NEURAL NETWORK-BASED CLUSTERING USING PAIR-WISE 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

In this work, we address the problem of finding a clustering of high-dimensional data using only pairwise constraints provided as input. Our strategy utilizes the back-propagation algorithm for optimizing neural networks to discover the clusters, while at the same time the features are also learned during the same training process. In order to do this, we design a novel architecture that can incorporate cost functions associated with KL divergence in order to minimize distance for similar pairs while maximizing the distance for dissimilar pairs. We also propose an efficient implementation that optimizes the parameters of the architecture more efficiently than a naive implementation, e.g. via Siamese networks. Experiments on Mnist and Cifar-10 show that the accuracy of the proposed approach is comparable or exceeds the results of classification. Reasonable clusters could also be discovered when only partial pairwise constraints were available.

1 INTRODUCTION

Neural networks have achieved immense success in many machine learning tasks in terms of classification, regression, and feature learning. Among these tasks, clustering has received less attention recently and neural approaches are usually addressed in an unsupervised fashion using techniques such as self-organizing maps (Kohonen, 1982), neural gas (Martinetz et al., 1993), or adaptive resonance theory (Grossberg, 1976). Several variations have also appeared, for example discussed in the work of Du (2010). In this paper, we explore the possibility of doing joint feature learning and clustering simultaneously using neural networks. The input of our problem is the raw data and weakly labeled information, specifically pairwise constraints. The output is a trained neural network which can extract features and assign the input data to a cluster. To our best knowledge, this is the first work to learn feature representation via clustering with neural networks.

This problem setting has several advantage over classification in the sense of label collection. The pairwise constraints usually represent similar/dis-similar pairs, or must/cannot links, which can be collected automatically based on spatial or temporal relationships. In some cases, it also may be an easier task for a human to provide pair-wise constraints rather than direct assignment of class labels (e.g. when dealing with attribute learning). Thus more choices of label collecting strategies would be available in the proposed scheme. In this work, we specifically focus on the formulation for jointy learning the features and clustering but do not explore the practices of weak label collection which will be explored in future work.

The main contributions of this work is that we: (1) show that clustering with pairwise constraints can be achieved with neural networks, (2) demonstrate that using KL-divergence can be a good strategy for pushing features to create a good clustering of the data, (3) show that clustering with partial constraints is viable in our proposed framework, and (4) present an architecture which can efficiently utilize denser pairwise information where multiple constraints are available for a sample.

1.1 RELATED WORKS

A common strategy to utilize the 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 Siamese architecture. The similar idea of leveraging temporal data is also be presented in the report of Goroshin et al. (2015). In addition, the triplet networks, which could be regarded as an extension of Siamese, gain 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 Siamese architecture, there is no report to explore 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 report showing how to use dense pairwise information directly.

Our proposed framework 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 are 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 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 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 USING NEURAL NETWORKS FOR JOINT FEATURE LEARNING AND CLUSTERING

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 softmax, 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

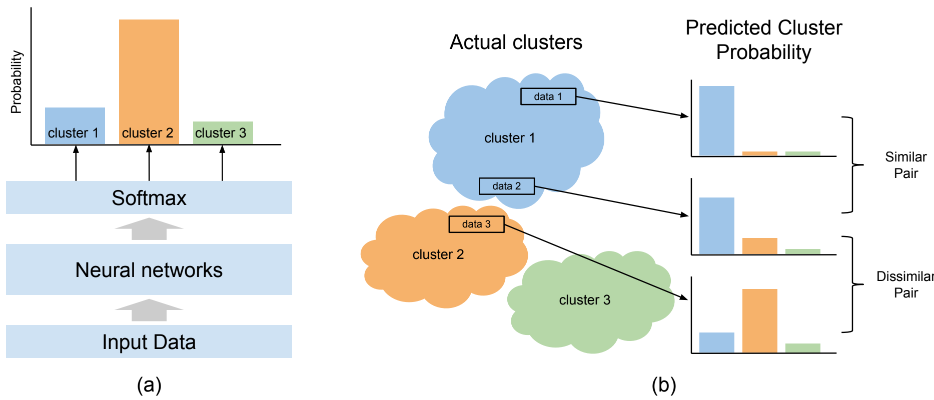

Figure 1: The illustration of (a) how neural networks output the distribution of possible clusters given a sample, (b) the example of predicted cluster distribution between similar/dissimilar pair.

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 to be 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:

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

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

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

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.

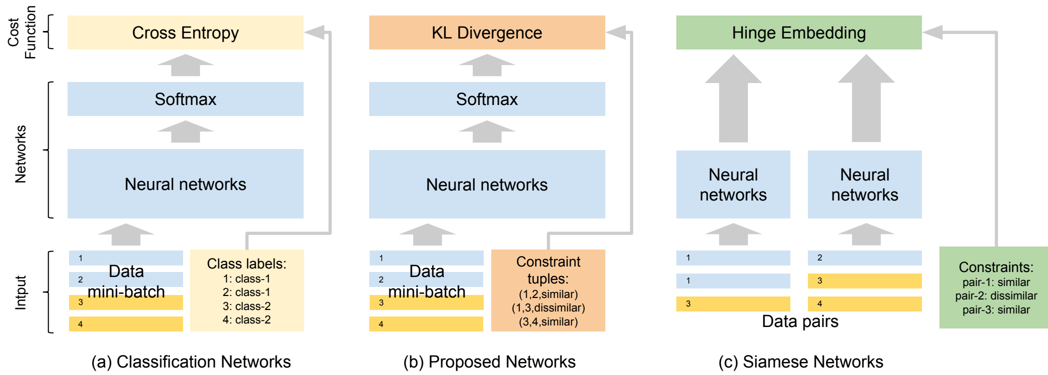

Figure 2: The comparison between (a) classification networks, (b) our proposed networks, and (c) Siamese networks. The parts that differ across architectures are shown with distinct colors. In (a) and (b), the numbers in the data represent the index of the input data in a mini-batch.

2.2 Efficient Architecture to Utilize Pairwise Constraints

Equation 2 is in a form 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 $n(n+1)/2 - n$ 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 $(i, j, relationship)$ 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.

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

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 networks 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 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.

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%

3.1 CLUSTERING VS CLASSIFICATION

We performed two sets of experiments to evaluate our approach. The first 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 comparable to classification, 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 optimal assignment. The results in Table 1 shows 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 didn't 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 were small, the proposed architecture and cost function proved effective.

3.2 CLUSTERING WITH PARTIAL CONSTRAINTS

The second set of experiments seek to demonstrate how the approach works with partial constraints. In this case, we also use a clustering metric to demonstrate how good the resulting clustering is. The constraints are uniformly sampled from the full set, i.e, $\#full\text{–}constraints = n(n+1)/2 - n$, 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 didn't address the fact that the amount of dissimilar pairs usually dominate the pairwise relationship (which is more realistic in many application domains), especially when 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.

We evaluate the resulting clusters with the purity measure and normalized mutual information (NMI). 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 is converged, i.e, feed the training data one more time to collect the outputs after the

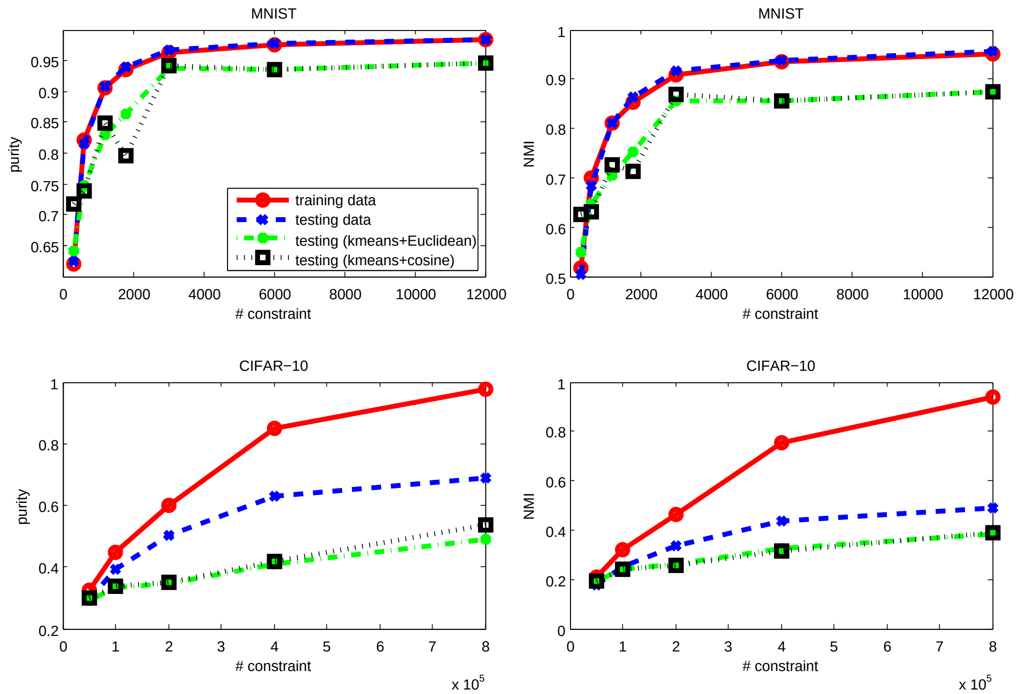

Figure 3: The results of clustering with partial pairwise constraints. The #constraint axis is the number of sampled pairwise relationship in the training data. The clustering and training is simutaneousy applied on the training data. The testing data is used to validate if the feature learned during clustering has generalizability. The k-means condition uses the features learned at the last hidden layer, which has a dimensionality of 500. The evaluation metric in the first column is purity, while the second column shows the NMI score.

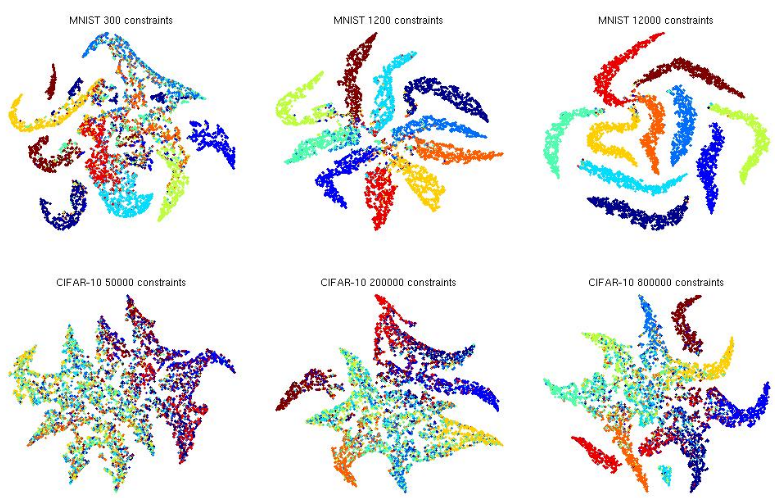

Figure 4: The visualization of clustering. The figure was created by using the outputs of softmax layer as the input for t-SNE (Van der Maaten & Hinton, 2008). Only testing data are shown. The networks used in the first row are trained with 300, 1200, and 12000 pairs of constraints in MNIST training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

training phase . As before, 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 is 2% constraints/sample, 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 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 on 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 no difference. Overall, Figure 3 shows that the clustering significantly benefits from the proposed framework.

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 $\sim$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.

4 Conclusion and Future Works

We introduce a novel criterion and efficient implementation to 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 trained with full pairwise constraints or semi-supervised with only partial constraints. We show 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 more efficiently implemented compared to a naive Siamese network implementation.

In future work, we plan to investigate the case where the number of clusters is not known. We also 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 hold 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

Bilenko, Mikhail, Basu, Sugato, and Mooney, Raymond 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.

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

Chopra, Sumit, Hadsell, Raia, and LeCun, Yann. 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.

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

Du, K-L. Clustering: A neural network approach. *Neural Networks*, 23(1):89–107, 2010.

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

Grossberg, Stephen. Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. *Biological cybernetics*, 23(3):121–134, 1976.

Hadsell, Raia, Chopra, Sumit, and LeCun, Yann. 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.

Han, Xufeng, Leung, Thomas, Jia, Yangqing, Sukthankar, Rahul, and Berg, Alexander C. 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.

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

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

Kohonen, Teuvo. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982. ISSN 0340-1200. doi: 10.1007/BF00337288.

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

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

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

Martinetz, T. M., Berkovich, S. G., and Schulten, K. J. 'neural-gas' network for vector quantization and its application to time-series prediction. *Trans. Neur. Netw.*, 4(4):558–569, July 1993. ISSN 1045-9227. doi: 10.1109/72.238311.

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

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

Schroff, Florian, Kalenichenko, Dmitry, and Philbin, James. 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.

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

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

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

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

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

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