CUNY School of Professional Studies

SPS.CUNY.EDU

Lecture 09
2020 Spring Data-622
k Nearest Neighbors

Raman Kannan

Instructor Email Address: Raman.Kannan@sps.cuny.edu

Acknowledgements:
Generous support from IBM Power Systems Academic Initiative
IBM PSAI provides computing infrastructure for free

Script for Algorithms

Develop the Intuition
Understand the assumptions
Develop the mathematics
Run the algorithms
Learn to interpret the result/output
Predict using the model
Learn to determine the performance
Distinguish training/testing error
Differentiate between overfitting/underfitting
Techniques to improve performance

Intuition (qualitative Model)

Birds of a feather flock together. That is the wisdom!

What is its relevance to our problem.

We are given a set of observations we are asked to learn that and then asked to classify when a new observation is presented.

So, if I tell you 23 birds are flocking together and a FEW of them are identified as Canadian Geese, is it appropriate to infer the rest of the unidentified birds as Canadian Geese.

This is the mental process we want to implement in an algorithm.

To do so algorithm should know "flocking together." Human brain does that effortlessly. Also, algorithms cannot handle FEW. What is your personal threshold?
Will you classify the unidentified bird as Canadian Geese if you identify one other bird to be Canadian Geese?
Will you require 10/or ALL the birds in the flock to be identified?
That number is the k in kNN. k determines the amount of work computational load. Bigger the K, longer it takes to identify a new Bird. What is the ideal K, for any dataset?

kNN Intuition 2 – together

Our brain has figured out to make sense out of together. It is a concept.

In the corridor we see two students walking in the same direction, one is sliding by the right side wall and the other clinging to the left wall. Our brain will not think of them as together.

We have an abstract sense of what goes with what – we use an assortment of features to determine this.

We tend to frozen in one bag and we dont pack soft vegetables with hard vegetables (tomatoes vs cabbage).

Our mind is seeking similarity (or disimilarity).

Our mind computes this in a way we cannot easily define.

Academics refer to this as “distance”.

Distance

ⓘ Not secure | axon.cs.byu.edu/papers/wilson.ml2000.drop.pdf

(Tversky, 1977); hyperrectangle distance functions (Salzberg, 1991; Domingos, 1995) and others. Several of these functions are defined in Figure 1 (Wilson & Martinez, 1997a).

Minkowsky:

$$D(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r \right)^{1/r}$$

Euclidean:

$$D(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\mathbf{x},\mathbf{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T Q(\mathbf{x}-\mathbf{y}) = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} (x_i - y_i) q_{ji} \right) (x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\mathbf{x},\mathbf{y}) = [\det V]^{1/m} (\mathbf{x}-\mathbf{y})^{\mathrm{T}} V^{-1} (\mathbf{x}-\mathbf{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} \frac{1}{sum_i} \left(\frac{x_i}{size_{\mathbf{x}}} - \frac{y_i}{size_{\mathbf{y}}} \right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_{\mathbf{x}}$ is the sum of all values in the vector $\mathbf{x}$.

Kendall's Rank Correlation:

$$D(\mathbf{x},\mathbf{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \mathrm{sign}(x_i - x_j)\,\mathrm{sign}(y_i - y_j)$$

sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

KNN Intuition 3 – feature selection

We see some students walking/hanging out together?

We have two classification problems at hand!
A)We want to identify students who will succeed and others who may need some encouragement!
B)We want to identify their gender!

We can all agree for B, we may consider the presence of a handbag or the apparel they are wearing.

We cannot use these attributes for A, instead we may look for their transcripts, prior schools they have attended, their attendance and yes their participation in classes and their homework grade. But definitely not handbag or apparel.

If we implement a NN algo using irrelevant features, GIGO will be in effect.

This problem domain is simple for a class room. But if we are engineering a system to identify target objects (friend or foe) or malignant or benign diagnostic we need more domain expertise.

Therefore, the crux of kNN

Is to know the appropriate distance function given problem domain?
- what features are relevant to compute this distance
- and not using irrelevant features is important

Is to know the appropriate N the number of neighbors in the domain?
- how to aggregate the results from the N neighbors
- majority vote, etc

KNN is inherently parallelizable – we can compute distances in parallel

There is no training, given a new observation, kNN must compute the D between the new and all the other known data points.

Let us look at some implementation

R Implementation

https://dataaspirant.com/2017/01/02/k-nearest-neighbor-classifier-implementation-r-scratch/

Data

Distance function

Iterate → Sort → Combine

Class/ label

```
euclideanDist <- function(a, b){
  d = 0
  for(i in c(1:(length(a)-1) ))
  {
    d = d + (a[[i]]-b[[i]])^2
  }
  d = sqrt(d)
  return(d)
}
```

```
knn_predict <- function(test_data, train_data, k_value){
  pred <- c()  #empty pred vector
  #LOOP-1
  for(i in c(1:nrow(test_data))){   #looping over each record of test data
    eu_dist =c()          #eu_dist & eu_char empty  vector
    eu_char = c()
    good = 0              #good & bad variable initialization with 0 value
    bad = 0

    #LOOP-2-looping over train data
    for(j in c(1:nrow(train_data))){

      #adding euclidean distance b/w test data point and train data to eu_dist vector
      eu_dist <- c(eu_dist, euclideanDist(test_data[i,], train_data[j,]))

      #adding class variable of training data in eu_char
      eu_char <- c(eu_char, as.character(train_data[j,][[6]]))
    }
# end of iteration continued next slide
```

kNN

Sorting/
Combining
Labeling

```
eu <- data.frame(eu_char, eu_dist) #eu dataframe created with eu_char & eu_dist
columns

    eu <- eu[order(eu$eu_dist),]      #sorting eu dataframe to gettop K neighbors
    eu <- eu[1:k_value,]              #eu dataframe with top K neighbors

    tbl.sm.df<-table(eu[,labelcol])
    cl_label<-  names(tbl.sm.df)[[as.integer(which.max(tbl.sm.df))]]

    pred <- c(pred, cl_label)

  }
  return(pred) #return pred vector
}
```

```
> dftest
  height weight foot
1      6    130    8
> dftrain
  gender height weight foot
1      M   6.00    180   12
2      M   5.92    190   11
3      M   5.58    170   12
4      M   5.92    165   10
5      F   5.00    100    6
6      F   5.50    150    8
7      F   5.42    130    7
8      F   5.75    150    9
> knn_predict(dftest,dftrain,3,1)
[1] "F"
>
```

standard Implementation

```
> require(class)
Loading required package: class
>

> class::knn(dftrain[,-1],dftest,dftrain$gender,k=3)
[1] F
Levels: F M
>
```

kNN Summary

K here denotes the number of neighbors to consider
– what will happen (bias and variance)
    – when k=1
    – when k=n (all the observations)

Knn is an instance based – non parametric
– non probabilisting classifier
– makes no assumption about linear/non-linearity in the data
– it can work in both scenarios

– parallelizable but scaling is non-trivial
    – curse of dimensionality
    – the distance function as a function of number of attributes
    – mind relevancy attributes
    – what distance is and how best to represent it

– easy to understand and to implement

References

http://www.learnbymarketing.com/tutorials/k-nearest-neighbors-in-r-example/

https://www.quora.com/What-is-the-k-nearest-neighbors-algorithm

https://github.com/santhalakshminarayana/Machine-Learning

Iris Data Set Classification using K-Nearest Neighbors (K-NN).ipynb

http://axon.cs.byu.edu/papers/wilson.ml2000.drop.pdf