Future Computing Technologies Lab: Creative Inquiry

LUNG TISSUE CLASSIFICATION

December 7, 2020

Bradley Selee
Clemson University
Department of Electrical and Computer Engineering
bselee@clemson.edu

1 Introduction

The data set I chose to work with was a Gene Expression Matrix (GEM) derived from lung tissue samples. This matrix had already been log transformed and quantile normalized. The data set was given to me by Alli Hickman as part of a research project in the Feltus Lab. This GEM is organized with tissue samples as columns and the gene expression values as rows; there are 1415 samples and 19648 genes.

The goal of this data set was to train a multilayer perceptron to classify lung tissue samples as cancerous or normal, and distinguish between the types of cancer. The data originates from The Cancer Genome Atlas (TCGA) and The Genotype-Tissue Expression (GTEx) project. This model is part of a larger project, with the overall goal to create a 'general purpose' classifier for multiple data sets.

2 Materials and Methods

The project used a linear neural network to classify samples with supervised learning. Because the data came from a genetics lab, an understanding of basic computational biology was required. I worked with several PhD students in the Feltus lab to better understand the biology aspect and ultimate goal of the project.

2.1 Understanding the Data

The very first step was to understand the data I was working with. This was necessary to understand how to preprocess the data and load the samples into the model. The lung data set contains two files. The first file is a GEM, which contains the gene expression from 1415 samples across 19648 genes. The second file is a label file. This file maps the sample to a specific condition: TCGA-LUAD, TCGA-LUSC, TCGA-Normal, and GTEx_normal. LUAD and LUSC represent different tumors, while Normal represents normal and pre-cancerous tissue. The GEM had to be transposed to align with the label file, then the two files were merged, giving a label to each sample.

The GEM was already log transformed and normalized using quantile normalization. This made preprocessing the data much easier. Quantile normalization is a technique that makes multiple distributions identical in properties [1]. An example of this is show in Figure 2.1.

2.2 Data Preprocessing

After understanding the data format, further processing was required to prepare the data for the model. Because the data set was already normalized, internal normalization was not needed. However, the lung data set had a lot of missing values which needed to be taken care of. A map the missing data is shown in figure 2.2 Because the lung data was log transformed, expression values of 0 become undefined and create missing

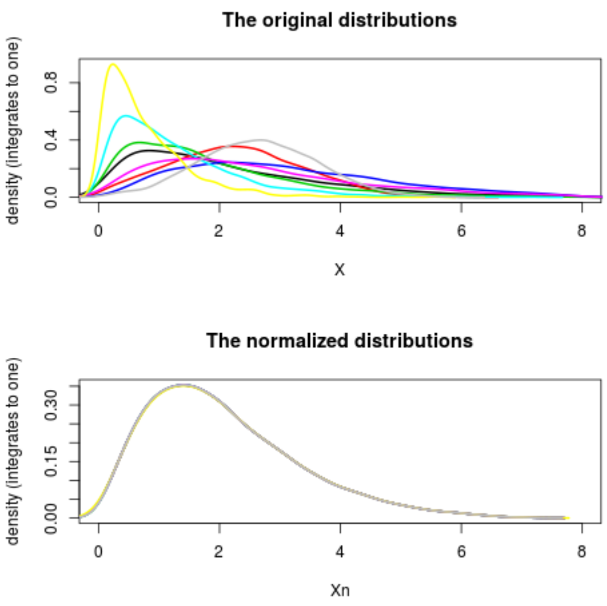

Figure 2.1: Distribution before and after quantile normalization

values in the transformed data set. Due to the low expression value, the missing data was replaced with the global minimum. Figure 2.3 shows the lung distribution after the missing data was replaced. The data shows a Gaussian distribution, the first peak is created by filling in the missing values, and the second peak shows the true distribution.

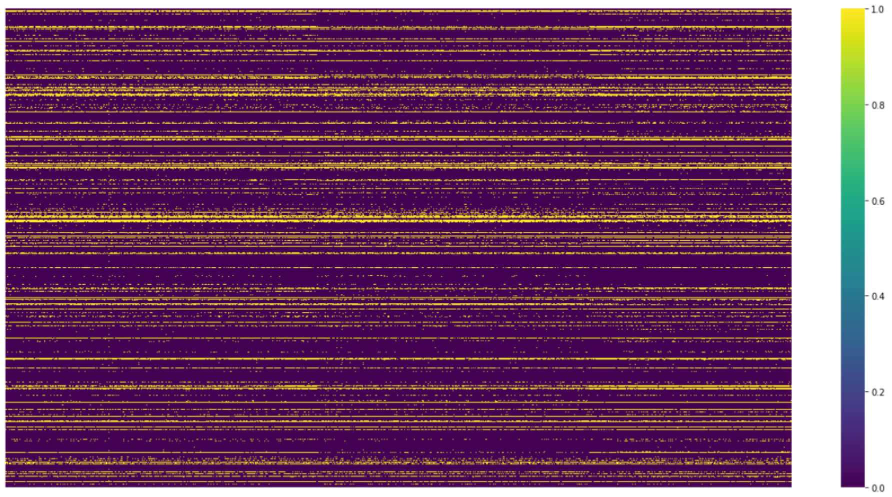

Figure 2.2: A visual of the gem showing the missing data. Yellow represents data that is missing.

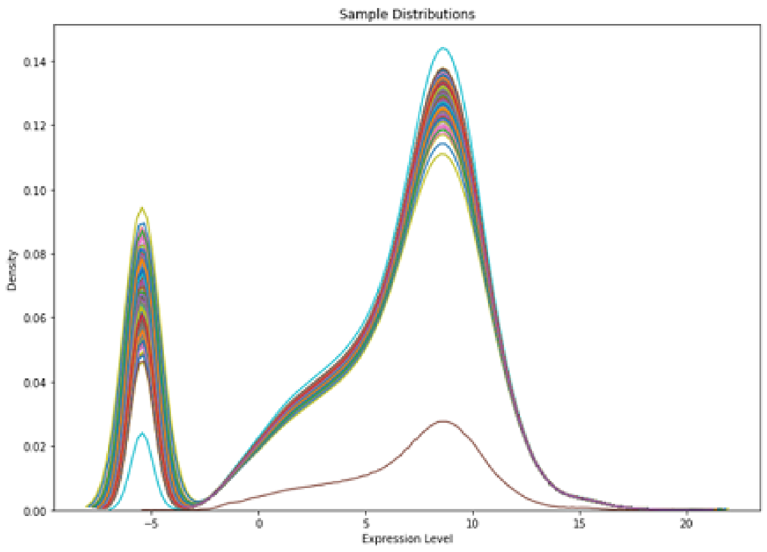

Figure 2.3: Distribution of the data set after preprocessing

For the most part, the preprocessing was not too difficult. There was a lot of confusion that arose within the lab discussions as to how the data should be normalized. Ultimately, we decided that because the data was already scaled using quantile normalization, no further normalization would be needed.

2.3 Creating Train and Test Data

After preprocessing, the data set was split into train and test sets. The data was very unbalanced, meaning out of the four classes they were all weighted differently. This imbalance had to be taken into account when splitting up the data.

Using sklearn to create a stratified split, a training and test set was created using 70% and 30% of the data, respectively. The goal of stratify is to weight the samples in the train and test set according to the weighting of the class labels.

2.4 Model Architecture

The model is a simple feed-forward neural network composed of an input layer, 3 hidden layers, and an output layer. The input layer size is number of genes in the data set. Each hidden layer is a linear layer that uses the ReLU activation function and contains 1024, 512, and 256 neurons. Finally, the output layer has a size of 4 because there are 4 different labels in the data set. The soft max function was not used on the output layer. The reasoning for this is because in PyTorch, the loss function for multi-class problems takes raw outputs from the model. If a softmax function were used, the outputs would no longer be raw values but rather a probability between 0-1.

2.5 Training and Evaluating

The model was trained on 70% of the data set and evaluated on 30% for 50 epochs. The learning rate was 0.001, which decreases by a factor of 10 every 50 epochs, and a batch size of 16.

3 Results

PyTorch does not keep track of features like accuracy and loss as easily as Keras. To collect the results, data frames were created to store the training/test accuracy and loss per epoch. Once the model finished training, a final forward pass was done with the test set in order to get the predictions for each class. This data was used to plot the confusion matrix. The results for this project are shown below.

After training the model for 50 epochs, the test accuracy plateaued at around 97%. With multi-class problems, the accuracy can be slightly misleading because it does not score well each individual label was classified. To account for this, F1 score was calculated which came out to 97%. The confusion matrix is also included, this shows how well the labels were predicted compared to the actual label of the sample. This can be thought of as a graphical representation of the F1 score. From these graphs, it is clear that the model has learned the samples with high accuracy.

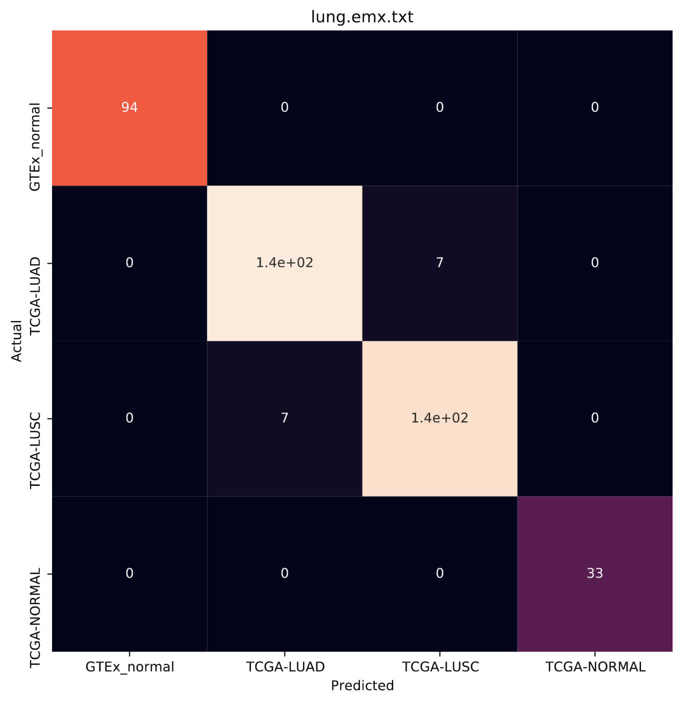

Figure 3.2: Training and Test loss per epoch

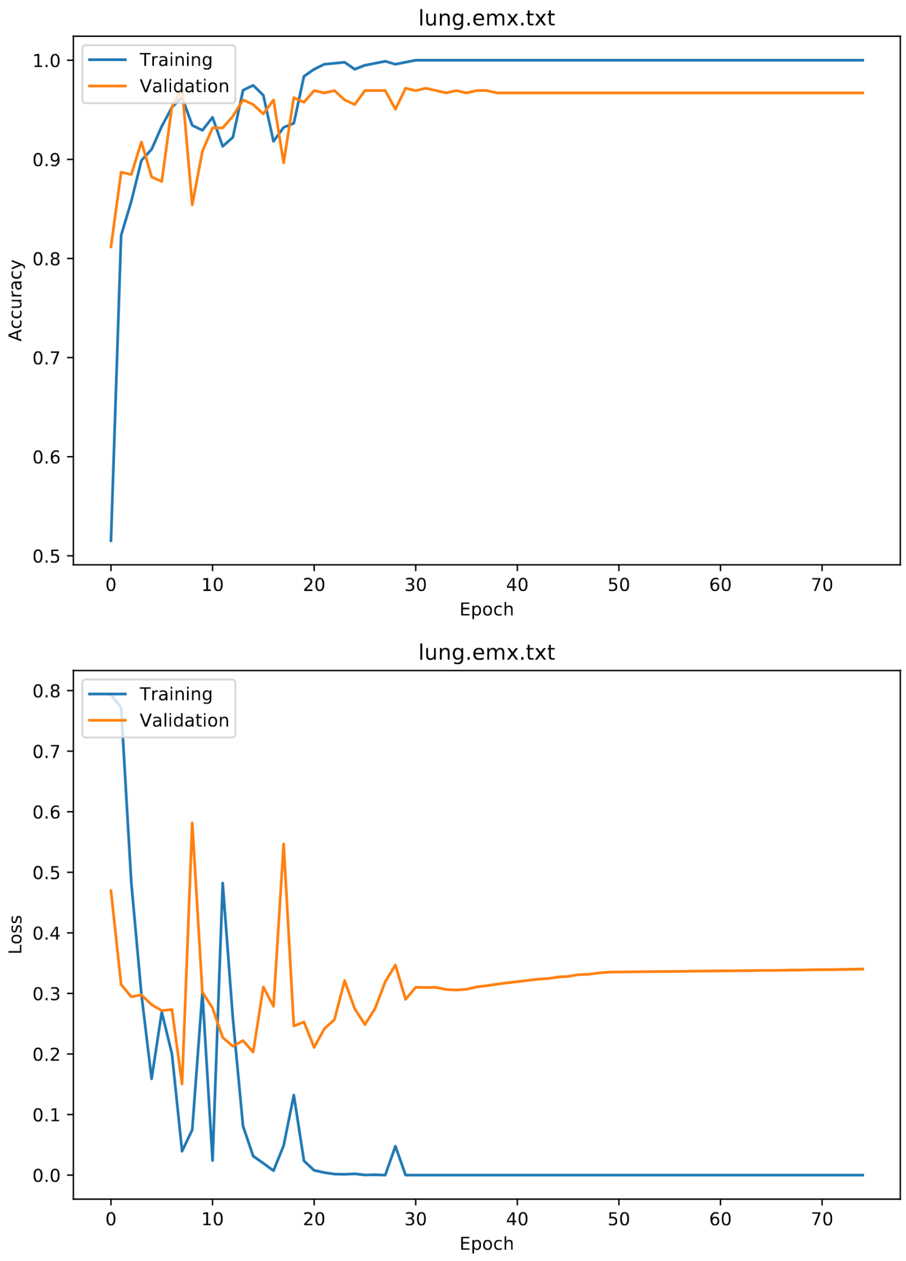

Figure 3.1: Training and Test accuracy per epoch

4 Experience

My overall experience in the CI was gratifying and the transition to online felt very smooth. I really enjoyed everyone screen sharing their progress each week and exploring new methods that they tried. It is hard to say how the CI could be improved due to the virus, but for continuing students I feel that the CI was managed well. I do feel that for new CI students, they would have had a better experience in person because it is hard to keep up with everyone online. The main challenges I faced this semester was mostly learning PyTorch as opposed to previously using Tensor Flow. The best way I overcame this issue was by reading a lot of documentation and testing my project on other data sets. The best resource of this CI is being exposed to practical uses of data science and machine learning, rather than just theory like in traditional classes. I definitely plan to go into some data science field in the future, as this field has recently been exploding and is now more important than ever. I plan to begin my masters with a thesis here at Clemson in Fall 2021. I would love for Dr. Smith to be my advisor and I plan to reach out to her once I understand more about the process. Seeing the work that her students do in the lab and their achievements after graduating seems like an amazing opportunity.

5 Conclusions and Future Work

From the high classification results, there is a clear relation between gene expression and lung tissue. In order to make this statement on any tissue, other data sets with different tissue would need to be tested. In contrast, there are some samples that were nearly impossible to classify. For future work towards this project, isolating these samples and looking into features that would make them hard to classify would be very interesting. Another approach would be to try and synthetically create RNA expression values, and see if they classify well on this same model. However, the approach to this is currently unknown to be and would require a lot of research.

References

[1] Y. Zhao. How to do quantile normalization correctly for gene expression data analyses. *Scientific Reports*, 10(1), 2020.