**תרגול 4**

חזרה על סטטיסטיקה

שונות- הפיזור של ההתפלגות, כמה התצפיות קורבות לממוצע

התפלגות נורמלית- מורכבת ממוצע ושונות

רגרסיה לינארית- משתמשים במשוואה של קו ישר, על מנת למדל את הקשר בין x לy.

מסמנים נקודה בyi, נסמן את המרחק של כל הנקודות yi לקו הממוצע של הנקודות yi .

ניקח את סכום המרחקים(השאריות-residual) , ככל שסכום ריבועי השאריות קטן יותר כך הפונקציה מתארת את הנתונים בצורה טובה יותר.

שלבים של למידת מכונה

1. איסוף נתונים
2. בדיקת המידע וויזואליזציה שלו
3. עיבוד מוקדם של המידע
4. בניית מודל
5. הערכה של המודל

סוגי למידת מכונה

Supervised learning- משהו שמנסים לחזות (מה המחיר של בית, האם זה כלב או חתול), יש משתנה-label שמנסים לחזות אותו, המשתנים שאנו משתמשים בהם כדי לחזות את המשתנה הם features, משמש לחיזוי

לדוגמה:

מקבלים נתונים של תפוחים וקאפקייקס, מכניסים את הנתונים למודל של למידת מכונה, המודל לומד מה זה תפוח ומזה קאפקייקס, לאחר מכן משתמשים בנתונים שלא היו באימון ומודל מוצלח יצליח להבין האם הנתונים מהפרט שהתקבל הוא תפוח או קאפקייק.

רגרסיה- מנסים לחזות משתנה מספרי

סיווג- מנסים לחזות חלוקה לקטגוריות

Unsupervised learning- מנסים להבין דברים על הנתונים, לדוגמה clustering- חלוקה של הנתונים לקבוצה, density estimation- לחזות איך הנתונים מתפלגים, צמצום ממדים.

לדוגמה:

מקבלים נתונים של תפוחים וקאפייקס, מכניסים לאימון על מודל של למידת מכונה ומגדירים שיחלק את הנתונים ל2 קבוצות. אין labels, הוא מחלק את הנתונים בצורה שנראית הגיונית, לא מנסים לחזות משהו, משמש לאנליזה.

Reinforcement learning- למידה מטעויות

על מנת להעריך כמה מוצלח המודל, מחלקים את המודל לtraining data ו test data -בדרך כלל חלוקה של 80% אימון 20% מבחן

שיטה נוספת היא חלוקה נוספת לvalidation data.

היפר-פרמטרים: פרמטרים שמשתמשים בהם כדי להגדיר את האלגוריתם, לדוגמה להחליט את עומק עץ ההחלטה (כמה רמות), על מנת לשפר את הביצועים של המודל.

קרוס ולידציה k fold- בונים 5 עצי החלטה שונים באמצעות שימוש בחלקים שונים מהנתונים על מנת להעריך כמה טובים הביצועים של כל סוג עץ.

במטריצה כל שורה זה עץ, המלבנים הכחולים בכל שורה זו הולדיציה- ה20% מהנתונים שהם המבחן של העץ.

קרוס ולידציה של leave one out, כל פעם להישאר חלק אחר מהנתונים מחוץ לאימון ולבסוף לבחון עליו.

עץ קלסיקפציה- לחזות לאיזו מחלקה שייכת הרשומה, בדוגמה: האם בן אדם אוהב סרטים או לא.

בודקים על פי כל תכונה בנפרד איך מתחלקים הרשומות

מחשבים את הgini עבור כל מחלקה שקיבלנו בכל חלוקה

טווח ערכי ג'יני: 0-1

A picture containing chart

Description automatically generatedText, letter

Description automatically generated

Chart

Description automatically generated

עבור כל חלוקה מחשבים בכל node כמה התקבלו מכל קבוצה של הlabel.

Total Gini- סך כל התצפיות בצד אחד חלקי סך כל התצפיות \* gini של אותו צד (סוכמים עבור כל Node את התוצאות).

**התמודדות בעצי החלטה עם משתנה נומרי:** מבצעים מיון בסדר עולה של הרשומות על פי התכונה, מחשבים את הממוצע עבור כל 2 תצפיות עוקבות, לאחר מכן מחשבים את רמת הטהורות באמצעות חישוב gini עבור כל ממוצע( נחלק ל2 קבוצות- תצפיות שקטנות שוות לממוצע ותצפיות שגדולות מהממוצע). בוחרים את הממוצע עם הערך gini הכי נמוך ומשתמשים בו לחלוקה על פי התכונה

עץ רגרסיה

Text, letter

Description automatically generated

Y גג הוא הממוצע של התצפיות לתכונה זו

Graphical user interface, text, application

Description automatically generated

Chart, scatter chart

Description automatically generated

החלוקה הראשונה שתמיד הכי עדיפה היא חלוקה שמאפשרת לקבל קבוצה אחת טהורה- שכל הנתונים בקבוצה זו שייכים לאותה מחלקה.

Pruning

הורדה של עלים על מנת להימנע מoverfitting.

עבור עצי רגרסיה:

1. נחשב את סכום ה SRR עבור עץ באמצעות סכימה של ערכי הSSR של כל העלים
2. מוסיפים קנס לסכום הSRR על פי רמת המורכבות של העץ
3. בוחרים את העץ עם הסכום SRR הכי נמוך

Graphical user interface, text, application

Description automatically generated

היפר פרמטרים עצי החלטה

Graphical user interface, text, application, email

Description automatically generated

**Bootstrap** a statistical procedure that resamples a single dataset to create many simulated samples.

לכל תצפית יש הסתברות שווה להיכלל בתוך הdataset החדש

הגודל של הdataset החדש יהיה בגודל של המקורי

תצפית יכולה להיבחר כמה פעמים לdataset החדש

**Bagging-** יוצרים n עצים מתוך datasets שונים של אותו data. לאחר מכן משתמשים בעצים שהתקבלו ביחד על מנת לחזות את הסיווג של הנתונים, על פי הבחירה של רוב העצים או הממוצע (אם זה משתנה מספרי).

**Random Forest**

יוצר n עצים במקום עץ אחד, אך לא לוקחים datasets מתוך הdata, אלא גם לוקחים רק חלק מהfeatures בכל שלב חלוקה- על מנת לבחור באיזה feature להשתמש בכל חלוקה בעץ, הוא בוחר כל פעם feature אחד מתוך רק חלק מהfeatures.

Graphical user interface, text, application, email

Description automatically generated

ניתן להשתמש בתצפיות שלא נכללו בdataset לבניית עץ ספציפי זה, על מנת לבדוק את רמת הדיוק של העץ-out of bag accuracy- We can measure how accurate our random forest is by the proportion of out-of-bag samples that were correctly classified.

Out-of-bag Error: the proportion of out-of-bag samples that were incorrectly classified.

Text

Description automatically generated

**AdaBoost**

תחילה כל התצפיות מקבלות את אותו משקל -1 חלקי מספר התצפיות

מחשבים את הGini עבור כל סטאמפ כמו בעץ החלטה (מחלקים את הנתונים על פי כל תכונה לפי הממוצע שמחלק הכי טוב).

Diagram

Description automatically generated

עבור כל סטאמפ מחשבים כמה הוא טעה

מספר הפעמים שהוא טעה חלקי מספר התצפיות=טעות כוללת

Diagram

Description automatically generated

מחשבים מה המשקל של הסטאמפ

כאשר המשקל הכי גבוה שיכול להיות לסטאמפ הוא 1

Diagram

Description automatically generated

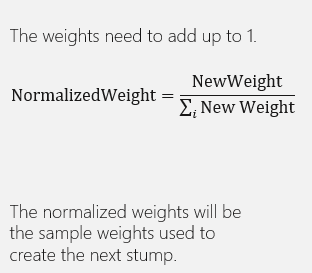
על מנת לבחור מה הסטאמפ הבא שנבדוק נבצע את התהליך הבא:

רוצים לתת חשיבות יותר גדולה לתצפיות שסווגו לא נכון, כדי שהסטאמפ הבא שיבדוק את התצפית יסווג אותה נכון

Table

Description automatically generated

כיוון שאנחנו רוצים שסך כל המשקלים של התצפיות יהיה שווה ל1 אז מנרמלים את משקלי התצפיות באמצעות חלוקת כל משקל של תצפית בסך כל משקלי התצפיות



Graphical user interface, text, application

Description automatically generated

שיטה 2

* עבור כל משקל של תצפית מגדירים טווח, כאשר הטווח של כל תצפית הוא המשקל של התצפית הקודמת עד סכום שמקלים של התצפיות עד התצפית הנוכחית
* בוחרים מספר רנדומלי בין 0 ל1 ובוחרים את התצפית שלטווח שלה המספר נכנס
* מוסיפים את התצפית שנבחרה לdataset חדש
* חוזרים על שלבים 2 ו3 n פעמים, כאשר n הוא מספר התצפיות בdataset
* כיוון שלתצפיות עם משקל גדול יותר יהיה טווח גדול יותר, יש יותר סיכוי שהן יבחרו, כי יש יותר סיכוי שהמספר הרנדומלי יהיה בטווח שלהן
* כך שבdataset החדש שנבנה השורות שסווגו לא נכון, יופיעו יותר פעמים ולכן החשיבות שלהן תהיה יותר גדולה, כיוון שהיא תשפיע יותר על הgini.
* מבצעים את תהליך שוב עם הdataset החדש, בוחרים את התנאי עם הפיצול הכי טוב, ומבצעים את הפיצול על הdataset החדש.
* אחרי שבנינו את כל הסטאמפס אז מחשבים עבור כל החלטה את סכום המשקלים של הסטאמפס שבחרו בהחלטה זו. ההחלטה עם המשקל הכי גבוה היא ההחלטה שנבחרת**.**

אם זה Regression AdaBoost אז מחשבים את הממוצע הממושקל שמתקבל מהתוצאות של כל הסטאמפים.

**היפר-פרמטרים:**

אפשר לקבוע כמה סטאמפים רוצים- הדיפולט הוא 50(n\_estimators)

בדיפולט adaboost בונה stumps- עצים בעומק 1 אפשר גם לקבוע עומק אחר (base\_estiamtor)

**Gradient Boosting**

**Regression**

1. מחשבים את הממוצע של כל ערכי הlabel (מה שמנסים לחזות)
2. מחשבים את השאריות – ההפרשים בין הערך האמיתי של הלייבל בכל תצפית לממוצע שחושב
3. בונים עץ רגרסיה ומנסים לחזות את השאריות עבור כל תצפית

לוקחים תצפית מהtrain ומנסים לעשות חיזוי: לוקחים את החיזוי הראשוני( הממוצע של הלייבלס) ומחברים עםlearning rate כפול החיזוי של השארית מהעץ  
זה יהיה החיזוי החדש. אם יש כמה תצפיות בכל עלה אז מחשבים את הממוצע שלהם.

הlearning rate הוא היפר-פרמטר, בdefault הוא 0.1

1. מחשבים שאריות חדשות: מחסירים את החיזוי החדש מהערך המקורי
2. חוזרים על שלבים 2-4 עד שמספר העצים הנדרש מתקבל או שהשאריות לא גורמות להבדל משמעותי

**היפר פרמטרים:**

Graphical user interface, text, application, email

Description automatically generated

**KNN**

Chart, scatter chart

Description automatically generated

Text, letter

Description automatically generated

ערכים מאוד קטנים של K מובילים לoverfitting

ערכים מאוד גדולים של K מובילים לbias מאוד גדול

הערך הכי טוב לK תלוי בנתונים, מנסים ובודקים מה נותן את החלוקה הטובה ביותר

על מנת לחלק featuresקטגורליים נמיר אותם ל0 ו1

אם משנים את המדגם מהנתונים ועבור כל מדגם נקבל תוצאה שונה- שונות גדולה –overfitting ,עבור מדגמים שונים יותר regressor מאוד שונה.

זה מצב לא טוב כי אנחנו רוצים להשתמש באותו regressor למדגמים שונים.

כאשר נשתמש ביותר מדי קלאסטרים אז נקבל bias גדול אבל שונות נמוכה, הregressor לא משתנה ממדגמים שונים אך עושה טעויות בסיווג לקלאסטרים

רוצים להגיע למצב מאוזן בין variance לbias

**קלסיפיקציה-** מסתכלים מהו הלייבל הכי נפוץ בין כל K השכנים ובוחרים בו

**רגרסיה-** מחשבים את הממוצע של הלייבל של כל K השכנים ובוחרים בו

**היפר-פרמטרים:**

Graphical user interface, text, application, email

Description automatically generated

Weight- עושה ממוצע משוקלל לפי המרחק של תצפית מהתצפית החדשה

Metric- באיזה סוג מרחק משתמשים, דיפולט זה אוקלידי

**KNN לעיבוד תמונה:**

ניתן להשתמש בKNN לעיבוד תמונה, באמצעות שימוש בפונקציה reshape ניתן להפוך כל פיקסל בתמונה לfeature ואז לבדוק את ערכו.

במקרה של תמונות שחור לבן ניתן לבדוק אם הפיקסל שחור ואז מקבל את הערך 1 או לבן ואז מקבל את הערך 0.

כך באמצעות KNN נקבל קלאסטרים לפי מה שחור ומה לבן

אם במקרים הבאים אותם פיקסלים יהיו לבנים פחות או יותר אז נוכל לזהות איזו ספרה מופיעה בתמונה

**KD Trees**

בונים L עצים כאשר בוחרים feature באופן רנדומלי ומחלקים לפי התצפית עם הערך של החציון של תכונה זו. מבצעים את התהליך מספר פעמים כך שכל צומת בעץ היא החציון של החלוקה לפי תכונה זו.

כאשר נותנים תצפית חדשה, מריצים אותה בכל העצים. כאשר מגיעים לעלה הרוולנטי בכל העצים- בוחרים את השכן הכי קרוב בכל עלה.

כאשר יש לנו את המרחקים הכי קרובים של העלים מכל העצים, בוחרים בעלה עם המרחק הכי קטן.

Chart, diagram

Description automatically generated

**LSH-Locality Sensitive Hashing**

יוצרים L hash tables , חותכים את הdata לקבוצות שונות באופן רנדומלי

כל התצפיות בחתך מסוים נכנסים ביחד לאותו hash code

בהינתן תצפית חדשה מחפשים באמצעות טבל הhash את השכן הקרוב ביותר

Chart, scatter chart

Description automatically generated

**היפר פרמטרים:**

Graphical user interface, text, application, email

Description automatically generated

**Clustering**

**K-Means**

* K-Means יכול ליצור קלאסטרים רק בצורת עיגול

**Graphical user interface

Description automatically generated with medium confidence**

**Elbow Method**

באמצעות שיטה זו בוחרים מה הK קלאסטרס שהכי כדאי להשתמש בהם.

באמצעות חישוב של sum of squared errors מוצאים את הנקודה שבה אין הבדל משמעותי בדיוק אם מוסיפים קלאסטר נוסף.

Sum of squared errors- מחשבים את הממוצע של הנתונים, לאחר מכן את המרחק של כל תצפית מהממוצע, מעלים בריבוע את המרחק של כל תצפית מהממוצע ואז סכומים את התוצאות

Chart, line chart

Description automatically generated

**היפר פרמטרים**

**Graphical user interface, text, application, email

Description automatically generated**

GMM-Gaussian Mixture Models

יוצר קלאסטרים שהם לא רק בצורת עיגול

תחילה בוחרים רנדומלית נקודות שהם יהיו מרכזי הקלאסטרים.

הנקודות האלו הופכות להיות ערכי הממוצע של התפלגות נורמלית

מחשבים עבור כל נקודה את ההסתברות שלה להיות חלק מהקלאסטר באמצעות log likelihood

מעדכנים את ערכי הפרמטרים של ההתפלגות על מנת להגדיל את הסיכוי שהנקודות יהיו חלק מהקלאסטר. לאחר מכן מחשבים שוב מה הסיכוי של כל נקודה להיות חלק מהקלאסטר וחוזרים על התהליך עד שאין כמעט שינוי במעבר בין הלקאסטרים.

**Text

Description automatically generated**

פאי k הוא אחוז התצפיות ששייך לקאסטר מספר k מסך כל התצפיות

**Text

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generatedChart

Description automatically generated**

**Agglomerative Clusterting**

Hierarchical Clustering

1. מגדירים כל תצפית להיות קלאסטר בפני עצמה
2. מבצעים השוואה של כל 2 תצפיות ביחד ובונים מטריצה אשר מבטאת כמה כל 2 תצפיות דומות, בדרך כלל מתבצע באמצעות מרחק אוקלידי, שומרים את מטריצת הדמיון
3. יוצרים קלאסטר ביחד של 2 התצפיות הכי דומות
4. חוזרים על שלבים 2-3 עד שכל התצפיות נמצאות בקלאסטר אחד או שמגיעים לכמות הרצויה של קלאסטרים

שיטות קישור- מגידרות איך מחשבים את המרחק בין קלאסטרים

Complete linkage- מחשבים את המרחק המקסימלי בין 2 הנקודות הכי רחוקות בכל קלאסטר

Single linkage- לוקחים את המרחק הכי קטן (בין 2 הנקודות הכי קרובות מכל קלאסטר)

Average linkage- מחשבים את המרחק בין הממוצע של כל קלאסטר

Graphical user interface, application

Description automatically generated

המטריצה היא מטריצת מרחקים

במקרה זה המרחק הכי קטן הוא בין תצפית 1 ל2

הופכים את תצפיות 1,2 לקלאסטר

מחפשים את המרחק הכי גדול בין הקלאסטר שנוצר לבין כל אחת מהתצפיות ויוצרים מזה מטריצה

מוצאים את המרחק הכי קטן כאשר הקלאסטר עכשיו נחשב כתצפית אחת, במקרה זה תצפיות 3,4

לכן 3,4 יהפכו לקלאסטר

מחשבים שוב את המרחק המקסימלי, בין הקלאסטרים, במקרה זה 0.8 ויוצרים מטריצה חדשה

לבסוף יוצרים קלאסטר של כל התצפיות יחד

מצד שמאל ניתן לראות את הדנדוגרמה שהתקבלה מתהליך זה, באמצעותה ניתן להבין כמה קלאסטרים כדאי לעשות: בודקים מה ההפרש הכי גדול בין המרחקים המקסימליים בדנדוגרמה בין מרחקים עוקבים’ , נבחר בקו האנכי הכי ארוך שמסמל את ההפרש הכי גדול ונחתוך אותו עם קו אופקי לפני החלוקה לקאסטרים הבאה. מספר החיתוכים של הקו האופקי עם קווים אנכיים יהיה מספר הקלאסטרים שכדאי לעשות (שאלגוריתם יחזיר).

Diagram

Description automatically generated with medium confidence

**היפר-פרמטרים:**

**Text, letter

Description automatically generated**

n\_clusters-מספר הקלאסטרים שהוא יגיע אליו

Affinity- איך לחשב את המרחק

Linkage-הדיפולט הוא word-מיזעור של השונות בתוך הקלאסטר

Distance\_threshold- ניתן להגדיר מרחק שמעליו לא מבצעים חיבור לקלאסטר

**\***לכל קבוצת נתונים מתאים Linkage אחר שיחלק את הנתונים בצורה הטובה ביותר, נדרש לנסות את הסוגים השונים.

Divesive Clustering

1. תחילה מתייחסים לכל התצפיות כקלאסטר אחד
2. יוצרים מטריצת קירבה(affinty) באמצעות המשוואה המופיעה אשר מחשבת את המרחק בין כל נקודה

A picture containing diagram

Description automatically generated

1. מחשבים את הסכום של המרחקים עבור כל קלאסטר אפשרי, כאשר מוסיפים תצפית לקלאסטרים הקיימים: במונה סוכמים את המרחק בין כל נקודה לשאר הנקודות ובמכנה מחשבים את מספר התצפיות בקלאסטר ומעלים בחזקת אלפא- ערך בין 0 ל2 שהוא היפר-פרמטר
2. בוחרים את הקלאסטר עם המרחק הכי גדול ומחלקים את המטריצה מחדש לפי הקלאסטר שיצרנו.
3. חוזרים על שלבים 2-4 עד עד שמגיעים למספר הקלאסטרים המבוקש או שכל קלאסטר מורכב מתצפית אחת

Graphical user interface

Description automatically generated with medium confidence

נניח כי יש 3 תצפיות

1. יוצרים מטריצת אפיניטי, כל תצפית עם עצמה שווה 1

2. מחשבים את הסכום לפי הנוסחה לכל פיצול אפשרי

אם תצפית 1 תהיה קלאסטר וכל השאר קלאסטר שני אז נקבל על פי המשוואה, כאשר אלפא שווה לחצי נקבל . כך גם עבור כל שאר התצפיות כשהן לבד.

עבור הקלאסטר 1,2 וכל השאר קלאסטר נקבל 1 עם 1 (1), 2 עם 2 (1) ו1 עם 2, שווה 0.4- זה הסכום במונה חלקי גודל הקלאסטר (2) בחזקת חצי=1.69

נבצע את זה לכל קלאסטר של 2 תצפיות.

3. נחלק לפי הסכום המקסימלי-1,3

לכן 1 ו3 הם קלאסטר ביחד ו2 בנפרד

4. חוזרים על תהליך זה עד שכולם בתוך קלאסטר 1

מסתכלים על הדנדוגרמה מלמעלה- קודם מפצלים את 2 ואז את 1 ו3

ניתן באמת לראות מההתחלה במטריצת אפיניטי ש1 ו3 הכי דומים (הערך הכי גבוה)

זה קורה בזמן אקספפונציאלי- מאוד לא יעיל, לכן יש כל מיני דרכים לשפר את הזמן ריצה.

אלתגוריתם שפחות בשימוש כי יש כאלה שעובדים יותר טוב.

Mean Shift

1. מגדירים שכל תצפית היא מרכז קלאסטר ובוחרים את הערך של h (רדיוס או bandwidth)
2. עושים קלאסטרינג ביחד של כל התצפיות שהן ברדיוס אחת של השנייה
3. מחשבים מחדש את המרכז של כל קלאסטר- שהוא הממוצע של כל התצפיות בקלאסטר

חוזרים על שלב 2-3 עד שאין שינוי בקלאסטרים

ערכים קטנים יותר של h יוצרים יותר קלאסטרים קטנים יותר

ערכים גדולים יותר של h יוצרים פחות קלאסטרים יותר גדולים

צריך למצוא את הh שמחלק את הנתונים בצורה הכי טובה

**היפר-פרמטרים:**

Graphical user interface, text, application, email

Description automatically generated

bandwidth- הערך לפיו מחליטים מה גודל הרדיוס

Max-iter- מספר האיטרציות המקסמילי שהאלגוריתם יעשה, כדי למנוע ממנו לרוץ עד אינסוף במקרים מסוימים

KDE-Kernel Density Estimators

אמדים לצפיפות של נתונים.לא משתמשים בפרמטר מסוים כדי להגדיר את צפיפות הנתונים (לא כמו התפלגות נורמלית לדגומה שמשתמשים בתוחלת ושונות)

אומדים את הצפיפות של ההתפלגות, הרבה פעמים משתמשים כדי להסתכל ויזואלית על המידע, יכול לשמש לקלאסטרינג משמש כשיטה מקבילה אבל באופן רציף להיסטוגרמה.

Text, letter

Description automatically generated

איקס הוא משתנה מקרי שיש לו צפיפות f , הh גם פה הוא bandwith, סוג של רדיוס. בודקים האם הנקודה X היא בין x-h וx+h

זאת אומרת מה ההסתברות שX הוא בין x-h לx+h כאשר h שואף ל0

נגדיר אינדיקטור שהוא שווה ל1 אם התצפית x שייכת לטווח ו0 אם לא

נעשה סכום של כל האחדים והאפסים ונחלק ב2 כפול h כפול מספר התצפיות – נקבל את ההסתברות שx שייך לתוך הטווח הזה.

זאת אומרת שנקבל את אילו תצפיות נכנסות לטווח x-h,x+h

Text, letter

Description automatically generated

הפונקציה W היא פונקציית משקל שאם היא מקבלת x שהערך מוחלט שלו קטן מ1 אז התוצאה היא 0.5 אחרת 0

ניתן לכתוב גם את המשוואה האחרונה מהשקופית הקודמת כמו המשוואה השנייה בשקופית זו

נגדיר את W שמקבלת , זאת אומרת שאם נמצא בטווח שבין x-h לx אז אם נחשב נקבל מספר קטן מh

לכן נקבל בתור ערך לW מרחק שהוא קטן h חלקי h לכן נקבל מW 0.5

אם יהיה לנו שהוא מספר קטן מx-h אז הערך שW יהיה מרחק גדול מh חלקי h ולכן נקבל מW 0

לכן אם נמצא בטווח x-h,x+h אז W יהיה שווה ל0.5, אחרת הוא יהיה שווה 0

כפי שהסברנו המשוואה השנייה היא לא רציפה אז ניתן להגדיר kernel estimator –המשוואה השלישית שבה הפונקציה K היא פונקציית kernel שיכולה להשתנות וh הוא הbandwidth

A picture containing text

Description automatically generatedChart, line chart

Description automatically generated

הkernel estimator אומד את הצפיפות לפי sum of bumps. כל פס אדום על ציר x הוא תצפית. עבור כל נקודה מגדירים kernel גאוסיני, כאשר כל נקודה היא התוחלת ויש שונות קבועה לכולן.

Chart, histogram

Description automatically generated

לכל הטווח, במקרה זה 0-4 עושה סכום של הbumps –ערכי הf(x) כאשר x הוא הקו שהיא תצפית והf(x) הוא הערך של ההתפלגות הגאוסיינית באותו ערך x. זה יוצר את ההתפלגות שרואים בקו האדום (הסכום של ערכי ההתפלגויות בנקודה)ניתן להגדיר kernel functions בצורות שונות אבל יעבוד על אותו העיקרון.

ככל שמגדילים את הbandwidth אז פונקציית הקרנל הופכת לחלקה יותר, כחלק שמקטינים אז הפונקציה עם יותר bumps

לא חייבים להגדיר bandwidth ידינית, אפשר לכתוב bw=“silverman” ואז זה מחשב את הbandwidth המתאים

**SVM Classifier**

**Graphical user interface, text, application

Description automatically generated**

מחפשים את קו האמצע בין 2 התצפיות הכי קרובות מכל מחלקה((hyperplane ויוצרים support vectors, אלו קוים אשר בתוכם יכולים להיכנס outlayers על מנת למנוע overfitting (soft margins). משתמשים בcross validation על מנת להחליט אילו soft margins הם הכי טובים- עם כמה שפחות variance וbias (כמה שפחות מיסקלסיפקציות אבל בלי overfitting).

עוברים לממד גובה יותר על מנת לאפשר חלוקה של הנתונים ל2 מחלקות. בחורים את הממד שהכי טוב להעלות אליו את הנתונים באמצעות cross validation.

The Kernel Trick: kernel functions only calculate the relationship between every pair of points as if they are in a higher dimension – they don’t actually do the transformation.

The kernel trick reduces the amount of computation required for SVM.

Chart

Description automatically generated

Chart

Description automatically generated

Chart

Description automatically generated with medium confidence

Text

Description automatically generatedחלוקה ליותר מ2 קלאסים:

Text

Description automatically generated

**PCA**

אלגוריתם להורדת ממדים- יותר מדי פיצ'רים מגדילים את הזמן ריצה וגם קשה לייצג את המידע מבחינה ויזואלית על מנת להסיק ממנו דברים.משמש למציאת קלאסטרים, משמש למצוא תצפיות חריגות.

רוצים למצוא את z אשר מחזיק כמה שיותר מידע מהדטא המקורי x עם כמה שפחות פיצ'רים

עדיין יש n תצפיות אבל j פיצ'רים אבל לא k כאשר j<k

1. בשלב הראשון מחשבים את מטריצת הco variance של הנתונים X

האלכסון הראשי יהיה השונות של כל פיצ'ר והשאר יהיו השונות המשותפת בין כל שני פיצ'רים לפי העמודה והשורה.השונות המשותפת של x1 עם x2 היא כמו השונות המשותפת של x2 עם x1

גודל המטריצה הוא KxK מספר הפיצרים\* מספר הפיצרים

2. לאחר מכן מבצעים פירוק עצמי של המטריצה- מקרה פרטי של SVD(Singular Value Decompostipm) שמוציא לנו את A- המטריצה של הוקטורים העצמיים של מטריצת הco variance.

3. לוקחים את המטריצה המקורית של הנתונים X ומכפילים אותה במטריצה של הוקטורים העצמיים A ונקבל את מטריצה Z שהיא המטריצה עם הפיצ'ירים המצומצמים.

העמודה הראשונה בZ תהיה הפיצ'ר שתופס הכי הרבה שונות מהנתונים, הטור לאחר מכן שתופס הכי הרבה שונות מתחתיו וכו'..

הכפלה של מטריצת הנתונים בוקטור העצמי הראשון תיתן את הטור הראשון =

כך הלאה לPC האחרים

ימשיכו להיווצר PCs עד שנקבל 90% מהנתונים מבחינת השונות (זה הדיפולט ניתן גם לשנות בהרצת האלגוריתם).

Diagram

Description automatically generated

הוקטורים מראים לנו אילו ערכים השפיעו על יצירת הקלאסטרים

לדוגמה ככל שsepal width משפיע יותר על התצפית, כך היא תלך יותר לכיוון הוקטור של sepal width, לכן התצפיות בקלאסטר הכחול הן בעלות sepal width גדול- ערך יותר גדול של התכונה= יותר קרוב לוקטור של התכונה והולך לכיוון שלו.

כאשר מציגים ויזואלית ניתן להשתמש רק ב2-3 PCs

אם מנסים לעשות חיזוי ניתן להשתמש ביותר

בדוגמה ציר x הוא PC1 וציר y הוא PC2. האחוזים הם כמה שונות כל PC תופס.

על מנת ליצור וקטור עצמי אשר תופס כמה ממדים של הנתונים מחשבים את סכום המרחקים של הטלי התצפיות(projections) על קו וקטור. עושים זאת עבור מספר וקטורים בעלי זווית שונה על מנת למצוא את הוקטור עם סכום המרחקים המרובעים הגדול ביותר(Sum of Squared Distances).

זה יהיה PC1.

Loading scores- ערך החשיבות שלוקחת כל תכונה על הווקטור העצמי (ערכים מ0 עד 1)

סכום המרחקים המרובעים גדול ביותר אשר יצר את PC1 נקרא הערך העצמי של PC1.

השורש של הערך העמי הוא הsingular value של PC1.

PC2 יהיה הקו האנכי לPC1 בראשית הצירים.

משתמשים בהטלי המרחקים של כל תצפית מהPCs על מנת למקם אותם בגרף כאשר PC1 הוא ציר X וPC2 הוא ציר Y.

ניתן להבין כמה אחוזים משונות הנתונים הPC מסביר באמצעות חלוקה של סכום המרחקים המרובעים של הPC במספר התצפיות פחות 1

Text

Description automatically generated

נסכום את התוצאות של שני הPCs ונחלק כל סכום מרחק המרובעים של PC בסכום הכולל וכך נקבל את אחוז השונות שהPC מסביר.

**t-SNE**

אלגוריתם להורדת ממדים

משמש למציאת קלאסטרים

משמש לויזואליזציה של הנתונים

מאפשר להבין את המבנה של הנתונים בצורה טובה

מחליטים מראש את מספר הממדים שיצא

זמן הריצה של t-sne הוא ארוך מאוד

1. מחשבים את המרחקים בין כל זוג נקודות בממד הגבוה (לפי נורמה אוקלידית) ואז נקבל מטריצה שבה יש את המרחק של כל נקודה מהשאר.
2. עושים סטנדריזציה למרחקים על מנת לקבל ערכים בין 0 ל1, משתמשים בהתפלגות גאוסיאנית על מנת ליצור פיזור טוב של המרחקים- עבור כל נקודה מייצרים התפלגות נורמלית כאשר מרכז ההתפלגות היא הנקודה וממקמים את כל מרחקי הנקודות ממנה על ההתפלגות הנורמלית. כאשר הסטיית תקן של ההתפלגות שווה 1.מחלקים את המרחק של כל נקודה מהנקודה שבמרכז ההתפלגות בסכום המרחקים של כל הנקודות ממנה על מנת לקבל סטדריזציה מ0 ל1.
3. שמים את הנקודות על גרף דו מימדי , מחשבים את המרחקים בין כל זוג נקודות בגרף זה ואז נקבל מטריצה שבה יש את המרחק של כל נקודה מהשאר (מרחקים בדו ממד)
4. האלכסון הראשי במטריצות יהיה 0 כי המרחק של נקודה מעצמה הוא 0.
5. מזיזים את המרחקים בין הנקודות בדו ממד על מנת שהמרחק יהיה יותר קרוב למרחק במטריצה ברב מימד (המימד הגבוה) ואז המטריצה בדו מימד ייצגו את המרחק בין הנקודות ברב מימד.
6. מחשבים את המרחק בין ההתפלגויות בין 2 ערכי המרחקים –של הממד הגובה והממד הנמוך ומנסים לצמצם את המרחק ביניהן- מעדכנים את המרחק בין הנקודות בצורה איטרטיבית כך שהמרחקים במימד הנמוך יהיו דומים למימדים במרחק הגבוה.

Text, letter

Description automatically generated

Text

Description automatically generated

Preplexity-כמה שכנים קרובים אנחנו רוצים

n\_iter-מספר האיטרציות עד שהאלגוריתם עוצר

\*t-SNE מאוד משופע מהיפר פרמטרים, לכן הוא משמש בעיקר לויוזאליזציה של הנתונים, אך לא לחיזוי.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **אלגוריתם** | **דרישות** | **יתרונות** | **חסרונות** | **Loss Function** |
| **עץ החלטה** | לא יכול nulls | ניתן לראות את תהליך קבלת ההחלטה ולתרגם אותו לשפות אחרות | בעל high variance – רגיש לסט נתונים שנותנים להתאמן עליו.  עלול ליצור עץ מאוד מסובך בשימוש ברגרסיה  לא בעל דיוק גבוה |  |
| **Adaboost** | לא יכול nulls, רגיש לoutlayers ולכן עדיף scaling |  |  |  |
| **Random Forest** |  | דיוק גבוה |  |  |
| **Gradiant Boosting** |  |  | רגיש לoutlayers, לוקח הרבה זמן |  |
| **KNN** | עדיף לעשות scaling, נדרש להגדיר מראש את מספר השכנים |  | -תצפיות חריגות עלולות להיכנס לקלאסטר אחר | סכום מרחקי הנקודות ממרכז הקלאסטרים |
| **K-Means** | נדרש להגדיר מראש את מספר השכנים, להוריד שורות כפולות, לא יכול לקבלnulls |  | יוצר רק קלאסטרים בצורת עיגול |  |
| **GMM** |  | מדויק יותר מK-Means, יכול ליצור קלאסטרים בצורות שונות |  |  |
| **Hierarchical Clustering** |  | לא דורש הגדרה מראש של מספר הקלאסטרים |  | Affinity -רמת דמיון בין קלאסטרים |
| **PCA** |  |  |  |  |
| **t-SNE** |  |  | בעל זמן ריצה ארוך, רגיש מאוד לשינויים בהיפר-פרמטרים | Kullback leibler divergence |
| **SVM** | דורש scaling של הנתונים כדי למנוע זמן ריצה ארוך מאוד | דיוק גבוה | בעל זמן ריצה ארוך |  |