Greedy Algorithm for Community Detection

Arnab Banerjee

August 29, 2019

1 Algorithm

Algorithm 1 Community Detection in Dynamic Networks

procedure GREEDYCOMDEC(A,DensityRatio)
▷ Here A is an array of T adjacency matrices and DensityRatio is our hyperparameter
nCom,comAssign,comDensity ← PerformSpectralClustering(A,DensityRatio)
nCom,comAssign,comDensity ←PerformIterations(nCom,comAssign,A,DensityRatio,comDensity)
return $nCom, comAssign, comDensity$
▷ The final output

Algorithm 2 Initial Spectral Clustering Step

procedure PERFORMSPECTRALCLUSTERING(A,DensityRatio)
▷ This procedure performs the initial spectral clustering task
$T \leftarrow$ Number of graphs
$N \leftarrow$ Number of nodes in each graph
$kInit \leftarrow InitialCommunities(A, N, T)$
$nCom \leftarrow kInit$
for each matrix A_t in A **do**
$comAssign \leftarrow SpectralClustering(A_t, nCom)$
$comDensity \leftarrow GetDensity(nCom[t], comAssign, A_t)$
for each matrix A_t in A **do**
for each node n in A_t **do**
$maxClass \leftarrow -1$
$max \leftarrow DensityRatio$
for each community c in nCom[t] **do**
$AdjCnt \leftarrow$ Count n's adjacent nodes belonging to community c
if comDensity[t,c]!=0 and AdjCnt/comDensity[t,c]≥max **then**
$max \leftarrow AdjCnt/comDensity[t, c]$
$maxClass \leftarrow c$
$comAssign[t, n] \leftarrow c$
else if Number of vertices belonging to community c equal 1 and 2*AdjCnt≥max **then**
$max \leftarrow 2 * AdjCnt$
$maxClass \leftarrow c$
$comAssign[t, n] \leftarrow c$
if maxClass=-1 **then**
$comAssign[t, n] \leftarrow 1 + max(nCom[t])$
$nCom[t] \leftarrow nCom[t] + 1$
$comDensity[t] \leftarrow GetDensity(nCom[t], comAssign, A_t)$
return $nCom, comAssign$
▷ The final output

Algorithm 3 Greedy Algorithm

procedure InitialCommunities(A,N,T) ▷ This procedure finds out the initial number of communities
    $K \leftarrow []$
    for each matrix A_t in A **do**
        $M = \begin{bmatrix} 0 & D-I \\ I & A \end{bmatrix}$
        ▷ Here D is a diagonal matrix of degree of nodes
        $k_t \leftarrow$ Number of eigen values of B greater than square root of B's norm
        $K \leftarrow KUk_t$
    return K ▷ Returns array of initial number of communities

Algorithm 4 Greedy Algorithm

procedure GetDensity(numCom,ComAssign,A_t) ▷ This procedure finds out the density of the communities of A_t
    $Edges \leftarrow$ Total edges belonging to each community in A_t
    $Vertices \leftarrow$ Total vertices belonging to each community in A_t
    for each c in numCom **do**
        if Edges[c]=0 **then**
            Density[c]=0
        else
            Density[c]=Edges[c]/Vertices[c]
    return $Density$ ▷ Returns array of community densities of A_t

Algorithm 5 Greedy Algorithm

procedure PerformIterations(numCom,ComAssign,A,DensityRatio,comDensity) ▷ This procedure performs iterations over the graphs
    $itr \leftarrow 1$
    $maxItr \leftarrow 3$
    while $itr <= maxItr$ **do**
        for each A_t in A **do**
            $comAssign, comDensity \leftarrow NewAssignments(A_t, numCom[t], ComAssign, t, DensityRatio, comDe$
    return $numCom, ComAssign, comDensity$ ▷ Returns density

Algorithm 6 Greedy Algorithm

procedure NEWASSIGNMENTS(A_t,k,ComAssign,t,DensityRatio,comDensity,T) ▷ This procedure performs iterations over the graphs
  for each node n in A_t **do**
    for each community c in k **do**
      if t>0 **then**
        $AdjCnt1 \leftarrow$ Number of edges with vertices of graph A_t which belonged to community c in time t-
        if ComDensity[t-1,k]!=0 **then**
          $parameter1 \leftarrow AdjCnt1/ComDensity[t-1,k]$
        else
          $parameter1 \leftarrow 2 * AdjCnt1$
      if t<T **then**
        $AdjCnt2 \leftarrow$ Number of edges with vertices of graph A_t which belonged to community c in time t-
        if ComDensity[t+1,k]!=0 **then**
          $parameter2 \leftarrow AdjCnt2/ComDensity[t+1,k]$
        else
          $parameter2 \leftarrow 2 * AdjCnt2$
      $AdjCnt3 \leftarrow C$Number of edges with vertices of graph A_t which belonged to community c in time t
      if ComDensity[t,k]!=0 **then**
        $parameter3 \leftarrow AdjCnt3/ComDensity[t,k]$
      else
        $parameter3 \leftarrow 2 * AdjCnt3$
      if parameter1≥DensityRatio and parameter1≥maxParam1 **then**
        $maxParam1 \leftarrow parameter1$
        $maxClass1 \leftarrow c$
      if parameter2≥DensityRatio and parameter2≥maxParam2 **then**
        $maxParam2 \leftarrow parameter1$
        $maxClass2 \leftarrow c$
    if maxClass1!=-1 and maxClass1=maxClass2 **then**
      $comAssign[t,n] \leftarrow maxClass1$
      $ComDensity[t] \leftarrow GetDensity(k, comAssign[t], A_t)$
    else if maxClass1!=maxClass2 and comAssign[t,n]!=maxClass1 and comAssign[t,n]!=maxClass2 **then**
      $comCon \leftarrow GetClassOnCondition(parameter1, parameter2, parameter3, maxClass1, maxClass2, c$
      $comAssign[t,n] \leftarrow comCon$
      $comDensity[t] \leftarrow GetDensity(k, comAssign[t], A_t)$
  return $k, ComAssign, comDensity$ ▷ Returns density

Algorithm 7 Greedy Algorithm

procedure GETCLASSONCONDITION(par1,par2,par3,c1,c2,c3)▷ This procedure finds out which class has the maximum value of the parameter
  $paralist \leftarrow [par1, par2, par3]$
  $classlist \leftarrow [c1, c2, c3]$
  return $classlist[paralist.argsort()[paralist.length - 1]]$