Improved Representation Learning Through Tensorized Autoencoders

Pascal Esser[*] Technical University of Munich

Satyaki Mukherjee[*] Technical University of Munich

Mahalakshmi Sabanayagam[*] Technical University of Munich

Debarghya Ghoshdastidar Technical University of Munich

Abstract

The central question in representation learning is what constitutes a good or meaningful representation. In this work we argue that if we consider data with inherent cluster structures, where clusters can be characterized through different means and covariances, those data structures should be represented in the embedding as well. While Autoencoders (AE) are widely used in practice for unsupervised representation learning, they do not fulfil the above condition on the embedding as they obtain a single representation of the data. To overcome this we propose a meta-algorithm that can be used to extend an arbitrary AE architecture to a tensorized version (TAE) that allows for learning cluster-specific embeddings while simultaneously learning the cluster assignment. For the linear setting we prove that TAE can recover the principle components of the different clusters in contrast to principle component of the entire data recovered by a standard AE. We validate this on planted models and for general, non-linear and convolutional AEs we empirically illustrate that tensorizing the AE is beneficial in clustering and de-noising tasks.

1 INTRODUCTION AND MOTIVATION

With the increasing use of very high dimensional data an important task is to find a good lower dimensional representation either to reduce noise in the data or to overcome the curse of dimensionality. An obvious question is therefore, what is a *good* representation and how can we guarantee that a given algorithm obtains it? Conceptually a latent representation should preserve certain structures of the data, while removing noise dimensions. Naturally there are different perspectives on what could constitute an important structure of the given data, such as an underlying topology or clustering structures. In this work we consider the latter from a statistical perspective and more specifically from the perspective of its covariance structure and latent representation. Suppose our dataset consists of k clusters that each have some inherent structure. Then we would want those separate structures to be again represented in the latent space. One of the most common approaches in deep learning for obtaining latent representations are Autoencoders (AE) (Kramer, 1991). Formally we write a general AE as the following optimization problem

$$\min_{\Phi,\Psi} \|\boldsymbol{X} - f_\Phi\left(g_\Psi\left(\boldsymbol{X}\right)\right)\| - \lambda * \text{penalty}, \tag{1}$$

where $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ is the *centred* data input matrix consisting of n samples of dimension d and $\widehat{\boldsymbol{X}} = f_\Phi\left(g_\Psi\left(\boldsymbol{X}\right)\right)$ the reconstruction. Let $g_\Psi(\cdot)$ be the *encoder function*, parameterized by Ψ mapping from the data dimension d onto the latent dimension h and $f_\Phi(\cdot)$ the *decoder function* parameterized by Φ mapping back to d. λ determines the strength of the penalty, that is chosen depending on the task or desired properties of the parameters.

It has been shown that in the linear setting the encoder recovers the principal components of the full dataset (Kunin et al., 2019). However this one representation does not necessarily capture the underlying cluster structure well as illustrated in Figure 1. A classical example of this is the so called *simpson's paradox* (Simpson, 1951), a known phenomenon in statistics where the trend in clusters does not align with the trend that appears in the full dataset, often observed in social-science and medical-science statistics (Wagner, 1982; Holt, 2016). Figure 1 shows that one representation for all the data points (shown in black) does not capture the structures of individual clusters well (shown by the colored arrows). To be able to model such structures we therefore introduce a modified AE architecture we term

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

[*]Equal contribution.

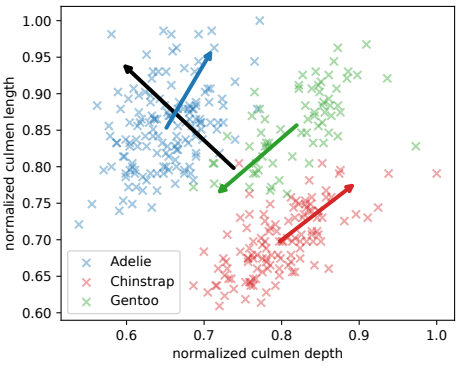

Figure 1: Illustration simpson's paradox in the 'penguin dataset' (Gorman et al., 2014). The three clusters and their first principal component are plotted in red, blue and green respectively and the principal direction for the full dataset in black.

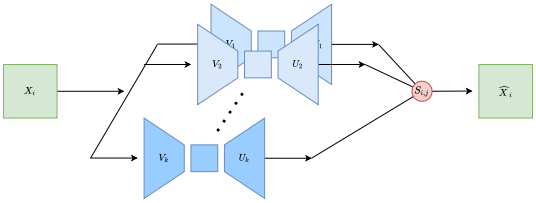

Figure 2: Illustration of the tensorized AE. An input $\boldsymbol{X}_i$, is passed through k AEs, with k-means penalty latent space that reconstruct $\boldsymbol{X}_i$. $\boldsymbol{S}_{i,j}$ weight the outputs of the AEs by class assignment.

Tensorized Autoencoders (TAE) that, in the linear setting, provably recovers the principal directions of *each cluster* while jointly learning the cluster assignment. This new AE architecture considers a single AE for each cluster allowing us to learn distinct cluster representations and is illustrated in Figure 2. Important to note that while this increases the number of parameters, the representation still remains the same dimension as before. In particular this still experimentally performs better than a single autoencoder with $d \times k$ encoding dimension. Formally we change (1) to the following

$$\min_{\{\Phi_j,\Psi_j\}_{j=1}^k,\boldsymbol{S}} \sum_{i=1}^{n}\sum_{j=1}^{k} \boldsymbol{S}_{j,i}\left[\left\|(\boldsymbol{X}_i - \boldsymbol{C}_j) - f_{\Phi_j}\left(g_{\Psi_j}\left(\boldsymbol{X}_i - \boldsymbol{C}_j\right)\right)\right\|^2 - \lambda * \text{penalty}_j\right], \tag{2}$$

where $\boldsymbol{S}$ is a $k \times n$ matrix, such that $\boldsymbol{S}_{j,i}$ is the probability that data point i belongs to class j, $g_{\Psi_j}(\cdot)$ and $f_{\Phi_j}(\cdot)$ are the *encoder* and the *decoder* functions respectively specific to points in class j, and C_j is a parameter centering the data passed to each autoencoder specific to class j. Specifically a k-means (Macqueen, 1967) penalty is considered to enforce a cluster friendly structure in the latent space (similar to the one proposed in Yang et al. (2017)).

To further illustrate the importance of the new formulation (2), that provides cluster specific representations, we look at two important representations learning downstream tasks.

Clustering. The main goal of clustering is to group similar objects into the same class in an unsupervised setting. While this problem has been extensively studied in traditional machine learning the time complexity significantly increases with high dimensional data and therefore existing works focus on projecting data into low-dimensional spaces and then cluster the embedded representations (Roth and Lange, 2003; Tian et al., 2014; Wang et al., 2016). Several methods have been developed that use deep unsupervised models to learn representations with a clustering focus that simultaneously learns feature representations and cluster assignments using deep neural networks (Xie et al., 2016; Dizaji et al., 2017; Wang et al., 2016; Xie and Xing, 2015; Wang et al., 2015). However all of these algorithms learn a single representation for the full dataset. Assuming we use an AE for learning the representation we formally consider (1) and perform the clustering on $g_\Psi(\boldsymbol{X})$ instead of $\boldsymbol{X}$. Now the question is, is one representation of the data sufficient? We investigate this question by considering different planted datasets as shown in Figure 3, where we compare the clustering obtained from k-means ++ (Arthur and Vassilvitskii, 2007) on the original features, the embedding obtained by a standard AE and the proposed TAE. In addition we also consider a simple variational autoencoder (VAE) (Kingma and Welling, 2013) and Deep Clustering (DC) (Yang et al., 2017) and observe that in this setting they perform very similar to standard AE. Therefore for the later experiments we only focus on the comparison to k-means ++ and standard AE. The advantage of TAE is that it captures the *directions* of the clusters whereas simple AE mis-classifies some points in clusters that are close in euclidean space. For those datasets the distance function is inherently linked to the shape of the clusters. We extend this analysis to real world data and more complex AE architectures in Section 3.

De-noising. Consider an image corrupted by noise, the task is to remove the noise from the image. In this setting a good representation is one that only returns the true data structure and removes the noise. Here we are not directly interested in the embedding but only in the obtained reconstruction of the decoder. However we again conjecture that having separate embeddings for each cluster is beneficial as it allows to learn more cluster specific representations.

We consider a De-Noising AE (DAE), formally defined (Buades et al., 2005; Cho, 2013) by considering $\boldsymbol{X} + \varepsilon$ as input in Eq 1 where ε is an additive noise term and the goal is to remove the noise from the data. An additional advantage in the denosing setting is that the cluster structure obtained

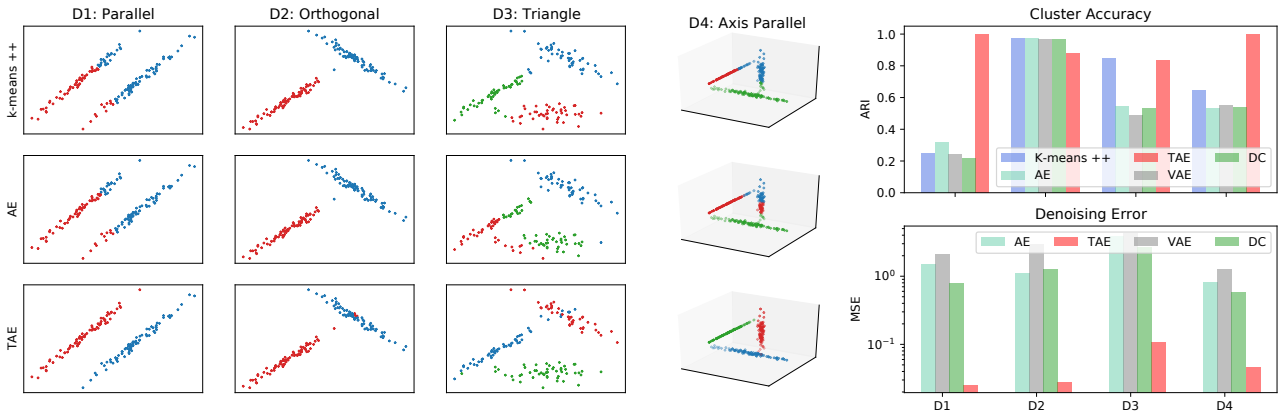

Figure 3: Illustration for cases where separate embedding for clusters are beneficial. **Left grid.** Plotted is the true data in $\mathbb{R}^2$. Top row: Clustering obtained from k-means ++ on the original features. Middle row: k-means ++ on the embedding obtained by a standard AE. Bottom row: k-means ++ on the proposed TAE. **Right plots.** Top: alignment of clustering with true labels measured by ARI (Rand, 1971). Bottom MSE for denosing on the same datasets.

by the TAE does not have to match the one imposed by the ground truth. This is beneficial in cases where there is not a unique way to cluster given data (e.g. when clustering cars, one might group by color or by type). We see this advantage empirically illustrated in Figure 3, right, where we see that the MSE of the simple AE is consistently significantly higher than for the TAE. We further validate this intuition empirically in Section 3.

Contributions This paper provides the following main contributions. (a) To learn different representations for each cluster in the data we formally propose tensorized autoencoders with k-means penalty in section 2, that simultaneously learns the embedding and cluster assignments. (b) In section 2.2 we prove that this architecture with linear encoder and decoder recovers the h leading eigenvectors of the different clusters instead of the eigenvectors of the whole data-set as done by standard linear AEs. (c) Empirically we demonstrate in section 3 that the general concept can be extended efficiently to the non-linear setting as well and TAE perform well in clustering and de-noising tasks on real data. (d) Finally in section 4 we show how TAE are connected to Expectation maximization. We provide related works and concluding remarks in sections 5–6 and all the proofs in Appendix.

Notation. We denote vectors as lowercase bold, $\boldsymbol{a}$, matrices as uppercase bold, $\boldsymbol{A}$ and tensors calligraphic as $\mathcal{A}$. Let $\|\cdot\|^2$ be the standard square norm and a function f parameterized by Θ as f_Θ. Furthermore let $\mathbb{1}_n$ be the all ones vector of size n and $\mathbb{I}_n$ the $n \times n$ identity matrix.

2 ANALYSIS OF TENSORIZED LINEAR AE WITH K-MEANS PENALTY

To get a better understanding of the behaviour of the proposed architecture in (2), we consider a simple *linear-AE* as this allows us to analytically derive the optimal parameterization. We extend the analysis empirically to a non-linear setting in section 3.

2.1 Formal Setup

For simplicity consider the clustering setting where $\widetilde{\boldsymbol{X}} := \boldsymbol{X} \in \mathbb{R}^{d\times n}$, with n being the number of data-points and d the feature dimension. Note that the analysis extends directly to the de-noising setting as well. For a two layer linear AE, let $\mathcal{U} \in \mathbb{R}^{k\times h\times d}$ and $\mathcal{V} \in \mathbb{R}^{k\times d\times h}$ be the encoding and decoding tensor respectively. Then for each $1 \le j \le k$, let $\boldsymbol{U}_j \in \mathbb{R}^{h\times d}$, be the *encoding* matrix by taking the appropriate slice of the tensor $\mathcal{U}$. Essentially $\boldsymbol{U}_j$ corresponds to the encoding function of the j'th cluster. We assume $\boldsymbol{U}_j$ is a projection matrix i.e. $\boldsymbol{U}_j\boldsymbol{U}_j^T = \mathbb{I}_h$. Similarly define $\boldsymbol{V}_j^{d\times h}$ from the *decoder* $\boldsymbol{V}$ (injection matrix).Finally let $\boldsymbol{C}_j$ be the cluster centers and λ be the weight assigned to the penalty. We treat it as a hyperparameter. From there we define the loss function as

$$\begin{aligned}\mathcal{L}_\lambda(\boldsymbol{X}) := \sum_{i=1}^{n}\sum_{j=1}^{k} \boldsymbol{S}_{j,i}\Big[\Big\|\big(\boldsymbol{X}_i - \boldsymbol{C}_j\big) - \boldsymbol{V}_j\boldsymbol{U}_j\big(\boldsymbol{X}_i - \boldsymbol{C}_j\big)\Big\|^2 \\ - \lambda\Big\|\boldsymbol{U}_j\big(\boldsymbol{X}_i - \boldsymbol{C}_j\big)\Big\|^2\Big], \\ \text{s.t.}\quad \mathbb{1}_k^T\boldsymbol{S} = \mathbb{1}_n^T,\ \boldsymbol{S}_{j,i} \ge 0,\end{aligned} \tag{3}$$

where we define $\boldsymbol{S}$ to be a $k \times n$ matrix, such that $\boldsymbol{S}_{j,i}$ is the probability that datapoint i belongs to class j. To ensure that the entries can be interpreted as probabilities we impose the above stated constraints. To further illuminate the

intuition behind S consider a dataset $\{X_i\}_{i=1}^n$ where associated to each datapoint X_i there are probabilities $\{S_{j,i}\}_{j=1}^k$ corresponding to how certain we are that X_i is sampled from the true distribution. These $\{S_{j,i}\}$ are latent variables that we learn from the dataset.

2.2 Parameterization at optimum

For the linear setting we derive the parameterization at the optimum but for reference we first recall the optimum of a standard linear AE:

Theorem 2.1 (Parameterization at Optimal for Linear AE)**.** *Baldi and Hornik (1989) showed that linear AE without regurlarization finds solutions in the principal component spanning subspace, but the individual components and corresponding eigenvalues cannot be recovered. Kunin et al. (2019) show that l_2 regularization reduces the symmetry solutions to the group of orthogonal transformations. Finally Bao et al. (2020) show that non-uniform l_2 regularization allows linear AE to recover ordered, axis-aligned principal components.*

From this we note that we only learn a single representation for the data and therefore cannot capture the underlying cluster structures. To give the intuition of this cost, before we characterize the optimal parameters of a linear TAE, lets consider a single datapoint X_i assigned to cluster C_j and its cost is,

$$S_{j,i}\left[\left\|\left(X_i - C_j\right) - V_j U_j\left(X_i - C_j\right)\right\|^2 - \lambda\left\|U_j\left(X_i - C_j\right)\right\|^2\right].$$

Ignoring C_j, the cost is simply the cost of X_i in AE weighted by the probability of it belonging to cluster j (i.e. $S_{j,i}$). With this intuition, we characterize the optimal parameters of a linear TAE in the following theorem.

Theorem 2.2 (Parameterization at Optimal for TAE)**.** *For $0 < \lambda \leq 1$, optimizing Eq. 3 results in the parameters at the optimum satisfying the following:*

i) ***Class Assignment.*** *While in Eq. 3 we define $S_{j,i}$ as the probability that X_i belongs to class j at the optimal $S_{j,i} = 1$ or 0 and therefore converges to a strict class assignment.*

ii) ***Centers.*** *C_j at optimum naturally satisfies the condition*

$$C_j = \frac{\sum_{i=1} S_{j,i} X_i}{\sum_{i=1} S_{j,i}}.$$

iii) ***Encoding / Decoding (learned weights).*** *We first show that $V_j^T = U_j$, and define*

$$\hat{\Sigma}_j := \sum_{i=1}^n S_{j,i}\left(X_i - C_j\right)\left(X_i - C_j\right)^T,$$

then the encoding corresponds to the top h eigenvectors of $\hat{\Sigma}_j$.

At the above values for the parameters, C_j and $\hat{\Sigma}_j$ acts as estimates for the means and covariances for each specific class respectively. Thus assuming that S gives reasonable cluster assignments, U_j and V_j combined gives the principal components of each cluster. While points ii) and iii) follow directly by deriving the parameterization at the optimal and we give the full derivation in the supplementary material, we give a short intuition on i) here. First we note that as per our current definition $\mathcal{L}$ is linear in S. Thus the global optimum of the loss with respect to the aforementioned linear conditions on S must be at some vertex of the convex polytope defined by the conditions. Since these conditions are $\mathbb{1}_k^T S = \mathbb{1}_n^T$ and $S_{j,i} \geq 0$, at any of the vertices of the corresponding polytope we have that $S_{j,i} = 1$ *or* 0. This combined with the fact that at global optimum C_j satisfies the condition $C_j = \frac{\sum_{i=1} S_{j,i} X_i}{\sum_{i=1} S_{j,i}}$ implies that the global optimum of the loss $\mathcal{L}$ in this expanded space is precisely same as that of the cost in the strict case we discuss in remark 2.1. This is important as it shows that even though we allow the optimization using gradient decent the parameterization at the optimal assigns each datapoint to one AE.

Let us compare Theorem 2.2 to the general notion proposed in the introduction: We would like an approach to obtain a separate meaningful representation (in the sense of recovering principal directions) for each cluster structure without having prior knowledge of which cluster a given datapoint belongs to. Theorem 2.2 shows that the proposed tensorized AE (Eq. 3) fulfills those requirements as TAE recover the top k eigenvectors for each cluster separately.

Remark 2.1 (Strict cost function)**.** Since we showed in Theorem 2.2 that Eq. 3 converges to a strict class assignment we can alternatively also define the loss function directly with strict class assignments as follows: Let $\overline{X}^{S(i)}$ be the center of all data-points which X_i belongs to. We define the loss function with strict class assignments as

$$\min_S \min_{V_{S(i)}, U_{S(i)}} \min_{C_{S(i)}} \sum_{i=1}^n \left\|\left(X_i - \overline{X}^{S(i)}\right) - V_{S(i)} U_{S(i)}\left(X_i - \overline{X}^{S(i)}\right)\right\|^2 - \lambda\left\|U_{S(i)} X_i - C_{S(i)}\right\|^2$$

Similar to Theorem 2.2 we again characterize the parameters at the optimal and most importantly note that with

$$\hat{\Sigma}_j = \frac{\sum_{i:S(i)=j}\left(X_i - \overline{X}^j\right)\left(X_i - \overline{X}^j\right)^T}{|\{i : S(i) = j\}|},$$

as long as $\lambda < 1$, the optimal projection U_j for the above cost is exactly the top h eigenvectors of $\hat{\Sigma}_j$.

2.3 Optimization

While in the previous section we discussed the optimum that is obtained by solving the optimization problem in Eq. 3 we now look at how to practically train the tensorized AE.

The general steps for learning the encoder and decoder are summarized as follows, where step 2 and 3 are repeated until convergence.

1. **Initialize** weights and cluster assignments according to k-means ++ (Arthur and Vassilvitskii, 2007)[1].
2. **Update the weights** for the encoder and decoder (using e.g. a GD step).
3. **Update the class assignment** $\boldsymbol{S}$. To do so we consider a number of different options:

   Option 1: using a GD step under constraints $\mathbb{1}_k^T \boldsymbol{S} = \mathbb{1}_n^T,\ \boldsymbol{S}_{j,i} \geq 0$.

   Option 2: using an un-constrained GD step and project $\boldsymbol{S}$ back onto the constraints.

   Option 3: using a Lloyd's step[2] on a strict class-assignment.

The choice of options in step 3 mostly depends on the framework of implementation. As noted, $\boldsymbol{S}$ as defined in Eq. 3 allows us to perform gradient updates on the class assignments. The advantage is that since $\boldsymbol{S}$ is not defined binary, frameworks such as CVXPY (Diamond and Boyd, 2016) or Keras (Chollet et al., 2015) can be used to perform a constrained gradient steps, however in popular frameworks such as PyTorch (Paszke et al., 2019), that as of the time of writing this paper do not directly support constraint optimization *Option 3* can be used.

While the general goal of the proposed setup is to learn good data representations, the exact train and test steps depend on the downstream task. In this work there are two main settings. For instance in *clustering* we directly apply the above steps and jointly learn the clustering and embedding. While in a simple AE we would have to apply a clustering algorithm onto the latent representations, in TAE the above steps directly provide the clusters. While for the linear case we prove that $\boldsymbol{S}_{ij}$ is binary, in practice, especially with more complex networks one has to compute $\operatorname{argmax}_j \boldsymbol{S}_{ij}$ to determine the class assignment. On the other hand in *de-nosing* we again jointly learn the embedding and cluster assignment but only train on the train set and in a second step use the learned TAE to de-noise images in the test set. While for a simple AE we can directly pass test data, with a TAE we use the approach presented in section 2.4 that allows us to assign the new datapoint to the appropriate AE to process it. This same setup could be used for tasks such as super resolution or inpainting. Importantly, while the results from Theorem 2.2 hold only for the exact linear formulation in Eq. 3, the main idea and training steps can be extended to arbitrary encoding and decoding functions which we illustrate empirically in section 3.

[1]Note on the initialization: while in this algorithm we consider a k-means++ in cases where we have access to some labels (e.g. in a semi-supervised setting) this can be replaced by considering random points with given labels for each cluster as initializations.

[2]Here the Lyod's step solves the linear problem on $\boldsymbol{S}$ assuming all other parameters are fixed.

2.4 Test on new data

In the context of De-noising and validating clustering on an independent test dataset or other downstream tasks, we generally need a way of encoding and decoding a new or test datapoint. In contrast to AE, where one simply passes any new data through the trained network, TAE additionally decides on the latent variable $\boldsymbol{S}$. Note then that essentially in our model the actual parameters are $\boldsymbol{C}_j, \boldsymbol{U}_j, \boldsymbol{V}_j$ whereas $\boldsymbol{S}$ is simply an encoding of our confidence of what the latent variable or label is for each $\boldsymbol{X}_i$. Thus to run the TAE on a new datapoint, we have to first estimate this latent variable.

Following this idea, let $\boldsymbol{s} \in \mathbb{R}^k$, with its j'th coordinate being $\boldsymbol{s}_j$.[3] Then given a new datapoint $\widetilde{\boldsymbol{X}}_i$ and given the trained parameters $\boldsymbol{C}_j, \boldsymbol{U}_j, \boldsymbol{V}_j$, we first find $\hat{\boldsymbol{s}}$ such that

$$\hat{\boldsymbol{s}} = \underset{\mathbb{1}_k^T \boldsymbol{s}=1; \boldsymbol{s}_j \geq 0}{\operatorname{argmin}} \sum_{j=1}^{k} \boldsymbol{s}_j \left[\left\| (\widetilde{\boldsymbol{X}}_i - \boldsymbol{C}_j) - \boldsymbol{V}_j \boldsymbol{U}_j (\widetilde{\boldsymbol{X}}_i - \boldsymbol{C}_j) \right\|^2 - \lambda \left\| \boldsymbol{U}_j (\widetilde{\boldsymbol{X}}_i - \boldsymbol{C}_j) \right\|^2 \right].$$

We use this setting for example for a de-noising task such that the de-noised reconstruction of $\widetilde{\boldsymbol{X}}_i$ would be given by $(\mathbb{I} - \boldsymbol{V}_{\hat{j}} \boldsymbol{U}_{\hat{j}})(\widetilde{\boldsymbol{X}}_i - \boldsymbol{C}_{\hat{j}}) + \boldsymbol{C}_{\hat{j}}$.

We consider the following two cases to present a heuristic that the step in general does not worsen the prediction. On the one hand if *clusters are well separated*, this implies that the separate AEs are quite different and therefore a wrong assignment would be significant. However well separated clusters also implies that the assignment of a new point is with high probability correct. On the other hand if the *clusters are not well separated* then the point might be assigned to the wrong cluster more easily, however similar clusters also implies similar AEs so even if we assign a new point to the wrong AE the reconstruction is still close to the one from the correct AE.

3 EXPERIMENTS WITH NON-LINEAR AND CONVOLUTIONAL NETWORKS

The above linear setting allows us to perform a thorough analysis of the model and prove that the parameterization at the optimal fulfills the desired criteria of learning a meaningful representation for each cluster instead of one for the whole dataset. However in an applied setting we would like

[3]Note then that this is a linear problem in the variables $\boldsymbol{s}_j$ and thus its solution is at some vertex of the bounding polytope, i.e. there is some $\hat{j}$ such that $\hat{\boldsymbol{s}}_{\hat{j}} = 1$. Thus the label assigned to $\widetilde{\boldsymbol{X}}_i$ is this j.

to take advantage of the expressive power of more complex, non-linear encoders and decoders as well. Therefore we first discuss how tensorized AE can be extended to a more general setting and additionally empirically validate its performance.

All further details on the implementation[4] and experiments are provided in the supplementary material.

3.1 Extension to Arbitrary AE Architectures

Given a single datapoint x and some specific architecture, let $\mathcal{L}_{\{\Phi_j,\Psi_j\}_{j=1}^k}(x)$ and $g_{\Psi_j}(\cdot)$ be the corresponding loss function and encoder parameterized by Ψ_j respectively. Then the corresponding tensorized autoencoder is generated by first considering k many independent copies of the above AE $\{g_{\Psi_j}(\cdot)\}_{j=1}^k$. Then the tensorized loss is defined by

$$\mathcal{L}(X) := \sum_{i=1}^{n}\sum_{j=1}^{k} S_{j,i}\left[\mathcal{L}_{\Phi_j,\Psi_j}\left(X_i - C_j\right) - \lambda\left\|g_{\Psi_j}\left(X_i - C_j\right)\right\|^2\right],$$

where C_j is defined as $C_j = \frac{\sum_{i=1} S_{j,i}X_i}{\sum_{i=1} S_{j,i}}$. While a more involved analysis is required to prove the exact latent representation in this case, the overall idea presented in (2). Again for the training we follow the steps presented in Section 2.3.

3.2 Clustering on Real Data

While on the toy data in Figure 3 we clearly observed how the analyzed approaches deal with different datastructures, we extend the analysis to real data and also more complex AE architectures.

For comparability of the number of parameters we furthermore compare *AE 1* with latent dimension d and *AE 2* with latent dimensions $d \times k$ in Figure 5. We note that their performance on the clustering task is very comparable and we therefore conclude that the performance of the TAE is not a direct result of the increased number of parameters but is achieved due to the different architecture.

To start the analysis we consider the penguin dataset presented in the introduction. Figure 4 (top left) shows the simpson's paradox for the penguin dataset and the other plots show the clusterings the different algorithms converge to. Notably we see that for k-means as well as for both AE architectures, the y-axis parallel decision boundaries are obtained where as for the TAE we obtain clusters that are closer to the true structures.

To further quantify the empirical performance of TAE we show the average and standard deviation over five runs for the 'penguin dataset' with two and four features as well as the 'iris dataset' (Fisher, 1936; Anderson, 1936) and 'MNIST' (LeCun and Cortes, 2010) (sub-sample of 50 datapoints for class $\{1, 2, 3, 4, 5\}$ each) for simple linear networks as well as two layer convolutional networks. We show this in Figure 5. For penguin dataset and iris we consider $d = 1$ and $d = 10$ for MNIST. We observe that the difference in the performance between AE and TAE is significant for the penguin dataset which we attribute to the above outlined clustering structures. For MNIST, we observe very similar performance of AE and TAE, which is most likely due to the fact that the underlying clustering structures have very similar properties, such there is no significant difference for the TAE to exploit.

[4]The code is available at: https://github.com/mahalakshmi-sabanayagam/tensorized_autoencoder

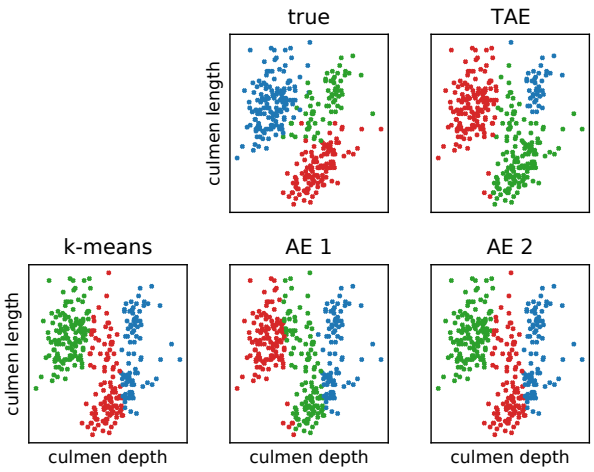

Figure 4: Illustration of the final clustering obtained for the *penguin dataset, two features* by the different algorithms.

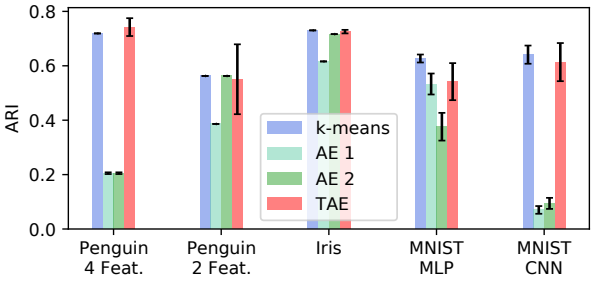

Figure 5: Comparison of k-means ++, standard AE with k-means ++ and TAE. AE 1 has latent dimension d while AE 2 has latent dimension $d \times k$. MNIST MLP shows the performance for a single hidden layer network, while MNIST CNN considers an encoder and decoder with two convolutional layers each. Again the latent dimension is d for both networks.

3.3 De-Noising on Real Data

Similar to the experiments on clustering real data we look at de-noising on various real datasets with standard AE and TAE. We use the same datasets we used in the clustering case to illustrate that TAE learns a reasonable clustering

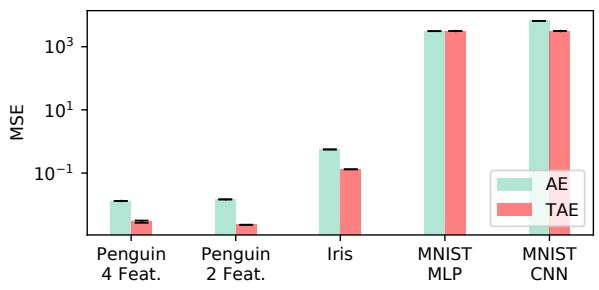

Figure 6: Real data de-noising. Note that the MSE is plotted in log scale

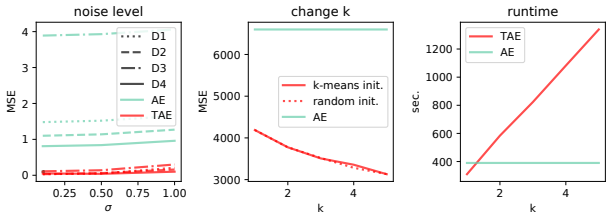

Figure 7: **(left)** De-nosing under change in noise level for the four toy datasets. **(middle)** De-nosing for five classes MNIST using AE and TAE with non-optimal k. **(right)** De-nosing for five classes MNIST. Analysis of run-time with changing number of k in TAE.

as well as a better reconstruction. The original data was corrupted by adding an isometric Gaussian to each data point. We illustrate this difference in Figure 6 and observe that TAE consistently performs the same or significantly better then the standard AE. For the penguin dataset we observe a better cluster recovery and a better de-noising performance. Interestingly for the Iris dataset the reconstruction obtained by TAE is significantly better the the one by the standard AE while their clustering performance is about the same. Finally for MNIST with CNN we get a markedly improved performance on using TAE, while using a single layer linear neural network as the underlying architecture doesn't gives us any significant improvement in performance. This suggests that it is important to choose a function class or underlying architecture that is capable of capturing a low dimensional representation of the data.

Let us consider the presented de-nosing setting for some further investigation of the model.

Different levels of noise. When considering de-noising an obvious question on the influence of the noise level on the performance of AE and TAE. To analyze this we go back to the toy datasets introduced in section 1 and observe in Figure 7 (left) that while with increasing noise level the the MSE increases for both approaches, TAE consistently outperforms the standard AE on all datasets.

Performance of TAE when k is not the true number of clusters. Let us consider the MNIST setting stated before with five classes. We observe in Figure 7 (middle) that the MSE is monotonously improved with k getting closer to k_{true}, all improving on standard AE. This indicates that even if the number of clusters is not known[5] TAE for de-noising tasks performs better then standard AE.

Importance of initialization of S. While we propose k-means++ as an initialization procedure for the cluster assignment, empirically the question is how important this assumption is. Figure 7 (middle) illustrates that in the previously considered MNIST setting there is no noticeable difference between the the k-means++ initialization and a random initialization. However we conjecture that the difference increases for more complex the dataset (in the sense of structures such as introduced in Section 1).

Computational demands. Finally an important point is the computational comparison between AE and TAE. By construction one can see that the computational complexity scales linearly with the number of considered clusters. We can observe that this also holds empirically as shown in Figure 7 (right), again for the MNIST setting considered in the previous two points.

4 CONNECTION TO EXPECTATION MAXIMIZATION

The astute reader would notice that the gradient descent step that we are proposing is similar to Expectation Maximization (EM) algorithm. To explain this connection more carefully (and propose a slightly modified algorithm in the case of Gaussian data), we first write down the EM algorithm itself.

Let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_n$ be data coming from from some distribution in the set of distributions with parameter θ with some latent or unobserved variables Z. Let $L(\theta; \boldsymbol{X}, Z)$ be the likelihood function of the parameters. The EM algorithm then seeks to maximise

$$Q(\theta) = \sum_{i=1}^{n} \frac{1}{n} \mathbb{E}_{Z|\boldsymbol{X}_i,\theta}[\log L(\theta; \boldsymbol{X}_i, Z)].$$

To make matters a little clearer let us see what this implies when the data comes from a mixture of Gaussians. Let

$$\boldsymbol{X} \sim \sum_{i=1}^{k} p_i N(\mu_i, \Sigma_i).$$

To imagine a latent variable imagine the data being generated as follows. First sample a random variable Z such

[5] This is a general problem in any clustering based tasks, and also present in deep clustering (Yang et al., 2017) works. A possible future approach would be along the lines of the semidefinite programming relaxation in (Yan et al., 2017), however this comes with computational overhead and the extension is not trivial.

that $p(Z = i) = p_i$. Then sample an observation from $N(\mu_z, \Sigma_Z)$. Thus in this case the latent variable Z is the assignment of the data to its respective cluster or the observed data's label. The parameters θ in this case is the vector $(p_1, ..., p_k, \mu_1, ..., \mu_k, \Sigma_1, ..., \Sigma_k)$. Then given data $\boldsymbol{X}_i$, the probability the corresponding Z is j is given by (using Bayes rule) :

$$\begin{aligned} S_{j,i} &= \frac{p_j f(\boldsymbol{X}_i; \mu_j, \Sigma_j)}{\sum_t p_t f(\boldsymbol{X}_i; \mu_t, \Sigma_t)} \\ &= \frac{p_j \sqrt{\det(2\pi\Sigma_j)} \exp\left(-\frac{1}{2}(\boldsymbol{X}_i - \mu_j)\Sigma_j^{-1}(\boldsymbol{X}_i - \mu_j)\right)}{\sum_t p_t \sqrt{\det(2\pi\Sigma_t)} \exp\left(-\frac{1}{2}(\boldsymbol{X}_i - \mu_t)\Sigma_t^{-1}(\boldsymbol{X}_i - \mu_t)\right)}. \end{aligned}$$

Then the expectation of the log likelihood $\mathbb{E}_Z[L(\theta; X, Z)]$ is :

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} S_{j,i}\left(\frac{\log\ \det(2\pi\Sigma_j)}{2} - \frac{\|\Sigma_j^{-1/2}(\boldsymbol{X}_i - \mu_j)\|^2}{2}\right).$$

Now note that ignoring the determinant term ($\log\ \det(2\pi\Sigma_j)$) maximizing this above quantity is exactly minimizing

$$\sum_{i=1}^{n}\sum_{j=1}^{k} S_{j,i}\left\|\Sigma_j^{-1/2}(\boldsymbol{X}_i - \mu_j)\right\|^2.$$

Note then that this is very similar to the term we are optimizing if we allow S to be independent of Σ_j and set

$$\Sigma_j^{-1} = (\mathbb{I} - \boldsymbol{V}_j\boldsymbol{U}_j)^T(\mathbb{I} - \boldsymbol{V}_j\boldsymbol{U}_j) - \lambda\boldsymbol{U}_j^T\boldsymbol{U}_j, \quad \text{and} \quad \mu_j = \boldsymbol{C}_j.$$

In other words we are fixing the covariance matrix to have numerically low rank, i.e. it has only h (few) of its eigenvalues are $1/\lambda$ (very large) whereas rest of them are 1 (small).

The connection between TAE and the EM algorithm is especially interesting as there is a vast literature on the theoretical properties of EM and therefore opens up future research directions for a more fine-grained analysis of TAEs.

5 RELATED WORK AND FUTURE APPLICATIONS

Theoretical analysis of AE. The theoretical understanding of simple AE is still limited and mainly summarized in Theorem 2.1 and formalizes the optimal parameterization of linear AE depending on the considered regularization (Baldi and Hornik, 1989; Kunin et al., 2019; Bao et al., 2020; Pretorius et al., 2018). While the considered proof techniques differ in our work we derive a similar result of the TAE. An important future direction for theoretical analysis for both simple AE and TAE is the extension to the non-linear setting.

Before going into the related work on clustering and de-noising we would like to note here that the focus of this work is to show the difference in learning a single representation for the data and a representation for each cluster in the data. The general literature of possible, task specific, AE models is vast and would exceed the limits of this related work section. Therefore we focus on the most relevant related work, which is the basic setup for clustering on embedding and de-noising with simple AEs.

Clustering. The main goal of clustering is to group similar objects into the same class in an unsupervised setting. While this problem has been extensively studied in traditional machine learning in terms of feature selection (Boutsidis et al., 2009; Alelyani et al., 2013), distance functions (Xing et al., 2002; Xiang et al., 2008) and group methods (Macqueen, 1967; von Luxburg, 2007; Li et al., 2004) (for a more comprehensive overview see Aggarwal and Reddy (2014)) the time complexity significantly increases with high dimensional data, previous work focuses on projecting data into low-dimensional spaces and then cluster the embedded representations (Roth and Lange, 2003; Tian et al., 2014; Wang et al., 2016). For there there several methods have been developed that use deep unsupervised models to learn representations with a clustering focus that simultaneously learns feature representations and cluster assignments using deep neural networks (Xie et al., 2016; Dizaji et al., 2017; Wang et al., 2016; Xie and Xing, 2015; Wang et al., 2015).

De-noising. We consider de-noising with AEs (Buades et al., 2005; Cho, 2013). While there are several extensions to more complex AE models and task specific setups (see e.g. Zhang et al. (2022)) in this work we focus on the question if learning cluster specific representations is beneficial for reducing the reconstruction error, which to the best of our Knowledge has been considered so far.

Possible future applications. We note that while in this paper we focus on *clustering and de-noising* the general concept can be extended to other AE based downstream tasks such as anomaly detection (Morales-Forero and Bassetto, 2019; Sakurada and Yairi, 2014; Zhou and Paffenroth, 2017), image compression (Theis et al., 2017; Ballé et al., 2017), super resolution (Zeng et al., 2017; Song et al., 2017) and machine translation (Cho et al., 2014; Sutskever et al., 2014).

Such extensions are especially of interest for future applications as Figure 3 and Section 3.3 show that TAE outperform standard AE especially when measured in the reconstruction quality. This indicates that tasks such as image compression can benefit from using TAE, as for such tasks the matching to the true clustering is not required.

6 CONCLUSION

This work presents a meta algorithm which can be used to better adapt any existing autoencoder architecture to datasets in which one might anticipate cluster structures.

By jointly learning the cluster structure and low dimensional representations of clusters, the proposed tensor autoencoder (TAE) directly improves upon Kmeans, applied to a dataset or to an encoding of the same generated by an AE. More importantly, in the context of de-noising or downstream learning tasks, while it is trivial to note that mathematically a TAE can never have worse MSE than a corresponding AE, we verify the same experimentally. On the surface the difference in performance might simply seem to be a matter of more parameters. However we show experimentally (see experiment on real data clustering, Figure 5) that even with the same total number of parameters TAE clusters better and hence gives a more accurate reconstruction than an AE. An open question in this regard would be to show such a result mathematically under certain data assumptions. Another open implementation problem would be to design a more a efficient gradient step for the TAE.

7 SOCIAL IMPACT

Usage of traditional autoencoders on large diverse communities often favour creating a single latent monolithic representations mostly representing a single majority. This might create situations where the interests of various smaller communities are ignored. An instance of this might be to use autoencoders to find a couple of parameters which are the most significant markers of a particular disease. Using a tensorized autoencoders might be of interest in these situations as this might create multiple representations for each community. On the other hand in cases where recovering more explicit clustering structures have negative implications TAE might not be favorable. A possible example here could be differential privacy. However to better understand this setting the interplay between properties of latent representations in AE and privacy will has to be more thoroughly investigated in the future.

Acknowledgements

This work has been supported by the German Research Foundation (DFG) through the Priority Program SPP-2298 (project GH-257/2-1), DFG-ANR PRCI "ASCAI" (GH 257/3-1) and Research Training Group GRK 2428.

Bibliography

Aggarwal, C. C. and Reddy, C. K., editors (2014). *Data Clustering: Algorithms and Applications*. CRC Press.

Alelyani, S., Tang, J., and Liu, H. (2013). Feature selection for clustering: A review. In *Data Clustering: Algorithms and Applications*.

Anderson, E. (1936). The species problem in iris. *Annals of the Missouri Botanical Garden*.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. *Proc. of the Annu. ACM-SIAM Symp. on Discrete Algorithms*.

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*.

Ballé, J., Laparra, V., and Simoncelli, E. P. (2017). End-to-end optimized image compression. In *Int'l Conf on Learning Representations (ICLR)*.

Bao, X., Lucas, J., Sachdeva, S., and Grosse, R. B. (2020). Regularized linear autoencoders recover the principal components, eventually. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6971–6981. Curran Associates, Inc.

Boutsidis, C., Drineas, P., and Mahoney, M. W. (2009). Unsupervised feature selection for the k-means clustering problem. In *Advances in Neural Information Processing Systems*.

Buades, A., Coll, B., and Morel, J.-M. (2005). A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal*.

Cho, K. (2013). Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images. *Proceedings of the 30th International Conference on Machine Learning*.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Diamond, S. and Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*.

Dizaji, K., Herandi, A., Deng, C., Cai, W., and Huang, H. (2017). Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. *International Conference on Computer Vision (ICCV)*.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*.

Gorman, K. B., Williams, T. D., and Fraser, W. R. (2014). Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus pygoscelis). *PLOS ONE*.

Holt, G. B. (2016). Potential simpson's paradox in multicenter study of intraperitoneal chemotherapy for ovarian cancer. *Journal of Clinical Oncology*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243.

Kunin, D., Bloom, J., Goeva, A., and Seed, C. (2019). Loss landscapes of regularized linear autoencoders. In *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

Li, T., Ma, S., and Ogihara, M. (2004). Entropy-based criterion in categorical clustering. *Proceedings of the International Conference on Machine Learning*.

Macqueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

Morales-Forero, A. and Bassetto, S. (2019). Case study: A semi-supervised methodology for anomaly detection and diagnosis. In *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*.

Pretorius, A., Kroon, S., and Kamper, H. (2018). Learning dynamics of linear denoising autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*.

Roth, V. and Lange, T. (2003). Feature selection in clustering problems. In *Advances in Neural Information Processing Systems*.

Sakurada, M. and Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *MLSDA'14: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*.

Simpson, E. H. (1951). The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society. Series B (Methodological)*.

Song, T.-H., Sanchez, V., ElDaly, H., and Rajpoot, N. M. (2017). Hybrid deep autoencoder with curvature gaussian for detection of various types of cells in bone marrow trephine biopsy images. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*.

Theis, L., Shi, W., Cunningham, A., and Huszár, F. (2017). Lossy image compression with compressive autoencoders. In *International Conference on Learning Representations*.

Tian, F., Gao, B., Cui, Q., Chen, E., and Liu, T.-Y. (2014). Learning deep representations for graph clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*.

Wagner, C. H. (1982). Simpson's paradox in real life. *The American Statistician*.

Wang, Z., Chang, S., Zhou, J., and Wang, M. (2016). Learning a task-specific deep architecture for clustering. Proceedings of the 2016 SIAM International Conference on Data Mining.

Wang, Z., Yang, Y., Chang, S., Li, J., Fong, S., and Huang, T. S. (2015). A joint optimization framework of sparse coding and discriminative clustering. In *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press.

Xiang, S., Nie, F., and Zhang, C. (2008). Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*.

Xie, J., Girshick, R., and Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis.

Xie, P. and Xing, E. P. (2015). Integrating image clustering and codebook learning.

Xing, E., Jordan, M., Russell, S. J., and Ng, A. (2002). Distance metric learning with application to clustering with side-information. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems*.

Yan, B., Sarkar, P., and Cheng, X. (2017). Provable estimation of the number of blocks in block models. In *International Conference on Artificial Intelligence and Statistics*.

Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. (2017). Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning*.

Zeng, K., Yu, J., Wang, R., Li, C., and Tao, D. (2017). Coupled deep autoencoder for single image super-resolution. *IEEE Transactions on Cybernetics*.

Zhang, C., Zhang, C., Song, J., Yi, J. S. K., Zhang, K., and Kweon, I. S. (2022). A survey on masked autoencoder for self-supervised learning in vision and beyond.

Zhou, C. and Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. *Knowledge Discovery in Databases '17: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

A PROOF OF THEOREM 2.2

For clarity let us first restate the considered loss function

$$\mathcal{L}_\lambda(X) := \sum_{i=1}^{n}\sum_{j=1}^{k} S_{j,i}\Big[\Big\|\big(X_i - C_j\big) - V_j U_j\big(X_i - C_j\big)\Big\|^2 - \lambda\Big\|U_j\big(X_i - C_j\big)\Big\|^2\Big],$$

$$\text{s.t.}\quad \mathbb{1}_k^T S = \mathbb{1}_n^T, \quad S_{j,i} \geq 0, \tag{4}$$

where we define S to be a $k \times n$ matrix, such that $S_{j,i}$ is the probability that datapoint i belongs to class j and the theorem :

Theorem A.2 (Parameterization at Optimal for TAE)**.** *For $0 < \lambda \leq 1$, optimizing Eq. 4 results in the parameters at the optimum satisfying the following:*

i) ***Class Assignment.*** *While in Eq. 4 we define $S_{j,i}$ as the probability that X_i belongs to class j at the optimal $S_{j,i} = 1$ or 0 and therefore converges to a strict class assignment.*

ii) ***Centers.*** *C_j at optimum naturally satisfies the condition*

$$C_j = \frac{\sum_{i=1} S_{j,i} X_i}{\sum_{i=1} S_{j,i}}.$$

iii) ***Encoding / Decoding (learned weights).*** *We first show that $V_j^T = U_j$, and define*

$$\hat{\Sigma}_j := \sum_{i=1}^{n} S_{j,i}\big(X_i - C_j\big)\big(X_i - C_j\big)^T,$$

then the encoding corresponds to the top h eigenvectors of $\hat{\Sigma}_j$.

Proof. (i) This part is trivial once we note that for a fixed C_j, U_j, V_j, the loss is linear in $S_{j,i}$. Thus the optimal must occur at the extreme points of the constraints

$$\mathbb{1}_k^T S = \mathbb{1}_n^T$$
$$S_{j,i} \geq 0.$$

Let us first prove part (iii). To get conditions on U_j, V_j let us fix S, C_j and get conditions of optimal in terms of the fixed quantities. Define

$$\hat{\Sigma}_j := \sum_{i=1}^{n} S_{j,i}(X_i - C_j)(X_i - C_j)^T$$

Let us now optimize $\boldsymbol{V}_j, \boldsymbol{U}_j$ for each fixed j.

$$
\begin{aligned}
&\min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \sum_{i=1}^{n} \boldsymbol{S}_{j,i} \left(\left\| (\boldsymbol{X}_i - \boldsymbol{C}_j) - \boldsymbol{V}_j \boldsymbol{U}_j (\boldsymbol{X}_i - \boldsymbol{C}_j) \right\|^2 - \lambda \left\| \boldsymbol{U}_j (\boldsymbol{X}_i - \boldsymbol{C}_j) \right\|^2 \right) \\
&= \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \sum_{i=1}^{n} \boldsymbol{S}_{j,i} \left(\left\| (\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j)(\boldsymbol{X}_i - \boldsymbol{C}_j) \right\|^2 - \lambda \left\| \boldsymbol{U}_j (\boldsymbol{X}_i - \boldsymbol{C}_j) \right\|^2 \right) \\
&= \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \sum_{i=1}^{n} \boldsymbol{S}_{j,i} \left((\boldsymbol{X}_i - \boldsymbol{C}_j)^T (\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j)^T (\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j)(\boldsymbol{X}_i - \boldsymbol{C}_j) - \lambda (\boldsymbol{X}_i - \boldsymbol{C}_j)^T \boldsymbol{U}_j^T \boldsymbol{U}_j (\boldsymbol{X}_i - \boldsymbol{C}_j) \right) \\
&= \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \sum_{i=1}^{n} \boldsymbol{S}_{j,i} \operatorname{Tr} \left[(\boldsymbol{X}_i - \boldsymbol{C}_j)^T (\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j)^T (\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j)(\boldsymbol{X}_i - \boldsymbol{C}_j) \right] - \lambda \boldsymbol{e}_j^T \boldsymbol{S} \boldsymbol{e}_i \operatorname{Tr} \left[(\boldsymbol{X}_i - \boldsymbol{C}_j)^T \boldsymbol{U}_j^T \boldsymbol{U}_j (\boldsymbol{X}_i - \boldsymbol{C}_j) \right] \\
&= \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \sum_{i=1}^{n} \boldsymbol{S}_{j,i} \operatorname{Tr} \left[(\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j)^T (\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j)(\boldsymbol{X}_i - \boldsymbol{C}_j)(\boldsymbol{X}_i - \boldsymbol{C}_j)^T \right] - \lambda \boldsymbol{e}_j^T \boldsymbol{S} \boldsymbol{e}_i \operatorname{Tr} \left[\boldsymbol{U}_j^T \boldsymbol{U}_j (\boldsymbol{X}_i - \boldsymbol{C}_j)(\boldsymbol{X}_i - \boldsymbol{C}_j)^T \right] \\
&= \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \operatorname{Tr} \left[(\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j)^T (\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j) \hat{\Sigma}_j \right] - \lambda \operatorname{Tr} \left[\boldsymbol{U}_j^T \boldsymbol{U}_j \hat{\Sigma}_j \right] \\
&= \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \operatorname{Tr} \left[((\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j)^T (\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j) - \lambda \boldsymbol{U}_j^T \boldsymbol{U}_j) \hat{\Sigma}_j \right] \\
&= \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \operatorname{Tr} \left[(\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j - \boldsymbol{U}_j^T \boldsymbol{V}_j^T + (1 - \lambda) \boldsymbol{U}_j^T \boldsymbol{U}_j) \hat{\Sigma}_j \right].
\end{aligned}
$$

We can now bound the above as follows

$$
\begin{aligned}
&\min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \operatorname{Tr} \left[(\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j - \boldsymbol{U}_j^T \boldsymbol{V}_j^T + (1 - \lambda) \boldsymbol{U}_j^T \boldsymbol{U}_j) \hat{\Sigma}_j \right] \\
&= \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \operatorname{Tr} \left[(\mathbb{I} + (1 - \lambda) \boldsymbol{U}_j^T \boldsymbol{U}_j) \hat{\Sigma}_j \right] - \operatorname{Tr} \left[(\boldsymbol{U}_j^T \boldsymbol{V}_j^T + \boldsymbol{V}_j \boldsymbol{U}_j) \hat{\Sigma}_j \right] \\
&= \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \operatorname{Tr} \left[(\mathbb{I} + (1 - \lambda) \boldsymbol{U}_j^T \boldsymbol{U}_j) \hat{\Sigma}_j \right] - 2 \operatorname{Tr} \left[\boldsymbol{U}_j \hat{\Sigma}_j \boldsymbol{V}_j^T \right] \\
&\geq \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \operatorname{Tr} \left[(\mathbb{I} + (1 - \lambda) \boldsymbol{U}_j^T \boldsymbol{U}_j) \hat{\Sigma}_j \right] - 2 \sqrt{\operatorname{Tr} \left[\boldsymbol{U}_j \hat{\Sigma}_j^{1/2} \hat{\Sigma}_j^{1/2} \boldsymbol{U}_j^T \right] \operatorname{Tr} \left[\boldsymbol{V}_j^T \hat{\Sigma}_j^{1/2} \hat{\Sigma}_j^{1/2} \boldsymbol{V}_j \right]} \qquad (5) \\
&\geq \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \operatorname{Tr} \left[(\mathbb{I} + (1 - \lambda) \boldsymbol{U}_j^T \boldsymbol{U}_j) \hat{\Sigma}_j \right] - 2 \sqrt{\operatorname{Tr} \left[\boldsymbol{U}_j \hat{\Sigma}_j \boldsymbol{U}_j^T \right] \sum_{i=1}^{h} \alpha_i} \qquad (6) \\
&= \min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \sum_{i=1}^{d} \alpha_i - 2 \sqrt{\operatorname{Tr} \left[\boldsymbol{U}_j \hat{\Sigma}_j \boldsymbol{U}_j^T \right] \sum_{i=1}^{h} \alpha_i} + (1 - \lambda) \operatorname{Tr} \left[\boldsymbol{U}_j \hat{\Sigma}_j \boldsymbol{U}_j^T \right],
\end{aligned}
$$

where the inequality 5 follows from applying Cauchy Schwarz inequality and the inequality 6 from observing that the columns of V are h orthonormal vectors to get $\operatorname{Tr} \left[\boldsymbol{V}_j^T \hat{\Sigma}_j \boldsymbol{V}_j \right] \leq \sum_{i=1}^{h} \alpha_i$.

Let us set $x = \sqrt{\operatorname{Tr} \left[\boldsymbol{U}_j \hat{\Sigma}_j \boldsymbol{U}_j^T \right]}$, $a = 1 - \lambda$, $b = 2\sqrt{\sum_{i=1}^{h} \alpha_i}$ and $c = \sum_{i=1}^{d} \alpha_i$. We can now observe that $0 \leq x = \sqrt{\operatorname{Tr} \left[\boldsymbol{U}_j \hat{\Sigma}_j \boldsymbol{U}_j^T \right]} \leq \sqrt{\sum_{i=1}^{h} \alpha_i} = \frac{b}{2}$ and the minimization problem becomes

$$
ax^2 - bx + c
$$

which is a quadratic and thus decreasing when $x \leq \frac{b}{2a}$. Therefore as $\frac{b}{2} \leq \frac{b}{2a}$ the minimum is achieved at $x = \frac{b}{2}$ thus we get

$$
\begin{aligned}
\min_{\boldsymbol{V}_j,\boldsymbol{U}_j} \operatorname{Tr} \left[(\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j - \boldsymbol{U}_j^T \boldsymbol{V}_j^T + (1 - \lambda) \boldsymbol{U}_j^T \boldsymbol{U}_j) \hat{\Sigma}_j \right] &\geq \sum_{i=1}^{d} \alpha_i - 2 \sqrt{\sum_{i=1}^{h} \alpha_i \sum_{i=1}^{h} \alpha_i} + (1 + \lambda) \sum_{i=1}^{h} \alpha_i \\
&= \sum_{i=1}^{d} \alpha_i - (1 + \lambda) \sum_{i=1}^{h} \alpha_i
\end{aligned}
$$

Reviewing the proof above we note that for the inequality to be achieved, we must satisfy the conditions :

- $\operatorname{Tr} \left[\boldsymbol{U}_j \hat{\Sigma}_j \boldsymbol{U}_j^\top \right] = \sum_{i=1}^{h} \alpha_i$ (from optimizing the quadratic above),

- $\boldsymbol{V}_j^T \hat{\Sigma}_j^{1/2} = \boldsymbol{U}_j \hat{\Sigma}_j^{1/2}$ (from equalizing C.S. used to get inequality 5),
- $\mathrm{Tr}\left[\boldsymbol{V}_j^\top \hat{\Sigma}_j \boldsymbol{V}_j\right] = \sum_{i=1}^h \alpha_i$ (from equalizing inequality 6).

Luckily the above conditions are only *all* satisfied for the following unique choice of orthonormal $\hat{\boldsymbol{U}}_j$ and $\hat{\boldsymbol{V}}_j$. If α_i and $\boldsymbol{w}_i$ are the eigenvalues (arranged in descending order) and eigenvectors of $\hat{\Sigma}_j$ respectively define

$$\hat{\boldsymbol{U}}_j = \begin{bmatrix} \boldsymbol{w}_1^T \\ \vdots \\ \boldsymbol{w}_h^T \end{bmatrix}, \quad \hat{\boldsymbol{V}}_j = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_h].$$

We then obtain,

$$\begin{aligned} \mathrm{Tr}\left[(\mathbb{I} - \hat{\boldsymbol{V}}_j \hat{\boldsymbol{U}}_j - \hat{\boldsymbol{U}}_j^T \hat{\boldsymbol{V}}_j^T + (1-\lambda)\hat{\boldsymbol{U}}_j^T \hat{\boldsymbol{U}}_j)\hat{\Sigma}_j\right] &= \sum_{i=1}^d \alpha_i - (1+\lambda)\sum_j^h \alpha_j \\ &= -\lambda \sum_{i=1}^h \alpha_i + \sum_{j=h+1}^d \alpha_j. \end{aligned}$$

Let us finally derive (ii).

Define

$$\boldsymbol{A} := (\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j - \boldsymbol{U}_j^T \boldsymbol{V}_j^T + (1-\lambda)\boldsymbol{U}_j^T \boldsymbol{U}_j).$$

Then,

$$\begin{aligned} &\min_{\boldsymbol{C}_j} \mathrm{Tr}\left[\left(\mathbb{I} - \boldsymbol{V}_j \boldsymbol{U}_j - \boldsymbol{U}_j^T \boldsymbol{V}_j^T + (1-\lambda)\boldsymbol{U}_j^T \boldsymbol{U}_j\right)\left(\sum_{i=1}^n \boldsymbol{S}_{j,i}(\boldsymbol{X}_i - \boldsymbol{C}_j)(\boldsymbol{X}_i - \boldsymbol{C}_j)^T\right)\right] \\ &= \min_{\boldsymbol{C}_j} \mathrm{Tr}\left[\boldsymbol{A}\left(\sum_{i=1}^n \boldsymbol{S}_{j,i}(\boldsymbol{X}_i - \boldsymbol{C}_j)(\boldsymbol{X}_i - \boldsymbol{C}_j)^T\right)\right] \\ &= \min_{\boldsymbol{C}_j} \mathrm{Tr}\left[\boldsymbol{A}\left(\sum_{i=1}^n \boldsymbol{S}_{j,i}(\boldsymbol{X}_i \boldsymbol{X}_i^T - \boldsymbol{X}_i \boldsymbol{C}_j^T + \boldsymbol{C}_j \boldsymbol{C}_j^T)\right)\right] \\ &= \min_{\boldsymbol{C}_j} \mathrm{Tr}\left[\boldsymbol{A}\left(\left(\sum_{i=1}^n \boldsymbol{S}_{j,i}\boldsymbol{X}_i \boldsymbol{X}_i^T\right) - \left(\sum_{i=1}^n \boldsymbol{S}_{j,i}\boldsymbol{X}_i\right)\boldsymbol{C}_j^T + \left(\sum_{i=1}^n \boldsymbol{S}_{j,i}\right)\boldsymbol{C}_j \boldsymbol{C}_j^T\right)\right]. \end{aligned}$$

As $\boldsymbol{U}_j, \boldsymbol{V}_j$ are variables varying in a space independent of $\boldsymbol{C}_j$, at optimality of the above, the partial derivative with respect to $\boldsymbol{C}_j$ must be 0. Thus we have,

$$\frac{\partial}{\partial \boldsymbol{C}_j} = -\boldsymbol{A}\left(\sum_{i=1}^n \boldsymbol{S}_{j,i}\boldsymbol{X}_i\right) + \boldsymbol{A}\left(\sum_{i=1}^n \boldsymbol{S}_{j,i}\right)\boldsymbol{C}_j = 0 \tag{7}$$

On the other hand from the derivation of (iii), we also have the condition that at optimality $\boldsymbol{V}_j = \boldsymbol{U}_j^\top = \hat{\boldsymbol{U}}_j^\top$. Thus at optimality

$$\boldsymbol{A} = \left(\mathbb{I} - (1+\lambda)\hat{\boldsymbol{U}}_j^\top \hat{\boldsymbol{U}}_j\right).$$

In particular when $\lambda > 0$, $\boldsymbol{A}$ (at optimality) is invertible (as its eigenvalues are 0 and $-\lambda$). Using this in equation 7 we get

$$\hat{\boldsymbol{C}}_j = \frac{\sum_{i=1}^n \boldsymbol{S}_{j,i}\boldsymbol{X}_i}{\sum_{i=1}^n \boldsymbol{S}_{j,i}}.$$

□

B EXPERIMENTAL DETAILS FOR RESULTS FROM THE MAIN PAPER

In this section we provide the setup for the experiments performed in the main paper. Further details can be see in the provided code.

B.1 Experiments on Toy Data

For all experiments we consider the theoretical setting where we use a two layer linear network and $\lambda = -0.1$. The Scatter plot is for a converged clustering from a randomly sampled run. The *cluster accuracy* and *de-nosing error* plot show the average over five random initialization after 100 epochs.

We consider $h = 1$

B.2 Notes on the penguin dataset

We consider two main versions of the dataset. The *two feature* version only considers 'culmen depth' and 'culmen length' as features (this version is illustrated in Figure 1 (main paper)). The *four feature* version additionally considers 'flipper length' in mm and 'body mass' in grams.

Since the features 'body mass' and 'culmen depth' are on very different scales we normalize all features for the experiments with four features.

B.3 Experiments on Clustering on Real Data

As a default network we consider a one hidden layer, ReLU network. For Penguin and Iris we consider $h = 1$ and for MNIST $h = 10$.

For the CNN we consider a network with two encoding CNN layers and two decoding CNN layers and the same hidden dimension as for the fully connected setting. For the exact architecture we refer to the provided code.

B.4 Experiments on De-Noising on Real Data

We generally consider the same setup as for the clustering tasks with a default network of one hidden layer, ReLU network. For Penguin and Iris we consider $h = 1$ and for MNIST $h = 10$.