Denoising single-cell RNA-seq datasets using deep autoencoders

Jesse Zhang (jessez)

December 8, 2015

Introduction

With cancer rates around the world increasing every year, the need to better understand human biology and develop cheaper, more effective ways to combat disease is more important than ever. Luckily, in the past few years the advent of recent high-throughput sequencing technologies combined with modern computing power has allowed researchers to study gene expression in biological samples at the single-cell resolution. The function of an individual cell can be described by how much the cell expresses certain genes. Genes encode proteins, which accomplish cellular functions through biochemical reactions. The translation process from a gene (DNA) to a protein involves an intermediate step: forming an mRNA molecule, or a processed copy of the gene sequence. By identifying and quantifying all RNA molecules in a cell, a scientist can gain a better understanding of the cell's function. Single-cell RNA-seq is a novel technology that sequences all RNA molecules within an individual cell. For M cells and N genes, the sequences are processed to obtain gene expression levels summarized by a $M \times N$ design matrix. This matrix can be used to identify known or discover new cell subpopulations.

Task Definition

The purpose of this project is to leverage the power of deep neural networks to clean the noisy design matrix such that biologically meaningful clusters can be uncovered using simple unsupervised learning algorithms such as k-means clustering. Specifically, the method will receive two inputs: an $M \times N$ design matrix and k, the desired number of clusters. The method will output a set of M labels assigning each of the M cells to one of the k clusters.

Dataset

The dataset used in this project was obtained from a study that quantified the effect of the cell cycle on gene expression [2]. The dataset involved analysis of 8989 gene expressions in

182 individual mouse embryonic stem cells (mESCs). Cell cycle status was the only source of heterogeneity for the 182 mESCs, and using cell sorting techniques, the cell cycle status (G1, S, or G2M) of each mESC was identified a priori. The methods explored in this project will attempt to assign individual cells to a cell cycle stage using only the knowledge that there are three types of cells.

Evaluation metric

Since the true labels on the 182 cells were known, an evaluation metric for the performance of any method developed would be:

$$Score = \max_{l \in P(\ell)} \sum_{i=1}^{M} 1\{l_i = L_i\} \tag{1}$$

Here, ℓ is the set of M labels generated by the method, L is the true set of M labels, and $1\{l_i = L_i\}$ is an indicator function that equals 1 if $l_i = L_i$ and 0 otherwise. $P(\ell)$ is the set of all permutations of the unique clustering labels. For example, $P([1,2,1,3]) = \{[1,2,1,3],[1,3,1,2],[2,3,2,1],[2,1,2,3],[3,1,3,2],[3,2,3,1]\}$. In words, this evaluation metric attempts to maximize the matching between the cluster labels with the true labels.

Approach

In addition to suffering from sequencing errors and technical noise, samples within a single-cell RNA-seq dataset often come from experiments done on separate days under slightly different conditions. Because of these confounding factors, the greatest directions of variance in the combined dataset often capture experiment ID rather than something biologically relevant, such as cell type. Additionally, single-cell RNA-seq experiments capture the expression levels of thousands of genes. Direct clustering on cells in this high-dimensional space results in the curse of dimensionality where signal from a few genes are overshadowed by noise from all the other genes. Studies often tackle this problem by preselecting genes based on statistical tests and biological intuition, but an ideal method would automatically generate relevant clusters without the preselection step.

When given enough data, deep neural networks have been shown to effectively learn abstract features from data points. A deep denoising autoencoder attempts to describe a set of high-dimensional samples using a small set of these abstract features [4]. The autoencoder learns the (nonlinear) mapping from the high dimensional input space to the low-dimensional abstract feature space by attempting to map the features back to the input. The autoencoder minimizes the reconstruction loss between the output and the input:

$$\min \|x - \hat{x}\|^2 \tag{2}$$

where x is a data point and $\hat{x}$ is the same point reconstructed by the autoencoder after mapping it to a lower dimensional feature vector. Because single-cell RNA-seq datasets

often capture the expression levels of thousands of genes, perhaps a deep neural network could uncover the structure hidden in these thousands of data points. **An important concept here is the fact that rather than using the M cells as training samples, we can instead use the N genes (or features) to train the autoencoder**.

Model

In this project, the $M \times N$ design matrix X ($X_{ij} \geq 0$ represents the expression level of gene j in cell i) is modeled in two ways:

1. Each column of in X is assumed to be the nonlinear combination of a small number (k) of key features.

2. X is the sum of a sparse matrix S and a low-rank matrix U that solves the optimization problem [3]:

$$\min_{U} \|X - U\|_1 + \lambda \sum_{i,j=1}^{N} W_{ij} \|\mathbf{u}_i - \mathbf{u}_j\|_2^2 \text{ subject to } \text{rank}(U) \leq c \tag{3}$$

   W is a gene similarity matrix W where W_{ij} is the similarity between genes i and j. If two genes i and j are similar (W_{ij} is high), then $\mathbf{u}_i$ and $\mathbf{u}_j$, the corresponding columns of U, should have similar values. D is the degree matrix, a diagonal matrix where D_{ii} equals the sum of the ith row of W (or ith column, since W is symmetric).

Intuitively, the biologically relevant structure underlying X should be simple and hence low-rank. For Approach 2, the S term is assumed to be sparse to account for outliers. One reason these outliers may exist is because during the amplification of the RNA transcripts, some transcripts may just happen to be amplified at earlier stages, leading to an exponential overexpression of these particular molecules.

For both approaches, U captures some important gene-gene similarity relationships such as correlation. For example, consider a dataset with two types of cells: type A and type B. Each of these cells can be distinguished by a small set of genes: G_A and G_B. Cells of type A show greater expression of genes in G_A, and cells in type G_B show greater expression of genes in G_B. Therefore for type A cells, we expect positive correlation between genes in G_A, and for type B cells, we expect little correlation between genes in G_A (since they are not relevant to G_B). Because the similarity relationships amongst genes need not be linear, a deep neural network is the ideal choice for capturing these complex relationships. Approach 1 captures the similarity relations implicitly in the way the key features are nonlinearly combined to form the columns of X. Approach 2 captures the similarity relationships explicitly in W.

Algorithms

A deep denoising autoencoder is first trained using the N length-M data points (genes). Up to 5 layers (number of layers from input layer to middle hidden layer, including the input

and middle hidden layers) were tested. The denoising autoencoder is implemented using the *Keras* python package and trained using *Keras*' implementation of stochastic gradient descent and backpropogation. To prevent overfitting during training, each element in a length-M data point had a 0.2 probability of being set to 0 before the data point was used to train the autoencoder (dropout probability = 0.2). Four activation functions were tested: linear, sigmoid, rectified linear, and hyperbolic tangent. The trained autoencoder was used in two ways, based on the two approaches (models) described above.

1. Encode and decode each of the N points, resulting in a "clean" $M \times N$ matrix U. Noise, which lacks a pattern, will be shaved away because the autoencoder will fail to find a good way of representing the noise in the low k-dimensional space.

2. (a) Encode each of the N points, resulting in a $k \times N$ matrix $X^{(E)}$ (E for "encoded") where k is the number of neurons in the innermost hidden layer.

   (b) Use these "clean" representations of the genes to construct W where W_{ij} is the similarity between genes i and j ($\mathbf{x}_i^{(E)}$ is the ith column of $X^{(E)}$):

$$W_{ij} = \exp\left(-\frac{\|\mathbf{x}_i^{(E)} - \mathbf{x}_j^{(E)}\|^2}{2\sigma^2}\right) \tag{4}$$

   (c) Find U by solving the optimization problem described in Eq. 3. While this problem is not convex, we can approximate its solution by solving the relaxed optimization problem [3]:

$$\min_U \|U\|_* + \gamma\|X - U\|_1 + \lambda \sum_{i,j=1}^{N} W_{ij}\|\mathbf{u}_i - \mathbf{u}_j\|_2^2 \tag{5}$$

   where $\|U\|_*$ denotes the nuclear norm of U, or the sum of its singular values ($tr(U^T U)$). This now convex optimization problem can be solved using the alternating direction method of multipliers (ADMM) [1].

After obtaining a U matrix, or the denoised version of X, using either of the approaches described above, labels for the M cells are generated by using k-means clustering on the M rows of U. k-means assigns each of the M points to a cluster such that the average distance between points and their assigned cluster centers is minimized. In other words, k-means attempts to solve the optimization problem

$$\min_{z,\mu} \sum_{i=1}^{M} \|\mathbf{u}_i^T - \mu_{z_i}\|^2 \tag{6}$$

where $\mathbf{u}_i^T$ is the ith row of U, z_i is the cluster assignment of cell i, and μ_j is the mean of all points in cluster j. Due to the random nature of k-means (different centroid initializations can lead to different cluster assignments), the clustering was repeated 10 times, generating a evaluation metric distribution for each set of cluster assignments.

Baseline

As a baseline, the quality of the labels generated from clustering on the unprocessed $M \times N$ design matrix X will be scored using the evaluation metric defined in Eq. 1.

Oracle

The true assignments of cells to cell cycle stage are known from cell staining experiments performed by the creators of the dataset [2]. These true labels are used as an oracle.

Results

Baseline

Fig. 1 shows the results of applying k-means clustering directly to the unprocessed design matrix X. While this approach appears to get one of the clusters roughly correct (green cluster in Fig. 1a and blue cluster in Fig. 1b), Fig 1. shows how the visual structure in the other two labels (green and red in Fig. 1b) were lost by k-means. Fig. 1c shows how different runs of k-means can generate different scores according to the evaluation metric, but overall the distribution has significant mass around 0.65 and never exceeds 0.775. The fact that the k-means algorithm can generate heavily varying results demonstrates the lack of clustering structure in X. In summary, the evaluation metric to beat is 0.65, and ideally the distribution of the evaluation metric will have small variance.

Approach 1: Encode then decode

Several different hyperparameters were tested for constructing the deep denoising autoencoder. At two layers, the autoencoder only had one hidden layer, which was tested with 3, 5, 10, 20, 50, and 90 neurons. The linear activation function performed best here, achieving a mean evaluation metric of 0.72 when the hidden layer had 90 neurons. All other combinations of activation functions and number of neurons achieved evaluation metrics of below 0.62, which is worse than the baseline. Even the best scoring autoencoder was still in a relatively low quantile of X's evaluation metric distribution (Fig. 1c).

At three layers, combinations of 50 and 90 neurons at the first hidden layer and 3, 5, 10, and 20 neurons at the second hidden layer were tested. The best scoring autoencoder used 90 and 10 neurons at hidden layers 1 and 2, respectively, and scored 0.775. All other activation functions scored below 0.68. Deepening the autoencoder architecture appears to improve the clustering of the cells.

At four layers, combinations of 50, 80, and 120 neurons in hidden layer 1, 10, 20, and 40 neurons in hidden layer 2, and 2, 3, and 5 neurons in hidden layer 3 were tested. The rectified linear activation function performed poorly, never scoring above 0.63. The linear activation function scored up to 0.77. The hyperbolic tangent activation function performed

the best. At 120, 20, and 5 neurons for hidden layers 1, 2, and 3, the autoencoder managed to score 0.83 using the hyperbolic tangent activation function.

Finally at five layers, combinations of 100 and 120 neurons in hidden layer 1, 40, 75, and 80 neurons in hidden layer 2, 10, 20, and 30 neurons in hidden layer 3, and 2 and 5 neurons in hidden layer 4 were tested. The linear activation function could not surpass 0.76. The rectified linear and sigmoid functions both performed poorly. Several neuron combinations using the hyperbolic tangent activation function achieved scores over 0.8. The highest scoring autoencoder had 120, 40, 30, and 5 neurons in its 4 layers, and it achieved a scored of 0.87, a significant improvement over the baseline of 0.65.

As demonstrated by the experiments above, the linear activation function performs best when the autoencoder has a shallower architecture. As the depth increases, the hyperbolic tangent activation function produces significant improvement in the results. With the increased depth, the linear activation function fails to capture more abstract nonlinear features, while the nonlinear hyperbolic tangent activation function indeed captures some of these abstract features.

Fig. 2 demonstrates visually the improvement in using U, a clean encoded/decoded X, rather than the original unprocessed X (Fig. 1). The labels in Fig. 2a matches the labels in Fig. 2b quite well (blue, red, green in Fig. 2a to red, green, blue in Fig. 2b, respectively). The remaining errors stem from a few cells that were lost in other clusters. This may be due to two reasons: 1) the data from these cells were inherently noisy, and therefore their relevant information was lost before X was created, or 2) the autoencoder failed to capture the information in these cells in its low-dimensional encoding. Reassuringly, the k-means clustering for the best-performing architecture appears relatively robust. Compared to Fig. 1c, Fig. 2c shows how the evaluation metric is significantly less varying and always scores above 0.86.

Approach 2: Encode then find low-rank approximation

Because Approach 2 more explicitly modeled the relationships between genes (captured by a deep encoding), Approach 2 was expected to perform better than Approach 1 if the modeling assumptions were correct and worse otherwise. Due to the computational load of solving for U (182*8989 variables), only the best performing encodings from Approach 1 were considered. γ was held at 1, but various values of λ from 10^{-5} to 10^{6} were tested. As λ increased, the method would focus more on ensuring that U captures the similarities described in W and less on being low-rank.

As a new baseline, principle component analysis (PCA), the simplest method for projecting the length-M genes into a low-dimensional space, was used to construct the gene similarity matrix W_{PCA}. For values of λ from 10^{-5} to 100, various U were obtained from solving Eq. 5 using $W = W_{\text{PCA}}$. Fig. 3a shows how for all values of λ, the obtained U does as well as or worse than the original unprocessed X. Increasing λ results in worse performance of U, suggesting that the similarities captured by W_{PCA} were incorrect.

Surprisingly, the best autoencoder from Approach 1 performed poorly for all values of λ tested (Fig. 3c). This is perhaps due to the way W was constructed from the low-dimensional

encoding of X. Due to the depth of the network and the usage of the hyperbolic tangent activation function, the mapping from the 182-neuron input layer to the 5-neuron hidden layer was nonlinear. The Gaussian kernel (Eq. 4) used to calculate similarities was based on the norm of the difference between two columns of the encoded X. As shown in Fig. 3c, this was not the correct way to estimate the similarity. Incorporating this W into the optimization problem only decreases the performance of the obtained U.

Of the autoencoders tested, the best was a 3-layer autoencoder that used a linear activation function and had 90 and 20 neurons in its two hidden layers. At $\lambda = 10^5$, this autoencoder scored about 0.71 (Fig. 3b), which is still not particularly impressive considering the baseline was 0.65.

Conclusion

This project explored two approaches of using deep denoising autencoders to clean a noisy $M \times N$ design matrix X obtained from a single-cell RNA-seq experiment. From the experimental results above, the first and simpler approach of encoding and decoding X outperformed the second approach of explicitly incorporating gene similarities in an optimization problem. With the baseline scoring at about 0.65 according to the evaluation metric (Eq. 1), Approach 1 did fantastic, consistently scoring 0.87 and above when the right network architecture was used. Due to computational limits, Approach 2 could not be tested as exhaustively, but for all cases tested, Approach 2 rarely performed better than the baseline. This may be because Approach 2 handles the similarities between genes poorly compared to Approach 1, which nonlinearly maps the encoding back to something that's easier to handle for k-means. While the algorithms for Approach 2 were more complicated, Approach 1 was able to learn more complex models due to its deep and nonlinear architecture. Approach 2 forced a linear relationship between encoded genes and between U and X that may have been incorrect.

Future work

The methods explored above can be further tested on other single-cell datasets. Ideally, one would find a deep model that applies to all single-cell datasets save a few tweaks. Also, k-means is the most naive clustering method; more complex clustering methods such as spectral clustering and pre-processing steps such as singular value decomposition could be used to get better results.

References

1. Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1), 1-122.

2. Buettner, F., Natarajan, K. N., Casale, F. P., Proserpio, V., Scialdone, A., Theis, F. J., ... & Stegle, O. (2015). Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nature biotechnology*, 33(2), 155-160.

3. Cands, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. *Journal of the ACM (JACM)*, 58(3), 11.

4. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504-507.

Figure 1: PCA visualization of X, the unprocessed design matrix, colored using k-means generated labels (a) and true labels (b). (c) shows a histogram of the evaluation metric computed for 200 runs of k-means.

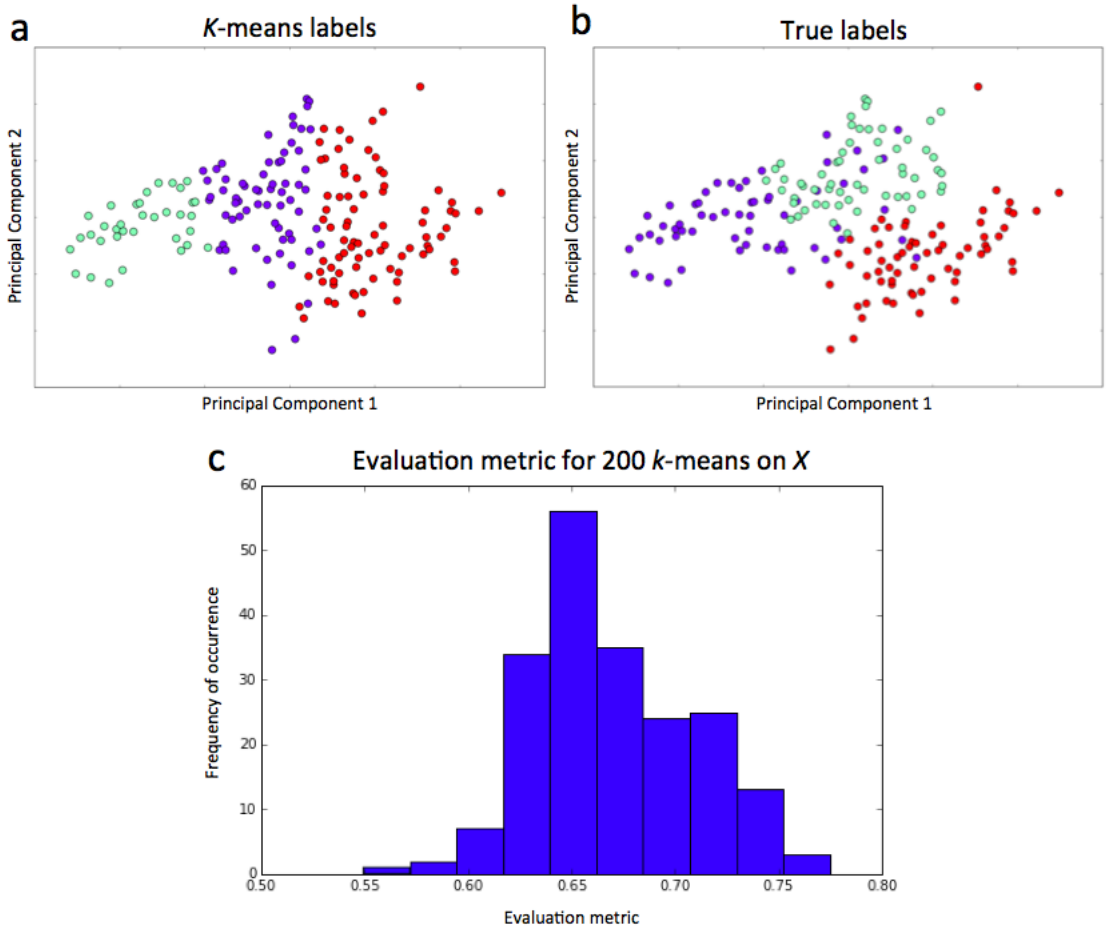

Figure 2: PCA visualization of U, the encoded then decoded design matrix X, colored using k-means generated labels (a) and true labels (b). The 4-layer autoencoder used a hyperbolic tangent function and had 120, 40, 30, and 5 neurons in its 4 layers. (c) shows a histogram of the evaluation metric computed for 200 runs of k-means.

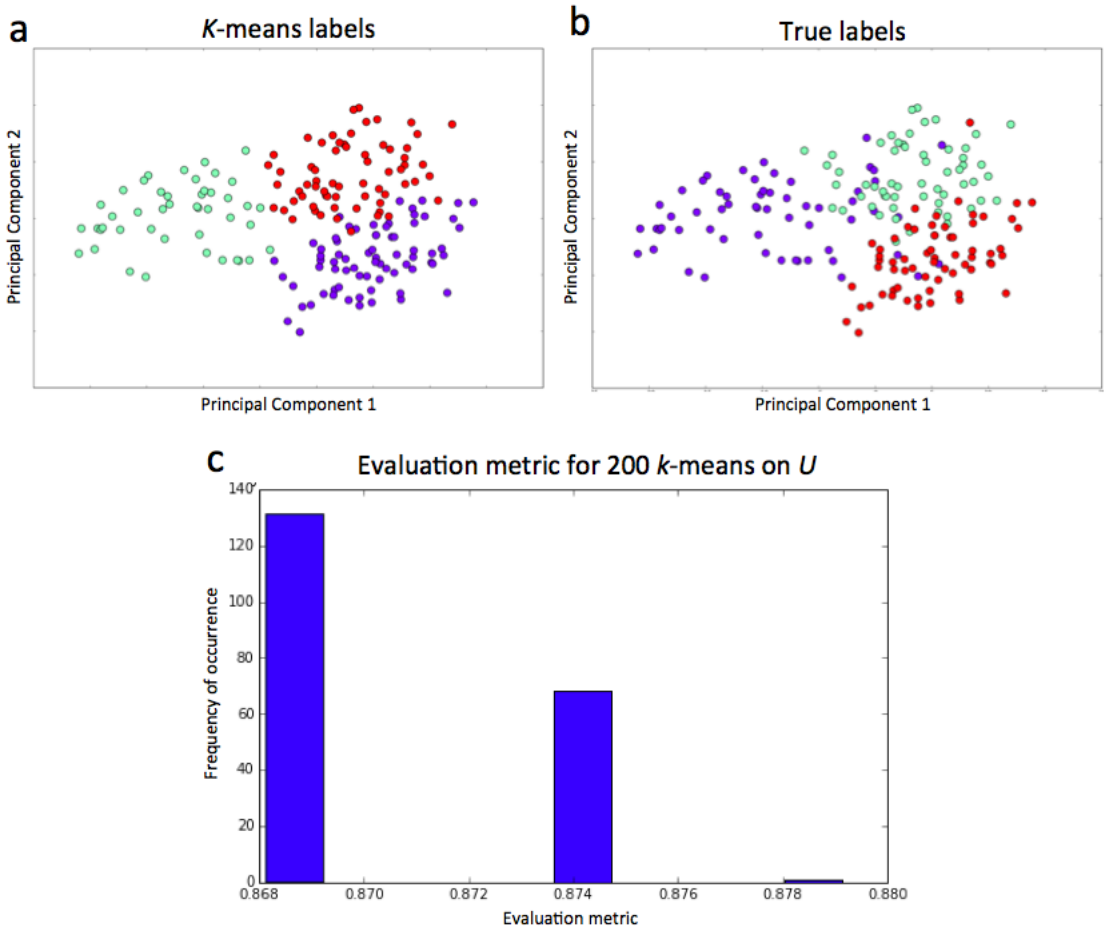

Figure 3: Plots of the performance of k-means (with error bars) on the U obtained from Eq. 5 when the similarity matrix W was obtained using (a) PCA of U into 2 dimensions, (b) 3-layer encoding of U into 5 dimensions (the hidden layers of the autoencoder had 90 and 20 neurons, and the autoencoder used a linear activation function), and (c) 5-layer encoding of U into 5 dimensions (the hidden layers of the autoencoder had 120, 40, 30, and 5 neurons, and the autoencoder used a hyperbolic tangent activation function).