יובל פרידמן ת.ז. 204169320 <u>חישוביות וקוגניציה תר' 1</u>

1.1

קיבולת הרשת היא מספר הזכרונות שאפשר לאחסן כך שתבניות הזיכרון יהיו מצבים יציבים ברשת.

הקיבולת תלויה במס' הנוירונים. נסביר מדוע:

נוירון 1 למשל, יציב כאשר מתקיים $\boxed{\operatorname{sgn}\left(\frac{N-1}{N}+\sum_{k=1}^{(N-1)(P-1)} Z_k\right)=1}$ כפי שראינו בכיתה, כאשר Z_k הוא מ"מ שמייצג את $\underbrace{\frac{1}{N}\rho_i^{\mu}\rho_j^{\mu}\rho_i^{1}\rho_j^{1}}_{i\neq j}$.

ראינו כי $E[Z_k]=0$ ו- $SD(Z_k)=\frac{1}{N}$,

ולמשפט הגבול המרכזי על המחבר בינומי בפונקציית ה-sgn מתפלג נורמלית עם תוחלת 0 וסטיית תקן ששווה בערך $\sqrt{\frac{P}{N}}$ (כאשר N, P גדולים).

כיון שכך, הסיכוי לאי יציבות הנוירון הוא $P\left(\underbrace{\sum_{k=1}^{(N-1)(P-1)} Z_k}_{\sim N\left(0,\frac{P}{N}\right)} < -\frac{N-1}{N} \approx -1\right)$

מקבלים ב- CDF של התפלגות נורמלית ידועה וניתן לחשב נוסחאות לחשב זאת.

כאשר ההתפלגות "צרה", כלומר סטיית התקן קטנה, אז הסיכוי לאי-יציבות קטן גם הוא כי השטח בקטע $[-1,-\infty)$ קטן גם הוא. ולכן להיפך במקרה שסטיית התקן גדלה.

במקרה שלנו, מס' הנוירונים קטן בחצי, ולכן סטיית התקן היא $\sqrt{\frac{P}{N/2}}=\sqrt{2}\sqrt{\frac{P}{N}}$ כלומר גדלה פי $\sqrt{2}$, ולכן הסיכוי לאי-יציבות גדל וקיבולת הרשת קטנה:

מתכונות היחידות של התפלגות נורמלית:
$$\sum_{k=1}^{(N-1)(P-1)} Z_k = \sqrt{\frac{P}{N}}\, Z \qquad (Z\sim N(0,1))$$

(כיון ש- $Var\left(\sum_k Z_k\right)=\frac{P}{N}\cdot 1=Var\left(\sqrt{\frac{P}{N}}Z\right)$ וגם $E\left[\sum_k Z_k\right]=E\left(\sqrt{\frac{P}{N}}Z\right)=0$)

$$\Rightarrow \underbrace{\sum_k Z_k}_{\text{המקרה הרגיל}} = \sqrt{\frac{P}{N}}\, Z$$

נשווה את הסיכוי לטעות בשני המקרים:

$$\Rightarrow P_{error}\left(\sum_k Z_k < -1\right) = P\left(\sqrt{\frac{P}{N}}Z<-1\right) = \Phi\left(-\sqrt{\frac{N}{P}}\right)$$

C.D.F נורמלי סטנדרטי

$$P_{error}^{\text{חדש}}\left(\sum_k Z_k < -1\right) = P\left(\sqrt{2}\sqrt{\frac{P}{N}}Z<-1\right) = \Phi\left(-\frac{1}{\sqrt{2}}\sqrt{\frac{N}{P}}\right)$$

Φ פונק' עולה, הארגומנט במקרה החדש גדל ולכן הסיכוי גדל (בהנחה של $\frac{N}{P}-1$ כפי שהיה קודם)

(2.2)

ננסה לבדוק את יציבות הפתרון ברשת החדשה לאחר התחלה ב-ρ_i^1, נבדוק את

סימון כבשיעור:

$$\rho_i^1 \cdot sgn\left(\sum_{j=1}^{N} J_{ij}\rho_j^1\right) = \rho_i^1 sgn\left(\sum_{\substack{j=1, j\neq i \\ j: J_{ij}\notin Q}}^{(1-q)N} \frac{1}{N}\sum_{\mu=1}^{p} \rho_i^\mu \rho_j^\mu \rho_j^1\right) =$$

נסמן:

$Q \equiv$ קבוצת הסינפסות שהוסרו (=0)

$q = |Q|$

$$= \rho_i^1 sgn\left(\frac{1}{N}\left[\sum_{\substack{j=1 \\ j\neq i \\ j: J_{ij}\notin Q}}^{(1-q)N} \rho_i^1 \underbrace{\rho_j^1 \rho_j^1}_{1} + \sum_{j}\sum_{\mu=2}^{p} \rho_i^\mu \rho_j^\mu \rho_j^1\right]\right) =$$

$$= \rho_i^1 sgn\left(\frac{1}{N}\left[(1-q)(N-1)\rho_i^1 + \sum_{j}\sum_{\mu=2}^{p} \rho_i^\mu \rho_j^\mu \rho_j^1\right]\right) = sgn\left(\frac{N-1}{N}(1-q) + \sum_{\substack{j=1 \\ j\neq i \\ j: J_{ij}\notin Q}}^{(1-q)N} \sum_{\mu=2}^{p} \frac{1}{N}\rho_i^\mu \rho_j^\mu \rho_i^1 \rho_j^1\right)$$

בנשאת sgn-ל ρ_i^1

נתבונן במונח הרעש, כיוון שאנו סוכמים רק על סינפסות שלא הושמטו,

עדין הנ"ל $z_k = \frac{1}{N}\rho_i^\mu \rho_j^\mu \rho_i^1 \rho_j^1$ שהוא נפל כמו קודם $SD[z_k] = \frac{1}{N}$, $E[z_k]=0$.

והמשתנה הגדול כגרלוני $\sum_{k=1}^{(N-1)(1-q)(p-1)} z_k$ מתפלג נורמלית עם תוחלת 0 וסטיית תקן:

$$\sigma = \sqrt{(N-1)(p-1)(1-q)}\,\frac{1}{N} \approx \frac{\sqrt{p(1-q)}}{N}$$

הסיכוי לטעות הוא הסיכוי שהרעש אותו פירשנו יהיה קטן מ-0:

$$\Rightarrow P_{error} = P\left(\sum^{(N-1)(1-q)(p-1)} z_k < -\frac{N-1}{N}(1-q) \approx -(1-q) = q-1\right) =$$

תקנון

$$\Phi\left(\frac{q-1}{\sqrt{p(1-q)/N}}\right) = \Phi\left(\frac{q-1}{\sqrt{1-q}}\cdot\sqrt{\frac{N}{p}}\right) = \Phi\left(-\sqrt{1-q}\cdot\sqrt{\frac{N}{p}}\right) = \boxed{\Phi\left(-\sqrt{\frac{N(1-q)}{p}}\right)}$$

Φ מונוטונית עולה והארגומנט עכשיו גדול יותר מהארגומנט ($-\sqrt{\frac{N}{p}}$ שהיה) בגישה לפני השינוי) ולכן הסיכוי לטעות עכשיו עלה.

משמעות הדבר היא שכשמורידים אחוז q סינפסות קיבלנו רשת קטנה.

Yuval_Friedmann_ex1_computation_and_cognition

November 11, 2019

[0]:
```
import numpy as np
import pandas as pd
import copy
import matplotlib.pyplot as plt
```

Q1

[0]:
```
def memory_patterns(n,p):
  patterns = np.random.choice([-1,1], (n,p))
  connections = np.zeros((n,n))
  for i in range(n):
    for j in range(i+1, n):
        connections[i,j] = np.dot(patterns[i,:], patterns[j,:])/n # loop of mu
        connections[j,i] = connections[i,j] # synapse matrix is symmetric

  np.fill_diagonal(connections,0)
  return patterns, connections
```

Q2

[0]:
```
# connections: j matrix, S: the network init vector which in {-1,+1}^n
def a_synchronic(connections, s, print_updates=False):
  j=0
  while True:
    j+=1
    s_t = copy.deepcopy(s)
    for i in range(connections.shape[0]):
      if s[i]*np.sign(np.dot(connections[i,:], s)) < 0: # update according to
 ↪the rule; sign(0)=1
        s[i] *= -1
    if print_updates:
      print(np.sum(s != s_t),"neurons updated on trail",j)
    if np.all(s == s_t):
      return s
```

Q3

[0]:
```python
def stability(connections, s, z):
  n = connections.shape[0]
  noisier = np.random.choice([-1,1],n,p=[z,1-z])
  s_noisy = s*noisier # change sign of s[i] w.p of z
  s_convergance = a_synchronic(connections=connections,s=s_noisy)
  return (n - np.sum(s_convergance == s))/n
```

Q4

[0]:
```python
def simmulation(n,p,z):
  # pandas apply function convert ints to floats so we have to convert it back
  n = n.astype(np.int)
  p = p.astype(np.int)
  patt, conn = memory_patterns(n,p)
  return stability(conn,patt[:,0],z) # initialize with the first memory pattern

alpha = np.repeat(np.arange(0.02,0.302,step=0.02),5)
n = np.repeat(1000,len(alpha))
z = 0.1
p = np.ceil(n*alpha).astype(np.int)
df_exp = pd.DataFrame({"N":n, "P":p, "alpha":alpha})
```

[6]:
```python
df_exp.head()
```

[6]:

	N	P	alpha
0	1000	20	0.02
1	1000	20	0.02
2	1000	20	0.02
3	1000	20	0.02
4	1000	20	0.02

[0]:
```python
# cell running is approx 105 sec.
df_exp["false_memory"] = df_exp.apply(lambda row: simmulation(n=row["N"],␣
 ↪p=row["P"], z=z), axis=1)
```

[0]:
```python
df_mean_results = df_exp.groupby(by="alpha").mean()
```

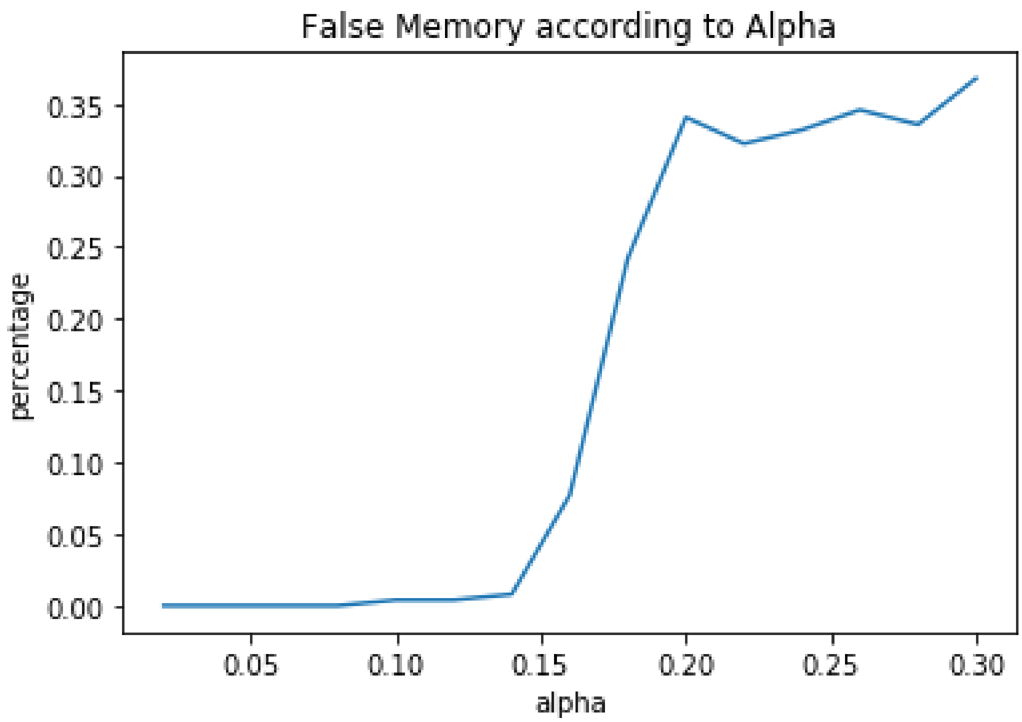
Q5 A glimpse of the area of the critical alpha value in the data frame results

[9]:
```python
df_mean_results.loc[0.1:0.23]
```

[9]:

alpha	N	P	false_memory
0.10	1000	100	0.0040
0.12	1000	121	0.0040
0.14	1000	140	0.0080
0.16	1000	160	0.0772
0.18	1000	180	0.2420
0.20	1000	200	0.3406
0.22	1000	220	0.3222

[10]:
```
plt.plot(df_mean_results.index, df_mean_results["false_memory"])
plt.title("False Memory according to Alpha")
plt.xlabel("alpha")
plt.ylabel("percentage")
plt.show()
```



This graph explains the principle of network capacity showed in class.
For alpha's below approx. 0.14 the network is stable and the error rate is low. From that value of alpha the error increasing exponentialy, and for larger alpha's the error rate converges to relatively high error rate.
Namely, For large alpha's the network is more likely to produce false memories. This is a consequence of the principle showed in class, which says the more natural real memories the network has(dimension of P), it is more likely to converge to false memories.