Exploring centroids initialization within Deep Convolutional Embedded Clustering

Leonardo G. Nogueira, Adriane B. S. Serapião

[1]Universidade Estadual Paulista (UNESP)
Instituto de Geociências e Ciências Exatas (IGCE)
Departamento de Estatística, Matemática Aplicada e Computação (DEMAC)
Rio Claro – SP – Brasil

leo_henry@outlook.com, adriane.serapiao@unesp.br

Abstract. *Deep clustering uses a deep neural network to learn deep feature representation for performing clustering tasks. In this paper, we explored the Deep Clustering Embedded Clustering (DCEC) method, which employs a standart clustering method to get initial weight for the neural model training incorporated to other clustering methods. The original DCEC uses K-Means with Euclidean distance for the clusters center initialization step. We have applied K-Means with Mahalanobis distance instead of Euclidean distance. In order to improve the DCEC performance, we have included the standart K-Harmonic Means clustering algorithm as well, which tries overcome the dependency of the K-Means performance on the clusters center initialization. The Kernel based K-Harmonic Means was also introduced in this study to reduce the effect of outliers and noise. We evaluated the performance of these clustering approaches wthin DCEC over benchmark image datasets and the results were better than the baseline.*

1. Introduction

Clustering is a fundamental task in data science and unsupervised machine learning. Clustering divides data into a number k of subsets It consists of grouping of similar objects into a set known as cluster [Tan et al. 2016]. Objects in the same cluster are likely to be different when compared to those in other groups (clusters). It deals with finding a structure in a collection of unlabeled data. To divide a data into k clusters, some methods were proposed in the literatura, like K-Means and Gaussian Mixture Model. These methods have shown good results in small feature dimensions, although with higher dimensions the clustering becomes inefficient and unreliable. A solution to this problem is to lower the dimension of the feature space to a dimension where the traditional clustering methods works with reliable and efficient results.

Recent works on clustering has been focused on using deep neural networks (DNN) [Goodfellow et al. 2016] and this pairing is commonly referred to as deep clustering. In order to surmount the limitation of the classical clustering methods over large dimensional spaces, deep-learning-based clustering methods initially train a feature extractor and take, in a separate step, a clustering algorithm to the lower features dimension in the embedded space.

Deep clustering algorithms have three main components [Guo et al. 2017b]: deep neural network, network loss, and clustering loss. DNNs are applied to learn low dimensional non-linear data representations from the dataset. Most used architectures are

based on autoencoders, generative models and convolutional networks. The network loss refers to the reconstruction loss of the DNN and it measures how different the reconstructed data are from the original data. Clustering loss guides the embedded features to be prone to forming clusters and it can be divided in algorithms for cluster assignment (to provide cluster assignments to the data points directly) and cluster regularization (to enforce the network to preserve suitable discriminant information from the data in the representations). The present paper focus on cluster assignment in deep clustering with autoencoders.

Different clustering methods may be incorporated to the DNN to implement the clustering loss, to be run on top the learnt data representations.

K-Means [Lloyd 1982] is probably the most known clustering method. It uses random clusters centers at initialization and generally Euclidean distance to label the data and update de centroids. This random initialization affects directly on the clustering performance. All data has the same weight which makes the algorithm attribute more data to dense centers, converging to a suboptimal local minimum.

The Euclidean distance creates a spherical cluster shape that may not be the optimal shape to represent a group of data. It increases the algorithm's sensitivity to noise and outliers. Changing the distance metric others shapes of cluster may represent better the group of data, like Mahalanobis distance, that forms an elliptical shape of cluster when the covariance matrix is not the identity matri (and is equal to Euclidean when the covariance matrix is the identity).

To solve the initialization problem K-Harmonic Means [Zhang et al. 1999] was proposed. The algorithm uses dynamic weights for each data that allows the same point to belong simultaneously to different clusters, altough it uses Euclidean distance as well and imposes the assumption of hyper-spherical clusters for the data. If the separation boundaries between clusters are nonlinear, the K-Harmonic Means fails to identify these clusters correctly. However, the algorithm is essentially insensitive to the initialization of the centers.

An alternative approach to raise the robustness to the factors presented by K-Harmonic Means is to map the data into a highdimensional nonlinear feature space and accomplish the clustering within this space. Kernel based K-Harmonic Means [Li et al. 2007] was proposed, incorporating spatial information from images. This approach is imune to initialization and less prone to oversegmentation, outliers and noise,

The purpose of the present paper is to evaluate the classical clustering algorithms decribed above, taking part of the clustering loss of deep convolutional embeddes clustering [Guo et al. 2017b], applied to the image clustering task.

The paper is so organized. Section II describes some related work. Section III presents the details of DCEC algorithm. Section IV describes the proposed initialization in embedded space. Section V evaluates the different proposed initializations. Section VI concludes and suggest future works.

2. Related work

According to the different network architectures and the nature of loss functions used, deep clustering models can be roughly based on autoencoders, generative model and di-

rect cluster optimization. Our interest is on autoencoders based models. Most of them use a pre-training scheme in which the encoder and decoder network parameters are initialized with the reconstruction loss before clustering loss is introduced.

Deep Clustering Network (DCN) [Yang et al. 2016] carries out reconstruction and K-Means clustering simultaneously. The loss compasses penalties on both the reconstruction and the clustering losses.

Deep Embedded Clustering (DEC) [Xie et al. 2016] model learns feature representations with feed forward autoencoder and cluster assignments using K-Means at the same time. DEC learns a mapping from the data space to a lower-dimensional feature space in which it iteratively optimizes a clustering objective.

Improved Deep Embedded Clustering (IDEC) [Guo et al. 2017a] is based on DEC and the algorithm adds reconstruction loss to DEC's objective. IDEC did not incorporate convolutional layers.

Discriminately Boosted Clustering (DBC) [Li et al. 2017] is built on DEC by using a unified clustering framework to learn image representations and cluster centers jointly based on a fully convolutional autoencoder instead of feed forward autoencoder and soft K-Means scores. It uses the same training scheme, reconstruction loss and cluster assignment hardening loss as DEC. DBC neglected the local structure preservation problem

Deep Convolutional Embedded Clustering (DCEC) algorithm [Guo et al. 2017b] presented an improvement of DEC by incorporating convolutional autoencoders structure to learn embedded features in an end-to-end way. A clustering oriented loss is directly built on embedded features to jointly perform feature refinement and cluster assignment. DCEC takes care of both convolutional networks and preservation of local structure of data generating distribution.

3. Deep Convolutional Embedded Clustering

The DCEC algorithm [Guo et al. 2017b] was proposed to improve the Deep Embedded Clustering (DEC)[Xie et al. 2016] algorithm by replacing the Stacked Autoencoder (SAE), which is initialized layer by layer with each layer trained as a denoising autoencoder to reconstruct the previous layer output, to the Convolutional Autoencoder (CAE). By adding the reconstruction loss (L_r) to the objective function, they propose that in DEC the feature space may be distorted by using only the clustering loss (L_c).

3.1. Convolutional AutoEncoders

An autoencoder is a neural network that is trained to attempt to copy its input to its output. A traditional autoencoder may be viewed as consisting of two parts, a encoder layer $f_W(x)$ and a decoder layer $g_U(x)$. It has a objective of minimizing the mean squared errors (MSE) between the input and the output over all samples:

$$\min_{W,U} \frac{1}{n} \sum_{i=1}^{n} \|g_U(f_W(x_i)) - x_i\|_2^2 \tag{1}$$

In fully connected autoencoder,

$$f_W(x) = \sigma(Wx) \equiv h$$

$$g_U(h) = \sigma(Uh) \tag{2}$$

where x and h are vectors, and σ is an activation function. After training, h is the embedded space that represents the input sample and can be used to try to reconstruct the original feature space by using layer-wise pretraining, performing the Stacked Auto Encoder (SAE). For using in images we can use a convolutional autoencoder, defined as:

$$f_W(x) = \sigma(W * x) \equiv h$$

$$g_U(h) = \sigma(U * h) \tag{3}$$

with x and h being matrices or tensors, and $*$ the convolutional operator. A Stacked Convolutional Auto Encoder (SCAE) is implemented Using this structure and the layer-wise pretraining.

In the Convolutional AutoEncoder (CAE) proposed in DCEC[Guo et al. 2017b], this layer-wise pretraining is removed. Its structure is composed by stacked convolutional layers to extract hierarchical features and flat into a vector, followed by a fully connected layer with only 10 units called embedded layer, transforming a 2D image into a 10 units feature space. To train it in a unsupervised way, a fully connected layer and convolutional transpose layers is used to transform the embedded feature back to the original image. The parameters of encoder $f_w(x) = h$ and decoder $g_{w'}(h) = x'$ are upgraded by minimizing:

$$L_r = \frac{1}{n}\sum_{i=1}^{n} \|g_{w'}(f_w(x)) - x\|_2^2 \tag{4}$$

with L_r being the Reconstruction Loss, x the original image and n the number of images.

The CAE idea is to have the lowest dimension as possible in the embedded layer, if embedded layer is large enough, the model may copy the input to output, learning useless features.

3.2. The Structure of DCEC

The DCEC model is composed by CAE and with a clustering layer connected in the embedded layer of CAE, as illustrated by The Figure 1. The clustering layer maps each embedded point z_i of input image x_i into a soft label. Then the clustering loss L_c is defined as Kullback-Leibler divergence (KL divergence) between the distribution of soft labels and the predefined target distribution [Guo et al. 2017b]. CAE is used to learn embedded features and the clustering loss guides the embedded features to be prone to forming clusters. The DCEC objective is to minimize:

$$L = L_r + \gamma L_c \tag{5}$$

where $\gamma > 0$ is the degree of distortion in the embedded space. When $\gamma = 1$ and $L_r \equiv 0$ the objective is the same as in DEC.

3.3. Clustering Layer and Clustering Loss

The clustering layer [Guo et al. 2017b] is composed by two parts, a soft assignment and a KL divergence minimization (l_c minimization). In the soft assignment the Student's

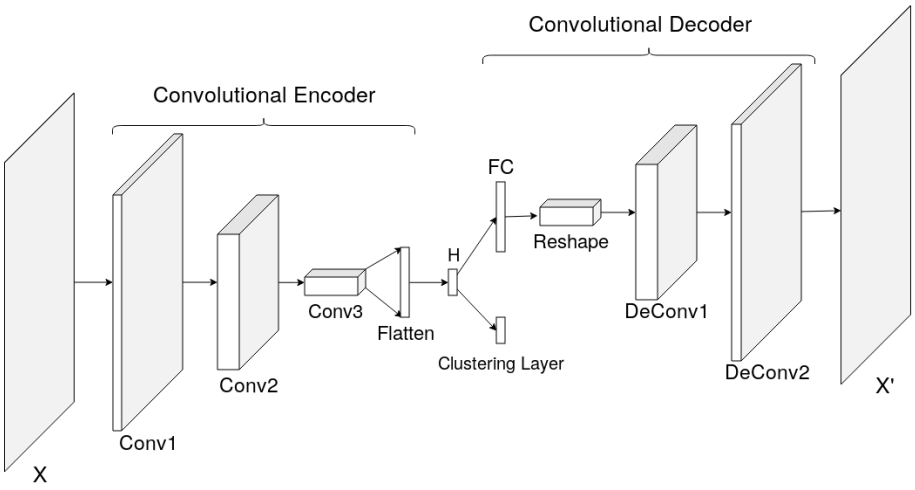

Figura 1. The structure of the Deep Convolutional Embedded Clustering, composed by CAE and a clustering layer connected to the embedded layer.

t-distribution is used to measure the similarity between z_i and μ_i, with z_i the embedded point and μ_i a centroid:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum_j (1 + \|z_i - \mu_j\|^2)^{-1}} \tag{6}$$

with q_{ij} being the probability of z_i being a point in the cluster with centroid μ_j. In DEC and DCEC the centroids initialization is made by using KM in the embedded layer and setting the centroids as weights in the cluster layer.

The clustering loss (L_c) is defined as:

$$L_c = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{7}$$

with p, the target distribution, defined as:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j (q_{ij}^2 / \sum_i q_{ij})} \tag{8}$$

4. Centroids Initialization in Embedded Space

To achieve an efficient clustering in the cluster layer, it needs a good initialization. In the orignal DCEC, the K-Means algorithm is used, employing the Euclidean distance for that purpose. We propose to substitute the initialization algorithm to alternatives that can results in better initialization, and consequently improve the model results. We propose to change the distance metric of K-Means and also to evaluate other approaches with the K-Harmonic Means and the Kernel based K-Harmonic Means algorithms.

4.1. K-Means

K-Means algorithm [Lloyd 1982] is one of the simplest unsupervised learning algorithms for clustering. It takes a bunch of unlabeled data points and tries to group them into "k" number of clusters, based on a distance metric. The general steps behind the K-Means clustering algorithm are:

1 Define the number of clusters (k).
2 Place randomly k central points (centroids) in different locations in the space of the problem.
3 For each data point, using a distance metric, assign it to the closest centroid.
4 Recalculate k new centroids as barycenters.
5 Repeat the assigning of data points to the new barycenter.
6 Repeat steps 4 and 5 until the barycenters do not move any more.

The algorithm is significantly sensitive to the initial randomly selected cluster centers. The K-Means aims to minimize the distance of each centroid to the cluster data. The standart metric used is the Euclidean distance, making the algorithm sensitive to noise and outliers. The Euclidean distance is calculed by:

$$D_E = \|z - \mu\| \tag{9}$$

with z a point in embedded space and μ a centroid in the cluster space.

K-Means and Euclidean distance work well only if the covariance structures of the clusters are nearly spherical and homogeneous in nature. To mitigate this shortfall in the K-Means algorithm, the Mahalanobis metric is applied to capture the variance structure of the clusters. It uses the covariance matrix V, between data and centroids in embedded space to calculate the distance. The Mahalanobis distance is given by:

$$D_M = \sqrt{(z - \mu)V^{-1}(z - \mu)^T} \tag{10}$$

4.2. K-Harmonic Means

K-Harmonic Means [Zhang et al. 1999] is a center-based clustering algorithm, which uses the harmonic average as components to its performance function. It overcomes the major drawback of K-Means, that is highly dependent on the initial identification of elements that represent the clusters. The algorithm has better results than K-Means in low dimension as shown by Harmerly and Elkan [Hamerly and Elkan 2002].

K-Harmonic Means algorithm goals to find the optimal clustering for the data set X, by minimizing the harmonic mean of the distance from each point to all the centers, $d_{ij} = \|z_i - \mu_j\|$.

$$Q(C) = \sum_{i=1}^{n} \frac{k}{\sum_{j=1}^{k} 1/(d_{ij})} \tag{11}$$

where k is the number of clusters, n is the number of data, $p \geq 2$ is a input parameter and C is the clusters.

The cluster update can be calculed using the membership u_{ij} and the weight w_i functions. The membership function calculates the probability of a point belong to a cluster. This means that a point influences in the orders centroids instead of only one, like K-Means.

$$u_{ij} = \frac{1/d_{ij}^2}{\sum_{j=1}^{k} 1/d_{ij}^2} \tag{12}$$

The weight function calculates the influence of the point to the centroids. It gives higher weight to points that are far from each center and lower to the near data. This aims

to spread the clusters to a better representation of data.

$$w_i = \frac{\sum_{j=1}^{k} 1/d_{ij}^2}{(\sum_{j=1}^{k} 1/d_{ij})^2} \tag{13}$$

The centroids update is given by:

$$C_j = \frac{\sum_{i=1}^{n} u_{ij} w_i x_i}{\sum_{i=1}^{n} u_{ij} w_i} \tag{14}$$

The K-Harmonic Means algorithm can be performed by initializing with random centroids and following the steps:

1 Update the centroids using (14).
2 Calculate the $Q(C)$ (11).
3 Repeat steps 1 and 2 until $Q(C)$ does not change.
4 Label the data by using argmax(u_{ij}).

This implementation has a discontinuity problem when $z_i = C_j$ leading to $d_{ij} \rightarrow 0$, to avoid this problem we use d_{ij} = max(d_{ij}, ϵ) with ϵ a small positive value.

4.3. Kernel based K-Harmonic Means

The K-Harmonic Means can surpass the initialization problem, but also use Euclidean distance. To create a non-euclidean distance measures, Kernel based K-Harmonic Means [Li et al. 2007] was proposed using Gaussian Radial Basis Function (RBF) Kernel. The Gaussian RBF Kernel is given by:

$$K(x, y) = \exp\left(\frac{-\|x - y\|^2}{\sigma^2}\right) \tag{15}$$

where σ is a parameter for non linear hyperplanes and controls (how far) the influence of new features on the decision boundary, with low values meaning ′far′ and high values meaning ′close′. In other words, it controls the shape of the kernel and the suppression to outliers.

The objective function of Kerner based K-Harmonic Means using Gaussian RBF is given by:

$$Q(C) = \sum_{i=1}^{n} \frac{2k}{\sum_{j=1}^{k} (1/(1 - K(z_i, c_j))} \tag{16}$$

with k is the number of centers.

The membership function u_{ij} and weight w_i are defined by:

$$u_{ij} = \frac{K(z_i, c_j)}{(1 - K(z_i, c_j))^2 \sum_{j=1}^{k} \frac{K(z_i, c_j)}{(1 - K(z_i, c_j))^2}} \tag{17}$$

$$w_i = \frac{\sum_{j=1}^{k} \frac{K(z_i, c_j)}{(1 - K(z_i, c_j))^2}}{\sum_{j=1}^{k} (\frac{1}{(1 - K(z_i, c_j))})^2} \tag{18}$$

The algorithm for the Kernel based K-Harmonic Means method is the same of K-Harmonic Means, replacing the equivalent equations.

5. Experiments

5.1. Datasets

The DCEC method was evaluated on the image datasets MNIST and USPS. For sake of comparision, we have used the same datasets as evaluated in [Guo et al. 2017b].

The MNIST handwritten digits dataset [Lecun et al. 1998] consists of a training set of 60000 grayscale images, representing 10 digits (0 to 9), and a test ser of 10000 images. The images are each of 28×28 pixel resolution, representing 784 features. We have usdes only the training set, which contains 600 observations per digit.

The USPS dataset is [Hull 1994] composed by normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations. Tje dataset contains 16×16 handwritten digit images, which are split into 7291 training images and 2007 test images. Onlys the trainins set was used in the experiments.

5.2. Experiment Setup

The platform of the experiments is a personal computer equipped with an Intel Core i7 (3^{th} generation), with 16 GB of DDR4 memory, and the operating system is Ubuntu Linux with 64-bit, 2.70 GHz CPU. The model was implemented using Keras and TensorFlow, a Python library that bolsters easy implementation of DL.

Comparing models. The purpose of this work is to compare different distance metrics used in the clustering method with DCEC. The distance metrics evaluated in the clustering layer were: Euclidean distance, Mahalanobis distance, K-Harmonic Means and RBF based K-Harmonic Means. In the original DCEC using Euclidean distance as metric, each time, the algorithm run 20 different random seed to get different set of initial centroids and it select the initial values which end up with the best clustering performance. Our implementation for DCEC with Mahalanobis distance, K-Harmonic Means and RBF based K-Harmonic Means run a single random values initialization.

Parameters settings. For the experiments with DCEC, we have used de original parameters used in [Guo et al. 2017b]. The encoder network structure is $\text{conv}^5_{32} \rightarrow \text{conv}^5_{64} \rightarrow \text{conv}^3_{128} \rightarrow \text{FC}_k$, where conv^n_f denotes a convolutional layer with f filters, kernel size of $n \times n$, number of cluster is k and stride length 2 as default. The decoder is a mirror of encoder. The CAE is pretrained end-to-end for 200 epochs using Adam with default parameters. The convergence threshold is set to δ= 0.1% and the update intervals T = 140. The implementation is based on Python and Keras. In RBF based K-Harmonic Means, the σ= was set to 1.

Evaluation measures. The four clustering models were evaluated by clustering accuracy (ACC), Normalized Mutual Information (NMI) and Adjusted Rank Index (ARI), which are common used in unsupervised learning task.

For each model we ran 10 training experiments and measured the indexes above. We then averaged the results of the 10 experiments. The standart deviation (s.d.) was computed for accuracy. We also constructed a 95% confidence interval (CI) around accuracy and investigated how accurate the estimates are. The values were expressed as p-values.

5.3. Results

In Tables 1 and 2 we compare the results of the clustering models for each dataset, respectively, MNIST and USPS.

Tabela 1. MNIST dataset evaluation

Distance metric	*ACC* $\pm$ *s.d.*	*CI*	*NMI*	*ARI*
K-Means (euclidean)	0.9012 ± 0.0382	0.0237	0.8590	0.8887
K-Means (Mahalanobis)	0.8805 ± 0.0535	0.0331	0.8514	0.9002
K-Harmonic Means	$\mathbf{0.9051} \pm 0.0703$	0.0459	**0.8765**	**0.9064**
RBF K-Harmonic Means	0.6752 ± 0.0606	0.0376	0.6394	0.7802

Tabela 2. USPS dataset evaluation

Distance metric	*ACC* $\pm$ *s.d.*	*CI*	*NMI*	*ARI*
K-Means (euclidean)	0.7938 ± 0.0095	0.0059	0.7523	**0.8320**
K-Means (Mahalanobis)	$\mathbf{0.7978} \pm 0.0730$	0.0452	**0.7582**	0.8243
K-Harmonic Means	0.7755 ± 0.0106	0.0066	0.7321	0.8144
RBF K-Harmonic Means	0.6600 ± 0.0665	0.0412	0.5921	0.7295

For MNIST dataset, DCEC with K-Harmonic Means outperformed all approaches in all performance measures. For USPS dataset, DCEC with Mahalanobis distance overcame both the original DCEC with Euclidean distance and the other approaches based on spherical shape in terms of accuracy, NMI and ARI. For all experiments, p-value related to CI is less than the significance level (0.05), which indicates that our results are statistically significant.

It is important to note that the centroids initialization of the winning aproaches was performed only once, obtaining better results than the original DCEC that worked with the best of 20 initializations.

By incorporating spatial information and non-linear transformation, RBF Kernel K-Harmonic Means within DCEC degraded clustering performance, contrary to expectations.

6. Conclusions

In this paper we have compared different traditional clustering algorithms as centroids initialization methods for deep convolutional embedded clustering, tailored for image clustering in large datasets. We have shown that our results (Mahalanobos distance and K-Harmonic Means) are compatible to previously proposed original DCEC network wuth Euclidean distance. However, DCEC with RBF based K-Harmonic Means was worse than all the approaches evaluated.

There are several interesting ways for improvements. A possible way is trying bigger and deeper architectures for convolutional networks. Another promising way is to try incorporating Swarm Clustering Algorithms at the clustering layer connected to embedded layer of autoencoder in DCEC, replacing the K-Means algorithm, in order to optimize the feature learning and clustering.

Referências

Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA.

Guo, X., Gao, L., Liu, X., and Yin, J. (2017a). Improved deep embedded clustering with local structure preservation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 1753–1759. AAAI Press.

Guo, X., Liu, X., Zhu, E., and Yin, J. (2017b). Deep clustering with convolutional autoencoders. In *ICONIP*.

Hamerly, G. and Elkan, C. (2002). Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 600–607, New York, NY, USA. ACM.

Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554.

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

Li, F., Qiao, H., and Zhang, B. (2017). Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognition*, 83:161–173.

Li, Q., Mitianoudis, N., and Stathaki, T. (2007). Spatial kernel k-harmonic means clustering for multi-spectral image segmentation. *IET Image Processing*, 1(2):156–167.

Lloyd, S. P. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137.

Tan, P.-N., Steinbach, M., and Kumar, V. (2016). *Introduction to Data Mining, (Second Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Xie, J., Girshick, R., and Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 478–487, New York, New York, USA. PMLR.

Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. (2016). Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*.

Zhang, B., Hsu, M., and Dayal, U. (1999). K-harmonic means: A data clustering algorithm. Technical report, Technical Report HPL1999-124, Hewlett-Packard Labs.