Movement State Classification with C.elegans

By Charlie Schumacher

16 November 2018

Abstract

The Caenorhabditis elegans nematode is frequently used as a model organism in the field of biology due to the relatively simple nature of its structure and the fact that its neurons are almost identical to humans. It is one of a small pool of organisms which has had its entire genome sequenced and boasts the fact that it is the only multicellular organism to have its entire connectome (neural wiring diagram) mapped. The latter fact makes this organism particularly interesting to neuroscientists and biologists who focus their research on the brain and its dynamical properties. In order to effectively track the subtle changes in observable behaviors when modifying C.elegans specimen, location and positioning data needs to be extracted from experimental recordings and labels applied to individual frames which indicate the movement pattern and length of its duration. This is an especially challenging task for C.elegans due to the limited range of movement types it can express and the constant turning even when moving forward. Historically, strong performance has been achieved by using a stratified sampling approach within a single experimental video to label the rest of the video. However, this approach requires a fully annotated video by a human expert to get the appropriate class distribution for training and testing sets and therefore does not contribute to the goal of reducing or eliminating the need for human expert annotation. One of the proposed solutions to this problem is learning movements labels from a pool of existing expert-labeled videos and generalizing them to new videos in order to identify the correct states. In this paper we examine the efficacy of state classification produced by a combination of temporal-engineered features, linear transformations, target-balancing technique, and meta-learners. Our preliminary results suggest that the high degree of noise in the data and severe target-class imbalance cannot be easily overcome through by these methods, and that additional features may need to be engineered as well as non-linear transformations of the dataset.

Introduction

The MedIX laboratory at DePaul University has partnered with biologists at Rosalind Franklin University to build a pipeline for generating C.elegans experiments and generating location features using machine learning techniques. Several experimental recordings have been generated and labeled by a panel of experts (using a majority voting scheme for determining labels) for natural worms as well as various mutants, and a huge number of recorded videos have been generated which do not have annotations yet. Because the process of manually annotating videos can be a very slow process, the goal is to use additional machine learning techniques to apply labels to new videos by training a classifier on the human-annotated videos. The obstacles which have been encountered thus far include a severe imbalance of movement states: the forward-NTD (no turn detected) state consistently represents a number of instances which is multiple orders of magnitude greater than other states such as forward-sharp, reverse-long, reverse-short, and stop.

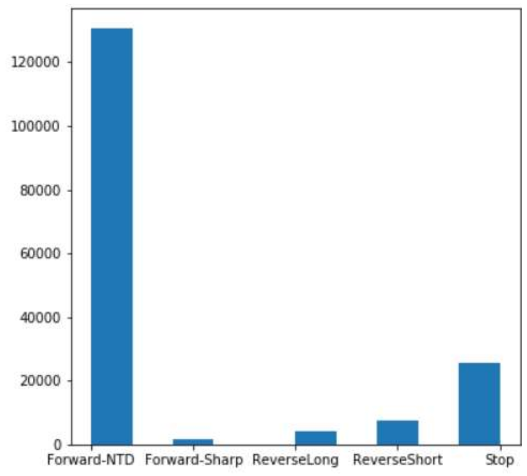

Figure 1: Distribution of target classes in our tph1 mutant dataset

To further complicate the task, the forward-NTD state appears nearly identical to the majority of other states because of the way that worms are constantly turning even when they are wiggling forward in a straight line. This makes turning in a forward motion difficult to distinguish from turning in a turning motion and requires a threshold angle to be set which determines whether or not a turn has occurred, and the engineering of temporal-based location features which can be used in these geometrical calculations. The potential benefit of a well-functioning automatic state-label is the ability to produce a significant number of experiments using C.elegans across a variety of mutations and under different environmental conditions, and quickly identify how the behavior of the worm is modified by analyzing subtle movement patterns rather than simply assessing speed of the worm (which is what is frequently done since it is far easier and reliably consistent).

Methodology and Related Work

In order to evaluate the possibility of using existing expert-labels videos to apply labels to new videos, three full length videos of tph1 mutants (serotonin deficient) with expert-labels are used as a training set to try and predict on an additional tph1 mutant dataset and evaluated by comparing our predicted labels to the expert labels. The dataset includes 67 continuous numerical features which describe location such as position of the head, tail, and centroid of the worm; and movement features which describe the speed, acceleration, etc. The target labels are five different states; the distribution of which can be seen in figure 1.

Limited resources can be found which describe the problem of movement state classification with the specimen of interest in our experiments. However, various well-documented preprocessing techniques exist which can improve that classification abilities when there is an imbalanced target distribution and noisy data, and these are evaluated for our problem in the following analysis. One of these interesting techniques calls for the identification of difficult to classify cases, Tomek-links, using nearest-neighbor analysis and removal of the majority class in those instances [3]. An additional variation to this approach combines the use of Tomek-links with random-under-sampling of the majority class within a training set in order to determine a more effective decision boundary and thin out a noisy and overwhelming majority class distribution [3]. Both these techniques are used in our analysis, in addition to the more aggressive balancing technique of cluster centroids which identifies cases which represent cluster centers in the majority class distribution and removes the rest of the points [6]. This approach aims to improve classification efforts by increasing the distance between target classes in the search space and representing the majority class using significant points that other cases tend to cluster around.

Given the nature of our task and temporal data, variations of our dataset were generated which add temporal features to each instance. In one of these variations, information encoding the attributes in the past five frames and future five frames is encoding into the instance (window-5 dataset). The goal of this method is to allow the classifier to make determinations about the current state based on location and movement attributes leading up to the current state and where the worm will be in the next few frames. The other variation which is utilized encodes fifteen frames in the past and fifteen frames in the future while taking every other frame (window-30 dataset). The aim with this variation is to allow the classifier to make determinations on the state using a larger window of the temporal data. Linear PCA is used in our analysis with the datasets that contain temporal features to reduce high correlation between the temporal features and reduce the number our classifier has to handle to a reasonable amount. PCA is often highly effective in cases with noisy data and correlated features, and it remedies the issue by combining and transforming the feature in such a way that causes them to be uncorrelated and capture the majority of variance in the data within a few components.

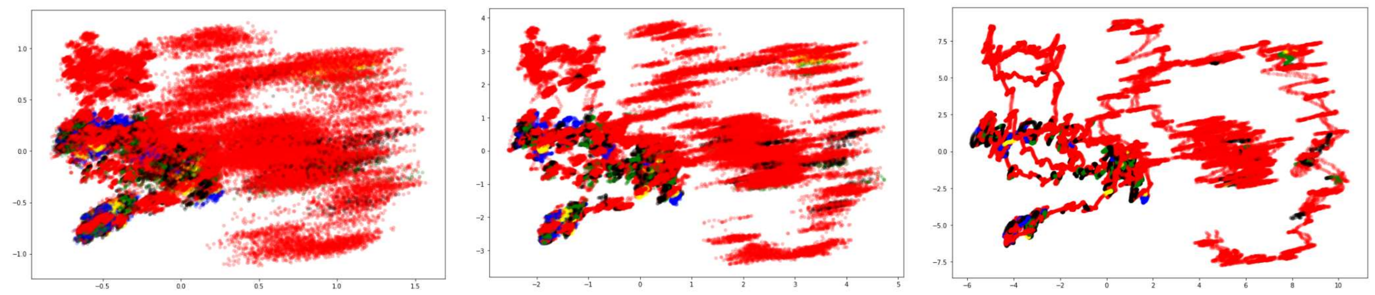

Figure 2: PC1 and PC2 on the training set. From left to right: original data, data with window-5 temporal features, and data with window-30 temporal features

It is evident from visual inspection of the primary principal components (figure 2) that the transformed data is still not easily separable. Adding the temporal features reduces the noise from the majority class and causes the transformed data to condense into tight non-linear form within the feature space. Using non-linear transformations is computationally expensive, and with datasets as large as the ones in our present corpus is becomes borderline intractable in the absence of significant computing power (this is left for exploration in future work).

Because of the intensely noisy patterns in the datasets, a series of weak-learning models and weak-ensembles are used to attempt to generalize various portions of the data. One of the methods of interest we evaluate is a combination of bagging and boosting [8], which suggests that incremental improvements can be gained in classification scenarios by injecting additional randomness into the classifier by training on overlapping datasets and adding weights to individual hard-to-classify cases and models which perform well (in the style of AdaBoost). For the sake of computational cost-efficacy since we have a very large dataset, a mild form of this approach is used which ignores the separate PCA transformation for each bagged-dataset and instead simply looks at bagging and boosting in conjunction when there the original dataset has already been linearly transformed [7]. Another technique of interest that is considered is use of a Rotation Forest [2]. This algorithm is similar to a Random Forest approach, except that each decision tree which is built and injected with randomness at feature selection stages is also combined with a bagged dataset (using 75% of the original training data) that has been linearly transformed. This technique is nearly as computationally expensive as the RotBoost algorithm, but when used in a weak manner with low numbers of estimators it can be calculated in reasonable time and is therefore deemed appropriate for use in our collection of weak learners. The full list of weak-classifiers/ensembles used contains:

- Bagging+Boosting combination (bagging 75% of dataset to produce 5 datasets, 3 boosting iterations on a decision stump)
- Rotation Forest (3 rotated trees built with linear PCA and minimum of 10 samples per leaf)
- Random Forest (3 tree built with a minimum of 30 samples per leaf)
- AdaBoost (5 boosting iterations built on a base decision stump)
- Decision Tree (minimum of 10 sample per leaf)
- Naïve Bayes

The parameters chosen for these classifiers were discovered through experimentation. Making these classifiers even weaker causes them to label every single instance as one class; making them any stronger than they are by allowing them to fit the training data better also causes them to generalize very poorly and label every instance as a single class. Setting any kind of maximum depth to the tree-based methods also caused extremely poor generalization; the limit of 10 samples per leaf was determined to be the best approach since it caused the models to at least show some variety in class labels. An example of how these parameters effect a Random Forest in the case of generalization can be seen in Figure 3.

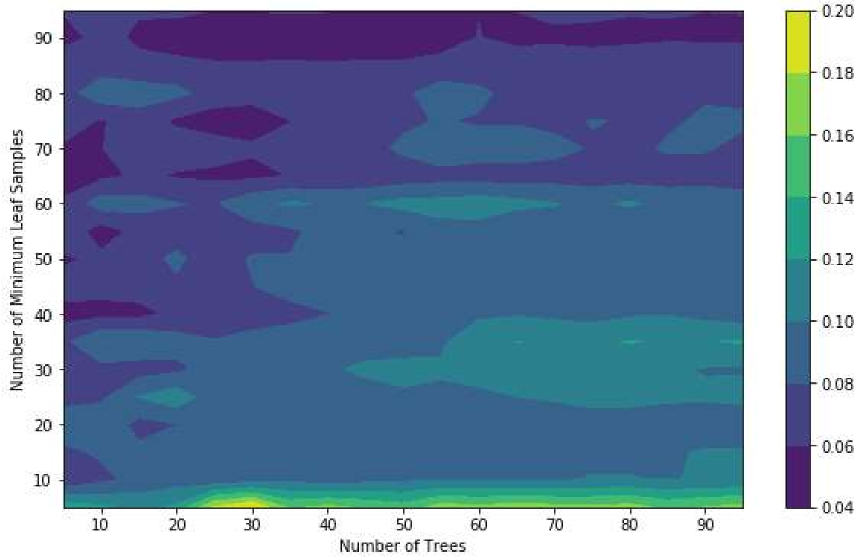

Figure 3: Grid search of parameters for random forest. Color indicates the global F1 score of the model when trained on the Cluster Centroid data. Overwhelmingly, this model does not generalize well to the new dataset.

Related literature exists that describe the impact of using a meta-learner on the classification outputs of several weaker, but well-tuned learners [1][4][5]. In general, these papers describe the phenomenon that the overall performance of a meta-learner is generally dictated by the performance of the components which it comprises. The primary benefit however, comes in the form of incremental improvement by balancing strengths and weaknesses of the different classifiers it is learning from. If one of the classifiers is biased towards a particular class under certain conditions but effectively identifies a different class, and two other classifiers in the ensembles have the inverse of this, weights could be assigned to the outputs of these classifiers within the meta-learner in such a manner that it reduces the weakness of the first class while still maintaining most of the good performance generated by the other two. Numerous approaches exist which can be used in meta-learning, but the one evaluated in this paper is a simple voting scheme which applies the final class label based on the majority vote of all the individual classifiers. The reason that we chose to use “weakly-tuned” classifiers and ensembles to feed into the meta-learner is due to the extremely noise characteristics of this dataset. As previously note, parameters we identified which produced the least skewed distribution of predicted class labels; the overwhelming majority of parameter choices for this dataset caused the models to predict every instance as one class which made for extremely poor predictions on minority classes and eliminated any of the incremental value added by a meta-learning layer.

Figure 4 visually depicts the flow of information in the pipeline constructed. Note that the normalization steps before PCA and use of Naïve Bayes have not been shown but were applied, and that the number of outputs from each sub-classifier into a meta-learner is equal to the total number of dataset variations used (12).

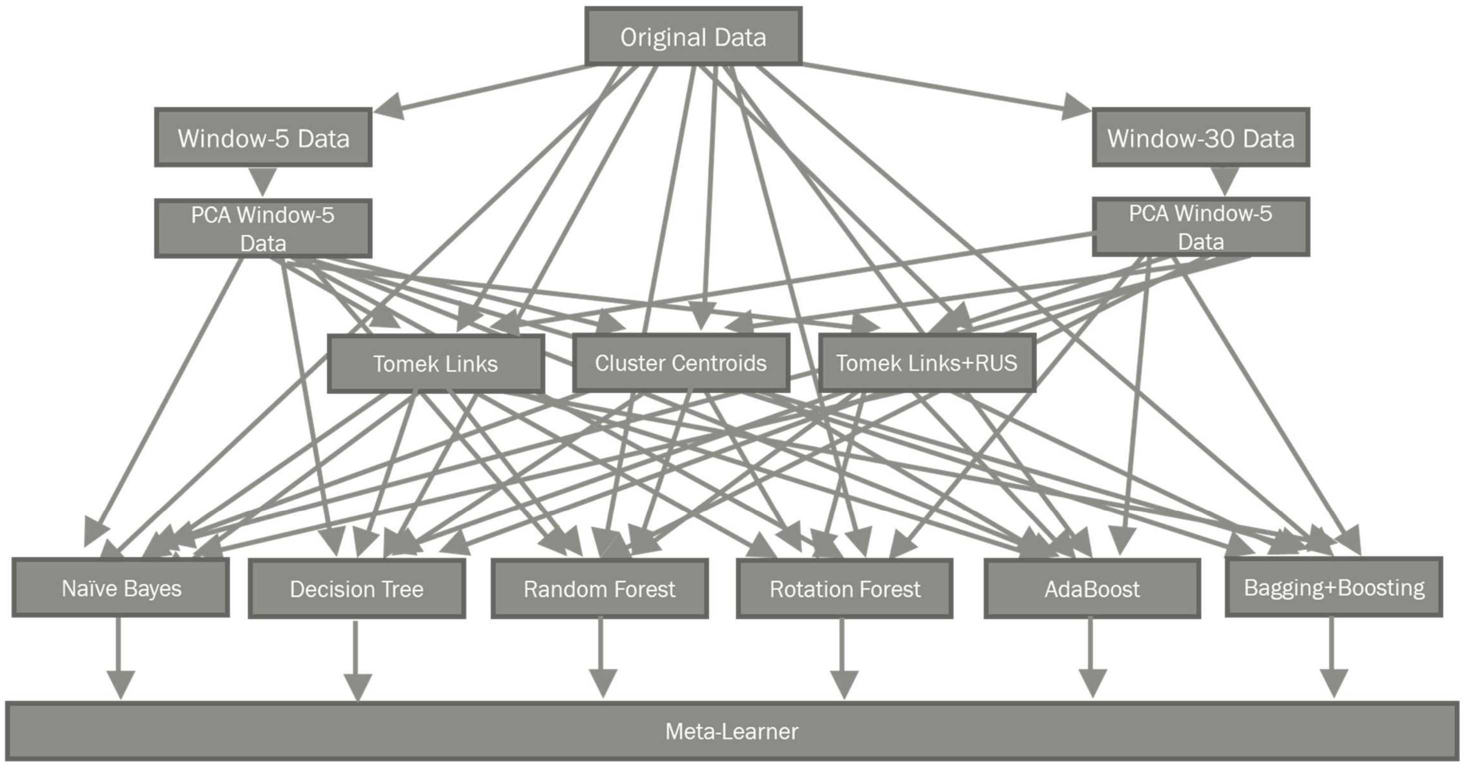

Figure 4: The classification pipeline used for the exploration detailed in this paper

Evaluation of the different individual models and the meta-learner is done through various metrics that reflect the ability of the model to accurately predict the meaningful portion of the overall dataset, as well as its ability to accurately distinguish between minority classes. The metrics used are as follows:

- Overall model accuracy
  - This is calculated as the total number of true positives in the model divided by the total number of cases. This metric is undoubtedly skewed because of the highly imbalanced nature of the data, but when considered in combination with other metrics it acts as a barometer to let us know whether or not an individual model or the stacked meta-learner is predicting the majority of instances as the majority class label or if it is tuned in such a manner that it predicts most of the labels as a minority class. For our purposes with this difficult dataset, an optimal model that we would like to see is getting enough of the minority classes predicted correctly even if it is at the expense of predictions on the majority class. Therefore, an attractive classifier should have a reasonably high accuracy while still producing a high global F1 score
- Global F1 score
  - In lieu of analyzing the individual metrics for each class, for each individual model, on each individual dataset, a summary metric approach is used through a global F1 score. This metric captures the unweighted average F1 scores of the individual class predictions in the model. This is a concise measure which indicates whether or not the model is achieving a good balance of true positives/(true positives+false positives) and true positives/(true positives+false negatives) for each class without letting the majority class skew the average (if we used a weighted

F1 score it would consistently produce results that are extremely low or extremely high based on how well it was tuned for the majority class).

- Precision
    - This is used in a global sense for the overall model as well. This measure is calculated as the number of true positives divided by the sum of true positives and false positives for each class, across the entire model. This metrics indicates whether or not the model is able to distinguish each of the classes well enough that it generates a low number of false positives.
- Recall
    - This metric is also used for the overall model and is calculated as the ratio of true positives divided by the sum of true positives and false negatives. This type of error indicates whether the model is able to achieve a reliable balance across classes of correctly predicted labels relative to the total number of instances for that label.

Results

Resubstitution analysis revealed that several of the models were able to effectively locate and distinguish the minority classes from the majority in our training set (this is indicated by a strong overall accuracy in combination with high global F1 and precision/recall). Across the board with each individual model and the meta-learner (except for AdaBoost), resubstitution performance was improved upon when using the window-30 dataset that had been transformed with PCA, and the variation of that which also employed Tomek-links to pruned instances of the majority class which caused minority class instances to be hard to classify correctly (tables in figure 5). The best performance was achieved by the stacked voting meta-learner and tied with the resubstitution performance generated by the combination of bagging and boosting. Both these methods were able to get at least 1% improvement upon the next best method in every metric category. Consistent with research in stacking methods, the meta-learner in use appeared to have its strongest performance be bounded by the performance of the best classifier used as inputs (the bagging and boosting combination).

The next best model with resubstitution was the Rotation Forest which expanded on the functionality of a Random Forest, followed by a combination of Random Forest and a simple Decision Tree. The AdaBoost model and Naïve Bayes both generated fairly poor performance, achieving Global F1, Precision, and Recall values well below 50%. AdaBoost achieved its best results on the raw data set and raw data set which utilized Tomek-links to clean up difficult cases. Naïve Bayes on the other hand, was able to produce results with a higher Global F1 when utilizing the raw dataset that transformed the majority class using cluster centroids. Given the assumptions of independence and expectation of a relatively balanced target distribution, it is possible that the usage of cluster centroids was able to reduce multicollinearity in the dataset, and likely that the significant portion of instances removed helped induce stronger performance by allowing it to train on a more balanced dataset which still captured the severely imbalanced class relationship.

Resubstitution Accuracy

Model	Raw	Raw + Tomek Links	Raw + Cluster Centroids	Raw + Tomek Links + RUS	PCA on window-5 data	PCA on window-5 + Tomek Links	PCA on window-5 + Cluster Centroids	PCA on window-5 + Tomek Links + RUS	PCA on window-30 data	PCA on window-30 + Tomek Links	PCA on window-30 + Cluster Centroids	PCA on window-30 + Tomek Links + RUS
AdaBoost	79.2%	79.9%	73.7%	72.9%	71.3%	71.5%	64.3%	64.2%	73.4%	73.3%	64.4%	62.2%
Bagging+Boosting	99.8%	99.8%	99.6%	99.6%	99.4%	99.5%	99.1%	99.0%	99.9%	99.8%	99.6%	99.7%
Decision Tree	97.0%	96.9%	95.0%	94.9%	94.2%	94.3%	91.4%	91.1%	97.5%	97.5%	94.8%	95.0%
Naïve Bayes	71.0%	69.7%	63.6%	62.9%	70.0%	69.8%	58.8%	58.8%	71.6%	71.6%	57.6%	57.6%
Random Forest	97.2%	97.1%	95.0%	94.4%	93.6%	94.0%	90.4%	90.3%	98.1%	98.2%	95.8%	95.8%
Rotation Forest	99.0%	98.9%	97.9%	97.7%	97.0%	97.1%	95.2%	95.0%	99.1%	99.2%	98.2%	98.2%
Stacked Ensemble	99.8%	99.8%	99.7%	99.6%	99.4%	99.5%	99.1%	99.1%	99.9%	99.9%	99.6%	99.7%

Resubstitution Global F1

Model	Raw	Raw + Tomek Links	Raw + Cluster Centroids	Raw + Tomek Links + RUS	PCA on window-5 data	PCA on window-5 + Tomek Links	PCA on window-5 + Cluster Centroids	PCA on window-5 + Tomek Links + RUS	PCA on window-30 data	PCA on window-30 + Tomek Links	PCA on window-30 + Cluster Centroids	PCA on window-30 + Tomek Links + RUS
AdaBoost	38.1%	41.8%	52.6%	44.4%	17.4%	23.0%	24.2%	24.0%	29.2%	29.2%	26.0%	32.9%
Bagging+Boosting	99.5%	99.6%	99.4%	99.2%	98.9%	98.9%	98.6%	98.5%	99.7%	99.6%	99.2%	99.4%
Decision Tree	92.4%	92.7%	91.8%	90.6%	86.9%	87.2%	86.0%	84.5%	94.1%	94.2%	89.4%	90.4%
Naïve Bayes	17.0%	16.9%	21.5%	15.9%	27.6%	28.0%	30.8%	30.9%	34.7%	34.7%	35.1%	35.1%
Random Forest	93.5%	93.5%	92.3%	89.7%	85.4%	86.0%	83.3%	82.6%	95.7%	96.0%	91.2%	91.6%
Rotation Forest	97.3%	97.2%	96.3%	95.4%	92.7%	93.1%	91.6%	90.5%	98.1%	98.2%	95.2%	95.6%
Stacked Ensemble	99.5%	99.6%	99.5%	99.3%	98.9%	98.9%	98.7%	98.5%	99.7%	99.7%	99.2%	99.4%

Resubstitution Precision

Model	Raw	Raw + Tomek Links	Raw + Cluster Centroids	Raw + Tomek Links + RUS	PCA on window-5 data	PCA on window-5 + Tomek Links	PCA on window-5 + Cluster Centroids	PCA on window-5 + Tomek Links + RUS	PCA on window-30 data	PCA on window-30 + Tomek Links	PCA on window-30 + Cluster Centroids	PCA on window-30 + Tomek Links + RUS
AdaBoost	45.4%	57.5%	57.5%	52.6%	24.2%	24.3%	34.2%	33.7%	36.1%	36.1%	50.7%	56.2%
Bagging+Boosting	99.7%	99.6%	99.7%	99.7%	99.2%	99.2%	99.3%	99.3%	99.8%	99.7%	99.7%	99.8%
Decision Tree	93.5%	93.9%	92.6%	92.4%	87.4%	87.7%	87.5%	86.9%	94.3%	94.5%	91.7%	92.5%
Naïve Bayes	17.4%	16.5%	31.1%	17.8%	35.5%	36.3%	36.5%	36.7%	39.2%	39.3%	39.0%	39.2%
Random Forest	95.8%	95.7%	94.7%	94.9%	89.6%	90.1%	90.8%	90.3%	96.7%	96.9%	96.5%	96.2%
Rotation Forest	98.4%	98.4%	97.5%	98.2%	94.5%	94.9%	96.0%	95.5%	98.3%	98.5%	98.3%	98.4%
Stacked Ensemble	99.7%	99.6%	99.7%	99.7%	99.2%	99.2%	99.2%	99.1%	99.8%	99.7%	99.7%	99.8%

Resubstitution Recall

Model	Raw	Raw + Tomek Links	Raw + Cluster Centroids	Raw + Tomek Links + RUS	PCA on window-5 data	PCA on window-5 + Tomek Links	PCA on window-5 + Cluster Centroids	PCA on window-5 + Tomek Links + RUS	PCA on window-30 data	PCA on window-30 + Tomek Links	PCA on window-30 + Cluster Centroids	PCA on window-30 + Tomek Links + RUS
AdaBoost	38.1%	40.4%	50.0%	43.5%	20.3%	23.7%	25.6%	25.4%	28.4%	28.4%	25.6%	29.7%
Bagging+Boosting	99.4%	99.5%	99.2%	98.7%	98.6%	98.6%	98.1%	97.7%	99.6%	99.6%	98.7%	99.0%
Decision Tree	91.3%	91.6%	90.9%	89.3%	86.4%	86.7%	84.7%	82.8%	94.0%	93.9%	87.9%	89.1%
Naïve Bayes	20.2%	20.2%	23.8%	20.2%	27.9%	28.4%	34.6%	34.8%	36.0%	36.0%	38.8%	38.7%
Random Forest	91.5%	91.6%	90.1%	86.2%	82.1%	82.8%	78.4%	78.1%	94.8%	95.1%	88.3%	88.8%
Rotation Forest	96.3%	96.1%	95.1%	93.1%	91.1%	91.5%	88.3%	87.0%	97.8%	98.0%	93.1%	93.6%
Stacked Ensemble	99.4%	99.5%	99.3%	99.0%	98.6%	98.6%	98.3%	97.8%	99.6%	99.6%	98.8%	99.1%

Figure 5: Resubstitution performance metrics for each individual model and the meta-learning stack on each dataset. Color scale indicates the relative best performance within each metrics window (green is better performance, red is worse)

Performance for all of the models including the meta-learner decreased dramatically upon generalization to a new and separate experimental recording (figure 6). Accuracy remained high for several of the models, but the there are no instances where the Accuracy was high and the Global F1 score was higher than 26% (AdaBoost produced the highest Global F1 with 26.1%), which indicates that those instances of high accuracy correspond to situations where the models are assigning all labels as the majority class, Forward-NTD. For the most part, models which were trained on the datasets processed with either Cluster Centroids or Tomek-links and Random-Under-Sampling methods produced very poor results with generalization. This indicates that the data thrown out during training (Cluster Centroids and RUS remove more of the majority class than Tomek-links) was important for the model to be able to generalize because almost all of the new instances were predicated to be a minority class (hence the very low overall accuracy from missing the majority class labels). The *relative* exceptions to this were the Bagging+Boosting, Random Forest, and Stacked Meta-Learners which were able to still classify a portion of the data more effectively than the other models, but they still only achieved Global F1 scores below 30% which is far less than desirable.

Generalization Accuracy

Model	Raw	Raw + Tomek Links	Raw + Cluster Centroids	Raw + Tomek Links + RUS	PCA on window-5 data	PCA on window-5 + Tomek Links	PCA on window-5 + Cluster Centroids	PCA on window-5 + Tomek Links + RUS	PCA on window-30 data	PCA on window-30 + Tomek Links	PCA on window-30 + Cluster Centroids	PCA on window-30 + Tomek Links + RUS
AdaBoost	94.5%	95.3%	5.7%	1.6%	95.4%	95.4%	64.3%	62.5%	56.5%	56.5%	3.4%	34.3%
Bagging+Boosting	38.7%	40.6%	44.1%	63.0%	35.5%	40.9%	67.1%	89.9%	90.4%	88.1%	53.5%	27.0%
Decision Tree	64.1%	66.0%	9.2%	19.0%	37.8%	42.6%	7.5%	14.2%	81.9%	82.5%	15.8%	45.8%
Naïve Bayes	95.1%	95.1%	3.8%	3.5%	95.4%	95.4%	89.5%	92.4%	95.4%	95.4%	90.9%	90.1%
Random Forest	81.6%	81.0%	15.6%	72.6%	83.6%	93.0%	14.5%	19.9%	74.7%	78.6%	49.6%	13.3%
Rotation Forest	78.5%	69.1%	13.2%	9.6%	43.9%	44.4%	10.2%	11.2%	82.4%	81.9%	40.3%	42.8%
Stacked Ensemble	38.8%	40.6%	43.8%	68.6%	35.5%	40.9%	67.3%	89.9%	90.4%	88.1%	55.4%	27.8%

Generalization Global F1

Model	Raw	Raw + Tomek Links	Raw + Cluster Centroids	Raw + Tomek Links + RUS	PCA on window-5 data	PCA on window-5 + Tomek Links	PCA on window-5 + Cluster Centroids	PCA on window-5 + Tomek Links + RUS	PCA on window-30 data	PCA on window-30 + Tomek Links	PCA on window-30 + Cluster Centroids	PCA on window-30 + Tomek Links + RUS
AdaBoost	26.1%	24.4%	9.5%	3.9%	24.4%	24.4%	16.7%	16.5%	18.3%	18.3%	1.5%	13.9%
Bagging+Boosting	14.2%	14.0%	15.0%	19.9%	14.4%	16.1%	20.7%	24.1%	23.8%	19.2%	14.4%	8.9%
Decision Tree	21.8%	17.9%	7.3%	9.2%	14.3%	15.5%	3.8%	5.7%	19.3%	24.3%	6.3%	15.0%
Naïve Bayes	24.4%	24.4%	1.8%	1.7%	24.4%	24.4%	23.6%	24.0%	24.4%	24.4%	23.8%	23.7%
Random Forest	19.8%	22.2%	6.6%	20.1%	25.5%	19.7%	5.9%	7.6%	24.9%	22.9%	18.2%	5.4%
Rotation Forest	25.0%	20.3%	9.1%	4.9%	16.4%	16.0%	4.9%	5.3%	24.7%	24.6%	13.3%	13.0%
Stacked Ensemble	14.0%	14.0%	14.8%	21.3%	14.4%	16.1%	20.7%	24.1%	23.8%	19.2%	14.7%	9.1%

Generalization Precision

Model	Raw	Raw + Tomek Links	Raw + Cluster Centroids	Raw + Tomek Links + RUS	PCA on window-5 data	PCA on window-5 + Tomek Links	PCA on window-5 + Cluster Centroids	PCA on window-5 + Tomek Links + RUS	PCA on window-30 data	PCA on window-30 + Tomek Links	PCA on window-30 + Cluster Centroids	PCA on window-30 + Tomek Links + RUS
AdaBoost	26.6%	23.9%	25.4%	24.0%	23.9%	23.9%	19.6%	19.8%	23.9%	23.9%	20.7%	25.1%
Bagging+Boosting	21.0%	20.7%	21.5%	22.2%	50.1%	24.9%	24.1%	24.2%	23.9%	19.6%	19.4%	19.8%
Decision Tree	24.4%	20.3%	23.1%	21.4%	24.6%	24.6%	20.8%	20.1%	19.9%	25.0%	20.7%	20.7%
Naïve Bayes	23.9%	23.9%	24.8%	0.9%	23.9%	23.9%	23.8%	23.8%	23.9%	23.9%	23.8%	23.8%
Random Forest	21.1%	22.2%	20.1%	21.4%	25.9%	20.9%	20.2%	19.3%	26.4%	24.3%	24.8%	20.8%
Rotation Forest	26.5%	21.8%	24.0%	20.7%	25.3%	24.7%	20.7%	20.5%	25.3%	25.2%	20.3%	19.7%
Stacked Ensemble	20.8%	20.7%	21.4%	22.6%	50.1%	24.9%	24.1%	24.2%	23.9%	19.6%	19.4%	19.7%

Generalization Recall

Model	Raw	Raw + Tomek Links	Raw + Cluster Centroids	Raw + Tomek Links + RUS	PCA on window-5 data	PCA on window-5 + Tomek Links	PCA on window-5 + Cluster Centroids	PCA on window-5 + Tomek Links + RUS	PCA on window-30 data	PCA on window-30 + Tomek Links	PCA on window-30 + Cluster Centroids	PCA on window-30 + Tomek Links + RUS
AdaBoost	25.7%	25.0%	25.3%	24.4%	25.0%	25.0%	18.4%	23.3%	19.0%	19.0%	16.6%	34.8%
Bagging+Boosting	20.4%	17.1%	28.3%	31.4%	41.8%	19.2%	30.3%	24.0%	23.8%	18.9%	19.8%	25.6%
Decision Tree	25.0%	22.0%	26.2%	24.1%	35.0%	36.1%	22.6%	22.8%	20.1%	25.6%	21.2%	25.7%
Naïve Bayes	24.9%	24.9%	25.0%	24.9%	25.0%	25.0%	23.4%	24.2%	25.0%	25.0%	23.8%	23.6%
Random Forest	24.3%	27.7%	20.2%	25.0%	26.7%	19.7%	25.2%	30.4%	33.0%	22.6%	24.4%	18.5%
Rotation Forest	26.0%	24.3%	21.1%	17.4%	36.9%	36.6%	20.2%	20.8%	25.6%	26.0%	22.7%	13.4%
Stacked Ensemble	19.8%	17.1%	27.0%	31.7%	41.8%	19.2%	30.4%	24.1%	23.8%	18.9%	20.2%	25.8%

Figure 6: Generalization performance metrics for each individual model and the meta-learning stack on each dataset. Color scale indicates the relative best performance within each metrics window (green is better performance, red is worse)

Additional noteworthy points are that in terms of Global F1, the Tomek-link and Tomek-link+RUS variations of the window-30 dataset produced more consistent results in terms of their unweighted F1 average, and that the stacked ensemble used was not the best model anymore and bested by an individual model on nearly every variation of the training datasets. This is most likely due to the fact that most of the models generalized very poorly to the new data, which cause the voting mechanism in the meta-learner to frequently mimic the poor classification abilities across the skewed target distribution.

In terms of the balance of Precision and Recall in the models, there were two combinations which stuck out: Bagging+Boosting on the window-5 dataset with no additional target balancing efforts used, and the performance by the meta-learner on the same dataset. The reason the stacked version did so well is likely because the of the strong performance from the Bagging+Boosting classifier and it swaying the meta-voting in the correct direction, and for that reason the Bagging+Boosting model is of greater interest. This model had strong resubstitution performance, a *less* abysmal overall accuracy score and Global F1 balance than other models and generated nearly twice as strong of Precision and Recall for the new video. The confusion matrix for the individual Bagging+Boosting model upon generalizing to an unseen experimental recording can be found in Figure 7. Part of the reason for the poor performance across all models with this testing set is that only 4 of the 5 movement states are even observed in the new data. However, the Bagging+Boosting model can make consistent predictions with that and doesn't identify any states as Reverse-Long, which is important to note that it can identify those well. The Bagging+Boosting model confuses the majority of Forward-NTD movements as either Forward-Sharp or Reverse-Short which causes a significant number of incorrect predictions It only makes two predictions that an instance is a Stop and get them both correct, but unfortunately misses the remaining 1473 of them. The reason that the model appears to drive strong Precision and Recall is because of the relatively small number of false positives on Forward-NTD and no false positive on Stop.

		Predicted			
	Label	**Forward-NTD**	**Forward-Sharp**	**Reverse-Short**	**Stop**
Actual	**Forward-NTD**	14685	8279	17367	0
	Forward-Sharp	0	62	63	0
	Reverse-Short	0	62	265	0
	Stop	282	582	609	2

Figure 7: Confusion matrix for Bagging+Boosting in generalizing to new video

The reason for very poor generalization performance is somewhat illuminated when investigating the transformed feature space of the test set by plotting the first two principal components which capture most of the variance in the data and compare it to those from the training dataset. This is shown in figure 8. The shape of the majority of the variance, the amount of noise generated by the majority class, and the locations of the minority classes in the transformed feature space all shift from training to testing. In addition to this, there is the consistent problem of overlapping target-class clusters that make it difficult to distinguish the minorities from both each other, and the majority class.

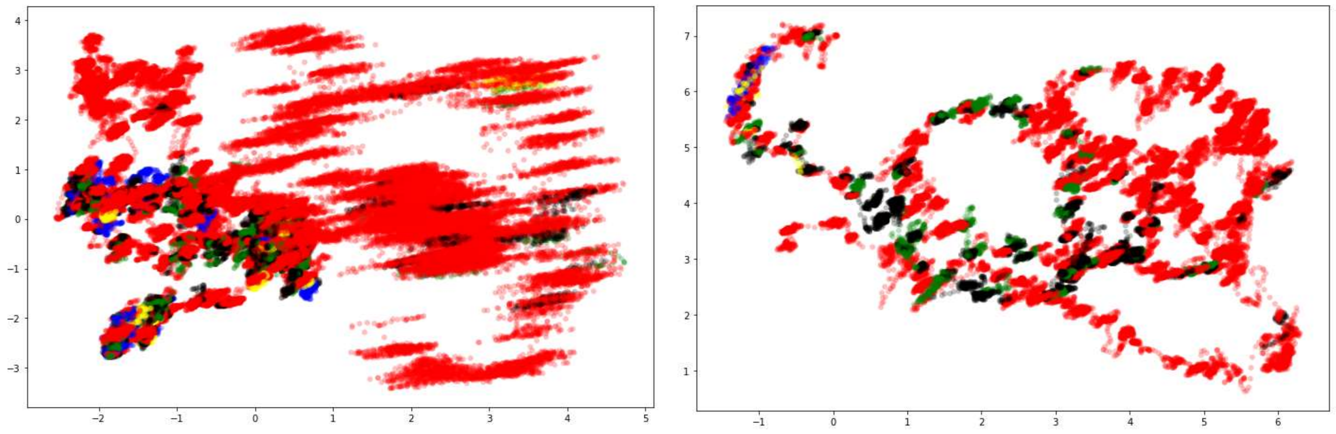

Figure 8: PC1 and PC2 of training set (left) and PC1 and PC2 of testing set (right). The changing levels of noise and location of minority classes shifts in the new video despite being the same mutant and environment

Conclusions and Future Work

The individual models explored in this paper as well as the meta-learner which generated predictions based on the predictions of all the individual learners were able to learn the training data effectively by using relatively flexible tuning parameters. The datasets which included temporal features and various combinations of target-balancing techniques to illuminate position and movements before and after a given instance helped improve the performance of the classifier when learning the training set but did not translate into a significant boost to performance upon generalization. Investigation into the distribution of the data and its variance in both training and testing along with knowledge of the severe class imbalance indicates that while these methods may be suitable for effective learning in other scenarios, that they are not robust to this dataset which is tremendously skewed and noisy. The meta-learning model tested was not effective at balancing out the weaknesses of individual learners and provided no incremental value on testing data, although additional tuning to the voting weights of these models has the potential to improve this. An interesting highlight amongst a series of lowlights is the strong resubstitution performance generated by a bagging and boosting combo, despite it being a heuristic to a more complex algorithm (RotBoost). The relatively strong generalization (but overall still poor) performance may be less impacted by the algorithm itself and more due to the nature of the extremely noisey data set since other algorithms generated even less correct labels for minority classes. This is something that is worth continuing to explore more and tune appropriately for other problems.

Future work in the area of building a state-labeler for C.elegans experimental videos includes the application of a proper RotBoost algorithm to improve upon a Bagging+Boosting combination, in addition to modifying RotBoost to utilize a kernel function since the data is clearly not linearly transformable and minority class instances tend to cluster together while the majority class is noisily dispersed through the feature space. Additional methods which we seek to explore are use of sequence classifiers such as the Hidden Markov Model and ensemble of Hidden Markov Models (training multiple HMMs of varying orders on sequences of data which correspond to the different states and using a meta-learner to make label predictions by using the sum of the transition probabilities to different states).

References

1. Campos, R., Canuto, S., Salles, T., Sá, C. C., & Gonçalves, M. A. (2017). Stacking Bagged and Boosted Forests for Effective Automated Classification. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 17*. doi:10.1145/3077136.3080815
2. Rodriguez, J., Kuncheva, L., & Alonso, C. (2006). Rotation Forest: A New Classifier Ensemble Method. *IEEE Transactions on Pattern Analysis and Machine Intelligence,28*(10), 1619-1630. doi:10.1109/tpami.2006.211
3. T, E., & M, A. (2016). Classification of Imbalance Data using Tomek Link(T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method. *Journal of Informatics and Data Mining,1*(2). doi:10.21767/2472-1956.100011
4. Tsoumakas, G., Katakis, I., & Vlahavas, I. (2004). Effective Voting of Heterogeneous Classifiers. *Machine Learning: ECML 2004 Lecture Notes in Computer Science,*465-476. doi:10.1007/978-3-540-30115-8_43
5. Vijayan, V. V., & Anjali, C. (2015). Computerized information system using stacked generalization for diagnosis of diabetes mellitus. *2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*. doi:10.1109/raics.2015.7488409
6. Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (n.d.). Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems*.
7. Zaman, M. F., & Hirose, H. (2011). Classification Performance of Bagging and Boosting Type Ensemble Methods with Small Training Sets. *New Generation Computing,29*(3), 277-292. doi:10.1007/s00354-011-0303-0
8. Zhang, C., & Zhang, J. (2008). RotBoost: A technique for combining Rotation Forest and AdaBoost. *Pattern Recognition Letters,29*(10), 1524-1536. doi:10.1016/j.patrec.2008.03.006