|H50) * P(\text{H50})}{P(9H|\text{H50}) * P(\text{H50}) + P(9H|\text{H60}) * P(\text{H60})} \\
&= \frac{0.5^{10} * 1/3}{(0.5^{10} * 1/3) + (0.55^9 * 0.45 * 1/3) + (0.6^9 * 0.4 * 1/3)} \\
&= 0.138.
\end{aligned}$$

Likewise,

$$\begin{aligned}
P(\text{H55}|9H) &= \frac{0.55^9 * 0.45 * 1/3}{(0.5^{10} * 1/3) + (0.55^9 * 0.45 * 1/3) + (0.6^9 * 0.4 * 1/3)} \\
&= 0.293,
\end{aligned}$$

and

$$P(\text{H60}|9H) = \frac{0.60^9 * 0.40 * 1/3}{(0.5^{10} * 1/3) + (0.55^9 * 0.45 * 1/3) + (0.6^9 * 0.4 * 1/3)}$$
$$= 0.569.$$

Part (b)

Let L be whether the student liked the lecture, and D be the discipline that student is from. We have choices of $D = S$ for science, $D = H$ for healthcare, $D = LA$ for liberal arts, and $D = E$ for engineering. We are therefore looking for $P(D = S|L)$.

$$\begin{aligned}
P(D = S|L) &= \frac{P(D = S \cap L)}{P(L)} \\
&= \frac{P(L|D = S) * P(D = S)}{P(L|D = S) * P(D = S) + ... + P(L|D = E) * P(D = E)} \\
&= \frac{0.90 * 0.15}{0.90 * 0.15 + 0.18 * 0.21 + 0 * 0.24 + 0.1 * 0.4} \\
&= 0.634.
\end{aligned}$$

Thus, the probability that the student is from Science, given that they liked the lecture, is 0.634.

Part (c)

Let P be whether the woman is pregnant, and let T be whether the test returns a positive result. Thus, we are looking for $P(P|T)$. We are given that if the woman is pregnant, the test returns "positive" 99% of the time (i.e. $P(T|P) = 0.99$). Otherwise, if the woman is not pregnant, the test returns "positive" 10% of the time (i.e. $P(T|\bar{P}) = 0.10$). Finally, we are given that 99% of the female population is not pregnant at any given point in time (i.e. $P(\bar{P}) = 0.99$.

$$\begin{aligned}
P(P|T) &= \frac{P(P \cap T)}{P(T)} \\
&= \frac{P(T|P) * P(P)}{P(T|P) * P(P) + P(T|\bar{P}) * P(\bar{P})} \\
&= \frac{0.99 * 0.01}{0.99 * 0.01 + 0.10 * 0.99} \\
&= 0.091.
\end{aligned}$$

The probability a woman is pregnant given that she received a positive test is about 0.091. This makes intuitive sense because a very small proportion of the overall population is pregnant, and there are quite a few false positives among the population that are not pregnant.

Part (d)

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

$$\begin{aligned}
[\mathbb{E}(\mathbf{Ax}+\mathbf{b})]_i &= \mathbb{E}\left(\sum_{j=1}^{n} a_{i,j}x_j + b_i\right) \\
&= \sum_{j=1}^{n} a_{i,j}\mathbb{E}(x_j) + b_i \\
\mathbb{E}(\mathbf{Ax}+\mathbf{b}) &= \mathbf{A}\mathbb{E}(\mathbf{x}) + \mathbf{b}
\end{aligned}$$

Part (e)

$\mathbf{cov}(\mathbf{Ax}+\mathbf{b}) = \mathbf{A}\mathbf{cov}(\mathbf{x})\mathbf{A}^T$:

$$\begin{aligned}
\mathbf{cov}(\mathbf{x}) &= \mathbb{E}\left[(\mathbf{x}-\mathbb{E}(\mathbf{x}))(\mathbf{x}-\mathbb{E}(\mathbf{x}))^T\right] \\
\mathbf{cov}(\mathbf{Ax}+\mathbf{b}) &= \mathbb{E}\left[((\mathbf{Ax}+\mathbf{b})-\mathbb{E}(\mathbf{Ax}+\mathbf{b}))((\mathbf{Ax}+\mathbf{b})-\mathbb{E}(\mathbf{Ax}+\mathbf{b}))^T\right] \\
&= \mathbb{E}\left[((\mathbf{Ax}+\mathbf{b})-(\mathbf{A}\mathbb{E}(\mathbf{x})+\mathbf{b}))((\mathbf{Ax}+\mathbf{b})-(\mathbf{A}\mathbb{E}(\mathbf{x})+\mathbf{b}))^T\right] \\
&= \mathbb{E}\left[(\mathbf{Ax}-\mathbf{A}\mathbb{E}(\mathbf{x}))(\mathbf{Ax}-\mathbf{A}\mathbb{E}(\mathbf{x}))^T\right] \\
&= \mathbb{E}\left[\mathbf{A}(\mathbf{x}-\mathbb{E}(\mathbf{x}))(\mathbf{x}-\mathbb{E}(\mathbf{x}))^T\mathbf{A}^T\right] \\
&= \mathbf{A}\mathbb{E}\left[(\mathbf{x}-\mathbb{E}(\mathbf{x}))(\mathbf{x}-\mathbb{E}(\mathbf{x}))^T\right]\mathbf{A}^T \\
&= \mathbf{A}\mathbf{cov}(\mathbf{x})\mathbf{A}^T
\end{aligned}$$

Problem 3

(a) $\nabla_{\mathbf{x}} \mathbf{x}^T \mathbf{A} \mathbf{y} = \mathbf{A}\mathbf{y}$:

$$\begin{aligned}
\mathbf{b} &= \mathbf{A}\mathbf{y} \\
\mathbf{x}^T b &= \sum_{i=1}^{n} x_i b_i \\
\frac{\partial \mathbf{x}^T \mathbf{b}}{\partial x_i} &= b_i = \sum_{j=1}^{m} a_{ij} y_j \\
\nabla_{\mathbf{x}} \mathbf{x}^T \mathbf{A} \mathbf{y} &= \mathbf{A}\mathbf{y}
\end{aligned}$$

(b) $\nabla_{\mathbf{y}} \mathbf{x}^T \mathbf{A} \mathbf{y} = \mathbf{A}^T \mathbf{x}$:

$$\begin{aligned}
\mathbf{b} &= \mathbf{A}\mathbf{y} \\
b_i &= \sum_{j=1}^{m} a_{ij} y_j \\
\mathbf{x}^T \mathbf{b} &= \sum_{i=1}^{n} x_i b_i = \sum_{i=1}^{n} x_i \sum_{j=1}^{m} a_{ij} y_j \\
\frac{\partial \mathbf{x}^T \mathbf{b}}{\partial y_j} &= \sum_{i=1}^{n} x_i a_{ij} \\
\nabla_{\mathbf{y}} \mathbf{x}^T \mathbf{A} \mathbf{y} &= \mathbf{A}^T \mathbf{x}
\end{aligned}$$

(c) $\nabla_{\mathbf{A}} \mathbf{x}^T \mathbf{A} \mathbf{y} = \mathbf{x}\mathbf{y}^T$:

$$\begin{aligned}
\mathbf{b} &= \mathbf{A}\mathbf{y} \\
b_i &= \sum_{j=1}^{m} a_{ij} y_j \\
\mathbf{x}^T \mathbf{b} &= \sum_{i=1}^{n} x_i b_i = \sum_{i=1}^{n} x_i \sum_{j=1}^{m} a_{ij} y_j \\
\frac{\partial \mathbf{x}^T \mathbf{b}}{\partial a_{ij}} &= x_i y_j \\
\nabla_{\mathbf{A}} \mathbf{x}^T \mathbf{A} \mathbf{y} &= \mathbf{x}\mathbf{y}^T
\end{aligned}$$

(d) $\nabla_{\mathbf{x}} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x} + \mathbf{b}$:

$$
\begin{aligned}
f &= \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} \\
\nabla_{\mathbf{x}}\mathbf{x}^T\mathbf{A}\mathbf{x} &= (\mathbf{A} + \mathbf{A}^T)\mathbf{x} \\
\mathbf{b}^T\mathbf{x} &= \sum_{i=1}^{n} b_i x_i \\
\nabla_{\mathbf{x}}\mathbf{b}^T\mathbf{x} &= \mathbf{b} \\
\nabla_{\mathbf{x}} f &= (\mathbf{A} + \mathbf{A}^T)\mathbf{x} + \mathbf{b}
\end{aligned}
$$

(e) $\nabla_{\mathbf{A}} tr(\mathbf{AB}) = \mathbf{B}^T$:

Let $f = tr(\mathbf{AB})$. Let $\mathbf{C} = \mathbf{AB}$.

$$
\begin{aligned}
c_{1,1} &= \sum_{i=1}^{n} a_{1,i} b_{i,1} \\
&\ldots \\
c_{n,n} &= \sum_{i=1}^{n} a_{n,i} b_{i,n} \\
tr(\mathbf{C}) &= \sum_{k=1}^{n} c_{k,k} \\
&= \sum_{k=1}^{n}\sum_{i=1}^{n} a_{k,i} b_{i,k} \\
\frac{\partial tr(\mathbf{C})}{\partial a_{i,j}} &= b_{j,i}
\end{aligned}
$$

Therefore, $\nabla_{\mathbf{A}} tr(\mathbf{AB}) = \mathbf{B}^T$.

(f) $\nabla_{\mathbf{A}} tr(\mathbf{BA} + \mathbf{A}^T\mathbf{B} + \mathbf{A}^2\mathbf{B}) = \mathbf{B}^T + \mathbf{B} + (\mathbf{AB} + \mathbf{BA})^T$:

Let $f = tr(\mathbf{BA} + \mathbf{A}^T\mathbf{B} + \mathbf{A}^2\mathbf{B})$. Let $\mathbf{C} = \mathbf{BA} + \mathbf{A}^T\mathbf{B}$.

$$
\begin{aligned}
c_{ij} &= \sum_{k=1}^{n} b_{ik} a_{kj} + \sum_{k=1}^{n} a_{ki} b_{kj} \\
&= \sum_{k=1}^{n} (b_{ik} a_{kj} + a_{ki} b_{kj}) \\
c_{ii} &= \sum_{k=1}^{n} (b_{ik} a_{ki} + a_{ki} b_{ki}) \\
tr(\mathbf{C}) &= \sum_{i=1}^{n}\sum_{j=1}^{n} (b_{ij} a_{ji} + a_{ji} b_{ji}) \\
\frac{\partial tr(\mathbf{C})}{\partial a_{ij}} &= b_{ji} + b_{ij} \\
\nabla_{\mathbf{A}} tr(\mathbf{C}) &= \mathbf{B}^T + \mathbf{B}.
\end{aligned}
$$

According to Equation 107 in *The Matrix Cookbook*, $\nabla_{\mathbf{A}} tr(\mathbf{A}^2\mathbf{B}) = (\mathbf{AB} + \mathbf{BA})^T$.

Therefore, $\nabla_{\mathbf{A}} f = \mathbf{B}^T + \mathbf{B} + (\mathbf{AB} + \mathbf{BA})^T$.

(g) $\nabla_{\mathbf{A}} ||\mathbf{A} + \lambda \mathbf{B}||_F^2 = 2\mathbf{A}^T + 2\lambda \mathbf{B}^T$:

Let $f = ||\mathbf{A} + \lambda \mathbf{B}||_F^2$. Let $\mathbf{C} = \mathbf{A} + \lambda \mathbf{B}$, and let $\mathbf{D} = \mathbf{C}^T \mathbf{C}$. Then, $f = tr(\mathbf{D})$.

$$\begin{aligned}
c_{ij} &= a_{ij} + \lambda b_{ij} \\
d_{ij} &= \sum_{k=1}^{n} c_{ik} c_{kj} \\
tr(\mathbf{D}) &= \sum_{i=1}^{n} d_{ii} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} c_{ji} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} + \lambda b_{ij})(a_{ji} + \lambda b_{ji}) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} a_{ji} + \lambda a_{ji} b_{ij} + \lambda a_{ij} b_{ji} + \lambda^2 b_{ij} b_{ji}) \\
\frac{\partial tr(\mathbf{D})}{\partial a_{ij}} &= (a_{ji} + \lambda b_{ji}) + (a_{ji} + \lambda b_{ji})
\end{aligned}$$

Therefore, $\nabla_{\mathbf{A}} f = 2\mathbf{A}^T + 2\lambda \mathbf{B}^T$.

Problem 4

We want to derive the optimal $\mathbf{W}$ in the objective function

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

In other words, we want to solve for $\mathbf{W}$ by taking the partial derivative of the objective function with respect to $\mathbf{W}$, set it to zero, and solve for $\mathbf{W}$:

$$\frac{\partial}{\partial \mathbf{W}}\left(\sum_{i=1}^{n} ||\mathbf{y}^{(i)} - \mathbf{W}\mathbf{x}^{(\mathbf{i})}||^2\right) = 0.$$

To simply the summation expression in the left side of the equation, we have the equations

$$\begin{aligned}
\frac{1}{2}\sum_{i=1}^{n} ||\mathbf{y}^{(i)} - \mathbf{W}\mathbf{x}^{(i)}||^2 &= \frac{1}{2}\sum_{i=1}^{n} ||\mathbf{y}^{(i)} - \mathbf{x}^{(i)^T}\mathbf{W}||^2 \\
&= \frac{1}{2}\sum_{i=1}^{n}\left(\mathbf{y}^{(i)} - \mathbf{x}^{(i)^T}\mathbf{W}\right)^T\left(\mathbf{y}^{(i)} - \mathbf{x}^{(i)^T}\mathbf{W}\right) \\
&= \frac{1}{2}\left(\mathbf{Y} - \mathbf{X}\mathbf{W}\right)^T\left(\mathbf{Y} - \mathbf{X}\mathbf{W}\right) \\
&= \frac{1}{2}\left(\mathbf{Y}^T\mathbf{Y} - 2\mathbf{Y}^T\mathbf{W}\mathbf{X} + \mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W}\right),
\end{aligned}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are concatenations of all individual $\mathbf{x}^{(i)^T}$ and $\mathbf{y}^{(i)}$, respectively, as shown in class.

Then, we can solve for the partial derivative:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{W}}\left(\sum_{i=1}^{n} ||\mathbf{y}^{(i)} - \mathbf{W}\mathbf{x}^{(\mathbf{i})}||^2\right) &= \frac{\partial}{\partial \mathbf{W}}\left(\frac{1}{2}\left(\mathbf{Y}^T\mathbf{Y} - 2\mathbf{Y}^T\mathbf{X}\mathbf{W} + \mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W}\right)\right) \\
&= 0 - \mathbf{X}^T\mathbf{Y} + (\mathbf{X}^T\mathbf{X} + \frac{1}{2}\mathbf{X}^T\mathbf{X})\mathbf{W} \\
&= \mathbf{X}^T\mathbf{X}\mathbf{W} - \mathbf{X}^T\mathbf{Y} \\
&[= 0].
\end{aligned}$$

Solving for $\mathbf{W}$, we derive $\mathbf{W} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$.

Problem 5

We can solve for $\mathbf{\Theta}$ in the same way that we did in the Problem 4, only now, we add the extra regularization term $\frac{\lambda}{2}\mathbf{\Theta}$ to the objective function.

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{\Theta}}\left(\sum_{i=1}^{n}||\mathbf{y}^{(i)}-\mathbf{\Theta}\mathbf{x^{(i)}}||^2+\frac{\lambda}{2}||\mathbf{\Theta}||_2^2\right) &= \frac{\partial}{\partial \mathbf{\Theta}}\left(\frac{1}{2}\left(\mathbf{Y}^T\mathbf{Y}-2\mathbf{Y}^T\mathbf{X}\mathbf{\Theta}+\mathbf{\Theta}^T\mathbf{X}^T\mathbf{X}\mathbf{\Theta}\right)+\frac{\lambda}{2}||\mathbf{\Theta}||_2^2\right)\\
&= 0-\mathbf{X}^T\mathbf{Y}+(\mathbf{X}^T\mathbf{X}+\frac{1}{2}\mathbf{X}^T\mathbf{X})\mathbf{\Theta}+\lambda\mathbf{\Theta}\\
&= (\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})\mathbf{\Theta}-\mathbf{X}^T\mathbf{Y}\\
&[= 0]
\end{aligned}$$

Solving for $\mathbf{\Theta}$, we derive $\mathbf{\Theta}^* = (\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$.

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

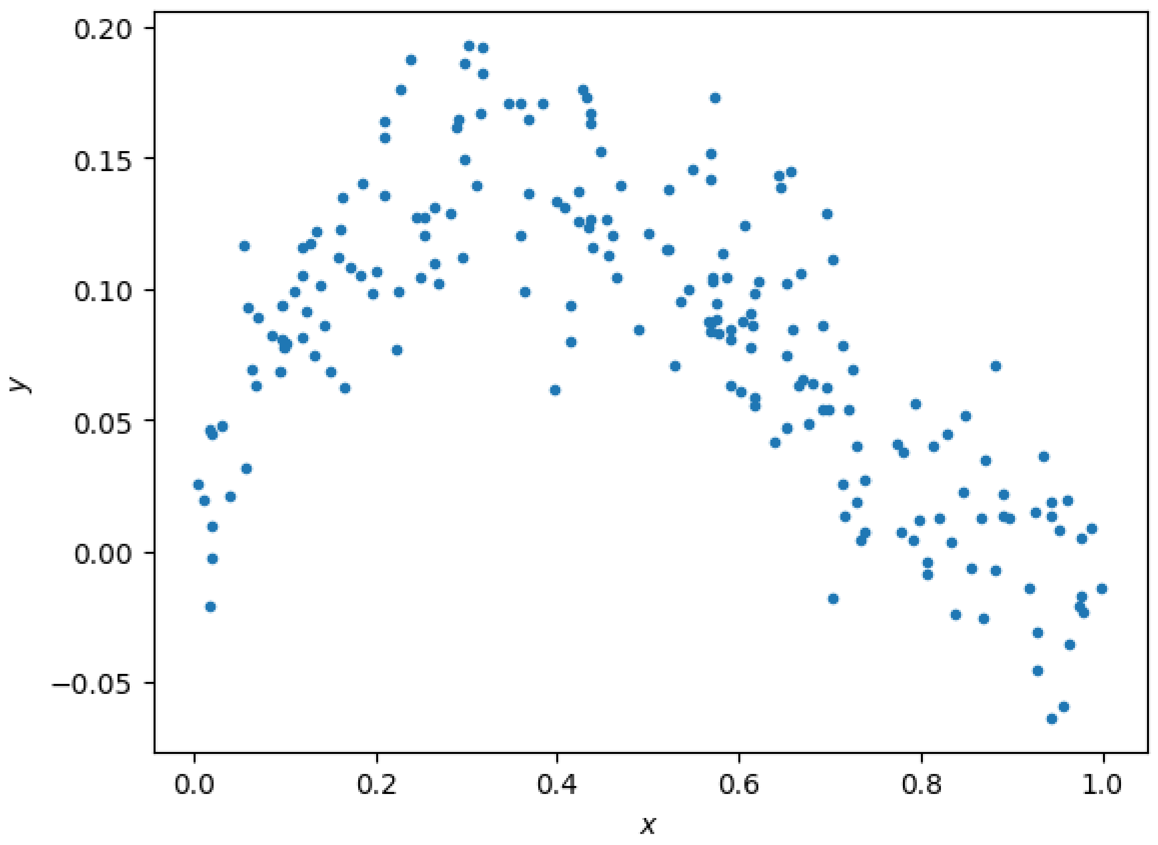
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

#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) The distribution of x is a uniform distribution ranging from 0 to 1.

(2) The distribution of ϵ is a normal distribution with mean=0 and stddev=0.03.

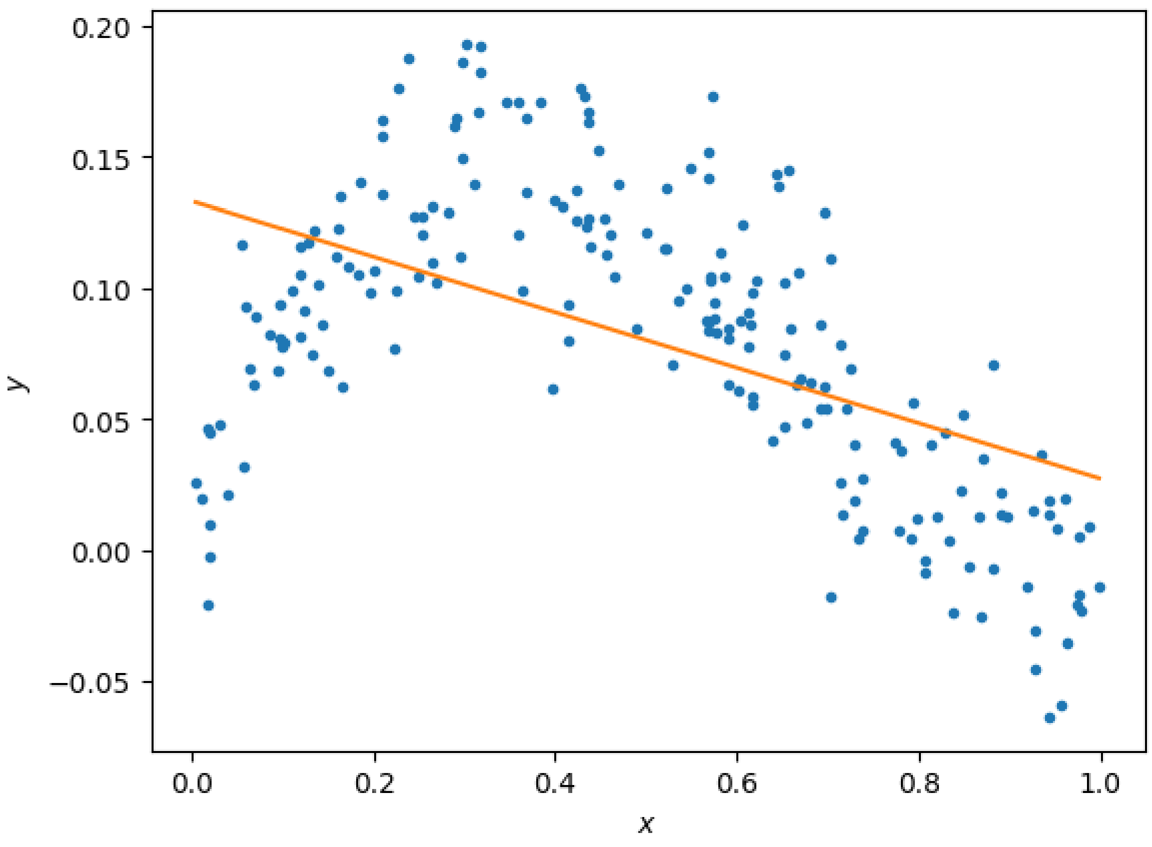
Fitting data to the model (5 points)

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

In [27]:
```
# 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 [

# theta = np.zeros(2) # please modify this line
theta = y.dot(xhat.T).dot(np.linalg.inv(xhat.dot(xhat.T)))

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

In [15]:
```
# 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[15]:
```
[<matplotlib.lines.Line2D at 0x11ecd4490>]
```



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 underfits the data.

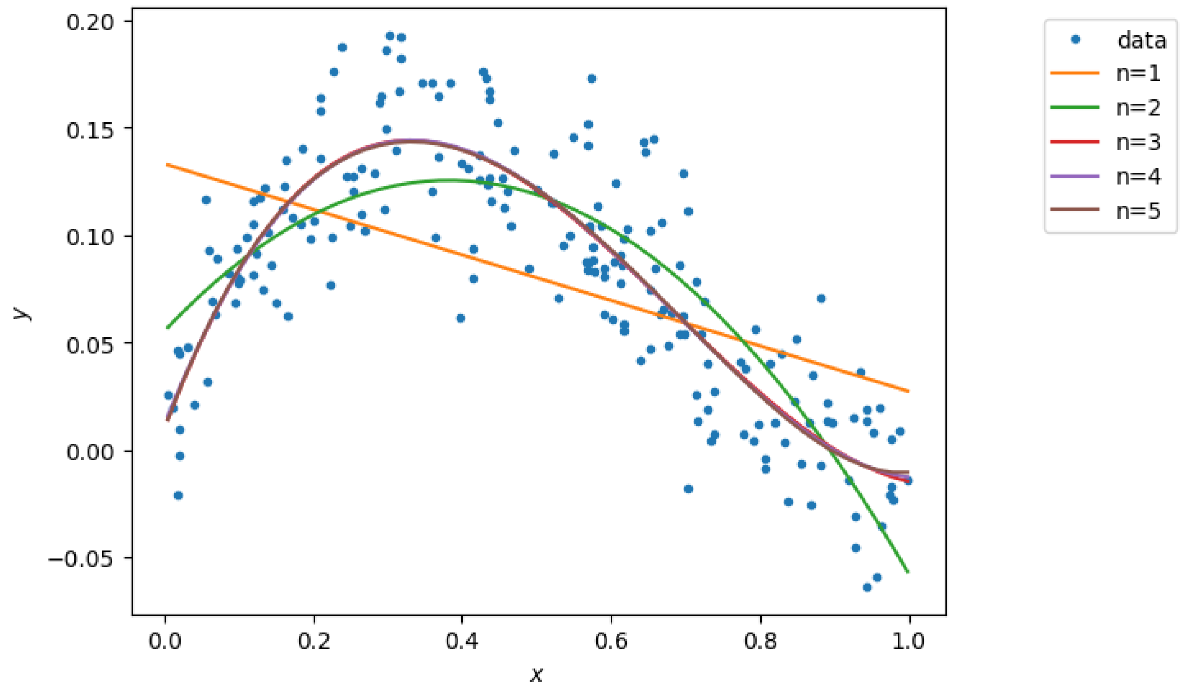
(2) We can increase the model's complexity to better fit the data. In this case, we can increase the order of the polynomial of the model.

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 [32]:
```
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 polynomi
#   i.e., thetas[0] is equivalent to theta above.
#   i.e., thetas[1] should be a length 3 np.array with the coefficients of t
#   ... etc.

xhat = np.vstack((x, np.ones_like(x)))
xhats.append(xhat)
for i in range(2, N+1):
    xhats.append(np.vstack((x**i, xhats[-1])))

for xhat in xhats:
    thetas.append(y.dot(xhat.T).dot(np.linalg.inv(xhat.dot(xhat.T))))

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

In [33]:
```
# 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 (5 points)

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

In [41]:
```
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 c
for i in range(N):
    y_pred = thetas[i].dot(xhats[i])
    training_errors.append(np.mean((y - y_pred)**2))

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

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

```
Training errors are:
 [0.0023799610883627007, 0.001092492220926853, 0.0008169603801105373, 0.000
816535373529698, 0.0008161479195525296]
```

QUESTIONS

(1) What polynomial has the best training error?

(2) Why is this expected?

ANSWERS

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

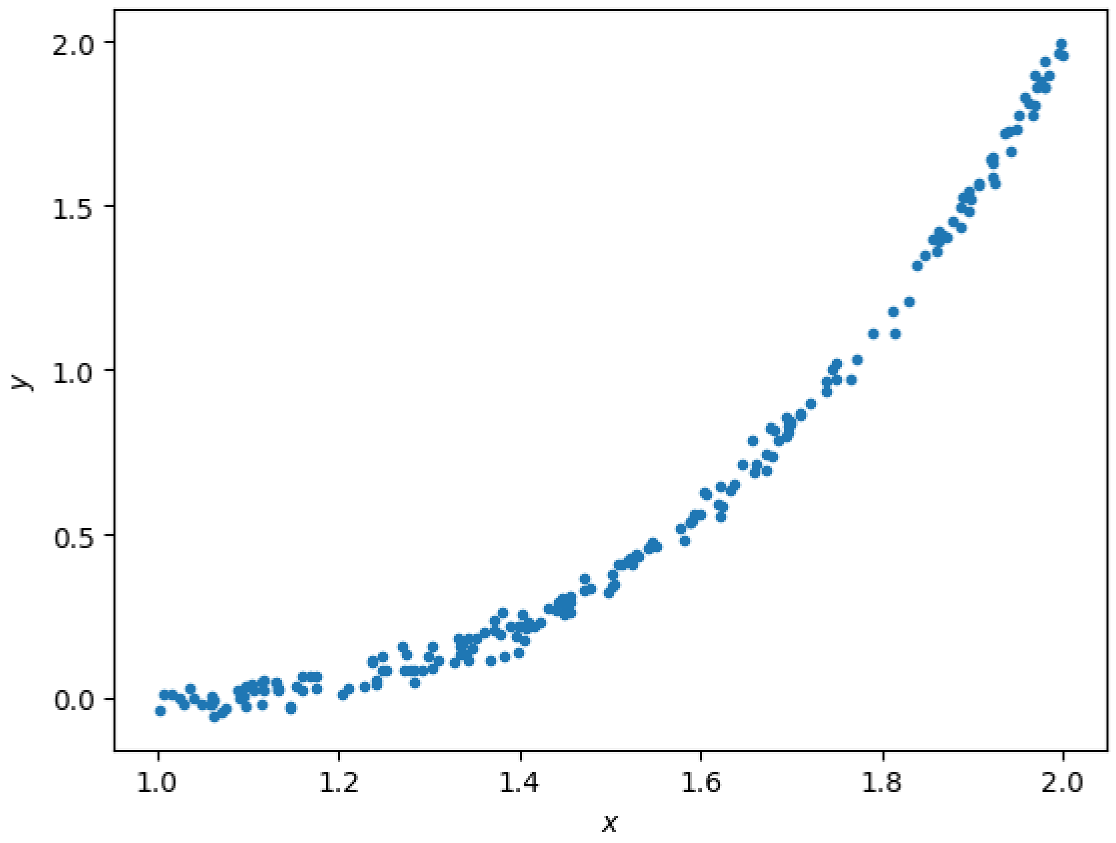
(2) This is the expected result because typically, the more complicated a model is, the better it fits the training data. In this case, complexity takes the form of higher-order polynomials. Furthermore, higher-order polynomials are guaranteed to do at least as well as their lower-order counterparts -- lower-order models are the same as higher-order polynomials, only with the higher-order terms zeroed out. Thus, the added complexity can only help the model's performance.

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 [42]:
```python
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('$y$')
```

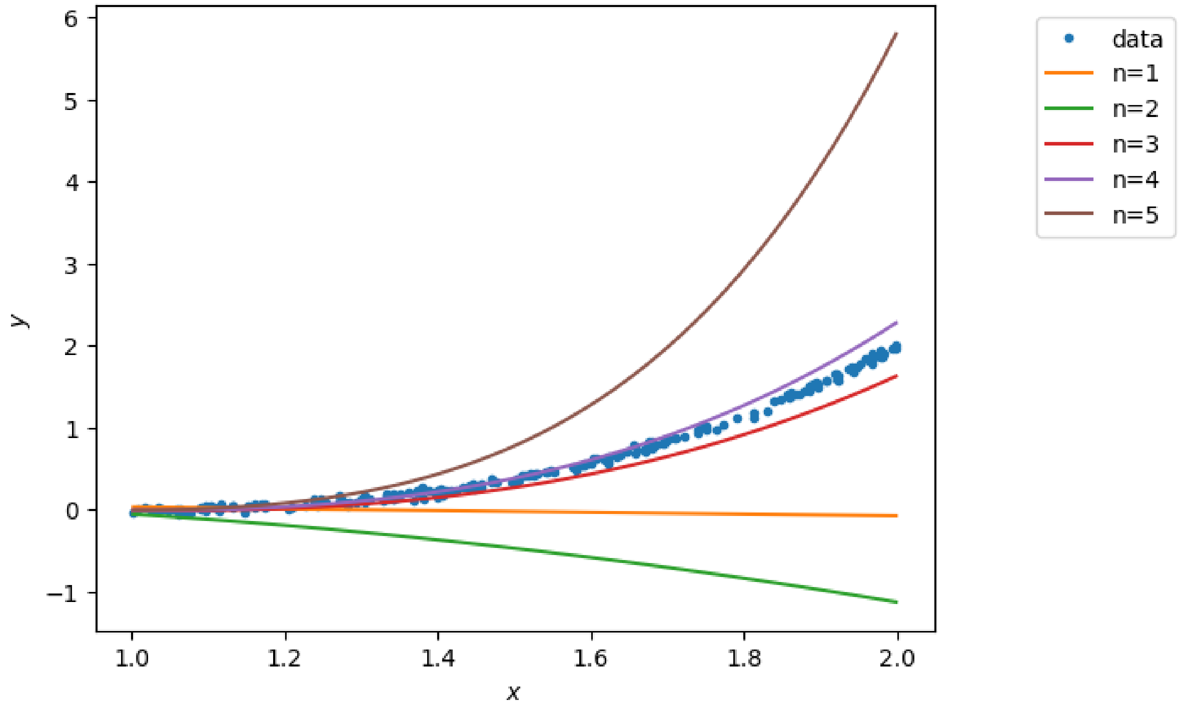
Out[42]:
```
Text(0, 0.5, '$y$')
```



In [43]:
```python
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 [44]:
```python
# 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 [47]:
```
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 orc
for i in range(N):
    y_pred = thetas[i].dot(xhats[i])
    testing_errors.append(np.mean((y - y_pred)**2))

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

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

```
Testing errors are:
 [0.8086165184550592, 2.1319192445058595, 0.03125697108310568, 0.0118707651
95994617, 2.1491021828637646]
```

QUESTIONS

(1) What polynomial has the best testing error?

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

ANSWERS

(1) The fourth-order polynomial has the best testing error.

(2) The fifth-order polynomial does not generalize well because it overfits the testing data. Even though it fit the training data the best, it is too complex for the testing data, which it has not seen before.

```
In [ ]:
```