יובל פרידמן 204169320

<u>חישוביות וקוגניציה תר' 2</u>

1

1.1

$$Var(y) = E[y^2] - E^2[y] = E[(w^Tx)^2] - E^2[(w^Tx)] =$$

$$E[(w^Tx)(x^Tw)] - w^TE^2[x] = w^TE[xx^T]w - w^T\cdot 0 =$$

$$w^TCw$$

1.2

נכתוב את הלגרנז'יאן, נגזור ונשווה לאפס:

$$L(w,c,\mu) = \overset{f(w)}{w^TCw} - \overset{g(w)}{\mu(\|w\|^2-1)}$$

נפתח את הביטוי הימני:

$$w^TCw = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}w_iw_j =$$

נפריד לביטויים שמכילים את w_1 ואלה שאינם:

$$w_1\sum_{j=1}^{n}C_{1j}w_j + w_2\sum_{j=1}^{n}C_{2j}w_j + \dots + w_n\sum_{j=1}^{n}C_{nj}w_j =$$

$$= C_{11}w_1^2 + w_1\sum_{j=2}^{n}C_{1j}w_j + w_2C_{21}w_1 + w_2\sum_{j=2}^{n}C_{2j}w_j + \dots + w_nC_{n1}w_1 + w_n\sum_{j=2}^{n}C_{nj}w_j$$

ולכן הנגזרת לפי w_1 היא:

$$\nabla_{w_1}f = 2C_{11}w_1 + \sum_{j=2}^{n}C_{1j}w_j + C_{21}w_2 + C_{31}w_3 + \dots + C_{n1}w_n =$$

$$2C_{11}w_1 + \sum_{j=2}^{n}C_{1j}w_j + \sum_{j=2}^{n}C_{j1}w_j = 2C_{11}w_1 + \sum_{\substack{j=1\\ j\neq 1}}^{n}(C_{1j}w_j + C_{j1}w_j) =$$

$$2C_{11}w_1 + \sum_{\substack{j=1\\ j\neq 1}}^{n}(C_{1j}+C_{j1})w_j \overset{C\text{ סימטרית}}{=} 2C_{11}w_1 + 2\sum_{\substack{j=1\\ j\neq 1}}^{n}C_{1j}w_j =$$

$$= 2\left(\sum_{j=1}^{n}C_{1j}w_j\right) = 2(Cw)_1$$

וכיון שלא השתמשנו בכך שw_1 היא הקורדינטה במקום הראשון ב-w, ניתן להכליל לכל w_i:

$$\nabla_{w_i}f = 2(Cw)_i$$

ובכתיב וקטורי:

$$\nabla_w f = 2Cw$$

כעת נפתח את הביטוי השמאלי:

$$\mu(\|w\|^2-1) = \mu\left(\sum_{i=1}^{n}w_i^2 - 1\right) \Rightarrow \nabla_{w_i}g = 2w_i\cdot\mu \Rightarrow \nabla_w g = 2\mu w$$

ובתוספת האילוץ קיבלנו:

$$2Cw - 2\mu w = 0 \quad s.t\ \|w\|^2=1 \iff Cw = \mu w$$

כלומר צריך למצוא w כך ש-w הוא ו"ע של C, וכן השונות תהיה:

$$Var(y) = w^T C w = w^T \mu w = \mu w^T w = \mu \|w\|^2 = \mu$$

כלומר, בנוסף לכך השונות תהיה שווה לע"ע של הו"ע שנבחר,
ולכן דרישנו לבחור בו"ע המתאים לע"ע הגבוה ביותר.

1.3

1.3.1

נתון: $X_1 \sim N(0,1)$, $X_2 \sim N(0,4)$, $X_1 \perp X_2$

מטריצת הקורלציה היא מטריצה כך ש:

$$(XX^T)_{ij} = C_{ij} = \begin{cases} Var(x_i) & i=j \\ cov(x_i, x_j) & i \neq j \end{cases}$$

ולכן לפי הנתון:

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$$

1.3.2

צ"ל: ו"ע של הע"ע הגבוה ביותר של C

נמצא פולינום אופייני:

$$p(\lambda) = |C - \lambda I| =$$

$$= \begin{vmatrix} 1-\lambda & 0 \\ 0 & 4-\lambda \end{vmatrix} = (1-\lambda)(4-\lambda) =$$

$$\Rightarrow \lambda_1 = 1, \quad \lambda_2 = 4$$

נמצא את הו"ע המתאים

$$\left[\begin{array}{cc|c} 1-4 & 0 & 0 \\ 0 & 4-4 & 0 \end{array}\right] \Leftrightarrow \left[\begin{array}{cc|c} -3 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \Leftrightarrow$$

$$-3w_1 = 0 \Leftrightarrow w_1 = 0$$

w_2 לכאורה יכול להיות כל ערך, אך נבחר את האילוץ $\|w\|^2 = 1$

ולכן במקרה שלנו $w_2 = 1 \Leftarrow w = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

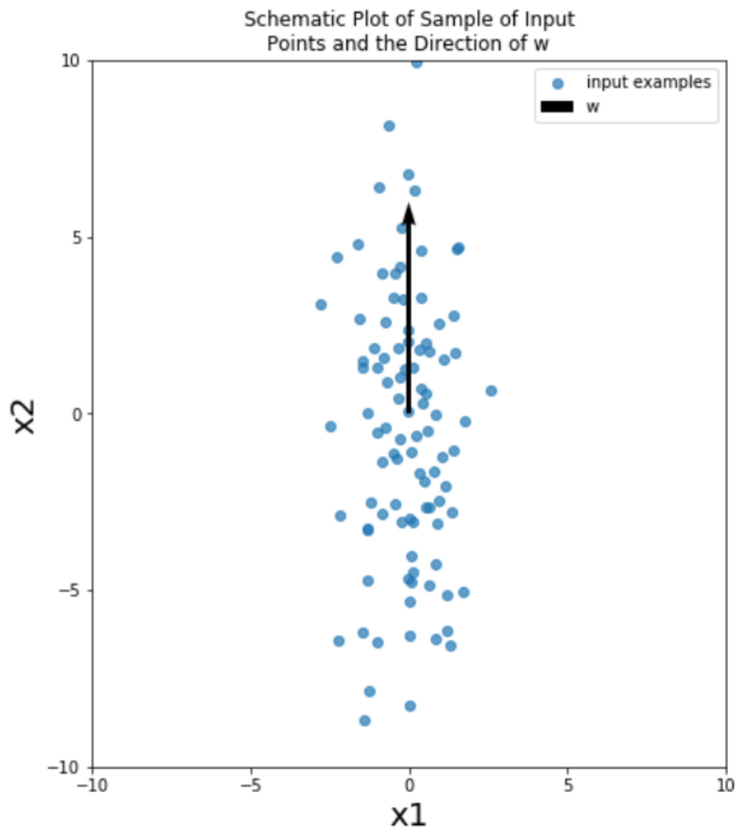
1.3.3 Appendix

1.4 $Var(y) = Var(wx) = w^2 Var(x) = w^2 E(x^2) \Rightarrow$

ככל שניקח w גדול יותר כך שונות הפלט תהיה גדולה יותר!

Appendix

Answer for question 1.3.3



בציור ניתן לראות נקודות שהוגרלו כאשר לממד הראשון שונות קטנה יותר מלממד השני, הוקטור שהתקבל מצביע על הכיוון בו השונות בדוגמאות היא גדולה.

Answer for question 1.5

בכיתה הגדרנו את בעיית הPCA בעזרת פונקציית אנרגיה, ודרשנו למזער את ערכה כפונקציה של הוקטורים u (וקטור הורדת הממד) ו-v (וקטור השחזור). ראינו כי מזעור של הפונקציה מתקבל בעזרת לקיחת u ו-v שווים זה לזה, וכן ע"י בחירתם להיות הוקטור העצמי המתאים לערך העצמי הגבוה ביותר של מטריצת הקורלציה של הקלט. אמנם, ראינו בכיתה כי עשויים להיות כמה וקטורים אופיטמליים עבור מזעור פונקציית האנרגיה, אך בשל כך בחרנו אותם להיות שייכים לקבוצת הוקטורים שהנורמה שלהם היא 1.

זהו בדיוק התהליך אותו הראינו בסעיף ב' בשאלה זו: מצאנו כי שונות הפלט הגדולה ביותר תתקבל על ידי בחירת הוקטור העצמי המתאים לערך העצמי הגבוה ביותר של מטריצת הקורלציה של הדוגמאות. במקרה של סעיף ב' הערך y שאת שונותו אנחנו מנסים למקסם, מייצג את הדוגמאות לאחר הקטנת הממד, ומקסום שונותו שקול למזעורה של פונקציית האנרגיה (המודדת כמה 'מייצגת' היתה הורדת הממד).

ex5_computation_and_cognition

December 12, 2019

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

0.0.1 Q1

[0]:
```
def sample_2d(p,f,sig_x):
  x1 = np.random.uniform(-1,1)
  determinator = np.random.choice([True,False], p=(p,1-p))
  if determinator:
    eps = np.random.normal(0,sig_x)
    x2 = np.sin(f*x1) + eps
  else:
    x2 = np.random.uniform(-1,1)
  return pd.Series([x1, x2])
```

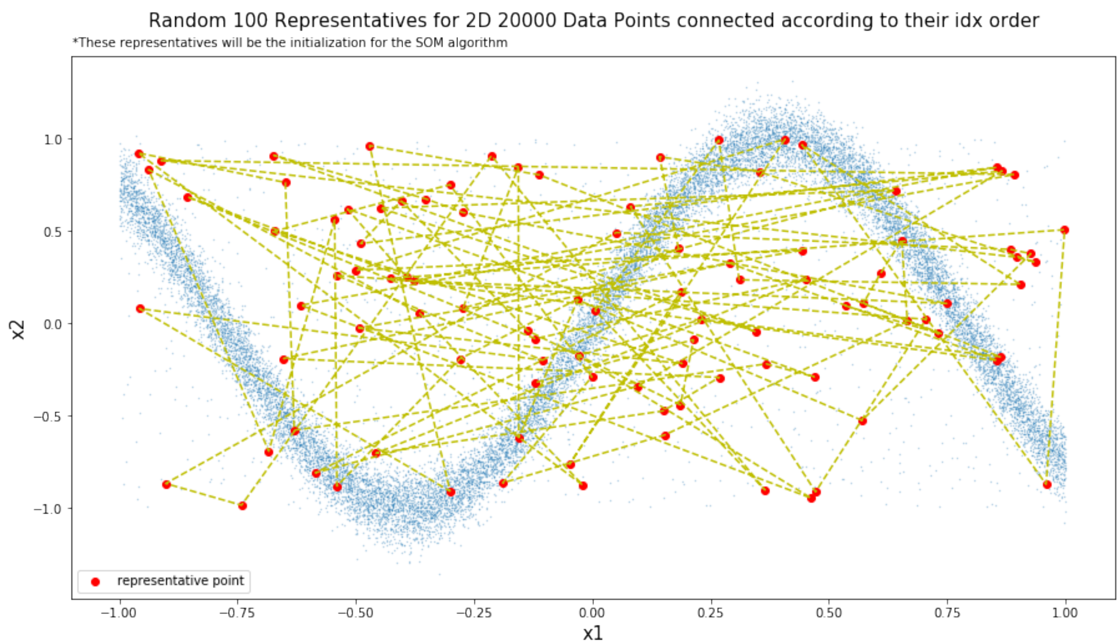
0.0.2 Q2

[0]:
```
p=0.95
f=4
sig_x=0.1
K=100
eta=1
sig=1

# setup
experiment = pd.DataFrame(np.column_stack((np.repeat(p,20000),np.
 ↪repeat(f,20000),np.repeat(sig_x,20000))),
             columns=["p","f","sig_x"])

# sampling according to the parameters
experiment[["x1","x2"]] = experiment.apply(lambda row:␣
 ↪sample_2d(row["p"],row["f"],row["sig_x"]), axis=1)
repres_mat = np.random.uniform(-1,1,size=(2,K))
```

[110]:
```
experiment[["x1","x2"]].plot(kind="scatter", x="x1", y="x2", s=0.1, alpha=0.5,␣
 ↪figsize=(15,8))
plt.scatter(x=repres_mat[0],y=repres_mat[1], c="r", label="representative␣
 ↪point")
plt.legend(loc="lower left")
plt.plot(repres_mat[0],repres_mat[1], c="y", linestyle="--")
plt.xlabel("x1" ,fontsize=15)
plt.ylabel("x2",fontsize=15)
plt.title("Random 100 Representatives for 2D 20000 Data Points connected␣
 ↪according to their idx order\n", fontsize=15)
plt.text(x=-1.1,y=1.5,s="*These representatives will be the initialization for␣
 ↪the SOM algorithm")
plt.show()
```

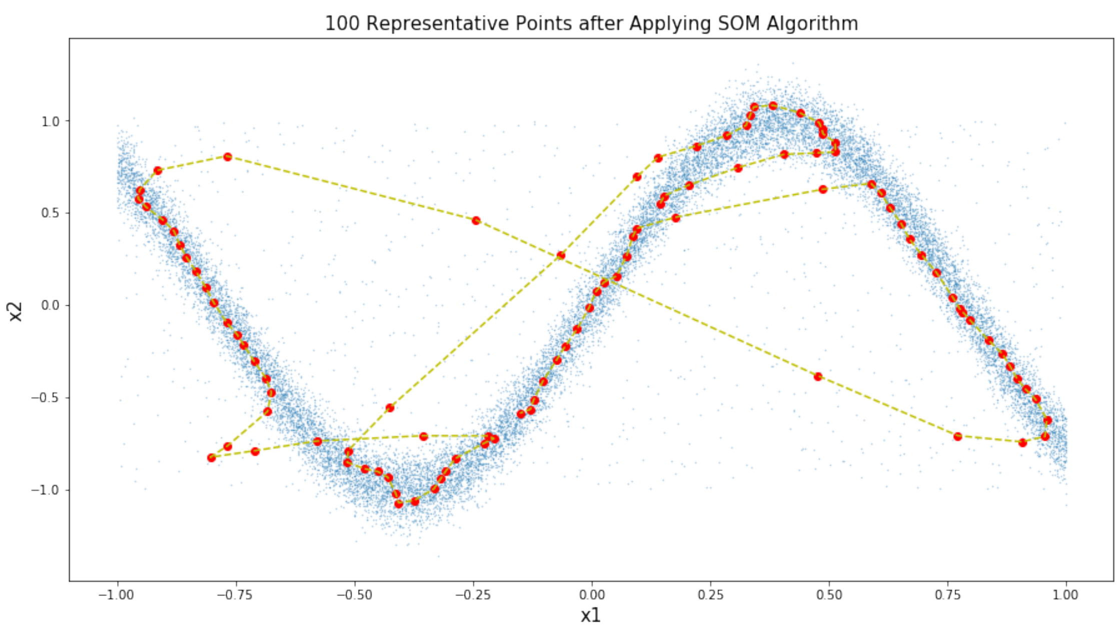


0.0.3 Q3

[0]:
```
for example in experiment[["x1","x2"]].values:
    exam_rep_diff = np.repeat(example,K).reshape(2,K) - repres_mat
    dists = np.apply_along_axis(np.linalg.norm,0, exam_rep_diff)
    l = np.argmin(dists)
    arg_for_pi = np.arange(repres_mat.shape[1]) - np.repeat(l,repeats=repres_mat.
 ↪shape[1])  # 0 < arg+l < K-1
    pis = np.exp(-(arg_for_pi)**2/(2*(sig**2)))
    pis /= np.sum(pis)
    repres_mat += eta*exam_rep_diff*pis
```

0.0.4 Q4

[112]:
```
experiment[["x1","x2"]].plot(kind="scatter", x="x1", y="x2", s=0.1, alpha=0.5,␣
 ↪figsize=(15,8))
plt.scatter(x=repres_mat[0],y=repres_mat[1], c="r", label="representative␣
 ↪point")
plt.plot(repres_mat[0],repres_mat[1], c="y", linestyle="--")
plt.title("100 Representative Points after Applying SOM Algorithm", fontsize=15)
plt.xlabel("x1" ,fontsize=15)
plt.ylabel("x2",fontsize=15)
plt.show()
```



0.0.5 Q5

Seif a
SOM algo updates representative with close indecis in a similar way, as the pi function of the updating rule forces it. Therefore the yellow line is quite going from one representative to the next one (there is noice so there are few points that violate this). Basically, the distances between representative points are similar in the most of the time. This is because SOM force the K's points to represent the data, and the data is close to be uniform in the two dimensions. The data is simply uniform in the first dimension, and in the second it is spread most of the time on the sinus curve (with a small noise). The places in the data which violated the uniformity, produce representatives that violated the equal distances.

Seif b
The magnitude of noise in the encoding doesn't affect the organization one after another of the representative, because the rule of close-idx-of-representatives update together doesn't change

neither. However, the noise do affect the ability of the algo to assign representatives which will represent the data in a satisfy way, and therefore as the noise will increase the representatives will represent the examples in a bad way.