

```
In [1]: import random
import numpy as np
from IPython.display import clear_output
import csv
import pandas as pd
import time
import matplotlib.pyplot as plt
```

Task 1

```
In [2]: def read_all_documents(path, D):
corpus = []
for f in range(1,D+1):
    file_path = f'{path}/{f}'
    with open(file_path, 'r') as file:
        corpus.append(file.read().split())
return corpus
```

```
In [3]: def LDA(corpus,K):
N_iters=500
N_words=0
a=5/K
b=0.01
for i in corpus:
    N_words=N_words+len(i)
vocab=[]
for i in corpus:
    for j in i:
        if j not in vocab:
            vocab.append(j)
pi_n=random.sample(range(0,N_words),N_words)
w_n=[]
for i in corpus:
    for j in i:
        w_n.append(vocab.index(j))
d_n=[]
for i in range(len(corpus)):
    for j in range(len(corpus[i])):
        d_n.append(i)
z_n=[]
for i in range(N_words):
    z_n.append(random.randint(0,K-1))
w_n=np.array(w_n)
z_n=np.array(z_n)
d_n=np.array(d_n)
c_d = np.zeros((D, K))
c_t = np.zeros((K, len(vocab)))
P=np.zeros(K)
for i in range(c_d.shape[0]):
    for j in range(c_d.shape[1]):
        c_d[i,j] = np.sum(z_n[np.where(d_n==i)[0]]==j)

for i in range(c_t.shape[0]):
    for j in range(c_t.shape[1]):
        c_t[i,j] = np.sum(w_n[np.where(z_n==i)[0]]==j)

for i in range(N_iters):
    for n in range(N_words):
        word=w_n[pi_n[n]]
        topic=z_n[pi_n[n]]
        doc=d_n[pi_n[n]]
        c_d[doc][topic]=c_d[doc][topic]-1
        c_t[topic][word]=c_t[topic][word]-1
        for k in range(K):
            P[k]=((c_t[k][word] + b)/(len(vocab)*b + np.sum(c_t[k,:])))*((c_d[doc,k] + a)/(K*a + np.sum(c_d[doc,:])))
        pnorm = sum(P)
        pnorm = pnorm if pnorm>0 else 1
        P = [pp/pnorm for pp in P]
        topic=np.random.choice(np.arange(K),p=P)
        z_n[pi_n[n]]=topic
        c_d[doc][topic]=c_d[doc][topic]+1
        c_t[topic][word]=c_t[topic][word]+1
return(z_n,c_d,c_t,vocab)
```

```
In [4]: path = 'pp4data/artificial'
D = 10
corpus = read_all_documents(path, D)
K=2
ans=(LDA(corpus,K))
```

```
In [5]: path = 'pp4data/20newsgroups/'
D = 200
corpus = read_all_documents(path, D)
K=20
newans=(LDA(corpus,K))
```

```
In [6]: data=[]
for j in range(2):
    newarr=sorted(range(len(ans[2][j])), key=lambda i: ans[2][j][i])[-3:]
    temp=[]
    for i in newarr:
        temp.append(ans[3][i])
    data.append(temp)
```

```
In [7]: f = open('csv_file', 'w')
writer = csv.writer(f)
writer.writerows(data)
f.close()

df=pd.read_csv("csv_file",header=None)
df
```

Out[7]:

	0	1	2
0	bank	dollars	loan
1	river	water	bank

```
In [8]: data=[]
for j in range(20):
    newarr=sorted(range(len(newans[2][j])), key=lambda i: newans[2][j][i])[-5:]
    temp=[]
    for i in newarr:
        temp.append(newans[3][i])
    data.append(temp)
```

```
In [9]: f = open('topicwords', 'w')
writer = csv.writer(f)
writer.writerows(data)
f.close()

df=pd.read_csv("topicwords",header=None)
df
```

```
Out[9]:
```

	0	1	2	3	4
0	mustang	apr	artidle	writes	edu
1	access	pat	shuttle	hst	mission
2	ship	night	life	etc	earth
3	interested	probe	problem	make	find
4	want	make	mail	geico	insurance
5	want	extra	cost	money	don
6	shift	sho	don	shifter	clutch
7	incoming	ics	uci	gif	edu
8	writes	spencer	toronto	edu	henry
9	oort	spacecraft	mars	part	system
10	sci	sky	gov	nasa	space
11	don	apr	article	writes	edu
12	capability	diesels	redesign	option	station
13	find	information	internet	george	science
14	mph	doesn	work	two	power
15	program	long	moon	bill	space
16	seat	toyota	price	cars	engine
17	dealer	speed	manual	ford	car
18	people	high	large	good	time
19	bmw	lights	change	service	oil

```
In [20]: P1=[]
for i in range(10):
    temp=[]
    for k in range(2):
        temp.append((ans[1][i,k] + 5/2)/(5 + np.sum(ans[1][i,:])))
    P1.append(temp)
P1=np.array(P1)
P1
```

```
Out[20]: array([[0.91346154, 0.08653846],
 [0.20175439, 0.79824561],
 [0.75641026, 0.24358974],
 [0.16666667, 0.83333333],
 [0.78205128, 0.21794872],
 [0.13068182, 0.86931818],
 [0.71978022, 0.28021978],
 [0.20967742, 0.79032258],
 [0.6875, 0.3125],
 [0.15853659, 0.84146341]])
```

```
In [21]: P2=[]
for i in range(200):
    temp=[]
    for k in range(20):
        temp.append((newans[1][i,k] + 5/20)/(5 + np.sum(newans[1][i,:])))
    P2.append(temp)
P2=np.array(P2)
P2
```

```
Out[21]: array([[0.10416667, 0.02083333, 0.02083333, ..., 0.02083333, 0.02083333,
 0.10416667],
 [0.05681818, 0.01136364, 0.03863636, ..., 0.05681818, 0.05681818,
 0.58409091],
 [0.14130435, 0.01086957, 0.01086957, ..., 0.01086957, 0.05434783,
 0.01086957],
 ...,
 [0.01666667, 0.01666667, 0.28333333, ..., 0.01666667, 0.08333333,
 0.01666667],
 [0.00438596, 0.60087719, 0.00438596, ..., 0.00438596, 0.00438596,
 0.00438596],
 [0.1712963, 0.00462963, 0.02314815, ..., 0.00462963, 0.06018519,
 0.02314815]])
```

Task 2

```
In [36]: ▶ def bagOfWords(corpus):  
    word=[]  
    for i in corpus:  
        for j in i:  
            word.append(j)  
    count=0  
    d={}  
    for i in word:  
        if i not in d:  
            d[i] = count  
            count=count+1  
    doc = []  
    for i,j in enumerate(corpus):  
        doc.extend([i for k in range(len(j))])  
    doc = np.array(doc)  
    w = []  
    for i in word:  
        w.append(d[i])  
    w = np.array(w)  
    doc_words = np.zeros((200,len(d)))  
    for i in range(doc_words.shape[0]):  
        for j in range(doc_words.shape[1]):  
            doc_words[i,j] = np.sum(w[np.where(doc==i)[0]]==j)  
    return doc_words  
words=bagOfWords(corpus)
```

```
In [37]: ▶ partition = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]  
labels = pd.read_csv(r'pp4data/20newsgroups/index.csv', header=None)  
y=labels[1].to_numpy().flatten()
```

```

In [38]: def discriminative(X,Y):
    N = X.shape[0]
    N_test = int(N/3)
    N_train = N-N_test
    index=[]
    for j in range(N):
        index.append(j)
    discriminative_result = {}
    logisticR_X = np.c_[np.ones(N),X]
    for times in range(1, 31):
        test_index = random.sample(index, N_test)
        train_index = [x for x in index if x not in test_index]
        xtrain = logisticR_X[train_index,:]
        ytrain = Y[train_index]
        xtest = logisticR_X[test_index,:]
        ytest = Y[test_index]
        for pv in partition:
            pv_N = int(pv*N_train)
            p_x = xtrain[:pv_N,]
            p_y = ytrain[:pv_N]
            W_old = np.zeros(p_x.shape[1])
            W_new = np.matrix(np.zeros(p_x.shape[1]))
            d = X.shape[1]
            for iteration in range(100):
                if not(np.linalg.norm(W_new-W_old)**2)/(np.linalg.norm(W_old)**2>0.001):
                    break
                a = np.dot(p_x, W_old)
                y = 1/(1 + np.exp(-a))
                R = np.diag(y*(1-y))
                W_new = W_old - np.dot(np.linalg.inv(0.01*np.identity(d+1) + np.dot(np.dot(p_x.T,R),p_x)),np.dot(p_x.T, (y-p_y)))
                W_old=W_new
            a = np.dot(p_x, W_new)
            y = 1/(1 + np.exp(-a))
            total = np.zeros((d+1,d+1))
            for i in range(y.size):
                total+= (y[i]*(1-y[i]))*np.dot(p_x[i,:], p_x[i,:])
            SN = 0.01*np.identity(d+1) + total
            prob = np.zeros(xtest.shape[0])
            for i in range(prob.size):
                phi = xtest[i,:]
                mu = np.dot(W_new.T, phi)
                sigma_square = np.dot(np.dot(phi.T, SN), phi)
                k = 1/np.sqrt(1 + (np.pi*sigma_square/8))

                prob[i] = 1/(1 + np.exp(-(k*mu)))

            pred_labels = np.where(prob>=0.5,1,0)
            misclassified_instances = np.sum(np.logical_xor(ytest, pred_labels))
            test_error_rate = misclassified_instances/ytest.size

        discriminative_result[(str(times),str(pv))] = test_error_rate

    return(discriminative_result)
def error(generative):
    g_avg = {}

    for p in partition:
        x = np.array([generative[(str(t),str(p))] for t in range(1,31)])
        g_avg[(str(p), "Discriminative")] = (np.mean(x), np.std(x))
    return g_avg

```

```

In [39]: start1 = time.time()
prediction1 = discriminative(P2,y)
end1 = time.time()
diff1 = end1-start1
print("Time taken for topic representation ",diff1," seconds")

start2 = time.time()
prediction2 = discriminative(words,y)
end2 = time.time()
diff2 = end2-start2
print("Time taken for Bag of words ",diff2," seconds")

```

C:\Users\Aaryan Agarwal\AppData\Local\Temp\ipykernel_23112\2718418435.py:25: RuntimeWarning: invalid value encountered in true_divide

```
if not(np.linalg.norm(W_new-W_old)**2)/(np.linalg.norm(W_old)**2>0.001):
```

Time taken for topic representation 0.6286466121673584 seconds

Time taken for Bag of words 16.87462830543518 seconds

```
In [40]: result1=error(prediction1)
         result2=error(prediction2)
```

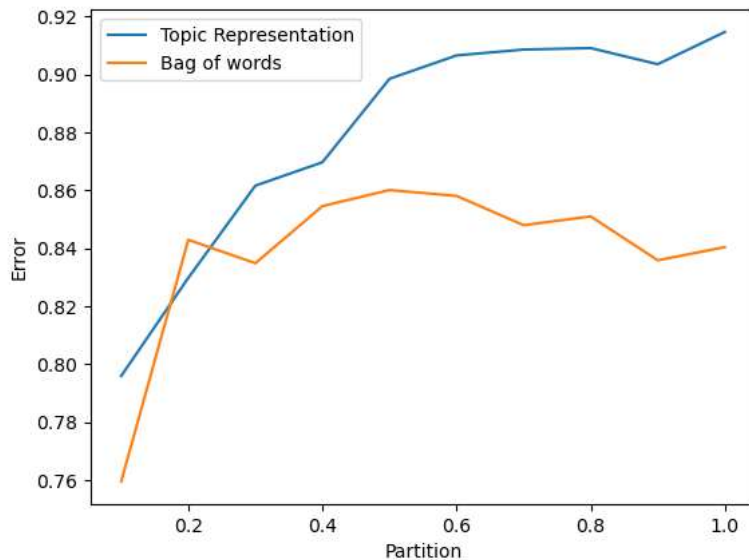
```
In [41]: result1
```

```
Out[41]: {('0.1', 'Discriminative'): (0.20404040404040408, 0.0684040523367152),
          ('0.2', 'Discriminative'): (0.17020202020202022, 0.05170543190616634),
          ('0.3', 'Discriminative'): (0.13838383838383836, 0.05445184635128709),
          ('0.4', 'Discriminative'): (0.13030303030303028, 0.04058042308404175),
          ('0.5', 'Discriminative'): (0.10151515151515151, 0.0391503736894616),
          ('0.6', 'Discriminative'): (0.09343434343434341, 0.02981509523867766),
          ('0.7', 'Discriminative'): (0.09141414141414139, 0.03862563394949271),
          ('0.8', 'Discriminative'): (0.09090909090909088, 0.03477154440011819),
          ('0.9', 'Discriminative'): (0.09646464646464643, 0.035096467986043815),
          ('1', 'Discriminative'): (0.08535353535353533, 0.034877750849736454)}
```

```
In [42]: result2
```

```
Out[42]: {('0.1', 'Discriminative'): (0.2404040404040404, 0.0706059883527325),
          ('0.2', 'Discriminative'): (0.15707070707070706, 0.04924954623451553),
          ('0.3', 'Discriminative'): (0.16515151515151516, 0.045848461481965114),
          ('0.4', 'Discriminative'): (0.14545454545454545, 0.05415121274214227),
          ('0.5', 'Discriminative'): (0.13989898989898989, 0.044892711835682594),
          ('0.6', 'Discriminative'): (0.14191919191919191, 0.04587070991781632),
          ('0.7', 'Discriminative'): (0.15202020202020206, 0.04074665343743424),
          ('0.8', 'Discriminative'): (0.14898989898989898, 0.03412362814466438),
          ('0.9', 'Discriminative'): (0.16414141414141417, 0.034789878988473785),
          ('1', 'Discriminative'): (0.15959595959595962, 0.034712808641292094)}
```

```
In [49]: plt.plot(partition,[1-result1[(str(p),'Discriminative')][0] for p in partition], label = 'Topic Representation')
         plt.plot(partition, [1-result2[(str(p),'Discriminative')][0] for p in partition], label = 'Bag of words')
         plt.xlabel('Partition')
         plt.ylabel('Error')
         plt.legend()
         plt.show()
```



```
In [ ]: 
```