NEURAL NETWORK-BASED CLUSTERING USING PAIRWISE CONSTRAINTS

Yen-Chang Hsu
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
yenchang.hsu@gatech.edu

Zsolt Kira
Georgia Tech Research Institute
Atlanta, GA 30318, USA
zkira@gatech.edu

ABSTRACT

This paper presents a neural network-based end-to-end clustering framework. We design a novel strategy to utilize the contrastive criteria for pushing data-forming clusters directly from raw data, in addition to learning a feature embedding suitable for such clustering. The network is trained with weak labels, specifically partial pairwise relationships between data instances. The cluster assignments and their probabilities are then obtained at the output layer by feed-forwarding the data. The framework has the interesting characteristic that no cluster centers need to be explicitly specified, thus the resulting cluster distribution is purely data-driven and no distance metrics need to be predefined. The experiments show that the proposed approach beats the conventional two-stage method (feature embedding with k-means) by a significant margin. It also compares favorably to the performance of the standard cross entropy loss for classification. Robustness analysis also shows that the method is largely insensitive to the number of clusters. Specifically, we show that the number of dominant clusters is close to the true number of clusters even when a large k is used for clustering.

1 INTRODUCTION

Performing end-to-end training and testing using deep neural networks to solve various tasks has become a dominant approach across many fields due to its performance, efficiency, and simplicity. Success across a diverse set of tasks has been achieved in this manner, including classification of pixel-level information into high level categories (Krizhevsky et al., 2012), pixel-level labeling for image segmentation (Long et al., 2015; Zheng et al., 2015), robot arm control (Levine et al., 2015), speech recognition (Graves & Jaitly, 2014), playing Atari games (Mnih et al., 2015) and Go (Clark & Storkey, 2015; Silver et al., 2016). All of the above techniques largely avoid sophisticated pipeline implementations and human-in-the-loop tuning by adopting the concept of training the networks to learn the target problem directly.

Clustering, a classical machine learning problem, has not yet been fully explored in a similar manner. Although there are some two-stage approaches that have tried to learn the feature embedding specifically for clustering, they still require using other clustering algorithms such as k-means to determine the actual clusters at the second step. Specifically, the first stage of previous works usually assume how the data is distributed in the projected space using human-chosen criteria such as self-reconstruction, local relationship preservation, sparsity (Tian et al., 2014; Huang et al., 2014; Shao et al., 2015; Wang et al., 2016; Chen, 2015; Song et al., 2013), fitting predefined distributions (Xie et al., 2015), or strengthening of neighborhood relationships (Rippel et al., 2015) to learn the feature embedding. Furthermore, all of these techniques then use a metric, such as Euclidean or cosine distance, in the second stage. This further introduces human-induced bias via strong assumptions and the chosen metric may not necessarily be appropriate for the embedded space. In other words, there has not been a method to solve the two sub-problems (learning a feature space and performing clustering within that feature space) jointly in an end-to-end manner.

In this work, we propose a framework which minimizes such assumptions by training a network that can directly assign the clusters at the output layer. We specifically use weak labels, in the form

of pairwise constraints or similar/dis-similar pairs, to learn the feature space as well as output a clustering. It is worth emphasizing that such weak labels could be obtained automatically (in an unsupervised manner) based on spatial or temporal relationships, or using a neighborhood assumption in the feature space similar to the above works. One could also get the weak labels from the ground-truth obtained from crowd-sourcing. In many cases, it may be an easier task for a human to provide pair-wise relationships rather than direct assignment of class labels (e.g. when dealing with attribute learning).

In order to adopt the raw data and weak labels for end-to-end clustering, we present the novel concept of constructing the cost function in a manner that incorporates contrastive KL divergence to minimize the statistical distance between predicted cluster probabilities for similar pairs, while maximizing the distance for dissimilar pairs. In the latter sections, we will show that the framework is extremely easy to realize by rearranging existing functional blocks of deep neural networks, so it has large flexibility to adopt new layer types, network architectures, or optimization strategies for the purpose of clustering.

One significant property of the proposed end-to-end clustering is that there are no cluster centers explicitly represented. This largely differs from all of the works mentioned above. Without the centers, no explicit distance metrics need to be involved for deciding the cluster assignment. The learning of the cluster assignments is purely data-driven and is implicitly handled by the parameters and the non-linear operations of the network. Of course the outputs of the last hidden layer could be regarded as the learned features, however it is not necessary to interpret it using predefined metrics such as Euclidean or cosine distance. The networks will find the best way to utilize the embedded feature space during the same training process in order to perform clustering. The experimental sections will demonstrate this property, in addition to strong robustness when the number of output clusters is varied. In such cases, the network tends to output a clustering that only utilizes the same number of nodes as there are clusters intrinsically in the data.

Furthermore, since the proposed framework can learn the cluster assignments using the proposed contrastive loss, it opens up the possibility of directly comparing its accuracy to the standard cross-entropy loss when full labels are available. This is achieved by developing an implementation that can efficiently utilize such dense pairwise information. This implementation strategy and experiments are also presented in sections 2.2 and 3.3, showing favorable results compared to the standard classification approach. Source code in Torch is provided on-line.

1.1 Related Works

A common strategy to utilize pairwise relationship with neural networks is the Siamese architecture (Bromley et al., 1993). The concept had been widely applied to various computer vision topics, such as similarity metric learning (Chopra et al., 2005), dimensionality reduction (Hadsell et al., 2006), semi-supervised embedding (Weston et al., 2008), and some applications to image data, such as in learning to match patches (Han et al., 2015; Zagoruyko & Komodakis, 2015) and feature points (Simo-Serra et al., 2015). The work of Mobahi et al. (2009) uses the coherence nature of video as a way to collect the pairwise relationship and learn its features with a Siamese architecture. The similar idea of leveraging temporal data is also presented in the report of Goroshin et al. (2015). In addition, the triplet networks, which could be regarded as an extension of Siamese, gained significant success in the application of learning fine-grained image similarity (Wang et al., 2014) and face recognition (Schroff et al., 2015). Despite the wide applicability of the Siamese architecture, there is no report exploring them from the clustering perspective. Furthermore, while some works try to maximize the information in a training batch by carefully sampling the pair (Han et al., 2015) or by formulating it as a triplet (Wang et al., 2014), there is no work showing how to use dense pairwise information directly and efficiently.

Our proposed implementation strategy can efficiently utilize any amount of pairwise constraints from a dataset to train a neural network to perform clustering. When the full set of constraints is given, it can compare to the vanilla networks trained using supervised classification. If only partial pairwise constraints are available, the problem is similar to semi-supervised clustering. There is a long list of previous work related to the problem. For example, COP-Kmeans (Wagstaff et al., 2001) forced the clusters to comply with constraints and Rangapuram & Hein (2012) added terms in spectral clustering to penalize the violation of constraints. The more closely related works perform

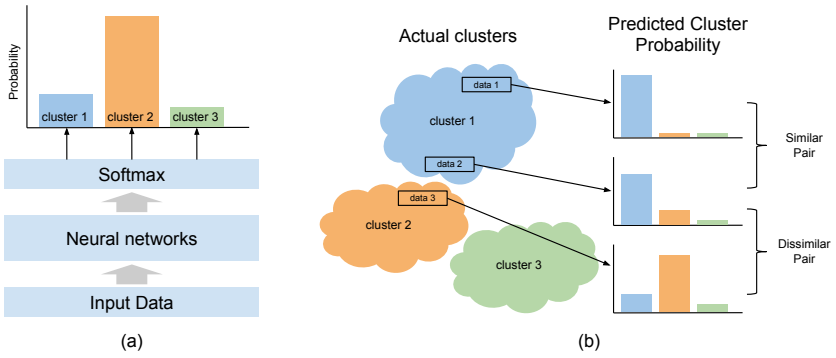

Figure 1: Illustration of (a) how neural networks output the distribution of possible clusters given a sample, (b) the example of predicted cluster distribution between similar/dissimilar pairs.

metric learning (Bilenko et al., 2004) or feature re-weighting (De Amorim & Mirkin, 2012) during the clustering process. The recent approaches TVClust and RDP-means (Khashabi et al., 2015) address the problem with probabilistic models. None of these approaches, however, jointly learn the feature space in addition to clustering.

In the next sections we will explain how to inject the concept of clusters into a neural network formulations. The experiments on two image datasets will be presented in the third section, demonstrating the efficacy of the approach.

2 The End-to-End Clustering Networks

Consider the vanilla multilayer perceptron (MLP) used for classification tasks: Each output node is associated with predefined labels and the optimization minimizes a cost function, such as cross entropy, that compares the output labels (or the distribution over the labels) provided by the network for a set of instances and the corresponding ground truth labels. We start from this model and remove the hard association between labels and network outputs. The idea is to only use pairwise information and define the output nodes in a manner such that they can represent a clustering of the data. In other words, which node will correspond to which cluster (or object class) is dynamically decided during the training process. To achieve this, we formulate an approach that only needs to modify the cost criterion above the softmax layer of any neural network which was designed for a classification task. We therefore present a new pairwise cost function for clustering that can take the place of, or be combined with, the traditional supervised classification loss functions. This flexibility allows the network to use both types of information, depending on which is available.

2.1 Pairwise KL-Divergence

While the output of the traditional softmax layer represents the probability that a sample belongs to the class labels (or clusters in our problem), the outputs of the whole softmax layer could be viewed as the distribution of possible clusters given a sample (Figure 1a). If the data only contains a single concept, such as in the case of hand-written digits, then the distributions between the softmax output for a similar pair should be similar. Conversely, the distribution over the class labels should be dissimilar if the pair belongs to different clusters (Figure 1b). The similarity between distributions could be evaluated by statistical distance such as Kullback-Leibler (KL) divergence. Traditionally this can be used to measure the distance between the output distribution and ground truth distribution. In our case, however, it can instead be used to measure the distance between the two output distributions given a pair of instances. Given a pair of distributions $\boldsymbol{P}$ and $\boldsymbol{Q}$, obtained by feeding data x_p and x_q into network f, we will fix $\boldsymbol{P}$ first and calculate the divergence of $\boldsymbol{Q}$ from $\boldsymbol{P}$. Assume the network has k output nodes, then the total divergence will be the sum over k nodes. To turn the divergence into a cost, we define that if $\boldsymbol{P}$ and $\boldsymbol{Q}$ come from a similar pair, the cost will be plain KL-divergence; otherwise, it will be the hinge loss (still using divergence). The indicator

functions I_s in equation 2 will be equal to one when (x_p, x_q) is a similar pair, while I_{ds} works in reverse manner. In other words:

$$\boldsymbol{P} = f(x_p), \boldsymbol{Q} = f(x_q),$$

$$KL(\boldsymbol{P} \parallel \boldsymbol{Q}) = \sum_{i=1}^{k} P_i log(\frac{P_i}{Q_i}), \tag{1}$$

$$loss(\boldsymbol{P} \parallel \boldsymbol{Q}) = I_s(x_p, x_q) KL(\boldsymbol{P} \parallel \boldsymbol{Q}) + I_{ds}(x_p, x_q) \max(0, margin - KL(\boldsymbol{P} \parallel \boldsymbol{Q})). \tag{2}$$

Since the cost should be calculated from fixing both P or Q (i.e. symmetric), the total cost L of the pair x_p, x_q is the sum of both directions:

$$L(\boldsymbol{P}, \boldsymbol{Q}) = loss(\boldsymbol{P} \parallel \boldsymbol{Q}) + loss(\boldsymbol{Q} \parallel \boldsymbol{P}). \tag{3}$$

To calculate the derivative of cost L, it is worth to note that the P in the first term of equation 3 (and Q in the second term) is regarded as constant instead of variable. Thus, the derivative could be formulated as:

$$\begin{aligned} \frac{\partial}{\partial Q_i} L(\boldsymbol{P}, \boldsymbol{Q}) &= \frac{\partial}{\partial Q_i} loss(\boldsymbol{P} \parallel \boldsymbol{Q}), \\ &= \begin{cases} -\frac{P_i}{Q_i} & \text{if} \quad I_s(x_p, x_q) = 1, \\ \frac{P_i}{Q_i} & \text{elseif} \quad KL(\boldsymbol{P} \parallel \boldsymbol{Q})) < margin, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

$$\begin{aligned} \frac{\partial}{\partial P_i} L(\boldsymbol{P}, \boldsymbol{Q}) &= \frac{\partial}{\partial P_i} loss(\boldsymbol{Q} \parallel \boldsymbol{P}), \\ &= \begin{cases} -\frac{Q_i}{P_i} & \text{if} \quad I_s(x_p, x_q) = 1, \\ \frac{Q_i}{P_i} & \text{elseif} \quad KL(\boldsymbol{Q} \parallel \boldsymbol{P})) < margin, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

With the defined derivatives of cost, the standard back-propagation algorithm can be applied without change.

2.2 Efficient implementation to Utilize Pairwise Constraints

Equation 2 is in the form of contrastive loss that is suitable to be trained with Siamese networks (Hadsell et al., 2006). However, when the amount of pairwise constraints increases, it is not efficient to enumerate all pairs of data and feed them into Siamese networks. Specifically, if there is a mini-batch that has pairwise constraints between any two samples, the number of pairs that have to be fed into the networks will be $n(n-1)/2$ where n is mini-batch size. However, a redundancy occurs when a sample has more than one constraint associated with it. In such cases the sample will be fed-forward multiple times. However, feed-forward once for each sample is sufficient for calculating the pairwise cost in a mini-batch. Figure 2c demonstrates an example for the described situation. The data with index 1 and 3 are fed-forward twice in vanilla Siamese networks to enumerate the three pairwise relationships: (1,2),(1,3), and (3,4). To avoid the redundancy of computation, we apply a strategy of enumerating the pairwise relationships only in the cost layer, instead of instantiating the Siamese architecture. This strategy simplified the implementation of neural networks which utilize pairwise relationship. Our proposed architecture is shown in the Figure 2b. The pairwise constraints only need to be presented to the cost layer in the format of tuples $T : (i, j, relationship)$ where i and j are the index of sample inside the mini-batch and $relationship$ indicates similar/dissimilar pair. Each input data is therefore only fed-forward once in a mini-batch and its full/partial pairwise relationships are enumerated as tuples.

Concretely, the gradients for the back-propagation in a mini-batch are calculated as:

$$\frac{\partial}{\partial f(x_i)} \hat{L} = \sum_{\forall j;(i,j) \in T} \frac{\partial}{\partial f(x_i)} L(f(x_i), f(x_j)). \tag{6}$$

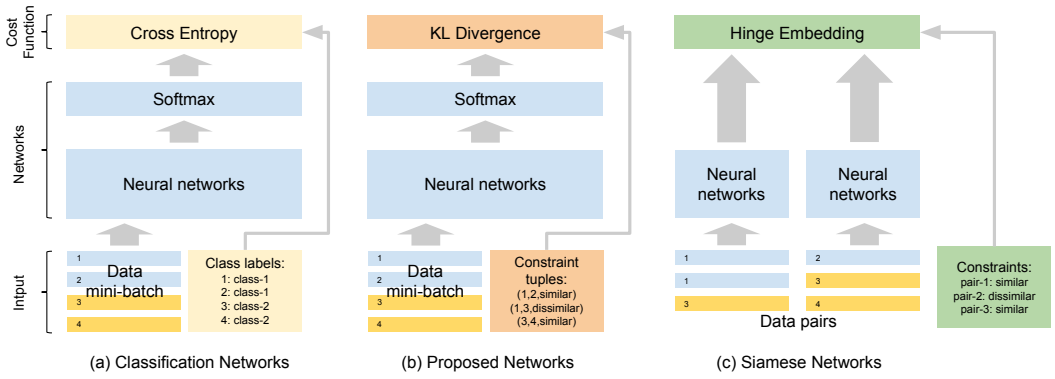

Figure 2: The comparison between (a) classification networks, (b) our proposed networks, and (c) Siamese networks. The parts that differ across architectures are shown with distinct colors. In (a) and (b), the numbers in the data represent the index of the input data in a mini-batch.

One could see our proposed architecture (Figure 2b) is highly similar to the standard classification networks (Figure 2a). As a result of this design, ideas in the above two sections could be easily implemented as a cost criterion in the torch *nn* module. Then a network could be switched to either classification mode or clustering mode by simply changing the cost criterion. We therefore implemented our approach in Torch, and have released the source on-line [1].

It is also worth mentioning that the presented implementation trick is not specifically for the designed cost function. Any contrastive loss could benefit from the approach. Since there is no openly available implementation to address this aspect, we include it in our released demo source.

3 Experiments

We evaluate the proposed approach on the MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky & Hinton, 2009) datasets. The two datasets are both normalized to zero mean and unit variance. The convolutional neural networks architecture used in these experiments is similar to LeNet (LeCun et al., 1998). The network has 20 and 50 5x5 filters for its two convolution layers with batch normalization (Ioffe & Szegedy, 2015) and 2x2 max-pooling layers. We use the same number of filters for both MNIST and CIFAR-10 experiments. The two subsequent fully connected layers have 500 and 10 nodes. Both convolutional and the first fully connected layers are followed by rectified linear units. The only hyper-parameter in our cost function is the $margin$ in equation 2. The margin was chosen by cross-validation on the training set. There is no significant difference when the margin was set to 1 or 2. However, it has a higher chance of converging to a lower training error when the margin is 2, thus we set it to the latter value across the experiments. To minimize the cost function, we applied mini-batch stochastic gradient descent.

3.1 Clustering with partial constraints

We performed three sets of experiments to evaluate our approach. The first experiment seeks to demonstrate how the approach works with partial constraints. In this case, we use a clustering metric to demonstrate how good the resulting clustering is. The constraints are uniformly sampled from the full set, i.e, $\#full\text{–}constraints = n(n-1)/2$, where n is the size of training set. The pairwise relationship is converted from the class label. If a pair has the same class label, then it is a similar pair, otherwise it is dissimilar. We did not address the fact that the amount of dissimilar pairs usually dominates the pairwise relationship (which is more realistic in many application domains), especially when the number of classes is large. In our experiments for this section, the ratio between the number of similar and dissimilar pairs is roughly 1:9.

[1] http://github.com/yenchanghsu/NNclustering

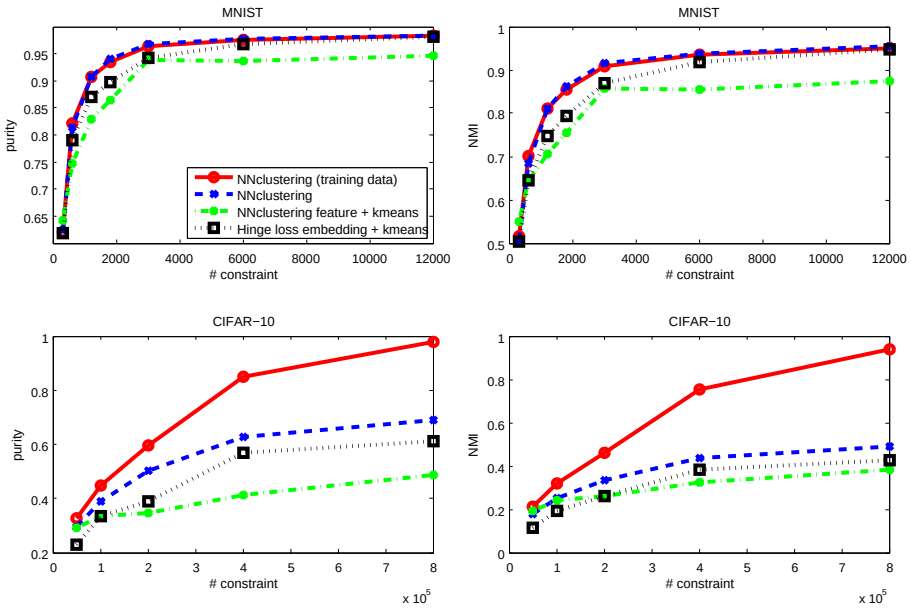

Figure 3: The results of clustering with partial pairwise constraints. The #constraint axis is the number of sampled pairwise relationship in the training data. The clustering and training is simultaneously applied on the training data (red line). The testing data (for blue, green, black lines) is used to validate if the feature space learned during clustering has generalizability. NNclustering is our proposed method. The *NNclustering feature + kmeans* uses the outputs at the last hidden layer (500-D) as the input for k-means. The baseline networks (black line) were trained with hinge loss of Euclidean distance. The evaluation metric in the first column is purity, while the second column shows the NMI score.

We evaluate the resulting clusters with the purity measure and normalized mutual information (NMI) (Strehl & Ghosh, 2003). The index of cluster for each sample is obtained by feed-forwarding the training/testing data into the trained networks. Note that we collect the clustering results of training data after the training error has converged, i.e, feed the training data one more time to collect the outputs after the training phase. We picked the networks which have the lowest training loss among five random restarts while the set of constraints are kept the same.

Figure 3 shows that on MNIST the clustering could still achieve high accuracy when constraints are extremely sparse. With merely 1200 constraints, which were randomly sampled from the pairwise relationship of full (60000 samples) training set, it achieves $>$0.9 purity and $>$0.8 NMI scores. Note that the training samples without any constraint associated to it has no contribution to the training. Thus, the scheme is not the same as the semi-supervised clustering framework in previous works (Wagstaff et al., 2001; Bilenko et al., 2004; De Amorim & Mirkin, 2012) where their unlabeled data contribute to calculating the centers of the clusters. The lack of explicit cluster centers provides the flexibility to learn more complex non-linear representations, so the proposed algorithm could still predict the cluster of unseen data without knowing the cluster centers. In the experiments with MNIST, we could see the performance of testing data has no degradation. It is mainly because the networks could learn the clustering with so few constraints such that most of the training data have no constraints and act like unseen data. Note that although no directly comparable results for MNIST have been reported for our specific problem formulation, results for the closest problem setting can be seen in Figure 4(b) of Li et al. (2009) which achieves similar results for a subset of the classes and a much smaller number of constraints (only about 3k). Hence, our results are competitive with theirs but our approach is scalable enough to allow the use of many more constraints and continues to improve while their approach seems to plateau.

To demonstrate the advantage of performing joint clustering and feature learning, we also applied the k-means algorithm with the features learned at the last hidden layer, which has 500 dimensions. The k-means algorithm used Euclidean or cosine distance and was deployed with 50 random restarts

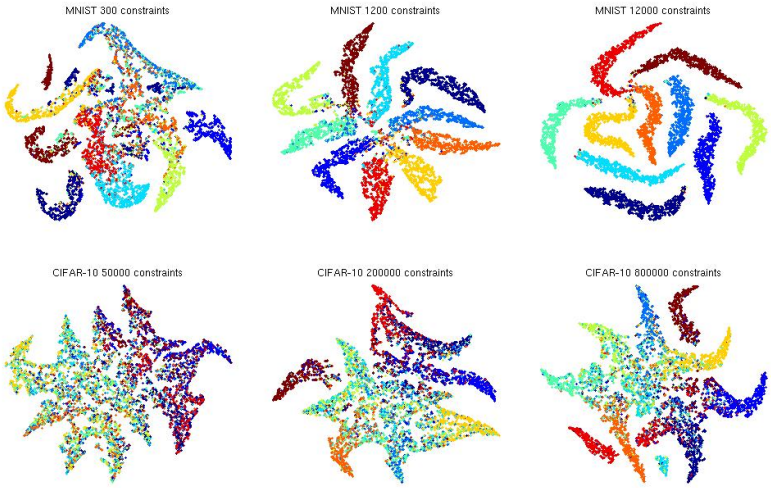

Figure 4: The visualization of clustering. The figure was created by using the outputs of the softmax layer as the input for t-SNE (Van der Maaten & Hinton, 2008). Only testing data are shown. The networks used in the first row are trained with 300, 1200, and 12000 pairs of constraints in MNIST training set. The second row is trained with 50k, 200k, and 800k constraints in CIFAR-10.

on the testing set. We report the clustering results of k=10 which has the lowest sum of point-to-centroid distances among 50 restarts. Since the dimensionality is relatively high, the performance of using Euclidean and cosine distance showed minor difference. The results in Figure 3 show that the jointly trained last layer utilize the outputs of last hidden layer much better than k-means.

To construct the baseline approach, we use the common strategy of training a Siamese networks with standard hinge loss embedding criteria in *torch nn* package, then perform k-means on the networks' outputs. The baseline networks have the same architecture except the softmax layer and the loss function. Figure 3 shows that the proposed clustering framework beats the baseline with a significant margin when the number of constraints is few in the easy dataset (purity is $\sim 5\%$ better in MNIST) or when the dataset is harder (purity is $15 \sim 50\%$ better in CIFAR-10).

The experiments with CIFAR-10 provides some idea of how the approach works on a more difficult dataset. The required constraints to achieve reasonable clustering is much higher. Eight constraints/sample (400,000 total constraints) is required to reach a 0.8 purity score with the same network. The performance on unseen data is also degraded because the networks is over-fitting the constraints. The degradation could possibly be mitigated by adding some regularization terms such as dropout. While any general regularization strategy could be applied in the proposed scheme, we do not address it in this work. Nevertheless, the clustering on the training set is still effective with sparse constraints, e.g., it is able to reach a purity of ~ 1 with only 16 constraints/sample on CIFAR-10. The visualization in Figure 4 provides more intuition about the clustering results trained with different numbers of constraints.

3.2 Robustness of Clustering

3.2.1 Adding Noise

Noisy constraints are likely to occur when the pairwise relationships are generated in an automatic/unsupervised way. We simulated this scenario by flipping the sampled pairwise relationship. Since the ratio of similar pair and dissimilar pair is 1:9, adding 10% noise will introduce equal amount of false-similar pair as the amount of true-similar pair. The clustering performance in Figure 5 (left) shows the reasonable tolerance against noise. We would like to point out that when noise is less than 10%, the performance degradation is reduced when the number of constraints increased. This means that the proposed method could achieve higher performance by adding more pairwise

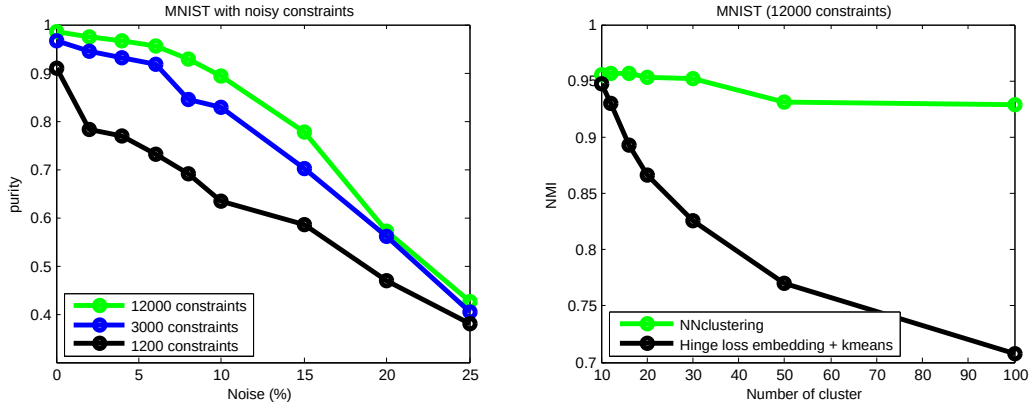

Figure 5: The robustness evaluation of the proposed clustering method. Left figure is the result of adding noisy constraints into MNIST, while the right figure simulates the case when the number of clusters is unknown.

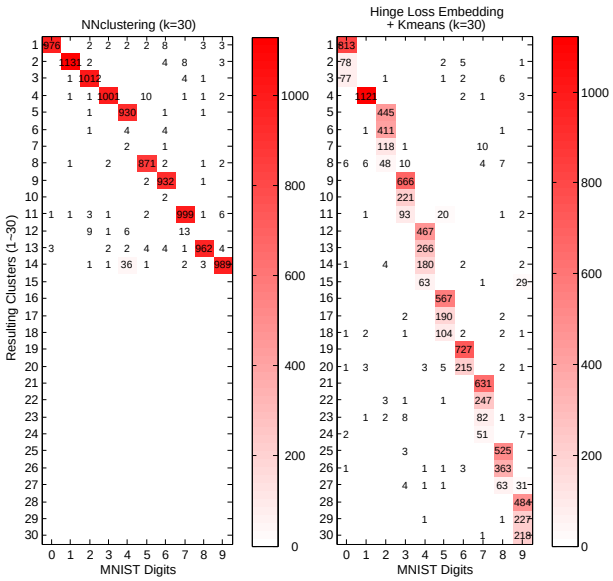

Figure 6: The contingency tables of resulting clusters. It only shows k=30 (same experiment in the right part of figure 5) for the ease of visualization. NNclustering produces similar result even when k=100. The numbers in the table show the amount of samples been assigned to the cluster, while the blank rows indicate empty clusters. Higher numbers in fewer positions is the preferred result for clustering.

information while keeping the ratio of noise the same. Real applications would benefit from this property since adding more weakly labeled data is cheap and the noise level of automatically generated constraints are usually the same.

3.2.2 Changing Number of Clusters

Another common scenario is that the number of target clusters is unknown. Since the purity metric is not sensitive to the number of clusters, NMI is more appropriate in this evaluation. We performed the experiment with 12,000 constraints for training, which include ≈1200 similar pairs in MNIST. The testing results in the right of figure 5 show that the proposed method is almost not affected by

Table 1: Comparing the testing accuracy between classification and clustering using same networks architecture. The clustering is trained with full pairwise relationships obtained from ground-truth class labels. The separated testing set (10,000 samples) is used in this evaluation.

Training approach	Classification	Clustering
Training data:		
MNIST 6 sample/class	**82.4%**	79.4%
MNIST 60 sample/class	94.7%	**95.1%**
MNIST 600 sample/class	98.3%	**98.8%**
MNIST full (≈6000 sample/class)	99.4%	**99.6%**
Training data:		
CIFAR-10 5 sample/class	21.3%	**22.0%**
CIFAR-10 50 sample/class	34.6%	**37.0%**
CIFAR-10 500 sample/class	**55.0%**	53.2%
CIFAR-10 full (5000 sample/class)	**73.7%**	73.4%

increasing the number of clusters. Even in the condition of 100 clusters (by setting 100 output nodes in our networks), the performance only decreases by a very small amount. In fact, in figure 6 most of the data were assigned to ≈10 major clusters and left other clusters being empty. In contrast, the kmeans-based approach (hinge loss embedding + kmeans) is susceptible to the number of clusters and usually divide a class into many small clusters.

3.3 Clustering VS Classification

The final set of experiments compares the accuracy of our approach with a pure classification task in order to get an upper bound of performance (since full labels can be used to create a full set of constraints) and see whether our approach can leverage pairwise constraints to achieve similar results. To make the results of clustering (contrastive loss) comparable to classification (cross-entropy loss), the label of each cluster is obtained from the training set. Specifically, we make the number of output nodes to be the same as the true number of classes, thus we could assign each output node with a distinct label using the optimal assignment. The results in Table 1 show our cost function achieved slightly higher or comparable accuracy in most of the experiment settings. The exception is MNIST with 6 samples/class. The reason is that the proposed cost function creates more local minimum. If the training data is too few, then the training will be more likely to be trapped in certain local minimum. Note that we also applied a random restart strategy (randomly initializing the parameters of the network) to find a better clustering result based on the training set, which is a common strategy used in typical clustering procedures. We ran 5 randomly initialized networks to perform clustering and chose the network that had the highest training accuracy and then used the resulting network to predict the clusters on the testing set.

We also performed the experiments using the same architecture applied to a harder dataset, i.e, CIFAR-10. We did not pursue optimal performance on the dataset, but instead used it to compare the performance difference of learning between the classification and clustering networks. The results show that they are fully comparable. Since CIFAR-10 is a much more difficult dataset compared to MNIST, the overall drop of accuracy on CIFAR-10 is reasonable. Even in the extreme case when the number of training samples is small, the proposed architecture and cost function proved effective.

4 Conclusion and Future Works

We introduce a novel framework and construct a cost function for training neural networks to both learn the underlying features while, at the same time, clustering the data in the resulting feature space. The approach supports both supervised training with full pairwise constraints or semi-supervised with only partial constraints. We show strong results compared to traditional K-means clustering, even when it is applied to a feature space learned by a Siamese network. Our robustness analysis not only shows good tolerance to noise, but also demonstrates the significant advantage of our method when the number of clusters is unknown. We also demonstrate that, using only pairwise constraints, we can achieve equal or slightly better results than when explicit labels are available

and a classification criterion is used. In addition, our approach is both easy to implement for existing classification networks (since the modifications are in the cost layer) and can be efficiently implemented.

In future work, we plan to deploy the approach using deeper network architectures on datasets that have a larger number of classes and instances. We hope that this work inspires additional investigation into feature learning via clustering, which has been relatively less explored. Given the abundance of available data and recent emphasis on semi or unsupervised learning as a result, we believe this area holds promise for analyzing and understanding data in a manner that is flexible to the available amount and type of labeling.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation and National Robotics Initiative (grant # IIS-1426998).

REFERENCES

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

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

Gang Chen. Deep learning with nonparametric clustering. *arXiv preprint arXiv:1501.03084*, 2015.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pp. 539–546. IEEE, 2005.

Christopher Clark and Amos Storkey. Training deep convolutional neural networks to play go. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1766–1774, 2015.

Renato Cordeiro De Amorim and Boris Mirkin. Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering. *Pattern Recognition*, 45(3):1061–1075, 2012.

Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised feature learning from temporal data. *arXiv preprint arXiv:1504.02518*, 2015.

Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772, 2014.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pp. 1735–1742. IEEE, 2006.

Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3279–3286, 2015.

Peihao Huang, Yan Huang, Wei Wang, and Liang Wang. Deep embedding network for clustering. In *2014 22nd International Conference on Pattern Recognition (ICPR)*, pp. 1532–1537. IEEE, 2014.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Daniel Khashabi, Jeffrey Yufei Liu, John Wieting, and Feng Liang. Clustering with side information: From a probabilistic model to a deterministic algorithm. *arXiv preprint arXiv:1508.06235*, 2015.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

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

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *arXiv preprint arXiv:1504.00702*, 2015.

Zhenguo Li, Jianzhuang Liu, and Xiaoou Tang. Constrained clustering via spectral regularization. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 421–428. IEEE, 2009.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CVPR (to appear)*, November 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 737–744. ACM, 2009.

Syama Sundar Rangapuram and Matthias Hein. Constrained 1-spectral clustering. *International conference on Artificial Intelligence and Statistics*, 2012.

Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. Metric learning with adaptive density discrimination. *arXiv preprint arXiv:1511.05939*, 2015.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 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.

Ming Shao, Sheng Li, Zhengming Ding, and Yun Fu. Deep linear coding for fast graph clustering. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 3798–3804. AAAI Press, 2015.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the International Conference on Computer Vision*, number EPFL-CONF-213228, 2015.

Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 117–124. Springer, 2013.

Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.

Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI*, pp. 1293–1299, 2014.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pp. 577–584, 2001.

Jiang Wang, Yang Song, Tommy Leung, Catherine Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 1386–1393. IEEE, 2014.

Zhangyang Wang, Shiyu Chang, Jiayu Zhou, and Thomas S Huang. Learning a task-specific deep architecture for clustering. *Proceedings of SIAM Conference on Data Mining (SDM)*, 2016.

Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, pp. 1168–1175, 2008.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. *arXiv preprint arXiv:1511.06335*, 2015.

Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1529–1537, 2015.