Generalizing self-organizing maps: large-scale training of GMMs and applications in data science

Alexander Gepperth[1][0000−0003−2216−7808]

Fulda University of Applied Sciences, Leipzigerstr. 123, 36037 Fulda, Germany
alexander.gepperth@cs.hs-fulda.de
www.gepperth.net/alexander

Abstract. This contribution shows that Gaussian Mixture Models can be considered generalizations of self-organizing maps. More precisely, we demonstrate that the training of self-organizing maps is an approximation to the training of Gaussian Mixture Models by gradient descent. As a consequence, the scores of a trained SOM can be treated as log-likelihoods of a GMM with tied, spherical covariance and used, e.g., for outlier detection, whereas sampling from trained SOMs is not well-defined. Furthermore, we outline how SGD-trained GMMs can be generalized to diagonal and more expressive covariance matrices and how this benefits typical data science applications such as outlier detection, sampling and generative classification.

Keywords: Self-Organizing Maps · Gaussian Mixture Models · Stochastic Gradient Descent

1 Introduction

The self-organizing map (SOM) model [5] has widely been used in data visualization and data exploration, and has been very influential in general. SOM training is however complicated by the fact that there is no loss function that is minimized by SOM training, and that the decay of the neighbourhood radius over time is tricky to tune. In addition, is it theoretically unclear how SOM scores, i.e., input-prototype distances, can be interpreted although they have been used for outlier detection on an ad-hoc basis.

This article proposes the Gaussian Mixture Model (GMM) as a "generalized self-organizing map", i.e., a probabilistic model that reduces to SOM in a suitable approximation. By virtue of this approximation, one can harness the beneficial properties of SOMs while at the same time providing SOM with a clear probabilistic interpretation, leading to applications in outlier detection and sampling with SOM-like models.

1.1 Related work

SOMs and GMMs have multiple common points, the most notable being that they are based on *distances* rather than scalar products as DNNs are, which is

why some authors include them in the group of *prototype-based models*. There are a few works that try to link the GMM and SOM models in the past. Notably, in [3], the authors propose a modified SOM model that is based on the minimization of a differentiable loss. This was new at the time, since it had been shown that the SOM learning rule cannot be derived from the minimization of *any* single differentiable loss function, see [4]. On the other hand, is was demonstrated in [9] that SOM-like topological visualization behavior can be imposed on GMMs if the common uniform prior over latent variables is replaced appropriately. However, gradient-based training of GMMs is not discussed in this article, and neither is a formal link between SOMs and GMMs. The possibilities of advanced GMMs such as the MFA model for image modeling were explored in [8], using SGD for using k-means initialization.

1.2 Contributions

This article makes the following contributions:

- mathematical proof that the SOM learning rule is an approximation to the training of GMMs by stochastic gradient descent
- provide a probabilistic analysis of SOM's ability to sample and perform outlier detection
- demonstrate the capabilities of SGD-trained GMMs as "generalized SOMs" w.r.t. sampling, outlier detection and generative classification

2 Theoretical contributions

2.1 Relation between GMM and SOM training

Gaussian Mixture Models The GMM model approximates an unknown distribution, given empirically by data samples $X = \{\boldsymbol{x}_n\}$, by expressing that distribution by a weighted superposition of K multivariate Gaussian densities $\mathcal{N}_k(\boldsymbol{x}) \equiv \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$: $p(X) = \prod_n \left(\sum_k \pi_k \mathcal{N}_k(\boldsymbol{x}_n)\right)$. When passing to log-probabilities, this transforms to $\mathcal{L}_0 = \log p(X) = \sum_n \log \sum_k \pi_k \mathcal{N}_k(\boldsymbol{x}_n)$, which is often termed the GMM log-likelihood. GMMs are trained by adapting the *weights* π_k, the *centroids* $\boldsymbol{\mu}_k$ and the covariance matrices $\boldsymbol{\Sigma}_k$ such that the negative log-likelihood is maximized.

GMM optimization by SGD Traditionally, GMMs are trained by the Expectation-Maximization (EM) algorithm which does not require learning rates and has good convergence properties. However, since EM is inherently a batch-type algorithm, i.e., it needs to process all samples at once to retain its convergence properties, it is unfeasible for large-scale training. Furthermore, convergence in practice is only ensured if centroids are initialized by, e.g., k-means. This is why recent publications [2] propose to optimize GMMs by SGD from random initial

conditions. In order for this, two modifications need to be made: first of all, the log sum inside the GMM log-likelihood is approximated by its largest term:

$$\mathcal{L}_1(X) = \mathcal{L}_0(X) = \sum_n \log \sum_k \pi_k \mathcal{N}_k(\boldsymbol{x}_n) \approx \sum_n \log \max_k \pi_k \mathcal{N}_k(\boldsymbol{x}_n) = \quad (1)$$
$$= \sum_n \max_k \log\left(\pi_k \mathcal{N}_k(\boldsymbol{x}_n)\right)$$

This is very numerically advantageous as the log removes the exponential inside the Gaussian density $\mathcal{N}$, but it also makes the gradients much more tractable. When optimizing this loss by SGD from random initial conditions, we often encounter *degenerate solutions* where one component describes all the data, and the remaining ones remain uninitialized. To remedy this, [2] proposes to add an annealing mechanism to the loss of eq. (1) which averages components over a Gaussian neighbourhood, thus obtaining the final GMM loss

$$\mathcal{L}_2(X) = \mathcal{L}_1(X) = \sum_n \max_k \log\left(\pi_k \mathcal{N}_k(\boldsymbol{x}_n)\right) \approx$$
$$\approx \sum_n \max_k \left[\sum_l g_{kl}(\sigma) \log\left(\pi_l \mathcal{N}_l(\boldsymbol{x}_n)\right)\right] \equiv \mathcal{L} \quad (2)$$

The kernel matrix g_{ij} follows a Gaussian profile with a time-dependent width $\sigma(t) \equiv \sigma$: $g_{kl} = \exp(-0.5 d(k,l)^2 \sigma^{-2})$, with $d(i,j)$ representing the Euclidean distance between components i and j when placed on a two-dimensional grid. It is easy to see that $g_{kl} \to \delta_{kl}$ in the limit $\sigma \to 0$, in which case the annealed loss $\mathcal{L}_2$ reduces to $\mathcal{L}_1$. Optimizing a GMM by SGD is therefore performed by maximizing $\mathcal{L}_2(X)$ w.r.t $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$ while letting $\sigma(t)$ decay from some initial value σ_0 to 0. This ensures that local optima are avoided by the optimization, as demonstrated and proven in [2]. An additional reparameterization ensures that weights are normalized: $\sum_k \pi_k = 1$.

From GMM to SOM In the GMM model, the covariance matrices $\boldsymbol{\Sigma}_k$ can have any form. Most commonly, they are chosen as spherical ($\boldsymbol{\Sigma}_k = \sigma_k \boldsymbol{I}$) or diagonal matrices ($\boldsymbol{\Sigma}_k = \mathrm{diag}[\sigma_{k1}, \sigma_{k2}, dots]$)) for simplicity. If we assume that all covariance matrices have a spherical, tied form of unit variance as $\boldsymbol{\Sigma}_k = I$, the covariance matrices effectively drop out of the GMM loss $\mathcal{L}_2$. The Gaussian component densities $\mathcal{N}_k$ thus simplify as

$$\mathcal{N}_k \equiv \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = (2\pi \det \boldsymbol{\Sigma}_k)^{-0.5} \exp(-0.5\left((\boldsymbol{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)\right)$$
$$\to \sqrt{2\pi} \exp(-0.5 ||\boldsymbol{x} - \boldsymbol{\mu}_k||^2). \quad (3)$$

If we furthermore assume that the weights are all tied to K^{-1}, they drop out of the loss as well, which finally becomes:

$$\begin{aligned}\mathcal{L}_2 &= \sum_n \max_k \left[\sum_l g_{kl}(\sigma) \log\left(\pi_l \mathcal{N}_l(\boldsymbol{x}_n)\right) \right] \\ &\rightarrow \sum_n \max_k \left[\sum_l g_{kl}(\sigma)(-0.5\log(2\pi) - 0.5||\boldsymbol{x}_n - \boldsymbol{\mu}_k||^2) \right] = \\ &= -0.5 \sum_n \max_k \left[\sum_l g_{kl}(\sigma)(||\boldsymbol{x}_n - \boldsymbol{\mu}_k||^2) \right] + \text{const.} \equiv \mathcal{L}_3 \end{aligned} \tag{4}$$

Discarding the constant and the leading factor which are irrelevant for gradient descent, we arrive at what was termed the "energy function" for the SOM model proposed by [3]. When optimizing $\mathcal{L}_3$ by stochastic gradient descent, we obtain the update rule

$$\boldsymbol{\mu}_l \leftarrow \boldsymbol{\mu}_{k^*} + \epsilon g_{k^* l}(\sigma)(\boldsymbol{\mu}_{k^*} - \boldsymbol{x}), \tag{5}$$

which of course has to be accompanied by an appropriate decrease regime for $\sigma = \sigma(t)$. The only difference to the conventional online SOM learning rule is that the best-matching unit (BMU) k^* is determined here as $k^* = \arg\max_k \left[\sum_l g_{kl}(\sigma)(||\boldsymbol{x}_n - \boldsymbol{\mu}_k||^2)\right]$, instead of $k^* = \arg\max_k ||\boldsymbol{x}_n - \boldsymbol{\mu}_k||$. In the limit of $\sigma \rightarrow 0$, these two formulations become identical. For all practical purposes, the modified BMU selection leads to the same results, as shown in [4].

2.2 Analysis of SOM capabilities

With the knowledge that SOMs are a particular limiting case of GMMs, we can derive the following statements w.r.t. particular SOM capabilities:

SOMs can perform limited outlier detection SOM scores $s_k \equiv -||vx - \boldsymbol{\mu}_k||^2$ correspond to GMM log-probabilities, for GMMs with tied weights and spherical unit covariance, see above. We can therefore compute the posterior probability that a given sample was produced by a component c as $p(\boldsymbol{x}|c) = \frac{\exp(s_c)}{\sum_l \exp(s_l)}$ and impose a threshold for the BMU probability. Alternatively, we can apply an application-derived threshold directly on the SOM scores to detect outliers. This is of course only possible under the assumption that data clusters do have approximately unit variance.

SOMs cannot be used for sampling For outlier detection, it is immaterial whether the unit variances fit the actual data well, since we are just want to rank SOM's components w.r.t. their degree of match to the data. For sampling, this is no longer true, since we need the variances to draw samples from a multivariate Gaussian distribution. By assuming unit variances for all components, all clusters in sampled data will therefore have unit radius regardless of their actual extent in data space. Thus, SOMs are not suitable for faithful sampling.

2.3 Advantages of GMMs w.r.t SOMs

GMMs do not share the limitations of SOMs outlined in section 2.2, and if they are trained by SGD, they are by construction capable of outlier detection and sampling. However, they do retain many important SOM properties, while at the same time fixing a serious issue in SOM training:

Automatic tuning of neighbourhood radius The correct decay regime for $\sigma(t)$ is always hard to tune for SOMs because there is no real criterion of SOM performance (since there is not loss function to be minimized). In contrast, SGD-trained GMMs provide a loss function and thus a simple criterion for reducing $\sigma(t)$: essentially, this should happen whenever the loss is stationary. It can be shown (see [2]) that σ defines a lower bound for the GMM loss $\mathcal{L}_2$, so reducing σ can allow the loss to decrease further. Performing $\sigma \to \gamma\sigma$ in case of stationary loss during training provides a viable strategy that does not require manual tuning at all. The only values to set are $\sigma_0 \equiv \sigma(t=0)$ and σ_∞, both of which can be set based on heuristics.

Data visualization property Since the loss $\mathcal{L}_2$ rewards configurations where similar centroids lie close together on the two-dimensional lattice, SGD-trained GMMs that optimize $\mathcal{L}_2$ retain the topological SOM property and can thus be used for data visualization and exploration as a drop-in replacement for SOMs.

Arbitrary covariance matrices One particular feature of GMMs is the freedom to choose a particular form for the covariance matrices. In practice, diagonal CMs are very common especially for high-dimensional data, since GMM training involves matrix inversions, which makes full covariance matrices too expensive in terms of memory and computation time. An interpolation between these two choices for a d-dimensional problem is given by the Mixture of Factor Analyzers (MFA) model which decomposes the CM as $\boldsymbol{\Sigma} = \boldsymbol{D} + \Gamma\Gamma^T$, where $\boldsymbol{D} = \mathrm{diag}(\sigma_1, \ldots, \sigma_d)$ and $\Gamma \in \mathbb{R}^{l \times d}$. The parameter $l \in [1, d]$ determines the complexity of the resulting CM. Just like GMMs, MFA instances can be trained by SGD, see [1].

3 Experiments

In all experiments of this section, we use GMM and SOMs with $K = 64$ components and periodic boundary conditions on a rectangular lattice. The initial neighbourhood radius for SOM and GMM experiments is set to a quarter of the diagonal of the lattice, i.e., $\sigma_0 = \sqrt{128}/4 \approx 3$. The asymptotic neighbourhood radius is set to $\sigma_\infty = 0.01$. For SOM experiments, the decay of the neighbourhood radius is performed as $\sigma = \sigma_0 \exp(-e)$, where $e > 1$ denotes the current epoch. For $E = 1$, the neighbourhood radius is kept at σ_0. For GMM training by SGD, the automatic annealing mechanism is used for controlling the "neighbourhood radius" $\sigma(t)$, with default parameters as given in [2].

We use the following image datasets for our experiments. All of them are relatively high-dimensional and well out of reach of conventional EM-trained GMMs.

MNIST [6] consists of 60.000 28×28 grayscale images of handwritten digits (0-9).
Fashion-MNIST [10] consists of 60.000 images of clothes in 10 categories and is structured like MNIST.
Fruits-360 [7] contains 100x100 images showing different types of fruits, from which we chose the 10 best-represented classes and downsample to 48x48 RGB.

3.1 Data visualization with SOMs, energy-based SOMs and GMMs

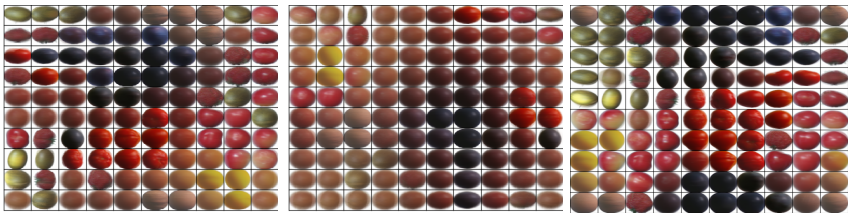

Fig. 1. Centroids obtained after training a SOM (left), energy-based SOM (middle) and GMM (right), arranged on a 10x10 grid, after training on the Fruits-360 dataset. We observe a similar topologival organization in all three cases.

This is a preliminary experiment meant to show that the data visualization property of SOMs actually carries over to GMMs, in this case with a diagonal covariance matrix. The $K = 100$ GMM centroids, obtained after 100 training epochs on the Fruits-360 dataset, are visualized in fig. 1, and we observe that they show the SOM-like topological organization where similar prototypes/centroids develop next to each other during training.

3.2 Conditional and unconditional sampling from GMMs

dataset	Fruits-360	MNIST	FashionMNIST
GMM	98.6	89.3	85.2
MFA, $l = 4$	98.6	89.9	88.8
MFA, $l = 25$	99.0	91.7	90.9

Table 1. Classification accuracies for linear classification based on MFA and GMM responsibilities on various datasets. We always use $K = 64$ components for GMM and MFA and use two values for the MFA latent dimension l. All accuracies are measured on the test sets of the respective datasets.

In this section, we demonstrate sampling from GMMs with diagonal covariance matrices, which is performed by assuming an uniform prior over components and drawing components from it. In addition, we show that coupling the

responsibilities of a GMM to a simple linear classifier allows for the conditional sampling of given classes. The logic behind this approach is simple: the normalized GMM/MFA responsibilities (or posterior probabilities) $\gamma_k(\boldsymbol{x}) \equiv \frac{p_k(\boldsymbol{x})}{\sum_i p_i(\boldsymbol{x})}$ are transformed as $\boldsymbol{y} = \boldsymbol{S}(W\boldsymbol{\gamma}(\boldsymbol{x}) + \boldsymbol{b})$, where $\boldsymbol{S}(\boldsymbol{y})$ denotes the softmax function. After being trained by cross-entropy, this linear classifier can be approximately inverted as $\hat{\boldsymbol{\gamma}} = W^T(\boldsymbol{c} - \boldsymbol{b})$ when fed with a one-hot encoded target vector to produce a distribution (generally non-uniform in nature) over components, from which one can draw one for sampling. As we can see in table 1, the obtained accuracies for the linear classifier are not excessive, since GMM responsibilities are not optimized for discrimination. Figure 2 shows both unconditional and conditional sampling results for GMMs trained for 100 epochs on MNIST, FashionMNIST and Fruits-360. Conditional sampling is used to generated samples of class 5, and we can visually confirm that virtually all produced samples belong to this class. All reported results are accuracies for single runs, measured on the test test of the respective benchmarks.

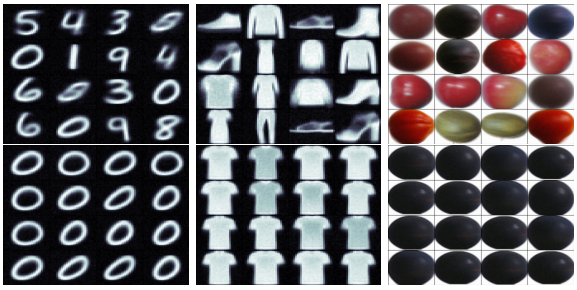

Fig. 2. Samples generated by GMMs with diagonal covariance matrix after training on MNIST(left column), FashionMNIST(middle column) and Fruits-360(right column). In the upper row, sampling is unconditional, whereas we conditionally sample from class 5 in the lower row.

3.3 Sampling with MFA instances

Since MFA instances can be considered to be GMMs whose covariance matrix interpolates between diagonal and full form, we can expect more diverse sampling results from MFA instances than from GMMs with diagonal covariance matrices. This is significant since, for high-dimensional data, full covariance matrices would require excessive memory.

We train an MFA model with a latent dimension of $l = 4$ on all datasets described in section 3 for 300 epochs using $K = 100$ components. The samples generated by selected components of this model are shown in fig. 3. To show that the variability is caused by the latent factors and not by different components being selected for sampling, we always draw samples from a single component.

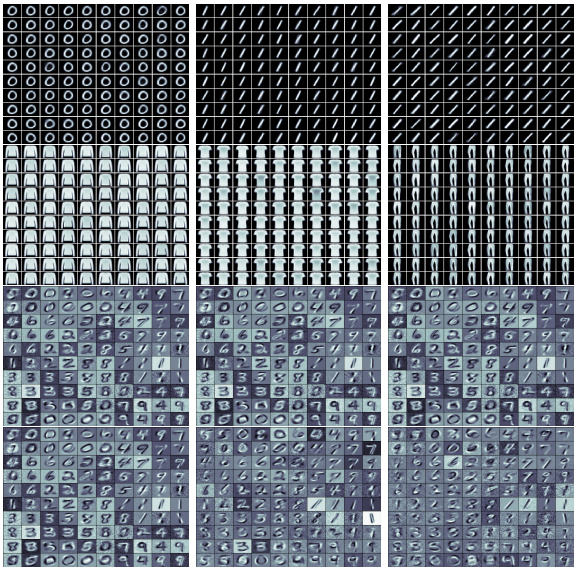

Fig. 3. Upper two rows: Sampling results for MFA on MNIST(first row) and FashionMNIST(second row). Each image shows 100 samples produced from the same component. Lower two rows: the first three latent directions for MFA on MNIST (3^{rd}row) and FashionMNIST(4^{th} row).

We indeed observe that samples are quite diverse in terms of slant and stroke even when coming only from one component. This is especially true when comparing MFA-generated samples to those obtained from diagonal-CM GMMs (see previous section).

The latent directions are shown in fig. 3. Latent directions are the columns of the matrices Γ_k, corresponding to (generalized, since the Γ_k are not full rank) principal directions. Latent direction i indicates the variations of sampled data that occur if we move along one axis i in the latent space and are useful for visualizing the sample variability captured by the MFA model.

3.4 Outlier detection experiments with MFA

For these experiments, we train a single MFA on class 5 of the MNIST and FashionMNIST datasets (Fruits was omitted since it is a very easy classification benchmark), and impose a threshold on the log-probability of the best-matching component in order for a given sample to be classified as an inlier. By varying this threshold, we obtain a receiver-operator characteristic (ROC) that indicates the quality of the outlier detector. As we can see in fig. 4, outlier detection is clearly something that GMMs can perform well. We observe that increasing the number of latent factors increase the quality of outlier detection. Increasing the number of components(not shown) has a similar effect which is however not as pronounced.

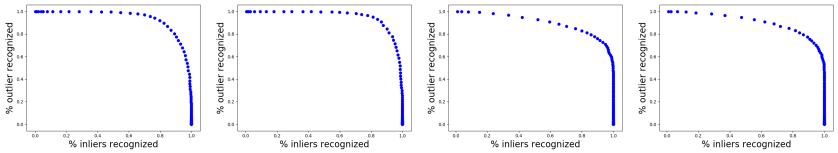

Fig. 4. Outlier detection quality. Shown, from left to right, are ROCs for MNIST(4 latent factors), MNIST(25 latent factors), FashionMNIST(4 latent factors) and FashionMNIST(25 latent factors). Inlier class is always class 5. The area under the curve is a performance indicator and follows a strict the-larger-the-better principle.

3.5 MFA: generative classification

Generative classification is in essence outlier detection with multiple outlier detection models $\mathcal{M}_i$, each model being trained on class i only. For testing, each model produces a score for the test sample, and the model with the highest score defines the predicted class for that sample: $p(c|\boldsymbol{x}) = \text{argmax}_i \mathcal{M}_i(\boldsymbol{x})$. If the outlier detection models are of sufficient quality, this scheme is not more computationally demanding than training a conventional DNN classifier, because although there are now several models, each model only gets trained on a small part of the data. We again found that this works best with MFA and a rather high latent dimensionality, e.g., $l = 25$. For reference, we also train conventional GMMs for this problems and compare the results (see table 2. All reported results are accuracies for single runs, measured on the test test of the respective benchmarks. It is important to note that generative classification is actually supervised learning: although the labels are not used while training the individual outlier detectors, they *are* used for assigning data samples *to* outlier detectors for training.

dataset	Fruits-360	MNIST	FashionMNIST
GMM	91.0	85.3	81.2
MFA, $l = 4$	92.6	85.9	83.3
MFA, $l = 25$	98.5	95.7	92.3

Table 2. Results for generative classification using MFA and GMM outlier detection models on various datasets.

4 Discussion and outlook

In this article, we argue that SGD-trained Gaussian Mixture Models can be used as a drop-in replacement for SOMs in data visualization tasks that SOMS are typically used for. In particular, training GMMs by SGD enables them to process large amount of high-dimensional data which is required for data visualization

and other typical data science tasks. We showed that SOMs can be considered an approximation to GMMs in a particular, simplifying case, and that SOMs do have a probabilistic interpretation. However, we argue that GMMs are much better suited than SOMs for performing typical data science functions such as outlier detection, sampling and generative classification because they offer a much greater flexibility in terms of, e.g., parameterizing the covariance matrix.

References

1. Gepperth, A.: Large-scale gradient-based training of mixture of factor analyzers. In: International Joint Conference on Neural Networks(IJCNN) (2022)
2. Gepperth, A., Pfülb, B.: Gradient-based training of gaussian mixture models for high-dimensional streaming data. Neural Processing Letters (2021), accepted
3. Heskes, T.: Energy functions for self-organizing maps. In: Kohonen maps, pp. 303–315. Elsevier (1999)
4. Heskes, T.: Self-organizing maps, vector quantization, and mixture modeling. IEEE transactions on neural networks **12**(6), 1299–1305 (2001)
5. Kohonen, T.: The self-organizing map. Proceedings of the IEEE **78**(9), 1464–1480 (1990)
6. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
7. Mureşan, H., Oltean, M.: Fruit recognition from images using deep learning. Acta Universitatis Sapientiae, Informatica **10**(1), 26–42 (2018). https://doi.org/10.2478/ausi-2018-0002, http://arxiv.org/abs/1712.00580
8. Richardson, E., Weiss, Y.: On GANs and GMMs. Advances in Neural Information Processing Systems **2018-December**(NeurIPS), 5847–5858 (2018)
9. Verbeek, J.J., Vlassis, N., Kröse, B.J.: Self-organizing mixture models. Neurocomputing **63**, 99–123 (2005)
10. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms pp. 1–6 (2017), http://arxiv.org/abs/1708.07747