EN.601.448/648 Computational genomics: Final Project - Midterm report

Ayush Agarwal (aagarw33)
Computer Science (Masters)

Lohita Sivaprakasam (lsivapr1)
Computer Science (Masters)

Satish Palaniappan (spalani2)
Computer Science (Masters)

Srivathsa Pasumarthi (psrivat1)
Computer Science (Masters)

April 30, 2019

Due: 4/12/2019

1 Problem

In this project, our primary aim is to perform taxonomic-classification of bacterial DNA using various machine learning and deep learning approaches. The DNA sequences for these single-cell microorganisms serve as a unique identifier (like a barcode) for that species and characterizes each of them completely. Following [1] as the base literature, we would like to perform the same classification using various other machine learning and deep learning approaches as described below and compare each method using defined metrics. Secondly, we would also like to use deep representation learning approaches to model meaningful representations of the DNA sequences and use these representations to perform hierarchical agglomerative clustering to validate if they correlate with their actual taxonomic hierarchy.

2 Data

We are using the same data as of [1] which is from the Ribosomal Database Project (RDP [2] - Download Link) and consists of around ∼550K samples of bacterial DNA along with the information about its taxa - phylum, class, order, family, genus. The source FASTA file was processed and samples whose sequence lengths were between 400 and 440 were picked; this choice was made only for keeping the sequence lengths almost constant while maximizing the total number of samples. The final data consists of ∼9174 samples spanning over 3 phyla, 5 classes and 19 orders.

In order to tackle the extreme imbalance in the *order* level, we have grouped the samples that are under-represented, into a single class named *Others*. The **train-test-validation split is 60%, 20% and 20%** respectively. The choice of dataset split was intuitive as

we wanted to give enough samples for parameter tuning and validation, given the relatively small number of samples.

Instead of doing multi-category classification, we are solving three separate classification problems by generating separate data files for phylum, class and order; this is a conscious choice in order tackle the problem of class imbalance in the lower levels of the hierarchy and to avoid multiplicative error transmission from top to bottom in the hierarchy.

3 Methods

3.1 Methods for Taxonomic-Classification

3.1.1 General Machine Learning methods (Features: k-mer counts)

Normalized k-mer counts were extracted from the dataset and stored as individual feature matrices for different K values. Using these features, SVM and Random Forest models were trained for classifying the phylum, class and order. The goal here is to find the ideal K value that represents the data with the right amount of context, using the basic ML methods. F1 scores were calculated for the different models separately, for each K value ranging from 3 to 6. SVM and Random Forest both perform almost equally well with an average F1 score of 0.94. Detailed analysis of results can be found in Section 5.1

3.1.2 General Machine Learning methods (Features: vector representation)

Another type of encoding used was the ordinal encoding of the gene data where each of the four types (A, C, G, T) and an additional type (to accommodate all other data apart from the above), were encoded as separate real number representations. This was constructed as a direct mapping and each sample's sequence was simply converted to a vector-based representation of the same length. Using these features, SVM and Random Forest classifiers were trained for the phylum and class level hierarchy. This method was directly compared to the previously explained k-mer based encoding method, to understand the importance of sequence encodings in our dataset. An average F1 score of about 0.79 was obtained for the phylum classification using both the models.

3.1.3 Convolutional Neural Networks (Features: QRCode-like image representation of DNA sequences)

Here we would like to explore an interesting way to represent a DNA sequence, which would be to visualize them as QRCode-like images. An example of how this encoding technique works is shown in Figure 1. As the sequences in the dataset are of varying length, we first take the length of the longest sequence and ceil its square root and take this number, D, as the max width and height of the image to be generated. Then we take each sequence in the dataset and reshape them to $D * D$, 2D arrays, by padding the 2D array with zeros where there is no information. Then we one-hot encode each cell based on the A, C, G, T base pairs, thus leading to an array of size $D * D * 4$, where 4 is the number of channels of the

image. Just for the sake of visualizing this as an image, we assigned 4 colours namely Red, Green, Blue and Orange for A, C, G, and T respectively (and black for no information). For our dataset here with N samples, as the maximum sequence length is 440bp, we generate a dataset of dimensions $N * 21 * 21 * 4$.

These DNA QRCodes were then used as the input features for training the Convolutional Neural Network who's architecture is as defined in Figure 2. This network was trained using a Negative Log-likelihood loss which was optimized using the Adam optimizer with a learning rate of 0.001 for 25 epochs with a batch size of 32.

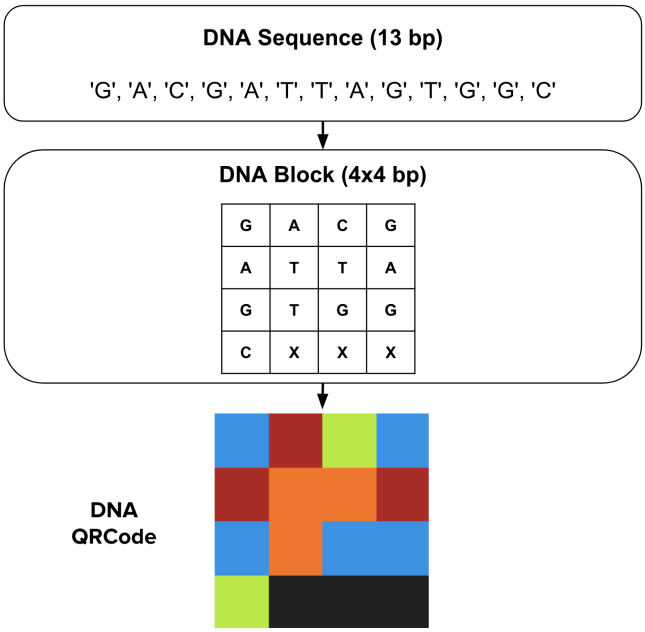


Figure 1: QR-code like DNA sequence representation

```
ConvNet(
  (layer1): Sequential(
    (0): Conv2d(4, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc): Linear(in_features=800, out_features=3, bias=True)
)
```

Figure 2: CNN Architecture

3.1.4 Recurrent Neural Networks - LSTM (Features: Raw DNA sequences)

Given the lack of explicit features and high dimensionality of the DNA sequences, recurrent neural networks(RNNs) provide an efficient way to encode the sequences for classification. Long Short Term Memory(LSTM) is a special type of RNNs, which is also composed of a chain of repeating modules of neural networks. Contrary to the standard RNNs that have a single tanh layer as the repeating module, LSTMs have four layers that interact with each other in order to handle the long-range dependencies with the aid of input, forget and output gates. In this approach, we are making use of LSTM networks due to their ability to process entire sequences and preserve the entire sequence's information irrespective of the gap length in huge sequences.

We have carried out experiments with the raw DNA sequences as the input to the LSTM and attempted to classify the sequences with respect to their phylum, class and order labels. The LSTM network was trained to closely predict the ground truth labels, by means of optimizing the negative log-likelihood loss with an Adam optimizer at a learning rate of 0.001. The obtained results and their analysis are presented in Section 5.4.

3.2 Representation Learning and Agglomerative Clustering

Deep representation learning has been used in recent years for various applications such as clustering, classification and data compression. Our goal here is to apply various representation learning techniques such as Variational Auto-Encoders (VAE) and LSTM Auto-Encoders to learn a representation of the DNA sequence and use that representation as features to perform hierarchical agglomerative clustering. The clustering is done to verify and validate that the learned representations actually correlate with their taxonomic hierarchy.

3.3 Ambitious Goals

These tasks will be done based on the availability of time and resources.

3.3.1 Ensemble Learning

We will attempt to learn multiple possibly uncorrelated learners to solve the same problem with sub-sampled data and combine these models to learn a meta-learner using stacking techniques such as an MLP, to get a much improved final prediction accuracy. This will help us leverage the advantages of multiple machine learning models.

3.3.2 Hybrid CNN-RNN approach

Here we borrow one of the key ideas from deep-learnt image captioning architectures which use both CNNs and RNNs to learn representations of data and combine them into a richer lower dimension space to get a hybrid data representation that facilitates feature interaction among different types of data. Using this as key, we combine the QRCode-like representation of a DNA sequence run over a CNN and the raw DNA sequence run over an RNN using fully connected layers to finally perform taxonomy-classification, which we believe will give better results.

3.3.3 Family Level Classification

When we move to the family level of the hierarchy, the dataset we have suffers from very high class-imbalance and with a high number of classes to predict as well. In order to train models in spite of this, we propose to group the classes with an almost same number of samples in a single dataset and produce N such datasets. Then we can train N different ML models on these datasets and bag their results to predict the final label, and if we are using Deep-learnt models like a CNN or RNN, we can use weighted loss functions to account for the class imbalance, where the weights are determined by the class imbalance proportions.

4 Evaluation metrics

We would like to use the following metrics used to evaluate any standard supervised classification or unsupervised clustering problem:

1. **Classification Problem:** Overall accuracy, Per category accuracy, False Positive Rate, F1 score (Good metric for classification as it is the harmonic combination of precision and recall) and weighted F1 and Accuracy scores.
2. **Clustering Problem:** Mutual Information Score, Homogeneity Score

5 Results

5.1 General Machine Learning methods (Features: k-mer counts)

Figure 3 and Figure 4 shows the K-mer wise F1 score counts. We observe that Random Forest classifier performs slightly better, but the peak is at k=5 for both classifiers. From these figures, we can infer that k=5 is an ideal setting in terms of the context in the DNA sequence that it provides and also in the number of features (4^5) which is not too little nor too much. Hence 5-mers represent the data precisely which 3-mers and 4-mers do not do due to both the lack of context and lack of number-of-features. There is a significant drop in F1 score for k=6 because, at this point, the number of features (4^6) becomes greater than the number of samples.

5.2 General Machine Learning methods (Features: vector representation)

The results obtained for the phylum classification were encouraging, with the simple encoding, leading to validation and test accuracies close to 88% for the random classifier and about 81% for SVM. The ROC curve for the same is as seen in Figure 5.

When the confusion matrix for the above classification was extracted, it was found that most errors lay in false positives, appearing where the positive class was the most dominant class in terms of samples. Although the skew in phylum classes is less, the above property indicates that classification on the basis of Class and Order may not perform well for the given encoding method.

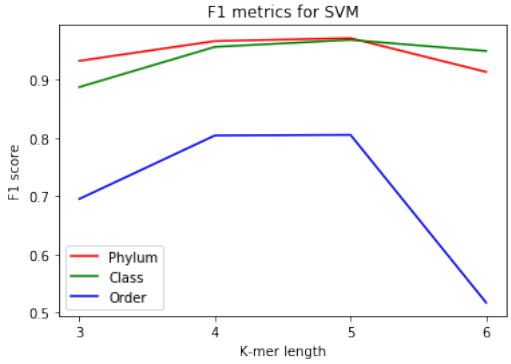


Figure 3: K-mer wise F1 scores for SVM

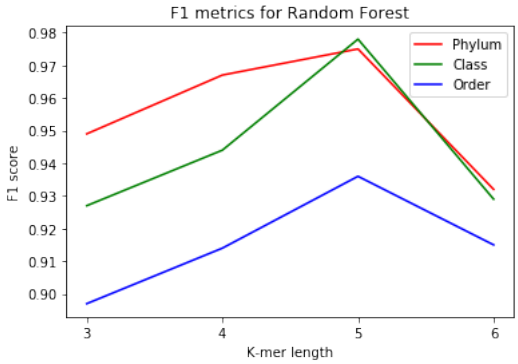


Figure 4: K-mer wise F1 scores for Random Forest classifier

As a next step, the same methods were applied to the Class level classification to test the previous hypothesis. As expected, poor results were obtained with F1 scores using both Random Forest and SVM, languishing at 20% and according to the confusion matrix, it was observed that most prediction allotments were done towards the class having the most number of samples. The ROC curve for the same can be seen in Figure 6.
Order level classification gave an even worse performance having an F1 score of 15% on the 10 order classes.

5.3 Convolutional Neural Networks (Features: QRCode-like image representation of DNA sequences)

We train the Convolutional Neural Network on the DNA QRCodes as described in Section 3.1.3. We repeat the training for each level of the taxonomy hierarchy, namely Phylum, Class and Order. We train on the training dataset and evaluate on the test dataset, but for time being we haven't used the validation dataset to optimize over the CNN parameters, which we will do later. The test accuracy of the Phylum, Class and Order classifiers are 99.29%, 98.85% and 93.57% respectively, and the accuracy vs epochs plot can be found in Figure 7. To see more plots, visualizations and results, have a look at the Jupyter Notebook hosted here.

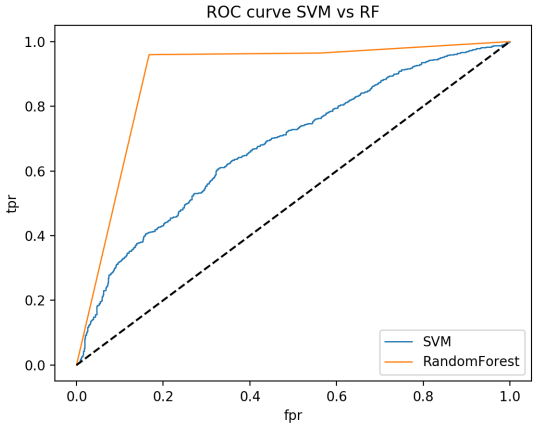


Figure 5: ROC curves for phylum classification - vector encoding

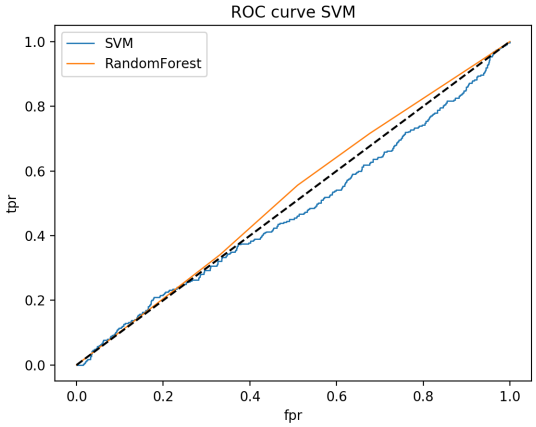


Figure 6: ROC curves for Class classification - vector encoding

From the accuracy plots it is also evident that the **CNN model as described in Figure 2 trained on the DNA QRCodes majorly outperforms all the models and methods we have experimented with**, including SVMs, Random Forests, and RNNs, by achieving the highest accuracy possible for each of the three levels of classification tasks. Other inferences from the accuracy plots are that the Phylum and Class models are well trained and do not suffer from the overfitting problem and generalize well to unseen samples, while the Order level CNN model, actually starts to slightly overfit the data and hence the gap between the train and test accuracy curves. Which can be alleviated by introducing regularization techniques like dropouts and L2 weight decay, which we will try out and report in our final proposal.

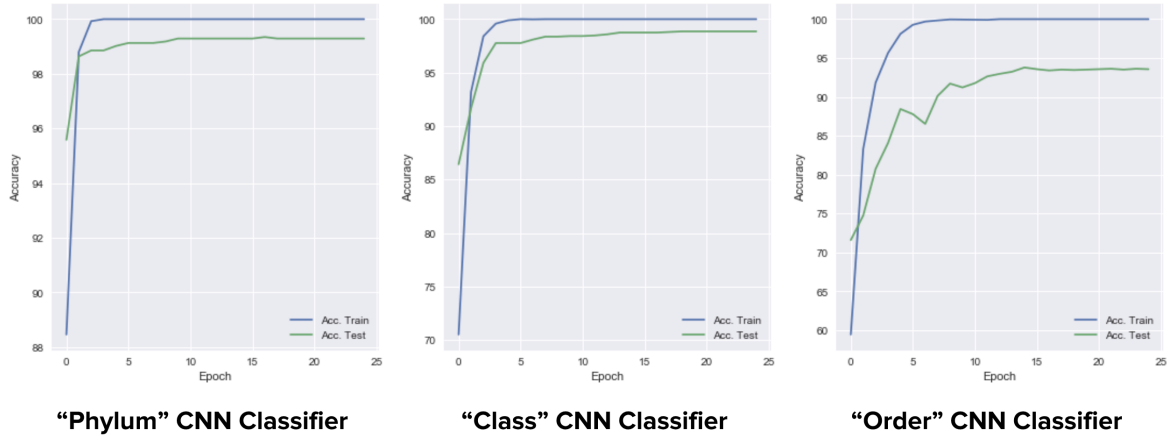


Figure 7: Accuracy vs Epochs Plot: CNN (QRCode)

Classifier	Train data		Test data	
	Accuracy	F1 Score	Accuracy	F1 Score
Phylum	0.97	0.97	0.976	0.97604
Class	0.978	0.9779	0.947	0.947
Order	0.987	0.9869	0.891	0.8912

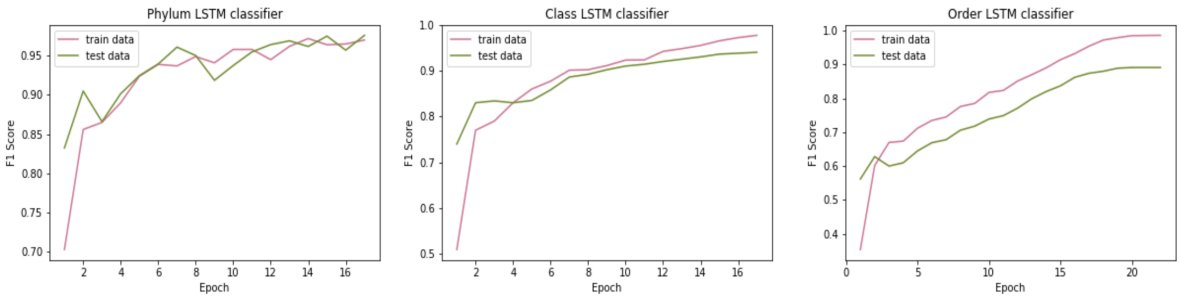
Table 1: LSTM Classifier Evaluation Results

5.4 Recurrent Neural Networks - LSTM (Features: Raw DNA sequences)

As stated in Section 3.1.4, we have attempted to classify the LSTM encoded DNA sequences based on three different levels, namely phylum, class and order. This approach gives us an insight into the degree to which the LSTM encoded DNA sequences are indicative of the phylum, class and order to which they belong. The F1 scores obtained on each of the classification tasks are presented in Table 1 below. The test accuracies for Phylum, Class and Order classifiers are 97.6%, 94.7% and 89.1% respectively. The F1 scores measured are the weighted F1 scores in order to account for a balanced distribution of samples with respect to their labels. The plots in Figure 8 illustrate how the F1 scores evolve over time with respect to the number of epochs. It can be observed from the results that the LSTM classifier's performance increases for coarse levels of the taxonomy hierarchy ie. from Order to Class to Phylum.

6 Significant Changes from Proposal

1. In the proposal, we filtered the DNA sequences of lengths between 1270 and 1370 and decided to use a dataset with 330K samples. But we realized that the dataset has a lot of duplicate sequences. We also wanted a compact and representative dataset to finish model training in the given time. Hence we decided to go with a dataset with 9174 samples as described above



"Phylum" LSTM Classifier **"Class" LSTM Classifier** **"Order" LSTM Classifier**

Figure 8: F1 Score Vs. Epochs Plot: LSTM Classsifier

2. We have decided to drop the genus level classification as many classes are heavily under-represented. We have moved the family level classification to reach goals, as we have to solve the problem differently (Ex: boosting and bagging) to alleviate the problem of class imbalance.

3. We proposed a multi-category classification approach, but we are solving three separate classification problems due to the reasons mentioned above.

7 Issues

We foresee a few issues in this project which are enlisted below:

1. Class imbalance at the family and genus level.

2. Sequences are of a varied length which might be a challenge for CNN and RNN approaches. We are currently planning to truncate/pad to obtain sequences of fixed length, though there are ways to handle varying length sequences using RNNs which we will try.

3. We are also sceptical about the deep representation learning approaches to actually model the distribution of each taxonomy-level groups.

References

[1] Rizzo R., Fiannaca A., La Rosa M., Urso A. *A Deep Learning Approach to DNA Sequence Classification*. Computational Intelligence Methods for Bioinformatics and Biostatistics. CIBB 2015.

[2] Cole, J. R., Q. Wang, J. A. Fish, B. Chai et al. *Ribosomal Database Project: data and tools for high throughput rRNA analysis*. Nucl. Acids Res. 42(Database issue):D633-D642 2014