End-to-end Differentiable Clustering with Associative Memories

Bishwajit Saha [1] **Dmitry Krotov** [2] **Mohammed J. Zaki** [1] **Parikshit Ram** [2 3]

Abstract

Clustering is a widely used unsupervised learning technique involving an intensive discrete optimization problem. Associative Memory models or AMs are differentiable neural networks defining a recursive dynamical system, which have been integrated with various deep learning architectures. We uncover a novel connection between the AM dynamics and the inherent discrete assignment necessary in clustering to propose *a novel unconstrained continuous relaxation of the discrete clustering problem*, enabling end-to-end differentiable **cl**ustering with **AM**, dubbed **`ClAM`**. Leveraging the pattern completion ability of AMs, we further develop a novel self-supervised clustering loss. Our evaluations on varied datasets demonstrate that **`ClAM`** benefits from the self-supervision, and significantly improves upon both the traditional Lloyd's k-means algorithm, and more recent continuous clustering relaxations (by upto 60% in terms of the Silhouette Coefficient).

1. Introduction

Clustering is a fundamental unsupervised technique used to analyze and organize large amounts of data based on their similarities. Various formulations and algorithms have been studied to identify an efficient clustering of the data. Among them k-means (MacQueen, 1967), Spectral Clustering (Donath & Hoffman, 1973), Hierarchical Clustering (Johnson, 1967), Density-based Clustering (Ester et al., 1996), and Expectation Maximization (Dempster et al., 1977) have been widely used. However, these formulations are computationally expensive involving an intensive combinatorial task due to their discrete nature and lack of differentiability: for example, exact k-means is NP-hard (Dasgupta, 2008), though approximations can be efficient. To address this limitation, recent approaches have focused on deep clustering, which combines representation learning and clustering while utilizing continuous relaxation of the discrete assignment problem (Ren et al., 2022; Zhou et al., 2022). However, This relaxation replaces the discrete assignment with *partial cluster assignments*, which violates a fundamental premise of clustering – each point belongs to only one cluster. Additionally, to the best of our knowledge, there is currently no differentiable clustering scheme that can seamlessly leverage the stochastic gradient descent or SGD (Nemirovski et al., 2009) based optimization frameworks (Duchi et al., 2011; Kingma & Ba, 2014) for unconstrained optimization while preserving the discrete nature of clustering. We believe that the problem of clustering can benefit from utilizing SGD based solutions and to that end, we look for ideas in a very distant field of associative memories.

Recently, traditional **associative memory** or AM models (Hopfield, 1982; 1984) have been reformulated to significantly increase their memory storage capacity and integrated with modern deep learning techniques (Krotov & Hopfield, 2016; Ramsauer et al., 2020; Krotov & Hopfield, 2021; Krotov, 2021; 2023). These novel models, called Dense Associative Memories, are fully differentiable systems capable of storing a large number of multi-dimensional vectors, called patterns or "memories", in their synaptic weights. We believe that this ability to learn synaptic weights of the AM in an end-to-end differentiable manner together with the discrete assignment (association) of each data point to exactly one memory makes AM uniquely suited for the task of differentiable clustering. Specifically, we make the following contributions:

- We develop a flexible mathematical framework for **cl**ustering with **AM** or **`ClAM`**, which is a *novel continuous unconstrained relaxation* of the discrete optimization problem of clustering that allows for clustering in an **end-to-end differentiable manner** while maintaining the discrete cluster assignment throughout the training, with linear time cluster assignment.
- We leverage the pattern completion capabilities of AMs to develop a **differentiable self-supervised loss** that improves the clustering quality.
- We empirically demonstrate that **`ClAM`** is able to consistently improve upon k-means by upto 60%, while being competitive to spectral and agglomerative clustering, and producing insightful interpretations.

[1]CS Department, Rensselaer Polytechnic Institute, Troy, NY, USA [2]MIT-IBM Watson AI Lab, IBM Research, Cambridge, MA, USA [3]IBM Research, Yorktown Heights, NY, USA. Correspondence to: Bishwajit Saha <sahab@rpi.edu>.

Published at the Differentiable Almost Everything Workshop of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. July 2023.

2. Associative Memories for Clustering

In this section, we present (i) the mathematical framework for **ClAM**, (ii) motivate its suitability for clustering, and (iii) present a way of learning the memories for clustering.

2.1. Associative memories: Mathematical Framework

In a d-dimensional Euclidean space, consider M memories $\boldsymbol{\rho}_\mu \in \mathbb{R}^d, \mu \in [M] \triangleq \{1, \dots, M\}$ (we will discuss later how the memories are learned). The critical aspects of this mathematical framework are the **energy function** and the **attractor dynamics** (Krotov & Hopfield, 2021; Millidge et al., 2022). A suitable energy for clustering should be a continuous function of a point (or a particle) $\boldsymbol{v} \in \mathbb{R}^d$. Additionally, the energy should have M local minima, corresponding to each memory. Finally, as a particle progressively approaches a memory, its energy should be primarily determined by that single memory, while the contribution of the remaining $M-1$ memories should be small. An energy function satisfying these requirements is given by

$$E(\boldsymbol{v}) = -\frac{1}{\beta} \log \left(\sum\nolimits_{\mu \in [M]} \exp(-\beta \|\boldsymbol{\rho}_\mu - \boldsymbol{v}\|^2) \right) \quad (1)$$

with a scalar $\beta > 0$ interpreted as an inverse "temperature". As β grows, the $\exp(\cdot)$ in equation 1 ensures that only the leading term remains significant, while the remaining $M-1$ terms are suppressed. The energy function, in this limit, is represented by a parabola centered on the nearest memory, dividing the space into M basins of attraction. The attractor dynamics control how $\boldsymbol{v}$ moves in the space over time, via $d\boldsymbol{v}/dt$, while ensuring that energy decreases. That is $dE(\boldsymbol{v})/dt < 0$, which ensures that a particle convergences to a local minimum. The particle dynamics is described by gradient descent on the energy landscape:

$$\tau \frac{d\boldsymbol{v}}{dt} = -\frac{1}{2} \nabla_{\boldsymbol{v}} E = \sum_{\mu \in [M]} (\boldsymbol{\rho}_\mu - \boldsymbol{v})\, \boldsymbol{\sigma}(-\beta \|\boldsymbol{\rho}_\mu - \boldsymbol{v}\|^2) \quad (2)$$

where $\tau > 0$ is a characteristic **time constant**, describing how quickly the particle moves on the energy landscape, and $\boldsymbol{\sigma}(\cdot)$ is the softmax function over the scaled distances to the memories. This is guaranteed to reduce the energy

$$\frac{dE(\boldsymbol{v})}{dt} \overset{\text{(a)}}{=} \nabla_{\boldsymbol{v}} E(\boldsymbol{v}) \cdot \frac{d\boldsymbol{v}}{dt} \overset{\text{(b)}}{=} -2\tau \left\| \frac{d\boldsymbol{v}}{dt} \right\|^2 \overset{\text{(c)}}{\leq} 0 \quad (3)$$

where (a) is the chain rule, (b) follows from equation 2, and the equality in (c) implying local stationarity. A valid update $\boldsymbol{\delta}^{t+1}$ for the point $\boldsymbol{v}$ from state $\boldsymbol{v}^t$ to $\boldsymbol{v}^{t+1} = \boldsymbol{v}^t + \boldsymbol{\delta}^{t+1}$ at a discrete time-step $t+1$ is via finite differences:

$$\boldsymbol{\delta}^{t+1} = \frac{dt}{\tau} \sum\nolimits_{\mu \in [M]} (\boldsymbol{\rho}_\mu - \boldsymbol{v}^t)\, \boldsymbol{\sigma}(-\beta \|\boldsymbol{\rho}_\mu - \boldsymbol{v}^t\|^2) \quad (4)$$

Given a dataset of points S, for each point $\boldsymbol{v} \in S$ one can corrupt it with some noise to produce a distorted point $\tilde{\boldsymbol{v}}$. This serves as an initial state of the AM network $\boldsymbol{v}^0 \leftarrow \tilde{\boldsymbol{v}}$. The AM dynamics is defined by the learnable weights $\boldsymbol{\rho}_\mu, \mu = 1, \dots, M$, which also correspond to the fixed points of the dynamics (memories). The network evolves in time for T recursions according to equation 4, where T is chosen to ensure sufficient convergence to a fixed point. The final state $\boldsymbol{v}^T$ is compared with the uncorrupted $\boldsymbol{v}$ to define the loss function

$$\mathcal{L} = \sum\nolimits_{\boldsymbol{v} \in S} \|\boldsymbol{v} - \boldsymbol{v}^T\|^2, \text{ where } \boldsymbol{v}^0 \leftarrow \tilde{\boldsymbol{v}} \quad (5)$$

which is minimized with backpropagation through time with respect to the AM parameters $\boldsymbol{\rho}_\mu$.

2.2. AM as a differentiable discrete $\arg\min$ solver

Consider the original k-means objective with a dataset $S \subset \mathbb{R}^d$, where we learn k prototypes $\boldsymbol{R} \triangleq \{\boldsymbol{\rho}_\mu, \mu \in [k]\}$ with $[k] \triangleq \{1, \dots, k\}$ by solving the following problem:

$$\min_{\boldsymbol{R}} \sum_{\boldsymbol{x} \in S} \|\boldsymbol{x} - \boldsymbol{\rho}_{\mu^\star_{\boldsymbol{x}}}\|^2, \text{s.t. } \mu^\star_{\boldsymbol{x}} = \arg\min_{\mu \in [k]} \|\boldsymbol{x} - \boldsymbol{\rho}_\mu\|^2 \quad (6)$$

The discrete selection of $\mu^\star_{\boldsymbol{x}}$ (for each $\boldsymbol{x}$) makes equation 6 a combinatorial optimization problem that cannot directly be solved via (stochastic) gradient descent. A common continuous relaxation of this problem is as follows:

$$\min_{\boldsymbol{R}} \sum\nolimits_{\boldsymbol{x} \in S} \sum\nolimits_{\mu \in [k]} w_\mu(\boldsymbol{x})\, \|\boldsymbol{x} - \boldsymbol{\rho}_\mu\|^2 \quad (7)$$

where (usually) $w_\mu(\boldsymbol{x}) \in [0, 1]$ and $\sum_{\mu \in [k]} w_\mu(\boldsymbol{x}) = 1\, \forall \boldsymbol{x} \in S$. Hence, these weights $\{w_\mu(\boldsymbol{x}), \mu \in [k]\}$ define a probability over the k prototypes, and are designed to put the most weight on the closest prototype $\boldsymbol{\rho}_{\mu^\star_{\boldsymbol{x}}}$, and as small a weight as possible on the remaining prototypes. Other weighting functions have been developed with similar properties (Xie et al., 2016), and essentially utilize a weighted sum of distances to the learned prototypes, resulting in something different in essence to the discrete assignment $\mu^\star_{\boldsymbol{x}}$ in the original problem (equation 6). Another subtle point is that this **soft weighted assignment** of any $\boldsymbol{x} \in S$ across all prototypes **at training time** does not match the **hard cluster assignment** to the nearest prototype **at inference**, introducing an incongruity between training and inference.

We propose a novel alternative continuous relaxation to the discrete k-means problem leveraging the AM dynamics (§2.1) that preserves the discrete assignment in the k-means objective. Given the prototypes (memories) $\boldsymbol{R} = \{\boldsymbol{\rho}_\mu, \mu \in [k]\}$, the dynamics (equation 2) and the updates (equation 4) ensure that any example (particle) $\boldsymbol{x} \in \mathbb{R}^d$ will converge *to exactly one of the prototypes* $\boldsymbol{\rho}_{\widehat{\mu}_{\boldsymbol{x}}}$ corresponding to a single basin of attraction. Furthermore, for appropriate β, $\widehat{\mu}_{\boldsymbol{x}}$ matches the discrete assignment $\mu^\star_{\boldsymbol{x}}$ in the k-means objective (equation 6). This implies that, for appropriately set β and T, $\boldsymbol{x}^T \approx \boldsymbol{\rho}_{\mu^\star_{\boldsymbol{x}}}$, allowing us to replace the desired per-example

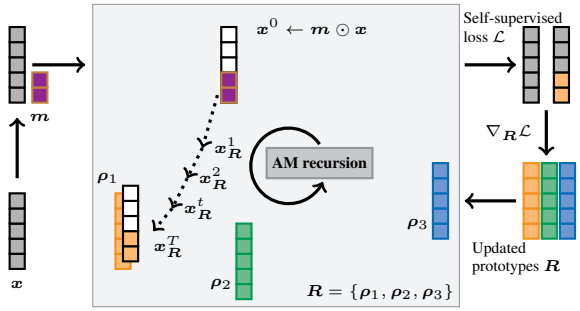


Figure 1: **ClAM Algorithm.** For $x \in S$, we first apply a mask (in purple) m to x to get the initial iterate x^0 for the AM recursion. With T recursions, we have a completed version x_R^T. The prototypes R are updated with the gradient $\nabla_R \mathcal{L}$ on the self-supervised loss $\mathcal{L}$ (equation 9).

loss $\|x - \rho_{\mu_x^\star}\|^2$ in the k-means objective (equation 6) with $\|x - x^T\|^2$, allowing us to rewrite k-means (equation 6) as the following continuous optimization problem:

$$\min_{R} \sum\nolimits_{x \in S} \left\| x - x_R^T \right\|^2, \text{ where } x^0 \leftarrow x \tag{8}$$

where x_R^T is obtained by applying update in equation 4 to any $x \in S$ through T recursion steps.

This is a significantly different relaxation of the k-means objective compared to the existing "weighted-sum" approaches. The tightness of this relaxation relies on the choices of β (inverse temperature) and T (the recursion depth) – we treat them as hyperparameters and select them in a data-dependent manner. See Appendix B.1 to understand the partitions induced by AMs.

2.3. ClAM: Clustering with AMs and self-supervision

The AM framework allows for a novel end-to-end differentiable unconstrained continuous relaxation (equation 8) of the discrete k-means problem leveraging the strong pattern completion abilities of AMs. We use standard masking from self-supervised learning – we apply a (random) mask $m \in \{0,1\}^d$ to a point $x \in S \subset \mathbb{R}^d$ and utilize the AM recursion to complete the partial pattern $x^0 = m \odot x$ and utilize the distortion between the ground-truth masked value and completed pattern as our loss as follows. Given a mask distribution $\mathcal{M}$ and denoting $\bar{m}$ as the complement of m:

$$\mathcal{L} = \sum\nolimits_{x \in S} \mathbb{E}_{m \sim \mathcal{M}} \|\bar{m} \odot (x - x_R^T)\|^2, \; x^0 \leftarrow m \odot x, \tag{9}$$

where x_R^T evolves from x^0 with T recursions of equation 4. Prototypes R are learned by minimizing the self-supervised pattern completion loss (equation 9) via SGD.

3. Empirical evaluation

We evaluate **ClAM** on 10 datasets of varying sizes – 7-16000 features, 101-60000 points. The number of clusters for each dataset are selected based on its number of underlying classes (the class information is not involved in clustering or hyperparameter selection). See details in Appendix A.1. First, we compare **ClAM** with established schemes, k-means (Lloyd, 1982), spectral & agglomerative clustering, and to DEC (Xie et al., 2016) and DCEC (Guo et al., 2017) (for 3 image sets since DCEC uses CAE). Then we ablate the effect of self-supervision in **ClAM**. Qualitatively, we visualize memory evolution through the learning.

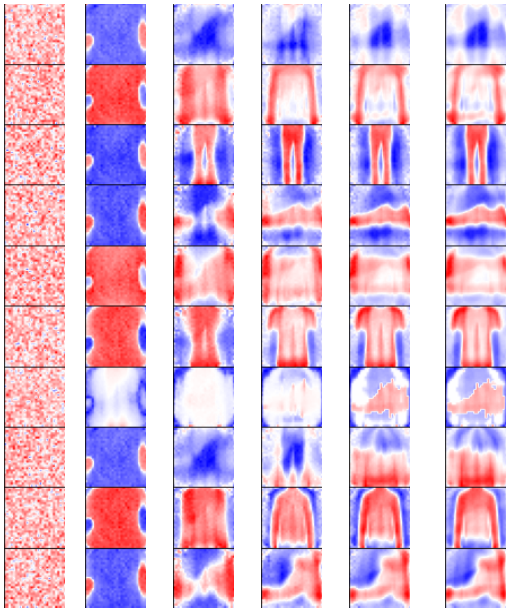
For implementation, we use Tensorflow (Abadi et al., 2016) for **ClAM** and `scikit-learn` (Pedregosa et al., 2011) for the clustering baselines and quality metrics. DCEC/DEC seed the optimization with the prototypes from the Lloyd's solution; we also consider a randomly seeded DEC^r. Further details are in Appendix A.3. We perform an elaborate hyperparameter search for all methods and utilize the configuration for each method corresponding to the best Silhouette Coefficient (SC) (Rousseeuw, 1987) on each dataset. The code is available at `https://github.com/bsaha205/clam`.

Q1: How does ClAM compare against baselines? We present the best SC obtained by all schemes in Table 1. **ClAM** consistently improves over k-means across all 10 datasets (up to 60%), and outperforms both versions of DEC, highlighting the advantage of the novel relaxation. Furthermore, Lloyd's seeding is critical in DEC – DEC^r does worse than base k-means (via Lloyd (1982)) in all cases – while ***ClAM*** *performs well with random seeding*. **ClAM** even improves upon DCEC, which uses rich image representations from a CAE. Overall, **ClAM** performs best on 5/10 datasets, showing significant improvements over even spectral and agglomerative clustering (these are not prototype-based, and hence can be more expressive). On the remaining datasets, both spectral and agglomerative clustering show improvements over **ClAM**. To understand this better, we also compare the clusters generated by the different methods to the ground-truth class structure in each of the datasets.The Normalized Mutual Information (NMI) & Adjusted Rand Index (ARI) (Hubert & Arabie, 1985) scores (Vinh et al., 2009) are shown in Table 3 from Appendix. The results indicate that the datasets on which spectral and agglomerative clustering have a significantly higher SC than k-means and **ClAM** (GCM, USPS, CTG, Segment), their corresponding NMI & ARI scores are significantly lower (see underlined entries in Table 3 from Appendix), indicating clusters that are misaligned with the ground-truth labels. Upon further investigation, we see that both spectral and agglomerative clustering end up with a single large cluster, and many really small (even singleton) ones, indicating that the clustering is overly influenced by geometric outliers.

Q2: How beneficial is self-supervision in ClAM? In **ClAM**, we use the pattern completion ability of AMs, and optimize

Table 1: **Silhoutte Coefficient (SC) obtained by ClAM, its variants and baselines (*higher is better*).** ▲(▼) show the performance gain (drop) by **ClAM**, given by $^{(a-b)}/_{b} \times 100\%$ where a is the SC for **ClAM**, and b is the SC for the baseline; positive (negative) values indicate performance gain (drop). The best performance for a dataset is in **boldface**. W/L denote Wins/Losses. Note ablations **ClAM** w/ (8) is **ClAM** without self-supervision.

Dataset	k-means	Spectral	Agglo	DCEC	DEC	DEC^r	ClAM	w/ (8)
Zoo	0.374$^{\blacktriangle 10\%}$	0.398$^{\blacktriangle 3\%}$	0.398$^{\blacktriangle 3\%}$	—	0.374$^{\blacktriangle 10\%}$	0.367$^{\blacktriangle 12\%}$	**0.412**	0.382
Ecoli	0.262$^{\blacktriangle 26\%}$	0.381$^{\blacktriangledown 13\%}$	**0.404**$^{\blacktriangledown 18\%}$	—	0.262$^{\blacktriangle 26\%}$	0.255$^{\blacktriangle 30\%}$	0.331	0.301
MovLib	0.252$^{\blacktriangle 3\%}$	0.247$^{\blacktriangle 5\%}$	0.247$^{\blacktriangle 5\%}$	—	0.258$^{\blacktriangle 1\%}$	0.184$^{\blacktriangle 41\%}$	**0.260**	0.248
Yale	0.117$^{\blacktriangle 10\%}$	0.120$^{\blacktriangle 8\%}$	0.114$^{\blacktriangle 13\%}$	0.089$^{\blacktriangle 45\%}$	0.123$^{\blacktriangle 5\%}$	0.088$^{\blacktriangle 47\%}$	**0.129**	0.121
USPS	0.135$^{\blacktriangle 1\%}$	0.228$^{\blacktriangledown 40\%}$	**0.231**$^{\blacktriangledown 41\%}$	0.119$^{\blacktriangle 14\%}$	0.135$^{\blacktriangle 1\%}$	0.108$^{\blacktriangle 26\%}$	0.136	0.127
Segment	0.357$^{\blacktriangle 35\%}$	**0.702**$^{\blacktriangledown 31\%}$	0.697$^{\blacktriangledown 31\%}$	—	0.357$^{\blacktriangle 35\%}$	0.287$^{\blacktriangle 68\%}$	0.483	0.357
FMNIST	0.126$^{\blacktriangle 10\%}$	0.075$^{\blacktriangle 84\%}$	0.096$^{\blacktriangle 44\%}$	0.121$^{\blacktriangle 14\%}$	0.129$^{\blacktriangle 7\%}$	0.070$^{\blacktriangle 97\%}$	**0.138**	0.131
GCM	0.118$^{\blacktriangle 62\%}$	0.205$^{\blacktriangledown 7\%}$	**0.288**$^{\blacktriangledown 34\%}$	—	0.115$^{\blacktriangle 66\%}$	0.054$^{\blacktriangle 254\%}$	0.191	0.142
MicePE	0.128$^{\blacktriangle 56\%}$	0.156$^{\blacktriangle 28\%}$	0.193$^{\blacktriangle 4\%}$	—	0.137$^{\blacktriangle 46\%}$	0.115$^{\blacktriangle 74\%}$	**0.201**	0.126
CTG	0.161$^{\blacktriangle 53\%}$	**0.449**$^{\blacktriangledown 45\%}$	0.387$^{\blacktriangledown 36\%}$	—	0.164$^{\blacktriangle 50\%}$	0.130$^{\blacktriangle 89\%}$	0.246	0.147
ClAM W/L	10/0	5/5	5/5	3/0	10/0	10/0	—	10/0



(a) n_0 (b) n_5 (c) n_{10} (d) n_{20} (e) n_{50} (f) n_{100}

Figure 2: **Evolution of ClAM prototypes.** We visualize the prototypes at the n^{th} training epoch for $n = 0, 5, 10, 20, 50, 100$ (with $T = 10$). The images pixels are colored as red for positive, white for zero and blue for negative values, with their intensity denoting magnitude.

for the self-supervised loss (equation 9). Here, we ablate the effect of this choice – we utilize a version of **ClAM** that does not use any masking and learns memories by minimizing the loss in equation 8. Table 1 (**ClAM** and **ClAM w/ (8)** columns) show their SC, highlighting the positive effect of self-supervision, with around 10% gain in all cases.

Q3: How to interpret ClAM? An interesting aspect of AMs is the prototype-based representation of memories (Krotov & Hopfield, 2016). We study this for **ClAM** with the Fashion-MNIST images. We use **ClAM** to partition the 60k images of fashion items into 10 clusters, and visualize the evolution of the 10 memories in figure 2. Each sub-figure in figure 2 corresponds to a particular epoch during the training, and shows all 10 memories stacked vertically. We can see that at epoch 0 (figure 2a) the memories are initialized with random values, but gradually reveal discernible patterns by epoch 10 (Figure 2c) and further sharpen by epoch 50 (figure 2e). By epoch 100 (figure 2f), the shapes have stabilized although some learning might still be happening. Some memories evolve into individual forms (such as rows 3–pants, 4–shoes, 8–bags, 9–long-sleeve shirts and 10–boots), while others evolve into mixtures of 2 forms (such as rows 1–pants and shoes, 6–pants and short-sleeve shirts, 7–shoes and long-sleeve shirts). The remaining evolve into less distinguishable forms though we can still visualize some shapes (like in rows 2–long-sleeve shirts and 5–shoes).

4. Limitations and Future Work

In this paper, we present a novel continuous relaxation of the discrete prototype-based clustering problem leveraging the AM dynamical system that allows us to solve the clustering problem with SGD based schemes. We show that this relaxation better matches the essence of the discrete clustering problem, and empirical results show that our **ClAM** approach significantly improves over standard prototype-based clustering schemes, and existing continuous relaxations. We note that **ClAM** is still a prototype-based clustering scheme, hence inherits the limitations of prototype-based clustering. Given the end-to-end differentiable nature of **ClAM**, we will extend it to clustering with kernel similarity functions and Mahanalobis distances, and to deep clustering where we also learn a latent space. We plan to explore new energy functions and update dynamics that enable spectral clustering. Finally, given **ClAM**'s flexibility, we want to automatically estimate the number of clusters on a per-dataset basis, much like Pelleg & Moore (2000) and Hamerly & Elkan (2003).

Acknowledgements

This work was supported by the Rensselaer-IBM AI Research Collaboration (http://airc.rpi.edu), a part of the IBM AI Horizons Network. The work was done during Bishwajit Saha's externship at MIT-IBM Watson AI Lab, IBM Research.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

Aurenhammer, F. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

Chakraborty, S., Paul, D., and Das, S. Automated clustering of high-dimensional data with a feature weighted mean shift algorithm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 6930–6938, 2021.

Dasgupta, S. *The hardness of k-means clustering*. Department of Computer Science and Engineering, University of California . . . , 2008.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Donath, W. E. and Hoffman, A. J. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, 1973. doi: 10.1147/rd.175.0420.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.

Guo, X., Liu, X., Zhu, E., and Yin, J. Deep clustering with convolutional autoencoders. In *International conference on neural information processing*, pp. 373–382. Springer, 2017.

Hamerly, G. and Elkan, C. Learning the k in k-means. *Advances in neural information processing systems*, 16, 2003.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10): 3088–3092, 1984.

Hubert, L. and Arabie, P. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Hull, J. J. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

Johnson, S. C. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krotov, D. Hierarchical associative memory. *arXiv preprint arXiv:2107.06446*, 2021.

Krotov, D. A new frontier for hopfield networks. *Nature Reviews Physics*, pp. 1–2, 2023.

Krotov, D. and Hopfield, J. J. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

Krotov, D. and Hopfield, J. J. Large associative memory problem in neurobiology and machine learning. In *International Conference on Learning Representations*, 2021.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. Feature selection: A data perspective. *ACM computing surveys (CSUR)*, 50(6):1–45, 2017.

Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

MacQueen, J. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pp. 281–297, 1967.

Millidge, B., Salvatori, T., Song, Y., Lukasiewicz, T., and Bogacz, R. Universal hopfield networks: A general framework for single-shot associative memory models. *arXiv preprint arXiv:2202.04557*, 2022.

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Pelleg, D. and Moore, A. W. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, volume 1, pp. 727–734, 2000.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K., et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

Ren, Y., Pu, J., Yang, Z., Xu, J., Li, G., Pu, X., Yu, P. S., and He, L. Deep clustering: A comprehensive survey. *arXiv preprint arXiv:2210.04142*, 2022.

Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Vinh, N. X., Epps, J., and Bailey, J. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual international conference on machine learning*, pp. 1073–1080, 2009.

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

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

Zhou, S., Xu, H., Zheng, Z., Chen, J., Bu, J., Wu, J., Wang, X., Zhu, W., Ester, M., et al. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *arXiv preprint arXiv:2206.07579*, 2022.

A. Experimental Details

A.1. Dataset details

To evaluate **ClAM**, we conducted our experiments on ten standard benchmark data sets. The datasets are taken from various sources such as Yale from ASU feature selection repository[1] (Li et al., 2017), USPS from Kaggle[2] (Hull, 1994), Fashion-MNIST from Zalando[3] (Xiao et al., 2017), GCM from Chakraborty et al. (2021) and the rest of the datasets from the UCI machine learning repository[4]. The statistics of datasets used in our experiment are given in Table 2.

Table 2: Descriptions of various benchmark datasets, used in our experiments.

Dataset	Short name	# Points	# Features	# Classes
Zoo	Zoo	101	16	7
Yale	Yale	165	1024	15
GCM	GCM	190	16063	14
Ecoli	Ecoli	336	7	8
Movement Libras	MovLib	360	90	15
Mice Protien Expression	MicePE	1080	77	8
USPS	USPS	2007	256	10
CTG	CTG	2126	21	10
Segment	Segment	2310	19	7
Fashion MNIST	FMNIST	60000	784	10

A.2. Metrics used

To evaluate the performance of **ClAM**, we use Silhouette Coefficient (SC) (Rousseeuw, 1987) as the unsupervised metric that is used to measure the quality of clustering. The score is between -1 and 1; a value of 1 indicates perfect clustering while a value of -1 indicates entirely incorrect clustering labels. A value of near 0 indicates that there exist overlapping clusters in the partition. To observe which existing clustering scheme is **ClAM** most similar to, we use Normalized Mutual Information (NMI) (Vinh et al., 2009) & Adjusted Rand Index (ARI) (Hubert & Arabie, 1985) between the obtained partition by **ClAM** and obtained partition by baselines. For NMI, a value of 1 indicates perfect clustering while a value of 0 indicates completely wrong class labels. ARI scores range between -1 and 1 and the interpretation is same as SC. We also use the NMI & ARI scores between the ground truth and the obtained partition from **ClAM** and the baselines to measure how they are aligned to true clustering labels.

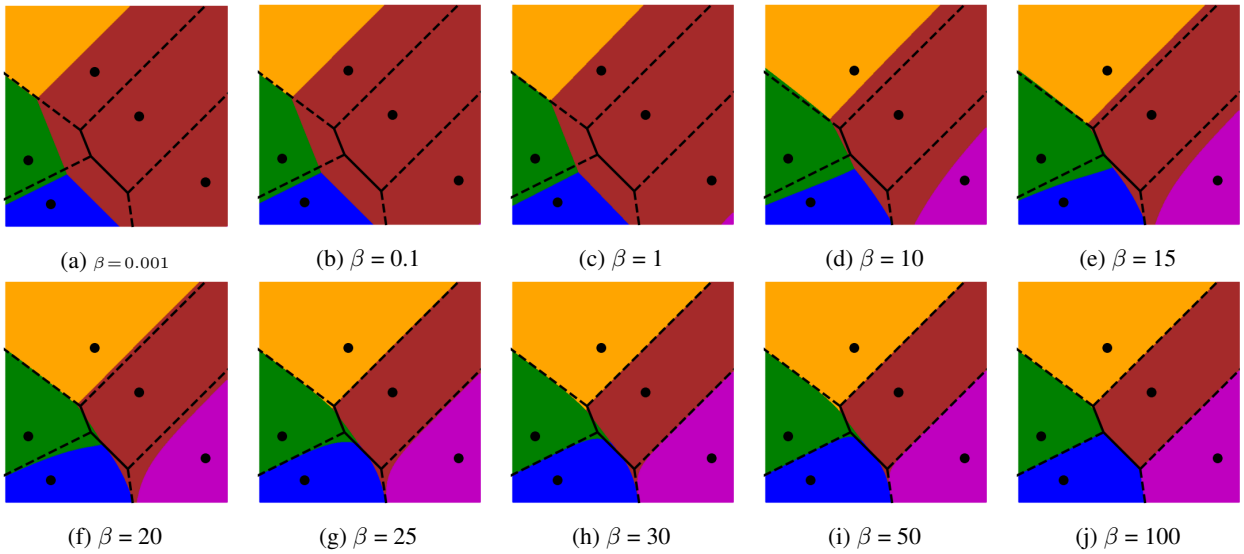
A.3. Implementation Details

We use Tensorflow (Abadi et al., 2016) numerical machine learning library to implement and evaluate our model. We train **ClAM** on a single node with 1 NVIDIA Geforce RTX 3090 (24GB RAM), and 8-core 3.5GHz Intel Core-i9 CPUs (32GB RAM). We train on the original data with masking where we utilize the distortion between the ground-truth masked value and completed pattern from **ClAM** as the loss function to find the best model. In the forward pass, in each step, the feature vector is updated in such a way so that gradually it moves toward one of the stored memories. In the backward path, the memories are learned to minimize the loss. Hyperparameters are tuned for each dataset to find the best result. For the baseline schemes of k-means, spectral, and agglomerative, we use the implementation from `scikit-learn` (Pedregosa et al., 2011) library and tune different hyperparameters to get the best results for each dataset. For DCEC (Guo et al., 2017), as it is based on convolutional autoencoders (CAE) and works with only image dataset, we evaluate on three image datasets to compare with **ClAM** (we leverage their Tensorflow implementation[5]). For the soft-clustering part of DEC (Xie et al., 2016) (where they utilize KL divergence loss between predicted and target probability distribution), besides their k-means-initialized cluster centers, we also employ random-initialized cluster centers (DEC^r) to study how randomization works in DEC soft clustering network.

[1] http://featureselection.asu.edu/
[2] https://www.kaggle.com/datasets/bistaumanga/usps-dataset
[3] https://github.com/zalandoresearch/fashion-mnist
[4] https://archive.ics.uci.edu/ml/index.php
[5] https://github.com/XifengGuo/DCEC

Table 3: **NMI and ARI score comparison among `ClAM` and the baseline clustering schemes in terms of ground truth (*higher is better*).** The best performance for each dataset is in **boldface**. The underlined entries for **Spec**tral and **Aggl**omerative mark low NMI but high SC in Table 1.

Dataset	Metric	k-means	Spectral	Agglomerative	DCEC	DEC	DEC^r	`ClAM`
Zoo	NMI	0.8330	0.8891	0.8429	N/A	0.8330	0.7992	**0.9429**
	ARI	0.7373	0.8664	0.9109	N/A	0.7373	0.6755	**0.9642**
Yale	NMI	0.6034	0.5744	**0.6723**	0.5428	0.5499	0.5068	0.6418
	ARI	0.3644	0.3226	**0.4567**	0.2817	0.2920	0.2398	0.4230
GCM	NMI	0.4385	0.1715	0.1882	N/A	0.4228	0.3923	**0.4476**
	ARI	0.1308	-0.0119	-0.0071	N/A	0.1188	0.0880	**0.2492**
Ecoli	NMI	0.6332	0.6606	**0.7111**	N/A	0.6332	0.5707	0.6633
	ARI	0.4997	0.6505	**0.7261**	N/A	0.4997	0.3893	0.7027
Movement Libras	NMI	0.6044	0.6118	0.6086	N/A	0.5959	0.3147	**0.6142**
	ARI	0.3260	0.3236	0.3144	N/A	0.3147	0.2223	**0.3351**
Mice Protien Exp	NMI	0.2373	0.0056	0.2596	N/A	0.2873	0.2951	**0.3108**
	ARI	0.1260	0.0019	0.1558	N/A	0.1756	0.1796	**0.1652**
USPS	NMI	0.5368	0.0777	0.0180	**0.6961**	0.5376	0.4538	0.5566
	ARI	0.4304	-0.0033	0.0002	**0.5907**	0.4306	0.3226	0.4537
CTG	NMI	0.3581	0.0391	0.0419	N/A	0.3507	**0.3587**	0.3154
	ARI	0.1780	0.0060	0.0078	N/A	0.1766	**0.1818**	0.1736
Segment	NMI	0.5846	0.0102	0.0085	N/A	0.5853	**0.6102**	0.5489
	ARI	0.4607	0.0005	0.0003	N/A	0.4612	**0.5038**	0.4331
FMNIST	NMI	0.5036	**0.6429**	0.0051	0.5948	0.5008	0.3339	0.5183
	ARI	0.3461	**0.4307**	0.0005	0.4113	0.3369	0.2279	0.3665



(a) $\beta = 0.001$ (b) $\beta = 0.1$ (c) $\beta = 1$ (d) $\beta = 10$ (e) $\beta = 15$

(f) $\beta = 20$ (g) $\beta = 25$ (h) $\beta = 30$ (i) $\beta = 50$ (j) $\beta = 100$

Figure 3: **Basins of attraction vs Voronoi partition for five clusters.** Partitions induced by AM basins of attraction with given memories (black dots) for different β are shown by the colored regions (T=10). Dashed lines show the Voronoi partition.

B. Additional Experimental Results

B.1. Understanding the partitions induced by AMs

An interpretation of the discrete assignment $\mu^\star_{\boldsymbol{x}}$ for each $\boldsymbol{x} \in S$ in the k-means objective (equation 6) is that the prototypes $\boldsymbol{R}$ induce a Voronoi partition (Aurenhammer, 1991) of the whole input domain, and the examples $\boldsymbol{x} \in S$ are assigned to the prototype whose partition they lie in. Voronoi partitions have piecewise linear boundaries, with all points in a partition having the same closest prototype. Given prototypes $\boldsymbol{R}$, the AM dynamics induces a partition of the domain into non-overlapping

basins of attraction. For an appropriately set β and T, the basins of attraction approximately match the Voronoi partition induced by the prototypes. However, when β is not "large enough", the cluster assignment happens in a more "collective" manner, involving all prototypes, leading to non-Voronoi partitions. We visualize the basins of attraction for different β in figure 3, contrasting them to the Voronoi partition. For a small β (figure 3a), the basins do not match the Voronoi partition. As β increases, the basins of attraction evolve to match the Voronoi partition (figure 3j). Furthermore, while the Voronoi partition boundaries are piecewise linear, the AM basins of attraction can have nonlinear boundaries (see boundaries in figure 3e). As mentioned earlier, we treat β and T as hyperparameters, and select them in a data-dependent fashion, so that the AM partition sufficiently matches the Voronoi one.