Local PurTree Subspace Spectral Clustering for Customer Transaction Data

Xiaojun Chen[1], Guowen Yuan[1], JianZhe Zhang[1], Joshua Zhexue Huang[1], and Qingyao Wu[2⋆]

[1] College of Computer Science and Software, Shenzhen University, Shenzhen, 518060, P.R. China,
xjchen@szu.edu.cn, 243864952@qq.com, 353699875@qq.com, zx.huang@szu.edu.cn
[2] School of Software Engineering, South China University of Technology, Guangzhou, P.R. China,
qyw@scut.edu.cn

Abstract. Clustering of customer transaction data is very important in retail and e-commerce companies. In this paper, we propose a local PurTree Subspace Spectral (LPSS) clustering algorithm for customer transaction data. In the new method, a recently proposed "Purchase Tree" is used to represent the customer transaction data. We propose a PurTree subspace distance to compute the distance between two trees, in which a set of sparse node weights are introduced to distinguish the importance of different nodes in a purchase tree. The new method learns a data similarity matrix from the local distances and a set of sparse node weights in the PurTree subspace distance simultaneously. An iterative optimization algorithm is proposed to optimize the proposed model. We also present an effective method to compute a regularization parameter in LPSS. We compare LPSS with five clustering algorithms on 10 benchmark data sets and the experimental results show the superiority of the new method.

Keywords: Clustering, customer segmentation

1 Introduction

Transaction data is the collection of daily shopping transactions of customers at a retail company. A transaction is a sequence of products (items) bought by a customer in one basket. Clustering of customer transaction data is one of the most critical tasks in successful modern marketing and customer relationship management [2, 16]. It is used to categorize customers into different groups based on their purchase behaviors, so that the customers in the same cluster bought more similar goods to each other than to those in other clusters. The early segmentation methods use general variables like customer demographics, lifestyle, attitude and psychology, because such variables are intuitive and easy to operate [12]. With the rapid increase of customer behavior data, the new study

⋆ Corresponding author.

turns to use product specific variables such as items purchased [21, 18, 13, 22]. Although some methods which cluster item set data were proposed [20–22], these methods suffer from the huge amount of transaction records. On the other side, most work employed hierarchical agglomerative clustering algorithm which are not scalable to large-scale transaction data [20–22, 10]. Over the past decades, many clustering algorithms have been proposed, such as spectral clustering [19], subspace clustering [11, 7], support vector clustering [1], multi-view clustering [4, 6], etc. However, few methods have been used for clustering transaction data.

Recently, a PurTreeClust clustering algorithm was proposed for large-scale transaction data [5]. In this algorithm, a product tree is first built from the transaction data. In the product tree, the leaf nodes denote products and the internal nodes represent product categories. A purchase tree is built for each customer, in which each product (item) bought by a customer corresponds to a leaf node in the product tree. A PurTree distance metric was defined to measure the dissimilarity between two purchase trees, and a PurTreeClust algorithm was proposed to cluster purchase trees. The PurTreeClust algorithm first builds a cover tree for indexing the purchase tree data set, then selects initial cluster centers with a fast leveled density estimation method. Finally, the clustering result is produced by assigning each customer to the nearest cluster centers. However, PurTreeClust has three shortcomings: 1) the node weights in the PurTree distance are set as equal values for a parent node's children nodes. However, since a product tree often consists of hundreds of thousands nodes, it is desired to learn a set of sparse node weights such that only a few important nodes are considered for computing the distance, and 2) it is difficult to set proper parameter for the PurTree distance, and 3) it lacks of optimization method to optimize the clustering result.

In this paper, we propose LPSS, a local PurTree Subspace Spectral clustering algorithm to solve the above three shortcomings. A PurTree subspace distance is proposed to compute the distance between two trees, in which a set of sparse node weights are learnt. The new model learns a local similarity matrix and a set of sparse node weights simultaneously. We present an iterative optimization algorithm to optimize the proposed model, and an effective method to compute a proper regularization parameter. A series of experiments were conducted on 10 benchmark data sets and the experimental results show the superior performance of LPSS.

The rest of this paper is organized as follows. Notations and preliminaries are given in Section 2. We propose the PurTree subspace distance in Section 3, and the LPSS algorithm in Section 4. The experimental results and analysis are presented in Section 5. Conclusions and future work are given in Section 6.

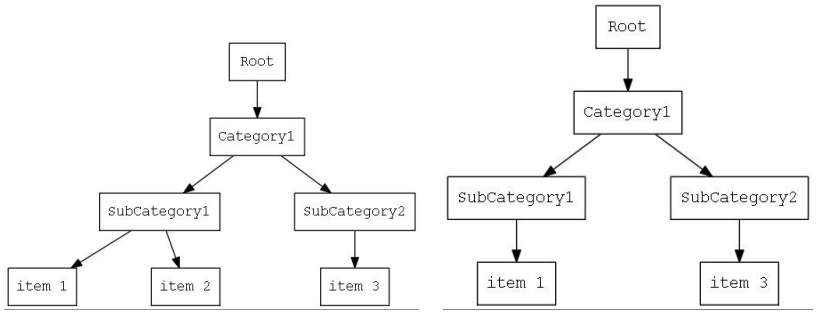
2 Notations and Preliminaries

In this section, we introduce the notations and preliminaries used in [5].

2.1 Notations

Let T be a rooted tree with nodes $N(T)$ and edges $E(T) \subseteq N(T) \times N(T)$. The root of T is denoted by $root(T)$. A node without children is a leaf, and otherwise an internal node. For an edge $(u, v) \in E(T)$, node u is the parent and v is a child, $P_v(T) = u$ and $v \in C_u(T)$. The descendants of v, the nodes in all paths reachable from v to a leaf node, is denoted as $des(v)$. The level of a node is defined by 1 + (the number of edges between the node and the root). $N^l(T)$ represents nodes in the l-th level of T. The height of tree T, denoted by $H(T)$, is the number of edges on the longest downward path between the root and a leaf node.

2.2 Product Tree and Purchase Tree



(a) A sample of product tree. (b) A sample of purchase tree.

Fig. 1. Samples of product tree and purchase tree.

Let Ψ be a rooted tree used to systematically organize the items with multiple levels of categories, in which each leaf node represents an item and each internal node represents a category. All leaf nodes in Ψ are assumed to have equal depth in [5]. A purchase tree φ is used to illustrate the items bought by a customer, which is a subgraph of Ψ, i.e., $N(\varphi) \subseteq N(\Psi)$, $E(\varphi) \subseteq E(\Psi)$. Given a leaf node $u \in N(\varphi)$, the path from $root(\varphi)$ to u also exists in Ψ. For each purchase tree φ, we have $H(\varphi) = H(\Psi)$. Figure 1(b) shows a sample of purchase tree, which is a subset of the product tree in Figure 1(a).

2.3 PurTree metric distance

Given a product tree Ψ and a set of n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$ where $\varphi_i \in \Psi$, let $H(\Phi)$ be the height of these purchase trees, $root(\varphi)$ be the empty root node of the purchase tree. The PurTree distance is defined as follows [5]

Definition 1 *Given a product tree Ψ and n purchase trees $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ where $\varphi_i \in \Psi$, the PurTree distance between φ_i and φ_j is defined as*

$$d(\varphi_i, \varphi_j) = \sum_{l=1}^{H(\Phi)} \beta_l \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \alpha_v \delta_v(\varphi_i, \varphi_j) \tag{1}$$

where $\delta_v(\varphi_i, \varphi_j)$ is the Jaccard distance of φ_i and φ_j on an internal node v, which is defined as

$$\delta_v(\varphi_i, \varphi_j) = 1 - \frac{|C_v(\varphi_i) \bigcap C_v(\varphi_j)|}{|C_v(\varphi_i) \bigcup C_v(\varphi_j)|} \tag{2}$$

and α_v is the node weight for node $v \in N(\Psi)$ which is defined as

$$\alpha_v = \begin{cases} 1 & \text{if } v = root(\Psi) \\ \frac{\alpha_u}{|C_u(\varphi_i) \bigcup C_u(\varphi_j)|} & \text{where } v \in C_u(\varphi_i) \bigcup C_u(\varphi_j) \end{cases} \tag{3}$$

and β_l is the l-th level weight. $\{\beta_1, ..., \beta_{H(\Phi)}\}$ is a geometric sequence with common ratio γ ($\gamma >= 0$) under constraint $\sum_{l=1}^{H(\Phi)} \beta_l = 1$, defined as

$$\beta_l = \begin{cases} \frac{1-\gamma}{1-\gamma^{H(\Phi)}} \gamma^{l-1} & \text{for } \gamma > 0 \text{ and } \gamma \neq 1 \\ \frac{1}{H(\Phi)} & \text{for } \gamma = 1 \\ 1 & \text{for } \gamma = 0 \text{ and } l = 1 \\ 0 & \text{for } \gamma = 0 \text{ and } 1 < l \leq H(\Phi) \end{cases} \tag{4}$$

3 PurTree Subspace Distance

In the PurTree distance defined in Eq. (1), the node weights α are set as fixed values and a parameter γ is used to control the level weights β. However, since a product tree often consists of hundreds of thousands nodes, it is desired to learn a set of sparse node weights such that only a few important nodes are considered for computing the distance. Moreover, it is difficult to set proper γ. In this paper, we propose a new PurTree subspace distance defined as follows

Definition 2 *Given a product tree Ψ and a set of n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$, $\varphi_i \in \Psi$, the PurTree subspace distance between two purchase trees φ_i and φ_j is defined as*

$$d(\varphi_i, \varphi_j, \Omega) = \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \omega_v \delta_v(\varphi_i, \varphi_j) \tag{5}$$

where $\Omega = \{\omega_v | v \in N(\Phi)\}$ *contains the node weights which are defined under the following constraint*

$$\begin{cases} \forall v \in N(\Psi), \omega_v \in [0,1] \\ \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v = 1 \\ \sum_{t \in C_v(\Psi)} \omega_t = \omega_v \end{cases} \tag{6}$$

and $\delta_v(\varphi_i, \varphi_j)$ *is defined in Eq. 2.*

The following theorem states the properties of Ω.

Theorem 1 $\forall 1 \leq l \leq H(\Phi)$, $\sum_{v \in N^l(\Phi)} \omega_v = \frac{1}{H(\Phi)}$.

Proof. $\forall 1 \leq l \leq H(\Phi)$, we can verify that

$$\sum_{v \in N^l(\Phi)} \omega_v = \sum_{v \in N^l(\Phi)} \sum_{t \in C_v(\Psi)} \omega_t = \sum_{v \in N^{l+1}(\Psi)} \omega_v$$

Then we have $\sum_{v \in N^l(\Phi)} \omega_v = \frac{1}{H(\Phi)}$.

Theorem 2 *Given two purchase trees* φ_i, φ_j *in* Φ, $d(\varphi_i, \varphi_j, \Omega) \in [0,1]$.

We omit the proof of Theorem 2 for simplicity.

Theorem 3 $d(\varphi_i, \varphi_j, \Omega)$ *defined in (5) is a metric.*

Proof. Since Jaccard distance is a metric, we can verify that $d(\varphi_i, \varphi_j, \Omega) = d(\varphi_j, \varphi_i, \Omega)$ (Symmetry), $d(\varphi_i, \varphi_i, \Omega) = 0$ (Reflexivity) and $d(\varphi_i, \varphi_j, \Omega) \geq 0$ for all φ_i and φ_j in Ψ (Positivity).

Given three Purchase trees φ_i, φ_j and φ_t,

$$\begin{aligned} & d(\varphi_i, \varphi_j, \Omega) + d(\varphi_j, \varphi_t, \Omega) \\ &= \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \omega_v \left[\delta_v(\varphi_i, \varphi_j) + \delta_v(\varphi_j, \varphi_t)\right] \\ &\geq \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_t)} \omega_v \delta_v(\varphi_i, \varphi_t) = d(\varphi_i, \varphi_t, \Omega) \end{aligned} \tag{7}$$

which indicates that the distance satisfies triangle inequality. Therefore, it is a metric.

4 Local PurTree Subspace Spectral Clustering

Given a product tree Ψ and n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$, we want to cluster Φ into c clusters. We want to learn a new graph represented by affinity

matrix $\mathbf{P} = [p_{ij}]_{n\times n}$, in which p_{ij} is the probability that two trees φ_i and φ_j are connected. In order to find c clusters from Φ, we hope that the graph constructed from $\mathbf{P}$ only consists of c connected components. To achieve this goal, we present the following problem to simultaneously learn the node weights ω and the connection probability matrix $\mathbf{P}$

$$\min_{\mathbf{P},\Omega} \sum_{i,j=1}^{n} \left[\sum_{l=1}^{H(\Phi)} \sum_{v\in N^l(\Phi)} \omega_v d_{ij}^v p_{ij} + \lambda p_{ij}^2 \right] + \eta \sum_{l=1}^{H(\Phi)} \sum_{v\in N^l(\Phi)} \omega_v^2 \tag{8}$$

subject to

$$\begin{cases} \forall i, \mathbf{p}_i^T \mathbf{1} = 1, & p_{ij} \in [0,1] \\ \sum_{l=1}^{H(\Phi)} \sum_{v\in N^l(\Phi)} \omega_v = 1, & \omega_v \in [0,1] \\ \sum_{t\in C_v(\Psi)} \omega_t = \omega_v & \\ rank(\mathbf{L}_P) = n - c & \end{cases} \tag{9}$$

where d_{ij}^v is the abbreviation of $\delta_v(\varphi_i, \varphi_j)$ which is computed according to Eq. (2), $\mathbf{L}_P = \mathbf{D}_P - \frac{\mathbf{P}^T+\mathbf{P}}{2}$ is the Laplacian matrix, the degree matrix $\mathbf{D}_P \in \mathcal{R}^{n\times n}$ is defined as a diagonal matrix in which $d_{ij} = \sum_{j=1}^{n} \frac{p_{ij}+p_{ji}}{2}$, λ and η are two regularization parameters.

The first term in (8) is the pairwise product of the PurTree distance and connection probability, in which a smaller distance $d(\varphi_i, \varphi_j, \Omega)$ will be assigned with a larger probability p_{ij}. The second and third terms are regularization terms of $\mathbf{P}$ and Ω, where λ and η are two regularization parameters. The rank constraint $rank(\mathbf{L}_P) = n-c$ is imposed to $\mathbf{L}_P$, such that the sparse graph constructed from P only consists of c connected components [8, 14]. The final clustering assignment can be made by assigning the objects in the same connected component in the graph constructed from $\mathbf{P}$ to the same cluster.

On the other hand, the constraint $rank(\mathbf{L}_P) = n - c$ is equivalent to the following problem [17]

$$\min_{\mathbf{F}\in R^{n\times c}, \mathbf{F}^T\mathbf{F}=I} 2\mu Tr(\mathbf{F}^T \mathbf{L}_P \mathbf{F}) \tag{10}$$

where μ is a large enough parameter.

Then we improve (8) to form the following Local PurTree Subspace Spectral clustering problem

$$\min \sum_{i,j=1}^{n} \left[\sum_{l=1}^{H(\Phi)} \sum_{v\in N^l(\Phi)} \omega_v d_{ij}^v p_{ij} + \lambda p_{ij}^2 \right] + \eta \sum_{l=1}^{H(\Phi)} \sum_{v\in N^l(\Phi)} \omega_v^2 + 2\mu T_r(\mathbf{F}^T \mathbf{L}_P \mathbf{F}) \tag{11}$$

subject to

$$\begin{cases} \forall i, \sum_{j=1} p_{ij} = 1, & p_{ij} \in [0,1] \\ \sum_{l=1}^{H(\Phi)} \sum_{v\in N^l(\Phi)} \omega_v = 1, & \omega_v \in [0,1] \\ \sum_{t\in C_v(\Psi)} \omega_t = \omega_v & \\ \mathbf{F}^T\mathbf{F} = I, & \mathbf{F} \in R^{n\times c} \end{cases} \tag{12}$$

We can apply the alternative optimization approach to solve problem (11).

4.1 Optimization of F

When $\mathbf{P}$ and ω are fixed, problem (11) becomes

$$\min_{\mathbf{F}\in\mathcal{R}^{n\times c},\mathbf{F}^T\mathbf{F}=I} Tr(\mathbf{F}^T\mathbf{L}_P\mathbf{F}) \tag{13}$$

The optimal solution $\mathbf{F}$ in problem (13) is formed by c eigenvectors of $\mathbf{L}_P$ which corresponds to its c smallest eigenvalues.

4.2 Optimization of P

When $\mathbf{F}$ and ω are fixed, problem (11) becomes

$$\begin{aligned} &\min_{\mathbf{P}} \sum_{i,j=1}^{n} \left[\left(\sum_{l=1}^{H(\Phi)} \sum_{v\in N^l(\Phi)} \omega_v d_{ij}^v + \mu \|f_i - f_j\|_2^2 \right) p_{ij} + \lambda p_{ij}^2 \right] \\ &st.\ \ \forall i, \sum_{j=1} p_{ij} = 1, p_{ij} \in [0,1] \end{aligned} \tag{14}$$

where $f_i \in \mathcal{R}^{c\times 1}$ is the transpose of the i-th row of $\mathbf{F}$.

Note that problem (14) is independent between different i, so we can solve it individually for each p_i (the vector form of $[p_{i1}, \ldots, p_{in}]$). Denote $d_{ij}^f = \|f_i - f_j\|_2^2$, $d_{ij}^\Psi = \sum_{l=1}^{H(\Phi)} \sum_{v\in N^l(\Phi)} \omega_v d_{ij}^v$, and $d_i \in \mathbb{R}^{n\times 1}$ as a vector with the j-th element as $d_{ij} = d_{ij}^\Psi + \mu d_{ij}^f$.

The Lagrangian function of problem (14) for p_i is

$$\mathcal{L}(p_i, \chi, \tau) = \lambda \sum_{j=1}^{n} p_{ij}^2 + \sum_{j=1}^{n} d_{ij} p_{ij} + \chi(\sum_{j=1}^{n} p_{ij} - 1) - p_i^T \tau \tag{15}$$

where χ and positive vector τ are Lagrangian multipliers.

It can be verified that the optima p_{ij} is

$$p_{ij} = \left(-\frac{1}{2\lambda} \left[d_{ij} + \chi \right] \right)_+ \tag{16}$$

where χ can be computed from $\sum_{j=1}^{n} p_{ij} = 1$ and $a_+ = \max(a, 0)$.

4.3 Optimization of $\boldsymbol{\omega}$

When $\mathbf{P}$ and $\mathbf{F}$ are fixed, problem (11) becomes

$$\min \sum_{i,j=1}^{n} \sum_{l=1}^{H(\Phi)} \sum_{v\in N^l(\Phi)} \omega_v d_{ij}^v p_{ij} + \eta \sum_{l=1}^{H(\Phi)} \sum_{t\in N^l(\Phi)} \sum_{v\in C_t} \omega_v^2 \tag{17}$$

subject to

$$\begin{cases} \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v = 1, \omega_v \in [0,1] \\ \sum_{t \in C_v(\Psi)} \omega_t = \omega_v \end{cases} \tag{18}$$

According to Theorem 1, the sum of node weights in each level is a constant value $\frac{1}{H(\Phi)}$. Therefore, we can sequently optimize the node weights level by level. We first let $\omega_{Root} = \frac{1}{H(\Phi)}$. Given an internal node $t \in N^l(\Phi)$, we fix ω_t and solve the following problem for $\{\omega_v | v \in C_t\}$

$$\min_{\sum_{v \in C_t} \omega_v = \omega_t} \sum_{i,j=1}^{n} \sum_{v \in C_t} \omega_v d_{ij}^v p_{ij} + \eta \sum_{v \in C_t} \omega_v^2 \tag{19}$$

The Lagrangian function of problem (19) is

$$\mathcal{L}(\omega_v | v \in C_t) = \sum_{v \in C_t} \omega_v \sum_{i,j=1}^{n} d_{ij}^v p_{ij} + \eta \sum_{v \in C_t} \omega_v^2 + \chi_t (\sum_{v \in C_t} \omega_v - \omega_t) - \sum_{v \in C_t} \tau_v \omega_v \tag{20}$$

where $t = P^v$, $\chi_t \geq 0$ and $\{\tau_v \geq 0 | v \in C_t\}$ are multipliers.

It can be verified that the optima ω_v is

$$\omega_v = \left(-\frac{1}{2\eta} \left[\sum_{i,j=1}^{n} d_{ij}^v p_{ij} + \chi_t \right] \right)_+ \tag{21}$$

where t is the children of v and χ_t can be computed from $\sum_{v \in C_t} \omega_v = \omega_t$.

4.4 Determination of λ

Since a transaction data set often consists of a large number of users, it is very useful to explore the local connectivity during the clustering process. Assuming that P is a sparse graph in which for each i, p_i contains at most k positive values for its k-nearest neighbors. However, it is impossible to select the real k-nearest neighbors at the start because Ω is unknown. To simplify the computing, we select approximate k nearest neighbors at the start, where the node weights $\omega_v \in \Omega$ are defined as

$$\omega_v = \begin{cases} \frac{1}{H(\Phi)} \text{ if } v = Root(\Psi) \\ \frac{\omega_t}{|C_t|} \text{ if } v \in C_t \end{cases} \tag{22}$$

Then we from a partially ordered set $\{d_{ij}^e | 1 \leq j \leq n, j \neq i\}$, in which d_{ij}^e is the j-th small value. The k-nearest neighbors $\mathcal{M}_i$ for φ_i can be formed by selecting k smallest values in $\{d_{ij}^e | 1 \leq j \leq n, j \neq i\}$.

According to Eq. (16), we know that p_{ij} is negatively proportional to d_{ij}. If we wish p_i contains at most k positive values, we can set a proper λ such that

only k-nearest neighbors $\mathcal{M}_i$ for φ_i are with positive p_{ij}. According to (16), we know that

$$\begin{cases} -\frac{d_{ik}+\chi}{2\lambda_i} > 0 \\ -\frac{d_{i,k+1}+\chi}{2\lambda_i} \leq 0 \end{cases} \tag{23}$$

where d_{ik} is the k-th smallest distance to φ_i.

According to the constraint $\sum_{j\in\mathcal{M}_i} p_{ij} = 1$, we have

$$\chi = -\frac{2\lambda_i}{k} - \frac{1}{k}\sum_{j\in\mathcal{M}_i} d_{ij} \tag{24}$$

So we have the following inequality for λ_i according to (23) and (24)

$$\frac{k}{2}d_{ik} - \frac{1}{2}\sum_{j\in\mathcal{M}_i} d_{ij} < \lambda_i \leq \frac{k}{2}d_{i,k+1} - \frac{1}{2}\sum_{j\in\mathcal{M}_i} d_{ij} \tag{25}$$

In problem (8), we wish all k neighbors be used for clustering. Therefore, λ can be set as its upper bound. However, it is impossible to compute the exact upper bound λ_i since the node weights ω change in each iteration. We turn to compute an approximate upper bound of λ with equal level weights of ω, and set λ_i as the approximate upper bound

$$\lambda_i = \frac{k}{2}d^e_{i,k+1} - \frac{1}{2}\sum_{j\in\mathcal{M}_i} d^e_{ij} \tag{26}$$

where d^e_{ij} is the PurTree subspace distance computed with Ω defined in Eq. (22).

The overall λ can be set to the mean of $\{\lambda_1, ..., \lambda_n\}$

$$\lambda = \frac{1}{2n}\sum_{i=1}^{n}\left(kd^e_{i,k+1} - \sum_{j\in\mathcal{M}_i} d^e_{ij}\right) \tag{27}$$

The optimal solution of p_{ij} for k-nearest neighbors is

$$p_{ij} = \left(-\frac{1}{2\lambda}\left[d_{ij} + \chi\right]\right)_+ \tag{28}$$

where χ can be computed from $\sum_{j\in\mathcal{M}_i} p_{ij} = 1$.

4.5 The Optimization Algorithm

The detailed algorithm to solve the problem (11) is summarized in Algorithm 1 [3]. Given a set of n purchase trees $\Phi = \{\varphi_1, \ldots, \varphi_n\}$, we want to cluster Φ into c clusters. F, ω and P in (11) are iteratively solved until it converges. The final clustering result can be obtained from the connected components in the graph constructed from $\mathbf{P}$. If the algorithm needs r iterations to converge, the complexity of LPPS algorithm is $O(rkn^2)$.

[3] In practice, μ can be set according to the method used in [17].

Algorithm 1 Local PurTree Subspace Spectral clustering algorithm to solve problem (11)

Input: A set of n purchase trees $\Phi = \{\varphi_1, \ldots, \varphi_n\}$, a regularization parameters η, the number of nearest neighbors k, and the number of clusters c.
Initialize ω according to Eq. (22) and find k nearest neighbors $\mathcal{M}_i$ for each φ_i (using the cover tree [3]), in which the distance is computed with the node weights defined in Eq. (22).
Initialize $\mathbf{P}$ by solving the problem (28), in which d_{ij}^f is set as 0.
repeat
    Update $\mathbf{F}$ by selecting c eigenvector of $\mathbf{L}_P = \mathbf{D}_P - \frac{\mathbf{P}^T + \mathbf{P}}{2}$ which corresponds to its c smallest eigenvalues.
    for $l = 2$ to $H(\Psi)$ **do**
        for $\forall\ v \in N^l(\Phi)$ **do**
            Update ω by solving the equation (21).
        end for
    end for
    Update P by solving the equation (28).
until (converges)
Output: Find the connected components in the graph constructed from $\mathbf{P}$, and assign the objects in the same connected component to the same cluster.

5 Experimental results and analysis

In this section, we present the experiments conducted on 10 real-life transaction data sets to investigate the effectiveness and scalability of the LPSS algorithm.

5.1 Comparison Scheme

10 real-life transaction data sets were used to investigate the effectiveness and scalability of the proposed algorithm. D_1 was built from a superstore's transactional data set[4], which consists of $8,399$ transaction records from 796 customers. $D_2 \sim D_6$ were built from five subsets of a super market's transactional data, which contains up to 25 million newest transaction records. $D_7 \sim D_{10}$ were built from four subsets of a year of purchase history transaction data from the kaggle competition[5], which contains more than 349 million transaction records. The characteristics of 10 data sets are shown in Table 1, in which $D_2 \sim D_6$ contain 5 levels and other five data sets contain 4 levels.

Since the 10 data sets in Table 1 contain no labels, we use internal validation method to evaluate the clustering results. Given c clusters $\mathcal{C}$, the commonly-used internal validation method $\log(W_k) = \log\{\sum_{l=1}^{c} \frac{1}{2|\mathcal{C}_l|} \sum_{i,j \in \mathcal{C}_l} d(\varphi_i, \varphi_j)\}$. However, $\log(W_k)$ is affected by c and the weights in both PurTree distance and PurTree subspace distance.

[4] `https://community.tableau.com/docs/DOC-1236`
[5] `http://www.kaggle.com/c/acquire-valued-shoppers-challenge/data`

Table 1. Characteristics of 10 customer transaction data sets.

Data sets (D)	Size	$\|N^2(D)\|$	$\|N^3(D)\|$	$\|N^4(D)\|$	$\|N^5(D)\|$
D_1	795	3	17	1264	
D_2	208	84	501	1198	9760
D_3	416	90	552	1339	14274
D_4	832	90	606	1457	18137
D_5	1665	91	651	1584	22822
D_6	3330	91	697	1714	28065
D_7	608	821	73164	89665	
D_8	1216	826	74500	91785	
D_9	2433	825	75737	93731	
D_{10}	4867	827	77424	96538	

We may get different $log(W_k)$ even for the same clustering result, with different node weights ω. So we normalize log(Wk) with the sum of all distances. In this paper, we normalize $log(W_k)$ with the sum of all distances and propose a normalized logarithm of within-cluster dispersion for evaluating a clustering result, which is computed as

$$\mathcal{NLW}(\mathcal{C}) = \log\{\sum_{l=1}^{c} \frac{1}{2|\mathcal{C}_l|} \sum_{i,j \in \mathcal{C}_l} d(\varphi_i, \varphi_j)\} - \log(\sum_{i,j=1}^{n} d(\varphi_i, \varphi_j)) \qquad (29)$$

where $\mathcal{C}$ consists of c clusters. The lower the $\mathcal{NLW}(\mathcal{C})$, the better the clustering result.

5.2 Results and Analysis

We used all ten data sets to compare the effectiveness of the LPSS algorithm with five clustering algorithms, i.e., DBSCAN, HAC, Ncut [15], RCut [9] and PurTreeClust [5]. The PurTree subspace distance is used for LPSS, and the PurTree distance is used for other five clustering methods. In this experiment, we selected 96 integers from 5 to 100 for c. The similarity matrices for DBSCAN, HAC, Ncut, RCut and PurTreeClust were computed as 1 minus the PurTree distance matrices, in γ was set as the same values used in [5], i.e., $\gamma = \{0, 0.2, 0.8, 1, 2, 1000\}$. 20 integers from 5 to 100 were used for k in DBSCAN, PurTreeClust and LPSS. The other parameters of all methods were set in the same strategy to make the experiments fair enough, i.e., $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\}$. For DBSCAN, we also set eps as 95 values from 0.25 to 0.5 and minPts as 95 values from 1% to 10% of the number of objects. For each clustering algorithm, we computed the average $\mathcal{NLW}$. The results are shown in Table 2, in which the mean and standard deviation of $\mathcal{NLW}$ are reported. From these results,

Table 2. Performance comparison of $\mathcal{NLW}$ ($Mean \pm Standard\ Deviation$) by six clustering algorithms on 10 benchmark data sets (The smallest value for each data set is highlighted in bold).

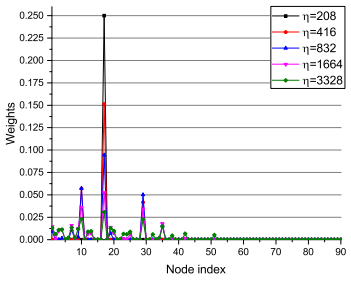
Data	DBSCAN	HAC	NCut	RCut	PurTreeClust	LPSS
D_1	-7.47 ± 1.23	-7.75 ± 1.24	-7.41 ± 0.24	-7.41 ± 0.24	**-7.83** ± 0.80	-7.63 ± 0.36
D_2	-6.03 ± 0.04	**-6.34** ± 0.90	-6.17 ± 0.40	-6.19 ± 0.40	-6.31 ± 0.80	-6.31 ± 0.35
D_3	-6.73 ± 0.04	-6.94 ± 0.64	-6.81 ± 0.20	-6.81 ± 0.20	-6.91 ± 0.64	**-7.05** ± 0.32
D_4	-7.42 ± 0.04	-7.59 ± 0.63	-7.48 ± 0.18	-7.48 ± 0.24	-7.62 ± 0.52	**-7.68** ± 0.36
D_5	-8.11 ± 0.08	-8.26 ± 0.45	-8.16 ± 0.15	-8.15 ± 0.15	-8.34 ± 0.38	**-8.38** ± 0.33
D_6	-8.80 ± 0.08	-8.92 ± 0.58	-8.84 ± 0.12	-8.84 ± 0.12	-8.89 ± 0.41	**-9.28** ± 0.34
D_7	-7.10 ± 0.08	-7.19 ± 0.16	-7.17 ± 0.12	-7.16 ± 0.12	-7.40 ± 0.49	**-8.63** ± 0.35
D_8	-7.80 ± 0.08	-7.85 ± 0.14	-7.83 ± 0.12	-7.83 ± 0.12	-7.90 ± 0.47	**-8.98** ± 0.34
D_9	-8.49 ± 0.08	-8.53 ± 0.13	-8.52 ± 0.12	-8.51 ± 0.12	-8.58 ± 0.49	**-9.63** ± 0.38
D_{10}	-9.18 ± 0.08	-9.21 ± 0.12	-9.20 ± 0.08	-9.20 ± 0.08	-9.30 ± 0.50	**-10.24** ± 0.35

we can see that LPSS produced the highest NLW on eight data sets. We noticed that LPSS produced much better results than other clustering algorithms when the product tree contains a large number of leaf nodes. For example, on D_7, D_8, D_9 and D_{10} which consists of more than 80 thousands of products, LPSS has more than 10% average improvement compared to the second best method PurTreeClust. On D_7, LPSS even produced nearly 17% average improvement compared to the second best method PurTreeClust. Besides LPPS, PurTreeClust produced better results than other methods.

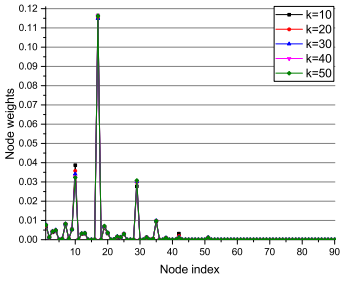
5.3 Parameter Sensitivity Study

We select D_4 to show the relationship between the node weights ω and η, k. For each k or η, we computed the average node weights and draw them in Figure 2. From these figures, we can see that with the increase of η, the node wrights become more evenly distributed. But with the increase of k, the node weights were almost stable. This indicates that the node wights were sensitive to η, but insensitive to k. Therefore, we can set smaller η in order to select fewer nodes for consideration during clustering. We also observe that the node weight distributions in different levels are highly correlated. To be specific, the average weight decreases level by level. This can be verified according to Theorem 1 which indicates that the sum of weights for nodes in different levels are equal, and the fact that the high level contains more nodes than low level.

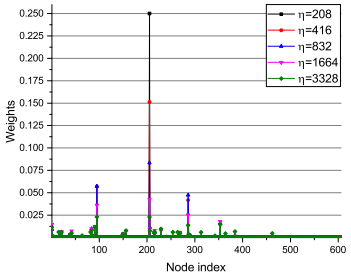
We compute the ratio of positive weights versus η and k, and draw the results in Figure 3. From these figures, we can see that the number of positive weights is sensitive to η, but insensitive to k. With the increase of the tree level, the number of positive weights decreases rapidly.



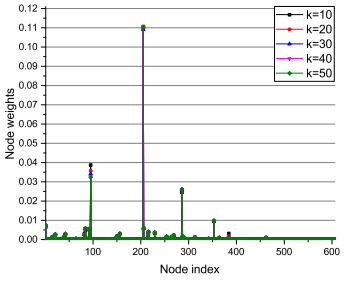
(a) $\{\omega_v | v \in N^2(\Phi)\}$ versus η.



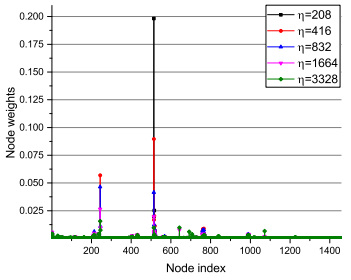
(b) $\{\omega_v | v \in N^2(\Phi)\}$ versus k.



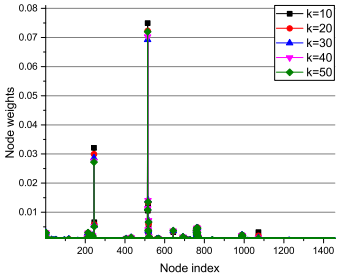
(c) $\{\omega_v | v \in N^3(\Phi)\}$ versus η.



(d) $\{\omega_v | v \in N^3(\Phi)\}$ versus k.



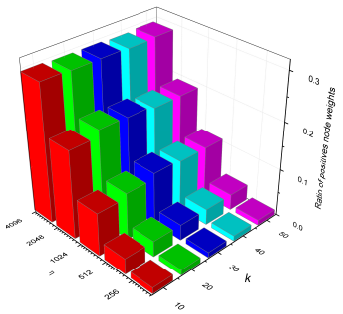
(e) $\{\omega_v | v \in N^4(\Phi)\}$ versus η.



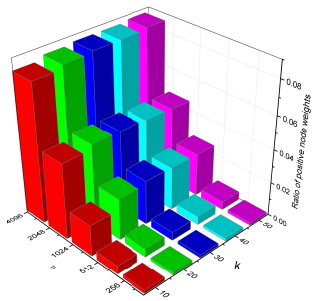
(f) $\{\omega_v | v \in N^4(\Phi)\}$ versus k.

Fig. 2. Node weights $\Omega = \{\omega_v\}$ versus η and k on D_4.

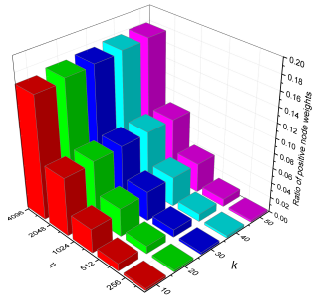
Finally, we analyze the change of $\mathcal{NLW}$ with both parameters η and k. The relationships between the mean and standard deviation of $\mathcal{NLW}$ and two parameters η and k are shown in Fig. 4. From these figures, we can see that the average $\mathcal{NLW}$ did not change too much with the change of η, and it is nearly stable with the change of k.



(a) The results for nodes in the second level.



(b) The results for nodes in the third level.



(c) The results for nodes in the fourth level.

Fig. 3. Ratio of positive weights versus η and k.

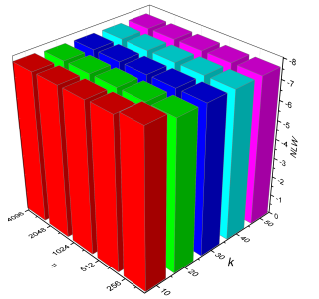


Fig. 4. Average $\mathcal{NLW}$ versus η and k.

6 Conclusions

We have presented LPSS, a local PurTree Subspace Spectral clustering algorithm for customer transaction data. In the new method, a recently proposed

"Purchase Tree" is used to represent a customer and we have proposed a PurTree subspace distance to compute the distance between two purchase trees, in which a set of sparse node weights are introduced to distinguish the importance of different nodes in a purchase tree. We have proposed an iterative optimization algorithm to optimize the new clustering model, in which a local similarity matrix and a set of sparse node weights are simultaneously learnt. An effective method is proposed to compute one regularization parameters in LPSS. A series of experiments were conducted on 10 real-life data sets and the experimental results have demonstrated the superior performance of the new method. In the future work, we will improve our method to handle larger scale transaction data.

Acknowledgment

This research was supported by NSFC under Grant no.61305059, 61473194 and 61502177, Fundamental Research Funds for the Central Universities under Grant D2172500.

References

1. Ben-Hur, A., Horn, D., Siegelmann, H.T., Vapnik, V.: Support vector clustering. The Journal of Machine Learning Research 2, 125–137 (2002)
2. Berson, A., Smith, S.J.: Building data mining applications for CRM. McGraw-Hill, Inc. (2002)
3. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: Proceedings of the 23rd international conference on Machine learning. pp. 97–104. ACM (2006)
4. Cai, X., Nie, F., Huang, H., Kamangar, F.: Heterogeneous image feature integration via multi-modal spectral clustering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011). pp. 1977–1984. IEEE (2011)
5. Chen, X., Huang, J.Z., Luo, J.: PurTreeClust: A Purchase Tree Clustering Algorithm for Large-scale Customer Transaction Data. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE). pp. 661–672 (May 2016)
6. Chen, X., Xu, X., Ye, Y., Huang, J.Z.: TW-k-means: Automated Two-level Variable Weighting Clustering Algorithm for Multi-view Data. IEEE Transactions on Knowledge and Data Engineering 25(4), 932–944 (2013)
7. Chen, X., Ye, Y., Xu, X., Huang, J.Z.: A feature group weighting method for subspace clustering of high-dimensional data. Pattern Recognition 45(1), 434–446 (2012)
8. Chung, F.R.: Spectral graph theory, vol. 92. American Mathematical Soc. (1997)
9. Hagen, L., Kahng, A.B.: New spectral methods for ratio cut partitioning and clustering. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 11(9), 1074–1085 (1992)
10. Hsu, F.M., Lu, L.P., Lin, C.M.: Segmenting customers by transaction data with concept hierarchy. Expert Systems with Applications 39(6), 6221–6228 (2012)
11. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans. Knowl. Discov. Data 3(1), 1–58 (2009)

12. Kuo, R., Ho, L., Hu, C.M.: Integration of self-organizing feature map and k-means algorithm for market segmentation. Computers & Operations Research 29(11), 1475–1493 (2002)
13. Lu, T.C., Wu, K.Y.: A transaction pattern analysis system based on neural network. Expert Systems with Applications 36(3), 6091–6099 (2009)
14. Mohar, B., Alavi, Y., Chartrand, G., Oellermann, O.: The laplacian spectrum of graphs. Graph theory, combinatorics, and applications 2(871-898), 12 (1991)
15. Ng, A.Y., Jordan, M.I., Weiss, Y., et al.: On spectral clustering: Analysis and an algorithm. Advances in neural information processing systems 2, 849–856 (2002)
16. Ngai, E.W., Xiu, L., Chau, D.C.: Application of data mining techniques in customer relationship management: A literature review and classification. Expert systems with applications 36(2), 2592–2602 (2009)
17. Nie, F., Wang, X., Huang, H.: Clustering and projected clustering with adaptive neighbors. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 977–986. ACM (2014)
18. Tsai, C.Y., Chiu, C.C.: A purchase-based market segmentation methodology. Expert Systems with Applications 27(2), 265–276 (2004)
19. Von Luxburg, U.: A tutorial on spectral clustering. Statistics and computing 17(4), 395–416 (2007)
20. Wang, K., Xu, C., Liu, B.: Clustering transactions using large items. In: Proceedings of the eighth international conference on Information and knowledge management. pp. 483–490. ACM (1999)
21. Xiao, Y., Dunham, M.H.: Interactive clustering for transaction data. In: Data Warehousing and Knowledge Discovery, pp. 121–130. Springer (2001)
22. Xiong, T., Wang, S., Mayers, A., Monga, E.: Dhcc: Divisive hierarchical clustering of categorical data. Data Mining and Knowledge Discovery 24(1), 103–135 (2012)