Name: Girish Tripathy
Roll no: 1811072
Subject: Machine learning

Vector and matrices

(1) $y \cdot z \Rightarrow y = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad z = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$

$$y^T z = \begin{bmatrix} 1 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$= 11$$

(2) $xy \Rightarrow x = \underset{2 \times 2}{\begin{bmatrix} 2 & 4 \\ 1 & 3 \end{bmatrix}} \quad y = \underset{2 \times 1}{\begin{bmatrix} 1 \\ 3 \end{bmatrix}}$

$$x \cdot y = \begin{bmatrix} 2 & 4 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

$$= \begin{bmatrix} 14 \\ 10 \end{bmatrix}$$

(3) Yes inverse exist. $\text{Det}(X) \neq 0$

$$\text{Inverse} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \times \frac{1}{|X|}$$

$$= \frac{1}{2} \begin{bmatrix} 3 & -4 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 3/2 & -2 \\ -1/2 & 1 \end{bmatrix}$$

(4) $X = \begin{bmatrix} 2 & 4 \\ 1 & 3 \end{bmatrix}$ Rank = 2

Because both the rows and column are independent to each other.

Calculus

(1) $y = x^3 + x - 5$

$\frac{dy}{dx} = 3x^2 + 1 - 0$

(2) $f(x_1, x_2) = x_1 \sin(x_2) e^{-x_1}$

$$\nabla f(x) = \begin{pmatrix} \partial_{x_1} f \\ \partial_{x_2} f \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \sin x_2 e^{-x_1} \cancel{\frac{dx_1}{dx_1}}^{1} \\ - x_1 \sin x_2 e^{-x_1} \end{pmatrix} \\ x_1 e^{-x_1} \cos x_2 \end{pmatrix}$$

Probability and ~~St~~atistics

$$S = \{1\ 1\ 0\ 1\ 0\} = \{T\ T\ H\ T\ H\}$$

1 $\text{Mean} = \dfrac{\sum_{i=1}^{N} x_i}{N} \quad x_i \in S$

$$= \frac{1+1+0+1+0}{5} = 3/5$$

(2) $\text{Variance} = \dfrac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2$

$\mu \to$ mean $= 3/5$

and $x_i \in S$.

$$= \frac{1}{5}\left[\left(\frac{2}{5}\right)^2 + \left(\frac{2}{5}\right)^2 + \left(\frac{-3}{5}\right)^2 + \left(\frac{2}{5}\right)^2 + \left(\frac{-3}{5}\right)^2\right]$$

$$= \frac{1}{5}\left[\frac{4+4+9+4+9}{25}\right] = \frac{1}{5}\left[\frac{30}{25}\right]$$

$$= \frac{6}{25}$$

(3) $P(H) = P(T) = 1/2$

$$\left(1/2\right)^5 = \frac{1}{32}$$

$$P(T\ T\ H\ T\ H) = P(T) \cdot P(T) \cdot P(H) \cdot P(T) \cdot P(H)$$

(4) We need to maximize the Prob. function

Maximise $_p$ $P(1\ 1\ 0\ 1\ 0)$ for some

Let $P(x=1) = P$ $\quad P(x=0) = 1-p$

$$\boxed{P(\text{sample}) = \prod_{i=1}^{5} p^{x_i} \cdot (1-p)^{(1-x_i)}}$$

we need to maximise this wrt. P.

Ideal way to maximize this is by takin' log.

$$f = \log P(\text{sample}) = \sum_{i=1}^{5}\left[\log\left(p^{x_i}\right) + \log\left((1-p)^{1-x_i}\right)\right]$$

$$\Rightarrow \sum_{i=1}^{5}\left[x_i \cdot \log p + (1-x_i)\log(1-p)\right]$$

$$f = \sum_{i=1}^{5} x_i \cdot (\log P) + \left(n - \sum_{i=1}^{5} x_i\right)\log(1-p)$$

To maximize f wrt P..

$$\frac{df}{dP} = 0$$

$$\frac{d}{dp}\left[\sum x_i (\log p) + \left(n - \sum x_i\right)\log(1-p)\right) = 0$$

$$\left\{ \frac{d}{dp}\log p = \frac{1}{p} \right\}$$

$$\Rightarrow \sum_{i=1}^{n} x_i \cdot \frac{1}{p} - \left(n - \sum_{i=1}^{n} x_i\right) \cdot \frac{1}{1-p} = 0$$

$$= \frac{\sum x_i}{p} - \frac{n + \sum x_i}{1-p} = 0$$

$$= \frac{\sum x_i (1-\cancel{p}) - np + \cancel{\sum x_i p}}{p(1-p)} = 0$$

$$= \frac{\sum x_i - np}{p(1-p)} = 0$$

$$\Rightarrow \boxed{p = \frac{1}{n} \sum_{i=1}^{N} x_i} = \frac{1}{5}(1+1+0+0+1)$$

$$\boxed{p = \frac{3}{5}}$$

(3) • $P(z = T \text{ and } y = b) = 0.1$

• $P(z = T / y = b)$

Conditional Probability

$$P(z=T/y=b) = \frac{P(z=T \text{ and } y=b)}{P(y=b)}$$

$$= \frac{0.1}{0.1+0.15} = \frac{0.1}{0.25} = 0.4$$

O notations

(1) $f(n) = \ln(n)$ $\quad g(n) = \lg(n)$

$\ln(n) = f(n) = O(\lg(n))$ and vice versa

Both are true

(2) $f(n) = 3^n$ $\quad g(n) = n^{100}$

$n^{100} = g(n) = O(f(n))$ is true only

(3) $f(n) = 3^n$ $\quad g(n) = 2^n$

$2^n = g(n) = O(3^n) = O(f(n))$

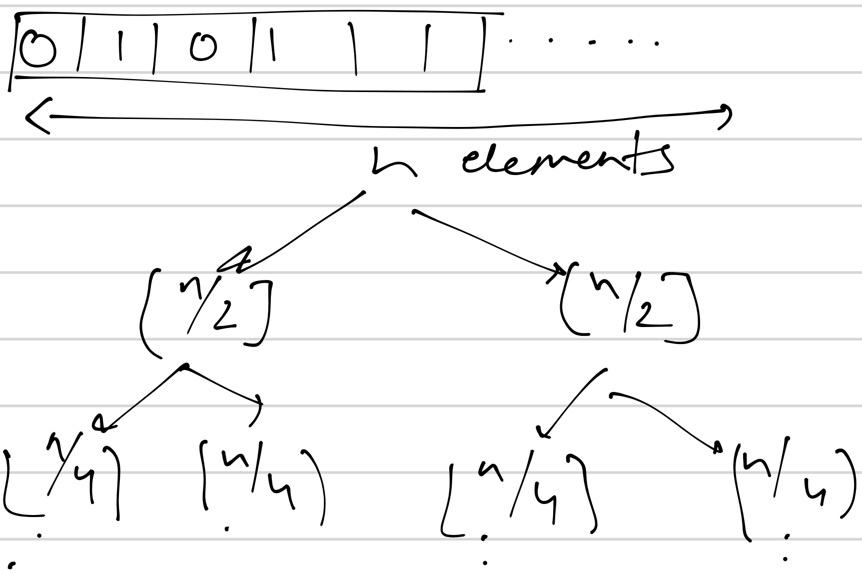
(4) $f(n) = 1000n^2 + 2000n + 4000$

$g(n) = 3n^3 + 1$

$f(n) = O(n^3) = O(3n^3+1)$

$= O(g(n))$.

Algorithm




```
transision(i, j)    # i = 1   j = n

mid = floor[i+j/2] # divides the array
a1 = array[mid]
a2 = array[mid+1]
  If a == b == 1:
    return transisition(i, mid)

else if a == b == 0:
    return transition(mid+1, j)
```

else:
    return mid

here $a = 0$ and $b = 1$
and lastentry $= 0$.

Running time of a Divide and Conquer algo.

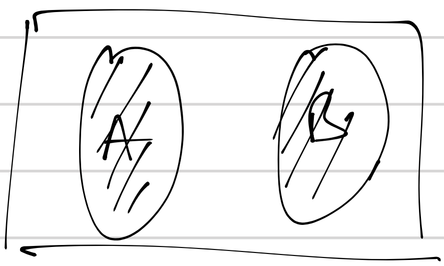
$$T(n) = T(n/2) + O(1)$$

for $n \leq 4 \Rightarrow T(n) = c$

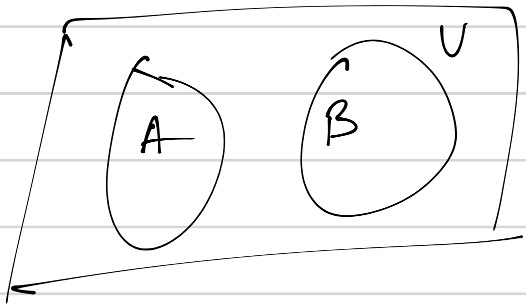
$\Rightarrow O(\log n)$.

Probability

(a) $P(A \cup B) = P(A \cap (B \cap A^c))$



$A \cup B$



$A^c = U - A$

$B \cap A^c = B$

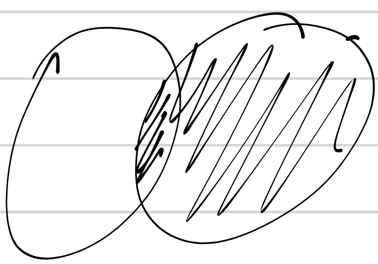
$A \cup B \neq A \cap B$

False

(b) $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

True

(c) $P(A) = P(A \cap B) + P(\underbrace{A^c \cap B}_{B})$



$\neq P(A)$

False

(d) $P\left(\frac{B}{A}\right) = P\left(\frac{A}{B}\right)$

$$P\left(\frac{B}{A}\right) = \frac{P(B \cap A)}{P(A)} \neq$$

$$P\left(\frac{A}{B}\right) = \frac{P(A \cap B)}{P(B)}$$

False

(e) $P(A_1 \cap A_2 \cap A_3)$

$$= P(A_3 \mid A_2 \cap A_1) P(A_2 \mid A_1) P(A_1)$$

$$RHS = \frac{P(A_3 \cap A_2 \cap A_1)}{\cancel{P(A_2 \cap A_1)}} \times \frac{\cancel{P(A_2 \cap A_1)}}{P\cancel{A_1}} \times P\cancel{A_1}$$

$= LHS$

True..

Discrete and Cont. Distribution

(a) ⟶ (h)
(b) ⟶ (e)
(c) ⟶ (f)
(d) ⟶ (g)

Mean Variance and entropy

(a) $Var(x) = E[(x - E(x))]^2$

$= E[x^2 + E(x)^2 - 2x E(x)]$

$= E[x^2] + E.[E(x)^2] - E.[2x E(x)]$

$= E[X^2] + (E(x))^2 - 2E[X \cdot E(x)]$

$= E[x^2] + [E(x)]^2 - 2E[x^2]$

$= \underline{E[x]^2 - E[x^2]}$

(b) Bernaulli random variable

Success p 1
failure $(1-p)$ 0

mean $= p(1) + (1-p)0$

$\boxed{\mu = p}$

Variance

$\sigma^2 = (1-p)(\underbrace{0-p}_{\text{distance from mean}})^2 + p(1-p)^2$

$= p^2 \cdot (1-p) + p(1-p)^2$

$= (1-p)(p^2 + p(1-p))$

$= (1-p)(\cancel{p^2} + p - \cancel{p^2})$

$\boxed{\sigma^2 = p(1-p)}$

Standard deviation

$\sqrt{\sigma^2} = \sqrt{p(1-p)}$

Eentropy $= -\log(p^p \cdot (1-p)^{(1-p)})$

$= -\log(p^p) - \log(1-p)^{1-p}$

$= -p\log p - (1-p)\log(1-p)$

Law of large Number and CL Theorem

(a) due to law of large number theorem which means their sample mean approaches their theoretical mean if large numbers are there.

(b) Central limit theorem

Linear algebra

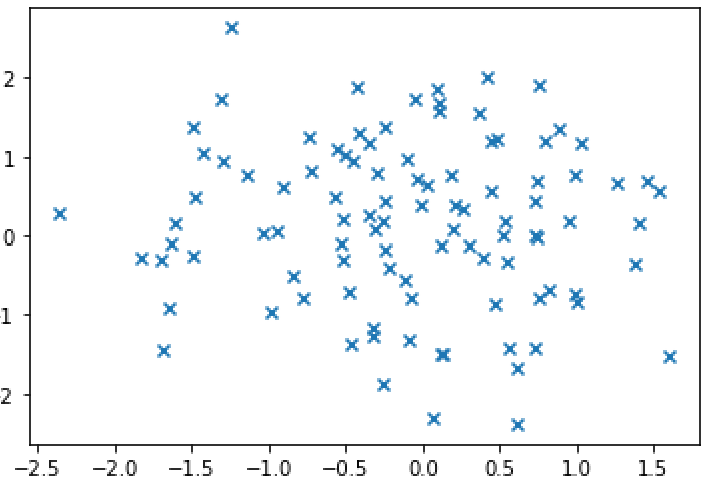
(-_-)

PROGRAMMING SKILLS

In [5]:
```
import numpy as np
from matplotlib import pyplot as plt
```

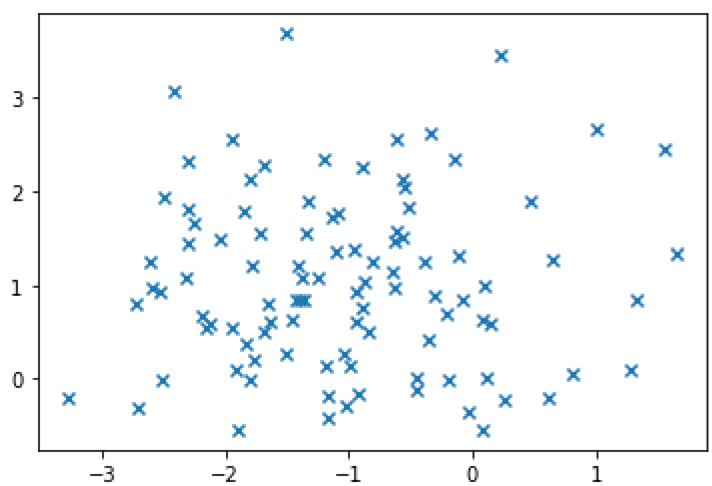
(a)

In [16]:
```
mean = [0 , 0]
cov = [[1,0],[0,1]]
x,y = np.random.multivariate_normal(mean,cov,100).T
plt.scatter(x,y,marker='x')
plt.show()
```



(b)

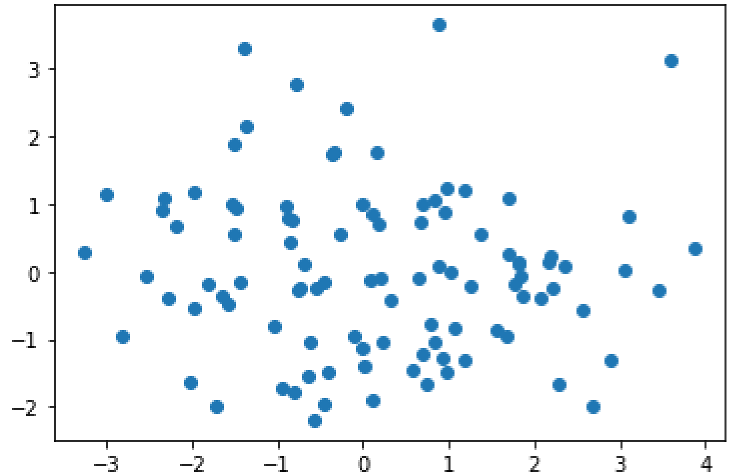
In [18]:
```
mean = [-1 , 1]
cov = [[1,0],[0,1]]
x,y = np.random.multivariate_normal(mean,cov,100).T
plt.scatter(x,y,marker='x')
plt.show()
```



(c)

In [20]:

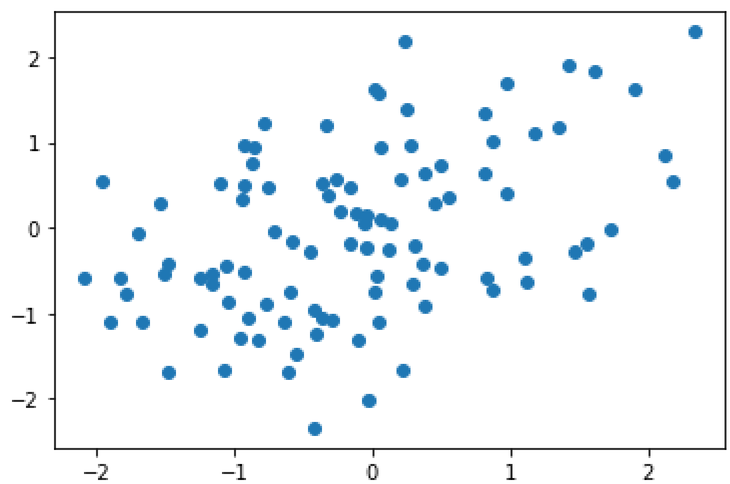
```
mean = [0,0]
cov = [[2,0],[0,2]]
x,y = np.random.multivariate_normal(mean,cov,100).T
plt.scatter(x,y,marker='o')
plt.show()
```



(d)

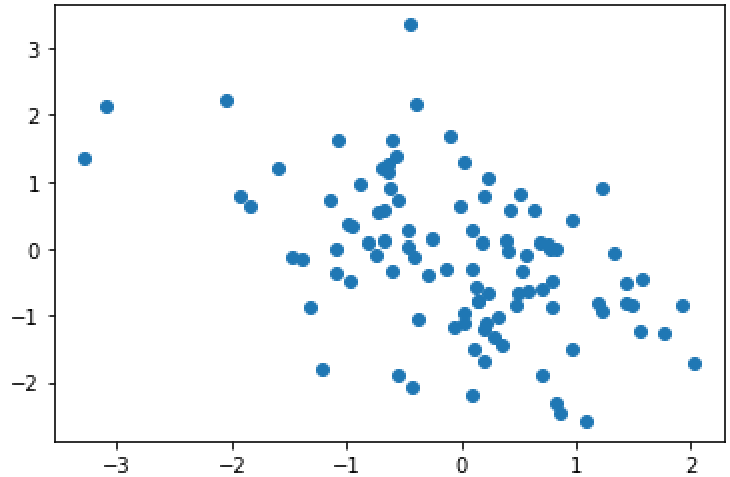
In [22]:

```
mean = [0,0]
cov = [[1,0.5],[0.5,1]]
x,y = np.random.multivariate_normal(mean,cov,100).T
plt.scatter(x,y,marker='o')
plt.show()
```



(e)

In [24]:
```
mean = [0,0]
cov = [[1,-0.5],[-0.5,1]]
x,y = np.random.multivariate_normal(mean,cov,100).T
plt.scatter(x,y,marker='o')
plt.show()
```



In [ ]:
```
```