Contents

מערכות לומדות תרגיל 6

עמית בסקין 312259013

PCA

חלק תיאורתי

1. יהא X משתנה מקרי עם התפלגות כלשהי מעל $\mathbb{R}^d$ כך שהתוחלת שלו היא $0_{\mathbb{R}^d}$ ומטריצת השונות המשותפת שלו היא $\Sigma_{d\times d}$.

צ.להוכיח: לכל $v\in\mathbb{R}^d$ עם $\|v\|_2=1$, השונות של $\langle v,X\rangle$ אינה גדולה מהשונות של שיכון ה־ PCA של X ב־ $\mathbb{R}$.

הוכחה:

לפי ההגדרה מתקיים:

$$Var\left(X\right)=\mathbb{E}\left[\left(X-\overline{X}\right)\left(X-\overline{X}\right)^t\right]$$

כלומר:

$$=\mathbb{E}\left[\left(X-\overline{0}\right)\left(X-\overline{0}\right)^t\right]=\mathbb{E}\left[XX^t\right]$$

אז:

$$Var\left\langle v,X\right\rangle=Var\left(v^tX\right)=\mathbb{E}\left[\left\langle v^tX,v^tX\right\rangle\right]=\mathbb{E}\left[v^tXX^tv\right]=$$

$$=v^t\mathbb{E}\left[XX^t\right]v=v^tVar\left(X\right)v=v^tCov\left(X\right)v=v^t\Sigma v$$

יהיו $\left(\mathrm{x}_i\right)_1^m\subseteq\mathbb{R}^d$ כך ש־ $\Sigma=\sum_1^m\mathrm{x}_i\mathrm{x}_i^t$ ויהיו $\left(u_i\right)_1^n$ הוקטורים העצמיים המובילים של Σ. נגדיר את U להיות המטריצה שהעמודות שלה הן $\left(u_i\right)_1^n$ ונתבונן בפירוק הספקטרלי של Σ, קרי: $\Sigma=U^tDU$

לפי מה שראינו בהרצאה, מתקיים:

$$\underset{W\in\mathbb{R}_{1\times d}\ ,\ U\in\mathbb{R}_{d\times1}}{\arg}\min\sum_{i=1}^m\left\|\mathrm{x}_i-UW\mathrm{x}_i\right\|^2=\left(u_1,u_1^t\right)$$

כלומר, $\left(u_1,u_1^t\right)$ הוא פיתרון לבעיית ה־ PCA עבור $\left(\mathrm{x}_i\right)_1^m$.

לפי מה שראינו בהרצאה, מתקיים:

$$trace\left(v^tDv\right) \le D_{1,1} = trace\left(u_1^tDu_1\right)$$

אבל:

$$trace\left(v^tDv\right) = trace\left(v^t\Sigma v\right) = v^t\Sigma v$$
$$trace\left(u_1^tDu_1\right) = trace\left(u_1^t\Sigma u_1\right) = u_1^t\Sigma u_1$$

ולכן:

$$v^t\Sigma v \le u_1^t\Sigma u_1$$

כאשר $Var\left(u_1X\right) = u_1^t\Sigma u_1$, קרי השונות של השיכון של X ב־ $\mathbb{R}$. מש"ל

חלק תכנותי

2. נבחן מקרה שבו יש לנו n דגימות מתוך $\mathbb{R}^k \subseteq \mathbb{R}^d$ עבור $k < d$, קרי $d-k$ השיעורים האחרונים של כל דגימה הם אפסים. נניח שהדגימות נדגמו עם רעש גאוסיאני אדיטיבי.

לפני שנטפל בדגימות הרועשות, ננסה להיפטר מהרעש ה"טהור" שקיבלנו ב־ $d-k$ הקורדינטות האחרונות של כל דגימה. נראה איך ניתן לבצע זאת באמצעות PCA:

נייצר דאטא $X_{n\times d}$ עם $n = d = 100$ כך ש־ X היא מדרגה 10, והערכים הסינגולריים שלה הם $\left(\sqrt{20-2i}\right)_0^9$.

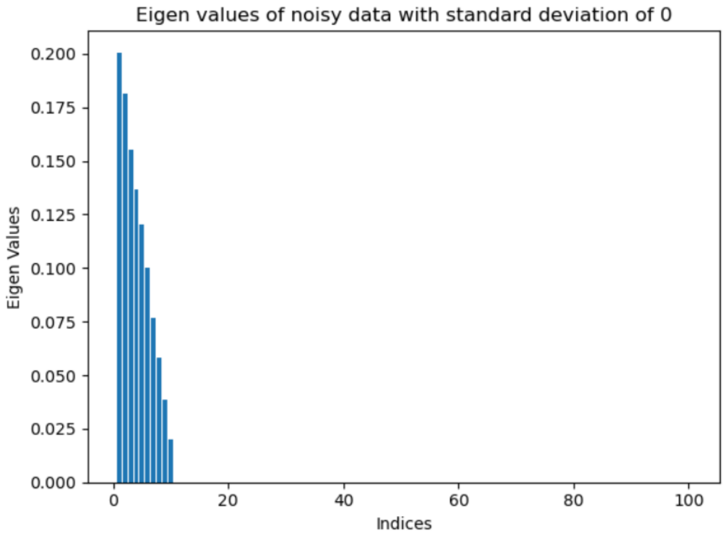
בהינתן σ, נדגום מטריצה רנדומלית $Z_{n\times d}$ כאשר כל הכניסות נדגמות באופן זהה ובלתי־תלוי מההתפלגות $\mathcal{N}\left(0,\sigma^2\right)$.

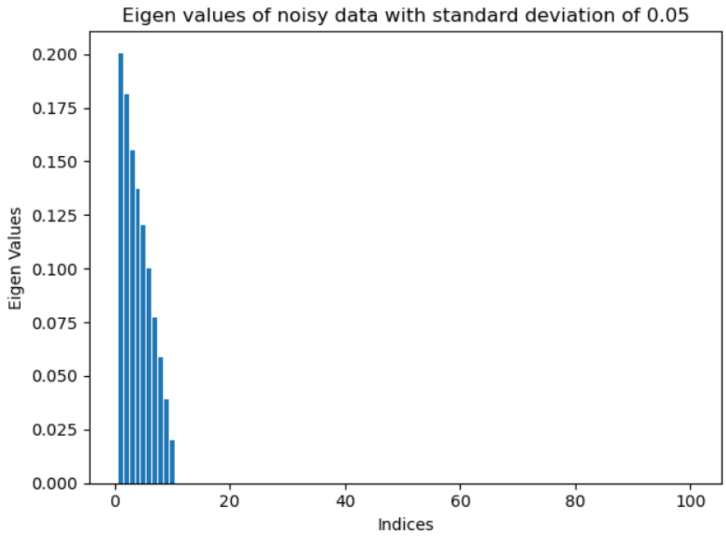
נייצר מטריצת דאטא רועשת: $Y = X + Z$.

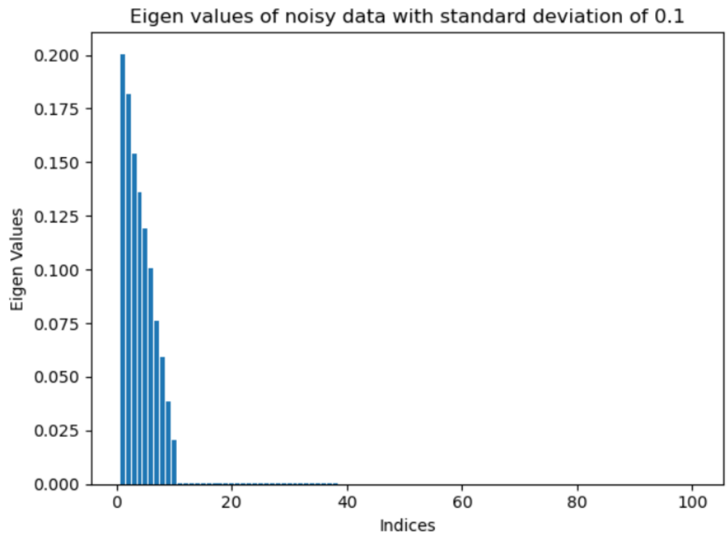
נחשב את הערכים העצמיים, $\left(\lambda_i\right)_1^{100}$ (בסדר יורד), של מטריצת השונות המשותפת של הדאטא הרועש (כל שורה היא דגימה).

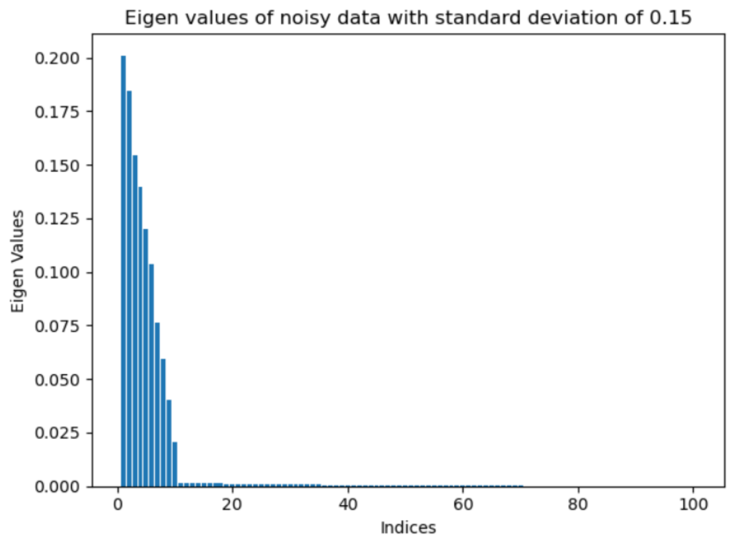
נשרטט גרף עמודות של הערכים העצמיים בסדר יורד: ציר ה־ x יהיה המספרים הסידוריים של הערכים העצמיים וציר ה־ y יהיה הערכים עצמם.

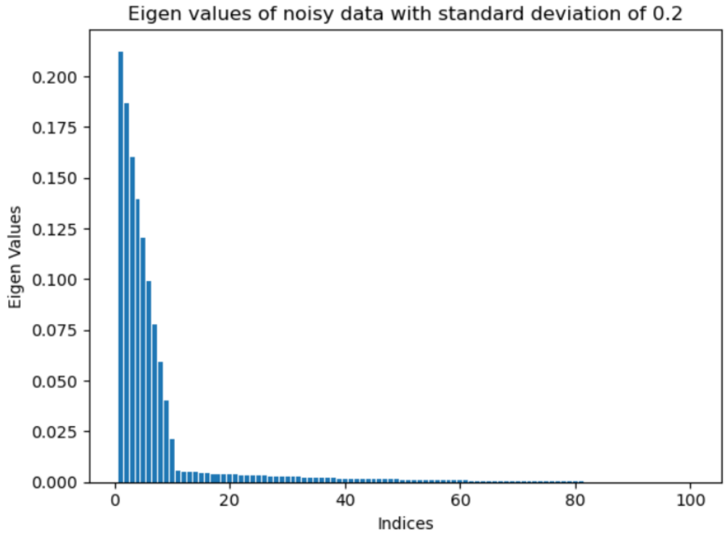
א. נבצע את הנ"ל לכל $\sigma \in \{0, 0.05, 0.1, 0.15, 0.2, 0.4, 0.6\}$:

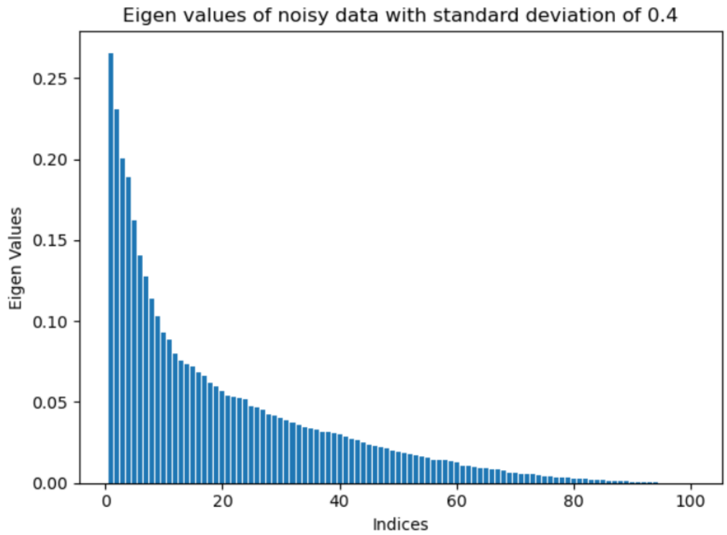


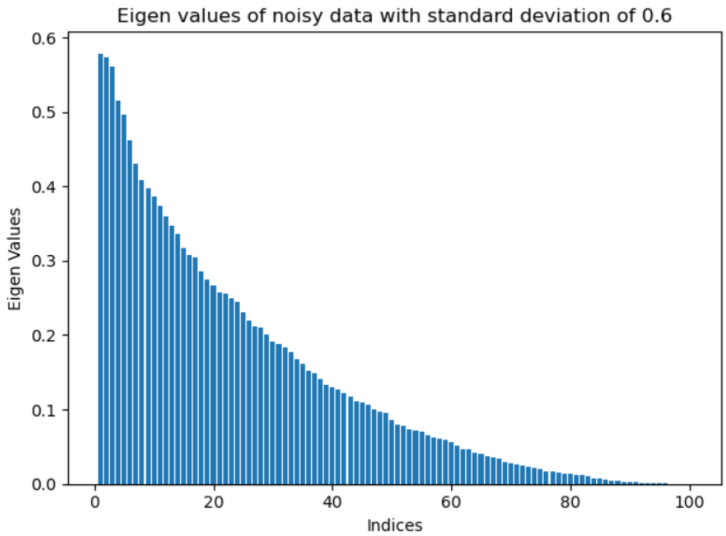

Eigen values of noisy data with standard deviation of 0.05
Eigen Values
Indices
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
20
40
60
80
100


Eigen values of noisy data with standard deviation of 0.1
Eigen Values
Indices
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
20
40
60
80
100


Eigen values of noisy data with standard deviation of 0.15
Eigen Values
Indices
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
20
40
60
80
100







ב. התופעה שעולה מהנ"ל: ככל שהרעש גדל כך הערכים העצמיים גדלים.

ג. נחקור את הסוגייה של בחירת מימד השיכון:

i. בהינתן הערכים העצמיים של מטריצת השונות המשותפת, צ.לחשוב על דרך להבדיל בין ערך עצמי שהתקבל כתוצאה

מרעש ״טהור״ ובין ערך עצמי שהתקבל מהדאטא המקורי.

צ.לתאר את השיטה וליישם אותה עבור הערכים מסעיף א:

כשסטיית התקן קטנה מ־ 0.6 נראה שניתן לקבוע באופן די ברור את הרף של הפרש בין שני ערכים עצמיים עוקבים כך שאם ההפרש קטן מהרף הזה אז מדובר בערכים עצמיים שהתקבלו כתוצאה מרעש טהור.

ניישם את השיטה על הערכים מסעיף א:

```
def determine_original_eigen_vals_amount():
    print()
    for sigma in SIGMAS:
        print('     Amount of original eigen values with '
              'standard devition of {} is '
              .format(sigma), determine_original_eigen_vals_amount_helper(
            get_eigen_vals(sigma)))


determine_original_eigen_vals_amount()
```

ונקבל:

```
Amount of original eigen values with standard devition of 0 is  10
Amount of original eigen values with standard devition of 0.05 is  10
Amount of original eigen values with standard devition of 0.1 is  12
Amount of original eigen values with standard devition of 0.15 is  12
Amount of original eigen values with standard devition of 0.2 is  11
Amount of original eigen values with standard devition of 0.4 is  11
Amount of original eigen values with standard devition of 0.6 is  1
```

כלומר, עבור סטיית תקן שקטנה מ־ 0.6 השיטה הצליחה פחות או יותר לקבוע מתי מדובר ברעש ״טהור״ אך עבור 0.6 כבר לא.

ii. צ.לחשב את המכפלה הפנימית שבין עשרת הוקטורים הסינגולריים המובילים של X ובין עשרת הוקטורים העצמיים המובילים של מטריצת השונות המשותפת של Y:

```
The inner product between the ten leading left singular vectors of X
with the ten leading eigenvectors of the covariance matrix of Y
with standard deviation of 0 is  [-0.095-0.014j  0.087-0.005j  0.015+0.11j   0.169-0.033j  0.13 -0.046j
  0.073+0.028j -0.045+0.012j  0.125+0.018j -0.005+0.045j -0.015-0.018j]

The inner product between the ten leading left singular vectors of X
with the ten leading eigenvectors of the covariance matrix of Y
with standard deviation of 0.05 is  [ 0.059 -0.045 -0.187 -0.018 -0.118  0.029 -0.277 -0.053  0.075 -0.053]

The inner product between the ten leading left singular vectors of X
with the ten leading eigenvectors of the covariance matrix of Y
with standard deviation of 0.1 is  [ 0.122  0.045  0.02   0.059 -0.02   0.002 -0.02  -0.08  -0.007  0.045]

The inner product between the ten leading left singular vectors of X
with the ten leading eigenvectors of the covariance matrix of Y
with standard deviation of 0.15 is  [-0.131  0.018 -0.138 -0.031  0.071 -0.079  0.101 -0.092  0.103 -0.185]

The inner product between the ten leading left singular vectors of X
with the ten leading eigenvectors of the covariance matrix of Y
with standard deviation of 0.2 is  [-0.008 -0.091  0.137  0.082  0.033 -0.197 -0.04  -0.079 -0.07  -0.078]

The inner product between the ten leading left singular vectors of X
with the ten leading eigenvectors of the covariance matrix of Y
with standard deviation of 0.4 is  [-0.087  0.036 -0.201  0.073 -0.058  0.042  0.035 -0.03  -0.12  -0.156]

The inner product between the ten leading left singular vectors of X
with the ten leading eigenvectors of the covariance matrix of Y
with standard deviation of 0.6 is  [ 0.04  -0.215 -0.006  0.087 -0.163  0.028 -0.126  0.184 -0.121 -0.166]
```

צ.לתאר את התוצאה שהתקבלה ולהסביר איך זה יכול לעזור לקבוע את השיטה מהסעיף הקודם:

לא ברורה לי התוצאה שהתקבלה אבל אני חושב שהמסקנה שאנו חותרים אליה היא שניתן לחשב את המכפלה הפנימית של עבור כולם, כלומר לא רק העשרה הראשונים, לפי זה נוכל לומר מתי מדובר ברעש ״טהור״.

אופטימיזציה קמורה ו־ SGD

חלק תיאורטי

3. א. יהיו $\{f_i : V \to \mathbb{R}\}_1^m$ פונקציות קמורות ו־ $\{\gamma_i \in \mathbb{R}_+\}_1^m$. נגדיר: $g : V \to \mathbb{R}$ על ידי $g(u) = \sum_1^m \gamma_i f_i(u)$. צ.להוכיח ש־ g קמורה.

הוכחה:

צ.להראות שלכל $x, y \in V$ ו־ $\alpha \in [0,1]$ מתקיים ש־

$$g(\alpha x + (1-\alpha) y) \leq \alpha g(x) + (1-\alpha) g(y)$$

קרי:

$$\sum_1^m \gamma_i f_i(\alpha x + (1-\alpha) y) \leq$$

$$\leq \alpha \sum_1^m \gamma_i f_i(x) + (1-\alpha) \sum_1^m \gamma_i f_i(y)$$

ואכן:

$$g(\alpha x + (1-\alpha) y) =$$

$$= \sum_1^m \gamma_i f_i(\alpha x + (1-\alpha) y)$$

ומכך שכל f_i קמורה:

$$\leq \sum_1^m \gamma_i (\alpha f_i(x) + (1-\alpha) f_i(y)) =$$

$$= \alpha \sum_1^m \gamma_i f_i(x) + (1-\alpha) \sum_1^m \gamma_i f_i(y) =$$

$$= \alpha g(x) + (1-\alpha) g(y)$$

מש"ל.

ב. יהיו $f, g : \mathbb{R} \to \mathbb{R}$. נגדיר $h : \mathbb{R} \to \mathbb{R}$ על ידי $h = f \circ g$. צ.להראות שאם f, g קמורות אז h אינה בהכרח קמורה.

הוכחה:

נראה דוגמה ל־ $f, g : \mathbb{R} \to \mathbb{R}$ ו־ $x, y \in \mathbb{R}$ ו־ $\alpha \in [0,1]$ כך ש־

$$f(g(\alpha x + (1-\alpha) y)) > \alpha f(g(x)) + (1-\alpha) f(g(y))$$

ניקח $g(x) = e^x$ ו־ $f(x) = -x$ ו־ $x = 2, y = 4$ ו־ $\alpha = \frac{1}{2}$. אכן f, g קמורות.

אז מצד אחד:

$$f\left(\frac{1}{2}\cdot 2+\left(1-\frac{1}{2}\right)\cdot 4\right)=$$

$$=f\left(g\left(1+2\right)\right)=f\left(g\left(3\right)\right)=f\left(e^{3}\right)=-e^{3}\approx-20$$

ומצד שני:

$$\frac{1}{2}f\left(g\left(2\right)\right)+\left(1-\frac{1}{2}\right)f\left(g\left(4\right)\right)=$$

$$=\frac{1}{2}f\left(e^{2}\right)+\frac{1}{2}f\left(e^{4}\right)=$$

$$=-\frac{1}{2}e^{2}-\frac{1}{2}e^{4}\approx-31$$

ג. תהא $f:C\to\mathbb{R}$ עם C קמורה. נגדיר את האפיגרף של f להיות:

$$E=\left\{ \left(u,t\right)\mid f\left(u\right)\leq t\right\}$$

צ.להוכיח ש־ f קמורה אםם האפיגרף של f קמורה.

הוכחה:

כיוון ראשון: נניח ש־ f קמורה ונראה ש־ E קמורה:

יהיו $\left(u,t\right),\left(v,s\right)\in E$. תהא $\alpha\in\left[0,1\right]$. צ.להראות ש־

$$\alpha\left(u,t\right)+\left(1-\alpha\right)\left(v,s\right)=$$

$$=\left(\alpha u+\left(1-\alpha\right)v,\alpha t+\left(1-\alpha\right)s\right)\in E$$

קרי ש־

$$f\left(\alpha u+\left(1-\alpha\right)v\right)\leq\alpha t+\left(1-\alpha\right)s$$

ואכן מכך ש־ f קמורה:

$$f\left(\alpha u+\left(1-\alpha\right)v\right)\leq\alpha f\left(u\right)+\left(1-\alpha\right)f\left(v\right)$$

ומכך ש־ $\left(u,t\right),\left(v,s\right)\in E$:

$$\leq\alpha t+\left(1-\alpha\right)s$$

מש"ל כיוון ראשון.

כיוון שני: נניח ש־ E קמורה ונראה ש־ f קמורה:

יהיו $x,y\in C$ ו־ $\alpha\in[0,1]$. צ.להוכיח ש־

$$f\left(\alpha x+\left(1-\alpha\right)y\right)\leq\alpha f\left(x\right)+\left(1-\alpha\right)f\left(y\right)$$

תחילה מכך ש־ C קמורה, אזי ש־ $\alpha x+\left(1-\alpha\right)y\in C$ ולכן $f\left(\alpha x+\left(1-\alpha\right)y\right)$ קיימת בכלל.

עתה בפרט מתקיים ש־ $f\left(x\right)\leq f\left(x\right)$ ו־ $f\left(y\right)\leq f\left(y\right)$ ולכן $\left(x,f\left(x\right)\right),\left(y,f\left(y\right)\right)\in E$.

אז מכך ש־ E קמורה, נקבל שגם:

$$\alpha\left(x,f\left(x\right)\right)+\left(1-\alpha\right)\left(y,f\left(y\right)\right)\in E$$

קרי:

$$\left(\alpha x+\left(1-\alpha\right)y,\alpha f\left(x\right)+\left(1-\alpha\right)f\left(y\right)\right)\in E$$

קרי:

$$f\left(\alpha x+\left(1-\alpha\right)y\right)\leq\alpha f\left(x\right)+\left(1-\alpha\right)f\left(y\right)$$

כנדרש.

ד. יהיו $\left\{f_i\colon V\to\mathbb{R}\right\}_{i\in I}$ קמורות. נגדיר: $f\colon V\to\mathbb{R}$ על ידי $f\left(u\right)=\sup_{i\in I}f_i\left(u\right)$. צ.להוכיח ש־ f קמורה.

הוכחה:

נסמן $E=epi\left(f\right)$.

הנחה סמויה: V קמורה (נובע מההגדרה של פונקציה קמורה, קרי מכך ש־ f_i קמורות).

נראה ש־ E קמורה ונקבל מהסעיף הקודם ש־ f קמורה:

$$E=\left\{\left(u,t\right)\mid f\left(u\right)\leq t\right\}=$$

$$=\left\{\left(u,t\right)\mid\sup_{i\in I}f_i\left(u\right)\leq t\right\}$$

יהיו $\left(u,t\right),\left(v,s\right)\in E$. תהא $\alpha\in[0,1]$. צ.להראות ש־

$$\alpha\left(u,t\right)+\left(1-\alpha\right)\left(v,s\right)=$$

$$=\left(\alpha u+\left(1-\alpha\right)v,\alpha t+\left(1-\alpha\right)s\right)\in E$$

קרי ש־

$$f\left(\alpha u+\left(1-\alpha\right)v\right)\leq\alpha t+\left(1-\alpha\right)s$$

$$\sup_{i\in I} f_i\left(\alpha u+(1-\alpha)v\right)\le \alpha t+(1-\alpha)s$$

אבל לכל i מתקיים ש־

$$f_i\left(\alpha u+(1-\alpha)v\right)\le \alpha t+(1-\alpha)s$$

נסמן $M=\sup_{i\in I} f_i\left(\alpha u+(1-\alpha)v\right)$.

אם I סופית אז:

$$\sup_{i\in I} f_i\left(\alpha u+(1-\alpha)v\right)=\max_{i\in I} f_i\left(\alpha u+(1-\alpha)v\right)$$

יהא k עם:

$$k\in\arg\max_{i\in I} f_i\left(\alpha u+(1-\alpha)v\right)$$

קרי:

$$\max_{i\in I} f_i\left(\alpha u+(1-\alpha)v\right)=$$

$$=f_k\left(\alpha u+(1-\alpha)v\right)$$

ומכך ש־ f_kקמורה:

$$\le \alpha t+(1-\alpha)s$$

כפי שרצינו.

אם I לא סופית:

מהגדרת הסופרמום, לכל n טבעי יש f_{i_n} עם:

$$\sup_{i\in I} f_i\left(\alpha u+(1-\alpha)v\right)<f_{i_n}\left(\alpha u+(1-\alpha)v\right)+\frac{1}{n}$$

כלומר:

$$\lim_{n\to\infty} f_{i_n}\left(\alpha u+(1-\alpha)v\right)=\sup_{i\in I} f_i\left(\alpha u+(1-\alpha)v\right)$$

אז מכך ש־ f_{i_n} קמורה לכל i_n, אזי ש־

$$f_{i_n}\left(\alpha u+(1-\alpha)v\right)-\frac{1}{n}<\alpha t+(1-\alpha)s$$

ולכן:

$$\lim_{n\to\infty}\left(f_{i_n}\left(\alpha u+(1-\alpha)v\right)-\frac{1}{n}\right)\le \alpha t+(1-\alpha)s$$

קרי:

$$\lim_{n\to\infty} f_{i_n}\left(\alpha u+(1-\alpha)v\right)\le \alpha t+(1-\alpha)s$$

קרי:

$$\sup_{i\in I} f_i\left(\alpha u+\left(1-\alpha\right)v\right)\le \alpha t+\left(1-\alpha\right)s$$

כנדרש.

4. יהיו $x\in\mathbb{R}^d$ ו־ $y\in\{\pm1\}$. נגדיר:

$$f\left(w,b\right)=l_{x,y}^{hinge}\left(w,b\right)=\max\left\{0,1-y\cdot\left(\langle w,x\rangle+b\right)\right\}$$

צ.להראות ש־ f קמורה.

הוכחה:

הפונקציה הקבועה 0 היא קמורה וכן הפונקציה $\left(w,b\right)\mapsto 1-y\cdot\left(\langle w,x\rangle+b\right)$ היא פונקציה קמורה, כלומר f היא מקסימום של פונקציות קמורות ומהסעיף הקודם מתקבל שהיא קמורה.

ב. צ.למצוא $g\in\partial l_{x,y}^{hinge}\left(w,b\right)$.

פיתרון:

נפריד למקרים:

אם $\max\left\{0,1-y\cdot\left(\langle w,x\rangle+b\right)\right\}=0$ אז לפי טענה מהתרגול מתקיים $\nabla 0\left(w,b\right)=\partial l_{x,y}^{hinge}\left(w,b\right)$ אבל $\nabla 0\left(w,b\right)=\bar{0}$ אז ניקח $g=\bar{0}$.

אם $\max\left\{0,1-y\cdot\left(\langle w,x\rangle+b\right)\right\}=1-y\cdot\left(\langle w,x\rangle+b\right)$ אז לפי טענה מהתרגול מתקיים:

נגדיר $h\left(w,b\right)=1-y\cdot\left(\langle w,x\rangle+b\right)$

$$\nabla h\left(w,b\right)-=\partial l_{x,y}^{hinge}\left(w,b\right)$$

קרי:

$$\begin{pmatrix}-yx_1\\-yx_2\\\vdots\\-yx_d\\-y\end{pmatrix}=-y\begin{pmatrix}x_1\\x_2\\\vdots\\x_d\\1\end{pmatrix}=-y\begin{pmatrix}\mathrm{x}\\1\end{pmatrix}$$

כלומר ניקח $g=-y\begin{pmatrix}\mathrm{x}\\1\end{pmatrix}$.

מש"ל

ג. יהיו $\left\{f_i\colon\mathbb{R}^d\to\mathbb{R}\right\}$ קמורות, ו־ $\left\{\xi_i\right\}_1^m$ כך ש־ $\xi_i\in\partial f_i\left(x\right)$ לכל $i\in[m]$. נגדיר $f\colon\mathbb{R}^d\to\mathbb{R}$ על ידי $f\left(x\right)=\sum_1^m f_i\left(x\right)$.

צ.להראות ש־ $\sum_1^m\xi_i\in\partial\sum_1^m f_i\left(x\right)$.

הוכחה:

מההגדרה של תת־גרדיינט, לכל i מתקיים שלכל $z \in \mathbb{R}^d$ מתקיים:

$$f_i\left(z\right) \geq f_i\left(x\right) + \left\langle \xi_i, z - x \right\rangle$$

ולכן:

$$\sum_1^m f_i\left(z\right) \geq \sum_1^m f_i\left(x\right) + \left\langle \sum_1^m \xi_i, z - x \right\rangle$$

קרי $\sum_1^m \xi_i \in \partial \sum_1^m f_i\left(x\right)$, כנדרש.

ד. יהיו $\left\{\left(x_i, y_i\right)\right\}_1^m \subseteq \mathbb{R}^d \times \left\{\pm 1\right\}$ ו־ $\lambda \geq 0$. נגדיר $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ על ידי:

$$f\left(w, b\right) = \frac{1}{m} \sum_1^m l_{x,y}^{hinge}\left(w, b\right) + \frac{1}{2} \lambda \left\Vert w \right\Vert^2$$

צ.להראות ש־ f קמורה, ולמצוא $\xi \in \partial f\left(w, b\right)$.

פיתרון:

מתקיים:

$$f\left(w, b\right) = \frac{1}{m} \sum_1^m \max\left\{0, 1 - y_i \cdot \left(\left\langle w, x_i \right\rangle + b\right)\right\} + \frac{1}{2} \lambda \left\Vert w \right\Vert^2$$

מ־ 3. ד. מתקיים ש־ $\max\left\{0, 1 - y_i \cdot \left(\left\langle w, x_i \right\rangle + b\right)\right\}$ קמורה לכל i.

־ 3. א. מתקיים ש־ $\frac{1}{m} \sum_1^m \max\left\{0, 1 - y_i \cdot \left(\left\langle w, x_i \right\rangle + b\right)\right\}$ קמורה ומכך ש־ $\frac{1}{2} \lambda \left\Vert w \right\Vert^2$ קמורה, אזי שגם $\frac{1}{m} \sum_1^m \max\left\{0, 1 - y_i \cdot \left(\left\langle w, x_i \right\rangle + b\right)\right\}+$ $\frac{1}{2} \lambda \left\Vert w \right\Vert^2$ קמורה.

נגדיר:

$$h\left(w, b\right) = \frac{1}{2} \lambda \left\Vert w \right\Vert^2$$

אז:

$$\nabla h\left(w, b\right) = \begin{pmatrix} \lambda w \\ 0 \end{pmatrix}$$

לכל i נגדיר:

$$g_i\left(w, b\right) = \begin{cases} \bar{0} & \text{if } \max\left\{0, 1 - y_i\left(\left\langle w, x_i \right\rangle + b\right)\right\} = 0 \\ -y_i \begin{pmatrix} \mathrm{x}_i \\ 1 \end{pmatrix} & \text{else} \end{cases}$$

אז משני הסעיפים הקודמים מתקבל ש־

$$\begin{pmatrix} \lambda w \\ 0 \end{pmatrix} + \frac{1}{m} \sum_1^m g_i\left(w, b\right) \in \partial f\left(w, b\right)$$

קרי:

$$\begin{pmatrix}\lambda w\\ 0\end{pmatrix}-\frac{1}{m}\sum_{y_i(\langle w,x_i\rangle+b)<1} y_i\begin{pmatrix}\mathrm{x}_i\\ 1\end{pmatrix}\in\partial f\left(w,b\right)$$

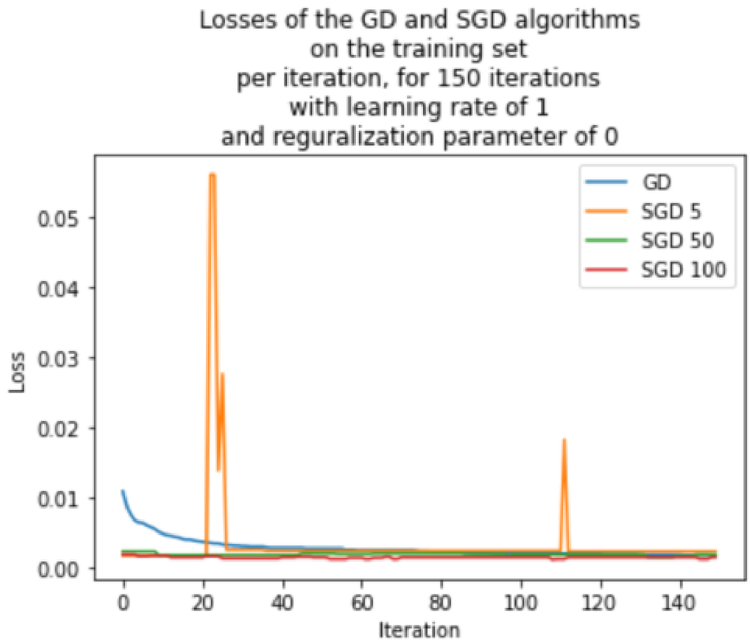
מש"ל

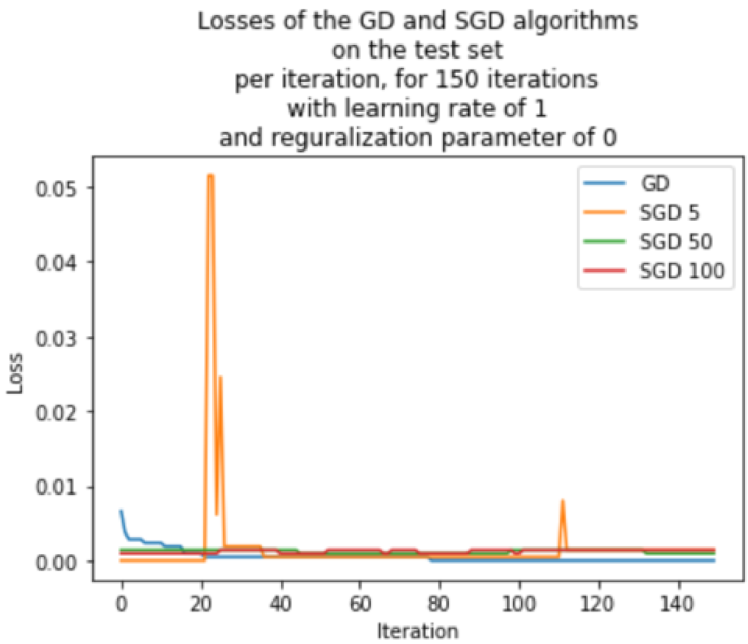
חלק תכנותי - SVM על $MNIST$

בחלק זה נפתור בעיית קלאסיפיקציה על הדאטא של $MNIST$ - תמונות של 28×28 פיקסלים של ספרות שנכתבו בכתב יד.

6. א. השתמשתי ב־ $\eta=1$.

ב. ש.לשרטט גרף של ה־ $loss$ של האימון עבור כל אחד מהאלגוריתמים בכל איטרציה, וכנ"ל עבור ה־ $loss$ של המבחן:





ג. SGD משתמש במיני באטצ'ים ומנסה לשערך את GD אבל עם פחות דגימות.

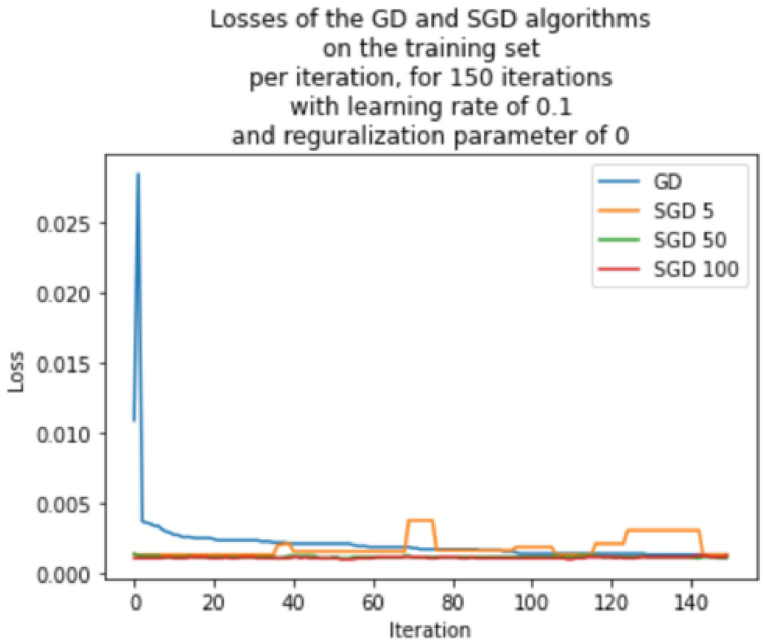
צ.לנתח את התוצאות מהסעיף הקודם. צ.להסביר כיצד השינוי בגודל הבאטצ'ים משפיע על הסיבוכיות החישובית ועל הדיוק של תת־הגרדיינט בתהליך הלמידה.

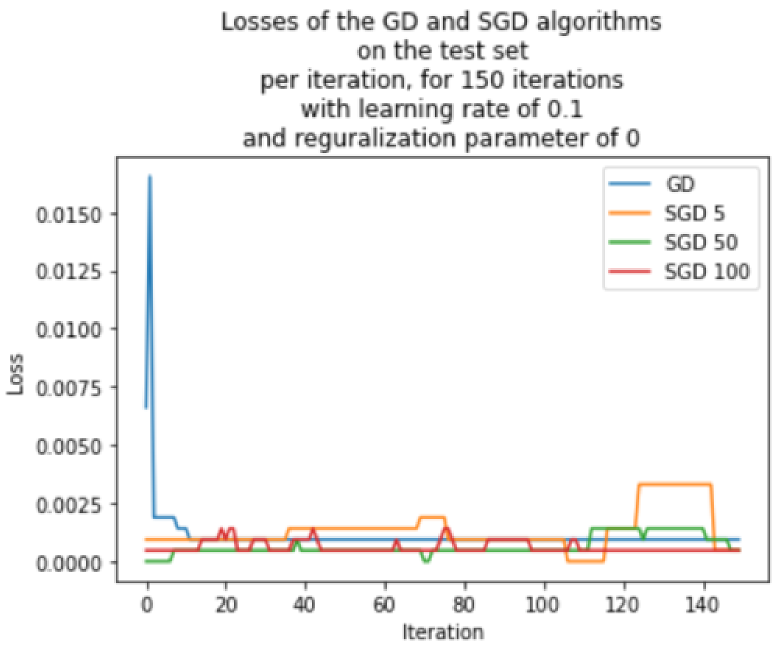
תשובה:

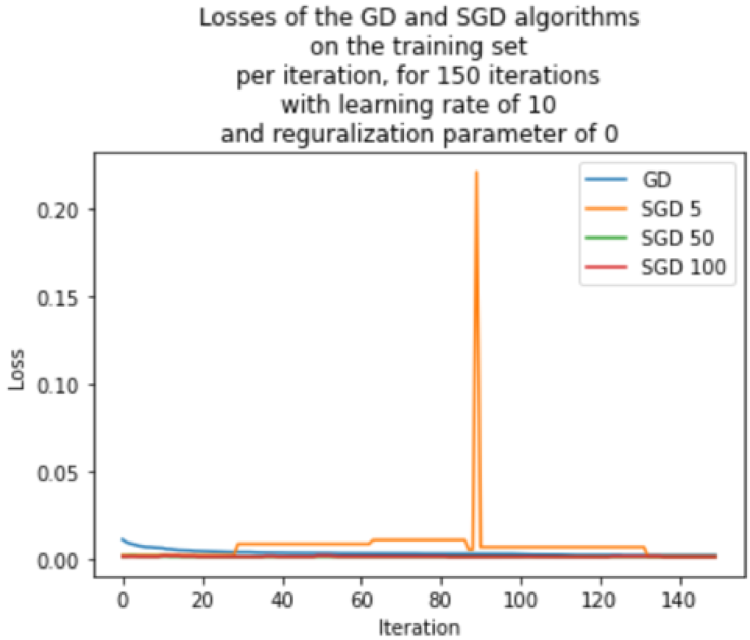
גודל הבאטצ' משפיע בעיקר בחישוב תת־הגרדיינט בכל איטרציה. כל חישוב כזה הוא לינארי בגודל הבאטצ'. נסמן ב־ T את מספר האיטרציות וב־ B גודל של באטצ' נתון. אז הסיבוכיות החישובית היא TB. לכן, עבור סקלר α כלשהו, אם לוקחים באטצ' מגודל αB, ההבדל בסיבוכיות החישובית הוא $TB\,|1-\alpha|$.

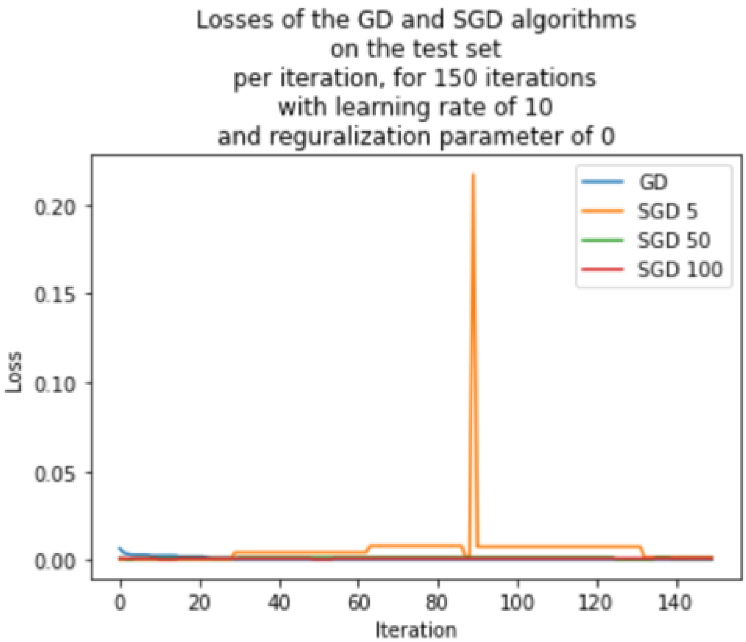
באשר לדיוק, ככל השגודל הבאטצ' גדול יותר, כך הוא יותר מייצג את הדאטא הנתון ולכן הדיוק גדל.

ד.צ.לחזור על סעיפים א ו־ ב עם $\eta = 0.1, 10$ ולבדוק מה ניתן ללמוד משיעור הלמידה במקרים הללו:









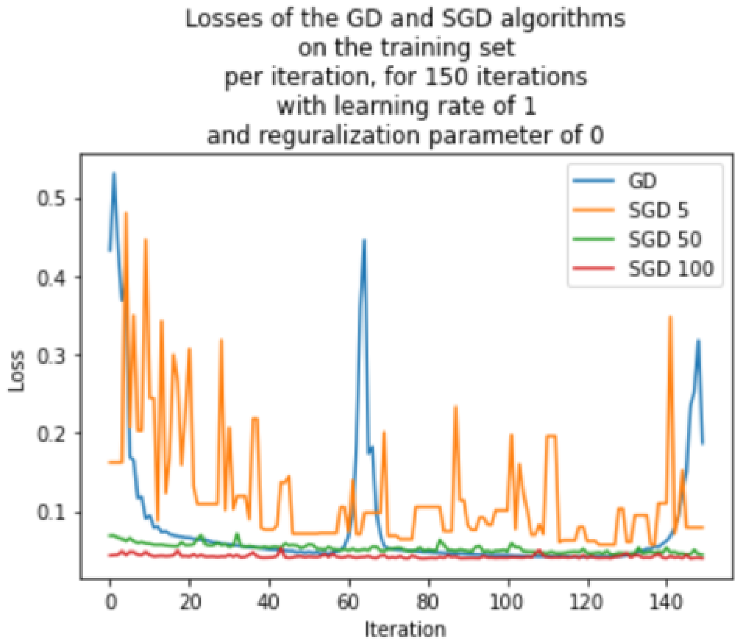
אז במקרה הזה, עבור מקדם למידה גדול יחסית, ה־ $loss$ קטן בצורה אחידה יותר על פני האיטרציות.

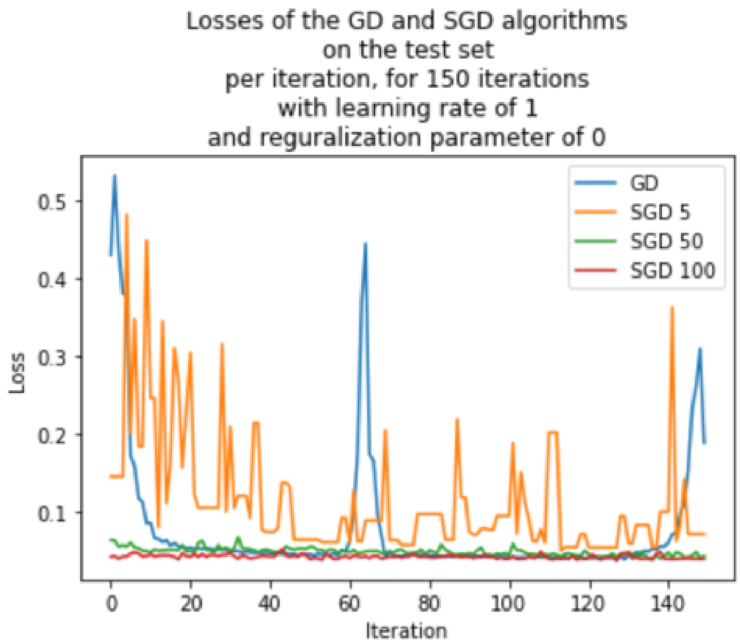
שאלה: עבור אילו מקרים כדאי לנו לקחת מקדם למידה גבוה / שיעור למידה נמוך? בשביל לענות על השאלה ניתן להשתמש בגרדיינט של $f(x) = 10^8 \cdot |x|$ ושל $g(x) = 10^{-8} \cdot |x|$ בנקודות שונות, ובהשפעה שלו ב־ GD.

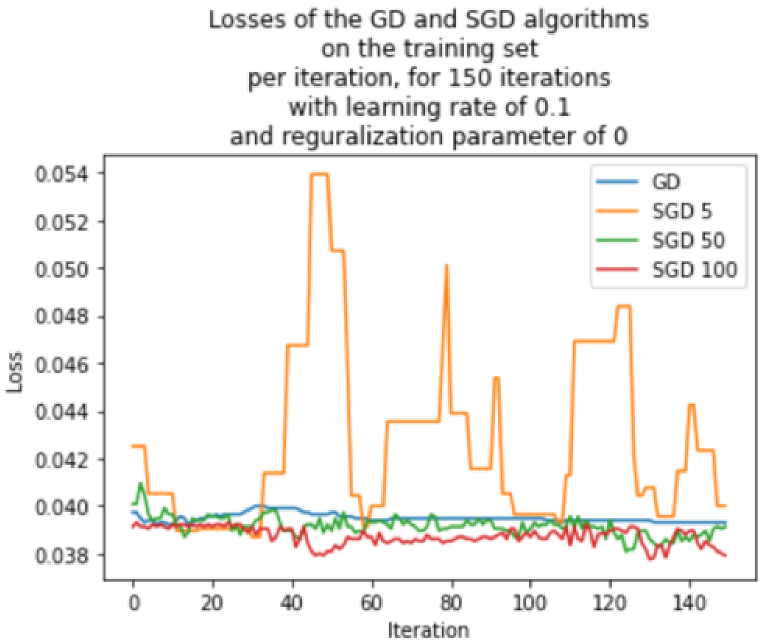
תשובה:

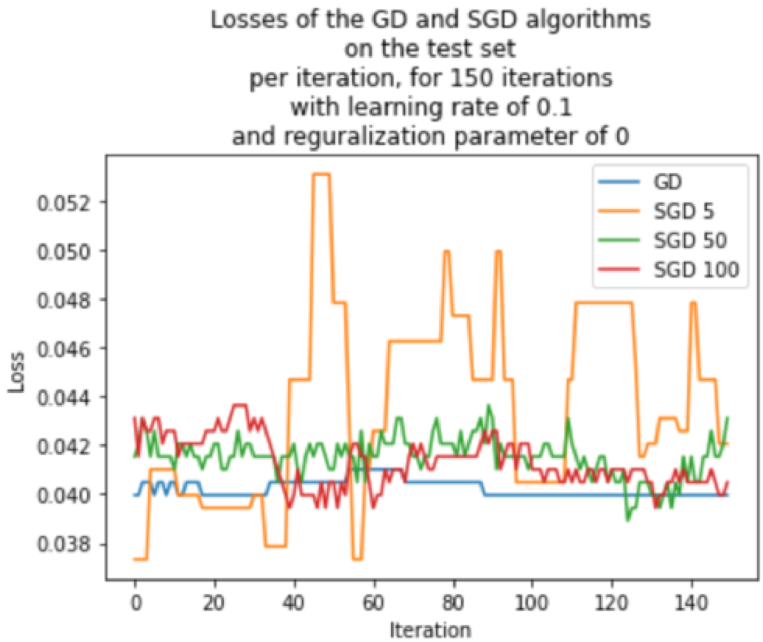
ככל שהתת־גרדיינט קטן יותר בערך מוחלט, כך כדאי לקחת מקדם למידה גבוהה יותר, על מנת ש"נספיק" למצוא את המינימום הגלובלי, שהרי הפונקציה שעבורה לקחנו תת־גרדיינט, מתקדמת לאט. ובאופן אנלוגי, ככל שהתת־גרדיינט גבוה יותר בערך מוחלט, כך כדאי לקחת מקדם למידה נמוך יותר, על מנת שלא "נפספס" את המינימום הגלובלי, שהרי הפונקציה שעבורה לקחנו תת־גרדיינט, מתקדמת מהר.

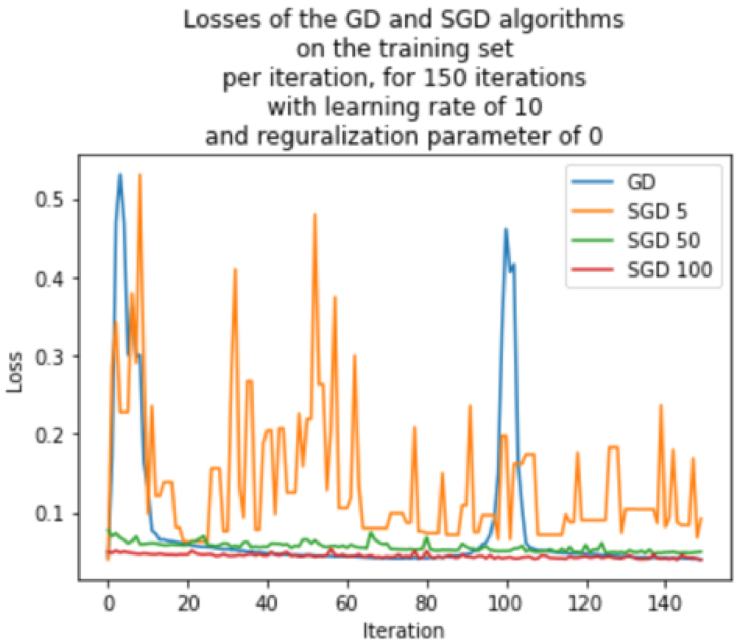
ה. צ.לבצע קלאסיפיקציה עבור זוג ספרות נוסף: למצוא זוג ספרות כך שה־ GD או ה־ SGD לא מתפקד היטב עבורו, ולהציע הסבר עבור התוצאות הגרועות:

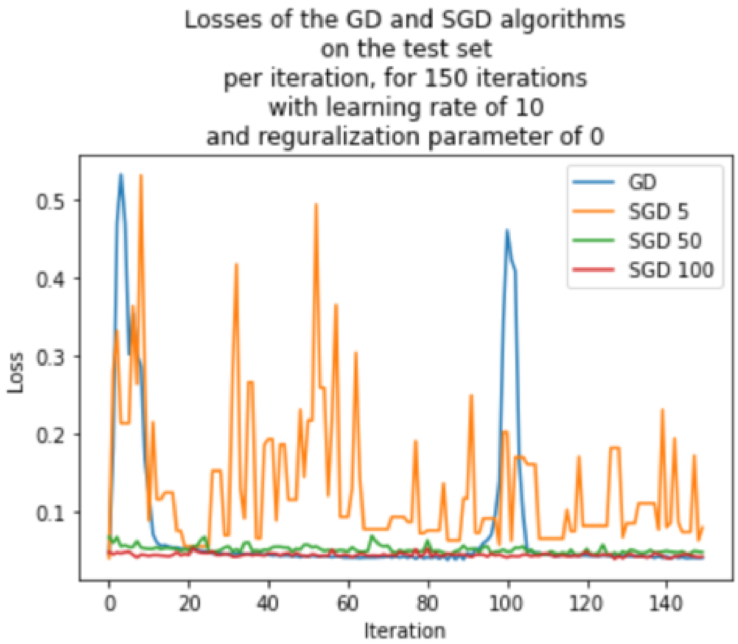

Losses of the GD and SGD algorithms
on the training set
per iteration, for 150 iterations
with learning rate of 1
and reguralization parameter of 0
GD
SGD 5
SGD 50
SGD 100
Loss
Iteration


Losses of the GD and SGD algorithms
on the test set
per iteration, for 150 iterations
with learning rate of 1
and reguralization parameter of 0
GD
SGD 5
SGD 50
SGD 100
Loss
Iteration


Losses of the GD and SGD algorithms
on the training set
per iteration, for 150 iterations
with learning rate of 0.1
and reguralization parameter of 0
GD
SGD 5
SGD 50
SGD 100
Loss
Iteration


Losses of the GD and SGD algorithms
on the test set
per iteration, for 150 iterations
with learning rate of 0.1
and reguralization parameter of 0
GD
SGD 5
SGD 50
SGD 100
Loss
Iteration





הסבר:

הדמיון בכתיב של 3 ו־ 5 גדול יבאופן משמעותי מאשר הדמיון שבין הכתיב של 0 ו־ 1 ולכן יותר קשה להם להבחין בין הספרות.

2 PCA.py

```
import numpy as np
import matplotlib.pyplot as plt


ROWS_DIM = 100
COLS_DIM = 100
SIGMAS = [0, 0.05, 0.1, 0.15, 0.2, 0.4, 0.6]


def get_samples():
    A = np.random.randn(ROWS_DIM, COLS_DIM)
    A_mean_mat = np.tile(np.mean(A, 0).reshape((1, -1)), (ROWS_DIM, 1))
    A -= A_mean_mat
    U, _, V = np.linalg.svd(A)
    D = np.sqrt(2 * np.arange(10, 0, -1))
    X = np.dot(np.dot(U[:, :10], np.diag(D)), V[:10, :])
    return X


X = get_samples()


def get_noised_samples(sigma):
    Z = np.random.normal(0, sigma ** 2, (ROWS_DIM, COLS_DIM))
    return X + Z


def get_eigenvals(sigma):
    Y = get_noised_samples(sigma)
    return np.flip(np.linalg.eigvalsh(np.cov(Y)))


def plot_eigenvals(sigma):
    eigenvals = get_eigenvals(sigma)
    indices = np.arange(len(eigenvals)) + 1
    plt.bar(indices, eigenvals)
    plt.xlabel('Indices')
    plt.ylabel('Eigenvalues')
    plt.title('Eigenvalues of noisy data with standard deviation of {}'.
              format(sigma))
    plt.show()


def plot_all_eigenvals():
    for sigma in SIGMAS:
        plot_eigenvals(sigma)


# plot_all_eigenvals()


def determine_original_eigenvals_amount_helper(eigenvals):
    counter = 1
    for i in range(len(eigenvals) - 1):
        curr = eigenvals[i]
        next = eigenvals[i + 1]
        if 0.95 * curr > next and next > 0.01 * curr:
            counter += 1
        else:
```

```python
            break
    return counter


def determine_original_eigenvals_amount():
    print()
    for sigma in SIGMAS:
        print('     Amount of original eigenvalues with '
              'standard devition of {} is '
              .format(sigma), determine_original_eigenvals_amount_helper(
            get_eigenvals(sigma)))


# determine_original_eigenvals_amount()


def get_original_and_noised_product_helper(sigma):
    Y = get_noised_samples(sigma)
    U, _, V = np.linalg.svd(X)
    X_left_singular_vecs = U[:10]
    Y_cov_eigenvecs = np.linalg.eig(np.cov(Y))[1][:10]
    inner_product = []
    for i in range(10):
        sing = X_left_singular_vecs[i]
        eig = Y_cov_eigenvecs[i]
        inner_product.append(sing @ eig)
    return inner_product


def get_original_and_noised_product():
    print()
    for sigma in SIGMAS:
        print()
        print('The inner product between the ten leading left singular '
              'vectors of X\nwith the ten leading eigenvectors of the '
              'covariance matrix of Y\nwith standard deviation of {} is '
              .format(sigma), np.round(get_original_and_noised_product_helper(
            sigma), 3))


# get_original_and_noised_product()


def run_all():
    plot_all_eigenvals()
    determine_original_eigenvals_amount()
    get_original_and_noised_product()


# run_all()
```

3 SVM.py

```
import numpy as np
from matplotlib import pyplot as plt


BATCHES = [5, 50, 100]
ETAS = [1, 0.1, 10]
REG_PARAM = 0
ITERS = 150


image_size = 28
no_of_different_labels = 2
image_pixels = image_size * image_size
data_path = r'dataset\mldata'
train_data = np.loadtxt(data_path + r"\mnist_train.csv",
                        delimiter=",")
test_data = np.loadtxt(data_path + r"\mnist_test.csv",
                       delimiter=",")

fac = 0.99 / 255
all_train_imgs = np.asfarray(train_data[:, 1:]) * fac + 0.01
all_test_imgs = np.asfarray(test_data[:, 1:]) * fac + 0.01

all_train_labels = np.asfarray(train_data[:, :1])
all_test_labels = np.asfarray(test_data[:, :1])


class DigitsData:
    """
    Training and test sets for two digits to classify
    """
    def __init__(self, d_1, d_2):
        binary_train_indices = np.logical_or((all_train_labels == d_1),
                                             (all_train_labels == d_2))
        binary_test_indices = np.logical_or((all_test_labels == d_1),
                                            (all_test_labels == d_2))

        binary_train_indices = binary_train_indices.reshape(
        binary_train_indices.shape[0])
        binary_test_indices = binary_test_indices.reshape(
            binary_test_indices.shape[0])

        x_train, y_train = all_train_imgs[binary_train_indices], \
                           all_train_labels[binary_train_indices]
        x_test, y_test = all_test_imgs[binary_test_indices], \
                         all_test_labels[binary_test_indices]

        y_train = y_train.reshape((len(y_train),))
        y_test = y_test.reshape((len(y_test),))
        y_train[y_train == d_1] = -1
        y_test[y_test == d_1] = -1
        y_train[y_train == d_2] = 1
        y_test[y_test == d_2] = 1
        self.x_train, self.y_train, self.x_test, self.y_test = \
            x_train, y_train, x_test, y_test


def get_subgradient(w, reg_param, data, labels):
    """
```

```
    Gets a subgradient according the hinge loss
    :param w: A d+1 by 1 numpy array where d is the dimension of each sample
    i.e. the parameters of the classifier where the last element is the bias
    :param reg_param: The regularization parameter in the sub-gradient
    :param data: The samples to calculate the subgradient over
    :param labels: The labels to calculate the subgradient over
    :return: A subgradient according the hinge loss
    """
    sample = data[0]
    sample_length = len(sample)
    m = len(data)
    only_w = w[: len(w) - 1]
    b = w[-1]
    sm = np.zeros(sample_length+1)
    for i in range(m):
        y_i = labels[i]
        x_i = data[i]
        extended_x_i = np.append(data[i], 1)
        curr = y_i * extended_x_i
        if y_i * (only_w @ x_i + b) < 1:
            sm += curr
    extended_w = np.append(only_w, 0)
    return reg_param * extended_w - 1/m * sm



def gd(data, labels, iters, eta, w, reg_param):
    """
    The sub gradient descent algorithm
    :param data: An n by d numpy array, where n is the amount of samples and
    d is the dimension of each sample
    :param labels: An n by 1 numpy array with the labels of each sample
    :param iters: An integer that will define the amount of iterations
    :param eta: A positive number that will define the learning rate
    :param w: A d+1 by 1 numpy array where d is the dimension of each sample
    i.e. the parameters of the classifier where the last element is the bias
    :param reg_param: The regularization parameter in the sub-gradient
    :return: A d+1 by 1 numpy array which contains the output of the sub
    gradient descent algorithm over "iters" iterations and a list of the
    vector result in each iteration
    """
    vecs = []
    for iter in range(iters):
        w += -eta * get_subgradient(w, reg_param, data, labels)
        vecs.append(w.copy())
    return 1/iters * np.sum(vecs, axis=0), vecs


def sgd(data, labels, iters, eta, w, batch):
    """
    The stochastic gradient descent algorithm
    :param data: An n by d numpy array, where n is the amount of samples and
    d is the dimension of each sample
    :param labels: An n by 1 numpy array with the labels of each sample
    :param iters: An integer that will define the amount of iterations
    :param eta: A positive number that will define the learning rate
    :param w: A d+1 by 1 numpy array where d is the dimension of each sample
    i.e. the parameters of the classifier where the last element is the bias
    :param batch: The amount of samples that the algorithm would draw and
    use at each iteration
    :return: A d+1 by 1 numpy array which contains the output of the
    sub-gradient descent algorithm over "iters" iterations and a list of the
    vector result in each iteration
    """
    vecs = []
    for iter in range(iters):
        samples_indices = np.random.choice(range(len(data)), batch)
        samples = np.take(data, samples_indices, axis=0)
```

```
        samples_labels = np.take(labels, samples_indices, axis=0)
        v_t = get_subgradient(w, 0, samples, samples_labels)
        w -= eta * v_t
        vecs.append(w.copy())
    return 1/iters * np.sum(vecs, axis=0), vecs


def get_errors(vecs, data, labels):
    """
    Gets a list of errors for each vector in vecs over the data and labels
    :param vecs: The vecs to calculate their errors
    :param data: The samples to calculate the errors over
    :param labels: The labels to calculate the errors over
    :return: A list of errors for each vector in vecs over the data and labels
    """
    errors = []
    for vec in vecs:
        errors.append(test_error(vec, data, labels))
    return errors


def get_hypothesis(w):
    """
    Gets the classifying hypothesis of the given vector
    :param w: The vector to get its classifying hypothesis
    :return: The classifying hypothesis of the given vector
    """
    def h_w(x):
        return np.sign(w[:-1] @ x + w[-1])
    return h_w


def test_error(w, test_data, test_labels):
    """
    Returns a scalar with the respective 0-1 loss for the hypothesis
    :param w: a d+1 by 1 numpy array where h_w(x) = sign(<w[:-1], x> + w[-1])
    :param test_data: An n by d numpy array with n samples
    :param test_labels: An n by 1 numpy array with the labels of the samples
    :return: A scalar with the respective 0-1 loss for the hypothesis
    """
    h_w = get_hypothesis(w)
    test_predictions = []
    for sample in test_data:
        test_predictions.append(h_w(sample))
    result = test_labels - test_predictions
    samples_amount = len(result)
    non_zeros_amount = np.count_nonzero(result)
    return non_zeros_amount / samples_amount


def plot_errors_helper(d_1, d_2, iters, gd_errors, sgd_errors_lst, set_str, eta,
                       reg_param):
    iters_lst = range(iters)
    plt.plot(iters_lst, gd_errors, label='GD')
    for i in range(len(BATCHES)):
        plt.plot(iters_lst, sgd_errors_lst[i], label='SGD {}'.format
        (BATCHES[i]))
    plt.legend()
    plt.xlabel('Iteration')
    plt.ylabel('Loss')
    plt.title('Losses of the GD and SGD algorithms\nclassifying the digits '
              '{} and {} on the {} set\nper iteration, for {} '
              'iterations\nwith learning rate of {}\nand reguralization '
            'parameter of {}'.format(d_1, d_2, set_str, iters, eta,
                                        reg_param))
    plt.show()
```

```python
def plot_errors(d_1, d_2, train_samples, train_labels,
                test_samples, test_labels, iters, eta, w, reg_param):
    w_gd, vecs_gd = gd(train_samples, train_labels, iters, eta, w, reg_param)
    gd_train_errors = get_errors(vecs_gd, train_samples, train_labels)
    gd_test_errors = get_errors(vecs_gd, test_samples, test_labels)
    sgd_train_errors_lst = []
    sgd_test_errors_lst = []
    for batch in BATCHES:
        w_sgd, vecs_sgd = sgd(train_samples, train_labels, iters, eta, w, batch)
        sgd_train_errors_lst.append(get_errors(vecs_sgd, train_samples,
                                               train_labels))
        sgd_test_errors_lst.append(get_errors(vecs_sgd, test_samples,
                                              test_labels))
    plot_errors_helper(d_1, d_2, iters, gd_train_errors, sgd_train_errors_lst,
                       'training', eta, reg_param)
    plot_errors_helper(d_1, d_2, iters, gd_test_errors, sgd_test_errors_lst,
                       'test', eta, reg_param)


def classify_digits(d_1 ,d_2):
    """
    Run all items for the given digits
    :param d_1: First digit
    :param d_2: Second digit
    :return: None
    """
    sets = DigitsData(d_1, d_2)
    sample = sets.x_train[0]
    w_length = len(sample) + 1
    zero_w = np.zeros(w_length)
    for eta in ETAS:
        plot_errors(d_1, d_2, sets.x_train, sets.y_train,
                    sets.x_test, sets.y_test, ITERS, eta, zero_w, REG_PARAM)
```

4 ex6 runme.py

```
from SVM import classify_digits


def main():
    classify_digits(0, 1)
    classify_digits(3, 5)


if __name__ == "__main__":
    main()
```