Comparison of Unsupervised Clustering Models to Analyze Image-Based, High-Throughput Cell Screens with Subtle Phenotypes

Wei Cui, Ruixi Fan, Huajin Wang
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213
`wcui2, ruixif, huajinw @ andrew.cmu.edu`

Abstract

Data analysis for imaging-based, high-throughput phenotype screens has been challenging for biologists and the pharmaceutical industry. Here, we implemented unsupervised learning algorithms, including k-means clustering and Gaussian mixture models, and used these models to analyze a large, high dimensional dataset. The dataset was previously extracted from microscopy images obtained form a high-throughput gene knockdown screen for lipid droplet phenotypes. The extracted dataset contains parameters of ∼150 features describing variables in ∼450,000 images. Our goal is to identify data points that are either similar to one of the 3 known controls, or with new phenotypes that are different from all controls. To enhance the performance of our models, we experimented different distance metrics to improve the goodness of clustering, and implemented parallel threads to improve computation time.

1 Introduction and related work

One of the important technologies widely used by biologists and pharmaceutical industry is imaging-based high-throughput screening. It is a powerful tool to discover meaningful targets (eg. genes or drugs) that contribute to certain biological defects or phenomenon. In recent years, automated high-throughput microscopy has made it possible to obtain millions of images within a reasonable amount of time. However, a bottleneck for successful target identification is the downstream data analysis. In addition to the difficulty in image segmentation, post-processing of the high-dimensional data generated from segmented images is also challenging, especially when the goal is to discover new phenotypes instead of known classes.

Statistical analysis for this type of work has traditionally been done by calculating Z scores to find outliers, which does not capture diversity of phenotypes. Various machine learning techniques have been applied to analysing gene expression data, however, these methods do not necessarily work well for discoveries in cell-based phenotypic screens due to the high dimensionality of data points and high cell-to-cell variation. In a recent work, a clustering algorithm was reported using Support Vector Machines (SVM) to generate distance metric to find phenotype dissimilarities (4) with improved success compared to previous work. However, one disadvantage of this algorithm is that it is computationally expensive, which may not be applicable to the much larger, genome-wide datasets like ours.

In this project, we will extract information from a genome-wide screen for lipid droplet phenotype in a cultured human cell line where every given gene of a genome is absent one at a time (gene knockdown). Because our data is unlabeled except for controls, it is unlikely to apply supervised classification models. Thus, we mainly explored unsupervised learning algorithms, including hierarchical clustering, k-means clustering, and Gaussian mixture models, to analyse our data. Because the objects of interest are highly heterogeneous and phenotypes caused by the absence of a given gene is often subtle, we need to find a model that is sensitive enough to distinguish subtle differences, while fast enough to be applicable for standard biology labs that are often not equipped with super computers.

2 Dataset

The dataset were generate by Wang and colleges from her previous work (1). Images of cells were obtained from a high-throughput gene knockdown screen performed on cells grown in 384-well culture plates to study lipid droplet phenotypes . Each well contain cells with either no knockdown (negative control), knockdown of a gene with known phenotype (positive control 1 or positive control 2), or knockdown of a gene with unknown phenotype that needs to be explored (unknown sample) (See Figure 1 for plate layout.) The dataset covers genes representing the whole human genomes ($\sim$18,000 genes), with at least triplicate wells for each gene and 7 images from each replicate. A total of 174 plates were used, each containing samples, negative and positive controls. In total there are at least 21 images for each gene of unknown effect, $\sim$30,000 images for negative control ('neg', 6%), $\sim$10,000 for positive control 1 ('pos1', 2%), and $\sim$20,000 for positive control 2 ('pos2', 4%). The abundance of control samples provides an objective channel to verify the quality of clustering results.

By using an established open-source image analysis software, CellProfiler (Base version)(2), microscopic features of images, such as granularity, size and number of lipid droplets, texture, distance from neighbours, were extracted from images and evaluated by single values per image. The output is stored in multiple csv files containing total of $\sim$450,000 data points with dimension (features) about 150.

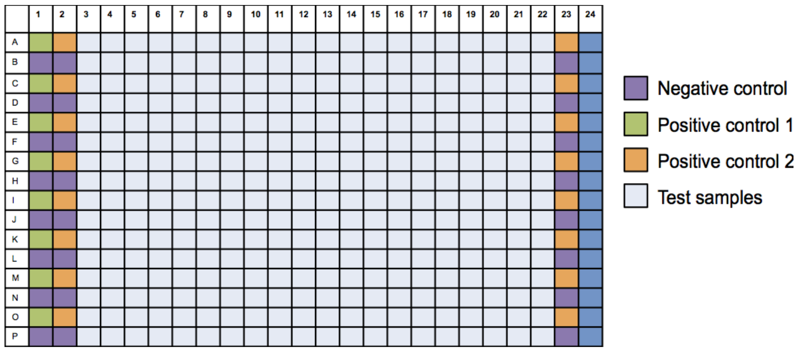

Figure 1. Sample layout on a 384-well plate. A total of 174 plates are included in the screen.

3 Methods and Results

3.1 Pre-processing

: As the raw files were automatically generated from CellProfiller in different batches and on two different equipments, significant pre-processing was required. We first unified data layout of individual output files and ensured that they have the same dimension. We then merged these files into a single csv file and trimmed columns that do not contain relevant feature information. Missing values were replaced with a small number ($epsilon = 10^{-6}$). To avoid interference from dominating features with especially large values, normalization was performed as:

$$z_{i,j} = \frac{(x_{i,j} - \mu_j)}{\sigma_j} \tag{1}$$

where $x_{i,j}$is the j th feature of the i th data point, μ_j is the mean and σ_j^2 is the variance of the j th feature. Additionally, for the convenience in parsing and array manipulation, we extracted gene symbols and corresponding line numbers, which were stored in a separate file.

3.2 Generation of training set

: To avoid laborious computation on the full dataset that is very large, we first generated a subset of randomly selected data points containing 1% of the full dataset to develop our models. We have confirmed that the composition of the dataset is representative of the full dataset. Once our models were tested to perform well on the subset, we applied them to the entire data for evaluation.

3.3 Dimension Reduction

: By first visually inspecting feature values, we think that the dataset may contain many potentially parallel features that do not provide additional information. Therefore, we used principal component analysis (PCA) to reduce dimensionality. The results suggest that more than 90% of the sample variance can be explained by the first 17 principal components (Figure 2). We used the output result from PCA for further analysis, which is expected to significantly reduce computation time and noise.

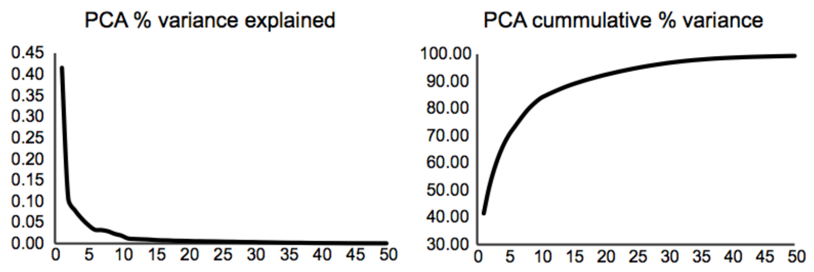

Figure 2. Percent variation explained and cumulative percent variation explained for PCA analysis.

We plotted data distribution after PCA with the first 3 components, which covers 60% of the samples variance. We found that the data are very densely populated, with few obvious outliers (Figure 3, left), even controls with known distinct phenotypes are not visibly separable (Figure 3, right). This suggest that significant difficulty will be expected in clustering our data.

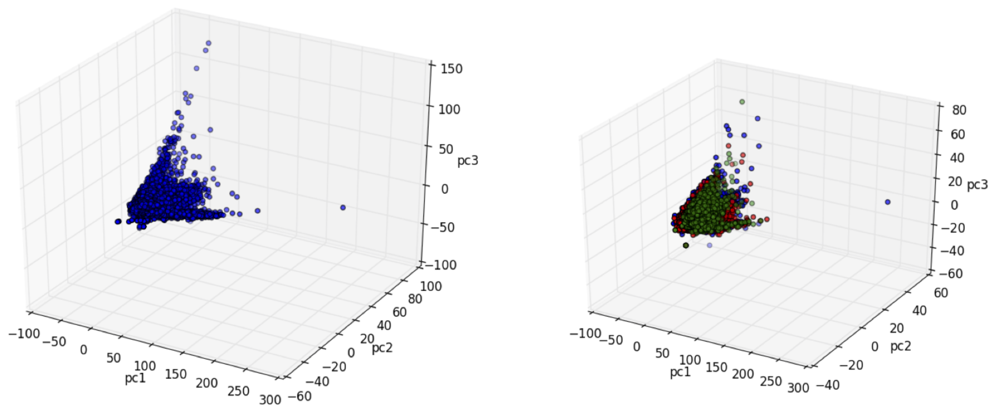

Figure 3. Sample distribution using the first 3 principle components. Left: all samples. Right: control samples. Red: ’pos1’, Green: ’pos2’, Blue: ’neg’.

3.4 Hierarchical Clustering

: We first implemented hierarchical clustering with complete-linkage criteria. To choose the optimal distance metric for hierarchical clustering, we planed to compare two different distance metrics between data points. The first is the basic euclidean distance as a baseline. We then attempted to implement a published method, where the distance metric is generated from pair-wise dissimilarity comparison based on classification accuracy by SVM (4). This method was reported to yield improved clustering result compared to methods such as PCA and factor analysis. However, one of the major drawbacks of using SVM dissimilarity distance metric is the high computational complexity. The time complexity to compute SVM dissimilarity distance metrics is $O(n^2)$. Computing hierarchical clustering using the generated metric runs at $O(n^2 logn)$. Thus, the total computation time for the whole analysis is $O(n^2 logn)$. Because the dataset in this publication contain only 779 genes, as compared to ours that has more than 18,000 genes, we speculated that the computation time required for this method may not be practical for our large genome-wide datasets with standard computing resource. In total we have about 430K data points thus it is not achievable to finish hierarchical clustering in a reasonable time. We conclude that with the current metrics, hierarchical clustering is not a practical algorithm for our data, and experiments confirmed out hypothesis.

3.5 k-means Clustering

: We plan to further find a computationally efficient algorithm with good clustering performance. k-mean is a NP-hard problems per se, however, efficient heuristic algorithms exist that lead to quick convergence. We implemented a commonly used k-means clustering algorithm (7) to cluster our data.

Initially, we have planed to optimize the algorithm by initializing with centroid positions obtained from hierarchical clustering, but since hierarchical clustering did not work for our data, we instead randomly initialized the centroids from a normal distribution, normalized to unit length. The algorithm was run multiple times and the outcome evaluated for quality. However, we could not obtain satisfying clustering result (see Evaluation of Methods section below for details), probably due to the high data density and noise.

We also experimented with different distance metrics to optimize our k-means algorithm, including euclidean, braycurtis, canberra, cityblock, correlation, and cosine. Unfortunately none of these distance metrics worked well. This is expected considering that random initialization failed to converge to a global optimum.

To obtain a reasonable clustering result as a baseline for our next algorithm, we used the python sklearn library to perform k-means clustering (8). The advantage of this algorithm is the application of k-means++ algorithm for centroid initialization, that leads to guaranteed convergence to a global optimum (9; 10).

An important factors to consider for k-means clustering is choosing the optimal number of clusters k. We first applied k-means with euclidean distance metrics and tried to find an optimal k using the elbow method. We analysed the total within-cluster variance of the clustering results with k values ranging from 2-20. However, no clear elbow point was revealed in the plot (Figure 3, left). We also tried to use Akaike information criterion (AIC) and Bayesian information criterion (BIC) (5) to estimate the goodness of clusters with different k values. Both AIC and BIC perform very similarly because the number of datapoints are much larger than the number of dimensions. BIC is calculated as follows:

$$BIC = n \cdot ln(\widehat{\sigma_e^2}) + k \cdot ln(n) \tag{2}$$

where $\widehat{\sigma_e^2}$ is sample variance, defined as:

$$\widehat{\sigma_e^2} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x_i})^2 \tag{3}$$

The plot for BIC is shown (Figure 3, right). Even though there are small roughness on the curve near k = 6 and k = 10, neither methods revealed an apparently optimal k.

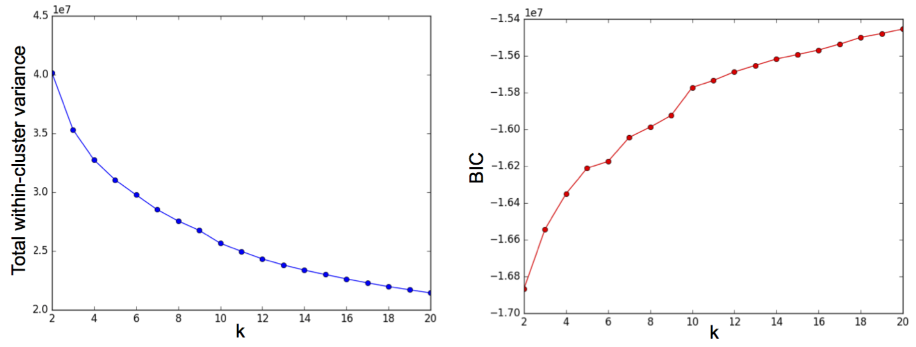

Figure 4. Left: Total sum of squares within clusters with increasing k values. Right: The BIC score of k-means clustering with increasing k values.

3.6 Gaussian Mixture Models

k-means is a special case of a mixture model, where some strong assumptions are made, such as the assumption that clusters are spherical and that each data point is assigned to a cluster with certainty ("hard assignment"). An alternative model that is similar to k-means but more generalized is the Gaussian mixture model (GMM). By allowing uncertainty in cluster assignments and incorporating covariance information, GMM is expected to perform better on complicated dataset like a biology phenotype screen. Additionally, GMM is guaranteed to converge, which would overcome the problem of getting stuck at local optima as with k-means.

Therefore, we implemented a GMM algorithm (6) to analyse our data. Using the evaluation methods shown below, we demonstrate that this model has improved performance compared to k-means, even when k-means++ initialization is use. The GMM algorithm we implemented is outlined as follows:

```
Initialize the means $\mu_k$, covariance $\Sigma_k$ and mixing coefficient $\pi_k$, where $k = 1...K$;
while Log-likelihood not converged do
    E-step: Evaluate the responsibilities for current data set:
        $$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \mathbf{\Sigma}_j)}$$
    M-step: Update the Gaussian parameters using the current responsibilities:
        $$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$
        $$\mathbf{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k^{new})(\mathbf{x}_n - \mu_k^{new})^T$$
        $$\pi_k^{new} = \frac{N_k}{N}$$
    where
        $$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$
    and Log-likelihood
        $$= \log p(\mathbf{X}|\mu, \mathbf{\Sigma}, \pi) = \sum_{n=1}^{N} ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)\}$$
end
```

One downside of GMM compared to k-means is that the improved performance is at the cost of increased time complexity. To reduce computation time, we implemented the GMM to execute in parallel using multiple threads. This parallel approach improved running time by 70% when executed on 4 cores, with computation time per iteration reduced from 4'30" to 1'20" for k = 6, and from 6'40" to 2' for k = 10, on a 4-core MacBook Pro. Thus, the total run time for the entire dataset to reach convergence was 4.5 hours for k = 6 ($\sim$ 200 iterations) and 33 hours for k = 10 ($\sim$ 1000 iterations). The computation time we achieved is realistic for a research lab with standard computing resource.

3.7 Evaluation of Methods

: We first evaluated the quality of PCA by calculating the percentage loadings on each principle components, as well as the cumulative percentage variance (Figure 2).

For clustering, it has always been challenging to evaluate the goodness of clusters. We tried a few commonly used methods such as calculating the total within-cluster variance, as well as AIC and BIC, to estimate the best k values to use, but did not succeed (Figure 3).

Because out data contain rich information including tens of thousands of control samples, we took advantage of these controls and developed a method to calculate the accuracy of our modes on positive and negative controls to assess whether data from the same known category cluster together. The goodness of clusters were then evaluated based on the following standards: 1) Percentage of all replicates of a given control or gene being assigned to the same cluster; 2) Whether the assigned label of a given control overlaps with that of another control.

According to these standards, the accuracy of label assignment is calculated as:

$$Accuracy = \frac{count\ of\ gene_i\ in\ cluster_k}{\sum_{k=1}^{K} count\ of\ gene_i} \tag{4}$$

The label assignment is then determined by majority voting:

$$Label(gene_i) = k;\ where\ k = argmax(count\ in\ cluster_k, for\ k = 1, ..., K) \tag{5}$$

We evaluated k values of 6 and 10 for both k-mean and GMM for their clustering performance according to these standards. The results show that both algorithms were able to separate 'pos2' from the rest relatively well, but the accuracy obtained from GMM is significantly improved, increasing from 0.55 to 0.72 with k=6, and from 0.47 to 0.70 with k = 10 (Figure4 and Table 1). However, both models had difficulty separating 'pos1' from 'neg'. This is not ideal but not surprising, because 'pos2' has a much more distinct phenotype, while the differences between 'pos1' and 'neg' are much more subtle and heterogeneous (can only be detected by well-trained eyes). For the task of separating 'pos1' and 'neg', k-means at k= 6 performed the best, with 0.34 for 'pos1' and 0.31 for 'neg'. Although GMM at k=6 achieved similar or slightly higher accuracy for these two controls, 'pos1' and 'neg' were wrongly assigned with the same label. Finally, using both methods, unknown samples almost always co-clustered with neg (Figure 4). This is expected as, due to the nature of genome-wide screens, over 90% of the unknown samples would behave similarly to the negative control).

Additional evaluation was also performed using selected unknown samples as test set. In the screen, the 'pos1' and 'pos2' are identical to 2 genes in unknown samples, known as 'FIT1' and 'BSCL2', respectively, each with 21 replicates. We therefore used 'FIT1 and 'BSCL2' as the first test set and examined the clustering accuracy of these two genes. Surprisingly, the results was much better than that obtained from 'pos1' and 'pos2', likely because less noise is introduced when less data points are included. At k = 6, the accuracy for 'FIT1' using k-means is 0.62, and using GMM is 0.86, while the accuracy for 'BSCL2' is 0.86 and 0.78, respectively. A higher accuracy score would correspond to a higher confidence in label assignment. Thus, it is possible that an imperial threshold can be set to determine the final label for unknown samples. As a second test set, we also randomly selected 10 genes from each cluster with clustering accuracy over 0.8 and visually inspected the original images where the dataset was extracted from. The results turned out mostly as expected, suggesting a decent performance of our models on strong phenotypes. However, this evaluation method is not applicable to untrained eyes, and is laborious. A more generalized and objective testing method is yet to be developed.

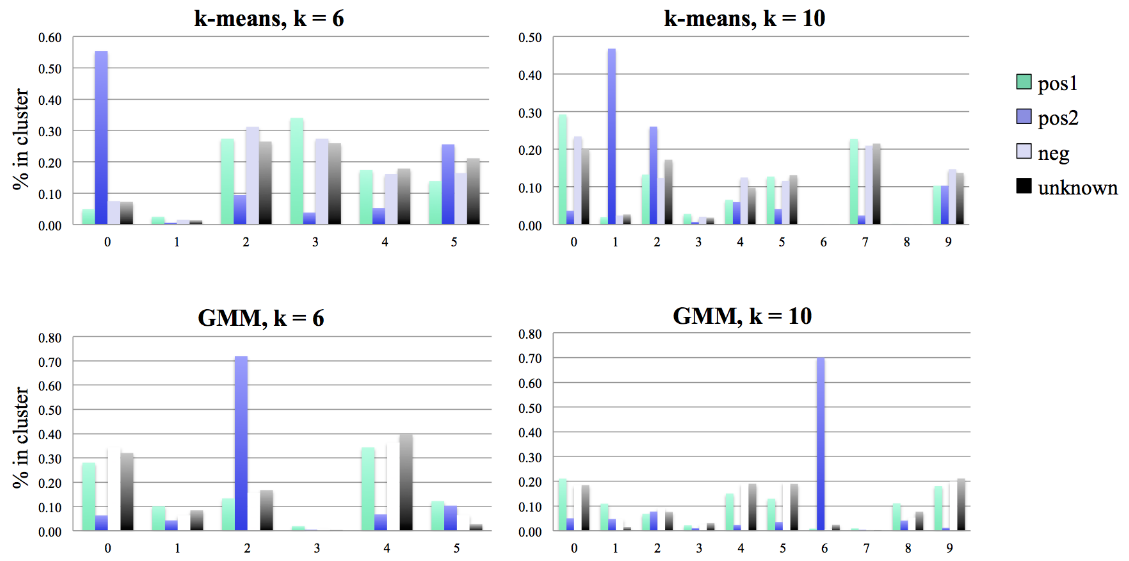

Figure 5. Comparison of label assignment accuracy with k-means and GMM. k values of 6 and 10 were compared using each model.

k = 6

k-means

	0	1	2	3	4	5
pos1	0.05	0.02	0.27	0.34	0.17	0.14
pos2	0.55	0.01	0.09	0.04	0.05	0.26
neg	0.07	0.01	0.31	0.27	0.16	0.16

GMM

	0	1	2	3	4	5
pos1	0.28	0.10	0.13	0.02	0.34	0.12
pos2	0.06	0.04	0.72	0.00	0.07	0.10
neg1	0.35	0.06	0.16	0.01	0.36	0.06

k = 10

k-means

	0	1	2	3	4	5	6	7	8	9
pos1	0.29	0.02	0.13	0.03	0.07	0.13	0.00	0.23	0.00	0.10
pos2	0.04	0.47	0.26	0.01	0.06	0.04	0.00	0.02	0.00	0.10
neg	0.23	0.02	0.12	0.02	0.12	0.12	0.00	0.21	0.00	0.15

GMM

	0	1	2	3	4	5	6	7	8	9
pos1	0.21	0.11	0.07	0.02	0.15	0.13	0.01	0.01	0.11	0.18
pos2	0.05	0.05	0.08	0.01	0.02	0.04	0.70	0.00	0.04	0.01
neg1	0.18	0.05	0.09	0.03	0.19	0.20	0.01	0.00	0.05	0.20

Table 1. Comparison of label assignment accuracy with k-means and GMM. k values of 6 and 10 were compared using each model. High-lighted cells indicate the predicted labels for each control. Orange: max value for 'pos2'; blue: max value for 'pos1' and 'neg'.

4 Discussion and Analysis

We implemented 3 different unsupervised clustering models to analyze a large dataset from a genome-wide, high-thoughput cell phenotype screen, and compared their performance, including running time and goodness of the clusters. The results demonstrate that both k-means and GMM are applicable to our data. k-means has the advantage of shorter running time, while GMM out-performs k-means in clustering accuracy, especially for strong phenotypes such as 'pos2'. It it not surprising that GMM performs better than k-means because it is a generalized model of k-means. By running parallel computing on a 4-core CPU, we saved computation time of GMM by 70%, which makes GMM a feasible model for information-rich, high-throughput data like ours. Another advantage of GMM over k-means is that the result of GMM does not depend on the random start. It is critically better for our highly dense data.

However, both algorithms had difficulty to separate more subtle phenotypes such as 'pos1' and 'neg'. A few intrinsic characteristics of the dataset makes the clustering task especially challenging:

1) There is great heterogeneity among samples, even among replicates with the same gene knockdown. This is is a common phenomenon in cell phenotype-based samples due to high cell-to-cell variation and image-to-image variation.

2) The dataset is densely populated. By inspecting the sample distribution, we confirmed that data points are clumped together and appear difficult to separate, even for controls (Figure 3).

3) Clusters are expected to have different sample sizes. It is expected that over 90% of the gene knockdowns would not have an effect on the phenotype and therefore the majority of the data points are similar to 'neg', while genes with new phenotype would only have a handful of instances. This hypothesis was not taken into consideration in our current model.

We propose a few approaches to further improve the clustering quality of our models in the future for subtle phenotypes.

First, to solve the problem of heterogeneity among replicates, we propose to first remove outliers for each gene before models are applied. Rather than taking the sample mean for each gene and risk loosing valuable information from replicates, we would keep only samples within a certain confidence interval.

Second, to address the vast difference in cluster sizes, more assumptions should be integrated into our model, for example, to assign clusters with different weights and add regularization terms. Alternatively, an outlier detection algorithm will need to be explored.

Third, a major challenge for unsupervised learning is the evaluation of the algorithm. Using our current method, the output for each gene is represented as the fraction of replicates being assigned to each $cluster_k$, and the final label is determined by the k value corresponding the the largest fraction. By using a high threshold, we were able to identify a cluster assignment with relatively high confidence. However, this majority voting strategy does not work well if one gene is distributed in 2 or more clusters with similar probability. It may be helpful to apply an additional mixture model that takes the output from GMM (the probability of being assigned to each label) as input to make deterministic label predictions.

Finally, the optimal k is yet to be determined for our analysis. One approach to consider is to use a sparse Dirichlet prior over the components, which will force the EM to choose the right k (11).

In summary, we compared the clustering performance of k-means and GMM on a genome-wide, cell phenotype screen, and achieved moderate success. We also optimized the running time of these algorithms to be more applicable to the real-world applications.

References

[1] Wang, Kuruvilla, et al. *Unpublished data.*

[2] CellProfiler (Broad Institute). `http://cellprofiler.org/`

[3] Kümmel, A., Selzer, P., Siebert, D., Schmidt, I., Reinhardt, J., Götte, M., … Gabriel, D. (2012). Differentiation and visualization of diverse cellular phenotypic responses in primary high-content screening. Journal of Biomolecular Screening, 17(6), 843–9.

[4] Zhang, X., Boutros, M., Boutros, M., Ahringer, J., Feng, Y., Mitchison, T., … Gentleman, R. (2013). A novel phenotypic dissimilarity method for image-based high-throughput screens. BMC Bioinformatics, 14(1), 336.

[5] Pelleg, D. Moore, A. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. Proc. 17TH Int. CONF. Mach. Learn. 727–734 (2000).

[6] Bilmes, J. A. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. (1998).

[7] http://stanford.edu/ cpiech/cs221/handouts/kmeans.html

[8] Pedregosa, F. et al. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12, 2825–2830 (2011).

[9] Arthur, D. Vassilvitskii, S. k-means++: The Advantages of Careful Seeding.

[10] Ostrovsky, R., Rabani, Y., Schulman, L. J. Swamy, C. The Effectiveness of Lloyd-Type Methods for the k-Means Problem.

[11] Görür, D. Rasmussen, C. E. Dirichlet Process Gaussian Mixture Models: Choice of the Base Distribution. (2010). doi:10.1007/s11390-010-1051-1