DEMO 1 - SUPPORT VECTOR MACHINES

QUESTION 1 -

This is an everyday problem that I try to solve since I'm quite active on LinkedIn - Given a set of profiles who see my post, and given the current post I've put up, how many of them would read my post fully, and how many of them would not?

The following are the attributes that I have learned of by trial-and-erro over a period of time ~

1. Manner of view - categorical (through profile, through mutual connect, through direct connect)

2. Impact Score of Opening Line - numerical

Can be measured by the following sub-attributes:

1.1 Presence of atleast one rhetoric / poetic device / question / title / emoticon - boolean

2.1 Presence of more than one of these on the opening line - numerical

3.1 Grammatical correctness - boolean

4.1 Number of descriptive words - numerical

5.1 Font and format size - categorical

3. Appeal Score of Picture (0, if none present) - numerical

Can be measured by the following sub-attributes:

3.1 Size of picture - categorical

3.2 Intersection between picture auto-tags and post tags - numerical

4. Intersection between hashtagged words on post and hashtags followed by the LinkedIn user - numerical

5. Average of the intersections between - numerical

-> keywords picked up from the first half of the post

-> keywords picked up from past posts viewed by that user

TARGET VARIABLE - Read Status - categorical (read post / did not read post)

QUESTION 2 - (Please scroll down to find the discussion)

KSVM SOLUTION ~

Note - I have not changed the curve, and have not cross-validated as of now. This experiments with C in most part.

C = 200 Accuracy = 0.8639144 OR 86%

Coefficients -

a1 = -3.565804e-04

a2 = -5.234880e-05

a3 = -1.650994e-04

a4 = 1.116171e-03

a5 = 1.007588e+00

a6 = -4.734253e-04

a7 = -5.133491e-05

a8 = -3.788560e-05

a9 = -6.765545e-05

a10 = 1.060083e-01

Constant -

a0 (constant) = 0.0814246

In [ ]:
```
#Code ~

library(kernlab)

data <- credit_card_data

model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),type="C-svc",kernel="vanilladot",C=200,scaled=TRUE)
err = model@error
pred <- predict(model,data[,1:10])
#see what fraction of the model's predictions match the actual classification
p = sum(pred == data[,11]) / nrow(data)
print(p)

a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
print(a)
#calculate a0
posa0 <- model@b
a0 <- -posa0
print(a0)
```

Experiments performed ~

1. Varying the degree, and then after narrowing down the degree, varying across the values of that degree.

Results -

Looping across degrees from 10^-12 to 10^-5, the accuracy is an all time low mark at approximately 55%. Then from there to 10^8, the accuracy reaches a plateau of around 86%, with a few cases where the accuracy dips to less than 70%.

1. Checking individual attribute correlation to target and varying attribute set selection.

Results - While individual correlations of each attribute with target variable vary from 0.6 to 0.8, the best combination of attributes for SVM still seems to be taking all the 10 attributes.

1. Experimenting with non-linear kernels for the same C value.

Accuracy Results -

3.1 Splinedot - 0.9785933 OR 98% (Improvement)

3.2 Polydot - 0.8639144 OR 86% (No change)

3.3 Tanhdot - 0.7217125 or 72% (Deterioration)

3.4 Laplacedot - 1 OR 100% (Improvement but unrealistic)

3.5 Besseldot - 0.9159021 OR 92% (Improvement)

3.6 Anovadot - 0.912844 OR 91% (Improvement)

3.7 Rbfdot - 0.9602446 OR 96% (Improvement)

Discussion for KSVM results

While there are many types of curves that are possible to use to fit this data, there are few observations I made -

1. The C value does not change in the way we would like it to change because the way the data is spaced out across the ten dimensions we were given, the margin and correctness trade-off do not let either contribute to an accuracy better than 86% with the vanilladot.

2. The data is definitely not linearly-separable by comparing the results of the splinedot and vanilladot. Had they been linearly separable, there would not have been so much difference in the accuracy.

3. While we don't know the attributes in this case, the KSVM model may not be a sensible model here, (and I may be wrong since I haven't cross validated test and train data), but after testing the same data with different test-train ratios, and cross validation with KNN (the next problem), I found most of the results to be much better. And it brings me to conclude that comparing a customer with customers around based on the distances in each feature, rather than comparing the customer to a LINE based on a few peripheral customers, may be more realistic and personalized for the customer in question, and for deciding whether to issue credit.

QUESTION 3 - (Please scroll down to find the discussion)

KNN SOLUTION ~

1. Including all the attributes -

Checking from k = 1 to 30;

the maximum accuracy ~ 84.77157

the k value ~ 20

2. Few of the better attribute sets -

Set1 = (3,4,5,6,7,8,9)

Checking k from 1-30;

the maximum accuracy ~ 88.32487

the k value ~ 2

Set2 = (3,4,5,6,7,9)

Checking k from 1-30;

the maximum accuracy ~ 88.32487

the k value ~ 3

Set3 = (3,5,7,9)

Checking k from 1-30;

the maximum accuracy ~ 88.32487

the k value ~ 13

Note - Note that for some cases -

a. You can remove attributes and it affects accuracy irreparably.

b. You can remove the attributes but you must adjust k by increasing it considerably (like for set3).

c. You can remove the attributes but you need adjust k by increasing it negligibly (like for set1 and set2).

3. Attributes 1,2, and 10, tend to skew the results and must not be as causal to the results as the rest.

In [ ]:
```
library(ISLR)
library(ggplot2)
library(reshape2)
library(plyr)
library(dplyr)
library(class)
library(combinat)
```

In [ ]:
```
#code - normalize and divide into test-train

normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x))) }


dt <- as.data.frame(lapply(credit_card_data[,c(3,5,6,7,9)], normalize))

set.seed(123)
dat <- sample(1:nrow(dt),size=nrow(dt)*0.7,replace = FALSE) #random selection of 70% data.

traina <- dt[dat,] # 70% training data
testa <- dt[-dat,] # remaining 30% test data

#Creating seperate dataframe for our target.
trainl <- credit_card_data[dat,11]
testl <- credit_card_data[-dat,11]

#taking square root to get the looping limits that follow in the next cell
sq <- NROW(trainl)**0.5
sq1=as.integer(sq)
sq2=sq1+1
```

In [ ]:
```
#code - experimenting with k

maxi = 0
k = 0
for(i in 1:28){
  knn1 <- knn(train=traina, test=testa, cl=trainl, k=i) #varying neighbouring number of points
  ACC1 <- 100 * sum(testl == knn1)/NROW(testl)

  print(ACC1)
  print(i)
  if(ACC1>maxi){
    maxi = ACC1 #finding highest accuracy
    k = i
  }
}
print(maxi)
print(k)
```

Discussion for KNN results

1. Unlike KSVM, the set of attributes does make a difference to the accuracy of the model. Which implies that some features for deciding credit are not as important as the others are. The features 1,2 and 10 do not play a bif role in detrmining credit.

2. The chances that a customer will pay off his credit debt is better determined with very specific attributes and with a value of k<5, rather than with a lot of attributes. And this shows that there may be a very strong relationship between a few attributes and target that could further be studied in terms of the domain to improve prediction and reduce errors.

EXTRA-CURRICULAR STUDIES -

From the sector that determines whether to issue credit to customers or not, the following are the most salient features obtained from a brief research -

1. Annual income
2. Job consistency
3. Credit interest
4. Personal living costs
5. Macroeconomic factors
6. Age in case of unemployment
7. Income history in case of unemployment
8. Healthcare and education debt status

THE END