Normalized-Cuts Algorithm

The normalized cuts algorithm can cluster a set of elements based only on the values of a similarity measure between all possible pairs of elements. The approach is a spectral clustering method. It’s based on the propertied of eigenvectors from a matrix computed using the similarities between each pairs of elements. In the normalized cuts scheme, the clustering is seen as a graph partitioning problem. The nodes of the graph are the elements and the weights on the graph edges connecting two nodes are the similarities measured between the two corresponding elements. In the process, we seek to partition the graph into subgraphs with high similarities between the nodes from the same subgraphs and low similarities between nodes from different subgraphs.

**How this algorithm works:**

Let G (V, E) be the graph with nodes V and an adjacent graph weight matrix W. The element w (u, v) of W if the similarities measured between the elements represented by the nodes u and v. We can break G into two disjoint sets A and B so that and by removing the edges connecting the two parts. The cost of removing those edges is computed as the total weight of edges that have been removed.

This cost is called a “cut” in the graph theory. We want to minimize this cut value when partitioning G. However, minimizing the cut value encourages cutting isolated nodes in G. To avoid that problem, the cost of removing edges from A to B is considered as a fraction of the total weight of connections from nodes in A to all nodes in the graph. This leads to the new cost measure called the normalized cut:

Where is the total connection weight from all nodes in A to all nodes in V. asso(B,V) is defined in the similar way. Our aim is to find a partition of G that minimize the normalized cut between the two parts.

**How to find the solution:**

Let x be a dimensional vector, that represents the partition of V into two sets A and B. if the node i is in A and if the node i is in B. We define:

Let the total weight of edges between the node i and all the other nodes in the graph. Let and let . It is proven in [87] that:

Subject to the constraints and . The change of variable: has been used. So if the node i is in A and if the node i is in B.

is called the Rayleigh quotient. The minimization of equation can be approximate by relaxing y to take real values and solving the generalized eigenvalue system:

. Equation can be rewritten as a standard eigensystem: = with .

By observing the following:

is a symmetric diagonally dominant real matrix with non-negative diagonal entries so it’s semidefinite; it has therefore only positive eigenvalues;

is an eigenvector of ; its corresponding eigenvalue is 0;

is symmetric so its eigenvectors are orthogonal to each other, in particular the eigenvector that corresponds to the second smallest eigenvalue is orthogonal to and thus satisfy the constraint with .

We can use the following theorem[63]: Let A be the real symmetric matrix with eigenvalues . The Rayleigh quotient is minimized by the eigenvector, under the constraint that x is orthogonal to the eigenvector associated with the j-1 smallest eigenvalues of A. The minimum is equal to the

In our case, we can conclude that minimizes under the constraint . So minimizes under the constraint

We need only to compute the eigenvector associated with the second smallest eigenvalue. We use the Lanczos algorithm to compute that vector. This algorithm iteratively approximates the eigenvectors until convergence [36]. In contrast to singular value decomposition, only two eigenvectors have to be computed to get the eigenvector we are looking for. This makes the Lanczos algorithm quicker, especially for large matrices.

The solution of the relaxed problem can be used as an approximate solution to the original discrete problem. We need an extra condition on the components of the solution: . In the ideal case, the components of the eigenvector take only two discrete values, which represents the two classes. If it is not the case, we need to choose a splitting point to partition the values of the components of the eigenvector. Here we simply choose 0 as a splitting point. All the positive components represent elements from one class and all the negatives ones represent elements from the other class. Other methods exist for choosing the splitting point [89].

We can then separate the set of elements into two sets, one containing the elements corresponding to positive values in the eigenvector and the other one containing elements corresponding to negative values in the eigenvector. We recursively cluster the resulting two element sets until we reach a given number of clusters.

This number of resulting clusters is usually not a power of two as one would expect given our description of the algorithm. This is due to the fact that the normalized cuts algorithm does not always separate the groups into two for each step. It is possible that the second eigenvector given by the Lanczos algorithm has only positive or only negative values. This can be due either to rounding errors in the computation of the eigenvector or to cases where the choice of 0 as a threshold value for selecting the two groups is a bad choice (the solution of the normalized cuts algorithm is only an approximation of the continuous case). It can also be due to the fact that the group we are trying to split only contains one element (due to previous splits). In that case, we do not split the group so the count of the number of group will not be a power of two.