K-Nearest Neighbor Classification

Agenda

- KNN Classification Algorithm
- Solving Business Problems using KNN Algorithm
- Hands-on
- Compare Multiple Classification Algorithms

Sample Business Problem

greatlearning

- Let’s assume a money lending company “XYZ” like UpStart, IndiaLends, etc.
- Money lending XYZ company is interested in making the money lending system comfortable & safe for lenders as well as for borrowers. The company holds a database of customer’s details.
- Using customer’s detailed information from the database, it will calculate a credit score(discrete value) for each customer.
- The calculated credit score helps the company and lenders to understand the credibility of a customer clearly.
- So they can simply take a decision whether they should lend money to a particular customer or not.

Sample Business Problem

- The customer's details could be:
  - Educational background details
    - Highest graduated degree
    - Cumulative grade points average (CGPA) or marks percentage
    - The reputation of the college
    - Consistency in his lower degrees
    - Whether to take the education loan or not
    - Cleared education loan dues
  - Employment details
    - Salary
    - Year of experience
    - Got any onsite opportunities
    - Average job change duration

Sample Business Problem

greatlearning

- The company(XYZ) use's these kinds of details to calculate credit score of a customer
- The process of calculating the credit score from the customer's details is expensive
- To reduce the cost of predicting credit score, they realized that the customers with similar background details are getting a similar credit score
- So, they decided to use already available data of customers and predict the credit score using it by comparing it with similar data
- These kinds of problems are handled by the K-nearest neighbor classifier for finding the similar kind of customers

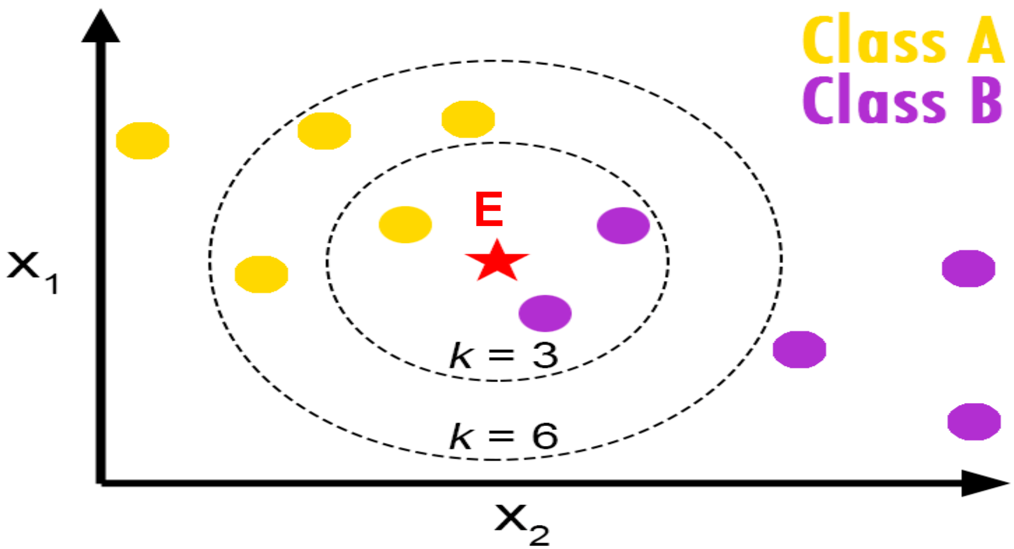
Introduction

- K-nearest neighbor classifier is one of the introductory supervised classifier, which every data science learner should be aware of
- Fix & Hodges proposed K-nearest neighbor classifier algorithm in 1951 for performing pattern classification task
- For simplicity, this classifier is called as KNN Classifier
- K-nearest neighbor classifier mostly represented as KNN, even in many research papers too
- KNN addresses the pattern recognition problems and also the best choices for addressing some of the classification related tasks
- The simple version of the K-nearest neighbor classifier algorithms is to predict the target label by finding the nearest neighbor class
- The closest class will be identified using the distance measures like Euclidean distance

K_Nearest Neighbour Algorithm

To determine the class of a new example E:

- Calculate the distance between E and all examples in the training set
- Select K-nearest examples to E in the training set
- Assign E to the most common class among its K-nearest neighbors



Distance Between Neighbors

Each example is represented with a set of numerical attributes

Jay:
Age=35
Income=95K
No. of credit cards=3

Rina:
Age=41
Income=215K
No. of credit cards=2

- “Closeness” is defined in terms of the Euclidean distance between two examples

- The Euclidean distance between X=(x_1, x_2, x_3,…x_n) and Y =(y_1,y_2, y_3,…y_n) is defined as:

$$D(X,Y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

$$\text{Distance (Jay,Rina)} = \sqrt{(35-41)^2+(95{,}000-215{,}000)^2+(3-2)^2}$$

K_Nearest Neighbours: Example

Customer	Age	Income	No. credit cards	Response
Jay	35	35K	3	No
Rina	22	50K	2	Yes
Hema	63	200K	1	No
Tommy	59	170K	1	No
Neil	25	40K	4	Yes
Dravid	37	50K	2	?

K_Nearest Neighbours: Example

Customer	Age	Income	No. credit cards	Response	Distance from Dravid
Jay	35	35K	3	No	$\sqrt{(35-37)^2+(35-50)^2+(3-2)^2}$ $= 15.16$
Rina	22	50K	2	Yes	15
Hema	63	200K	1	No	152.23
Tommy	59	170K	1	No	122
Neil	25	40K	4	Yes	15.74
Dravid	37	50K	2	?	0

K_Nearest Neighbours

Jay:
Age=35
Income=95K
No. of credit cards=3

Rina:
Age=41
Income=215K
No. of credit cards=2

Distance (Jay, Rina)=sqrt $[(35-45)^2 + \mathbf{(95{,}000-215{,}000)^2} + (3-2)^2]$

- Distance between neighbors could be dominated by some attributes with relatively large numbers (e.g., income in our example)
- **Important to normalize some features**
  (e.g., map numbers to numbers between 0-1)

  Example: Income
  Highest income = 500K
  Davis's income is normalized to 95/500, Rina income is normalized to 215/500, etc.)

K_Nearest Neighbours

Normalization of Variables

Customer	Age	Income	No. credit cards	Response
Jay	55/63= 0.175	35/200= 0.175	3/4= 0.75	No
Rina	22/63= 0.34	50/200= 0.25	2/4= 0.5	Yes
Hema	63/63= 1	200/200= 1	1/4= 0.25	No
Tommy	59/63= 0.93	170/200= 0.175	1/4= 0.25	No
Neil	25/63= 0.39	40/200= 0.2	4/4= 1	Yes
Dravid	37/63= 0. 58	50/200= 0.25	2/4= 0.5	Yes

K-Nearest Neighbor

- Distance works naturally with numerical attributes
  d(Rina,Johm)= $\sqrt{(35-37)^2+(35-50)^2+(3-2)^2}$ =**15.16**
- What if we have nominal attributes?

<u>Example</u>: Married

Customer	Married	Income	No. credit cards	Response
Jay	Yes	35K	3	No
Rina	No	50K	2	Yes
Hema	No	200K	1	No
Tommy	Yes	170K	1	No
Neil	No	40K	4	Yes
Dravid	Yes	50K	2	Yes

Non-Numeric Data

- Feature values are not always numbers
- Example
  - Boolean values: Yes or no, presence or absence of an attribute
  - Categories: Colors, educational attainment, gender
- How do these values factor into the computation of distance?

Dealing with Non-Neumeric Data

- Boolean values => convert to 0 or 1
  - Applies to yes-no/presence-absence attributes
- Non-binary characterizations
  - Use natural progression when applicable; e.g., educational attainment: GS, HS, College, MS, PHD => 1,2,3,4,5
  - Assign arbitrary numbers but be careful about distances; e.g., color: red, yellow, blue => 1,2,3
- How about unavailable data?
  (0 value not always the answer)

Preprocessing Your Dataset

- Dataset may need to be preprocessed to ensure more reliable data mining results
- Conversion of non-numeric data to numeric data
- Calibration of numeric data to reduce effects of disparate ranges
  - Particularly when using the Euclidean distance metric

Distance measures

- How to determine similarity between data points
  - using various distance metrics
- Let x = (x1,…,xn) and y = (y1,…yn) be n-dimensional vectors of data points of objects g1 and g2
  - g1, g2 can be two different genes in microarray data
  - n can be the number of samples

Distance measure

- Euclidean distance

$$d(g_1, g_2) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Manhattan distance

$$d(g_1, g_2) = \sum_{i=1}^{n} \left|(x_i - y_i)\right|$$

- Minkowski distance

$$d(g_1, g_2) = \sqrt[m]{\sum_{i=1}^{n} (x_i - y_i)^m}$$

Correlation distance

- Correlation distance

$$r_{xy} = \frac{Cov(X,Y)}{\sqrt{(Var(X) \cdot Var(Y)}}$$

  – Cov(X,Y) stands for covariance of X and Y
    - degree to which two different variables are related
  – Var(X) stands for variance of X
    - measurement of a sample differ from their mean

Correlation distance

- Variance

$$Var(X) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{X})^2}{n-1}}$$

- Covariance

$$CoVar(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y})}{n-1}$$

  – Positive covariance
    - two variables vary in the same way
  – Negative covariance
    - one variable might increase when the other decreases
  – Covariance is only suitable for heterogeneous pairs

Correlation distance

- Correlation

$$r_{xy} = \frac{Cov(X,Y)}{\sqrt{(Var(X) \cdot Var(Y)}}$$

  - maximum value of 1 if X and Y are perfectly correlated
  - minimum value of 0 if X and Y are exactly opposite
  - $d(X,Y) = (1 + r_{xy})/2$

Cosine distance

- Cosine distance
  - Used in document comparison

$$-\cos(\theta) = \frac{X^T Y}{\|X\| \|Y\|}$$

Summary of similarity measures

- Using different measures for clustering can yield different clusters
- Euclidean distance and correlation distance are the most common choices of similarity measures for microarray data
- Euclidean vs Correlation Example
  - g1 = (1,2,3,4,5)
  - g2 = (100,200,300,400,500)
  - g3 = (5,4,3,2,1)
  - Which genes are similar according to the two different measures?

k-NN Variations

- Value of k
  - Larger k increases confidence in prediction
  - Note that if k is too large, decision may be skewed
- Weighted evaluation of nearest neighbors
  - Plain majority may unfairly skew decision
  - Revise algorithm so that closer neighbors have greater “vote weight”
- Other distance measures

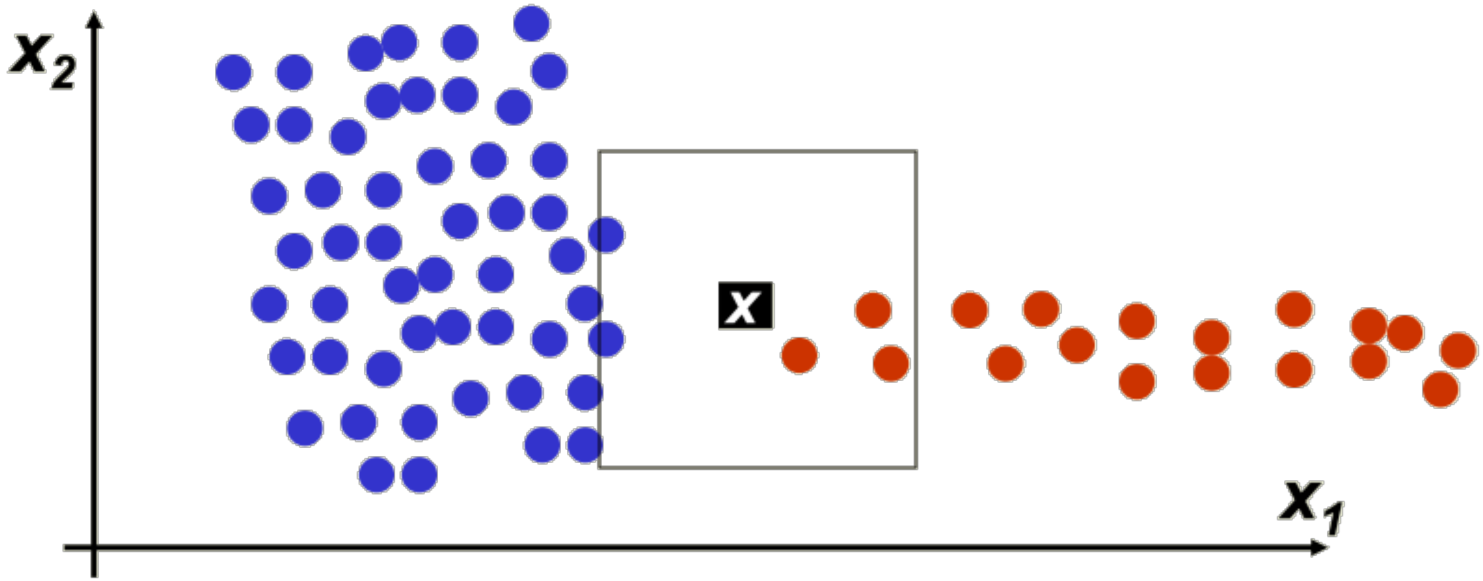
Other Distance Measures

- City-block distance (Manhattan dist)
  – Add absolute value of differences
- Cosine similarity
  – Measure angle formed by the two samples (with the origin)
- Jaccard distance
  – Determine percentage of exact matches between the samples (not including unavailable data)
- Others

Distance-Weighted Nearest Neighbor Algorithm

- Assign weights to the neighbors based on their 'distance' from the query point
  - Weight 'may' be inverse square of the distances
- All training points may influence a particular instance

How to Choose "K"?



- For k = 1, …,5 point x gets classified correctly
  - red class
- For larger k classification of x is wrong
  - blue class

How to Choose "K"?

- Selecting the value of K in K-nearest neighbor is the most critical problem.
- A small value of K means that noise will have a higher influence on the result i.e., the probability of overfitting is very high.
- A large value of K makes it computationally expensive and defeats the basic idea behind KNN (that points that are near might have similar classes).
- A simple approach to select K is $K = \sqrt{n}$
- It depends on individual cases, at times best process is to run through each possible value of K and test our result

KNN algorithm Pseudo Code

- Let (X_i, C_i) where $i = 1,2,\cdots,n$ be data points. X_i denotes feature values & C_i denotes labels for X_i for each i
- Assuming the number of classes as c, $C_i \in \{1,2,3,\cdots,c\}$ for all values of i
- Let x be a point for which label is not known
- We would like to find the label class using k-nearest neighbor algorithms.

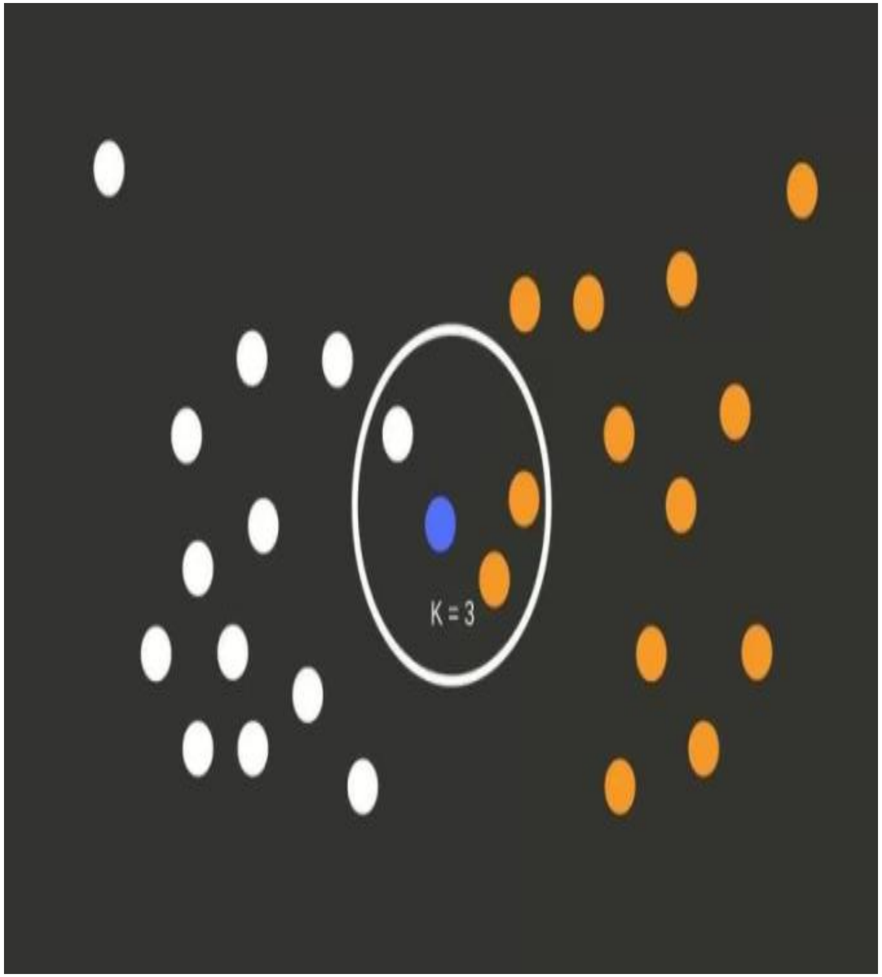
KNN algorithm Pseudo Code

- Calculate $d(x, x_i)$, $i = 1,2, \cdots, n$; where d denotes the Euclidean distance between the points.
- Let's consider a setup with n training samples, where x_i is the training data point.
- The training data points are categorized into c classes.
- Using KNN, we want to predict class for the new data point.
- So, the first step is to calculate the distance(Euclidean) between the new data point and all the training data points.
- Next step is to arrange all the distances in non-decreasing order.
- Assuming a positive value of k and filtering k least values from the sorted list.
- Now, we have k top distances.
- Let k_i denotes no. of points belonging to the i^{th} class among k points.
- If $k_i > k_j$ for all $i \neq j$ then put x in class i

KNN algorithm: Example

greatlearning

- Let’s consider the image shown here where we have two different target classes white and orange circles.
- We have total 26 training samples.
- Now we would like to predict the target class for the blue circle
- Considering k value as three, we need to calculate the similarity distance using similarity measures like Euclidean distance.
- If the similarity score is less which means the classes are close.
- In the image, we have calculated distance and placed the less distance circles to blue circle inside the Big circle.



Condensed Nearest Neighbor Data Reduction Rule

- Working on a big dataset can be an expensive task.
- Using the condensed nearest neighbor rule, we can clean our data and can sort the important observations out of it.
- This process can reduce the execution time of the machine learning algorithm. But there is a chance of accuracy reduction.
- The steps to condense are to divide data points into the following:
  - Outliers: Observations that lie at an abnormal distance from all the data points. Most of these are extreme values. Removing these observations will increase the accuracy of the model.
  - Prototypes: Minimum points in training set required to recognize non-outlier points.
  - Absorbed points: These are points that are correctly identified to be non-outlier points.

Nearest Neighbor Algorithm

- Nearest neighbor is a special case of k-nearest neighbor class.
- In this case k value is 1 ($k = 1$).
- In this case, new data point target class will be assigned to the 1^{st} closest neighbor

Radius Neighbor Algorithm

greatlearning

- KNN classifier is also considered to be an instance based learning / non-generalizing algorithm.
- It stores records of training data in a multidimensional space.
- For each new sample & particular value of K, it recalculates Euclidean distances and predicts the target class
- So, it does not create a generalized internal model
- Similar to KNN classifier, we can use Radius Neighbor Classifier for classification tasks.
- As in KNN classifier, we specify the value of K, similarly, in Radius neighbour classifier the value of R should be defined.
- The RNC classifier determines the target class based on the number of neighbors within a fixed radius, r, for each training data point.

Radius Neighbor Algorithm

greatlearning

- Measuring similarity with distance between the points using Euclidian method
- Other distance measurement methods include Manhattan distance, Minkowski distance, Mahalanobis distance, Bhattacharya distance etc.
- Scikit-learn implements two different nearest neighbors classifiers:
  - K Nearest Neighbor Classifier
  - Radius Neighbor Classifier
- Radius Neighbor Classifier implements learning based on number of neighbors within a fixed radius ‘r’ of each training point, where ‘r’ is a floating point value specified by the user
- Determining the optimal K is the challenge in K Nearest Neighbor classifiers. In general, larger value of K supresses impact of noise but prone to majority class dominating
- Radius Neighbor Classifier may be a better choice when the sampling is not uniform. However, when there are many attributes and data is sparse, this method becomes ineffective due to curse of dimensionality

Advantages and Disadvantages

greatlearning

- Advantages
  - Makes no assumptions about distributions of classes in feature space
  - Don’t need any prior knowledge about the structure of data in the training set
  - No retraining is required if the new training pattern is added to the existing training set
  - Can work for multi-classes simultaneously
  - Easy to implement and understand
  - Not impacted by outliers
  - KNN executes quickly for small training data sets
  - Performance asymptotically approaches the performance of the Bayes Classifier
- Disadvantages
  - Fixing the optimal value of K is a challenge
  - Will not be effective when the class distributions overlap
  - Does not output any models. Calculates distances for every new point (lazy learner)
  - For every test data, the distance should be computed between test data and all the training data. Thus a lot of time may be needed for the testing
  - Computationally intensive (O(N^2))


Demo Using Python

Sample Code in Python

```
X = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X, y) KNeighborsClassifier(...)
(neigh.predict([[1.1]])) [0]

(neigh.predict_proba([[0.9]])) [[ 0.66666667 0.33333333]]
```

Summary

- KNN classification algorithm
  - Different distance measures
  - KNN algorithm
  - Advantages and disadvantages
- Case study 1 (using KNN)
- Hands-on 1
- Case study 2 (comparing KNN with other classification methods)
- Hands-on 2

Thanks!