שיעור 3

3 באפריל 2019

1 Unsupervised Learning

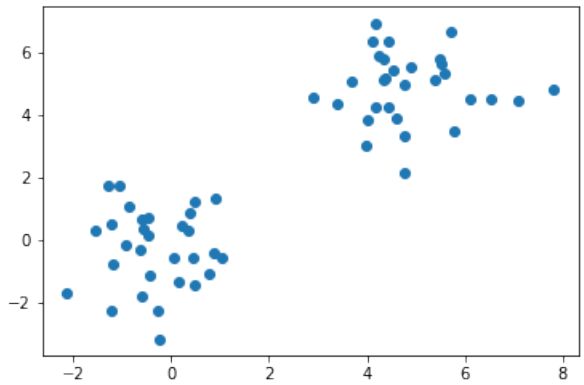
בשיעורים הקודמים הנחנו את היסודות לפרמול הבעיה בה אנו מתעסקים: הגדרנו מהו מודל, בחנו שיטות לאימון והתאמה של מודלים, הבנו כיצד למדוד מודלים וראינו דוגמאות למודלים שימושיים. העמקנו בדיון על עצי החלטה ובחנו נגזרות שונות של המודל, ובראשן את אלגוריתם random forest. שיטות אלו הן שיטות מתחום ה־supervised learning, מסגרת בה עבדנו עד כה, כאשר נתון לנו בסט האימון תיוג של המידע, כלומר תחזית רצויה לכל נקודה במידע. בשיעור זה נעסוק במשימות unsupervised learning, בהן דבר לא נתון לנו מלבד המידע עצמו, וכל המידע שנבקש לעשות בו שימוש "מקודד" בהתפלגות של המידע.
אילו בעיות הן בעיות supervised learning?

1. מציאת קבוצות חברים ברשת חברתית.
2. מציאת clusters, קבוצות, במידע.
3. הקטנת הייצוג של המידע, או מציאת פיצ'רים טובים יותר.

אנו נעסוק בעיקר בשתי הנקודות האחרונות, אותן ניתן לפגוש בהקשרים רבים בעבודה עם מידע. נעיר בשלב זה כי בבעיות מעשיות פעמים רבות, אף שהבעיה הייתה עשויה להינתן לניסוח כבעיית supervised, לא יעמוד לרשותנו תיוג איכותי דיו ובכמויות גדולות דיין לפתרון בעיית supervised learning, ונרצה ליישם שיטות מתחום ה־unsupervised learning לפתרונה.
כשאנו עוסקים בבעיות unsupervised learning, ובאופן כללי כשאנו מנהלים את חיינו, חשוב שנזכור את התובנה הבאה: "**דברים לא קורים רק בגלל שאנחנו רוצים שהם יקרו**". בפרט, לא נוכל לקוות "להצליח" במשימות unsupervised learning מבלי להסביר לעצמנו קודם היטב באיזו סיטואציה נצליח, מהן מגבלות האלגוריתמיקה שלנו ומהן מגבלות ההבנה שלנו את המידע. בעיני זו התובנה החשובה ביותר שעליכם לדעת בבואכם להתעסק במשימות מסוג זה, ונראה כמה וכמה דוגמאות לכך בהמשך השיעור.

1.1 Clustering

בבעיות אלו אנו מעוניינים לקבל חלוקה של המידע לקבוצות, שלא על־סמך תיוג כלשהו, אלא על־סמך התפלגות המידע בלבד. דוגמאות לצרכים אלו ניתן לראות בפילוח אוכלוסיה לפי תכונות מסוימות, מציאת קבוצות במידע גנטי, ועוד ועוד. נתבונן בדוגמא הבאה:

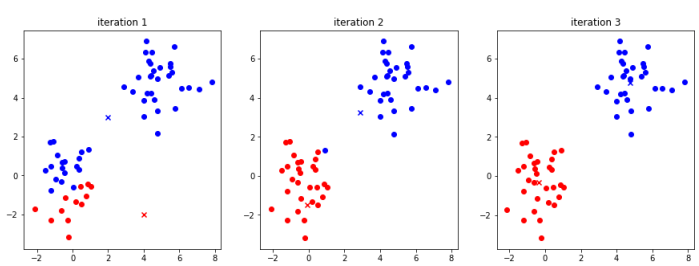


איור 1:

אנו מסוגלים לראות בקלות כי יש שתי קבוצות מובחנות במידע. מובן כי בבעיות מעשיות המידע שלנו הוא לרוב אינו דו־מימדי, ולא נוכל להשתמש באינטואיציה האנושית שלנו כדי לחלק אותו לקבוצות. נכיר כעת כמה אלגוריתמים שיאפשרו לנו למצוא clusters במידע.

1.1.1 KMeans

נכיר כעת את אלגוריתם ה־clustering הפשוט והנפוץ ביותר. בשיטה זו אנו נדרשים לספק את כמות ה־clusters, k. האלגוריתם פועל כך: בהינתן k, נתחיל בהגרלה של k מרכזי clusters. אלו k נקודות שרירותיות שנגדיר במרחב הפיצ'רים. האלגוריתם איטרטיבי, ובכל שלב נבצע את שתי הפעולות הבאות: ראשית, נשייך כל נקודה לנקודת מרכז ה־cluster הקרובה אליה. כלומר, לכל מרכז cluster אנו משייכים נקודות, כך שכל נקודה משויכת למרכז יחיד. בשלב השני נעדכן את מיקום מרכז ה־cluster כך שיהיה הממוצע (הבאריסנטר) של אוסף הנקודות ששייכנו אליו. נחזור על אלגוריתם עד להתכנסות.



איור 2:

עבור חלוקה ל־k קבוצות, נסמנן ב־$S_1, ..., S_k$ ונגדיר את

$$R_k = \sum_{i=1}^{k} \sum_{x \in S_i} |x - \mu(S_i)|^2 = \sum_{x} |x - \mu(s(x))|^2$$

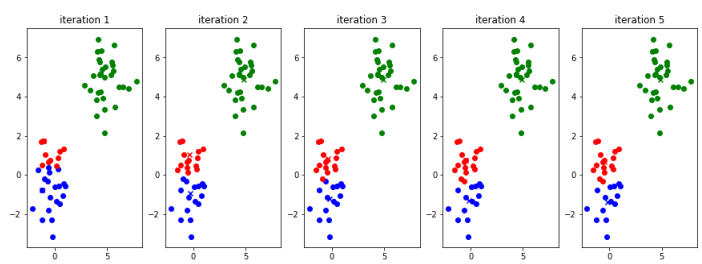
כאשר $\mu(S_i)$ הוא מרכז המסה של הקבוצה S_i (או הממוצע של כל הנקודות בה) ו־$s(x) = S_i$ עבורה $x \in S_i$. זהו סכום המרחקים של כל x ממרכז ה־cluster שלו. אנו מצפים כי בחלוקה טובה לקבוצות S_i מרחק זה יהיה קטן. למעשה, kmeans מנסה למזער את הגודל הזה. נראה זאת ע"י כך שנראה כי בכל איטרציה של האלגוריתם, R_k בהכרח קטן (או נשאר קבוע).
אכן, בשלב ההשמה באיטרציה, כאשר כל נקודה מקבלת את $\mu(S_i)$ הקרוב אליה, אנו בהכרח לא מגדילים את R_k, שכן כל ביטוי $|x - \mu(s(x))|$ בהכרח יקטן או לא ישתנה.
בשלב חישוב המרכזים החדשים אנו מחליפים את הביטויים הישנים $\mu(S_i)$ באלו עבורם $\sum_{x \in S_i} |x - \mu(S_i)|$ מקבל מינימום. אכן,

$$\frac{d}{dm} \sum_{x \in S_i} (x - m)^2 = 2 \sum_{x \in S_i} (x - m)$$

כך שמינימום מתקבל ל־

$$m = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

לכן גם כאן אנו בהכרח לא מגדילים את R_k.
קיבלנו, בסך הכל, כי R_k בהכרח יורד במהלך האלגוריתם.
מהי נקודת התורפה הגדולה ביותר של KMeans? כמובן, לא תמיד נדע כיצד לקבוע את k.
נחזור לדוגמא הקודמת, אלא שהפעם נבחר $k = 3$:

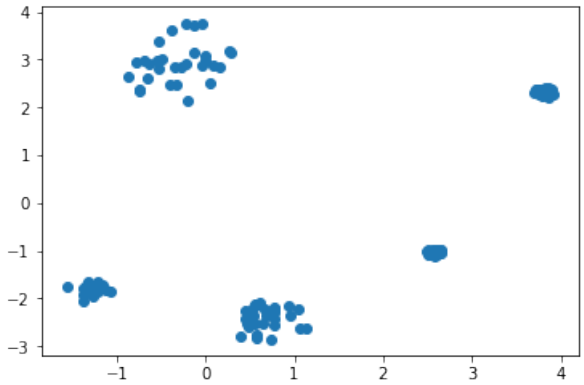


איור 3:

נשים לב כי הפעם קיבלנו פיצול "מלאכותי" של אחד ה־clusters לשניים. האם יש לנו דרך לדעת מהי הכמות "הנכונה" של clusters שאנו מצפים לקבל? האם נוכל לבצע בחירה מושכלת של k? יש שיטות היוריסטיות רבות המנסות לתת לנו k אופטימלי, ולמען האמת, אף אחת מהן לא מתאימה בכל המקרים. עם זאת, בכל זאת נציג את המפורסמת בהן.

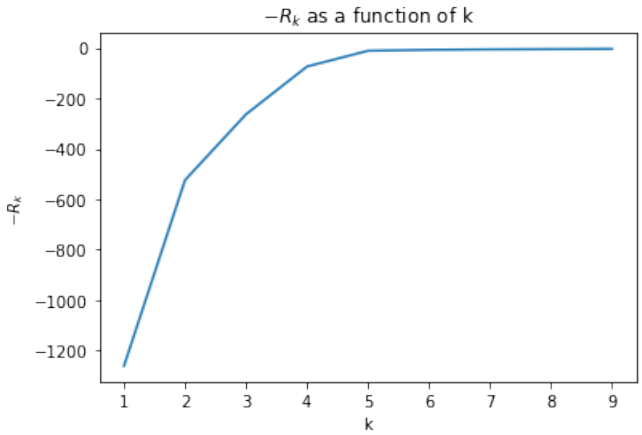
1.1.2 שיטת הברך בגרף

נתבונן כעת בדוגמא מסובכת מעט יותר:



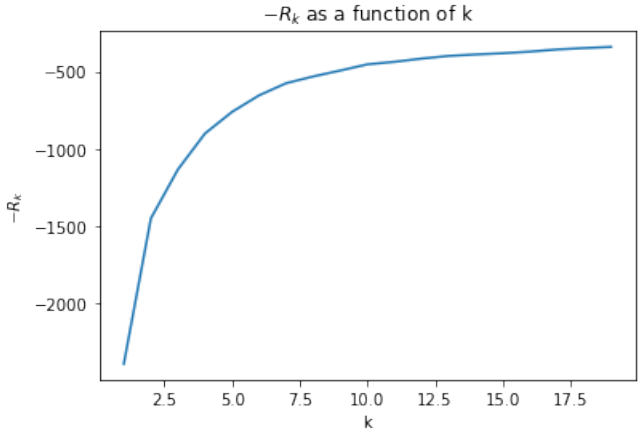
איור 4:

נוכל לחשב, לכל k אפשרי, את R_k. נציג את הגרף של $-R_k$ כתלות ב־k:



איור 5:

נשים לב לתופעת הברך בגרף: סביב $k \sim 4,5$ מופיעה בגרף "ברך", השתנות של השיפוע באופן חד לקראת התמתנותו. כלומר, סביב ערך k זה נעשית הירידה הגדולה ביותר ב־R_k. לכן נוכל לומר כי זהו ה־k המתאים ביותר, זה המתאר בצורה הטובה ביותר את המידע. ל־k גדולים יותר ה־clusters הופכים מנוונים, ואילו ל־k קטנים יותר הם לא מתארים טוב מספיק את המידע, ושינוי קטן ב־k מביא לשינוי גדול ב־R_k.
כפי שטענתי קודם, שיטה זו שימושית, אך פעמים רבות נמצא אותה לא מספקת. במקרים רבים אנו עשויים להיתקל בגרפים מהצורה

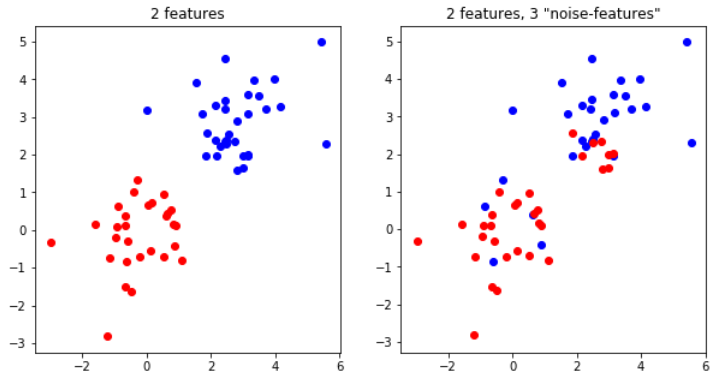


איור 6:

כעת הרבה פחות ברור לנו מה צריך להיות k, אך בכל זאת נוכל להעריך כי $5 \leq k \leq 10$.
נוסיף הערה מעניינת יותר על האלגוריתם: מלבד העובדה כי אנו נאלצים לקבוע את k, יש ל־KMeans חסרונות נוספים. אחד מהם, למשל, הוא היותו מוגדר עבור מרחבים אוקלידיים בלבד. פעמים רבות לא נרצה לעבוד עם מטריקה אוקלידית, בעיקר עקב התנהגות לא נחמדה שלה במימדים גבוהים. במקרים כאלה נעדיף לעבוד עם מטריקות אחרות, כדוגמת מטריקות L_p ל־$p < 1$ ($|v|_{L_p} = \sqrt[p]{\sum v_i^p}$), מטריקת הקורלציה ועוד. אולם לא נוכל להתאים באופן פשוט את KMeans למדוד מרחקים באמצעות מטריקות אלו ולחשב את הבאריסנטר באמצעותן בקלות.
נשתמש באלגוריתם KMeans כדי לדון בבעיה כללית יותר של משימות unsupervised learning.

1.1.3 הגדרת מרחב הפיצ'רים

כשעסקנו ב־supervised learning תיארנו כל נקודה כוקטור במרחב הפיצ'רים. האלגוריתמים שהצגנו היו, במידה רבה, אדישים לאיכות של הפיצ'רים השונים. יתר על כן, עץ החלטה, למשל, יודע לתת "משקל" נמוך יותר לפיצ'רים לא משמעותיים. האלגוריתמים מסוגלים להתעלם מהפרעות בדמות פיצ'רים גרועים בזכות התיוג: הוא מאפשר לנו לקודד אינפורמציה בין הפיצ'רים ובין מטרה מסוימת, ו"להחליט" מי הם הפיצ'רים הטובים.
ב־unsupervised learning אין לנו הפריווילגיה הזו, ולא נוכל לדעת אילו פיצ'רים הם פיצ'רים טובים, ואילו הם "רעש". כיוון שכך ה"רעש" עשוי להשתלט על התפלגות הפיצ'רים, ואנו עשויים לקבל תוצאות מוטות מאוד. למשל, ב־KMeans אנו מחשבים מרחקים בין נקודות. בחישוב של מרחק אוקלידי, אם חלק לא מבוטל מהקואורדינטות הוא רעש, נקבל כי מדידת המרחק מאבדת משמעות. נקבל פגיעה בביצועי האלגוריתם:

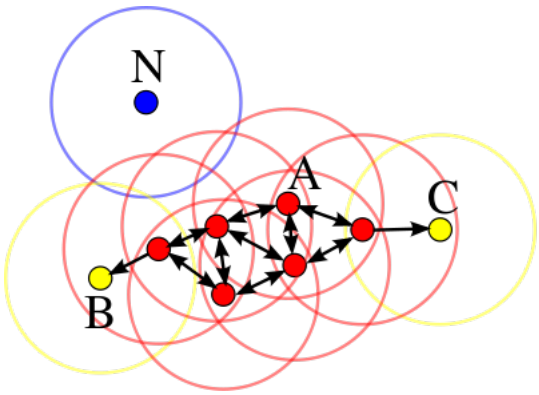


איור 7:

כלומר, נצטרך להיות זהירים בפיצ'רים שלנו ולוודא כי איננו נתונים להטיות לא צפויות. נדגיש כי אחת הבעיות בדוגמא נובעת מכך שכאמור, המטריקה בה משתמש KMeans היא המטריקה האוקלידית, שאינה רובוסטית לרעש בפיצ'רים מסוימים.
זו גם דוגמא טובה מאוד למסר מתחילת השיעור: דברים לא קורים רק כי אנחנו רוצים שהם יקרו. המציאות, פעמים רבות, מסובכת, ונצטרך להתאמץ כדי שדברים "יעבדו באופן חלק". נעבור להתבונן באלגוריתם clustering אחר במהותו, אך נפוץ במידה דומה.

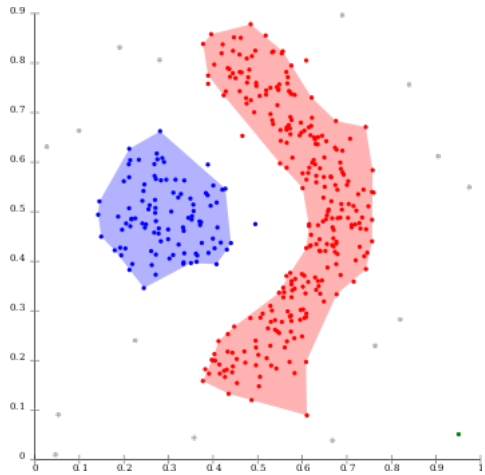
1.1.4 DBSCAN

אלגוריתם זה אינו עושה שימוש כלל במרכזי ה־clusters, ולמעשה יש לו יתרון משמעותי על KMeans: הוא אינו מצריך מאיתנו להעריך את כמות ה־clusters k. עם זאת, כידוע, אין מתנות חינם. נבחר $\varepsilon > 0$ ו־$n \in \mathbb{N}$. נעבור על כל הנקודות. לכל נקודה, נוסיף ל־cluster בו היא נמצאת את כל הנקודות ברדיוס ε ממנה, רק אם יש יותר מ־n נקודות כאלו. אם נקודה אינה נמצאת ב־cluster, נתייחס אליה כאל cluster נפרד. לבסוף, אל כל הנקודות שנשארו בודדות ב־cluster משלהן נתייחס כאל outliers, חריגים, ולא נייחס להן אף cluster.



איור 8:

הפרמטרים ε, n קובעים למעשה קריטריון של צפיפות: ה־clusters שלנו יהיו קבוצות מצפיפות גדולה מ־$\frac{n}{vol(B(\varepsilon))}$, כאשר $B(\varepsilon) = \{x \in \mathbb{R}^n \,|\, |x| < \varepsilon\}$, כדור ברדיוס ε.
ל־DBSCAN יש כמה יתרונות ברורים: ראשית, הוא גמיש מאוד, ומאפשר לנו להתאים כמות משתנה של clusters, בצורות שונות.



איור 9:

אכן, איננו רגישים כלל לצורה המרחבית של ה־clusters, אלא רק לצפיפות בתוכם.
יתרון נוסף של האלגוריתם הוא העובדה שניתן לשנות את המטריקה בה הוא עושה שימוש בקלות רבה. כפי שהזכרנו קודם לכן, ישנה חשיבות רבה לבחירה נכונה של המטריקה, במיוחד במידע המיוצג במרחב פיצ'רים ממימד גבוה.
כמו־כן, DBSCAN נותן לנו מושג ברור של outlier, חריגה. נקבל סף ברור להגדרת אנומליות על המידע, דבר שעשוי להיות שימושי במשימות רבות. נשים לב כי אין לנו מושג ברור שכזה ל־KMeans.
ל־DBSCAN יש חסרון בולט: הוא רגיש מאוד לפרמטרים ε, n, במיוחד לn קטנים. הדבר בא לידי ביטוי במיוחד בהבדל בין $n = 1$ ל־$n = 2$. עבור $n = 1$ כל נקודה היא ב־cluster, תופעה שלאו דווקא נקבל ל־$n = 2$. כדי להפעיל את DBSCAN בהצלחה, מוטב שתהיה לנו הערכה לצפיפות שאנו מצפים לקבל בתוך clusters, או לצפיפות הממוצעת בהתלפגות המידע שלנו. לרוב ניתן לקבל הערכות שכאלו באמצעות ניסויים חוזרים של הפעלת DBSCAN וחישוב גדלים סטטיסטיים והיסטוגרמות על המידע: למשל, נוכל לשאול כיצד מתפלג המשתנה המקרי המחזיר מרחק בין שתי דגימות שרירותיות הנדגמות מתוך ההתפלגות.
בנוסף, נשים לב כי אין לנו דרך "טבעית" לשייך נקודה חדשה, שלא הופיעה בסט האימון, ל־cluster כלשהו שהתקבל באמצעות DBSCAN. זאת בניגוד לאלגוריתם KMeans, בו יש לנו מושג ברור של מרכז ה־cluster, ונקודה חדשה תשוייך מיד ל־cluster שאל המרכז שלו היא הכי קרובה.

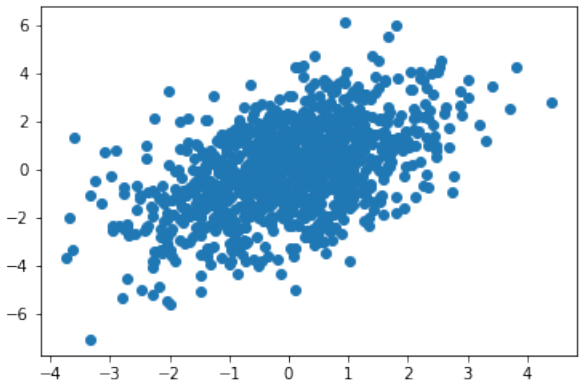
1.2 הורדת מימדים

כזכור, כחלק מתהליך העבודה עם מידע אנו מחלצים פיצ'רים לשימוש המודלים. יש חשיבות לקיום של פיצ'רים שיתארו היטב את המודל, אך מודלים רבים לא יתמודדו היטב עם פיצ'רים

"גרועים". מודלים פשוטים, כדוגמאת רגרסיה ליניארית, וודאי שלא ירוויחו מכך, ומודלים מורכבים יותר כמו עץ ההחלטה שניתחנו בשיעור שעבר עשויים להיפגע, במיוחד כאשר אנו מנסים להזריק אקראיות למודלים שלנו, ועושים שימוש בפיצ'רים "גרועים". בעיה נוספת עשוייה להיווצר כאשר יש לנו כמות רבה של פיצ'רים מנוונים. למשל, אם ננסה לאמן מודל random forest כאשר כל פיצ'ר מופיע כמה פעמים אנו עלולים לבנות עצים עם מעט מאוד פיצ'רים יחודיים, שלא יתרמו לחיזוק ה־ensemble.
לכמות רבה של פיצ'רים יש חסרונות נוספים, הבולט שבהם מתבטא ב"קללת המימדים" - לא נרחיב עליה כאן, אף שאתם יותר ממוזמנים לקרוא על הנושא.
התהליך של בחירת פיצ'רים מבין כלל הפיצ'רים הקיימים נקרא feature selection, ויש מגוון דרכים לעשותו. תהליך אחר שניתן לבצע הוא תהליך של הקטנת מימדים, dimensionality reduction, בו אנו מפעילים טרנספורמציה כלשהי על מרחב הפיצ'רים כדי להקטין אותו, ולהיוותר עם כמות קטנה יותר של פיצ'רים חדשים, יותר "אינפורמטיביים", אף שאולי בעלי משמעות אחרת.

1.2.1 PCA

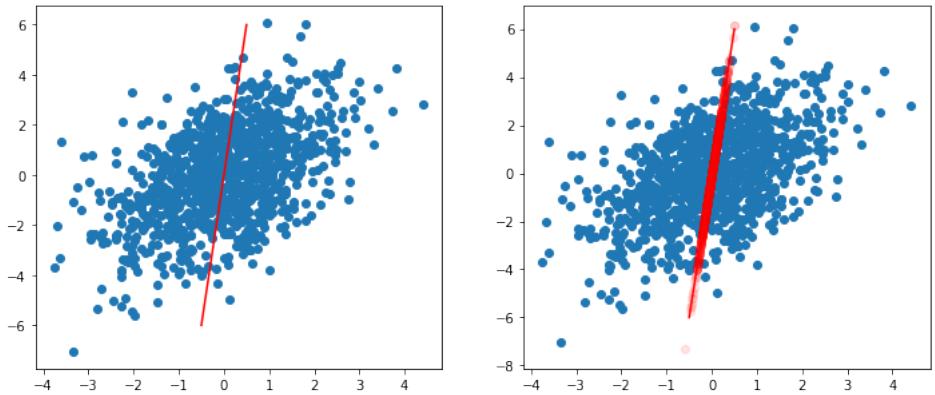
נדון כעת בדרך הנפוצה ביותר להקטנת מימדים. שיטה זו היא unsupervised, כלומר משתמשת בפיצ'רים בלבד, מבלי להתייחס כלל לתיוג. לכן הורדת המימדים בהכרח תיעשה ע"פ פרמטר כלשהו של התפלגות המידע במרחב הפיצ'רים - זו כל האינפורמציה שנותרנו איתה.
נתבונן בנקודות הבאות במישור דו־מימדי הלקוחות מהתפלגות מסוימת:



איור 10:

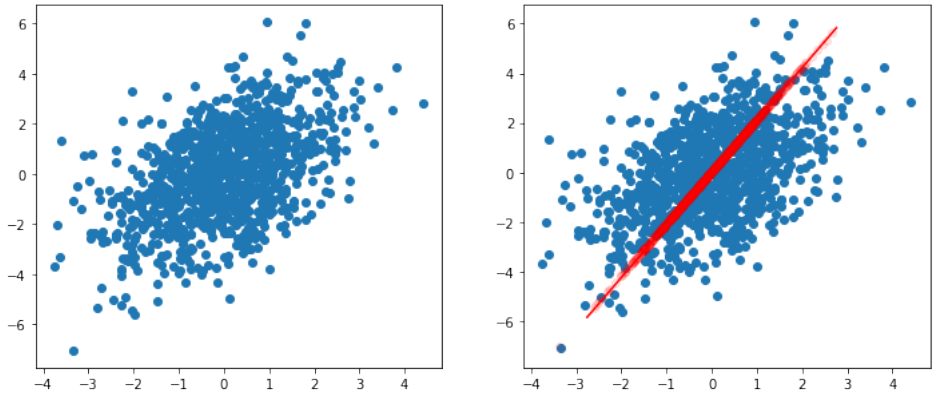
נוכל לחשב את השונות של המידע. כזכור, השונות, $Var(X) = \mathbb{E}((X - \mathbb{E}(X))^2)$, היא מדד לכמה המידע מפוזר סביב הממוצע, לכמה ההתפלגות "מרוחה". אולם השונות היא גם מעין מדד לכמה "אינפורמציה" יש בהתפלגות. אם השונות היא 0 פירוש הדבר הוא שכל הנקודות מרוכזות בממוצע, והפיצ'ר לא יעניין אותנו, כיוון שהוא לא יוסיף לנו אינפורמציה. למשל, בדוגמא שלנו, השונות היא ≈ 2.61. המידע נתון בשני פיצ'רים, ונניח שנרצה לעבור לפיצ'ר יחיד. נוכל לשאול מהי השונות בכל "ציר" אפשרי בנפרד. למשל, נוכל להטיל את המידע על הציר הבא:

איור 11:



הציר החדש שבחרנו הוא פיצ'ר, קומבינציה של הפיצ'רים האחרים עד כה. לכל דוגמא אנו יכולים לבדוק היכן על הציר היא מוטלת. נוכל לשאול, מהי השונות של המידע כאשר אנו מטילים אותו אל הפיצ'ר הזה בלבד. בדוגמא זו התשובה היא ≈ 1.89. כלומר, אנו כעת מתארים את המידע שלנו באמצעות פיצ'ר אחד, במקום שניים, אך "הפסדנו" חלק מהשונות. נוכל לבחור ציר להטיל עליו כך שמקסימום מהשונות תישאר:

איור 12:



כעת השונות המתקבלת בפיצ'ר החדש היא של ≈ 2.09. כלומר, הקטנו את כמות הפיצ'רים אך שמרנו על מקסימום מהשונות.

באלגוריתם PCA נרצה להקטין את כמות הפיצ'רים שלנו, מ־n ל־$m < n$. נבחר את m הפיצ'רים שישמרו על כמה שיותר מהשונות. נעיר כי דרך המימוש הסנדרטית של האלגוריתם היא בכלים של אלגברה ליניארית שטרם למדתם, לכן נתאר היורסיטיקה של מימוש האלגוריתם:

1. עבור m פעמים:

(א) נמצא פיצ'ר חדש, קומבינציה (ליניארית) של כל הפיצ'רים הקיימים, כך שהשונות עליו היא מקסימלית

(ב) נשמור את הפיצ'ר ברשימת הפיצ'רים החדשים

(ג) נטיל את כל הפיצ'רים למרחב הניצב לפיצ'ר שבחרנו

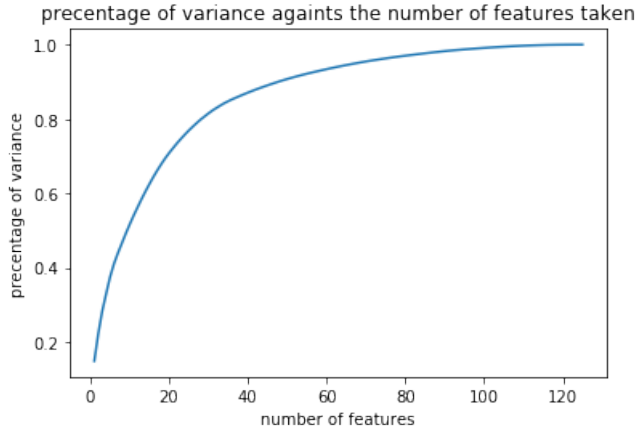
נתאר את תהליך ההטלה בפורמליזם של אלגברה ליניארית: כל דוגמא אנו מתארים כוקטור במרחב n מימדי, הנפרש ע"י הפיצ'רים - הוקטור $e_i = (0, ..., 1, ..., 0)$ מקבל 1 בפיצ'ר ה־i ו־0 באחרים, ואוסף הוקטורים הללו פורש את המרחב שלנו. כל דוגמא נוכל לכתוב כ־$a_1 \cdot e_1 + ... + a_n \cdot e_n$. נניח שמצאנו וקטור חדש, שהטלה עליו ממקסמת את השונות. כלומר, מצאנו וקטור $v = v_1 \cdot e_1 + ... + v_n \cdot e_n$. כעת אנו מעוניינים להטיל את כל הוקטורים למרחב הניצב לוקטור v, כלומר

$$u \mapsto u - < u, \frac{v}{|v|} > \cdot v$$

נקבל בסוף התהליך את m הפיצ'רים ששומרים על כמה שיותר מהשונות, כלומר "מהאינפורמציה" בפיצ'רים המקוריים.

1.2.2 בחירת המימד

בשימוש ב־PCA נוכל לקבוע לכמה פיצ'רים אנו רוצים לבצע הורדת מימדים ע"י כך שנצייר גרף של אחוז השונות איתה נותרנו כתלות בכמות הפיצ'רים שבחרנו לקחת:



איור 13:

כעת נוכל להתבונן בגרף ולבחור את כמות הפיצ'רים הרצויה. גם כאן נוכל להפעיל שיטות של חיפוש הברך בגרף. אולם בגרף זה יש לנו משמעות יותר לציר ה־y, ונוכל למשל לבחור בכמות פיצ'רים שתשמור 80% מהשונות.

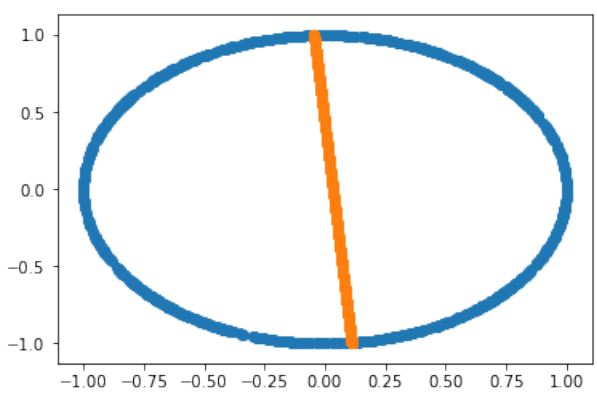
1.2.3 נרמול המידע

כאמור, PCA הוא אלגוריתם ממשפחת האלגוריתמים ה־unsupervised. ככזה הוא לא מניח דבר על הקלט שלו, וכל הנחה שהייתם רוצים שיקח, עליכם לתת לו מראש. לכן ישנה

חשיבות רבה מאוד לנרמול של המידע: במידע של שני פיצ'רים, כאשר פיצ'ר אחד מקבל ערכים בתחום $[-1,1]$ והשני בתחום $[-100,100]$, מובן כי חלק גדול מהשונות ימצא בפיצ'ר השני. זאת כיוון שהשונות לא מניחה נרמול מסוים, ומחשבת "מרחק ממוצע מהממוצע". אם הפיצ'ר שלנו "מנופח", המרחק מהממוצע "יתנפח" גם כן. לכן חשוב לנרמל את הפיצ'רים של המידע כך שיהיו באותה הסקאלה בטרם אנו משתמשים ב־PCA. למעשה, לקח זה נכון, כמובן, גם לשיטות ה־clustering שדנו בהן קודם לכן, ונכון באופן כללי לכל עבודה שהיא ב־unsupervised learning (כמו גם, למעשה, לעבודות ב־supervised learning, גם אם בעוצמה פחותה).

1.2.4 ליניאריות המודל

נשים לב כי PCA הוא מודל הקטנת מימדים ליניארי: כלומר, כל פיצ'ר חדש הוא צירוף ליניארי של הפיצ'רים המקוריים. הדבר הופך את המודל לקל מאוד לחישוב, אך למוגבל ביותר, כיוון שישנן הורדות מימד "פשוטות" שהוא לא יוכל לקבל - הוא יבצע הורדות מימד ליניאריות בלבד. דוגמא טובה לכך ניתן לראות במידע המתפלג על הספירה S^1:



איור 14:

ברור כי הורדת מימדים מוצלחת הייתה יכולה להיות $(x,y) \mapsto (\theta)$, כלומר $(x,y) \mapsto (tan^{-1}\frac{y}{x})$. אולם זו, כאמור, אינה פונקציה ליניארית של x,y, כך שלא נוכל לקבל אותה ב־PCA. PCA הוא אלגוריתם חזק ומאוד שימושי, אך יש להכיר בחולשות שלו.