Data Min Knowl Disc (2018) 32:385–416
https://doi.org/10.1007/s10618-017-0539-5

Infinite ensemble clustering

Hongfu Liu[1] · **Ming Shao**[2] · **Sheng Li**[3] · **Yun Fu**[1,4]

Received: 1 February 2017 / Accepted: 7 August 2017 / Published online: 20 August 2017

Abstract Ensemble clustering aims to fuse several diverse basic partitions into a consensus one, which has been widely recognized as a promising tool to discover novel clusters and deliver robust partitions, while representation learning with deep structure shows appealing performance in unsupervised feature pre-treatment. In the literature, it has been empirically found that with the increasing number of basic partitions, ensemble clustering gets better performance and lower variances, yet the best number of basic partitions for a given data set is a pending problem. In light of this, we propose the Infinite Ensemble Clustering (IEC), which incorporates marginalized denoising auto-encoder with dropout noises to generate the expectation representation for infinite basic partitions. Generally speaking, a set of basic partitions is firstly generated from the data. Then by converting the basic partitions to the 1-of-K codings, we link

Responsible editor: Pierre Baldi.

✉ Hongfu Liu
liu.hongf@husky.neu.edu

Ming Shao
mshao@umassd.edu

Sheng Li
lisheng1989@gmail.com

Yun Fu
yunfu@ece.neu.edu

1 Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA

2 Department of Computer and Information Science, University of Massachusetts Dartmouth, Dartmouth, MA, USA

3 Adobe Research, San Jose, CA, USA

4 College of Computer and Information Science, Northeastern University, Boston, MA, USA

the marginalized denoising auto-encoder to the infinite basic partition representation. Finally, we follow the layer-wise training procedure and feed the concatenated deep features to K-means for final clustering. According to different types of marginalized auto-encoders, the linear and non-linear versions of IEC are proposed. Extensive experiments on diverse vision data sets with different levels of visual descriptors demonstrate the superior performance of IEC compared to the state-of-the-art ensemble clustering and deep clustering methods. Moreover, we evaluate the performance of IEC in the application of pan-omics gene expression analysis application via survival analysis.

Keywords Ensemble clustering · Denoising auto-encoder · K-means

1 Introduction

Ensemble clustering, also known as *consensus clustering*, emerges as a promising way for multi-source, heterogeneous data clustering, and recently attracts increasing academic attention due to the robust and high-quality partitions. It aims to find a single partition that mostly agrees with multiple existing basic ones (Strehl and Ghosh 2003). It is of recognized benefits in generating robust partitions, discovering novel structures, handling noisy features, and integrating solutions from multiple sources (Nguyen and Caruana 2007).

Recently, representation learning attracts substantial research attention, which has been widely adopted as the unsupervised feature pre-treatment (Bengio et al. 2013). The layer-wise training and the followed deep structure are able to capture the visual descriptors from coarse to fine (Bengio et al. 2007; Hinton et al. 2006). Notably, there are a few deep clustering methods proposed recently, working well with either feature vectors (Shao et al. 2015) or graph Laplacian (Huang et al. 2014; Li et al. 2014), towards high-performance generic clustering tasks.

Tremendous efforts have been made in ensemble clustering and deep representation, which lead us to wonder whether these two powerful tools can be strongly coupled for the unsolved challenging problems. For example, it has been widely recognized that with the increasing number of basic partitions, ensemble clustering achieves better performance and lower variance (Wu et al. 2015; Luo et al. 2011). However, the best number of basic partitions for a given data set still remains an open problem. Too few basic partitions cannot exert the capacity of ensemble clustering, while too many basic partitions lead to unnecessary computational resource waste. Unfortunately, we cannot foreknow the best number of basic partitions, which leads the problem we address here to how to fuse infinite basic partitions for ensemble learning.

Here we aim to fuse infinite basic partitions for ensemble clustering. For a set of basic partitions, we can randomly remove some labels in the basic partitions to generate the extra incomplete basic partitions. Such process is just the same with denoising auto-encoder with dropout noises. If we repeat the process infinite times, it leads to the marginalized denoising auto-encoder. Therefore, our model links the marginalized denoising auto-encoder to ensemble clustering and leads to a natural integration named "Infinite Ensemble Clustering" (IEC), which is simple yet effective and efficient. To

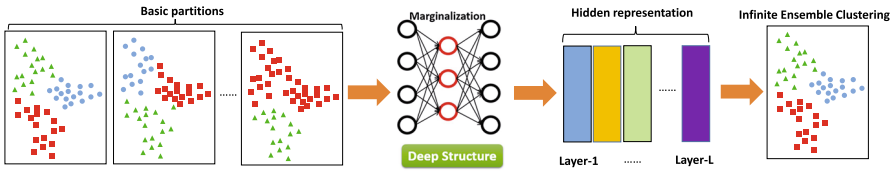

Fig. 1 Framework of IEC. We apply marginalized denoising auto-encoder to generate infinite ensemble members by adding drop-out noise and fuse them into the consensus one. Then K-means is conducted on the concatenated hidden representation for infinite ensemble clustering

that end, we first generate a moderate number of basic partitions, as the basis for the ensemble clustering. Second, we convert the preliminary clustering results from the basic partitions to 1-of-K codings, which disentangles dependent factors among data samples. Then the codings are fed into marginalized denoising auto-encoders for generating the infinite representation. Two different deep representations of IEC are provided with the linear or non-linear model. Finally, we run K-means on the learned representations to obtain the final clustering. The framework of IEC is demonstrated in Fig. 1. The whole process is similar to marginalized denoising auto-encoder (mDAE). Several basic partitions are fed into the deep structure with drop-out noises in order to obtain the expectation of the co-association matrix. Extensive results on diverse vision data sets show that our IEC framework works fairly well with different visual descriptors, in terms of time efficiency and clustering performance, and moreover some key impact factors are thoroughly studied as well. The pan-omics gene expression analysis application shows that IEC is a promising tool for real-world multi-view and incomplete data clustering. We highlight our contributions as follows.

- We propose a framework called Infinite Ensemble Clustering (IEC) which integrates the deep structure and ensemble clustering. By this means, the complex ensemble clustering problem can be solved with a stacked marginalized denoising auto-encoder structure in an efficient way.
- Within the marginalized denoising auto-encoder, we fuse infinite ensemble members into a consensus one by adding drop-out noises, which maximizes the capacity of ensemble clustering. Two versions of IEC are proposed with different deep representations.
- Extensive experimental results on numerous real-world data sets with different levels of features demonstrate IEC has obvious advantages on effectiveness and efficiency compared with the state-of-the-art deep clustering and ensemble clustering methods, and IEC is a promising tool for large-scale data clustering.
- The pan-omics gene expression analysis application illustrates the effectiveness of IEC to handle multi-view and incomplete data clustering.

This paper is an extension of our conference paper (Liu et al. 2016). The new contents include (1) we add more information about the preliminary knowledge on ensemble clustering and auto-encoder for comprehensive understanding, (2) we provide the non-linear version of IEC and compare the clustering performance of the linear and non-linear models, and (3) a thorough application on pan-omics gene expression

analysis is employed to fully evaluate the performance of IEC via survival analysis and demonstrate the practicability of IEC with multi-view and incomplete data clustering.

The rest of this paper is organized as follows. In Sect. 2, we introduce the related work on ensemble clustering and auto-encoder. Then some preliminaries and problem definition are given in Sect. 3. The framework and details of IEC are demonstrated in Sect. 4. Extensive experimental results and gene expression analysis are showcased in Sects. 5 and 6, respectively, followed by the conclusion in Sect. 7.

2 Related work

Here we introduce the related work in terms of ensemble clustering and auto-encoder, and highlight the differences between existing methods and ours.

2.1 Ensemble clustering

Ensemble clustering aims to fuse various existing basic partitions into a consensus one, which can be divided into two categories: with or without explicit global objective functions. In a global objective function, a utility function is employed to measure the similarity between a basic partition and the consensus one at the partition level. Then the consensus partition is achieved by maximizing the summarized utility function. In the inspiring work, Topchy et al. (2003) proposed a Quadratic Mutual Information based objective function for consensus clustering, further they used the expectation-maximization algorithm with a finite mixture model for consensus clustering (Topchy et al. 2004). Wu et al. put forward a theoretic framework for K-means-based Consensus Clustering (KCC), and gave the sufficient and necessary condition for KCC utility functions that can be maximized via a K-means-like iterative process (Wu et al. 2015, 2013; Liu et al. 2015b, 2016). In addition, there are some other interesting objective functions for consensus clustering, such as the ones based on nonnegative matrix factorization (Li et al. 2007), kernel-based methods (Vega-Pons et al. 2010), simulated annealing-based method (Lu et al. 2008), etc.

Another kind of methods do not set explicit global objective functions, which transforms it into graph partition problem. In one pioneer work, Strehl and Ghosh (2003) (GCC) developed three graph-based algorithms for consensus clustering. More methods, however, employ co-association matrix to calculate how many times two instances jointly belong to the same cluster. By this means, some traditional graph partitioning methods can be called to find the consensus partition. Fred and Jain (2005) (HCC) is the most representative one in the link-based methods, which applied the agglomerative hierarchical clustering on the co-association matrix to find the consensus partition, while SEC applies the spectral clustering on the co-association matrix with a weighted K-means solution (Liu et al. 2015a, 2017a). Other methods include Relabeling and Voting (Ayad and Kamel 2008), Locally Adaptive Cluster based methods Domeniconi and Al-Razgan (2009), Robust Spectral Ensemble Clustering (Tao et al. 2016) and Simultaneous Clustering and Ensemble (Tao et al. 2017), etc. There are still many other algorithms for ensemble clustering. Readers with interests can refer to some survey papers for more comprehensive understanding (Vega-Pons and Ruiz-Shulcloper 2011).

2.2 Auto-encoder

Auto-encoder is a building block of deep structure that learns hidden and compressed representations (i.e., codings) from data (Bengio 2009), which has been widely used in numerous applications. Denoising auto-encoder (DAE) and stacked DAE are two representitive variants of auto-encoder, which learn effective representations by reconstructing input data from artificial corruptions (Vincent et al. 2008). Marginalized denoising auto-encoder (mDAE) approximately marginalizes out the corruptions during training, taking into account infinitely many corrupted copies of training data (Chen et al. 2014, 2012). Due to the flexibility and impressive learning capability, auto-encoder and its variants have been successfully applied to many scenarios, such as face recognition (Meina et al. 2014), domain adaptation (Chen et al. 2012; Ding et al. 2015), and image classification (Xie et al. 2015).

Most recently, a few auto-encoder based methods have been proposed for graph clustering. Song et al. (2013) augmented the loss function of auto-encoder by incorporating a constraint of the distance between samples and centroids. Huang et al. (2014) built a deep embedding network using auto-encoder, and incorporated locality-preserving and group sparsity constraints to the loss function of deep network for clustering-oriented representations. Tian et al. (2014) revealed the similarity between auto-encoder and spectral clustering, and presented a GraphEncoder method based on sparse auto-encoder. Shao et al. (2015) proposed a deep linear coding approach, which jointly learns feature transforms and discriminative codings for fast graph clustering. However, the connection between auto-encoder and ensemble clustering has not been explored.

In this paper, we aim to build the connection between ensemble clustering and auto-encoder for unsolved challenged problems, and apply marginalized denoising auto-encoder to fuse infinite basic partitions for ensemble clustering.

3 Preliminaries and problem definition

In this section, we introduce the preliminary knowledge in terms of ensemble clustering and marginalized denoising auto-encoder, and then formulate the research problem.

3.1 Ensemble clustering

The goal of ensemble clustering is to find a single partition which agrees with existing basic partitions as much as possible. Different from the traditional clustering methods, which aim to separate a bunch of data instances into different groups that the instances in the same group are more similar to each other, ensemble clustering fuses several different partitions into a consensus one. The input of traditional clustering methods is the data matrix, while the input of ensemble clustering is a set of basic partitions. Here basic partitions might be generated by the same clustering algorithm with different parameters, or by the same clustering algorithm with different features or even by several different clustering algorithms. In essence, ensemble clustering is a fusion problem, rather than a clustering problem.

Given a set of r basic partitions of the data matrix $\mathbf{X}$: $\mathcal{H} = \{\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \ldots, \mathbf{H}^{(r)}\}$ with the number of clusters in $\mathbf{H}^{(i)}$ to be K_i, the goal is to fuse all the basic partitions into a consensus partition $\mathbf{H}^*$. Generally speaking, ensemble clustering can be roughly divided into two categories in terms of measuring the similarity in different levels.

The first category designs the utility function to measure the similarity between the final consensus partition and basic ones. Usually the following formulation is used to solve ensemble clustering.

$$\Gamma(\mathbf{H}^*, \mathcal{H}) = \sum_{i=1}^{r} U(\mathbf{H}^*, \mathbf{H}^{(i)}), \tag{1}$$

is maximized with respect to $\mathbf{H}^*$, where $\Gamma : \mathcal{N}^{n\times K} \times \mathcal{N}^{n\times Kr} \mapsto \mathcal{R}$ is a *consensus function*, and $U : \mathcal{N}^{n\times K} \times \mathcal{N}^{n\times K} \mapsto \mathcal{R}$ is a *utility function*, $i = 1, 2, \ldots, r$.

Here utility function plays a role in measuring the similarity of two partitions. Therefore, the choice of the utility function is critical for the success of a consensus clustering. In the literature, many external measures originally proposed for cluster validity have been adopted as the utility functions for consensus clustering, such as Normalized Mutual Information (Strehl and Ghosh 2003), Category Utility Function (Mirkin 2001), Quadratic Mutual Information (Topchy et al. 2003), and Rand Index (Lu et al. 2008). These utility functions, together with the consensus function, largely determine the quality of consensus clustering.

For better understanding of utility function, a *contingency matrix* is often employed for computing the difference of two partitions. In Table 1, $n_{kj}^{(i)}$ denotes the number of data objects shared by both cluster $C_j^{(i)}$ in $\mathbf{H}^{(i)}$ and cluster C_k in $\mathbf{H}^*$, $n_{k+} = \sum_{j=1}^{K_i} n_{kj}^{(i)}$, and $n_{+j}^{(i)} = \sum_{k=1}^{K} n_{kj}^{(i)}$, $1 \le k \le K$, $1 \le j \le K_i$. Let $p_{kj}^{(i)} = n_{kj}^{(i)}/n$, $p_{k+} = n_{k+}/n$, and $p_{+j}^{(i)} = n_{+j}^{(i)}/n$, we then have the *normalized contingency matrix* for utility computation. For instance, the well-known Category Utility Function (Mirkin 2001) can be computed as follows:

$$U_c(\mathbf{H}^*, \mathbf{H}^{(i)}) = \sum_{k=1}^{K} p_{k+} \sum_{j=1}^{K_i} (p_{kj}^{(i)}/p_{k+})^2 - \sum_{j=1}^{K_i} (p_{+j}^{(i)})^2. \tag{2}$$

Note that a larger U_c indicates a higher similarity.

Table 1 The contingency matrix

$\mathbf{H}^*$	$\mathbf{H}^{(i)}$				$\sum$
	$C_1^{(i)}$	$C_2^{(i)}$	$\cdots$	$C_{K_i}^{(i)}$	
C_1	$n_{11}^{(i)}$	$n_{12}^{(i)}$	$\cdots$	$n_{1K_i}^{(i)}$	n_{1+}
C_2	$n_{21}^{(i)}$	$n_{22}^{(i)}$	$\cdots$	$n_{2K_i}^{(i)}$	n_{2+}
.	.	.	$\cdots$	.	.
C_K	$n_{K1}^{(i)}$	$n_{K2}^{(i)}$	$\cdots$	$n_{KK_i}^{(i)}$	n_{K+}
$\sum$	$n_{+1}^{(i)}$	$n_{+2}^{(i)}$	$\cdots$	$n_{+K_i}^{(i)}$	n

Another category of ensemble clustering is to employ a co-association matrix to summarize r basic partitions as follows:

$$\mathbf{S}(\mathbf{x}, \mathbf{y}) = \frac{1}{r} \sum_{i=1}^{r} \delta(\mathbf{H}^{(i)}(\mathbf{x}), \mathbf{H}^{(i)}(\mathbf{y})), \tag{3}$$

where $\delta(\cdot)$ denotes the Kronecker delta function, which returns 1 with two identical input values and returns 0 with different input values. We can regard $\mathbf{S}$ as a similarity matrix between a pair of instances, which simply counts the co-occurrence number in the same cluster in each basic partition. By this means, ensemble clustering problem is redefined as a classical graph partition problem, so that based on the co-association matrix $\mathbf{S}$, some clustering rules or loss functions can be derived in order to obtain the final consensus clustering.

Although ensemble clustering can be roughly generalized into two categories, based on co-association matrix or utility function, Liu et al. (2015a) built a connection between the methods based on co-association matrix and utility functions and pointed out the co-association matrix plays a determinative role in the success of ensemble clustering. Thus, here we focus on the methods based on co-association matrix. Next, we introduce the impact of the number of basic partitions by the following theorem.

Theorem 1 (Stableness Luo et al. (2011)) *For any $\epsilon > 0$, there exists a matrix $\boldsymbol{S}_0$, such that*

$$\lim_{r \to \infty} P(||\boldsymbol{S} - \boldsymbol{S}_0||_{\mathrm{F}}^2 > \epsilon) = 0,$$

where $||\cdot||_{\mathrm{F}}^2$ denotes the squared Frobenius norm with $||\boldsymbol{A}||_{\mathrm{F}}^2 = \mathrm{trace}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})$.

The assumptions behind Theorem 1 lie in (1) all the basic partitions come from the same data set, (2) the basic partitions are independently generated from the same strategy (for example, K-means with different cluster numbers), (3) p_{xy} is the probability of a pair of instances x and y belonging to the same cluster, $p_{xy} \sim Ber(p_{xy})$, where $Ber(p_{xy})$ is the Bernoulli distribution with the probability p_{xy}. We can see that these assumptions are reasonable for real-world applications. From the above theorem, we have the conclusion that although basic partitions might be greatly different from each other, the normalized co-association matrix becomes stable with the increase of the number of basic partitions r. From our previous experimental results (Liu et al. 2015b) in Fig. 2, it is easy to observe that with the increasing number of basic partitions, the performance of ensemble clustering goes up and becomes stable. However, the best number of basic partitions for a given data set is difficult to set. Too few basic partitions can not exert the capacity of ensemble clustering, while too many basic partitions lead to unnecessary computational resource waste. Therefore, fusing infinite basic partition is addressed in this paper, instead of answering the best number of basic partitions for a given data set. According to Theorem 1, we expect to fuse infinite basic partitions to maximize the capacity of ensemble clustering. Since we cannot generate infinite basic partitions, how to obtain a stable co-association matrix $\mathbf{S}$ and calculate $\mathbf{H}^*$ in an efficient way is highly needed,

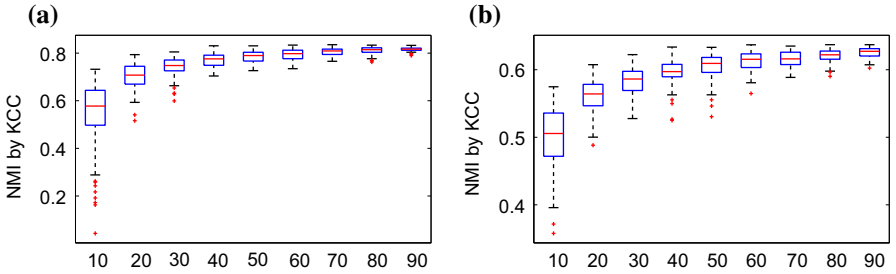

Fig. 2 Performance of different numbers of basic partitions via KCC on *mm* and *reviews* data sets. X-axis is the number of basic partitions. With increasing numbers of basic partitions, the performance goes up and the variance becomes narrow. Y-axis denotes the performance of KCC via Normalized Mutual Information (NMI). **a** *mm*. **b** *reviews*

which is also one of our motivations. In Sect. 4, we employ mDAE to equivalently obtain the "infinite" basic partitions and achieve the expectation of co-association matrix.

3.2 Marginalized denoising auto-encoder

An auto-encoder (Bengio 2009) is an artificial neural network used for unsupervised learning of efficient codings. The aim of an auto-encoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. Internally, it has a hidden layer that describes a code used to represent the input. The network may be viewed as consisting of two parts: an encoder function g and a decoder function h that produces a reconstruction.

To obtain a robust representation, denoising auto-encoders (DAEs) have been successfully used for a wide range of machine learning tasks (Ghifary et al. 2015; Carreira-Perpinn and Raziperchikolaei 2015). Usually DAE is implemented as a single-hidden-layer neural network where the input is the corrupted data by certain noises and the output is the clean data. The goal of DAE is to make the output to be as close as possible to the clean data $\mathbf{x}$ after learning. Usually a loss function $\ell(\mathbf{x}, \mathbf{y})$ is employed to measure the reconstruction error as follows:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}\ell(\mathbf{x}_i, f(\widetilde{\mathbf{x}}_i^j)), \tag{4}$$

where n is the number of data points, m is the times of corrupted data, $\mathbf{x}_i$ is the ith clean data point and $\widetilde{\mathbf{x}}_i^j$ is the ith data point in j-th corruption, and $f(\widetilde{\mathbf{x}}_i^j) = g \circ h(\widetilde{\mathbf{x}}_i^j)$ is the output of $\mathbf{x}_i^j$, where g and h are the encoder and decoder, respectively.

According to the linear or non-linear function of g and h, DAE can be roughly divided into the linear or non-linear version. For example, let $\mathbf{W}$ be the mapping function between the corrupted inputs $\widetilde{\mathbf{x}}$ and the clean data $\mathbf{x}$. Then we have the

following the squared reconstruction loss:

$$\frac{1}{2mn}\sum_{j=1}^{m}\sum_{i=1}^{n}||\mathbf{x}_i - \mathbf{W}\widetilde{\mathbf{x}}_i^j||^2. \tag{5}$$

After getting the one-layer hidden representation $\mathbf{z} = h(\widetilde{\mathbf{x}})$, we can continue to use this strategy by using $\mathbf{z}$ as the input to obtain deep representation for feature generation, which is called stacked denoising auto-encoder.

The disadvantage of DAE is to explicitly corrupt $\mathbf{x}$ by m times to get multiple $\widetilde{\mathbf{x}}$, which enlarges the training samples and increases the computational cost. Recently, Chen et al. (2014, 2012) proposed the marginalized denoising auto-encoder (mDAE) to overcome this challenge with $m \to \infty$ by taking use of the expected average loss as follows,

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{p(\widetilde{\mathbf{x}}_i|\mathbf{x}_i)}[\ell(\mathbf{x}_i, f(\widetilde{\mathbf{x}}_i))]. \tag{6}$$

For a long time, auto-encoder and its variants are regarded as a powerful feature generation tool. Actually it can also be used as an optimization tool. In the following, we will give another interpretation of auto-encoder.

3.3 Problem definition

Deep structure and clustering techniques are powerful tools for computer vision and data mining applications. Especially, ensemble clustering attracts a lot of attention due to its appealing performance. However, these two powerful tools are usually used separately. Notice that the performance of ensemble clustering heavily depends on the basic partitions. As mentioned before, co-association matrix $\mathbf{S}$ is the key factor for the ensemble clustering and with the increase of basic partitions, the co-association matrix becomes stable. According to Theorem 1, the capability of ensemble clustering goes to the upper bound with the number of basic partitions $r \to \infty$, Then we aim to seamlessly integrate deep concept and ensemble clustering in a one-step framework: *Can we fuse infinite basic partitions for ensemble clustering in a deep structure?*

4 Infinite ensemble clustering

Here we first uncover the connection between ensemble clustering and auto-encoder. Next, marginalized denoising auto-encoder is applied for the expectation of co-association matrix, and finally we propose our method and give the corresponding analysis.

4.1 From ensemble clustering to auto-encoder

It seems that there exists no explicit relationship between ensemble clustering and auto-encoder due to their respective tasks. The aim of ensemble clustering is to find

a cluster structure based on basic partitions, while auto-encoder is usually used for better feature generation. However, by taking a close look at the objective function in Eqs. 4 and 6, we find that auto-encoder can be regarded as an optimization method for minimizing the loss function.

Recalling that the goal of ensemble clustering is to find a single partition which agrees the basic ones as much as possible, we can understand it in the opposite way that the consensus partition has the minimum loss to present all the basic ones. After we summarize all the basic partitions into the co-association matrix $\mathbf{S}$, spectral clustering or some other graph partition algorithms can be conducted on the co-association matrix to obtain the final consensus result. Taking spectral clustering as an example, we aim to find an $n \times K$ low-dimensional space to represent the original input. Each column of low-dimensional matrix is a base for spanning the space. Then K-means can be run on that for the final partition. Similarly, the function of auto-encoder is also to learn a hidden representation of d dimensions by "carrying" as much as possible information of the input, where d is a user pre-defined parameter. Therefore, to some extent, spectral clustering and auto-encoder have the similar function to learn new representations according to minimizing certain objective function; the difference is that in spectral clustering, the dimension of new representation is K, while auto-encoder produces d dimensions. From this view, auto-encoder is more flexible than spectral clustering.

Therefore, we have another interpretation of auto-encoder, which not only can generate robust features, but also can be regarded as an optimization method for minimizing the loss function. By this means, we can feed the co-association matrix into auto-encoder to get the new representation, which has the similar function with spectral clustering, and run K-means on that to obtain the consensus clustering. For the efficiency issue, it is not a good choice to use auto-encoder on the ensemble clustering task due to the large space complexity of co-association matrix $O(n^2)$. We will address this issue in the next subsection.

4.2 The expectation of co-association matrix

According to Theorem 1, with the number of basic partitions going to infinity, the co-association matrix becomes stable. Before answering how to generate infinite ensemble members, we first solve how to increase the number of basic partitions given the limited ones. The naive way is to apply some generation strategy on the original data to produce more ensemble members. The disadvantages lie in two folds: (1) time consuming, (2) sometimes we only have the basic partitions, and the original data are not accessible. Therefore, without the original data, producing more basic partitions with the limited one is like a clone problem. However, simply duplicating the ensemble members does not work. Here we make several copies of basic partitions and corrupt them with erasing some labels in basic partitions to get new ones. By this means, we have extra incomplete basic partitions and Theorem 1 also holds for incomplete basic partitions (Liu et al. 2017a).

By this strategy, we just amply the size of ensemble members, which is still far from the infinity. To solve this challenge we use the expectation of co-association

matrix instead. Actually, $\mathbf{S}_0$ is just the expectation of $\mathbf{S}$, which means if we obtain the expectation of co-association matrix as an input for auto-encoder, our goal can be achieved. Since the expectation of co-association matrix cannot be obtained in advance, we intend to calculate it during the optimization.

Inspired by the marginalized Denoising auto-encoder (Chen et al. 2012), which involves the expectation of certain noises during the training, we corrupt the basic partitions and marginalize them for the expectation. We aim to fuse infinite incomplete basic partitions for ensemble clustering. Here the incomplete basic partitions can be obtained by removing labels from the complete ones, regarding as the partitions on subsets of the data set. For a set of basic partitions, we can randomly remove some labels in the basic partitions to generate the extra incomplete basic partitions. Such process is just the same as denoising auto-encoder with dropout noises. If we repeat the process infinite times, it leads to the marginalized denoising auto-encoder. If we take a look at Eq. 6, the function f can be linear or non-linear. In this paper, for efficiency issue we use the linear version of mDAE (Chen et al. 2012) since it has a closed-form formulation. By this means, our model links the marginalized denoising auto-encoder to ensemble clustering and leads to a natural integration.

4.3 Linear version of IEC

So far, we solve the infinite ensemble clustering problem with marginalized denoising auto-encoder. Before conducting experiments, we notice that the input of mDAE should be the instances with independent and identical distribution; however, the co-association matrix can be regarded as a graph, which disobeys this assumption. To solve this problem, we introduce a binary matrix $\mathbf{B}$.

Let $\mathbf{B}$ be a binary $n \times d$ matrix, where d equals $\sum_{i=1}^{r} K_i$ and each row $b(x)$ represents one data point as follows:

$$b(x) = \langle b(x)_1, \ldots, b(x)_r \rangle,\ b(x)_i = \langle b(x)_{i1}, \ldots, b(x)_{iK_i} \rangle,$$
$$b(x)_{ij} = \begin{cases} 1, & \text{if } \mathbf{H}^{(i)}(x) = j \\ 0, & \text{otherwise} \end{cases}.$$

We can see that the binary matrix $\mathbf{B}$ concatenates all the basic partitions with 1-of-K_i coding, where K_i is the number of clusters in $\mathbf{H}^{(i)}$. With the binary matrix $\mathbf{B}$, we have $\mathbf{B}\mathbf{B}^{\mathrm{T}} = r\mathbf{S}$. It indicates that the binary matrix $\mathbf{B}$ has the same information with the co-association matrix $\mathbf{S}$. Since $\mathbf{B}$ obeys the independent and identical distribution with respect to data points, we can put the binary matrix as input for marginalized denoising auto-encoder.

For linear version of IEC, the corresponding mapping for $\mathbf{W}$ between input and hidden representations is in closed-form (Chen et al. 2012):

$$\mathbf{W} = \mathbb{E}[\mathbf{P}]\mathbb{E}[\mathbf{Q}]^{-1}, \tag{7}$$

where $\mathbf{P} = \mathbf{B}\mathbf{B}^{\mathrm{T}} = r\mathbf{S}$ and $\mathbf{Q} = \mathbf{B}^{\mathrm{T}}\mathbf{B} = \mathbf{\Sigma}$. We add the constant 1 at the last column of $\mathbf{B}$ and corrupt it with p level drop-out noise. Let $\mathbf{q} = [1-p, \ldots, 1-p, 1] \in \mathbb{R}^{d+1}$,

Algorithm 1 The algorithm of Infinite Ensemble Clustering

Input: $\mathbf{H}^{(1)}, \ldots, \mathbf{H}^{(r)}$,: r basic partitions;
l: number of layers for mDAE;
p: noise level;
K: number of clusters.
Output: optimal $\mathbf{H}^*$;
1: Build the binary matrix $\mathbf{B}$;
2: Apply l layers stacked linear or non-linear mDAE with p noise level to get the mapping matrix $\mathbf{W}$;
3: Run K-means on $\mathbf{BW}^{\mathrm{T}}$ to get $\mathbf{H}^*$.

we have $\mathbb{E}[\mathbf{P}]_{ij} = \mathbf{\Sigma}_{ij}\mathbf{q}_j$ and $\mathbb{E}[\mathbf{Q}]_{ij} = \mathbf{\Sigma}_{ij}\mathbf{q}_i\tau(i, j, \mathbf{q}_j)$. Here $\tau(i, j, \mathbf{q}_j)$ returns 1 with $i = j$, and returns $\mathbf{q}_j$ with $i \neq j$. After getting the mapping matrix, $\mathbf{BW}^{\mathrm{T}}$ is used as the new representation. By this means, we can recursively apply marginalized denoising auto-encoder to obtain deep hidden representations. Finally, K-means is called to run on the hidden representations for the consensus partition. Since only r elements are non-zeros in each row of $\mathbf{B}$, it is very efficient to calculate $\mathbf{\Sigma}$. Moreover, $\mathbb{E}[\mathbf{P}]$ and $\mathbb{E}[\mathbf{Q}]$ are both $(d+1)\times(d+1)$ matrics. Finally, K-means is conducted on all the hidden representations. Therefore, our total time complexity is $O(ld^3 + IKnld)$, where l is the number of layers of mDAE, I is the iteration number in K-means, K is the number of clusters, and $d = \sum_{i=1}^{r} K_i \ll n$. This indicates our algorithm is linear to n, which can be applied for large-scale clustering. Since K-means is the core technique in IEC, the convergence is guaranteed according to the objective function value.

4.4 Non-linear version of IEC

For the non-linear version IEC, we follow the non-linear mDAE with second-order expansion and approximation (Chen et al. 2014) and have the following objective function:

$$\ell(\mathbf{x}, f(\mu_{\mathbf{x}})) + \frac{1}{2}\sum_{d=1}^{D}\sigma_{\mathbf{x}_d}^2\sum_{h=1}^{D_h}\frac{\partial^2\ell}{\partial z_h^2}\left(\frac{\partial z_h}{\partial \widetilde{\mathbf{x}}_d}\right)^2, \tag{8}$$

where l is the loss function in Eq. 5, $\mu_{\mathbf{x}} = \mathbf{x}$ is the mean of $\mathbf{x}$, $\sigma_{\mathbf{x}_d}^2 = \mathbf{x}^2 p/(1-p)$ is the variance of $\mathbf{x}$ in dth dimension with the noise level p, f is the sigmoid function, D and D_h are the dimensions of the input and hidden layers, respectively. The detailed understanding about the non-linear objective function can be found in Chen et al. (2014). A well-known framework Theano[1] is applied for the non-linear mDAE optimization.

Similarity, we feed the binary matrix $\mathbf{B}$ into the non-linear version of mDAE for the mapping function $\mathbf{W}$ and calculate the new presentation for clustering. The algorithm is summarized in Algorithm 1. Compared the two versions of IEC, the linear version has a closed-form expression with low computational cost, while the non-linear version is an approximation and the GPU is a must for the expensive computation.

[1] http://deeplearning.net/software/theano/.

5 Experimental results

In this section, we first introduce the experimental settings, then showcase the effectiveness and efficiency of IEC compared with the state-of-the-art deep clustering and ensemble clustering methods. Finally, some impact factors of IEC are thoroughly explored.

5.1 Experimental settings

Data Sets Thirteen real-world image data sets with true cluster labels are used for experiments. Table 2 shows their important characteristics, where #MinClass, #MaxClass, CV and Density denote the instance number of the smallest and biggest clusters, coefficient of variation statistic that characterizes the degree of class imbalance, and the ratio of non-zeros elements, respectively. In order to demonstrate the effectiveness of our IEC, we select the data sets with different levels of features, such as pixel, SURF and deep learning features. The first two are characters and digits data sets,[2] the middle ones are the objects and digits data sets[3,4] and the last four data sets are with the deep learning features.[5] In addition, these data sets contain different types of images, such as digits, characters, objects. Figure 3 shows some samples of these data sets.

Comparative algorithms To validate the effectiveness of the IEC, we compare it with several state-of-the-art methods in terms of deep clustering methods and ensemble clustering methods.

- K-means is the baseline method.
- MAEC (Chen et al. 2012) applies mDAE to get new representations and runs K-means on it to get the partition. Here MAEC1 uses the orginal features as the input, and MAEC2 uses the Laplace graph as the input.
- GEncoder (Tian et al. 2014) is short for GraphEncoder, which feeds the Laplace graph into the sparse auto-encoder to get new representations.
- DLC (Shao et al. 2015) jointly learns the feature transform function and discriminative codings in a deep mDAE structure.
- GCC (Strehl and Ghosh 2003) is a general concept of three benchmark ensemble clustering algorithms based on graph: CSPA, HGPA and MCLA, and returns the best result.
- HCC (Fred and Jain 2005) is an agglomerative hierarchical clustering algorithm based on the co-association matrix.
- KCC (Wu et al. 2015) is a K-means-based consensus clustering which transfers the ensemble clustering into a K-means optimization problem.
- SEC (Liu et al. 2017a) employs spectral clustering on co-association matrix and solves it by weighted K-means.

[2] http://archive.ics.uci.edu/ml.

[3] https://www.eecs.berkeley.edu/~jhoffman/domainadapt.

[4] http://www.cad.zju.edu.cn/home/dengcai.

[5] http://www.cs.dartmouth.edu/~chenfang.

Table 2 Experimental data sets

Data set	Type	Feature	#Instance	#Feature	#Class	#MinClass	#MaxClass	CV	Density
Letter	Character	Low-level	20,000	16	26	734	813	0.0301	0.9738
MNIST	Digit	Low-level	70,000	784	10	6313	7877	0.0570	0.1914
COIL100	Object	Middle-level	7200	1024	100	72	72	0.0000	1.0000
Amazon	Object	Middle-level	958	800	10	82	100	0.0592	0.1215
Caltech	Object	Middle-level	1123	800	10	85	151	0.2087	0.1638
Dslr	Object	Middle-level	157	800	10	8	24	0.3857	0.1369
Webcam	Object	Middle-level	295	800	10	21	43	0.1879	0.1289
ORL	Face	Middle-level	400	1024	40	10	10	0.0000	1.0000
USPS	Digit	Middle-level	9298	256	10	708	1553	0.2903	1.0000
Caltech101	Object	High-level	1415	4096	5	67	870	1.1801	1.0000
ImageNet	Object	High-level	7341	4096	5	910	2126	0.3072	1.0000
Sun09	Object	High-level	3238	4096	5	20	1264	0.8970	1.0000
VOC2007	Object	High-level	3376	4096	5	330	1499	0.7121	1.0000

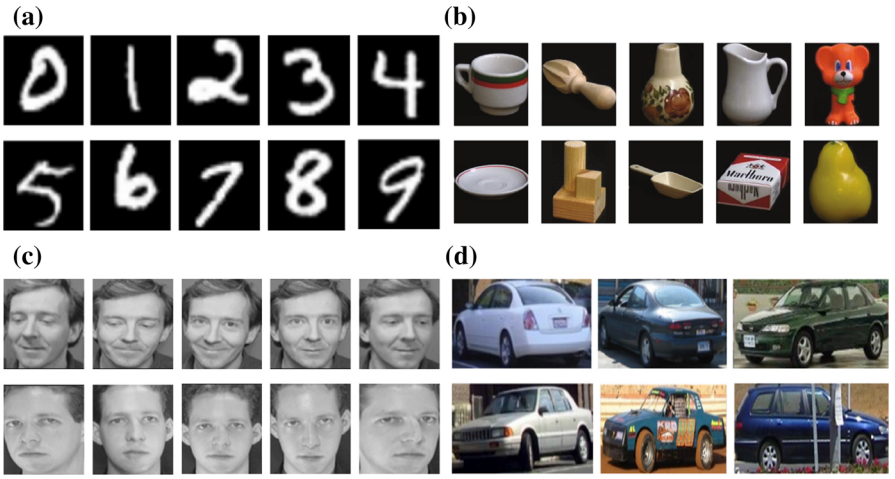

Fig. 3 Sample images. **a** *MNIST* is a 0–9 digits data sets in grey level, **b** *COIL100* is an object data set with 100 categories, **c** *ORL* contains faces of 40 people with different poses and **d** *Sun09* is an object data set with different types of cars

In the ensemble clustering framework, we employ Random Parameter Selection (RPS) strategy to generate basic partitions. Generally speaking, *k-means* is conducted on all features with different numbers of clusters, varying from K to $2K$. 100 basic partitions via RPS are produced to feed into the comparative methods. Note that we set 5 layers in our linear model and 1 layer for the non-linear model, and set the dimension of the hidden layers as the same with the one of input layer. For all clustering methods, we set K to be the true number of clusters and use the default parameter setting of each algorithm for a fair comparison. In the non-linear IEC, we set the learning rate to be 0.1, training iterations to be 300 suggested by Chen et al. (2014).

Validation metric Since the label information is available to these data sets, here we use two external metrics *accuracy* and *Normalized Mutual Information* (NMI) to measure the performance.

Accuracy is the average performance of label matching results between resulted labels and ground truth labels. Since clustering is an unsupervised process, we need to map the label order as a permutation operation and maximize this fraction as the final clustering accuracy. Given an instance $\mathbf{x}_i$, let r_i and s_i be the obtained cluster label and the label provided by the ground truth, respectively. The accuracy is defined as follows:

$$accurary = \frac{\sum_{i=1}^{n} \delta(s_i, map(r_i))}{n}, \tag{9}$$

where $\delta(x, y)$ denotes the Kronecker delta function that equals one if $x = y$ and equals zeros otherwise, and $map(r_i)$ is the permutation mapping function that maps each cluster label r_i to the ground truth s_i. The best mapping is applied by the Kuhn–Munkres algorithms.

Normalized Mutual Information measures the mutual information entropy between resulted cluster labels and ground truth labels, followed by a normalization operation

which guarantees that NMI ranges from 0 to 1. Here we use the variables in Table 1 to give the formulation of mutual information for two partitions $\mathbf{H}^*$ and $\mathbf{H}^{(i)}$.

$$MI(\mathbf{H}^*, \mathbf{H}^{(i)}) = \sum_{c_k \in \mathbf{H}^*, c_j \in \mathbf{H}^{(i)}} p_{kj}^{(i)} \log \frac{p_{kj}^{(i)}}{p_{k+} p_{+j}^{(i)}}. \tag{10}$$

In our experiments, the normalized mutual information NMI is used for evaluating the clustering performance, as follows:

$$NMI(\mathbf{H}^*, \mathbf{H}^{(i)}) = \frac{MI(\mathbf{H}^*, \mathbf{H}^{(i)})}{\max(H(\mathbf{H}^*), H(\mathbf{H}^{(i)}))}, \tag{11}$$

where $H(\mathbf{H}^*)$ and $H(\mathbf{H}^{(i)})$ are the entropies of $\mathbf{H}^*$, $\mathbf{H}^{(i)}$ respectively.

Note that accuracy and NMI are both positive measurements, which means the larger, the better.

Environment All the experiments except the non-linear IEC were run on a Windows standard platform of 64-bit edition, which has two Intel Core i7 3.4GHz CPUs and 32GB RAM. The non-linear IEC was conducted on a Ubuntu 14.04 of 64-bit edition with a NVIDA TITAN X GPU.

5.2 Clustering performance

Tables 3 and 4 show the clustering performance of different algorithms in terms of accuracy and NMI. The best results are highlighted in bold font. "N/A" denotes that there is no result due to out of memory. As can be seen from the tables, three observations are very clear. (1) In the deep clustering method, MAEC1 performs the best and the worst on *Amazon* and *COIL100*, respectively; on the contrary, MAEC2 gets reasonable result on *COIL100* but low quality on *Amazon*. Since we focus on the unsupervised clustering task, the default setting should be conducted for practical use. However, GEncoder suffers from the worst performance in all the comparative methods, even worse than K-means, although we try our best to tune the number of neurons in the hidden layers in a large range. The high computational cost prohibits MAEC2 and GEncoder from handling large-scale data sets. Since clustering belongs to the unsupervised learning, only relying on deep structure makes little effect to improve the performance. Instead DLC jointly learns the feature transform function and discriminative codings in a deep structure, which has the satisfactory results. (2) In most cases, ensemble clustering is superior to the baseline method, even better than deep clustering methods. The improvement is obvious when applying ensemble clustering methods on the data sets with high-level features, since high-level features have more structural information. However, ensemble methods do not work well on *SUN09*. One of the reasons might be the unbalanced class structure, which prevents the basic clustering algorithm K-means from uncovering the true structure and further harms the performance of ensemble methods. (3) Our method IEC gets the best results on most of 13 data sets. It is worthy to note that the improvements are over nearly 8%, 8% or

Table 3 Clustering performance of different algorithms measured by accuracy

Data sets	Baseline	Deep clustering method				Ensemble clustering method				
	K-means	MAEC1	MAEC2	GEncoder	DLC	GCC	HCC	KCC	SEC	IEC (ours)
Letter	0.2485	0.1163	N/A	N/A	**0.3087**	0.2598	0.2447	0.2461	0.2137	0.2633
MNIST	0.4493	0.3757	N/A	N/A	0.5498	0.5047	0.4458	0.6026	0.5687	**0.6086**
COIL100	0.5056	0.0124	0.5206	0.0103	0.5348	0.5382	0.5332	0.5032	0.5210	**0.5464**
Amazon	0.3309	**0.4395**	0.2443	0.2004	0.3653	0.3486	0.3069	0.3434	0.3424	0.3904
Caltech	0.2457	0.2787	0.2102	0.2333	0.2840	0.2289	0.2386	0.2618	0.2680	**0.2983**
Dslr	0.3631	0.4140	0.3185	0.2485	0.4267	0.4268	0.3949	0.4395	0.4395	**0.5159**
Webcam	0.3932	0.5085	0.3220	0.3430	**0.5119**	0.4305	0.3932	0.4203	0.4169	0.4983
ORL	0.5475	0.0450	0.3675	0.2050	0.5775	**0.6300**	0.6025	0.5450	0.5850	**0.6300**
USPS	0.6222	0.6290	0.4066	0.1676	0.6457	0.6211	0.6137	0.6857	0.6157	**0.7670**
Caltech101	0.6898	0.4311	0.5060	0.7753	0.7583	0.5152	0.7336	0.7611	0.9025	**0.9866**
ImageNet	0.6675	0.6601	0.3483	0.2892	0.6804	0.5765	0.7054	0.5986	0.6571	**0.7075**
Sun09	0.4360	0.4750	0.3696	0.3854	0.4829	0.4424	0.4235	0.4473	0.4732	**0.4899**
VOC2007	0.4565	0.4138	0.3874	0.4443	0.5130	0.5195	0.5044	**0.5364**	0.5124	0.5178

Table 4 Clustering performance of different algorithms measured by NMI

Data sets	Baseline	Deep clustering method				Ensemble clustering method				
	K-means	MAEC	MAEC2	GEncoder	DLC	GCC	HCC	KCC	SEC	IEC (ours)
Letter	0.3446	0.1946	N/A	N/A	**0.3977**	0.3444	0.3435	0.3469	0.3090	0.3453
MNIST	0.4542	0.3086	N/A	N/A	0.5195	0.4857	0.5396	0.4651	0.5157	**0.5420**
COIL100	0.7719	0.0769	0.7794	0.0924	0.7764	0.7725	0.7815	0.7761	0.7786	**0.7866**
Amazon	0.3057	**0.3588**	0.1982	0.0911	0.3001	0.2882	0.3062	0.2947	0.2595	0.3198
Caltech	0.2043	0.1862	0.1352	0.1132	0.2104	0.1774	0.2094	0.2031	0.1979	**0.2105**
Dslr	0.3766	0.4599	0.2900	0.1846	0.4614	0.4113	0.4776	0.4393	0.4756	**0.5147**
Webcam	0.4242	0.5269	0.2316	0.3661	**0.5280**	0.4344	0.4565	0.4502	0.4441	0.5201
ORL	0.7651	0.2302	0.6268	0.4431	0.7771	0.7987	0.7970	0.7767	0.7858	**0.8050**
USPS	0.6049	0.4722	0.4408	0.0141	0.5843	0.6219	0.5187	0.6363	0.5895	**0.6409**
Caltech101	0.7188	0.4980	0.5200	0.6922	0.7669	0.6536	0.7747	0.7881	0.8747	**0.9504**
ImageNet	0.4287	0.4827	0.1556	0.0064	0.4117	0.3902	**0.4375**	0.4366	0.4366	0.4358
Sun09	0.2014	**0.2787**	0.0576	0.0481	0.2315	0.2026	0.2091	0.1803	0.1927	0.2197
VOC2007	0.2697	0.2653	0.1118	0.1920	0.2651	0.2588	0.2564	0.2607	0.2511	**0.2719**

Table 5 Execution time of different ensemble clustering methods by second

Data sets	GCC	HCC	KCC	SEC	IEC (5 layers)
Letter	383.89	1717.88	11.39	8.35	55.46
MNIST	112.44	19,937.69	11.98	3.79	51.55
COIL100	21.27	170.02	4.99	3.09	14.93
Amazon	3.93	1.61	0.17	0.08	1.21
Caltech	3.55	2.12	0.23	0.11	1.43
Dslr	2.27	0.09	0.04	0.06	0.70
Webcam	2.09	0.14	0.04	0.05	0.90
ORL	6.81	0.04	0.21	0.21	14.11
USPS	7.66	160.41	1.73	0.53	5.48
Caltech101	1.21	1.68	0.15	0.09	0.53
ImageNet	3.83	52.47	1.40	0.32	1.76
Sun09	2.36	10.01	0.33	0.13	0.82
VOC2007	2.05	10.97	0.32	0.16	0.82

22% on *Dslr*, *USPS* and *Caltech101*, respectively, which are rare in clustering field. Usually the performance of ensemble clustering goes up with the increasing number of basic partitions. In order to show the best performance of the comparative ensemble clustering methods, we use 100 basic partitions. Here we can see that there still exists large space to improve via infinite ensemble members.

For efficiency, to make fair comparisons here we only report the execution time of ensemble clustering methods. Although additional time is needed for generating basic partitions, *k-means* and parallel computation make it quite efficient. Table 5 shows the average time of ten runs via these methods. GCC runs three methods on small data sets but runs two methods on large data sets, and HCC runs fast on data sets containing few instances but struggles as the number of instances increases due to its $O(n^3)$ time complexity. KCC, SEC and IEC are all K-means-based methods, which are much faster than other ensemble methods. Since our method only applies mDAE on basic partitions which has the closed-form solution and then runs K-means on the new representations, therefore IEC is suitable for large-scale image clustering.

In the end of this subsection, we compare the clustering performance of linear and non-linear IEC in Fig. 4. Here we employ 5-layer linear model and one-layer non-linear model. From Fig. 4, we can see that the non-linear model has 2–6% improvements on *Webcam* and *ORL* over the linear one in terms of accuracy. However, the non-linear model is an approximate calculation while linear model has closed-form representation. Besides, the non-linear model spends longer time to train even with the GPU accelerator. Taking the effectiveness and efficiency into comprehensive consideration, we choose the linear version of IEC as our default model for further analysis.

5.3 Inside IEC: factor exploration

Next we thoroughly explore the impact factors of IEC in terms of the number of layers, the generation strategy of basic partitions, the number of basic partitions, and the noise level, respectively.

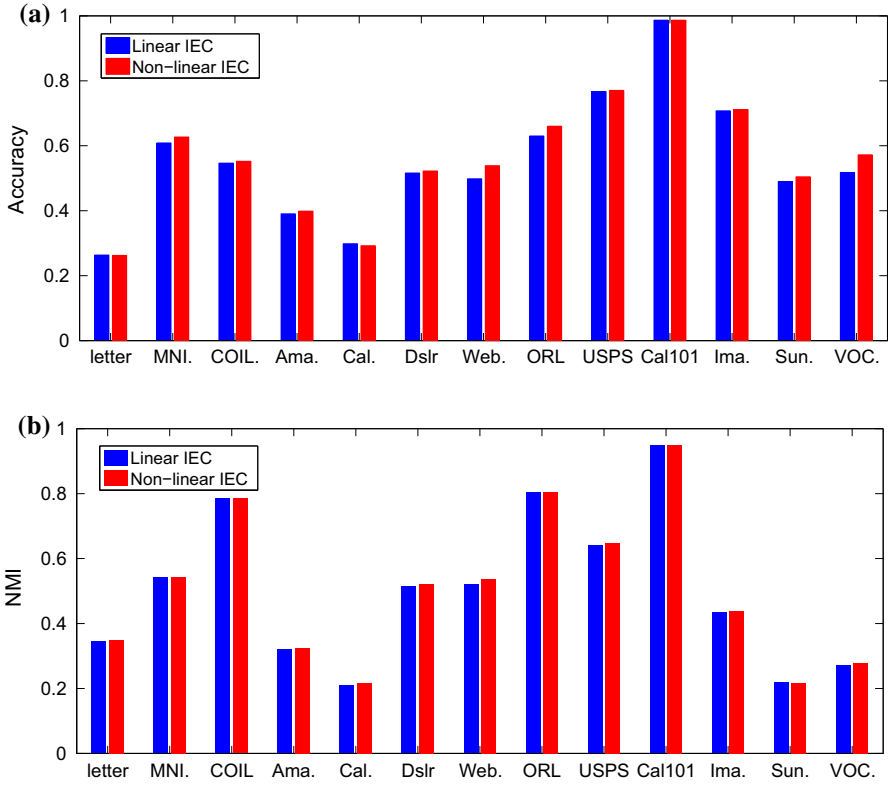

Fig. 4 Performance of linear and non-linear IEC on 13 data sets. **a** Accuracy. **b** NMI

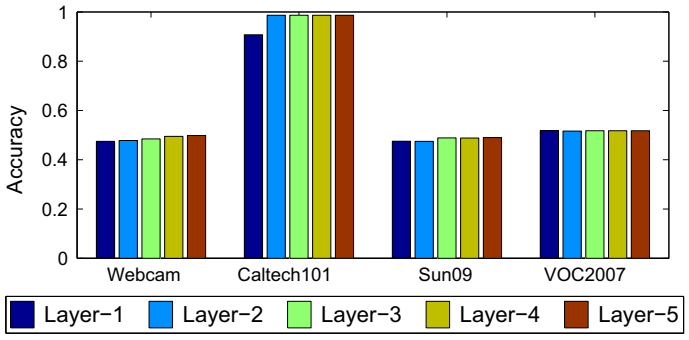

Fig. 5 Performance of IEC with different layers

Number of layers Since stacked mDEA is used to fuse infinite ensemble members, here we explore the impact of the number of layers. As can be seen in Fig. 5, the performance of IEC goes slightly up or keeps still with the increase of layers. Except that the second layer has large improvements over the first layer on *Caltech101*, IEC demonstrates the stable results on different layers, because only one-layer marginalized

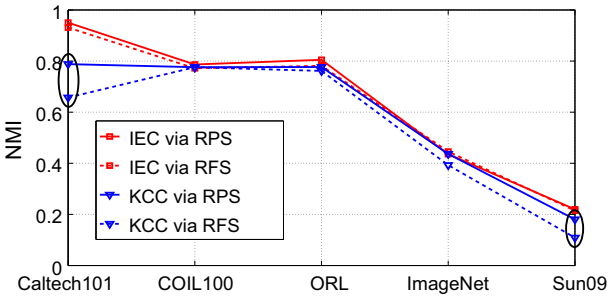

Fig. 6 Impact of basic partition generation strategies

denoising auto-encoder calculates the expectation of co-association matrix. The stable performance might result from the fully connected network structure in mDAE. Here the default value of the number of layers is set to be 5.

Generation strategy of basic partitions So far we rely solely on Random Parameter Selection (RPS) to generate basic partitions, with the number of clusters varying in $[K, 2K]$. In the following, we demonstrate whether the generation strategy will impact the performance of IEC. Here Random Feature Selection (RFS) is proposed as a comparison, which still uses *k-means* as the basic clustering algorithm with randomly selecting 50% original features to obtain 100 basic partitions. Figure 6 demonstrates the performance of KCC and IEC via RPS and RFS on 5 data sets. As we can see that, IEC exceeds KCC in most cases of RPS and RFS. When we take a close look, the performance of IEC via RPS and RFS is almost the same, while KCC produces large gaps between RPS and RFS on *Caltech101* and *Sun09* (See the ellipses). This indicates that although the generation of basic partitions is of high importance to the success of ensemble clustering, IEC helps to alleviate the impact.

Number of basic partitions The key problem of this paper is to use limited basic partitions to achieve the goal of fusing infinite ensemble members. Here we discuss the impact of the number of basic partitions to ensemble clustering. Figure 7 shows the performance of 4 ensemble clustering methods on four data sets. Generally speaking, the performance of HCC, KCC and GCC goes up with the increase of the number of basic partitions and becomes stable when enough basic partitions are given, which is consistent with Theorem 1. It is worthy to note that for large-scale data sets, generating basic partition suffers from high time complexity even with ensemble process. Thus, it is appealing that IEC achieves the high performance with limited basic partitions and is suitable for large-scale data clustering.

Noise level The core idea of this paper is to obtain the expectation of co-association matrix via adding the drop-out noise. Figure 8 shows the results of IEC with different noise levels on four data sets. As can be seen, the drop-out noise probability does affect the results even after marginalization. IEC gains the improvements with increasing noise levels; when the noise level is large enough, i.e., 0.1, IEC becomes stable. Note that if we set the noise level to zero, IEC would equivalently degrade into KCC. Thus, the dropout noise plays a key part in IEC, which is necessary and crucial to the improvement over KCC.

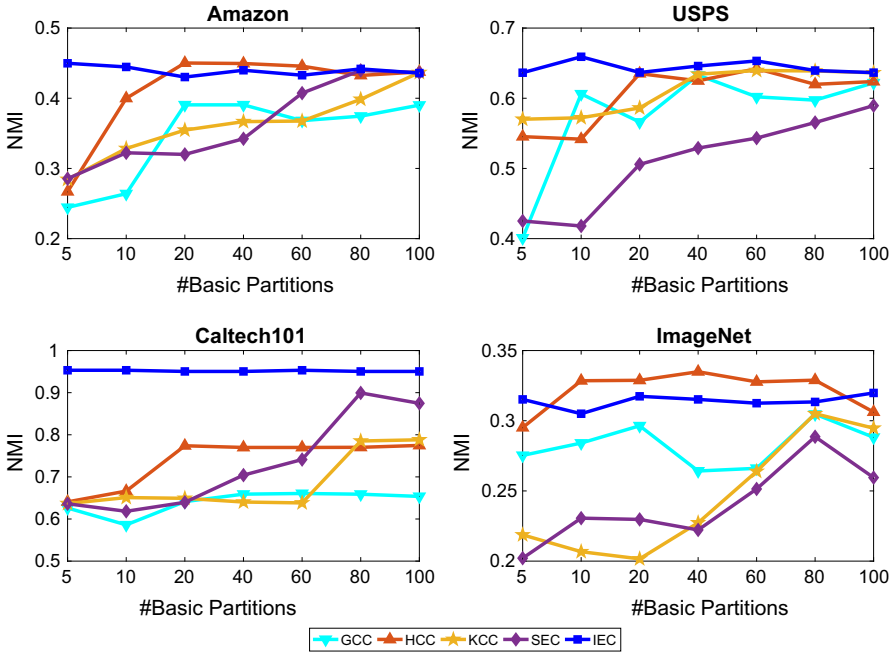

Fig. 7 Impact of the number of basic partitions via different ensemble methods on four date sets

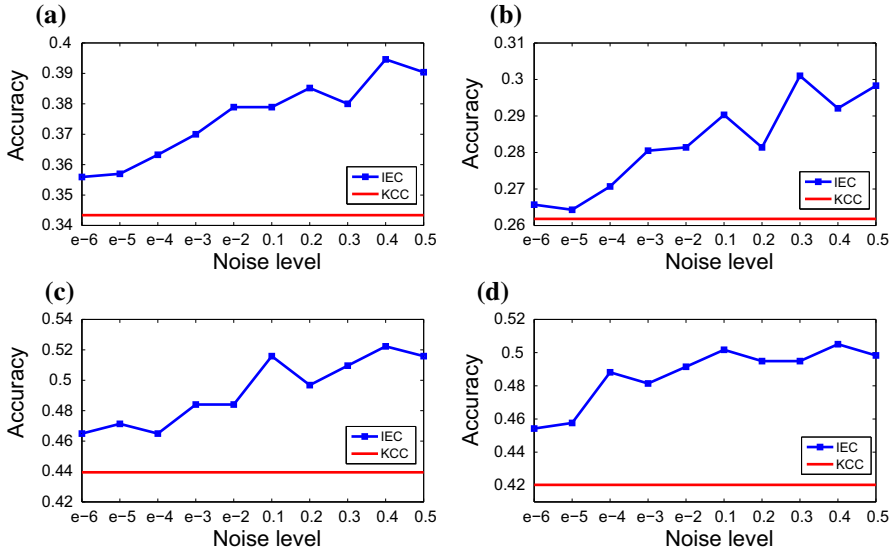

Fig. 8 Performance of IEC with different noise levels on four data sets. Y-axis denotes the noise level with a nonlinear scale. **a** Amazon. **b** Caltech. **c** Dslr. **d** Webcam

Table 6 Some key characteristics of 13 data sets from TCGA

Database	#Class	Protein (190) #Instance	miRNA (1046) #Instance	mRNA (20531) #Instance	SCNA (24952) #Instance
BLCA	4	127	328	326	330
BRCA	4	742	728	1065	1067
COAD	4	330	242	292	450
HNSC	4	212	471	502	508
KIRC	4	454	247	523	524
LGG	3	258	441	445	443
LUAD	3	237	441	496	502
LUSC	4	195	317	476	479
OV	4	408	474	262	575
PRAD	7	161	414	418	418
SCKM	4	206	416	436	438
THCA	5	370	503	502	503
UCEC	4	394	393	162	527

The numbers of clusters are obtained from the original papers which publish the data sets

6 Application on pan-omics gene expression analysis

With the rapid development of techniques, it becomes much more easier to collect diverse and rich molecular data types from genome to transcriptome, proteome, and epigenome (Uhlen et al. 2016; Zhu et al. 2015). The pan-omics gene expressions, which is also known as multi-view data, provide great opportunities to characterize human pathologies and disease subtypes, identify driver genes and pathways, and nominate drug targets for precision medicine (Biankin et al. 2015; Bolouri et al. 2016; Liu et al. 2017b). Clustering, an unsupervised exploratory analysis, has been widely used for patient stratification or disease subtyping (Chen et al. 2013; Chang et al. 2005). To fully demonstrate the effectiveness of IEC in real-world applications, here we employ IEC for pan-omics gene analysis and compare it with several widely used clustering methods in the biological domain. In the following, we introduce the gene expression data sets and the experimental setting, evaluate the performance of different clustering methods by survival analyses and finally apply IEC for the missing pan-omics gene expression analysis.

6.1 Experimental setting

Data sets Thirteen pan-omics gene expression data sets with survival information from TCGA[6] are used for evaluating the performance of patient stratification. These data sets denote the gene expression of the patients with 13 major cancer types and each data set contains 4 different types of molecular data, including protein expres-

[6] https://cancergenome.nih.gov/.

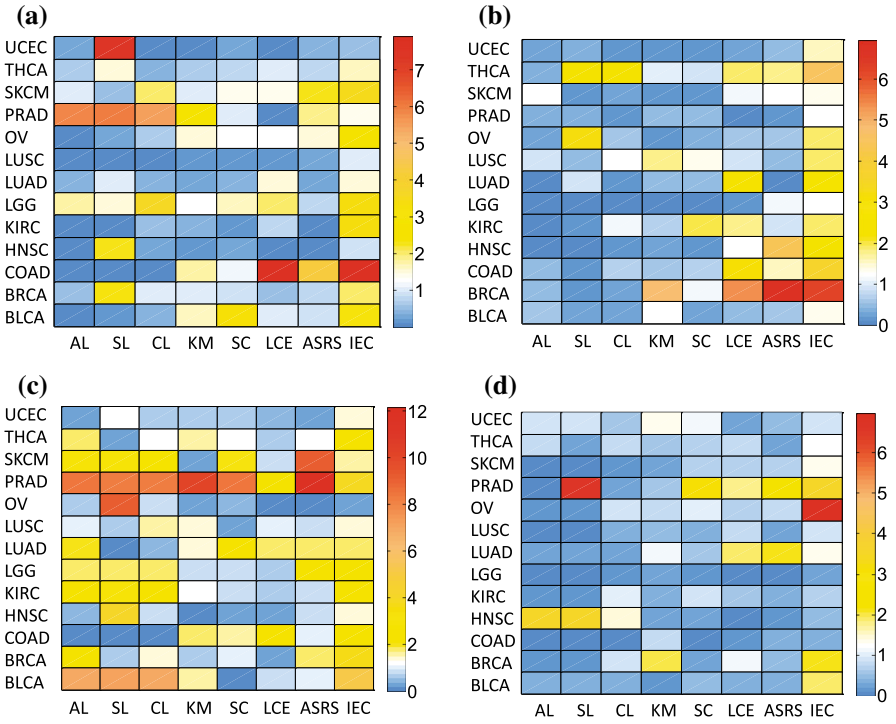

Fig. 9 Survival analysis of different clustering methods in the one-omics setting. X-axis denotes different clustering methods, and Y-axis denotes different data sets. The *color* represents the $-\log(p \text{ value})$ of the survival analysis. The larger value indicates the more significant difference among different subgourps according to the partition by different clustering methods. For better visualization, we set the *white color* to be $-\log(0.05)$ so that the warm colors mean the pass of hypothesis test and the cold colors mean the failure of hypothesis test. The detailed numbers of p value can be found in Tables 7, 8, 9 and 10 in "Appendix", where the p value is in bold with passed survival hypothesis test. **a** protein. **b** miRNA. **c** mRNA. **d** SCNA (Color figure online)

sion, microRNA (miRNA) expression, mRNA expression (RNA-seq V2) and somatic copy number alterations (SCNAs). These cancer types include bladder urothelial carcinoma (BLCA), breast cancer carcinoma (BRCA), colon adenocarcinoma (COAD), head and neck squamous cell carcinoma (HNSC), kidney renal clear cell carcinoma (KIRC), acute myeloid leukemia (LAML), brain lower grade glioma (LGG), lung adenocarcinoma (LUAD), lung squamous cell carcinoma (LUSC), ovarian serous cystadenocarcinoma (OV), prostate adenocarcinoma (PRAD), skin cutaneous melanoma (SKCM), thyroid carcinoma (THCA), and uterine corpus endometrial carcinoma (UCEC). Table 6 shows some key characteristic of 13 data sets from TCGA. These four types of molecular data have different dimensions. For example, the protein expression has 190 dimensions, miRNA expression has 1046 dimensions, mRNA expression has 20,531 dimensions and SCNA has 24,952 dimensions. It is also worthy to note that the numbers of subjects on different molecular types on each data set are different due to the missing data or device failure.

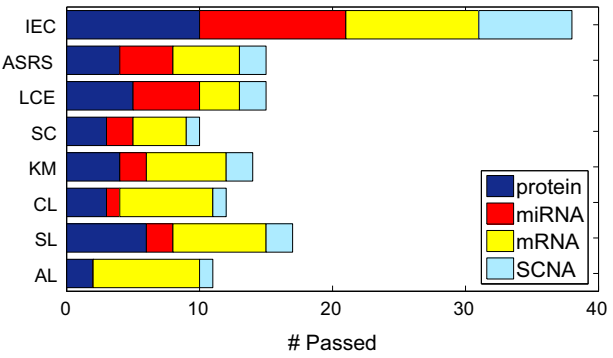

Fig. 10 Number of passed hypothesis tests of different clustering methods

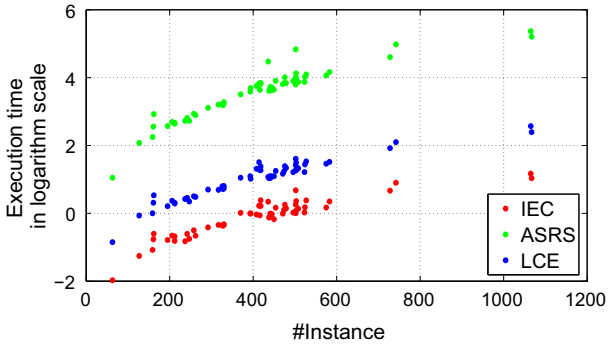

Fig. 11 Execution time in logarithm scale of different ensemble clustering methods on 13 cancer data sets with 4 different molecular types

Comparative algorithms Since we focus on the gene expression analysis, some widely used clustering methods in biological domain are chosen for comparison in terms of traditional clustering and ensemble clustering methods.

- Agglomerative hierarchical clustering, K-means (KM) and spectral clustering (SC) are baseline methods. Here agglomerative hierarchical clustering with the group-linkage, single-linkage and complete-linkage denotes as AL, SL and CL.
- LCE (Iam-on et al. 2010) is a link-based cluster ensemble method, which accesses the similarity between two clusters, builds refined co-association matrix, and applies spectral clustering for the final partition.
- ARSR (Galdi et al. 2014) is short for Approximated Sim-Rank Similarity (ASRS) matrix, which is based on a bipartite graph representation of the cluster ensemble in which vertices represent both clusters and data points and edges connect data points to the clusters to which they belong.

Similar to the setting in Sect. 5, we still use Random Parameter Selection (RPS) strategy with the number of clusters varying from K to $2K$ to generate 100 basic partitions for the ensemble clustering methods LCE, ARSR and IEC. And for all clustering methods, we set K to be the true number of clusters for fair comparisons.

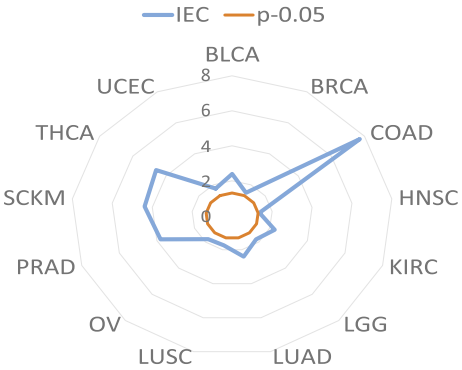

Fig. 12 Survival analysis of IEC in the pan-omics setting. The value denotes the $-\log(p$ value) of the survival analysis. The detailed numbers of p value can be found in Table 11 in "Appendix"

Validation metric For these 13 molecular data without label information, we employ survival analyses to evaluate the performance of different clustering methods. Survival analysis considers the expected duration of time until one or more events happen, such as death, disease occurrence, or other experience of interest (Miller and Rupert 2011). Based on the partition obtained by different clustering methods, we divide the objects or patients into several different groups. Then survival analyses are conducted to calculate whether these groups have significant differences by log-rank test.

The log-rank test is a hypothesis test to compare the survival distributions of two or more groups. The null hypothesis that every group has the same or similar survival function. The expected number of subjects surviving at each time point in each group is adjusted for the number of subjects at risk in the groups at each event time. The log-rank test determines if the observed number of events in each group is significantly different from the expected number. The log-rank statistic has an asymptotic chi-squared distribution with one degree of freedom, and the p value is calculated using the chi-squared distribution. When the p value is smaller than 0.05, it typically indicates that those groups differ significantly in survival times. Here *survival* library in R package[7] is used for the log-rank test.

Environment All the experiments were run on a Windows standard platform of 64-bit edition, which has two Intel Core i7 3.4GHz CPUs and 32GB RAM.

6.2 One-omics gene expression evaluation

Since these 13 data sets have different numbers of instances within four different types, we first evaluate these widely used clustering methods in biological domain and IEC in the one-omics setting. That means that we treat these 13 data sets with four modular types as 52 independent data sets, and then run clustering methods and evaluate the performance of survival analysis by p value. For the ensemble methods, LCE, ARSR and IEC, RPS strategy is employed to generate 100 basic partitions.

Figure 9 shows the survival analysis performance of different clustering methods on one-omics setting, where colors denote the $-\log(p$ value) of the survival analysis.

[7] https://cran.r-project.org/web/packages/survival/index.html.

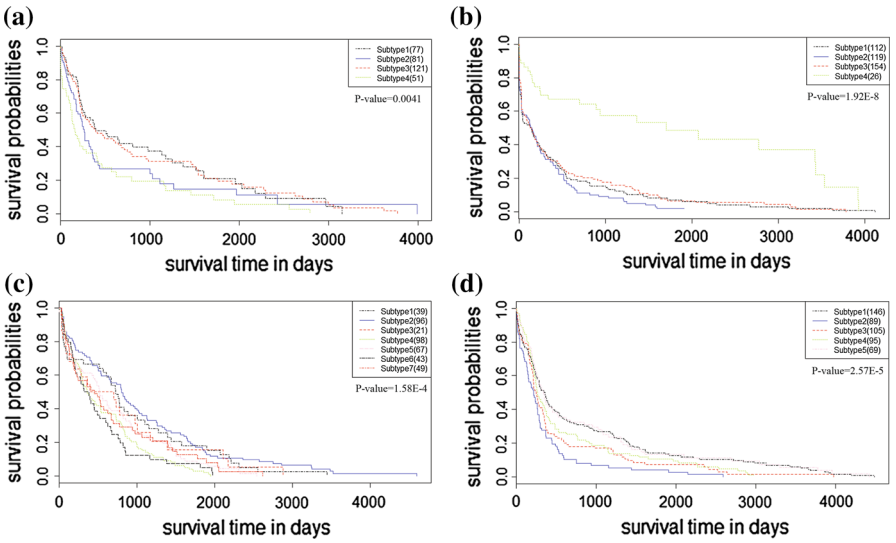

Fig. 13 Survival curves of four cancer data sets by IEC. **a** BLCA. **b** COAD. **c** PRAD. **d** THCA

For better comparison, we set $-\log(0.05)$ as the white color so that the warm colors (yellow, orange and red) mean the pass of hypothesis test and the cold colors (blue) mean the failure of hypothesis test. From this figure, we have three observations. (1) Generally speaking, traditional clustering methods, such as agglomerative hierarchical clustering, K-means and spectral clustering deliver poor performance, especially AL has no pass on the miRNA modular data. Compared with these traditional clustering methods, ensemble methods fuse several diverse basic partitions and enjoy more passes on these data sets. (2) IEC shows the obvious advantages over other competitive methods with more bright area and more passes of hypothesis tests. On KIRC data set with protein expression, BLCA and UCEC data sets with miRNA expression, BLCA, OV, SCKM and THCA data sets with SCNA, only IEC passes the hypothesis tests. Figure 10 shows the number of passed hypothesis tests of these clustering methods on four different modular types. On these 52 independent data sets, IEC has 38 passes hypothesis tests with the passing rate over 73.0%, while the second best method only has the 32.7% passing rate. Such evidence gives the strong support that IEC is a promising tool for gene expression analysis over rivals. The benefits of IEC lie in two aspects. One is that IEC is an ensemble clustering method, which incorporates several basic partitions in a high-level fusion fashion; the other is that the latent infinite partitions make the results resist to noises. (3) Different types of molecular data have different capacities to uncover the cluster structure for survival analysis. For example, most of methods pass the hypothesis tests on mRNA, while few of them pass the hypothesis tests on SCNA. For a certain data set or cancer, we cannot pre-know what is the best molecular data type for passing the hypothesis test of survival analysis. This leads to the pan-omics gene expression evaluation in the next subsection.

Figure 11 shows the execution time in logarithm scale of LCE, ASRS and IEC on 13 cancer data sets with 4 different molecular types. Since IEC enjoys the roughly

linear time complexity to the number of instance, IEC has significant advantages over LCE and ASRS in terms of efficiency. For example, IEC is 2–4 times faster than LCE and 20–66 times faster than ASRS. This indicates that IEC is a suitable ensemble clustering tool for real-world applications in large-scale.

6.3 Pan-omics gene expression evaluation

In this subsection, we continue to evaluate the performance of IEC with missing values. In the pan-omics application, it is quite normal to collect the data with missing values or missing instances. For example, these 13 cancer data sets in Table 6 have different numbers of instances in different types. To handle the missing data, a naive way is to remove the instances with missing values so that a smaller complete data set can be achieved. However, this way is a kind of waster since collecting data is very expensive especially in biology domain. Although there exist missing values in the pan-omics gene expression in Table 6, we can still employ the IEC to finish the partition.

To achieve this, we generate 25 incomplete basic partitions for each one-omics gene expression by running K-means on incomplete data sets and the missing instances are labelled as zeros. Then IEC is applied to fuse 100 incomplete basic partitions into the consensus one. Figure 12 shows the survival analysis of IEC on 13 pan-omics data sets. We can see that by integrating pan-omics gene expression, IEC passes all the hypothesis tests on 13 cancer data sets. Recall that in the one-omics setting, IEC fails the hypothesis tests on some data sets. This indicates that even incomplete pan-omics gene expression is conductive to uncover the meaningful structure. Figure 13 shows the survival curves of four cancer data sets by IEC.

7 Conclusion

In this paper, we proposed a novel ensemble clustering algorithm Infinite Ensemble Clustering (IEC) to fuse infinite basic partitions. Generally speaking, we built a connection between ensemble clustering and auto-encoder, and applied marginalized denoising auto-encoder to fuse infinite incomplete basic partitions. The linear and non-linear versions of IEC were provided. Extensive experiments on 13 data sets with different levels of features demonstrated our method IEC had promising performance over the state-of-the-art deep clustering and ensemble clustering methods; besides, we thoroughly explored the impact factors of IEC in terms of the number of layers, the generation strategy of basic partitions, the number of basic partitions, and the noise level to show the robustness of our method. Finally, we employed 13 pan-omics gene expression cancer data sets to illustrate the effectiveness of IEC in the real-world applications.

Acknowledgements This work is supported in part by the NSF IIS award 1651902, ONR Young Investigator Award N00014-14-1-0484, and U.S. Army Research Office Young Investigator Award W911NF-14-1-0218.

Appendix

See Tables 7, 8, 9, 10 and 11.

Table 7 Survival analysis of different clustering algorithms on protein expression data

Data set	AL	SL	CL	KM	SC	LCE	ASRS	IEC
BLCA	0.8400	0.6230	0.3210	**0.0241**	**0.0005**	0.0881	0.1030	**0.0008**
BRCA	0.2660	**0.0008**	0.0988	0.0997	0.1130	0.3060	0.1460	**0.0092**
COAD	0.8750	0.9530	0.8430	**0.0157**	0.0738	**1.20E−8**	**4.82E−5**	**1.50E−8**
HNSC	0.7540	**0.0050**	0.5520	0.7340	0.5110	0.9840	0.5960	0.1340
KIRC	0.7640	0.9140	0.2460	0.4120	0.6560	0.1680	0.7590	**0.0003**
LGG	**0.0182**	**0.0305**	**0.0002**	0.0563	0.0198	**0.0094**	0.1780	**0.0004**
LUAD	0.3730	0.8350	0.3220	0.4790	0.3990	**0.0293**	0.5070	**0.0267**
LUSC	0.9050	0.9290	0.9340	0.6670	0.6050	0.6550	0.5420	0.0982
OV	0.8090	0.5450	0.1900	**0.0275**	**0.0446**	**0.0485**	**0.0327**	**0.0026**
PRAD	**1.19E−6**	**9.78E−7**	**3.16E−6**	**0.0011**	0.0918	0.8140	**0.0124**	**0.0041**
SKCM	0.0848	0.2860	**0.0100**	0.0929	**0.0411**	**0.0381**	**0.0059**	**3.00E−4**
THCA	0.2380	**0.0255**	0.3470	0.1910	0.1480	0.0799	0.1370	**0.0187**
UCEC	0.4530	**3.00E−8**	0.9860	0.9860	0.4550	0.8450	0.3700	0.2930
#Significance	2	6	3	4	3	5	4	**10**

The values in the table represent the p value of log-rank test

Table 8 Survival analysis of different clustering algorithms on miRNA expression data

Data set	AL	SL	CL	KM	SC	LCE	ASRS	IEC
BLCA	0.2780	0.5880	0.5940	0.0616	0.5620	0.3410	0.2400	**0.0490**
BRCA	0.3110	0.6350	0.5410	**1.53E−5**	0.0717	**3.97E−6**	**1.12E−7**	**5.15E−7**
COAD	0.3290	0.6430	0.2070	0.2290	0.1960	**8.88E−4**	**0.0246**	**0.0002**
HNSC	0.8900	0.8820	0.7650	0.5760	0.6770	0.0605	**4.45E−5**	**0.0048**
KIRC	0.7970	0.6420	0.0692	0.2180	**0.0093**	**0.0180**	0.1090	**0.0140**
LGG	0.8820	0.9640	0.8940	0.9850	0.9000	0.7450	0.0640	0.0550
LUAD	0.8350	0.1200	0.7410	0.2870	0.3580	**0.0038**	0.8260	**0.0020**
LUSC	0.1060	0.3450	0.0565	**0.0152**	**0.0394**	0.1310	0.3120	**0.0136**
OV	0.5540	**0.0007**	0.2410	0.6290	0.4190	0.2340	0.2340	**0.0125**
PRAD	0.4570	0.4250	0.6500	0.3330	0.3200	0.8720	0.6270	0.0519
SKCM	0.0619	0.6870	0.4920	0.6390	0.6940	0.0663	0.0575	**0.0440**
THCA	0.4660	**0.0064**	**0.0053**	0.0892	0.1100	**0.0119**	**0.0157**	**2.95E−5**
UCEC	0.5280	0.4570	0.6290	0.6870	0.6080	0.5530	0.3520	**0.0258**
#Significance	0	2	1	2	2	5	4	**11**

The values in the table represent the p value of log-rank test

Table 9 Survival analysis of different clustering algorithms on mRNA expression data

Data set	AL	SL	CL	KM	SC	LCE	ASRS	IEC
BLCA	**1.06E−7**	**8.88E−8**	**1.06E−7**	**0.0258**	0.6860	0.1280	0.0938	**5.53E−6**
BRCA	**5.35E−3**	0.1740	**0.0401**	0.1760	0.0840	0.5980	**0.0155**	**0.0002**
COAD	0.8930	0.8960	0.8720	**0.0163**	**0.0296**	**0.0048**	0.0743	**0.0028**
HNSC	0.2950	**8.53E−5**	0.1350	0.7470	0.5440	0.6290	0.1440	0.0392
KIRC	**0.0025**	**0.0012**	**0.0036**	0.0612	0.1450	0.2420	0.1550	**0.0038**
LGG	**0.0156**	**0.0156**	**0.0155**	0.1270	0.1230	0.2650	**0.0023**	**0.0055**
LUAD	**0.0109**	0.8290	0.3190	**0.0429**	**0.0034**	**0.0189**	**0.0157**	**0.0165**
LUSC	0.0990	0.2100	**0.0241**	**0.0355**	0.4740	0.0769	0.1360	**0.0371**
OV	0.2210	**4.92E−10**	0.1700	0.6360	0.3780	0.8720	0.7660	0.4530
PRAD	**4.29E−9**	**4.49E−9**	**5.88E−9**	**7.29E−11**	**4.10E−9**	**0.0070**	**6.75E−13**	**0.0001**
SKCM	**0.0012**	**0.0012**	**0.0015**	0.5230	**0.0006**	0.1350	**5.91E−10**	**0.0204**
THCA	**0.0147**	0.5650	0.0713	**0.0244**	0.0561	0.2380	0.0710	**0.0048**
UCEC	0.5790	0.0594	0.1930	0.1850	0.2460	0.3670	0.4890	0.0437
#Significance	8	7	7	6	4	3	5	**10**

The values in the table represent the *p* value of log-rank test

Table 10 Survival analysis of different clustering algorithms on SCNA data

Data set	AL	SL	CL	KM	SC	LCE	ASRS	IEC
BLCA	0.3710	0.3710	0.3810	0.6340	0.3580	0.4340	0.3800	**0.0120**
BRCA	0.6540	0.6540	0.1160	**0.0090**	0.4790	0.0798	0.3520	**0.0073**
COAD	0.9320	0.9320	0.9010	0.1600	0.7920	0.7670	0.4660	0.3900
HNSC	**0.0003**	**0.0003**	**0.0380**	0.5280	0.5730	0.8280	0.7710	0.2940
KIRC	0.6580	0.7510	0.0929	0.4390	0.1060	0.2690	0.3710	0.2210
LGG	0.8800	0.9950	0.6430	0.5710	0.6130	0.8750	0.9740	0.4930
LUAD	0.5420	0.5420	0.5880	0.0763	0.2390	**0.0121**	**0.0080**	**0.0456**
LUSC	0.8900	0.8190	0.3870	0.3560	0.3810	0.1710	0.5540	0.1290
OV	0.7500	0.7500	0.1270	0.1710	0.0904	0.1730	0.1380	**1.08E−7**
PRAD	0.8410	**2.40E−7**	0.5060	0.2640	**0.0008**	**0.0160**	**0.0046**	**0.0003**
SKCM	0.8730	0.8140	0.6790	0.5660	0.1970	0.2210	0.2040	**0.0444**
THCA	0.1530	0.5180	0.1440	0.2670	0.1960	0.1360	0.5440	**0.0496**
UCEC	0.1100	0.1100	0.2310	**0.0484**	0.0673	0.4860	0.3450	0.1210
#Significance	1	2	1	2	1	2	2	**7**

The values in the table represent the *p* value of log-rank test

Table 11 Survival analysis of IEC on pan-omics gene expression

BLCA	0.0041	BLCA	0.0327	COAD	1.92E−8	HNSC	0.0423	KIRC	0.0054
LGG	0.0054	LUAD	0.0160	LUSC	0.0040	OV	0.0163	PRAD	1.58E−4
SKCM	4.14E−4	THCA	2.57E−5	UCEC	0.0178				

The values in the table represent the *p* value of log-rank test

References

Ayad H, Kamel M (2008) Cumulative voting consensus method for partitions with variable number of clusters. IEEE Trans Pattern Anal Mach Intell 30(1):160–173

Bengio Y (2009) Learning deep architectures for AI. Found Trends® Mach Learn 2(1):1–127

Bengio Y, Lamblin P, Popovici D, Larochelle H et al (2007) Greedy layer-wise training of deep networks. Advances in neural information processing systems (NIPS-06), pp 153–160

Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell 35(8):1798–1828

Biankin AV, Piantadosi S, Hollingsworth SJ (2015) Patient-centric trials for therapeutic development in precision oncology. Nature 526(7573):361–370

Bolouri H, Zhao LP, Holland EC (2016) Big data visualization identifies the multidimensional molecular landscape of human gliomas. In: Proceedings of the national academy of sciences

Carreira-Perpinn M, Raziperchikolaei R (2015) Hashing with binary autoencoders. In: Proceedings of computer vision and pattern recognition

Chang HY, Nuyten DS, Sneddon JB, Hastie T, Tibshirani R, Sorlie T et al (2005) Robustness, scalability, and integration of a wound-response gene expression signature in predicting breast cancer survival. In: Proceedings of the national academy of sciences

Chen M, Xu Z, Weinberger K, Sha F (2012) Marginalized stacked denoising autoencoders for domain adaptation. In: Proceedings of international conference on machine learning

Chen G, Sullivan PF, Kosorok MR (2013) Biclustering with heterogeneous variance. In: Proceedings of the national academy of sciences

Chen M, Weinberger K, Sha F, Bengio Y (2014) Marginalized denoising autoencoders for nonlinear representation. In: Proceedings of international conference on machine learning

Ding Z, Shao M, Fu Y (2015) Deep low-rank coding for transfer learning. In: Proceedings of AAAI conference on artificial intelligence

Domeniconi C, Al-Razgan M (2009) Weighted cluster ensembles: methods and analysis. ACM Trans Knowl Discov Data 2(4):17

Fred ALN, Jain AK (2005) Combining multiple clusterings using evidence accumulation. IEEE Trans Pattern Anal Mach Intell 27(6):835–850

Galdi P, Napolitano F, Tagliaferri R (2014) Consensus clustering in gene expression. In: International meeting on computational intelligence methods for bioinformatics and biostatistics

Ghifary M, Kleijn W, Zhang M, Balduzzi D (2015) Domain generalization for object recognition with multi-task autoencoders. In: Proceedings of international conference on computer vision

Hinton GE, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554

Huang P, Huang Y, Wang W, Wang L (2014) Deep embedding network for clustering. In: Proceedings of international conference on pattern recognition

Iam-on N, Boongoen T, Garrett S (2010) Lce: a link-based cluster ensemble method for improved gene expression data analysis. Bioinformatics 26(12):1513–1519

Kan M, Shan S, Chang H, Xilin C (2014) Stacked progressive auto-encoders (SPAE) for face recognition across poses. In: Proceedings of computer vision and pattern recognition

Li T, Chris D, Jordan M (2007) Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In: Proceedings of international conference on data mining

Li S, Jiang Y, Zhou Z (2014) Partial multi-view clustering. In: Proceedings of AAAI conference on artificial intelligence

Liu H, Liu T, Wu J, Tao D, Fu Y (2015a) Spectral ensemble clustering. In: Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining

Liu H, Wu J, Tao D, Zhang Y, Fu. Dias Y (2015b) A disassemble-assemble framework for highly sparse text clustering. In: Proceedings of SIAM international conference on data mining

Liu H, Shao M, Li S, Fu Y (2016) Infinite ensemble for image clustering. In: Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining

Liu H, Wu J, Liu T, Tao D, Fu Y (2017a) Spectral ensemble clustering via weighted k-means: theoretical and practical evidence. IEEE Trans Knowl Data Eng 29(5):1129–1143

Liu H, Zhao R, Fang H, Cheng F, Fu Y, Liu Y-Y (2017b) A novel clustering method for patient stratification. Bioinformatics 167:1–8

Lu Z, Peng Y, Xiao J (2008) From comparing clusterings to combining clusterings. In: Proceedings of AAAI conference on artificial intelligence
Luo D, Ding C, Huang H, Nie F (2011) Consensus spectral clustering in near-linear time. In: Proceedings of international conference on data engineering
Miller J, Rupert G (2011) Survival analysis. Wiley, New York
Mirkin B (2001) Reinterpreting the category utility function. Mach Learn 45(2):219–228
Nguyen N, Caruana R (2007) Consensus clusterings. In: Proceedings of IEEE international conference on data mining
Shao M, Li S, Ding Z, Fu Y (2015) Deep linear coding for fast graph clustering. In: Proceedings of international joint conference on artificial intelligence
Song C, Liu F, Huang Y, Wang L, Tan T (2013) Auto-encoder based data clustering. In: Iberoamerican congress on pattern recognition. Springer, Berlin, Heidelberg, p 117–124
Strehl A, Ghosh J (2003) Cluster ensembles—a knowledge reuse framework for combining partitions. J Mach Learn Res 3:583–617
Tao Z, Liu H, Li S, Fu Y (2016) Robust spectral ensemble clustering. In: Proceedings of conference on information and knowledge management
Tao Z, Liu H, Fu Y (2017) Simultaneous clustering and ensemble. In: Proceedings of AAAI conference on artificial intelligence
Tian F, Gao B, Cui Q, Chen E, Liu T (2014) Learning deep representations for graph clustering. In: Proceedings of AAAI conference on artificial intelligence
Topchy A, Jain A, Punch W (2003) Combining multiple weak clusterings. In: Proceedings of international conference on data mining
Topchy A, Jain A, Punch W (2004) A mixture model for clustering ensembles. In: Proceedings of SIAM international conference on data mining
Uhlén M, Hallström BM, Lindskog C, Mardinoglu A, Pontén F, Nielsen J (2016) Transcriptomics resources of human tissues and organs. Mol Syst Biol, 12(4):862:1–12
Vega-Pons S, Ruiz-Shulcloper J (2011) A survey of clustering ensemble algorithms. Int J Pattern Recognit Artif Intell 25(3):337–372
Vega-Pons S, Correa-Morris J, Ruiz-Shulcloper J (2010) Weighted partition consensus via kernels. Pattern Recognit 43(8):2712–2724
Vincent P, Larochelle H, Bengio Y, Manzagol P-A (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of international conference on machine learning
Wu J, Liu H, Xiong H, Cao J (2013) A theoretic framework of k-means-based consensus clustering. In: Proceedings of international joint conference on artificial intelligence
Wu J, Liu H, Xiong H, Cao J, Chen J (2015) K-means-based consensus clustering: a unified view. IEEE Trans Knowl Data Eng 27(1):155–169
Xie G-S, Zhang X-Y, Liu C-L (2015) Efficient feature coding based on auto-encoder network for image classification. In: Proceedings of Asian conference on computer vision
Zhu Q, Wong AK, Krishnan A, Aure MR, Tadych A, Zhang R et al (2015) Targeted exploration and analysis of large cross-platform human transcriptomic compendia. Nat Methods 12(3):211–214