אופטימיזציה ללמידת מכונה

הסבר לגבי האלגוריתם

שמות מגישים:

עוביידה חטיב, 201278066

אסאל חדאד, 207718701

הרצת האלגוריתם

הרצת הקוד מתבצעת ע"י אחת מבין שתי הפקודות הבאות:

```
python good_subspace.py --n n_val --d d_val --k k_val --epsilon eps_val
python good_subspace.py --points npy_file --k k_val --epsilon eps_val
```

כאשר:

- n: מספר הנקודות.
- d: ממד המרחב הכללי.
- k: ממד המרחב של ה- Flat.
- epsilon: רמת הקירוב\דיוק.
- points: קובץ txt. המכיל קואורדינטות של נקודות.

ההבדל בין שתי הפקודות הוא שבשנייה הנקודות נתונות כארגומנט, ובכך ערכיהם של n,d מאותחלות כמספר הנקודות ומימד הנקודות בהתאמה (מצורף קובץ txt. המכיל נקודות כדוגמה). בפקודה הראשונה, מנגד, הנקודות לא נתונות והאלגוריתם מייצר אותם באופן רנדומלי, כך שיהיו n במספרם ובעלות ממד d. יצירת הנקודות נעשית ע"י בחירת תת מרחב רנדומלי מממד k, בחירת נקודות על המרחב הזה והוספת רעש כלשהו להם.

הפלט של האלגוריתם

הפלט כולל את:

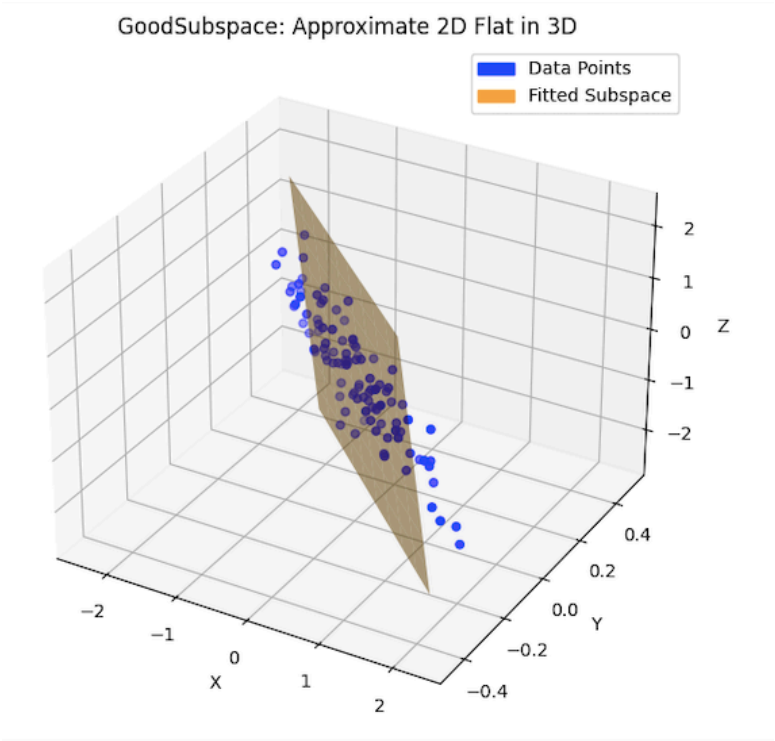
- הנוסחה הפרמטרית של תת המרחב (Flat) שנמצא.
- ה- RD cost, העלות האופטימלית לפי ה- PCA, והודעה של האם האלגוריתם עמד בדרישת הקירוב (העלות האופטימלית * (1+ε)).
- אם המרחב הוא דו-ממדי או תלת-ממדי, תוצג גם ויזואליזציה של ההתאמה ובה מוצגים הנקודות וה- Flat שנוצר.

דוגמת הרצה 1

python good_subspace.py --n 100 --d 3 --k 2 --epsilon 0.5

```
The Fitted Subspace:
  x(α) = [-0.0424, 0.0218, -0.1217] + α1*[-0.2153, 0.0370, 0.9758]
                                    + α2*[0.9555, -0.1982, 0.2184]

Optimal Cost (PCA): 0.482672
Allowed Bound: 2.2500 × 0.482672 = 1.086011
RD Cost: 0.895169
     ✅ The Algorithm satisfies the guarantee.
```

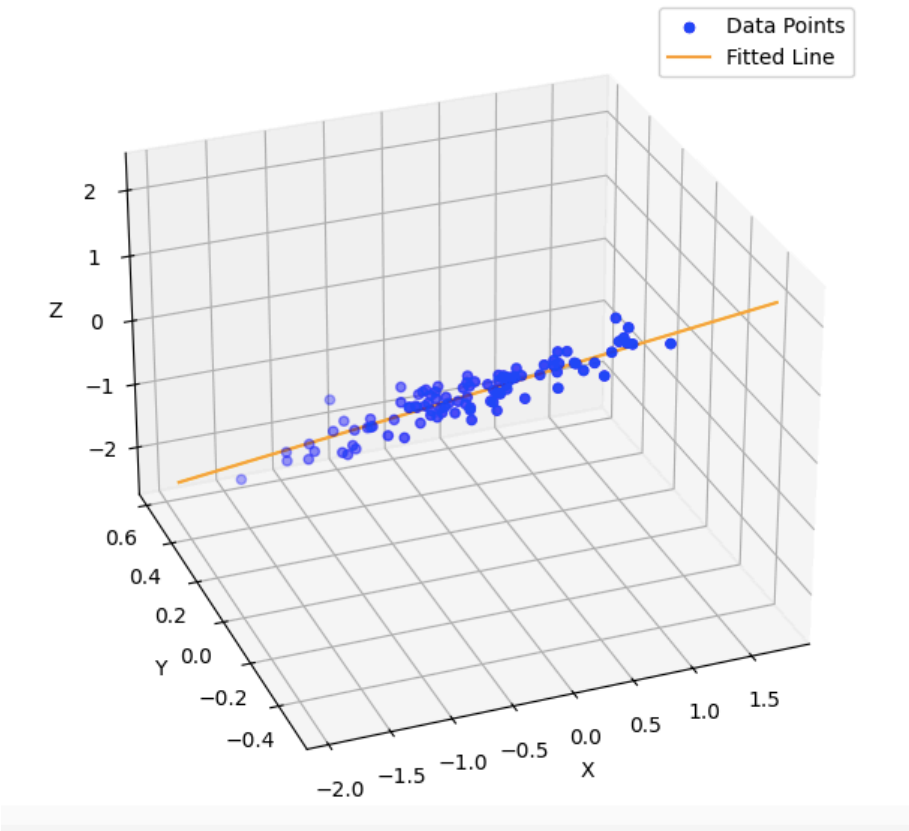


דוגמת הרצה 2

python good_subspace.py --n 100 --d 3 --k 1 --epsilon 0.1

```
The Fitted Subspace:
  x(α) = [-0.0891, 0.0322, -0.0983] + α1*[-0.6062, 0.1827, -0.7740]

Optimal Cost (PCA): 0.668118
Allowed Bound: 1.1000 × 0.668118 = 0.734929
RD Cost: 0.679274
     ✅ The Algorithm satisfies the guarantee.
```

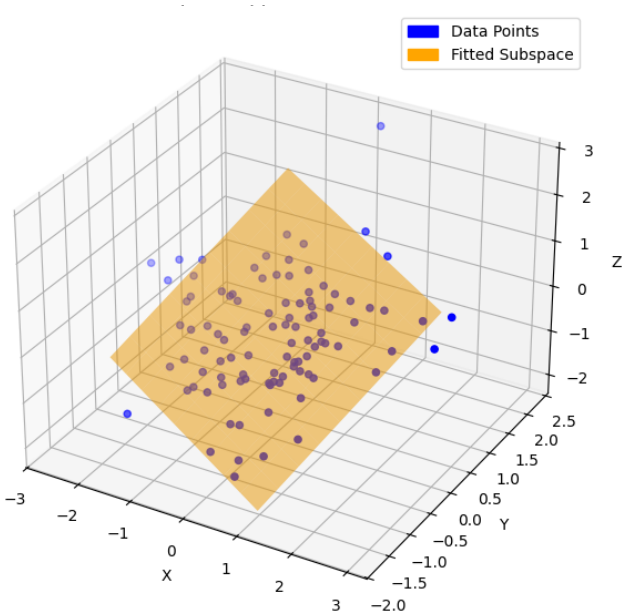


דוגמת הרצה 3

python good_subspace.py --points points_3d.txt --k 2 --epsilon 0.5

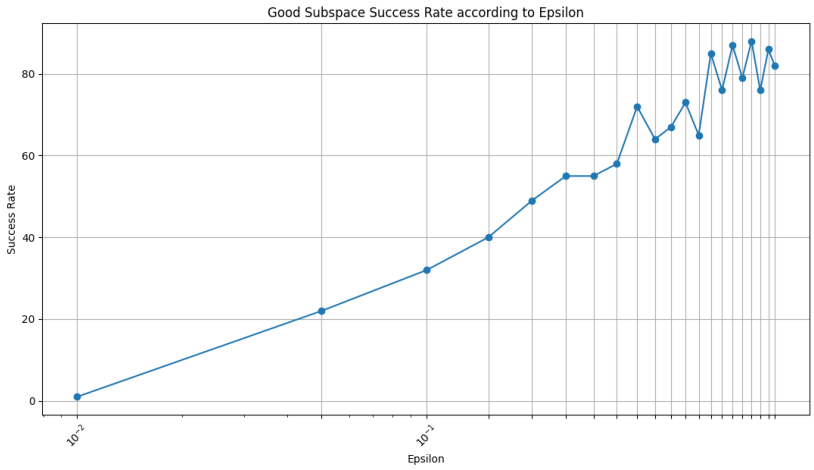
```
The Fitted Subspace:
  x(α) = [0.0788, -0.0491, -0.0631] + α1*[0.9219, -0.2620, -0.2854]
                                    + α2*[0.3867, 0.5779, 0.7186]

Optimal Cost (PCA): 0.515116
Allowed Bound: 2.2500 × 0.515116 = 1.159011
RD Cost: 0.527254
     ✅ The Algorithm satisfies the guarantee.
```

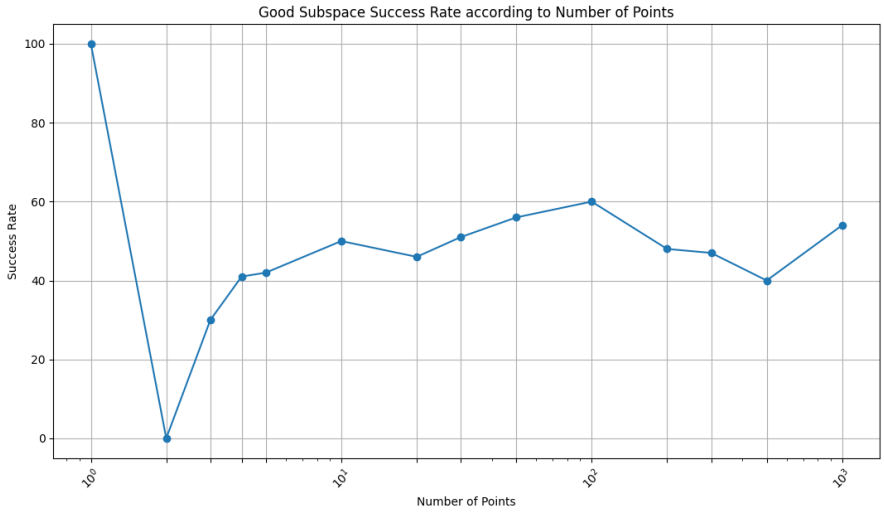


השוואה בין ערכי פרמטרים שונים:

- **מידת השגיאה** (ε): האלגוריתם נבחן תחת ערכי אפסילון שונים שבין הטווח 0-1, ע"י השוואה בין סטים של 100 טסטים שהשתמשו באותה קונפיגורציה מלבד ערך האפסילון. התוצאות, המצורפות למטה, הראו קשר ישיר וחזק בין ערך השגיאה ומידת הצלחת האלגוריתם.

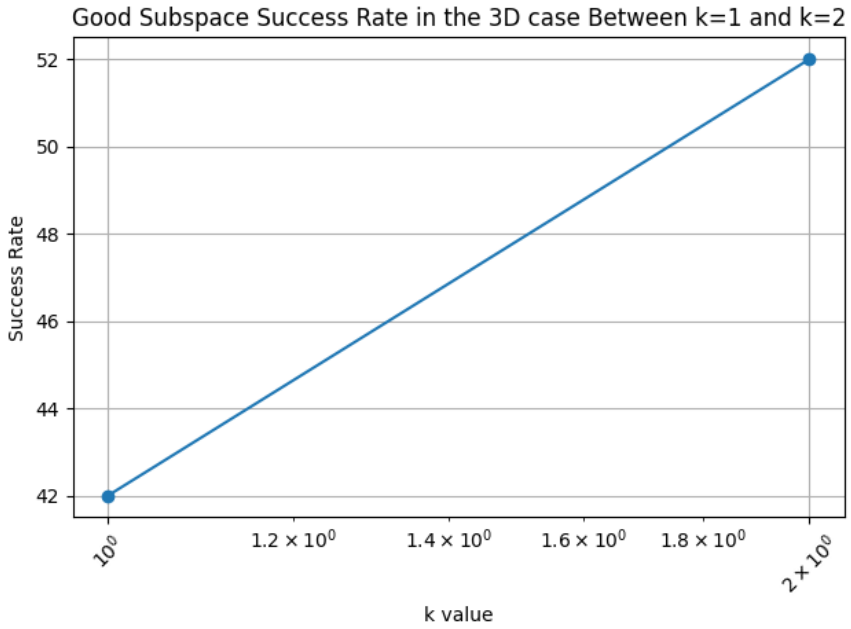


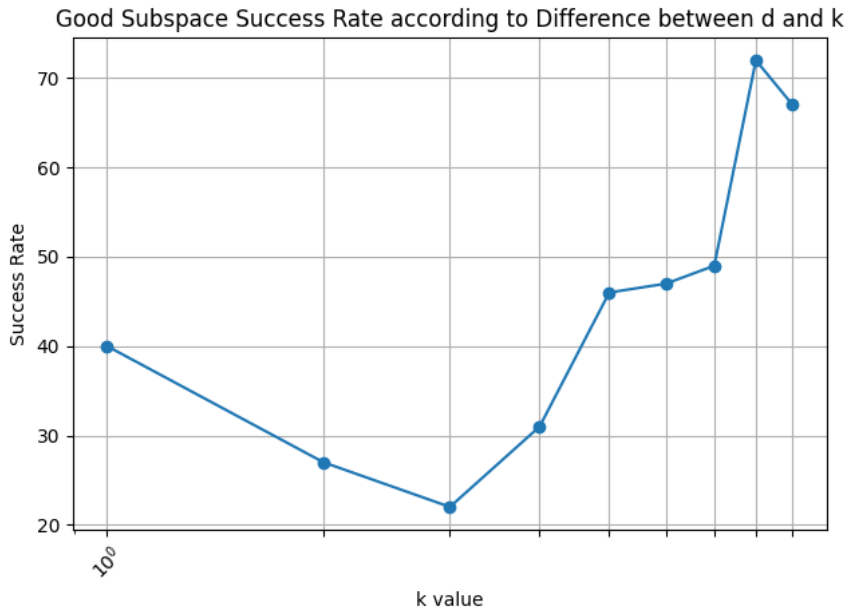
- **מספר הנקודות:** בדומה להשוואה הקודמת, השוואה נוספת בוצעה עבור מספר הנקודות שהאלגוריתם קיבל. גורם זה נחשד כבעל השפעה על הצלחת האלגוריתם בשל כך שיותר נקודות משמעה יותר מידע על המרחב. למרות שאחוז ההצלחה עלה בהתאם למספר הנקודות למספרים קטנים בטווח 1-10, מגמה זאת לא נצפתה במספרי נקודות גדול.



- **הפרש בין d ל-k:** הנחנו שזהו גורם משפיע גם כן על סמך כך שהאלגוריתם היה מצליח יותר עבור מרחב תלת-מימדי (d=3) במקרה שהתת-מרחב ממימד 2 (k=2) לעומת כאשר התת מרחב ממימד 1 (k=1), כפי שרואים בגרף הראשון למטה

המשווה בין שני המקרים. דבר זה נמדד באמצעות כך שהשווינו בין ערכי ה- k-ים השונים שבין 1-9 כאשר המרחב הוא ממימד 10, התוצאות תאמו חלקית לציפיות שלנו, וכנראה שיש גורמים אחרים המעורבים.





ההשוואה עם אלגוריתם SVD-KSM

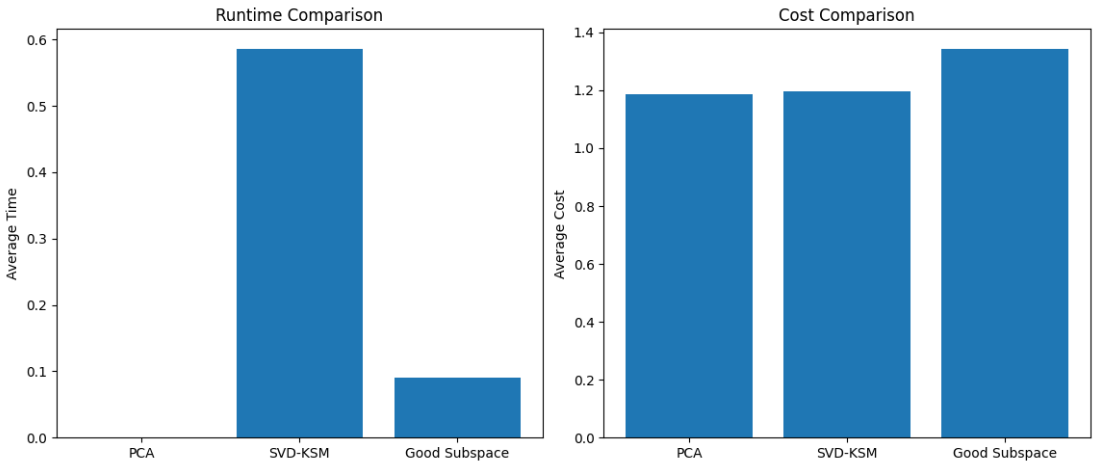
השווינו את התוצאות עם מימוש של אלגוריתם SVD-KSM הפותר את אותה בעיה, בנוסף ל-PCA. ההשוואה נעשתה במונחים של התוצאות שהתקבלו (ה- RD Score במקרה של

האלגוריתם שלנו, והמרחק האורתוגונלי במקרה של אלגוריתם SVD-KSM), וזמן הריצה.

ההשוואה לקחה בחשבון את המקרים הבאים:

```
test_cases = [
    {'n': 50, 'd': 2, 'k': 1},
    {'n': 50, 'd': 3, 'k': 2},
    {'n': 50, 'd': 3, 'k': 1},
    {'n': 200, 'd': 2, 'k': 1},
    {'n': 200, 'd': 3, 'k': 2},
    {'n': 200, 'd': 3, 'k': 1}]
```

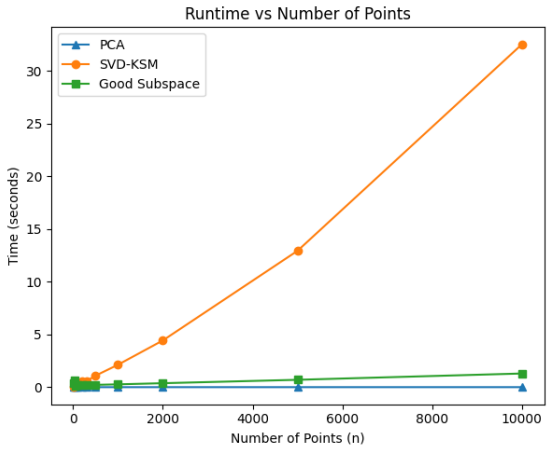
ולהלן התוצאות שהתקבלו:

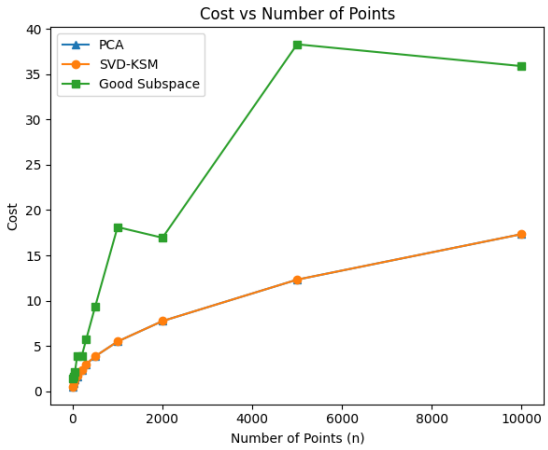


כמו כן, נעשו השוואות יותר מעמיקות שלקחו בחשבון את המשתנים המשותפים בין שני האלגוריתמים, שהם n,d,k. בשל אופיו הרנדומלי של האלגוריתם ועל מנת לצמצם את אפקט הרעש, כל מקרה מהמקרים המצוינים למטה נבחן ע"י 5 ריצות שהתמצעו.

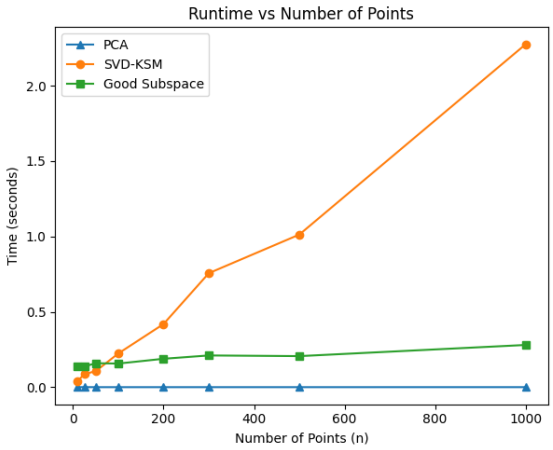
<u>ההשוואה כפונקציה של מספר הנקודות</u> (n): ערכי ה- d,k קובעו להיות 3 ו- 2 בהתאמה.

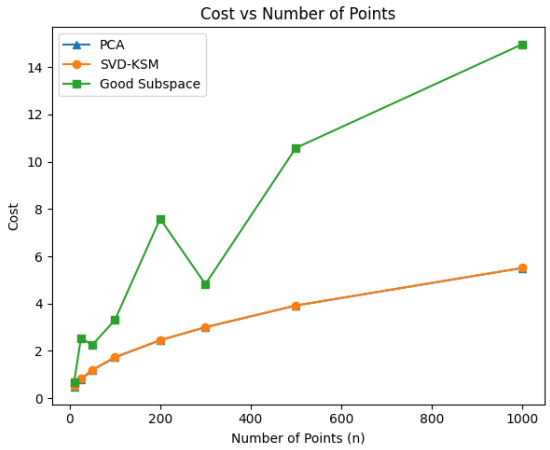
להלן הגרף המציג את התוצאות של שני האלגוריתמים על פני ערכי n שונים:



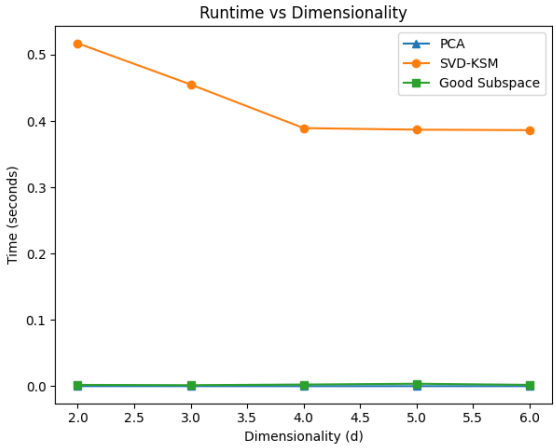


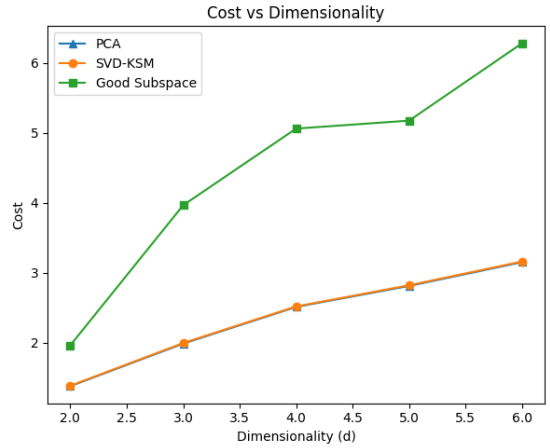
גרף היותר ממוקד בערכים עד 1000:





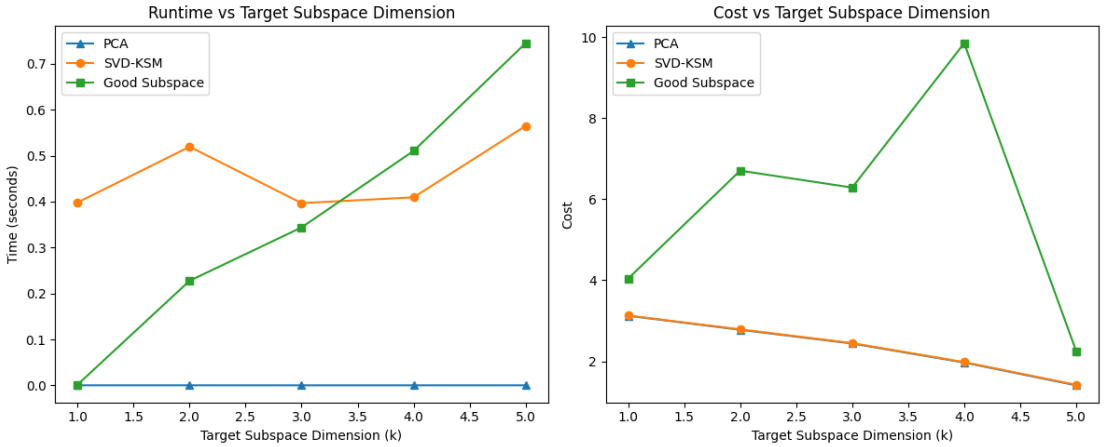
<u>ההשוואה כפונקציה של הממד</u> (d): ערכי ה- n,k קובעו להיות 200 ו- 1 בהתאמה. להלן הגרף המציג את התוצאות של שני האלגוריתמים על פני ערכי ה- d שבין 2 ל- 6:





<u>ההשוואה כפונקציה של ממד המטרה</u> (k): ערכי ה- n,d קובעו להיות 200 ו- 6 בהתאמה.

להלן הגרף המציג את התוצאות של שני האלגוריתמים על פני ערכי ה- k שבין 1 ל- 5:



אלגוריתם ה- Good Subspace מגלה יעילות מבחינת זמן ריצה, באופן יותר טוב לעומת אלגוריתם SVD-KSM, ולא נראה שהוא מושפע רבות מהשתנות ערכי המשתנים השונים.

מבחינת תוצאות האלגוריתמים, תוצאות המרחק האורתוגונלי של SVD-KSM כמעט תמיד היו טובות יותר משל האלגוריתם שלנו. ההפרש הזה גדל ככל ש- n היה יותר גדול, עד ל- n=5000 שם המרחקים התחילו להצטמצם. מגמה דומה נצפתה גם בממד (d) כאשר ככל שהממד נהיה גדול יותר ההפרש היה גדל לטובת ה- SVD-KSM. מהתוצאות שלנו לא נראה שלערך ה- k הייתה השפעה כלשהי על ערכי התוצאה.