Topic02a : Prelim Problem Set I

Case 1

Represent the following representations into its vectorized form using LaTeX.

> **Problem 1.a. System of Linear Equations**
>
> $$\begin{cases} -y + z = \frac{1}{32} \\ \frac{1}{2}x - 2y = 0 \\ -x + \frac{3}{7}z = \frac{4}{5} \end{cases}$$
>
> **Problem 1.b. Linear Combination**
>
> $$\cos{(\theta)}\hat{i} + \sin{(\theta)}\hat{j} - \csc{(2\theta)}\hat{k}$$
>
> **Problem 1.c. Scenario**
>
> A conference has 200 student attendees, 45 professionals, and has 15 members of the panel. There is a team of 40 people on the organizing committee. Represent the *percent* composition of each *attendee* type of the conference in matrix form.

Express your answers in LaTeX in the answer area.

Solution

> **Problem 1.a. System of Linear Equations (Solution)**
>
> $$-y + z = \frac{1}{32}$$
> $$-y = \frac{1}{32} - z$$
> $$\frac{y}{-1} = \frac{1/32}{-1} - \frac{z}{-1}$$
> $$y = -\frac{1-32z}{32}$$
> $$Substitute: y = -\frac{1-32z}{32}$$
> $$\frac{1}{2}x - 2(-\frac{1-32z}{32}) = 0$$
> $$\frac{1}{2}x + (\frac{1-32z}{16}) = 0$$
> $$\frac{1}{2}x = (-\frac{1-32z}{16})$$
> $$2.\frac{1}{2}x = 2.(-\frac{1-32z}{16})$$
> $$x = -\frac{1-32z}{8}$$
> $$Substitute: x = -\frac{1-32z}{8}$$

```
import numpy as np
```

> **Problem 1.a. System of Linear Equations**

```
A = np.array([[-1, 1],[1/2, 2],[-1, 3/7]])
A
```

```
array([[-1.        ,  1.        ],
       [ 0.5       ,  2.        ],
       [-1.        ,  0.42857143]])
```

```
B = np.array([1/32, 4/5])
B
```

```
array([0.03125, 0.8    ])
```

```
# X = np.linalg.inv(A).dot(B)
np.dot(A,B)
```

```
array([0.76875   , 1.615625  , 0.31160714])
```

> **Problem 1.b. Linear Combination**

```
theta = np.radians(30)
R = np.array([[np.cos(theta), np.sin(theta),np.arcsin(2*theta)]])
R
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: RuntimeWarning: invalid

array([[0.8660254, 0.5      ,       nan]])
```

> **Problem 1.c. Scenario**

```
Conference = np.array([200,45,15,40])
percentage = Conference/Conference.sum(axis=0)*100
print(percentage)
```

```
[66.66666667 15.          5.         13.33333333]
```

Attendees	Number of People	Percentage
$Attendees$	200	66.6%
$Professionals$	45	15%
$Panels$	15	5%
$Org\ Committee$	40	13.33%

Case 2

> **Problem 2.a: Vector Magnitude**

> The magnitude of a vector is usually computed as:
> $$||v|| = \sqrt{a_0^2 + a_1^2 + \ldots + a_n^2}$$
> Whereas v is any vector and a_k are its elements wherein k is the size of v. Re-formulate $||v||$ as a function of an inner product. Further discuss this concept and provide your user-defined function.

> **Problem 2.b: Angle Between Vectors**

> Inner products can also be related to the Law of Cosines. The property suggests that:
> $$u \cdot v = ||u|| \cdot ||v|| \cos(\theta)$$
> Whereas u and v are vectors that have the same sizes and θ is the angle between u and v.

> Explain the behavior of the dot product when the two vectors are perpendicular and when they are parallel.

Solution

> **Problem 2.a: Vector Magnitude**

```
def vector_magnitude(x):
  return np.sqrt(sum(i**2 for i in x))
```

```
a = 5*np.random.randn(6)
a
```

```
array([ 3.93964854, -7.15139717, -3.86628941, -1.8916867 , -0.48208711,
       -3.90560029])
```

```
np.linalg.norm(a)
```

```
10.03374835526016
```

```
vector_magnitude(a)
```

```
10.03374835526016
```

> **Problem 2.b: Angle Between Vectors**

```
def angle_vectors(a,b):
  inner = np.inner(a, b)
  norms = np.linalg.norm(a) * np.linalg.norm(b)

  cos = inner / norms
  rad = np.arccos(np.clip(cos, -1.0, 1.0))
  deg = np.rad2deg(rad)
  print("Radiant: ", rad)
  print("Degree: ", deg)
  return deg,rad
```

```
a = np.array([9, 5])
b = np.array([-6, 2])
print(a,b)
```

```
[9 5] [-6  2]
```

```
angle_vectors(a,b)
```

```
Radiant:  2.3127435948008137
Degree:  132.51044707800082
(132.51044707800082, 2.3127435948008137)
```

Case 3

For the final cases analysis we will be looking at series of equations building up a single feed-forward computation of a logistic regression. The case will not require you to learn fully what is logistic regression.

$$X = \begin{bmatrix} —(x^{(1)})^T— \\ —(x^{(2)})^T— \\ \vdots \\ —(x^{(m)})^T— \end{bmatrix}, Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}, \text{and } \theta = \begin{bmatrix} \theta^{(1)} \\ \theta^{(2)} \\ \vdots \\ \theta^{(m)} \end{bmatrix}$$

The dataset X has m entries with n features while Y is the vector containing the groud truths of a the entries of X, and θ are the parameters or weights of the vectors. We first compute the vector product of the dataset and the parameters as:

$$z = x^{(i)}\theta^{(i)} = X \cdot \theta$$

$$\text{Eq. 3.1}$$

We then solve for the hypothesis of the logistic regression alogrithm as:

$$h_\theta(x) = g(z)$$

$$\text{Eq. 3.2}$$

Where g is an acitvation function that maps the values of the hypothesis vector between a range of 0 and 1. We computed the activation as a sigmoid function:

$$g(z) = \frac{1}{1+e^{-z}}$$

$$\text{Eq. 3.3}$$

Finally we compute the loss of the logistic regression algorithm using J. Wheras $J(\theta)$ is a function that computes the logistic loss of the hypothesis with respect to the ground truths y. it is then computed as:

$$J(\theta) = \frac{1}{m}\sum_{i=0}^{m} = [-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

$$\text{Eq. 3.4}$$

Double-click (or enter) to edit

> **Problem 3.a: Matrix Equivalences**
>
> In Eq. 1, z can also be solved as $X \cdot \theta$ which is the vectorized form of $x^{(i)}\theta^{(i)}$. However, it can also be expressed as $\theta^T \cdot X$. Prove the equality of $X \cdot \theta$ with $\theta^T \cdot X$ in this case.
>
> **Problem 3.b: Matrix Shapes**
>
> Determine the shape of h_θ if X has a shape of $(300, 5)$.
>
> **Problem 3.c: Vectorization**
>
> Express $J(\theta)$ into its vectorized form.
>
> **Problem 4.c: Computational Programming (Also Laboratory 2)**
>
> Encode Equations 3.1 to 3.4 as the class `LRegression` wherein:
>
> - `LRegression` should be instantiated with a dataset X, a ground truth vector y, and a parameter vector θ. Each parameter should have a data type of `numpy.array`.

> - It should further have `methods` reflecting to at least the four (4) aforementioned

Problem 3.a: Matrix Equivalences

```
import numpy as np

X = np.exp(50)
theta = np.exp(50)

z = np.dot(X,theta)

z
```

```
2.688117141816135e+43
```

```
X = np.exp(50)
theta = np.exp(50)

z = np.dot(X,theta.T)

z
```

```
2.688117141816135e+43
```

$$X = \begin{bmatrix} —(x^{(1)})^T— \\ —(x^{(2)})^T— \\ \vdots \\ —(x^{(m)})^T— \end{bmatrix} \text{ and } \theta = \begin{bmatrix} \theta^{(1)} \\ \theta^{(2)} \\ \vdots \\ \theta^{(m)} \end{bmatrix}$$

$$\text{Computing for the matrix product X}\theta,\text{ we have:}$$

$$X\theta = \begin{bmatrix} —(x^{(1)})^T\theta— \\ —(x^{(2)})^T\theta— \\ \vdots \\ —(x^{(m)})^T\theta— \end{bmatrix} = \begin{bmatrix} —\theta^T(x^{(1)})— \\ —\theta^T(x^{(2)})— \\ \vdots \\ —\theta^T(x^{(m)})— \end{bmatrix}$$

> $X \cdot \theta$ = $\theta^T \cdot X$
> $\begin{bmatrix} —(x^{(1)})^T— \\ —(x^{(2)})^T— \\ \vdots \\ —(x^{(m)})^T— \end{bmatrix} \cdot \begin{bmatrix} \theta^{(1)} \\ \theta^{(2)} \\ \vdots \\ \theta^{(m)} \end{bmatrix} = \begin{bmatrix} \theta^{(1)} & \theta^{(2)} & \dots & \theta^{(m)} \end{bmatrix} \cdot \begin{bmatrix} —(x^{(1)})^T— \\ —(x^{(2)})^T— \\ \vdots \\ —(x^{(m)})^T— \end{bmatrix}$
> $\begin{bmatrix} (x^{(1)})^T\theta^{(1)} + (x^{(2)})^T\theta^{(2)} \dots + (x^{(m)})^T\theta^{(m)} \end{bmatrix}$ =
> $\begin{bmatrix} (x^{(1)})^T\theta^{(1)} + (x^{(2)})^T\theta^{(2)} \dots + (x^{(m)})^T\theta^{(m)} \end{bmatrix}$

Problem 3.b

$$X = \begin{bmatrix} x^1 & x^2 & x^3 & x^4 & x^5 \\ x^6 & x^7 & x^8 & x^9 & x^{10} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x^{296} & x^{297} & x^{298} & x^{299} & x^{300} \end{bmatrix}, \theta = \begin{bmatrix} (\theta^1) \\ (\theta^2) \\ \vdots \\ (\theta^k) \end{bmatrix}$$

$$g(z) = \frac{1}{1+e^{-z}}$$
$$z = X \cdot \theta$$
$$h_\theta(x) = g(X \cdot \theta)$$
$$h_\theta(x) = g(z)$$

$$h_\theta(x) = \begin{bmatrix} \theta^1(x) & \theta^2(x) & \theta^3(x) & \theta^4(x) & \theta^5(x) \\ \theta^6(x) & \theta^7(x) & \theta^8(x) & \theta^9(x) & \theta^{10}(x) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \theta^{296}(x) & \theta^{297}(x) & \theta^{298}(x) & \theta^{299}(x) & \theta^{300}(x) \end{bmatrix}$$

Problem 3.c

$$J(\theta) = \frac{1}{m}\sum_{i=0}^{m} = [-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

$$\begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix} = \frac{1}{m}\begin{bmatrix} \sum_{i=0}^{m}[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] \\ \sum_{i=0}^{m}[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] \\ \sum_{i=0}^{m}[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] \\ \vdots \\ \sum_{i=0}^{m}[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] \end{bmatrix}$$

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_0(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Problem 4.c

```
import numpy as np

class LRegression:
  def _init_(self, dataset_X, truth_vector_y, vector_theta):
    self.dataset_X = dataset_X
    self.truth_vector_y = truth_vector_y
    self.vector_theta = vector_theta

  def vector_magnitude(self):
    vec_mag = np.linalg.norm(self.vector_theta)
    return vec_mag

  def vector_product(self):
    vec_prod = np.dot(self.dataset_X, self.vector_theta)
    return vec_prod

  def sigmoid_function(self):
    sig = 1 / (1 + np.exp(self.vector_theta))
    return sig

  def logistic_loss(self):
    m = self.truth_vector_y.shape[0]
    cos = -(1/m) * np.sum( np.dot(np.log(self.vector_theta), self.truth_vector_y.T) + np.dot(
    return cos
```

0s completed at 8:47 PM