Compression of Differential Expression Data with Deep Autoencoders

Tony Kabilan Okeke[1]
[1]School of Biomedical Engineering, Drexel University, USA

Course: BMES 547
Instructor: Ahmet Sacan
Date: 2023-03-23
Data: Digital Expression Explorer 2

Abstract

The paper presents a neural network model for predicting enriched Gene Ontology (GO) terms from RNA-Seq differential expression analysis (DEA) data. We trained an autoencoder model using log fold change (logFC) values from mouse RNA-Seq datasets curated from the Digital Expression Explorer 2 repository. We hypothesized that the autoencoder's latent space would capture the essential features and patterns of the logFC values to predict enriched GO terms. We normalized the feature matrix, removed low-variance genes, and compared their approach to previous studies. The results demonstrated that the autoencoder was able to reconstruct the log fold change values with a mean squared error of 0.92.

1 Introduction

Differential expression analysis (DEA) is a commonly used technique to identify genes that are differentially expressed between two or more biological conditions. Gene Ontology (GO) enrichment analysis, on the other hand, is used to interpret the biological functions and pathways associated with the differentially expressed genes. Both DEA and GO enrichment analysis is commonly used in the analysis of RNA-seq data. However, both can be affected by various sources of noise and bias, such as batch effects, sample heterogeneity, and gene annotation errors [1]. Thus, it is crucial to develop robust and accurate methods to integrate DEA and GO enrichment analysis in a data-driven manner.

To address this issue, we are developing a neural network model to predict the enriched GO terms from the log fold change (logFC) values computed by DEA. Neural networks are powerful machine learning models that can learn complex patterns and relationships from high-dimensional data. In this project, we aim to develop an autoencoder model that can learn a compressed latent representation of the logFC values. The autoencoder is composed of an encoder that maps the input logFC values to a lower-dimensional latent space and a decoder that reconstructs the input logFC values from the latent space. We hypothesize that the latent space of the autoencoder will capture the essential features and patterns of the logFC values. In our future work, we will use the latent space to predict the enriched GO terms from the logFC values.

There have been several previous studies that have used neural networks to predict gene expression or gene function from transcriptomic data. One such study is D-GEX, which proposed a deep neural network model to predict gene expression levels from a set of landmark genes [2]. The D-GEX model was shown to accurately predict the expression levels of thousands of genes with lower error rates that the current standard (linear regression) [2]. Our work differs from D-GEX in that we focus on the integration of DEA and GO enrichment analysis. Additionally, by first learning a latent representation of the DEA results, we will also be able to reduce noise and bias in the data, improving the accuracy of the downstream predictions.

2 Dataset

We used the *Digital Expression Explorer 2* (DEE2) project as the primary source of RNA-seq data for our analysis. The DEE2 project is a curated repository of uniformly processed RNA-seq data collected from public repositories, including the Sequence Read Archive (SRA) hosted by the National Center for Biotechnology Information (NCBI) [3].The raw fastq files are downloaded from the SRA, undergo quality control and alignment to the reference genome, and are quantified using the STAR and Kallisto tools. The feature count and metadata tables are available for download from the DEE2 website, and through the 'getDEE2' R package. A metadata table which lists all curated datasets for each species is also available for download. DEE2 provides 532,711 processed RNA-seq runs (SRRs) from mouse experiments and 475,547 processed RNA-seq runs from human experiments, as well as other species [3].

For our analysis, we curated a subset of the mouse RNA-seq data, which consisted of 11,130 unique datasets with over 350,000 individual runs. We obtained additional metadata for these datasets from the Gene Expression Omnibus (GEO) database. We filtered out datasets that had only one sample, or that did not have at least two distict conditions or labels. This resulted in a final collection of 7,130 datasets with gene counts for 48,978 genes.

3 Methods

We used the 'limma' R package [4] to perform differential expression analysis on each of the 7,130 mouse RNA-seq datasets based on the metadata downloaded from GEO. We extracted the log2 fold change (logFC) values for all genes and combined them into a feature matrix. We then performed Gene Ontology (GO) enrichment analysis on the differentially expressed genes using the 'GOfuncR' R package [5]. The significantly enriched GO terms ($p < 0.05$) were combined into a single matrix and were one-hot encoded to be used as features for the classification model. Finally, we normalized the feature matrix using the 'StandardScaler' function from 'Scikit-Learn' [6], which transforms the data to have zero mean and unit variance. The dataset was then split into a training, validation and test sets with 70%, 15%, and 15% of the data, respectively.

We used 'VarianceThreshold' from 'Scikit-Learn' [6] to remove low-variance genes from the data, leaving 21,953 genes in the feature matrix. We then visualized the data using PCA and t-SNE with 2 components each. PCA is a linear technique that preserves the maximum variance of the data, while t-SNE is a non-linear technique that preserves the local structure of the data.

We built a simple autoencoder using the 'Keras' library with 'TensorFlow' as the backend [7]. The autoencoder consisted of a fully connected, feed-forward neural network with one hidden layer and 2000 neurons. The hidden layer had a rectified linear unit (ReLU) activation function, while the output layer had a sigmoid activation function. We trained the autoencoder using the 'Adam' optimizer with a learning rate of 0.001 and a batch size of 128. The loss function was the mean squared error between the input and output. We trained the model to reconstruct the feature matrix (logFC values) for 50 epochs and evaluated its performance by calculating the loss on the validation set. Figure 1 shows the architecture of the autoencoder. To identify the optimal number of hidden layers and neurons, we performed a random search over the hyperparameter space. We also used the 'reduceLRonPlateau' callback to reduce the learning rate by a factor of 0.5 when the validation loss did not improve for 5 consecutive epochs. The model's performance was evaluated using the cross-validation loss on the validation set. The model was implemented in Python 3.9.16 through Google Colaboratory.

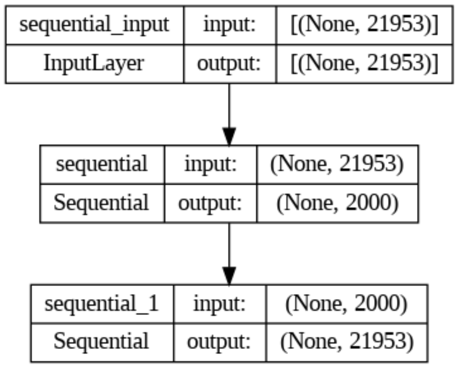
sequential_input | input: | [(None, 21953)]
InputLayer | output: | [(None, 21953)]
sequential | input: | (None, 21953)
Sequential | output: | (None, 2000)
sequential_1 | input: | (None, 2000)
Sequential | output: | (None, 21953)

Figure 1: Architecture of the autoencoder model.

4 Results

First, we performed principal component analysis (PCA) on the logFC values to visualize the data in a lower-dimensional space. The first two principal components explained 2.61% and 1.79% of the total variance, respectively. Figure 2a shows the scatter plot of the first two principal components. The points are colored by whether the GO0005622 (the most common GO term) term is enriched or not. No clear separation between the two classes is observed.

We then visualized the data using t-distributed stochastic neighbor embedding (t-SNE). Figure 2b shows the scatter plot of the first two t-SNE components. The points are colored by whether the GO0005622 term is enriched or not. Again, no clear separation between the two classes is observed.

Next, we needed to determine the optimal number of neurons and hidden layers for the autoencoder. To identify the optimal number of neurons, we trained the autoencoder with varying numbers of neurons in a single hidden layer; we evaluated the model's performance using the loss on the validation set. We tested the following numbers of neurons: 1, 10, 100, 200, 400, 600, 800, 1000, and 2000. Figure 2c shows the loss on the training and validation sets for each number of neurons. The loss on

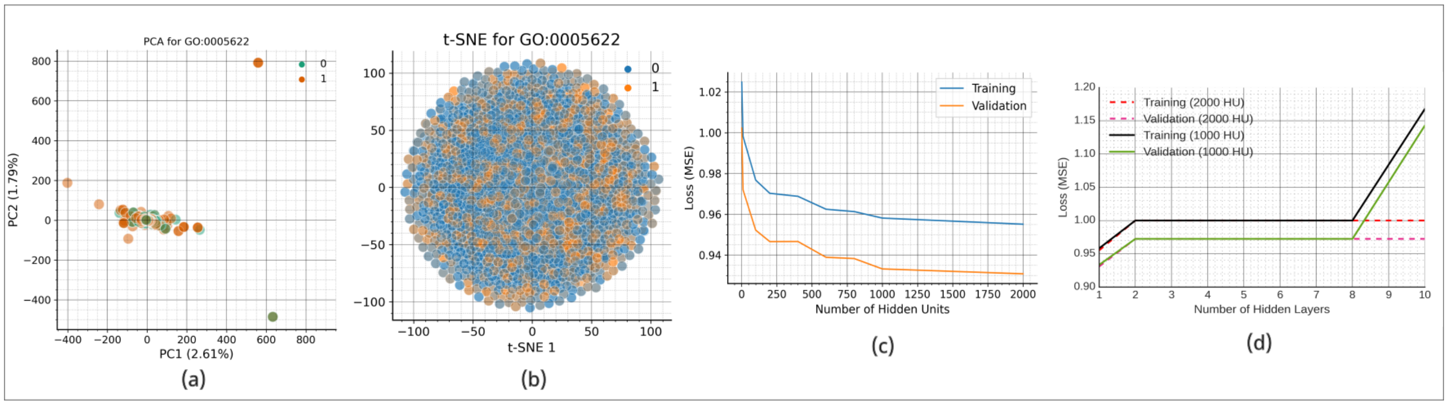

Figure 2: Results from evaluating the autoencoder. a) Scatter plot showing first 2 principal components of the data. Explained variances shown in axis labels. b) Scatter plot showing first 2 t-SNE components of the data. c) Autoencoder loss on the validation set with varied number of neurons. d) Autoencoder loss on the validation set with varied number of hidden layers.

the validation set decreased as the number of neurons increased, and plateaus at around 2000 neurons.

To identify the optimal number of hidden layers, we trained the autoencoder with varying numbers of hidden layers. The total number of neurons in the network was first fixed at 2000 (the optimal number of neurons from the previous experiment), and subsequently at 1000. The neurons were distributed so each hidden layer had half the number of neurons of the previous layer. We tested the following numbers of hidden layers: 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10. Figure 2d shows the loss on the training and validation sets for each number of hidden layers. The loss on the validation set increased as the number of hidden layers increased, with the lowest loss observed at 1 hidden layer.

Finally, we trained the autoencoder with the optimal number of neurons and hidden layers (2000 neurons in a single hidden layer) over 50 epochs. The performance of the final model was evaluated using 4-fold cross-validation. The average cross-validation loss was 0.9632.

5 Discussion

Our results provide insights into the use of autoencoders for unsupervised feature learning from gene expression data. The performance of the autoencoder was evaluated based on its ability to reconstruct the input data, and we believe that this is model captures the underlying structure of the data well. However, we acknowledge some limitations of our study. One major limitation was the memory constraints we faced during training, which limited our ability to use all 48978 genes. In addition, we were only able to evaluate a limited number of hyperparameters due to computational limitations. Future studies could explore more advanced architectures such as sparse and variational autoencoders to improve the performance of the model. Despite these limitations, our study provides a foundation for future work in this area. We plan to explore the use of the latent space of the autoencoder to perform feature extraction for a classifier to predict the enrichment of Gene Ontology terms.

In summary, our work highlights the potential of autoencoders for unsupervised feature learning from gene expression data, and provides a starting point for future studies to explore more advanced architectures and evaluate the performance of the model on predicting biological annotations. However, we acknowledge the limitations of our study and the need for further research to address these limitations and advance the field.

References

[1] C. M. Koch, S. F. Chiu, M. Akbarpour, A. Bharat, K. M. Ridge, E. T. Bartom, and D. R. Winter, "A beginner's guide to analysis of rna sequencing data," *American journal of respiratory cell and molecular biology*, vol. 59, no. 2, pp. 145–157, 2018.

[2] Y. Chen, Y. Li, R. Narayan, A. Subramanian, and X. Xie, "Gene expression inference with deep learning," *Bioinformatics*, vol. 32, no. 12, pp. 1832–1839, 2016.

[3] M. Ziemann, A. Kaspi, and A. El-Osta, "Digital ex-

pression explorer 2: a repository of uniformly processed rna sequencing data," *Gigascience*, vol. 8, no. 4, p. giz022, 2019.

[4] M. E. Ritchie, B. Phipson, D. Wu, Y. Hu, C. W. Law, W. Shi, and G. K. Smyth, "limma powers differential expression analyses for rna-sequencing and microarray studies," *Nucleic acids research*, vol. 43, no. 7, pp. e47–e47, 2015.

[5] S. Grote, *GOfuncR: Gene ontology enrichment using FUNC*, 2022. R package version 1.18.0.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.