Bioinformatics, 36(20), 2020, 5045–5053
doi: 10.1093/bioinformatics/btaa623
Advance Access Publication Date: 9 July 2020
Original Paper

Gene expression

Exploring generative deep learning for omics data using log-linear models

Moritz Hess *, Maren Hackenberg and Harald Binder

Institute of Medical Biometry and Statistics, Faculty of Medicine and Medical Center – University of Freiburg, 79104 Freiburg, Germany

*To whom correspondence should be addressed.

Associate Editor: Pier Luigi Martelli

Received on October 6, 2019; revised on June 28, 2020; editorial decision on June 29, 2020; accepted on July 2, 2020

Abstract

Motivation: Following many successful applications to image data, deep learning is now also increasingly considered for omics data. In particular, generative deep learning not only provides competitive prediction performance, but also allows for uncovering structure by generating synthetic samples. However, exploration and visualization is not as straightforward as with image applications.

Results: We demonstrate how log-linear models, fitted to the generated, synthetic data can be used to extract patterns from omics data, learned by deep generative techniques. Specifically, interactions between latent representations learned by the approaches and generated synthetic data are used to determine sets of joint patterns. Distances of patterns with respect to the distribution of latent representations are then visualized in low-dimensional coordinate systems, e.g. for monitoring training progress. This is illustrated with simulated data and subsequently with cortical single-cell gene expression data. Using different kinds of deep generative techniques, specifically variational autoencoders and deep Boltzmann machines, the proposed approach highlights how the techniques uncover underlying structure. It facilitates the real-world use of such generative deep learning techniques to gain biological insights from omics data.

Availability and implementation: The code for the approach as well as an accompanying Jupyter notebook, which illustrates the application of our approach, is available via the GitHub repository: https://github.com/ssehztirom/Exploring-generative-deep-learning-for-omics-data-by-using-log-linear-models.

Contact: hess@imbi.uni-freiburg.de

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

High-dimensional omics data, like gene expression data, are now collected in large amounts under controlled conditions, e.g. single-cell data from mouse models, or derived from patients in clinical settings. Frequently, the goal is to better understand the heterogeneity of the data in terms of subpopulations that differ in the activity of molecular networks. In contrast to standard supervised approaches such as regularized regression or random forests, generative deep learning approaches, building e.g. on variational autoencoders (VAEs) (Kingma and Welling, 2013; Rezende *et al.*, 2014) learn a usually low-dimensional latent representation of the observed variables in an unsupervised manner. This latent representation can then be visually inspected and be used to assess the similarity of potential subpopulations. In fact, the performance of generative deep approaches in learning latent representations that allow for better capturing the underlying structure has been demonstrated in various studies (Ding *et al.*, 2018; Eraslan *et al.*, 2019; Lopez *et al.*, 2018).

Since these generative approaches learn the joint distribution of observed and latent variables, they allow synthetic observations to be generated with a similar structure as observed in the empirical data. For this reason, deep generative models are very useful, e.g. imputing missing values, which has been demonstrated for gene expression by Wang *et al.* (2018).

However, while deep learning approaches allow for the uncovering of hidden patterns in opaque and complex data, the models are themselves complex and opaque, which does not easily allow researchers to infer how learned latent representations relate to the observed variables. This is considered a large challenge which must be solved to unleash the full potential of deep learning to better understand biological networks (Camacho *et al.*, 2018).

For applications with image data, several approaches have been proposed to investigate how observed variables relate to latent representations learned by a deep architecture (Montavon *et al.*, 2018). Recently, approaches which combine deep learning with causal inference (Harradon *et al.*, 2018) also in combination with generative models (Besserve *et al.*, 2018) have been proposed.

Still, there are several difficulties associated with transferring these approaches to omics data. First, image data possess a spatial structure with strong local correlations and second, interpretation of image data is facilitated by the presence of clearly defined hierarchical concepts of the relationship of objects. For instance, a face is composed of a nose, a mouth and eyes, while eyes have lids. Thus, the individual variable (pixel) is usually of little importance compared to the composition of pixels into objects. In omics data however, the individual observed variable (gene) might be of great importance, and we are rather interested in identifying patterns observed in a small number of 'essential' observed variables. The extraction of these patterns is particularly relevant in unsupervised learning settings for several reasons. First, patterns, reflecting the essential structure, which is differentiated in the subpopulations of the empirical data, can be potentially useful for e.g. investigating cell differentiation processes in single-cell data. Second, identified patterns can serve as substitutes for external labels in judging the quality of models in the unsupervised setting, where prediction performance as a primary criterion for judging model quality is not available.

Here, we propose an approach (Fig. 1a) that extracts patterns from synthetic samples and corresponding latent representations learned by a deep generative approach such as VAEs or deep Boltzmann machines (DBMs) (Salakhutdinov and Hinton, 2009). Our approach is based on a rather simple, easily interpretable external model that allows one to infer the connections of latent representations with observed variables (features), and consequently adds to their explainability. Specifically, we train log-linear models on synthetic data, sampled from the trained generative models, to identify groups of features that are jointly associated with the states of latent variables. To obtain a low-dimensional representation of patterns seen in these groups of features, latent state information is again used (Fig. 1b). This allows us, for example, to monitor the training progress.

In the following, we provide a brief description of a generative deep learning framework, which encompasses VAEs and DBMs, before introducing the proposed approach of applying log-linear models to extract patterns from synthetic data generated from the deep

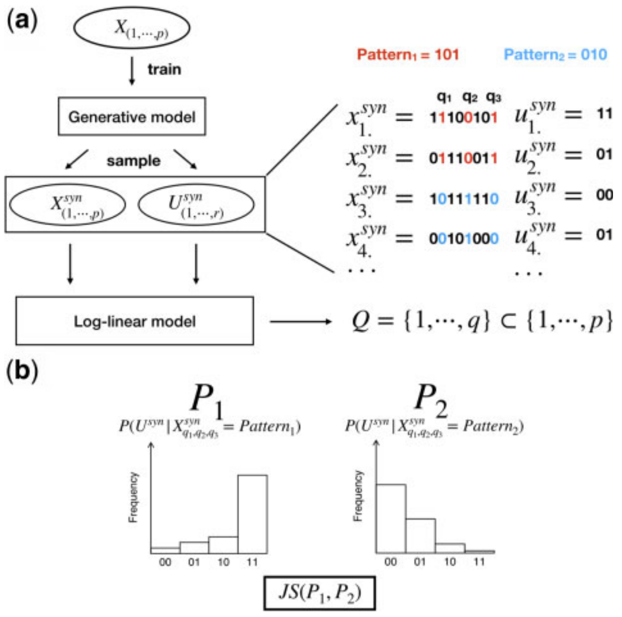
(a)
$X_{(1,\cdots,p)}$
train
Generative model
sample
$X^{syn}_{(1,\cdots,p)}$
$U^{syn}_{(1,\cdots,r)}$
Log-linear model
$Q = \{1,\cdots,q\} \subset \{1,\cdots,p\}$
Pattern₁ = 101
Pattern₂ = 010
q₁ q₂ q₃
$x^{syn}_{1.}$ = 11100101 $u^{syn}_{1.}$ = 11
$x^{syn}_{2.}$ = 01110011 $u^{syn}_{2.}$ = 01
$x^{syn}_{3.}$ = 10111110 $u^{syn}_{3.}$ = 00
$x^{syn}_{4.}$ = 00101000 $u^{syn}_{4.}$ = 01
(b)
P_1
$P(U^{syn} | X^{syn}_{q_1,q_2,q_3} = Pattern_1)$
P_2
$P(U^{syn} | X^{syn}_{q_1,q_2,q_3} = Pattern_2)$
Frequency
00 01 10 11
$JS(P_1, P_2)$

Fig. 1. Extraction of patterns from omics data (**a**) and identification of the similarity of these patterns (**b**). (a) Generative models learn low-dimensional latent representations (U) of high-dimensional categorical data (X) which in this example harbor two subpopulations. These subpopulations differ in multivariate patterns (Pattern_1 and Pattern_2) in a set (Q) of three features. We sample synthetic data (X^{syn}) and the corresponding latent representations (U^{syn}) from trained models. We apply log-linear models to the synthetic data to infer the set Q of features in X which are jointly correlated with the latent variables in U. (b) Having obtained Q (which contains the features q_1, q_2 and q_3 in this example) the dissimilarity of synthetic data, that differ in these features is inferred. In detail, we compute the Jensen–Shannon divergence between the distributions (P_1, P_2) of combinations of states of the here two latent variables in U, obtained for the synthetic data with a common pattern in Q

models. We suggest a stepwise approach to fit the log-linear models to build up sets of features, which represent the most important structure in the data. An approach for visualizing the relationship between groups of synthetic data, each group carrying a unique pattern, is proposed. The aforementioned approaches are illustrated with artificial data, mimicking cell differentiation processes and real single-cell gene expression data. Finally, we provide some remarks on limitations and potential generalization to other settings. Basic usage of the proposed approach is illustrated in an accompanying Jupyter notebook.

2 Materials and methods

2.1 Generative deep learning

A generative approach, in the present setting, is an approach that given some training dataset, i.e. the empirical distribution, can generate synthetic observations that should exhibit the most important structural properties observed in the empirical distribution.

More formally, training is based on n observations $x_i = (x_{i1}, \ldots, x_{ip})', i = 1, \ldots, n,$ each described by p features. After training on these original observations, a generative approach can generate synthetic observations $x_i^{\text{syn}} = (x_{i1}^{\text{syn}}, \ldots, x_{ip}^{\text{syn}})', i = 1, \ldots, n_{\text{syn}}$. The deep generative approaches that will be considered in this article also learn a latent representation for which values $u_i^{\text{syn}} = (u_{i1}^{\text{syn}}, \ldots, u_{ir}^{\text{syn}})', i = 1, \ldots, n_{\text{syn}},$ often corresponding to the states of hidden nodes in the deep network, can be obtained for each synthetic observation. This latent representation is characterized by a number of latent variables that is typically smaller than the number p of features (Fig. 1a).

We now briefly describe how the two specific deep generative approaches that we will consider in the following, deep Boltzmann machines (DBMs) and variational autoencoders (VAEs), fit into this framework (for a visualization, see Supplementary Fig. S1).

DBMs are Markov random fields that consist of a visible layer which contains variables that represent the states observed for the p features in the n observations, and at least one latent layer containing at least one latent variable u. In our scenario, the features can have two or more states and the frequency of states is Bernoulli or Softmax distributed. Latent variables have two states whose frequency is Bernoulli distributed. DBMs learn the joint distribution $P(x)$ of the training data x, using Gibbs sampling to infer the expectation for the states of observed and latent variables unconditional on the training data. DBMs are energy-based models. During training, the weights which connect observed and latent variables are set to retrieve a low energy for frequently observed patterns. Consequently, in a trained DBM, the states of the latent variables are associated with patterns in the features, i.e. in the training data. In our approach, we focus on the variables in the visible layer and the variables in the highest hidden layer.

The objective in training VAEs is to retrieve a usually lower-dimensional latent representation u, drawn from distribution $P(u)$, conditional on the training data x that emerges from the higher-dimensional observed empirical distribution $P(x)$. To facilitate learning, a standard normal distribution (z) is used as a prior for $P(u)$. During training, the mean μ and standard deviation σ of the normal distribution are learned. Usually a deep neural net, the encoder, is used to transform the training data, observed from $P(x)$, into the lower-dimensional space of $P(u)$. Here, we dichotomize samples drawn from $P(u)$ at the median to retrieve binary, categorical data. We model joint patterns between the latent variables in u and the variables in the output layer, representing a transformation of a sample from $P(u)$ into the space of $P(x)$.

2.2 Log-linear models

The aim in the following is to identify a subset containing q of the p features that are jointly associated with the latent variables (Fig. 1a). Specifically, joint patterns between the features in the synthetic data $x_i^{\text{syn}}, i = 1, \ldots, n_{\text{syn}}$ and their corresponding latent representation $u_i^{\text{syn}}, i = 1, \ldots, n_{\text{syn}}$, are investigated. We assume the states of the latent variables and the features to be sampled from a discrete distribution. Consequently, we observe at least two levels (l) for all variables/features. This implies that the features and the latent variables have to be on a binary or categorical scale. If values of latent representations are continuous, they need to be categorized beforehand. This applies to the VAE as used in our study since in the VAE, the states of latent variables are drawn from a multivariate normal distribution. We model the joint patterns between features and one or many latent variables based on the frequency of the combinations of states of the features and the latent variables in the synthetic data. The resulting contingency table is then analyzed using log-linear models.

We fit the log-linear models using a stepwise approach to build up a subset, comprising q of the p features that are jointly associated with the latent variables. Given a set of q^* features that has already been identified to be correlated with a latent variable u (omitting indices for simplicity), the addition of the feature with index j to the set is evaluated by the log-linear model

$$\begin{aligned}\log(m_{l_u,l_1\ldots l_q^*,l_j}) = \mu + \lambda_{l_u}^{U} + \lambda_{l_1}^{X_{(1)}} + \cdots + \lambda_{l_q^*}^{X_{(q^*)}} + \lambda_{l_j}^{X_j} \\ + \cdots \\ + \lambda^{UX_{(1)}\cdots X_{(q^*)}}.\end{aligned} \tag{1}$$

$m_{l_u,l_1\ldots l_q^*,l_j}$ indicates the expected frequency in the contingency table for level l_u of the latent variable, levels $l_1,\ldots,l_q^*$ of the features already in the set, and level l_j of the feature that is a candidate for addition. The parameter μ specifies the baseline frequency, i.e. it normalizes for the overall number of counts in the table. The parameters $\lambda_{l_u}^{U}$ specify the main effects of the respective levels of the latent variable, i.e. the marginal frequency for level l_u of the latent variable. $\lambda_{l_1}^{X_{(1)}},\ldots,\lambda_{l_q^*}^{X_{(q^*)}}$ and $\lambda_{l_j}^{X_j}$ specify main effects of the levels of features, i.e. their marginal frequency in the table. $\lambda^{UX_{(1)}\cdots X_{(q^*)}}$ represents the highest-order interaction between the latent variable and all features, except j. Lower-order interactions between the latent variable and all features, except j are represented by $\cdots$.

By comparing the fit of the above model to the full model, i.e. a model that includes all interaction terms between the state of the latent variable, all features already in the group and the change in the candidate feature j, we can evaluate whether knowledge of the state of the latent variable is needed to explain changes in the candidate feature in the context of the group. In other words whether the latent variable, the candidate and the already selected observed variables form joint patterns. Specifically, the difference in the G^2 goodness-of-fit statistic between these two models is considered. If there are several latent variables, the maximum of the differences is used. This is calculated for all features not yet in the set, and the feature with the largest difference is added. Starting from a set of size zero, this is repeated until a set of the desired size is reached.

If alternatively a significance test for each addition is desired, a permutation approach could be used to obtain a null distribution, where each feature is permuted separately to remove joint patterns. For each permutation, the whole process described above, i.e. fitting a generative model and log-linear modeling, would be performed, to obtain a valid null distribution that does not overlook potential overfitting in the analyses in the original data. The algorithm then starts with the best pair of features in the set to initially improve over the null distribution.

2.3 Low-dimensional latent representation of patterns

After having identified a set of q features that are particularly correlated with the latent variables, it is first of interest to retrieve the most frequent different combinations of discrete states observed for the q features, i.e. different patterns. Since the synthetic data are categorical, either as directly obtained from the generative model, or due to categorization, the most frequent patterns can be determined by simply counting the number of synthetic samples, carrying a specific pattern. In addition, it will be of interest to infer whether different patterns are in fact related to distinct subpopulations of samples in the synthetic data obtained from a deep generative approach, as this will point toward distinct subpopulations in the original data.

To investigate the relationship between synthetic data that are grouped based on their pattern in the q features, a distance measure is needed to infer the dissimilarity between these groups. For visualization purposes, the distances might then be mapped onto a two-dimensional space. Instead of determining dissimilarity at the feature level, we suggest using the latent representation, to extract what the generative model considers as similar or as dissimilar (Fig. 1b). Using the latent representations is beneficial because of usually reduced dimensionality and generally improved linear separability.

As multiple samples will share a distinct pattern, there will be several combinations of values for the latent variables for each pattern, i.e. there is a distribution over the states of latent variables for each pattern. We suggest quantifying the dissimilarity between the distributions of latent variables corresponding to the patterns by the Jensen–Shannon divergence

$$D_{JS}(P_1 \parallel P_2) = \frac{1}{2}D_{KL}(P_1 \parallel M) + \frac{1}{2}D_{KL}(P_2 \parallel M) \tag{2}$$

with

$$M = \frac{1}{2}(P_1 + P_2). \tag{3}$$

P_1 and P_2 representing the frequencies of observed latent variable configurations for two patterns and D_{KL} denoting the Kullback–Leibler divergence

$$D_{KL}(P_1 \parallel P_2) = \sum_{u\in U} P_1(u)\log\left(\frac{P_1(u)}{P_2(u)}\right). \tag{4}$$

Here, u is an example of the observed configurations U of latent variables, and $P_1(u)$ and $P_2(u)$ are the frequencies of the latent variable pattern u in two populations of samples which differ in the pattern in the q features extracted from the p features using the log-linear models. From equations (3) and (4), we see that the Jensen–Shannon divergence represents a symmetric version of the Kullback–Leibler divergence.

Having obtained a matrix of dissimilarities between groups of synthetic data carrying a common pattern, a low-dimensional representation can be obtained by e.g. multidimensional scaling (Kruskal, 1964).

2.4 Training and evaluation of VAEs and DBMs

Our proposed method is intended to work with different deep generative architectures. We here demonstrate the usage with DBMs and VAEs. Since the focus is on the development of a method which extracts the structure learned by a generative approach, we aim to keep the architectures as simple as possible. Thus, we use the typical parameter settings for the two deep architectures. Since we are modeling binary data, we use the sigmoid function as activation function for the nodes modeling the observed variables. For the latent variables, we use the activation functions which were used in the original publications. In the following, we describe the details of the models. More details on the configurations used are shown in Supplementary Tables S1–S5.

We train standard VAEs as described by Kingma and Welling (2013) to model Bernoulli distributed data. The VAEs learn a latent representation u containing 5–20 variables. We use a model architecture comprising one hidden layer in both the encoder and decoder network, while the number of latent variables in both hidden layers equals the number of features (Supplementary Fig. S1). Except for the terminal layer of the decoder network, where the activations of nodes are modeled to be Bernoulli distributed, we use the Tangens hyperbolicus as activation function. Parameters are optimized using the ADAM optimizer with a learning rate of 0.0001.

The DBMs used consist of visible and latent variables whose activation is Bernoulli distributed, arranged in one visible and two hidden layers (Supplementary Fig. S1). The number of nodes in the first hidden layer equals the number of modeled features, while we perform a reduction in the second hidden layer. These numbers were selected based on recommendations by Hinton (2012). We train

DBMs as described in Hess *et al.* (2017). In brief, we pursue a two-step training procedure, where layerwise pretraining, serving to initialize the weights between two adjacent layers to meaningful values, is followed by jointly fine-tuning the weights, while allowing for top-down feedback from deeper layers. For both steps, we use a learning rate of 0.001.

For the single-cell gene expression data, we tune the number of epochs for the VAE and the DBM training, which in case of the DBM are partitioned into pretraining epochs and fine-tuning epochs. Since the training of VAEs and DBMs relies on start values, i.e. initializations, for the weights which are randomly set, we also tune the start values of the model weights by evaluating ten different initializations of the model weights for different training epochs. For better comparability of the results from VAE and DBM, we tune the number of optimal training epochs and initialization of model parameters based on the distance between synthetic data and test data as inferred through a multivariate measure of the similarity of two distributions. The purpose here is to have an external metric which can be easily evaluated for both architectures. Specifically, we use the Cramér statistic (Baringhaus and Franz, 2004), calculated based on the five principal components of the synthetic data and the test data and which we aim to minimize.

VAEs and DBMs are implemented in Julia (Bezanson *et al.*, 2017), using the Flux package (Innes, 2018) for VAEs and the BoltzmannMachines package (Lenz *et al.*, 2019) for DBMs.

2.5 Processing of RNA-Seq data

We normalize RNA-Seq data for sequencing depth using DESeq (Anders and Huber, 2010). We then dichotomize the expression data at the median.

2.6 Annotating synthetic data by patterns

We infer the capability of the log-linear modeling approach to extract relevant patterns from the trained generative models by annotating synthetic data sampled from the model with labels from empirical observations (Supplementary Fig. S2). Using the log-linear modeling approach, we extract the q relevant features as described in Section 2.2. We then transfer the label (y_j) of an empirical observation (x_j) to a synthetic observation (x_j^{syn}) sampled from the generative model based on the similarity in the extracted pattern in the q variables. Specifically, we use a binary similarity criterion where the label is only assigned in the case of a perfect match

$$x_{j,(1,\dots,q)} = x_{j,(1,\dots,q)}^{\text{syn}}. \tag{5}$$

If there is no match, we randomly assign a label to the sampled synthetic observation. We then visually and quantitatively examine the distance between synthetic data carrying a specific label and the empirical observation from which the label has been transferred. Specifically, we investigate the first two principal components computed for a joint dataset of standardized synthetic data and empirical data. In our application, the empirical observations are gene expression vectors obtained from single cells, while the labels indicate the cell type. To quantitatively judge the quality of assigning the samples to clusters, we use an internal clustering index, specifically the Davies–Bouldin index (DBi) (Davies and Bouldin, 1979). The DBi is defined as the ratio of the spread within a cluster and the distance between clusters. Consequently, a lower DBi indicates a better clustering. We compute the DBi using the R package ‘clusterCrit’ (Desgraupes, 2018).

2.7 Non-negative matrix factorization

For comparison with the log-linear modeling approach, we extract the most information carrying variables with non-negative matrix factorization (NMF) (Lee and Seung, 1999). NMF factorizes a non-negative matrix V, where n observations are arranged in columns and p variables are arranged in rows, into two non-negative matrices W and H. W has p rows and k columns and H has k rows and n columns. k indicates the number of latent factors to be learnt. Specifically, we aim at minimizing the error of reconstructing V by the product of W and H. By selecting k so that $k*p+k*n < n*p$, data compression can be achieved. By requiring both the W and the H matrix to contain only non-negative entries, we achieve sparseness of W, meaning that the observed variables are not densely connected with the latent factors. Since W indicates, how strongly observed variables, here genes, are connected with the latent factors, i.e. how important they are for the reconstruction, we choose the genes with the highest entries in the W matrix. Specifically, for each row in W, we select the highest loading of the respective gene on any of the k learnt latent factors, and then pick the rows, i.e. the genes, with the highest maximum loadings. There are two main versions of the NMF algorithm which aim at minimizing two different objectives (Lee and Seung, 2001). One aims at minimizing the Euclidean distance between the original data and the reconstructed version, while the other aims at minimizing the Kullback–Leibler divergence. In fact, we use the algorithm, based on the Kullback–Leibler divergence, as described in Brunet *et al.* (2004). Compared to the algorithm that builds on the Euclidean distance, it has been demonstrated to deliver better performance with gene expression data (Brunet *et al.*, 2004). Since NMF is iteratively fit, requiring (random) start values, we evaluate ten different random initializations. We also vary the number of latent factors (k) from 1 to 10. From the resulting 100 factorizations, we then select the best factorization based on the DBi. We perform the factorizations with the R implementation NMF (Gaujoux and Seoighe, 2010).

3 Applications

We demonstrate the ability of our approach to uncover structure in omics data learned by generative models such as VAEs or DBMs based on artificial and real single-cell gene expression data. We use the aforementioned log-linear models to (i) select features, representing the essential signal in the data, (ii) identify subpopulations in the data, characterized by specific patterns in the selected features and (iii) investigate the relation of the identified subpopulations based on their low-dimensional latent representations. We show how the extraction and investigation of patterns can guide the tuning of hyperparameters such as the number of training epochs or can aid in understanding the structure of real high-dimensional gene expression data.

3.1 Artificial data

3.1.1 Simulation design

The structure of our artificial dataset is inspired by single-cell gene expression data characterizing different cell types or distinct stages of a cell during a differentiation process. Some sets of genes are exclusively expressed within a cell type, while some genes are highly expressed across different cell types. In the following, we will speak of ‘genes’ when we refer to the features.

Specifically, we create a stair-like structure, where the hypothetical genes characterizing a specific cell type are arranged in blocks with some overlap between genes. These overlapping genes represent genes with high expression shared between two cell types. Interpreting the data structure as a representation of a cell differentiation process, each stage is thus characterized by the abundance of specific genes. The overlap of some genes between one stage and the following stage models a smooth transition between one stage in cell development and the next. In our dataset, we discriminate between 10 distinct cell types or stages based on the expression of 32 genes: each cell type is characterized by high expression of 5 genes and the sets of genes overlap by 2 genes. For instance, the first 5 genes are highly expressed in the first cell type, while genes 4 and 5 are also highly expressed in the second cell type, together with genes 6, 7 and 8. In the third cell type, genes 7 and 8 are still highly expressed, as well as genes 9, 10 and 11, and so on, resulting in a stair-like structure (Fig. 2).

We model gene expression as Bernoulli distributed binary variables. Among the genes generally highly expressed within a cell stage, the presence of a gene in the observations belonging to that cell type

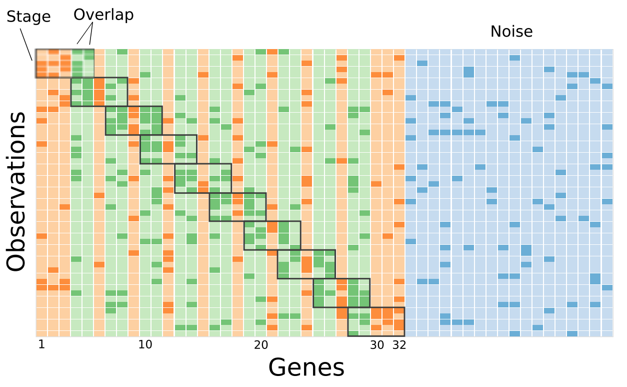

Fig. 2. Examples of simulated single cell expression data. The values of the first 32 genes characterize 10 different cell stages (black boxes), where each stage is characterized by high expression (dark color) of 5 genes while subsequent stages overlap (green) by 2 genes. The black boxes encircle the characterizing genes of one cell stage. For example, observations 1–5 represent one cell type that is characterized by high expression of genes 1–5

is encoded as 1 drawn from a Bernoulli distribution with probability $P = 0.6$ and absence is encoded as 0. To account for technical and biological noise, we model a gene to be present in any other cell stage with probability 0.1. Furthermore, we add 18 genes which purely represent noise and are again modeled to be present according to a Bernoulli distribution with $P = 0.1$.

3.1.2 Feature and pattern extraction

In this section, we demonstrate the capability of the log-linear modeling approach to extract relevant data features and the corresponding typical value combinations of those features over the course of training epochs. For demonstration purposes, we carry out this task with a VAE, but we have found the results to also generalize to the training process of a DBM.

Following the modeling scheme described earlier, we create 1000 artificial observations, train a VAE for 1000 epochs and apply log-linear modeling every 5 training epochs on a set of 10 000 synthetic data samples to build a set of 7 genes most strongly correlated with the latent states. We then extract the 10 most frequent patterns exhibited by those genes and investigate how the extracted patterns vary over the course of training (Fig. 3).

At the beginning of the training process, mostly noise genes are included in the set of selected genes, indicating that the VAE has not yet learned to identify the genes that characterize the specific cell stages (and potentially overlaps). After about 20 training epochs, however, the genes from within a cell stage, specifically from the overlaps become the features most strongly correlated with the latent variables, i.e. the VAE builds its latent representation of the data such that it focuses specifically on the genes governing the differentiation process. After around 30 epochs, the VAE relies mostly, and after 75 epochs almost exclusively, on overlap genes. This indicates that the VAE identifies the essential structure in the data, since overlap genes occur more frequently in the data than the genes exclusively abundant in one cell type. Often, the two genes that jointly form an overlap tend to be selected in pairs.

As the VAE learns to focus on the overlap genes, the stair-like structure of the input data becomes more clearly visible in the most frequently generated patterns (see the patterns obtained at epoch 75 in Fig. 3). This implies that the VAE gradually learns the typical value combinations of the overlap genes found in the training data. There, only genes from one overlap or two neighboring overlaps are highly expressed within a simulated cell stage while genes from the other overlaps are weakly expressed. Each of the most frequent patterns thus describes the state of genes characteristic for one or two specific cell stages: either one or two genes from one overlap or two genes from two neighboring overlaps are selected.

The aforementioned behavior is robust to the initialization of the model weights although the sets of genes, corresponding patterns and the number of epochs after which stable states are observed

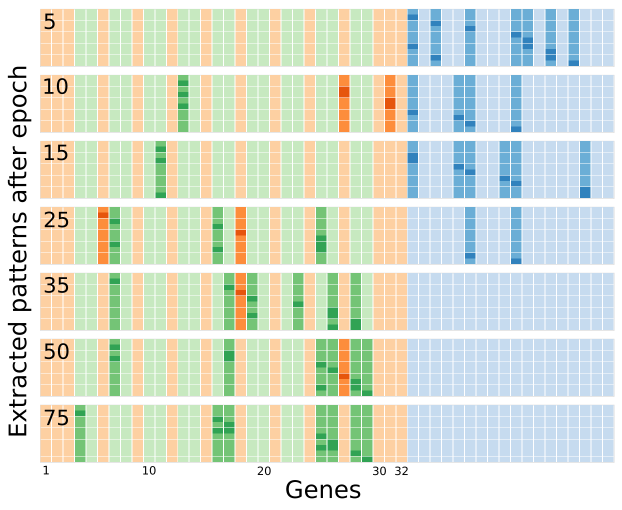

Fig. 3. Features selected by the VAE over the course of training and patterns in the selected features. The 10 most frequent patterns in the selected 7 features after training for a given number of epochs are shown. Orange = genes characterizing the cell stages, green = genes expressed in two adjacent cell stages. Bright color = feature not selected, medium color = feature selected, value '0', dark color = feature selected, value '1'

during the training process might vary. This suggests that by our modeling approach, we can uncover local optima of the loss function and to some extent also assess training variability. Finally, by looking at how selected genes and their value patterns change over time, we gain insight into the development of the VAE internal representation of the data and can monitor the training process. In particular, these analyses allow the researcher to inspect the quality of the model fit over different training epochs.

3.1.3 Dissimilarity of uncovered patterns

In the previous section, we identified the genes which represent the essential structure in the data and their typical patterns, representing the subpopulations, i.e. different cell stages or types found in the data. Here, we use the latent representation of frequent patterns derived over the course of training, i.e. at different states in the training process, to visualize how the dissimilarity of the identified patterns changes as training progresses. On the one hand, this allows one to monitor the training process by investigating the concordance of observed genes with their latent representations, e.g. for tuning the number of training epochs. On the other hand, it provides a robust distance measure to study the relation between the identified subpopulations.

Here, we use the same stair-like structure as before, i.e. simulating expression of 32 genes discriminating 10 cell stages, but we increase the number of noise genes to 168 and the number of observations to 5000. Consequently, we model the expression of 200 genes in 5000 simulated cells.

On this dataset, we train a DBM for 2, 20, 50, 100 and 200 epochs, respectively, and at each state select 4 features. To compare these five states, we have to make sure that all features, selected at the five states, are considered. Consequently we form the union of the features selected at the five states. In this example, this results in a total of 11 features that are further investigated. We then extract the 10 most frequent patterns observed in the 11 features for each state, i.e. each investigated training epoch. To compare the distance of the most frequent patterns at all five states during training, we again form the union of the, in this example, five sets of the most frequently observed patterns obtained at a specific state, resulting in a total of 12 patterns (a scheme of how unions of features and patterns are formed can be found in the Supplementary Fig. S3). These 12 binary patterns comprise 11 selected genes; in each pattern, there is at most one gene with value 1 while all other genes have value 0. For each of the investigated epochs, i.e. different model states, we

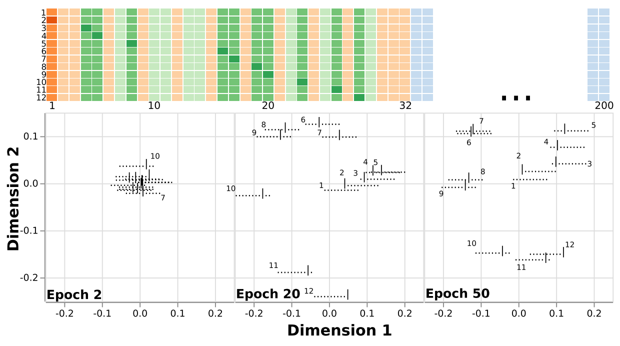

Fig. 4. Dissimilarity of frequent patterns over the course of DBM training. Each plot corresponds to the states of the DBM after being trained for the specified number of epochs. Each object corresponds to a pattern observed in the selected 11 genes. Dashes indicate a 0 and vertical bars a 1 observed for a selected feature. Extracted patterns above the plots are numbered according to the objects depicted on the dissimilarity plots. Bright color = feature not selected, medium color = feature selected, value '0', dark color = feature selected, value '1'

then obtain a two-dimensional representation of the common patterns as described earlier (Fig. 4).

At the beginning of training, all patterns are clustered together, indicating that the DBM has not yet learned a latent representation of the dataset characteristics. As the training process continues, however, the patterns become arranged in a more structured way. Specifically, the patterns form a spiral-like structure after 20 epochs. The spiral starts at the center of the plot with the pattern consisting only of 0 s. As we trace the patterns with a value 1 generated in the first, second, third features and so on, we observe their representations spiraling outward in a counter-clockwise direction from the center. Moving outward along the spiral thus reflects the gradual change in highly expressed genes from the first to the last gene in the selected set. At higher epochs, we observe a similar spiral-like structure, while the rotation of the spiral changes from counter-clockwise to clockwise (epoch 50) and back (epoch 200, shown in the supplementary Jupyter notebook).

Repeating the entire process 100 times to confirm the robustness of our method and findings, we stably observed the aforemenioned pattern representation in a circular or spiral-like shape after only 5–10 training epochs, and then maintained in an overall stable way, as is shown in Figure 4. We also frequently observe the aforementioned change in rotation and moderate shifts in the arrangements of patterns despite the general structure of the representation being stably maintained. Taken together, the extraction of patterns and further inferring the dissimilarity between patterns based on their latent representations allows us to monitor how the DBM learns the inherent characteristics of the input data. This information can e.g. be used to tune the number of epochs in the absence of an external criterion. In addition, the visualization of the internal representation of the DBM in two-dimensional space also serves to assess its modifications and overall stability during the course of training.

3.1.4 Scalability

To investigate the performance of the method with respect to the number of features to be modeled, we increased the number of noise variables from 168 to 368 and 968 resulting in additional datasets with 500 and 1000 features and 5000 training examples. After training a DBM on these larger datasets, we observed that we could still extract relevant structure, reflected in the selection of a large number of information carrying variables (Supplementary Figs S4 and S5). However, compared with the smaller number of features, the method selected a greater number of (up to 4) noise variables.

3.1.5 Runtime

We benchmarked the speed of our implementation in extracting variables from synthetic data. We varied the total amount of features in the data and the number of features to be selected by the log-linear models. Taken together, the method scales better than linearly with the number of input variables and approximately exponentially with the number of information carrying variables to be selected (Supplementary Table S6).

3.2 Single-cell gene expression data

As the log-linear models were found to be capable of extracting patterns that characterize distinct subpopulations in simulated data, we test here whether the approach can extract biologically meaningful signals from real single-cell RNA-Seq data. To answer this question, we investigate expression in 1525 single cells extracted from the cortex of mice (Tasic *et al.*, 2016) (Supplementary Fig. S6).

These cells are characterized by the differential abundance of 105 cell type-specific marker genes based on which different subtypes of neurons and non-neural cells can be differentiated. Our question is whether our approach can extract the genes from the joint set of marker genes which best discriminate between the cell types. In analogy to the simulation experiment, since we removed the non-neural cells prior to training, the non-neural marker genes serve as noise variables which should not be selected by a well-trained deep generative model. Before training, we dichotomized the expression data at the median.

3.2.1 Selected marker genes

We train VAEs and DBMs on 63.2% of the expression data described earlier and evaluate the generative models on the remainder of the data for selecting the optimal number of epochs as described in 'Training and evaluation of VAEs and DBMs'. We use the same architecture as described in Section 2, but here the encoder network of the VAE comprises two hidden layers with 105 nodes in the first and 52 nodes in the second layer (see Supplementary Tables S3 and S4 for the architectures of VAEs and DBMs, respectively).

After training the DBMs and VAEs until convergence, we use log-linear models to select the features, i.e. the genes representing the essential structure in the data that has been learned by the model. For both the VAE and DBM, we allow the approach to select 12 genes based on 10 000 examples of synthetic data sampled from the generative model for observed and the associated latent variables. Eight of the 12 selected genes, namely Calb2, Car4, Cdh13, Penk, Pvalb, Rorb, Sst and Vip are selected based on generated data from both the VAE and DBM, indicating that both approaches have learned similar structure in the data. We further monitor the selection of genes over the course of training in the VAE and the DBM (Fig. 5).

During the first epochs, a large proportion of the selected genes are non-neural marker genes. This indicates that the VAE and DBM had not learned the structure in the data yet, since there are no non-neural cells in the training data and correspondingly the expression of these genes should follow no pattern. From epoch 20 onwards, mostly GABAergic and Glutamatergic marker genes are selected. Of those, the genes Pvalb, Sst and Vip are of interest since their

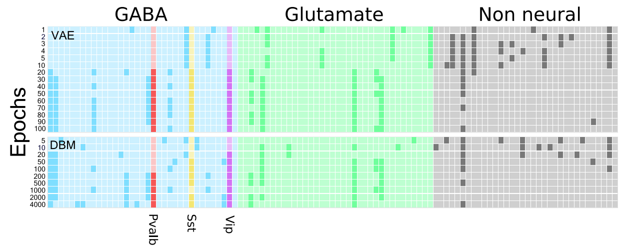

Fig. 5. Genes, extracted during the training of VAEs and DBMs. Genes were selected based on synthetic observations and the corresponding latent representation drawn from trained VAEs and DBMs using log-linear models. Each square represents a gene (horizontal) at a specific epoch (vertical). Squares are color coded according to the gene class. Bright colors indicate non-selected genes and dark colors indicate selected genes. The genes Pvalb, Sst and Vip are highlighted. Due to the different procedures used for training VAEs and DBMs, absolute numbers of training epochs are not directly comparable between both architectures

expression is an important marker which is known to discriminate three large subgroups of GABAergic neurons. This indicates that both models learned the essential structure of the data.

3.2.2 Relation of patterns in the selected genes to Sub populations in the data

To test whether patterns in the selected genes allow one to discriminate between sub populations in the data, we annotate generated synthetic data with the cell type label of empirically observed expression data based on the pattern in the selected genes (see Section 2.6 and Supplementary Fig. S2).

After annotating the synthetic data with labels from the empirically observed data, we investigate whether synthetic data, annotated with a given cell type label, are similar overall to the empirical observations of cell types from which the label has been transferred (Fig. 6). By visually inspecting the similarity, we observe that the synthetic data annotated with a certain label is in fact found in close proximity to cells of a certain type from which the label has been transferred.

We also observe that we can discriminate between a larger number of different cell types, when increasing the number of selected genes used for annotation (Fig. 6, from ‘4 genes’ to ‘12 genes’). This indicates that all selected genes are informative for the discrimination between cell types. Overall these results indicate, that the log-linear approach was capable of extracting those genes which contribute most to the structure in the data since pattern in the genes allow for discriminating between different cell types.

To confirm that the annotation of synthetic data is robust to variation in the selected genes, i.e. that the data is less well annotated using a random selection of marker genes, we visually investigate the sample labeling based on sets of randomly selected genes. In addition, we investigate the clustering performance using an internal clustering evaluation metric, specifically the DBi. For 100 randomly selected sets of 8 genes, we always observed worse concordance between synthetic data carrying a specific label and the cell type from which the label has been transferred as compared with the genes selected by the log-linear models. Correspondingly, the DBi was always higher for the random subsets, indicating worse clustering. This applied to both the VAE and the DBM (examples shown in Supplementary Figs S7 and S8).

3.2.3 Robustness of obtained patterns against dichotomization and model initialization

We dichotomized the quantitative gene expression data at the median before training the deep generative models. Thus our results

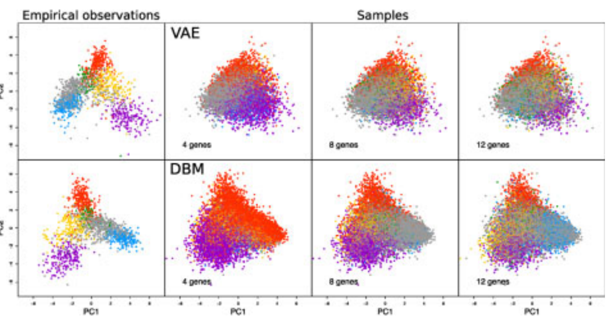

Fig. 6. Identification of cell populations based on extracted genes. Synthetic data sampled from a VAE (upper row) or a DBM (lower row) are annotated with the labels of empirically observed expression data based on pattern matching. The patterns are made up from the expression values observed in 4–12 genes, selected using log-linear models. Synthetic data drawn from each generative model, as well as expression data are jointly standardized, and principal components (PC) are computed for the joint set. Shown are the first two PCs for the empirical expression data (left) and 10, 000 samples (right), annotated with labels from the empirical data. Colors refer to the cell type label. For both the VAE and the DBM, the same empirical observations are shown. Purple = Vip cells, yellow = Sst cells, red = Pvalb cells, blue = L4 cells, green = L5b cells and gray = other cells

might be affected by the cut-point used for dichotomization. Consequently, using DBMs as an example, we tested how different cut-points affect performance. We split the training data at cut-points ranging from the first to the ninth decile (Supplementary Fig. S9) and for each cut-point trained 10 DBMs with different random initializations of the parameters. Based on samples from the DBMs, we then extracted the essential variables using log-linear models and inspected the clustering. We obtained generally similar clustering results at each cut-point as revealed by visual inspection (Supplementary Fig. S10). In terms of the DBi, the best results were obtained when splitting at the third quantile, while the median split resulted in the worst performance (Supplementary Table S7).

3.2.4 Comparison with NMF

To illustrate the potential benefit of using deep models such as DBMs in combination with log-linear models to extract the essential structure from omics data such as single-cell RNA-Seq data and to obtain a performance reference, we also used an established, non-deep technique, specifically NMF. NMF has been demonstrated to be able to extract the information carrying genes from RNA-Seq data (Zhu *et al.*, 2017). Deep variants of NMF have been proposed as well, e.g. in Yu *et al.* (2018). As we did not want to perform an extensive comparison study, we chose a rather straightforward non-deep version of NMF, as used in Brunet *et al.* (2004). Consequently, this comparison is not meant as a benchmarking with state-of-the-art approaches, but provides a performance context for our approach by showing results from an established technique. We learned latent factors using NMF on the training dataset also used for DBM training. We then selected the 12 genes which are most strongly connected with the latent factors learned by NMF.

Using NMF, we were able to extract nine of the genes also selected by log-linear models applied to synthetic data sampled from DBMs, namely Calb2, Car4, Cdh13, Parm1, Penk, Pvalb, Rorb, Sst and Vip. Investigating the clustering of test data based on the 12 genes, we observed a more interpretable result for the genes extracted by the DBM and log-linear models [compare Supplementary Figures S11 (NMF) and 12 (DBM)]. This is essentially due to the genes Gria3 and Enpp2, selected by NMF but not by DBMs, which contribute to the overall structure but do not form joint patterns with the other selected genes. By contrast, Bcl6 and Mybpc1 selected by the DBM and log-linear models but not by NMF, form joint patterns with Pvalb and Parm1. In addition, the genes selected by NMF less accurately reflect the distance between the cell types. When comparing the distance between all cells belonging to the two major groups of cell types, specifically GABAergic and glutamatergic cells, we observe that the genes selected by the DBM and log-linear models reflect the expected similarity of cell types slightly better (Supplementary Fig. S13). Here, GABAergic and glutamatergic cells can be better separated when using the genes selected by DBMs and the log-linear models. The advantage of our proposal over NMF is found to be even more pronounced when investigating genes with a less sparse expression pattern, specifically 81 neurotransmitter genes (Supplementary Fig. S14). Here, the overlap between the genes selected by NMF and DBM is smaller (3 of 8 selected genes). This large difference is reflected in a considerably better clustering of glutamatergic and GABAergic cell types when considering the genes selected by DBMs and log-linear models (Supplementary Figs S15, S16 and S17).

4 Discussion

In this article, we demonstrate how log-linear models can be used to identify the essential structure in omics data, learned by generative deep models. By investigating the correlation between observed and latent variables, patterns in the features are identified that contribute most to the overall structure in the data. We also showed that the distribution of the states of latent variables can be used to estimate the similarity between synthetic data carrying a specific pattern. Using artificial and real gene expression data, we showed how these methods can be employed to ‘learn what the model has learned’ over the course of training, i.e. monitoring the training process, or to compare different generative models, such as DBMs and VAEs.

Using the real single-cell gene expression data, we confirmed that our approach can in fact can uncover biologically relevant signals, such as essential marker genes such as Vip, Sst and Pvalb which allow for discriminating between major subpopulations of cell types [see Tasic *et al.* (2016) for more information]. This means that our approach allows for the identification of multivariate patterns in high-dimensional data which characterize subpopulations in the data. This, in turn, allows for an interpretation of the differentiation of subpopulations in a holistic way, by studying the combinations of genes that are characteristic of different subpopulations. Compared with traditional unsupervised methods such as NMF, this property allows for the extraction of features that may better reflect the underlying biology. Here, we provided some context with respect to performance by showing results from a shallow, non-deep NMF approach as used in Brunet *et al.* (2004). Since there are a multitude of deep NMF approaches (e.g. Trigeorgis *et al.*, 2017; Yu *et al.*, 2018), deep versions of NMF might have performed significantly better in this comparison. However, a comprehensive comparison with such techniques would have been beyond the scope of the present paper. The variant of NMF considered here also has already been investigated with RNA-Seq data (Zhu *et al.*, 2017). Using Kullback–Leibler divergence instead of Euclidian distance, we also followed advice from the literature to improve the performance of NMF (Brunet *et al.*, 2004).

We demonstrated that our approach can be combined with different generative architectures such as VAEs and DBMs. Since we use a rather simple model that does not rely on additional hyperparameters or random initializations, the method leads to robust results.

4.1 Generalizations

Here, we focused on gene expression data but the approach is in principle applicable to any kind of categorical omics data or data that can be categorized without sacrificing a lot of information. We investigated only standard VAEs and DBMs but our method is in principle applicable to every deep generative model which allows users to draw synthetic data from the learned joint distribution of latent variables and features.

4.2 Limitations

Despite the promising results, there might be some limitations.

First, our method requires the synthetic observations and the corresponding latent representations to be categorical, which here for the single-cell gene expression data and the latent variables of the VAE is achieved by dichotomization. Naturally, this implies some loss of information. However, we argue that modeling categorical data has the huge benefit that potentially very complex joint distributions of continuous random variables do not have to be modeled by an external model, but rather somewhat less complex distributions can be modeled in high-dimensional contingency tables. Without such a reduction in complexity, the external model would potentially have to be as complex and therefore as opaque as the generative model. Indeed, the results from the single-cell gene expression data indicate that our approach was capable of extracting multivariate patterns that represent essential structure in the data. We also showed that the results are robust to the cut-point used for dichotomization. Additionally, dichotomizing the continuous latent representations of the VAE was unlikely to have resulted in a large loss of information since forming several categories of the normally distributed values of the latent variables instead of dichotomization did not improve the performance of the VAE. This supports our hypothesis that the reduction in complexity allowed stable and meaningful groups of features and patterns to be obtained, while compensating for the information loss due to dichotomization.

One also might argue that in our applications, we chose arbitrary numbers of features to be selected in the stepwise fit of the log-linear models. Consequently, it might not be guaranteed that meaningful features are selected. However, in both applications, we observed that all selected features contributed to the interpretability. In the single-cell application, for instance, we observed, that increasing the number of selected genes led to better capturing the diversity of cells (Fig. 6), indicating that all selected genes indeed contributed to the essential structure in the data. We also think that this behavior will generalize to other datasets since we aim to extract compact patterns, comprising about 10 features. Given the high number of features in high-dimensional data, one can assume that there would hardly ever be a situation where less than 10 features contribute to the structure, except if something has gone wrong in the experiment. However, if an objective criterion for feature selection is desired, a permutation-based *P*-value could be considered as described in Section 2.

Modeling large contingency tables with log-linear models can be time consuming when more than 10 features are intended to be selected, i.e. the resulting contingency table is at least 11-dimensional. However, this time can be reduced by parallel computing. In addition, being able to investigate joint patterns in 8–12 variables was sufficient in all investigated applications to infer a substantial amount of structure in the data, and larger structures cannot reasonably be expected to be identified given a limited number of observations. In situations with a high amount of sparsity in the data, e.g. a high number of genes that are exclusively expressed in a small number of samples, multiple deep generative models might be trained on partitions of the data, as described in Hess *et al.* (2017). Log-linear models would then be applied to each partition, and for each partition, the essential information-carrying variables would be extracted.

Despite the insight into the training process our method can provide, we strongly recommend to combine it with an external evaluation criterion such as the log-likelihood. This step is important if no ground truth is available, which was not the case in our applications. If no ground truth is available, using an external evaluation criterion can help to avoid, extracting patterns from an insufficiently trained model.

5 Conclusion

We propose a method to extract interpretable patterns from generative deep models, that represent the essential structure learned by the model. These patterns are especially valuable for better understanding complex omics data. Using real gene expression data, we demonstrate, that the approach can learn the essential, biologically meaningful, patterns in the data. Taken together, our method allows for an easy inspection of what a generative model has learned and is robust to the generative architecture selected.

Acknowledgements

The authors would like to thank James Balmford for proofreading the article.

Funding

The work of M.H. is supported by the Federal Ministry of Education and Research (BMBF) project ‘Generatives Deep-Learning zur explorativen Analyse von multimodalen Omics-Daten bei begrenzter Fallzahl’ (GEMOLS: Generative deep learning for exploring multimodal omics data with limited size, Fkz. 031L0250A).

Conflict of Interest: none declared.

References

Anders,S. and Huber,W. (2010) Differential expression analysis for sequence count data. *Genome Biol.*, **11**, R106.

Baringhaus,L. and Franz,C. (2004) On a new multivariate two-sample test. *J. Multivar. Anal.*, **88**, 190–206.

Besserve,M. *et al.* (2018) Counterfactuals uncover the modular structure of deep generative models. *arXiv preprint arXiv : 1812.03253v2*. https://arxiv.org/abs/1812.03253v2.

Bezanson,J. *et al.* (2017) Julia: a fresh approach to numerical computing. *SIAM Rev.*, **59**, 65–98.

Brunet,J.-P. *et al.* (2004) Metagenes and molecular pattern discovery using matrix factorization. *Proc. Natl. Acad. Sci.*, **101**, 4164–4169.

Camacho,D.M. *et al.* (2018) Next-generation machine learning for biological networks. *Cell*, **173**, 1581–1592.

Davies,D.L. and Bouldin,D.W. (1979) A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.*, **PAMI-1**, 224–227.

Desgraupes,B. (2018) *clusterCrit: Clustering Indices.* R package version 1.2.8. https://cran.r-project.org/web/packages/clusterCrit/index.html.

Ding,J. *et al.* (2018) Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nat. Commun.*, **9**, 2002.

Eraslan,G. *et al.* (2019) Single-cell RNA-seq denoising using a deep count autoencoder. *Nat. Commun.*, **10**, 390.

Gaujoux,R. and Seoighe,C. (2010) A flexible r package for nonnegative matrix factorization. *BMC Bioinformatics*, **11**, 367.

Harradon,M. *et al.* (2018) Causal learning and explanation of deep neural networks via autoencoded activations. *arXiv preprint arXiv:1802.00541v1*. https://arxiv.org/abs/1802.00541v1.

Hess,M. *et al.* (2017) Partitioned learning of deep Boltzmann machines for SNP data. *Bioinformatics*, **33**, 3173–3180.

Hinton,G.E. (2012) A practical guide to training restricted Boltzmann machines. In: Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg, pp. 599–619.

Innes,M. (2018) Flux: elegant machine learning with Julia. *J. Open Source Softw.*, **3**, 602.

Kingma,D.P. and Welling,M. (2013) Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114v10. https://arxiv.org/abs/1312.6114v10.

Kruskal,J.B. (1964) Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, **29**, 1–27.

Lee,D.D. and Seung,H.S. (1999) Learning the parts of objects by non-negative matrix factorization. *Nature*, **401**, 788–791.

Lee,D.D. and Seung,H.S. (2001) Algorithms for non-negative matrix factorization. In: Leen,T.K. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. **13**. MIT Press, Massachusetts, pp. 556–562.

Lenz,S. *et al.* (2019) Unsupervised deep learning on biomedical data with boltzmannmachines. jl. *bioRxiv* 578252. https://www.biorxiv.org/content/10.1101/578252v1.abstract.

Lopez,R. *et al.* (2018) Deep generative modeling for single-cell transcriptomics. *Nat. Methods*, **15**, 1053–1058.

Montavon,G. *et al.* (2018) Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.*, **73**, 1–15.

Rezende,D.J. *et al.* (2014) Stochastic backpropagation and approximate inference in deep generative models. arXiv preprint arXiv: 1401.4082v3. *https://arxiv.org/abs/1401.4082v3*.

Salakhutdinov,R. and Hinton,G. (2009) Deep Boltzmann machines. *In: Proc. International Conference on Artificial Intelligence and Statistics*, pp. 448–455.

Tasic,B. *et al.* (2016) Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nat. Neurosci.*, **19**, 335–346.

Trigeorgis,G. *et al.* (2017) A deep matrix factorization method for learning attribute representations. *IEEE Trans. Pattern Anal. Mach. Intell.*, **39**, 417–429.

Wang,X. *et al.* (2018) Conditional generative adversarial network for gene expression inference. *Bioinformatics*, **34**, i603–i611.

Yu,J. *et al.* (2018) Learning the hierarchical parts of objects by deep non-smooth nonnegative matrix factorization. *arXiv preprint* arXiv: 1803.07226v1, https://arxiv.org/abs/1803.07226v1.

Zhu,X. *et al.* (2017) Detecting heterogeneity in single-cell RNA-seq data by non-negative matrix factorization. *PeerJ*, **5**, e2888.