(20942) מבוא ללמידה חישובית | סיכום הרצאה 12

מנחה: ד"ר שי מימון
סמסטר: 2022א'
נכתב על ידי: מתן כהן

חלק 1 Unsupervised Learning with Clustering

1 Supervised vs Unsupervised Learning

למידה מונחית	למידה לא מונחית
מצויות תצפיות אשר לכל תצפית יש תווית	מצויות תצפיות ללא תוויות או מנחה ועל בסיס סט התצפיות אנחנו צריכים להבין מה התבנית/מבנה המידע. נתייחס למידע (שהוא multi-dimensional vector) אשר ממנו נסיק מאפיינים של פונקצית ההתפלגות ללא כל עזרה חיצונית
ישנה אפשרות לבחון את ההתאמה של המודל ואת התוצאות שלנו	קשה לבחון את טיב האלגוריתמים שלנו. הרבה פעמים "בורחים" לשיטות היריסטיות על מנת לתת מוטיבציה לאלגוריתמים ולשפוט את איכות התוצאות שלנו. המצב הנ"ל גורר הרבה אלגוריתמים שונים. כמו-כן אין לנו ground truth שאליו אנחנו יכולים להשוות את הביצועים שלנו בשביל לבדוק איזה חזאי טוב יותר.

1.1 Unsupervised Learning

במצב בוא סט התצפיות שלנו מממד נמוך, יש המון שיטות המאפשרות לנו לשערך את פונקצית הצפיפות של X (סט התצפיות).
כאשר נדבר על מצב בו סט התצפיות מממד גבוה (מה שלרוב קורה) - השיטות שלנו לשערוך פונקצית הצפיפות לא עובדות בצורה אופטימלית.
מה שמנסים לעשות במקרים כאלו זה לקבל מודלים גלובליים גסים כמו Gaussian Mixture (נשמע עליו בהמשך) או לאפיין באמצעות מאפיינים סטטיסטיים מסוימים את פונקצית הצפיפות.

1.1.1 מה הכוונה ב"מאפיינים סטטיסטיים מסויימים"?

המטרה היא להבין איפה ההסתברות של X מקבלת ערכים **גבוהים**.
שיטות ה PCA ו- K-means clustering הן שיטות שכאלו.
בשיטת ה-Clustering רוצים למצוא איזורים במרחבו של X בהם ההסתברות מאוד גבוהה.
בשיטת ה-PCA מנסים להתייחס למידע כאילו הוא נמצא על יריעה מממד הרבה יותר נמוך - את היריעה הנ"ל מנסים למצוא.

2 Cluster Analysis

הגדרה: קיבוץ של אובייטקים לתתי-קבוצות או "קלאסטרים", כך שכל תת-קבוצה מאופיינת בכך שכל האובייקטים באותה קבוצה מאוד דומים אחד לשני (מוגדר על ידי פונקצית דמיון כלשהי - similarity function) ואובייקטים בקבוצות שונות שונים אחד מהשני.

קל להבין שישנן דרכים שונות ומגוונות להגדרת הקירבה בין אובייטקים בעזרת similarity function.

2.1 A Clustering Model

קלט

- סט של **אלמנטים** (אובייקטים)
- **פונקצית מרחק** סימטרית ($d(x_1, x_2) = d(x_2, x_2)$) המקיימת $d(x, x) = 0$ שתקבע עד כמה אובייקטים דומים או שונים.
- חלק מאלגוריתמי הקלאסטרינג כוללים גם פרמטר K אשר יגדיר את מספר הקבוצות שנרצה לקבל בתוצאה שלנו.
  - ישנן שיטות שיכולות ללמוד את מספר הקלאסטרים מתוך המידע

פלט

- **חלוקה** של קבוצת הדוגמאות שלנו לתתי-קבוצות

3 K-Means Clustering

- המטרה היא חלוקת המידע שלנו למספר קלאסטרים שנקבע מראש (K)
- מפרמל את הבעיה בעזרת **סנטרויד** (**centroid**) שיהיה נתון לכל קבוצה - כל נקודה בקבוצה תהיה קרובה לסנטרויד שלה יותר מאשר לכל סנטרויד אחר.
- האלגוריתם מעצם הגדרתו לא מבטיח התכנסות לאופטימום גלובלי
  - יתרה מזאת הרבה פעמים ההתכנסות תהיה לאופטימום מקומי

3.1 רקע

עבור K כלשהו ו- N מדידות מתקיימים:

$$\{\mathbf{x}_n\}_{n=1}^{N} \quad \mathbf{x}_n \in \mathbb{R}^d, \quad \boldsymbol{\mu}_k \in \mathbb{R}^d,\ k = 1, \ldots, K$$

כאשר $\boldsymbol{\mu}_k$ הוא פרוטוטייפ (**סטנרויד**) **והוא לא חייב להיות מסט הדוגמאות שלנו.**

כמו-כן נגדיר את r_{nk} בתור **שיוך נקודה $\mathbf{x}_n$ לקלאסטר k:**

$$r_{nk} = \begin{cases} 1 & \mathbf{x}_n \ \textbf{belongs to cluster k} \\ 0 & \textbf{else} \end{cases}$$

ופונקצית המטרה (בהנחה של r_{nk} ו- $\boldsymbol{\mu}_k$ ידועים):

$$J = \sum_{n=1}^{K} \sum_{k=1}^{K} r_{nk} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2$$

את הביטוי הנ"ל נרצה להביא למינימום כיוון שהוא מבטא את מרחקי הריבועים בין כל דגימה שנמצאת בקבוצה מסוימת לבין הסנטרויד של אותה קבוצה.

במילים אחרות מדובר **במרחקים בתוך קבוצה.**

הצעה: כעת נוכל למצוא השמה של כל נקודה $\mathbf{x}_n$ לקלאסטר k ואת הסנטרואידים שלנו בצורה אופטימלית ולכן נוכל לעבור על **כל ההשמות האפשריות** ולבחור את המינימום - כך נוכל למצוא את המינימום הגלובלי.

בעיה: אם סט הדוגמאות שלנו גדול הבעיה הקומבינטורית שהגדרנו מאוד קשה - משמע נעדיף להשתמש באלגוריתמים חמדניים שימצאו לנו פתרונות מספיק טובים גם אם לא האופטימליים ביותר.

3.2 K-Means Derivation

3.2.1 מציאת r_{nk}

נניח שקיימים לנו $\boldsymbol{\mu}_k$ לכל $k = 1, \ldots, K$ ונרצה לאפטם את הבעיה שלנו על פי r_{nk}, ובפרט עבור נקודה $\mathbf{x}_{n'}$ כלשהי.
נוכל להסיק מכך שמכלל הסכימה של פונקצית המטרה שלנו, נצטרך רק את אותה סכימה עבור אותה דוגמה ספציפית ולכן נתבונן בדוגמה כלשהי n' ונסיק:

$$\min_{r_{n'k}} \sum_{n=1}^{K} \sum_{k=1}^{K} r_{nk} \left\|\mathbf{x}_n - \boldsymbol{\mu}_k\right\|^2 \equiv \min_{r_{n'k}} \sum_{k=1}^{K} r_{n'k} \left\|\mathbf{x}_{n'} - \boldsymbol{\mu}_k\right\|^2$$

וכעת מבחינה אינטואיטיבית נרצה ש $r_{n'k}$ יהיה 1 רק עבור אותו k שהמרחק $\left\|\mathbf{x}_{n'} - \boldsymbol{\mu}_k\right\|^2$ יהיה מינימלי עבורו ולכן:

$$\boxed{\begin{aligned} j &= \underset{k\in\{1,\ldots,K\}}{\operatorname{argmin}} \left\|\mathbf{x}_{n'} - \boldsymbol{\mu}_k\right\|^2 \qquad & r_{nj} &= 1 \\ & & r_{nk} &= 0 \;\; \forall k \neq j \end{aligned}}$$

3.2.2 מציאת $\boldsymbol{\mu}_k$

נרשום את הבעיה החדשה (בדומה להסבר של 3.2.1 גם כאן נוכל להשמיט סכימה):

$$\min_{\boldsymbol{\mu}_\ell} J \equiv \min_{\boldsymbol{\mu}_\ell} \sum_{n=1}^{N} r_{n\ell} \left\|\mathbf{x}_n - \boldsymbol{\mu}_\ell\right\|^2$$

נוכל לגזור ולבדוק:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\mu}_\ell} \sum_{n=1}^{N} r_{n\ell} \left\|\mathbf{x}_n - \boldsymbol{\mu}_\ell\right\|^2 = 2 \cdot \sum_{n=1}^{N} r_{n\ell} \left(\mathbf{x}_n - \boldsymbol{\mu}_\ell\right) = 0 \\ \iff \sum_{n=1}^{N} r_{n\ell} \mathbf{x}_n = \left(\sum_{n=1}^{N} r_{n\ell}\right) \cdot \boldsymbol{\mu}_\ell \end{aligned}$$

ולכן:

$$\boxed{\boldsymbol{\mu}_\ell = \frac{\sum_{n=1}^{N} r_{n\ell} \mathbf{x}_n}{\sum_{n=1}^{N} r_{n\ell}}, \quad \ell = 1, \ldots, K}$$

ובעצם, נבחין כי המכנה הוא מספר הנקודות בקלאסטר ℓ - נסמנו ב- N_ℓ
בצורה דומה נבחין כי המונה זה **סכום כלל הנקודות** ששייכות **לקלאסטר** ℓ
ובעצם:

$$\boldsymbol{\mu}_\ell = \frac{\text{sum of points in cluster } \ell}{\text{number of points in cluster } \ell}$$

במלים אחרות, $\boldsymbol{\mu}_\ell$ הוא ממוצע הנקודות בקלאסטר ℓ.
כעת, בעזרת ממוצעים אלו נוכל להתבונן בכל אחת מהנקודות ולבדוק לאילו מהממוצעים היא הכי קרובה - אל אותו ממוצע נשייך את הנקודה. את פעולה זו נבצע שוב ושוב עד אשר הכל מתייצב ואין יותר שינויים (ההשמות לא משנות את הקבוצות) או עד אשר הגענו למגבלת איטרציות.

3.3 אתחול האלגוריתם

האתחול של האלגוריתם יכול להיות אקראי וישנן 2 אופציות מוכרות:

(1) אתחול על פי נציגים אקראיים מהדוגמאות

(2) אתחול על פי השמה מסוימת - חלוקה לקבוצות על פי הגיון התחלתי

באופן כללי נהוג לאתחל את האלגוריתם בעזרת קביעת הנציגים (סנטרואידים) בצורה אקראית מתוך סט הדוגמאות שלנו בתקווה שנקבל חלוקה טובה.

- תוצאות האלגוריתם משתנות בהתאם לאתחול ולכן ניתן לבחון אתחולים שונים
- אתחול על פי K-Means++

3.4 אתחול על פי K-Means++

אתחול זה הוא שיפור לאופן האתחול בו בחירת הנקודות תהיה בצורה כמה שיותר "מפוזרת" על פי סט הדוגמאות שלנו ופועל באופן הבא:

(1) נבחר תחילה את הסנטרויד הראשון אקראית מכלל הנקודות הקיימות

(א) נבחר את הסנטרויד הבא בצורה אקראית על פי הסתברות לא אחידה - פרופורציונלית למרחק הנקודות מהסטנרויד הקודם - משמע לאחר בחירת סטנרויד כלל הנקודות מקבלות התסברויות על פי מרחקן מהסנטרויד שנבחר - ככל שרחוקות יותר ההסתברות גבוהה יותר וכך נבחר את הסנטרויד הבא על פי אותן הסתברויות

(ב) כך נקבל פונקצית הסתברות לבחירת הסנטרואידים.

3.5 בחירת פונקצית המרחק

כידוע, פונקצית המרחק הריבועית **מאוד רגישה** ל-outliers ולפיכך במצבים בהם יש לנו כאלו, זוהי לא הבחירה הטובה ביותר. לשם כך נוכל להגדיר פונקציות אחרות שימדדו את המרחק בין הנקודות אך נתמודד עם הבעיה במציאת האופטימום, שכן לא כל הפונקציות הן ריבועיות ופשוטות לעבודה.

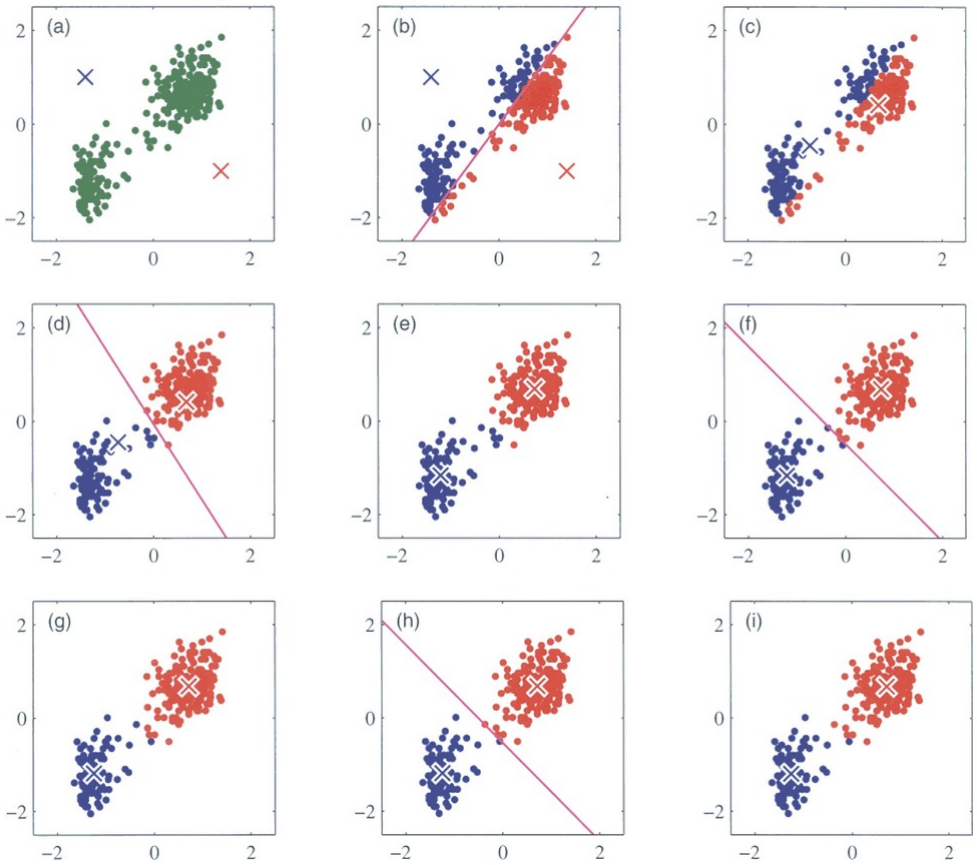
דרך לפשט את הדבר היא אילוץ הסנטרואידים להיות **אך ורק נקודות מתוך הקלאסטר** וכך בבדיקת ערך המינימום

$$\min_{\boldsymbol{\mu}_\ell} J \equiv \min_{\boldsymbol{\mu}_\ell} \sum_{n=1}^{N} r_{n\ell} \left\|\mathbf{x}_n - \boldsymbol{\mu}_\ell\right\|^2$$

נוכל לקבע נקודה $\boldsymbol{\mu}_m$ אחת בכל פעם ולמצוא את האופטימום.

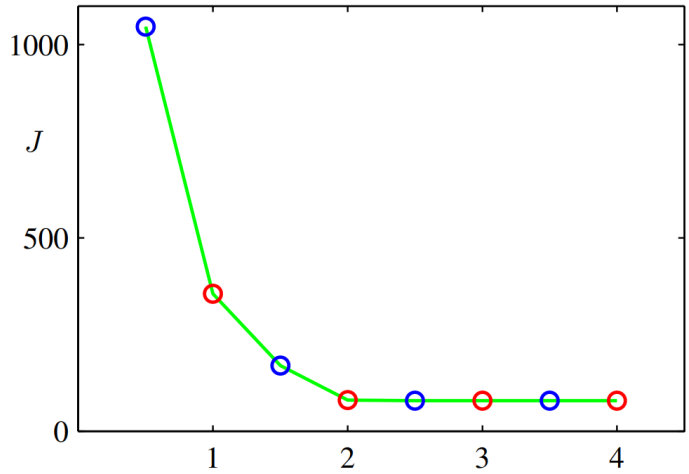
3.6 דוגמאות

3.6.1 חלוקת נקודות במרחב



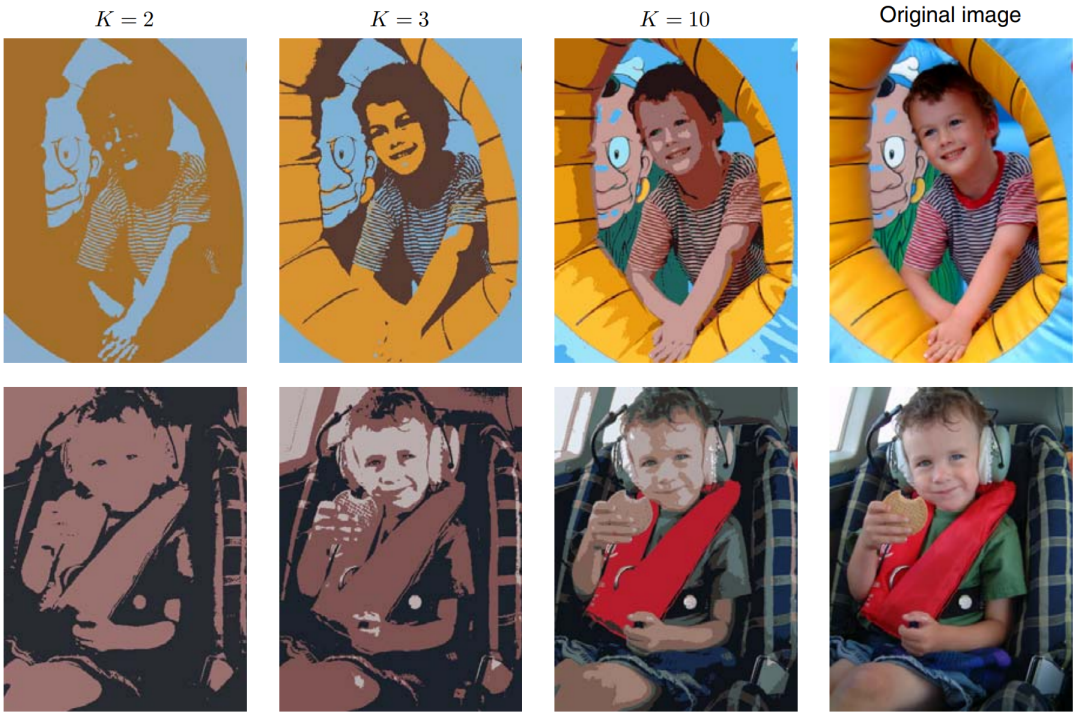
ב- (a) התבצעה קביעה "רעה" של סנטרואידים
ב- (b) רואים את הקו המפריד בין החלוקות וחלוקה ראשונית לקלאסטרים
ב- (c) רואים את את הסנטרואידים החדשים שנוצרו על פי מיצוע הדוגמאות
ב- (d) רואים את החלוקה החדשה לקלאסטרים על פי הסטנרואידים שהתקבלו ב- (c)
בתמונות $(e) - (h)$ התהליך חוזר על עצמו עד אשר אין שינויים יותר
ב- (i) קיבלנו את התוצאה הסופית

ניתן להבחין בקצב הירידה של ערכי J:



3.6.2 סגמנטציית תמונות

הכוונה כאן היא לגרום לכך שאיזורים דומים בתמונה יהיו אחידים



התמונה המקורית כוללת $240 \times 180 = 43,200$ פיקסלים
ישנם 3 צבעים - R, G, B כאשר כל צבע מיוצג על פי 8 ביטים - ערכים מ 0 ועד 255 (מידת בהירות של הצבע)
לכן על פיקסל מיוצג על ידי 24 ביטים ולכן כל תמונה מיוצגת על ידי $24 \times 43,200 = 1,036,800$ ביטים!
נוכל להשתמש ב- K-Means ועל פי בחירת K נוכל להגדיר את מספר הערכים האפשריים עבור מידת ה- RGB.
בצורה זו נוכל להגביל את מספר הביטים בתמונה ובכך להקטין את כמות המידע שצורכת.
נבחין כי עבור $K = 2$ נקבל תמונה עם מספר ביטים מאוד נמוך וכתוצאה מכך גם תמונה פחות ברורה.
בנוסף, נוכל להבחין כי כיוון שישנן K אפשרויות לייצג כל פיקסל, נוכל להגיד כי כל פיקסל מיוצג על ידי $\log_2 K$ ביטים.
ישנם N פיקסלים סה"כ, וכל אחד מהסנטרואידים מורכב מ 24 ביטים.
מכך נוכל להסיק שהנוסחה לחישוב מספר הביטים בתמונה עם K ערכים אפשריים עבור מידת RGB היא

$$24K + N \log_2 K$$

3.7 בחירת ערכי K

3.7.1 הכשלון של הולידציה

בנוסף, צריך לזכור שמקרים של Supervised Learning השתמשנו בסט ולידציה על מנת לקבוע את ההיפר-פרטמטרים שלנו אך כאן **זה כבר לא יעבוד לנו**!

אם נגדיל את ערכי K אנחנו נגדיל את מספר הסנטרואידים וכתוצאה מכך אנחנו נקטין את המרחקים בין כל אחת מהנקודות לסנטרויד הכי קרוב אליה - משמע ככל שנגדיל את K פונקצית המחיר J תקטן ונגיע לבסוף למצב שבו כל נקודה שייכת לקלאסטר משל עצמה.

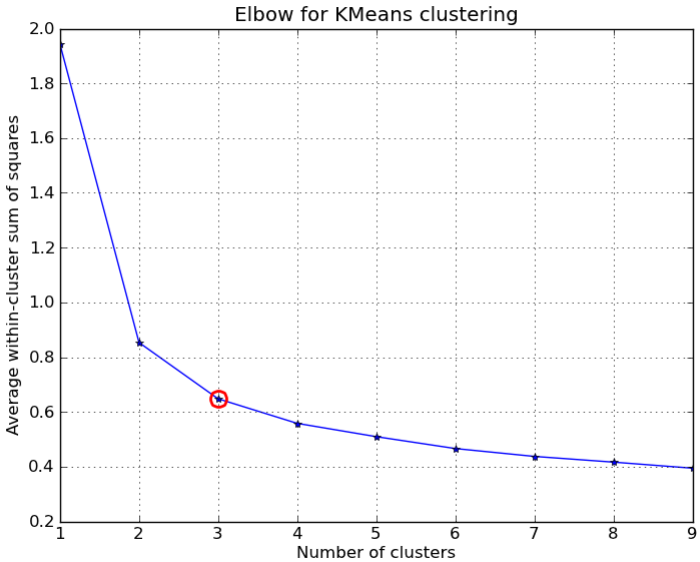
מהסיבה הזו בדיוק שיטות הולידציה לא תעבודנה.

3.7.2 Elbow Method

נבחין כי קיים איזשהו K^* אשר מייצג את המידע שלנו בצורה טובה.

כל עוד לא הגענו לאותו K^* - כל הגדלה של K תפריד קבוצות ויהיה שיפור משמעותי **ולהפחתה משמעותית** בפונקצית המחיר J.

במידה והגענו לאותו K^*, כל הגדלה נוספת תגרום להפרדה **מלאכותית** של קבוצות ותגרום **להפחתה מאוד קטנה** של פונקצית המחיר!



חלק 2 Dimensionality Reduction

רקע

הגדרה. הפחתת ממד הוא תהליך בו נלקח מידע **מממד גבוה** (כמות גדולה של פיצ'רים) וממופה למרחב מממד **הרבה יותר נמוך**.

מוטיבציות להפחתת ממדים

- ככל שהדוגמאות מממד גבוה יותר - ככה הבעיה יותר מורכבת **מבחינה חישובית**
  - Curse of Dimensionality - ככל שאנחנו בממדים יותר גבוהים, ביצועי המודל פוחתים
- במצב בו יש הרבה פיצ'רים אנחנו עלולים לבצע overfitting ולכן אנחנו פוגעים **במידת ההכללה**
- קל בהרבה מקרים להתבונן על המידע שלנו בממדים נמוכים (נניח ממד 2) וכך **להסיק ממנו מסקנות**

1 PCA - Principal Component Analysis

1.1 הקדמה ומוטיבציה

- בהרבה מקרים נבחין כי המידע שלנו, אף על פי שהוא **מממד מאוד גבוה**, בפועל הוא שוכב על **יריעה** כלשהי **מממד הרבה יותר נמוך**
- נרצה למצוא את אותו ממד שמייצג את האינפורמציה החבויה במדידות הללו
- אחת הגישות הכי פופולריות להורדת ממד היא PCA ומשמשת בין היתר להורדת ממד, דחיסה, קבלת תובנות על איזה פיצ'רים מייצגים את המידע שלנו ואף ויזואליזציה למידע שלנו.

1.1.1 מדוע נרצה להקטין את הממד?

ראינו כבר בשיעורים הקודמים שישנה אופציה להשתמש בטרנספורמציות כאלו ואחרות על מנת להגדיל את ממד הבעיה שלנו על מנת להתמודד עם בעיות לא ליניאריות בשימוש עם מודלים לינארים.

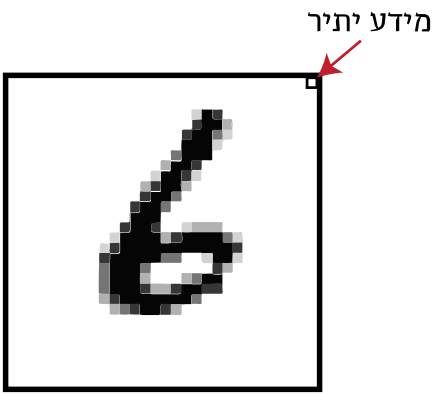
הגדלת הממד עזרה לנו להקטין את ה E_{in} ושילמנו בכך **שפגענו ביכולת להכליל** ואף בהגעה ל-**overfitting.**

כעת, נוכל להסתכל על הדרך ההפוכה - אם נוכל להקטין את הממד שלנו ללא פגיעה ב- E_{in} , נוכל גם לשפר **את ההכללה** שלנו.

1.1.2 מהות השימוש ב- PCA

השימוש ב- PCA עוזר לנו להפטר מממדים שהם לא אינפורמטיביים/יש בהם יתירות (האינפורמציה בהם קיימת בממדים אחרים) או ממדים שמהווים רעש שהוא לא רלוונטי למידע שלנו.

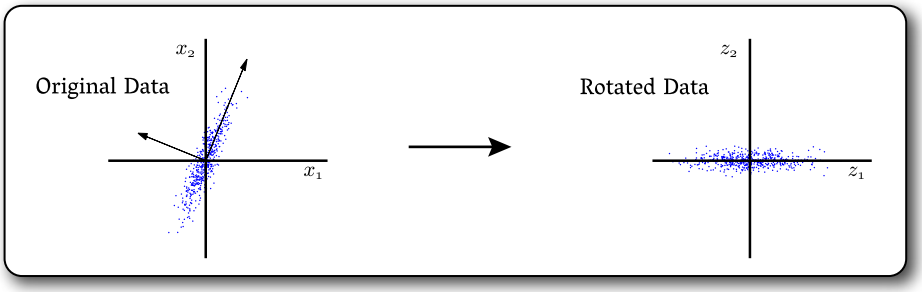
הורדת הממד במקרים כאלו תוכל **לשפר לנו את ההכללה**!



כאן, ישנם המון פיקסלים שהם כמעט לבנים ולא תורמים לנו מידע - הפחתה בהם תוכל לעזור לנו בהכללת הבעיה!

1.2 אופן הפעולה של PCA

- לאחר שהבנו ש-PCA מנסה לתמצת את המידע שלנו, נבין כיצד הוא עושה זאת.
- הרעיון העומד מאחורי PCA הוא הסתכלות על המידע במערכת צירים חדשה
- את מערכת הצירים הוא מנסה לסובב בצורה כזו שממדים חשובים יבלטו וככה נוכל לשמר אותם וממדים פחות חשובים נוכל לזרוק



בדוגמה זו רואים שהמידע על ציר z_1 מתפרש על כלל הציר ועליו ניתן להבחין בין נקודה אחת לאחרת

מאידך, בציר z_2 קשה מאוד לראות הבדלים בין הנקודות וישנו סיכוי שהן שם כתוצאה מרעש כלשהו!

כתוצאה מכך נוכל "לזרוק" את המידע מציר z_2 ולנסות להתמודד יותר טוב עם הבעיה שלנו.

מילת המפתח כאן היא **שונות!**

1.3 שתי הגדרות - אלגוריתם אחד

ניתן להגדיר את PCA בשתי דרכים שונות:

הגדרה 1. ההיטל האורתוגונלי של המידע אל תת-מרחב ממד נמוך יותר (principal subspace) כך **שהשונות** של הנקודות המוטלות היא **מקסימלית**.

הגדרה 2. ההטלה הלינארית של המידע שלנו אל ממד נמוך יותר אשר מביאה **למינימום** את **השגיאה הריבועית הממוצעת** בין הנקודות המקוריות לבין הנקודות המוטלות.

1.4 פרמול השונות המקסימלית (הגדרה 1)

נתייחס לסט דוגמאות $\{\mathbf{x}_n\}_{n=1}^N$ אשר $\mathbf{x}_n \in \mathbb{R}^D$.

המטרה שלנו היא להטיל את המידע שלנו לתת-מרחב בעל ממד $M < D$ כאשר אנחנו **ממקסמים** את **השונות** של הנקודות המוטלות.

1.4.1 הטלה למרחב ממד $M = 1$

על מנת לבצע את ההטלה למרחב ממד $M = 1$ נבחר וקטור $\mathbf{u}_1 \in \mathbb{R}^D$ אשר יקבע את הכיוון שאליו נטיל את הנקודות ונתייחס לנורמה של $\mathbf{u}_1$ בתור 1 - משמע $\mathbf{u}_1^T\mathbf{u}_1 = 1$.

נסמן ב- $\mathbf{z}_n$ את הנקוה $\mathbf{x}_n$ מוטלת בעזרת $\mathbf{u}_1$:

$$\mathbf{z}_n = \underbrace{\frac{(\mathbf{x}_n^T \cdot \mathbf{u}_1)}{(\mathbf{u}_1^T \cdot \mathbf{u}_1)}}_{1} \cdot \mathbf{u}_1$$

נרצה להתבונן בממוצע הערכים של הנקודות המוטלות (sample mean of projected data points) (ללא הכיוון):

$$\frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n^T \cdot \mathbf{u}_1 = \left(\frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n^T\right) \cdot \mathbf{u}_1 = \overline{\mathbf{x}}^T \cdot \mathbf{u}_1$$

ולחשב את שונות הערכים של הנקודות המוטלות (sample variance of projected data points):

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{x}_n^T\mathbf{u}_1-\overline{\mathbf{x}}^T\mathbf{u}_1\right)^2 &= \frac{1}{N}\sum_{n=1}^{N}\left(\underbrace{(\mathbf{x}_n-\overline{\mathbf{x}})^T\cdot\mathbf{u}_1}_{\overset{\text{scalar}}{=}\mathbf{u}_1^T\cdot(\mathbf{x}_n-\overline{\mathbf{x}})}\right)^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{u}_1^T\cdot(\mathbf{x}_n-\overline{\mathbf{x}})\cdot(\mathbf{x}_n-\overline{\mathbf{x}})^T\cdot\mathbf{u}_1 \\ &= \mathbf{u}_1^T\cdot\underbrace{\left[\frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\overline{\mathbf{x}})\cdot(\mathbf{x}_n-\overline{\mathbf{x}})^T\right]}_{S_{D\times D}\text{ - sample covariance of the data points}}\cdot\mathbf{u}_1\end{aligned} \tag{1.1}$$

מטריצת S כוללת את איברי S_{ij} אשר כל אחד מהם הוא ה-covariance בין פיצ'ר i של $\mathbf{x}$ לבין פיצ'ר j של $\mathbf{x}$.
כמו-כן מטריצה S היא **סימטרית** ולכן

$$S = S^T \tag{1.2}$$

ולכן נגדיר את **השונות**:

$$\boxed{J = \mathbf{u}_1^T\cdot S\cdot\mathbf{u}_1}$$

וכעת נרצה לפתור את בעית האופטימיזציה:

$$\begin{aligned}\max_{\mathbf{u}_1}\ & \mathbf{u}_1^T\cdot S\cdot\mathbf{u}_1 \\ \textbf{s.t}\ & \|\mathbf{u}_1\|^2 = 1\end{aligned} \tag{1.3}$$

לשם כך נשתמש בכופלי לגראנג' כאשר את האילוץ נכתוב כאילוץ **שוויון ל-0**:

$$\max\ \underbrace{\mathbf{u}_1^T\cdot S\cdot\mathbf{u}_1+\lambda\left(1-\mathbf{u}_1^T\mathbf{u}\right)}_{\tilde{J}}$$

נגזור על פי $\mathbf{u}_1$ ו- λ ונשווה לאפס:

$$\begin{aligned}\frac{\partial\tilde{J}}{\partial\mathbf{u}_1} &= \left(\underbrace{S+S^T}_{\overset{1.2}{=}2S}\right)\mathbf{u}_1-2\lambda\mathbf{u}_1=\mathbf{0} \\ &\overset{1.2}{\iff} S\mathbf{u}_1=\lambda\mathbf{u}_1\end{aligned} \tag{1.4}$$

$$\frac{\partial\tilde{J}}{\partial\lambda} = 1-\|\mathbf{u}_1\|^2 = 0 \iff \|\mathbf{u}_1\|^2 = 1$$

כעת, קל להבחין מ- 1.4 שבעצם $\mathbf{u}_1$ הוא **וקטור עצמי** של S. נכפיל משמאל $\mathbf{u}_1^T$ ונקבל:

$$S\mathbf{u}_1 = \lambda\mathbf{u}_1 \xrightarrow{\mathbf{u}_1^T\cdot()} \mathbf{u}_1^T S\mathbf{u}_1 = \lambda\underbrace{\|\mathbf{u}_1\|^2}_{1} \Rightarrow \boxed{\mathbf{u}_1^T S\mathbf{u}_1 = \lambda}$$

משמע, קיבלנו ש-λ זו בעצם השונות של המידע המוטל שלנו J:

$$J = \mathbf{u}_1^T S\mathbf{u}_1 = \lambda$$

וכיוון שאנחנו רוצים **שהשונות שלנו תהיה מקסימלית** נרצה לבחור את הוקטור העצמי $\mathbf{u}_1$ שלו יש את הערך העצמי λ המקסימלי!

1.4.2 סיכום תהליך הפעולה

- הגדרנו את מטריצה S מסדר $D \times D$ - מטריצת השונות המשותפת של הדוגמאות שלנו שהיא ממשית וסימטרית
  - מכך שהיא ממשית וסימטרית נוכל להסיק 2 מסקנות חשובות:
    (1) היא לכסינה אוניטרית ו**כלל הערכים העצמיים שלה הם ממשיים**
    (2) הוקטורים העצמיים שלה **אורתוגונליים**
    (3) ניתן להשתמש בפירוק הספרטקלי ולרשום אותה בצורה מתאימה
  - כמו-כן כיוון שזו מטריצת השונות המשותפת היא **חיובית למחצה** (P.S.D)
    - ⋆ לכן הערכים העצמיים שלה **הם אי-שליליים**
    - ⋆ בד"כ הערכים הם חיוביים ממש
- רצינו למצוא את הוקטורים והערכים העצמיים של S:

$$S\mathbf{u}_i = \lambda_i\mathbf{u}_i, \quad i = 1, \dots, D$$

- באופן כללי יכלנו להגדיר מטריצה $U = \begin{bmatrix}\mathbf{u}_1 & \dots & \mathbf{u}_D\end{bmatrix}$ מטריצה שעמודותיה הם הו"ע של S ולרשום ומטריצה $\Lambda = diag\{\lambda_1, \dots, \lambda_D\}$:

$$S \cdot U = U \cdot \Lambda$$

- בעזרת מה שהגדרנו נוכל להשתמש בפירוק הספקטרלי ובעובדה שמתכונות U נקבל: $UU^T = U^TU = I$ (U^T ההפכית של U)

$$S = U\Lambda U^{-1} = U\Lambda U^T$$

- בעזרת הפירוק נוכל למצוא את הוקטור העצמי של S ששייך לע"ע המקסימלי של S.

1.5 פרמול השגיאה המינימלית (הגדרה 2)

בשלב הראשון נגדיר סט קורדינטות חדשות $\mathbf{u}_i$ שהם וקטורים שיגידרו לנו את מערכת הצירים כאשר כל ה-$\mathbf{u}_i$-ים מנורמלים אחד לשני והנורמה שלהם היא 1 - אותם וקטורים $\mathbf{u}_i$ מהווים בסיס **אורתונורמלי** לאותו מרחב מממד $\mathbb{R}^D$ שממנו הגיעו הדוגמאות שלנו.

כיוון ש $\mathbf{u}_i$ הם וקטורי הבסיס, נכול לייצג כל נקודה $\mathbf{x}_n$ בעזרת וקטורים אלו:

$$\mathbf{x}_n = \sum_{i=1}^{D} \alpha_{ni}\mathbf{u}_i \tag{1.5}$$

$$\xrightarrow{\mathbf{u}_j^T\cdot()} \mathbf{u}_j^T\mathbf{x}_n = \sum_{i=1}^{D} \alpha_{ni}\left(\mathbf{u}_j^T\mathbf{u}_i\right) \overset{\mathbf{u}_i^T\mathbf{u}_j=0,\ i\neq j}{=} \alpha_{nj} \tag{1.6}$$

$$\xrightarrow{1.5+1.6} \mathbf{x}_n = \sum_{i=1}^{D} \left(\mathbf{x}_n^T\mathbf{u}_i\right)\mathbf{u}_i$$

כעת נוכל לקרב את הנקודה $\mathbf{x}_n$ בעזרת $M < D$ ממדים בלבד:

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^{M} z_{ni}\mathbf{u}_i + \sum_{i=M+1}^{D} b_i\mathbf{u}_i$$

ונרצה להביא **למינימום** את המרחק בין הנקודה מממד M לנקודה המקורית:

$$J = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 = \frac{1}{N}\sum_{n=1}^{N}\left\|\mathbf{x}_n - \left(\sum_{i=1}^{M} z_{ni}\mathbf{u}_i + \sum_{i=M+1}^{D} b_i\mathbf{u}_i\right)\right\|^2$$

נגזור על פי z_{mj} ונשווה לאפס:

$$\begin{aligned}
\frac{\partial J}{\partial z_{mj}} &= \frac{2}{N}\mathbf{u}_j^T\left(\mathbf{x}_m - \left(\sum_{i=1}^{M} z_{mi}\mathbf{u}_i + \sum_{i=M+1}^{D} b_i\mathbf{u}_i\right)\right) \\
&= \frac{2}{N}\left(\mathbf{u}_j^T\mathbf{x}_m - \left(\sum_{i=1}^{M} z_{mi}\mathbf{u}_j^T\mathbf{u}_i + \sum_{i=M+1}^{D} b_i\mathbf{u}_j^T\mathbf{u}_i\right)\right) \\
&= \frac{2}{N}\left(\mathbf{u}_j^T\mathbf{x}_m - z_{mj}\right) = 0, \quad j = 1,\ldots,M
\end{aligned}$$

ונקבל:

$$\boxed{z_{nj} = \mathbf{u}_j^T\mathbf{x}_n, \quad \text{for } j = 1,\ldots,M}$$

נגזור על פי b_j ונשווה לאפס (נבחין ש $j = M+1,\ldots,D$ כי רק שם מוגדר b_j):

$$\begin{aligned}
\frac{\partial J}{\partial b_j}] &= \frac{2}{N}\sum_{n=1}^{N}\mathbf{u}_j^T\left(\mathbf{x}_n - \left(\sum_{i=1}^{M} z_{ni}\mathbf{u}_i + \sum_{i=M+1}^{D} b_i\mathbf{u}_i\right)\right) \\
&= \frac{2}{N}\sum_{n=1}^{N}\left(\mathbf{u}_j^T\mathbf{x}_n - b_j\right) = 0, \quad \boldsymbol{j = M+1,\ldots,D}
\end{aligned}$$

ונקבל:

$$\boxed{b_j = \mathbf{u}_j^T\cdot\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n = \mathbf{u}_j^T\overline{\mathbf{x}}, \quad \text{for } \boldsymbol{j = M+1,\ldots,D}}$$

נציב חזרה את מה שמצאנו בשביל לקבל את השגיאה:

$$\begin{aligned}
\mathbf{x}_n - \tilde{\mathbf{x}}_n &= \sum_{i=1}^{D} \left(\mathbf{x}_n^T \mathbf{u}_i\right) \mathbf{u}_i - \sum_{i=1}^{M} \left(\mathbf{x}_n^T \mathbf{u}_i\right) \mathbf{u}_i - \sum_{i=M+1}^{D} \left(\overline{\mathbf{x}}^T \mathbf{u}_i\right) \mathbf{u}_i \\
&= \sum_{i=M+1}^{D} \left(\mathbf{x}_n^T \mathbf{u}_i\right) \mathbf{u}_i - \sum_{i=M+1}^{D} \left(\overline{\mathbf{x}}^T \mathbf{u}_i\right) \mathbf{u}_i \\
&= \sum_{i=M+1}^{D} \left[\left(\mathbf{x}_n - \overline{\mathbf{x}}\right)^T \mathbf{u}_i\right] \mathbf{u}_i
\end{aligned}$$

והשגיאה הנ"ל נמצאת בתת-המרחב האורתוגונלי למרחב עליו הטלנו את המדידות שלנו.
נרצה להביא את הביטוי המייצג את ממוצע מרחקי הריבועים של השגיאות הנ"ל למינימום:

$$\frac{1}{N} \left\|\mathbf{x}_n - \tilde{\mathbf{x}}_n\right\|^2$$

נבטא את הממוצע הנ"ל על פי מה שמצאנו:

$$\begin{aligned}
J &= \frac{1}{N} \sum_{n=1}^{N} \left(\sum_{i=M+1}^{D} \left[\left(\mathbf{x}_n - \overline{\mathbf{x}}\right)^T \mathbf{u}_i\right] \mathbf{u}_i \right)^T \left(\sum_{i=M+1}^{D} \left[\left(\mathbf{x}_n - \overline{\mathbf{x}}\right)^T \mathbf{u}_i\right] \mathbf{u}_i \right) \\
\mathbf{u}_i^T \mathbf{u}_j = 0,\ \forall i \neq j \quad &= \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} \left(\underbrace{\left(\mathbf{x}_n - \overline{\mathbf{x}}\right)^T \mathbf{u}_i}_{\overset{\text{scalar}}{=} \mathbf{u}_1^T \cdot (\mathbf{x}_n - \overline{\mathbf{x}})} \right)^2 \\
\text{same trick at } 1.1 \quad &= \sum_{i=M+1}^{D} \mathbf{u}_i^T S \mathbf{u}_i
\end{aligned}$$

דוגמה. עבור $M = 1,\ \ D = 2$:

$$J = \mathbf{u}_2^T S \mathbf{u}_2$$

נרצה לפתור:

$$\min_{\mathbf{u}_2} \ \mathbf{u}_2^T S \mathbf{u}_2, \ \ \textbf{s.t} \ \left\|\mathbf{u}_2\right\|^2 = 1, \ \mathbf{u}_2^T \mathbf{u}_1 = 0$$

רואים שהבעיה כאן דומה ל- 1.3 וכיוון שמדובר כאן בבעיית **מינימום** (ולא מקסימום) נרצה לשנות את התשובה שלנו ולהגדיר את $\mathbf{u}_2$ להיות:

הוקטור העצמי של S המתאים לערך העצמי **הקטן ביותר** של $\mathbf{u}_2 = S$

מכך ש $\mathbf{u}_2$ הוא הו"ע של ע"ע הכי נמוך - בהכרח $\mathbf{u}_1$ הוא ו"ע של ע"ע הכי גדול - משמע קיבלנו את אותה תוצאה כמו בהגדרה 1.
ומכך נוכל להסיק ש-

$$J = \lambda_2$$

———

הערה. באופן כללי נוכל להגיד ש:

$$J = \sum_{i=M+1}^{D} \lambda_i$$

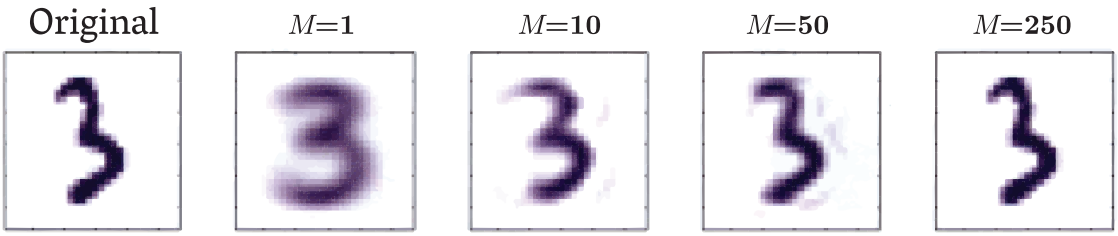
בהרבה מאוד מקרים, S יכולה להיות סינגולרית (לא הפיכה $\Leftarrow$ ע"ע 0) נבחין כי הורדת הממד לא מוסיפה לשגיאה שכן $\lambda_i = 0$ עבור אותם ערכים עצמיים. באופן כללי אם נתבונן במטריצת השונות המשותפת נראה **ערכים מאוד קטנים שקרובים ל-0 ואותם נוכל לזרוק.**

1.6 דחיסה - Compression

אחד השימושים של PCA הוא דחיסה של מידע וניתן לבטא את הקירוב בצורה הבאה:

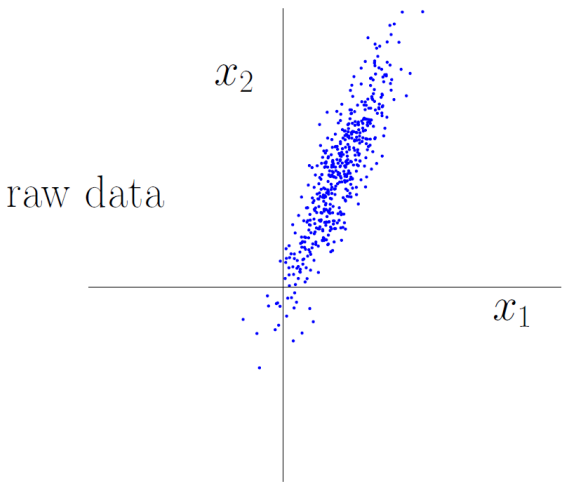
$$\begin{aligned}\tilde{\mathbf{x}}_n &= \sum_{i=1}^{M} \left(\mathbf{x}_n^T \mathbf{u}_i\right) \mathbf{u}_i + \sum_{i=M+1}^{D} \left(\overline{\mathbf{x}}^T \mathbf{u}_i\right) \mathbf{u}_i \\ &= \overline{\mathbf{x}} + \sum_{i=1}^{M} \left(\mathbf{x}_n^T \mathbf{u}_i - \overline{\mathbf{x}}^T \mathbf{u}_i\right) \mathbf{u}_i\end{aligned}$$

- ככל שנדחוס יותר - נאבד יותר מידע ונעלה את השגיאה



1.7 Data Pre-Processing

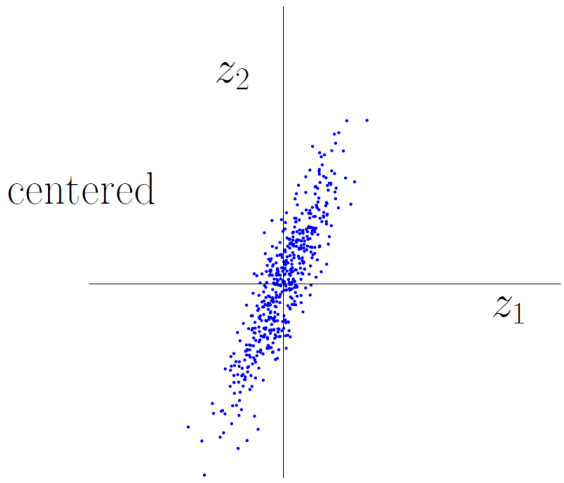
שימוש נוסף ל- PCA הוא pre-processing למידע שלנו - שינויים קטנים מלפני אימון המודל שלנו על מנת לשפר את הביצועים.



1.7.1 מרכוז - Centering

נשתמש בטרנספורמציה שתניב לנו נקודות ממורכזות בעזרת הממוצע $\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$:

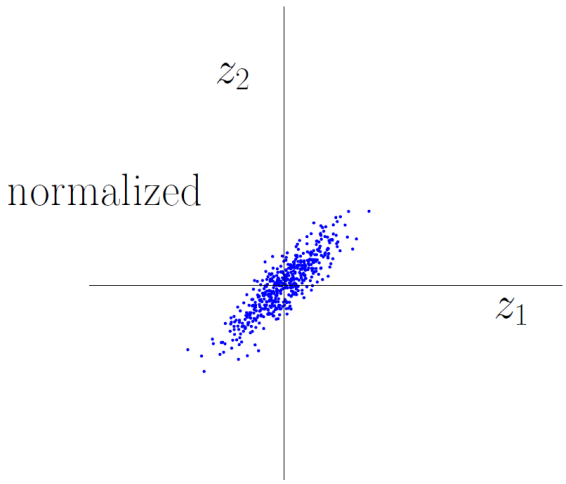
$$\mathbf{z}_n = \mathbf{x}_n - \overline{\mathbf{x}}$$



1.7.2 נרמול - Normalization

משמעות הדבר היא לגרום לכך שלכל הפיצ'רים תהיה אותה שונות - משמע דואג לכך שבכל אחד מהצירים השונות תהיה 1

$$\rho_{ij} = \frac{1}{N}\sum_{n=1}^{N}\frac{x_{ni}-\overline{x}_i}{\sigma_i}\cdot\frac{x_{nj}-\overline{x}_j}{\sigma_j}$$



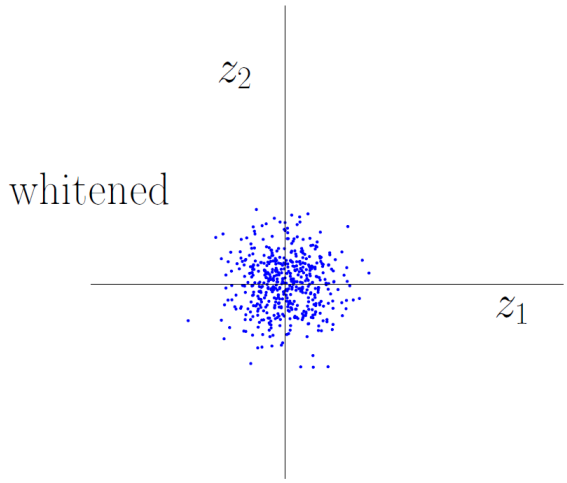
1.7.3 Whitening

אם נרצה להשלים את התמונה ולקבל פיצ'רים שהם **גם מנורמלים וגם חסרי קורלציה** נשתמש ב-Whitening
נשתמש במטריצת השונות המשותפת S, מטריצת הוקטורים העצמיים U ומטריצת הערכים העצמיים L ונבצע טרנספורמציה למידע:

$$\mathbf{y}_n = L^{-\frac{1}{2}} U^T \left(\mathbf{x}_n - \overline{\mathbf{x}}\right)$$

כך שאם נחשב את מטריצת השונות המשותפת של המידע החדש נקבל את מטריצת הזהות:

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\mathbf{y}_n\mathbf{y}_n^T &= L^{-\frac{1}{2}}U^T\left(\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{x}_n-\overline{\mathbf{x}}\right)\left(\mathbf{x}_n-\overline{\mathbf{x}}\right)^T\right)UL^{-1/2}\\ &= L^{-1/2}U^TULU^TUL^{-1/2} = I\end{aligned}$$



1.7.4 Test Set

צריך לזכור שסט **המבחן או הולידציה** שלנו לא עובר את אותו תהליך כמו סט האימון!

סט המבחן שלנו **לא עובר אף טרנספורמציה על פי המידע שלו עצמו** - **גם לא בתהליך ה-Pre-Processing**
כאשר עושים מרכוז או סטנדרטיזציה או כל תהליך אחר - מבצעים אותם **אך ורק על סט האימון**!
על מנת למרכז את המידע בזמן המבחן משתמשים במה שמצאנו על פי סט האימון!

דוגמה. אם רוצים לעשות מרכוז של המידע
תחילה נבצע מרכוז על סט האימון ונמצא ערך של $\overline{\mathbf{x}}$ - זהו הממוצע!
בעת שימוש בסט המבחן או הולידציה שלנו, **לא נמצא ערך חדש** לממוצע - אלא **נשתמש בערך שכבר מצאנו** בעת האימון.