Math 7243 Machine Learning - Fall 2021

Instructor: He Wang

Test 1.

Student Name: ____________________ __________/**50**

Rules and Instructions for Exams:

1. Unless otherwise specified, to receive full credits you must show **all** necessary work. The grading is based on your work shown. Only a final result from computer will receive zero point.

2. You need to finish the exam yourself. Any discussions with the other people will be considered as **academic dishonesty**. **Cheating, Unauthorized Collaboration, and Facilitating Academic Dishonesty are not allowed.** You can read a description of each here http://www.northeastern.edu/osccr/academic-integrity-policy/

3. This is an open exam. You are allowed to look at textbooks, and use a computer.

4. You are **not** allowed to discuss with any other people.

5. You are **not** allowed to ask questions on any internet platform.

6. For programming questions, if there is no specific instruction, you can only use numpy library. You should **not** use any build in function from Scikit-learn or StatsModels libraries.

1. (10 points) Calculate the **gradient** and **Hessian matrix** of the following functions and find the $\mathbf{argmin}_\theta$ of each function. Here the norm $||\ \ ||$ is the standard l_2-norm. You can use any results in the lecture notes.
(1) Let $\vec{b} \in \mathbb{R}^d$ and let $J(\vec{\theta}) = ||\vec{\theta} - \vec{b}||^2$.

$$J(\vec{\theta}) = ||\vec{\theta} - \vec{b}||^2 = (\vec{\theta} - \vec{b})^T(\vec{\theta} - \vec{b}) = \vec{\theta}^T\vec{\theta} - 2\vec{b}^T\vec{\theta} - \vec{b}^T\vec{b}$$

So, the gradient of $J(\vec{\theta})$ is

$$\Delta J(\vec{\theta}) = 2\vec{\theta} - 2\vec{b}$$

The Hessian matrix of $J(\vec{\theta})$ is

$$H(J(\vec{\theta})) = 2I_d$$

$\mathbf{argmin}_\theta(J)$ is $\vec{b}$

(2) Let $X \in \mathbb{R}^{n\times d}$ and $\vec{b} \in \mathbb{R}^d$. Suppose $\text{rank}(X) = d$. Let $F(\vec{\theta}) = ||X\vec{\theta}||^2 + \vec{\theta}^T\vec{b}$.

$$J(\vec{\theta}) = \vec{\theta}^T X^T X\vec{\theta} + \vec{b}^T\vec{\theta}$$

So, the gradient of $J(\vec{\theta})$ is

$$\Delta J(\vec{\theta}) = 2X^T X\vec{\theta} + \vec{b}$$

The Hessian matrix of $J(\vec{\theta})$ is

$$H(J(\vec{\theta})) = 2X^T X$$

$\mathbf{argmin}_\theta(J)$ is $-(2X^T X)^{-1}\vec{b}$

2. (10 points) In this question, you may use Python (with only numpy library) to solve the matrix equation. Consider the following data points

x_1	x_2	y
1.1	2	2.3
2.2	4	4.3
3.1	6	6.3
4.2	8	7.8
5.3	10	9.8

a). Fit a linear model $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ to this dataset when the loss is RSS= $||X\vec{\theta} - \vec{y}||^2$. You should report the best fit function and the RSS cost value.

(1) We will use the formula $\beta = (X^T X)^{-1} X^T \vec{y}$ to do the calculation.
The coefficient we get is $\vec{\theta}$ =array([0.6 , -0.83333333, 1.35833333])
The RSS cost is $RSS(\theta) = ||X\vec{\theta} - \vec{y}||^2 = 0.06667$

b). Fit a linear function to this dataset when the loss is the Ridge Loss $J(\theta) = ||X\vec{\theta} - \vec{y}||^2 + \lambda(\theta_1^2 + \theta_2^2)$ with $\lambda = 1$ and with $\lambda = 10$. You should report the best fit function and the **RSS** cost value. (Warning: Do not put penalty on θ_0)

(2) We will use the formula $\beta = (X^T X + \lambda I)^{-1} X^T \vec{y}$ to do the calculation. However, we don't want to put penalty on β_0.
We need to centralize our data by calculate $x' := x^{(i)} - \overline{x}$ and $y' := y^i - \overline{y}$

—

x.mean(axis=0) =array([3.18, 6.])
y.mean()=6.1
x-x.mean(axis=0) =array([[-2.08, -4.], [-0.98, -2.], [-0.08, 0.], [1.02, 2.], [2.12, 4.]])
y-y.mean()=array([-3.8, -1.8, 0.2, 1.7, 3.7])

—

In this new centralized data, we have the linear model $y' = \beta_1 x_1' + \beta_2 x_2'$ with no intersection.
So, now, we can use the formula to the new data (X, y), and calculate the coefficient $\beta = (X^T X + \lambda I)^{-1} X^T \vec{y}$
$\overline{y} = 6.1$ and
So $y - \overline{y} = \beta_1(x_1 - \overline{x}_1) + \beta_2(x_2 - \overline{x}_2)$.
When $\lambda = 1$, $[\beta_1, \beta_2]$= array([0.35998318, 0.71981341])
The Cost value is 0.0967
When $\lambda = 10$ $[\beta_1, \beta_2]$= array([0.31523096, 0.60886392])
The Cost value is 1.009285

3. (10 points) Consider the data

$x^{(i)}$	0	0.2	0.4	0.6	0.8	1	1.2	1.4
$y^{(i)}$	5.1	6.4	6.1	8.2	9.5	8.6	12	14.8

The data file $\{\vec{x}^{(i)}, y^{(i)}\}$ for $i = 1, 2, ..., n = 8$ is drawn (with noise) from

$$f(x) = \theta_0 + \theta_1 e^x$$

(1) Find a **closed formula** for parameters $\vec{\theta}$ to minimize the RSS loss

$$J(\vec{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - f(x^{(i)})\right)^2$$

The least squares solution is

$$\vec{\theta} = (X^T X)^{-1} X^T \vec{y}$$

where $X = \begin{bmatrix} 1 & \exp(\vec{x}^{(1)}) \\ 1 & \exp(\vec{x}^{(2)}) \\ \vdots & \vdots \\ 1 & \exp(\vec{x}^{(n)}) \end{bmatrix}$ and $\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$

(2) **Find the function** $f(x)$ fitting the data using the result in (1).

Define a new column x=np.exp(x).
Then, use the formula $(X^T X)^{-1} X^T \vec{y}$ to find the $\vec{\theta} = [2.26789, 2.94362]$

$$f(x) = 2.26789 + 2.94362 e^x$$

The cost is 4.54324

4. (10 points) Consider the data

$x^{(i)}$	0	0.2	0.4	0.6	0.8	1	1.2	1.4
$y^{(i)}$	3.2	4.2	5.6	5.2	7.7	8.8	13.9	18.7

The data file $\{\vec{x}^{(i)}, y^{(i)}\}$ for $i = 1, 2, ..., n = 8$ is drawn (with noise) from the function:

$$g(x) = \theta_0 + e^{\theta_1 x}.$$

Fit the data to the function $g(x)$ by minimizing the RSS loss

$$J(\vec{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - g(x^{(i)})\right)^2.$$

(1) Find the **gradient** of the cost function $J(\vec{\theta})$.

$$J(\vec{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - g(x^{(i)})\right)^2 = \sum_{i=1}^{n} \left(y^{(i)} - \theta_0 - e^{\theta_1 x}.)\right)^2.$$

$$\frac{\partial}{\partial \theta_0} = -2 \sum_{i=1}^{n} \left(y^{(i)} - g(x^{(i)})\right)$$

$$\frac{\partial}{\partial \theta_1} = -2 \sum_{i=1}^{n} \left(y^{(i)} - g(x^{(i)})\right) \theta_1 e^{\theta_1 x}$$

(2) Write down the update formula for gradient decent using α for the learning rate.

$$\theta_0^{next} = \theta_0 + \alpha 2 \sum_{i=1}^{n} \left(y^{(i)} - g(x^{(i)})\right)$$

$$\theta_1^{next} = \theta_1 + 2 \sum_{i=1}^{n} \left(y^{(i)} - g(x^{(i)})\right) \theta_1 e^{\theta_1 x}$$

(3) Use gradient decent(GD) to find θ_* to minimize $J(\vec{\theta})$. You should try different learning rates and recording the cost function values to see what is the best α. Turn in any associated computations, your learning rate, cost values, and the parameters.

The model I used is $f(x) = 3 + e^{2x} + noise$

5. Consider the categorical learning problem consisting of a data set with two labels:

Label 1: (contains 6 points)

X_1	0.2	0.6	2	2.6	3.1	3.8
X_2	3.4	1.8	2	2.7	3.5	1.5

Label 2: (contains 5 points)

X_1	-0.7	-2.1	-2.5	-3	-3.9
X_2	-2.9	-2.8	-1.3	-2	-1.5

(1) (7 points) For each label above, the data follow a multivariate normal distribution Normal(μ_i, Σ) where the covariance Σ is the same for both labels. Fit a pair of LDA functions to the labels by computing the covariances Σ, means μ_i, and proportion ϕ of data. You may use Python (with only numpy library)

(a) You should report the values for ϕ, μ_i and Σ.

$\mu_1 = \begin{bmatrix} 2.05 \\ 2.48 \end{bmatrix}$

$\mu_2 = \begin{bmatrix} -2.44 \\ -2.1 \end{bmatrix}$

$\Sigma = \dfrac{1}{11-2} \sum_{i=1}^{1} 1(x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T = \begin{bmatrix} 1.732 & -0.4025 \\ -0.4025 & 0.636 \end{bmatrix}$

$\phi = 6/11$

(b) Give the **formula for the line** forming the decision boundary.

The line forming the discretion boundary is $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ such that $\log(\dfrac{P(label=1|\vec{x})}{P(label=2|\vec{x})})$

$P(label = k|\vec{x}) = \dfrac{P(\vec{x}|y=k)P(y=k)}{P(\vec{x})}$

Simplify the calculation with constants, we have the equality

$$\vec{x}^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \log(\phi_1) = \vec{x}^T\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 + \log(\phi_2)$$

Plug in the information from (1) we have the line

$$5.21x_1 + 10.68x_2 = 0.62$$

(2) (3 points) For each label above, use **logistic regression** to classify the data. You should report the **logistic function** $p(Y=1|\vec{x}) = \dfrac{1}{1+e^{-\theta^T\vec{x}}}$ and the **formula for the line** forming the decision boundary. (In this question, you can use any Python library including Scikit-learn.)

$$p(y=1|\vec{x}) = \frac{1}{(1+\exp(-(0.2237 + 0.6839x_1 + 0.8666x_2))}$$

The decision boundary is $0.2237 + 0.6839x_1 + 0.8666x_2 = 0$

5. **(continue)**

(3)(2 bonus points) Find the probability $P(y=1|\vec{x})$ for a test point $\vec{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ for both LDA model and the logistics model in the above two questions.

For LDA

$$P(label=1|\vec{x}) = \frac{P(\vec{x}|y=1)P(y=1)}{P(\vec{x})} = \frac{P(\vec{x}|y=1)P(y=1)}{P(\vec{x}|y=1)P(y=1)+P(\vec{x}|y=2)P(y=2)}$$

$$= \frac{\phi_1 \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x}-\mu_1)^T\Sigma^{-1}(\vec{x}-\mu_1)\right)}{\phi_1 \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x}-\mu_1)^T\Sigma^{-1}(\vec{x}-\mu_1)\right) + \phi_2 \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x}-\mu_2)^T\Sigma^{-1}(\vec{x}-\mu_2)\right)}$$

When $\vec{x} = (0,0)$,

$$P(label=1|\vec{x}) = 0.3486$$

For Logistic regression: $p(y=1|\vec{x}) = \frac{1}{(1+\exp(-(0.2237+0.6839x_1+0.8666x_2))} = 0.55$ when $\vec{x} = (0,0)$

(4) (2 bonus points) Find the boundary using the QDA method. (You may use a computer, but only with numpy library)

We assume the covariance Σ_1 and Σ_2 for each label are different. In this case,

$\Sigma_1 = \frac{1}{6-1}\sum_{i=1}^{6}(x^{(i)}-\mu_{y^{(i)}})(x^{(i)}-\mu_{y^{(i)}})^T = \begin{bmatrix} 9.995 & -1.215 \\ -1.215 & 3.883 \end{bmatrix}$

$\Sigma_2 = \frac{1}{5-1}\sum_{i=1}^{5}(x^{(i)}-\mu_{y^{(i)}})(x^{(i)}-\mu_{y^{(i)}})^T = \begin{bmatrix} 5.592 & -2.61 \\ -2.61 & 2.14 \end{bmatrix}$

Simplify the calculation with constants, we have the equality

$$-\frac{1}{2}\log|\Sigma_1| - \frac{1}{2}(\vec{x}-\mu_1)^T\Sigma^{-1}(\vec{x}-\mu_1) + \log(\phi_1) = -\frac{1}{2}\log|\Sigma_2| - \frac{1}{2}(\vec{x}-\mu_2)^T\Sigma^{-1}(\vec{x}-\mu_2) + \log(\phi_2)$$

Plug in the information from (1) we have the quadratic curve