Cell Classifications by Single-cell RNA Sequences

Dicong Qiu
Carnegie Mellon University
dq@cs.cmu.edu

1 Introduction

The dataset used for this homework are drawn from 104 separate scRNA-seq experiments, and only the data points from 79 experiments are labelled while the data points from the remaining 25 experiments are used as test data without labels. The first challenge is to split the insufficient labelled samples into the training and validation datasets. Samples with the same experiment ID shall be put into the only one dataset to simuldate experiment bias, where some labels only exist in one experiment. Details of this problem will be discussed in section 2.

Each data point has more than more than 20,000 dimensions, so the second challenge would be reducing the them to 100 dimensions. Several approaches are discussed in section 3 to address it.

And finally, two types of classifiers, SVMs and neural networks, along with some explored training methods for neural network classifiers will then be discussed in sections 4 and 5, respectively.

2 Datasets Splitting

Two methods are explored to split the labelled data into training and validation datasets. The first method is splitting the data randomly regardless of the experiment IDs. But this method neglects the potential experiment bias in the test data, because the validation experiment IDs may share between the training and validation datasets, where the validation dataset cannot simulate the potential experiment bias in the test dataset.

So another method is applied, which splits the labelled data with some constraints: (1) the training dataset shall contain all labels; (2) the training dataset shall contain at least $2/3$ of the data with the same label. Under these condictions, there are actually not too many choices in data splitting. An appropriate splitting is shown as figure 1, where about 95% of labelled data are in the training dataset.

3 Data Dimension Reduction

Several data dimension reduction methods are explored, namely **principal components analysis (PCA)**, **kernel PCA** and **neural networks**.

In PCA and kernel PCA, all data including the labelled data and test data are used to fit the PCA model, because we want to capture the data distribution structure of the training dataset to provide better features for training the classifiers, and we also want to capture the structure of the test data so that the test data after dimension reduction still have better separable features for classification. And then the dimensions of labelled dataset and the test dataset are reduced down to 100 dimensions using PCA or kernel PCA.

PCA is a common dimension reduction method, but it has a strong assumption that the principal components have a linear structure and are orthogonal. But this property does not always hold true. In order to capture the nonlinear data structure, kernel PCA and neural networks with a structure shown in figure 3 are also explored. But the drawback for kernel PCA is a predefined kernel, such as a RBF kernel, which may be an inappropriate assumption, but one can never know what is the best kernel without prior knowledge of the domain. And it can lead to even worse outcome, like figure 2b in contrast to figure 2a, where kernel PCA and PCA are respectively to reduce the dimension of

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

1000 samples to 2 principal components as a toy example. Even though neural network approach can capture any hidden data structure, but it requires a massive amount of training data to train at least 2 million parameters as indicated in figure 3, which is not applicable to the insufficient dataset we have and leads to instability and overfitting issues.

4 Classifiers

Two types of classifiers, **SVMs** and **neural networks**, have been explored to classify the data after dimension reduction from the previous section.

A multiple classes SVM is explored as a classifier to classify the dimension-reduced data. To overcome the potential nonlinear structure hidden in the data, a RBF kernel is used. The SVM converges quickly and reaches a training accuracy of 100%. However, it takes a very long time to converge when the training dataset grows, and it is almost not feasible to train when the entire training dataset is used. Another drawback is that this method is not as easily scalable as neural networks, where stochastic gradient descent can be used to train incremental data from the previous trained network.

As for the neural network classifiers, different structures are explored, with different combinations of depths from 2 to 6 layer(s) and activation functions of **tanh**, **relu** and **sigmoid**. It has been identified by multiple experiments that 5-layer networks are better for this problem domain, which are deep enough to capture the data feature and shallow enough to avoid experiment bias (overfitting). Some performance examples are shown in figures 4a, 4b, 4c, 4d and 4e, of neural networks with 2 to 6 layer(s), respectively. Furthermore, neural networks with all ReLU layers except for the last sigmoid layers, seem to perform better than those with most tanh layers, because ReLU layers reduce vanishing gradient problem and fit the input data more naturally. Examples that show ReLU layers perform potentially better than tanh layers are shown in figures 5a and 5b.

In conclusion of this section, an optimal classifier is a neural network with input of 100 dimensions (data input after dimension reduction) and one-hot output with 46 dimensions (46 potential labels), which has 4 ReLU layers ($[100, 64, 64, 64]$) and the last sigmoid layer to output one-hot labels.

5 Training Neural Network Classifiers

There are different ways to train neural networks, including **batch training** (averaging all the gradients of training data), **stochastic gradient descent (SGD)** (with one training data point at a time), **SGD with minibatch**, and **Adam**. Adam is good at training neural networks with large dataset, which has been also explored but it is not applicable to our situation. How to train the neural network depends on the distribution of the test dataset, of which no prior knowledge is given. So a reasonable assumption is that the distribution of the test data will be similar to that of the labelled data. So there will be no need to **re-weight samples**. And as described in section 2, the validation dataset is designed to best simulate the test dataset considering experiment bias, so the goal of training will be better predict the validation data labels. As shown in figures 6a, 6b and 6c, the bigger the batch size is, the more stable the validation accuracy will be, but it is more limited by an upper bound. And a minibatch size of 256 combines both stability and the potential to reach higher validation accuracy, so this configuration is adopted.

Another important technique is **early stopping**, which can avoid potential overfitting issues. It can be applied to matrices like training/validation loss/accuracy. According to experiment results, for instance figures 7a and 7b, applying early stopping to validation accuracy with patience of 5 episodes is more reasonable and performs better than other configurations.

6 Conclusion

Dimensions of all data are reduced from more than 20,000 to 100. The labelled data are then split into training and validation datasets according criteria described in 2. A neural network with 4 ReLU layers and the last sigmoid layer is used as the classifier, which is trained with minibatch of size 256 and early stopping. The final train accuracy of 93.6297% and validation accuracy of 46.9979% are reched, as shown in figure 7b.

Appendix 1: Figures

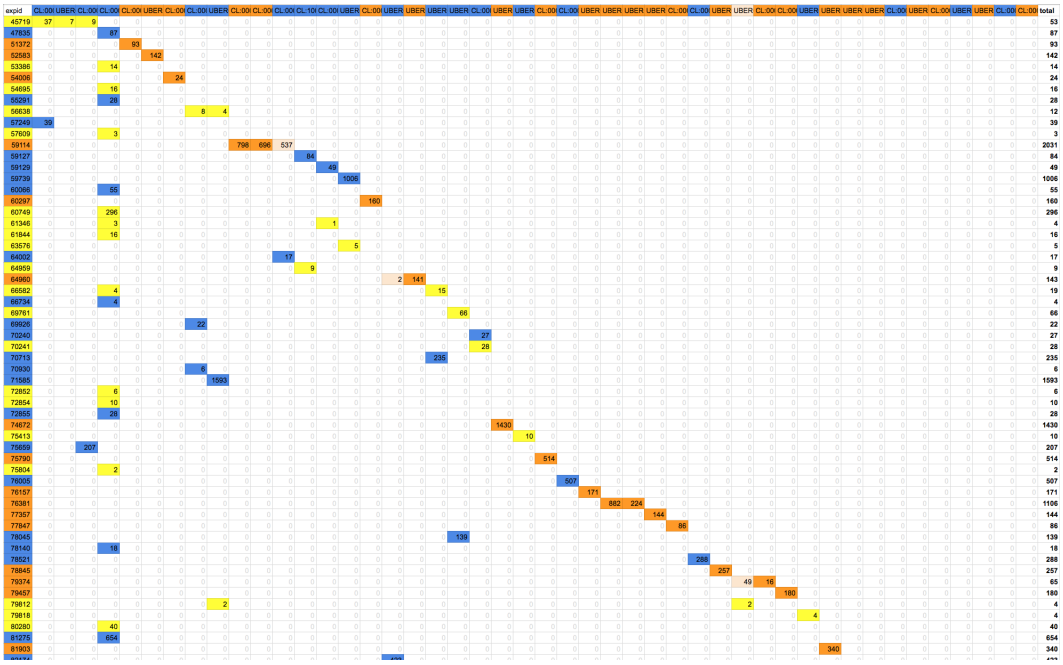
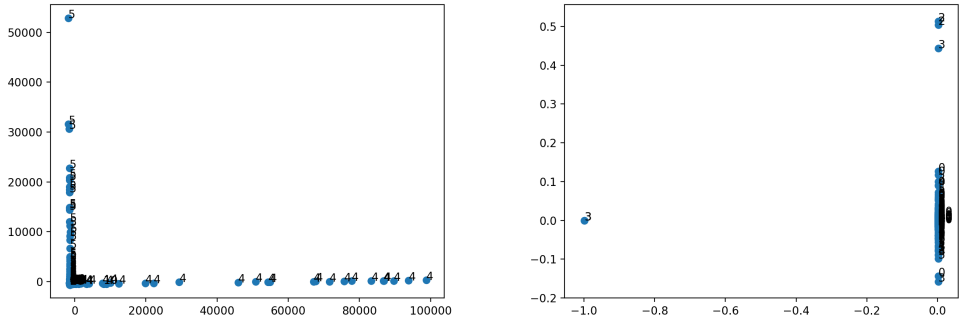


Figure 1: partial datasets splitting result



(a) PCA (b) kernel PCA (RBF kernel)

Figure 2: PCA and kernel PCA with 2 components on 1000 scRNA-seq samples

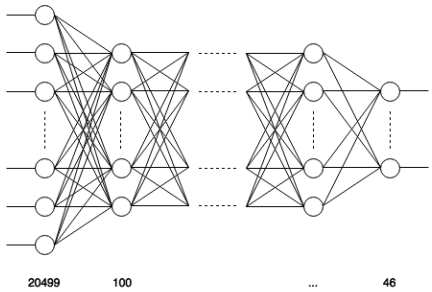
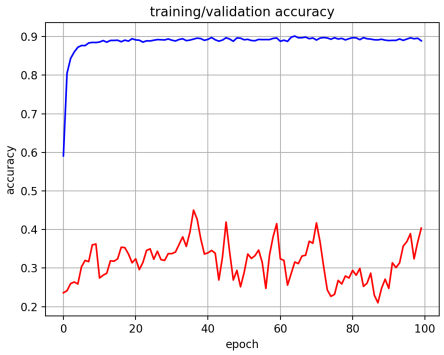
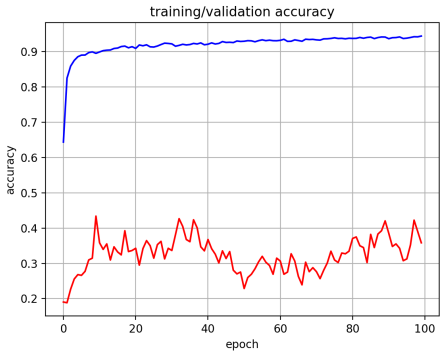


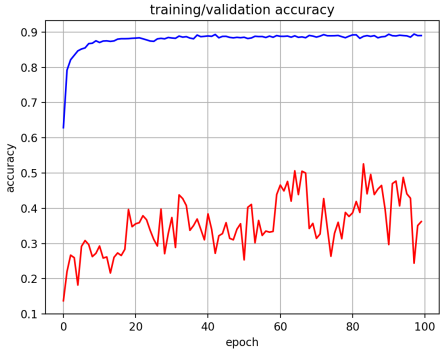
Figure 3: neural networks structure for dimension reduction



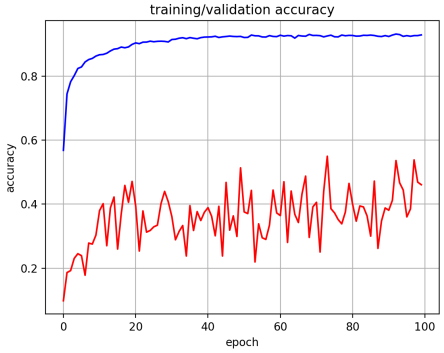
(a) 2 layers



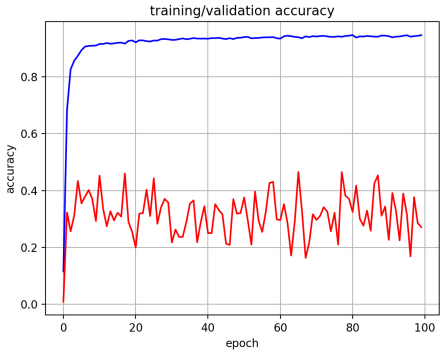
(b) 3 layers



(c) 4 layers



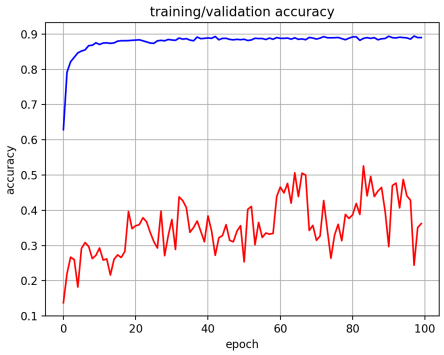
(d) 5 layers



(e) 6 layers

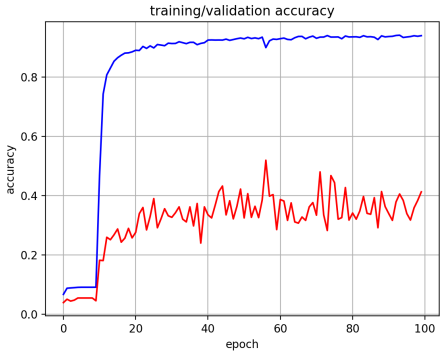
Figure 4: performance examples of neural network classifiers with different depths (blue curves indicate training accuracy and red curves indicate validation accuracy)

(a) ReLU layers (4-layer network)

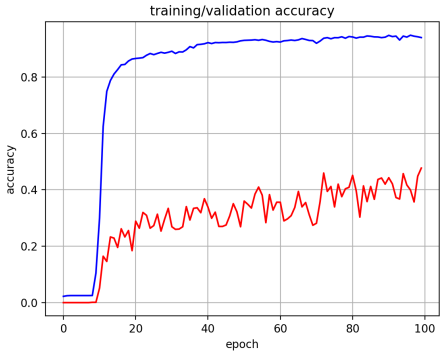


(b) tanh layers (4-layer network)

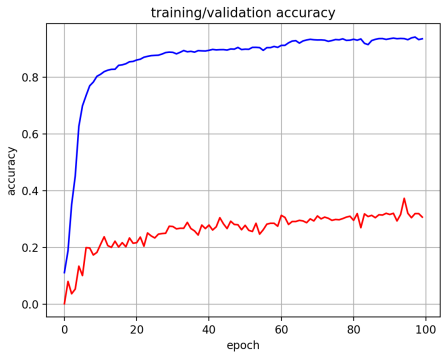
Figure 5: performance examples of neural network classifiers with different activation functions (blue curves indicate training accuracy and red curves indicate validation accuracy)



(a) minibatch size of 128

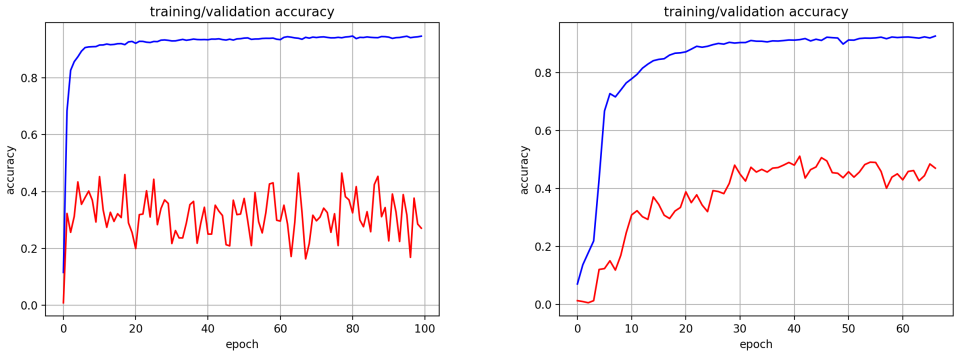


(b) minibatch size of 256



(c) minibatch size of 512

Figure 6: performance examples of neural network classifiers with different minibatch sizes (blue curves indicate training accuracy and red curves indicate validation accuracy)



(a) same configurations without early stopping

(b) same configurations with early stopping

Figure 7: performance examples of neural network classifiers for early stopping (blue curves indicate training accuracy and red curves indicate validation accuracy)