Knowledge Integration in Deep Clustering

Nguyen-Viet-Dung Nghiem✉[1], Christel Vrain[1], and Thi-Bich-Hanh Dao[1]

Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, F-45067, Orléans, France
`nguyen-viet-dung.nghiem@etu.univ-orleans.fr`
`{christel.vrain, thi-bich-hanh.dao}@univ-orleans.fr`

Abstract. Constrained clustering that integrates knowledge in the form of constraints in a clustering process has been studied for more than two decades. Popular clustering algorithms such as K-means, spectral clustering and recent deep clustering already have their constrained versions, but they usually lack of expressiveness in the form of constraints. In this paper we consider prior knowledge expressing relations between some data points and their assignments to clusters in propositional logic and we show how a deep clustering framework can be extended to integrate this knowledge. To achieve this, we define an expert loss based on the weighted models of the logical formulas; the weights depend on the soft assignment of points to clusters dynamically computed by the deep learner. This loss is integrated in the deep clustering method. We show how it can be computed efficiently using Weighted Model Counting and decomposition techniques. This method has the advantages of both integrating general knowledge and being independent of the neural architecture. Indeed, we have integrated the expert loss into two well-known deep clustering algorithms (IDEC and SCAN). Experiments have been conducted to compare our systems IDEC-LK and SCAN-LK to state-of-the-art methods for pairwise and triplet constraints in terms of computational cost, clustering quality and constraint satisfaction. We show that IDEC-LK can achieve comparable results with these systems, which are tailored for these specific constraints. To show the flexibility of our approach to learn from high-level domain constraints, we have integrated implication constraints, and a new constraint, called span-limited constraint that limits the number of clusters a set of points can belong to. Some experiments are also performed showing that constraints on some points can be extrapolated to other similar points.

Keywords: Deep Clustering · Knowledge Integration · Constrained Clustering.

1 Introduction

Clustering is an important task in Data Mining, which aims at partitioning data instances into groups (clusters) such that instances in the same cluster are similar and instances in different clusters are dissimilar. Prior knowledge has been integrated into the clustering process by means of constraints, leading to a new field called Constrained Clustering. Constraints can be instance-level constraints, mainly must-link, resp. cannot-link constraints, which state that two instances must be, resp. cannot be in the same cluster. Constraints can also specify requirements that the clusters must satisfy. Many works have been developed to integrate constraints: by enforcing constraints [25], by balancing clustering quality and constraint satisfaction [8], or by learning a metric taking into account the constraints [28,3]. Most clustering approaches are based on a distance between objects leading to the difficulty of choosing the right representation of data. The emergence of deep learning and its ability to learn new data representation in a lower dimension space have led to deep clustering approaches [26,12,6]. The integration of constraints into deep clustering has been studied in [14,30] but most of the work focused only on instance-level constraints. To the best of our knowledge few work consider the integration of different types of constraints, such as [30], where a loss is defined for each type of constraints and the loss criterion is therefore a combination of the loss for each kind of constraints. Such a framework has two drawbacks: the integration of a new family of constraints requires the design of the corresponding loss, and defining the global loss needs to set the parameters to combine the different losses for the constraints.

In our work, we take another point of view which is to define a general constraint satisfaction score. Then, an expert loss, based on this constraint satisfaction score, is introduced into a deep clustering framework for backpropagation. We show that the constraint satisfaction score can be computed through its transformation into a Weighted Model Counting problem [21]. Based on logical formulas, our approach has the advantage of being flexible, so that different types of knowledge can be integrated without designing specific losses. The framework is summarized in Figure 1.

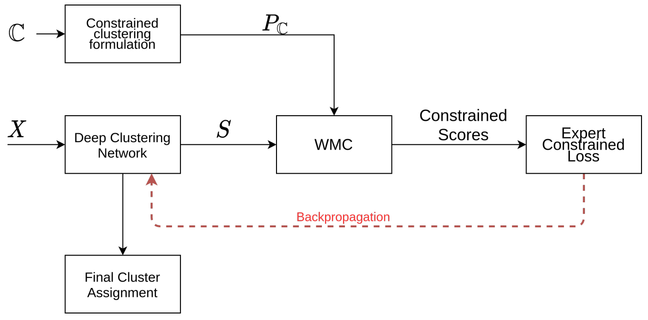

Fig. 1. Overview of our DC-LK framework. The constrained clustering problem is formulated in a logical form $P_{\mathbb{C}}$. A deep clustering framework is used to compute a soft assignment S of data points $\mathbf{X}$ to clusters. The constrained score is computed based on S and the constraint problem $P_{\mathbb{C}}$ and is used to define the expert loss. It is backpropagated to the deep clustering network.

Our contributions are:

- We propose a logical formulation of the constrained clustering problem and a unified definition of expert loss to integrate constraints into a deep clustering framework.
- Given a constraint set, the expert loss is based on a constraint satisfaction score, defined thanks to the notion of semantic models of a logical formulae, thus making it independent from the type of constraints. Moreover, this can be computed by Weighted Model Counting.
- We show that our framework can be integrated into different clustering frameworks by considering two well-known deep clustering methods, namely IDEC and SCAN, and extending them to integrate knowledge.
- Experiments on five datasets with randomly generated constraint sets show that our framework is competitive with state-of-the-art deep constrained clustering systems on pairwise and triplet constraints.
- To illustrate the genericity of our approach, we introduce a new type of constraints, called a span-limited constraint.
- We analyze the efficiency of our framework both on runtime and on constraint satisfaction for complex constraints.
- We show that satisfying constraints when training the model allows to improve the satisfaction of unseen constraints on test data.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. The formulation of the constrained clustering problem and of the expert loss is presented in Section 3. Section 4 presents knowledge integration into two deep clustering frameworks IDEC and SCAN using the expert loss. Section 5 describes the experiments and analyzes the results and Section 6 concludes and discusses future work.

2 Related Work

Constrained clustering. Many approaches have now been developed for constrained clustering. Most of them focus on pairwise (must-link/cannot-link) constraints and the early work consisted in adapting classic methods such as k-means or spectral clustering to enforce them [5,18]. Other constraints have been introduced as for instance cardinality constraints [22]. Nevertheless all these approaches are usually designed for one kind of constraints whereas the expert knowledge is often multiform including both pairwise constraints, cardinality constraints and much more complex constraints as given for instance in [9], thus requiring new frameworks for constrained clustering. It has been shown that declarative frameworks such as ILP [1,19], SAT [11] or Constraint Programming [8] allow to integrate a large variety of constraints, while satisfying all the constraints.

Deep clustering. Recently, deep clustering approaches have been extensively proposed following the success of deep neural networks in supervised learning. Several research directions have been considered: adapting to clustering well-known supervised learning architectures such as convolutional neural network

[6], changing data representation through an autoencoder and then enforcing the clustering structure on the latent space [26,12]. Another approach is to mine the nearest neighbor based on pretext features (an embedding for a specific task such as inpainting patches, predicting noise, instance discrimination). It helps to promote similar predictions of the neighbors, thus, improving the clustering quality [23]. More ambitious approaches have been proposed as for instance generative models that both cluster data and generate samples for a given clustering [15,20], but they usually suffer from relatively low performances.

Deep learning with knowledge. Knowledge integration can be seen as a generalization of semi-supervised learning. While label information is easy to represent, expert knowledge can be various and thus expressed in many different ways. To tackle this problem, several work [27,29] have studied the integration of knowledge expressed in logic in Deep Supervised Learning: knowledge is then enforced on each individual input instance. [29] gives a precise formulation of the loss regardless of the logical form (whether it is represented in CNF, DNF or in a arbitrary form) at the price of a high computational complexity. [27] learns a knowledge loss (using a logic graph embedder) for a specific logical form (d-DNNF), which is much faster but requires a substantial amount of constraints.

In a clustering setting, even if each point receives a label, the output on all the data is expected to represent a partition (or another structure). This means that the constraints are not put on the output of a single point, but they can link several outputs, which is much more challenging.

[14] integrates triplet constraints in a deep clustering framework. DCC [30] has proposed a deep clustering framework, which can integrate several types of constraints such as pairwise, triplet or cardinality. However for each type of constraint, a specific loss is designed. This differs from our approach, where the same definition of expert loss is given for any type of constraints, as soon as the constraint can be expressed using a logical formulation.

3 Expert loss for knowledge integration

3.1 Expert knowledge representation

In this paper, we are interested in integrating knowledge in a deep clustering system. We suppose that we have n points $x_1, \ldots, x_n$ and that we want to learn a partition of them into k clusters. We also have expert knowledge on the desired partition, which is expressed by expert constraints and written in propositional logic. Let β_{ij}, $i \in \{1, ..., n\}, j \in \{1, ..., k\}$ be formulas, such that β_{ij} is $True$ when point i is assigned to cluster j. Using β, the predicate meaning that two points u, v are in the same cluster can be expressed by

$$Together(u, v) \overset{\text{def}}{=} \wedge_{i\in[1,k]}((\beta_{ui} \wedge \beta_{vi}) \vee (\neg\beta_{ui} \wedge \neg\beta_{vi}))$$

and the predicate meaning that two points i, j are in different clusters:

$$Apart(u, v) \overset{\text{def}}{=} \wedge_{i\in[1,k]}(\neg\beta_{ui} \vee \neg\beta_{vi})$$

The constraints we consider are:

– must-link (cannot-link) constraints stating that two points u, v must be (resp. cannot be) in the same cluster are expressed by the statement $Together(u, v)$ (resp. $Apart(u, v)$).
– triplet constraints (a, p, n) expressing that a is closer to p than to n:

$$Together(a, n) \implies Together(a, p)$$

– more complex implication constraints, as for instance:

$$Together(a, b) \wedge Together(c, d) \wedge Apart(a, e) \Rightarrow Together(e, f) \wedge Apart(c, e)$$

– a new type of constraints, called *span-limited constraint* that expresses that a given group I of points cannot be dispatched in more than a given number m of clusters.

$$\vee_{J\subset\{1,\ldots,k\},|J|=m} \wedge_{i\in I} \vee_{j\in J}\beta_{ij}$$

3.2 Constraint-satisfaction score

We consider a deep clustering algorithm that produces a soft assignment S of points to clusters, where S_{ij} denotes the soft assignment value of point i to cluster j. We aim at integrating in this system a new loss, called **expert loss** that takes into account the satisfaction of constraints. This loss has to be generic so as to integrate different kinds of constraints, this explains why we rely on propositional logic. This expert loss is based on the weighted models of the logical formulas, where the weights depend on the soft assignment of points to clusters dynamically computed by the deep learner.

Given a partition $\mathbf{p}$ and the soft assignment matrix S, we define the partition score between the partition and S by:

$$Score(\mathbf{p}, S) = \prod_{i\in[1,n]} S_{i\mathbf{p}_i} \tag{1}$$

where $\mathbf{p}_i$ denotes the assignment of i in the partition $\mathbf{p}$.

Given a set of constraints $\mathbb{C}$, we denote by $\mathbb{P}_{\mathbb{C}}$ the set of partitions that satisfy all the constraints in $\mathbb{C}$. To measure how likely the soft assignment S is with respect to the constraint set $\mathbb{C}$, we define a constraint-satisfaction score as follows:

$$Score(\mathbb{C}, S) = \sum_{p\in\mathbb{P}_{\mathbb{C}}} Score(\mathbf{p}, S) \tag{2}$$

To illustrate this, let us suppose that we have only 2 points and 2 clusters. Then the score of the partition that assigns each point to cluster 1 is $S_{11} * S_{21}$. If we add a must-link constraint betwwen the two points, then two partitions satisfy this constraint, and $Score(\mathbb{C}, S) = S_{11} * S_{21} + S_{12} * S_{22}$.

3.3 Constraint-satisfaction score computed by a WMC problem

The constraint-satisfaction score (2) can be computed directly. However, enumerating all the partitions in $\mathbb{P}_{\mathbb{C}}$ is expensive ($\mathcal{O}(k^n)$). We show that this problem can be converted into a Weighted Model Counting problem with an appropriate choice of formulas and of weights.

Let us recall that if α is a logical formula defined on a set of variables $\mathbf{Y}$, where each variable Y_i is associated with a weight w_i, then the weighted model counting (WMC) of α is defined by [21]:

$$WMC(\alpha, w) = \sum_{\mathbf{y} \models \alpha} \prod_{i:\mathbf{y} \models Y_i} w_i \prod_{i:\mathbf{y} \models \neg Y_i} (1 - w_i) \tag{3}$$

In (3), the weights w_i and $(1-w_i)$ can occur several times. Sentential Decision Diagrams (SDD) [10] can be used for a more efficient representation to compress the computation into an arithmetic tree.

Theorem 1. *Let $\mathbf{B}$ be a set of logical variables $\{B_{ij} : i \in [1,n], j \in [1,k]\}$. We define β_{ij} as follows:*

$$\begin{aligned} &\beta_{ij} \overset{def}{=} B_{ij} \wedge \textstyle\bigwedge_{t\in[1,j-1]} \neg B_{it} \textit{ for all } j \in [1, k-1], \\ &\beta_{ik} \overset{def}{=} \textstyle\bigwedge_{t\in[1,k-1]} \neg B_{it} \end{aligned} \tag{4}$$

Let w_B the weight for the variables defined by:

$$w_B(B_{ij}) = \begin{cases} S_{ij}/(1 - \sum_{t\in[1,j-1]} S_{it}) & \textit{if } \sum_{t\in[1,j-1]} S_{it} < 1 \\ 1 & \textit{otherwise} \end{cases} \tag{5}$$

Given a set of constraints $\mathbb{C}$ expressed using β. Then we have:

$$Score(\mathbb{C}, S) = WMC(\beta \wedge \mathbb{C}, w_B) \tag{6}$$

Theorem 2. *With the definition of β by (4), the formula stating that each point belongs to a single cluster is expressed by (7) and is a tautology, i.e. (7) $\equiv \top$.*

$$\bigwedge_i (\beta_{i1} \wedge \ldots \wedge \beta_{ik} \bigwedge_{j,l\in[1,k]:j\neq l} (\neg\beta_{ij} \vee \neg\beta_{il})) \tag{7}$$

In (4), β_{ij} true means the point i is assigned to cluster j and not to any other cluster j' such that $j' < j$. In Theorem 2, the fact that (7) is a tautology means using β we ensure the result is a partition, which is required in a clustering problem. The proofs of the two theorems are given in the supplementary material[1].

Translation of expert constraints in terms of B The constraints are expressed in terms of β. For sake of efficiency, expert knowledge is expressed with B. For instance, a cannot-link constraint is written:

$\wedge_{i\in[1,k]}(\neg\beta_{ui} \vee \neg\beta_{vi}) \iff \wedge_{i\in[1,k]} \left[\neg B_{ui} \vee \neg B_{vi} \vee_{t\in[1,i-1]} (B_{ut} \vee B_{vi})\right]$

[1] https://github.com/dung321046/Knowledge-Integration-in-Deep-Clustering

3.4 Decomposition of the problem

The score that we have defined in (2) takes into account all the constraints of $\mathbb{C}$ together in a single loss term. In order to have a greater impact when learning but also for complexity reasons, the whole problem is decomposed into a set of sub-problems c, one for each expert constraint $c \in \mathbb{C}$. Given a constraint $c \in \mathbb{C}$, we define $Score(c, S)$ in the same way as in (2):

$$Score(c, S) = \sum_{p \in \mathbb{P}_c} Score(\mathbf{p}, S)$$

where $\mathbb{P}_c$ is the set of partitions satisfying c, and we define $Score(\mathbb{C}, S)$ by:

$$Score(\mathbb{C}, S) = \prod_{c \in \mathbb{C}} Score(c, S) \tag{8}$$

3.5 Expert loss

The expert loss L_{expert} is defined as

$$L_{expert} = -\log Score(\mathbb{C}, S) = -\sum_{c \in \mathbb{C}} \log Score(c, S) \tag{9}$$

4 Integrating knowledge in deep clustering frameworks

We present here our framework called DC-LK (Deep Clustering with Logical Knowledge), whose general scheme is shown in Figure 1. Given $\mathbf{X}$ a set of n points, k a number of clusters and $\mathbb{C}$ a set of constraints expressing expert knowledge, it computes a cluster assignment $p = \{p_1, p_2, ..., p_n\}$, $p_i \in \{1, \ldots, k\}$, expressing that point i belongs to cluster p_i, guided by the expert constraints. First, expert constraints are formulated in logic and represented as SDD structures. Data $\mathbf{X}$ is processed through a deep clustering network thus computing a soft cluster assignment $S = (S_{ij})$ that represents the likelihood of point i to belong to cluster j. The expert constraint loss depending on S is computed by Weight Model Counting and this loss is integrated with the deep learner loss for back-propagation. Any deep clustering learner [12,4,23] that computes a soft cluster assignment S could be used.

We consider two methods for integrating the expert loss in a deep clustering learner. Since at each epoch there are two objectives, one is for the clustering task, the other one is to satisfy the constraint, we propose two main methods to combine them: separated back-propagation and joined back-propagation. The separated back-propagation calculates and backpropagates the clustering loss (which depending on the architecture can be composed of several losses) and the expert loss separately. In contrast, the joined method combines all the losses into a weighted sum for back-propagation.

In this work, we have integrated our expert loss into two deep clustering frameworks IDEC [12] and SCAN [23].

4.1 IDEC-LK

IDEC The neural structure of IDEC is an autoencoder, which allows to learn new representation of data $\mathbf{Z} = encode(\mathbf{X})$. K-means is applied to find the cluster centers $(\mu_1, \ldots, \mu_k)$ in the embedding space. The soft cluster assignments of all points (to all clusters) is computed based on Student's t-distribution:

$$S_{ij} = \frac{(1+ \parallel z_i - \mu_j \parallel^2)^{-1}}{\sum_{j'}(1+ \parallel z_i - \mu_{j'} \parallel^2)^{-1}} \tag{10}$$

For learning, IDEC uses the clustering loss and the reconstruction loss. The reconstruction loss is the mean square distance between the original data $\mathbf{X}$ and the reconstructed output $\tilde{\mathbf{X}} = decode(encode(\mathbf{X}))$.

$$L_{recon} = \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 \tag{11}$$

The clustering loss is based on the Kullback–Leibler difference between the soft-assignment S_{ij} and an "ideal" target assignment P, which amplifies the separation between the clusters, defined by: $P_{ij} = (S_{ij}^2/f_j)/(\sum_{j'} S_{ij'}^2/f_{j'})$ where $f_j = \sum_{i=1}^{n} S_{ij}, j = 1, \ldots, k$ are the soft cluster frequencies. Then, the clustering loss is:

$$L_{clustering} = \sum_i \sum_j P_{ij} \log \frac{P_{ij}}{S_{ij}} \tag{12}$$

Expert integration We have implemented separated back-propagation for IDEC-LK, in order for constraints to have a stronger impact on learning. It may be less efficient in run time, since forward and backward are done twice, but this is not a problem, given the simple architecture of IDEC. Moreover, for efficiency reasons, we use mini-batch learning for constraint sets. So, the loss of IDEC-LK model is defined as:

$$\begin{aligned} L_1 &= \lambda_r \times L_{recon} + \lambda_c \times L_{clustering} \\ L_2 &= \lambda_e \times L_{expert} \end{aligned} \tag{13}$$

where λ_r, λ_c and λ_e are coefficients controlling each loss, L_{recon} is the reconstruction loss for the autoencoder, $L_{clustering}$ is the IDEC clustering loss based on KL divergence, L_{expert} is our expert loss. The detailed algorithm is given in 1.

4.2 SCAN-LK

SCAN The neural structure of SCAN is a convolutional neural network (CNN), which is pretrained by SimCLR[7] using contrastive learning. With a suitable K value, it is observed that the K nearest neighbors of a point in the pretext embedding are instances of the same cluster. Let us denote $S_i \in \mathbb{R}^k_{[0,1]}$ the soft

Algorithm 1 Training process of IDEC-LK

```
Input: Input data: $\mathbf{X}$, Number of clusters: $k$, Constraint set: $\mathbb{C}$, Maximum
    iterations: $MaxIter$; Coefficients: $\lambda_r$, $\lambda_e$, $\lambda_c$
Output: Cluster assignment $\mathbf{p}$
 1: Initialize parameters with pre-trained autoencoder
 2: Initialize $\mu$ with K-means on the representations learned by pre-trained
    autoencoder
 3: Generate $\mathbb{T}$ - a set of SDD structures from all $c \in \mathbb{C}$
 4: for $iter := 1$ to $MaxIter$ do
 5:     for $batch := 1$ to $NumConstrainedBatches$ do
 6:         $X_{batch} = \{x : x \in C_{batch}\}$
 7:         Calculate $Z_{batch} = encode(X_{batch})$
 8:         Forward distribution $S$ via $t$-distribution with $Z, \mu$; (Eq. (10)) from
    the set of points $x \in C_{batch}$
 9:         Feed $S$ to SDD structures $\mathbb{T}$ to calculate $L^{batch}_{expert}$
10:         Backpropagate $L_2$ and update parameters
11:     end for
12:     for $batch := 1$ to $BatchAllInputs$ do
13:         Calculate $Z_{batch} = encode(X_{batch})$
14:         Forward distribution $S_{batch}$ via $t$-distribution with $Z_{batch}, \mu$ (Eq. (10))
15:         Calculate target distribution $P_{batch}$
16:         Feed $Z_{batch}$ to the decoder to obtain the reconstruction $\tilde{X}_{batch}$
17:         Calculate $L^{batch}_{recon}$, $L^{batch}_{clustering}$, (Eq. (11), (12), respectively)
18:         Backpropagate $L_1$ and update parameters
19:     end for
20: end for
21: $\mathbf{p} = \{\arg\max_{j\in[1,k]} Q_{ij} : i \in [1, n]\}$
22: Return $\mathbf{p}$
```

assignment vector of x_i and $\mathcal{N}_{x_i}$ the neighborhood of x_i. The loss function is defined by:

$$L = \lambda_{nn} \times \frac{1}{n} \sum_{i=1}^{n} \sum_{x_j \in \mathcal{N}_{x_i}} \log\langle S_i \cdot S_j \rangle + \lambda_{entropy} \times \sum_{h=1}^{k} S^*_h \log S^*_h \qquad (14)$$

where $\lambda_{nn}, \lambda_{entropy}$ are the coefficients, $\langle\cdot\rangle$ is the dot product operator, $S^*_h = \frac{1}{n}\sum_{i=1}^{n} S_{ih}$. The first term enforces the similarity in predictions of x_i with its neighbors while the second term enforces the even distribution of points to all k clusters.

Expert integration In this case, we chose joined back-propagation because forward and backward operations are too expensive in SCAN architecture. This leads to a more complicated handling of constraints involved in batches (which are randomly generated in SCAN architecture). For each batch, the algorithm searches for expert constraints containing points in the batch and completes the batch by the points involved in these constraints. To limit the size of the batch, for each point at most one constraint involving it is chosen at random. The total loss of SCAN-LK model is defined as:

$$L = \lambda_{nn} \times L_{nearest} + \lambda_{entropy} \times L_{entropy} + \lambda_{expert} \times L_{expert} \tag{15}$$

where $\lambda_{nn}, \lambda_{entropy}$ and λ_{expert} are coefficients controlling each loss. When the stopping condition is reached (for instance the change of the loss is under a given threshold), a final assignment is computed for all i by taking $p_i = \arg\max_j S_{ij}$.

Algorithm 2 Training process of SCAN-LK

```
Input: Input data: $\mathbf{X}$, Number of clusters: $k$, Constraint set: $\mathbb{C}$, Maximum
    iterations: $MaxIter$; Coefficients: $\lambda_{nn}$, $\lambda_{entropy}$, $\lambda_{expert}$
Output: Cluster assignment $\mathbf{p}$
 1: Initialize parameters with SimCLR
 2: Generate $\mathbb{T}$ - a set of SDD structures from all $c \in \mathbb{C}$
 3: for $iter := 1$ to $MaxIter$ do
 4:     for $batch := 1$ to $BatchAllInputs$ do
 5:         Load $X_{batch}$ - points in the batch
 6:         Load $C_{batch}$ - expert constraints containing $X_{batch}$ and $X_{batch}^{constrained}$ -
    all points in $C_{batch}$
 7:         Forward $X_{batch}^{constrained}$ to obtain $S_{batch}^{constrained}$
 8:         Feed $S_{batch}^{constrained}$ to SDD structures $\mathbb{T}$ to calculate $L_{expert}^{batch}$
 9:         Calculate $L_{nearest}^{batch}$, $L_{entropy}^{batch}$
10:         Backpropagate $L = \lambda_{nn} \times L_{nearest}^{batch} + \lambda_{entropy} \times L_{entropy}^{batch} + \lambda_{expert} \times$
    $L_{expert}^{batch}$ and update parameters
11:     end for
12: end for
13: Forward all data $\mathbf{X}$ to obtain $S$.
14: $\mathbf{p} = \{\arg\max_{j \in [1,k]} S_{ij} : i \in [1,n]\}$
15: Return $\mathbf{p}$
```

5 Experiments

Experiments are conducted to address the following aims: (i) to evaluate IDEC-LK and SCAN-LK on clustering quality and to compare them with other systems on pairwise and triplet constraints; (ii) to evaluate our system on constraints that have never been used in deep clustering experiments, namely implication constraints and span-limited constraints. Our experiment could be reproduced using the avaiable source[2].

[2] https://github.com/dung321046/Knowledge-Integration-in-Deep-Clustering

5.1 Experiment Settings

Datasets. We use five datasets, which are challenging and also used in many recent deep constrained clustering methods [14,30]. **MNIST** contains 70,000 handwritten single-digits from 10 classes. Among them 60,000 images are used to perform clustering, the remaining ones are used to evaluate the interest of span-limited constraints. Similarly, **STL10** has 5,000 color images for clustering and expert learning and 8,000 images for testing. **Fashion** has 60,000 images associated to a label from 10 classes. **CIFAR10** consists of 50,000 color images in 10 classes. **Reuters** contains around 810,000 English news stories labeled with a category tree [17].

Experiment setting. For all the experiments, we first run SDAE[24] and SimCLR[7] for learning a new representation space. The same pre-trained model is given as input to the clustering algorithms. We compare our system to IDEC [12] (unconstrained), PCK-means [2] (pairwise constraints), MPCK-means [3] (pairwise constraints), and DCC [30] (pairwise and triplet constraints).

No supervised information is used for setting the parameters, therefore for SDAE, IDEC, SCAN and DCC, we use the default parameters. Our hyperparameters are detailed in the supplementary. We put two stopping conditions: either when the maximum number of epochs is reached or when the percentage of assignments that differ from the previous epoch is less than 0.01%.

Experiments are run on a 2.6 GHz Intel Core i7 processor and a NVIDIA GeForce RTX 2060 graphics card.

5.2 Experiments and Analysis for Clustering quality

In this section, we study the impact of pairwise and triplet constraints on the clustering quality. For all datasets, the true class of objects is available and we use it to evaluate the accuracy of the clustering. We consider two measures: Normalized Mutual Information (NMI) and clustering accuracy (ACC), with a one-to-one mapping between clusters and labels, computed by the Hungarian algorithm [16].

For testing the influence of pairwise constraints, we consider 4 numbers of constraints (10, 100, 500, 1000). For each test case, we randomly generate five sets of constraints, we run the system only once for each set of constraints and we report the mean and the standard deviation in Table 1 and Table 2. We do the same for triplet constraints (see Figure 2). All results are given in the supplementary materials.

Pairwise constraints. In MNIST, Fashion and Reuters datasets, IDEC-LK has competitive performances to the state-of-the-art methods. In terms of complexity, the times to run each epoch of DCC and of IDEC-LK are quite the same. However, the convergence of IDEC-LK is slower than for DCC (See Table 1).

In CIFAR10 and STL10, all the systems based on SDAE (IDEC, DCC, IDEC-LK) have a poor performance. The performance of IDEC with CIFAR10 data is reported in the supplementary material. With 1000 pairwise constraints, SCAN-LK helps to improve the clustering performance of CIFAR10 and STL10, with a ratio of 3% and 6% respectively.

Table 1. Comparison on clustering quality with 1000 pairwise constraints. Green (blue) numbers are for the best (second-best) values, respectively.

Data	Models	NMI	ACC	Time (s)
MNIST	DCC	0.8689 ± 0.0008	0.8815 ± 0.0007	277 ± 9
MNIST	MPCK-means	0.7589 ± 0.0171	0.7788 ± 0.0413	211 ± 3
MNIST	PCK-means	0.7463 ± 0.0228	0.7698 ± 0.0543	32.97 ± 15.90
MNIST	IDEC-LK	0.8680 ± 0.0017	0.8826 ± 0.0012	388 ± 27
Fashion	DCC	0.6000 ± 0.0019	0.5241 ± 0.0039	140 ± 16
Fashion	MPCK-means	0.5749 ± 0.0138	0.5312 ± 0.0292	205 ± 4
Fashion	PCK-means	0.5714 ± 0.0212	0.5314 ± 0.0293	37.02 ± 13.19
Fashion	IDEC-LK	0.6009 ± 0.0019	0.5230 ± 0.0034	358 ± 17
Reuters	DCC	0.5655 ± 0.0086	0.7477 ± 0.0030	3.46 ± 0.41
Reuters	MPCK-means	0.5262 ± 0.0330	0.7251 ± 0.0412	167 ± 2
Reuters	PCK-means	0.5174 ± 0.0288	0.7343 ± 0.0377	14.80 ± 2.34
Reuters	IDEC-LK	0.5927 ± 0.0105	0.7563 ± 0.0079	27.52 ± 9.88

Table 2. Comparison on clustering quality between the baselines and SCAN-LK with 1000 pairwise constraints.

Data	Models	NMI	ACC	#Unsat
CIFAR10	SCAN	68.30	79.39	183 ± 17
CIFAR10	SCAN-LK	71.81 ± 0.19	82.11 ± 0.27	55 ± 12.77
STL10	SCAN	65.11	75.58	194.67 ± 2.52
STL10	SCAN-LK	72.48 ± 0.79	83.57 ± 0.95	4.33 ± 0.58

Triplet constraints. The triplet constraints have less impact on the clustering quality than the pairwise ones because they convey conditional information on the points. Relying on a Kolmogorov-Smirnov test with $p = 0.05$ [13], IDEC-LK has better clustering quality in Reuters while it has similar performances with DCC in the other datasets.

5.3 Experiments and Analysis for Constraint Satisfaction

In this section, we aim at illustrating two points: first our method can leverage complex domain knowledge and second, it can learn from it, that is, it does not only aim at satisfying the constraints but it is also able to satisfy unseen constraints of the same type. The second point is crucial because acquiring

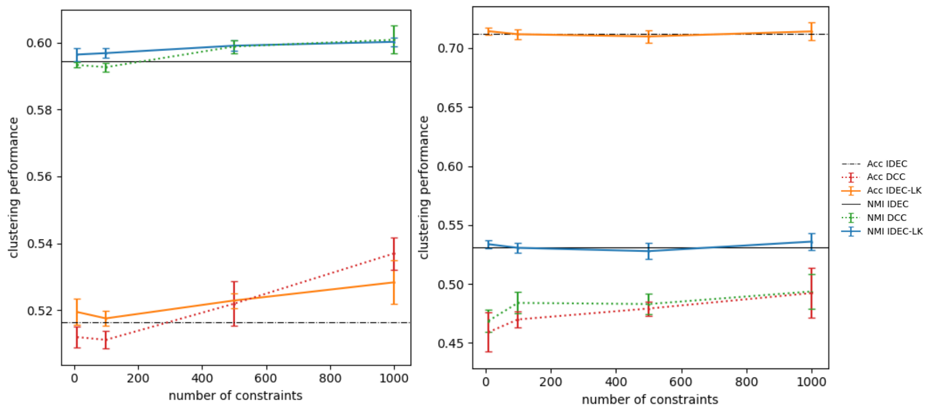

Fig. 2. The clustering performances with triplet constraints on Fashion (left) and Reuters (right) dataset

constraints is expensive, and the set of training constraints is only a minute fraction of all possible interpretations of the domain knowledge.

Implication Constraint. Introduced in Section 3.1, the first part (if-clause) is denoted as P and the second part (then-clause) as Q. To study the interest of such constraints, we generate 5 sets of 100 constraints at random based on the ground truth. For each constraint, the number of Together/Apart constraints in P is 3, the number of Together/Apart in Q is 1 and we define the notion of P-Q distribution: around 20% constraints satisfy $P = \perp$, the remaining 80% is $(P = \top, Q = \top)$.

Table 3. Comparison between IDEC and IDEC-LK on the satisfaction of implication constraints.

Data	Models	$\overline{Score}$	#Unsat
MNIST	IDEC	0.8777 ± 0.0118	13.6 ± 1.5
MNIST	IDEC-LK	0.8856 ± 0.0130	12.4 ± 1.7
Fashion	IDEC	0.7620 ± 0.0442	24.8 ± 4.6
Fashion	IDEC-LK	0.7743 ± 0.0449	23.2 ± 4.5
Reuters	IDEC	0.8290 ± 0.0376	17.8 ± 4.7
Reuters	IDEC-LK	0.8357 ± 0.0381	17.0 ± 5.0

In Table 3, IDEC-LK shows the improvement of the average constrained scores, denoted by $\overline{Score}$, compared to the value of IDEC. In all three datasets, the constraint satisfaction score has been improved, and the number of unsatisfied constrained is reduced.

Span-limited Constraint. In this experiment, we run IDEC-LK algorithm with MNIST and SCAN-LK with STL10. We aim at testing the interest of span-limited constraints that state that a group of points can be dispatched on a fixed number of clusters. For generating such constraints for MNIST dataset, we have selected four groups from the labels: $\mathcal{G} = \{G(3,9), G(6,8), G(1,7), G(2,5)\}$ (pairs of digits that share some similar shapes) and stated that elements in each group must be dispatched into two clusters. To create a group $G(u,v)$, we have selected randomly 100 images of either digit u or digit v . To test whether such constraints have a true impact, we chose them so that a quarter of them have been assigned to a wrong cluster by SDAE+K-Means (without constraints). For generating span-limited constraints for STL10 dataset, we have applied the same process with $\mathcal{G} = \{G(airplane, bird), G(cat, dog), G(deer, horse)\}$ and we have randomly selected 1,000 images for each group. Because the matching between labels and clusters is unknown, for each group $G(u,v)$, we set the two clusters to be the ones with the highest number of points in $G(u,v)$.

Figure 3 shows the changes in satisfaction of span-limited constraints. A quarter of the images in each spanning group initially does not belong to the two major clusters obtained with IDEC. After training with IDEC-LK with *all* points in all the four groups, the points that were already in these clusters remain, while the other points have changed to belong to one of the two clusters.

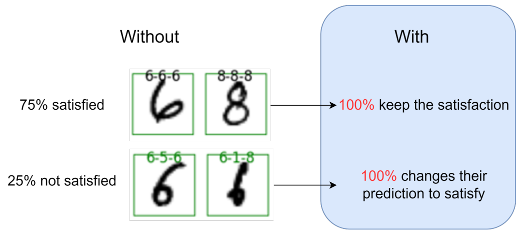
Without
With
75% satisfied
6-6-6
8-8-8
100% keep the satisfaction
25% not satisfied
6-5-6
6-1-8
100% changes their prediction to satisfy

Fig. 3. Training results with span-limited constraints of cluster 6 and 8 in MNIST. IDEC (Without) vs IDEC-LK (With)

Let us notice that the model is learned using constraints on the train set. In order to analyse the capacity of generalizing constraints, we study the ability of the model to satisfy constraints on the test set. Figure 4 presents the satisfaction of span-limited constraints from a test set, which are unknown when training the model: we consider two datasets MNIST ($10,000$ test points, IDEC) and STL10 ($8,000$ test points, SCAN) and for each we compare the behavior of two models, respectively learned without/with the constraints on the train set. All data in the test set are neither used in the clustering process, nor in knowledge integration. We can observe that the number of unsatisfied constraints in the test set is reduced by 85.7% for MNIST and 46.79% for STL10.

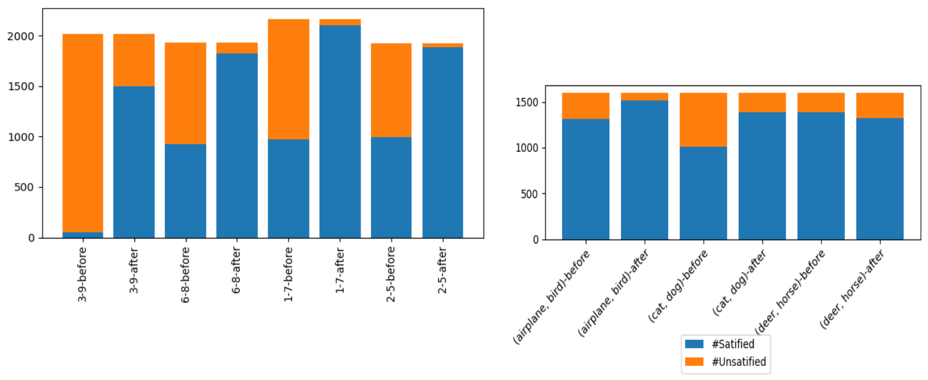

Fig. 4. Study on the effect of learning on constraints on test data: MNIST with IDEC-LK and STL10 with SCAN-LK, span-limited constraints

6 Conclusion

Our work is the first proposal of a general framework for integrating knowledge in constrained clustering problems. We propose an expert loss for integrating expert constraints and we show how it can be computed through Weight Model Counting. Relying on logic allows to express many kinds of constraints and we show the flexibility and adaptability of our method by considering new constraints such as implication constraints or span-limited constraints. This general framework has been embedded in deep clustering systems such as IDEC and SCAN. In our experiments, we obtain similar performance to other systems with well-known constraint types, but we also show the ability to integrate new constraints and even to generalize the constraints to unseen points. We plan to embed our proposal in other deep clustering architectures to show the generality of our approach.

The main limitation of this work is the complexity for computing the SDD trees, so that it prevents from incorporating cluster-level constraints or constructing a single loss for the whole constraint set. So we aim at reducing complexity by introducing new formulations or approximation schemes. However, let us notice that SDD trees are computed only once at the beginning of the learning process.

References

1. Babaki, B., Guns, T., Nijssen, S.: Constrained clustering using column generation. In: CPAIOR 2014. pp. 438–454 (2014)
2. Basu, S., Banjeree, A., Mooney, E., Banerjee, A., Mooney, R.J.: Active semi-supervision for pairwise constrained clustering. In: SDM. pp. 333–344 (2004)
3. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: ICML 2004. pp. 11–18 (2004)
4. Bo, D., Wang, X., Shi, C., Zhu, M., Lu, E., Cui, P.: Structural deep clustering network. In: Proceedings of The Web Conference 2020. pp. 1400–1410 (2020)

5. Bradley, P., Bennett, K., Demiriz, A.: Constrained k-means clustering. Tech. Rep. MSR-TR-2000-65, Microsoft Research (2000)
6. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: ECCV. pp. 132–149 (2018)
7. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML. pp. 1597–1607. PMLR (2020)
8. Dao, T.B.H., Duong, K.C., Vrain, C.: Constrained clustering by constraint programming. Artificial Intelligence **244**, 70–94 (2017)
9. Dao, T.B.H., Vrain, C., Duong, K.C., Davidson, I.: A Framework for Actionable Clustering using Constraint Programming. In: ECAI 2016. pp. 453–461 (2016)
10. Darwiche, A.: Sdd: A new canonical representation of propositional knowledge bases. In: IJCAI (2011)
11. Davidson, I., Ravi, S.S., Shamis, L.: A SAT-based Framework for Efficient Constrained Clustering. In: ICDM 2010. pp. 94–105 (2010)
12. Guo, X., Gao, L., Liu, X., Yin, J.: Improved deep embedded clustering with local structure preservation. In: IJCAI 2017. pp. 1753–1759 (2017)
13. Hodges, J.L.: The significance probability of the smirnov two-sample test. Arkiv för Matematik **3**(5), 469–486 (1958)
14. Ienco, D., Pensa, R.G.: Deep triplet-driven semi-supervised embedding clustering. In: DS. pp. 220–234. Springer (2019)
15. Jiang, Z., Zheng, Y., Tan, H., Tang, B., Zhou, H.: Variational deep embedding: An unsupervised and generative approach to clustering (2016)
16. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)
17. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: Rcv1: A new benchmark collection for text categorization research. JMLR **5**, 361–397 (2004)
18. Lu, Z., Carreira-Perpinan, M.A.: Constrained spectral clustering through affinity propagation. In: IEEE CVPR. pp. 1–8. IEEE (2008)
19. Mueller, M., Kramer, S.: Integer Linear Programming Models for Constrained Clustering. In: DS 2010. pp. 159–173 (2010)
20. Mukherjee, S., Asnani, H., Lin, E., Kannan, S.: Clustergan: Latent space clustering in generative adversarial networks. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 4610–4617 (2019)
21. Sang, T., Beame, P., Kautz, H.A.: Performing bayesian inference by weighted model counting. In: AAAI. vol. 5, pp. 475–481 (2005)
22. Tang, W., Yang, Y., Zeng, L., Zhan, Y.: Optimizing MSE for clustering with balanced size constraints. Symmetry **11**(3) (2019)
23. Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., Van Gool, L.: Scan: Learning to classify images without labels. In: ECCV. pp. 268–285 (2020)
24. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A., Bottou, L.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research **11**(12) (2010)
25. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S.: Constrained K-means Clustering with Background Knowledge. In: ICML 2001. pp. 577–584 (2001)
26. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: ICML 2016. pp. 478–487 (2016)
27. Xie, Y., Xu, Z., Kankanhalli, M.S., Meel, K.S., Soh, H.: Embedding symbolic knowledge into deep networks. In: NIPS. pp. 4233–4243 (2019)
28. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning with application to clustering with side-information. In: NIPS. vol. 15, p. 12 (2002)

29. Xu, J., Zhang, Z., Friedman, T., Liang, Y., Broeck, G.: A semantic loss function for deep learning with symbolic knowledge. In: ICML. pp. 5502–5511 (2018)
30. Zhang, H., Zhan, T., Basu, S., Davidson, I.: A framework for deep constrained clustering. Data Mining and Knowledge Discovery **35**(2), 593–620 (2021)