Single Cell Analysis Using Dimensional Reduction and SVM

Derun Gu, Zhengyu Chen, Zihao He
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
`{derung,zhengyuc,zihaohe}@andrew.cmu.edu`

Abstract

Different cells in human body express different subsets of genes at different levels, which empowers people to determine the type of a cell solely based on its gene expression profile using single-cell RNA sequencing *(scRNA-seq)*. This report shows an algorithm combining PCA and SVM to predict cell type out of its gene expression profile with an accuracy of 52.2%. Experiments indicate that the algorithm possesses the strong ability of generalization and the training and inference process is fast and efficient. Using model stacking approach to combine SVM with random forest and neural network, we can further achieve 53.5% accuracy.

1 Introduction and Background

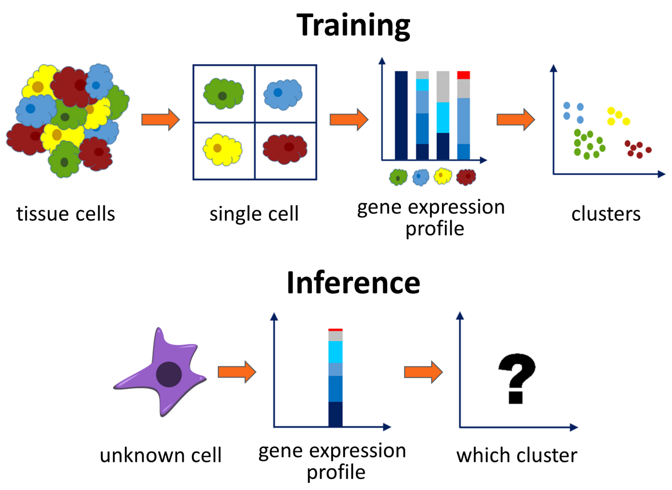


Figure 1: General view of training and inference process

Genes contain information of instructing how to build a molecule, and different cells in our body express different subsets of these genes at different levels. A technique named single-cell RNA sequencing (*scRNA-seq*) can be used to measure the activity levels of some annotated genes in a single cell, the result of which is termed the gene expression profile. The same type of cells (e.g. lung cell, brain cell, skin epidermis) are closely related to its gene expression profile and tend to have a similar distribution. Thus, this property makes it possible to predict the type of a cell solely from its gene expression profile. Accurate prediction is important in many cases. For example, the ability to tell the composition of a cancer biopsy sample has a huge impact on the type of treatment prescribed, and it's also helpful for identifying and characterizing new cell types and cell states[1, 2].

Given the training pairs of a gene expression profile and its ground truth cell type, the goal of the project is to classify the type of an unknown cell based on its gene expression profile. Specifically, the training input is a set of samples in some experiments denoted by $\{(\boldsymbol{x}_i, y_i) | 1 \leq i \leq N\}$ where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in C_k = \{blood\ cell, cardiac\ muscle\ cell, ...\}$. Cells from another set of experiments are further tested, denoted by $\{(\boldsymbol{x}_i, y_i) | 1 \leq i \leq M\}$ where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in C_{k'} \subseteq C_k$. In our setting, we have $N = 21389, M = 2855, p = 20499, k = 46, k' = 21$. Since the test set contains different experiments from the training set, the distribution may not be exactly the same. The classifier model itself must possess balanced abilities of learning and generalization.

2 Related Works

Amit Zeisel et al. [7] use scRNA-seq to perform a molecular classification of regions in the mouse brain. They developed *BackSPIN* algorithm, an unsupervised bi-clustering method based on sorting points into neighborhoods. Lin etc.[3] predict the label by one or two layered fully connected neural networks, some of the which incorporate prior biological knowledge, and can even correctly group cells not used in the training. Alavi etc.[4] implement a web server which can download, process and annotate publicly available scRNA-seq datasets online and employ fast matching methods to determine cell types of scRNA-seq data posted to the server. Jang etc.[5] experiment on several types of classifiers (e.g. SVM, random forest etc.) on the scRNA-seq data and it turns out that SVM has the best accuracy.

3 Dataset and Preprocessing

As specified in the first section, the dataset contains a training set and a testing set, each with 21389 and 2855 data samples correspondingly. The dimension of each sample is 20499. The numbers of classes in training and testing set are 46 and 21 respectively.

Due to the high dimension ($p = 20499$) of the original input, the training overhead is too large and is prone to be affected by unrelated noises. Hence, we first apply a dimensional reduction algorithm on the datasets to extract key information out of the input. We also do normalization on the low-dimension data before fitting into any model to remove the scale effect.

We choose Principal Component Analysis (PCA) as our final choice of dimensional reduction and record the first 40 principal components of the training set. Then the mean and standard deviation of the PCA output are stored. For training and testing, we project the data onto those 40 principle components and apply normalization according to the saved values.

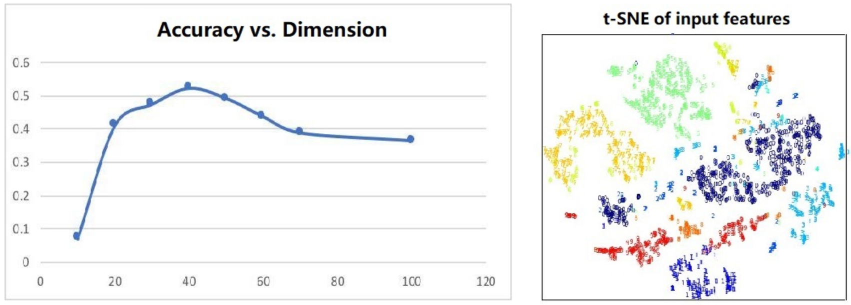


Figure 2: On the left is the accuracy plot after SVM vs. dimension of the PCA output. On the right is a 2D t-SNE representing the plot of the first 10 classes after PCA.

We also tried other reduced dimensions. However, in experiments, we found that when the dimension is small, we can lose some important information, and when the dimension is large, some unrelated information encoded by minor principal components can mislead the classification. By tuning carefully, we choose 40 as our final choice. The trend is shown on the left in figure 2. On the right shows the visualization of the output. Different colors represent different classes and it shows that

PCA maintains the spatial locality within each cluster, thus linear projection is enough for further classification. We have also tried non-linear dimensional reduction methods like Locally Linear Embedding (LLE) and auto-encoder but failed to observe a significant improvement.

4 Methods

The single cell analysis pipeline consists of data preprocessing, classification, and model stacking. The preprocessing part has been introduced in details in section 3, therefore this section will mainly focus on our classifiers and stacking approach. The best accuracy of a single model we have achieved is based on SVM. We also combine other classifiers including random forest and neural network to produce a better ensemble model.

4.1 Support Vector Machine

The input dimension is now 40 and 20k data samples are still relatively sparse in the space. SVM is suitable for this kind of problems as it assumes that there exists a hyperplane separating the input data into different classes, which is easy to be done in high-dimensional space. Here we employ one-vs.-rest scheme and RBF kernel for multi-classification. The final label is the one with the largest margin to the decision boundary, which means the data point is more likely to be on this side. Note that although it is not typically what we train the SVM for, it often works well in practice. And in this case, it's especially important to normalize the input first because SVM is sensitive to the input scale to compare the margin distance.

4.2 Random Forest

We tried a series of methods based on decision tree and found that using random forest on lower-dimension data works the best. The detailed analysis is as below. The experiment results can be found in this section and also section 5.2.

4.2.1 Decision Tree

Decision trees are commonly used in the classification problem. It is proved to have a high fitting ability and is easy to train. Here we use the raw high-dimensional data as our input and the output is one of the 46 classes. There are still some other details need to consider.

- Splitting Criteria. The most common two splitting criteria are information gain and Gini impurity. We have tried both and did not observe a significant difference in accuracy. The theoretical computational costs for both are also the same. However, since entropy needs to compute logarithm while Gini only needs to compute square, Gini impurity runs faster in experiments, and we adopt it as our final choice.
- Early Stop. When investigating our decision tree model, we found that there can be some internal nodes having very few samples and are still splitting, which leads to overfitting. In order to avoid this, we set the minimum sample size to 100 for any internal nodes and avoid further splitting on nodes with too few samples.
- Path Pruning. It is possible that even a node contains a lot of samples, after splitting, one of its children has very few samples. To avoid this, we prune the whole path when a leaf node contains too few samples.

The basic decision tree model achieves 33.6% accuracy. After implementing all these techniques, the accuracy goes up to 38.2%.

4.2.2 Random Forest

Random forest is a method of dealing with high dimensional data with a limited sample size. It builds many decision trees, each of which focusing on only a small part of the features. The final output is produced by aggregating all results using a method like a majority voting. This is extremely effective in preventing overfitting.

In this problem, we build 100 decision trees and each of them using 200 features. Each tree uses the same techniques we mentioned above in section 4.2.1. The accuracy further goes up from 38.2% to 41.6%.

Noticing that dimension reduction helps to extract the most useful part of the data, we also build a random forest on the pre-processed low-dimension data. Since the dimension is smaller, each tree now selects 20 random features. This leads to an approximate 4% increase and achieves 46.0% accuracy.

4.2.3 Gradient Boosting Decision Tree and Adaboost

We further tried to use gradient boosting decision tree and Adaboost to improve our decision tree and random forest performance. However, since the testing data has a different class distribution from the training data, and the training dataset size is also not large enough, these methods tend to overfit on the training data. Adaboost shows a gradual decrease in testing accuracy with more iterations, while GBDT shows limited improvement based on the decision tree baseline.

4.3 Neural Network

Deep Neural Network is a kind of method which is broadly used in recent years. It has shown great improvement in wide areas like image recognition, neural language model etc. Thus we want to try whether this works equally well on this specific task. For this specific task, our input data is gene sequence, and our output is the prediction of one of the classes. There's little locality founded in the sequence of the genes (at least not found with our biology knowledge), therefore we use fully-connected networks to train the classification model. Due to a large amount of data, we use AWS to accelerate our computing.

A baseline is created using a simple one hidden layer network, which has a test accuracy of 33.7%. Further experiments are done using different sets of parameters. Moreover, we think that some features given might be relatively less useful but do increase our computational burden and mislead the classifier. Therefore, we also test the performance of the classifier by choosing different amounts of features.

4.4 Ensemble Stacking

We have trained a small neural network for model stacking. The input is the concatenated probability vectors generated by the decision tree, neural network and support vector machine models. There is one hidden layer with exactly the same number of neurons as the input. The output is a 46-dimension vector which produces the final probability using softmax. The network is trained on training data. However, when we evaluate the model on testing data, we found that the performance does not become better than using the single SVM model.

After investigating the results, we notice that the training strategies of SVM are different when with probability and without probability, which means the label with the highest probability may not always equal to the predicted label without using probability. In this specific problem, SVM model without using probability gives a 3% higher accuracy than the model using probability, and our stacking approach is using the SVM with probabilities. Therefore we also add a one-hot encoder with the SVM predicted label at the end of the input vector. The new stacking approach gives 1.3% increase in accuracy compared with a single SVM model.

We also tried other ensemble methods at the same time to combine models and produce a classifier with higher accuracy. The simplest approach is to use probability weighting. We use a grid search to determine the best weight of each model on training samples. The resulting ratio is random forest: neural network: SVM with probability: SVM without probability = 2: 1: 3: 8. This means that the SVM model without using probability (which is also the one with the highest single model accuracy) dominates the other three models. The accuracy increase is also small ($\approx 0.6\%$). By further investigating, we found that if the single SVM prediction is correct, then we can simply follow this prediction. However, if SVM prediction is not correct, then the other three probability models tend to make similar mistakes. This results in the confusing ratio, which makes single SVM dominate the final output. Compared with the stacking approach, this ensemble method seems to be a simplified version of it. Therefore we adopt the model stacking approach described above as our final choice.

5 Experiments and Results

This section demonstrates our main results in a compact way for each part of our model. Comments and design justifications of our model are also provided along with the results.

5.1 Support Vector Machine

Table 1: Accuracy vs γ

γ	Accuracy
0.001	44.4%
0.005	52.0%
0.01	**52.2%**
0.05	45.7%
0.1	40.1%

Table 2: Accuracy vs C

C	Accuracy
0.01	14.4%
0.1	44.1%
1	**52.2%**
10	44.0%
100	40.8%

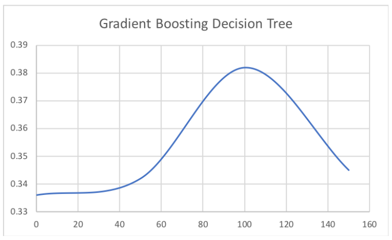
Table 1 and table 2 show the accuracy under different hyperparameters γ for the RBF kernel radius and C for the penalty weight. C is fixed at 1 with regard to Table 1 and γ is set to 0.01 in Table 2. We finally choose $\gamma = 0.01$ and $C = 1$ and achieved $\mathbf{52.2}\%$ accuracy on test set. The experiments are implemented using *scikit-learn*[6] in Python. The training and testing process takes around 30 seconds and 3 seconds.

5.2 Random Forest

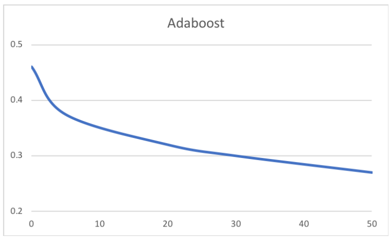
Table 3: Comparison Between Decision Tree Based Methods

Method	Accuracy
Decision Tree Baseline	33.6%
DT with Early Stop	36.5%
DT with Early Stop and Path Pruning	38.2%
Random Forest	41.6%
PCA + Random Forest	**46.0%**

The detailed analysis of decision tree, random forest and PCA has been introduced before in section 3 and 4.2. The above table gives a uniform view of how well they perform. As we can see that applying early stop and path pruning to decision tree avoids overfitting and increase its generalization ability. Using random forest further improves it and has a better performance in the testing dataset. With the pre-processed data using PCA, there is another significant improvement in accuracy. All these techniques have proved to be helpful in increasing the model's generalization ability.



(a) GBDT with 150 iterations. We calculate the accuracy every 25 iterations and the peak accuracy is about 38%, 5% higher than decision tree baseline



(b) Adaboost + Random Forest with 50 iterations. We calculate the accuracy at iteration 5, 20, 30, and 50. The accuracy keeps decreasing

Figure 3: Accuracy of boosting methods with number of iterations.

Boosting methods including Adaboost and Gradient Boosting Decision Tree are not very helpful in this problem setting since they usually work well when training and testing data have similar distribution and training dataset is large enough to prevent overfitting. Additionally, since the accuracy using decision tree is not very high, it also restricts the performance of GBDT which is based on it.

5.3 Neural Networks

Table 4: Accuracy using different architecture

Layer	Neuron	Accuracy
1	50	33.7%
2	50	35.9%
3	50	39.2%
4	50	38.8%
1	100	35.2%
2	100	37.4%
3	100	**40.3%**
4	100	40.1%
1	150	34.9%
2	150	37.0%

According to the experiments, we can find that generally deeper networks give us better performance when the network has few layers, but this is not guaranteed. Experimentally, best results are found when we have about 3 hidden layers. In the above table, 3 hidden layers with 100 neurons each give us 40.3% accuracy.

Table 5: Accuracy using different number of features

Feature Number	Layer	Neuron	Accuracy
100	3	100	21.3%
200	3	100	27.7%
500	3	100	28.6%
1000	3	100	32.7%
5000	3	100	36.9%
10000	3	100	**41.2%**
all	3	100	40.3%

In this experiment, we test the performance of a 3-hidden-layer network with different numbers of randomly picked features. It does not work well at the beginning when there is only a very small number of features. With the increase in the number of features, it generally has better accuracy. However, we can see that we do not get the best performance when using all features, this proves that some dimension of features are redundant and can make our classifier overfit the data.

Based on the above experiments, we found that for Neural Network method, the system reaches the best performance when the network is of middle size (small if compared to the huge networks used in the industry nowadays). This may partially due to the fact that we have more than 20k dimension of features but we only have about 20k samples as at the same time. So the network may easily overfit when it becomes too large. Additionally, some noisy information also affects the performance of the system.

5.4 Ensemble Methods

Table 6: Comparison of Ensemble Methods

Method	Accuracy
SVM	52.2%
Random Forest	46.0%
Neural Network	41.2%
Stacking	**53.5%**
Averaging	52.8%

From the above table, we can see that using a shallow neural network for probability stacking can produce better ensemble results compared with averaging probabilities of three models. The ensemble model achieves slightly better accuracy than the single SVM model.

6 Conclusion and Future Work

We presented three different methods to classify the type of a single cell based on its scRNA-seq expression. The best single classification model is built based on SVM and achieves 52.2% accuracy. First, PCA is used to extract the most important information and to accelerate training, and the lower dimensional data is normalized using Z-scores. The RBF kernel SVM with the one-vs-rest scheme is imposed for multi-class classification in this model.

Besides SVM, we also tried random forest and neural networks. The random forest approach shows good generalization ability with the help of dimensional reduction and overfitting prevention techniques. The Neural Network method reaches its best performance when we have about 3 hidden layers with 10k randomly picked samples. The data is insufficient so this may partially account for why NN does not work as well as SVM or random forest in our experiments.

At the same time, we also designed a combined model which achieves an accuracy of 53.5%. The model stacking approach makes use of all three models and produces a slightly better model than each of the individual one as expected.

In the future, there are several things to be examined.

1. Using hierarchical classification. With more domain knowledge, we can first group cells to some hyper-classes and build another model to predict the hyper-class first. Then we can train and use our current model in that hyper-class to do the final prediction.
2. Ensemble methods such as AdaBoost on SVM will be tested to improve the accuracy. Since both Adaboost and SVM are initially used for binary classification, it is not easy to combine them and let them work well in a multi-class setting. Some existing algorithms and software packages can be a good start point for our next-step exploration.
3. Noticing the positive effect of dimensional reduction on SVM and random forest, we will try to merge PCA with NN to see whether it can result in a better performance.

References

[1] Darmanis,S., Sloan,S.A., Zhang,Y., Enge,M., Caneda,C., Shuer,L.M., Gephart,M.G.H., Barres,B.A. and Quake,S.R. (2015) A survey of human brain transcriptome diversity at the single cell level. Proc. Natl. Acad. Sci. U.S.A., 112, 7285–7290.

[2] Poulin,J.-F., Tasic,B., Hjerling-Leffler,J., Trimarchi,J.M. and Awatramani,R. (2016) Disentangling neural cell diversity using single-cell transcriptomics. Nat. Neurosci., 19, 1131–1141.

[3] Lin,C., Jain,S., Kim,H., Bar-Joseph,Z. Using neural networks for reducing the dimensions of single-cell RNA-Seq data. Nucleic Acids Research, Volume 45, Issue 17, 29 September 2017, Pages e156, `https://doi.org/10.1093/nar/gkx681`

[4] Alavi,A., Ruffalo,M., Parvangada,A., Huang,Z., Bar-Joseph,Z. scQuery: a web server for comparative analysis of single-cell RNA-seq data. bioRxiv 323238; `https://doi.org/10.1101/323238`

[5] Jang,H., Al-Saffar.S. Classifying Single-Cell Types from Mouse Brain RNA-Seq Data using Machine Learning Algorithms. `http://hankyujang.com/Papers`

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

[7] Zeisel,A. et al. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. Science 347, 1138-1142 (2015).