Data Compression by Volume Prototypes for Streaming Data

Kenji Tabata, Maiko Sato, Mineichi Kudo *

Division of Computer Science
Graduate School of Information Science and Technology
Hokkaido University, Sapporo 060-0814, JAPAN

Abstract

In these years, we often deal with an enormous amount of data in a large variety of pattern recognition tasks. Such data require a huge amount of memory space and computation time for processing. One of the approaches to cope with these problems is using *prototypes*. We propose *volume prototypes* as an extension of traditional *point prototypes*. A volume prototype is defined as a geometric configuration that represents some data points inside. A volume prototype is akin to a data point in the usage rather than a component of a mixture model. We show a one-pass algorithm to have such prototypes for data stream, along with an application for classification. An oblivion mechanism is also incorporated to adapt concept drift.

Key words: volume prototypes, one-pass algorithm, streaming data, concept drift

1 Introduction

In these years, we often deal with an enormous amount of data or a data stream in a large variety of pattern recognition tasks. Such data require a huge amount of memory space and computation time for processing. So we need compression techniques in which many data are represented by a small subset of them. Such representative data are referred to as *prototypes*.

In this paper, we propose *volume prototypes* for this goal and show an algorithm that runs in a linear order of data size. A volume prototype is defined as a geometric configuration that represents some data points in the inside of

* Corresponding author. Tel:+81-11-706-6852, Fax:+81-11-706-7393
Email address: mine@main.ist.hokudai.ac.jp (Mineichi Kudo).

itself. A volume prototype has a specific region which enables data condensation based on the local data distributions (Fig.1). Moreover, they are available as a replacement of point prototypes. We may consider several geometric configurations as candidates; for example, a hyper-sphere, a hyper-ellipsoid, a hyper-rectangle, and so on. Maybe the only one necessary condition is the convexitivity. In this paper, we deal with a hyper-ellipsoid prototype such as Fig.1(b). Concretely speaking, a hyper-ellipsoid prototype has a center $\boldsymbol{\mu}$, a covariance matrix Σ, a Mahalanobis radius r and the number n of inside samples. Such a prototype is denoted by $\mathbf{p} = (\boldsymbol{\mu}, \Sigma, r, n)$.

2 Related works

Prototype generation/selection has been widely studied in the literature. The condensed nearest neighbor [1] and the reduced nearest neighbor [2] are typical in the nearest neighbor searching and classification. In this paper, we concentrate on prototype approaches for a massive data or a data stream. Especially we focus on one-pass algorithms, because they are significantly effective for streaming data that can be accessed only once.

Such approaches for massive data are divided into four groups: prototype selection, density estimation by mixture models, clustering and classification. We will review them in this order.

In the literature [3–6], there have been some studies of prototype selection for a huge dataset. A synergy effect of combining the editing and condensing nearest neighbor techniques on a massive data is discussed in [3]. Mitra *et al.* proposed an algorithm for data condensation using SVMs [4]. In that algorithm, only a subset of data points crucial for classification is chosen. The problem of large memory requirement for training SVMs is resolved by adopting an active incremental learning algorithm. Kim and Oommen also gave a prototype reduction scheme (PRS) for a huge dataset [5]. In the algorithm, a dataset is divided into some blocks recursively, and each block size is suppressed to be small. Then any traditional PRS can be employed to process each block. Recently, Beringer and Hüllermeier [6] have proposed editing strategies for nearest neighbor classification. They developed an algorithm that can adapt to changes of the underlying data stream (concept drift/shift). All these methods are one-pass algorithms and provide some point prototypes.

Density estimation algorithms for a huge dataset are also found in the literature [7–10]. Zhang *et al.* extended a kernel method to achieve a fast density estimation for very large databases [7]. Arandjelović and Cipolla realized a real-time density estimation by incremental learning of GMM [8]. The incremental EM algorithm [9] attempts to reduce the computational cost needed

by EM algorithm. This is made by adopting partial E-steps. It divides a whole dataset into some blocks and performs a partial E-step on each block in a cyclic way. The lazy EM algorithm [10] takes the same strategy with the incremental EM algorithm. However, it performs a partial E-step only on a significant subset of data.

Clustering methods for a huge dataset are also seen in the literature [11–16]. Charikar *et al.* proposed a one-pass clustering algorithm in a streaming model [11]. It produces a constant factor approximation of the solution of the k-median problem in storage space $O(k\ poly \log n)$ for size n data. An application of k-median approach is seen in clustering and classification of protein sequences [12]. Bradley *et al.* showed an efficient clustering method for large databases [13]. Their method is based on classifying regions into three kinds of regions: the regions that must be maintained as they are, the regions that are compressive, and the regions that are discardable. This algorithm requires at most one scan of the database. The algorithm BIRCH [14] by Zhang *et al.* achieves an efficient clustering for a huge dataset by constructing a CF-tree that is a hierarchical summary of clusters. Data is scanned only once to construct a CF-tree. The algorithm FEKM [15] by Goswami *et al.* produces almost the same clusters as the original k-means algorithm. Their algorithm is basically one-pass, and obtains clusters close to the correct clusters of the k-means with a few number of additional scans. A similar idea is seen in [16].

Domingos and Hulten proposed a general framework that allows any learning algorithm to deal with a huge dataset, and showed applications to k-means clustering [17] and EM algorithm [18]. For classification, they also provide a very fast decision tree (VFDT) algorithm for data streams [19].

Some classification methods have been proposed for a huge dataset [20,21]. The algorithm SEA [20] proposed by Street and Kim classifies a sample by a majority vote of some classifiers. The component classifiers are constructed from some blocks of a data stream. Aggarwal *et al.* showed a classification model which is not only one-pass but also adaptable to concept drift [21].

The volume prototype approach is related to all of above four groups of studies, especially it is strongly related to clustering and mixture models (Table 1). However, there are clear differences from them. A major difference is that volume prototype approach is at a lower level than clustering or mixture models. That is, volume prototypes are regarded as one of intermediate expressions, more close to data, between data points and components as seen in clustering or mixture models. Indeed, volume prototypes are able to be used for clustering or for mixture modelling, instead of data points.

The name *volume prototype* is not new. Kaymak and Setnes [22] have already used the terminology, but their volume prototypes are clusters with a

boundary. This direction has been studied as cluster volumes or the volume of clusters [23,24]. On the contrary, our volume prototypes are closer to data themselves in number and in size as will be shown. Typically, the number of volume prototypes is more than the number of clusters and/or the number of components of mixture models and the sizes are smaller than clusters and/or components.

The difference from clustering comes from the fact that volume prototypes overlap each other. In clustering, usually clusters do not overlap each other, although some methods allow partitions with overlapping [25,26]. In addition, a set of volume prototypes does not always include every data point. This nature would be useful for efficient and robust compression. The latter means that some data points are removed as outliers.

The difference from mixture models is that a volume prototype has a clear boundary unlike a component of a mixture. It means that we can count the number of data points inside a volume prototype. Once a mixture model was constructed and then all the data points were discarded, we cannot know where data points were anymore. In contrast, in volume prototypes, we loose the concrete positions of data points, but can know its rough region. In mixture models, it is often difficult to determine the number of components appropriately, especially in streaming data. On the other hand, in volume prototype approaches, we do not need to be so careful the number of prototypes. It usually suffices to have a sufficient number of volume prototypes, because they converge to dense parts, modes, of the underlying distribution, as will be seen. In general, a larger number of prototypes are found compared to the number of components in a mixture model even if model selection is applied to the mixture model. This is mainly because the goals are different.

Comparing point prototypes and volume prototypes, it is difficult to judge which is better. The main difference comes from the fact that the former represents data points globally as a set, while the latter represents a variety of local data points. In other words, a point prototype keeps the first degree of statistics locally, while a volume prototype keeps the first and second degrees of statistics locally. The difference affects the compression rate. If the second degree of statistics reflects the local distribution faithfully, the volume prototypes would realize a higher compression rate, with a small number of prototypes, than that of point prototypes. Such an example is seen in the case that the underlying distribution consists of a small number of Gaussian modes.

For the goal of data compression, there are other types of methodology. Learning Vector Quantization (LVQ) [27] is widely used for data compression. It is close to clustering and the goal is to find some centers to which every sample is assigned to. Very recently, Lughofer [28] extended a vector quantization

technique for incremental clustering in which a quite similar technique to ours has been proposed in terms of (axis-parallel) ellipsoidal clusters. However, their clusters are close to traditional clusters and thus each cluster is larger than a volume prototype. One-class classification is another type of methodology [29,30]. It aims at finding only a (target) class distribution and is applied mainly to outlier or novelty detection. It is also useful to avoid unnecessary retraining of classifiers for when samples being regarded as outliers are obtained. However, one-class classification is not always appropriate for general purpose such as clustering and data compression.

3 Basic idea

In this paper, we assume that we have an almost infinite number of data, that is, a data stream. First, we show the basic strategy of updating the current prototypes.

(1) We deal with the samples one by one in the presented order and bring up the prototypes gradually.
(2) Only when a sample falls into the *acceptance region* (Fig.2) of a prototype, we update the prototype using the sample. One sample can be used for updating more than one prototype.

Here, the acceptance region is determined according to the statistics of data included in the prototype. We only use the samples within the acceptance region for updating. A volume prototype grows or reduces in the middle of the updating process. We expect that each prototype converges to a certain *final prototype* if the underlying data generation mechanism is stable.

The algorithm consists of two steps. The first step is the "mode estimation" step (shortly, ME step) and the second step is the "convergence" step (shortly, C step). The learning scheme is illustrated in Fig.3. In this scheme, the first ME step requires multiple scans over the first N samples, but the second C step requires only one-time scan for the remaining samples. As a result, this algorithm is regarded as a one-pass algorithm.

4 Algorithm

In this section, we show the concrete algorithm VP to generate volume prototypes. The algorithm is shown in Fig. 4.

The outline is as follows. With multiple scans over the first N samples in

different orders, we obtain M *initial prototypes*. Then choose L' *seed prototypes* using a set covering criterion (ME Step). We bring up those seed prototypes by updating them using the samples fell in their acceptance regions (C Step). When we want to have *final prototypes*, we carry out prototype selection using the same set covering criterion again to have L final prototypes.

4.1 Details

Dataset: We consider a data stream

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N, \boldsymbol{x}_{N+1}, \ldots$$

of D-dimensional vectors.

Prototype: Let $\boldsymbol{p} = (\boldsymbol{\mu}, \Sigma, r, n)$ be a prototype. Here, $\boldsymbol{\mu}$ is the prototype center, Σ is the covariance matrix, r is the Mahalanobis radius and n is the number of samples included in the prototype. Here, r is a significant parameter that affects the behavior of the prototype growth. For a random vector $\boldsymbol{X}$ generated according to a normal distribution $N(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma)$, it is known that the Mahalanobis (squared) distance $(\boldsymbol{X}-\boldsymbol{\mu})'\Sigma^{-1}(\boldsymbol{X}-\boldsymbol{\mu})$ obeys χ^2_D. Therefore, r^2 is set to $\chi^2_D(\theta)$ ($100 \times \theta$ % point of chi-squared statistics with D degrees of freedom). Here, θ reflects to what degree we require each prototype to cover a local area. If we take $\theta = 0.9$, we expect that 90% of surrounding points are covered.

Included sample set: Let $\boldsymbol{S}_p$ be the set of data points included in the prototype $\boldsymbol{p}$, which is specified by

$$\boldsymbol{S}_p = \{\boldsymbol{x} | (\boldsymbol{x}-\boldsymbol{\mu})'\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \leq r^2\}.$$

Then $n = |\boldsymbol{S}_p|$. Unfortunately we cannot count the exact number n because the acceptance region moves in the middle of process. Therefore we approximate n by the number of samples used for updating the prototype. That is, when a prototype is updated, we increment the value of n by one.

Acceptance region: We specify the acceptance region $\boldsymbol{A}_p$ of prototype $\boldsymbol{p} = (\boldsymbol{\mu}, \Sigma, r, n)$, by the acceptance Mahalanobis radius R as

$$\boldsymbol{A}_p = \{\boldsymbol{x} | (\boldsymbol{x}-\boldsymbol{\mu})'\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \leq R^2\}. \tag{1}$$

Here, $R(\geq r)$ is determined from r and n by

$$R = r + \sqrt{\frac{\chi^2_D(0.95)}{n}}. \tag{2}$$

Here, 0.95 means to take the 95% confidence region of the sample mean in Mahalanobis radius. It is noted that R approaches to r as n goes to infinity.

Updating procedure: When a sample $\boldsymbol{x}$ falls into the acceptance region of

prototype $\boldsymbol{p}_{t-1}$ at tth update, $\boldsymbol{p}_{t-1}$ is updated to $\boldsymbol{p}_t$ by

$$n_t = n_{t-1} + 1, \tag{3}$$

$$\boldsymbol{\mu}_t = \frac{(n_t - 1)\boldsymbol{\mu}_{t-1} + \boldsymbol{x}}{n_t}, \tag{4}$$

$$\Sigma_t = \frac{(n_t - 1)\Sigma_t + \boldsymbol{x}\boldsymbol{x}' + (n_t - 1)\boldsymbol{\mu}_{t-1}\boldsymbol{\mu}'_{t-1} - n_t\boldsymbol{\mu}_t\boldsymbol{\mu}'_t}{n_t}. \tag{5}$$

In addition, radius R of the acceptance region is also updated by (2).

Initialization of ME step: We initialize the center $\boldsymbol{\mu}_0$ and the covariance matrix Σ_0 by

$$\boldsymbol{\mu}_0 = \boldsymbol{x}_1, \tag{6}$$

and

$$\Sigma_0 = \lambda^2 \boldsymbol{I}. \tag{7}$$

Here,

$$\lambda^2 = \frac{1}{DN} \sum_{i=1}^{N} \min_{j \neq i, j = \{1,2,\ldots,N\}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$$

($\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ is the Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$) and $\boldsymbol{I}$ is the unit matrix. In addition, we set the initial value of n_0 to

$$n_0 = D + 1. \tag{8}$$

Seed prototype selection: After obtaining M initial prototypes, we select some of them by a set-covering criterion. In a greedy manner, we select the largest prototype in the number of samples first, then the prototype covering largest the samples that are not covered by the first prototype, and so on. This procedure is repeated until no prototype is found to increase the covering ratio. As a result, we have L' seed prototypes.

C step: In this step, we just bring up L' seed prototypes obtained in ME step with the remaining samples (Fig.3). Update is carried out by (3)–(5) using samples falling into their acceptance regions (1).

Choice of final prototypes: When one wants to have a set of volume prototypes, we choose some prototypes from the currently held L' prototypes. This is also made in a greedy manner in which selection is terminated when all N_v (latest) samples are covered or no improvement is found in covering. As a result, we obtain L final prototypes.

5 An example

We applied the proposed VP algorithm to a huge synthetic dataset to obtain volume prototypes. The dataset **Moon** comes from a 2-dimensional 2-class (ω_1 and ω_2) problem (Fig.5) which is used in [31]. Among 100 000 (=50 000

$\times$ 2 classes) points, the first 1 000 (500 per class) samples were used in ME step and the remaining 99 000 samples in C step. The algorithm was applied class by class.

The results in the individual steps are shown in Fig.5(a)-(c). With $M = 50$, we obtained 50 initial prototypes in each class, and then 70 (37 for ω_1 and 33 for ω_2) seed prototypes after prototype selection. To obtain the final prototypes, 49,500 samples in each class were used for updating the prototypes. After selection, we had 34 (18 for ω_1 and 16 for ω_2) final prototypes. Another example with 10^7 samples is shown in Fig.5(d).

In Fig.5, the decision boundaries of a locally-quadratic classifier, that will be given later, are shown in some cases. In the last stage, 100 000 samples were represented by 34 final prototypes. That is, $200000 (= 100000 \times 2)$ real values are reduced to $272 (= 34 \times 8)$ real values. The reduction rate is 0.14%. In Fig.5(d), 32 volume prototypes were obtained at reduction rate 0.0013%.

In Fig. 6, we can see how volume prototypes change for different values of θ. We have more but smaller volume prototypes for a smaller value of θ, while less but larger volume prototypes for a larger value of θ.

From Fig. 5(c),(d) and Fig. 6, we notice that

(1) the set of final prototypes approximates the distribution from inside when $\theta \leq 0.9$.
(2) each prototype is located on the ridge of the distribution.

6 Behavior analysis

Here, let us make clear the behavior of volume prototypes, especially where they move to, how fast they move and whether they shrink or not, as the number t of updatings increases.

The behavior of a volume prototype is very complicated. The center $\boldsymbol{\mu}_{t-1}$ and covariance matrix Σ_{t-1} are updated to $\boldsymbol{\mu}_t$ and Σ_t through (3) – (5). The factors $\boldsymbol{\mu}_{t-1}$, Σ_{t-1}, r, n_{t-1} affect the updating as well as $\boldsymbol{X}_t$ generated from the underlying distribution f. Here let us derive the typical behavior under some strong but generally acceptable assumptions.

First, let us analyze the direction and speed of movement of a volume prototype center. Let a prototype after tth updating be $\boldsymbol{p}_t = (\boldsymbol{\mu}_t, \Sigma_t, r_\theta, n_t)$. For simplicity, we assume that $\Sigma_t = \Sigma_0$ and $R^2 = r_\theta^2 = \chi_D^2(\theta)$ for $t = 1, 2, \ldots$ Here, $n_t = n_0 + t$. Then, the acceptance region $\boldsymbol{A}_{p_t}$ of $\boldsymbol{p}_t$ is given by

$$\boldsymbol{A}_{p_t} = \{\boldsymbol{x} \mid (\boldsymbol{x} - \boldsymbol{\mu}_t)'\Sigma_0^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_t) \leq r_\theta^2\}.$$

With a sample $\boldsymbol{X}_{t+1} \in \boldsymbol{A}_{p_t}$, the center $\boldsymbol{\mu}_t$ is updated to $\boldsymbol{\mu}_{t+1}$ by

$$\begin{aligned}\boldsymbol{\mu}_{t+1} &= \frac{n_0 + t}{n_0 + t + 1}\boldsymbol{\mu}_t + \frac{1}{n_0 + t + 1}\boldsymbol{X}_{t+1} \\ &= \boldsymbol{\mu}_t + \frac{\boldsymbol{X}_{t+1} - \boldsymbol{\mu}_t}{n_0 + t + 1} = \boldsymbol{\mu}_t + \frac{\Delta\boldsymbol{X}_{t+1}}{n_0 + t + 1}.\end{aligned}$$

Then the expected updated center is given by

$$E_{\boldsymbol{X}_{t+1} \in \boldsymbol{A}_{p_t}}\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \frac{E_{\boldsymbol{X}_{t+1}}\Delta\boldsymbol{X}_{t+1}}{n_0 + t + 1}.$$

If Σ_0 is small enough to allow f to be linearly approximated in $\boldsymbol{A}_{p_t}$ (Fig. 7) as $f(\boldsymbol{x}) = f(\boldsymbol{\mu}_t) + \nabla f(\boldsymbol{\mu}_t)'(\boldsymbol{x} - \boldsymbol{\mu}_t)$, we can have

$$E_{\boldsymbol{X}_{t+1}}\Delta\boldsymbol{X}_{t+1} = \frac{r_\theta^2}{D+2}\Sigma_0\frac{\nabla f(\boldsymbol{\mu}_t)}{f(\boldsymbol{\mu}_t)},$$

(for the proof, see Fukunaga [31](pp.534–535)). Therefore, we have

$$E_{\boldsymbol{X}_{t+1} \in \boldsymbol{A}_{p_t}}\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \frac{1}{n_0 + t + 1}\frac{r_\theta^2}{D+2}\Sigma_0\frac{\nabla f(\boldsymbol{\mu}_t)}{f(\boldsymbol{\mu}_t)}. \tag{9}$$

From Eq. (9), we see that 1) the direction of displacement is $\Sigma_0\nabla f(\boldsymbol{\mu}_t)$, 2) the speed $E\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}_t$ is proportional to $\frac{r_\theta^2}{D+2}$ and $\frac{\nabla f(\boldsymbol{\mu}_t)}{f(\boldsymbol{\mu}_t)}$, therefore, 3) the speed is low when $\nabla f(\boldsymbol{\mu}_t)$ is small or $f(\boldsymbol{\mu}_t)$ is large depending on the previous position $\boldsymbol{\mu}_t$, and 4) the speed reduces at ratio $1/(n_0 + t + 1)$ for the increase of t, hence, the accumulated amount is approximately $\log t$. Note that $r_\theta^2/(D+2)$ does not decrease linearly to $1/D$ because r_θ depends on both D and θ. Indeed, this value changes in a little strange way as shown in Fig. 8 ($\theta = 0.9$). In $\theta = 0.9$, up to around $D = 8$, the value increases, hence the movement speed of the center increases, but, after $D = 8$, the value decreases toward one, hence the speed decreases, as the dimensionality D increases.

We can obtain more intuition when $\Sigma_0 = \alpha\Sigma_*$ with a constant α (< 1), where Σ_* is the covariance matrix of the underlying normal distribution $f(\boldsymbol{x}) =$

$N(\boldsymbol{x}; \mu_*, \Sigma_*)$. Then, for Eq. (9), we have a simpler equation:

$$\begin{aligned} E_{\boldsymbol{X}_{t+1}}\boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t + \frac{1}{n_0+t+1}\frac{r_\theta^2}{D+2}\Sigma_0\frac{\nabla f(\boldsymbol{\mu}_t)}{f(\boldsymbol{\mu}_t)} \\ &= \boldsymbol{\mu}_t + \frac{1}{n_0+t+1}\frac{\alpha r_\theta^2}{D+2}\Sigma_*\Sigma_*^{-1}(\boldsymbol{\mu}_* - \boldsymbol{\mu}_t) \quad \text{(for Gaussian } f\text{)} \\ &= \boldsymbol{\mu}_t + \frac{1}{n_0+t+1}\frac{\alpha r_\theta^2}{D+2}(\boldsymbol{\mu}_* - \boldsymbol{\mu}_t) \end{aligned}$$

Therefore, the expected $\boldsymbol{\mu}_{t+1}$ is the point between the global mean $\boldsymbol{\mu}_*$ and $\boldsymbol{\mu}_t$ at ratio $\frac{1}{n_0+t+1}\frac{\alpha r_\theta^2}{D+2} : (1 - \frac{1}{n_0+t+1}\frac{\alpha r_\theta^2}{D+2})$ and climbs up to the steepest ascend direction.

Experiment 1

Data : $N(\mathbf{0}, I_D), D = 2, 4, 8, 16, 32, 64$
Sample size: Out of 10^5 samples, the first $N = 10^3$ samples were used in ME step.
The number of prototypes: $M = L = 10$ (No prototype selection)
Radius: $r^2 = {r_\theta}^2 = \chi_D^2(\theta)$, $\theta = 0.85, 0.90, 0.95$
Procedure: Just after ME step, we have $\boldsymbol{p}_0 = (\boldsymbol{\mu}_0, \Sigma_0, r, n_0)$, where n_0 depends on the prototype.
Results: The top row of Fig. 9. Here, the horizontal axis is the logarithm of the number of samples.

From Fig. 9 (the top row), we can confirm that the speed is linear to $\log t$ within a range in which $\frac{\nabla f(\boldsymbol{\mu}_t)}{f(\boldsymbol{\mu}_t)}$ is almost the same. The speed (the slope) increases up to $D = 8$ and then decreases for larger D. As θ increases, thus r_θ does so, they converge more quickly. These observations show the validity of our analysis.

As a result, the centers of volume prototypes move at speed of the reciprocal of the number of samples and converge to one of local peaks of the underlying distribution, and they will stay there (because ∇f is close to zero) or approach to a higher place (dense part) very slowly.

6.2 Convergence of covariance matrix

Next, let us analyze whether volume prototypes vanish or not with a sufficiently large number of samples.

For simplify, let us assume that $f(\boldsymbol{x}) = N(\boldsymbol{x}; \mathbf{0}, \Sigma_*)$ and a prototype $\boldsymbol{p}_t =$

$(\mathbf{0}, \Sigma_t, r_\theta, n_t)$ has already converged to the origin in the center. In addition, we assume that $\Sigma_t = \alpha_t \Sigma_*$, that is, the configuration is the same as Σ_* and only the scale is different by α_t.

Then the behavior of Σ_t is identical to that of α_t. Let $Y^2 = \boldsymbol{X}'\Sigma_*^{-1}\boldsymbol{X}$. Then Y^2 obeys χ^2_D and $EY^2 = D$. Then, the acceptance region $\boldsymbol{A}_{p_t} = \{\boldsymbol{x} | \boldsymbol{x}'\Sigma_t^{-1}\boldsymbol{x} \leq r_\theta^2\}$ is rewritten by $\boldsymbol{A}_{p_t} = \{\boldsymbol{x} | \boldsymbol{x}'\Sigma_*^{-1}\boldsymbol{x} \leq \alpha_t r_\theta^2\}$. With $f(y^2) = \chi^2_D(y^2)$, we have

$$\begin{aligned} E_{\boldsymbol{X}\in\boldsymbol{A}_{p_t}} Y^2 &= \int_0^{\alpha_t r_\theta^2} y^2 f(y^2) dy^2 \Big/ \int_0^{\alpha_t r_\theta^2} f(y^2) dy^2 \\ &= \frac{2\gamma\left(\frac{D}{2}+1, \alpha_t r_\theta^2\right)}{\gamma\left(\frac{D}{2}, \alpha_t r_\theta^2\right)}, \end{aligned} \tag{10}$$

where $\gamma(\cdot,\cdot)$ is the incomplete gamma function defined by $\gamma(a,x) = \int_0^x e^{-t} t^{a-1} dt$. Note that $E_{\boldsymbol{X}\in\boldsymbol{A}_{p_t}} Y^2 \to D$ as $r_\theta^2 \to \infty$.

From Eq. (10) and the fact that $E_{\boldsymbol{X}\in\boldsymbol{A}_{p_t}} Y^2 = \mathrm{tr}\left(\Sigma_*^{-1} E_{\boldsymbol{X}\in\boldsymbol{A}_{p_t}} \boldsymbol{X}\boldsymbol{X}'\right)$, we have

$$E_{\boldsymbol{X}\in\boldsymbol{A}_{p_t}} \boldsymbol{X}\boldsymbol{X}' = \frac{2\gamma\left(\frac{D}{2}+1, \alpha_t r_\theta^2\right)}{D\gamma\left(\frac{D}{2}, \alpha_t r_\theta^2\right)} \Sigma_*. \tag{11}$$

On the other hand, the covariance Σ_t is updated to Σ_{t+1} by $\boldsymbol{X}_{t+1}$ as

$$\Sigma_{t+1} = \frac{n_0 + t}{n_0 + t + 1}\Sigma_t + \frac{1}{n_0 + t + 1}\boldsymbol{X}_{t+1}\boldsymbol{X}'_{t+1}.$$

By taking the expectation over $\boldsymbol{X}_{t+1} \in \boldsymbol{A}_{p_t}$, from (11), we have

$$\begin{aligned} \alpha_{t+1}\Sigma_* = E_{\boldsymbol{X}_{t+1}\in\boldsymbol{A}_{p_t}} \Sigma_{t+1} &= \Sigma_t + \frac{E\boldsymbol{X}_{t+1}\boldsymbol{X}'_{t+1} - \Sigma_t}{n_0 + t + 1} \\ &= \alpha_t \Sigma_* + \frac{\frac{2\gamma\left(\frac{D}{2}+1, \alpha_t r_\theta^2\right)}{D\gamma\left(\frac{D}{2}, \alpha_t r_\theta^2\right)} - \alpha_t}{n_0 + t + 1}\Sigma_*. \end{aligned}$$

That is,

$$\alpha_{t+1} = \alpha_t + \frac{\frac{2\gamma\left(\frac{D}{2}+1, \alpha_t r_\theta^2\right)}{D\gamma\left(\frac{D}{2}, \alpha_t r_\theta^2\right)} - \alpha_t}{n_0 + t + 1},$$

$$\alpha_{t+1}D = \left(1 - \frac{1}{n_0 + t + 1}\right)\alpha_t D + \frac{1}{n_0 + t + 1}\frac{2\gamma\left(\frac{D}{2}+1, \alpha_t r^2\right)}{\gamma\left(\frac{D}{2}+1, \alpha_t r^2\right)}. \tag{12}$$

At time t, Σ_t and $\boldsymbol{A}_t$ correspond to $\alpha_t D$ and $\alpha_t r_\theta^2$, respectively (Fig. 10 and Fig. 11). Note that $r_\theta^2 > D$ for large θ, say $\theta = 0.9$. Then the contribution by $\boldsymbol{X}_{t+1}$ corresponds to $2\gamma\left(\frac{D}{2}+1, \alpha_t r^2\right)/\gamma\left(\frac{D}{2}, \alpha_t r_\theta^2\right)$. Thus a pair $(\alpha_t D, \alpha_t r_\theta^2)$ produces $2\gamma\left(\frac{D}{2}+1, \alpha_t r^2\right)/\gamma\left(\frac{D}{2}, \alpha_t r_\theta^2\right)$ (Fig. 11).

When $\alpha_t > 1$, since $2\gamma\left(\frac{D}{2}+1, \alpha r_\theta^2\right)/\gamma\left(\frac{D}{2}, \alpha r_\theta^2\right) < D < \alpha_t D < \alpha_t r_\theta^2$, we have $\alpha_{t+1}D < \alpha_t D$ from (12). On the other hand, when α_t is smaller than α^* defined below, we have $\alpha_t D < 2\gamma\left(\frac{D}{2}+1, \alpha_t r_\theta^2\right)/\gamma\left(\frac{D}{2}, \alpha_t r_\theta^2\right) < \alpha_t r_\theta^2$, then $\alpha_{t+1}D > \alpha_t D$. The latter situation is illustrated in Fig. 11.

To find the convergence value α^*, it suffices to solve the equation

$$\alpha D = \frac{2\gamma\left(\frac{D}{2}+1, \alpha r_\theta^2\right)}{\gamma\left(\frac{D}{2}, \alpha r_\theta^2\right)}.$$

Let us denote $\alpha^* = \alpha^*(D, \theta)$ by making clear the dependency. Several values of $\alpha^*(D, \theta)$ are shown in Table. 2. Here $r_\theta^2 = \chi_D^2(\theta)$. As a result, the covariance matrix Σ_t approaches to $\alpha^*(D, \theta)\Sigma_*$ as $t \to \infty$. In Table. 2, $\alpha^*(D, \theta) = 0$ for $D = 2$ and $\theta < 0.86$. However, when Σ_t becomes small, the probability that a sample falls in the acceptance region of the prototype becomes small too, hence, the number t of updatings decreases. As a result, such a small prototype survives for a long time. It should be noted that the volume ratio is $\alpha^*(D, \theta)^D$, so that, even when $\alpha^*(D, \theta)$ is close to one, it produces a large reduction in volume for large dimensionality D.

We conducted the same experiment as Experiment 1. The result is shown in Fig. 9. In Fig. 9 (the bottom row), we plotted the value of $\mathrm{tr}\Sigma_t/D$ as an estimate of α_t because $\mathrm{tr}\boldsymbol{I}_D = D$. From Fig. 9, we can see that α_t approaches to the theoretical equilibrium value $\alpha^*(D, \theta)$.

In total, we can say that volume prototypes converge to local peaks at a speed proportional to the reciprocal of the number of samples and then the covariance matrices converge to $\alpha^*(D, \theta)$ times the local covariance matrices.

7 Robustness and validity

The robustness of volume prototypes can be described in several ways. As for a single volume prototype, we cannot expect robustness both in the position (the mean) and in the configuration (the covariance matrix). This is because, each volume prototype strongly depends on the initial position that is determined by a random sample. Indeed, volume prototypes are substitutions of samples generated randomly. After obtaining many samples to grow, a volume prototype reaches at a local top of a hill. In this respect, some volume prototypes may reach at almost the same place if they start from a surrounding region of the point. In other words, as long as a sufficient number of samples were given, the final volume prototypes do not strongly depend on the initial prototypes. Indeed, in comparison between Fig. 5(c) with 10^5 samples and Fig. 5(d) with 10^7 samples, we see that the degree of affection of initial prototypes is smaller in the latter. Therefore, the number of initial prototypes is more crucial than individual positions of initial prototypes.

Robustness is found in a set of volume prototypes. A set of volume prototypes gives an inner approximation of the underlying distribution so that the set is robust against the outliers. This is a remarkable characteristic of volume prototypes. Such a characteristic is not seen in mixture models or in clustering. In other words, a set of volume prototypes approximates well the inner part of the underlying distribution, but looses the outer configuration. Such a characteristic would be useful to find the modes. Given a sufficient number of samples, some volume prototypes converge to one of the modes. Then we could find every mode by combining close prototypes into one.

8 Usage of volume prototypes for classification

When we use volume prototypes for designing classifiers, there are several ways:

(1) Use only the centers of volume prototypes as point prototypes. (Weighted point prototype strategy)
(2) Use the centers and covariances of volume prototypes. (Locally-quadratic strategy)
(3) Generate a necessary number of samples on the basis of volume prototypes. (Bootstrap strategy)

In the first strategy, we use only the centers with weights each of which is calculated from the number of samples included in the corresponding volume prototype. Then we can use a weighted nearest neighbor that is a popular tech-

nique and has been used up to now with several modifications (for example, see [32]).

In the second strategy, we use the second degree of statistics, that is, covariance Σ, of a volume prototype in addition to the center μ. A distance-weighted k-nearest neighbor [33] is also popular, in which the Euclidean distances of a query point and its k nearest neighbors are considered in addition to the majority class of them. Then it is natural to introduce the Mahalanobis distance using μ and Σ instead of the Euclidean distance.

In the last strategy, we consider a bootstrap data generator to generate a necessary number of samples for designing classifiers. Unlike above two strategies, such an approach is applicable to a wide range of classifiers. We can do it by sampling from the distribution modeled by the set of volume prototypes as a (detailed) mixture model. Unlike a likelihood-based mixture model, we do not need to determine the number of components nor to be careful about irregularly small components.

In the following, we describe the latter two strategies in detail.

8.1 Locally-quadratic classifier

In the second strategy, we propose a natural extension of a quadratic classifier. The classifier that we propose is illustrated in Fig.12 with some typical classifiers. This locally-quadratic classifier is regarded as an intermediate one between the mixture Bayes classifier and the nearest neighbor classifier. That is, this classifier is expected to provide an optimal complexity between the mixture models and the nearest neighbor classifier.

When each of class-conditional probability densities is given as a mixture model, we classify a class-unknown sample $\boldsymbol{x}$ to a class c^* by the rule

$$c^* = \arg\max_c \sum_{k=1}^{K_c} P(c)P(k|c)f(\boldsymbol{x}|\boldsymbol{\theta}_{c,k}),$$

where K_c is the number of components of class c, $f(\boldsymbol{x}|\boldsymbol{\theta}_{c,k})$ is the component density of the kth component of class c, $P(c)$ is the prior probability of class c, and $P(k|c)$ is the prior probability of the kth component of class c. The main problem of mixture models is that it is difficult to determine the number of components K_c and the parameters $\boldsymbol{\theta}_{c,k}$ appropriately, especially in a huge dataset.

To exploit volume prototypes $\boldsymbol{p} = (\boldsymbol{\mu}, \Sigma, n)$ for classification, we consider a normal density function $f(\boldsymbol{x}|\boldsymbol{\theta}) = N(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma)$ for each of them. Then, we can

regard each volume prototype as a component of a mixture model, but volume prototypes are closer to data points. So, instead of above rule, we use

$$c^* = \arg\max_c \max_k P(c)P(k|c)f(\boldsymbol{x}|\boldsymbol{\theta}_{c,k}),$$

where $f(\boldsymbol{x}|\boldsymbol{\theta}_{c,k})$ is calculated by the kth volume prototype of class c. This rule is a nearest neighbor rule using a probabilistic distance between $\boldsymbol{x}$ and the kth prototype of class c. On the other hand, this rule can be also seen as a quadratic classifier between prototypes of different two classes closest to the sample. In these senses, the classifier is between the nearest neighbor classifier and the quadratic classifier using global statistics. In practice, we build this rule with estimated $P(c)$'s and prototypes $\mathbf{p}(\boldsymbol{\mu}_{c,k}, \Sigma_{c,k}, n_{c,k})$'s by

$$f(\boldsymbol{x}|\boldsymbol{\theta}_{c,k}) = N(\boldsymbol{x}|\boldsymbol{\mu}_{c,k}, \Sigma_{c,k}),$$

and

$$P(c) = \frac{O_c}{O}, \quad P(k|c) = \frac{n_{c,k}}{N_v} \quad (N_v = \sum_{k=1}^{L_c} n_{c,k}).$$

Here, O is the total number of samples obtained so far, O_c is the number of samples of class c among O samples, and N_v is the number of last samples in each class that were kept in VP algorithm. Here, $n_{c,k}$ is modified to satisfy $\sum_k n_{c,k} = N_v$ by taking into account the degree of overlapping among volume prototypes. The value of k runs from one to L_c (the number of final prototypes in class c).

8.2 Bootstrap strategy

Here, we describe the third strategy. Regarding each volume prototype as a component of a mixture model, we generate samples according to $P(c)$, $P(k|c)$ and $N(\boldsymbol{x}|\boldsymbol{\mu}_{c,k}, \Sigma_{c,k})$ of kth prototype of class c, in this order. That is, the empirical distribution

$$\sum_c \sum_{k|c} P(c)P(k|c)N(\boldsymbol{x}|\boldsymbol{\mu}_{c,k}, \Sigma_{c,k})$$

is used for generating synthetic samples. Based on the latest N_v samples, we estimate these statistics and generate N' samples from this model. The generated samples seem to be close to the latest samples, but they are distributed according to the volume prototypes learned from all previous samples. As a result, they reflect as a whole the knowledge obtained from all previous samples. In general, the number of volume prototypes is more than that of components in a usual mixture model, so that the degree of approximation of the underlying distribution is expected to be higher than that of a mixture model. We call this strategy VP-P (transform from volume prototypes to point prototypes).

9 Experiments

9.1 Prototype generation

In this experiment, we used four kinds of 2-dimensional datasets:

(1) **Circle**: Two clusters. One cluster is concentrated on the origin and it is surrounded by the other cluster at a distance.
(2) **4-Cross**: Four Gaussians that perpendicularly cross at four corners.
(3) **5-Gaussian**: Five Gaussians. Some of them are closely located.
(4) **3-Gaussian with noise**: Three Gaussians with noise at mixture ratio 1%, 10% and 25%. The noise is generated according to the uniform distribution and is mixed at random.

In each dataset, 100 000 samples were generated according to a specified distribution. We found volume prototypes by VP algorithm with $M = 100$, $N = N_v = 1000$ and $\theta = 0.85, 0.90, 0.95$.

The found volume prototypes are shown in Fig. 13. We can see that the volume prototypes approximate the underlying distribution from inside and they are on the ridge. For $\theta = 0.85$, the specified number M of initial prototypes is clearly less than needed.

The robustness against noise is confirmed from Fig. 14. Up to 10% in noise ratio, the volume prototypes are stably converged into the three modes. In 25%, some volume prototypes exist ouside of the modes. Note that they are located away from the modes. This means that the other seed prototypes were all converged to the nodes, but they were not attracted because they were regarded as the samples from the uniform distribution. One might discard them by considering the density, that is, the ratio of the number of samples to the volume. It is also noted that the covariance matrices of the volume prototypes become larger according to the increase of noise.

9.2 Classification of two-dimensional synthetic data

We have compared volume prototypes with point prototypes in **Moon**. For generation of point prototypes, we used "very fast k-means clustering" (VFKM) [17]. We have used 10^5 samples (5×10^4 in each class) and obtained volume prototypes with $N = 500, M = 50, \theta = 0.9$. Then, using bootstrap strategy (VP-P), we generated 100 synthetic samples from the obtained volume prototypes. VFKM was applied to the same data set with $K = 100$ so that 100 cluster centers were obtained. In both methods, the process was carried class

by class.

The result is shown in Fig. 15. It is clear that VFKM finds spread and well-organized cluster centers and that VP-P generates many data in the dense area. In the sense of approximation, VFKM may be better than VP-P, but the samples generated by VP-P is robust against outliers. It is noted that some cluster centers obtained by VFKM are found inside the opposite class. Thus, for distribution modeling and/or for classification, VP-P may be better than VFKM. Indeed, when we used the generated samples by VP-P and those by VKFM, as the training samples, the recognition rates for another testing data of 200 000 points were 98.24% and 97.93%, respectively, by 1-NN method. When we use only centers of volume prototypes, we had 93.23%. By the locally-quadratic classifier with selected volume prototypes, we attained 98.21%.

Since VFKM needs more than linear time in the number of samples, VP with linear time is faster than it. We have confirmed it in a larger set of samples. With 10^7 samples (Fig. 5(d)), VFKM needed about 100 000 seconds in the same setting, while VP needed 320 seconds. Note that generation of points using volume prototypes is carried out by the same short time as in the above experiments.

9.3 Classification of high-dimensional real data

Next, to confirm the validity of VP algorithm for high-dimensional datasets, we used a multi-featured dataset of 10 handwritten numerals represented in terms of four different feature sets with $D = 47, 64, 76, 240$ [34]. In each class, the number of prototypes was set to $M = 10$, the first $N = 20$ samples were used for ME step and the remaining 180 samples were used for C step. Algorithm VP was invoked class by class. The radius was $r^2 = \chi^2_D(\theta)$ with $\theta = 0.9$. The latest sample number was $N_v = 180$ because no more sample was available. To estimate the classification rate of classifier, we used 10-fold cross validation.

The result is shown in Table 3 and Fig. 16. For comparison, the recognition rates by SVM with a RBF kernel and by VFDT (Very Fast Decision Tree) [19], in which the default parameter values were used, are also shown.

From Fig. 16, we see that 1) 10 prototypes are closely located but they represent well the spread of the class distribution (Fig. 16(a), (c)), 2) the sizes are a little smaller than that of the covariance matrix of all the samples of the class (two classes are chosen for visual interpretability in Fig. 16(b), (d)). Note that the prototype sizes look larger than their real sizes when they are projected on a 2–dimensional space, because the volume increases exponentially in dimensionality. As a whole, volume prototypes well represent the configuration

of dataset. The recognition rates of the locally-quadratic classifier are comparable to those of SVM and are better than those of VFDT. This means that the volume prototypes preserve the characteristics of the original data.

10 Adaptation to Population/Concept Drift

One of important properties expected for any algorithm that finds prototypes for streaming data is that it can adapt to the change of underlying distribution over time. It seems happen in many cases that the underlying distribution moves when much time passes. So, we expect that our algorithm can adapt to such a *population/concept drift*.

It is natural to introduce an *oblivion coefficient* $\alpha \geq 0$ so as to lessen the importance of a sample obtained one time-step before at ratio $e^{-\alpha}$. It is equivalent to assume, at current time t, a probability distribution on $\{1, 2, \ldots, t\}$:

$$\{e^{-\alpha(t-1)}/\beta(t), e^{-\alpha(t-2)}/\beta(t), \ldots, 1/\beta(t)\},$$

where $\beta(t) = \sum_{i=1}^{t} e^{-\alpha(t-i)}$. It is clear that

$$\beta(t) = \frac{1 - e^{-\alpha t}}{1 - e^{-\alpha}} \to \begin{cases} t & (\alpha \to 0) \\ 1 & (\alpha \to \infty) \end{cases} \quad \text{and} \quad \beta(t) \to \frac{1}{1 - e^{-\alpha}} \text{ as } t \to \infty \text{ for fixed } \alpha > 0.$$

That is, $\alpha = 0$ means "no forgetting," and $\alpha = \infty$ means "no memory" and uses only one (current) sample essentially. An intermediate value of α means that new samples are more important than old samples and the essential number $\beta(t)$ of samples is finite in the limit.

Let us assume that a volume prototype have included n samples at time $s_1, s_2, \ldots, s_n$ and the current time is t. Let $\beta_t = \sum_{i=1}^{n} e^{-\alpha(t-s_i)}$ and $\beta_t^F = \beta(t)$. Then some statistics of this volume prototype are defined as

$$p_t = \beta_t / \beta_t^F \qquad \text{(Probability of the volume prototype)} \tag{13}$$

$$\boldsymbol{\mu}_t = \frac{1}{\beta_t} \sum_{i=1}^{n} e^{-\alpha(t-s_i)} \boldsymbol{x}_{s_i} \qquad \text{(Mean)} \tag{14}$$

$$\Sigma_t = \frac{1}{\beta_t} \sum_{i=1}^{n} e^{-\alpha(t-s_i)} (\boldsymbol{x}_{s_i} - \boldsymbol{\mu}_t)(\boldsymbol{x}_{s_i} - \boldsymbol{\mu}_t)' \qquad \text{(Covariance)} \tag{15}$$

It should be noted that p_t can be seen as the degree of importance and it decreases as time t increases as long as n does not increase (no more sample is included in it).

Fortunately, we can derive the update procedure of above statistics. We show

this in two cases: at time t, (A) the case that sample $\boldsymbol{x}$ is added to the volume prototype as the nth sample for it and (B) the case that $\boldsymbol{x}$ is not added but n samples were already included. For case (A), we have

$$\beta_t = e^{-\alpha}\beta_{t-1} + 1, \tag{16}$$

$$\boldsymbol{\mu}_t = \frac{(\beta_t - 1)\boldsymbol{\mu}_{t-1} + \boldsymbol{x}}{\beta_t}, \tag{17}$$

$$\Sigma_t = \frac{(\beta_t - 1)\Sigma_{t-1} + \boldsymbol{x}\boldsymbol{x}' + (\beta_t - 1)\boldsymbol{\mu}_{t-1}\boldsymbol{\mu}'_{t-1} - \beta_t\boldsymbol{\mu}_t\boldsymbol{\mu}'_t}{\beta_t}. \tag{18}$$

For case (B), we keep the mean and covariance as $\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1}$ and $\Sigma_t = \Sigma_{t-1}$ and update only β_t by

$$\beta_t = e^{-\alpha}\beta_{t-1}. \tag{19}$$

We also use the initialize value

$$\beta_0 = \beta(n_0) = \frac{1 - e^{-\alpha n_0}}{1 - e^{-\alpha}}. \tag{20}$$

It is clear that equations (16)–(18) are identical with the original equations (3)–(5) when $\alpha = 0$, that is, the "remember forever" situation.

The covering criterion is also defined naturally. It suffices to replace the number n included in a volume prototype with the weighted sum β_t of samples included so far.

When the underlying distribution moves gradually with time, we need usually to add new prototypes and to delete old prototypes. However, in our algorithm, we do not need to do these explicitly. This is because volume prototypes can trace the moving distribution as long as the speed is not so high. In addition, if a prototype has included no sample after a certain time, then its probability $p_t = \beta_t/\beta_t^F$ decreases to zero as time goes. That is, old volume prototypes will be forgotten in nature.

Eventually, we do not have to do both in principle. The only concern is how to determine the value of oblivion coefficient α. It depends on how fast the distribution moves and to what degree we consider the latest samples important. Nevertheless, it could be possible to tune the value by examining how many latest samples have not been covered by the keeping volume prototypes.

In the case that we want to forcibly remove volume prototypes, we can remove volume prototypes that have a small value of p_t. However, instead of removal, it is more natural to select important prototypes by a selection procedure when we need to use the prototypes for some applications. Even if we leave all volume prototypes, for example, in a classification scheme, a volume prototype with a small value of p_t does not affect much to the construction of classifiers.

We conducted an experiment with a synthetic two-dimensional data. Figure 17 shows that the volume prototypes trace to the moving distribution. A single Gaussian moves and changes its covariance as time goes. In Figure 17, 50 seed volume prototypes grow and trace the change of the distribution and some of them are forgotten because of no recent sample included in them.

11 Discussion

In the proposed VP algorithm, it is a little questionable about how to determine the radius parameter θ of the acceptance region. From Fig. 6 and Fig. 13, we see that there is no big difference in different values of θ. In Fig. 13, in the case of $\theta = 0.85$, the specified number $M = 100$ of initial prototypes was smaller than needed. On the contrary, with $\theta = 0.95$, volume prototypes seem to be a little larger than desired. From these observations, we recommend $\theta = 0.9$ for $D = 2$. The balance of θ and M for high-dimensional datasets is necessary to be studied in the next phase.

Although many one-pass algorithms have been proposed for clustering and mixture models, but most of them need higher cost than VP algorithm in each round of updating. Indeed, although we do not describe the algorithms here, we have confirmed that EM algorithm using volume prototypes is fairly faster than incremental EM algorithm [9] and k-means using volume prototypes is comparable to SKM (scalable k-means) [16]. These results will be shown in another paper with the algorithms. One advantage of volume prototype approaches is that classification, clustering and mixture modelling can be made separately from the same set of volume prototypes, after processing a massive data.

Difference between volume prototypes and point prototypes is summarized as follows. 1) A set of volume prototypes gives a higher-compressed representation of data compared with a set of point prototypes in case that the underlying distribution is of a small number of modes, but the latter is advantageous to the former for finding a uniform representation of those data, 2) a set of volume prototypes gives an inner approximation of distribution and thus is robust against outliers, while a set of point prototypes gives an entire approximation and is affected by outliers, 3) the number of point prototypes is not so easy to be determined, but the number of volume prototypes is semi-automatically determined, and 4) the calculation cost of VP algorithm is cheaper than that of many clustering and vector quantization algorithms for data streams, mainly because no iteration procedure is included in VP algorithm.

It is noted that the calculation of Mahalanobis distance in VP algorithm re-

quires a matrix inversion in each round but we can exploit a sequential updating formula of the inverse matrix with only one inversion operation at the beginning (For example, see Appendix C in [35]).

12 Conclusion

In this paper, we have proposed volume prototypes for data compression. Volume prototypes are more informative than point prototypes because the former holds the second degree of statistics in addition to the first degree of statistics. We proposed a one-pass algorithm to have such prototypes and showed a locally-quadratic classifier using them. Volume prototypes give an inner approximation of the underlying distribution. Therefore, a set of volume prototypes are robust against outliers.

We have discussed the differences between this approach using the volume prototypes and the other approaches including point prototypes, mixture models and clustering. As a result, it was confirmed that such a volume-prototype approach is promising in data compression for a massive dataset or a data stream because of its high compression ability, the robustness and its wide applicability for classification and clustering.

For adapting to a gradual change in distribution (concept drift), we also have shown another version of the algorithm, in which volume prototypes with an oblivion coefficient trace the moving distribution as long as the amount of change is not so large. We need to improve the algorithm to cope with a sudden change in distribution (concept shift). Automatic adaptation to the speed of drift is also important. We are planning to write another paper for discussing on this isssue. In the paper, we will also consider some ways to realize semi-supervised learning [36] by volume prototypes, in which a limited number of labeled samples will be effectively used for labelling volume prototypes instead of all the remaining samples.

Acknowledgment

The authors would like to thank the anonymous reviewers for their critical comments to improve the quality of this paper.

References

[1] P.E. Hart (1968), The condensed nearest neighbor rule, IEEE Trans. Information Theory, IT-14(3):515–516.

[2] G.W. Gates (1972), The reduced nearest neighbor rule, IEEE Trans. Information Theory, IT-18(3):431–433.

[3] B.V. Dasarathy, J.S. Sánchez and S. Townsend (2000), Nearest Neighbour Editing and Condensing Tools – Synergy Exploitation. Pattern Analysis and Applications, 3(1):19–30.

[4] P. Mitra, C. A. Murthy, Sankar K. P (2000) Data Condensation in Large Databases by Incremental Learning with Support Vector Machines. Proceedings of the 15th International Conference on Pattern Recognition :708–711.

[5] S-W. Kim, B. J. Oommen (2004) Enhancing Prototype Reduction Schemes with Recursion: A Method Applicable for "Large" Data Sets. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 34(3):1384–1397.

[6] J. Beringer and E. Hüllermeier (2007): Efficient instance-based learning on data streams. Intelligent Data Analysis 11(6), 627–650.

[7] T. Zhang, R. Ramakrishnan, M. Livny (1999) Fast Density Estimation Using CF-kernel for Very Large Databases. Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining :312–316.

[8] O.Arandjelović, R. Cipolla (2005) Incremental Learning of Temporally-Coherent Gaussian Mixture Models. Proceedings of British Machine Vision Conference.

[9] R. M. Neal, G. E. Hinton (1998) A view of the EM algorithm that justifies incremental, sparse, and other variants. Learning in Graphical Models :355–371.

[10] B. Thiesson, C. Meek, D. Heckerman (2001) Accelerating EM for Large Databases. Machine Learning 45:279–299.

[11] M. Charikar, L. O'Callaghan, R. Panigrahy (2003) Better Streaming Algorithms for Clustering Problems. Proceedings of the 35th Annual ACM Symposium on the Theory of Computing :30–39.

[12] P.A. Vijaya, M.N. Murty and D.K. Subramanian (2006), Efficient median based clustering and classification techniques for protein sequences. Pattern Analysis and Applications, 9(2-3):243–255.

[13] P. S. Bradley, U. Fayyad, C. Reina (1998) Scaling Clustering Algorithms to Large Databases. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining :9–15.

[14] T. Zhang, R. Ramakrishnan, M. Livny (1996) BIRCH: An Efficient Data Clustering Method for Very Large Databases. Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Databases Systems.

[15] A. Goswami, R. Jin, G. Agrawal (2004) Fast and Exact Out-of-Core K-Means Clustering. Proceedings of the 4th IEEE International Conference on Data Mining:83–90.

[16] F. Farnstrom, J. Lewis and C. Elkan(2000) Scalability for Clustering Algorithms Revisited, ACM SIGKDD Explorations, 2(1):51–57.

[17] P. Domingos, G. Hulten (2001) A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering. Proceedings of the 18th International Conference on Machine Learning :106–113.

[18] P. Domingos, G. Hulten (2002) Learning from Infinite Data in Finite Time. Advances in Neural Information Processing Systems 14 :673–680.

[19] G. Hulten and P. Domingos (2003) A toolkit for mining high-speed time-changing data streams. http://www.cs.washington.edu/dm/vfml.

[20] W. N. Street, Y. Kim (2001) A Streaming Ensemble Algorithm(SEA) for Large-Scale Classification. The ACM International Conference on Knowledge Discovery and Data Mining.

[21] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, Philip S. Yu (2004) On Demand Classification of Data Streams. Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining :503–508.

[22] U. Kaymak and M. Setnes (2002), Fuzzy clustering with volume prototypes and adaptive cluster merging. IEEE Transactions on Fuzzy Systems, 10(6):705–712.

[23] X.L.Xie and G. Beni (1991), A validity measure for fuzzy clustering, IEEE Transactions on Pattern Anal. Machine Intell., 13:841–847.

[24] A. Keller and F. Klawonn (1999), Clustering with volume adaptation for rule learning, Proceedings of 7th Cong. Intelligent Techniques Soft Computing, Aachen.

[25] J. Bezdek, R. Hathaway (1988) Recent convergence results for the fuzzy c-means clustering algorithms. Journal of Classification 5(2):237–247.

[26] S. Nascimento, B. Mirkin, F. Moura-Pires (2000) A Fuzzy Clustering Model of Data and Fuzzy c-Means. The 9th IEEE International Conference on Fuzzy Systems, :302–307.

[27] T. Kohonen (1989): Self-Organization and Associative Memory (3rd ed.), Springer, Berlin.

[28] E. Lughofer (2008): Extensions of Vector Quantization for Incremental Clustering. Pattern Recognition 41(3), 995–1011.

[29] D. M. J. Tax and R. P. W. Duin (2004): Support Vector Data Description, Machine Learning, 54(1), 45–66.

[30] B. Schölkopf, et al. (2001): Estimating the Support of a High Dimensional Distribution. Nural Computation, 13(7), 1443–1471.

[31] K. Fukunaga (1990), Introduction to statistical pattern recognition (2nd edition), Academic Press.

[32] S. Cost and S. Salzberg (1993): A Weihted Nearest Neighbor Algorithm for Learning with Symbolic Features. Machine Learning 10, 57–78.

[33] S. A. Dudani (1976): The distance-weighted k-nearest-neighbor rule. IEEE Trans. Syst. Man Cybern., 6, 325–327.

[34] Asuncion, A., Newman, D.: UCI machine learning repository (2007)

[35] C. M. Bishop (2006), Pattern Recognition and Machine Learning, Springer.

[36] Xiaojin Zhu (2006), Semi-Supervised Learning Literature Survey. Technical report, Computer Science Department University of Wisconsin, Madison.

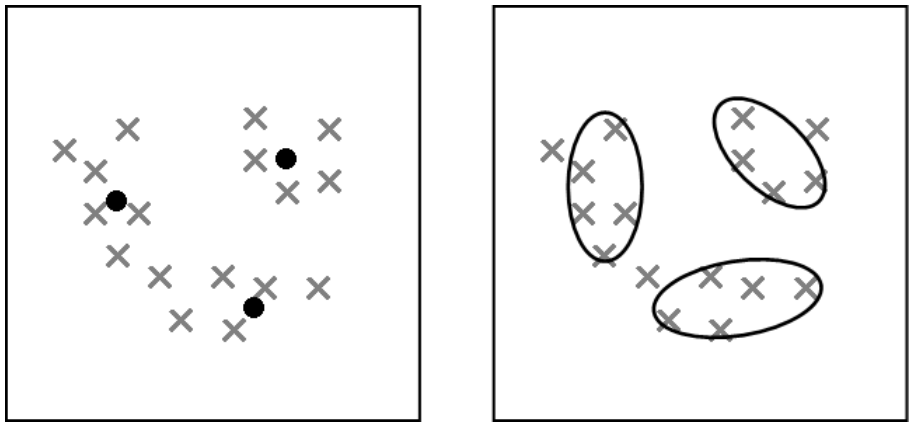


Fig. 1. An example of (a) *point prototypes* (•) and (b) *volume prototypes* (ellipsoids) for the same samples.

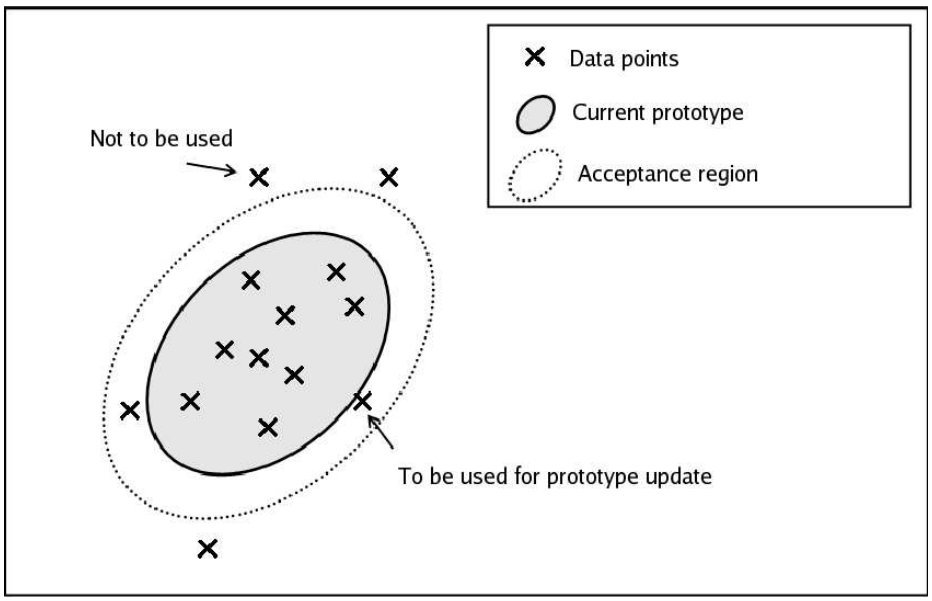


Fig. 2. An example of the *acceptance region* for prototype updating. Only samples falling into the acceptance region are used for updating.

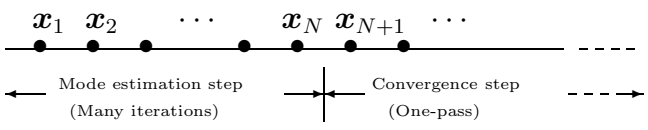


Fig. 3. Learning scheme

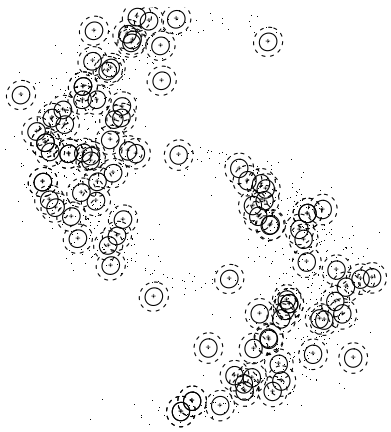
Inputs:(Unlimited samples) $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N, \ldots$
N is the number of samples for **ME Step**.
M is the number of initial prototypes.
N_v is the number of latest samples to be kept.
θ is the value to determine the acceptance radius $r = \chi^2_D(\theta)$.
Outputs: $L(\leq M)$ volume prototypes.

Procedure for obtaining volume prototypes
ME step:
Repeat M times the following for the first N samples.
1. Choose a random permutation σ to reorder the samples to $\boldsymbol{x}_{\sigma_1}, \boldsymbol{x}_{\sigma_2}, \ldots, \boldsymbol{x}_{\sigma_N}$.
2. Initialize a prototype by (6)–(8).
3. Use $\boldsymbol{x}_{\sigma_2}, \ldots, \boldsymbol{x}_{\sigma_N}$ in order to update the *initial prototype* by (3)–(5) (using only the samples in the acceptance region (1).

Select greedily some prototypes in a set covering criterion.
(We have $L'(\leq M \leq N!)$ *seed prototypes* at this stage.)
C step:
For each sample $\boldsymbol{x}_i (i = N+1, N+2, \ldots)$ do the following.
1. Update the seed prototypes in which $\boldsymbol{x}_i$ falls by (3)–(5).
2. Keep the latest N_v samples $\boldsymbol{x}_{i-N_v+1}, \boldsymbol{x}_{i-N_v+2}, \ldots, \boldsymbol{x}_i$.

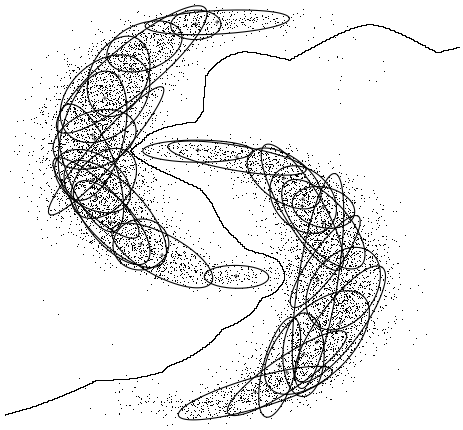
Exploit time:
From L' prototypes, select greedily $L(\leq L')$ prototypes until no improvement is found for covering N_v samples.
Count the number of samples falling in each prototype in N_v samples.
Here a sample is divided by k when k prototypes share it.

Fig. 4. Volume Prototype Algorithm (VP)

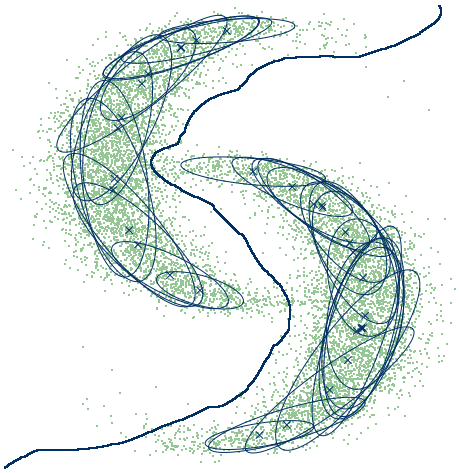


(a) Initial prototypes with 1000 samples

(b) Selected seed prototypes

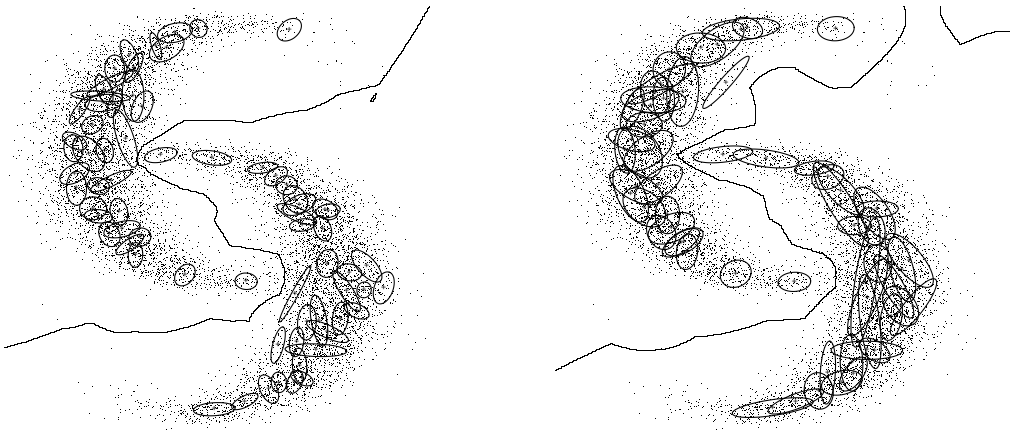


(c) Selected prototypes (10^5 samples)

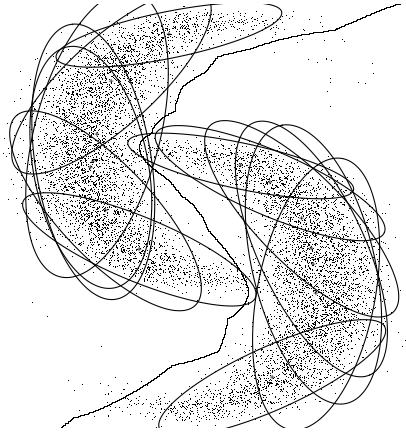


(d) Selected prototypes (10^7 samples)

Fig. 5. Generation process of volume prototypes ($N = 500, M = 50, \theta = 0.9$ in each class). The solid lines show the boundaries of prototypes and the dotted lines show their acceptance regions. (a) 100 (50/class) initial prototypes with a set of first $1,000$(= 500/class $\times$ 2 class) samples. (b) 70 (37 and 33 for two classes) seed prototypes after prototype selection with the decision boundary by a locally-quadratic classifier. (c) 34 (18 and 16 for two classes) volume prototypes (after prototype selection) generated 100,000 samples (50,000/class) with the decision boundary. (d) selected volume prototypes with another set of 10^7 samples (5×10^6/class)



(a) $\theta = 0.8$ (b) $\theta = 0.85$



(c) $\theta = 0.95$

Fig. 6. Final volume prototypes ($N = 500, M = 50$ in each class). Only selected final prototypes with 10^5 samples are shown.

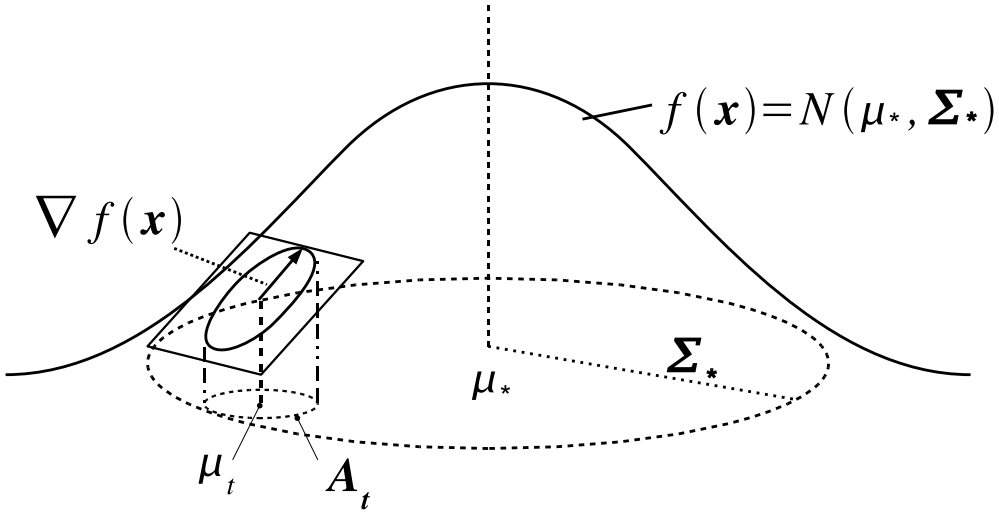


Fig. 7. Movement of a prototype center

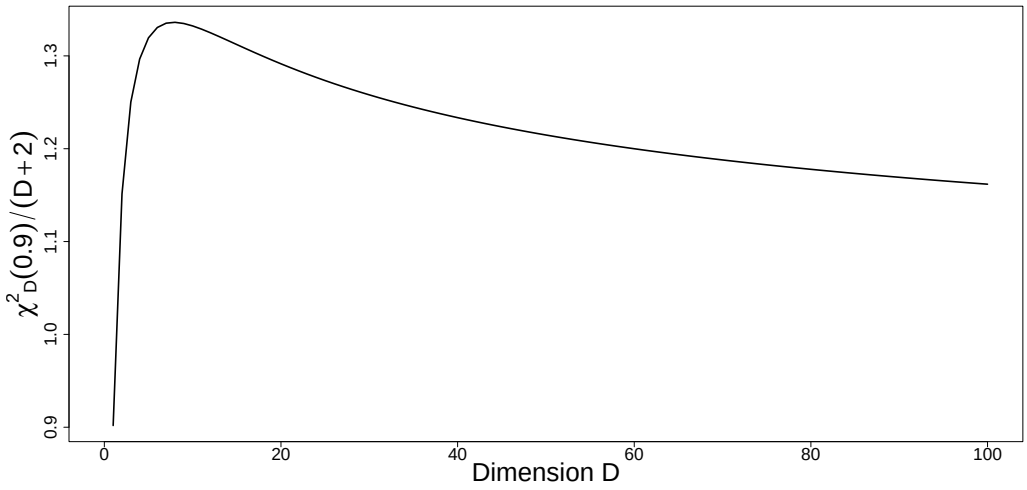


Fig. 8. $\chi^2_D(0.9)/(D+2)$ vs. dimensions

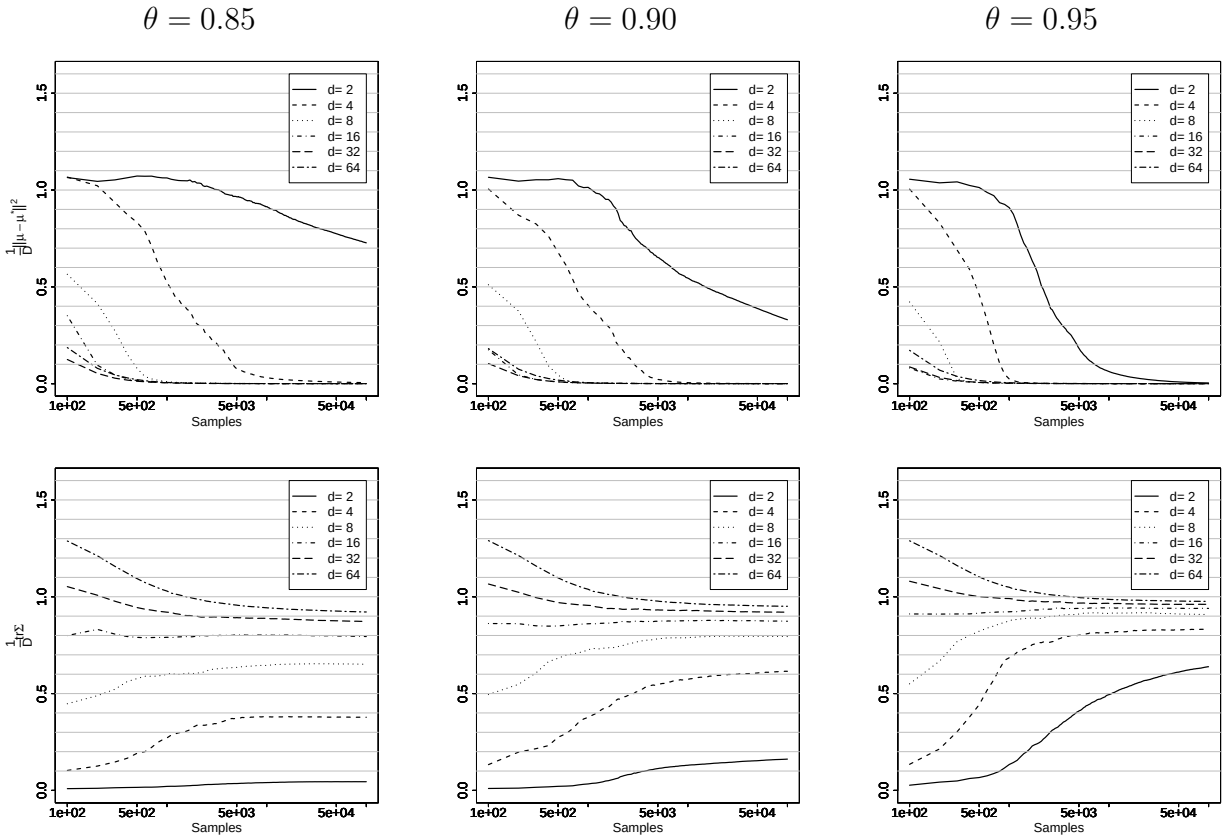


Fig. 9. The behavior of prototypes for $f(\boldsymbol{x}) = N(\boldsymbol{x}; \mathbf{0}, \boldsymbol{I}_D)$. The distance $\|\boldsymbol{\mu}_t\|^2/D$ (the top row) and $\mathrm{tr}\Sigma_t/D$ (the bottom row) are plotted for increasing number of sample in log-scale in $D = 2, 4, 8, 16, 32, 64$. From left to right, $\theta = 0.85, 0.90, 0.95$.

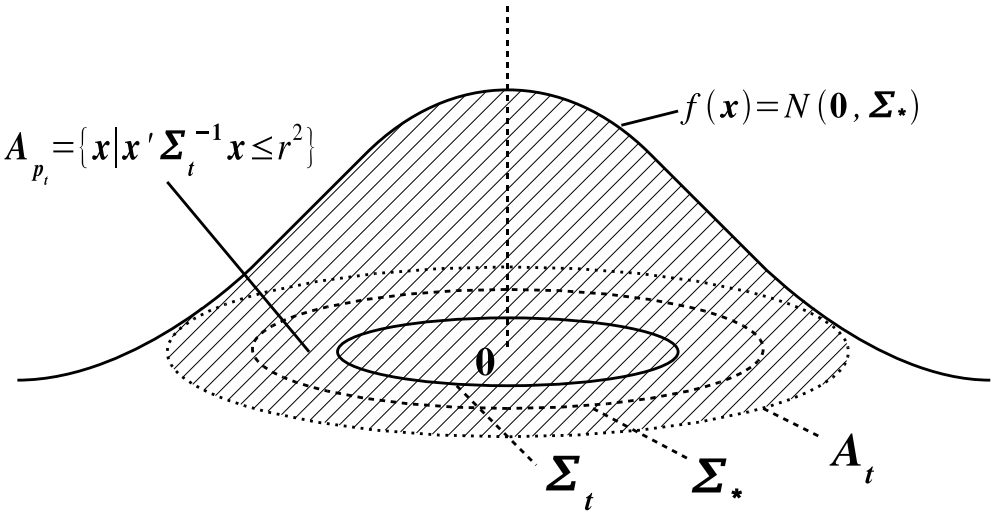


Fig. 10. Acceptance region of prototype $\boldsymbol{p}_t$.

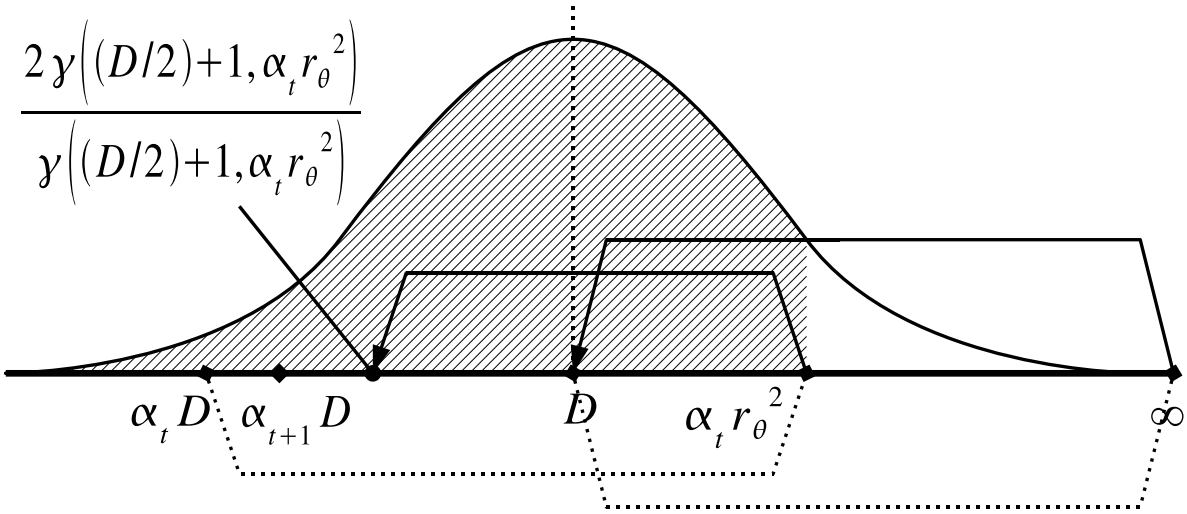
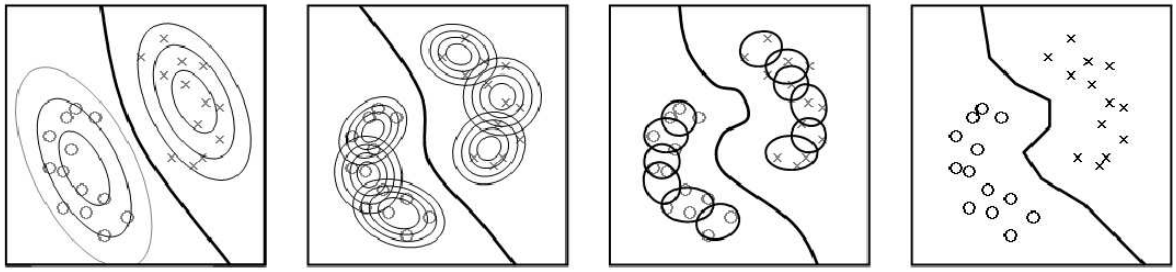


Fig. 11. Update scheme of covariance matrix.



Quadratic Mixture Bayes Locally-quadratic Nearest neighbor

Fig. 12. Comparison of classifiers (locally-quadratic classifier is the proposed one). From left to right, they become more complex in terms of the number of components.

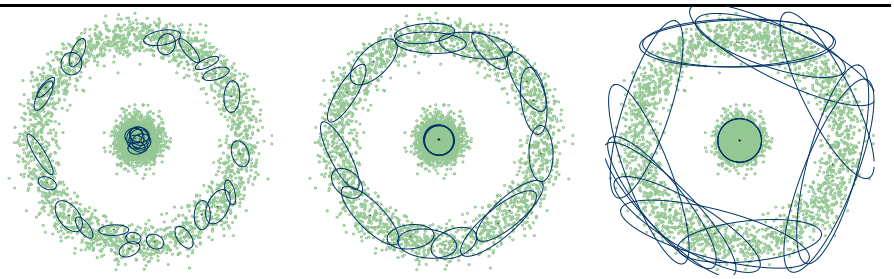
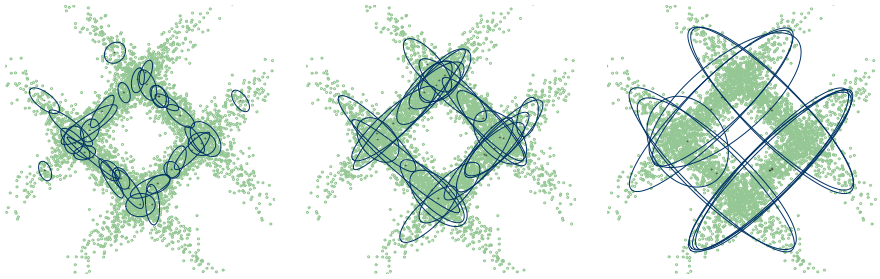
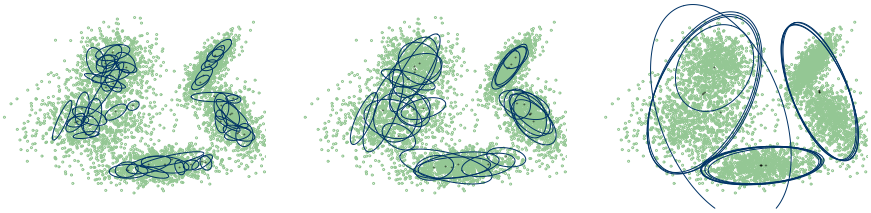
Dataset	Radius parameter θ ($r = \chi^2_D(\theta)$)		
	0.85	0.90	0.95
Circle			
4-Cross			
5-Gaussian			

Fig. 13. Volume prototypes in three datasets with 10^5 samples. In each dataset, 100 initial volume prototypes are generated and some of lastly generated volume prototypes are selected in a set-cover criterion ($M = 100, N = 1000, \theta = 0.85, 0.90, 0.95$).

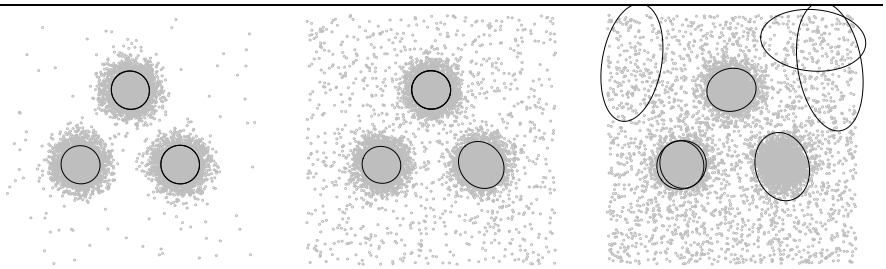
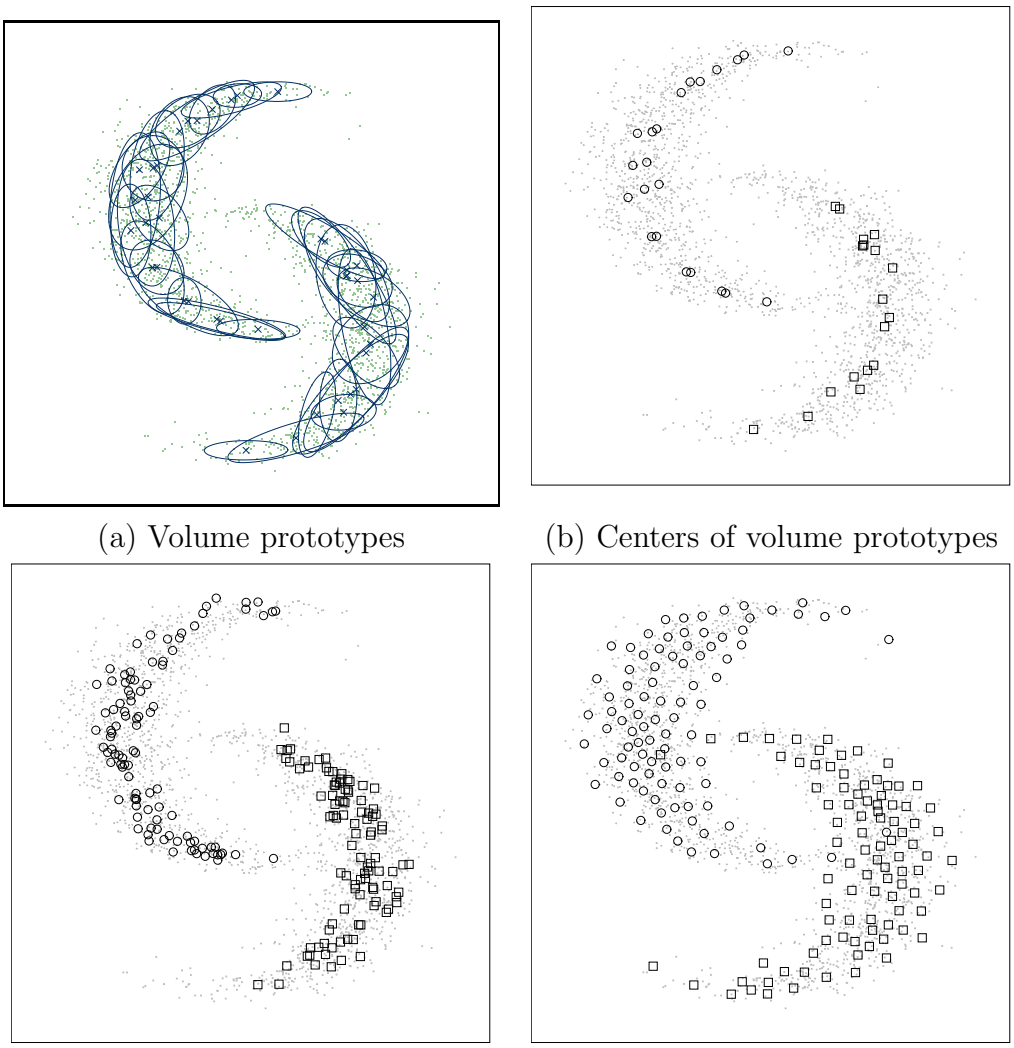
Dataset	Noise ratio		
	1%	10%	25%
3-Gaussian			

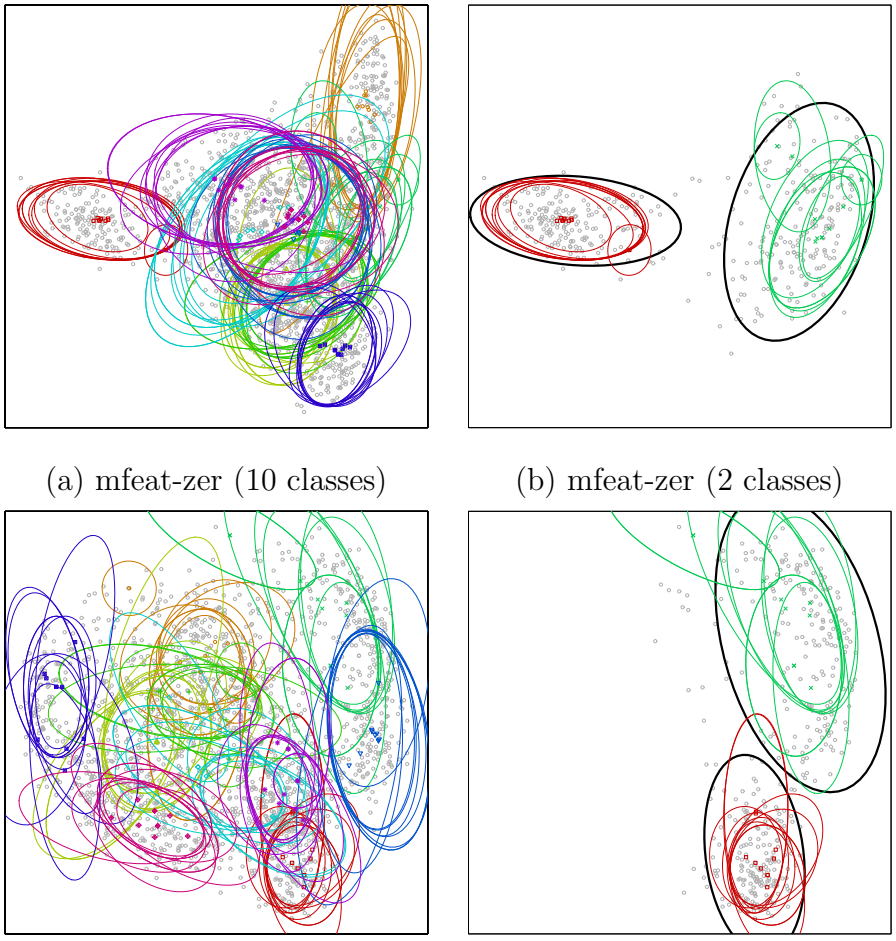
Fig. 14. Volume prototypes in noise environment. In 10^5 samples, noise was mixed randomly at ratio of 1%, 10% and 25%. The parameters of VP algorithm are $M = 100, N = 1000, \theta = 0.90$.



(a) Volume prototypes (b) Centers of volume prototypes

(c) VP-P (d) VFKM

Fig. 15. Comparison between VFKM and VP-P. In (c) 100 synthetic data are generated from volume prototypes by VP-P and; in (d) 100 cluster centers were obtained by VFKM. Small dots are latest 1000 samples in each class.



(a) mfeat-zer (10 classes)

(b) mfeat-zer (2 classes)

(c) mfeat-pix (10 classes)

(d) mfeat-pix (2 classes)

Fig. 16. Volume prototypes in each dataset (10 classes). The number of volume prototypes is 10 in each class. The radius is $r^2 = \chi^2_D(0.9)$. The groups of various marks are the centers of prototypes and the group of ellipsoids with a same color is the volume prototypes of the class. The covariance matrix of data points is also shown as a black bold ellipsoid in (b) and (d) for chosen two classes.

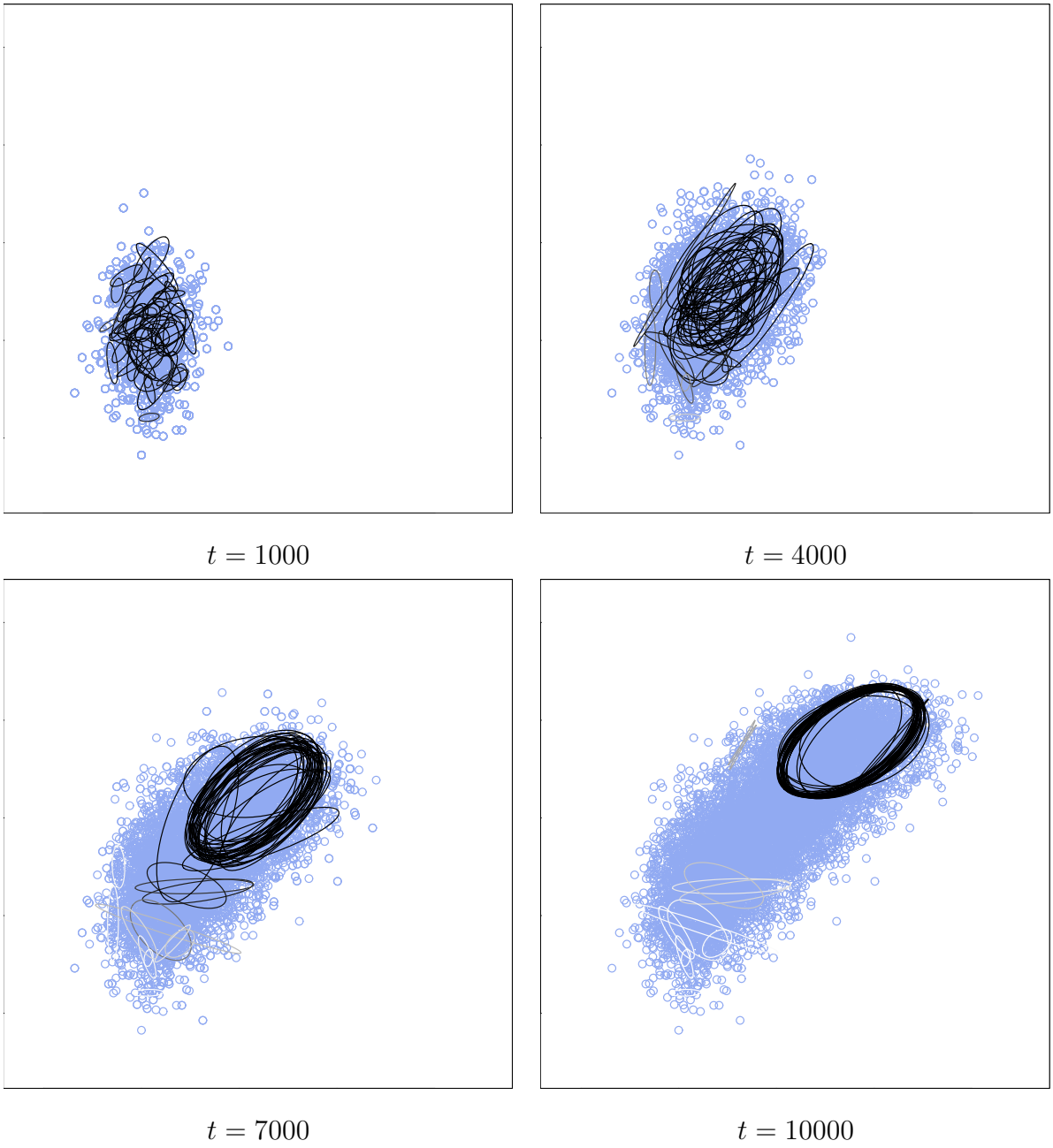


Fig. 17. Volume prototypes with an oblivion coefficient α. The underlying distribution is a single Gaussian in which the mean moves and the covariance changes gradually as time goes. The probability (importance) of a prototype is shown by the grey level of the corresponding ellipsoid (white: low and black: high). The number of prototypes is 50. The used parameter values are $\alpha = 0.001$, $N = 1000$, $N_{\text{total}} = 10000$, $r = 0.9$. Thus $\beta(N_{\text{total}}) \simeq 1000$. No selection of prototypes is performed.

Table 1
Some differences among volume prototypes, clustering and mixture models.

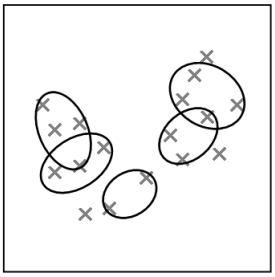
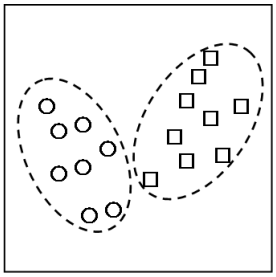
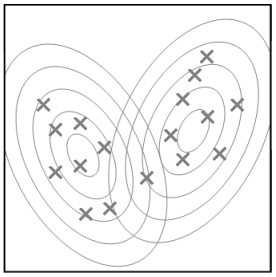
	Volume prototypes	Clustering	Mixture models
Goal	To condense (large) data appropriately	To analyze relative locations and distributions of clusters	To estimate mechanisms of data generation
Definition	Local prototypes with a volume	Partition of data without overlapping	Sum of component distributions to estimate original distributions
Example			

Table 2
Equilibrium value α^* of $\alpha = 2\gamma\left(\frac{D}{2}+1, \frac{\alpha r_\theta^2}{2}\right)/D\gamma\left(\frac{D}{2}, \frac{\alpha r_\theta^2}{2}\right)$.

θ	D					
	2	4	8	16	32	64
0.85	0.000 [a]	0.367	0.628	0.769	0.850	0.901
0.90	0.346	0.634	0.781	0.862	0.911	0.940
0.95	0.715	0.835	0.899	0.936	0.958	0.972

[a] For $\theta < 0.86$ and $D = 2$, the value becomes zero.

Table 3
Recognition rates of the locally-quadratic classifier using volume prototypes generated by VP ($\theta = 0.90$), the support vector machine with a RBF kernel, and VFDT in each feature set of "mfeat" dataset. The figures in parentheses show the dimensionality.

Classifier	Recognition rate (%)			
	mfeat-zer	mfeat-kar	mfeat-fou	mfeat-pix
	(47)	(64)	(76)	(240)
Locally-Quad.	83.5	82.5	97.3	97.9
SVM with RBF	80.9	82.0	97.5	97.9
VFDT	— [a]	82.5	76.7	88.8

[a] Appopriate result was not obtained.