Deep Constrained Clustering - Algorithms and Advances

Hongjing Zhang [1] Sugato Basu [2] Ian Davidson [1]

Abstract

The area of constrained clustering has been extensively explored by researchers and used by practitioners. Constrained clustering formulations exist for popular algorithms such as k-means, mixture models, and spectral clustering but have several limitations. We explore a deep learning formulation of constrained clustering and in particular explore how it can extend the field of constrained clustering. We show that our formulation can not only handle standard together/apart constraints without the well documented negative effects reported but can also model instance level constraints (level-of-difficulty), cluster level constraints (balancing cluster size) and triplet constraints. The first two are new ways for domain experts to enforce guidance whilst the later importantly allows generating ordering constraints from continuous side-information.

1. Introduction

Constrained clustering has a long history in machine learning with many standard algorithms being adapted to be constrained (Basu et al., 2008) including EM (Basu et al., 2004), K-Means (Wagstaff et al., 2001) and spectral methods (Wang & Davidson, 2010). The addition of constraints generated from ground truth labels allows a semi-supervised setting to increase accuracy (Wagstaff et al., 2001) when measured against the ground truth labeling.

However, there are several limitations in these methods and one purpose of this paper is to explore how deep learning can make advances to the field beyond what other methods have. In particular we find that existing non-deep formulations of constrained clustering have the following limitations:

- *Limited Constraints and Side Information.* Constraints are limited to simple together/apart constraints typically generated from labels. In some domains, experts may more naturally give guidance at the cluster level or generate constraints from continuous side-information.
- *Negative Effect of Constraints.* For some algorithms though constraints improve performance when averaged over many constraint sets, individual constraint sets produce results worse than using no constraints (Davidson et al., 2006). As practitioners typically have one constraint set their use can be "hit or miss".
- *Intractability and Scalability Issues.* Iterative algorithms that directly solve for clustering assignments run into problems of intractability (Davidson & Ravi, 2007). Relaxed formulations (i.e. spectral methods (Lu & Carreira-Perpinan, 2008; Wang & Davidson, 2010)) require solving a full rank eigen-decomposition problem which takes $O(n^3)$.
- *Assumption of Good Features.* A core requirement is that good features or similarity function for complex data is already created.

Since deep learning is naturally scalable and able to find useful representations we focus algorithmically on the first and second challenges but experimentally explore the third and fourth. Though deep clustering with constraints has many potential benefits to overcome these limitations it is not without its challenges. Our major contributions in this paper are summarized as follows:

- We propose a deep constrained clustering formulation that cannot only encode standard together/apart constraints but new triplet constraints (which can be generated from continuous side information), instance difficulty constraints, and cluster level balancing constraints (see section 3).
- Deep constrained clustering overcomes a long term issue (Davidson et al., 2006) with constrained clustering of profound practical implications: overcoming the negative effects of individual constraint sets.
- We show how the benefits of deep learning such as scalability and end-to-end learning translate to our deep constrained clustering formulation. We achieve better clustering results than traditional constrained clustering methods (with features generated from an autoencoder) on challenging datasets (see Table 3).

[1]Department of Computer Science, University of California, Davis, USA [2]Google Research, Mountain View, USA. Correspondence to: Ian Davidson <davidson@cs.ucdavis.edu>.

Our paper is organized as follows. First, we introduce the related work in section 2. We then propose four forms of constraints and introduce how to train the clustering network with these constraints in section 3. Then we compare our approach to previous baselines and demonstrate the effectiveness of new types of constraints in section 4. Finally, we discuss the future work and conclude in section A.

2. Related Work

Constrained clustering is an important area in the research communities of machine learning. There is a large body of work on constrained clustering that takes into account the *side information* to improve the clustering performance (Wagstaff & Cardie, 2000; Wagstaff et al., 2001; Xing et al., 2003; Bilenko et al., 2004; Wang & Davidson, 2010). Here the side information is typically labeled data which is used to generate *pairwise* together/apart constraints used to partially reveal the ground truth clustering to help the clustering algorithm. Such constraints are easy to encode in matrices and enforce in procedural algorithms though not with its challenges. In particular, it was shown (Davidson et al., 2006) performance improves with larger constraint sets when **averaged** over many constraint sets generated from the ground truth labeling. However, for a significant fraction (just not the majority) of these constraint sets performance is *worse* than using no constraint set.

Moreover, side information can exist in different forms beyond labels (i.e. continuous data) and domain experts can provide guidance beyond pairwise constraints. Some work in the supervised classification setting (Joachims, 2002; Schultz & Joachims, 2004; Schroff et al., 2015; Gress & Davidson, 2016) seek alternatives such as relative/triplet guidance but to our knowledge such information has not been explored in the non-hierarchical clustering setting. Complex constraints for hierarchical clustering have been explored (Bade & Nürnberger, 2008; Chatziafratis et al., 2018) but these are tightly limited to the hierarchical structure (i.e. x must be joined with y before z) and not directly translated to non-hierarchical (partitional) clustering.

Motivated by the success of deep neural networks in supervised learning, unsupervised deep learning approaches are now beginning exploring (Xie et al., 2016; Jiang et al., 2016; Yang et al., 2016a;b; Guo et al., 2017; Shaham et al., 2018). There are approaches (Yang et al., 2016a; Shaham et al., 2018) which learn an encoding that is suitable for a clustering objective first and then applied an external clustering method. Our work builds upon the most direct setting (Xie et al., 2016; Guo et al., 2017) which encodes one self-training objective and finds the clustering allocations for all instances within one neural network.

Most recently, the semi-supervised clustering networks with pairwise constraints have been explored: Work (Hsu & Kira, 2015) uses pairwise constraints to enforce small divergence between similar pairs while increasing the divergence between dissimilar pairs assignment probability distributions. However, this approach did not leverage the unlabeled data, which requires lot's of labeled data to achieve good results. Fogel et al. proposed an unsupervised clustering network (Fogel et al., 2018) by self-generating pairwise constraints from mutual KNN graph and extends it to semi-supervised clustering by using labeled connections queried from human. However, this method can't make out-of-sample predictions and requires user-defined parameters for generating constraints from mutual KNN graph. Their approach is also slower than us because they need to compute the whole pairwise similarity to build the neighbor's graph.

3. Deep Constrained Clustering Framework

Here we outline our proposed framework for deep constrained clustering. The reader can see training process for each type's of constraints in Appendix. As is the norm in the field we add constraints to an accepted clustering method. We first describe the deep embedded clustering framework (DEC (Xie et al., 2016)) and then our extensions to handle constraints. Instance level pairwise constraints which allow encoding preferences of together and apart have been extensively studied (Basu et al., 2008) which we describe next. We then describe three new types of constraints that are fundamentally different from pairwise constraints. Instance difficulty constraints allow a domain scientist to specify which instances are challenging to clusters. Triplet constraints specify similarity amongst triples in that instance i is more similar to instance j than instance k. They are useful for generating constraints from *continuous or ordinal* side information such as age, income etc. Finally, we explore cardinality cluster level constraints that encode the preferred size of clusters. This type of information is useful in the semi-supervised setting since we typically cluster data points into as many clusters as there are labels and hence strong prior expectations of the cluster sizes are known.

3.1. Deep Embedded Clustering

We choose to apply our constraints formulation to the deep embedded clustering framework. Deep Embedded Clustering (DEC) (Xie et al., 2016) starts with pre-training an autoencoder ($x_i = g(f(x_i))$ but then removes the decoder. The remaining encoder ($z_i = f(x_i)$) is then fine-tuned by optimizing an objective which takes first z_i and converts it to a soft allocation vector of length k which we term $q_{i,j}$ indicating the degree of belief instance i belongs to cluster j. Then q is self-trained on to produce p a unimodal "hard" allocation vector which allocates the instance to primarily only one cluster.

Conversion of z to Soft Cluster Allocation Vector. Soft assignment q_{ij} is the similarity between embedded point z_i and cluster centroid u_j measured by Student's t-distribution (Maaten & Hinton, 2008), the v is a constant as $v = 1$:

$$q_{ij} = \frac{(1 + ||z_i - \mu_j||^2/v)^{-\frac{v+1}{2}}}{\sum_{j'} (1 + ||z_i - \mu_{j'}||^2/v)^{-\frac{v+1}{2}}} \tag{1}$$

Conversion of Q To Hard Cluster Assignments P. The above normalized similarities between embedded points and centroids can be considered as soft cluster assignments Q. However, we desire a target distribution P that better resembles a hard allocation vector, p_{ij} is defined as:

$$p_{ij} = \frac{{q_{ij}}^2 / \sum_i q_{ij}}{\sum_{j'} ({q_{ij}}^2 / \sum_i q_{ij'})} \tag{2}$$

Loss Function. Then the algorithm's loss function is simply to minimize the distance between P and Q as follows.

$$\ell_C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3}$$

The DEC framework requires the initial centroids given (μ) to calculate Q are representative. The initial centroids are set using k-means clustering. However, there is no guarantee that the clustering results over an auto-encoders embedding yield a good clustering. We believe that constraints can help overcome this issue which we test later.

3.2. Different Types of Constraints

To enhance the clustering performance and allow for more types of interactions between human and clustering models. We propose four types of guidance which are pairwise constraints, instance difficulty constraints, triplet constraints and global size constraints.

3.2.1. Pairwise Constraints

Pairwise constraints are well studied and used in both clustering and ranking problems. They are capable of defining any ground truth set partitions, we also extend the DEC framework to work with pairwise constraints such as must-link constraints and cannot-link constraints (Wagstaff et al., 2001).

We encode the loss for must-link constraints set ML as:

$$\ell_{ML} = - \sum_{(a,b) \in ML} \log \sum_j q_{aj} * q_{bj} \tag{4}$$

Similarly loss for cannot-link constraints set CL is:

$$\ell_{CL} = - \sum_{(a,b) \in CL} \log (1 - \sum_j q_{aj} * q_{bj}) \tag{5}$$

Intuitively speaking, the must-link loss prefers instances with same soft assignments and the cannot-link loss prefers the opposite cases. However, optimizing the must-link loss may run into issue of mapping all the points into same embedding space. To mitigate this problem, we add the reconstruction loss together with the must-link loss to prevent the learned embedding fall into trivial solution. The detail is introduced in the subsection 3.3.

3.2.2. Instance Difficulty Constraints

As mentioned before, there is no guarantee of pulling samples near margins towards the "correct" cluster based on self-crafted distribution P. There exist some instances which are hard to identify the real cluster, thus pushing the soft assignment of these instance towards "correct" cluster is not necessary. To counter these cases we propose a new type of constraint called instance difficulty constraints, intuitively this type of constraints will make the "easier" instances as dense as possible and leave the "harder" instances as they are in the latent feature space.

We encode user supervision with an $n \times 1$ constraint vector M. Let $M_i \in [-1, 1]$ be an instance difficulty indicator, $M_i > 0$ means the instance i is easy to be distinguished between others, $M_i = 0$ means no extra difficulty information is provided and $M_i < 0$ means instance i is hard to distinguish and prone to be an outlier. We initialize the constraint vector M as all zeros so that no difficulty supervision is provided. The loss function is formulated as:

$$\ell_I = \sum_{t \in \{M_t < 0\}} M_t \sum_j {q_{tj}}^2 - \sum_{s \in \{M_s > 0\}} M_s \sum_j {q_{sj}}^2 \tag{6}$$

The instance difficulty loss function aims to encourage the easier instances have sparse clustering assignments and also prevent the difficult instances having sparse clustering assignments, the absolute value of M_i means confidence degree. This loss will help the model training process converge faster on easier instances and increase our model's robustness towards difficult instances.

3.2.3. Triplet Constraints

Although pairwise constraints are capable of defining any ground truth set partitions, must-links are hard to get in some domains, especially when the clustering target attribute is in the continuous domain. Thus we seek triplet constraints, which is weaker constraints that indicate the relationship within a triplet.

Given an anchor instance a, positive instance p and negative instance n. The loss function for this triplet (a, p, n) can be represented as:

$$\ell_T = \max(d(q_a, q_n) - d(q_a, q_p) + \theta, 0) \tag{7}$$

where $d(q_a, q_b) = \sum_j q_{aj} * q_{bj}$ and $\theta > 0$. The larger value of $d(q_a, q_b)$ represents larger similarity between a and b. The variable θ controls the gap distance between positive and negative instances. ℓ_T works by pushing the positive instance's assignment closer to anchor's assignment and preventing negative instance's assignment being closer to anchor's assignment.

3.2.4. Global Size Constraints

Experts may more naturally give guidance at a higher level but not just for several instances. We are motivated to provide global constraints which can satisfy human's need. We explore the global clustering size constraints, which means each cluster should have the same size. Denote the total number of clusters as k, total training instances number as n, the global size constraints loss function is as follows:

$$\ell_G = \sum_{c \in \{1,..k\}} (\sum_{i=1}^{n} q_{ic}/n - \frac{1}{k})^2 \tag{8}$$

Our global constraints loss function works by minimizing the distance between expected cluster ratio and actual cluster ratio. The actual cluster ratio is calculated by averaging the soft-assignments. To guarantee the effectiveness of global size constraints, we need to assume that during our mini-batch training the batch size should be large enough comparing to the clusters number. Otherwise, the actual cluster ratio can't be well approximated and the training process won't converge.

3.3. Prevent Trivial Solution

Previous deep clustering method (Yang et al., 2016a) has shown one critical problem is how to avoid trivial solutions in this unsupervised task. In our framework the must-link constraints we mentioned before also faces this issue.Thus we combine the reconstruction loss with the must-link loss. Denote the encoding network as $f(x)$ and decoding network as $g(x)$, the reconstruction loss for instance x_i is:

$$\ell_R = \ell(g(f(x_i)), x_i) \tag{9}$$

where ℓ is the least-square loss: $\ell(x, y) = ||x - y||^2$.

Furthermore, DEC has an improved version as IDEC (Guo et al., 2017) which keeps the reconstruction branch with the clustering loss used in equation (3), we also follow the improvements and use IDEC as our base clustering method in following experiments.

3.4. Efficient Training Strategy

3.4.1. Parameter initialization

We use the same initialization strategy as DEC/IDEC which uses a stacked denoising autoencoder's (SDAE) weights to initialize our network, the unsupervised representation learned by SDAE naturally facilitates the learning of clustering representations. K-means clustering is applied to the initial forward pass' latent embedding z, we calculate out the k initial centroids μ_j where $j = \{1, ...k\}$.

3.4.2. Training Strategy

We optimize our networks in two branches. The first branch is a combination of the default IDEC branch with the instance difficulty constraints or global size constraints. The IDEC branch consists of clustering loss ℓ_C and reconstruction loss ℓ_G, this part is purely unsupervised learning. The instance difficulty loss ℓ_I can be treated as an addictive loss towards the IDEC branch which speeds up the learning on "easier" instances and improves model robustness regarding "difficult" instances. Moreover, we apply the global size constraints for all the training instances by concatenating ℓ_G to IDEC branch.

Our second branch consists of pairwise or triplet constraints which includes supervised information, we split the total pairwise or triplet constraints into constraints batches which contain a list of pairs or triplets. During the mini-batch training, we run forward pass for all the instances in the constraints batch and update the network via backpropagating the corresponding loss. The whole network is optimized in an alternative way as one round for clustering branch and one round for pairwise/triplet constraints branch. The training procedure is summarized in Algorithm 1.

Algorithm 1 Deep Constrained Clustering Framework

```
Input: data X, maximum epochs $m$, cluster numbers $k$, batch size $N$ and constraints batch size $N_C$.
Output: latent embeddings $Z$, cluster assignment $S$.
Initialize centroids $\mu$ via k-means on embedding $Z$.
for $epoch = 1$ to $m$ do
  for $batch = 1$ to $N$ do
    Calculate $\ell_C$ via Eqn (3), $\ell_R$ via Eqn (9).
    Calculate $\ell_I$ via Eqn (6) or $\ell_G$ via Eqn (8).
    Calculate total loss as $\ell_C + \ell_R + \{\ell_I || \ell_G\}$.
    Update network parameters based on total loss.
  end for
  for $batch = 1$ to $N_C$ do
    Calculate $\ell_P$ via Eqn (4, 5) or $\ell_T$ via Eqn (7).
    Update network parameters based on $\{\ell_P || \ell_T\}$ .
  end for
  Forward pass to compute $Z$ and $S_i = \text{argmax}_j\, q_{ij}$.
end for
```

4. Experiments

All data and code used to perform these experiments are available online (http://github.com/blueocean92) to help with reproducibility. In our experiments we aim to address the following questions:

- How does our end-to-end deep clustering approach using traditional constraints compare with traditional constrained clustering methods? The latter are given an auto-encoding representation.
- Are the new types of constraints we create for deep clustering method useful in practice?
- Is our end-to-end deep constrained clustering method more robust to the well known negative effects of constraints (Davidson et al., 2006)?

4.1. Datasets

To study the performance and generality of different algorithms, we evaluate the proposed method on two image datasets and one text dataset:

MNIST: Consists of 70000 handwritten digits of 28-by-28 pixel size. The digits are centered and size-normalized in our experiments (LeCun et al., 1998).

FASHION-MNIST: A Zalando's article images-consisting of a training set of 60000 examples and a test set of 10000 examples. Each example is a 28-by-28 grayscale image, associated with a label from 10 classes. This dataset is more complicated than the MNIST (Xiao et al., 2017).

REUTERS-10K: This dataset contains English news stories labeled with a category tree (Lewis et al., 2004). Following the DEC paper (Xie et al., 2016), we used 4 root categories: `corporate/industrial`, `government/social`, `markets` and `economics` as labels and excluded all documents with multiple labels. We randomly sampled a subset of 10000 examples and computed TF-IDF features on the 2000 most frequent words.

4.2. Evaluation Metric

We adopt standard metrics for evaluating clustering performance which measure how close the clustering found is to the ground truth result. Specifically, we employ the following two metrics: normalized mutual information(**NMI**)(Strehl et al., 2000; Xu et al., 2003) and clustering accuracy(**ACC**)(Xu et al., 2003). For data point x_i, let l_i and c_i denote its true label and predicted cluster respectively. Let $l = (l_1, ...l_n)$ and similarity $c = (c_1, ...c_n)$. **NMI** is defined as:

$$\mathbf{NMI}(l, c) = \frac{\mathbf{MI}(l, c)}{max\{H(l), H(c)\}} \quad (10)$$

where $\mathbf{MI}(l, c)$ denotes the mutual information between l and c, and H denotes their entropy. The **ACC** is defined as:

$$\mathbf{ACC}(l, c) = \max_m \frac{\sum_{i=1}^{n} \mathbf{1}\{l_i = m(c_i)\}}{n} \quad (11)$$

where m ranges over all possible one-to-one mappings between clusters and labels. The optimal assignment of m can be computed using the Kuhn-Munkres algorithm (Munkres, 1957). Both metrics are commonly used in the clustering literature and with higher values indicating better clustering results. By using them together we get a better understanding of the effectiveness of the clustering algorithms.

4.3. Implementation and Experiment Results

4.3.1. Implementation

Basic Deep Clustering Implementation. To be comparable with DEC and Improved DEC, we set the encoder network as a fully connected multilayer perceptron with dimensions $d - 500 - 500 - 2000 - 10$ for all datasets, where d is the dimension of input data(features). The decoder network is a mirror of the encoder. All the internal layers are activated by ReLU (Nair & Hinton, 2010) nonlinearity function. For a fair comparison with baseline methods, we used the same greedy layer-wise pre-training strategy to calculate the auto-encoders embedding. To initialize clustering centroids, we run k-means with 20 restarts and select the best solution. We choose Adam optimizer with initiate learning rate 0.001 for all the experiments.

For pairwise constraints experiments. We randomly select pairs of instances and generate the corresponding pairwise constraints between them. To ensure transitivity we calculate the transitive closure overall must-linked instances and then generate entailed constraints from the cannot-link constraints (Davidson & Ravi, 2007). Since our loss function for must-link constraints is combined with reconstruction loss, we use grid search and set the penalty weight for must-link as 0.1.

For instance difficulty constraints experiments. To simulate human-guided instance difficulty constraints, we use k-means as a base learner and mark all the incorrect clustering assignments as difficult instances with confidence 0.1, we also mark the correct instances as easy instances with confidence 1. The reason we set different confidences is that the k-means learner is a weak learner, observing the visualization of generated instance difficulty constraints for MNIST in Figure 1 we can see the first row's instances are pretty easy but the second row's instances are not necessarily difficult. That's why we put low confidence in the generated difficult instance constraints. We also set the penalty weight

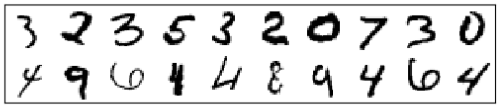


Figure 1. Example of instance difficulty constraints. Top row shows the "easy" instances and second row shows the "difficult" instances.

for instance difficulty loss as 0.1 through all experiments.

For triplet constraints experiments. Triplet constraints are useful when continuous side information exists as there is no explicit category information to generate together/apart constraints. Triplet constraints can state that instance i is more similar to instance j than instance k. To simulate human guidance on triplet constraints, we randomly select n instances as anchors (i), for each anchor we randomly select two instances (j and k) based on the similarity between the anchor. The similarity is calculated as the euclidian distance d between two instances' trusted embedding. The trusted embedding is extracted from our deep clustering network trained with 100000 pairwise constraints. For different dataset, we set different distance thresholds to determine positive/negative instances. The thresholds are set to ensure human can distinguish between positive and negative instances. Figure 2 shows the generated triplets constraints. Through grid search we set the triplet loss margin $\theta = 0.1$.

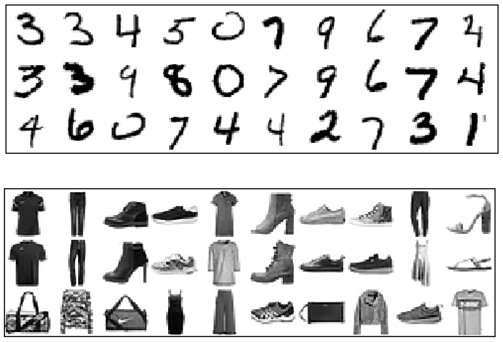


Figure 2. Examples of the generated triplet constraints for MNIST and Fashion. The three rows for each plot shows the anchor instances, positive instances and negative instances correspondingly.

For global size constraints experiments. We apply global size constraints to MNIST and Fashion datasets since they satisfy the balanced size assumptions. The total cluster number is set to 10. We set the penalty term for global constraints as 1 via validation.

4.3.2. Experiments on instance difficulty

In Table 1, we report the average test performance of deep clustering framework without any constraints. In comparison, we report the average test performance of deep clustering framework with instance difficulty constraints in Table 2. Note that the model learned with instance difficulty constraints outperforms the baseline method in all datasets, this shows the weak instance difficulty guidance can leverage the clustering results. To demonstrate the effectiveness of speeding up the learning process, we also show the number of epochs to train until converging. With instance difficulty guidance the model converges faster.

Table 1. The baseline results for our data sets using Improved DEC (Guo et al., 2017) averaged over 20 trials.

	MNIST	Fashon	Reuters
Acc	0.8829 ± 0.0005	0.5874 ± 0.0008	0.7520 ± 0.0007
NMI	0.8612 ± 0.0009	0.6327 ± 0.0011	0.5416 ± 0.0173
Epoch	87.60 ± 12.53	77.20 ± 11.28	12.90 ± 2.03

Table 2. Experiments using instance difficulty constraints (mean $\pm$ std) averaged over 20 trials. c.f. with baselines in Table 1

	MNIST*	Fashion*	Reuters*
Acc	0.9102 ± 0.0034	0.6217 ± 0.0006	0.7801 ± 0.0013
NMI	0.8808 ± 0.0014	0.6495 ± 0.0004	0.5602 ± 0.0021
Epoch	29.70 ± 4.25	47.60 ± 6.98	9.50 ± 1.80

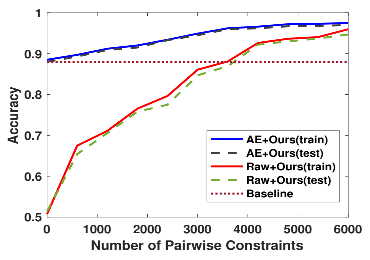
4.3.3. Experiments on pairwise constraints

We randomly generate 6000 pairs of constraints which are a small fractions of possible pairwise constraints for MNIST($2 * 10^{-6}$), Fashion($2 * 10^{-6}$) and Reuters($6 * 10^{-5}$).

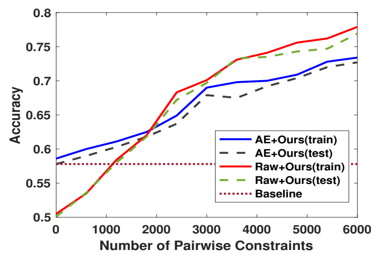
To better understand the contribution of pairwise constraints, we've tested our method with both auto-encoders features and raw data. As can be seen from Figure 3: the clustering performance improves consistently as the constraint amount increases in both settings. Moreover, with just 6000 pairwise constraints the performance on Reuters and MNIST increased significantly especially for the setup with raw data. We also notice that learning with raw data in Fashion achieves a better result than using autoencoder's features. This shows that the autoencoder's features may not always be suitable for DEC's clustering objective. Overall we show pairwise constraints can help reshape the representation and improve the clustering results.

We also compare the results with recent work (Hsu & Kira, 2015): our approach(autoencoders features) outperforms the best clustering accuracy reported for MNIST by a margin of 16.08%, 2.16% and 0.13% respectively for 6, 60 and 600 samples/class. Unfortunately we can't make a comparison with Fogel's algorithm (Fogel et al., 2018) due to the issue in their code repository.

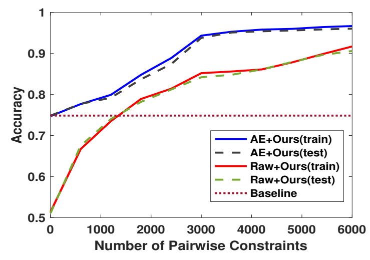
Negative Effects of Constraints. Earlier work (Davidson et al., 2006) showed that for existing constrained clustering algorithms, that the addition of constraints *on average* helps clustering but many individual constraint sets can hurt performance in that it is less than using no constraints. Here we recreate these results even though these classic methods use auto-encoded representations. In Table 3, we report the average performance with 3600 randomly generated pairwise constraints. For each dataset, we randomly generated 100 sets of constraints to test the negative effects of constraints(Davidson et al., 2006). In each run, we fixed the



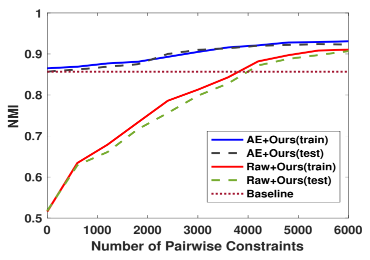
(a) MNIST



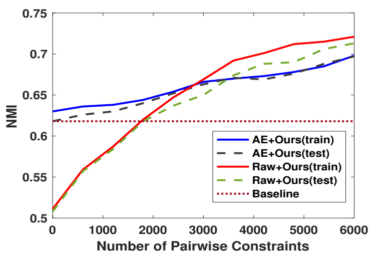
(b) Fashion



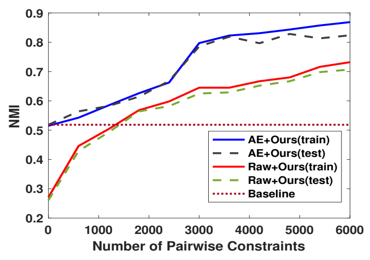
(c) Reuters



(d) MNIST



(e) Fashion



(f) Reuters

Figure 3. Clustering accuracy and NMI on training test sets for different number of pairwise constraints. AE means an autoencoder was used to seed our method. The horizontal maroon colored baseline shows the IDEC's (Guo et al., 2017) test set performance.

random seed and the initial centroids for k-means based methods, for each method we compare its performance between constrained version to unconstrained version. We calculate the negative ratio which is the fraction of times that unconstrained version produced better results than the constrained version. As can be seen from the table, our proposed method achieves significant improvements than traditional constrained clustering algorithms (Wagstaff et al., 2001; Bilenko et al., 2004; Wang & Davidson, 2010).

Table 3. Pairwise constrained clustering performance (mean $\pm$ std) averaged over 100 constraints. Due to the scalability issues we apply flexible constrained spectral clustering with downsampled data(3000 instances and 180 constraints). Negative ratio is the fraction of times using constraints resulted in poorer results than not using constraints.

	Flexible CSP*	COP-KMeans	MPCKMeans	Ours
MNIST Acc	0.628 ± 0.07	0.816 ± 0.06	0.846 ± 0.04	$\mathbf{0.963 \pm 0.01}$
MNIST NMI	0.587 ± 0.06	0.773 ± 0.02	0.808 ± 0.04	$\mathbf{0.918 \pm 0.01}$
Negative Ratio	19%	45%	11%	**0 %**
Fashion Acc	0.417 ± 0.05	0.548 ± 0.04	0.589 ± 0.05	$\mathbf{0.681 \pm 0.03}$
Fashion NMI	0.462 ± 0.03	0.589 ± 0.02	0.613 ± 0.04	$\mathbf{0.667 \pm 0.02}$
Negative Ratio	23%	27%	37%	**6 %**
Reuters Acc	0.554 ± 0.07	0.712 ± 0.0424	0.763 ± 0.05	$\mathbf{0.950 \pm 0.02}$
Reuters NMI	0.410 ± 0.05	0.478 ± 0.0346	0.544 ± 0.04	$\mathbf{0.815 \pm 0.02}$
Negative Ratio	28%	73%	80%	**0 %**

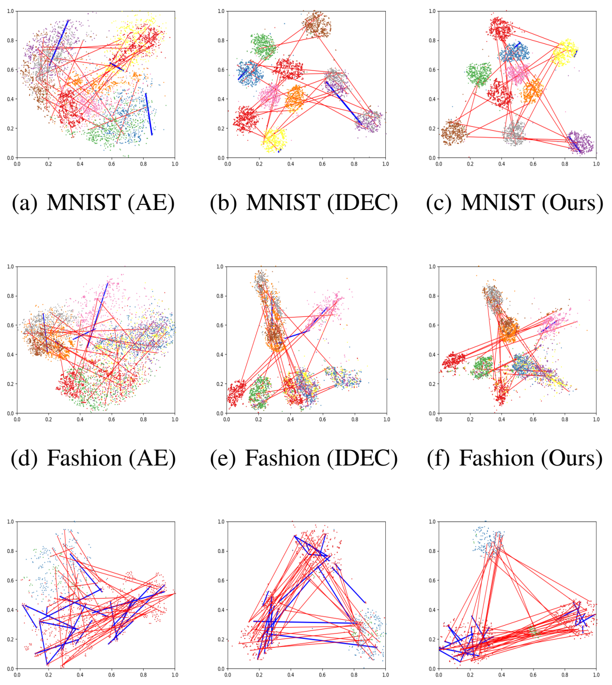
Figure 4 shows the embedded representation of a random subset of instances and its corresponding pairwise constraints using t-SNE to the learned embedding z. Based on Figure 4, we can see the autoencoders embedding is noisy and lot's of constraints are inconsistent based on the definition given (Davidson et al., 2006). Further, we visualize the IDEC's latent embedding and find out the clusters are better separated. However, the inconsistent constraints still exist (blue lines across different clusters and redlines within a cluster), these constraints tend to have negative effects on traditional constrained clustering methods. Finally observed from our method's results we can see the clusters are well separated, the must-links are well satisfied(blue lines are within the same cluster) and cannot-links are well satisfied(red lines are across different clusters).

Combing the visualization with the performance evaluation results in Table 3 we show our model can counter "negative effects" described earlier (Davidson et al., 2006) efficiently via reshaping the latent space based on pairwise constraints. This result has profound practical significance as practitioners typically only have one constraint set to work with.

4.3.4. Experiments on triplet constraints

We experimented on MNIST and FASHION datasets with triplet constraints. Figure 2 visualizes example triplet constraints (based on embedding similarity), note the positive instances are closer to anchors than negative instances.

In Figure 5, we show the clustering Acc/NMI improves consistently as the number of constraints increasing. Comparing with Figure 3 we can find the pairwise constraints can bring slightly better improvements, that's because our

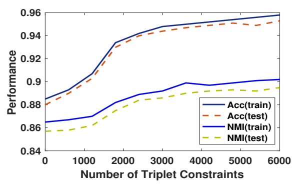


(a) MNIST (AE) (b) MNIST (IDEC) (c) MNIST (Ours)

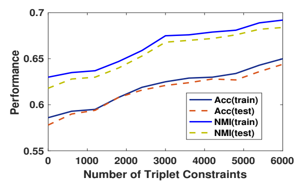
(d) Fashion (AE) (e) Fashion (IDEC) (f) Fashion (Ours)

(g) Reuters (AE) (h) Reuters (IDEC) (i) Reuters (Ours)

Figure 4. We visualize (using t-SNE) the latent representation for a subset of instances and pairwise constraints, we visualize the same instances and constraints for each row. The red lines are cannot-links and blue lines are must-links.

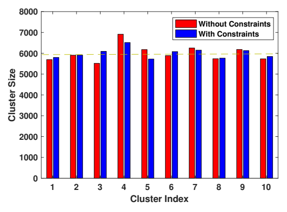


(a) MNIST(Triplet)

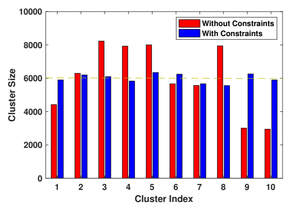


(b) Fashion(Triplet)

Figure 5. Evaluation of the effectiveness of triplet constraints. Both measures range from 0 to 1 and the larger the better.



(a) MNIST



(b) Fashion

Figure 6. Evaluation of the global size constraints. This plot shows each cluster's size before/after adding global size constraints.

triplets constraints are generated from a continuous domain and there is no exact together/apart information encoded in the constraints. Triplet constraints can be seen as a weaker but more general type of constraints.

4.3.5. Experiments on global size constraints

To test the effectiveness of our proposed global size constraints, we have experimented on MNIST and Fashion training set since they both require balanced cluster sizes (see Figure 6). Note that the ideal size for each cluster is 6000 (each data set has 10 classes), we can see that blue bars are more evenly distributed and closer to the ideal size.

We also evaluate the clustering performance with global constraints on MNIST (Acc:0.91, NMI:0.86) and Fashion (Acc:0.57, NMI:0.59). Comparing to the baselines in table 1 we find the performance improved slightly on MNIST but dropped slightly on Fashion.

5. Conclusion and Future Work

The area of constrained partitional clustering has a long history and is widely used. Constrained partitional clustering typically is mostly limited to simple pairwise together and apart constraints. In this paper, we show that deep clustering can be extended to a variety of fundamentally different constraint types including instance-level (specifying hardness), cluster level (specifying cluster sizes) and triplet-level. The latter allows generation of constraints from continuous side information which is explored in other fields but to our knowledge not in constrained clustering.

Our deep learning formulation was shown to advance the general field of constrained clustering in several ways. Firstly, it achieves better experimental performance than well-known k-means, mixture-model and spectral constrained clustering in both an academic setting and a practical setting. For the former on average (i.e. repeated over many constraint sets) our method outperforms standard methods by a large margin (see Table 3). For the later we show that our approach has little of the negative impact of constraints meaning that if a practitioner has just one constraint set (as they typically do), our method is far more likely to perform better than using no constraints.

Most importantly, we were able to show that our method achieves all of the above but still retains the benefits of deep learning such as scalability, out-of-sample predictions and end-to-end learning. We found that even though standard non-deep learning methods were given the same autoencoding representations of the data used to initialize our methods the deep constrained clustering was able to adapt these representations even further. Future work will explore new types of constraints, using multiple constraints at once and extensions to the clustering setting.

References

Bade, K. and Nürnberger, A. Creating a cluster hierarchy under constraints of a partially known hierarchy. In *Proceedings of the 2008 SIAM international conference on data mining*, pp. 13–24. SIAM, 2008.

Basu, S., Bilenko, M., and Mooney, R. J. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 59–68. ACM, 2004.

Basu, S., Davidson, I., and Wagstaff, K. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, 2008.

Bilenko, M., Basu, S., and Mooney, R. J. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 11. ACM, 2004.

Chatziafratis, V., Niazadeh, R., and Charikar, M. Hierarchical clustering with structural constraints. *arXiv preprint arXiv:1805.09476*, 2018.

Davidson, I. and Ravi, S. Intractability and clustering with constraints. In *Proceedings of the 24th international conference on Machine learning*, pp. 201–208. ACM, 2007.

Davidson, I., Wagstaff, K. L., and Basu, S. Measuring constraint-set utility for partitional clustering algorithms. In *Knowledge Discovery in Databases: PKDD 2006*, pp. 115–126. Springer, 2006.

Fogel, S., Averbuch-Elor, H., Goldberger, J., and Cohen-Or, D. Clustering-driven deep embedding with pairwise constraints. *arXiv preprint arXiv:1803.08457*, 2018.

Gress, A. and Davidson, I. Probabilistic formulations of regression with mixed guidance. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pp. 895–900. IEEE, 2016.

Guo, X., Gao, L., Liu, X., and Yin, J. Improved deep embedded clustering with local structure preservation. In *International Joint Conference on Artificial Intelligence (IJCAI-17)*, pp. 1753–1759, 2017.

Hsu, Y.-C. and Kira, Z. Neural network-based clustering using pairwise constraints. *arXiv preprint arXiv:1511.06321*, 2015.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

Joachims, T. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142. ACM, 2002.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

Lu, Z. and Carreira-Perpinan, M. A. Constrained spectral clustering through affinity propagation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8. IEEE, 2008.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

Munkres, J. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Schultz, M. and Joachims, T. Learning a distance metric from relative comparisons. In *Advances in neural information processing systems*, pp. 41–48, 2004.

Shaham, U., Stanton, K., Li, H., Nadler, B., Basri, R., and Kluger, Y. Spectralnet: Spectral clustering using deep neural networks. *arXiv preprint arXiv:1801.01587*, 2018.

Strehl, A., Ghosh, J., and Mooney, R. Impact of similarity measures on web-page clustering. In *Workshop on artificial intelligence for web search (AAAI 2000)*, volume 58, pp. 64, 2000.

Wagstaff, K. and Cardie, C. Clustering with instance-level constraints. *AAAI/IAAI*, 1097:577–584, 2000.

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pp. 577–584, 2001.

Wang, X. and Davidson, I. Flexible constrained spectral clustering. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 563–572. ACM, 2010.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xie, J., Girshick, R., and Farhadi, A. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pp. 478–487, 2016.

Xing, E. P., Jordan, M. I., Russell, S. J., and Ng, A. Y. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pp. 521–528, 2003.

Xu, W., Liu, X., and Gong, Y. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 267–273. ACM, 2003.

Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. *arXiv preprint arXiv:1610.04794*, 2016a.

Yang, J., Parikh, D., and Batra, D. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5147–5156, 2016b.

*

A. Appendix

A.1. General Network Structure

We first visualize the whole network structure(which includes all the types of constraints) for training which in the Figure 7. In our current work we only explore one type of constraints at each run. Normal batch here means all the training instances and clustering batch means the instances within the list of pairwise constraints or triplet constraints.

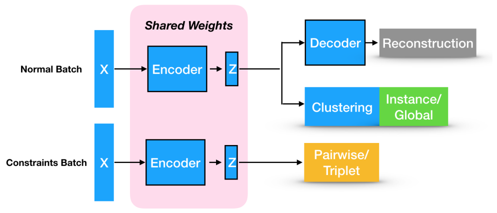


Figure 7. Network Structure for Deep Constrained Clustering's Training

We visualize the prediction network structure in Figure 8. Y is the output which are the same as soft assignments Q introduced in the paper.

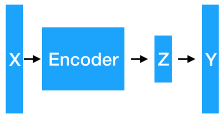


Figure 8. Network Structure for Deep Constrained Clustering's Prediction

A.2. Learning with Instance/Global Constraints

We visualize the training process with instance difficulty constraints or global size constraints in the Figure 9. As can be seen from the figure these two types of constraints are treated as addictive loss concatenating to the clustering loss.

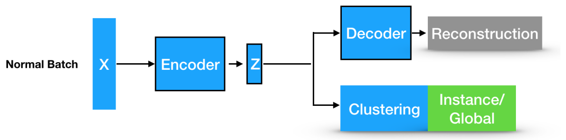


Figure 9. Training Process for Deep Constrained Clustering with Pairwise/Triplet Constraints

A.3. Learning with Pairwise/Triplet Constraints

We visualize the training process with pairwise constraints or triplet constraints in the Figure 10. As can be seen from the figure we train on the same network with two branches alternatively, one branch for pure unsupervised loss and one branch for supervised loss (pairwise/triplet constraints).

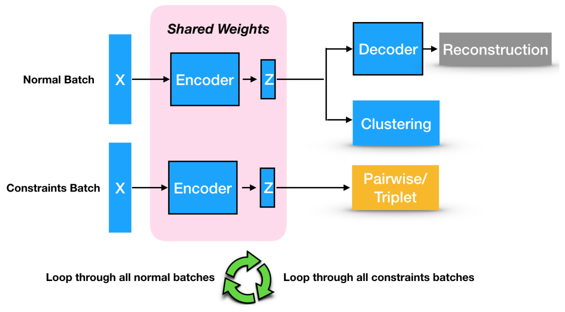


Figure 10. Training Process for Deep Constrained Clustering with Pairwise/Triplet Constraints