Week 14

Last week, I read some papers about Non-negative Matrix Factorization. Because my course project is about Semi-supervised Network Representation Learning based on Multi-view Dataset, I should study the articles about multi-view clustering or embedding such as:

- Paper One: Relational Learning via Collective Matrix Factorization
- Paper Two: Multi-view Clustering via Joint NMF

Existing multi-view clustering algorithms can be roughly classified into three categories:

1) Optimize certain loss function in which multi-view information has been integrated.
2) Project multi-view data into a lower dimensional space and then cluster them through conventional clustering method.
3) Cluster each single-view data and get many corresponding results, and then fuse different clustering results into single one result.

The algorithms in these two papers are different. The former one is trying to integrating multi-view data directly, which belongs to 1), while the latter one utilizes a method called late fusion during the learning process, which belongs to 3).

Paper One

Relational Learning via Collective Matrix Factorization

Motivation

Relational Learning aims to predict the unknown values of a relation. We usually handle this issue by learning embedding factors. In a domain of real world with multiple relations, we can utilize one to predict another. That's the multi-view embedding issue. In this paper, authors give us their ideas to solve it: simultaneously learn the latent factors into the same latent representation matrix.

Related Theories

Since there may be different value types and error distributions, authors use Bregman divergences to measure error, allowing nonlinear relationships between the parameters and outputs.

Bregman divergence: $$D_F(p.q) = F(p) - F(q) - \langle \nabla F(q), (p-q) \rangle$$

Where F is a convex two-order differentiable function, $\nabla F(q)$ represents the gradient of F at q, < > represents the dot product operation.

Thus, the $L(p,q) = F(q) + \langle \nabla F(q), (p-q) \rangle$ term in the above formula, indicates the linear performance of F at q.

Main Theory

Using the shared coefficient matrix but different basis matrices across views as shown below:

$$\sum_{v=1}^{n_v} \lambda_v \|X^{(v)} - U^{(v)}(V^{(*)})^T\|_F^2.$$

Authors use stochastic optimization methods to solve the above objective function, as well as handle with large, sparse datasets. Moreover, by deriving Newton update for the projection, the optimization becomes more efficient. But since the optimization method has no relations with my course project design, so I do not record them here.

From the above objective function, we can clearly see that the method in this paper (often called as Collective NMF) is trying to embed the latent information into V directly through the proper objective function. But the next algorithm proposed in paper two is another classification of multiview methods.

Paper Two

Multiview Clustering via Joint NMF

Motivation

A great number of real-world datasets are naturally comprised of many views. Therefore, some clustering algorithms working on single-view cannot work on multiple-view dataset directly. The authors want to promote the classic NMF method such that it adapt on multi-view datasets. They propose an effective method utilizing NMF and PLSA algorithms, which can give us a meaning and comparable clustering solutions.

Related Theories

PLSA: A kind of Topic model, closely related to NMF. It can give a better probability expression to this model.
Specifically, PLSA is a topic model, considering that God chooses a subject in a certain probability when writing an article, and then chooses a word in a certain probability under the subject, and repeats the process to complete an article:

$$p(d_i, w_j) = p(z_k|d_i)p(w_j|z_k)$$

Where d, w, z indicates documents, words, and topics.

Therefore, the solution to PLSA model is to compute $p(z_k|d_i)$ and $p(w_j|z_k)$. The classic method is EM algorithm:
EM algorithm is an iterative algorithm which contains E-step and M-step.
E-step:

Firstly, compute $p(z_k \mid d_i, w_j)$ according to the following formula. $p(z_k \mid d_i)$ and $p(w_j \mid z_k)$ are obtained from the last M-step. The initial values of them are randomly given.

$$p(z_k|d_i, w_j) = \frac{p(z_k|d_i)p(w_j|z_k)}{\sum_k p(z_k|d_i)p(w_j|z_k)}$$

M-step:

$p(z_k \mid d_i)$ and $p(w_j \mid z_k)$ are obtained according to the following formulas, where $p(z_k \mid d_i, w_j)$ is calculated in the E-step.

$$p(z_k|d_i) = \frac{\sum_j p(z_k|d_i, w_j)}{\sum_j \sum_k p(z_k|d_i, w_j)} \quad p(w_j|z_k) = \frac{\sum_i p(z_k|d_i, w_j)}{\sum_i \sum_j p(z_k|d_i, w_j)}$$

Main Theory

For traditional NMF algorithm, V_j^v is the low dimensional expression of j node based on basic matrix U^v. And the measure of coefficient matrix V^v and consensus matrix V^* is defined as: $D(V^{(v)}, V^*) = ||V^{(v)} - V^*||_F^2$

Since then, our overall goal should be:

$$\sum_{v=1}^{n_v} ||X^{(v)} - U^{(v)}(V^{(v)})^T||_F^2 + \sum_{v=1}^{n_v} \lambda_v ||V^{(v)} - V^*||_F^2$$

With the constraints such that, $\forall 1 \le k \le K, ||U_{.,k}^{(v)}||_1 = 1 \text{且} U^{(v)}, V^{(v)}, V^* \ge 0$

Moreover, authors also introduce Q:

$$Q^{(v)} = Diag(\sum_{i=1}^{M} U_{i,1}^v, \sum_{i=1}^{M} U_{i,2}^v, \ldots, \sum_{i=1}^{M} U_{i,K}^v)$$

in order to constraint U.

Therefore, the final objective function is supposed to be:

$$O = \sum_{v=1}^{n_v} ||X^{(v)} - U^{(v)}(V^{(v)})^T||_F^2 + \sum_{v=1}^{n_v} \lambda_v ||V^{(v)}Q^{(v)} - V^*||_F^2$$

Optimization:

In order to minimize the above objective function, and get the consensus matrix V^*, basic matrices $\{U^{(1)}, U^{(2)}, \ldots, U^{(n_v)}\}$ and coefficient matrices $\{V^{(1)}, V^{(2)}, \ldots, V^{(n_v)}\}$, we firstly fix V^* to update U^v and V^v, then fix U^v and V^v to update V^*:

Input: Nonnegative matrix $\{X^{(1)}, X^{(2)}, ..., X^{(n_v)}\}$, parameters $\{\lambda_1, \lambda_2, ..., \lambda_{n_v}\}$, number of clusters K

Output: Basic matrices $\{U^{(1)}, U^{(2)}, ..., U^{(n_v)}\}$, coefficient matrices $\{V^{(1)}, V^{(2)}, ..., V^{(n_v)}\}$ and consensus matrix V^*

1. Normalize each view $X^{(v)}$ such that $\left\|X^{(v)}\right\|_1 = 1$
2. Initialize each $U^{(v)}$, $V^{(v)}$ and V^*
3. Repeat

   For v=1 to n_v do

   Repeat

   - Fix V^* and $V^{(v)}$, update $U^{(v)}$:

$$U_{i,k} \leftarrow U_{i,k} \frac{(XV)_{i,k} + \lambda_v \sum_{j=1}^{N} V_{j,k} V^*_{j,k}}{(UV^TV)_{i,k} + \lambda_v \sum_{l=1}^{M} U_{l,k} \sum_{j=1}^{N} V^2_{j,k}}.$$

   - Normalize $U^{(v)}$ and $V^{(v)}$ as:

$$U \leftarrow UQ^{-1},\ V \leftarrow VQ.$$

   - Fix V^* and $U^{(v)}$, update $V^{(v)}$:

$$V_{j,k} \leftarrow V_{j,k} \frac{(X^TU)_{j,k} + \lambda_v V^*_{j,k}}{(VU^TU)_{j,k} + \lambda_v V_{j,k}}.$$

   Until $\|X - UV^T\|_F^2 + \lambda_v \|VQ - V^*\|_F^2$ converge.

   End for

   - Fix $U^{(v)}$ and $V^{(v)}$, update V^*:

$$V^* = \frac{\sum_{v=1}^{n_v} \lambda_v V^{(v)} Q^{(v)}}{\sum_{v=1}^{n_v} \lambda_v} \geq 0.$$

Until $O = \sum_{v=1}^{n_v} ||X^{(v)} - U^{(v)}(V^{(v)})^T||_F^2 + \sum_{v=1}^{n_v} \lambda_v ||V^{(v)}Q^{(v)} - V^*||_F^2$ converge.

Summary

The algorithm can produce a Interpretable result, where each element in V^* can be

viewed as sum of $P(d|k)^{(v)}$ with the weight of $P(d)^{(v)}$. What's more, V^* can be seen as a network representation, such that we can utilize some classic machine learning algorithm.