CMSC 25300: Mathematical Methods for Machine Learning

Problem Set 1

Hung Le Tran

06 Oct 2023

Problem 1.2 (Matrix Multiplication)

Solution

(a) Write

$$\mathbf{X} = \begin{bmatrix} 2 & 2 & 0.5 & 0 & 0 \\ 1 & 2 & 0.5 & 1 & 0 \\ 1 & 0 & 1 & 3 & 3 \end{bmatrix}$$

The $i-$th row of $\mathbf{X}$ represents the ingredients needed by the $i-$th meal. The $j-$th column of $\mathbf{X}$ represents the amount of $j-$th ingredient needed across all meals.
Meals in order: Omelette, pancakes, muffins
Ingredients in order: Eggs, milk, butter, flour, berries.
(b) Define

$$\mathbf{w} = \begin{bmatrix} 1 \\ 0.2 \\ 0.5 \\ 0.1 \\ 0.4 \end{bmatrix}$$

then

$$\mathbf{Y} = \mathbf{X}\mathbf{w} = \begin{bmatrix} 2 & 2 & 0.5 & 0 & 0 \\ 1 & 2 & 0.5 & 1 & 0 \\ 1 & 0 & 1 & 3 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 0.2 \\ 0.5 \\ 0.1 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 2.65 \\ 1.75 \\ 3 \end{bmatrix}$$

(c) For one portion of omelette and pancakes,

$$\mathbf{Z} = \mathbf{Y} \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} = 4.4$$

(d) For 3 portions of every meal,

$$\mathbf{Z} = \mathbf{Y} \begin{bmatrix} 3 & 3 & 3 \end{bmatrix} = 22.2$$

(e)

```python
import numpy as np
import pandas as pd

X = np.array([[2, 2, 0.5, 0, 0], [1, 2, 0.5, 1, 0], [1, 0, 1, 3, 3]])

w = np.array([1, 0.2, 0.5, 0.1, 0.4])

Y = np.matmul(X, w)
print(Y)

w1 = np.array([1, 1, 0])
w2 = np.array([3, 3, 3])

print(np.matmul(Y, w1))
print(np.matmul(Y, w2))

```

$\square$

Problem 1.3

Solution

(a) Yes.

$$\mathbf{X}\mathbf{w} = \begin{bmatrix} 4 & 1 & 1 \\ -3 & 2 & 2 \\ 1 & -3 & 2 \\ 5 & 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 9 \\ 7 \\ 1 \\ 16 \end{bmatrix} = \mathbf{y}$$

(b) $w_i = 1$, all other 0.
(c) $w_i = a, w_j = b$, all other 0.

(d) $w_i = 1$, all other 0.
(e) $w_2 = 10, w_1 = -1$, all other 0.
(f)

$$\mathbf{XB} = \begin{bmatrix} 4 & 1 & 1 \\ -3 & 2 & 2 \\ 1 & -3 & 2 \\ 5 & 1 & 3 \end{bmatrix} \begin{bmatrix} 0 & 1 & -1 & 1 \\ 2 & 3 & 1 & 1 \\ 0 & 2 & -2 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 9 & -5 & 7 \\ 4 & 7 & 1 & 3 \\ -6 & -4 & -8 & 2 \\ 2 & 14 & -10 & 12 \end{bmatrix}$$

(g)

```python
import numpy as np
X = np.array([[4, 1, 1], [-3, 2, 2], [1, -3, 2], [5, 1, 3]])
w = np.array([1, 2, 3])

# part a
Y = np.matmul(X, w)
print(Y)

# part b, c, d, e
w_b = np.array([0, 1, 0, 0]) # 2nd row
w_c = np.array([2, 3, 0, 0]) # 2 * 1st row + 3 * 2nd row
w_d = np.array([0, 0, 1]) # 3rd column
w_e = np.array([-1, 10, 0]) # 10 * 2nd column - 1st column

print(np.matmul(w_b, X))
print(np.matmul(w_c, X))
print(np.matmul(X, w_d))
print(np.matmul(X, w_e))

# part f
B = np.array([[0, 1, -1 ,1], [2, 3, 1, 1], [0, 2, -2, 2]])
XB = np.matmul(X, B)
print(XB)

```

□

Problem 1.4

Solution

(a) Rank of aa^T is 1. There exists $w = a^T$ such that $aa^T = aw$, and a is of shape 3×1.
(b) The third column is the first column scaled by (-0.5), while the second column is not. Therefore the matrix has rank = 2.
More concretely, we can point out U, V of shape 4×2 and 2×3 respectively:

$$U = \begin{bmatrix} 1 & 5 \\ 3 & 3 \\ 9 & 1 \\ 4 & 10 \end{bmatrix}, V = \begin{bmatrix} 1 & 0 & -0.5 \\ 0 & 1 & 0 \end{bmatrix}$$

and UV trivially equals to the given matrix. $\square$

Problem 1.5

Solution

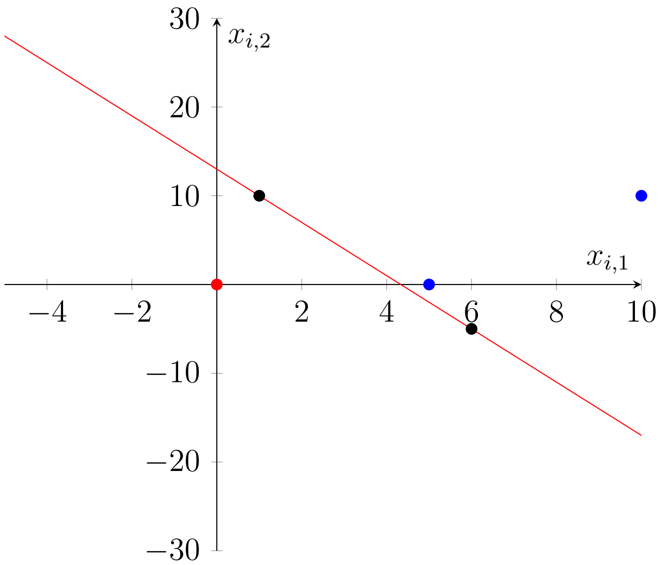
Since (1, 10) and (6, -5) are on the decision boundary, their predicted label is 0. Combining this with the label of (5, 0), we have the system of equations:

$$\begin{aligned} w_1 + 10w_2 + w_3 &= 0 \\ 6w_1 - 5w_2 + w_3 &= 0 \\ 5w_1 + w_3 &> 0 \end{aligned}$$

which implies $w_1 = 3w_2 \equiv 3t, \Rightarrow w_3 = -w_1 - 10w_2 = -13t$. The last condition requires $15t - 13t > 0 \Rightarrow t > 0$

There is an infinite number of $\mathbf{w}$ that satisfies the above conditions, but it is unique up to positively scaling $\begin{bmatrix} 3 \\ 1 \\ -13 \end{bmatrix}$.

Plot:



For $x = (x_{i,1}, x_{i,2})$ above the red decision boundary, the model would predict $+1$.
Correct classifications:
(0, 0), below the line, $\hat{y} = 0 + 0 - 13 = -13 < 0$
(10, 10), above the line, $\hat{y} = 30 + 10 - 13 > 0$

$\square$

Problem 1.6

Solution

(a)

$$p(\mathbf{z_i}) = w_1 z_{i,1}^2 + w_2 z_{i,1} + w_3 z_{i,2}^2 + w_4 z_{i,2} + w_5 z_{i,1} z_{i,2} + w_6$$

(b) Each row of $\mathbf{X}$:

$$\mathbf{x_i} = \begin{bmatrix} z_{i,1}^2 & z_{i,1} & z_{i,2}^2 & z_{i,2} & z_{i,1}z_{i,2} & 1 \end{bmatrix}$$

and

$$\mathbf{X} = \begin{bmatrix} --\mathbf{x_1}-- \\ --\mathbf{x_2}-- \\ \vdots \\ --\mathbf{x_n}-- \end{bmatrix}$$

then with

$$\mathbf{w} = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 \end{bmatrix}$$

we can get predictions $\mathbf{y} = \mathbf{X}\mathbf{w}$

(c)

```
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
# n = number of points
# z = points where polynomial is evaluated
# p = array to store the values of the interpolated polynomials
n = 100
z_1 = np.linspace(-1, 1, n)
z_2 = np.linspace(-1, 1, n)
w_size = 6
w = np.random.rand(w_size)
X = np.zeros((n, w_size))
# TODO : generate X - matrix

X = np.zeros((w_size, n))
X[0] = z_1 * z_1
X[1] = z_1
X[2] = z_2 * z_2
X[3] = z_2
X[4] = z_1 * z_2
X[5] = np.ones(n)

X = X.T

# TODO : evaluate polynomial at all points z
p = np.matmul(X, w)
# and store the result in p
# do NOT use a loop for this
# plot the datapoints and the best - fit polynomials
fig = plt.figure()
# syntax for 3 - D projection
ax = plt.axes (projection = "3d")
ax.plot3D (z_1, z_2, p, "green")
```

```
    ax.set_xlabel("z_1")
    ax.set_ylabel("z_2")
    ax.set_zlabel("y")

    ax.set_title("polynomial with coefficients w =% s "% w)
    plt.show()

```

□