Exploiting Ordinal Structure in Data for Identifying Driver Synonymous Mutations

Elvin Lo
Stat 195 Final Project, Spring 2025

1 Introduction

Single-nucleotide variants (SNVs) are mutations in the genome where a single nucleotide is altered. Synonymous SNVs (sSNVs) are those SNVs that occur in protein-coding regions, but do not change the amino acid sequence. Despite that sSNVs do not impact protein structure, growing evidence suggests that sSNVs can still influence gene expression, mRNA stability, splicing, or translation efficiency, and thus may play a role in diseases like cancer [3]. Distinguishing these salient sSNVs, referred to as *driver sSNVs*, remains a significant challenge due to their subtle effects. Distinguishing driver sSNVs from passenger sSNVs *in vitro* from large-scale sSNVs is expensive, since the process is labor-intensive and time-consuming. Hence, as high-throughput sequencing continues to generate large volumes of mutational data, it is increasingly desirable to develop machine learning (ML) methods that can identify candidate driver sSNVs based on features extracted from genomic, structural, and evolutionary data. Such tools could accelerate biological discovery and support precision oncology by highlighting potentially pathogenic synonymous mutations that would otherwise be overlooked.

2 Related Work

So, we are interested in developing ML methods to identify driver sSNVs. Of course, the notion of which sSNVs are "drivers" is not well-defined: given a dataset of tumor samples and all the sSNVs that are present, it is not obvious how to formulate the problem of predicting which of those sSNVs are drivers. Hence, past works [1, 2, 5] employ the heuristic of recurrence level, or the number of times a mutation has been observed in different cases (tumor samples). The implicit assumption here is that sSNVs common amongst known tumor samples are more likely to drivers. It is intuitive that such a correlation should exist, but of course there may exist highly recurrent passenger sSNVs or rare driver sSNVs. Nonetheless, the recurrence level heuristic lets us make better use of our available data and still provides some meaningful signal.

Specifically, past works [1, 5, 2] use the recurrence level heuristic to formulate the identification of driver sSNVs as a binary classification problem. In particular, they threshold sSNVs with recurrence level $r \geq k$ as positive samples ($k = 7$) and those with $r = 1$ are negative samples, and then apply various tree-based classification algorithms. (The differences between the papers lies largely in the choice of features they used.) Other works on similar driver sSNV-related problems exist [6], including some like [4] which explore deep learning-based methods, but tabular features (hand-crafted with domain knowledge) and tree-based algorithms remain the dominant approach especially for finding driver sSNVs for cancer (which the authors posit to be particularly nuanced even compared to other diseases).

However, the thresholding approach used in past work presents two issues:

1. First, this framing does not leverage any of the samples with recurrence levels $1 < r < k$, thereby wasting much of our available data. In particular, the available sSNV data after using such a threshold consists of 2000–3000 samples, of which few samples are "positive" samples.

2. Second, the choice of $k = 7$ is arbitrary, e.g., [1] do not provide reasoning for this choice other than that their model worked well with this choice (after trying many other values of k), and that very few samples had $k > 7$ so that any larger k would be impractical for the dataset size.

In this project, we develop a statistically principled ML method that overcomes these shortcomings

3 Proposal

Our proposed method resolves these issues by reformulating the problem. We label sSNVs by their explicit recurrence level, so that we have data classes for each r. With this modification, we no longer need to discard the samples with recurrence levels $1 < r < k$, so we have richer data to train our model on. So, our problem has dataset $D = \{(X_i, Y_i)\}_{1 \leq i \leq n}$, with $X_i \in \mathbb{R}^p$ and $Y_i \in \{1, 2, \ldots, K\}$ with $K \approx 10$ (but with fewer samples for larger class labels).

Now, our goal roughly is to know which samples have large class labels (highly recurrent), without explicitly specifying a threshold. We note that under this problem formulation, what we are really interested in is not really the classification accuracy (i.e., how many of the Y_j's in the test set did our model correctly predict), but we are interested in some sense of how far our model's predictions tended to be from the true recurrence level. Such a metric is meaningful because of the ordinal structure in recurrence levels and also because our dataset's recurrence level values themselves are estimates (i.e., they estimate how recurrent an sSNV is by just examining a known database, which is in a sense just a small sample of the population). Our choice of metric for this report will be the mean absolute error (MAE), which is the mean over the test samples of the absolute difference between the prediction (fitted response) and label (true response Y_j).

So, to accomplish our goal, we modify the objective in tree-based classification algorithms to respect the ordinal structure in the recurrence level (i.e., samples with r as 6 and 7 should be closer than samples with r as 1 and 7).

3.1 Decision Trees with Misclassification Costs

To define our weighted decision tree, we first define the misclassification cost $W(i,j) = |i-j|$, intuitively the "cost" of misclassifying a sample with class i as class j. Now, we modify the standard decision tree algorithm by using W to weight the objective as follows.

- At a particular node of the tree with data $R \subseteq D$, we denote the relative frequencies of each class by $p_k(R) = \frac{1}{|R|}|\{(X,Y) \in R : Y = k\}|$ and then define the weighted Gini

$$I_W(R) = \sum_{i=1}^{K}\sum_{j=1}^{K} p_i(R)p_j(R)W(i,j) = \sum_{i \neq j} p_i(R)p_j(R)|i-j|.$$

  This is the natural weighted generalization of the Gini coefficient, which is a common purity measure for decision trees.

- Now, we build the tree via the typical splitting criterion, i.e., we recursively split the data by selecting the feature and threshold at each node that minimize the weighted average of the weighted Gini impurity of the resulting child nodes.

- We continue splitting until a stopping condition is met (e.g., minimum node size or maximum depth), and assign each final region L to the best class, i.e.,

$$\hat{y}(L) = \operatorname*{argmin}_{k \in \{1,\ldots,K\}} \sum_{j=1}^{K} p_j(L)W(k,j).$$

Of course, in some cases the vanilla decision tree method can also perform well on ordinal data, e.g., consider the trivial example that each feature is roughly monotone in the class label. However, in more complicated examples, enforcing ordinal structure like our weighted decision tree offers benefits. For an extreme example, consider an example where a particular feature is "bimodal" in the class label in the sense that the feature is very similar for both some very small classes and large classes, but very different in medium-valued classes (e.g., a feature is large for samples with $k < 3$ and $k > 7$, but small for samples with $3 \leq k \leq 7$). In this case, it is still possible that the vanilla decision tree would choose to split on this feature if doing so can significantly improve the unweighted Gini, but the weighting function in the weighted decision tree will discourage such a split. (Of course, this does not directly imply that the weighted decision tree will outperform the vanilla decision tree, but this example is meant to provide some intuition for the kinds of behaviors that W might introduce into the decision tree.)

3.2 Extending Misclassification Costs to Other Tree-Based Methods

We may extend our design of the weighted decision tree (using the absolute difference weighting W) to other tree-based methods in a natural way. In particular, we may extend the weighted decision tree to random forests exactly as with vanilla random forests: we train many decision trees, each on a different bootstrap

sample, and at each split inside a tree we also randomly restrict the choice of features considered. Then, once each tree is grown fully, our final prediction is the median vote across all trees. The merit of the random forest method remains, in that it decorrelates the trees and greatly reduces variance compared to a single tree. The weight function W may be introduced to other classification methods also, e.g., we could define a cost-sensitive loss if we were to implement XGBoost.

4 Experiments

The original dataset was unfortunately not accessible, and finding similar alternative datasets with tabular data in related domains was challenging. Hence, we provide simulations to demonstrate the proposed misclassification cost-weighted RF, benchmarked against a vanilla RF model on the same multi-class classification problem. While there is not a completely natural way to benchmark the cost-weighted RF against the method of Bi et al. [1] since the problem formulations are different (multi-class versus binary), we provide some results as described in the Appendix.

Our simulated dataset mirrors key traits of driver-mutation recurrence. The class labels $k \in \{1, \ldots, 10\}$ are drawn with an exponentially decaying prior, as highly recurrent mutations are of course more rare. We sample $N = 6000$ data points with feature dimensionality $p = 25$ and split train-test 70-30. The twenty-five features are split into three (correlated) groups: (i) ten Gaussian measurements whose means grow linearly with k plus a shared "severity" latent factor, capturing the intuitive monotone signal that more severe mutations exhibit stronger molecular signatures; (ii) eight bimodal Gaussian-mixture features whose mixing weights oscillate with k, so distant classes (e.g., 1 and 10) occasionally share a mode, forcing learners to reckon with ordinal cost rather than naïve purity; and (iii) seven heavy-tailed Cauchy variables whose locations drift with k but whose outliers add realistic measurement noise. Two latent factors inject intra-group correlation to emulate pathway-level biological dependencies. While the marginals and heavy tails exaggerate some distributional aspects relative to real single-nucleotide variants-and ignore mutation-context covariates such as trinucleotide sequence-the hierarchy, imbalance, and structured overlap jointly recreate key challenges of ranking driver versus passenger mutations, thereby providing a meaningful test-bed for our methods. To demonstrate all tree algorithms, we use 100 decision trees per random forest and use the square root heuristic for the max number of features per tree (so the max features is set to 5).

Our results in Table 2 report the MAE and classification accuracy of the proposed weighted RF and the vanilla RF, including standard errors in the MAE. We see that the weighted RF is markedly better in MAE, as desired, though the vanilla RF still has better classification accuracy alone. Hence, the specified weighting function W succeeded in imposing the ordinal structure in this experiment. More elaborate results such as the confusion matrix are in the appendix.

Method	MAE	Classification Accuracy
Weighted RF	1.0911 ± 0.0240	0.313
Vanilla RF	1.2417 ± 0.0298	0.332

Table 1: Experimental results comparing the proposed weighted RF with the vanilla RF on the multi-class classification problem.

5 Future Directions

Future work can explore methodological improvements to the cost-aware RF and expand the application scope. Methodological improvements include modifying the current linear weighting function $W(i, j) = c|i - j|$, e.g., we could take a more data-driven approach to choosing W to better align with real clinical costs. Of course, the cost-aware approach to RFs may be extended to more methods like gradient-boosted trees. Theoretical analyses may also be of interest.

On the application side, it may be productive to apply the cost-aware method to real datasets in various problems with ordinal structure, e.g., disease severity or tumor grade prediction problems. Of course, it is important to consider the nature of the data features; while our tree-based methods work well on tabular data, working with image data or other modalities would require adapting the misclassification cost-based weighting to the appropriate loss functions.

References

[1] Bi, C., Shi, Y., Xia, J., Liang, Z., Wu, Z., Xu, K., and Cheng, N. (2025). Ensemble learning-based predictor for driver synonymous mutation with sequence representation. *PLOS Computational Biology*, 21(1):e1012744.

[2] Cheng, N., Bi, C., Shi, Y., Liu, M., Cao, A., Ren, M., Xia, J., and Liang, Z. (2023). Effect predictor of driver synonymous mutations based on multi-feature fusion and iterative feature representation learning. *IEEE Journal of Biomedical and Health Informatics*, 28(2):1144–1151.

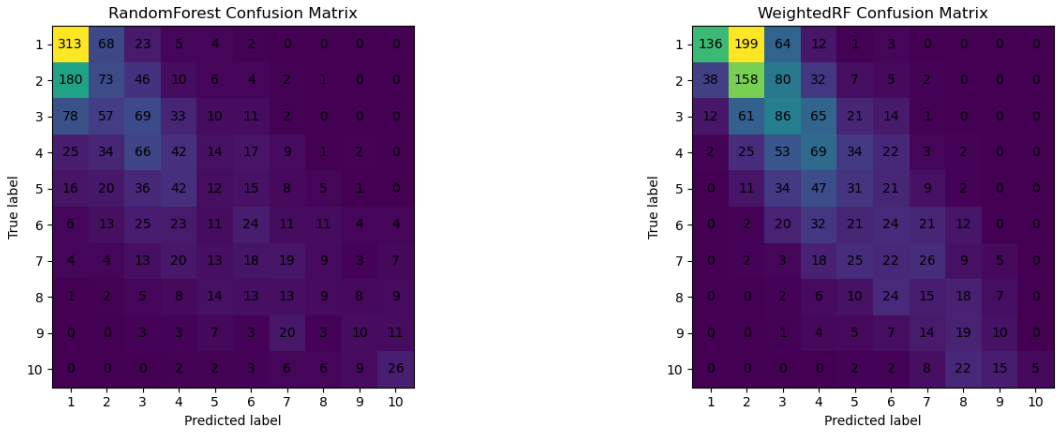
[3] Kaissarian, N. M., Meyer, D., and Kimchi-Sarfaty, C. (2022). Synonymous variants: necessary nuance in our understanding of cancer drivers and treatment outcomes. *Journal Of The National Cancer Institute*, 114(8):1072–1094.

[4] Quang, D., Chen, Y., and Xie, X. (2014). Dann: a deep learning approach for annotating the pathogenicity of genetic variants. *Bioinformatics*, 31(5):761–763.

[5] Rogers, M. F., Gaunt, T. R., and Campbell, C. (2020). Cscape-somatic: distinguishing driver and passenger point mutations in the cancer genome. *Bioinformatics*, 36(12):3637–3644.

[6] Rogers, M. F., Shihab, H. A., Mort, M., Cooper, D. N., Gaunt, T. R., and Campbell, C. (2018). Fathmm-xf: accurate prediction of pathogenic point mutations via extended features. *Bioinformatics*, 34(3):511–513.

A Confusion Matrices from Multi-Class Classification Experiments



(a) Random Forest

(b) Weighted Random Forest

Figure 1: Confusion matrices for the misclassification cost-weighted and vanilla random forest classifiers.

B Benchmarking Results Under the Binary Formulation

To benchmark the proposed multi-class classification methods against the binary random forest (RF) method considered by Bi et al. [1], we use the same original 70-30 split and then discard all samples with recurrence level $1 < r < 7$. We then run the binary classification random forest alongside the weighted and vanilla multi-class RFs, where we binarize the multi-class predictions by simply thresholding 0 if the prediction is below 7 and 1 otherwise. Of course, this thresholding is not really fair, since, for example, the model may predicted a 6 instead of a 7 for a "positive" sample (and such an error should be weighted much less than predicting a much smaller recurrence level on a positive sample).

Method	Classification Accuracy
Binary classification RF	0.9473
Weighted RF with thresholding	0.8877
Vanilla RF	0.8571

Table 2: Experimental results comparing the methods under the binary classification problem formulation using thresholding.

We see that baseline binary RF from Bi et al. [1] performs the best, but the weighted RF still outperforms the vanilla RF and of course the classification accuracies of both multi-class methods could be arbitrarily increased by thresholding a smaller value like 6. We do not conduct a full ablation study due to computational and time constraints, unfortunately.