Applied Data Science

Session 21: K-Nearest Neighbour Algorithm

Dr. Soharab Hossain Shaikh

1

K- Nearest Neighbours Algorithm (KNN)

2

Instance-Based Learning

- **Idea:**
  - Similar examples have similar label.
  - Classify new examples like similar training examples.
- **Algorithm:**
  - Given some new example ***x*** for which we need to predict its class y
  - Find most similar training examples
  - Classify ***x*** "like" these most similar examples

5

3

K-Nearest Neighbour Classification Algorithm

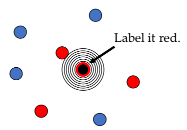
- Training method:
  - Save the training examples – No explicit training phase
- At prediction time:
  - Find the ***k*** training examples $(x_1,y_1),\ldots,(x_k,y_k)$ that are closest to the test example ***x***
  - Predict the **most frequent class** among those y_i's.

4

4

1-Nearest Neighbour

- One of the simplest of all machine learning classifiers
- Simple idea: label a new point the same as the closest known point



6

5

1-NN – Important Aspects

A distance metric- **Euclidean**

- When different units are used for each dimension
  → Normalize each dimension by standard deviation
- For discrete data, can use hamming distance
  → D(x1, x2) = number of features on which x1 and x2 differ
- Others (e.g., normal, cosine)

How many nearby neighbors to look at? **One – for Nearest Neighbour**

How to fit with the local points?

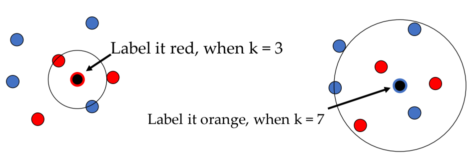
- Just predict the same output as the nearest neighbor.

7

6

K-Nearest Neighbour

- Generalizes 1-NN to smooth away noise in the labels
- A new point is now assigned the most frequent label of its ***k*** nearest neighbors



9

7

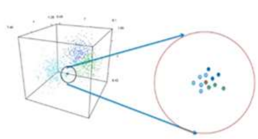
Instance-based Learning

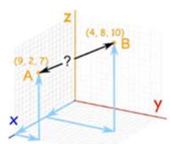
- Instance-based learning is often termed *lazy* learning, as there is typically no "transformation" of training instances into more general "statements"
- Instead, the presented training data is simply stored and, when a new query instance is encountered, a set of similar, related instances is retrieved from memory and used to classify the new query instance
- Hence, instance-based learners never form an explicit general hypothesis regarding the target function.
- They simply compute the classification of each new query instance as needed.

10

8

Euclidean Distance – Similarity Measure





$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Measuring similarity with distance between the points using Euclidean method.

9

Example

- Determining **decision on scholarship** application based on the following features:
  - Household income (annual income in thousands of rupees)
  - Number of siblings in family
  - High school grade (on a scale of 1.0 – 10.0)

- Intuition (reflected on data set): Award scholarships to high-performers and to those with financial need.

11

10

Dealing with Non-Numeric Data

- Feature values are not always numbers

  e.g. - Boolean values: Yes or No, presence or absence of an attribute

  Categories: Colors, Gender

- 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: 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

12

11

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

13

12

Data Preprocessing

- 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

14

13

Importance of Feature Scaling

- The Euclidean distance formula has the implicit assumption that the different dimensions are comparable.
- Features that span wider ranges affect the distance value more than features with limited ranges.
- Suppose household income was instead indicated in thousands of rupees per month and that grades are given on a 50-100 scale.

15

14

Standardization

- Transform raw feature values into z-scores

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

x_{ij} is the value for the i^{th} sample and j^{th} feature

μ_j is the average of all x_{ij} for feature *j*

σ_j is the standard deviation of all x_{ij} over all input samples

- Those dimensions which have larger possible range of values will dominate the result of the distance calculation using Euclidian formula.
- To ensure all the dimensions have similar scale, we normalize the data on all the dimensions/ attributes.
- There are multiple ways of normalizing the data. We will use Z-score standardization.

16

15

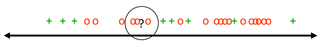
Other Distance Measures

- **City-block distance (Manhattan distance)**
  - Add absolute value of differences (state-space heuristic search)
- **Cosine similarity** - Measure angle formed by the two samples (with the origin) (NLP – comparing two word-vectors)
- **Correlation-based similarity** (e.g. comparing two images)

17

16

K-NN and Irrelevant Features



> Every feature may not be equally important.

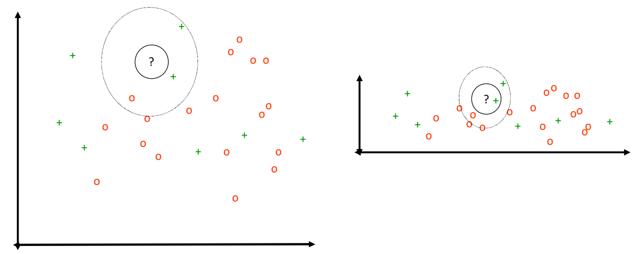
> Feature Selection is important – we may use weighted distance measures – give different weights to different features

Distance computation makes sense on relevant features.
(e.g. Decision on scholarship - high school grade and household income are better features compared to gender and geographic location of the residence of a candidate)

18

17

Importance of Feature Scaling in KNN



18

18

K-NN is Incremental

- All training instances are stored
- Model consists of the set of training instances
- Adding a new training instance only affects the computation of neighbors, which is done at execution time (i.e., lazily)

22

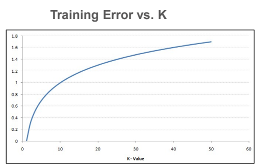
19

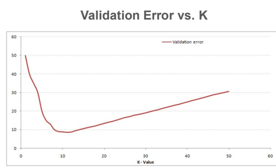
Choosing k

- Determining the optimal K is a challenge.
- Higher values of *k* provide smoothing - reduces the risk of overfitting due to noise in the training data
- if *k* is too high, we will miss out on the method's ability to capture the local structure in the data.
- In the extreme, *k* = *n* = the number of records in the training dataset - all records mapped to the majority class in the training data
- In general, larger value of *k* suppresses impact of noise but prone to majority class dominating.
- If *k* is too low, we may be fitting to the noise in the data.

20

How to choose the optimum value of k?





- To get the optimal value of K, you can segregate the training and validation from the initial dataset
- Now plot the validation error (alternatively test accuracy) curve to get the optimal value of K
- This value of K should be used for all predictions

21

k-NN Time Complexity

- Suppose there are **m** instances and **n features** in the dataset
- Nearest neighbor algorithm requires computing **m** distances
- Each distance computation involves scanning through each of the **n** features
- Running **time complexity** is proportional to **m x n**

19

22

Pros and Cons of KNN

Advantages

- Not impacted by Outliers (why???)
- Makes no assumptions about distributions of classes in feature space

Disadvantages

- Fixing the optimal value of K is a challenge.
- Does not output any models. Calculates distances for every new point.

23

Applications of KNN

- Pattern recognition in Optical character recognition
- Concept search in semantic models in NLP
- Detect similar buying patterns in customer analytics
- Text classification

24

How to *build* a KNN Classifier Model in *Scikit-Learn*

25

KNN Classifier in Scikit-Learn

sklearn.neighbors.KNeighborsClassifier

class sklearn.neighbors.**KNeighborsClassifier**(*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs*) [source]

- Build an estimator model like **KNeighborsClassifier()**
- Use **fit()** function to ***Train*** the model with **Training Dataset**
- Use **predict()** function to ***Test/Evaluate*** the model with **Test Dataset**
- Use **predict()** function to make ***prediction/inference*** on **New Unseen Data**

26

Example Code

```
>>> 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(...)
>>> print(neigh.predict([[1.1]]))
[0]
>>> print(neigh.predict_proba([[0.9]]))
[[0.66666667 0.33333333]]
```

27

28

Case-study

29

Context:

The dataset to be considered consists of a wide variety of intrusions simulated in a military network environment.

It was created in an environment to acquire raw TCP/IP dump data for a network by simulating a typical US Air Force LAN. The LAN was focused like a real environment and blasted with multiple attacks.

For each TCP/IP connection, 41 quantitative and qualitative features are obtained from normal and attack data (3 qualitative and 38 quantitative features) .

The class variable has two categories:

- Normal
- Anomalous

30

30

Dataset

https://www.kaggle.com/what0919/intrusion-detection

Data basically represents the packet data for a time duration of 2 seconds.

1-9 Columns: basic features of packet (type 1)

10-22 columns: employ the content features (type 2)

23-31 columns: employ the traffic features with 2 seconds of time window (type 4)

32-41 columns: employ the host based features

31

31

Let’s go to the Coding Demo…

32

To be continued in the next session…..

33