Network Date Homework 7

Jiaying Liu

December 3, 2018

Jaccard coefficient

To predict the status of edge, we can use the scoring method to predict. to fin the score method, we can use different kind of measurment. In the following section, I will use the jaccard coefficient to define the score.

For the jaccard coeffieient, it is the interect of i and j neighborhood divide by the union of i and j neighborhood. The equation is

$s(i,j) = |N_i^{obs} \cap N_j^{obs}|/|N_i^{obs} \cup N_j^{obs}|$

To note that, when we use the neighborhood function, it is actually include the vertex of i and j itself, so when we calcaute the number of interect node, we need to delete it. However when we calcaute the number of union, the situation change. we need to subtract 2 only for those pairs haven't been connected, because the union function has already deleted 2 duplicated items for node i and node j.

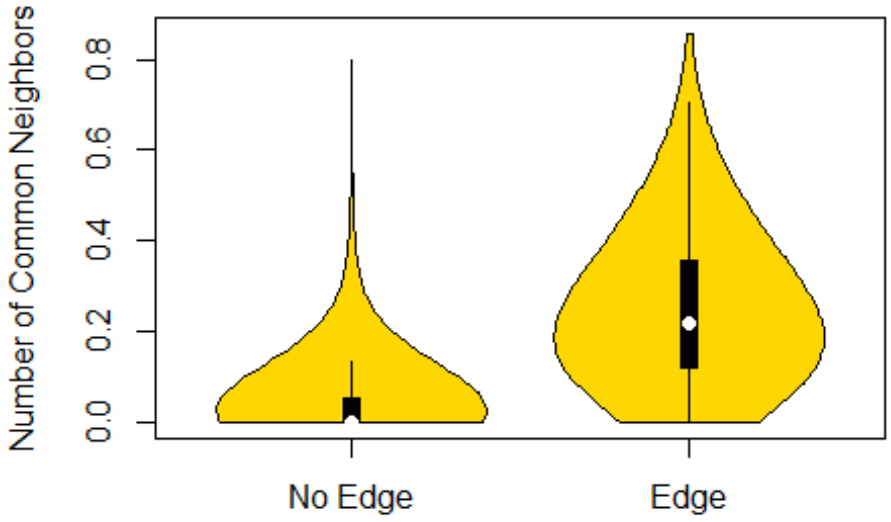
With the knowledge about this method, we begin to use R to calculate it.

```
nv <- vcount(fblog)
ncn <- numeric()
fblog2=upgrade_graph(fblog)
A <- get.adjacency(fblog2)
for(i in (1:(nv-1))){
 ni <- neighborhood(fblog2, 1, i)
 nj <- neighborhood(fblog2, 1, (i+1):nv)
 nbhd.ij1 <- mapply(intersect, ni, nj, SIMPLIFY=FALSE)
 nbhd.ij2 <- mapply(union, ni, nj, SIMPLIFY=FALSE)
 temp <- (unlist(lapply(nbhd.ij1, length))- 2* A[i, (i+1):nv])/(unlist(
lapply(nbhd.ij2, length))- 2*abs( A[i, (i+1):nv]-1))

 ncn <- c(ncn, temp)
}
```

In the following section, we create the violin plot for the score method we create above. For the violin plot, a good plot is that for two main part of no edge or edge should have a shreahold that can seperate it clearly. And in the following figure, we can see that it separete well for two differnt status.

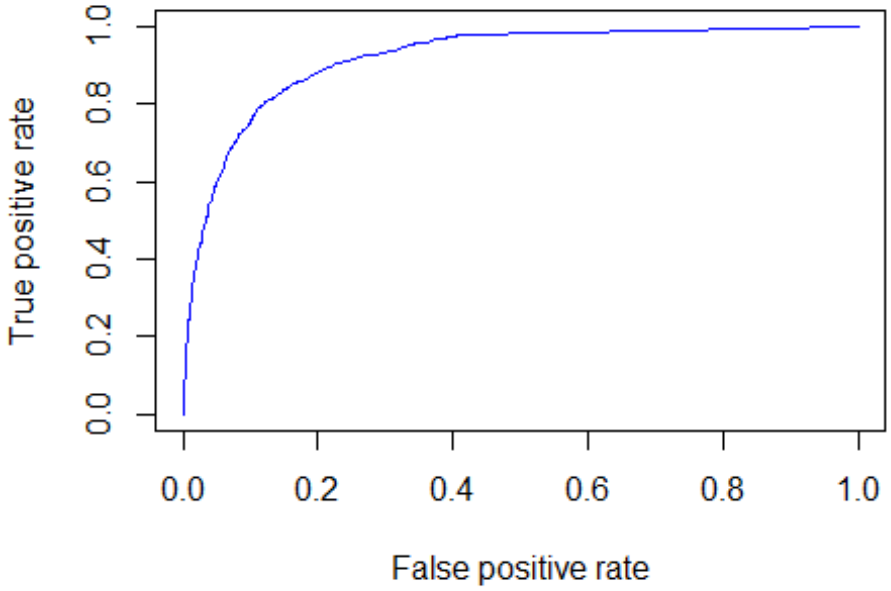
```
Avec <- A[lower.tri(A)]

## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
vioplot(ncn[Avec==0], ncn[Avec==1],
   names=c("No Edge", "Edge"),col='gold')
title(ylab="Number of Common Neighbors")
```



And we can also use the ROC curve to evaluate the preformation of jaccard coefficient. And we can see that for this method, the AUC is 0.91774. It is a very good value of AUC.

```
pred <- prediction(ncn, Avec)
perf <- performance(pred, "auc")
roc.perf=performance(pred,measure = "tpr", x.measure = "fpr")
slot(perf, "y.values")

## [[1]]
## [1] 0.91774

plot(roc.perf,col='blue')
```



Weighting more heavily those common neighbors not highly connected

In this section, we will use another method to calcuate the score.

Weighting more heavily those common neighbnors not highly connected. The equaltion is that

$$s(i,j) = \sum^{k \in N_i^{obs} \cap N_j^{obs}} \frac{1}{log|N_k^{obs}|}$$

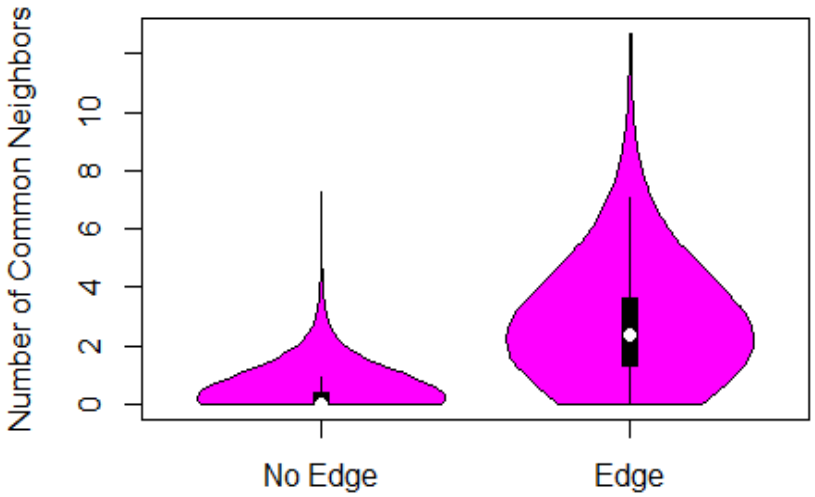
To note that, in the above method, we use the neighborhood statement which will include the node itself. However this is not the statement we want. K is belong to the interect of neighborhoods of i and j, but we need to delete the i and j itself in case k have the value of i and j. So to solve this problem, rather than use the neighborhood statement, we turn to use NEIGHBORS statement which would not include the node itself. Then it perfectly solve our problem. The other is just same as above. summation of inverse log number of neighbor in k node. And the code would show below.

```
nv <- vcount(fblog)
ncn <- numeric()
fblog2=upgrade_graph(fblog)
A <- get.adjacency(fblog2)
```

```
for ( i  in (1:(nv-1))){
  ni=neighbors(fblog2,i)
  ncn_1=numeric()
    for ( j in (i+1):nv){
      nj=neighbors(fblog2,j)
      in_ij=intersect(ni,nj)
      score=numeric()
      for (k in in_ij){
        nk=neighbors(fblog2,k)
        len=1/log(length(nk))
        score=c(score,len)
      }
      ncn_1=c(ncn_1,sum(score))
    }
  ncn=c(ncn,ncn_1)
}
```

For the violin plot, we can see that the plot also sepreate the two status well

```
#library(vioplot)
Avec <- A[lower.tri(A)]

## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient

vioplot(ncn[Avec==0], ncn[Avec==1],
   names=c("No Edge", "Edge"))
title(ylab="Number of Common Neighbors")
```



For the ROC curve and the AUC value, We find out the value is 0.933 which is better than jaccard coefficient.

```
pred <- prediction(ncn, Avec)
perf <- performance(pred, "auc")
roc.perf=performance(pred,measure = "tpr", x.measure = "fpr")
slot(perf, "y.values")

## [[1]]
## [1] 0.9331702

plot(roc.perf,col='blue')
```