Generating Whole Transcriptomic Profiles Using Compressed Sensing and Machine Learning

Kamal Obbad
SUNet ID: kobbad
& *AndraFehmiu*
SUNet ID: afehmiu
& *DaneHankamer*
SUNet ID: dhank

December 14, 2019

Project Category: General Machine Learning

1 Introduction

RNA profiles are informative for understanding various cellular types and processes. Unfortunately, methods for deriving genome-wide transcriptomic profiles are expensive and time consuming. Typically, measurements are taken for all ∼35,000 genes in order to generate a full gene expression profile. Interestingly, methods developed in compressed sensing (cs) were recently applied to accurately estimate genome-wide RNA expression profiles from a relatively small number of random composite measurements [2]. Since gene expression data is highly structured, cs methodologies promise to make transcriptomic profiling more time and cost efficient by acquiring compressed measurements and using those to recover structured, high-dimensional data– in our case, a gene expression profile with latent modular structure. Here, we propose to expand on this past work by implementing compressed sensing using generative models from neural networks as described in Bora et al [1]. By doing so, we aim to efficiently acquire gene expression levels via random composite measurements and show that similarity between pairs of expression profiles can be approximated with few composite measurements instead of being calculated from the full expression profiles.

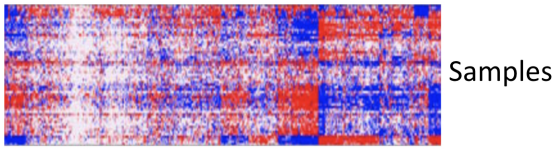
2 Related Work

Dagupta and Gupta, along with Johnson et al., prove that a small number of composite measurements can map to full gene expression levels using a random matrix Y [3] [6]. Specifically, mathematical results on low-dimensional projections suggest pairwise similarity of samples calculated in low-dimensional space Y can approximate their similarity in the original high-dimensional space. Their research validates our approach for mapping from a small number of measurements to the entire RNA expression profile. Hurley et al. outline a suite of tools for performing gene network inference on RNA abundance data. In order to better understand the structure our feature space and the connections between different genes and samples, we use one of their visualization techniques to detect correlation between specific genes across various samples in our highly structured data [5]. Cleary et al. describe state-of-the-art matrix factorization methods for acquiring high dimensional gene expression levels using composite measurements [2]. Their research largely serves as the basis for our project, as they employ Singular Value Decomposition (SVD), k-SVD, sparse Non-Negative Matrix Factorization (sNMF), and Sparse Modular Activity Factorization (SMAF) for the gene expression task. We seek to replicate their results using our dataset, as well as build on their models using neural networks. Bora et al. explore compressed sensing techniques that do not use standard sparse learning algorithms [1]. Instead, they use generative models to show that if the model is L-Lipschitz, then a sufficient number of random Gaussian measurements suffices for the same recovery guarantefes as sparsity techniques. We aim to use this method for our neural network to perform standard compressed sensing without sparsity constraints. Furthermore, Flenner and Hunter cleverly propose combining non-negative matrix factorization with a deep neural network in order to leverage the predictability and structure modeling of NMF with the power and accuracy of deep

neural networks [4]. We apply a similar stacking technique by feeding the results from our matrix factorization models into our neural network in hopes of building an even more robust model than the current state-of-the-art.

3 Dataset/Features

In our data collection and processing steps, we use the publicly available datasets from ARCHS4, which provide 167,726 samples of human RNA-seq data across all 35,000 genes [7]. However, in order to run our models in a reasonable amount of time, our subsample utilizes 10,000 RNA-seq samples (rows) and 500 randomly selected genes (columns). We perform minimal preprocessing to ensure that our methods are robust to the various approaches for measuring gene expression data. Thus, our preprocessing step only includes standardizing the data so that each gene expression falls in the range $[0, 1]$, where small values represent low expression levels of a particular gene and high values correspond to visible gene activity. Each of our models uses a 80/20 split for the training versus testing data, in which 8,000 samples are randomly chosen for the training set and the remaining 2,000 samples comprise the test set. We then perform gene network inference on our dataset to better understand the interactions between various genes. A visualization of this process is shown below:



As seen in the figure, there is a clear clustering pattern of genes with similar expression properties across the different samples. This non-random structure of the feature space allows us to confidently employ matrix factorization techniques and generative models to find near-optimal solutions to our task of generating full expression profiles.

4 Method/Models

4.1 Matrix Factorization

The aim of our project is to generate sparse and modular representations of gene expressions in low-dimensional space to then recover the full, high-dimensional gene expression levels. To do so, we apply various matrix factorization methods, such as: Singular Value Decomposition (SVD), sparse Negative Matrix Factorization (sNMF), and Sparse Modular Activity Factorization (SMAF), as well as machine learning techniques, such as: Neural Networks (NNs) and Generative Adversarial Models (GANs). These methods enable us to reconstruct approximate gene abundance profiles in original, high-dimensional space by "decompressing" few random composite measurement, which are linear combinations (or weighted sums) of gene expression levels, and finding latent, sparse representations of gene expression in low-dimensional space without accessing the full space of the original data. The weight of each gene expression level is nonzero and picked randomly.

Firstly, we generate a random, Gaussian measurement matrix A such that: $Y = A * X$.
We assume that each entry of our matrix A is Gaussian i.i.d. and that A $\in \mathbb{R}_{dxg}$, where $d = \#$ of random composite measurements and $g = \#$ of genes. Thus, A represents the weight of m random linear combinations of gene abundance profiles. In addition, X $\in \mathbb{R}_{gxn}$ where $g = \#$ of genes and $n = \#$ of samples. Thus, X represents the original expression values of the n samples in 35,238-dimensional space. Y $\in \mathbb{R}_{dxn}$ represents the samples in low-dimensional space and enables us to encode the transcriptome profile of a sample as a lower dimensional vector of random compositions, i.e. Y_i where i represents the i-th sample.
After obtaining Y, we recover the gene expression levels of the randomly picked composite measurements in the high-dimensional space, which is possible due to the sparsity of this high-dimensional data structure. This enables us to "decompress" the measurements by leveraging this latent structure and estimate $X_{testing}$, which is a matrix of predicted gene expression levels in 35,238 dimensional space. The procedure we followed to apply compressed sensing in recovering original gene expression levels via matrix factorization methods is as follows:

1. Compute a module dictionary U from training data, $X_{training}$ via matrix factorization.

$$U, W = X_{training} \tag{1}$$

2. Simulate random composite measurements Y on the test samples in low-dimensional space, where A is the random Gaussian measurement matrix.

$$Y = A * (X_{testing} + noise) \tag{2}$$

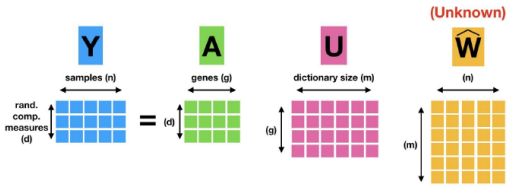
3. Obtain matrix representing module activity levels W using the low-dimensional gene abundance profile matrix Y, the measurement matrix A, and the module dictionary matrix U.

$$\hat{W} = Y * (A * U)^{-1} \tag{3}$$

4. Recover original high-dimensional gene expression levels of the randomly picked samples.

$$X_{testing} = \hat{W} * U \tag{4}$$

A graphical representation of this learning process is shown below, where we learn $\hat{W}$ in order to compute $X_{testing}$:



4.2 Generative Models

Next, we explored using generative models to reconstruct full gene expression data from composite measurements. As our baseline, we optimized a multi-layer perceptron to approximate gene expression from composite measurements. The MLP takes in measurement Y $\in \mathbb{R}_{dx1}$ and outputs predicted gene expression vector X $\in \mathbb{R}_{gx1}$. We trained this neural net by using full batch gradient descent to minimize mean squared error.
Additionally we attempted to train a GAN using full gene expression data that would produce a generator G: $\mathbb{R}_k \rightarrow \mathbb{R}_g$ where k is the dimension of representation space vector z. We would then reconstruct full gene expression from measurements by optimizing the following objective where A is the random measurement matrix and y is the random composite measurements for one sample:

$$loss(z) = \|AG(z) - y\| \tag{5}$$

Finally, we attempted to stack both our best performing matrix factorization technique, SMAF, with a generative model and see if that could improve performance. Using mean squared error, we trained a neural net using predicted gene expression data generated from SMAF as input and the actual gene expression values as target output.

5 Experiments/Results

For all our algorithms, we had two main metrics that we used to judge performance. 1) Pearson Correlation : This was used to determine if reconstructed gene expression correlated strongly with actual gene expression. We averaged Pearson correlation across all test data and judged our model off of that. 2)Sample-to-sample distance: We wanted to observe if reconstruction preserved sample-to-sample structure. First, we calculated pairwise euclidean distances between each reconstructed gene expression sample creating a $Z \in R^{nxn}$ where Z_{ij} is euclidean distance between samples i and j. Then we repeated the calculation for the actual gene expression samples. Lastly, we flatten the two matrices and calculate their Pearson correlation. Our results are summarized in Table 1.
To generate the module dictionary U, we perform matrix factorization using four methods: SVD(very similar to PCA taught in lecture), k-SVD, sNMF and SMAF (an algorith created by Cleary et al developed to fit the sparsity needs of this compressed sensing problem). This module dictionary consisting of 500 modules of 5,000 randomly sampled genes in our dataset is used in compressed sensing to recover the high-dimensional gene expression levels via step 4. described in the Method section above.
We imposed sparsity constraints in our matrix factorization algorithms to ensure that the resulting matrix representing the module activity levels W used in compressed sensing has exactly k nonzero values and thus, is column-wise

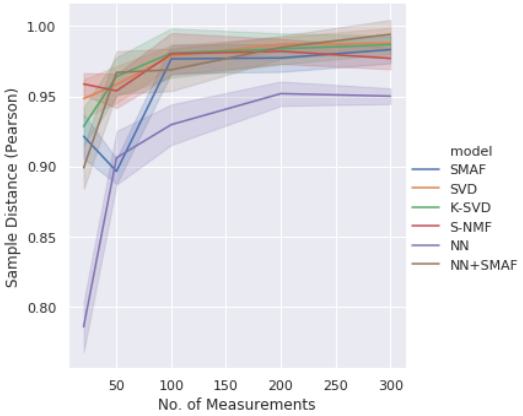


Figure 1: Sample-to-sample distance

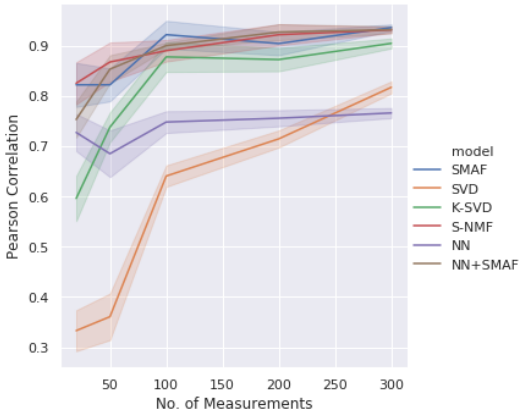


Figure 2: Average pearson correlation

k-sparse. We achieved this by setting k equal to 15, such that no more than 15 modules appear in a single, random composite measurement. Prior literature shows that k=15 works well for gene expression data. In SMAF, we ensured k-sparsity in the resulting matrix by identifying: 1). a non-negative, sparse module dictionary matrix U and 2). sparse module activity levels per sample in matrix W. The former is achieved by enforcing sparsity via an L1-relaxation while updating U using a Lasso non-negative regularizer when optimizing the objective function and the latter is achieved by implementing the Orthogonal Matching Pursuit algorithm while updating W, which finds a solution to $X_i = Uw_i$, such that $\|w_i\|_0 \leq k$, where k is the maximum sparsity level per sample. In SVD and sNMF, to enforce the k-sparsity, we kept only the k-vectors corresponding to the k-largest singular values using a truncated decomposition. Beyond k, our sparsity constraint, there were no other hyperparameter we had to optimize for our matrix factorization based approaches. We randomly split our data 20:80 into testing and training sets and ran the experiment many times to get accurate estimates of correlation and sample-to-sample distance along with error bars as shown in figure 1 and 2.
To train our neural nets, we randomly split our data 20:80 into testing and training sets. We then simulated taking random composite measurements with measurement matrix A as described in methods. For hyperparameter (learning rate and L2 regularization term) tuning we further split our training set 20:80 into a validation and training set. We then performed uniform grid search and found that a learning rate of .001 worked best along with λ of 1.0. All inputs and target output vectors used for training the neural net were standardized to have each element fall between [0,1]. For our testing data, we "un-standardized" our outputs using the standardization factors calculated from the training set.

Finally, we attempted to stack our neural net approach with our best performing matrix factorization technique, SMAF. We trained a neural net [architecture: (number of genes) - (500) - (500) - (500) - (number of genes), fully connected, ReLu hidden layer activation functions, identity output activation function] to take as input reconstructed gene expression vectors from SMAF and used the actual gene expression vectors as target output.

6 Discussion

For each of our models, we simulated taking d (d$\in \{20, 50, 100, 200, 300\}$) random composite measurements and tried to reconstruct full gene expression data. As seen in figure 1 and 2, increasing the number of random composite measurements led to better overall Pearson coefficients and more accurately preserved sample-to-sample distance after reconstruction, which is consistent with the fact that more measurements provides a better picture of the entire gene expression profile. SMAF was the most robust of all our matrix factorization methods and performed well even when taking a low number of random composite measurements.

As seen in the results in figure 1 and 2, SVD performs worse of all algorithms in compressed sensing. We expected these results because this matrix factorization algorithm does not ensure that the resulting module activity levels matrix W will have the necessary sparse structure to perform well in compressed sensing. However, adding the k-sparsity constraint significantly improves the performance of this algorithm, as seen with k-SVD. But, k-SVD does not perform as well as SMAF. This also makes sense because k-SVD represents samples with different activity levels

of the same 15 active modules corresponding to 15 eigenvectors with the largest eigenvalues (since k=15 in our case). In contrast, SMAF represents samples with different activity levels of different 15 active modules randomly chosen from the module dictionary U.

We found that our trained neural net was generally not able to out-perform matrix factorization approaches like SMAF and sparse NMF. However, when we combined our neural net approach with SMAF, we were able to out-perform our other models at higher measurements. For instance at 300 measurements, SMAF+NN produced an average Pearson correlation of .932 vs .930 for SMAF alone. Additionally at 300 measurements, SMAF + NN produced a sample-to-sample distance correlation of .99 vs .986 for SMAF. Though these results are not statistically significant, with more training data and CPU resources we may be able to significantly beat SMAF.

Lastly, as described in Methods, we spent a lot of time attempting to train a GAN that could be used to reconstruct gene expression data from random composite measurements and the random mixture matrix A. However, we found the GAN extremely unstable to train and were not able to have it perform reliably. Due to the nature of our data, we were unable to visualize GAN training and had challenges validating its performance and tuning it.

Table 1: Results

Table 2: Sample Distance

# of Measurements	20	50	100	200	300
SMAF	0.921	0.897	0.981	0.976	0.986
SVD	0.948	0.962	0.978	0.984	0.988
k-SVD	0.933	0.958	0.981	0.984	0.986
sparse NMF	0.953	0.960	0.975	0.984	0.982
NN	0.785	0.911	0.925	0.956	0.957
NN+SMAF	0.905	0.971	0.974	0.981	0.994

Table 3: Pearson Corr.

# of Measurements	20	50	100	200	300
SMAF	0.820	0.804	0.908	0.917	0.930
SVD	0.319	0.364	0.629	0.727	0.821
k-SVD	0.574	0.744	0.874	0.871	0.906
sparse NMF	0.825	0.864	0.906	0.916	0.929
NN	0.709	0.702	0.759	0.757	0.761
NN+SMAF	0.765	0.859	0.891	0.922	0.932

7 Conclusion/Future Work

We found that multiple matrix factorization techniques are able to perform very well for reconstructing gene expression data from random composite measurements. Gene expression data is highly structured so it makes sense that matrix factorization techniques that represent gene expression data sparsely are able to perform well. Similarly to how images can be reconstructed from a few measurements (due to being sparse in a fourier or wavelet basis), we show that relatively few measurements are able to accurately reconstruct high dimensional gene expression data. Today, for gene expression, state-of-the-art is SMAF. But, our results indicate that it may be possible to enhance the performance of SMAF by stacking it with a neural network.

Neural nets benefit greatly from large datasets. Due to CPU constraints, we had to train all of our algorithms on modestly sized training sets. As a next step, we can train our neural network on an order of magnitude more data and see how that impacts performance of both the standalone neural network and NN+SMAF. Additionally, we treated mean squared error (MSE) as a proxy for Pearson's correlation. Though this works, we also attempted to directly maximize Pearson's correlation as our objective function for training our NN. We found that using correlation as an objective function quickly led to training difficulties such as exploding gradients but with more time we may have been able to solve this. Lastly, Bora et al demonstrated how a GAN could be used for compressed sensing of images. Interestingly, despite a lot of effort, we were never able to successfully train a GAN. GAN performance is usually evaluated by human judgement which is difficult to do with gene expression data. Additionally, it is unclear what a GAN is learning. Our neural net minimized MSE which is related to Pearson's correlation coefficient but a GAN could be learning to model any of the structure within the gene expression data which may not necessarily strongly correlate with our chosen accuracy measurements. With more data, we could likely improve performance of our GAN and generator for reconstructing full gene expression.

8 Contribution

Kamal contributed to code to run baselines and experiments. He also reviewed the write up. Dane helped design and run the experiments, as well as created graphics for the report and contributed to the result interpretation. Andra prepared the milestone writeup and contributed to running the baseline model as well as interpreting the results. code: https://www.dropbox.com/s/48yi8gqsr0glb2u/cs229_project.zip?dl=0

References

[1] A. Bora, A. Jalal, E. Price, and A. G. Dimakis. Compressed sensing using generative models. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 537–546, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/bora17a.html`.

[2] B. Cleary, L. Cong, A. Cheung, E. S. Lander, and A. Regev. Efficient generation of transcriptomic profiles by random composite measurements. *Cell*, 171(6):1424 – 1436.e18, 2017. ISSN 0092-8674. doi: https://doi.org/10.1016/j.cell.2017.10.023. URL `http://www.sciencedirect.com/science/article/pii/S009286741731245X`.

[3] S. Dasgupta and A. Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. doi: 10.1002/rsa.10073. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/rsa.10073`.

[4] J. Flenner and B. Hunter. A deep non-negative matrix factorization neural network. 2017. URL `http://www1.cmc.edu/pages/faculty/BHunter/papers/deep-negative-matrix.pdf`.

[5] D. Hurley, H. Araki, Y. Tamada, B. Dunmore, D. Sanders, S. Humphreys, M. Affara, S. Imoto, K. Yasuda, Y. Tomiyasu, K. Tashiro, C. Savoie, V. Cho, S. Smith, S. Kuhara, S. Miyano, D. Charnock-Jones, E. Crampin, and C. Print. Gene network inference and visualization tools for biologists: Application to new human transcriptome datasets. *Nucleic acids research*, 40:2377–98, 12 2011. doi: 10.1093/nar/gkr902.

[6] W. B. Johnson, J. Lindenstrauss, and G. Schechtman. Extensions of lipschitz maps into banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, Jun 1986. ISSN 1565-8511. doi: 10.1007/BF02764938. URL `https://doi.org/10.1007/BF02764938`.

[7] A. Lachmann. Expression (gene level) human data, Jun 2019. URL `https://amp.pharm.mssm.edu/archs4/download.html`.