Classifying Single-Cell Types from Mouse Brain RNA-Seq Data using Machine Learning Algorithms

Hankyu Jang
Indiana University Bloomington
School of Informatics and Computing
hankjang@indiana.edu

Samer Al-Saffar
Indiana University Bloomington
School of Informatics and Computing
sialsaff@umail.iu.edu

Abstract

The cerebral cortex carries the cognitive and sensory functions of the mammalian body as well as any social behavior. Normal brain function depends on a variety of differentiated cell types, such as neurons, glia, and vasculature. Definitive identification of these cell types has proven to be more difficult than other cells of the mammalian body due to their high specialization. Here we attempt to classify cell types from RNA-Seq data obtained from brain cells of mice using machine learning approaches. We are using four different supervised machine learning algorithms, and our goal is to determine which one is the best for classifying untagged cells.

Keyword: Single-Cell RNA-Seq, Support Vector, Random Forests, KNN, Neural Network

Introduction

The brain is the most complex organ in all vertebrates and most invertebrates, its cortex carries out highly specialized cognitive functions of the nervous system. It consists of highly differentiated cells that run all the electrophysiological functions besides regulating other demands of the brain, such as nutrients and energy supply, waste disposal, immunity needs, and also serves as a protective framework. While these cells have all been classified according to their location, function, molecular markers, morphology, and heterogeneity, there is no classification system where one cell can be definitely identified under all studied properties (1). A more recent approach for classification is to utilize the advancement in single-cell biology, and particularly when applying RNA-Seq techniques, as these proved to be a powerful discriminatory tool to differentiate cells types (2). This method has been successfully applied to characterize cells in other organs. The brain, however, exhibits more complexity and variety of cells and thus it is still a challenge to classify these cell types both experimentally and computationally.

The field of machine learning has been widely utilized in the many applications of genetics and genomics, a review of these can be found in reference (3). In this study, we attempt to classify brain cortex cells in mice by applying different machine learning algorithms on quantitative RNA-Seq data obtained from a previous study by Amit Zeisel et al. We used four supervised algorithms with labeled training data set and unlabeled test data set. Our goal is to test the suitability of these different algorithms in obtaining the "best" classification of cell types. The four algorithms utilized in this study are: Support Vector Machine using four types of Kernels, Random Forests, K-Nearest Neighbor, and Multi-Layer Perceptron (Neural Network).

Related Work

Amit Zeisel et al. used quantitative single-cell RNA-Seq to perform a molecular classification of the primary somatosensory cortex and the hippocampal regions in the mouse brain, based on 3005 single-cell transcriptomes. Individual RNA molecules were counted and tagged using unique molecular identifiers, and confirmed by single-molecule RNA fluorescence in situ hybridization. For computational analysis, they used clustering to discover molecularly distinct classes of cells. However, standard hierarchical clustering resulted in fragmented clusters because most genes were not informative in most pairwise comparisons and contributed at best only noise. Therefore, biclustering seemed a better option as it can overcome this problem by simultaneously clustering genes and cells.

This stems from the observation that some genes (even though they may belong to the same gene group) may be co-expressed via a diversity of coherence models. One convincing argument is that a gene may participate in multiple pathways that may or may not be co-active under all conditions. It is biologically more meaningful to cluster both genes and conditions in gene expression data (10). Here both the gene expression profiles and molecular count data of the cells (e.g. age, sex, wellness, diameter, etc.) were used for biclustering. Zeisel et al. developed BackSPIN algorithm, a divisive two-way, unsupervised biclustering method based on sorting points into neighborhoods (SPIN) (9), which essentially sorts the expression matrix by cell-cell or gene-gene similarity. In contrast to the SPIN algorithm which does not identify clusters, this algorithm aimed to identify groups of cells/genes in an unsupervised manner. BackSPIN revealed nine major classes of cells: S1 and CA1 pyramidal neurons, interneurons, oligodendrocytes, astrocytes, microglia, vascular endothelial cells, mural cells (that is, pericytes and vas-

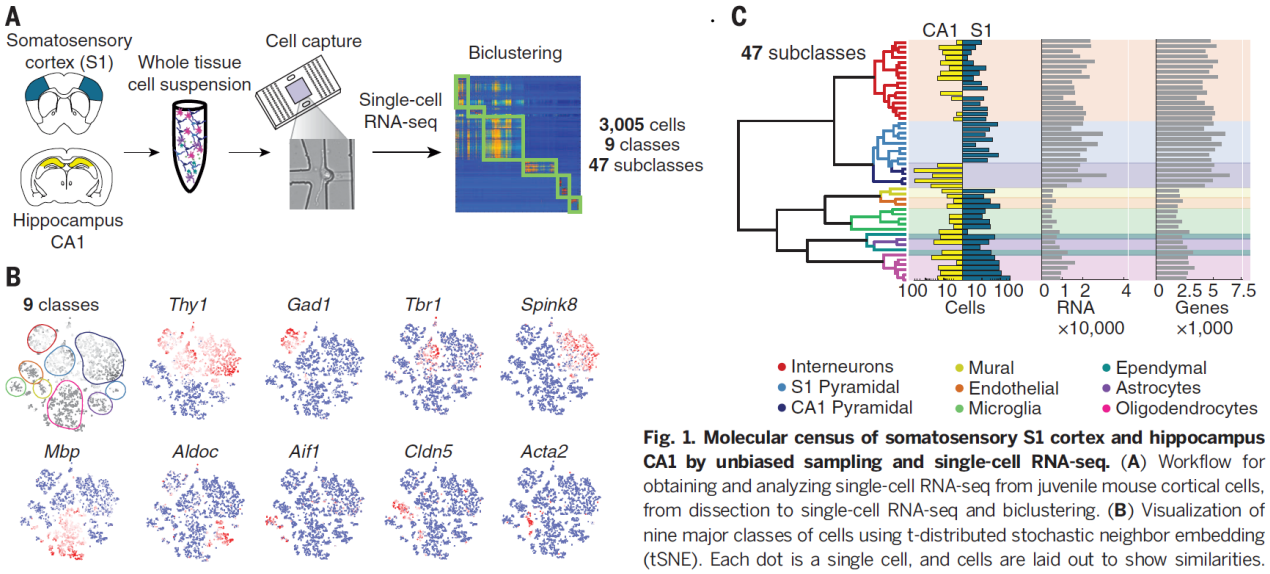

Fig. 1. Molecular census of somatosensory S1 cortex and hippocampus CA1 by unbiased sampling and single-cell RNA-seq. (**A**) Workflow for obtaining and analyzing single-cell RNA-seq from juvenile mouse cortical cells, from dissection to single-cell RNA-seq and biclustering. (**B**) Visualization of nine major classes of cells using t-distributed stochastic neighbor embedding (tSNE). Each dot is a single cell, and cells are laid out to show similarities. Colored contours correspond to the nine clusters in (A) and fig. S3. Expression of known markers is shown using the same layout (blue, no expression; white, 1% quantile; red, 99% quantile). (**C**) Hierarchical clustering analysis on 47 subclasses. Bar plots show number of captured cells in CA1 and S1, number of detected polyA+ RNA molecules per cell, and total number of genes detected per cell.

Figure 1: Paper by Amit Zeisel et al.

cular smooth muscle cells), and ependymal cells (Fig. 1).

Background

The following ML classifiers were used to carry out this study:

Support Vector Machines

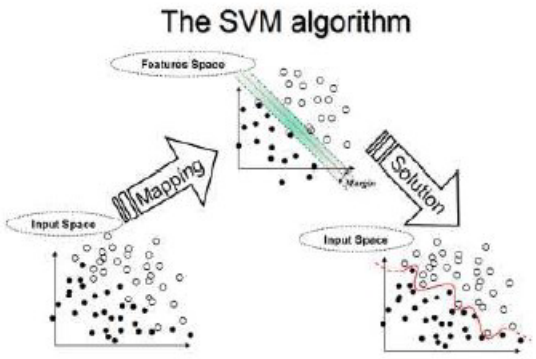

Figure 2: SVM algorithm

One of the simplest and most common algorithms used in machine learning, it was first developed in the 1960s and later got popularized and extensively adopted in the 1990s when it further got improved, it is still widely used today when performance is needed without much adjustment (4). SVM are supervised machine learning methods used for classification and regression analysis. It can efficiently perform both linear and non-linear classification.

In practice, the SVM algorithm functions using kernel classifiers. There are mainly three kinds of kernel SVMs: linear, polynomial, and radial kernel, the last was used in this study. By using the different types of kernels, the classifiers transform the original data according to the kernel type. Using the transformed dataset SVM can find non-linear decision boundary. Here radial kernel transforms the data using a Gaussian kernel that may give better classification result.

Random Forest

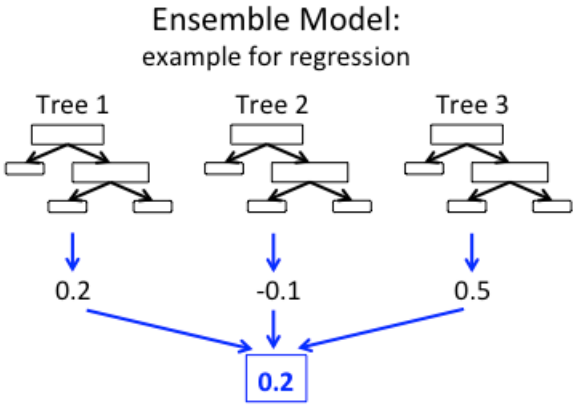

Figure 3: Random Forest algorithm

Random forest is perhaps the most powerful and most popular machine learning algorithm and has gained huge popularity in machine learning applications during the last

decade due to their good classification performance, scalability, and ease of use (6). A random forest can be seen as an ensemble of decision trees (a non-parametric supervised learning method used for classification and regression). The idea behind random forest is to combine weak learners to build a more robust model, a strong learner, that has a better generalization error and is less susceptible to overfitting. This unique combination of tree classifiers provided the random forest classifier with prediction accuracy and model interpretability that is superior to other machine learning methods.

The random forest algorithm can be summarized in four simple steps: 1) Draw a random bootstrap sample of size n (randomly choose n samples from the training set with replacement). Bootstrap is an ensemble technique that combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model; 2) Build a decision tree from the bootstrap sample. At each node: Randomly select d features without replacement. Then split the node using the feature that provides the best split according to the objective function; 3) Steps 1 and 2 are repeated for k times; 4) Aggregate the prediction by each tree to assign the class label by that has been predicted by the majority of classifiers (7).

K-Nearest Neighbors Algorithm (KNN)

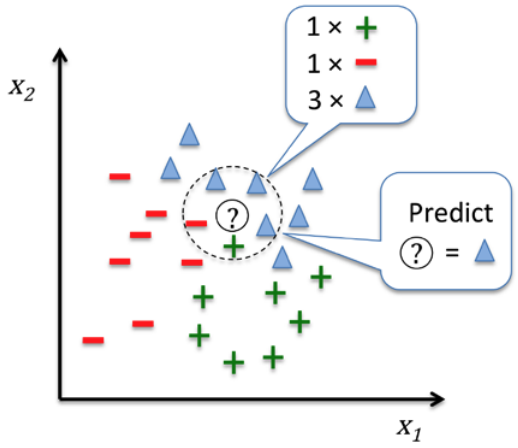

Figure 4: K-Nearest neighbor algorithm

KNN is fundamentally different from the other supervised learning algorithms. It is a typical example of a lazy learner. It is called so not because of its apparent simplicity, but because the model representation for KNN is the entire training dataset; it has no model other than storing (memorize) the entire dataset, so there is no learning required. The KNN algorithm itself is fairly straightforward and can be summarized by the following steps: 1) Choose the number of k and a distance metric; 2) Find the k Nearest neighbors of the sample that we want to classify; 3) Assign the class label by majority vote.

The following figure illustrates how a new data point "?" is assigned the triangle class label based on majority voting among its five Nearest neighbors. Based on the chosen distance metric, the KNN algorithm finds the k samples in the training dataset that are closest (most similar) to the point that we want to classify. The class label of the new data point is then determined by a majority vote among its k Nearest neighbors (7).

Neural Network

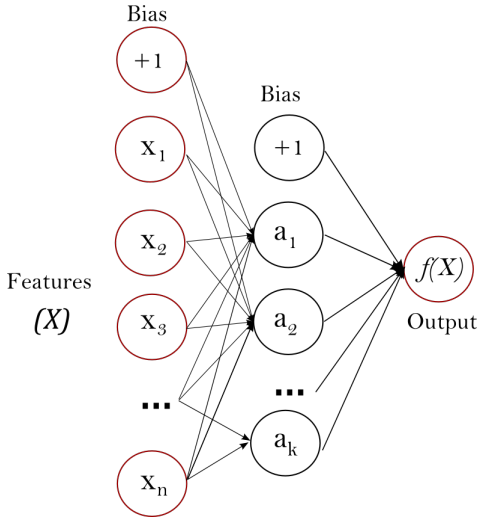

Figure 5: Neural Network algorithm

A perceptron is a single neuron model that was a precursor to larger neural networks. Multi-layer Perceptron is the most popular feedforward neural network model. They are currently one of the most active research topics in the machine learning field. The power of neural networks come from their ability to learn the representation in a training data and how to best relate it to the output variable that we want to predict. In this sense, neural networks learn a mapping. Mathematically, they are capable of learning any mapping function and have been proven to be a universal approximation algorithm.

Multi-layer Perceptron is a supervised learning algorithm that learns a function by training on a dataset, with a number of dimensions for input and a number of dimensions for output. Given a set of features and a target, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers.

The predictive capability of neural networks comes from the hierarchical or multi-layered structure of the networks. The data structure can pick out (learn to represent) features at different scales or resolutions and combine them into higher-order features. For example, from lines to collections of lines to shapes (8).

Experiments

The analysis is carried out using Python version 3.6.0 with all the machine learning tools being imported from the scikit-learn library (http://scikit-learn.org/stable/). Furthermore, the scikit-learn library has many convenient functions to pre-process data and to fine-tune and evaluate different models. In this paper, NumPy, and matplotlib libraries were used.

Two experiments were carried out using the same machine learning algorithms. In the first experiment, the original obtained data set were used where it has the shape (3005,4998). In the second experiment, the feature reduction algorithm PCA (principle component analysis) was implemented to reduce the dimension of the data, as 4998 seemed a large set of features, and the results from both experiments were compared.

Data set

The data used for the analysis is obtained from a previous study (1). It is arranged in a file where the columns represent the transcriptome profiles of 3005 cells, and the rows reflect the gene expression levels of 4998 genes for each cell. The file also contains annotation data, such as molecule count, age, wellness, and cell type classification. Classified cells fall into 1 of 9 possible classes: Interneurons, S1 Pyramidal, CA1 Pyramidal, Mural, Endothelial, Microglia, Ependymal, Astrocytes, and Oligodendrocytes.

Experiment 1: Classifying the Original Data

To find the good working parameter sets to classify the cells into 9 different types, we tuned the parameters for each of the four classifiers discussed above. Here are the different parameters sets we experimented on the dataset.

- K Nearest neighbor
  - number of neighbors: (3, 5, 7, ... , 19)
  - weights tuple: (”uniform”, ”distance”)
- Random Forest
  - criterion tuple: (”gini”, ”entropy”)
  - number of trees: 4, 8, 16, ... , 4096
  - minimum number of samples required to split an internal node: 2, 4, 8, ... , 32
- SVM
  - kernel: ’linear’, ’poly’, ’rbf’, ’sigmoid’
  - penalty parameter C of the error term: $2^{-3}, 2^{-1}, ..., 2^{15}$
  - gamma (Kernel coefficient for ’rbf’, ’poly’ and ’sigmoid’): $2^{-15}, 2^{-13}, ..., 2^{3}$
  - degree of the polynomial kernel function: 1, 2, 3, 4
- Neural Network
  - hidden layers: 27 different hidden layers (16, 16, 16) to (64, 64, 64)
  - activation function for the hidden layer:
    - identity: $f(x) = x$
    - logistic: $f(x) = \frac{1}{1+\exp(-x)}$
    - tanh: $f(x) = tanh(x)$
    - relu: $f(x) = max(0, x)$
  - solver for weight optimization: adam (stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy)
  - alpha(L2 penalty parameter): $2^{-8}, 2^{-7}, ..., 1$

The best parameter sets found for each classifier are:

- K Nearest neighbor: n=3,weights=uniform
- Random Forest: criterion=gini,n=128,minss=4
- SVM: kernel=poly,C=0.125,gamma=0.001953125, degree=1
- Neural Network: hls=(32, 32, 64), alpha=0.00390625,activation=identity,solver=adam

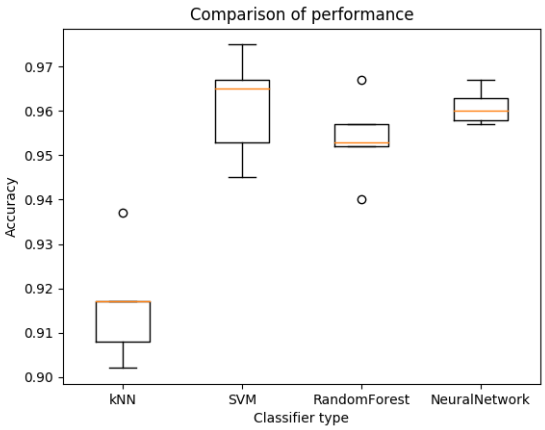

Figure 6: boxplot of 5-fold cross validation

Experiment 2: PCA Dimension Reduction

From the graph, we can see that SVM, Random Forest, and Neural Network classifiers successfully classified the cells into 9 different groups with high accuracy. However, kNN classifier was not performing well compared to other clasisfier. Hence, we decided to reduce dimension of the dataset to increase the accuracy found from kNN classifier.

Here the PCA algorithm was successfully implemented and the data features were reduced to 7, 13, 25, 50, 100, 200, 400, 500, 800, 1600, respectively. PCA algorithm finds the basis vectors that best decribes the dataset. For instance, PCA dimension reduction with number of basis = 100 means that the algorithm will select the 100 basis vectors with high eigenvalues. Dimension of the features are reduced to 100 by projecting the features with high variance to the 100-dimension space.

We used the parameter sets found in Experiment 1 to test the performance of each classifier on the 10 different dimension reduced datasets. Figure 7 shows the comparison of the classifiers to PCA dimension reduction datasets.

For kNN classifier, using 50 basis vectors gave the highest accuracy. Figure 8 shows the comparison of performance similar to that of Figure 6, but changed kNN to using PCA dimension reduction data.

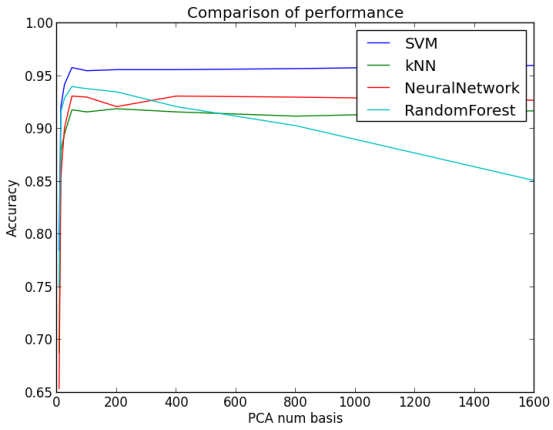

Figure 7: comparison of the classifiers to PCA dimension reduction datasets

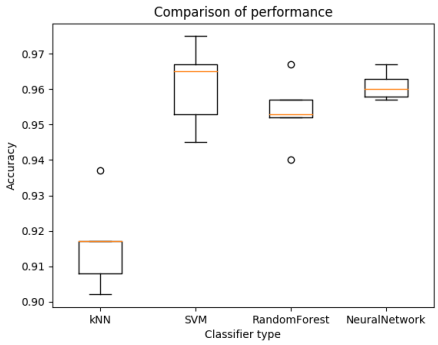

Figure 8: boxplot of 5-fold cross validation

Conclusion

In this paper, we compared the performance of four different classifiers in classifying the mouse brain cells into 9 different types. We tried various parameters sets to find a good working parameter set in this problem. Then using the parameter sets for each classifier, we reduced the dimension of the dataset by using PCA. After reducing the dimension, kNN was the only classifier that had better performance.

Our experiment has limitations. The parameter sets we tried may not be enough to find the best parameter sets in this classification problem. Also, the good working parameter sets found in Experiment 1 may not be the good working parameter sets for Experiment 2. Furthermore, the label of the dataset we used are not the ground truth. It's manually clustered into 9 different groups in the previous work from Amit Zeisel.

Despite limitations, the results demonstrate that mouse brain cells can be classified into 9 different groups only if the gene activation levels are given. In the future, we plan to use the parameter sets found for each classifiers, and test them on the different brain cell dataset. When reducing dimension of the data, other dimension reduction techniques could be used to enhance the performance of the classifiers.

References

1. Zeisel, A. et al. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. Science 347, 1138-1142 (2015).
2. Sandberg, R. Entering the era of single-cell transcriptomics in biology and medicine. Nature Methods 11, 2224 (2014).
3. Maxwell, W. Libbrecht and William Stafford Noble. Machine learning applications in genetics and genomics. Nature Reviews Genetics 16, 321332 (2015).
4. Vapnik, V. The Nature of Statistical Learning Theory. Springer (1995).
5. Scikit-learn documentations. http://scikit-learn.org/.
6. Leo Breiman. Random forests. Machine Learning, 45:532 (2001). Retrieved from https:// www.stat.berkeley.edu/ breiman/randomforest2001.pdf.
7. Raschka, S. Python Machine Learning. Packt publishing (2015).
8. Trevor, H. et al. The Elements of Statistical Learning, 2nd edition. Springer (2009).
9. D. Tsafrir, et al., Sorting points into neighborhoods (SPIN): data analysis and visualization by ordering distance matrices. 21, 23012308 (2005).
10. Kung, S.Y., and Mak, Man-Wai. A Machine Learning Approach to DNA Microarray Biclustering Analysis. IEEE (2005).