Learning to Cluster using Local Neighborhood Structure

Rómer Rosales[*+] ROMER@CSAIL.MIT.EDU
Kannan Achan[*] KANNAN@PSI.TORONTO.EDU
Brendan Frey[*] FREY@PSI.TORONTO.EDU

[*]Probabilistic and Statistical Inference Laboratory, University of Toronto, Toronto, ON M5S 3G4, CANADA
[+]Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA

Abstract

This paper introduces an approach for clustering/classification which is based on the use of local, high-order structure present in the data. For some problems, this local structure might be more relevant for classification than other measures of point similarity used by popular unsupervised and semi-supervised clustering methods. Under this approach, changes in the class label are associated to changes in the local properties of the data. Using this idea, we also pursue to *learn how to cluster* given examples of clustered data (including from different datasets). We make these concepts formal by presenting a probability model that captures their fundamentals and show that in this setting, learning to cluster is a well defined and tractable task. Based on probabilistic inference methods, we then present an algorithm for computing the posterior probability distribution of class labels for each data point. Experiments in the domain of spatial grouping and functional gene classification are used to illustrate and test these concepts.

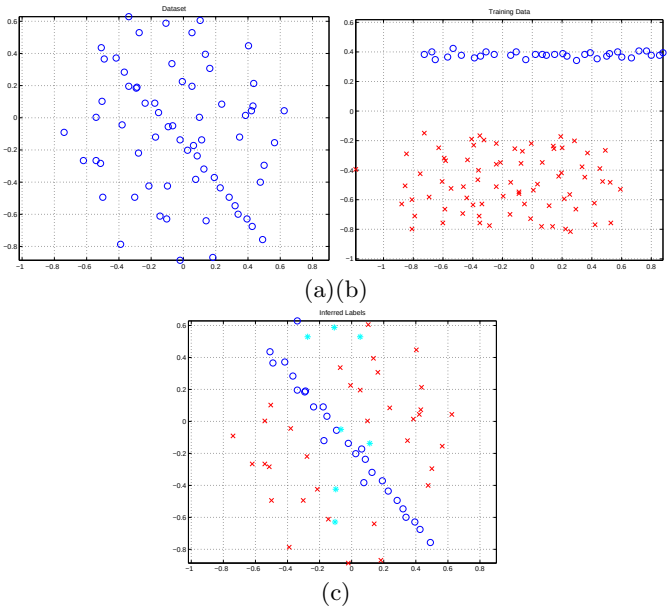

Figure 1. Illustration of the basic clustering/classification idea introduced. In the dataset in (a) the properties of the high-order local structure of the data are relevant for classification. A training set is provided (b) that allows us to learn how to cluster (and avoid the ill-defined concept of unsupervised classification) and produce the results in (c). Classes are shown as $\circ$ and $+$, uncertain labels are shown as a $*$ (blue, red, and cyan respectively if color enabled).

1. Introduction and Related Work

A fundamental problem in data analysis is that of clustering/classification with partly or only unlabeled data. This is known as semi-supervised or unsupervised classification respectively. Here, we will refer to *clustering* or *classification* as the problem of assigning a class label to (unlabeled) points in a dataset. Some well studied type of approaches for this task consider (sometimes implicitly) the information provided by the underlying *global* structure of the dataset.

In cases where there exists a relevant distance measure associated to the data coordinate system, pairwise distances could define a global function of the data (favoring certain properties, such as high intra-cluster or low inter-cluster similarity), that could be (approximately) optimized over all class assignments. In the case of unsupervised classification, a well known approach that uses this idea is spectral clustering (*e.g.,* (Shi & Malik, 2000; Ng et al., 2002)). Further, if the data is known to lie in a submanifold of the space, the structure of this submanifold might be critical for classification since it may provide a sometimes more meaningful way to

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

compute pairwise similarities for unsupervised or semi-supervised classification(Szummer & Jaakkola, 2002; Belkin & Niyogi, 2004). These concepts are not limited to clustering and have also been used as the basis for dimensionality reduction ((Roweis & Saul, 2000; Tenenbaum et al., 2000)). One way to formalize the role played by the global data distribution was recently proposed in the context of semi-supervised learning by introducing Information Regularization (Corduneanu & Jaakkola, 2003).

The above provides a rather general viewpoint, but will serve us to better illustrate the ideas introduced in this paper. The assumption in the above approaches was that changes in the class labels should occur in areas of low data density as opposed to areas of high data density, where class labels should remain the same. In this paper, we pursue a somewhat different notion. For some problems, the higher-order properties of the local data structure, rather than just its high/low density, could be the basis to correct classification. Under this concept, changes in class labels should be associated to changes in the local structure (also referred as *neighborhood structure* or *local topology*) of the dataset.

Global, unsupervised clustering algorithms usually suffer from the problem that a distance measure or class dependent pairwise likelihood is set in an ad-hoc way (*e.g.,* (Shi & Malik, 2000; Ng et al., 2002; Kannan et al., 2000)). Moreover, in many cases, it is set independently of the preferred clustering properties for the problem at hand. Semi-supervised classification provides a way around this problem by incorporating some labeled examples. In semi-supervised classification, the role of the unlabeled examples is generally to uncover, somehow, the structure of the data (*e.g.,* a low-dimensional manifold) (Szummer & Jaakkola, 2002; Belkin & Niyogi, 2004). In (Meila & Shi, 2001; Szummer & Jaakkola, 2002) the data points were associated to a Markov random walk whose transition matrix was computed *locally*; as a consequence it implicitly caused the unlabeled data density to play a major role in defining the pairwise similarity between points. The underlying assumption in these methods is that areas of low data density (rather than areas of high data density) should propitiate changes in cluster/class label. However, this is just one possible assumption on the joint distribution of class labels and data points.

In some clustering problems, what determines the point classes might be more related to the high-order (local) structure of the data. To illustrate this, let us use the simple example in Fig. 1(a). In general, most methods would produce a rather objectionable result for this example. In the case of an unsupervised clustering task, we might be motivated to cluster the data into a path of high density and a cloud formed by sparser *noise* points (Rosales & Frey, 2003). Of course, this might be just one subjective answer that many could agree upon in the absence of more information. However, it is useful at illustrating two ideas: (1) unsupervised clustering is in general ill-defined if we do not restrict the criteria used to characterize a *good clustering* and (2) the local data structure might be more relevant in order to distinguish clusters in some type of problems; as a consequence, different measures could be associated with each class. In this paper we focus on both of these ideas.

A reasonable approach to better define what we mean by clustering is to learn to cluster from examples (previously labeled data), such as that shown in Fig. 1(b). Different versions of this idea have been tried (Bach & Jordan, 2004; Xing et al., 2003), and it is in general a difficult problem. Successful clustering paradigms often involve a complicated function of the distance measure and the labels. Popular clustering approaches such as spectral clustering suffer from the fact that the problem of learning the distance measure from data, recently proposed in (Bach & Jordan, 2004), does not have a straightforward solution. Thus, approximations are needed even for the parameterized measure used in that approach. By using the point of view of Markov random walks, (Meila & Shi, 2001) also provided a rather elegant approach for learning the measure. In other contexts, such as the convex problem formulated by (Xing et al., 2003), learning distance measures is a less demanding task computationally. However, it all appears as if for computational tractability, the form of the measure and the type of objective function are rather limited. Our formulation provides a simple and natural way to learn about clusters from prelabeled data and produce results such as that shown in Fig. 1(c). In addition, none of these approaches attempts to learn or use the concept of class dependent distance measures introduced here.

In order to formalize the above and introduce our method for learning to cluster by using the local neighborhood structure (LC-LNS), first we define a generative model of data clustering that uses the local data structure to define what a cluster should be. We then explain how we could easily learn from previously labeled data using this framework and how to infer the posterior probability distribution over class labels for each data point. Finally we experimentally evaluate the methods and concepts introduced here by means of two different application domains.

2. Modeling Local Structure

Let $\mathcal{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_N\}$ be a set of observed data points which we would like to classify or cluster into classes or groups. Each point $\mathbf{z}_i$ is expected to belong to one of M classes. We will employ the random variable c_i taking values in the discrete space $\mathcal{C} = \{1, ..., M\}$ to represent the class associated to data point $\mathbf{z}_i$. Our goal will be to infer the class labels $C = (c_1, ..., c_N) \in \mathcal{C}^N$ of all points in $\mathcal{Z}$. We will consider local neighborhoods (groups of points) indexed by some set $\mathcal{G}$. Each neighborhood $\alpha \in \mathcal{G}$ will contain a subset of points from $\mathcal{Z}$ whose indices are given by the set η_α, with $\eta_{\alpha j}$ the j-th element of the neighborhood set. Denote the super set of all neighborhoods $\eta = \{\eta_\alpha | \alpha \in \mathcal{G}\}$. Associated to each local neighborhood α is a random variable $\mathbf{y}_\alpha$ that will *describe* the neighborhood; we let $\mathbf{y} = \{\mathbf{y}_\alpha | \alpha \in \mathcal{G}\}$. In other words, $\mathbf{y}_\alpha = f(\{\mathbf{z}_i\}_{i \in \eta_\alpha})$ for some function f. Our approach makes use of local neighborhood descriptions f, which in practice could be of diverse nature, and thus the domain of $\mathbf{y}$ could vary. For example, $\mathbf{y}_\alpha$ could encode simple binary relationships between data points in the neighborhood α (*i.e.,* $f(.) = \{f_{ij}\}$ for all pairs (i, j) of elements in any given neighborhood).

Since we consider the problem of classification, a neighborhood α will be assigned the random variable $\mathbf{x}_\alpha$ to describe its class. A particular aspect of this approach is the use of local high order descriptions of the data. In the extreme case, one neighborhood could be potentially formed by elements from the M different classes. Given our definition of $\mathcal{C}$, in this extreme case the domain of $\mathbf{x}_\alpha$ would be the Cartesian product space $\mathcal{C}^K$, where K is the neighborhood size. However, it is sensible to assume that *most* (*e.g.,* more than half) elements in a local neighborhood belong to a single common class, we will use $K_{in} < K$ to denote this (*in-class*) number. This is a reasonable assumption since it is equivalent to expecting that when a local neighborhood is chosen at random, most of the elements will belong to one particular (but unknown) class. Still, even under this representation, the number of ways to form a neighborhood by assigning a class to each point could be large. In this paper, we consider a local neighborhood representation where K_{in} points belong to a common class, and the rest ($K_{out} = K - K_{in}$) may be *out* of this common class.

Let $\mathcal{S}$ denote the domain of $\mathbf{x}_\alpha$. Using the representation above, $\mathcal{S}$ will have cardinality $M\binom{K}{K_{out}}$. This is, from K elements, choose any K_{out} elements to be considered *not part of* the common class. We will call this common class, the neighborhood class, with the caveat that sometimes not all the elements in a neighborhood belong to this class. It is natural to define $\mathbf{x}_\alpha = (\ell_\alpha, s_\alpha)$, with $\ell_\alpha \in \mathcal{C}$ representing the neighborhood class and s_α encoding the choice of in-class and out-of-class elements. We will use $s_\alpha = (s_{\alpha 1}, ..., s_{\alpha K})$, with $s_{\alpha i} \in \{0, 1\}$ denoting if its $i - th$ element is in the neighborhood class (1) or not (0). Finally, denote $\eta^1_\alpha(s_\alpha) = \{k | k = \eta_{\alpha j} \text{ and } s_{\alpha_j} = 1\}$ (*i.e.,* simply put, the set of indices of the points $\mathbf{z}_i$ that are in-class), likewise for η^0_α.

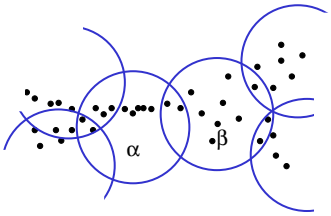

Figure 2. Two full neighborhoods and part of several others. The elements in each neighborhood are in the sets η_α and η_β respectively. The function f computes a description of the full neighborhood using its elements. The potentials ψ are a function of the points in overlapping areas. In this example α and β have different neighborhood structure.

2.0.1. Example

Let $K = 5$ and $K_{out} = 2$, then $s_\alpha = (0, 1, 1, 0, 1)$ is a valid state. Moreover, it denotes that as far as the random variable $\mathbf{x}_\alpha$ is concerned, from the five elements in α, the second, third, and fifth elements belong to class ℓ_α, the first and fourth ones may not belong to class ℓ_α but potentially to another class (they are *wild-cards*). In Sec. 3.1 it will be clear the role played by K and K_{out} in the trade off between algorithmic complexity and modeling power.

This representation allows us to define the conditional probability of a neighborhood description given its class in a simple form. In particular, we consider class-conditional mixture distributions of the form:

$$p(\mathbf{y}_\alpha | \mathbf{x}_\alpha) = \sum_\omega p(\omega | \ell_\alpha) p(\mathbf{y}_\alpha | \mathbf{x}_\alpha, \omega), \tag{1}$$

with ω as the index for the mixture component. Of particular interest will be:

$$p(\mathbf{y}_\alpha | s_\alpha, \omega) = \mathcal{N}(\mathbf{y}_\alpha(s_\alpha); \mu_{\ell_\alpha \omega}, \Sigma_{\ell_\alpha \omega}), \tag{2}$$

with $\omega \in \{1, ..., T\}$, T the desired number of mixture components, and $\mathbf{y}_\alpha(s_\alpha) = f(\{\mathbf{z}_i\}_{i \in \eta^1_\alpha(s_\alpha)})$ ($f(.)$ is computed using only the neighborhood elements $\mathbf{z}_i$ for which $s_{\alpha i} = 1$). This is a class-conditional Gaussian mixture defined on the range of f; thus the subscripts (l_α, ω) are simply the class and mixture indices for the mean and covariance.

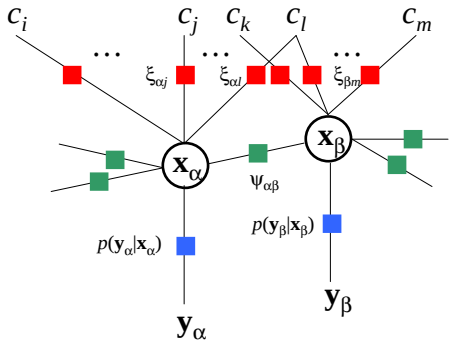

Figure 3. Example factor graph generated for a specific neighborhood instance. For simplicity we have ignored the class-dependent mixture random variables $\omega_{\alpha i}$.

So far, we have emphasized the use of attributes of local neighborhoods to characterize local structure. However, the larger the neighborhood the more complex its structure can potentially become. At a large enough scale, computational modeling can become infeasible. Generally speaking, some datasets may require a very complex model; however here we define a simple large scale model which is valid for our clustering approach and whose complexity is manageable. We choose to let multiple neighborhoods share elements; we can thus enforce large scale consistency simply by noticing that the class of a given point should agree among all the neighborhoods to which the point belongs. This is a fairly general statement and simple to specify formally.

We shall use a neighborhood-based Markov model, where two neighborhoods are probabilistically related if they share at least one element (in this case we say that the neighborhoods are *proximal*). Specifically, if $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$ are two proximal neighborhoods:

$$\psi(\mathbf{x}_\alpha, \mathbf{x}_\beta) \propto \exp\{-\sum_{(i,j)\in\mathcal{P}_{\alpha\beta}} \phi(s_{\alpha i}, s_{\beta j})\}^{\delta(\ell_\alpha \neq \ell_\beta)}, \quad (3)$$

where $\mathcal{P}$ indexes common element pairs, and $\delta(.)$ is the indicator function (see Fig. 2). We let $\phi(1,1) = 1$ and zero otherwise (as in the logical AND); thus, if two proximal neighborhoods have a different class, their compatibility decreases with the number of times their common elements disagree (variations on this could also be of interest). Note that using this definition, same-class neighborhoods pairs are equally compatible, independently of the number of shared elements.

2.1. Learning Neighborhood Structure

One of the advantages of this setting is that learning to cluster from labeled data can be easily defined since we only need to learn class-dependent neighborhood distributions[1]. Let us for the moment assume that, besides the unlabeled dataset $\mathcal{Z}$, we were given one or more labeled datasets $\mathcal{L}^i$, consisting of data points and their corresponding labels. We can formulate the learning problem as that of estimating class-dependent local neighborhood structure. This is possible for a number of distributions $p(\mathbf{y}|\mathbf{x})$, including distributions of the form in Eq. 1(*e.g.,* using the EM algorithm).

Given a set of neighborhoods $\eta^i = \{\eta_1^i, ..., \eta_{L_i}^{(i)}\}$, for each labeled dataset i, we fit the corresponding distribution using a maximum likelihood approach, assuming that neighborhoods and datasets where drawn independently at random. Using this criterion, the goal is to maximize:

$$p(\mathcal{L};\theta) = \prod_{i=1}^{} \prod_{\alpha=1}^{L_i} \prod_{c} p(\mathbf{y}_\alpha^i; \theta_c), \quad (4)$$

for each class c with parameters θ_c. From Eq. 2, $\theta_c = (\mu_{c\omega}, \Sigma_{c\omega}, p(\omega|c))$ for each ω. Eq. 4 factorizes across neighborhoods and classes; thus, learning becomes a simple task. Note that the neighborhood structure distributions are defined in a K_{in} dimensional space since ℓ_α is given (*i.e.,* the data is labeled).

3. Inferring the Clusters

We should remark that $\mathbf{x}_\alpha$ defines the state of the neighborhood α as a whole. So far no explicit concept has been associated to the class of individual data points $C = (c_1, ..., c_N)$, since the different neighborhoods that a point may belong to could have different classes. In order to make the class concept explicit, we define the potential ξ as follows:

$$\xi(c_i, \mathbf{x}_\alpha) = \begin{cases} 1 & \text{if } c_i = \ell_\alpha \\ 0 & \text{otherwise;} \end{cases} \quad (5)$$

this potential is incorporated in the model only if data point i is in the neighborhood α. The full joint probability over all the states and classes in the neighborhoods is then given by:

$$P(\mathbf{y}, \mathbf{x}, C) = \frac{1}{Z} \prod_{\alpha} p(\mathbf{y}_\alpha|\mathbf{x}_\alpha) \prod_{(\alpha,\beta)\in \text{proxim.}} \psi(\mathbf{x}_\alpha, \mathbf{x}_\beta) \prod_{(\alpha,i)} \xi(c_i, \mathbf{x}_\alpha) \quad (6)$$

A factor graph depicting the relationships between the defined random variables is shown in Fig. 3. A reason for formalizing our model using a factor graph is that complexity analysis and general algorithms for inference can be easily derived from it.

[1]Intra class dependencies could also be studied, but this is not considered in this paper.

3.1. Complexity Analysis and Inferring Neighborhoods

As we have seen above, $\mathbf{x}_\alpha$ is a discrete random variable, defining the state of a neighborhood α. It takes values in the set $\mathcal{S}$, with $|\mathcal{S}| = M\binom{K}{K_{out}} < MK^{\min(K_{in},K_{out})}$. If we do not allow elements without a class (*i.e.,* $K_{out} = 0$), then the representation for $\mathbf{x}_\alpha$ reduces to a more common representation (we can think of the neighborhood in a similar way as we would think of a single point). Using the neighborhood idea, this would mean that all the elements in the neighborhood are assumed to belong to a single class (*i.e.,* $|\mathcal{S}| = \mathcal{C}$). However, this assumption is too restrictive since the final clustering assignment would be constrained to agree with the subdivision (neighborhoods) given by η.

An ideal approach would be to let the neighborhood definitions unknown and try to discover them. This is a very complicated combinatorial problem by itself. However, our approach is designed with this issue in mind, and is thus geared towards solving this particular problem, but with some constraints on the neighborhoods η allowed.

Specification of the neighborhood is done (implicitly) using our representation. Since the elements i in the neighborhood for which $s_{\alpha i} = 0$ are not modeled by $p(\mathbf{y}_\alpha|\mathbf{x}_\alpha)$, it is as if the neighborhoods were non-static and specified by the s_α itself; $\mathbf{x}_\alpha = (s_\alpha, \ell_\alpha)$ is indeed a representation for neighborhood element ownership and class label. Thus, neighborhoods of the type represented by s_α can be accounted for during inference.

3.2. Algorithms for inference

Here our goal is to find a posterior distribution $p(c_i|\mathbf{y})$ for each point index i. This is a well studied problem once the joint probability distribution has been specified. However, for the distribution specified in Eq. 6, there is no known exact algorithm for efficiently computing the desired posteriors. We will resort to the sum-product algorithm (Pearl, 1988; Kschischang et al., 2001), which is not exact in the factor graph associated to Eq. 6 (due to loops introduced by the dependencies), but has been shown to perform surprisingly well in such graphs (*e.g.,* (McEliece et al., 1998)). Also, empirically it has been found to perform well in models with a large number of hard constraints.

The sum-product algorithm is a message passing method that computes local updates on the marginal posterior probabilities based on local dependencies between variables. The sum-product update equations are equivalent to iteratively solving self-consistent equations resulting from finding the zero-gradient points of the *Bethe* approximation to the Gibbs free energy associated to the joint probability in Eq. 6 (Yedidia et al., 2000).

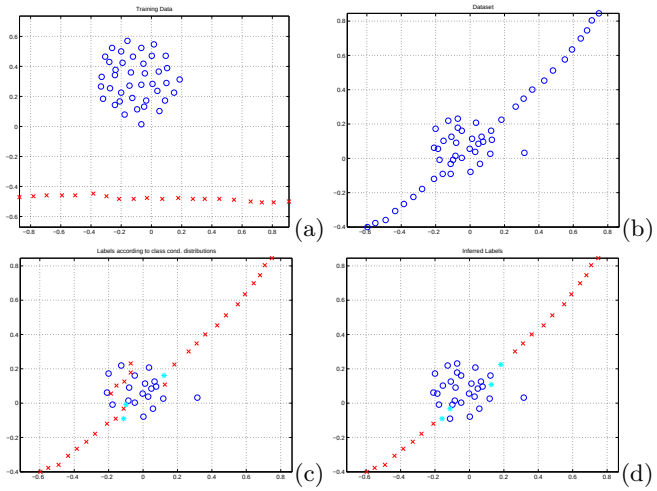

Figure 5. Illustration that the class likelihood alone is not enough for inference, thus the information across neighborhoods is needed. (a) shows the example dataset, (b) shows the problem at hand, (c) shows inference using the class likelihoods alone, and (d) shows full inference.

In our particular implementation, we randomly choose the neighborhoods. This initial step defines the factor graph to be used. We use the defined class-conditional probabilities to compute messages to each variable $\mathbf{x}_\alpha$. The posterior probabilities for $\mathbf{x}_\alpha$ are initialized according to this likelihood, then a parallel version of the sum-product algorithm is used to update the posterior distributions in the subgraph associated to the $\mathbf{x}_\alpha$ variables. The marginal posterior distributions for the classes c_i are then computed by marginalizing with respect to the states and multiplying appropriately (also equivalent to sum-product messages).

4. Experimental Results

To evaluate these concepts and algorithms, we employed datasets where local structure seems relevant for classification: visual/spatial clustering, sampled-manifold uncovering, and gene function prediction.

4.1. Learning Spatial Clustering and Uncovering Sampled Manifolds

In this set of tests we used synthetic spatial data in two dimensions. The scaling properties of the algorithms do not depend on the dimensionality of the elements in the dataset; we have used two dimensions

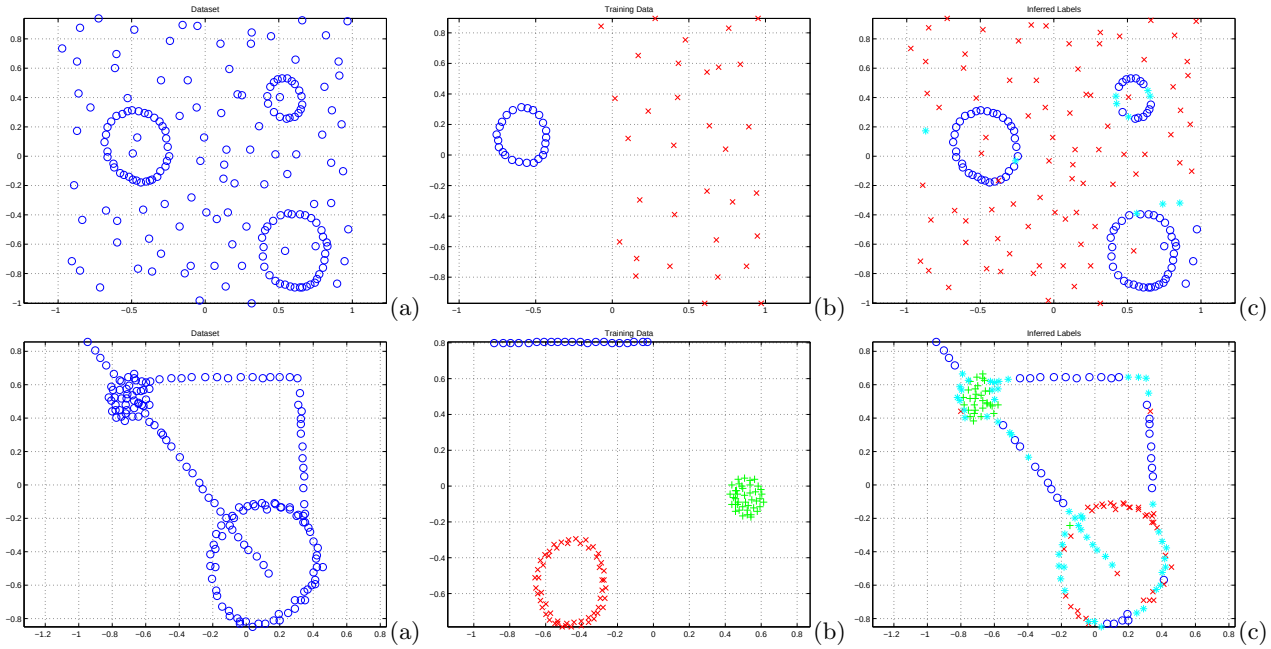

Figure 4. Two of the toy datasets used for our tests. Input datasets (a), datasets used for learning to cluster (b), and inferred labels (c). In each graph, class labels are color and symbol coded. A cyan $*$ denotes a high entropy posterior for the particular point (low confidence).

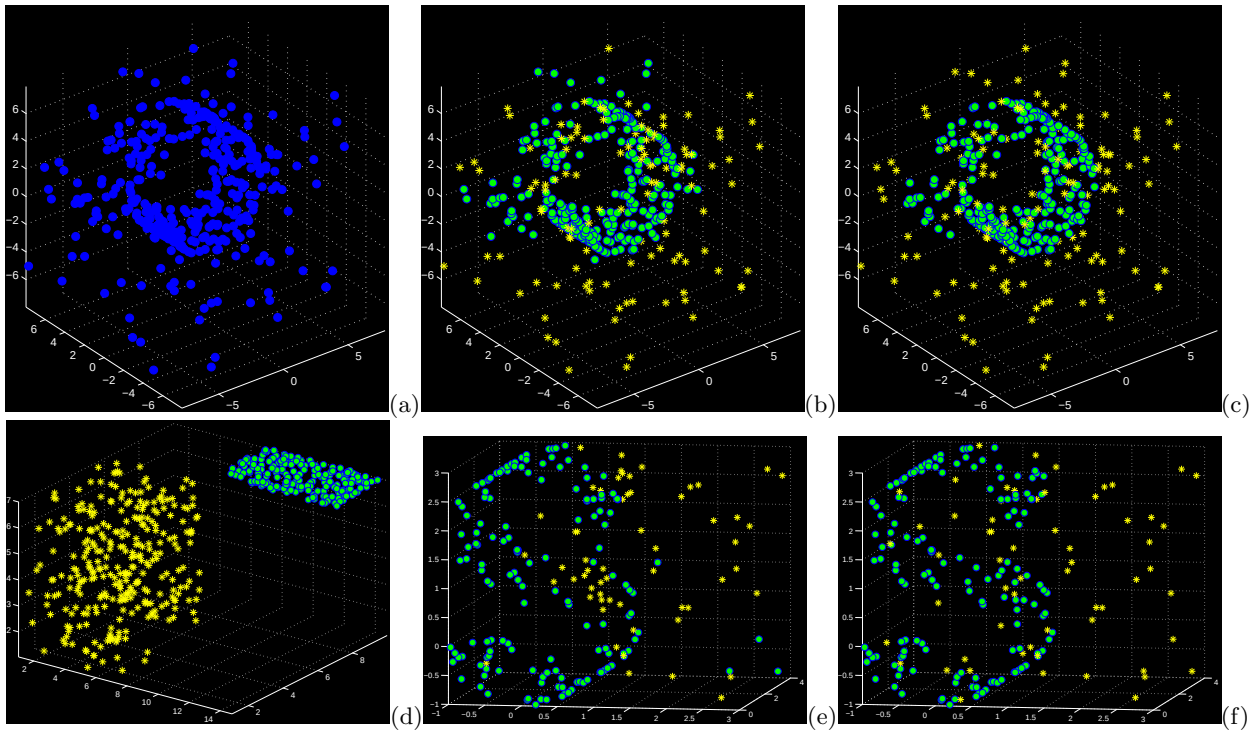

Figure 6. Uncovering sampled manifolds hidden in noise. Input Swiss-Roll sampled manifold + noise (a), inferred classification (b), ground-truth (c), training data for both Swiss-Roll and S-Manifold (d), S-Manifold inferred classification, S-Manifold ground-truth.

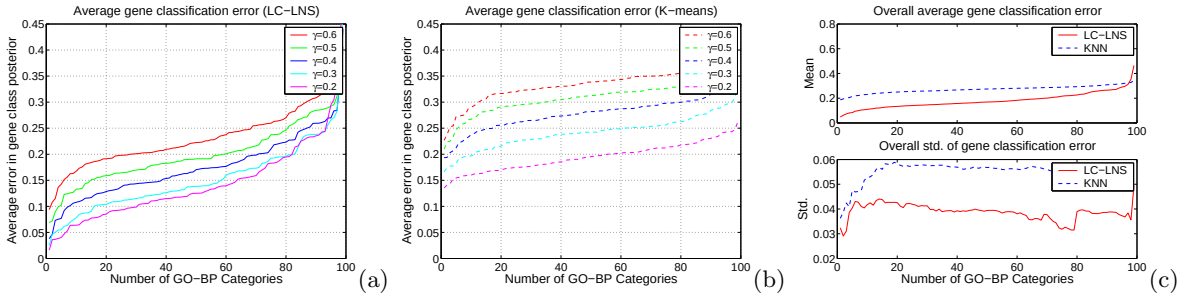

Figure 7. Gene classification results for (a) LC-LNS (our method), (b) KNN classifier for different decision proportions γ shown in decreasing order starting at the top curve (and in color if available), and (c) for overall error and standard deviation across experiments. The graphs show the sorted (not the cumulative) error values across GO-BP categories.

to ease visualization. We set $K = 10$, $K_{out} = 3$, and $f : \Re^K \rightarrow \Re^D$ to be the collection of pairwise (Euclidean) distances between the elements in the neighborhood; thus $D = K(K-1)/2$ (f provides a description of the neighborhood structure as a vector in $\Re^D$). We sorted the vector components to make the representation invariant to ordering and normalized them to achieve local scale invariance. We chose the neighborhoods to be centered at random point locations and to be defined by the K nearest neighbors. The number of neighborhoods was set to 60% of the number of data points. For learning, the class conditional neighborhood distributions were set as Eq. 2 with $T = 3$. The algorithm does not require any further parameter setting since the rest can be learned from data. Several datasets are shown in Figs. 1,4, and 5.

Using the same settings, we applied the algorithm to sampled versions of well-known test manifolds hidden in higher dimensional *noise*. Fig. 6 shows a couple of example experiments. The results are of good-quality even in noticeably complex patterns considering the combinatorial nature of the task. These examples illustrates potential use of this method in data visualization as well.

4.2. Functional Gene Classification

In our next set of experiments our goal is to correctly classify gene function based on gene expression data. We used mouse gene expression data obtained under a series of different (55) experimental conditions. The genes in this dataset were determined using GenomeScan, generating a set of 41K putative gene sequences which were used to produce DNA microarrays containing 60-mer oligonucleotides (see (Zhang et al., 2004)). We used Gene Ontology Biological Process (GO-BP) annotations (Ashburner et al., 2000), which defined a gene classification criterion (related to gene function). We set $K = 8$; K_{out}, f, T, and the number of neighborhoods were as in the previous experiment.

Our reason to believe that the classification concepts here presented are at all useful in this scenario is related to the possibility that gene function could be predicted by the pattern of gene expression in which they are involved and that this pattern might be shared by same-function genes. Thus, different classes might (presumably) be distinguished by their collective patterns of gene expression.

We considered 99 GO-BP categories, those for which the number of labeled genes was at least 80 (for statistical significance) and those which did not contain a high number of labeled genes (because those are too broad in function). We randomly partitioned each category into two sets of genes, 80% for training and the rest for testing, and built binary classifiers using the method described in this paper. The error was measured using the mean absolute difference between the inferred posterior probability for the class label found by our method and the (1/0) probability derived from ground-truth. This experiment was repeated a number of times (10). Since we were faced with the question regarding *when a gene function should be predicted*, we set our algorithms to make decisions about the class of at least a γ proportion of the (unlabeled) genes observed. This was done by choosing to classify the genes with highest confidence (posterior probability).

Figs. 7(a)-(b) show classification results for our method (LC-LNS) and a K-nearest neighbor classifier (KNN) respectively. In addition to LC-LNS being superior overall, the methods had very distinct performance properties. LC-LNS performed really well in many classes (classes whose genes could be uncovered by considering their collective expression pattern) and really badly in few classes (presumably those which could not), whereas KNN performance was relatively even across classes but not excellent for any of them. Fig. 7(c) summarizes the overall performance across

experiments and γ values, showing a lower standard deviation on the prediction error overall for LC-LNS. These experiments suggest that the expression pattern of groups of genes (rather than only the similarity in gene expression) is important for correct functional gene classification for some categories. Further, it is worth pointing out that LC-LNS is more general, since we could potentially perform gene classification across species by learning the function patterns in one species to classify genes, by the same functions, in another.

5. Discussion and Future Work

In this paper, we have introduced a general classification concept driven by the idea that, in some problems, high-order local structure of the data could be relevant for classification. Under this concept, changes in class label are associated to changes in local data structure. Many successful unsupervised and semi-supervised classification methods are (implicitly or explicitly) built upon the idea that changes in class labels should occur in areas of low data density. From this viewpoint, this work establishes a connection between these two type of approaches.

We presented a fully probabilistic model based on this classification idea and derived a simple and natural criterion for learning to cluster. The proposed method also gave us the freedom to account for class dependent cluster properties, unlike previous methods which have used a global distance measure (not class specific). Based on results from probabilistic inference, we used the sum-product algorithm for computing the posterior distributions of class labels.

There are several extensions and ideas that we believe are worth pursuing: (1) For certain problems it might be relevant to learn the inter-class neighborhood structure rather than just the within-class structure, (2) A more interesting definition of ψ could be one that allows the preservation of the neighborhood structure itself. Interesting further applications of this method include problems related to inferring graphs (*e.g.,* graph denoising and related areas). The concepts presented here extrapolate almost naturally to graphs, since neighborhood structure could be seen as graph connectivity structure.

Acknowledgments

We thank Quaid Morris for providing valuable remarks regarding our experiments using GO-BP categories and for making the gene data available to us. We also thank our reviewers for their helpful comments.

References

Ashburner, M. et al. (2000). Gene ontology: tool for the unification of biology. The gene ontology consortium. *Nat. Genet.*, *25*, 25–29.

Bach, F. R., & Jordan, M. I. (2004). Learning spectral clustering. *Neural Inf. Processing Systems*.

Belkin, M., & Niyogi, P. (2004). Semi-supervised learning on Riemannian manifolds. *Journal of Machine Learning Research (to appear)*.

Corduneanu, A., & Jaakkola, T. (2003). On information regularization. *Uncert. in Artificial Intelligence*.

Kannan, R., Vempala, S., & Vetta, A. (2000). On clusterings: good, bad and spectral. *41st Foundations of Computer Science (FOCS 00)*.

Kschischang, F., Frey, B., & Loeliger, H. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*.

McEliece, R., MacKay, D., & Cheng, J. (1998). Turbo decoding as an instance of pearl's belief propagation algorithm. *IEEE J. Sel. Areas in Comm.*, *16*.

Meila, M., & Shi, J. (2001). Learning segmentation with random walks. *Neural Inf. Processing Systems*.

Ng, A., Jordan, M., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Advances in Neural Inf. Processing Systems*.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. Morgan-Kaufman.

Rosales, R., & Frey, B. (2003). Generative models of affinity matrices. *Uncert. in Artificial Intelligence*.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence*, *22*, 888–905.

Szummer, M., & Jaakkola, T. (2002). Partially labeled classification with markov random walks. *Neural Inf. Processing Systems*.

Tenenbaum, J., Silva, V. D., & Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*.

Xing, E., Ng, A., Jordan, M., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. *Neural Inf. Processing Systems*.

Yedidia, J., Freeman, W., & Weiss, Y. (2000). Generalized belief propagation. *Neural Inf. Processing Systems* (pp. 689–695).

Zhang, W. et al. (2004). The functional landscape of mouse gene expression. *Submitted*.