**Applying Classification Models to Predict Categorical Identity**

Zeping Zhang r0787288

Tao Zhou r0778423

1,2 Faculty of Engineering Technology, Campus Group T, Leuven

Vesaliusstraat 13, 3000 Leuven, Belgium

zeping.zhang@student.kuleuven.be

Tao.zhou1@student.kuleuven.be

# Abstract

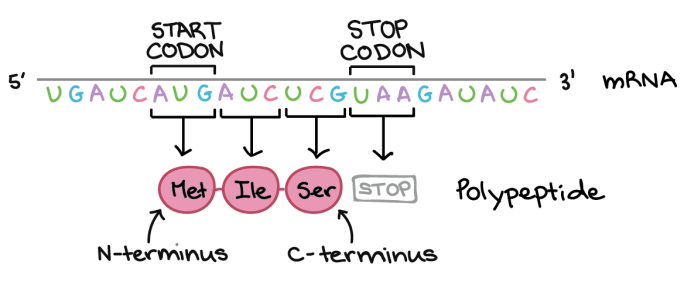
The aim of this paper is to predict the 'Kingdom’ of organisms by codon usage frequencies. We process thousands of data in a UCI database, which are divided into training set and test set. Then we balance the training set with SMOTE and ENN. Neural Network, Random Forest and Support Vector Machine were used to build the model. In each model, we use 5-fold cross validation to tune the parameters. Finally, we evaluated the models with accuracy and Macro-F1. Our models fit the dataset with the accuracy of more than 80% and Macro-F1 of more than 70%.

## Keywords

Codon Usage Frequency, Neural Network, Random Forest, Support Vector Machine

1. **Introduction**

Different organisms not only differ in the amino acid sequence of their proteins but also the degree to which they use synonymous codons for different amino acids. The inherent redundancy of the genetic code allows the same amino acid to be specified by one to five different codons, so there are many different nucleic acids to describe the primary structure of a given protein. Therefore, the coding DNA sequence can carry more information than is required to code the amino acid sequence. To simplify the procedure of nucleic acids classification, instead of analyzing individual nucleotide sequences themselves, it’s much easier to study it from the usages of different synonymous codons. With the information of codons usages of different organisms provided on UCI, the input to our algorithm is the codon usage frequencies in the genomic coding DNA of a large sample of diverse organisms from different taxa tabulated in the CUTG database. We then use Neural Network, Random Forest and Support Vector Machine to output a predicted Kingdom of that Specific species.



*Figure 1. Codons in mRNA*

1. **Related work**

As early as 1994, M.W. Craven was keenly aware that as laboratories around the world produced more and more DNA sequence data, scientists could use efficient computational analysis techniques such as machine learning for gene identification[1]. But at that time, only signal search and content search were used for gene identification. In 2000, Suzuki, K has compiled the frequencies of each of the 257 468 complete protein-coding sequences (CDSs) from the international DNA sequence database[2]. A list of the codon usage of genes and the sum of the codons used by each organism can be obtained to further analyze for variations In codon usage among different genomes. In the field of bioinformatics, various machine learning methods like Support Vector Machine (SVM) or Artificial Neural Network (ANN) are used to analyze and classify gene sequences[3]. The Back Propagation Neural Network is very effective to classify the hypertension gene sequence and identify the disease with 90% accuracy rate for small number of sample (80). Also, deciphering functional interactions between proteins is one of the great challenges in biology. Sequence-based homology-free encoding schemes have been increasingly applied to develop promising protein–protein interaction (PPI) predictors by means of statistical or machine learning methods. Here scientists analyze the relationship between codon pair usage and PPIs in yeast. They show that codon pair usage of interacting protein pairs differs significantly from randomly expected. This motivates the development of a novel approach for predicting PPIs, with codon pair frequency difference as input to a Support Vector Machine predictor, termed as CCPPI[4]. Zurab Bzhalava’s research pays more attentions on machine learning for detection of viral sequences in human metagenomic datasets[5]. He trained Random Forest and Artificial Neural Network using metagenomic sequences taxonomically classified into virus and non-virus classes. The algorithms achieved accuracies well beyond chance level and two codons (TCG and CGC) were found to have a particularly strong discriminative capacity.

1. **Dataset and Features**

The dataset examined codon usage frequencies, which is the number of a specific codon divided by the number of total codons, in the genomic coding DNA of a large sample of diverse organisms from different taxa tabulated in the CUTG database. The first column 'Kingdom' is a 3-letter code, which is the output of the dataset. The second column 'DNAtype', The third column ’SpeciesID’, The forth column ’Ncodons’ and the fifth column ’SpeciesName’ are not related to our topic. The headers from the sixth column to the sixty-ninth column are nucleotide bases, including 'UUU', 'UUA', 'UUG', 'CUU', etc. and entries are the codon usage frequencies, which are used as input. Because the input in the data set is codon frequency, so we don’t need to do preprocessing, including normalization or standardization. The resultant dataset then consists of 13026 organisms of which 126, 2919, 220, 18, 2523, 1345, 2077, 572, 215, 180 and 2831belong to the archaea, bacteria, bacteriophage, plasmid, plant, invertebrate, vertebrate, mammal, rodent, primate and virus kingdoms, respectively. Firstly, we exclude the genome entries classified as ‘plm’ mostly to avoid extremely imbalanced classes in our machine learning models, since there are only 18 plasmids. Then, We divide the entire dataset into 80% for training data and 20% for test data.

Table 1. Number of samples in the original training dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Kingdom | archaea | bacteria | bacteriophage | plant | invertebrate |
| Number | 103 | 2353 | 179 | 2026 | 1071 |
| Kingdom | vertebrate | mammal | rodent | primate | virus |
| Number | 1671 | 444 | 172 | 143 | 126 |

But it is obvious that after excluding this class, the unbalanced classes still exist, and the class with the most samples has more than two thousand more samples than the class with the fewest samples. From the perspective of the data set, we need to change the original unbalanced data set distribution to obtain a more balanced data distribution, and then use the corresponding learning algorithm to obtain a suitable model. The main method to change the data distribution is resampling, including undersampling and oversampling. When there is a lot of data and the class imbalance is not that large, undersampling may be effective[6]. But our data set is quite unbalanced so that we need to delete a large amount of data. Therefore, we have chosen a fairly popular oversampling method SMOTE. The main idea of SMOTE is to achieve the purpose of balancing the categories by generating new samples from some minority samples that are close to each other[7]. But in the process of training the model, we found that this method can generate noisy samples by interpolating new points between marginal outliers and inliers. This issue can be solved by cleaning the space resulting from over-sampling. We combine over- and undersampling using SMOTE and Edited Nearest Neighbours to solve the problem of unbalanced sample category distribution and overfitting[8].

Table 2. Number of samples in the resampled training dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Kingdom | archaea | bacteria | bacteriophage | plant | invertebrate |
| Number | 2353 | 2041 | 2349 | 2100 | 2158 |
| Kingdom | vertebrate | mammal | rodent | primate | virus |
| Number | 2164 | 2329 | 2352 | 2352 | 2057 |

1. **Methods**

We used three methods to solve this classification problem. The algorithms of each model are described as follow.

**4.1 Neural Network**

Neural networks are a class of models that are built with layers. The architecture of it could be seen in the figure below.

Diagram, schematic

Description automatically generated

*Figure 2. Neural Network*

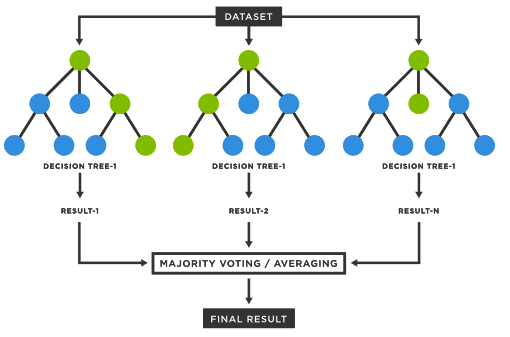
The main procedure of implementing this model is first to perform forward propagation to obtain the corresponding loss. Then, backpropagate the loss to get the gradients. After that, use the gradients to update the weights of the network. In this way, we can train the weight parameters and obtain a basic neural network model.

The cost function of forward propagation algorithms is as below. The is the output, which is also shown in the figure above. K is the number of labels; in our case it is 10.

Backpropagation is a method to update the weights in the neural network by taking into account the actual output and the desired output. For each node in layer, we compute an “error term” that measures how much that node was “responsible” for any errors in our output. Then the (unregularized) gradient for the neural network cost function can be obtained and the weight can also be updated. The additional term regularization is also added when computing the backpropagation. After the calculation of cost function and gradient computation, we used scipy's minimization to learn a good set parameters.

**4.2 Random Forest**

The Random Forest (RF) classifier is an ensemble method that combines the results of different decision trees by voting across them. RF classifiers are advantageous in preventing over-fitting and handling large datasets with high dimensionality.

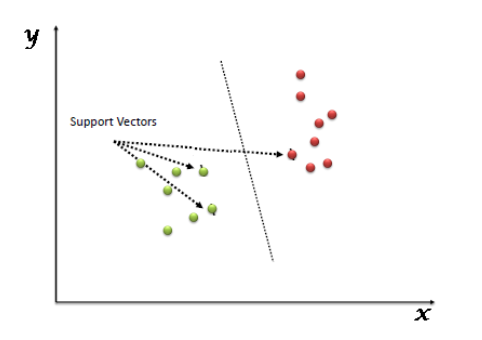


*Figure 3. Random Forest*

In this paper, each decision tree arrives at a different prediction based on the predictors and the training data used, both of which are randomly chosen.

**4.3 Support Vector Machine**

The Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well [9].



*Figure 4. Support Vector Machine*

Since the SVM code of the assignment does not support multi-classification and the number our training data exceeds 20,000, we are advised to use highly optimized SVM toolboxes such as scikit-learn.

1. **Results and Discussion**

**5.1 Statistical Scores**

The following metrics are commonly used to assess the performance of classification models[10]. TP, FP, TN, FN each represents true positives (observations that were correctly classified as positives), false positives (observations that were incorrectly classified as positives), true negatives (observations that were correctly classified as negatives), and false negatives (observations that were incorrectly classified as negatives). Macro F1-averaging is performed by first computing the F1-score per class/label and then averaging them[11]. We decide to use accuracy and macro-F1 score as our main metrics to evaluate our model. Because in this way we can select a model based on a balance between precision and recall.

|  |  |  |
| --- | --- | --- |
| Metric | Formula | Interpretation |
| Accuracy | (TP+TN)/(TP+TN+FP+FN) | Overall performance of model |
| Precision | TP/(TP+FP) | How accurate the positive predictions are |
| Recall Sensitivity | TP/(TP+FN) | Coverage of actual positive sample |
| Specificity | TN/(TN+FP) | Coverage of actual negative sample |
| F1 score | 2TP/(2TP+FP+FN) | Hybrid metric useful for unbalanced classes |

Table 3. commonly used metrics

**5.2 Neural Network**

For this model, we split the dataset into training set (60%), validation set (20%) and test set (20%). Then we do the data balancing to the training set data. We didn’t use k-fold to this model because it will take much longer time to train and test each fold of data and we do have enough training set to train the model. We use the training dataset to train the weight parameters and use validation set to select the hyperparameters of the model. At the end, we use the test dataset to do the final evaluation of the model.

The training sequence was first to select the hidden layer size also known as the units’ number of each layer. Then we select the best parameters of the first layer according to the validation dataset accuracy. After the first layer had been decided, the feature of the first layer was fixed and the second layer was trained as well. We added two more hidden layers in total and observed how it influenced the model.

Our first version of the model has only one hidden layer with the randomly chosen parameters. We got an acceptable result with the validation accuracy of 76.94%. Then we changed the hidden layer size to see how it influenced the result.

*Figure 5. Validation accuracy according to the hidden size.*

As we can see in the figure5, the accuracy changes quite randomly when changing the hidden layer size. As the number of input units is 63 and the output is 10, we didn’t further add the units’ number. We choose the hidden layer size as 32 since it once reached the highest level. Then we chose the parameters of the first layer. First, we choose the lambda, it is the parameter that influence the regularization degree. The trendy of validation accuracy according to lambda is not easily observed as well. But there is an obvious drop down after reaching the highest point 0.3. Thus, we took 0.3 as the lambda of the first layer. The last parameter of the first layer is maxiter. It indicates the amount of training times. Both too high and too low are not good for the training. If it is too high, it will increase the over fitting degree and the accuracy of the validation will decrease. For the maxiter value of the first layer, we took 500 as it has the highest result.

After choosing the parameters of the first hidden layer, we added the second hidden layer to the model. We kept the parameters of the first fixed and used the same method the train the weight parameters of the second hidden layer. The increase of validation accuracy can be observed. The accuracy range of one hidden layer is 75.86% ~81.32%. After tunning the parameters, the accuracy range of two hidden layers is 80.32%~83.24%. The difference between best solution (maxiter=600; lambda=0.3; hidden layer size=32) of two models is about 1.92%. The accuracy of model is not improved so much after adding the second hidden layer.

The third hidden layer is also added to see if it can also improve the model. The tunning method and adding principle stay the same. Then we got the best accuracy of 84.36% with the parameters of (maxiter=500; lambda=0.8; hidden layer size=32). This is the best validation accuracy we got so far, so we choose this model as the final one of the neural networks.

After accruing the model, we use test dataset to evaluate the behavior of the trained model. From the classification report, we find that the accuracy of the test dataset is 84% and the macro-F1 score is 74%. Compared to the training dataset (accuracy:93%; macro-F1 score:92%), the test error is lower than the training error and the training error is pretty low already, the model is overfitting. It also has high variance. Also, we can see it from the figure4 that class0, class2 and class8 have more mistakes (accuracy less than 50%), because the amount of data of these classes is very small. Thus, it is hard to predict these classes.

**Calendar

Description automatically generated**

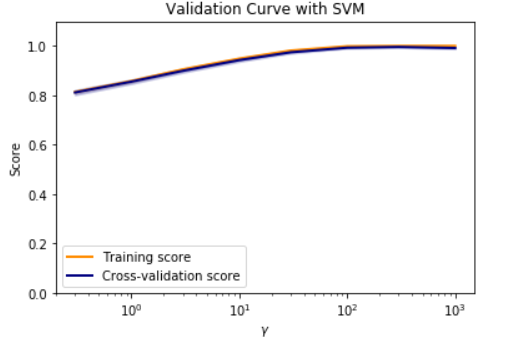
*Figure 6. Confusion matrix of the test dataset.*

**5.2 Random Fores**

The number of trees is initialized to be 500 for building the random forest model, and arrived at reduced branches based on the random feature selection in the tree splitting process. Using 5-fold cross-validation, we tuned the number of trees from 500 and arrived at an optimal value of 300 for the kingdom classification task with an overall accuracy 0.9143 , and 0.8680 of the macro-F1 score.

**5.3 Support Vector Machine**

According to Andrew Ng[11], we choose to use SVM with Radial Basis Function (RBF) kernel when n (number of features) is small (1-1000), m (number of samples) is intermediate (10-10000). When training an SVM with the Radial Basis Function (RBF) kernel, two parameters must be considered: C and gamma[12]. The parameter C, common to all SVM kernels, trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly. gamma defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected. validation\_curve[13] is used to tune these two parameters. 5-fold cross-validation is used in this method and the results are the accuracy of training and the accuracy of cross-validation which determine whether these two parameters fit the SVM model.



*Figure 7. Validation Curve when tuning gamma*

Both parameters are initialized to be 0.01 for building the SVM model, and arrived at an optimal value of C= 10 and gamma= 300 for the kingdom classification task with an overall accuracy 0.9431, and 0.8998 of the macro-F1 score.

1. **Conclusion and Future Work**

According to the result, the Support Vector Machine model has the highest performance (0.8998 macro-F1 score), the next highest model is Random Fores model with the macro-F1 score of 0.8680. The lowest one is neural network as its macro-F1 score is 0.74. The reason of the first two is higher than the neural network could be that they use the k-fold method to train the model. It could make the model has more training data and improve the model. One other reason could be that neural network need more data to make predictions less mistakes, SVM and RF require less input. The dataset input is not large enough to neural network.

About future work, due to the long running time, we didn’t implement the k-fold validation and draw the learning curve, we would like to explore more if we have more time. Also , we could calibrate better when choosing the hyperparameters .

1. **Contributions**

Zeping is responsible for the data balancing and Neural Network. Tao contributes to the related work, data balancing, Random Forest and SVM.

1. **References**

[1] Craven, Mark W., and Jude W. Shavlik. "Machine learning approaches to gene recognition." IEEE Expert 9.2 (1994): 2-10.

[2] Suzuki, K., Krenker, A., Bester, J., and Kos, A. Introduction to the Artificial Neural Networks, Artificial Neural Networks - Methodological Advances and Biomedical Applications. 2011. https://doi: 10.5772/644

[3] S. Zaman and R. Toufiq, "Codon based back propagation neural network approach to classify hypertension gene sequences," 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2017, pp. 443-446, doi: 10.1109/ECACE.2017.7912945.

[4] Zhou, Yuan, et al. "Can simple codon pair usage predict protein–protein interaction?." Molecular BioSystems 8.5 (2012): 1396-1404.

[5] Bzhalava, Zurab, et al. "Machine Learning for detection of viral sequences in human metagenomic datasets." BMC bioinformatics 19.1 (2018): 1-11.

[6] Korstanje, Joos. “SMOTE | Towards Data Science.” Medium, 30 Aug. 2021, towardsdatascience.com/smote-fdce2f605729.

[7] "Issue 6: Unbalanced Data Processing." ZhiHu column, 2016, zhuanlan.zhihu.com/p/21406238.

[8] “4. Combination of over- and under-Sampling — Version 0.8.1.” Imbalanced-Learn, 2021, imbalanced-learn.org/stable/combine.html#combine.

[9] Ray, Sunil. “SVM | Support Vector Machine Algorithm in Machine Learning.” Analytics Vidhya, Aug. 2021, [www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code](http://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code).

[10] Amidi, A. (n.d.). Machine Learning tips and tricks cheatsheet. From stanford.edu: https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks

[11] Ng, Andrew. “| Online Courses & Credentials From Top Educators. Join for Free.” Coursera, 2021, www.coursera.org/learn/machine-learning/supplement/pSe2X/lecture-slides.

[12] “1.4. Support Vector Machines.” Scikit-Learn, 2021, scikit-learn.org/stable/modules/ svm.html#svm-kernels.

[13] “Sklearn.Model\_selection.Validation\_curve.” Scikit-Learn, 2021, scikit-learn.org/stable/modules/generated/sklearn.model\_selection.validation\_curve.html#sklearn.model\_selection.validation\_curve.