Clustering High Dimensional Categorical Data via Topographical Features

Chao Chen CHAO.CHEN@QC.CUNY.EDU
CUNY Queens College & Graduate Center, New York, NY, USA

Novi Quadrianto N.QUADRIANTO@SUSSEX.AC.UK
SMiLe CLiNiC, University of Sussex, Brighton, UK

Abstract

Analysis of categorical data is a challenging task. In this paper, we propose to compute topographical features of high-dimensional categorical data. We propose an efficient algorithm to extract modes of the underlying distribution and their attractive basins. These topographical features provide a geometric view of the data and can be applied to visualization and clustering of real world challenging datasets. Experiments show that our principled method outperforms state-of-the-art clustering methods while also admits an embarrassingly parallel property.

1. Introduction

Categorical data analysis has been critical in many domains. In biomedical informatics, basic attributes of a patient include blood type, gender and race. Disease-specific attributes include the subtypes of disease and the treatments a patient has received. DNA/RNA sequence data have categorical values, that is, different nucleobases. These modern categorical data exhibit high dimensionality, insufficient samples and inhomogeneity. To handle these challenges, new methods for visualizing and exploring complex datasets are crucially needed.

In this paper, we develop a method to characterize the geometry of a categorical dataset by computing its topographical features, i.e., modes and their associated attractive basins. These features provide a geometric description of a given dataset and naturally lead to practical tools for visualization and clustering. For a given data and an estimation of its underlying distribution, we compute *modes*, i.e., local maxima whose probability is bigger than all nearby points. By exploiting modes, we then decompose the discrete domain into regions called the *basins of attraction*. Within each basin of attraction, every point converges to the same mode as directed by a discrete version of gradient ascent. See Figure 1 for a schematic illustration.

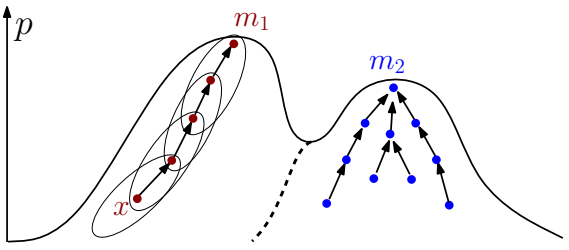


Figure 1: The topographical landscape of a distribution. There are two modes, m_1 and m_2. Their attractive basins are separated by a valley (dashed line). Each data point converges to one of the two modes following a discrete gradient ascent path.

It is computationally prohibitive to compute all the modes and their attractive basins. Instead, we take a data-centered approach and compute the discrete gradient ascent paths, each of which starts from a data point and ends at a mode. These paths associate data points with different modes and form a natural clustering. The paths merging into a same mode provide an outline of the attractive basin. The computation of each discrete gradient ascent step can be viewed as finding a neighboring point with the highest probability. This is challenging as the neighborhood of a high dimensional point can be of exponential size.

To address this issue, we propose to adopt the classic tree-structured graphical model to approximate the underlying probabilistic density function from given data. We leverage the tree structure of the model and develop a dynamic programming algorithm that finds the neighbor with the highest probability efficiently. While the tree model brings computational advantage, it also provides a satisfying description of high-dimensional data, as have been proven theoretically and practically (Liu et al., 2011; Minka & Qi, 2003). Our algorithm is efficient; given a tree model, computing one gradient ascent step is linear to the dimension. The method is linear to the data size. But it is extremely easy to parallelize as we can compute gradient ascent paths for all data independently.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

Our method offers a different view from most clustering methods. Unlike the top-down methods that derive clusters using a mixture of parametric models, our method does not hold any geometric or probabilistic assumption on each cluster. It respects the intrinsic geometry of the underlying distribution and uses it to partition the data. Compared with bottom-up approaches that agglomerate data based on similarities between them, our method is based on a probabilistic model and is more principled.

1.1. Related Work

Clustering is a classic yet very useful technique for modern data analysis (Xu & Wunsch, 2005). Many clustering algorithms have been proposed and they are possibly far beyond what we could keep track of. The classical K-Means (MacQueen, 1967) associates data from each cluster to a centroid point. The algorithm minimizes an objective function which is the total squared Euclidean distance between each data point and its associated centroid. The algorithm iteratively updates the centroid of each cluster and the membership of each data point, until it converges. The initial centroids and the number of clusters are critical for the quality of the result. Various algorithms have been developed to find a robust initialization and to estimate the number of clusters (Bradley & Fayyad, 1998). Probabilistic approaches to clustering such as Gaussian mixture model (GMM), Dirichlet process Gaussian mixture models (DPGMM) (Rasmussen, 2000), and Pitman-Yor diffusion trees (Knowles & Ghahramani, 2015) are also available.

For categorical data, methods such as K-Modes and ROCK are at one's disposal. K-Modes (Huang, 1998) optimizes the same objective function as K-Means, except that the squared Euclidean distance function is replaced by the Hamming distance and vectors of modes of categorical attributes instead of means are used to represent cluster centers. Since K-Modes is essentially same as K-Means, each iteration costs a linear time with respect to both the data size and the dimension. ROCK (Guha et al., 1999) is a type of hierarchical clustering algorithm. This method starts by initializing each data point as a cluster, and iteratively merge clusters with large similarity until a lowerbound of threshold is reached. The similarity between two clusters is measured by the amount of common neighbors the clusters share. Due to the need to compute list of neighbors and links among pairs of points, the time complexity of ROCK is quadratic to the data size.

Mixture models can also be used to cluster discrete data. The latent class analysis (LCA) (McCutcheon, 1987; Bishop, 2006) method assumes a binary valued discrete domain and fits a mixture of discrete distributions. Each component of the mixture is a product of D Bernoulli distributions, corresponding to the D independent random variables. It is straightforward to generalize LCA to multiple-valued discrete domain (Linzer et al., 2011). We call the generalized model the mixture of products of discrete distributions (MPD) and use it as one of our baselines. Like other mixture model, MPD are learned using an EM algorithm. Each iteration is linear to both the data size and the dimension. However, unlike other methods, this method is also linear to L, the maximal number of distinct values a variable can have. The reason is that the distribution of a multi-valued variable is expressed as the distribution of L binary-valued variables. For completeness, we also refer to more advanced latent class model (Zhang, 2004) for which the independence assumption is dropped.

Chen *et al*. (2014) also computed modes based on a given tree-model. The difference between our method and their method (called TMode) is twofold. First, unlike our method, TMode is *exponential* to the degree of the underlying tree-model. This could be computationally prohibitive as the tree degree could be linear to the dimension. Second, our method not only computes modes, but also computes attractive basins, which are a natural way to associate all data to their corresponding modes. This is not provided by TMode. As we will demonstrate in the experiment section, simply computing modes and associating data to their nearest modes will not work well in clustering. We also note that mode-based clustering method has been proposed before (Cheng, 1995). Such method, however, is built for continuous domain. Furthermore, it is based on kernel density estimation, which suffers from curse of dimensionality (Hastie et al., 2009) and does not suit high dimension.

Topographical features. For a scalar function, topographical features, e.g., modes, ridges, attractive basins, and their combinatorial structure, the Morse-Smale complex, have been studied extensively in the classic Morse theory (Milnor, 1963). In recent years, with the development of topology data analysis (Edelsbrunner & Harer, 2010; Carlsson, 2009), topographical features of a probabilistic density function have been used to analyze modern data (Chazal et al., 2014; Chen et al., 2015; Bubenik, 2015; Chen & Edelsbrunner, 2011). While the theoretical foundation, e.g., the theory of *persistent homology* (Edelsbrunner & Harer, 2010), has been well studied for general context, the algorithms for these topographical features (Chen & Kerber, 2011; 2013) are typically exponential to the dimension and thus are impractical for high-dimensional data.

2. Background

We review discrete graphical models, tree-structured graphical models, and how to estimate a tree-structured model from given data. A probabilistic graphical model (Wainwright & Jordan, 2008; Nowozin & Lampert, 2010) represents a joint distribution over a set of interdependent random variables, $X = (X_1, \ldots, X_D)$, using a graph $G = (\mathcal{V}, \mathcal{E})$ and a potential function f. The set of D vertices/nodes $\mathcal{V}$ corresponds to the set of D random vari-

ables, each of which can be assigned a discrete value $x_i \in \mathcal{L} = \{1, \ldots, L\}$. The edges encode the dependence between different variables. A value assignment to all random variables $x = (x_1, \ldots, x_D)$ is called a *configuration*. We denote by $\mathcal{X} = \mathcal{L}^D$ the domain of all configurations. The potential function $f : \mathcal{X} \to \mathbb{R}$ assigns to each configuration a real value, which is inversely proportional to the logarithm of the probability distribution,

$$p(x) = \exp(-f(x) - A), \tag{2.1}$$

where $A = \log \sum_{x \in \mathcal{X}} \exp(-f(x))$ is the *log-partition function*. Assuming these variables satisfy the Markov properties, the potential function can be written as $f(x) = \sum_{(i,j) \in \mathcal{E}} f_{i,j}(x_i, x_j)$, where $f_{i,j} : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ is the potential function for edge (i, j) [1]. For convenience, we assume any two different configurations have different potential function values. We call a configuration of a subgraph B a *partial configuration*. For a given configuration x, we may denote by x_B its configurations over B. We denote by $f_B(x_B)$ the potential of the partial configuration, which is only evaluated over edges within B. When the context is clear, we drop the subscript B and write $f(x_B)$.

Tree-structured graphical models. We focus on graphical models whose underlying graph G is a tree. A *tree distribution*, i.e., one which can be represented by a tree structure, $T = (\mathcal{V}, \mathcal{E})$, has the following factorization:

$$P(X = x) = p(x) = \prod_{(i,j) \in \mathcal{E}} \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \prod_{k \in \mathcal{V}} p(x_k). \tag{2.2}$$

Here $p(x_i, x_j)$ is the bivariate marginal density of the variable X_i and X_j, and $p(x_k)$ is the univariate marginal density of the variable X_k. The potential can be written as

$$\begin{aligned} f_{i,j}(x_i, x_j) = &- \log(p(x_i, x_j)) + (1 - 1/d_i) \log(p(x_i)) \\ &+ (1 - 1/d_j) \log(p(x_j)), \end{aligned} \tag{2.3}$$

in which d_i and d_j are the degrees of nodes i and j.

For an unknown distribution p^*, we approximate it by finding the *oracle tree distribution*, q^*, namely, the tree distribution with the minimal Kullback-Leibler (KL) divergence from p^*, that is, $q^* := \operatorname{argmin}_{q \in \mathcal{P}_T} D(p^* || q)$, where $\mathcal{P}_T$ is the family of all tree distributions and $D(p||q) := \sum_{x \in \mathcal{X}} p(x)(\log p(x) - \log q(x))$. Bach and Jordan (Bach & Jordan, 2003) proved that when the unknown distribution, p^*, is a tree distribution, the oracle tree distribution, q^*, has the same marginal univariate and bivariate distributions as p^*. Furthermore, it has been shown that the tree structure of the oracle tree distribution is the maximum spanning tree when the edge weights are the pairwise mutual information, formally, $T^* := \operatorname{argmax}_T \sum_{(i,j) \in \mathcal{E}(T)} I_{ij}$, where $\mathcal{E}(T)$ is the edge set of the tree graph T and

$$I_{ij} := \sum_{(x_i, x_j) \in \mathcal{L}^2} p^*(x_i, x_j) \log \frac{p^*(x_i, x_j)}{p^*(x_i)p^*(x_j)}.$$

In reality, we do not know the true marginal univariate and bivariate distributions. Instead, we are given N samples from the true distribution. We thus calculate empirical mutual information $\hat{I}_{ij}$ using the empirical marginal distributions $\hat{p}(x_i, x_j)$ and $\hat{p}(x_k)$. The estimated tree is then obtained by computing the maximum spanning tree on estimated edge weights using Kruskal's algorithm (Cormen et al., 2001): $\hat{T} = \operatorname{argmax}_T \sum_{(i,j) \in \mathcal{E}(T)} \hat{I}_{ij}$. The potential function on each edge can be estimated similarly using the estimated marginal univariate and bivariate distributions (as in Equation (2.3)). This estimation algorithm, first proposed by Chow and Liu (Chow & Liu, 1968), has various theoretical guarantees (Liu et al., 2011) and will be the basis of our method. It is linear to the data size and quadratic to the dimension.

3. Method

Our method clusters discrete data by exploiting the topographical landscape of the underlying probabilistic distribution. We partition the whole discrete domain into different regions, called the attractive basins, each of which is represented by a local maximum (mode) of the distribution. We do not explicitly compute all modes and attractive basins. Instead, we focus on each individual data and find out the mode it is associated with. For a data point, x, we compute its gradient ascent path by running a neighborhood-search algorithm: start from x and iteratively move to the neighbor with the highest probability. The point at which the neighborhood-search algorithm converges is a mode and is the representative of x. Figure 1 illustrates the procedure; x is associated with the local maximum m_1. We also show several other data points associated with another local maximum m_2. The valley (dashed curve) in between the two mountains separates the domain into two parts/attractive basins, represented by the two modes, respectively.

First, we provide formal definitions of these concepts in the discrete setting and subsequently introduce our algorithm in details.

Definitions. We define the *neighborhood* of a point x as the Hamming ball with a fixed radius δ, formally,

$$\mathcal{N}_\delta(x) = \{y \in \mathcal{X} \mid \operatorname{dist}_{\mathrm{H}}(x, y) \leq \delta\},$$

where the *Hamming distance*, $\operatorname{dist}_{\mathrm{H}}(x, y)$, measures the number of random variable at which x and y disagree. For example, in Figure 2a, the vertices of a three-dimensional

[1] W.l.o.g., we drop unary potentials f_i, as they can be absorbed into binary potentials. That is, any potential function with unary potentials can be rewritten as a potential function without them.

cube correspond to a three-dimensional binary-valued discrete domain. The neighborhood of the point (010) with $\delta = 1$ is highlighted. When $\delta = 2$, the neighborhood of (010) will be everything except for the point (101).

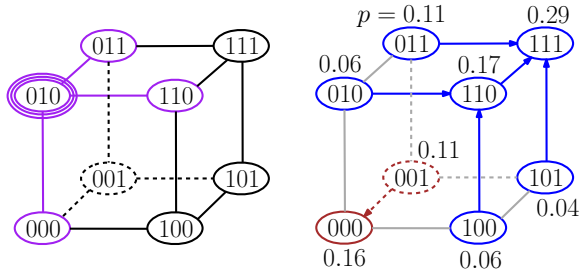
For each point $x \in \mathcal{X}$, we say the *next step* of x is the maximal probability point within the δ-neighborhood, namely,

$$\text{next}(x, \delta) = \underset{y \in \mathcal{N}_\delta(x)}{\text{argmax}}\, p(y).$$

When the next step from x is still x, $\text{next}(x, \delta) = x$, we say x is a *local maximum*, called a δ-*mode*. In other words, a mode x has a higher probability than all its neighbors. In Figure 2b, for the given probability distribution, we use arrows to show what is the next step of each point. There are only two modes, (000) and (111). These $\text{next}(.)$ and modes define a natural *decomposition of the domain*. In Figure 2b, the domain is decomposed into two parts (red and blue), represented by the two modes (000) and (111) respectively. This provides a natural way of clustering the data. Our main clustering algorithm follows.

The parameter δ, called the *scale*, controls the granularity of the clustering of the data/decomposition of the domain. As we increase δ, the neighborhood $\mathcal{N}_\delta(x)$ increases and modes may disappear. In Figure 2b, when $\delta = 2$, $\text{next}((000)) = (110)$ and $\text{next}(x) = (111)$ for any other x. There is only one mode (111), representing the whole domain. Intuitively, increasing δ makes the landscape less bumpy and smooths out modes. In our algorithm, we assume a fixed δ and drop the symbol when the context is clear. We use simplified notations, e.g., $\mathcal{M}$, $\mathcal{N}(x)$, $\text{next}(x)$, etc. In practice, we decide δ empirically.

The main algorithm. Our main algorithm is summarized in Algorithm 1. We first estimate a discrete distribution, p. Next, we run the neighborhood search procedure starting from each data, s_i, and associate it to the mode it converges to. All points that are associated to a same mode are grouped into one cluster. Their cluster IDs, c_i's, are assigned accordingly.



(a) The neighborhood of (010) is highlighted with purple color.

(b) Each arrow points to the highest probability neighbor from each point.

Figure 2: A three-dimensional binary-valued discrete domain and the discrete gradient directions when $\delta = 1$. For the given probability in (b), there are two clusters (blue and red).

There are two issues that need to be addressed. First, we need a model to represent the estimated discrete distribution. This model should be easy to estimate from the data. Second, we need an efficient algorithm to compute the next step for any given point. An exhaustive search of the neighborhood is infeasible as the neighborhood of x, $\mathcal{N}_\delta(x)$, is a Hamming ball with exponential size.

To this end, we choose to use the tree-structured discrete graphical model as it strikes a good balance between the flexibility of the model and the computational efficiency. As we mentioned in the background section, a tree-structured model is a flexible model with an efficient estimation method, i.e., the Chow-Liu algorithm. Furthermore, the tree-structure can be exploited in any inference tasks. In our case, we introduce an efficient algorithm that computes the next step of any point y, $\text{next}(y)$, using a dynamic programming algorithm. Our algorithm, `Compute-Next`,

Algorithm 1 `Discrete-Clustering`

Input: Data set $\mathcal{S} = \{s_1, \ldots, s_N\}$
Output: Cluster label $\mathcal{C} = \{c_1, \ldots, c_N\}$

The set of modes $\mathcal{M} \leftarrow \emptyset, m \leftarrow 0$
Estimate a discrete probability distribution using the tree model (Chow-Liu in Sec. 2)
for all $s_i, i = 1, \ldots, N$ **do**
  $x \leftarrow s_i$
  $y \leftarrow \text{next}(x)$ using `Compute-Next` (Alg. 2)
  while $y \neq x$ **do**
    $x \leftarrow y$
    $y \leftarrow \text{next}(x)$ using `Compute-Next` (Alg. 2)
  end while
  if $y \notin \mathcal{M}$ **then**
    $m \leftarrow m + 1$
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{y\}$, y is the m-th mode
  end if
  $c_i \leftarrow$ the index of y in $\mathcal{M}$
end for

Algorithm 2 `Compute-Next`

Input: A tree G, a potential function f, a scale δ and a given configuration y
Output: $\text{next}(y, \delta) = \text{argmin}_{z \in \mathcal{N}_\delta(y)} f(z)$

Build a rooted tree as in Figure 3a
$V \leftarrow$ the post-order traversal of the tree
for $v \in V, v \neq \widehat{r}$ **do**
  $u \leftarrow$ the parent of v
  for all $\ell_v \in \mathcal{L}, \tau \in [0, \delta]$ **do**
    Compute $\text{MSG}_{v \to u}(\ell_v, \tau)$
  end for
end for
$f(y^*) \leftarrow \min_{\ell_r} \text{MSG}_{r \to \widehat{r}}(\ell_r, \delta)$
Recover y^* through backtracking
return y^*

efficiently searches through the exponential-size neighborhood of y and finds the point with the highest probability. Unlike previous heuristic methods, such as Iterated Conditional Modes (ICM) (Besag, 1986), our method exploits the tree structure and finds the optimal solution exactly.

3.1. Computing the Next Step for a Tree Model

By definition (Eq. (2.1)), the smaller the potential of a configuration is, the bigger its probability is. Therefore, the next step of a configuration y, $\mathrm{next}(y)$ is the configuration within the neighborhood of y with the minimal potential, formally,

$$y^* = \mathrm{next}(y) = \operatorname*{argmin}_{z \in \mathcal{N}(y)} f(z).$$

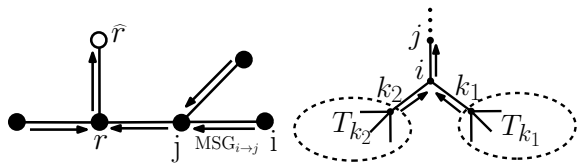
We compute y^* using a message-passing algorithm. We select an arbitrary node as the root, and thus a corresponding child-parent relationship between any two adjacent nodes. We add an extra pseudo node $\widehat{r}$ as the parent of the root, r. A message is a function that is passed from each node to its parent after collecting messages from all its children. See Figure 3a for an illustration of the message passing process. We visit all nodes and compute their messages in a post-order traversal (Cormen et al., 2001). This ensures that when we visit a node and compute its message, all its children have been visited.

Each message is a real-valued function with two parameters, a value and an integer. Denote by T_i as the subtree rooted at node i. The message from vertex i to its parent j, $\mathrm{MSG}_{i\to j}(\ell_i, \tau)$, is the minimal potential one can achieve within the subtree T_i given a fixed value ℓ_i at i and a constraint that the partial configuration of the subtree is no more than τ away from y_{T_i}, formally,

$$\mathrm{MSG}_{i\to j}(\ell_i, \tau) = \min_{\substack{z_{T_i}: z_i = \ell_i, \\ \mathrm{dist_H}(z_{T_i}, y_{T_i}) \le \tau}} f(z_{T_i}),$$

where $\ell_i \in \mathcal{L}$ and $\tau \in [0, \delta]$. Here the message does not depend on node j and the parameter τ is no more than δ. By definition, $\min_{\ell_r} \mathrm{MSG}_{r\to\widehat{r}}(\ell_r, \delta)$ is the potential of the optimal configuration, y^*. We recover y^* using the standard backtracking strategy of message passing. See Algorithm 2 for the pseudocode. It remains to explain details of the computation of each message and the backtracking.

Computing messages. For each $\ell_i \in \mathcal{L}$ and $\tau \in [0, \delta]$, we compute the message from i to j, $\mathrm{MSG}_{i\to j}(\ell_i, \tau)$. When



(a) (b)

Figure 3: Message passing.

i is a leaf node, the solution is trivial. When i has only one child, k, we compute the message as follows. If ℓ_i is equal to y_i, then we enumerate through all possible values of node k. For each value ℓ_k, we find the optimal partial configuration of T_k with the value ℓ_k at node k and within distance τ from y . Note that the potential of such configuration has been computed as the message from k to i, $\mathrm{MSG}_{k\to i}(\ell_k, \tau)$. For each of the optimal partial configuration of T_k, we extend it to a partial configuration of T_i using ℓ_i for node i. The potential is $f_{i,k}(\ell_i, \ell_k) + \mathrm{MSG}_{k\to i}(\ell_k, \tau)$. We compare potentials of all the L partial configurations, corresponding to L different value choices for node k, and select the smallest as our message. When ℓ_i is not equal to y_i, we know that any partial configuration we consider differs from y at i. Thus we only consider all partial configurations of T_k with value ℓ_k at node k and distance upperbound $\tau - 1$ from y. Unifying both $\ell_i = y_i$ and $\ell_i \neq y_i$, we have

$$\mathrm{MSG}_{i\to j}(\ell_i, \tau) = \min_{\ell_k} \left\{ f_{i,k}(\ell_i, \ell_k) + \mathrm{MSG}_{k\to i}(\ell_k, \hat{\tau}) \right\}, \tag{3.1}$$

where $\hat{\tau} = \tau$ if $\ell_i = y_i$ and $\hat{\tau} = \tau - 1$ otherwise.

In a more general case, when i has t children, $k_1, \ldots, k_t$ (Figure 3b), we consider partial configurations of the union of all subtrees $T_{k_1} \cup \ldots \cup T_{k_t}$ within distance $\hat{\tau}$ from y_{T_i}, where $\hat{\tau}$ is either τ or $\tau - 1$, depending on whether $\ell_i = y_i$ or not. We want to find the optimal partial configurations for all subtrees so that their total Hamming distance from y_{T_i} is no more than $\hat{\tau}$ and they jointly form a partial configuration of T_i with the minimal potential. In this case, we decompose the solution space into different subspaces, each corresponding to a tuple $(\tau_1, \ldots, \tau_t)$, specifying a distance upperbound for each subtree. For each tuple satisfying $\sum_{a=1}^{t} \tau_a = \hat{\tau}$, we find the optimal partial configuration for each subtree within the corresponding upperbound, τ_a. In total, there are at most $(\hat{\tau}+1)^{t-1}/((t-1)!)$ possible upperbound tuples satisfying the constraints that their sum is $\hat{\tau}$. However, we have an efficient way to search through all of them.

For the a's subtree with root k_a, and for each possible upperbound $\tau_a \in [0, \hat{\tau}]$, we compute the optimal potential of subtree T_{k_a} plus the potential of edge (i, k_a),

$$F(a, \tau_a) = \min_{\ell_{k_a}} \left(f_{i,k_a}(\ell_i, \ell_{k_a}) + \mathrm{MSG}_{k_a\to i}(\ell_{k_a}, \tau_a) \right) \tag{3.2}$$

This gives us $\hat{\tau} + 1$ different values for the a's subtree, associated with upperbound $\tau_a = 0, \ldots, \hat{\tau}$. Going through all t subtrees, we have a $t \times (\hat{\tau} + 1)$ matrix. For any tuple $(\tau_1, \ldots, \tau_t)$, the optimal potential of the partial configuration of T_i corresponds to selecting the τ_a's number from the a's row and sum them up.

Our goal is to decide the best tuple achieving a minimal total potential under the constraint that the total upperbound is $\hat{\tau}$. This can be solved using a standard dynamic

programming (DP) approach (Cormen et al., 2001). Formally, the optimization objective function is

$$\min_{(\tau_1,\ldots,\tau_t):\sum_{a=1}^{t}\tau_a=\hat{\tau}} \sum_{a=1}^{t} F(a,\tau_a) \tag{3.3}$$

Backtracking. In the backtracking process, we visit all nodes in a pre-order traversal (Cormen et al., 2001), so that each node is visited after its parent. For each node i, we fix an optimal pair of parameters (ℓ_i^*, τ_i^*). For the root note r, we fix $\tau_r^* = \delta$ and $\ell_r^* = \operatorname{argmin}_{\ell_r} \mathrm{MSG}_{r\to\hat{r}}(\ell_r, \delta)$. For any subsequent node, k, we fix the optimal parameters based on the optimal parameters fixed for its parent i, (ℓ_i^*, τ_i^*). When calculating the message of the parent using the fixed optimal parameters, we choose an optimal upperbound for each of its children (optimal τ_a's in Equation (3.3)). These numbers are used as τ^*'s of the children nodes $k_1, \cdots, k_t$. For each k_a, we fix the optimal value ℓ_{k_a} as the one minimizing Equation (3.2) with $\ell_i = \ell_i^*$ and $\tau_a = \tau_{k_a}^*$. Finally, all the ℓ_i^*'s give the optimal configuration y^*.

3.2. Complexity

Denote by d the degree of the tree. For each i and each parameter setting of the message (ℓ_i, τ), we compute the matrix of $F(a, \tau_a)$'s and run dynamic programming to compute Eq. (3.3). The table has $O(d\delta)$ entries. Computing each entry of the table take $O(L)$. Furthermore, running DP on the table takes $O(d\delta^2)$. Since we have $O(L\delta)$ input parameter pairs (ℓ_i, τ), the complexity of computing each message is $O(dL\delta^2(L+\delta))$. Therefore, the complexity of `Compute-Next` is $O(DdL\delta^2(L+\delta))$.

Our algorithm runs through every data point once. For each data, the gradient ascent path is computed, which include the computation of $\mathrm{next}(x)$ for K times, where K is the maximal number of gradient ascent steps a data point can take. Finally, we need to maintain a set of no more than N modes, each of which is a length D configuration. Therefore, the overall complexity of $O(N(KDdL\delta^2(L+\delta) + D\log N))$. Assuming δ is constant, our algorithm has $O(ND(KdL^2 + \log N))$. It is linear to the dimension and almost linear to the data size.

Parallelization. The desirable property of our method is that it can be very easily parallelized. Once the tree model is estimated. We can execute the neighborhood search for all data in parallel and record the modes they converge to. In the end, we just need to go through all data once and identify data associated with the same mode as one cluster. Being embarrassingly parallel makes our method rather appealing compared with others discrete clustering methods; to the best of our knowledge, none of these methods can be easily parallelized.

4. Experiments

We validate our method on various synthetic and real world categorical datasets. We focus on the clustering task and show that our method outperforms various existing clustering methods. We use synthetic, UCI and biological datasets. See Table 1 for a summary of different datasets.

Synthetic datasets. We create a synthetic categorical dataset with two elongated and two isotropic clusters. The domain has 110 dimensions. Each dimension can take 4 different categorical values ($L = 4$). We randomly corrupt 5% or 10% of the data, i.e., for each data point, 5% or 10% of its attributes are randomly selected to be corrupted. The value of each corrupted attribute is changed to a random value. In Figure 4, we visualize the data with 5% corruption rate.

UCI datasets. We use several categorical datasets from the UCI repository (Lichman, 2013), namely, Votes, Molecular Biology, Lymphography, Soybean and Mushroom. These data come from various domains such as social science, biomedicine and biology.

Biological datasets. We use DNA barcoding datasets from (Kuksa & Pavlovic, 2009). Each dataset is a collection of DNA samples with equal length/dimension. The attribute values of each dimension include different nucleobases such as cytosine (C), guanine (G), adenine (A) and thymine (T). These data are well aligned and thus are comparable. Overall, we have 5 different datasets: ACG,[2] Bats of Guyana, Birds, Fish of Australia and Hesperiidae. Each dataset is used twice, using the genera (denoted by "G") and the species (denoted by "S") of the DNA samples as the ground truth respectively.

4.1. Baseline Methods

We compare our method to various state-of-the-art clustering methods for continuous data, including K-Means (MacQueen, 1967), Dirichlet process Gaussian mixture model (DPGMM) (Rasmussen, 2000), affinity propagation (AP) (Frey & Dueck, 2007) and spectral clustering (SC) (Ng et al., 2002). We include these methods because they generally have computational advantage and can still perform reasonably well on discrete data (as shown in the results). To apply these continuous methods, we map a variable of L possible categorical values to L binary variables. The ℓ-th variable has value one if and only if the original variable has value l. This way we can map the categorical data into a binary-valued high dimensional domain. We further relax the domain into a high dimensional zero-one cube by allowing each variable to take real values between zero and one. This preprocessing delivers good baseline results for continuous methods

We also compare to methods designed for categorical data. We implemented baselines such as mixture of products of discrete distributions (MPD) using the pyMix package (Georgi et al., 2010). Note that the latent class analysis

[2] DNA samples from the species-rich fauna of Area de Conservacion Guanacaste (ACG) in northwestern Costa Rica.

(a) Groundtruth. (b) K-Means. (c) DPGMM. (d) AP. (e) SC. (f) K-Modes. (g) ROCK.

Figure 4: A synthetic categorical data with four clusters. Two of them are elongated clusters. The domain has 110 dimensions. Each dimension can take four discrete values. The data are plotted using multidimensional scaling so that their pairwise Hamming distance is approximated by their pairwise 2D Euclidean distance. (a): the groundtruth clustering, which is also our clustering result. (b) to (g): Results of other methods.

Table 1: Data Information: dimension (D), number of samples (N) and number of clusters (C).

Data	D	N	C	Data	D	N	C	Data	D	N	C
Synthetic	110	520	4	Soybean	35	307	19	Birds G/S	990	2589	289/658
Votes	16	435	2	Mushroom	22	8124	2	Fish G/S	901	754	113/211
Molecule	57	106	2	ACG G/S	663	4267	206/573	Hesp. G/S	664	2185	148/364
Lymph.	18	148	4	Bats G/S	659	840	50/96				

Table 2: Comparison in terms of NMI and running time.

Synthetic Data									
	K-Means	DPGMM	AP	SC	MPD	TMode	K-Modes	ROCK	Ours
5% Corrupted	0.75	0.75	0.73	0.08	0.72	0.63	0.74	0.47	**1.00**
10% Corrupted	0.75	0.74	0.72	0.05	0.70	0.63	0.74	0.47	**0.90**
UCI									
	K-Means	DPGMM	AP	SC	MPD	TMode	K-Modes	ROCK	Ours
Votes	0.50	0.52	0.35	0.32	0.49	0.40	0.46	0.34	**0.53**
Molecule	0.28	0.24	0.14	0.05	**0.43**	0.03	0.03	0.39	0.39
Lymph.	0.03	0.04	0.16	0.06	0.26	0.28	0.16	**0.41**	0.28
Soybean	0.69	0.67	0.67	0.38	**0.74**	0.68	0.59	0.55	0.68
Mushroom	0.37	0.36	0.33	0.17	0.14	0.41	0.33	0.28	**0.44**
DNA Barcoding									
	K-Means	DPGMM	AP	SC	MPD	TMode	K-Modes	ROCK	Ours
ACG G	0.60	0.49	0.53	0.42	0.63	0.42	0.75	0.76	**0.79**
ACG S	0.80	0.50	0.61	0.62	0.84	0.49	0.86	0.88	**0.89**
Bats G	0.81	0.79	**0.82**	0.39	0.84	0.49	**0.82**	0.72	**0.82**
Bats S	0.91	0.79	0.89	0.48	**0.92**	0.79	0.87	0.80	0.89
Birds G	0.61	0.35	0.48	0.40	0.78	0.16	0.80	**0.82**	**0.82**
Birds S	0.79	0.45	0.58	0.56	0.82	0.19	0.88	**0.90**	0.89
Fish G	0.88	0.44	0.89	0.59	0.84	0.77	**0.90**	0.84	0.88
Fish S	**0.94**	0.44	0.75	0.89	0.91	0.81	0.91	0.92	**0.94**
Hesp. G	0.61	0.43	0.45	0.47	0.70	0.13	0.75	0.78	**0.81**
Hesp. S	0.80	0.48	0.57	0.61	0.87	0.15	0.87	0.89	**0.90**
Average Running Time (seconds)									
	K-Means	DPGMM	AP	SC	MPD	TMode	K-Modes	ROCK	Ours
	30.2	138.9	112.4	1689.7	1872.5	829.5	473.9	2013.0	540.6

(LCA) method is only a special case of MPD. We compare to non-probabilistic methods including K-Modes (Huang, 1998) and ROCK (Guha et al., 1999).

Finally, we compare to TMode by Chen *et al*. (2014). The original method only computes modes. We decide the clusters by associating each data to its closest mode in terms of the Hamming distance. Note that TMode method is exponential to the degree of the tree model. For data set with high dimension, e.g., DNA Barcoding datasets, the average tree degree is 65. To ensure TMode finishes in a reasonable amount of time, we restrict the tree degree to no more than eight during model estimation and use this degree-restricted tree for TMode method. This restriction partially contributes to the relatively low performance of the TMode. This shows the advantage of our new algorithm, which runs on trees with arbitrary degree.

Metric and parameters. We compare all methods in terms of the normalized mutual information (Strehl & Ghosh, 2003), $NMI(X,Y) = I(X,Y)/\sqrt{H(X)H(Y)}$, which is commonly used for clustering evaluation. Our method automatically decides the number of clusters as the number of modes. The only parameter we need is the scale parameter δ. Empirically, we observe $\delta = 1$ is the best choice, although $\delta = 2$ and $\delta = 3$ also work well. For the competing methods, we provide the true number of clusters to K-Means, K-Modes and mixture models.

4.2. Results and Discussion

Results are reported in Table 2. For methods that depend on initialization, we run five times and report the mean score. We report the average running time in Table 2. Our method is reasonably efficient compared with others. The continuous methods are generally more efficient.

On synthetic data, our method outperforms other methods with a large margin. See Figure 4 for a visualization of several methods. Our method successfully identifies all four clusters. The main reason is that it is able to capture the probability gap between the modes even though they are close-by in terms of Hamming distance. Meanwhile, the whole elongated cluster becomes a single attractive basin. All data in this cluster gradient ascent to the right mode. Other methods fail to produce good results. K-Means and K-Modes cut each elongated cluster into two halves because they depend on distance based objective functions. DPGMM makes the same mistake, due to the limitation of the model; the elongated and bent cluster cannot be expressed by a single anisotropic Gaussian. Proximity-graph-based methods, such as affinity propagation and ROCK, tend to produce over-segmentations.

On real world data, our method outperforms other methods on a majority of the datasets. As expected, other categorical clustering methods, e.g., MPD, K-Modes and ROCK, are strong competitors. K-Means has good scores in general, although it is based on Euclidean distance.

5. Conclusion

In this paper, we proposed a new method to compute topographical features for high dimensional categorical data and used it in the clustering task. These topographical features provide a geometric description of the data, which can be very useful for analysis. In particular, our method has shown very good performance on the clustering task of a wide range of challenging real world datasets. This gives us the confidence to continue our effort in transforming it into a powerful tool for modern data analysis. Some of the interesting future directions are: 1) proving theoretical guarantees for the algorithm; 2) computation of other topographical features such as ridges and the Morse-Smale complex; and 3) extend to loopy graph and exploit the connection between topographical features and intrinsic graph structures, e.g., the soft-clique (Quadrianto et al., 2012).

Acknowledgment. We thank anonymous reviewers for valuable comments. We thank Vladimir Pavlovic, Han Liu, and Viktoriia Sharmanska for helpful discussions.

References

Bach, Francis R and Jordan, Michael I. Beyond independent components: trees and clusters. *The Journal of Machine Learning Research*, 4:1205–1233, 2003.

Besag, Julian. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 259–302, 1986.

Bishop, Christopher M. *Pattern Recognition and Machine Learning*, volume 4. springer New York, 2006.

Bradley, Paul S and Fayyad, Usama M. Refining initial points for k-means clustering. In *International Conference on Machine Learning (ICML)*, pp. 91–99, 1998.

Bubenik, Peter. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015.

Carlsson, Gunnar. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

Chazal, Frédéric, Glisse, Marc, Labruère, Catherine, and Michel, Bertrand. Convergence rates for persistence diagram estimation in topological data analysis. In *International Conference on Machine Learning (ICML)*, pp. 163–171, 2014.

Chen, Chao and Edelsbrunner, Herbert. Diffusion runs low on persistence fast. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 423–430. IEEE, 2011.

Chen, Chao and Kerber, Michael. Persistent homology computation with a twist. In *European Workshop on Computational Geometry*, volume 11, 2011.

Chen, Chao and Kerber, Michael. An output-sensitive algorithm for persistent homology. *Computational Geometry*, 46(4):435–447, 2013.

Chen, Chao, Liu, Han, Metaxas, Dimitris, and Zhao, Tianqi. Mode estimation for high dimensional discrete tree graphical models. In *Neural Information Processing Systems (NIPS)*, pp. 1323–1331, 2014.

Chen, Yen-Chi, Genovese, Christopher R., Tibshirani, Ryan J., and Wasserman, Larry. Non-parametric modal regression. *The Annals of Statistics*, 2015.

Cheng, Yizong. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

Chow, C and Liu, C. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

Cormen, Thomas H, Leiserson, Charles E, Rivest, Ronald L, Stein, Clifford, et al. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001.

Edelsbrunner, Herbert and Harer, John. *Computational Topology: an Introduction*. AMS, 2010.

Frey, Brendan J and Dueck, Delbert. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

Georgi, Benjamin, Costa, Ivan Gesteira, and Schliep, Alexander. Pymix-the python mixture package-a tool for clustering of heterogeneous biological data. *BMC Bioinformatics*, 11(1):1, 2010.

Guha, Sudipto, Rastogi, Rajeev, and Shim, Kyuseok. ROCK: A robust clustering algorithm for categorical attributes. In *International Conference on Data Engineering (ICDE)*, pp. 512–521, 1999.

Hastie, Trevor, Tibshirani, Robert, Friedman, Jerome, Hastie, T, Friedman, J, and Tibshirani, R. *The Elements of Statistical Learning*, volume 2. Springer, 2009.

Huang, Zhexue. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

Knowles, David A. and Ghahramani, Zoubin. Pitman yor diffusion trees for bayesian hierarchical clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(2):271–289, 2015.

Kuksa, Pavel and Pavlovic, Vladimir. Efficient alignment-free dna barcode analytics. *BMC Bioinformatics*, 2009.

Lichman, Moshe. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Linzer, Drew A, Lewis, Jeffrey B, et al. polca: An r package for polytomous variable latent class analysis. *Journal of Statistical Software*, 42(10):1–29, 2011.

Liu, Han, Xu, Min, Gu, Haijie, Gupta, Anupam, Lafferty, John, and Wasserman, Larry. Forest density estimation. *Journal of Machine Learning Research*, 12:907–951, 2011.

MacQueen, James. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pp. 281–297. Oakland, CA, USA., 1967.

McCutcheon, Allan L. *Latent class analysis*. Number 64. Sage, 1987.

Milnor, J.W. *Morse theory*, volume 51. Princeton University Press, 1963.

Minka, Thomas and Qi, Yuan. Tree-structured approximations by expectation propagation. In *Neural Information Processing Systems (NIPS)*, 2003.

Ng, Andrew Y, Jordan, Michael I, and Weiss, Yair. On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems (NIPS)*, pp. 849–856, 2002.

Nowozin, Sebastian and Lampert, Christoph H. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3-4): 185–365, 2010.

Quadrianto, Novi, Chen, Chao, and Lampert, Christoph H. The most persistent soft-clique in a set of sampled graphs. In *International Conference on Machine Learning (ICML)*, 2012.

Rasmussen, Carl E. The infinite gaussian mixture model. In *Neural Information Processing Systems (NIPS)*, 2000.

Strehl, Alexander and Ghosh, Joydeep. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.

Wainwright, Martin J. and Jordan, Michael I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Xu, Rui and Wunsch, Donald. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3): 645–678, 2005.

Zhang, Nevin L. Hierarchical latent class models for cluster analysis. *The Journal of Machine Learning Research*, 5: 697–723, 2004.