COMP 565: Assignment 4: VAE on scRNA-seq data

This assignment is worth 8% of your total grade and due at **23:59 on November 13, 2023**.

The Google Drive drive A4 contains the starter Jupyter notebook `comp565_fall2023_A4_starter.ipynb`, already implemented `etm.py` for ETM, and a mouse pancreas single-cell RNA-seq dataset.

Setting up Python environment & libraries on your computers

Set up the Python computing environment by installing the necessary Python libraries specified in the top of the `comp565_fall2023_A4_starter.ipynb` script. You will need Python 3.7 above to run the code properly.

The easiest way to install most of these Python libraries is via either command line command `pip` or Anaconda (`https://www.anaconda.com/`) command line command `conda`. Both are free.

For example, in single-cell analysis, a very popular library is called `scanpy`. You can use `pip` to install it (`https://scanpy.readthedocs.io/en/stable/installation.html`):

```
pip install scanpy
```

In this assignment, we will be using PyTorch to train the deep learning variational autoencoder (VAE). To install `pytorch` (`https://pytorch.org/get-started/locally/`), enter in command line: `conda install pytorch torchvision torchaudio -c pytorch`

If you have trouble, installing these libraries, let the TA or the instructor know as soon as possible to get help. Do it early.

Mouse Pancreas single-cell RNA-sequencing data

In the Google Drive directory there is a folder called `data`, which contains a mouse scRNA-seq pancreas dataset obtained from [1]. The dataset has 1886 cells measured over 4194 genes by scRNA-seq. The dataset were processed and filtered for the purpose of this assignment. Here are the individual files:

- `MP.pickle`: The single-cell expression count matrix (matrix of integers) of dimension 1886 cells by 4194 genes plus 2 columns indicating the batch IDs and cell types
- `MP_genes.pickle`: gene names for the 4194 genes
- `sample_info.csv`: a 1886 by 3 information matrix with the rows as the1886 cells and columns for the cell IDs, batch IDs (not used in this assignment), and cell type labels
- `cell_IDs.pkl`: cell IDs as a list of strings

The code at the beginning of the file `comp565_fall2023_A4_starter.ipynb` will load the scRNA-seq data into a Pandas DataFrame `df` as well as the M gene names `genes` using a Python library called `pickle`. It will reorder the rows and columns and save the final $N \times M$ matrix X into a Numpy `ndarray` for our subsequent matrix factorization tasks.

In addition, using `anndata` library we create an AnnData object called `mp_anndata`, which as the input `ndarray` matrix X and the ground-truth cell type labels for the 1886 cells for evaluation purpose.

Evaluating clustering by Adjusted Rand Index

Adjusted Rand Index is a popular metric used to evaluate the unsupervised clustering by comparing the consistency between the predicted clusters and the ground-truth clusters (i.e., cell type labels in this assignment). In `comp565_fall2023_A4_starter.ipynb` file, you are provided with a function called `evaluate_ari(cell_embed, adata)`, which takes $N \times K$ input cell embedding `cell_embed` with K as the embedding dimensions and the annotated cell label data as `adata`. It will first run UMAP to compute the distance between cells based on their embeddings and then run Louvain clustering using the cells-cells distance matrix from UMAP to cluster cells into groups defined by the resolution parameter (default: 0.15). Finally, it computes the ARI based on the Louvain clusters and the ground-truth cell type using `adjusted_rand_score` from `Scikit-learn`. We will be using `evaluate_ari` to evaluate the cell embedding quality by the autoencoders.

1 (2%) Implement a wrapper function to train ETM

The same Google Drive directory contains another Python file called `etm.py`, which implements a simpler strip-down version of the scETM (Figure 1) [2]. The code is the same as the original ETM implementation [3] that is available from `https://github.com/adjidieng/ETM`.

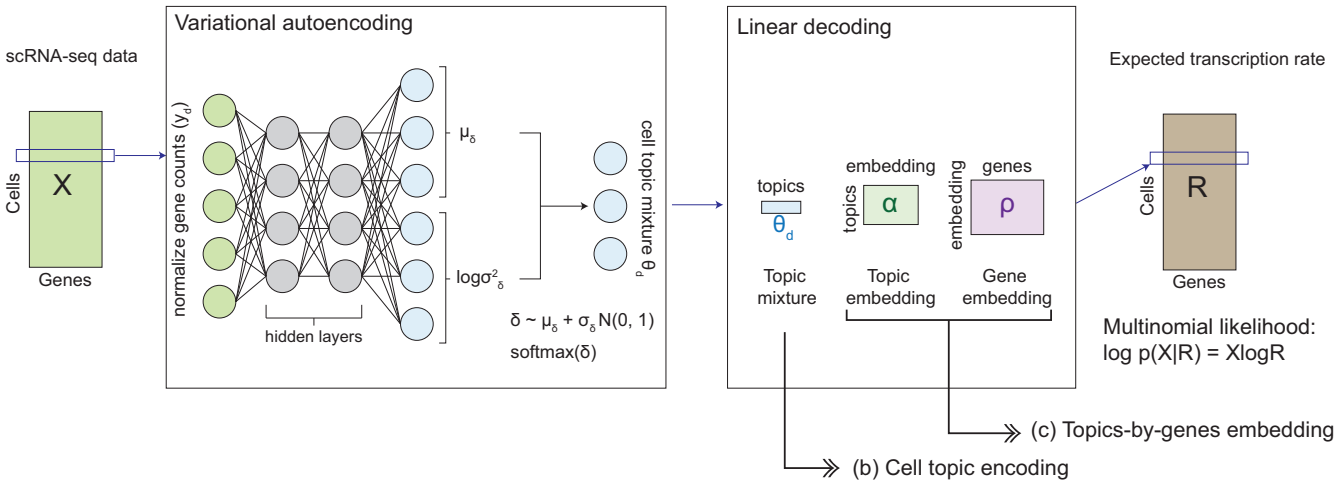

Figure 1: Modeling scRNA-seq data using single-cell embedded topic model (scETM) [2]. Each scRNA-seq profile serves as an input to a variational autoencoder (VAE) as the normalized gene counts. The encoder network produces a stochastic sample of the latent topic mixture (θ_d for cell $d = 1, \ldots, N$), which can be used for clustering cells. The linear decoder learns topic embedding and gene embedding.

The code to instantiate an ETM model object is already written for you in `comp565_fall2023_A4.py` file with pre-specified hyperparameters (i.e., topic number, hidden size, learning rates, weight decay penalty, etc). You want to start with these settings. Once you get the model working, you may play around with these parameters to get higher ARI. But that is not mandatory.

Implement the wrapper function called `train_scETM` that uses the helper functions `train_scETM_helper` to train a ETM and another completed helper function called `get_theta` (do not change) to compute cell embedding topic mixture θ. The latter two functions have been provided to you. **Do not modify them.**

PyTorch by default trace the error to calculate the gradient for backpropagation. When evaluating ARI, you want to turn off this automatic gradient trace by coding under the code block `with torch.no_grad()`.

Note here that `get_theta` takes the normalized gene counts by the total count per cell stored in `X_tensor_normalized` as the input to the encoder neural network (Figure 1b left-most side). The final reconstruction categorical likelihood is based on the unnormalized count data stored in `X_tensor` (Figure 1b right-most side).

Here ARI is computed based on the Louvain clustering the cell embedding θ against the ground-truth cell types stored in the `mp_anndata` object.

Training a neural network (i.e., the VAE in our case) requires many iterations because of the gradient descent updates with small learning rate. Run the model for 1000 iterations. Record the negative ELBO loss and ARI at each iteration and then use `monitor_perf` to display the

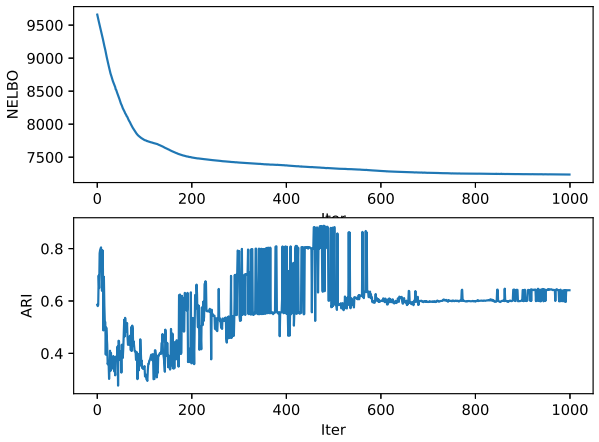

Figure 2: Negative ELBO and ARI of scETM at each iteration of the scETM training.

training progress (Figure 2). Have a cup of coffee (or tea) and let it train for about 5-10 minutes. You may try the same code on a NVIDIA GPU machine to observe a major speed-up compared to a CPU machine. But that's not required in this assignment.

2 (2%) Generate t-SNE or UMAP to visualize cell embeddings

Use the `model` object of ETM saved from the previous training over the 1000 iterations to infer final cell topic embedding θ and generate the two-dimensional t-SNE or UMAP plot. The plot was generated using a `Scanpy` functions `sc.tl.tsne` and `sc.pl.umap`. Learn how to use it from its documentation.

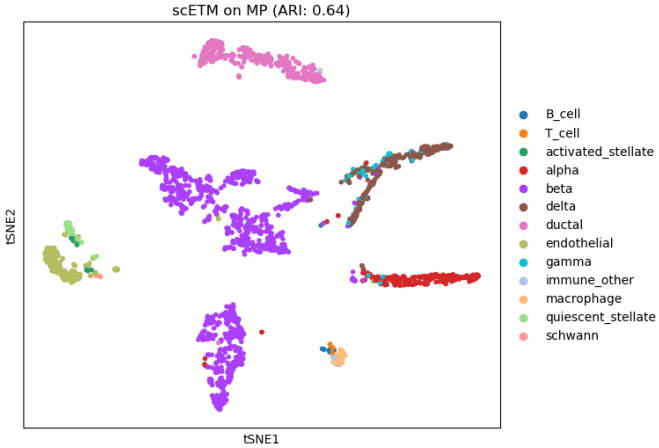

Figure 3: UMAP plot for the MP cell-embedding using θ from the 16-topic scETM. Cells are colored based on one of the 13 cell types.

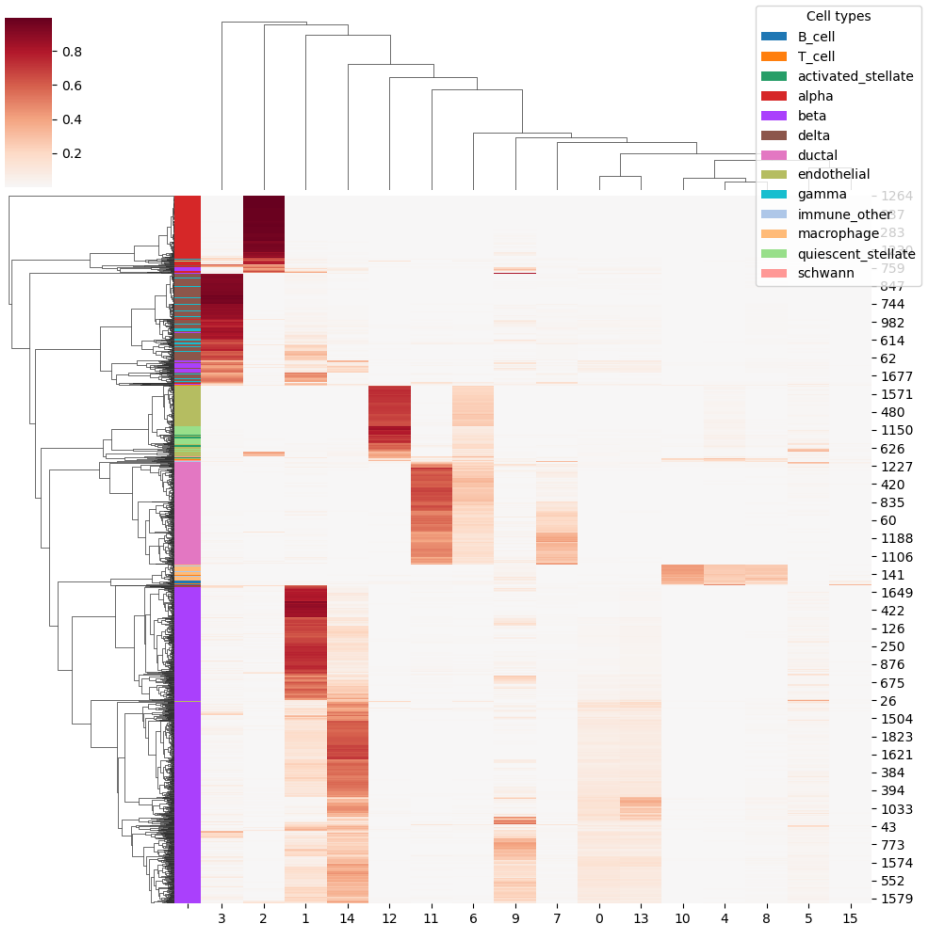

Figure 4: Cells by topics heatmap.

3 (2%) Plot heatmap for the cells under each topic

An alternative way to present the cell cluster is by heatmap. Heatmap is more effective in identifying which topics correlates well with which cell types. Generate a heatmap plot for the same cells-by-topics matrix θ using *all of the 1886 cells over the 16 topics*. Your heatmap should look similar to Fig. 4, which was generated using `seaborn.clustermap`.

From here, we see that topic 2 correlates well with alpha cell type, topic 12 with endothelial, topic 11 with ductal cell type, and so forth. Note that you will get different topic indices matching with different cell types as the order of these topics are not the same at different runs.

4 (2%) Plot heatmap for the top genes under each topic

To get cell-type-specific gene signature, we can visualize the genes by topics heatmap. Here we will plot the top 5 genes per topic in heatmap. Your heatmap should look similar to Fig. 5,

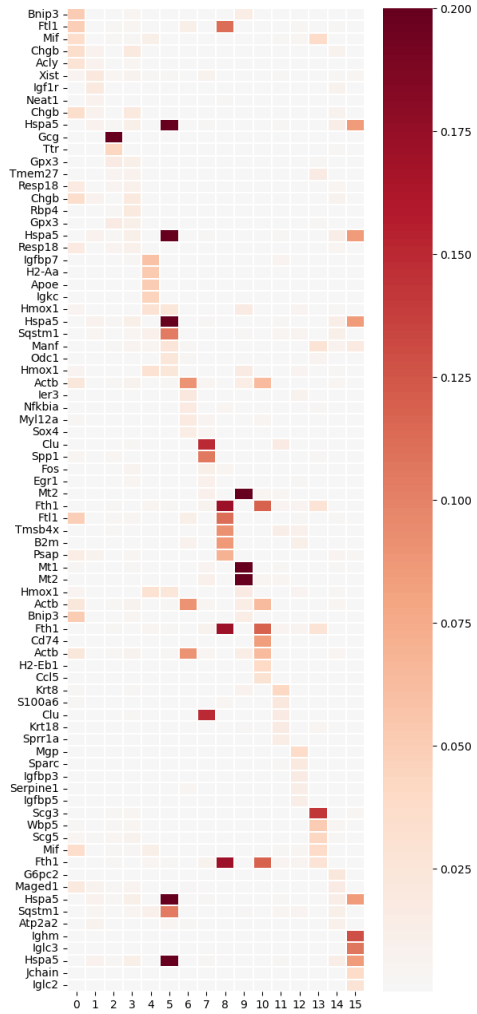

Figure 5: Heatmap of the top 5 genes per topic.

which was generated using `seaborn.heatmap`. In plotting the heatmap, I capped the max value at 0.2 instead of letting it set to 1 to make the red intensities more prominent for some of the genes with low absolute value under some of the topics.

What cell-type-specific gene signatures can you find in your analysis? For example, now we know that topic 2 corresponds to alpha cell type (Fig. 4). The gene *Gcg*, which codes for protein Glucagon has the highest probability under topic 2. Based on an online source, Glucagon is indeed a pancreatic hormone that counteracts the glucose-lowering action of insulin by stimulating glycogenolysis and gluconeogenesis!

5 Deliverables

Submit the completed `comp565_fall2023_A4.ipynb` on all four tasks.

References

[1] Maayan Baron, Adrian Veres, Samuel L Wolock, Aubrey L Faust, Renaud Gaujoux, Amedeo Vetere, Jennifer Hyoje Ryu, Bridget K Wagner, Shai S Shen-Orr, Allon M Klein, et al. A single-cell transcriptomic map of the human and mouse pancreas reveals inter-and intra-cell population structure. *Cell systems*, 3(4):346–360, 2016.

[2] Yifan Zhao, Huiyu Cai, Zuobai Zhang, Jian Tang, and Yue Li. Learning interpretable cellular and gene signature embeddings from single-cell transcriptomic data. *Nature Communications*, 12(1):5261, 2021.

[3] Adji B Dieng, Francisco JR Ruiz, and David M Blei. Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453, 2020.