GENE ONTOLOGY (GO) PREDICTION USING MACHINE LEARNING METHODS

HAOZE WU [*]

Department of Mathematics and Computer Science, Davidson College

Box 5996, Davidson, North Carolina, 28035, USA

anwu@davidson.edu

YANGYU ZHOU [*]

Biology Department, Davidson College

Box 6891, Davidson, North Carolina, 28035, USA

jozhou@davidson.edu

We aimed to apply machine learning to predict whether a gene is involved in axon regeneration. We extracted 31 features from different databases and trained five machine learning models. Our optimal model, a Random Forest Classifier with 50 sub-models, yielded a test score of 85.71%, which is 4.1% higher than the baseline score. We concluded that our models have some predictive capability. Similar methodology and features could be applied to predict other Gene Ontology (GO) terms.

Keywords: Gene Ontology; Machine Learning; Axon Regeneration

1. Introduction

Demand for bioinformatics methods to analyze genomic information has never been greater. Researchers are particularly interested in the overall functions of genes. Despite the best curation efforts, functions of genes remain incomplete and biased toward much-studied genes and pathways.

Machine learning methods have been extensively used to identify functional elements of a gene, such as transcription start sites, promoters, and transcription factor binding sites [1]. However, few studies have applied machine learning to predict whether a

[*] HaoZe Wu and YangYu Zhou are co-authors, they contributed equally to this work.

gene has certain function, that is, whether a gene is involved in a particular biological process.

As a proof-of-concept, in this paper, we aim to predict whether a gene is involved in axon regeneration in *C. elegans*. To establish the dataset, we find definitively positive or negative samples from literature and engineer features with the information provided with several on-line databases. We train five classifiers and assess the performance of each one of them.

In the following sections, we will first briefly introduce some biology background and show how we extract features and establish the dataset. After that, we will present our methodology of training and evaluating different machine learning models. Finally, we will present and discuss our results.

2. BACKGROUND

As a proof-of-concept, we chose to test the performance of our tool on axon regeneration (GO:0031103) in model organism *Caenorhabditis elegans* genome. We chose to analyze the genome of *C. elegans*, because of its well-annotated genome and vast amount of knowledge available in the scientific community.

Gene ontology associates a gene and the biological function of the gene's product. There are 24500+ gene ontology terms, each term describes a biological function. The structure of GO terms resembles a directed acyclic graph (DAG), with the most general term (biological process, GO: 0008150) at the root of the graph, and the most specific terms (including the one we are concerned with, axon regeneration, GO:0031103) at the leaves.

Damaged axons can sometime regenerate, restoring connections to target tissues after injury or disease. However, axon regeneration is initiated by intrinsic and extrinsic signals at the site of injury, such as neuron type, age, type of surrounding tissues and more. It is thought that the heterogeneity arises partially from regulation at the genetic level, including methylation of genome and chromatin structure. Transcription of certain key genes and synthesis of proteins are also required for successful axon regeneration.

To achieve the highest predictive capabilities, eleven features were selected (Table 1).

Table 1. Features, and their sources, used as input to the machine learning models.

#	Feature	Description	Database
1	Gene length	The length of a gene (in basepair) starting from its transcription start site (TSS)	UCSC
2	RNA-seq expression	Gene expression level under standard conditions	modENCODE
3	Mean GO distance	The mean distance of all GO terms of a gene to GO:0031103 (axon regeneration) in the graph representation of all GO terms	AmiGo2 WormBase
4	Minimum GO distance	The minimum distance among the distances between a gene's GO terms and GO: 0031103 (axon regeneration).	AmiGo2 WormBase
5	ChIP-seq score	ChIP-seq expression profile on young adult worms.	WormBase
6	Avg. Sequence conservation	Average conservation score of 15% length of the gene upstream of its TSS	WormBase
7	Std dev Sequence conservation	Standard deviation of score of 15% length of the gene upstream of its TSS	WormBase
8	Ratio of positive interacting proteins	Among the proteins that interact with a gene, what ratio of them are products of positive genes.	STRING
9	Number of positive interacting proteins	Among the proteins that interact with a gene, how many of them are products of positive genes.	STRING
10	Phenotype association	Whether a gene has the phenotypes that are most common in the dataset	STRING
11	GO enriched terms	Whether a gene has the GO terms that are most common in the dataset.	WormBase

2.1. ***Sequence conservation***

Conserved sequences are identical DNA sequences across different species. A region of high conservation score indicates that this segment of DNA is highly identical among species, suggesting that it may encode proteins of important biological functions so that it has been protected throughout evolution despite speciation.

2.2. ***Protein-protein interaction***

Proteins often interact with other proteins when performing biological functions such as signal transduction and cellular metabolism. Multiple transcription factors often act together when regulating gene expressions.

2.3. *Phenotype association*

Phenotype is the observable characteristics of an organism, such as morphology and behavior. The phenotype largely results from the genotype, the expression of genes. In this paper, the phenotype of a gene refers to the abnormal changes when the gene is deleted.

2.4. *ChIP-seq*

Chromatin Immunoprecipitation (ChIP) in combination with massive parallel DNA sequencing is a technique that identifies the binding sites of DNA binding proteins. Promoters of genes usually are high in ChIP-seq score due to high levels of transcription factor binding.

2.5. *RNA-seq*

RNA sequencing quantifies the levels of transcriptions of genes genome-wide. Similar to ChIP-seq data, RNA-seq measures how active the gene is at a given time. We used RNA-seq data of young adult worms living at standard conditions.

3. Data Preprocessing

Our samples are different genes, and our target variable is a boolean value that represents whether a gene is involved in axon regeneration.

To determine whether a gene is involved in axon regeneration, we searched literature for definitive positive examples (genes that are for sure involved in axon regeneration) and negative examples (for sure not involved). We found three papers that describe 347 positive genes and 1401 negative genes in total[2,3,4].

Since most of the datasets have different formats, we wrote programs to extract useful information and compiled them into a structured format. Here we will briefly introduce the algorithms we used to extract some features and make further clarification on some features.

3.1. *GO distance*

Each gene has several gene ontology terms. To get the distance between each of a gene's GO terms to the GO term GO:0031103 (axon regeneration), we first created a graph representation of all GO terms that captured the relationships between them, and then obtained the shortest distance between two GO terms with the Dijkstra's algorithm.

3.2. *Protein interaction*

We first constructed a graph representation of protein interactions. Each vertex constitutes a protein, and each edge *xy* constitutes an interaction between protein *x* and *y*. Given a gene, we searched its neighborhood and looked for the neighbors that are protein products of the known positive examples.

The data we obtained from different databases were well-established, so that we did not have a lot of missing values. We imputed missing values with mean imputation. Then, we randomly chose 20% of the dataset (350 samples) as the testing set, and used the rest 80% (1398 samples) for training and validation. From now on in this paper, for sake of convenience, we will refer to the 80% samples for training and cross-validation as the ``dataset." We will refer to the rest 20% samples as the ``testing set," as usual. Finally, we mean-normalized the dataset and the testing set.

4. Experiment

We trained five different classifiers, *K*-nearest Neighbor (*K*-NN), Random Forest (RF), Logistic Regression, Support Vector Machine (SVC), and Neural Network (NN). The Neural Network model we used was a single-layer feedforward Neural Network Classifier. We used the Python Scikit-learn implementations of these models scikit-learn [5].

We evaluated a model primarily based on its accuracy. In this paper, we will use the words ``score" and ``accuracy" interchangeably. We will use ``test score" to refer to the accuracy of a model on the test set; and we will use ``validation score" to refer to the mean cross-validation accuracy of a model.

Besides the scores, we were also interested in the models' performance on predicting positive samples. Therefore, we recorded the precision and recall on the test set for each model.

For each classifier, we tuned several hyperparameters. For each hyperparameter, we conducted a grid search for the optimal value. Table 2 shows the hyperparameters we tuned for each model, and the ranges of the grid searches.

Table 2. Hyperparameters tuned for different models.

Model	Hyperparameters and Range of Grid Search
Logistic	- Penalty parameter {0.001, 0.01, 0.1, 1, 10}
K-NN	- # of neighbors (1-13 with step size 2) - Distance metric {Euclidean, manhattan} - Weight function {uniform, distance}
RF	- # of components (10 – 100 with step size 10)
SVC	- Kernel {linear, rbf, sigmoid, 2nd poly} - Penalty parameter {0.001, 0.01, 0.1, 1, 10}
NN	- # of perceptrons (2 – 16 with step size 2) - Penalty parameter {0.0001, 0.001, 0.01, 0.1, 1} - Learning rate {adaptive, constant} - Weight optimization algorithm {l-bfgs, sgd, adam}

Besides, for *K*-NN, SVC, and Logistic, we tried to use Adaboost to enhance their performance. We also tried to enhance the last two classifiers with Bagging. For each model enhanced with these two meta-algorithms, we tuned the number of sub-classifiers of Adaboost or Bagging, with a grid search from 10 to 50, step size 10.

We decided not to use Bagging to enhance *K*-NN because that would create similar effect to a weighted version of *K*-NN, while we would tune the weight function as a hyperparameter for the *K*-NN classifier.

To compare models of a classifier with different values of hyperparameters, we conducted cross-validation. However, due to the small size of our dataset, we wanted to maximize the size of the training set while receiving stable validation scores. Therefore, we used leave-one-out cross-validation, that is, for each sample in the dataset, we trained a model with the rest of the samples, and predicted the target variable of that sample. In this case, the validation set is the entire dataset, and the cross-validation score would be the mean of the accuracies of all predictions (either 100% or 0%).

For each optimized classifier, we reported its validation score, test score, (test) precision, and (test) recall. We also compared the validation scores of the five models horizontally, and chose the classifier with the highest score on the validation set as the model that has the highest predictive capability.

To test whether our models have predictive capability, we also compared our models with the baseline model. Our training set contained 79.9% negative samples, and our test set contained 81.1% negative samples. If a model always predicted false, it would have a validation score of 79.9%, and a test score of 81.1%. Therefore, we decided that to outperform the baseline model, a model must have a validation score of more than 79.9% and a test score of more than 81.1%.

5. Results

In this section, we introduce the training results of the five models in the order of their performances (from low to high). Then we compare the five models horizontally.

5.1. *Logistic Regression*

We tuned the penalty parameter *C* for the Logistic Regressor and found that when *C* = 1, the model had the highest validation score. The model has a validation score of 79.01%, which is worse than the validation score of the baseline model. Therefore, we considered the Logistic Regressor a weak learner and used Adaboost to enhance its result.

As shown by figure 1, as the number of sub-models increased, there was no significant improvement of the validation score. The model with 30 sub-models has the highest validation score. And this model has a test score of 80.57%, a precision of 42.86% and a recall of 9.09%. Though Adaboost did improve the accuracy of the Logistic Regressor, the boosted test score is still lower than the baseline.

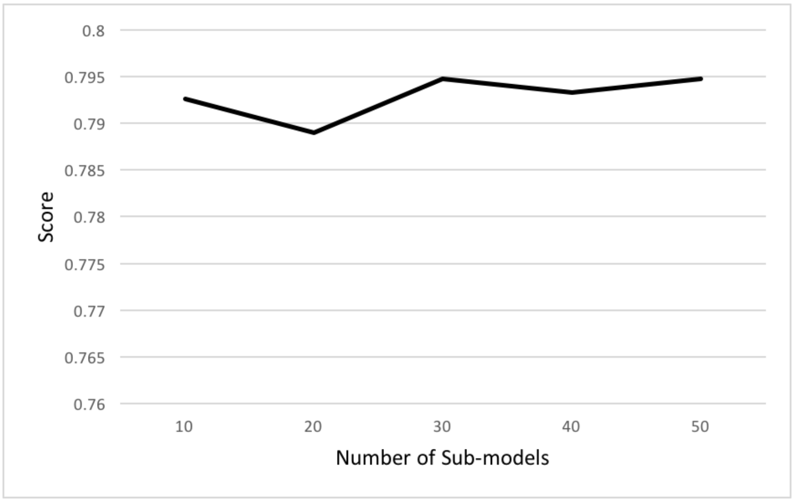

Fig. 1. Validation score vs. number of sub-models for Adaboost (Logisitic, C =1).

On the other hand, Bagging led to even less improvement. Our best Bagging Logistic Regressor contains 40 sub-models, and has a validation score of 79.28%. This minimal improvement suggests that our original model did not have high variation. This is supported by figure 2 which shows that as the number of training examples increased, the training scores and the validation scores came close to each other. This means our model did not have a high variation.

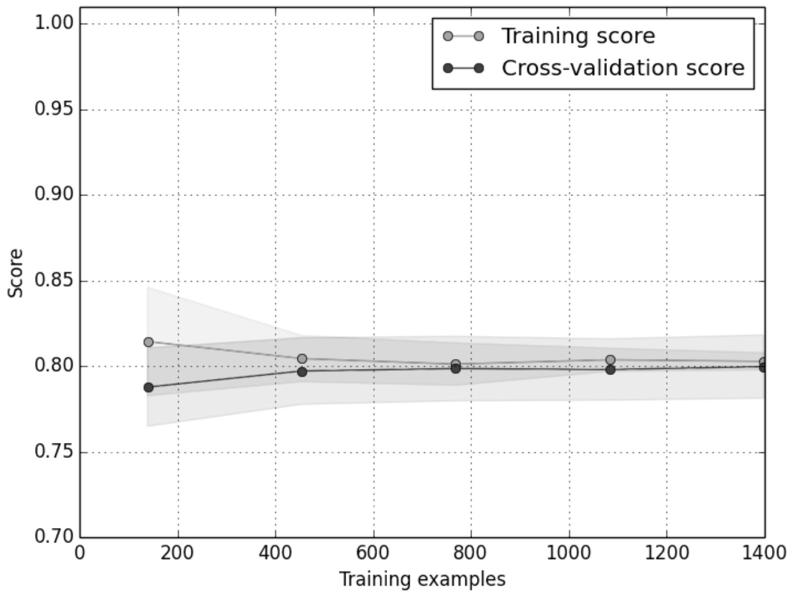

Fig 2. Learning Curve (Logisitic, C = 1).

None of our Logistic Regression model has a test score higher than the baseline model. Therefore, we concluded that Logistic Regressor does not have predictive capability.

5.2. *Neural Network (NN)*

We discovered that the optimal learning rate was "adaptive," and the optimal penalty parameter was 0.01.

Figure 3 shows the validation scores of Neural Network with different weight optimization algorithms and numbers of perceptrons. The validation scores generally increased as the number of perceptrons increased. ``l-bfgs" and ``adam" yields comparable optimal results, while ``adam" is slightly better. Our best Neural Network model uses ``adam" as the weight optimization algorithm, and has 12 perceptrons in the hidden layer. It yielded a test score of 83.14%, a precision of 61.29%, and recall of 28.79%.

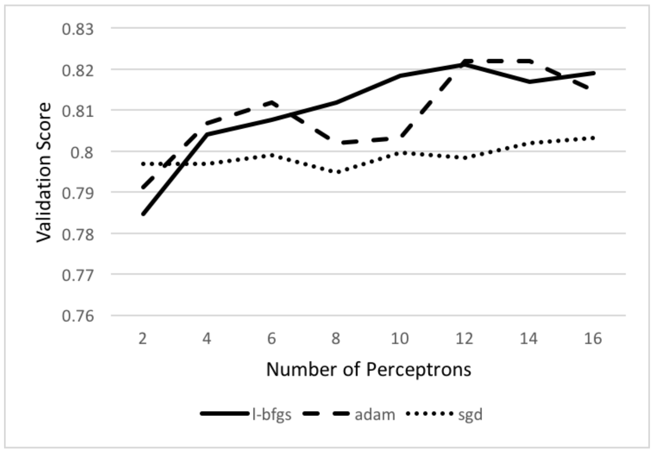

Fig. 3. Cross-validating the number of perceptrons and the weight optimization algorithm for NN.

5.3. *Support Vector Classifier (SVC)*

Figure 4 shows the validation scores of SVC models with different kernels and penalty parameters. As the penalty parameter increased, the validation scores increased for models using the ``rbf'' and the ``2nd degree poly'' kernels. The models using the ``sigmoid'' kernel always yielded the baseline accuracy, 79.9%. The models using the ``linear'' kernel yielded results worse than the baseline accuracy.

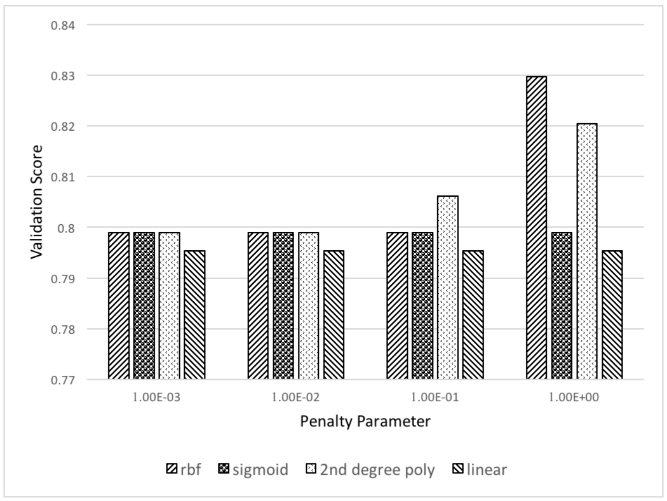

Fig. 4. Cross-validating the kernel and penalty parameter for SVC.

We tried to use Bagging and Adaboost to enhance the SVC, however, we received worse validation scores.

We discovered that the optimal penalty parameter of SVC is 1, and the optimal kernel is ``rbf." Our optimal SVC model has a test score of 84.00%, a precision of 85.71%, and a recall of 18.18%.

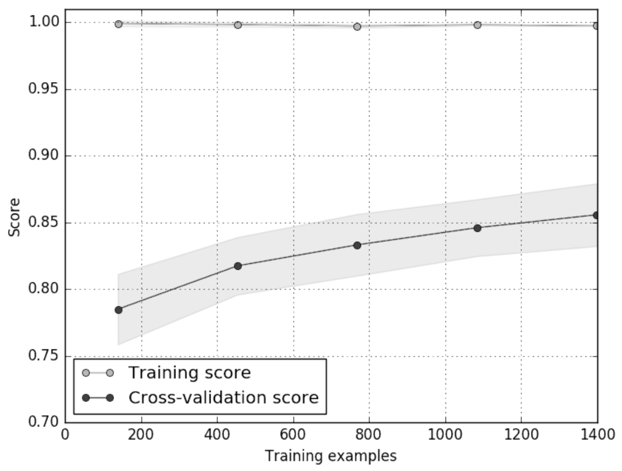

Fig. 5. Learning Curve (SVC, 'rbf' kernel, C = 1).

5.4. *K-nearest Neighbors (K-NN)*

As shown by figure 6, models that used manhattan distance metric, and weighed the neighbors using the ``distance" weight function (weighing neighbors by the inverse of their distances to the input instance), were strictly better than the other models. On the other hand, we observed that the validation scores for manhattan distance models, and Euclidean distance models continued to increase as the number of neighbors n increased, and there was no sign of convergence. Thus, we hypothesized that we had not found the optimal values of *n* for those two types of models and decided to cross-validate *n* by conducting a grid search with a wider range.

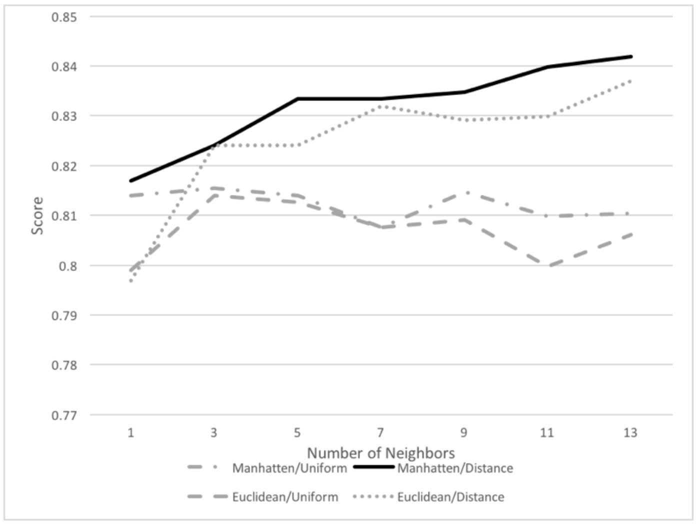

Fig. 6. Validation scores of K-NN.

Figure 7 supports our hypothesis. For the Euclidean distance models, the validation scores increased after n > 13 and started to decrease when n ≥ 17. For the manhattan distance models, the validation scores fluctuated and showed signs of convergence after n > 13. We discovered that the optimal *K*-NN classifier is a manhattan distance model that searches for 21 neighbors. This model yielded a validation score of 84.26%, a test score of 85.14%, a precision of 100%, and a recall of 21.21%.

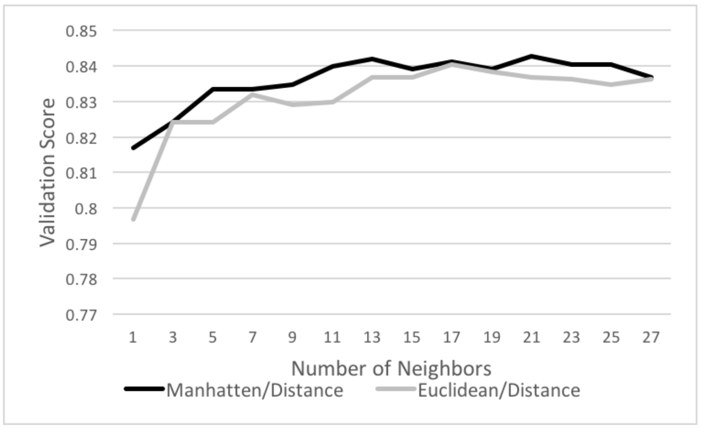

Fig. 7. Further cross-validation on the number of neighbors.

We observed that for the unweighted *K*-NN models, the optimal value of *n* was only 3, which was significantly smaller than that for the weighted *K*-NN models. We hypothesized that weighted *K*-NN models correctly captured the characteristics of the dataset that predicative information was available in a fairly large neighborhood of an input instance, while the closer a neighbor was, the more important information it could provide.

We tried to use Adaboost to enhance the *K*-NN, but there was no significant improvement.

5.5. ***Random Forest (RF)***

As shown by figure 8, the optimal number of sub-models *n* for Random Forest is 50. Normally, as *n* increases, the accuracy increases until convergence. However, this was not the case for us, the model's performance improved gradually when n ≤ 50, but when n > 50, the performance dropped and started to fluctuate.

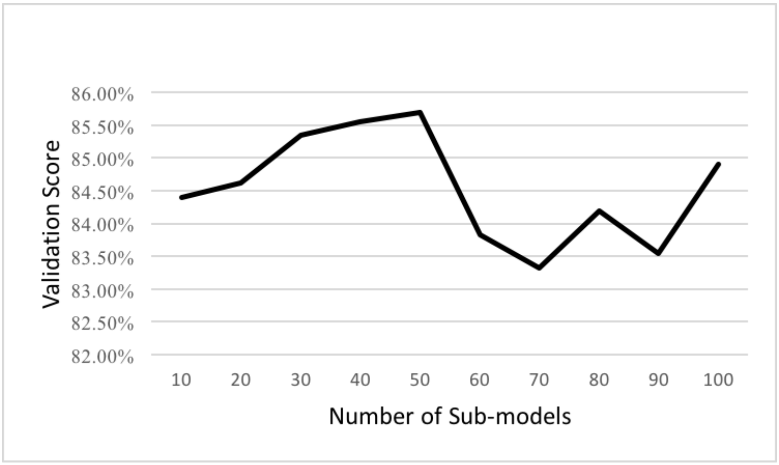

Fig. 8. Validation score vs. number of sub-models for Random Forest.

Our best Random Forest model, with n = 50, yielded a test score of 85.71%, precision of 83.33%, and a recall of 30.30%.

5.6. ***Overall Analysis***

In this subsection, we first compare the performance of different classifiers; then we talk about some unrelated observations and hypotheses of our experiments, including

- The inefficiency of Adaboost;
- The insufficiency of training samples;

- The precision/recall trade-off;
- The decision boundary between positive and negative samples.

Table 3. Comparison between different types of classifiers.

Model	Validation Score	Test Score	Precision	Recall
Logistic	79.68%	80.57%	42.86%	9.09%
NN	82.19%	83.14%	61.29%	28.79%
SVC	82.98%	84.00%	85.71%	18.18%
K-NN	84.26%	85.14%	100%	21.21%
RF	85.69%	85.71%	83.33%	30.30%

Table 3 shows the validation score, test score, test precision, and test recall of the optimal model for each type of classifier. Except for the Logistic Regressor, all models outperformed the baseline model. The Random Forest Classifier had the best performance, and *K*-NN was the second best model. The former yielded a test score of 85.71%, and the latter 85.14%, which are both higher than the baseline test score 81.1%. The fact that most of our models outperformed the baseline model suggests that some features have certain predictive capability. However, more predicative features are needed in order to achieve a higher performance, as our best model only outperformed the baseline model by 4.6%.

We observed that Adaboost generally resulted in marginal improvement or even mild decrease in the validation scores. We believed that this can be explained by the high dimensionality of the feature space, and the small size of our dataset. Adaboost is a bias-increasing method which theoretically produces weak models that barely outperform the baseline model. However, a very small dataset and a high-dimensional feature space might worsen the overfitting problem and result in sub-models with accuracies lower than the baseline.

We believed that the small size of training set might also lead to sub-optimally trained models. As shown by figure 5, the validation score curve of SVC does not seem to converge when the size of the training set is about 1400. It is likely that if we acquired more training examples, we would be able to obtain models with higher scores.

In terms of precision and recall, all of our models tended to have high precisions and low recalls. Intuitively, this suggests that a model would decide that a sample is positive only when the model is very ``confident" about it. We believed this high confidence threshold can be explained by the unbalanced nature of our dataset, where the models had more incentives to predict negative than predict positive.

Finally, the different behaviors of different classifiers shed light on the decision boundary between the negative and positive samples in the feature space. Logistic

Regression, and SVC with linear kernel yielded results worse than the baseline. On the other hand, Random Forest, *K*-NN, and SVC with 'rbf' kernel yielded the highest accuracy. Based on this observation, we hypothesized that the decision boundary is not linear.

6. Conclusion

We started out aiming to create a classifier that predicts whether a gene is involved in axon regeneration in *C. elegans*. We found that a Random Forest Classifier with 50 sub-estimators has the highest predicative capability. It yielded a test score of 85.71%, a precision of 83.33%, and a recall of 30.30%. Since the test score is 4.6% higher than the baseline test score, we decided that our model has certain predicative capability and some of the features we extracted were effective. We also believed that similar methodologies in extracting gene features as ours could be applied to the prediction of whether a gene is involved in other biological processes, besides axon regeneration. To improve predicative capability of our models, we would 1) increase sample size through literature search and lab works, 2) extract more predicative features.

References

1. Libbrecht, M. W., and Noble, W. S. 2015. Machine learning applications in genetics and genomics. Nature Reviews Genetics 16:321–332.
2. Bejjani, R. E., and Marc, H. 2012. Neural regeneration in c. elegans. *Annual Review of Genetics* 46:499–513.
3. Lizhen Chen, Zhiping Wang, A. G.-R. T. H. D. Y. S. O. R. B. B. Z. W. Y. J., and Chisholm, A. D. 2011. Axon regener- ation pathways identified by systematic genetic screening in c. elegans. *Neuron* 71(6):1043–1057.
4. Paola Nix, Marc Hammarlund, L. H.-M. L. E. M. J., and Bastiani, M. 2014. Axon regeneration genes identified by RNAi screening in c. elegans. The Journal of Neuroscience
34(2):499–513.
5. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit- learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.