Initialization Explanation

Shizhe Liang

2020-9-28

Roughly speaking, multi-view clustering differ from traditional clustering in the sense that, instead of looking at one representation of the data, we are analyzing the data from multiple perspectives (views) and aiming to produce a clustering that is compatible with all views.

Recall that for traditional K-Means clustering, if X is the (single-view) data matrix, with each row vector corresponding to a data point, G is the cluster indicator matrix and F is the centroid matrix, with each row representing a centroid, then we are essentially solving the optimization problem

$$\min_{G,F} ||X - GF||_F^2$$

where $||...||_F^2$ is the frobenius norm.

For multi-view K-Means clustering, we have M data matrices $X_1, ..., X_M$, one for each view. The $X_v's$ have the same number of rows (the number of data points), but the number of columns can be different. We also have M centroid matrices $F_1, ..., F_M$, one for each view. However, we need a common cluster indicator matrix G for all views. In addition, we introduce a weight vector $\alpha = (\alpha_1, ..., \alpha_M)$. Intuitively, we want the "good" views to have larger weights and "non-informational" views have lower weights. For the RMKMC algorithm, we are looking at the following optimization problem

$$\min_{G,(F_v)_v,(\alpha_v)_v} \sum_{v=1}^{M} ||X_v - GF_v||_{2,1}$$

where $||...||_{2,1}$ is the matrix (2,1)-norm.

The above optimization problem is not convex, but if we fix any two of the three variable $\{G, (F_v)_v, (\alpha_v)_v\}$, the induced problem on the remaining variable is convex. Indeed, in the original paper, the algorithm starts with a randomly initialized G and uniformly initialized $(\alpha_v)_v$, then each optimization iteration fixes the other two variables and optimizes $(F_v)_v, G, (\alpha_v)v$ sequentially.

The key observation that gives rise to my K-Means++ style initialization is that we can also start with any $(F_v)_v$ and uniformly initialized $(\alpha_v)_v$ then run the optimization iterations, with the ordering of variables changed. In particular, my initialization will invoke ordinary K-Means++ initialization on each view to produce the starting $(F_v)_v$, then the rest of the algorithm will run as smoothly.

Testing on UCI's handwritten numerals data set shows that this initialization indeed outperforms the random initialization.