Efficient Algorithms and Information Geometric Approach for Kernel-based Clustering

Graduate School of Systems and Information Engineering

University of Tsukuba

March 2008

Ryo Inokuchi

Contents

List of Figures

List of Tables

Acknowledgement

This thesis contains work that I did during the years 2003-2008 at graduate school of systems and engineering, University of Tsukuba. During that period I received a lot of help from faculty, students and friends.

First of all, I would like to thank gratefully and sincerely my supervisor Professor Sadaaki Miyamoto. He allowed the freedom to study this topic, and gave provided many valuable suggestions and technical hands-on assistances. It fundamentally changed the way I do research and turned me into a better researcher. I also thank him for providing an excellent environment for research without distractions and for making himself available whenever I needed.

I would also thank my thesis committee members Professor Toshiyuki Inagaki, Professor Takehisa Onisawa, Dr. Yasunori Endo and Dr. Mika Sato-Ilic for reviewing this thesis in detail and for giving me many valuable comments to improve this thesis. I would thank again Dr. Yasunori Endo, and Dr. Mika Sato-Ilic who provided valuable discussions, brilliant ideas, and helpful support.

I benefited from interactions with several graduate students at soft computing research group, University of Tsukuba. Yohei Kuroda, Kenta Arai, and Yuichi Kawasaki for helpful comments on different parts of the thesis. Older members Kiyotaka Mizutani, Hideyuki Haruyama and Ryuichi Murata provided much information about the doctoral studies. Despite not helping directly on topics related to the thesis, I benefited from interactions with Yukihiro Hamasuna, Tetsuya Nakamura, and Wataru Hashimoto. These interactions improved my understanding of my study and helped me write a better thesis.

Finally, I would like to thank my family for their dedication and the many years of support.

Chapter 1

Introduction

1.1 Background

In recent years, data from many natural and social phenomena are stored into huge databases in the world wide network of computers. For example, biological functions are coded as sequences of DNA and RNA [18], and market behaviors are expressed as time-series data [15]. However, it is difficult to analyze by hand since data are massive and complex. Hence, advanced data analysis techniques to get valuable knowledge from data using computing power of today are required. *Machine learning* [7] is an area in which new statistical methods for specific purposes have been actively studied.

In the context of machine learning, *supervised learning* is to predict an output for unseen data based on a given set of objects whose element $z = (x, y)$ is a pair of an input $x \in \mathcal{X}$ and an output $y \in \mathcal{Y}$, where $\mathcal{X}$ is referred to as the *input space*, and $\mathcal{Y}$ is referred to as the *output space*. If y is discrete, the problem is called *classification*, while the problem is called *regression* if y is continuous. In the classification problem, an output y shows the class or category for an input x. However, it is very expensive to get a large amount of labeled data since the labeling is often done by hand.

There are larger amounts of unlabeled data, and it needs very low-cost to obtain them. *Unsupervised learning* is based on only a set of objects whose element is an input x. Intuitively, unsupervised classification is a very difficult problem. Therefore, the purpose of unsupervised learning techniques is to extract features and structures behind data, rather than classification.

Data clustering is the most popular unsupervised classification method. Clustering is to divide a set of objects into some categories which are referred to as *clusters*. It should be noted that a *class* in supervised learning and a *cluster* in unsupervised

learning is similar, but quite different. A class in supervised learning is considered as an artificial category given beforehand, while a cluster is considered as a more natural category based on a natural similarity between objects.

Machine learning methods can be significantly improved by prior knowledge.

The most commonly used prior knowledge is a generative process of data. Suppose that all variables are generated from probability distributions. In the classification problem, *generative model* is to estimate class-conditional probability $p(y|x)$ using generative processes. It is derived from the property $p(y|x) \propto p(x|y)p(y)$ given by Bayesian framework. Finally, we assign the label having the maximum class-conditional probability $p(y|x)$.

However, it is very difficult to assume appropriate distributions and processes. If assumptions are not suitable, the result of analysis is insignificant. This issue is one of the motivations in this dissertation.

Another learning paradigm is *discriminative model*. It is to design a deterministic function $y = f(x)$ or estimate $p(y|x)$ directly. Since generative processes are not considered, almost all discriminative classifiers are heuristic. However, there are a few successful discriminative classifiers such as support vector machines (SVMs) [10]. They have the better performance than generative models, without any generative assumptions [35].

Clustering techniques are also classified into generative models and discriminative models.

The method of *Hard c-means* (K-means) [32] is the most popular discriminative clustering method. Hard c-means is to divide a set of objects into c clusters using a simple procedure. Assignments to clusters are deterministic, that is, an object x is assigned to one cluster exclusively. There are many extensions of hard c-means.

In contrast, a *mixture model* [34] has been frequently used to model clusters. It is a mixture of several probability distributions, and each component is considered as each cluster. The difference from hard c-means is that assignments are probabilistic using $p(y|x)$.

Methods of *fuzzy c-means* [37] has been remarked by many researchers. It is an extension of hard c-means using the concept of fuzzy sets. It should be noted that methods of fuzzy c-means do not require the details of fuzzy set theory.

Methods of fuzzy c-means have a property that assignments are soft despite not being a generative model. In other words, it does not assume a specific distribution.

To import prior knowledge into classifiers, a generative model is thus very useful.

There is, however, another approach to import prior knowledge. *Kernel methods* [46, 20] are very effective methods to extend discriminative classifiers such as SVMs. It doesn't require a generative model, and it requires only knowledge of the similarity between objects.

The basic idea of kernel methods is as follows. To represent rich information, data in the input space $\mathcal{X}$ are mapped into the *feature space* which is often high-dimensional. In general, it allows $\mathcal{X}$ not to be the Euclidean space, for example, graph, string, and structural data. It thus has a very useful property in application. If data in the high-dimensional feature space are directly used, the computational effort and the capacity of memory are not enough. The nice property of kernel methods is that it can compute the similarity in the feature space using a certain function defined in the input space. It has already been observed that kernel methods frequently allow a linear classifier to be a nonlinear classifier.

Since hard c-means is a linear classifier, cluster boundaries are linear. *Kernel c-means* [19, 39] and kernel fuzzy c-means [40] can obtain nonlinear classifiers. However, it has the high computational effort.

In kernel methods, $\mathcal{X}$ can be a non-Euclidean space, which implies that an adequate kernel for non-Euclidean data should be designed. There are so many designed kernels for graph, sequences and text [48]. However, this type of the kernel requires expert's knowledge for the similarity between the objects.

A more natural construction of the kernel is the *Fisher kernel* [25], which is derived from generative processes. The Fisher kernel is an inner product between the parameter derivative of log-likelihood. If appropriate distributions are given, this kernel allows discriminative classifier to obtain the superior performance. The Fisher kernel has advantages of both the generative model and the discriminative model. However, it is also crucial to give a suitable distribution.

From the view of the *information geometry* [2], let us consider the feature space of the Fisher kernel. In this space, a datum is expressed as a single probability distribution: $\mathrm{x} \to p(x)$, that is, datum is corresponding to a single parameter of the parameterized probability distribution $p(x|\theta)$. The parameter space is often not the Euclidean space. It is defined as a *Riemannian manifold*, and an inner product is computed by the Riemannian metric. The Fisher kernel is an inner product on the tangent space which is an approximation of the parameter space.

A more precise kernel on the manifold is the *information diffusion kernel* [31]. It is a kind of the heat kernel on the Riemannian manifold. However, it is quite difficult to

derive the kernel analytically, although we can obtain complete kernels from certain types of distributions.

1.2 Motivation

Kernel methods are very useful in application of which a typical example is the support vector machine. However, there are two crucial issues. First, kernel-based clustering algorithms such as kernel hard c-means have high computational efforts. It is not practical when massive data should be analyzed. Second, although kernels based on generative models have sound mathematical frameworks and usefulness in application to the non-Euclidean data, these methods still suffer from the difficulty of building specific models including parameter estimations and model selections.

For the first issue, we propose another approach to reduce the same risk functional of kernel hard c-means. This approach is called *on-line learning*. A new algorithm is derived from deep observations of on-line learning. After that, another algorithm is obtained from the view of the *reproducing kernel Hilbert space*. These algorithms are faster than kernel hard c-means, and the performance is as good as the kernel fuzzy c-means.

Another approach to an efficient algorithm of kernel-based clustering is to introduce sparseness, that is, a major part of memberships of kernel hard c-means is set to a specific value. Since solutions of kernel hard c-means are linear combinations of a set of objects with memberships, it is possible to reduce computations of solutions.

For the second issue, we consider problems in two specific kernels: the Fisher kernel and the information diffusion kernel.

Inspired by the Fisher kernel, we propose a new kernel using a relation between methods of fuzzy c-means and mixture models. Since our kernels can be constructed of fuzzy clusters and without a specific probability distribution, we can obtain a way to design a kernel from a more flexible framework than a generative model.

Although possibility to clustering on a manifold is to use a natural distance defined on that manifold. We consider data on hypersphere that arises in applications. Instead of the information diffusion kernels, we apply the Kullback-Leibler divergence and the Hellinger distance to hard c-means directly. These two metrics are approximations of the geodesic distance.

1.3 Outline of this dissertation

This dissertation is divided into two parts and a conclusion. We divide kernel methods for clustering into kernel machines and kernel designs.

In the former part, we discuss efficient algorithms of kernel clustering machines. In the latter part, we discuss information geometrical approaches for kernels.

The outline of this dissertation is as follows.

As preliminaries, Chapter 2 presents fuzzy clustering algorithms, and Chapter 3 presents the basic property of kernel methods.

In Chapter 4, we propose on-line learning of kernel-based clustering.

In Chapter 5, we discuss the advanced topics of clustering. Sparseness is very important property in machine learning context. We introduce sparseness into fuzzy clustering.

In Chapter 6, we design the kernel using cluster information. This kernel is similar to the Fisher kernel, but it doesn't depend on generative models since it uses the concept of fuzziness.

In Chapter 7, we discuss the design of the dissimilarity on the multinomial manifold. Inspired by the information diffusion kernel, we apply the KL-divergence and the Hellinger distance to fuzzy clustering.

Finally, in Chapter 8, the conclusion of this dissertation summarizes the proposed methods, their features and significances, and problems to be solved in near future.

Chapter 2

Methods of c-Means Clustering

This chapter presents methods of hard c-means and fuzzy c-means. After clustering is defined as a learning problem, two elementary algorithms, competitive learning and hard c-means are shown. Each algorithm is derived from a different optimization method. And then, we introduce fuzzy c-means algorithms including the standard fuzzy c-means and the possibilistic clustering. We especially deal with the entropy-regularized hard c-means. It has a statistical property that it is equivalent to the EM algorithm for mixture models.

2.1 Risk Minimization Framework

First, we define clustering as a learning problem [20, 54].

There are n pairs of an input $x \in \mathcal{X}$ and the corresponding output $y \in \mathcal{Y}$, where $\mathcal{X}$ is often referred to as an input space, and $\mathcal{Y}$ is referred to as an output space. $\mathcal{X}$ is usually R^p, and $\mathcal{Y}$ depends on the objectives. In the case of classification problems, $\mathcal{Y}$ is discrete and y is the class label corresponding to x. In the case of regression problems, $\mathcal{Y}$ is continuous and y is the objective variable corresponding to an explaining variable x. Clustering is regarded as one of classification problems.

Let us consider that one seeks a relation between an input and the output, $y = f(\boldsymbol{x})$ from a given set of objects $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^p$. In other words, one finds an

optimal function f^* from a function space $\mathcal{F}$. However, $\mathcal{F}$ is too big, and therefore it should be restricted by a particular class of $\mathcal{F}$. Let $f(\boldsymbol{x}, \boldsymbol{w})$ be a restricted class which is parameterized by a parameter vector $\boldsymbol{w} \in \mathcal{W}$.

The goal of a learning problem is to find an optimal parameter $\boldsymbol{w}^*$. A guiding principle for learning problems is to minimize the *expected risk*.

Definition 1 (**Expected Risk**) Let $\mathcal{L} : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ be a loss function.

$$\mathcal{R}_{exp}(\boldsymbol{w}) = \int \mathcal{L}(y, f(\boldsymbol{x}, \boldsymbol{w}))p(\boldsymbol{x}, y)d\boldsymbol{x}dy \tag{2.1}$$

is called expected risk of a function $f \in \mathcal{F}$.

In general, it cannot be calculated since a joint probability distribution $p(\boldsymbol{x}, y)$ is unknown. The *empirical risk* is an approximation of the expected risk and it can be calculated using only a given set of objects.

Definition 2 (**Empirical Risk**) Assume that a set of objects X is given.

$$\mathcal{R}_{emp}(\boldsymbol{w}) = \frac{1}{n}\sum_{k=1}^{n} \mathcal{L}(y, f(\boldsymbol{x}_k, \boldsymbol{w})) \tag{2.2}$$

is called empirical risk of a function $f \in \mathcal{F}$.

In (2.1), replacing $p(\boldsymbol{x}, y)$ by the empirical distribution

$$p(\boldsymbol{x}) = \frac{1}{n}\sum_{k=1}^{n} \delta(\boldsymbol{x} - \boldsymbol{x}_k) \ , \tag{2.3}$$

the empirical risk (2.2) can be obtained.

Clustering is to divide a set of objects into subsets consisting of similar objects. Each subset is referred to as a *cluster*. Various types of clustering methods exist but we focus on representative-based clustering methods.

Let c be the number of clusters, and let $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c) \in \mathbb{R}^p$ denote a set of representatives or *cluster centers*. In representative-based clustering methods,

$$\mathcal{L}(y_k, f(\boldsymbol{x}_k, V)) = D(\boldsymbol{x}_k, \boldsymbol{v}_{y_k}) \tag{2.4}$$

is a well-defined loss function where $D(\boldsymbol{x}_k, \boldsymbol{v}_i)$ is referred to as a *distortion measure*. It measures the error on the position of $\boldsymbol{x}$ when it is replaced by a representative v_y. In this chapter, the squared Euclidean distance

$$D(\boldsymbol{x}_k, \boldsymbol{v}_i) = \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 \tag{2.5}$$

is used as the distortion measure.

$$y_k = \arg \min_{1 \leq i \leq c} \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 \tag{2.6}$$

Letting

$$u_{ik} = \begin{cases} 1 & i = y_k \\ 0 & i \neq y_k \ , \end{cases} \tag{2.7}$$

we therefore obtain the empirical risk of clustering problems

$$J(V) = \frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} D(\boldsymbol{x}_k, \boldsymbol{v}_i) \ , \tag{2.8}$$

Our goal is to optimize the objective function (2.8) .

2.2 Competitive Learning

The simplest way to minimize J is the *stochastic gradient descent* method [8, 22].

$$-\Delta \boldsymbol{v}_i = -\alpha(t) \frac{\partial J}{\partial \boldsymbol{v}_i} = \alpha(t) \sum_{k=1}^{n} u_{ik} (\boldsymbol{x}_k - \boldsymbol{v}_i) \tag{2.9}$$

It is often referred to as the *batch competitive learning*. A learning rate $\alpha(t)$ at time t should be satisfied with

$$\sum_{t=1}^{\infty} \alpha(t) = \infty, \quad \sum_{t=1}^{\infty} \alpha^2(t) < \infty \ , \tag{2.10}$$

to guarantee the convergence [44]. For example, $\alpha(t) \propto 1/t$ satisfies the above condition.

Setting a data stream $\boldsymbol{x}^t, \ t = 1, \ldots, T$ and letting x_l be the current vector, the *on-line competitive learning* method is obtained by dropping the averaging operation in (2.9). The loss function $\mathcal{L}$ is not differentiable on points located the voronoi boundaries induced by cluster centers, but we can ignore the non-differentiable points [29, 8] in this case and apply the on-line gradient descent method.

$$-\Delta \boldsymbol{v}_i = u_{il} \alpha(t) (\boldsymbol{x}_l - \boldsymbol{v}_i) \tag{2.11}$$

In a batch algorithm, the number of iterations until convergence is much less than that of an on-line algorithm. But, the computational effort is greater than that of an on-line algorithm since a batch algorithm has an average operation. An on-line algorithm can handle additional data. However, this advantage causes non-robustness to noises.

2.3 Hard c-Means

K-means [32] is the most popular clustering algorithm. In the context of fuzzy clustering, K-means is referred to as the *hard c-means* [37]. The hard c-means (HCM) algorithm uses a more efficient optimization method than gradient descent methods [9].

Given a previous cluster centers $\bar{V}$, the objective function of the hard c-means is

$$J_{hcm}(V, \bar{V}) = \frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{c} \bar{u}_{ik} \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 \ , \tag{2.12}$$

In the HCM algorithm, u_{ik} is determined by a previous vector $\bar{V}$:

$$\bar{u}_{ik} = \begin{cases} 1 & i = \arg \min\limits_{1 \le i \le c} \|\boldsymbol{x}_k - \bar{\boldsymbol{v}}_i\|^2 \\ 0 & i \neq \arg \min\limits_{1 \le i \le c} \|\boldsymbol{x}_k - \bar{\boldsymbol{v}}_i\|^2 \ . \end{cases} \tag{2.13}$$

Since the objective function is linear with respect to u_{ik}, u_{ik} is driven to 0 or 1. The objective function J_{hcm} is therefore solved with respect to V, keeping $\bar{V}$ fixed.

Solving

$$\frac{\partial J}{\partial \boldsymbol{v}_i} = 0 \ , \tag{2.14}$$

we obtain

$$\boldsymbol{v}_i = \frac{1}{|G_i|} \sum_{\boldsymbol{x}_k \in G_i} \boldsymbol{x}_k \ , \tag{2.15}$$

where the i-th cluster denotes G_i.

Since u_{ik} is the best assignment, and induce the following property,

$$J(V) - J_{hcm}(V, \bar{V}) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 - \sum_{k=1}^{n} \sum_{i=1}^{c} \bar{u}_{ik} \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 \le 0 \ , \tag{2.16}$$

which help us understanding the convergence. From inequality (2.16),

$$J(V) - J(\bar{V}) = J(V) - J_{hcm}(V, \bar{V}) + J_{hcm}(V, \bar{V}) - J_{hcm}(\bar{V}, \bar{V}) \tag{2.17}$$

$$\le J_{hcm}(V, \bar{V}) - J_{hcm}(\bar{V}, \bar{V}) \le 0 \tag{2.18}$$

Each iteration of the algorithm thus decreases the original objective function J monotonically. The batch competitive learning is very similar to hard c-means if $\alpha(t)$ is fixed with $1/|G_i|$, however, these two algorithm is not equivalent. The optimization of hard c-means is a kind of *EM algorithms*, referred to in a later section.

2.4 Standard Fuzzy c-Means

The HCM algorithm and the competitive learning provide deterministic assignments to clusters. They thus work for well-separated clusters, but do not for overlapping clusters. There are two common-used concepts to handle overlapping clusters: the fuzzy set model and the mixture model.

Fuzzy c-means methods are an extension of hard c-means inspired by the fuzzy set model. The most famous fuzzy c-means method is *standard fuzzy c-means* [17, 5]. It is very simple way to fuzzify the HCM algorithm.

Let us introduce a relaxing constraint often referred to as the *probabilistic constraint*,

$$M_f = \{(u_{ik}) : u_{ik} \in [0,1], \sum_{i=1}^{c} u_{ik} = 1 \text{ for all } k\} \ . \tag{2.19}$$

As you see in the former section, we however cannot obtain fuzzy memberships since J_{hcm} is still linear with respect to u_{ik}. Hence, J_{hcm} should be non-linear. In the standard fuzzy c-means (sFCM) method, the modified objective function is used.

$$J_{sfcm} = \frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m D_{ik} \tag{2.20}$$

u_{ik} is derived using the Lagrange multipliers. The Lagrangian is written as

$$L = J_{sfcm} + \sum_{k=1}^{n} \mu_k (\sum_{i=1}^{c} u_{ik} - 1) \tag{2.21}$$

Solving L for u_{ik}, the following is obtained,

$$\frac{\partial L}{\partial u_{ik}} = m(u_{ik})^{m-1} D_{ik} + \mu_k = 0 \tag{2.22}$$

Vanishing the coefficient μ_k, we finally obtain

$$u_{ik} = \frac{1}{Z_k} D_{ik}^{-\frac{1}{m-1}} \tag{2.23}$$

where Z_k is a normalizer. In (2.23),

$$Z_k = \sum_{i=1}^{c} D_{ik}^{-\frac{1}{m-1}}. \tag{2.24}$$

On the other hand, there is no constraints for v_i, hence v_i takes a usual form,

$$\boldsymbol{v}_i = \frac{\sum_{k=1}^{n} (u_{ik})^m \boldsymbol{x}_k}{\sum_{k=1}^{n} (u_{ik})^m} \ . \tag{2.25}$$

2.5 Entropy Regularization

2.5.1 Cluster overlap measure

There are many criteria measuring degrees of cluster overlaps, but we choice the *Shannon's conditional entropy* [4, 45]:

$$H(Y|X) = -E_{XY}[\log p(y|\boldsymbol{x})] \tag{2.26}$$

$$= -\int \sum_{y} \log p(y|\boldsymbol{x}) p(\boldsymbol{x}, y) d\boldsymbol{x} \tag{2.27}$$

Since $p(\boldsymbol{x}, y) = p(\boldsymbol{x})p(y|\boldsymbol{x})$ (2.27) is written by

$$-\sum_{y} \int p(\boldsymbol{x}, y) \log \frac{p(\boldsymbol{x}, y)}{p(\boldsymbol{x})} d\boldsymbol{x} = -\sum_{y} KL\big(p(\boldsymbol{x}, y)||p(\boldsymbol{x})\big) \tag{2.28}$$

where

$$KL(p||q) = \int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x} \tag{2.29}$$

is the *Kullback-Leibler divergence*, which measures the difference between two probability distributions. The Shannon's conditional entropy can measure a sum of overlaps between y-th cluster and $p(\boldsymbol{x})$. $KL\big(p(\boldsymbol{x}, y)||p(\boldsymbol{x})\big)$ is decreasing as overlap is increasing.

The computation of $H(y|\boldsymbol{x})$ requires unknown $p(\boldsymbol{x}, y)$. Replacing $p(\boldsymbol{x})$ with the empirical distribution, we obtain

$$H(y|\boldsymbol{x}) = -\frac{1}{n} \sum_{k=1}^{n} \sum_{y} p(y|\boldsymbol{x}_k) \log p(y|\boldsymbol{x}_k) \tag{2.30}$$

2.5.2 Regularized Hard c-Means

We revisit the HCM algorithm. It cannot admit cluster overlap. We seek more smooth memberships than hard memberships obtained from the HCM algorithm.

The regularized hard c-means method using the Shannon's conditional entropy is proposed. It is often referred to as the *entropy-regularized fuzzy c-means* (eFCM) algorithm. The objective function is

$$J_{efcm}(V,U) = \frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}D_{ik} + \frac{\nu}{n}\sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}\log u_{ik} \ , \tag{2.31}$$

where $\nu > 0$ is the regularization parameter. It should be noted that the Shannon's conditional entropy is equivalent to the fuzzy entropy [42]. We therefore replace $p(y = i|x)$ by u_{ik}. The HCM algorithm provides a higher penalty on a point having a harder membership.

u_{ik} is also derived using the Lagrange multipliers, and we omit the derivation in detail. Finally, we obtain

$$u_{ik} = \frac{1}{Z_k}\exp\Big(-\frac{D_{ik}}{\nu}\Big) \ . \tag{2.32}$$

where Z_k is a normalizer. In (2.32),

$$Z_k = \sum_{i=1}^{c}\exp\Big(-\frac{D_{ik}}{\nu}\Big) \ . \tag{2.33}$$

On the other hand, v_i is the almost same with the HCM algorithm,

$$\boldsymbol{v}_i = \frac{\sum\limits_{k=1}^{n} u_{ik}\boldsymbol{x}_k}{\sum\limits_{k=1}^{n} u_{ik}}. \tag{2.34}$$

2.5.3 KL-divergence Regularization

Both of the HCM algorithm and the FCM algorithm follow the property that all clusters are the same size. It causes that small clusters eat away large clusters.

To obtain clusters having different sizes, Miyamoto et al. [41] and Ichihashi et al. [23] independently proposed an extension of the eFCM method. The entropy regularization term makes $p(\boldsymbol{x}, y)$ come close to $p(\boldsymbol{x})$. To avoid this issue, a new regularization term makes all memberships within a cluster come close to the uniform value, that is, makes $p(y|\boldsymbol{x})$ come close to $p(y)$:

$$\int KL\big(p(y|\boldsymbol{x})||p(y)\big)d\boldsymbol{x} = \frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}\log\frac{u_{ik}}{\alpha_i} \ , \tag{2.35}$$

where $p(y)$ is a prior of y-th cluster. Setting $p(y=i)=\alpha_i$ which is often referred to as the *cluster size variable*. If $\alpha_i = 1/c$, that is, all clusters are the same size, it is equivalent to the entropy regularization. The new objective function is therefore

$$J_{\alpha fcm}(V,U,\alpha) = \frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}D_{ik} + \frac{\nu}{n}\sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}\log\frac{u_{ik}}{\alpha_i} \ . \tag{2.36}$$

The cluster size variable α_i is obviously constrained with:

$$M_\alpha = \{\, \alpha = (\alpha_1,\ldots,\alpha_c) \,:\, \sum_{i=1}^{c}\alpha_i = 1,\ \alpha_j \geq 0,\ \forall j \,\}.$$

we obtain the same form (2.34) of v_i and

$$u_{ik} = \frac{1}{Z_k}\alpha_i \exp\Big(-\frac{D_{ik}}{\nu}\Big) \tag{2.37}$$

$$\alpha_i = \frac{1}{n}\sum_{k=1}^{n} u_{ik} \ . \tag{2.38}$$

It should be noted that the KL-divergence regularization is useful to build a desirable membership function, by setting functions $p(x), q(x)$ in $KL(p||q)$ for the objectives.

2.6 Possibilistic Clustering

2.6.1 Standard Method

Due to the probabilistic constraint (2.19), fuzzy c-means algorithms are not robust to outliers. Assume more relaxed constraint $u_{ik} \in \mathbb{R}^+$. Since it does not include influences from other clusters, it leads to a possibilistic clustering algorithm [30]. The objective function is

$$J_{pcm} = \frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m D_{ik} + \sum_{i=1}^{c}\frac{\eta_i}{n}\sum_{k=1}^{c}(1-u_{ik})^m \ . \tag{2.39}$$

The regularization term make u_{ik} come close to a membership value as big as possible. u_{ik} is given by

$$u_{ik} = \frac{1}{1+\left(\frac{D_{ik}}{\eta_i}\right)^{\frac{1}{m-1}}} \ , \tag{2.40}$$

On the other hand, v_i remains the same:

$$\boldsymbol{v}_i = \frac{\sum_{k=1}^{n} (u_{ik})^m x_k}{\sum_{k=1}^{n} (u_{ik})^m} \ . \tag{2.41}$$

2.6.2 I-divergence Regularization

Let the eFCM algorithm extend to the possibilistic version. The Shannon's conditional entropy term is rewritten by *I-divergence* [12]:

$$I(p||q) = \int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x} + \Big\{ \int p(\boldsymbol{x}) d\boldsymbol{x} - \int q(\boldsymbol{x}) d\boldsymbol{x} \Big\} \tag{2.42}$$

where $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ no longer need to be probability distributions but can be any non-negative functions. If the probabilistic constraint is used, the second term is vanished and it is equals to $KL(p||q)$. A new regularizer is defined as

$$\Omega_I(U) = \frac{1}{n} \sum_{k=1}^{n} \Big\{ \sum_{i=1}^{c} u_{ik} \log u_{ik} - (\sum_{i=1}^{c} u_{ik} - 1) \Big\} \ . \tag{2.43}$$

The objective function therefore is rewritten by

$$J_{epcm} = \frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} D_{ik} + \frac{\nu}{n} \sum_{k=1}^{n} \Big\{ \sum_{i=1}^{c} u_{ik} \log u_{ik} - (\sum_{i=1}^{c} u_{ik} - 1) \Big\} \ . \tag{2.44}$$

u_{ik} is given by

$$u_{ik} = \exp\Big(- \frac{D_{ik}}{\nu} \Big), \tag{2.45}$$

On the other hand, $\boldsymbol{v}_i$ remains the same:

$$\boldsymbol{v}_i = \frac{\sum_{k=1}^{n} u_{ik} \boldsymbol{x}_k}{\sum_{k=1}^{n} u_{ik}} \tag{2.46}$$

2.7 EM Algorithm

2.7.1 Latent Variable Model

we have reviewed discriminative clustering method in which any generative distribution is not required. On the other hand, clustering methods using generative models have been frequently used.

In the generative model, all variables are classified into observed variables and unobserved variables so-called *latent variable*. For example, in the clustering setting, we cannot observe clusters from which data are generated.

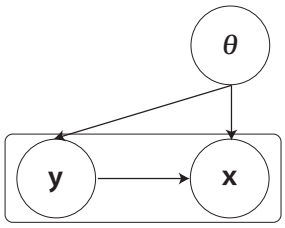

Fig. 2.1 An observed variable $\boldsymbol{x}$ is generated from a hidden variable y. Both of $\boldsymbol{x}$ and y depend on a parameter variable θ .

In general, it is very complicated to estimate parameters in latent variable models. *Expectation maximization (EM)* algorithm [13, 43] gives us an elegant way to solve this problem. The complete data log-likelihood is

$$\log p(X|\theta) = \log \sum_y p(y, X|\theta) \tag{2.47}$$

The problem to obtain parameters maximizing (2.47) is non-linear and hence direct solutions are uses of the Newton's method and other non-linear optimization techniques. However, solutions are not given directly since y is not observed.

The EM algorithm maximizes an approximation of the complete data log-likelihood. The Jensen's inequality for convex (upward) function

$$f\big(E[x]\big) \geq E\big[f(x)\big]$$

leads to the lower bound of (2.47),

$$\log \sum_y p(y, X|\theta) = \log \sum_y p(y|X, \theta^t) \frac{p(y, X|\theta)}{p(y|X, \theta^t)} \tag{2.48}$$

$$= \log \Big\langle \frac{p(y, X|\theta)}{p(y|X, \theta^t)} \Big\rangle_{p(y|X,\theta^t)} \tag{2.49}$$

$$\geq \Big\langle \log \frac{p(y, X|\theta)}{p(y|X, \theta^t)} \Big\rangle_{p(y|X,\theta^t)} = \sum_y p(y|X, \theta^t) \log \frac{p(y, X|\theta)}{p(y|X, \theta^t)} \tag{2.50}$$

The EM algorithm repeats two steps until the convergence to maximize the lower bound of the complete data log-likelihood.

[E step]

$$\mathcal{F}(\theta, \theta') = \sum_y p(y|X, \theta^t) \log \frac{p(y, X|\theta)}{p(y|X, \theta^t)} \tag{2.51}$$

$$= \sum_y p(y|X, \theta^t) \log \frac{p(y|X, \theta)p(X|\theta)}{p(y|X, \theta^t)} \tag{2.52}$$

$$= -\sum_y p(y|X, \theta^t) \log \frac{p(y|X, \theta^t)}{p(y|X, \theta)} + \log p(X|\theta) \tag{2.53}$$

$$= -KL\big(p(y|X, \theta^t)||p(y|X, \theta)\big) + \log p(X|\theta) \tag{2.54}$$

The E step maximizes the KL-divergence between $p(y|X, \theta^t)$ and $p(y|X, \theta)$. We compute a posterior $p(y|X, \theta)$ by the Bayes formula.

$$p(y|X, \theta) = \frac{p(y, X|\theta)}{\sum_y p(y, X|\theta)} \tag{2.55}$$

[M step]

$$\mathcal{F}(\theta, \theta') = \sum_y p(y|X, \theta^t) \log \frac{p(y, X|\theta)}{p(y|X, \theta^t)} \tag{2.56}$$

$$= \langle \log p(y, X|\theta) \rangle_{p(y|X,\theta^t)} + H\big(p(y|X, \theta^t)\big) \tag{2.57}$$

To maximize $\mathcal{F}(\theta, \theta')$ with respect to θ, $H\big(p(y|X, \theta^t)\big)$ is not including θ and then ignored, the objective function therefore is

$$Q(\theta, \theta') = \langle \log p(y, X|\theta) \rangle_{p(y|X,\theta^t)} \tag{2.58}$$

The solution derived from

$$\frac{\partial Q(\theta, \theta')}{\partial \theta} = 0 \tag{2.59}$$

2.7.2 The connection between EM and Fuzzy c-Means

The mixture model is very popular for the setting of clustering [34]. As you see in Fig 2.2, y indicating each cluster only depends on π and it is independent from θ contributing to generate x. Since the mixture model is a latent variable model, it can be applied to the EM algorithm.

$$p(X|\theta) = \sum_y p(y)p(X|y, \theta) = \sum_y \pi_y p(X|y, \theta) \tag{2.60}$$

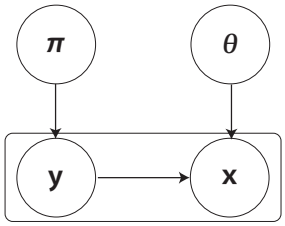

Fig. 2.2 The graphical plate model of the mixture model. The mixture coefficient is introduced.

In practice, $p(X|y,\theta)$ is required. Let $p(X|y,\theta)$ be the regular exponential family [3],

$$p(\boldsymbol{x}|y,\theta) = \frac{1}{Z_\theta}\exp\Big(-D_A(\boldsymbol{x},\boldsymbol{\mu}_y)\Big) \tag{2.61}$$

where $D_A(\boldsymbol{x},\boldsymbol{\mu}_y)$ is an adaptive distortion measure between $\boldsymbol{x}$ and the mean of y-th cluster μ_y. The adaptive parameter A is given by a matrix such as the covariance matrix. Letting A be the unit matrix I,

$$\mathcal{F}(\theta,\theta') = \sum_y p(y|X,\theta^t)\log\frac{p(y,X|\theta)}{p(y|X,\theta^t)} \tag{2.62}$$

$$= \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}D_{ik} + \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}\frac{u_{ik}}{\alpha_i} \tag{2.63}$$

It is obviously equivalent to the eFCM algorithm. [1, 23].

Chapter 3

Basic Concept of Kernels

This chapter provides the basic concept of kernels. After an intuitive example, a Mercer kernel is introduced. The reproducing kernel Hilbert space (RKHS) is important to understand the representer theorem. Next, the most popular kernel-based clustering algorithm, kernel hard c-means method is described using an expression in the RKHS.

3.1 Mercer Kernel

3.1.1 Kernel Trick

Let us consider a linear classifier,

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b, \tag{3.1}$$

where $\boldsymbol{x} \in \mathbb{R}^p$ is an p-dimensional vector, $\boldsymbol{w} \in \mathbb{R}^p$ is a weight vector, and $b \in \mathbb{R}$ is a bias variable.

The general linear classifier can classify linearly separable data using a $p-1$ hyperplane. However, if data cannot be separated linearly, it is a significant issue (but not be fatal) .

A non-linear classifier such as multilayer perceptron is effective to this issue. But it has a difficulty of learning such as local minima.

Suppose that data are mapped into the high-dimensional feature space.

$$\Phi : \mathbb{R}^p \to \mathbb{R}^q, \quad \boldsymbol{x} \to \Phi(\boldsymbol{x}) \tag{3.2}$$

The mapping Φ is a nonlinear mapping into a q-dimensional space, and q is much greater than p. A mapped object $\Phi(\boldsymbol{x})$ is called a *feature vector*.

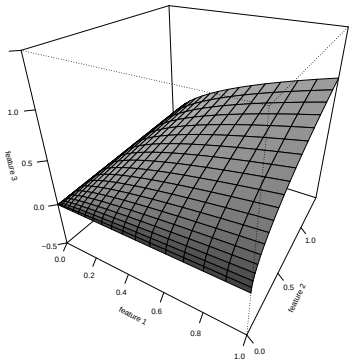

Fig. 3.1 A nonlinear mapping of the unit square $[0,1]^2 \subset \mathbb{R}^2$ into $\mathbb{R}^3$ by equation (3.3). The mapped unit square forms a two-dimensional sub-manifold in $\mathbb{R}^3$.

For example, two-dimensional data are mapped into three-dimensional data,

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3, \quad \Phi(\boldsymbol{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2). \tag{3.3}$$

As Fig.3.1, two-dimensional data are not linearly separable, but mapped data are linearly separable.

However, in practice, a feature vector is very higher-dimensional. Due to the limit of computational resources, it is impossible to use direct computations of feature vectors.

There is an elegant technique to avoid the above issue. Assume that a weight vector $\boldsymbol{w}$ is a linear combination of a given set of objects,

$$\boldsymbol{w} = \sum_{k=1}^{n} \alpha_k \boldsymbol{x}_k, \tag{3.4}$$

where $\alpha_k \in \mathbb{R}$ is a coefficient. A linear classifier is rewritten as below,

$$f(\boldsymbol{x}) = \sum_{k=1}^{n} \alpha_k \langle \boldsymbol{x}, \boldsymbol{x}_k \rangle. \tag{3.5}$$

Hence, we obtain a linear classifier in the feature space,

$$f(\Phi(\boldsymbol{x})) = \sum_{k=1}^{n} \alpha_k \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_k) \rangle. \tag{3.6}$$

Let a kernel function

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle \tag{3.7}$$

be given. Hence, a linear classifier in the feature space can be expressed with only the kernel.

$$f(\boldsymbol{x}) = \sum_{k=1}^{n} \alpha_k k(\boldsymbol{x}, \boldsymbol{x}_k) \tag{3.8}$$

We can operate the computation in the feature space using the function in the original space. This technique is referred to as *kernel trick*.

Definition 3 (Kernel) The kernel is the inner product function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$,

$$k(x, x') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle. \tag{3.9}$$

A kernel is a function in the general input space $\mathcal{X}$.

3.1.2 Mercer Kernel

Not any symmetric function k can serve as a kernel. The necessary and sufficient condition of k to be a kernel is given by Mercer's theorem.

Theorem 4 (Mercer's Theorem) Suppose $k \in L_\infty(\mathcal{X} \times \mathcal{X})$ is a symmetric function, i.e., $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$, such that the integral operator $T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$ given by

$$(T_k f)(\cdot) = \int_{\mathcal{X}} k(\cdot, \boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x}$$

is positive semidefinite, that is,

$$\int_{\mathcal{X}} \int_{\mathcal{X}} k(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}) f(\boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x}' \geq 0 \tag{3.10}$$

for all $f \in L_2(\mathcal{X})$. Let $\psi_i \in L_2(\mathcal{X})$ be the eigenfunction of T_k associated with the eigenvalue $\lambda_i \geq 0$ and normalized such that $\|\psi_i\|^2 = \int_{\mathcal{X}} \psi_i^2(\boldsymbol{x}) d\boldsymbol{x} = 1$, i.e.,

$$\forall \boldsymbol{x} \in \mathcal{X} : \quad \int_{\mathcal{X}} k(\boldsymbol{x}, \boldsymbol{x}') \psi_i(\boldsymbol{x}') d\boldsymbol{x}' = \lambda_i \psi_i(\boldsymbol{x}).$$

Then, $(\lambda_i)_{i \in \mathbb{N}} \in l_1$, $\psi_i \in L_\infty(\mathcal{X})$, and k can be expanded in a uniformly convergent series, i.e.,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{\infty} \lambda_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{x}')$$

holds for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$.

For example, *Gaussian kernel* is the most popular mercer kernel,

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \exp\Big(- cnst\|\boldsymbol{x} - \boldsymbol{x}'\|^2\Big). \tag{3.11}$$

3.2 Reproducing Kernel Hilbert Space

Assume that k is a real-valued positive definite kernel, and $\mathcal{X}$ a nonempty set. We define a map from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ into $\mathbb{R}$, denoted as $\mathbb{R}^{\mathcal{X}} := \{f : \mathcal{X} \to \mathbb{R}\}$, via

$$\Phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}} \tag{3.12}$$

$$\boldsymbol{x} \mapsto k(\cdot, \boldsymbol{x}) \tag{3.13}$$

We first define a vector space. This is done by taking linear combinations of the form

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, \boldsymbol{x}_i) \tag{3.14}$$

Here, $m \in \mathbb{N}$, $\alpha_i \in \mathbb{R}$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \mathcal{X}$ are arbitrary. Next, we define a dot product between another function

$$g(\cdot) = \sum_{j=1}^{m'} \beta_j k(\cdot, \boldsymbol{x}'_j) \tag{3.15}$$

where $m' \in \mathbb{N}$, $\beta_j \in \mathbb{R}$ and $\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_{m'} \in \mathcal{X}$, as

$$\langle f, g \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_k \beta_j k(\boldsymbol{x}_i, \boldsymbol{x}'_j) \tag{3.16}$$

This expression explicitly contains the expansion coefficients, which need not be unique. To see that it is nevertheless well-defined, note that

$$\langle f, g \rangle = \sum_{j=1}^{n'} \beta_j f(\boldsymbol{x}'_j) \tag{3.17}$$

using $k(\boldsymbol{x}'_j, \boldsymbol{x}_i) = k(\boldsymbol{x}_i, \boldsymbol{x}'_j)$. However, the sum does not depend on the particular expansion of f. Similarly, for g, note that

$$\langle f, g \rangle = \sum_{i=1}^{n} \beta_i f(\boldsymbol{x}_i) \tag{3.18}$$

The last two equations also show that $\langle \cdot, \cdot \rangle$ is bilinear. It is symmetric, as $\langle f, g \rangle = \langle g, f \rangle$. Moreover, it is positive definite, since positive definiteness of k implies that for any function f, written as (3.14), we have

$$\langle f, f \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0 \tag{3.19}$$

The latter implies that $\langle \cdot, \cdot \rangle$ is actually itself a positive definite kernel, defined on our space of functions. To see this, note that given functions $f_1, \ldots, f_n$, and coefficients $\gamma_1, \ldots, \gamma_n \in \mathbb{R}$, we have

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \gamma_i \gamma_j \langle f_i, f_j \rangle = \Big\langle \sum_{i=1}^{n} \gamma_i f_i, \sum_{j=1}^{n} \gamma_j f_j \Big\rangle \geq 0 \tag{3.20}$$

Here, the left hand equality follows from the bi-linearity of $\langle \cdot, \cdot \rangle$ and the right hand inequality from (3.20). For the last step in proving that it qualifies as a dot product, we will use the following interesting property of Φ, which follows directly from the definition: for all functions, we have

$$\langle k(\cdot, \boldsymbol{x}), f \rangle = f(\boldsymbol{x}) \tag{3.21}$$

k is the representer of evaluation. In particular,

$$\langle k(\cdot, \boldsymbol{x}), k(\cdot, \boldsymbol{x}') \rangle = k(\boldsymbol{x}, \boldsymbol{x}') \tag{3.22}$$

By virtue of these properties, positive definite kernel k are also called *reproducing kernels*. By (3.21) and Proposition 2.7, we have

$$|f(x)|^2 = |\langle k(\cdot, \boldsymbol{x}), f \rangle|^2 \leq k(\boldsymbol{x}, \boldsymbol{x}) \cdot \langle f, f \rangle. \tag{3.23}$$

which shows that $\langle f, f \rangle = 0$ only if $f(x) = 0$ for all $x \in \mathcal{X}$, i.e., $f = 0$.

Theorem 5 (Representer Theorem) Let k be a Mercer kernel on $\mathcal{X}$. Define $\mathcal{F}$ as the RKHS induced by k. Then any $f \in \mathcal{F}$ minimizing the regularized risk

$$R_{reg}(f, \boldsymbol{x}) = g_{emp}(\boldsymbol{x}, f) + g_{reg}(\|f\|), \tag{3.24}$$

admits a representation of the form

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, \boldsymbol{x}_i), \quad \boldsymbol{\alpha} \in \mathbb{R}^m \tag{3.25}$$

3.3 Algorithm of Kernel Hard c-Means

Assume the dissimilarity in RKHS,

$$D_{ik} = \|k(\cdot, \boldsymbol{x}_k) - f_i(\cdot)\|_H^2 \tag{3.26}$$

The cluster center in feature space is expressed

$$f_i(\cdot) = \frac{1}{|G_i|} \sum_{y_k = i} k(\cdot, \boldsymbol{x}_k) \tag{3.27}$$

This form satisfies the representer theorem. In practice, we compute the expansion of dissimilarity,

$$D_{ik} = \langle k(\cdot, \boldsymbol{x}_k), k(\cdot, \boldsymbol{x}_k) \rangle - 2\langle k(\cdot, \boldsymbol{x}_k), f_i \rangle + \langle f_i, f_i \rangle \tag{3.28}$$

From the property, we obtain

$$D_{ik} = k(\boldsymbol{x}_k, \boldsymbol{x}_k) - \frac{2}{|G_i|} \sum_{\boldsymbol{x}_j \in G_i} k(\boldsymbol{x}_k, x_j) + \frac{1}{|G_i|^2} \sum_{\boldsymbol{x}_j \in G_i} \sum_{\boldsymbol{x}_l \in G_i} k(\boldsymbol{x}_j, \boldsymbol{x}_l) \tag{3.29}$$

Kernel hard c-means works well except for a local minima. Since kernel hard c-means satisfies the representer theorem, the basic hard c-means problem is equivalent to

$$\min_v \sum_{i=1}^{c} \sum_{\boldsymbol{x}_k \in G_i} \Big(- \langle \boldsymbol{x}_k, \boldsymbol{v}_i \rangle + \frac{1}{2} \|\boldsymbol{v}_i\|^2 \Big) \tag{3.30}$$

$$= \min_v - \sum_{i=1}^{c} \sum_{\boldsymbol{x}_k \in G_i} \langle \boldsymbol{x}_k, \boldsymbol{v}_i \rangle + \frac{1}{2} \sum_{i=1}^{c} |G_i| \cdot \|\boldsymbol{v}_i\|^2 \tag{3.31}$$

If $|G_i|$ is corresponding to λ_i, it is a regularized risk functional. Mapping into RKHS, the interpretation of hard c-means is,

$$- \sum_{i=1}^{c} \sum_{\boldsymbol{x}_k \in G_i} f(\boldsymbol{x}_k) + \frac{1}{2} \sum_{i=1}^{c} |G_i| \cdot \|f_i\|^2. \tag{3.32}$$

Hence, we understand that the cluster center (3.27) takes the form (3.25).

This algorithm has $O(n^3)$ computation, and hence to reduce the complexity of a kernel algorithm should be studied, which we consider in the next chapter.

Chapter 4

Efficient Algorithms of Kernel-based Clustering

In this chapter, we propose a new kernel method for competitive learning clustering, and its variations and extensions. It is shown that the order of computation is reduced when compared with the existing method of kernel fuzzy c-means.

4.1 Kernel-based Competitive Learning

In contrast to batch learning, on-line learning has a lower computational effort. We hence propose a method of kernel competitive learning clustering based on this idea.

Assume that a dissimilarity between an object and a cluster center in the feature space,

$$D_{ik} = \|\Phi(\boldsymbol{x}_k) - \boldsymbol{m}_i\|_{\mathcal{H}}^2, \tag{4.1}$$

where $\boldsymbol{x}_k \in \mathcal{X}$ and $\boldsymbol{m}_i \in \mathbb{R}^{\mathcal{X}}$.

The updating rule of on-line competitive learning clustering in the feature space is

$$\boldsymbol{m}_i^t = \boldsymbol{m}_i^t + \delta_{ih}\alpha(t)(\Phi(\boldsymbol{x}_l) - \boldsymbol{m}_i^t). \tag{4.2}$$

Note that $\boldsymbol{m}_i$ is a cluster center in the feature space, and $\boldsymbol{x}_l$ is an input object at time t.

Since we do not know a concrete form of $\Phi(\boldsymbol{x})$, we cannot explicitly use $\boldsymbol{m}_i$. Hence the algorithm should be rewritten using (4.1) and the kernel function instead. We

therefore calculate D_h^{t+1} using D_{hk}^t and D_{hl}^t. For simplicity, put $\alpha = \alpha(t)$, $K_{kk} = k(\boldsymbol{x}_k, \boldsymbol{x}_k)$, and $K_{kl} = k(\boldsymbol{x}_k, \boldsymbol{x}_l)$. Then,

$$\begin{aligned} & D_{hk}^{t+1} - (1-\alpha)D_{hk}^t + \alpha(1-\alpha)D_{hl}^t \\ & \quad = K_{kk} + \alpha^2 K_{ll} - (1-\alpha)K_{kk} + \alpha(1-\alpha)K_{ll} - 2\alpha K_{kl} \\ & \qquad + \{(1-\alpha)^2 - (1-\alpha) + \alpha(1-\alpha)\}\|\boldsymbol{m}_h^t\|^2 \\ & \quad = \alpha(K_{kk} - 2K_{kl} + K_{ll}). \end{aligned}$$

We thus have

$$D_{hk}^{t+1} = (1-\alpha)D_{hk}^t - \alpha(1-\alpha)D_{hl}^t + \alpha(K_{kk} - 2K_{kl} + K_{ll}). \tag{4.3}$$

When we compare (3.29) and (4.3), we note that (3.29) has double summations in the last term which require $O(n^2)$ calculation. In contrast, equation (4.3) does not have such a summation. Since (3.29) and (4.3) are repeated n times for one cycle of the major loop of the iteration, the latter formulation has a lower computational complexity $O(n)$ when compared with $O(n^3)$ in kernel hard c-means.

4.1.1 On-line competitive learning in RKHS

There is another interpretation of kernel on-line competitive learning from RKHS. Recall the dissimilarity in RKHS,

$$D_{ik} = \|k(\cdot, \boldsymbol{x}_k) - f_i\|_{\mathcal{H}}^2 \tag{4.4}$$

Stochastic gradient descents in RKHS are shown below.

$$\Delta f_i = -\delta_{ih}\alpha(t)(k(\cdot, \boldsymbol{x}_k) - f_i) \tag{4.5}$$

For the purpose of practical computations, one can write f_i as a kernel expansion

$$f_i(\boldsymbol{x}) = \sum_{k=1}^{n} \beta_{ik} k(\boldsymbol{x}_k, \boldsymbol{x}) \tag{4.6}$$

where the coefficients are updated via

$$\beta_{hl}(t+1) = (1-\alpha)\beta_{hl}(t) + \alpha, \tag{4.7}$$

$$\beta_{hk}(t+1) = (1-\alpha)\beta_{hk}(t), \ k \neq l, \tag{4.8}$$

Thus we assume that a reference vector is represented as a linear combination of all data in the high-dimensional feature space.

$$f_i = \sum_{k=1}^{n} \beta_{ik} k(\cdot, \boldsymbol{x}_k), \tag{4.9}$$

After the learning process, we can compute distances between a new unknown data point and reference vectors using the last coefficients, and then it can be assigned to a class by the nearest prototype rule: x should be in class l if and only if $l = \arg\min_{1 \le i \le c} d_i(x)$, where

$$d_i(x) = k(\boldsymbol{x}, \boldsymbol{x}) - 2\sum_{j=1}^{n} \beta_{ij} k(\boldsymbol{x}_j, \boldsymbol{x}) + \sum_{j=1}^{n}\sum_{l=1}^{n} \beta_{ij}\beta_{il} K_{jl}, \tag{4.10}$$

Notice that

$$d_i(\boldsymbol{x}) = \|k(\cdot, \boldsymbol{x}) - f_i\|_{\mathcal{H}}^2.$$

4.2 Sequential Algorithm of Kernel Hard c-Means

4.2.1 Sequential Hard c-Means

A variation of the hard c-means algorithm has been considered by [16]. Their procedure is sometimes called a sequential algorithm, since it updates cluster centers immediately after an object moves from a cluster to another cluster.

Suppose that a center of cluster G_r is the nearest cluster centers to an object $\boldsymbol{x}_k \in \mathbb{R}^p$ belonging to cluster G_q. x_k moves from cluster G_q to cluster G_r, then the updating equations of $\boldsymbol{v}_q$, $\boldsymbol{v}_r \in \mathbb{R}^p$ are as follows:

$$\boldsymbol{v}_q' = \frac{N_q}{N_q - 1}\boldsymbol{v}_q - \frac{\boldsymbol{x}_k}{N_q - 1} \tag{4.11}$$

$$\boldsymbol{v}_r' = \frac{N_r}{N_r + 1}\boldsymbol{v}_r + \frac{\boldsymbol{x}_k}{N_r + 1} \tag{4.12}$$

where N_q and N_r are the numbers of elements in clusters G_q and G_r, respectively.

Next, we apply the present idea of on-line learning to the sequential algorithm of kernel hard c-means [39]. Data space is not limited in the Euclidean space, that is, $\boldsymbol{x}_k \in \mathcal{X}$. The updating equations of $\boldsymbol{m}_r, \boldsymbol{m}_q \in \mathbb{R}^{\mathcal{X}}$ in the feature space are

$$\boldsymbol{m}_q' = \boldsymbol{m}_q - \frac{1}{N_q - 1}(\Phi(\boldsymbol{x}_k) - \boldsymbol{m}_q) \tag{4.13}$$

$$\boldsymbol{m}_r' = \boldsymbol{m}_r + \frac{1}{N_r + 1}(\Phi(\boldsymbol{x}_k) - \boldsymbol{m}_r) \tag{4.14}$$

The algorithm of kernel sequential hard c-means is hence derived from (4.3).

$$D'_{qk} = \frac{N_q}{N_q - 1} D_{qk} + \frac{N_q}{(N_q - 1)^2} D_{ql} - \frac{1}{N_q - 1}(K_{kk} - 2K_{kl} + K_{ll}), \tag{4.15}$$

$$D'_{rk} = \frac{N_r}{N_r + 1} D_{rk} - \frac{N_r}{(N_r + 1)^2} D_{rl} + \frac{1}{N_r + 1}(K_{kk} - 2K_{kl} + K_{ll}). \tag{4.16}$$

4.2.2 Sequential Fuzzy c-Means

In this section, fuzzy c-means is considered as a generalization of hard c-means. Notice the constraint for membership,

$$M_f = \{(u_{ik}) : u_{ik} \in [0, 1], \sum_{i=1}^{c} u_{ik} = 1 \text{ for all } k\}. \tag{4.17}$$

The movement of cluster centers is replaced by the change of membership,

$$u'_{ik} = u_{ik} + \Delta u_{ik}. \tag{4.18}$$

Hence we obtain the following:

$$(U_i + \Delta u_{ik})\boldsymbol{v}'_i = U_i v_i + \Delta u_{ik} \boldsymbol{x}_k, \tag{4.19}$$

$$U_i = \sum_{k=1}^{n} u_{ik}. \tag{4.20}$$

The updating formula of cluster centers are

$$\boldsymbol{v}'_i = \boldsymbol{v}_i + \frac{\Delta u_{ik}}{U_i + \Delta u_{ik}}(\boldsymbol{x}_k - \boldsymbol{v}_i). \tag{4.21}$$

If the constraint of memberships are restricted as $U_i = N_i$, $\Delta u_{qk} = -1, \Delta u_{rk} = 1$. It then is equivalent to sequential hard c-means.

In a fuzzy version of sequential hard c-means, $\boldsymbol{v}_i$ are given by

$$\boldsymbol{v}_i = \frac{\sum_{k=1}^{n} u_{ik} \boldsymbol{x}_k}{\sum_{k=1}^{n} u_{ik}}. \tag{4.22}$$

There are variations for calculating membership u_{ik}: popular membership functions are

$$u_{ik} = \frac{1}{Z_k} D_{ik}^{-\frac{1}{m-1}}, \tag{4.23}$$

$$u_{ik} = \frac{1}{Z_k} \exp\big(-\lambda D_{ik}\big). \tag{4.24}$$

where m and λ are positive parameters. The first is used in the standard fuzzifier, while the second is used in the entropy regularized hard c-means. In the case of (4.23), U_i, Δu_{ik} are given by:

$$U_i = \sum_{k=1}^{n} (u_{ik})^m \tag{4.25}$$

$$\Delta u_{ik} = (u'_{ik})^m - (u_{ik})^m \tag{4.26}$$

Obviously sequential fuzzy c-means are employing the proposed method. We define a function g with respect to Δu_{ik} as follows:

$$g(\Delta u_{ik}) = \frac{\Delta u_{ik}}{U_i + \Delta u_{ik}} \tag{4.27}$$

The algorithm of kernel sequential fuzzy c-means is derived from (4.3).

$$\begin{aligned} D'_{ik} = (1 - g(\Delta u_{ik}))D_{ik} - g(\Delta u_{il})(1 - g(\Delta u_{hl}))D_{il} \\ + g(\Delta u_{ik})(K_{kk} - 2K_{kl} + K_{ll}) \end{aligned} \tag{4.28}$$

4.3 Illustrative Examples

4.3.1 Examples of Kernel On-line Competitive Learning

Two kinds of artificially generated 150 points on a plane were analyzed. The results by kernel on-line competitive learning clustering (**K-LVQC**) are shown in Fig. 4.1 where two clusters are represented by $\times$, and $+$, and Fig. 4.2 where three clusters are represented by $*$, $\times$, and $+$.

Since clustering is considered, the original data were not divided into the two or three classes. It is moreover obvious that the methods of ordinary hard c-means and fuzzy c-means cannot separate the outer circle and the inner circle and the ball within the outer circle. Thus these figures show the effectiveness of the kernel-based method for having nonlinear boundary between clusters.

The same output is obtained by the sequential algorithms of hard c-means and fuzzy c-means proposed here, although we omit the details.

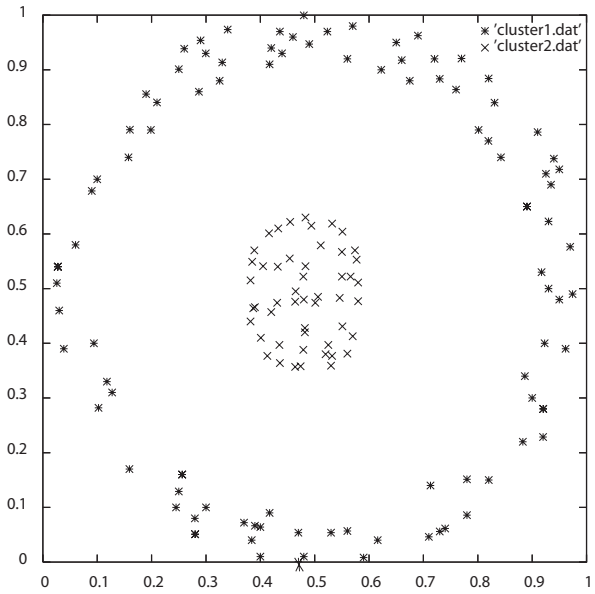

Fig. 4.1 Result of **KLVQC**. Data of 'a ring around a ball'. Parameters are $(c = 2, \alpha = 0.4, cnst = 20)$. The algorithm **KLVQC** can obtain clusters having non-linear cluster boundaries.

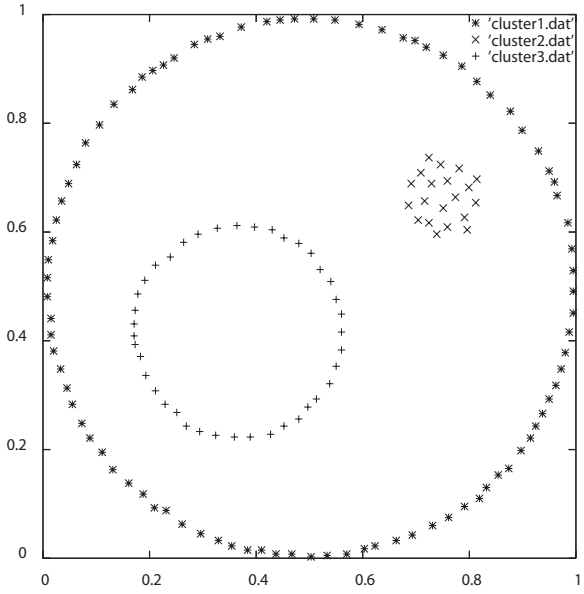

Fig. 4.2 Result of **KLVQC**. Data of 'a ball and a ring inside a large ring'. Parameters are $(c = 3, \alpha = 0.4, cnst = 20)$. The algorithm **KLVQC** can obtain clusters having more complex cluster boundaries.

Table. 4.1 Number of classification errors in Iris data by different methods

algorithm	*no. of errors*	*algorithm*	*no. of errors*
K-LVQC	14.98	LVQC	16.00
K-sFCM	17.00	sFCM	16.00
K-eFCM	13.90	eFCM	17.02

4.4 Numerical Examples Using Real Data

Two kinds of well-known data sets[*1], i.e., *Iris data* and *Wisconsin breast cancer data* were used to compare different algorithms of clustering including the present kernel competitive learning clustering.

Iris data set has 4 dimensional vectors of 150 objects in which 3 different types of Iris are included. Wisconsin breast cancer data set has 9 dimensional vectors of 683 objects in which two classes of *benign* and *malignant* are included. In addition to on-line competitive learning (**LVQC**) and kernel on-line competitive learning (**K-LVQC**) considered here, four other methods of fuzzy c-means, i.e., the standard fuzzifier (**sFCM**) and entropy-regularized hard c-means (**eFCM**) without a kernel function, the standard fuzzifier with the kernel (**K-sFCM**), and the entropy-regularized hard c-means with the kernel (**K-eFCM**) were tested.

Numbers of misclassified objects by these six methods are shown in Tables 4.1 and 4.2. Numbers of misclassified objects are averages of 100 trials with different initial values in each method. It is seen that the present method is as good as other methods and the use of the kernel is effective in the real data sets.

Tables 4.3 and 4.4 show the processor time until convergence by the four methods of **K-LVQC**, **K-eFCM**, **LVQC**, and **eFCM**. The processor time is the average of 100 trials with different initial values in each method. Those of **K-sFCM** and **sFCM** are longer than those of **K-eFCM** and **eFCM**, respectively.

Apparently, **K-eFCM** is time consuming while **K-LVQC** is much more efficient, while the classification capability is not worse than **K-eFCM** and **eFCM**.

[*1] The data sets have been downloaded from UCI repository of machine learning databases and domain theories, FTP address:ftp://ftp.ics.uci.edu/pub/machine-learning-databases

Table. 4.2 Number of classification errors in BCW data by different methods

algorithm	*no. of errors*	*algorithm*	*no. of errors*
K-LVQC	20.00	LVQC	27.00
K-sFCM	21.00	sFCM	28.00
K-eFCM	23.00	eFCM	27.00

Table. 4.3 Processor time for clustering of Iris data by the four different methods

algorithm	*processor time(sec)*	*algorithm*	*processor time(sec)*
K-LVQC	2.05×10^{-2}	LVQC	3.90×10^{-3}
K-sFCM	103.39	sFCM	1.71×10^{-2}
K-eFCM	4.25	eFCM	8.70×10^{-3}

Table. 4.4 Processor time for clustering of BCW data by the four different methods

algorithm	*processor time(sec)*	*algorithm*	*processor time(sec)*
K-LVQC	1.39×10^{-1}	LVQC	3.30×10^{-3}
K-sFCM	1827.94	sFCM	9.10×10^{-2}
K-eFCM	126.54	eFCM	1.45×10^{-2}

4.5 Discussion

In this chapter, we have proposed new methods of kernel-based clustering derived from the on-line competitive learning. It has lower computational complexity than kernel hard c-means and other batch clustering algorithms.

We have observed that kernel on-line competitive learning can obtain nonlinear classification boundaries.

We moreover have considered kernel methods for a sequential algorithm of hard c-means. Another algorithm of sequential hard c-means can be developed for fuzzy

version.

Various data sets should be tested to examine effectiveness of nonlinear separation of clusters.

Chapter 5

Efficient Algorithm with Sparseness

In this chapter, we propose a sparse possibilistic clustering with l_1 regularization to reduce membership parameters. The exact value regularization cannot be applied directly, since memberships are always non-negative. To solve this difficulty, we introduce the baseline constant into the regularizer. Illustrative examples are shown to evaluate the effectiveness of the proposed method.

5.1 Possibilistic Clustering and Sparseness

The goal of data mining is to discover knowledge from large amounts of data. In that sense, clustering techniques have been frequently used to detect meaningful subgroups.

Fuzzy c-means is one of the most popular clustering methods. In this algorithm, a generative model is not required, and cluster centers and posterior probabilities of clusters (memberships) are obtained via optimization of the objective functions. The constraints are probabilistic in the sense that the sum of memberships over all clusters is unity.

Fuzzy c-means is a soft partitioning method, rather than a cluster detection. It is not robust to noise, and irrelevant data must be included in any clusters, since all individuals are assumed to belong to at least one cluster. Hence, cluster mining

algorithms are required to detect important clusters or meaningful subsets..

Possibilistic clustering [30] is promising as a cluster mining algorithm. This method has a relaxation of probabilistic constraints and it can obtain high density regions. Still, irrelevant data which don't contribute to form cluster centers are contained in clusters.

Feature selection and dimensionality reduction in the machine learning are an important problem, and so far, many sparse classifiers and regressors have been proposed. l_1 regularizer [57] is the most successful technique to induce sparseness. Applying this technique to the possibilistic clustering, a major part of parameters becomes zero. Only non-zero parameters are essential for modeling.

5.2 Membership Regularizations

Many membership regularization techniques have been actively studied to obtain a variety of membership functions. Assume that the objective function J has a singular solution.

To solve the ill-posed problem, Tikhonov's regularization method has been used.

$$\min_U J(U, V) + \lambda\Omega(U) \tag{5.1}$$

where $\Omega(U)$ is a penalty term.

The choice of a penalty term is significant to decide a form of a membership function. The most popular penalty function is the quadratic function which is related to the Gaussian prior in the Bayesian framework.

$$\Omega(U) = \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}^2 \tag{5.2}$$

However, (5.2) cannot be expected to induce the strong sparseness. l_1 function (exact value function) [57] which is related to the Laplace prior can induce the strong sparseness of the parameters. Almost parameters become zero.

$$\Omega(U) = \sum_{k=1}^{n}\sum_{i=1}^{c} |u_{ik}| \tag{5.3}$$

The partial derivatives of the regularized objective function with respect to u_{ik} will

be uniformly zero, giving

$$\left|\frac{\partial J}{\partial u_{ik}}\right| = \lambda \quad \text{if } |u_{ik}| > 0,$$
$$\left|\frac{\partial J}{\partial u_{ik}}\right| < \lambda \quad \text{if } |u_{ik}| = 0.$$

This implies that if the partial derivatives of J falls below λ, u_{ik} will be set exactly to zero. We cannot use this function directly due to non-negativity of u_{ik}.

5.3 Sparse Possibilistic Clustering

The exact value regularization cannot be applied directly, since memberships are always non-negative. To solve this difficulty, we introduce the baseline constant [53] into the regularizer. The parameter α is a baseline constant to regularize the positive u_{ik}. The following objective function is considered.

$$J''(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m D_{ik} + \lambda \sum_{i=1}^{c}\sum_{k=1}^{n}|u_{ik} - \alpha| \tag{5.4}$$

The second term on right-hand side of (5.4) is a regularization term.

In the possibilistic clustering, each membership can be solved separately,

$$\min_{u_{ik}} (u_{ik})^m D_{ik} + \lambda|u_{ik} - \alpha|. \tag{5.5}$$

Let us decompose $u_{ik} - \alpha = \xi^+ - \xi^-$, where all elements of ξ^+ and ξ^- are nonnegative. Then, the problem (5.5) can be rewritten as

$$\min_{u_{ik},\xi^+,\xi^-} (u_{ik})^m D_{ik} + \lambda(\xi^+ + \xi^-) \tag{5.6}$$

The constraints are

$$u_{ik} - \alpha \le \xi^+,\ u_{ik} - \alpha \ge -\xi^-,\ \xi^+,\ \xi^- \ge 0. \tag{5.7}$$

Introducing the coefficients, $\beta^+, \beta^-, \delta^+, \delta^- \ge 0$, the Lagrangian is written as

$$\begin{aligned} L =& (u_{ik})^m D_{ik} + \lambda(\xi^+ + \xi^-) \\ &+ \beta^+(u_{ik} - \alpha - \xi^+) - \beta^-(u_{ik} - \alpha + \xi^-) - \delta^+\xi^+ - \delta^-\xi^- \end{aligned}$$

Solving L for ξ^+ and ξ^-, the following equations are obtained,

$$\frac{\partial L}{\partial \xi^+} = \lambda - \beta^+ - \delta^+ = 0 \tag{5.8}$$
$$\frac{\partial L}{\partial \xi^-} = \lambda - \beta^- - \delta^- = 0 \tag{5.9}$$

Since $\delta^+, \delta^- \geq 0$, the inequalities $\beta^+ \leq \lambda$ and $\beta^- \leq \lambda$ are induced from (5.9). Using (5.9), the Lagrangian is simplified as

$$L = (u_{ik})^m D_{ik} + (\beta^+ - \beta^-)(u_{ik} - \alpha) \tag{5.10}$$

Solving the Lagrangian for u_{ik}, we obtain

$$\frac{\partial L}{\partial u_{ik}} = m(u_{ik})^{m-1} D_{ik} + \beta^+ - \beta^- = 0 \tag{5.11}$$

If $\beta = \beta^+ - \beta^-$, this equation is solved as

$$u_{ik} = \Big[\frac{-\beta}{mD_{ik}}\Big]^{\frac{1}{m-1}} \tag{5.12}$$

The dual problem is written as

$$\arg \min_{-\lambda \leq \beta \leq \lambda} \Big[\frac{-\beta}{mD_{ik}}\Big]^{\frac{m}{m-1}} D_{ik} + \beta \Big\{ \Big[\frac{-\beta}{mD_{ik}}\Big]^{\frac{1}{m-1}} - \alpha \Big\} \tag{5.13}$$

Ignoring the constraint, this problem is solved as

$$\beta = -m(\alpha)^{m-1} D_{ik} \tag{5.14}$$

where the corresponding primal solution is α. If $\beta \leq -\lambda$, that is, $m(\alpha)^{m-1} D_{ik} \geq \lambda$, the optimal solution is $\beta = -\lambda$, and the corresponding primal solution is

$$u_{ik} = \Big[\frac{\lambda}{mD_{ik}}\Big]^{\frac{1}{m-1}} \tag{5.15}$$

On the other hand, the case of $\beta \geq \lambda$ is not realized since $m(\alpha)^{m-1} D_{ik}$ is always positive. Finally, we obtain

$$u_{ik} = \begin{cases} (\frac{\lambda}{mD_{ik}})^{\frac{1}{m-1}} & m(\alpha)^{m-1} D_{ik} \geq \lambda \\ \alpha & m(\alpha)^{m-1} D_{ik} \leq \lambda \end{cases} \tag{5.16}$$

For simplicity, a new coefficient $\mu = \lambda/m$ is introduced, then (5.4) and (5.16) are rewritten as

$$\min \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m D_{ik} + \mu \sum_{i=1}^{c} \sum_{k=1}^{n} |u_{ik} - \alpha|, \tag{5.17}$$

$$u_{ik} = \begin{cases} (\frac{\lambda}{mD_{ik}})^{\frac{1}{m-1}} & (\alpha)^{m-1} D_{ik} \geq \mu \\ \alpha & (\alpha)^{m-1} D_{ik} \leq \mu \end{cases} \tag{5.18}$$

Assume that α is sufficiently small, a cluster center $\boldsymbol{v}_i$ can be computed only with compact set S_i:

$$S_i = \big\{k \in \{1, \ldots, n\} | u_{ik} \neq \alpha\big\}. \tag{5.19}$$

Thus, $\boldsymbol{v}_i$ is solved as

$$\boldsymbol{v}_i = \frac{\sum\limits_{\boldsymbol{x}_k \in S_i} (u_{ik})^m \boldsymbol{x}_k}{\sum\limits_{\boldsymbol{x}_k \in S_i} (u_{ik})^m}. \tag{5.20}$$

5.4 Detection of Non-linear Regions

The proposed method can be easily extended to the kernel machine. Instead of the Euclidean distance, we use a modified distance as dissimilarity.

$$\begin{aligned} D_{ik} = K_{kk} &- \frac{1}{U_i} \sum_{j \in S} (u_{ij})^m K_{jk} \\ &+ \frac{1}{U_i^2} \sum_{j \in S} \sum_{l \in S} (u_{ij} u_{il})^m K_{jl} \end{aligned} \tag{5.21}$$

where $K_{ij} = K(x_i, x_j)$ is a kernel function.

$$K(x_i, x_j) = \exp\Big(- \frac{\|x_i - x_j\|^2}{\sigma^2} \Big) \tag{5.22}$$

In general, kernelized clustering has a high computational complexity because all sets are required to compute a distance, but the proposed method can compute a distance using only reduced sets, and hence it is an efficient method for a kernel machine.

5.5 Illustrative Examples

Artificially generated 272 points on a plane were analyzed. We analyze the proposed method in control of the parameter C. In this case, we set the baseline constant $\alpha = 0.01$, and the fuzzified parameter $m = 2$. This parameter setting is valid for sparse solutions for v_i. We did clustering this data under $c = 2$.

The result is shown in Fig.5.1, where $*$ and $\times$ show two clusters, and $+$ have a baseline membership value which is almost equal to zero. The sparseness is controlled by parameter C. Fig.5.2 shows a more sparse solution. The proposed method can remove outliers which are far from cluster centers.

And, we analyze the kernel method to detect non-linear regions. The result is shown in Fig.5.3. The proposed method is successful to detect non-linear regions. Fig.5.4 shows a more sparse solution than Fig.5.3.

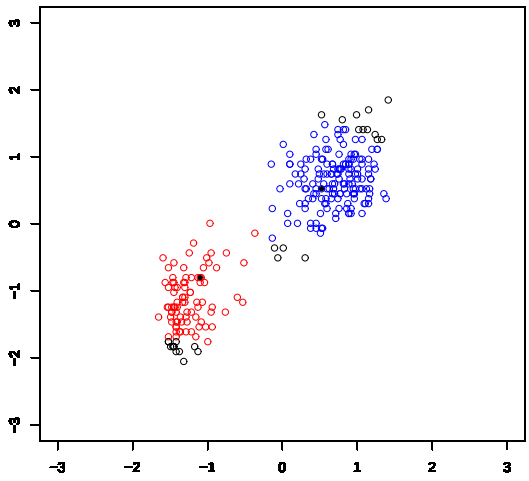

Fig. 5.1 The result of sparse method ($C = 0.010$). The proposed method can obtain compact clusters. The sparseness is shown by not containing outliers.

5.6 Discussion

We have proposed a sparse possibilistic clustering with l_1 regularization to reduce membership parameters. Illustrative examples show the sparseness of memberships induced by the proposed method and the kernelized version can obtain complicated density regions. It is successful to obtain compact clusters and reject outliers. It is a first step of cluster mining algorithms.

However, we need some techniques to decide the parameter C.

Another approach to select meaningful sets is the selection method via kernel PCA [46]. New sparse algorithms will be inspired by it.

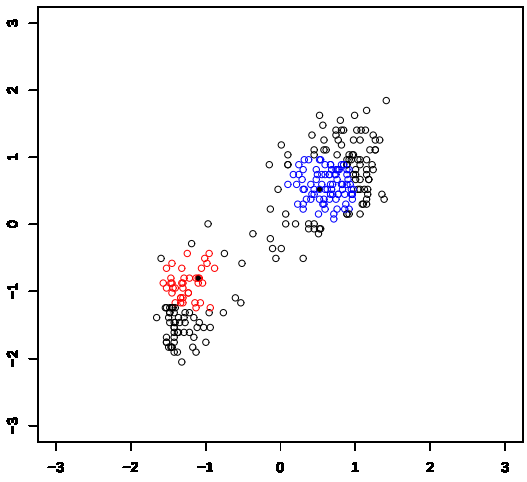

Fig. 5.2 The result of sparse method ($C = 0.005$). The parameter C can control the spareness of the algorithm.

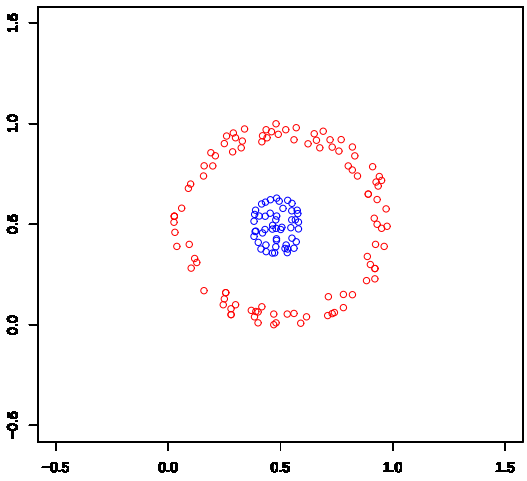

Fig. 5.3 The result of kernel method ($C = 0.02, \sigma = 0.09$). It is shown that appropriate clusters are obtained.

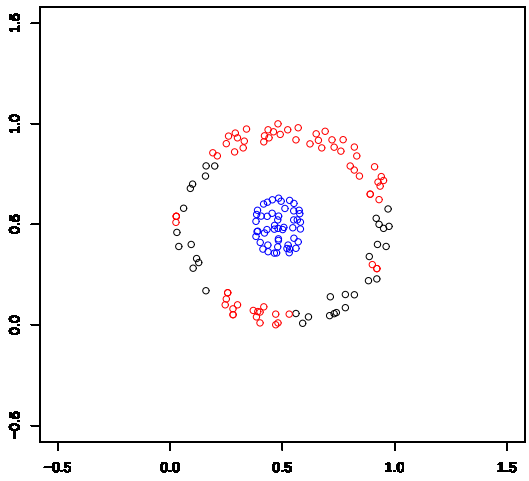

Fig. 5.4 The result of kernel method ($C = 0.001, \sigma = 0.10$). It it difficult to interpret non-active vectors.

Chapter 6

A Nonparametric Fisher Kernel Using Fuzzy Clustering

The Fisher kernel has been increasingly applied to discriminative classifiers in order to extract features from probabilistic models. It has been observed that the Fisher kernel classifiers have successful results if appropriate distributions for data are already known.

The Fisher kernel refers to the inner product in the feature space of the Fisher score, and it allows converting discrete data into continuous vectors.

However, when assuming that the distribution is unknown, parameters cannot be estimated (or make no sense), and hence an appropriate Fisher score cannot be obtained. We then have to obtain a quantity simulating the Fisher score with no assumption of the distribution.

Fuzzy clustering is one of answers to obtain the Fisher score from a nonparametric model. Recently it is elucidated that fuzzy clustering is highly related to a probabilistic model. For example, an entropy-regularized hard c-means is equivalent to a regular exponential family by setting parameters adequately.

Observing these facts, we propose a new nonparametric Fisher kernel derived from fuzzy clustering instead of a probabilistic model. An illustrative example shows the effectiveness of the proposed method.

6.1 The Fisher kernel

Let $\mathcal{X}$ denote the domain of data, which is discrete or continuous. Let us also assume that a probabilistic model $p(\boldsymbol{x}|\boldsymbol{\theta})$, $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathcal{X}$, $\boldsymbol{\theta} \in \boldsymbol{R}^p$ is available. Given a parameter estimate $\hat{\boldsymbol{\theta}}$ from given data, the feature vector which is called the Fisher score is obtained as

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = (\frac{\partial \log p(\boldsymbol{x}|\hat{\boldsymbol{\theta}})}{\partial \theta_1}, \cdots, \frac{\partial \log p(\boldsymbol{x}|\hat{\boldsymbol{\theta}})}{\partial \theta_p})^T. \tag{6.1}$$

The Fisher kernel refers to the inner product in this space.

$$K_{ij}^f = k^f(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^T I^{-1} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_j), \tag{6.2}$$

where I is the Fisher information matrix

$$I = \frac{1}{n} \sum_{k=1}^{n} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_k) \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_k)^T. \tag{6.3}$$

In the following, we will discuss how to determine $p(\boldsymbol{x}|\boldsymbol{\theta})$. Let us assume that true distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$ is determined as a mixture model of true class distributions:

$$p(\boldsymbol{x}|\boldsymbol{\pi}) = \sum_{i=1}^{c} \pi_i p(\boldsymbol{x}|y=i), \tag{6.4}$$

where $\sum_{i=1}^{c} \pi_i = 1$. The Fisher score for $p(\boldsymbol{x}|\boldsymbol{\pi})$ is

$$\frac{\partial \log p(\boldsymbol{x}|\boldsymbol{\pi})}{\partial \pi_i} = \frac{p(\boldsymbol{x}|y=i)}{\sum_{i=1}^{c} \pi_i p(\boldsymbol{x}|y=i)} = \frac{P(y=i|\boldsymbol{x})}{\pi_i}. \tag{6.5}$$

In the case when we know the true distribution, the class information is fully preserved, that is, the loss function L is

$$L(U) = \sum_{i=1}^{c} E_x(U_i(x) - P(y=i|\boldsymbol{x}))^2 = 0, \tag{6.6}$$

where the function U is the best estimator of the posterior distribution. In general, however, the true distribution is unknown, in which case we should consider the method of fuzzy clustering in the next section.

6.2 A Nonparametric Fisher Kernel

We propose a nonparametric Fisher kernel derived from fuzzy clustering. It is derived from the relation between regular exponential families and KL-divergence regularized hard c-means.

In the regular exponential family, the density function is

$$p(\boldsymbol{x}|y=i,\boldsymbol{\theta}) = \frac{1}{Z_{\boldsymbol{\theta}}} \exp\Big(- D_A(x,\mu_i)\Big). \tag{6.7}$$

where μ_i is an expectation of $p(x|y,\boldsymbol{\theta})$, $Z_{\boldsymbol{\theta}}$ is a normalizer, and A is an parameter of adaptive distortion measure. The most famous adaptive distortion measure is the Mahalanobis distance $D : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$,

$$D_A(\boldsymbol{x},\boldsymbol{\mu}_i) = (\boldsymbol{x}-\boldsymbol{\mu}_i)^T A_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i) \tag{6.8}$$

By contrast to a mixture of regular exponential family and fuzzy c-means in Chapter. 2, the nonparametric Fisher score is

$$\frac{\partial \log p(\boldsymbol{x}_k|\boldsymbol{\pi})}{\partial \pi_i} = \frac{p(y=i|\boldsymbol{x}_k)}{\pi_i} \approx \frac{u_{ik}}{\alpha_i}. \tag{6.9}$$

where $\alpha_i = \frac{1}{n}\sum_{k=1}^n u_{ik}$. Once the Fisher score is obtained using a clustering algorithm to a large amount of unlabeled data, the Fisher kernel is derived from (6.2). This kernel can be applied to a kernel classifier such as that for support vector machines.

6.3 Illustrative Examples

Artificially generated 150 points on a plane were analyzed. This clustering problem is the almost same as that in [50]. First, seven clusters are obtained by the algorithm of entropy-regularized hard c-means as shown in Fig. 6.1, and then the Fisher score is derived from the membership values. We applied two algorithms of the hard c-means and the entropy-regularized hard c-means to data mapped into the Fisher feature space. Fig. 6.2 shows the result from hard c-means algorithm, which is the same as that in [50]. On the other hand, Fig. 6.3 shows a successful result from the entropy-based fuzzy c-means. While Tsuda et. al [50] develop a new and complicated algorithm due to the failure of the hard c-means, we are successfully using the simple algorithm of fuzzy c-means. Fuzzy c-means algorithm has robustness for nuisance dimensions in the feature space of the Fisher score.

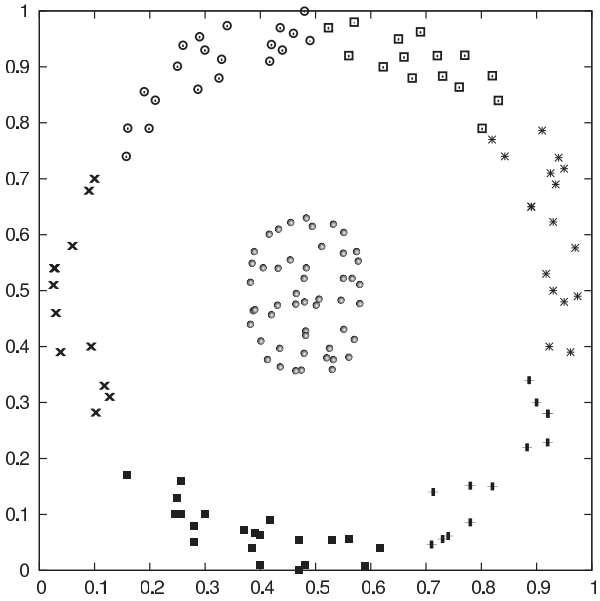

Fig. 6.1 Seven clusters obtained from entropy-regularized fuzzy c-means. In this artificial data, there are two true clusters. However, we cannot obtain true clusters using a simple clustering method. Instead, we approximate true clusters with a number of 'rough' clusters. This implies that $p(\boldsymbol{x})$ or a global information of data is estimated.

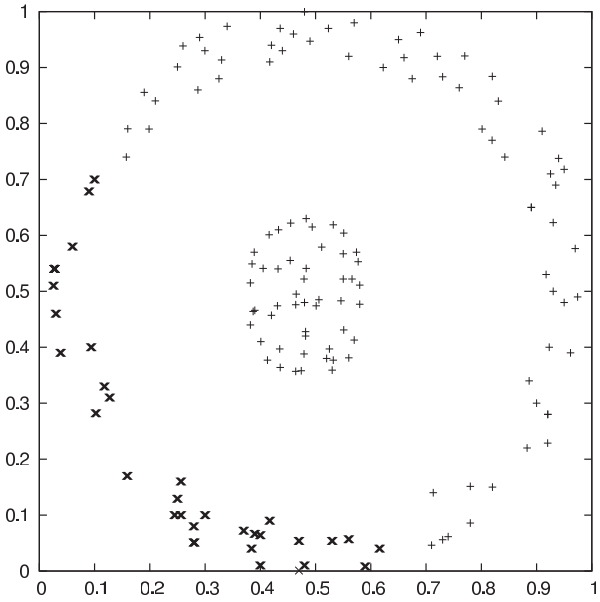

Fig. 6.2 Two clusters obtained from HCM in the feature space. The same failure was shown in [50]. It is caused by nuisance dimensions.

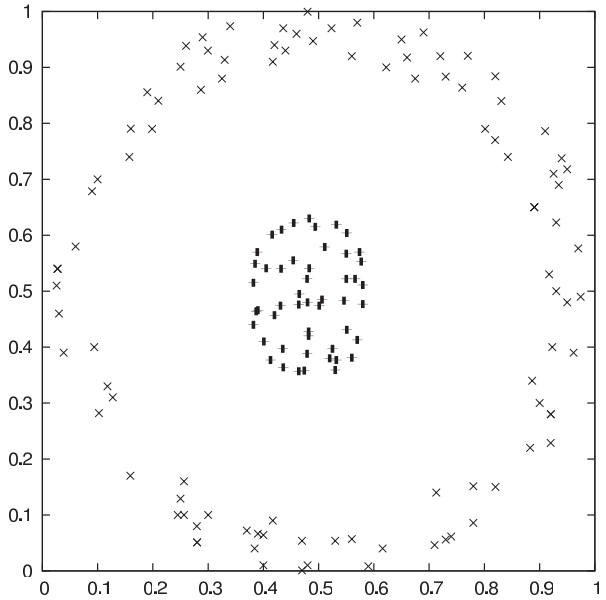

Fig. 6.3 Two clusters obtained from eFCM in the feature space. This implies that fuzzy c-means methods are robust to nuisance dimensions.

6.4 Discussion

We have proposed a nonparametric Fisher kernel derived from fuzzy clustering instead of a probabilistic model. In the proposed kernel, a nonparametric Fisher score using fuzzy clustering works as well as the ordinary Fisher score. The proposed kernel does not assume a probabilistic distribution. The numerical examples have shown nonlinearly separated clusters from kernelized fuzzy c-means using the nonparametric Fisher kernel.

Future studies include semi-supervised classification problems using the proposed method, and theoretical studies of relations between the proposed method and mutual information (MI) kernels [47] including the Fisher kernel.

Chapter 7

The Method of c-means Clustering on a Manifold

In this chapter, we discuss fuzzy c-means clustering on the statistical manifold of the multinomial distribution. It is difficult to apply directly the geodesic distance as a metric. Instead, we apply the Kullback-Leibler (KL) divergence and the Hellinger distance to fuzzy c-means. In particular, the Hellinger distance is an approximation of the geodesic distance.

7.1 Data Representation

In clustering algorithms for discrete data, data representation is a very crucial part. Since raw data are very sparse and high-dimensional, it is not appropriate to analyze them directly. Low-dimensional representations are hence required.

Recently, a novel representation has been used in several studies [11, 26]. In this representation, datum x is mapped to a probability distribution $p(\boldsymbol{x})$. It is often called probabilistic mapping [26]. If $p(\boldsymbol{x})$ is replaced with a particular distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$, datum x is mapped to a parameter $\boldsymbol{\theta}$ because $p(\boldsymbol{x}|\boldsymbol{\theta})$ is identified by $\boldsymbol{\theta}$.

The multinomial distribution is very important to represent documents and biological sequences. For example, a document is represented as the term frequency (TF) in many publications [33]. It realizes a probabilistic mapping because TF representation is the maximum likelihood estimator of the multinomial distribution.

The choice of metric is also an essential part in clustering algorithms. In general,

the Euclidean distance has often been used. However, it is not appropriate for the above representation.

The space of all parameters $\boldsymbol{\theta}$ is a Riemannian manifold with the Fisher information metric [2]. The multinomial manifold forms multidimensional simplex. The Riemannian manifold is usually "curved" while the Euclidean space is "flat". Therefore, the Euclidean distance is not suitable as a metric in this setting.

The geodesic distance is a suitable distance for the manifold [31]. But, it is difficult for almost all distributions to derive distances analytically, except for the multinomial distribution.

7.2 The Multinomial Manifold

We now present a brief introduction of the multinomial manifold. For further details, see [2, 27].

A probability density function of the multinomial distribution is

$$p(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{(\sum_{j=1}^{m+1} x_j)!}{\prod_{j=1}^{m+1} x_j!} \prod_{j=1}^{m+1} \theta_j^{x_j} \tag{7.1}$$

x_j is the number of occurrences corresponding to jth event of all $m+1$ events, θ_j is a parameter.

A family of (7.1),

$$S = \{p(\boldsymbol{x}, \boldsymbol{\theta}) | \boldsymbol{\theta} \in \Theta\}$$

is a $m+1$-dimensional manifold with the coordinate system $\boldsymbol{\theta} \in \Theta$. The parameter space Θ is the open m-dimensional simplex,

$$P_m = \Big\{ \boldsymbol{\theta} \in R^{m+1} : \forall j,\ \theta_j\ > 0, \quad \sum_{j=1}^{m+1} \theta_j = 1 \Big\}.$$

S is a Riemannian space with the Fisher information metric,

$$g_{ij}(\boldsymbol{\theta}) = E_{\boldsymbol{\theta}} \Big[\frac{\partial \log p(\boldsymbol{x}, \boldsymbol{\theta})}{\partial \theta_i} \frac{\partial \log p(\boldsymbol{x}, \boldsymbol{\theta})}{\partial \theta_j} \Big] \tag{7.2}$$

$$= \frac{\delta_{ij}}{\theta_i} \tag{7.3}$$

Now we consider another coordinate system $\xi = (\xi_1, \ldots, \xi_{m+1})$,

$$\xi_1 = 2\sqrt{\theta_1},\ \ldots,\ \xi_{m+1} = 2\sqrt{\theta_{m+1}}.$$

The Jacobi matrices are diagonal matrices,

$$J_{\alpha i} = \frac{\partial \xi_\alpha}{\partial \theta_i} = \frac{1}{\sqrt{\theta_i}} \delta_{\alpha i},$$
$$\bar{J}_{i\alpha} = \frac{\partial \theta_i}{\partial \xi_\alpha} = \frac{1}{2} \xi_\alpha \delta_{i\alpha}. \tag{7.4}$$

The Riemannian metric with ξ is derived as

$$g_{\alpha\beta}(\xi) = \sum_{i=1}^{m+1} \sum_{j=1}^{m+1} \bar{J}_{i\alpha} \bar{J}_{j\beta} g_{ij}$$
$$= \delta_{\alpha\beta} \tag{7.5}$$

The new coordinate system ξ has a useful feature to calculate the geodesic distance.

$$\sum_{\alpha=1}^{m+1} (\xi_\alpha)^2 = 4 \tag{7.6}$$

Hence, S is considered as m-dimensional positive sphere of radius 2 with ξ.

The squared geodesic distance on the sphere with the Euclidean metric

$$ds^2 = \sum_{\alpha=1}^{m+1} (d\xi_\alpha)^2$$
$$= \sum_{\alpha=1}^{m+1} \sum_{\beta=1}^{m+1} \delta_{\alpha\beta} d\xi_\alpha d\xi_\beta \tag{7.7}$$

It is equivalent to the squared geodesic distance with the Fisher information metric,

$$ds^2 = \sum_{\alpha=1}^{m+1} \sum_{\beta=1}^{m+1} g_{\alpha\beta}(\xi) d\xi_\alpha d\xi_\beta \tag{7.8}$$

Hence, S with ξ is an isometric embedding into R^{m+1}. The Riemannian distance between ξ_α and ξ'_α is the great circle.

$$\sum_{\alpha=1}^{m+1} \xi_\alpha \xi'_\alpha = 4 \sum_{i=1}^{m+1} \sqrt{\theta_i} \sqrt{\theta'_i} = 4 \cos \eta$$

Finally, we obtain the Riemannian distance,

$$D_G(\boldsymbol{\theta}, \boldsymbol{\theta}') = 2 \arccos \Big(\sum_{j=1}^{m+1} \sqrt{\theta_j \theta'_j} \Big). \tag{7.9}$$

The direct use of the geodesic distance is difficult to analyze. We introduce other familiar dissimilarity measures. The Kullback-Leibler (KL) divergence has been frequently used to measure the distance between probability distributions. For the multinomial family, the KL-divergence is defined by

$$D_{KL}(\boldsymbol{\theta}||\boldsymbol{\theta}') = \sum_{j=1}^{m+1} \theta_j \log \frac{\theta_j}{\theta'_j}. \tag{7.10}$$

The Hellinger distance have been also used.

$$D_H(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sqrt{\sum_{j=1}^{m+1} \left(\sqrt{\theta_j} - \sqrt{\theta'_j}\right)^2} \tag{7.11}$$

It is related to D_G,

$$D_H(\boldsymbol{\theta}, \boldsymbol{\theta}') = 2\sin(D_G(\boldsymbol{\theta}, \boldsymbol{\theta}')/4)$$

Thus, as $\boldsymbol{\theta}' \to \boldsymbol{\theta}$, D_H approximates $\frac{1}{2}D_G$ to the second order:

$$D_H(\boldsymbol{\theta}, \boldsymbol{\theta}') = \frac{1}{2} D_G(\boldsymbol{\theta}, \boldsymbol{\theta}') + O(D_G^3(\boldsymbol{\theta}, \boldsymbol{\theta}')) \tag{7.12}$$

We show equal distance contours from one point in $\mathbb{P}_2$. The contour by the geodesic distance is shown in Fig. 7.1. Compared to the contour by the Euclidean distance shown in Fig. 7.2, the parameter space is much different from the usual Euclidean space. The contour by the KL-divergence is shown in Fig. 7.3, and the contour by the Hellinger distance is shown in Fig. 7.4. The Hellinger distance and the KL-divergence are better than the Euclidean distance to use on the simplex.

7.3 Hard c-Means using the KL-Divergence

Probability vectors to be clustered are denoted by $\boldsymbol{p}_k = (\theta_{1k}, \ldots, \theta_{k,m+1}) \in P_m$, $k = 1, \ldots, n$.

As is well-known, hard c-means clustering using the KL-divergence is based on the optimization of an objective function.

$$J = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} D_{KL}(\boldsymbol{p}_k||\boldsymbol{v}_i) \tag{7.13}$$

Note that $\boldsymbol{p}_k$ and $\boldsymbol{v}_i$ are probability vectors defined by

$$\sum_{j=1}^{m+1} \theta_{kj} = \sum_{j=1}^{m+1} v_{ij} = 1. \tag{7.14}$$

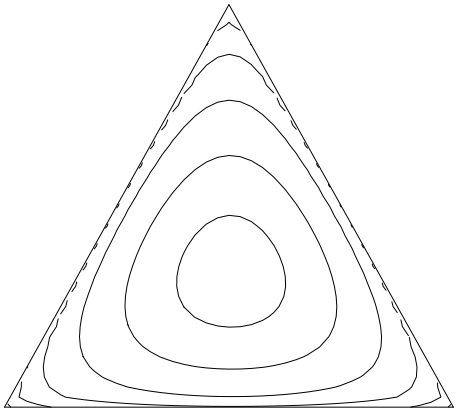

Fig. 7.1 The contours of the geodesic distance. They are ideal contours on the simplex.

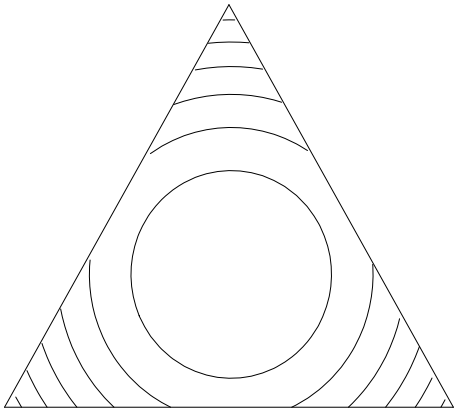

Fig. 7.2 The contours of the Euclidean distance. They are inappropriate on the simplex. The geometry of the simplex is not reflected in the measure.

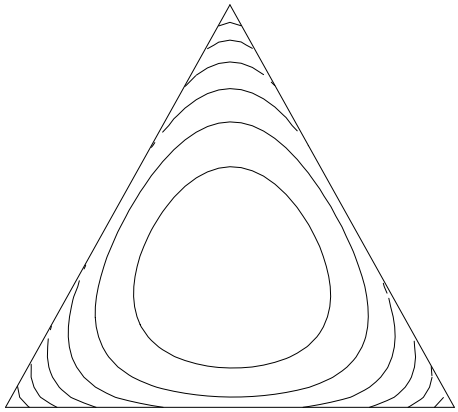

Fig. 7.3 The contours of the KL-divergence. They are better contours on the simplex. The KL-divergence have been frequently used to compare two probability distributions.

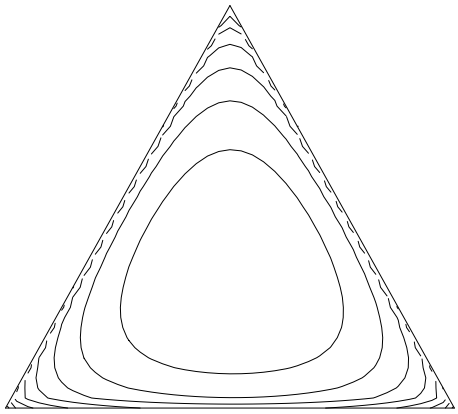

Fig. 7.4 The contours of the Hellinger distance. They are also better contours on the simplex. The Hellinger distance is an approximation of the geodesic distance.

The constraints is assumed for U,

$$M_c = \{\, U = (u_{ik}) \,:\, \sum_{i=1}^{c} u_{ik} = 1,\ u_{jk} \in \{0,1\},\ \forall j,k \,\}. \tag{7.15}$$

A cluster assignment of $\boldsymbol{p}_k$ is the same as the usual hard c-means,

$$h = \arg\min_{1 \le i \le c} D_{KL}(\boldsymbol{p}_k || \boldsymbol{v}_i). \tag{7.16}$$

Cluster centers are derived from the Lagrange multiplier method with the constraint.

$$L = J + \sum_{i=1}^{c} \mu_i (\sum_{j=1}^{m+1} v_{ij} - 1)$$

$$\frac{\partial L}{\partial v_{ij}} = -\sum_{k=1}^{n} \frac{u_{ik}\theta_{kj}}{v_{ij}} + \mu_i = 0$$

As a result, cluster centers are calculated in the same way as the usual hard c-means.

$$v_{ij} = \frac{\sum\limits_{k=1}^{n} u_{ik}\theta_{kj}}{\sum\limits_{k=1}^{n} u_{ik}} \tag{7.17}$$

7.4 Hard c-Means using the Hellinger Distance

We also apply the Hellinger distance to hard c-means. The Hellinger distance is closely related with the geodesic distance [31]. Now we consider the squared Hellinger distance,

$$D_H^2(\boldsymbol{p}_k, \boldsymbol{v}_i) = \sum_{j=1}^{m+1} \left(\sqrt{\theta_{kj}} - \sqrt{v_{ij}} \right)^2. \tag{7.18}$$

Therefore, The objective function is

$$J(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} D_H^2(\boldsymbol{p}_k, \boldsymbol{v}_i). \tag{7.19}$$

The cluster assignment is almost the same as (7.16).

$$h = \arg\min_{1 \le i \le c} D_H^2(\boldsymbol{p}_k, \boldsymbol{v}_i) \tag{7.20}$$

Cluster centers are also derived from the Lagrange multiplier method with the constraint.

$$L = J + \sum_{i=1}^{c} \mu_i (\sum_{j=1}^{m+1} v_{ij} - 1)$$

$$\frac{\partial L}{\partial v_{ij}} = -\sum_{k=1}^{n} \frac{u_{ik}(\sqrt{\theta_{kj}} - \sqrt{v_{ij}})}{\sqrt{v_{ij}}} + \mu_i = 0$$

As a result, cluster centers are calculated in the different way from the usual hard c-means.

$$v_{ij} = \frac{\sum_{k=1}^{n} \sum_{l=1}^{n} u_{ik} u_{il} \theta_{kj} \theta_{lj}}{\sum_{k=1}^{n} \sum_{l=1}^{n} u_{ik} u_{il} \langle \boldsymbol{p}_k, \boldsymbol{p}_l \rangle} \tag{7.21}$$

7.5 Fuzzy c-Means Based on the Multinomial Manifold

Hard c-means is effective for data having well-separated clusters, but is not much effective for data having overlapping clusters. Fuzzy c-means is useful to analyze overlapping data.

Fuzzy c-means use the relaxed constraint,

$$M_f = \{\, U = (u_{ik}) \,:\, \sum_{i=1}^{c} u_{ik} = 1,\ u_{jk} \geq 0,\ \forall j, k \,\}, \tag{7.22}$$

compared to the hard constraints M_c. Therefore, fuzzy c-means obtain soft assignment to clusters.

Standard fuzzy c-means method is proposed by [5]. The objective function is

$$J(V, U) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^{\nu} D(\boldsymbol{p}_k, \boldsymbol{v}_i) \tag{7.23}$$

where $D(\boldsymbol{p}_k, \boldsymbol{v}_i)$ is the KL-divergence or the Hellinger distance and ν is a parameter. The solution for U is

$$u_{ik} = \frac{D(\boldsymbol{p}_k, \boldsymbol{v}_i)^{-\frac{1}{\nu-1}}}{\sum_{j=1}^{c} D(\boldsymbol{p}_k, \boldsymbol{v}_j)^{-\frac{1}{\nu-1}}} \tag{7.24}$$

The solution of V using the KL-divergence is

$$v_{ij} = \frac{\sum_{k=1}^{n} (u_{ik})^{\nu} \theta_{kj}}{\sum_{k=1}^{n} (u_{ik})^{\nu}} \tag{7.25}$$

The solution of V using the Hellinger distance is

$$v_{ij} = \frac{\sum_{k=1}^{n} \sum_{l=1}^{n} (u_{ik} u_{il})^{\nu} \theta_{kj} \theta_{lj}}{\sum_{k=1}^{n} \sum_{l=1}^{n} (u_{ik} u_{il})^{\nu} \langle \boldsymbol{p}_k, \boldsymbol{p}_l \rangle} \tag{7.26}$$

Fuzzy c-Means by the entropy regularization [38] is a regularized hard c-means to obtain a smoothing membership. The objective function is

$$J(V, U, \boldsymbol{\alpha}) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} D(\boldsymbol{p}_k, \boldsymbol{v}_i) + \lambda^{-1} \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} \log \frac{u_{ik}}{\alpha_i}. \tag{7.27}$$

The second term of the right-hand side of (7.27) is the regularization term. $\alpha = (\alpha_1, \ldots, \alpha_c)$ is a cluster volume size variable with the constraint

$$A = \{ \alpha = (\alpha_1, \ldots, \alpha_c) \ : \ \sum_{i=1}^{c} \alpha_i = 1, \ \alpha_j \geq 0, \ \forall j \ \}. \tag{7.28}$$

The solution for U is

$$u_{ik} = \frac{\alpha_i \exp(-\lambda D(\boldsymbol{p}_k, \boldsymbol{v}_i))}{\sum_{j=1}^{c} \alpha_j \exp(-\lambda D_(\boldsymbol{p}_k, \boldsymbol{v}_j))}. \tag{7.29}$$

The solution for V is the same as (7.25). The solution for α is

$$\alpha_i = \frac{1}{n} \sum_{k=1}^{n} u_{ik}. \tag{7.30}$$

7.6 Illustrative Example

Artificially generated 60 points on $2d$-simplex were analyzed. We applied hard c-means with three different dissimilarities to data. Fig. 7.5 shows the result from hard c-means with the Euclidean distance, Fig. 7.6 shows the result from hard c-means with the KL-divergence, and Fig. 7.7 shows the result from hard c-means with the Hellinger distance. Fig. 7.7 is quite different from other two clustering results. The Hellinger distance may be the most appropriate dissimilarity on the manifold.

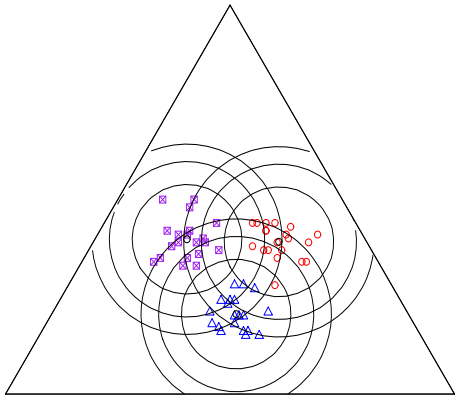

Fig. 7.5 The result using Euclidean distance. If the feature space is the Euclidean, it is the best result. But the feature space is the simplex, and not the Euclidean.

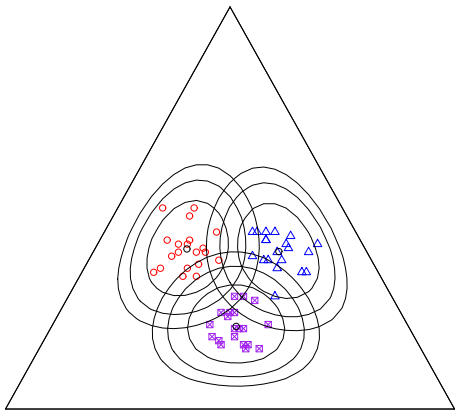

Fig. 7.6 The result using KL-divergence. The geometry of the simplex is reflected in the result.

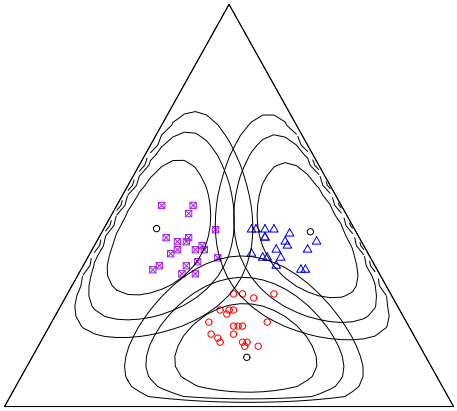

Fig. 7.7 The result using Hellinger distance. The Hellinger distance is the most effective measure on the simplex.

Artificially generated 150 points on $2d$-simplex were analyzed. We applied two fuzzy c-means methods with three different dissimilarities to data. Fig. 7.8, Fig. 7.9, and Fig. 7.10 show the results from standard fuzzy c-means (sFCM) with three metrics. Parameter ν is 2.0. Fig. 7.11, Fig. 7.12, and Fig. 7.13 show the results from entropy-based fuzzy c-means (eFCM) with three metrics. Parameter λ is 5.0. The results show fuzzy c-means with the KL-divergence or the Hellinger distance seems to be better than others. Fuzzy c-means with the Hellinger distance emphasizes the geometry strongly.

7.7 Discussion

In this paper, we have discussed hard c-means and fuzzy c-means on the multinomial manifold. Instead of the geodesic distance, the KL-divergence and the Hellinger distance have been used. The Hellinger distance is a good approximation of the geodesic distance compared to the KL-divergence. These metrics are more useful to detect clusters on the manifold than the Euclidean distance.

Future studies include real-world applications such as document clustering, and developments of clustering algorithms using the information geometry.

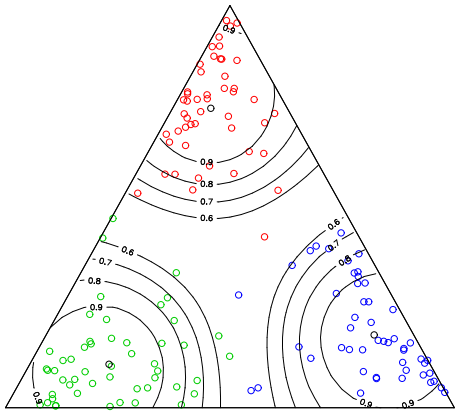

Fig. 7.8 Equal membership contours by sFCM using the Euclidean distance.

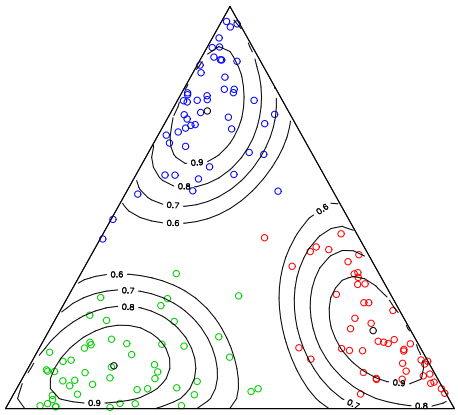

Fig. 7.9 Equal membership contours by sFCM using the KL-divergence.

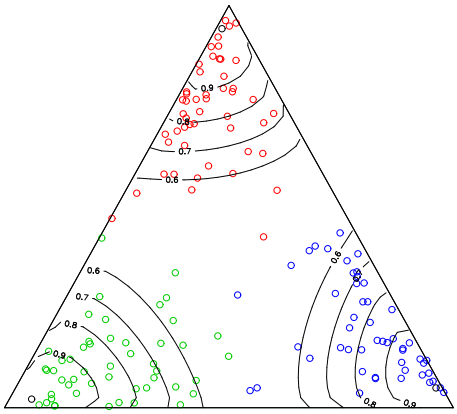

Fig. 7.10 Equal membership contours by sFCM using the Hellinger distance.

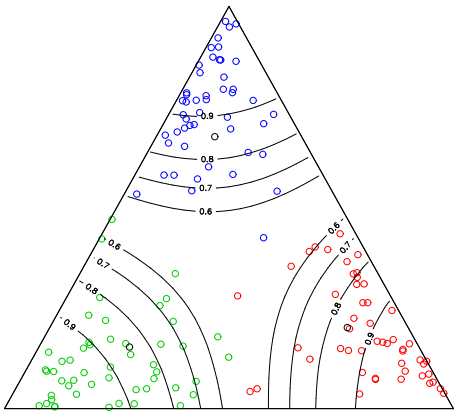

Fig. 7.11 Equal membership contours by eFCM using the Euclidean distance.

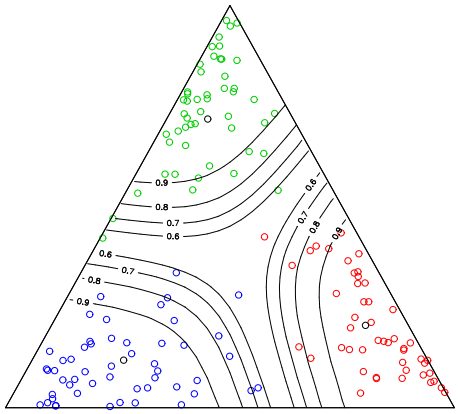

Fig. 7.12 Equal membership contours by eFCM using the KL-divergence.

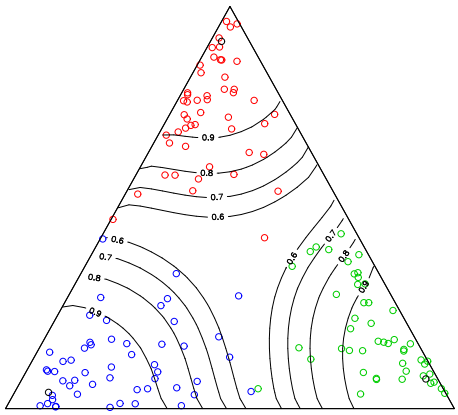

Fig. 7.13 Equal membership contours by eFCM using the Hellinger distance.

Chapter 8

Conclusion

In this dissertation, we discuss two aspects of kernel-based clustering: efficient algorithms and the information geometrical approach to design kernels.

Other kernel-based clustering algorithms have very high computational efforts in general, which is a very crucial issue for applications. We proposed efficient algorithms of kernel-based clustering using on-line learning. The proposed methods have a very lower computational complexity $O(n)$ while kernel hard c-means method has a computational complexity $O(n^3)$. And, from the view of the reproducing kernel Hilbert space, we obtained an extended on-line algorithm to handle additional objects. We moreover proposed a sequential fuzzy c-means method from the on-line learning scheme. It is highly related to the on-line version of the EM algorithms.

Another approach to an efficient algorithm is introducing sparseness. It aims to reduce the computation of coefficients required by solutions of clustering. we proposed possibilistic clustering with l_1 regularization. The Lagrangian dual problem of this algorithm is very complex but is derived by checking the violation of given constraints. In this algorithm, a major part of membership parameters is set to a specific value. Hence, only active vectors are required as solutions.

We propose a kernel derived from the connection between fuzzy clusters and generative models. The proposed kernel is more flexible than the original Fisher kernel since it does not require a generative process. The Fisher kernel requires a full generative model $p(x)$, but it is redundant because we want to know only the label y or the posterior $p(y|x)$, not a full model. In contrast, our kernel is formed with a class-conditional posterior and prior. This information is estimated by discriminative clustering such as fuzzy c-means. In addition, the dependence on the model selection is avoided.

An information diffusion kernel matrix is an affinity matrix extracted from some distance function. This implies that an appropriate distance function in data space leads to a new kernel. In this dissertation, we focused on the multinomial manifold. The geodesic distance, however, cannot be directly applied, and we therefore applied the Kullback-Leibler divergence and the Hellinger distance to hard c-means method directly. These two metrics are approximations of the geodesic distance.

For further studies, the statistical property of standard fuzzy c-means method has to be revealed. It is not certain that the standard fuzzy c-means method is completely heuristic. If a relation between the standard fuzzy c-means method and the EM algorithm is clarified, it is possible that our ways to design kernels can be applied to all other fuzzy c-means methods.

Bibliography

[1] S. Akaho. EM algorithm : applications to clustering and recent trends. *SOFT*, 12(5):2–10, 2000.

[2] S. Amari and H. Nagaoka. Methods of Information Geometry. *10th European Conference on Machine Learning*, 70:1–25, 2000.

[3] A. Banerjee, S. Merugu, I.S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

[4] Y. Bengio and Y. Grandvalet. Semi-supervised learning by entropy minimization. *Advances in Neural Information Processing Systems*, 17, 2005.

[5] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers Norwell, MA, USA, 1981.

[6] J.C. Bezdek, J. Keller, R. Krishnapuram, and N.R. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer Academic Publishers, 1999.

[7] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] L. Bottou. Stochastic learning. *Lecture Notes In Artificial Intelligence*, 3176:146–168, 2004.

[9] L. Bottou, Y. Bengio, et al. Convergence properties of the K-means algorithms. *Advances in Neural Information Processing Systems*, 7:585–592, 1995.

[10] C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[11] M. Collins, S. Dasgupta, and R. Schapire. A generalization of principal component analysis to the exponential family. *Advances in Neural Information Processing Systems*, 14, 2002.

[12] I. Csiszar. I-Divergence Geometry of Probability Distributions and Minimization Problems. *The Annals of Probability*, 3(1):146–158, 1975.

[13] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[14] I.S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, 2004.

[15] A. Doucet and N. De Freitas. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[16] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley New York, 1973.

[17] J.C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1974.

[18] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids,(1998), 1998.

[19] M. Girolami. Mercer kernel-based clustering in feature space. *Neural Networks, IEEE Transactions on*, 13(3):780–784, 2002.

[20] R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, 2002.

[21] F. Höpper, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. Wiley, 1999.

[22] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley-Interscience, 2000.

[23] H. Ichihashi, K. Honda, and N. Tani. Gaussian mixture PDF approximation and fuzzy c-means clustering with entropy regularization. *Proc. of the 4th Asian Fuzzy System Symposium*, pages 217–221, 2000.

[24] T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 149–158, 1999.

[25] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. *Advances in Neural Information Processing Systems*, 11:487–493, 1998.

[26] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844, 2004.

[27] R.E. Kass. The Geometry of Asymptotic Inference. *Statistical Science*, 4(3):188–219, 1989.

[28] J. Kivinen, AJ Smola, and RC Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004.

[29] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[30] R. Krishnapuram and J.M. Keller. A possibilistic approach to clustering. *IEEE*

Transactions on Fuzzy Systems, 1(2):98–110, 1993.

[31] J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *The Journal of Machine Learning Research*, 6:129–163, 2005.

[32] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1(281-297):14, 1967.

[33] C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[34] G.J. McLachlan and K.E. Basford. *Mixture models. Inference and applications to clustering*. Marcel Dekker, 1988.

[35] T. Minka. Discriminative models, not discriminative training. Technical report, Technical Report MSR-TR-2005-144, Microsoft Research, October 2005. ftp://ftp.research.microsoft.com/pub/tr/TR-2005-144.pdf, 2005.

[36] S. Miyamoto. *Fuzzy Sets in Informational Retrieval and Cluster Analysis*. Kluwer Academic Publisher, 1990.

[37] S. Miyamoto. *Introduction to Cluster Analysis (in Japanese)*. Morikita, 1999.

[38] S. Miyamoto and M. Mukaidono. Fuzzy c-means as a regularization and maximum entropy approach. *Proc. of the 7th International Fuzzy Systems Association World Congress (IFSA ' 97)*, 2:86–92, 1997.

[39] S. Miyamoto and Y. Nakayama. Algorithms of hard c-means clustering using kernel functions in support vector machines. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 7(1):19–24, 2003.

[40] S. Miyamoto and D. Suizu. Fuzzy *c*-means clustering using kernel functions in support vector machines. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 7(1):25–30, 2003.

[41] S. Miyamoto and K. Umayahara. Fuzzy c-means with variables for cluster sizes. *Proc. of the Fuzzy System Symposium*, 16:537–538, 2000.

[42] T. Murofushi. Conditional Probabilities and Fuzzy Entropy. *Journal of Japan Society for Fuzzy Theory and Systems*, 2(1):95–99, 1990.

[43] R.M. Neal and G.E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. *Learning in Graphical Models*, pages 355–368, 1998.

[44] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[45] S.J. Roberts, R. Everson Jr, and I. Rezek. Minimum entropy data partitioning.

Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470), 2, 1999.

[46] B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press Cambridge, MA, USA, 2001.

[47] M. Seeger. Covariance Kernels from Bayesian Generative Models. *Advances in Neural Information Processing Systems*, 14, 2002.

[48] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[49] K. Tsuda, S. Akaho, M. Kawanabe, and K.R. Muller. Asymptotic properties of the Fisher kernel. *Neural Computation*, 16(1):115–137, 2004.

[50] K. Tsuda, M. Kawanabe, and K.R. Muller. Clustering with the Fisher score. *Advances in Neural Information Processing Systems*, 15:729–736, 2003.

[51] K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.R. Müller. A New Discriminative Kernel From Probabilistic Models. *Advances in Neural Information Processing Systems*, 14, 2002.

[52] K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18:268–275, 2002.

[53] K. Tsuda and T. Kudo. Clustering graphs by weighted substructure mining. *Proceedings of the 23rd international conference on Machine learning*, pages 953–960, 2006.

[54] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

[55] G. Wahba. *Spline Models for Observational Data*. Society for Industrial Mathematics, 1990.

[56] C.K.I. Williams and C.E. Rasmussen. Gaussian processes for regression. *Advances in Neural Information Processing Systems*, 8:514–520, 1996.

[57] P.M. Williams. Bayesian regularisation and pruning using a Laplace prior. *Neural Computation*, 7:117–143, 1995.

Appendix

Pseudo codes

Algorithm 1 Batch competitive learning

Require: A set of objects $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^p$

Require: The number of clusters $c \in \mathbb{N}$

Require: A set of cluster center $V^0 = (\boldsymbol{v}_1^0, \ldots, \boldsymbol{v}_c^0) \in \mathbb{R}^p$

Require: An initial learning rate α^0

Require: A damping coefficient η

Require: The truncation number $\mathcal{T}$

$t = 0$

while $t < \mathcal{T}$ **do**

for $k = 1, \ldots, n$ **do**

$\boldsymbol{x}_k \in G_h \Longleftrightarrow h = \arg\min_{1 \le i \le c} \|\boldsymbol{x}_k - \boldsymbol{v}_i^t\|^2$

end for

for $i = 1, \ldots, c$ **do**

$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \alpha(t) \sum_{\boldsymbol{x}_k \in G_i} (\boldsymbol{x}_k - \boldsymbol{v}_i^t)$

end for

$\alpha(t+1) = \alpha(t) - \eta\alpha^0$

$t = t + 1$

end while

Algorithm 2 On-line competitive learning

Require: A stream of objects $X = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^t) \in \mathbb{R}^p$

Require: The number of clusters $c \in \mathbb{N}$

Require: A set of cluster center $V^0 = (\boldsymbol{v}_1^0, \ldots, \boldsymbol{v}_c^0) \in \mathbb{R}^p$

Require: An initial learning rate α^0

Require: A damping coefficient η

Require: The truncation number $\mathcal{T}$

$t = 0$

while $t < \mathcal{T}$ **do**

$\boldsymbol{x}^t \in G_h \Longleftrightarrow h = \arg\min_{1 \le i \le c} \|\boldsymbol{x}^t - \boldsymbol{v}_i^t\|^2$

$\boldsymbol{v}_h^{t+1} = \boldsymbol{v}_h^t + \alpha(t)(\boldsymbol{x}^t - \boldsymbol{v}_h^t)$

$\alpha(t+1) = \alpha(t) - \eta\alpha^0$

$t = t + 1$

end while

Algorithm 3 Hard c-means

Require: A set of objects $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^p$

Require: The number of clusters $c \in \mathbb{N}$

Require: A set of cluster center $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c) \in \mathbb{R}^p$

Require: A tolerance criterion $\epsilon \in \mathbb{R}^+$

repeat

$\bar{V} = V$

for $k = 1, \ldots, n$ **do**

$\boldsymbol{x}_k \in G_h \Longleftrightarrow h = \arg\min_{1 \le i \le c} \|\boldsymbol{x}_k - \bar{\boldsymbol{v}}_i\|^2$

end for

for $i = 1, \ldots, c$ **do**

$\boldsymbol{v}_i = \frac{1}{|G_i|} \sum_{\boldsymbol{x}_k \in G_i} \boldsymbol{x}_k$

end for

until $\max_{1 \le i \le c} \|\boldsymbol{v}_i - \bar{\boldsymbol{v}}_i\|^2 < \epsilon$

Algorithm 4 Standard fuzzy c-means

Require: A set of objects $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^p$

Require: The number of clusters $c \in \mathbb{N}$

Require: A set of cluster center $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c) \in \mathbb{R}^p$

Require: A parameter $m \geq 1$

Require: A tolerance criterion $\epsilon \in \mathbb{R}^+$

repeat

$\bar{V} = V$

for $k = 1, \ldots, n$ **do**

for $i = 1, \ldots, c$ **do**

$u_{ik} = \frac{1}{Z_k} \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^{-\frac{2}{m-1}}$

end for

end for

for $i = 1, \ldots, c$ **do**

$\boldsymbol{v}_i = \sum_{k=1}^{n} (u_{ik})^m \boldsymbol{x}_k / \sum_{k=1}^{n} (u_{ik})^m$

end for

until $\max_{1 \leq i \leq c} \|\boldsymbol{v}_i - \bar{\boldsymbol{v}}_i\| < \epsilon$

Algorithm 5 Fuzzy c-means using the entropy regularization

Require: A set of objects $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^p$

Require: The number of clusters $c \in \mathbb{N}$

Require: A set of cluster center $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c) \in \mathbb{R}^p$

Require: A parameter $\lambda \in \mathbb{R}^+$

Require: A tolerance criterion $\epsilon \in \mathbb{R}^+$

repeat

$\bar{V} = V$

for $k = 1, \ldots, n$ **do**

for $i = 1, \ldots, c$ **do**

$u_{ik} = \frac{1}{Z_k} \exp\big(- \lambda \|\boldsymbol{x}_k - \bar{\boldsymbol{v}}_i\|^2 \big)$

end for

end for

for $i = 1, \ldots, c$ **do**

$\boldsymbol{v}_i = \sum_{k=1}^{n} u_{ik} \boldsymbol{x}_k / \sum_{k=1}^{n} u_{ik}$

end for

until $\max_{1 \leq i \leq c} \|\boldsymbol{v}_i - \bar{\boldsymbol{v}}_i\| < \epsilon$

Algorithm 6 Fuzzy c-means using the KL-divergence regularization

Require: A set of objects $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^p$

Require: The number of clusters $c \in \mathbb{N}$

Require: A membership matrix $(U)_{ik} = u_{ik}, u_{ik} \in [0, 1]$

Require: A parameter controlling regularization $\lambda \in \mathbb{R}^+$

Require: A tolerance criterion $\epsilon \in \mathbb{R}^+$

repeat

$\bar{U} = U$

for $i = 1, \ldots, c$ **do**

$\boldsymbol{v}_i = \sum_{k=1}^{n} \bar{u}_{ik} \boldsymbol{x}_k / \sum_{k=1}^{n} \bar{u}_{ik}$

end for

for $i = 1, \ldots, c$ **do**

$\alpha_i = \sum_{k=1}^{n} \bar{u}_{ik} / n$

end for

for $k = 1, \ldots, n$ **do**

for $i = 1, \ldots, c$ **do**

$u_{ik} = \frac{1}{Z_k} \alpha_i \exp\big(- \lambda \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 \big)$

end for

end for

until $\max_{i,k} |u_{ik} - \bar{u}_{ik}| < \epsilon$

Algorithm 7 Possibilistic clustering using the I-divergence regularization

Require: A set of objects $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^p$

Require: The number of clusters $c \in \mathbb{N}$

Require: A set of cluster center $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c) \in \mathbb{R}^p$

Require: A parameter controlling regularization $\lambda \in \mathbb{R}^+$

Require: A tolerance criterion $\epsilon \in \mathbb{R}^+$

repeat

$\bar{V} = V$

for $k = 1, \ldots, n$ **do**

for $i = 1, \ldots, c$ **do**

$u_{ik} = \exp\big(-\lambda \|\boldsymbol{x}_k - \bar{\boldsymbol{v}}_i\|^2\big)$

end for

end for

for $i = 1, \ldots, c$ **do**

$\boldsymbol{v}_i = \sum_{k=1}^{n} u_{ik}\boldsymbol{x}_k / \sum_{k=1}^{n} u_{ik}$

end for

until $\max_{1 \le i \le c} \|\boldsymbol{v}_i - \bar{\boldsymbol{v}}_i\| < \epsilon$

Algorithm 8 EM algorithm for a latent variable model

Require: A set of objects $X = (x_1, \ldots, x_n) \in \mathcal{X}$

Require: A latent variable model $p(x, z; \theta)$

Require: A initial parameter θ

Require: A tolerance criterion $\epsilon \in \mathbb{R}^+$

repeat

$\theta^t = \theta$

E step

$p(z|X, \theta^t) = \max \mathcal{F}(\theta^t, \theta)$

M step

$\theta = \max_{\theta} \mathcal{F}(\theta^t, \theta)$

until $|\theta - \theta^t| < \epsilon$

Algorithm 9 Kernel Hard c-means

Require: A kernel matrix $(K)_{ij} = k(x_i, x_j)$

Require: The number of clusters $c \in \mathbb{N}$

Require: Initial clusters $G = (G_1, \ldots, G_c)$

Require: A tolerance criterion $\epsilon \in \mathbb{R}^+$

repeat

$\bar{G} \Leftarrow G$

for $i = 1, \ldots, c$ **do**

for $k = 1, \ldots, n$ **do**

$$D_{ik} = K_{kk} - \frac{2}{|\bar{G}_i|} \sum_{x_j \in \bar{G}_i} K_{jk} + \frac{1}{|\bar{G}_i|^2} \sum_{x_j \in \bar{G}_i} \sum_{x_l \in \bar{G}_i} K_{jl}$$

end for

end for

for $k = 1, \ldots, n$ **do**

$$x_k \in G_h \Longleftrightarrow h = \arg \min_{1 \leq i \leq c} D_{ik}$$

end for

until $G = \bar{G}$

Algorithm 10 Kernel Fuzzy c-means using the entropy regularization

Require: A kernel matrix $(K)_{ij} = k(x_i, x_j)$

Require: The number of clusters $c \in \mathbb{N}$

Require: A membership matrix $(U)_{ik} = u_{ik} \in [0, 1]$

Require: A tolerance criterion $\epsilon \in \mathbb{R}^+$

repeat

$\bar{U} = U$

for $i = 1, \ldots, c$ **do**

for $k = 1, \ldots, n$ **do**

$D_{ik} = K_{kk} - \frac{2}{U_i} \sum_{j=1}^{n} u_{ij} K_{jk} + \frac{1}{U_i^2} \sum_{j=1}^{n} \sum_{l=1}^{n} u_{ij} u_{il} K_{jl}$

end for

end for

for $k = 1, \ldots, n$ **do**

for $i = 1, \ldots, c$ **do**

$u_{ik} = \frac{1}{Z_k} \exp\big(- \lambda D_{ik} \big)$

end for

end for

until $\max_{i,k} |u_{ik} - \bar{u}_{ik}| < \epsilon$

Algorithm 11 Kernel on-line competitive learning

Require: A kernel matrix $(K)_{ij} = k(x_i, x_j)$

Require: The number of clusters $c \in \mathbb{N}$

Require: Initial clusters $G = (G_1, \ldots, G_c)$

Require: An initial learning rate α^0

Require: A damping coefficient η

Require: The truncation number $\mathcal{T}$

  for $i = 1, \ldots, c$ **do**

    for $k = 1, \ldots, n$ **do**

      $D_{ik}^0 = K_{kk} - \frac{2}{|\bar{G}_i|} \sum_{x_j \in \bar{G}_i} K_{jk} + \frac{1}{|\bar{G}_i|^2} \sum_{x_j \in \bar{G}_i} \sum_{x_l \in \bar{G}_i} K_{jl}$

    end for

  end for

  $t = 0$

  while $t < \mathcal{T}$ **do**

    $x_l = x^t$

    $x_l \in G_h \Longleftrightarrow h = \arg \min_{1 \leq i \leq c} D_{il}$

    for $k = 1, \ldots, n$ **do**

      $D_{hk}^{t+1} = (1 - \alpha) D_{hk}^t - \alpha(1 - \alpha) D_{hl}^t + \alpha(K_{kk} - 2K_{kl} + K_{ll})$

    end for

    $\alpha = \alpha - \eta\alpha^0$

    $t = t + 1$

  end while

Algorithm 12 Kernel on-line dual competitive learning

Require: A kernel matrix $(K)_{ij} = k(x_i, x_j)$

Require: The number of clusters $c \in \mathbb{N}$

Require: A coefficient matrix $(B)_{ik} = \beta_{ik} \in \mathbb{R}$

Require: An initial learning rate α^0

Require: A damping coefficient η

Require: The truncation number $\mathcal{T}$

for $i = 1, \ldots, c$ **do**

for $k = 1, \ldots, n$ **do**

$D_{ik}^0 = K_{kk} - \frac{2}{U_i} \sum_{j=1}^{n} \beta_{ij} K_{jk} + \frac{1}{U_i^2} \sum_{j=1}^{n} \sum_{l=1}^{n} \beta_{ij} \beta_{il} K_{jl}$

end for

end for

$t = 0$

while $t < \mathcal{T}$ **do**

$x_l = x^t$

$x_l \in G_h \Longleftrightarrow h = \arg \min_{1 \leq i \leq c} D_{il}$

$\beta_{hl}^{t+1)} = (1 - \alpha)\beta_{hl}^t + \alpha$

for $k = 1, \ldots, n$ **do**

$\beta_{hk}^{t+1} = (1 - \alpha)\beta_{hk}^t, \ k \neq l$

$D_{hk}^{t+1} = (1 - \alpha)D_{hk}^t - \alpha(1 - \alpha)D_{hl}^t + \alpha(K_{kk} - 2K_{kl} + K_{ll})$

end for

$\alpha = \alpha - \eta\alpha^0$

$t = t + 1$

end while

List of Publications

1. Journal Papers
   (a) Ryo Inokuchi, Sadaaki Miyamoto,
   Fuzzy c-means algorithms using Kullback-Leibler divergence and Hellinger distance based on multinomial manifold,
   Journal of Advanced Computational Intelligence and Intelligent Informatics, (submitted).
   (b) Ryo Inokuchi, Sadaaki Miyamoto,
   Kernel methods for clustering: competitive learning and c-means,
   International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol.14, No.4, pp. 481–493, 2006.
   (c) Ryo Inokuchi, Sadaaki Miyamoto,
   LVQ clustering and SOM using a kernel function,
   Journal of the Japan Society for Fuzzy Theory and Intelligent Informatics, Vol.17, No.1, pp. 88–94, 2005. (in Japanese)

2. Conference Proceedings
   (a) Ryo Inokuchi, Sadaaki Miyamoto,
   Sparse possibilistic clustering with L1 regularization,
   Proc. of the 2007 IEEE International Conference on Granular Computing (*IEEE GrC 2007*), pp. 442–445, Nov. 2–4, 2007, Silicon Valley, USA.
   (b) Ryo Inokuchi, Sadaaki Miyamoto,
   c-means clustering on the multinomial manifold,
   Proc. of the 4th Modeling Decisions for Artificial Intelligence (*MDAI 2007*), Aug. 16–18, 2007, Kitakyushu, Japan, V. Torra et al. (Eds.), *Lecture Note in Artificial Intelligence*, LNAI 4617, pp. 261–268, 2007.
   (c) Ryo Inokuchi, Sadaaki Miyamoto,
   Nonparametric Fisher kernel using fuzzy clustering,
   Proc. of the 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (*KES 2006*), Oct. 9–11, 2006, Bournemouth,UK, B. Gabrys et al. (Eds.), *Lecture Notes in Computer Science*, LNCS 4252, pp. 78–85, 2006.
   (d) Ryo Inokuchi, T. Nakamura, Sadaaki Miyamoto,

Kernelized cluster validity measures and application to evaluation of different clustering algorithms,
Proc. of the 2006 IEEE World Congress on Computational Intelligence (*WCCI 2006*), pp. 763–769, Jul. 16–21, 2006, Vancouver, Canada.

(e) Ryo Inokuchi, Sadaaki Miyamoto,
Kernel competitive learning networks,
Proc. of the 2nd Modeling Decisions for Artificial Intelligence (*MDAI 2005*), Jul. 25–27, 2005, Tsukuba, Japan. (CD-ROM)

(f) Ryo Inokuchi, Sadaaki Miyamoto,
LVQ clustering and SOM using a kernel function,
Proc. of the 2004 IEEE International Conference on Fuzzy Systems (*FUZZ-IEEE 2004*), pp. 1497–1500, Jul. 25–29, 2004, Budapest, Hungary.

3. Award

(a) WCCI 2006 Session Best Presentation Award (2006.7)
"Kernelized cluster validity measures and application to evaluation of different clustering algorithms"