Visualization of high-dimensional data using autoencoders

Theo Dedeken

Supervisor(s): Yvan Saeys, Sofie Van Gassen

Abstract— **Advances in flow cytometry allow to measure increasing amounts of parameters per cell, generating high-dimensional datasets. Visualizing these datasets in their raw form is tedious and confusing. Dimensionality reduction can be used to reduce the amount of dimensions in the datasets. When reducing to two dimension we can visualize the dataset in one scatter plot. We present two autoencoder models that can perform dimensionality reduction on a high-dimensional dataset. We compare their results to UMAP over a few datasets and calculate quality metrics based on distance ranking and k-ary neighborhoods. Autoencoders retain the structure of the data more broadly than UMAP, while still producing decent visualizations.**

Keywords— **visualization, autoencoder, dimensionality reduction, mnist, flow cytometry, mass cytometry**

I. Introduction

THE field of computational flow cytometry is fast evolving. Flow cytometry measures multiple dimensions of cells that flow in a stream through a system of photonic detectors. Recent advancements in technology allow for the measurement of millions of cells with up to 30 parameters per cell. Related to this measurement technique is mass cytometry that uses mass spectrometry to measure up to 100 parameters per cell.

Visualizing this high-dimensional data is a difficult task. Once the dimensionality goes beyond three dimensions it becomes hard to visualize all these dimensions at once. A possible solution is to select a few dimensions at a time to make a plot, but by doing this a lot of the structure of the dataset is not visible. This approach is also not scalable for datasets with a lot of dimensions.

Dimensionality reduction can reduce the amount of dimensions while preserving a lot of the structure in the dataset. Reducing to two dimensions allows us to visualize the dataset in a standard scatter plot.

Autoencoders are multilayer neural networks with a small central layer to reconstruct high-dimensional input data. As autoencoders project input data on a code layer with lower dimension we can use them as dimensionality reduction methods by first training them on the high-dimensional data. Afterwards the encoder part of the network can be used as a reducer for the input data but also for any other data of the same type. This categorizes the autoencoders as parametric dimensionality reduction.

II. Methods

A. Autoencoder

The use of autoencoder networks to reduce the dimensionality of data was first proposed by Hinton [1] in 2006. The low-dimensional codes that are generated in the small central layer by training on the high-dimensional data can be seen as a reduction of this data. Our autoencoder model is inspired by this work but there have been some modifications using some new insights in machine learning.

Both the encoder and decoder networks have multiple layers. For these layers we use the ReLU [2] activation function as this has superior training performance for deep learning models. It is able to better maintain the gradient of the network when modifying the weights deeper in the network. No activation function is added before the central layer to allow for more freedom in the low-dimensional representations. The optimizer used is Adam [3] a modern gradient descent algorithm with the addition of momentum to overcome local minima. It has been found that this method converges faster than other popular optimizers such as RMSprop, AdaGrad and AdaDelta. The data is divided in batches for training. An Early Stopping [4] measure was added to prevent the network from overfitting and to improve execution times where additional training would not have resulted in significant improvements.

B. DC-Kmeans

Autoencoders only optimize to have the output of the network match the input of the network. This alone does not generate low-dimensional codes that present a nice visualization of the data. To solve this Tian and Song [5, 6] independently created similar solutions that add a clustering objective function to the standard loss function of an autoencoder network We use the notation DC-KMeans, short for DeepCluster Kmeans, proposed by Tian.

We can implement this clustering by adding a regulariser on the central layer of the network that will minimize this extra objective function. This regulariser minimizes the following function:

$$\begin{aligned} \min : \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i||^2 + \lambda * ||\boldsymbol{y}_i - \boldsymbol{c}_i^*||^2 \\ s.t. \qquad \boldsymbol{y}_i = f^t(\boldsymbol{x_i}) \; i = 1, \dots, N \end{aligned} \tag{1}$$

$$\boldsymbol{c}_i^* = \text{argmin}_{c_j^{t-1}} ||y_i - c_j^{t-1}||^2, j = 1, \dots, k \tag{2}$$

where $f^t(\cdot)$ is the encoder function of the network at the t^{th} iteration, c_j^{j-1} is the j^{th} cluster center computed at the $(t-1)^{th}$ iteration and c_i^* is the closest cluster center of the i^{th} sample in the central layer. At each optimization step first the network is updated using the cluster centers computed in the previous iteration, afterwards the cluster centers are updated using the output of the changed network:

$$c_j^t = \frac{\sum_{x_i \in C_j^{t-1}} f^t(x_i)}{|C_j^{t-1}|} \tag{3}$$

where C_j^{t-1} is the set of points belonging to cluster c_j at iteration $t-1$. At the start of training, cluster centers are initialized randomly.

III. Evaluation

In their paper on the quality assessment of dimensionality reduction Lee and Verleysen [7] propose a unifying framework for quality measures for dimensionality reduction based on distance ranking and k-ary neighborhoods This framework is based on the co-ranking matrix, which can be used to calculate a few metrics.

A. Co-ranking matrix

To construct the co-ranking matrix we first need to calculate the pairwise ranking for all points in both the high-dimensional space and the low-dimensional space. This is done by calculating the distances for each point to all other points and ordering them by distance. The rank of point j in relation to point i is then the position in this ordering, noted p_{ij} for the high-dimensional space and r_{ij} for the low-dimensional space. The co-ranking matrix can then be defined as:

$$\begin{aligned} &\mathbf{Q} = [q_{mn}]_{1\le m,n\le N-1} \\ \text{with}\quad &q_{mn} = |\{(i,j) : p_{ij} = m \wedge r_{ij} = n\}| \end{aligned} \tag{4}$$

Each of the co-ranking metrics is calculated over a k-ary neighborhood, using k we can divide the co-ranking matrix into regions (Fig. 1). Using these regions we can calculate some rank-based criteria.

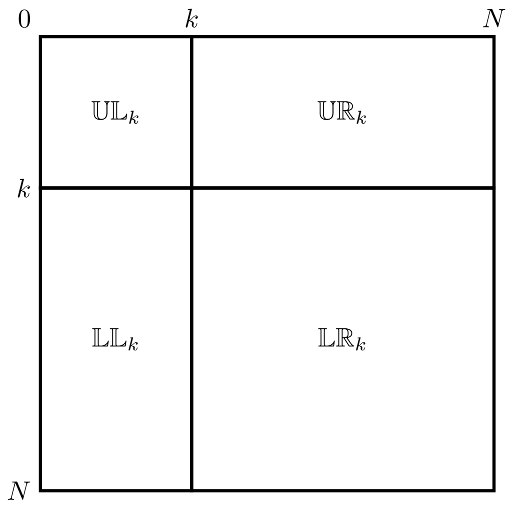

Fig. 1. Regions in the co-ranking matrix. Depending on the value of k, the co-ranking matrix can be divided into four regions.

B. Trustworthiness and continuity

Trustworthiness en continuity [8] are related metrics that look at the points in that are not present in both the low-dimensional and high-dimensional k-ary neighborhood. Faraway points that become neighbors decrease trustworthiness and neighbors that are embedded faraway from each other decrease the continuity. The formulas for both measures are as follows:

$$T(k) = 1 - \frac{2}{Nk(2N-3k-1)} \sum_{(m,n)\in\mathbb{LL}_k} (m-k)q_{mn} \tag{5}$$

$$C(k) = 1 - \frac{2}{Nk(2N-3k-1)} \sum_{(m,n)\in\mathbb{UR}_k} (s-k)q_{mn} \tag{6}$$

C. Mean Relative Rank Errors

The Mean Relative Rank Errors (MRRE) [9] are similar but cover different parts of the co-ranking matrix:

$$E_n(k) = \frac{1}{H_k} \sum_{(m,n)\in\mathbb{UL}_k\cup\mathbb{LL}_k} \frac{|m-n|}{n} q_{mn} \tag{7}$$

$$E_v(k) = \frac{1}{H_k} \sum_{(m,n)\in\mathbb{UL}_k\cup\mathbb{UR}_k} \frac{|m-n|}{m} q_{mn} \tag{8}$$

with $H_k = N \sum_{k=1}^{K} \frac{|N-2k+1|}{k}$.

If we mirror the co-ranking matrix over the anti-diagonal we can calculate the same metrics but over the neighborhood of the k furthest neighbors. This gives a measure of how well the global structure of the data is kept in the reduction.

IV. Results

We compare our autoencoder models with UMAP [10], a recent non parametric dimensionality reduction method with arguably better visualizations than t-SNE [11]. For our comparison we use 3 datasets:
MNIST A dataset of 28x28 pixel grayscale images of handwritten digits. There are 10 digit classes (0 through 9) and 70000 total images. This is treated as 70000 different 784 dimensional vectors.
Flow cytometry The flow cytometry datasets were provided by the VIB-UGent Center for Inflammation Research. They contain samples from 10 mice, 5 wild type and A20 fl x NKp46 iCre. The samples were stained with 11 different markers and were accompanied with a manual labeling. From this dataset 50 000 elements were sampled.
Mass cytometry The VIB-UGent Center for Inflammation Research also provided a mass cytometry dataset. This dataset contains a sample where IFNa & LPS is stimulated. The samples were stained with 33 different markers and were accompanied with a manual labeling. From this dataset 50 000 elements were sampled.

Each of these datasets got visualized by UMAP, the standard autoencoder and DC-Kmeans. The UMAP parameters are kept at the defaults, with the exception of the number of neighbors which is set to 30. The autoencoders are built with three hidden layers in both the encoder and decoder part, their sizes are 1024 – 512 – 256. The learning rate is set to 0.001 and batch size to 100.

A. Visualization

In Fig. 2 we see the visualizations of the different datasets for UMAP and both autoencoder models. UMAP forms well separated clusters of data in most cases, in the case of MNIST some groups are connected, they correspond to the digit sets $\{4, 7, 9\}$ and $\{3, 5, 8\}$, which are visually quite similar. The standard autoencoder tends to form a star pattern around zero as values inside the network generally stay close to zero because of the activation functions. The different groups of digits are not so well separated, because of the coloring we do see that the groups are not overlapping much. Using the extra clustering objective we see the formation of more uniform clusters. The separation between clusters is again less than UMAP but we do see that there are regions with lower density separating the groups of the datasets which is enough if a cluster algorithm like DBSCAN [12] is used.

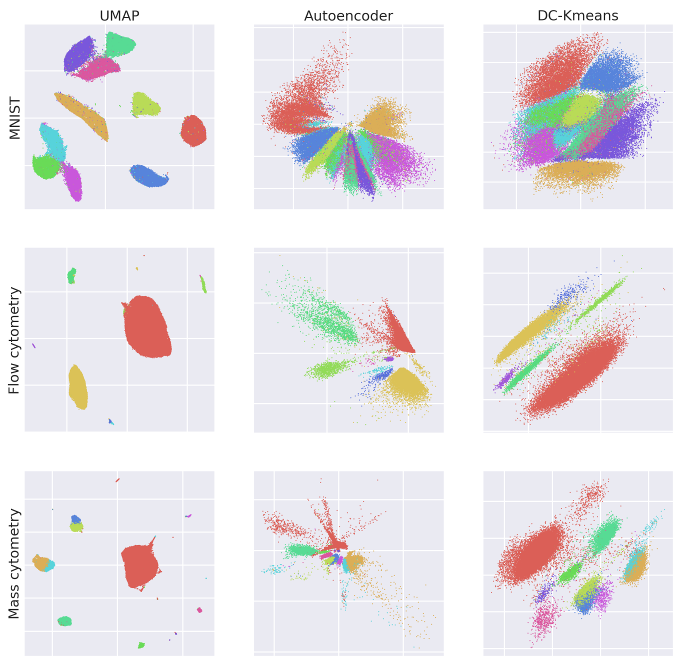

Fig. 2. Visual comparison of UMAP and autoencoder embeddings for a number of real world datasets. Points are colored using the class information of the datasets.

B. Quality measures

In Fig 3 the results for the quality measures for each model and dataset are given. The measures are calculated based on a neighborhood of 30 elements, the metrics based on the mirrored matrix are prefixed with ‘i’.

It is clear to see that UMAP preserves the local structure better but both autoencoders are not far behind. With the clustering objective we do see that trustworthiness and E_n can be lower. This means that some points that are faraway from each other in the high-dimensional space are in each others neighborhood in the visualization.

When we look at the preservation of neighborhoods of furthest points we see that autoencoders score better than UMAP. We can conclude that autoencoders are more capable in keeping the global structure of the dataset intact.

C. Execution times

In table I we see the execution times for the different models and datasets. We see that the amount of dimensions do not have

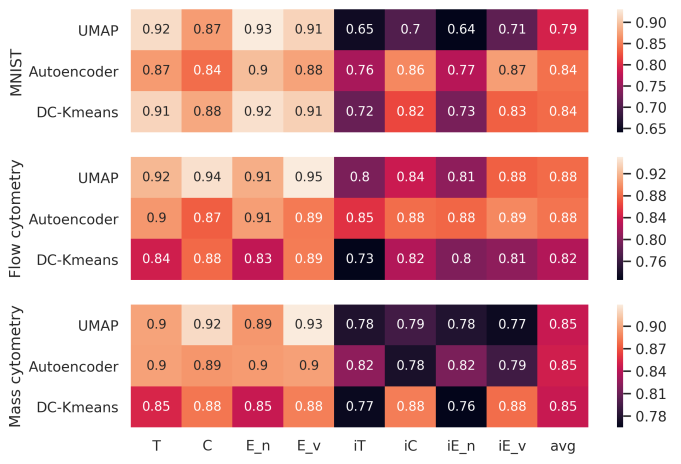

Fig. 3. Quality measures of UMAP and autoencoder embeddings for a number of real world datasets. A higher score means a better result for the measure.

that much of an effect for the execution times for UMAP. This is due to the algorithm, that is used for determining the nearest neighbors of each point, not being dependent on the amount of dimensions. In the case of the autoencoder models we do see that time increases linearly with the amount of dimensions. This is to be expected as the trainable parameters of the networks increase linearly with the amount of dimensions of the input data. We also see that the addition of the extra clustering objective made the DC-Kmeans model a lot slower. The time complexity increases quadratically with the amount of elements to be trained instead of linearly like the standard autoencoder.

TABLE I
EXECUTION TIMES FOR UMAP AND AUTOENCODER MODELS FOR A NUMBER OF REAL WORLD DATASETS. TIMES ARE IN MM:SS FORMAT.

	UMAP	Autoencoder	DC-Kmeans
MNIST	01:36	03:29	28:12
Flow cytometry	01:01	00:52	08:58
Mass cytometry	01:13	00:53	09:43

V. CONCLUSION

Autoencoders can generate a low-dimensional representation that captures the local and global structure of a high-dimensional dataset. Visually this representation is quite poor. Additional objective functions can improve this but this comes at a cost in time complexity. We can conclude that maintaining the local structure is more important than the global structure to get a nice visual result. This means that distorting the global structure is sometimes necessary to for a clear visualization.

The parametric property of autoencoders can prove very useful as previously trained networks can be used to quickly visualize other datasets of the same type of data. This is not possible with non-parametric methods such as t-SNE and UMAP.

References

[1] G E Hinton and R R Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.

[2] Vinod Nair and Geoffrey E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, USA, 2010, ICML’10, pp. 807–814, Omnipress.

[3] Diederik P. Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2015.

[4] N. Morgan and H. Bourlard, “Generalization and parameter estimation in feedforward nets: Some experiments,” in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed., pp. 630–637. Morgan-Kaufmann, 1990.

[5] Kai Tian, Shuigeng Zhou, and Jihong Guan, “Deepcluster: A general clustering framework based on deep learning,” in *Machine Learning and Knowledge Discovery in Databases*, Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, Eds., Cham, 2017, pp. 809–825, Springer International Publishing.

[6] Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan, “Auto-encoder based data clustering,” in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, José Ruiz-Shulcloper and Gabriella Sanniti di Baja, Eds., Berlin, Heidelberg, 2013, pp. 117–124, Springer Berlin Heidelberg.

[7] John Lee and Michel Verleysen, “Quality assessment of nonlinear dimensionality reduction based on k-ary neighborhoods,” in *Proceedings of the Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery at ECML/PKDD 2008*, Yvan Saeys, Huan Liu, Iñaki Inza, Louis Wehenkel, and Yves Van de Pee, Eds., Antwerp, Belgium, 15 Sep 2008, vol. 4 of *Proceedings of Machine Learning Research*, pp. 21–35, PMLR.

[8] Jarkko Venna and Samuel Kaski, “Neighborhood preservation in nonlinear projection methods: An experimental study,” 09 2001, vol. 2130.

[9] John A. Lee and Michel Verleysen, *Nonlinear Dimensionality Reduction*, Springer Publishing Company, Incorporated, 1st edition, 2007.

[10] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger, “Umap: Uniform manifold approximation and projection,” *The Journal of Open Source Software*, vol. 3, no. 29, pp. 861, 2018.

[11] Laurens van der Maaten and Geoffrey Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, “A density-based algorithm for discovering clusters 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*. 1996, KDD’96, pp. 226–231, AAAI Press.

[13] Yvan Saeys, Sofie Van Gassen, and Bart N. Lambrecht, “Computational flow cytometry: helping to make sense of high-dimensional immunology data,” *Nature Reviews Immunology*, vol. 16, pp. 449, June 2016.