Pattern Recognition Letters 37 (2014) 85–93

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Context-sensitive intra-class clustering

Yingwei Yu [a,*], Ricardo Gutierrez-Osuna [b], Yoonsuck Choe [b]

[a] IHS Global, Inc., 8584 Katy Freeway, Suite 400, Houston, TX 77024, USA
[b] Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843-3112, USA

ARTICLE INFO

Article history:
Available online 10 May 2013

Keywords:
Clustering
Intra-class clustering
ICC
Linear discriminant analysis
LDA
Semi-supervised learning

ABSTRACT

This paper describes a new semi-supervised learning algorithm for intra-class clustering (ICC). ICC partitions each class into sub-classes in order to minimize overlap across clusters from different classes. This is achieved by allowing partitioning of a certain class to be assisted by data points from other classes in a context-dependent fashion. The result is that overlap across sub-classes (both within- and across class) is greatly reduced. ICC is particularly useful when combined with algorithms that assume that each class has a unimodal Gaussian distribution (e.g., Linear Discriminant Analysis (LDA), quadratic classifiers), an assumption that is not always true in many real-world situations. ICC can help partition non-Gaussian, multimodal distributions to overcome such a problem. In this sense, ICC works as a preprocessor. Experiments with our ICC algorithm on synthetic data sets and real-world data sets indicated that it can significantly improve the performance of LDA and quadratic classifiers. We expect our approach to be applicable to a broader class of pattern recognition problems where class-conditional densities are significantly non-Gaussian or multi-modal.

1. Introduction

Many existing clustering methods based on unsupervised learning can partition data into clusters using the distribution of data points and the relative distance between the data points. A small amount of class information can help refine such clustering results by providing contextual information in a semi-supervised manner. In such a semi-supervised clustering situation, not only the distance among data points within the object class (e.g., targets to be recognized), but also the position of the context class (e.g., backgrounds or confounders) relative to the object class becomes important. As an example, the context class can serve as a boundary within the object class, and therefore influence how to best subdivide the object class.

An example of how a context class can affect the partitioning is shown in Fig. 1. Data points for the object class marked "+" were generated from a single Gaussian distribution (see Fig. 1 caption for details). Without the context data the distribution is best grouped into a single cluster as shown by the ellipse in Fig. 1(a). However, in the presence of intervening context data (marked "∘"), a different strategy is needed (Fig. 1(b)). Namely, when the context class is present, the object class can be split into two clusters: one to the lower left and the other to the upper right. The intra-class clustering algorithm in this paper is designed specifically to account for the presence of such intervening context classes.

Unlike other clustering algorithms that partition data into clusters exclusively based on intrinsic information (Jain, 2010), our context-sensitive clustering method operates in a *semi-supervised* mode by utilizing external information from context classes in addition to intrinsic information from the object class. As defined by Chapelle et al. (2006), "semi-supervised" clustering algorithms make use of external information, or "side-information". This semi-supervised learning method of context-sensitive clustering can lead to improvements in classification performance, as we will see shortly.

As an example, Fisher's linear discriminant analysis (LDA) (Duda et al., 2001) can find an optimal projection to discriminate data from different classes under the assumption that each class is a unimodal Gaussian with the same covariance matrix. However, LDA can significantly underperform in two situations (Zhao, 2000): (1) when the class discriminant information is in the variance of the data set, not just in the mean (Wang et al., 2004) or (2) when the class data is markedly non-Gaussian. For the first problem, one approach is to use the combined distribution of all other classes to split the object class in a context-sensitive manner. For the second problem we can require that the intra-class clustering method generates unimodal sub-clusters that are Gaussian.

Expectation Maximization (EM) (Dempster et al., 1977) based algorithms can divide non-Gaussian data into unimodal clusters, but EM is not context-sensitive, so the resulting clusters may have overlapping distributions. This problem is related to the first

* Corresponding author. Tel.: +1 281 844 4955.
E-mail addresses: yingweiy@gmail.com (Y. Yu), rgutier@cs.tamu.edu (R. Gutierrez-Osuna), choe@cs.tamu.edu (Y. Choe).

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2013.04.031

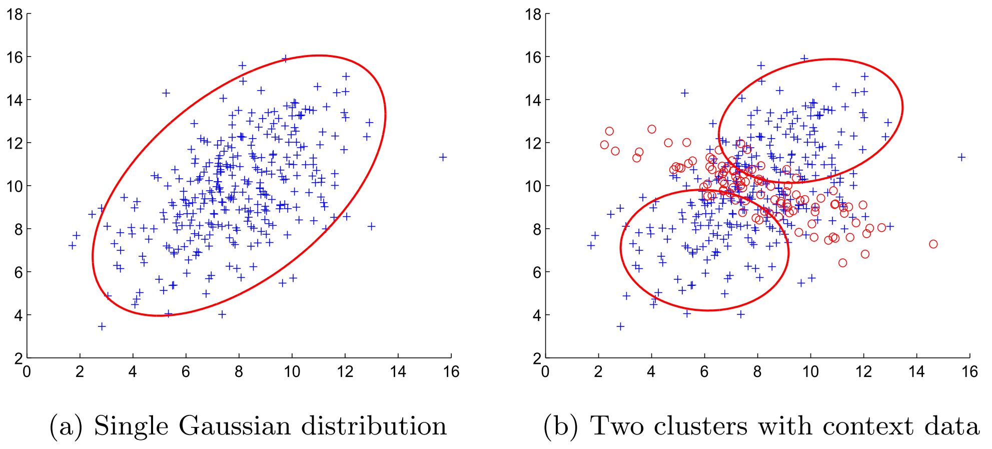


Fig. 1. Illustration of how object class and context class can interact. (a) The data points of the object class (marked "+") in the plot can be best modeled by a single Gaussian (mean at $(8, 10)$, with covariance $\begin{bmatrix} 10 & 30 \\ 30 & 30 \end{bmatrix}$) when no context data are present. (b) Data in the object class (marked "+") are those in (a), but whenthe context data are present (marked "$\circ$") with mean at $(8, 10)$, and covariance $\begin{bmatrix} 6 & -3 \\ -3 & 2 \end{bmatrix}$, the object class splits into two sub-clusters.

problem of LDA we discussed earlier. For example, the illustration in Fig. 2(a) shows data points from two Gaussian distributions (see figure caption for details). When EM is applied blindly, three of the resulting clusters significantly overlap. A better approach would be one that minimizes overlap among the clusters (see Fig. 2(b)).

To address these issues, this paper proposes an intra-class clustering algorithm that can generate non-overlapping unimodal clusters (e.g., as those in Fig. 2(b)). To achieve this, we propose a new semi-supervised learning algorithm called context-sensitive intra-class clustering algorithm (ICC). ICC can be used in unsupervised mode to cluster data as shown in Fig. 2(b) when there is no context class. Or, in semi-supervised mode, it can identify unimodal sub-clusters that reduce overlap within and across classes. The separation of data into single unimodal clusters makes the resulting distribution suitable for LDA as well as for low-cost Gaussian classifiers (e.g., the quadratic classifier). Our context-sensitive clustering algorithm is novel compared to other clustering algorithms in that it can not only cluster the object data based on the distance between samples, but also take into account the intervening nature of the context class.

The rest of the paper is organized as follows. In Section 2, we describe in detail our intra-class clustering algorithm. Section 3 presents experimental results of the proposed method on synthetic data and real-world data. Finally, Section 4 presents a brief discussion of our algorithm, followed by the conclusion.

2. Proposed algorithm: intra-class clustering

In most pattern recognition problems, the input space is very high dimensional which leads to serious problems due to curse of dimensionality as well as high computational cost. These issues can in part be overcome by applying dimensionality reduction algorithms. However, low-dimensional projections of the data are not guaranteed to minimize the overlap among different classes. If such overlap in low dimensional mapping can be alleviated by breaking down the classes into non-overlapping sub-classes, a highly efficient and accurate algorithm can be derived.

Our method starts by projecting the data in 2D space (or in 3D). Next, from these projections, density maps are generated for both the object class and the context class. Finally, the difference of the densities is calculated. A number of dimensionality reduction

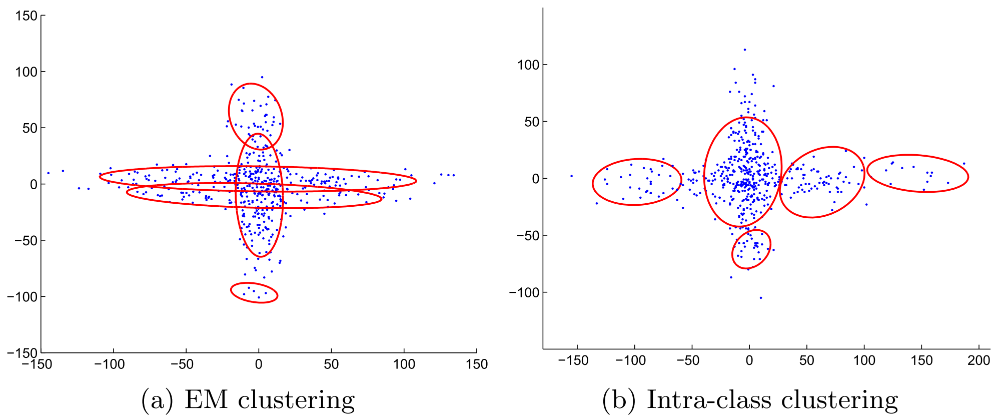
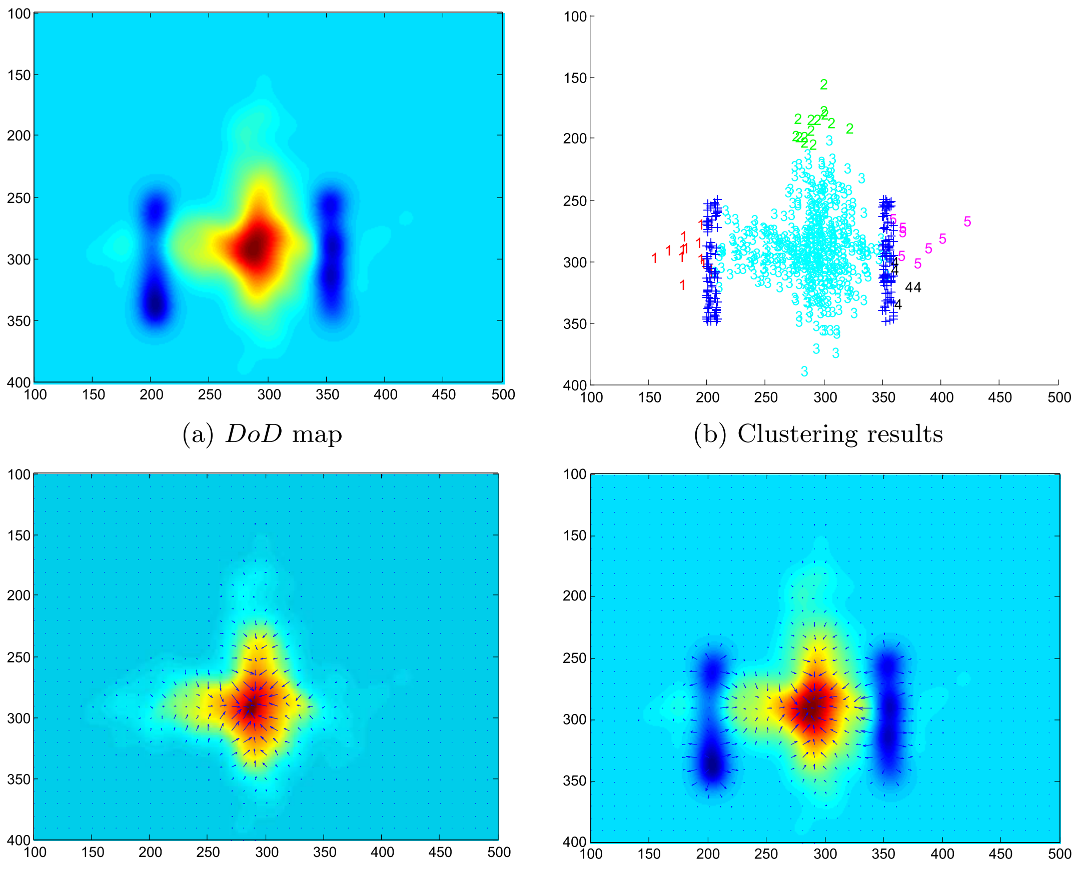


Fig. 2. EM vs. ICC in unsupervised mode. (a) The data are generated by two overlapping Gaussian distributions where both of their means are at $(0, 0)$. The first Gaussian has covariance $\begin{bmatrix} 10 & 0 \\ 0 & 40 \end{bmatrix}$, while the second one $\begin{bmatrix} 60 & 0 \\ 0 & 10 \end{bmatrix}$. Each Gaussian has 500 data points. Assume that we do not know the true number of underlying distributions and guessed that there are five. Using the EM algorithm, we can fit five Gaussian distributions to the data. There is obvious overlap between the resulting distributions near the center. (b) The intra-class clustering method clusters the data into five non-overlapping unimodal Gaussians. This method ensures that the resulting clusters are unimodal and reduces the unnecessary overlap between the clusters.


(a) *DoD* map (b) Clustering results

(c) Gradient of the object class (d) Gradient of *DoD*

Fig. 3. Difference of density (*DoD*) map and context-sensitive intra-class clustering result. (a) The difference of density (DoD) map of an object class and a context class is shown (color key: red > yellow > green > cyan > blue). The object class has the same distribution as the random dots in Fig. 2 in red, green, cyan, black, and pink. The context class is the two vertical distributions of data points marked "+" in (b) and in dark blue. (b) The object class is partitioned into five clusters and labeled by numbers 1–5. The context class data is represented by "+". Note that the resulting sub-cluster distribution is sensitive to the location of the context class: the boundary between label 1 and 3 to the left, and the boundary between 3 and the subclusters 4 and 5 to the right. We can see that the context class data (marked "+") serve as the boundary between object class sub-clusters. (c) The gradient of the object class data only. Without the interference of the context class, there are no local maxima at the left and the right wings. (d) The gradient on *DoD*. The context class data exerts a strong influence, thus separating the object class data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

techniques may be used for the first step, such as PCA or manifold learning methods such as ISOMAP (Tenenbaum et al., 2000) or Locally Linear Embedding (Roweis and Saul, 2000). However, discriminatory information may be hidden in higher dimension than the projected low dimension, resulting in overlaps in the data from different classes after dimensionality reduction. Even though there can be inevitable overlap, ICC can effectively deal with the problem, by generating non-overlapping sub-clusters using context class data as a boundary.

2.1. Main algorithm

The main routine in intra-class clustering (or ICC) is shown in Algorithm 1. The sub-routines *GetDoDMap* and *GetClustersByGradientAscent* used in the main algorithm will be defined in Algorithms 2 and 3 in the following subsections. The main steps of the algorithm are as follows:

1. PCA step:
   (a) Project data points into low-dimensional space using PCA.
2. Difference-of-density estimation step:
   (a) Perform Gaussian kernel density estimation on both the object class and the context class, then
   (b) Find the differenece between the two (Difference-of-Density, DoD).
3. Gradient Ascent Step:
   (a) Calculate the gradient of DoD and perform gradient ascent to locate the local maxima.

Algorithm 1. Intra-class Clustering Algorithm – Main

```
1: function ICC ($W, n, u, v, z, s$) ▷ $W$: the input data set containing $n$ classes, $(u, v)$: the resolution of the DoD map, $z$ and $s$: the filter size and standard deviation of Gaussian for calculating the DoD map.
2:     $D \leftarrow$ PCA ($W$) ▷ Dimensionality reduction by PCA to a 2D point set $D$
3:     for $i = 1$ to $n$ do
4:         $B \leftarrow D_i$ ▷ Set the object class to be $D_i$, the $i$th class in $D$.
5:         $C \leftarrow \overline{D_i}$ ▷ Set the context class to be all other classes except $D_i$
6:         $M \leftarrow$ GetDoDMap ($B, C, u, v, z, s$) ▷ Get DoD map
7:         $L_i \leftarrow$ GetClustersByGradientAscent ($M, B$) ▷ Get cluster labels $L$
8:     end for
9:     return $L$
10: end function
```

2.2. Difference of density (DoD) function

One of the motivations of our algorithm comes from "semi-supervised" learning. Following Chapelle et al. (2006), we represent *side information* in the data by means of pair-wise constraints. As an example, a *must-link* constraint may be used to specify that a pair of points must belong to the same cluster. Likewise, a *cannot-link* constraint may be used to specify that a pair of points should not belong to the same cluster. In our case, a *cannot-link* constraint may be used to push apart data points that are separated by examples from the context class. To implement this idea computationally, we compared the *difference* of density (DoD) between the object and the context class data. (The density can be estimated using Gaussian kernel methods (Parzen, 1962).) DoD is able to address the *cannot-link* property enforced by the *context* class, because the density of the context class data has a separating effect on the grouping of data points in the object class: The context class density appears as valleys that separate the mountains (the object class distribution) in the DoD map. Likewise, gradient ascent on the DoD identifies groups of data points that belong to the same cluster, which is equivalent to establishing *must-link* constraints.

The density $d_i(\mathbf{x})$ for a data point $\mathbf{x} \in \Re^p$ (p-dimensional real space) in class ω_i can be estimated by means of kernel density estimation (Parzen, 1962):

$$d_i(\mathbf{x}) = \frac{1}{n}\sum_{\mathbf{x_e}\in\omega_i} G(\mathbf{x}, \mathbf{x_e}, \Lambda) \quad (1)$$

where n is the total number of examples, $\mathbf{x_e}$ is a sample from the object class, and $\mathbf{\Lambda}$ is a constant covariance matrix for the Gaussian kernel G. The Gaussian centered at $\mathbf{y}$ in p dimensions is defined as:

$$G(\mathbf{x}, \mathbf{y}, \Lambda) = \frac{1}{(2\pi \det\Lambda)^{\frac{p}{2}}} \exp\left(-\frac{(\mathbf{x}-\mathbf{y})^T\Lambda^{-1}(\mathbf{x}-\mathbf{y})}{2}\right) \quad (2)$$

This kernel density estimation step can be a very expensive operation when conducted in high dimension, but in our case it is done in a very low-dimensional space, so it is computationally efficient (Algorithm 1, line 2). Similarly, the density $\overline{d_i(\mathbf{x})}$ of the context class $\overline{\omega_i}$ (all classes except ω_i) with respect to object class ω_i can be defined as:

$$\overline{d_i(\mathbf{x})} = \frac{1}{n}\sum_{\mathbf{x_i}\in\overline{\omega_i}} G(\mathbf{x}, \mathbf{x_i}, \Lambda), \quad (3)$$

The difference of density function (*DoD*) for object class i can then be defined as:

$$DoD_i(\mathbf{x}) = d_i(\mathbf{x}) - \overline{d_i(\mathbf{x})}. \quad (4)$$

The routine that calculates the DoD map is defined in Algorithm 2.

Fig. 3(a) illustrates the DoD function of the object class (same as the distribution in Fig. 2(b)); the added context class is marked "+" in Fig. 3(b). Note that the resulting cluster distribution is sensitive to the position of the context class. As shown in Fig. 3(b), the boundary between labels 1 and 3 on the left, and the boundary between sub-cluster 3 and sub-clusters 4 and 5 on the right are dictated by the region occupied by the context data "+". The gradient of the object class density is shown in Fig. 3(c). Note that without the interference of the context class, there are no local maxima at the left and the right wings. The gradient of the *DoD* map is shown in Fig. 3(d). The context class data exerts a strong influence on the gradient vectors of the object class, and as a result, they separate the object class data into multiple subclusters.

The covariance matrix Λ in Eq. (2) is a user-adjustable parameter, which can be treated in a manner akin to the kernel bandwidth in non-parametric density estimation. Intuitively, Λ can be used to control the number of local maxima in the *DoD* map.

Algorithm 2. Generating the DoD Map

```
1: function GETDODMAP($B, C, u, v, z, s$)    ▷ $B$: PCA-projected object class data set, $C$: PCA-projected context class data set, $(u, v)$: the resolution of the DoD map, $z$ and $s$: the filter size and standard deviation of Gaussian in calculating the DoD map.
2:     $M \leftarrow$ Zeros $(u, v)$    ▷ Zero matrix of size $u \times v$
3:     $B \leftarrow$ Normalize$(B, [1, u])$    ▷ Fit data points into range $[1, u]$
4:     $C \leftarrow$ Normalize$(C, [1, u])$    ▷ Fit data points into range $[1, u]$
5:     for each data point $\mathbf{r} = (r_1, r_2)$ in data sets $B$ do
6:         $M(r_1, r_2) \leftarrow M(r_1, r_2) + 1$    ▷ object class: add 1
7:     end for
8:     for each data point $\mathbf{r} = (r_1, r_2)$ in data sets $C$ do
9:         $M(r_1, r_2) \leftarrow M(r_1, r_2) - 1$    ▷ context class: subtract 1
10:    end for
11:    $M \leftarrow M * $ Gaussian$(z, s)$    ▷ Convolve $M$ with a $z \times z$ Gaussian, std $s$
12:    return $M$
13: end function
```

2.3. Clustering by finding local maxima

A number of density clustering algorithms (Kowalewski, 1995; Hader and Hamprecht, 2003) use the density function as a map, and assign examples on the same "mountain" (local peak) to the same cluster. Mean shift also takes a similar approach (Fukunaga and Hostetler, 1975). Our algorithm can be thought of as a density clustering algorithm with the exception that we use the difference of density (DoD) rather than the plain density. Once the density clustering step is complete, the data in each new cluster will share one local maximum on the *DoD* map. To search for the local maximum of sample $\mathbf{x}$ in the object class, we use the gradient of the *DoD* map (calculated through finite difference as shown in Fig. 3(d)). This gradient can be used to iteratively find the position of the local maximum $\mathbf{y}$ as follows ($\mathbf{y}$ is initialized to $\mathbf{x}$'s position on the *DoD* map):

$$\mathbf{y} \leftarrow \mathbf{y} + \eta \nabla DoD_i, \quad (5)$$

where $\mathbf{y}$ is initialized to $\mathbf{x}$, and η is the learning rate (see, e.g., Press et al., 1992, p. 421).

Once a local maximum for each example is found, all examples sharing the same local maximum are assigned to the same subclass label (Hader and Hamprecht, 2003). The procedure is summarized in Algorithm 3.

Algorithm 3. Density Clustering by Gradient Ascent

```
1: function GETCLUSTERSBYGRADIENTASCENT$M, B$    ▷ $M$: DoD map, $B$: the projected object class data set in 2-D.
2:     $G \leftarrow$ Gradient (M)    ▷ Get the gradient vectors of M
3:     Create an empty list $L$
4:     Create an empty list $T$
5:     for each data point $r$ in data set $B$ do    ▷ Local gradient ascent loop
6:         $y \leftarrow r$
7:         $y_0 \leftarrow (0, 0)$
8:         while $y$ is different from $y_0$ do
9:             $y \leftarrow y + \eta G(y)$    ▷ $\eta$: step size
```

Algorithm 3. Density Clustering by Gradient Ascent

```
10:         $y_0 \leftarrow y$
11:       **end while**
12:       $L \leftarrow$ Enqueue $(y, L)$
13:       **if** $y$ not in $T$ **then**
14:         $T \leftarrow$ Enqueue $(y, T)$
15:       **end if**
16:     **end for**
17:     **for** each item $y$ in $L$ **do**
18:       $i \leftarrow$ Search $(y, T)$          ▹ Get the index $i$ of item $y$ in table $T$
19:       Replace item $y$ in $L$ with the index $i$
20:     **end for**
21:     **return** $L$
22: **end function**
```

2.4. ICC as a preprocessing step for a classifier

For actual classification tasks, the proposed ICC algorithm is employed as a data *pre-processing step* before the application of a proper pattern classifier. The general procedure of applying intra-class clustering for classification is described in Algorithm 4.

Algorithm 4. Classifier with ICC-Preprocessing

```
1: **function** CLASSIFY(Training data set:$W$, Testing data set:$S$)
2:     $(u, v) \leftarrow$ DoD map resolution
3:     $z \leftarrow$ Gaussian filter size for DoD map calculation
4:     $s \leftarrow$ Gaussian filter's standard deviation for DoD map calculation
5:     $n \leftarrow$ number of classes in $W$
6:     $L \leftarrow$ ICC$(W, n, u, v, z, s)$
7:     Create a Map $P$ to map sub-classes labels $L$ to their original class labels
8:     $[D, v] \leftarrow$ LDA$(W, L)$          ▹ $D$: projected data points, $v$: LDA vectors
9:     $S' \leftarrow Sv$          ▹ Project test set using LDA vectors
10:    $R \leftarrow$ QuadraticClassifier$(D, L, S')$          ▹ Standard quadratic classifier
11:    $Y \leftarrow$ LookUp$(R, P)$          ▹ Recover original labels from map $P$
12:    **return** $Y$
13: **end function**
```

3. Experiments

We tested the ICC algorithm in supervised mode (Algorithm 4) for classification on both synthetic data (experiment 1) and real data (experiments 2 and 3). The specific parameters were as follows: the grid size $u = 600$, $v = 200$ for the first experiment (synthetic data), while $u = 120$, $v = 500$ for the second experiment (real world data). For the Gaussian filter, we chose window size $z = 100$, and standard deviation $s = 25$. In the third experiment, we tested ICC with different (u, v) configurations, and observed the effects of changing grid size $(u \times v)$. In all of our experiments, we set the learning rate η to 1.

3.1. Experiment 1: test on synthetic data

In this experiment, we evaluated ICC on a synthetic data set containing 3 classes in 30 dimensions. The data were generated as follows. First, we drew random samples from three-dimensional Gaussian distributions, one Gaussian per class. The means were $(8, 10, 0)$, $(8, 10, 0)$, $(0, 4, 0)$; and the covariances were as follows:

$$\begin{pmatrix} 2 & 3 & 0 \\ 3 & 6 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 6 & -3 & 0 \\ -3 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

respectively. Then, we added 27 noisy dimensions (uniformly random values in the range [0.0, 0.2]). Finally, we rotated the data using a randomly generated matrix (uniformly random values in the range [0.0, 0.5]). As a result, each feature dimension consisted of a weighted sum of three signal channels and 27 noise channels.

We generated 700 data points, and split them into a training set with 650 data points and a test set 50 data points. As shown in Fig. 4(a), PCA of the training set reveals a high level of overlap between classes 1 and 2. The particular structure of the data poses significant difficulty for LDA, as shown in Fig. 4(b), because classes 1 and 2 are largely overlapping. However, by using the context-sensitive clustering algorithm (ICC), the two overlapping Gaussians of class 1 and 2 can be separated into 4 sub-classes (each class has two local maxima in their *DoD* map). The final results from ICC + LDA (Fig. 4(d)) show markedly better separability.

Results on the test set are summarized in Table 1. ICC + LDA outperformed all other methods ($p < 10^{-4}$, t-test, $n = 20$), whereas EM + LDA was not significantly better than LDA (for EM, we allowed up to 2 sub-clusters per class). This is because the context-sensitive approach can preserve the information from the context class, and reduce the overlap between all clusters.

To guard against the possibility that ICC is better merely due to the fact that ICC can use arbitrarily many sub-clusters, we allowed EM to subdivide each class into even more sub-classes. ICC's performance was still significantly better than EM: for 3 sub-classes, 4.0% higher in accuracy (ICC + LDA = 87.4% vs. EM + LDA = 83.4%, $p = 0.047$, t-test, $n = 20$). (Note: accuracies are slightly different from the original experiment since these were fresh new random trials albeit with the same Gaussian parameters Table 1.) ICC + LDA and EM + LDA achieve similar accuracy only when the number of sub-class was increased to 5 (ICC = 87.4% vs. EM = 86.0%, $p = 0.24$, t-test, $n = 20$).

3.2. Experiment 2: test on real data – olfactory database

To evaluate the efficiency of our algorithm on a realistic task, we tested ICC on experimental data from an olfactory database. The data were generated by a gas sensor array containing 60 features (Gutierrez-Osuna and Nagle, 1999). The data contained five classes, each class consisting of various cola flavors from a different manufacturer: (1) Coca-Cola Co. (Coke, Diet Coke, and Cherry Coke), (2) Pepsi Cola Co. (Pepsi, Cherry Pepsi), (3) Dr. Pepper Snapple Grp. (Dr. Pepper), (4) Carolina Beverage Corp. (Cheerwine), and (5) other (RC Cola, Eckerd Cola, Eckerd Dr. Riffic). The data set was split into a training set with 150 examples, and a test set with 25 examples. Fig. 5(a) shows the PCA plot of the training set projected onto the first and the second eigenvectors, whereas Fig. 5(b) shows the projection of the data onto the first and the second LDA dimensions.

As shown in Fig. 5(a), the structure of the data poses a problem for LDA: nearly all classes are multimodal (e.g. class 5). Therefore, breaking these classes into sub-classes is likely to improve the separability in LDA. Following application of ICC (with around 8–10 clusters in each trial), class boundaries become quite clear (fig. 5(d)). EM (with 10 clusters) improves matters but there is a major overlap between class 1 and class 5. As shown in Table 2, both methods outperform LDA: ICC + LDA outperforms LDA by 16.0%, a statistically significant difference ($p = 6.05 \times 10^{-7}$, t-test,$n = 20$),

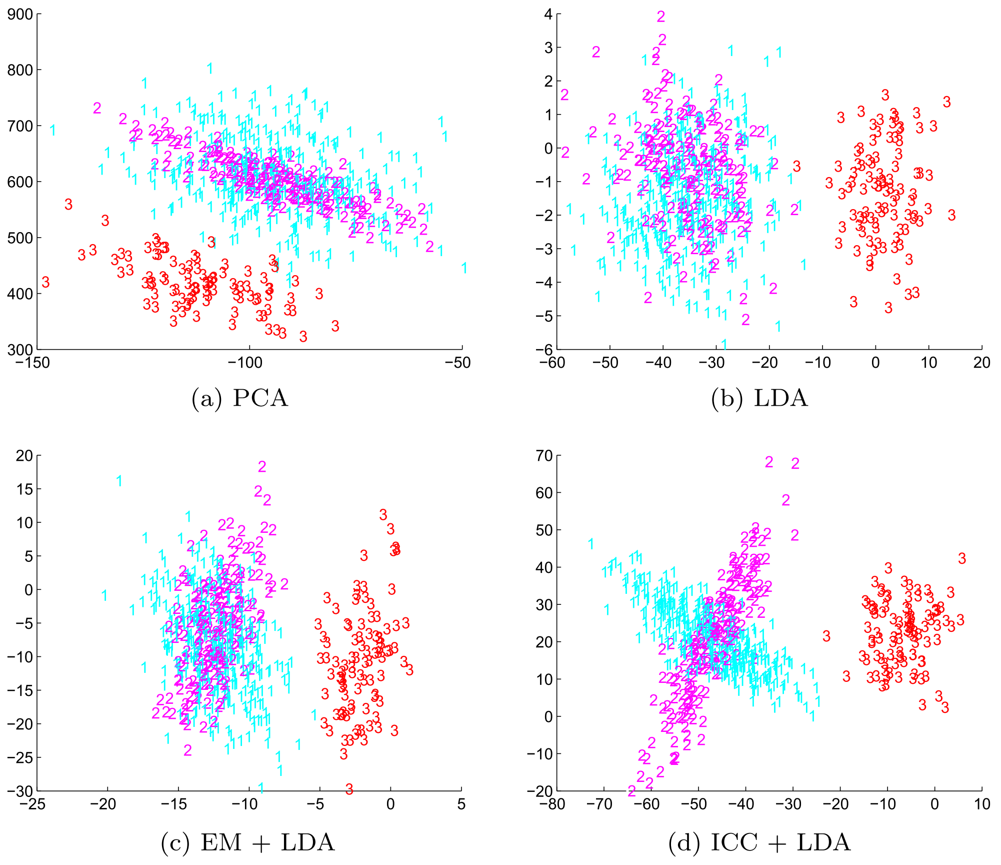


Fig. 4. Experiment 1 results. (a) The PCA plot of the synthetic data set shows that each class has a unimodal Gaussian distribution, and class 1 and class 2 have a large degree of overlap. Such a structure poses significant difficulty for LDA as shown in (b), even with the EM step (c). Using our context sensitive intra-class clustering algorithm (ICC), classes 1 and 2 can be separated into four sub-clusters. (d) With more detailed cluster information, the resulting LDA maximally separates the three classes.

Table 1
Experiment 1 classification accuracy (Test set).

Pre-processing method	Accuracy (%)	Std.
PCA only	54.3	0.0684
LDA only	70.6	0.0602
EM + LDA	78.7	0.0852
ICC + LDA	88.0	0.0426

whereas EM + LDA outperforms LDA by 3.4%, a difference that was not statistically significant ($p = 0.154$, t-test, $n = 20$).

3.3. Experiment 3: test on real data – North Texas vowel database

We further tested with a larger dataset, the North Texas vowel database with 3434 samples (Assmann et al., 2008). The dataset was obtained by recording the acoustic signals of 12 vowels from 5 groups of subjects, which include adult males and females, and children of 3, 5, and 7 years of age. Up to 10 speakers were recorded in each group. This is a challenging dataset where each single class (each vowel) includes samples from different age and sex groups. Thus, intra-class clustering can help delineate such internal structure. The dataset contains 8 features that were extracted from the sound recordings with the following features: (1) F0 (mean fundamental frequency over all voiced frames), (2) F1i (initial F1 estimate, 20% point), (3) F2i (initial F2 estimate, 20% point), (4) F3i (initial F3 estimate, 20% point), (5) F1f (final F1 estimate, 80% point), (6) F2f (final F2 estimate, 80% point), (7) F3f (final F3 estimate, 80% point), and (8) DUR (vowel duration in milliseconds). Our task was to predict the index of the vowels: from 1 to 12. In each trial, we randomly selected 90% of the samples from the dataset as training data and the other 10% as testing data. In each test, we ran 20 trials to get the average result.

We performed two groups of tests with this dataset. The first group was aimed at testing the effect of the ICC DoD grid size. The second group compared the ICC algorithm with other methods.

3.3.1. Tests with different grid size

In the first group of tests, we applied ICC on two dimensional PCA space. To test the effect of varying DoD grid size, we tested grid sizes of 100×100, 150×150, 200×200, 250×250, and 300×300 (the filter size was fixed at 60×60). Fig. 6 compares the results of these grid configurations, the number of total subclusters in the 12 classes found by ICC, as well as the average running time of each configuration.

Fig. 6(b) shows that when the filter size is fixed, larger grid size results in more clusters. We observed that the total number of subclusters increases from 15 to 108 when the grid size changes from 100×100 to 300×300. The grid size of 200×200 gave the best accuracy of 78.5% as shown in Fig. 6(a), and with this configuration each class contained 3 to 4 clusters on average. Even though there are small variations in the accuracy with different grid sizes, we can observe that ICC is not too sensitive to these parameters: the results are within the range of 76% to 79% when the grid becomes 9 times larger in area.

In this experiment, each class data was in a Gaussian-like distribution, but there was a significant overlap among the classes, which raised the difficulty for traditional classifiers. This type of data with heavy overlap is suitable for ICC because it decomposes each class into non-overlapping sub-clusters using the context

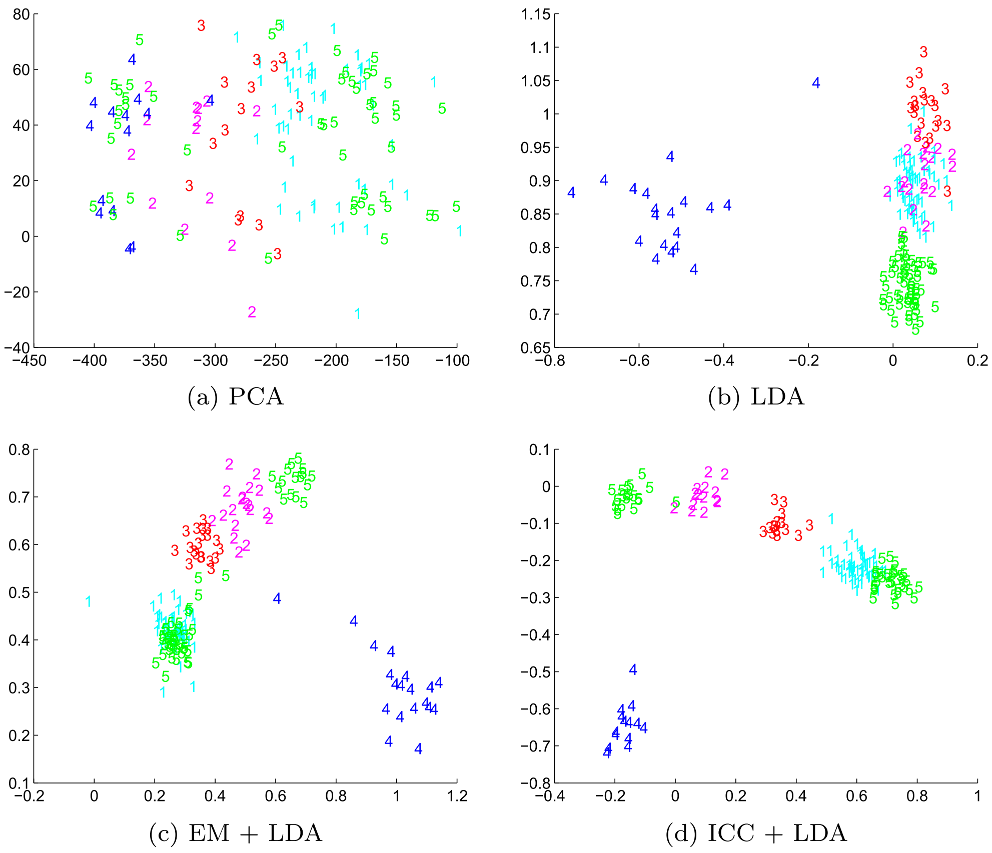


Fig. 5. Experiment 2 Results. (a) PCA of the olfactory data shows that several classes are multimodal non-Gaussian (e.g., classes 4 and 5). ICC separates each class data into unimodal Gaussians, which leads to improvements in discrimination performance when LDA is applied, e.g., compare (b) and (d) Since ICC also takes into account the context class distribution, ICC + LDA can reduce the overlap between classes as compared to EM + LDA in (c). See text for details and Table 2 for accuracies.

Table 2
Experiment 2 classification accuracy (Test set).

Pre-processing method	Accuracy (%)	Std.
PCA only	50.4	0.0815
LDA only	68.2	0.0779
EM + LDA	71.6	0.0788
ICC + LDA	84.2	0.0908

Table 3
Experiment 3B classification accuracy (test set).

Methods	Accuracy (%)	Std.	Time (s)
ICC + LDA	79.2	0.027	10
EM + LDA	79.2	0.0183	120
LDA only	75.9	0.027	<1
PCA only	41.73	0.0254	<1

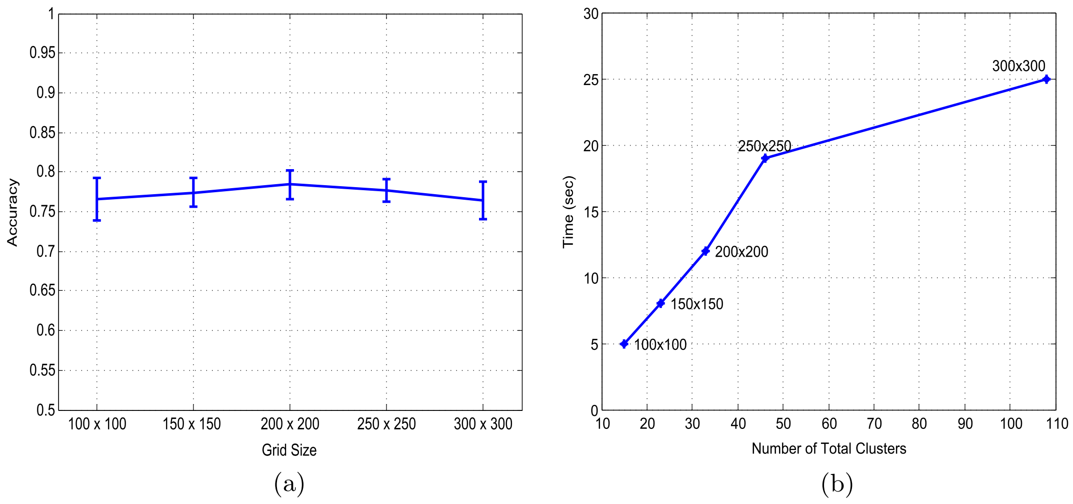


Fig. 6. Experiment 3A: effects of grid size ($u \times v$) in ICC. (a) Accuracy as a function of grid size. The error bars are the standard deviations of each grid size configuration. (b) The running time of ICC as a function of total sub-clusters. The grid size ($u \times v$) is marked along the data point.

class data as boundaries, and these sub-clusters help improve the classification accuracy. As shown in the results, when each class is divided into 3 or 4 clusters on average, best accuracy can be achieved.

The running time shown in Fig. 6(b) was recorded on a computer with AMD Phenom (tm) 9500 quad-core processor at 2.2 GHz and 8G memory. The running time of ICC grows nearly linearly as the grid size increases.

3.3.2. Tests with different methods

We compared ICC performance with other methods (EM + LDA, LDA only, and PCA only) as in the two previous experiments. The training samples were first projected onto the first two dimensions on PCA, and then ICC applied. In ICC, we used grid size 130×200 with filter size 60×60, and it created about 43 total sub-clusters on average. For the EM method, each class was divided into the same number of sub-clusters as ICC.

As the results show in Table 3, ICC and EM have similar accuracies, and they both outperform all other methods. Due to smaller overlap among the PCA 2D projections of the different classes in this data set, and EM subdividing each class into 3 to 4 clusters as ICC did, EM was able to achieve the accuracy as high as ICC. However, EM took about 12 times longer to run than ICC.

4. Discussion

Intra-class clustering was first used to improve facial feature detection by Lucey et al. (2003). In this approach, a Gaussian mixture model was employed to partition the data from each class into several Gaussian distributions. The parameters of each Gaussian model were estimated by EM. However, EM-based clustering algorithms cannot overcome issues related to the presence of context classes. Moreover, as shown in our experiments, context-sensitive intra-class clustering outperformed EM in enhancing the separability in subsequent LDA transform and classification.

A possible concern regarding our algorithm is that of computational cost. Kernel density estimation can be very expensive for high-dimensional data. However, we circumvent this problem by operating in projected low-dimensional space. Possible problems due to class overlap introduced by dimensionality reduction are overcome by generating non-overlapping sub-clusters. (Note: In cases where a shallow valley in the object class distribution overlaps with a deeper valley in the context class, a small local peak can result in the DoD at the overlapping valley. However, this apparent artifact is not a problem since it simply indicates the high local "contrast" of the object class in comparison to the surrounding context class.).

For n samples in d dimension (or number of attributes), the time complexity of PCA transform to 2D space is $O(d^2 n)$ (Sharma and Paliwal, 2007). Next, as shown in the main ICC algorithm (see Algorithm 1), the convolution step and gradient search will be repeated for each class. Assuming that the grid size is $u \times u$, convolution takes $O(u^2 \log u)$, and the gradient ascent step for seeking the local maxima is linear with respect to the diameter of the search region (Kowalewski, 1995), i.e., $O(un)$. Therefore, the overall time complexity is $O\left((d^2 + cu)n + cu^2 \log u\right)$, where c is the number of classes. Regarding space complexity, the algorithm is also very efficient. Besides the space needed for the input data and its PCA transform $(O(nd + d^2))$, the searching operations are carried out on the *DoD* map, which is a matrix with size $u \times v$ as shown in the Algorithm 1. In our experiments, the grid sizes u or v are within a few hundreds. In sum, the space complexity is $O(nd + d^2 + uv)$.

In order to test the computation time of ICC when the number of samples grows, we conducted a test by using a synthetic dataset with two classes as shown in Fig. 3. In the test, the total number of samples ranged from 500 to 10 K. The first class contained 60% of the samples, while the second class contained the other 40% of the samples. The running time as a function of total sample numbers is shown as the Fig. 7. With ICC, the first class was divided into 3 clusters, and the second one was divided into two clusters. ICC was applied on 2D PCA projections, and the grid size was 100×100, and filter size was 50×50. Our results show that as the number of samples grows, the running time increases linearly. We also compared the running time of EM vs. ICC. In the test, EM was allowed to generate the same number of clusters as ICC. Our results show that EM is orders of magnitude slower than ICC. This test was performed on a computer with AMD Phenom (tm) 9500 quad-core processor at 2.2 GHz with 8G memory.

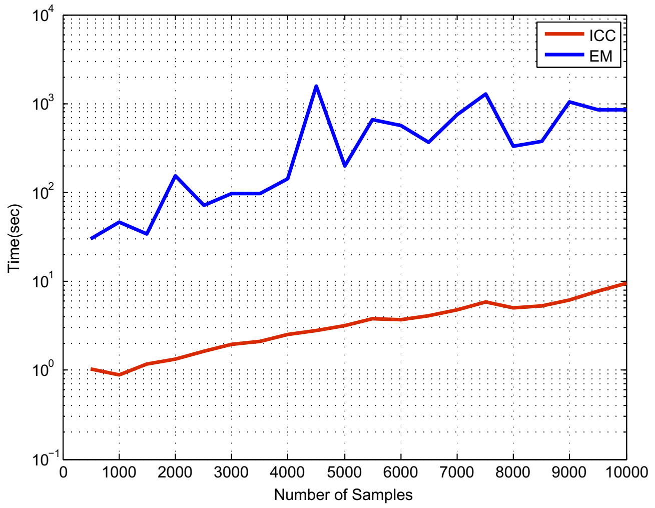


Fig. 7. Computation time of ICC and EM as a function of the number of total class samples. We tested the performance using a synthetic dataset with 2 attributes and 2 classes as in Fig. 3. In the test, the total number of samples was a variable, and the running time of EM vs. ICC was compared. With ICC, the first class was divided into three clusters, and the second one into two clusters. ICC was applied on 2D PCA projections, where the grid size was 100×100, and the filter size was 50×50. We allow the same number of clusters to be generated for EM. The result shows that ICC is orders of magnitude faster than EM. This test was performed on a computer with AMD Phenom (tm) 9500 quad-core processor at 2.2 GHz, and 8G memory.

We can further improve the performance by evaluating the gradient only along the path when searching for the subcluster centers. This method can be implemented by using Newton's method, which can iteratively and efficiently find the local maxima. One potential advantage of using this is that we do not need to operate ICC on the PCA 2D space, and the searching can be done directly on the original data in high dimension.

Our ICC algorithm is closely related to discriminative classifiers (DC) (Ng and Jordan, 2001). DC shares many characteristics with ICC: (1) DC is a density-based classification method, (2) DC is context-sensitive, since it utilizes information from all non-target classes, and (3) it uses gradient-ascent to find the local maxima. ICC extends DC in the following ways: (1) ICC operates in semi-supervised learning mode while DC does so in supervised learning mode only. (2) ICC can reduce the overlap across distributions from different classes by decomposing a class into several sub-classes. (3) ICC can be used as a data preprocessing method.

ICC can also be seen as an extension of mean-shift analysis (MSA) (Fukunaga and Hostetler, 1975), an unsupervised mode seeking algorithm. MSA looks for the local maxima in the space of sample density. MSA is non-parametric just like ICC, but it does not utilize any contextual information (side-information) to detect the modes in the data distribution. In contrast, ICC searches for peaks in the space of the DoD, thus utilizing the constraints from the context class.

5. Conclusion

In this paper, we proposed a computationally cheap algorithm that can separate an arbitrary data distribution into non-overlapping unimodal clusters, while utilizing intervening context data distributions to further separate the clusters. Computational results with synthetic data and real-world data showed that the method can help improve the performance of LDA and standard classifiers such as the quadratic classifier. We expect our approach to be applicable to a broader class of pattern recognition problems where class-conditional densities are significantly non-Gaussian or multi-modal.

Acknowledgment

This publication is based in part on work supported by Award No. KUSC1–016-04, made by King Abdullah University of Science and Technology (KAUST).

References

Assmann, P., Nearey, T.M., Bharadwaj, S.V., 2008. Analysis and classification of a vowel database. Canadian Acoustics 36 (3), 148–149.
Chapelle, O., Scholkopf, B., Zien, A., 2006. Semi-Supervised Learning. MIT Press.
Dempster, A., Laird, N., Rubin, D., 1977. Maximum likelihood from incomplete data via the EM algorithm. Royal Statistical Society 39, 1–38.
Duda, R.O., Hart, P., Stork, D.G., 2001. Pattern Classification, second ed. Wiley, New York.
Fukunaga, K., Hostetler, L.D., 1975. The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Transactions on Information Theory 21 (1), 32–40.
Gutierrez-Osuna, R., Nagle, H.T., 1999. A method for evaluating data-preprocessing techniques for odor classification with an array of gas sensors. IEEE Transactions on Systems, Man, and Cybernetics B 29 (5), 626–632.
Hader, S., Hamprecht, F.A., 2003. Efficient density clustering using basin spanning trees. In: Between Data Science and Applied Data Analysis. Springer, pp. 39–48.
Jain, A.K., 2010. Data clustering: 50 years beyond k-means. Pattern Recognition Letters 31, 651–666.
Kowalewski, F., 1995. A gradient procedure for determining clusters of relatively high point density. Pattern Recognition 28, 1973–1984.
Lucey, S., Sridharan, S., Chandran, V., 2003. Improved facial feature detection for AVSP via unsupervised clustering and discriminant analysis. EURASIP Journal on Applied Signal Processing 2003 (3), 264–275.
Ng, A.Y., Jordan, M.I., 2001. On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In: Advances in Neural Information Processing Systems, pp. 841–848.
Parzen, E., 1962. On estimation of a probability density function and mode. Annals of Mathematical Statistics 33, 1065–1076.
Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 1992. Numerical Recipes in C, second ed. Cambridge University Press.
Roweis, S., Saul, L., 2000. Nonlinear dimensionality reduction by locally linear embedding. Science 290 (5500), 2323–2326.
Sharma, A., Paliwal, K.K., 2007. Fast principal component analysis using fixed-point algorithm. Pattern Recognition Letters 28, 1151–1155.
Tenenbaum, J.B., de Silva, V., Langford, J.C., 2000. A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323.
Wang, M., Perera-Lluna, A., Gutierrez-Osuna, R., 2004. Principal discriminants analysis for small-sample-size problems: application to chemical sensing. In: Proceedings of the third IEEE SENSORS 2004, vol. 2, Vienna, pp. 591–594.
Zhao, W.Y., 2000. Discriminant component analysis for face recognition. In: 15th International Conference On Pattern Recognition, pp. 818–21.