**הרצאה 4**

**KNN-K Nearest Neighbor**

על פי אלגוריתם זה מבצעים סיווג של כל נקודה לפי k הנקודות הכי קרובות אליה.

אין מבנה נתונים שאותו שומרים, אלא פשוט מקבלים עבור כל נקודה את הK נקודות הכי קרובות- כל K נקודות הכי קרובות זה קבוצה

Leave one out verification

מאמנים את המודל על N-1 נקודות (מאמנים- בודקים מה הK שייתן מספר הטעויות הכי נמוך)

מסווגים את הנקודה שלא אימנו עליה

בודקים לפי הסיווג האמיתי האם המודל הצליח לסווג את הנקודה נכון

**עבור אלגוריתם KNN:**

מבצעים KNN כשאר K=N-1

לכן עבור כל קבוצה תישאר נקודה אחת בחוץ

ואז נבדוק לאיזו קבוצה נקודה זו שייכת על מנת לבדוק האם האלגוריתם מדויק

K fold cross validation

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
   1. Take the group as a hold out or test data set
   2. Take the remaining groups as a training data set
   3. Fit a model on the training set and evaluate it on the test set
   4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.

הסיבוכיות של הרצת KNN

KND כאשר

K -מספר השכנים שצריך לסווג

N דוגמאות

D מספר הממדים של הנתונים

ANN-Approximate Nearest Neighbor

מחזיר פחות או יותר את הנקודות הכי קרובות, גרסה הרבה יותר מהירה אך יש יותר סיכוי לטעויות.

בדרך כלל הטעויות לא גורמות לסיווג לא נכון לקלאסים ולכן עדיין ניתן להשתמש באלגוריתם זה.

עץ KD: K dimensions tree

K=מספר הממדים=מספר התכונות של האובייקט

מוצאים את החציון של תכונה מסוימת, חותכים את הנתונים ל2 קבוצות לפי החציון של תכונה זו,

מבצעים את תהליך זה שוב ושוב, כאשר כל חלוקה היא לפי תכונה אחרת

מגדירים מראש את עומק העץ לכן מספר התכונות שמשתמשים בהם, הם עומק העץ שהגדרנו.

התכונה שעל פיה חותכים כל פעם נבחרת באופן רנדומלי(התכונה שנבחרת היא התכונה שמספרה הוא התוצאה שיוצאת מעומק מודולו k)- האלגוריתם לא מבדיל בחשיבות בין התכונות.

על מנת למצוא את ה NN-Nearest Neighbor יורדים בעץ על מנת למצוא את הנקודה שלה מחפשים את השכנים, כאשר מגיעים לעלה, בודקים את המרחק בנקודה בעלה זה לנקודה הרצויה ושומרים את הנקודה כהמרחק הכי קצר הנוכחי, עולים שלב אחד בעץ לכיוון השורש ובודקים האם הנקודה בצומת זה יותר קרובה לנקודה הרצויה, אם כן מחליפים אותה כהמרחק הכי קצר הנוכחי, כך הלאה עד שמוצאים את הנקודות הכי קרובות.

ככל שמספר הממדים הולך וגדל (K) האלגוריתם נהיה פחות יעיל ויכול לעשות יותר טעויות.

הסיבוכיות של האלגוריתם היא O(logn) וככל שמספר הk גדל מתקרבים יותר לסיבוכיות n.

Randomized KD Tree

בונים מספר עצי KD באופן רנדומלי- במקום להשתמש בחישוב של המודולו בוחרים תכונה רנדומלית לחתוך לפיה וכך נקבל מספר עצים שונים. בעל ביצועים טובים יותר מKD tree.

בונים כמה עצי KD רנדומליים ובודקים את הNNs שמתקבלים בכל עץ.

ניקח את האיחוד של כל התוצאות ונחפש בנקודות באיחוד את הNNs- nearset neighbors.

כל נקבל תשובה בסיכוי יותר גבוה שבה נקבל את ה NNs האמיתיים.

D-הממד של הdata

N- מספר נקודות

d- המרחק בין הדוגמה לNN שנמצא

-המרחק האמיתי בין הנקודה לNN שלה

*- המרחק בין הנקודה לNN המקורב שלה*

????

**Locality Sensitive Hashing -LSH**

מגרילים k חתכים אקראיים

L-מספר תאים

בהינתן דוגמה כל שהי x בודקים האם הדוגמה מקיימת את התנאי

זאת אומרת האם הדוגמה גדולה מהערך של כל חתך או לא, עבור כל חתך מקבלים תשובה (false)0 או1(true).

הוקטור שמתקבל של אפסים או אחדים עבור כל חתוך עבור דוגמה זו משמש כkey בhash table

והנקודה נשמרת כdataשל key זה.

מבצעים את תהליך זה עבור כל נקודה.

כך יכול להיות מצב שכמה נקודות הן בעלי אותו key

כך נקבל תאים שיש בהם נקודות אשר כל תא מכיל נקודות שמתאימות לוקטור של אפסים ואחדים של תא זה.

על מנת למצוא את הNN של נקודה נחפש את הנקודות הכי קרובות בתא של נקודה זו.

ככל שk יותר גדול מספר האלמנטים בתא בממוצע יורד ולכן החיפוש יהיה יותר מהיר.

ככל שk יורד הזמן עולה

ככל שk עולה הדיוק יורד- יש יותר סיכוי שהשכן הקרוב ביותר יהיה בתא אחר

על מנת לבחור את הK וL האופטימליים בודקים את רמת הטעות עבור כל K ועבור כל K בודקים את מספר התאים המינימלי L שמאפשר את מספר החתכים K.

עבור כל הזוגות L,K אשר נותנים את השגיאה הכי נמוכה, מחפשים את הזוג L,K עם הזמן ריצה הכי מהיר ואלו יהיו ה L,K הכי אופטימליים.

\*כיוון שהאלגוריתמים של KNN לא יודעים לזהות אילו תכונות אינן בעלות חשיבות ואילו לא, נדרש לבצע feature selection- להבין אילו תכונות הן החשובות ביותר ורק איתן לבצע KNN.

כך גם נגדיל את הדיוק וגם את זמן הריצה של האלגוריתם.

סיבוכיות זמן האלגוריתם:

N- מספר התצפיות (נקודות)

D- מספר הממדים (תכונות)

K- מספר החלוקות

L- מספר החזרות על תהליך החלוקה

בדרך כלל בוחרים

ואז סיבוכיות הזמן היא:

לעומת KNN רגיל שסיבוכיות הזמן שלו היא O(ND)