Lecture 4: Resampling Methods

Resampling methods are an indispensable tool in modern statistics. They involve repeatedly drawing samples from a training set and refitting a model of interest on each sample in order to obtain additional information about the fitted model. For example, to estimate the variability of a linear regression fit, we can repeatedly draw different samples from the training data, fit a linear regression to each new sample, and then examine the extent to which the resulting fits differ. Such an approach may allow us to obtain information that would not be available from fitting the model only once using the original training sample.

Resampling approaches can be computationally expensive, because they involve fitting the same statistical method multiple times using different subsets of the training data. However, due to recent advances in computing power, the computational requirements of resampling methods generally are not prohibitive. In this chapter, we discuss two of the most commonly used resampling methods, cross-validation and the bootstrap. Both methods are important tools in the practical application of many statistical learning procedures. For example, cross-validation can be used to estimate the test error associated with a given statistical learning method in order to evaluate its performance, or to select the appropriate level of flexibility. The process of evaluating a model's performance is known as model assessment, whereas the process of selecting the proper level of flexibility for a model is known as model selection. The bootstrap is used in several contexts, most commonly to provide a measure of accuracy of a parameter estimate or of a given statistical learning method.

Cross-Validation

In linear models, we discuss the distinction between the test error rate and the training error rate. The test error is the average error that results from using a statistical learning method to predict the response on a new observation — that is, a measurement that was not used in training the method. Given a data set, the use of a particular statistical learning method is warranted if it results in a low test error. The test error can be easily calculated if a designated test set is available. Unfortunately, this is usually not the case. In contrast, the training error can be easily calculated by applying the statistical learning method to the observations used in its training. But as we saw before, the training error rate often is quite different from the test error rate, and in particular the former can dramatically underestimate the latter.

In the absence of a very large, designated test set that can be used to directly estimate the test error rate, a number of techniques can be used to estimate this quantity using the available training data. Some methods make mathematical adjustment to the training error rate in order to estimate the test error rate. In this section, we instead consider a class of methods that estimate the test error rate by holding out a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations. First, for simplicity we assume that we are interested in performing regression with a quantitative response. Next, we consider the case of classification with a qualitative response. As we will see, the key concepts remain the same regardless of whether the response is quantitative or qualitative.

The Validation Set Approach

Suppose that we would like to estimate the test error associated with fitting a particular statistical learning method on a set of observations. The validation set approach, displayed in Figure 1, is a very simple strategy for this task. It involves randomly dividing the available set of observations into two parts, a training set and a validation set or hold-out set. The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set. The resulting validation set error rate—typically assessed using MSE in the case of a quantitative response—provides an estimate of the test error rate.

FIGURE 1. A schematic display of the validation set approach. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.

Using MSE as a measure of validation set error, are shown in the left-hand panel of Figure 2. The validation set MSE for the quadratic fit is considerably smaller than for the linear fit. However, the validation set MSE for the cubic fit is actually slightly larger than for the quadratic fit. This implies that including a cubic term in the regression does not lead to better prediction than simply using a quadratic term.

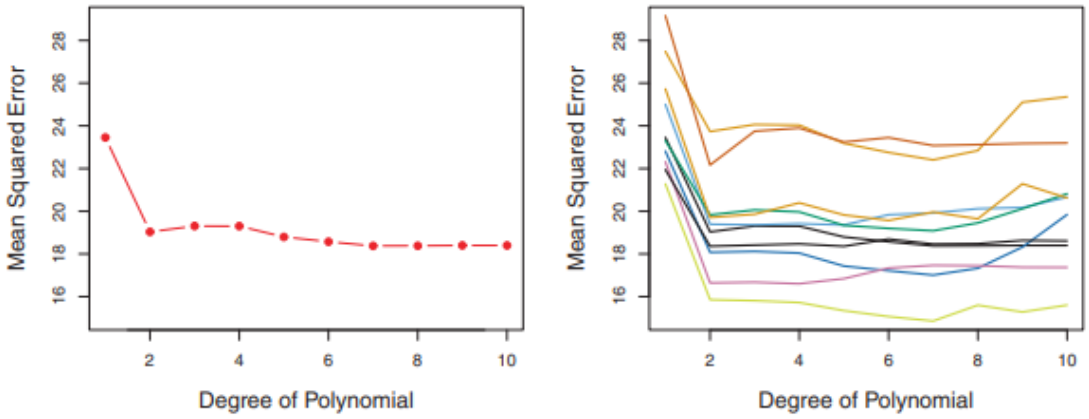

FIGURE 2. The validation set approach was used on the Auto data set in order to estimate the test error that results from predicting mpg using polynomial functions of horsepower. Left: Validation error estimates for a single split into training and validation data sets. Right: The validation method was repeated ten times, each time using a different random split of the observations into a training

set and a validation set. This illustrates the variability in the estimated test MSE that results from this approach.

Recall that in order to create the left-hand panel of Figure 2, we randomly divided the data set into two parts, a training set and a validation set. If we repeat the process of randomly splitting the sample set into two parts, we will get a somewhat different estimate for the test MSE. As an illustration, the right-hand panel of Figure 2 displays ten different vali- dation set MSE curves from the Auto data set, produced using ten different random splits of the observations into training and validation sets. All ten curves indicate that the model with a quadratic term has a dramatically smaller validation set MSE than the model with only a linear term. Furthermore, all ten curves indicate that there is not much benefit in including cubic or higher-order polynomial terms in the model. But it is worth noting that each of the ten curves results in a different test MSE estimate for each of the ten regression models considered. And there is no consensus among the curves as to which model results in the smallest validation set MSE. Based on the variability among these curves, all that we can conclude with any confidence is that the linear fit is not adequate for this data.

1. As is shown in the right-hand panel of Figure 2, the validation estimate of the test error rate can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.

2. In the validation approach, only a subset of the observations—those that are included in the training set rather than in the validation set—are used to fit the model. Since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation set error rate may tend to overestimate the test error rate for the model fit on the entire data set.

In the coming subsections, we will present cross-validation, a refinement of the validation set approach that addresses these two issues.

Leave-One-Out Cross-Validation

Leave-one-out cross-validation (LOOCV) is closely related to the validation set approach, but it attempts to address that method's drawbacks.

Like the validation set approach, LOOCV involves splitting the set of observations into two parts. However, instead of creating two subsets of comparable size, a single observation (x_1, y_1) is used for the validation set, and the remaining observations $\{(x_2, y_2), \dots, (x_n, y_n)\}$ make up the training set. The statistical learning method is fit on the $n - 1$ training observations, and a prediction $\widehat{y_1}$ is made for the excluded observation, using its value x_1. Since (x_1, y_1) was not used in the fitting process, $MSE_1 = (y_1 - \widehat{y_1})^2$ provides an approximately unbiased estimate for the test error. But even though MSE1 is unbiased for the test error, it is a poor estimate because it is highly variable, since it is based upon a single observation (x_1, y_1).

We can repeat the procedure by selecting (x_2, y_2) for the validation data, training the statistical learning procedure on the $n - 1$ observations $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, and computing

$MSE_2 = (y_2 - \widehat{y_2})^2$. Repeating this approach n times produces n squared errors, $MSE_1, \ldots, MSE_n$. The LOOCV estimate for the test MSE is the average of these n test error estimates:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1} MSE_i \tag{1}$$

A schematic of the LOOCV approach is illustrated in Figure 3.

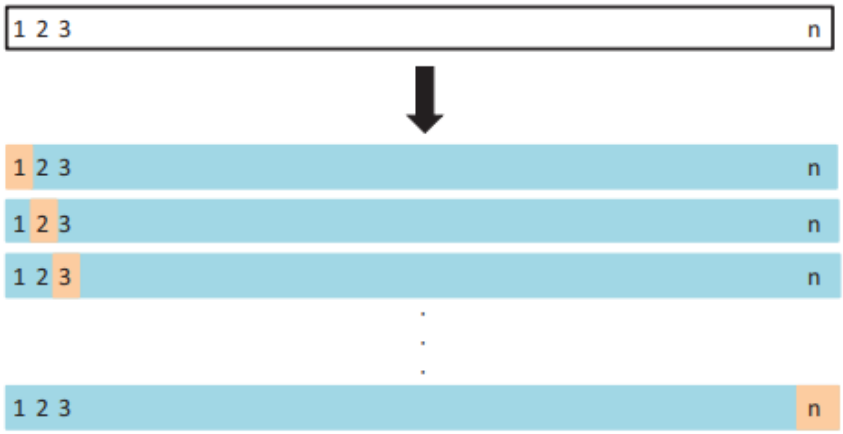

FIGURE 3. A schematic display of LOOCV. A set of n data points is repeatedly split into a training set (shown in blue) containing all but one observation, and a validation set that contains only that observation (shown in beige). The test error is then estimated by averaging the n resulting MSE's. The first training set contains all but observation 1, the second training set contains all but observation 2, and so forth.

LOOCV has a couple of major advantages over the validation set approach. First, it has far less bias. In LOOCV, we repeatedly fit the statistical learning method using training sets that contain $n - 1$ observations, almost as many as are in the entire data set. This is in contrast to the validation set approach, in which the training set is typically around half the size of the original data set. Consequently, the LOOCV approach tends not to overestimate the test error rate as much as the validation set approach does. Second, in contrast to the validation approach which will yield different results when applied repeatedly due to randomness in the training/validation set splits, performing LOOCV multiple times will always yield the same results: there is no randomness in the training/validation set splits.

We used LOOCV on the Auto data set in order to obtain an estimate of the test set MSE that results from fitting a linear regression model to predict mpg using polynomial functions of horsepower. The results are shown in the left-hand panel of Figure 4.

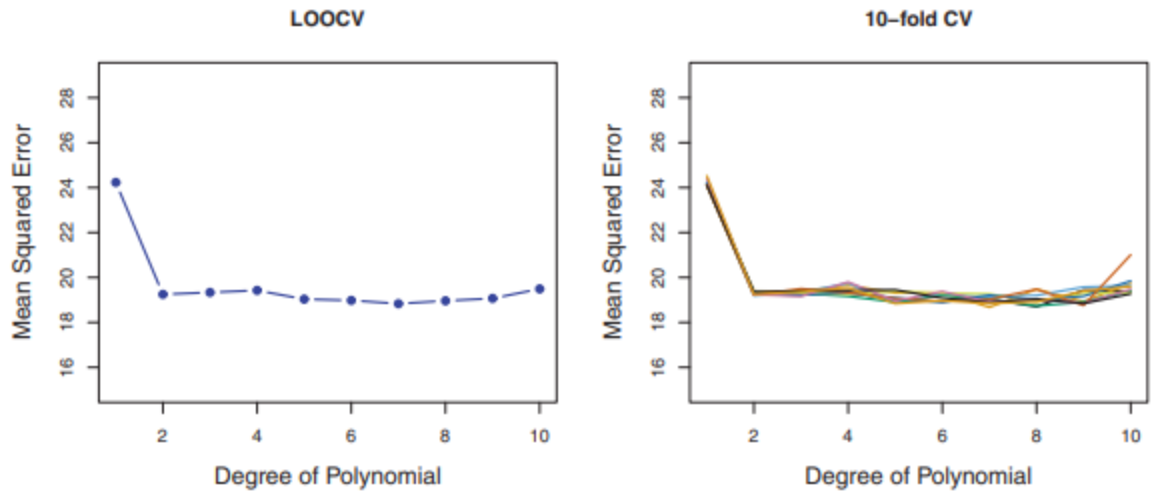

FIGURE 4. Cross-validation was used on the Auto data set in order to estimate the test error that results from predicting mpg using polynomial functions of horsepower. Left: The LOOCV error curve. Right: 10-fold CV was run nine separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves.

LOOCV has the potential to be expensive to implement, since the model has to be fit n times. This can be very time consuming if n is large, and if each individual model is slow to fit. With least squares linear or polynomial regression, an amazing shortcut makes the cost of LOOCV the same as that of a single model fit! The following formula holds:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2 \tag{2}$$

where $\hat{y}_i$ is the ith fitted value from the original least squares fit, and h_i is the leverage. This is like the ordinary MSE, except the ith residual is divided by $1 - h_i$. The leverage lies between 1/n and 1, and reflects the amount that an observation influences its own fit. Hence the residuals for high-leverage points are inflated in this formula by exactly the right amount for this equality to hold.

LOOCV is a very general method, and can be used with any kind of predictive modeling.

k-Fold Cross-Validation

An alternative to LOOCV is k-fold CV. This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $k - 1$ folds. The mean squared error, MSE_1, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error, $MSE_1, MSE_2, \ldots, MSE_k$. The k-fold CV estimate is computed by averaging these values,

$$CV_{(n)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{3}$$

Figure 5 illustrates the k-fold CV approach.

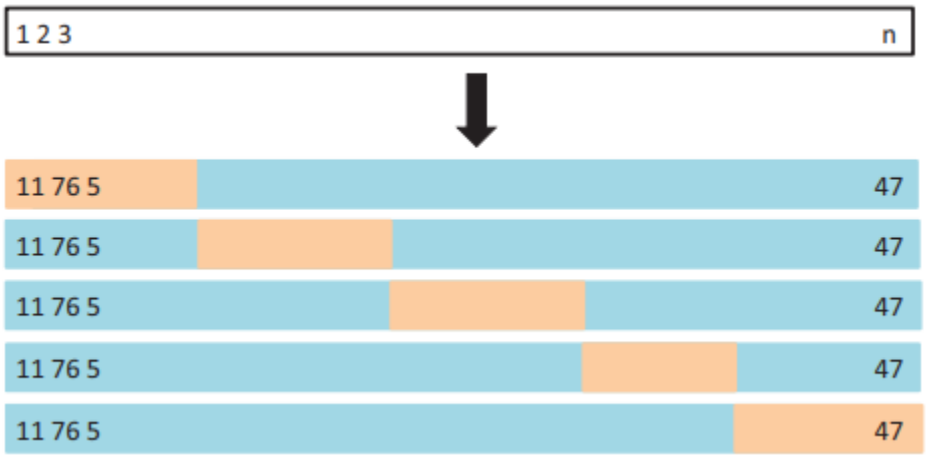

FIGURE 5. A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.

It is not hard to see that LOOCV is a special case of k-fold CV in which k is set to equal n. In practice, one typically performs k-fold CV using $k = 5$ or $k = 10$. What is the advantage of using $k = 5$ or $k = 10$ rather than $k = n$? The most obvious advantage is computational. LOOCV requires fitting the statistical learning method n times. This has the potential to be computationally expensive (except for linear models fit by least squares, in which case formula (2) can be used). But cross-validation is a very general approach that can be applied to almost any statistical learning method. Some statistical learning methods have computationally intensive fitting procedures, and so performing LOOCV may pose computational problems, especially if n is extremely large. In contrast, performing 10-fold CV requires fitting the learning procedure only ten times, which may be much more feasible. As we see later sections, there also can be other non-computational advantages to performing 5-fold or 10-fold CV, which involve the bias-variance trade-off.

The right-hand panel of Figure 4 displays nine different 10-fold CV estimates for the Auto data set, each resulting from a different random split of the observations into ten folds. As we can see from the figure, there is some variability in the CV estimates as a result of the variability in how the observations are divided into ten folds. But this variability is typically much lower than the variability in the test error estimates that results from the validation set approach (right-hand panel of Figure 2).

When we perform cross-validation, our goal might be to determine how well a given statistical learning procedure can be expected to perform on independent data; in this case, the actual estimate

of the test MSE is of interest. But at other times we are interested only in the location of the minimum point in the estimated test MSE curve. This is because we might be performing cross-validation on a number of statistical learning methods, or on a single method using different levels of flexibility, in order to identify the method that results in the lowest test error. For this purpose, the location of the minimum point in the estimated test MSE curve is important, but the actual value of the estimated test MSE is not.

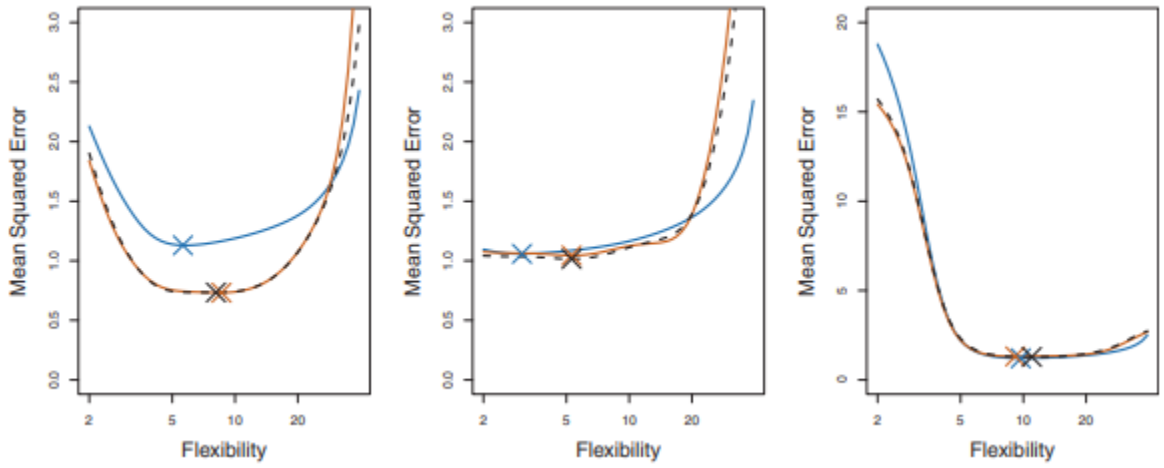

FIGURE 6. True and estimated test MSE for the simulated data sets in Lecture 1 Figures 9 (left), 10 (center), and 11 (right). The true test MSE is shown in blue, the LOOCV estimate is shown as a black dashed line, and the 10-fold CV estimate is shown in orange. The crosses indicate the minimum of each of the MSE curves.

Bias-Variance Trade-Off for k-Fold Cross-Validation

We mentioned that k-fold CV with $k < n$ has a computational advantage to LOOCV. But putting computational issues aside, a less obvious but potentially more important advantage of k-fold CV is that it often gives more accurate estimates of the test error rate than does LOOCV. This has to do with a bias-variance trade-off.

It was mentioned that the validation set approach can lead to overestimates of the test error rate, since in this approach the training set used to fit the statistical learning method contains only half the observations of the entire data set. Using this logic, it is not hard to see that LOOCV will give approximately unbiased estimates of the test error, since each training set contains $n - 1$ observations, which is almost as many as the number of observations in the full data set. And performing k-fold CV for, say, $k = 5$ *or* $k = 10$ will lead to an intermediate level of bias, since each training set contains $(k - 1)n/k$ observations—fewer than in the LOOCV approach, but substantially more than in the validation set approach. Therefore, from the perspective of bias reduction, it is clear that LOOCV is to be preferred to k-fold CV.

However, we know that bias is not the only source for concern in an estimating procedure; we must also consider the procedure's variance. It turns out that LOOCV has higher variance than does k-fold CV with $k < n$. Why is this the case? When we perform LOOCV, we are in effect

averaging the outputs of n fitted models, each of which is trained on an almost identical set of observations; therefore, these outputs are highly (positively) correlated with each other. In contrast, when we perform k-fold CV with $k < n$, we are averaging the outputs of k fitted models that are somewhat less correlated with each other, since the overlap between the training sets in each model is smaller. Since the mean of many highly correlated quantities has higher variance than does the mean of many quantities that are not as highly correlated, the test error estimate resulting from LOOCV tends to have higher variance than does the test error estimate resulting from k-fold CV.

To summarize, there is a bias-variance trade-off associated with the choice of k in k-fold cross-validation. Typically, given these considerations, one performs k-fold cross-validation using $k = 5\ or\ k = 10$, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.

Cross-Validation on Classification Problems

In this chapter so far, we have illustrated the use of cross-validation in the regression setting where the outcome Y is quantitative, and so have used MSE to quantify test error. But cross-validation can also be a very useful approach in the classification setting when Y is qualitative. In this setting, cross-validation works just as described earlier in this chapter, except that rather than using MSE to quantify test error, we instead use the number of misclassified observations. For instance, in the classification setting, the LOOCV error rate takes the form

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} Err_i \tag{4}$$

where $Err_i = I(y_i \neq \hat{y}_i)$. The k-fold CV error rate and validation set error rates are defined analogously.

As an example, we fit various logistic regression models on the two-dimensional classification data displayed in Lecture 1 Figure 13. In the top-left panel of Figure 7, the black solid line shows the estimated decision boundary resulting from fitting a standard logistic regression model to this data set. Since this is simulated data, we can compute the true test error rate, which takes a value of 0.201 and so is substantially larger than the Bayes error rate of 0.133. Clearly logistic regression does not have enough flexibility to model the Bayes decision boundary in this setting. We can easily extend logistic regression to obtain a non-linear decision boundary by using polynomial functions of the predictors, as we did in the regression setting.

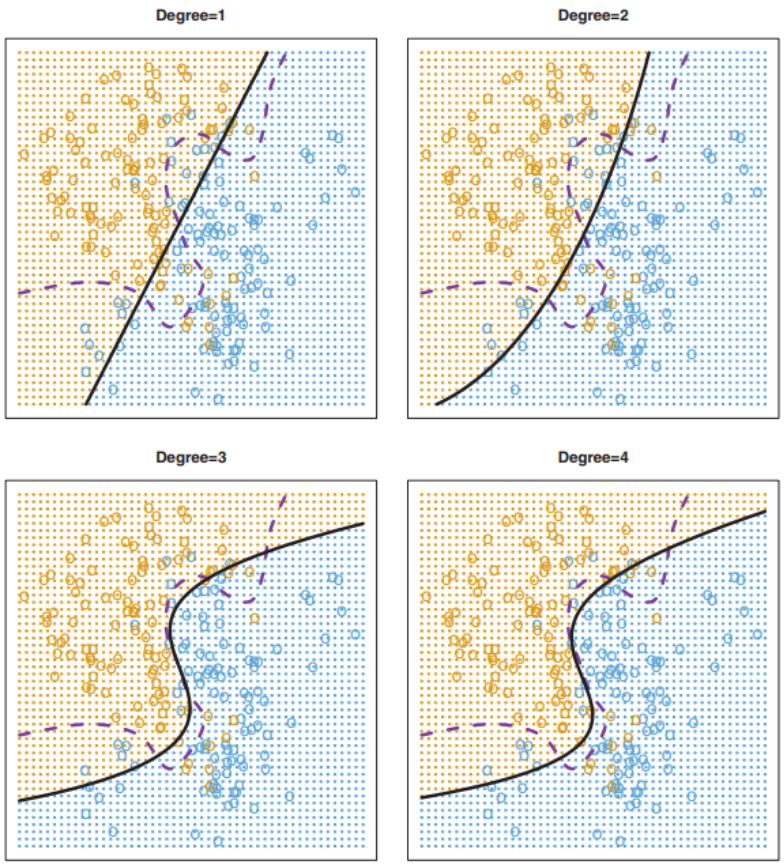

FIGURE 7. Logistic regression fits on the two-dimensional classification data displayed in Figure 2.13. The Bayes decision boundary is represented using a purple dashed line. Estimated decision boundaries from linear, quadratic, cubic and quartic (degrees 1–4) logistic regressions are displayed in black. The test error rates for the four logistic regression fits are respectively 0.201, 0.197, 0.160, and 0.162, while the Bayes error rate is 0.133.

The Bootstrap

The bootstrap is a widely applicable and extremely powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method. As a simple example, the bootstrap can be used to estimate the standard errors of the coefficients from a linear regression fit. In the specific case of linear regression, this is not particularly useful, since we saw that standard statistical software such as R outputs such standard errors automatically. However, the power of the bootstrap lies in the fact that it can be easily applied to a wide range of statistical learning methods, including some for which a measure of variability is otherwise difficult to obtain and is not automatically output by statistical software.

Here, we illustrate the bootstrap on a toy example in which we wish to determine the best investment allocation under a simple model. We will also explore the use of the bootstrap to assess the variability associated with the regression coefficients in a linear model fit.

Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities. We will invest a fraction α of our money in X, and will invest the remaining $1-\alpha$ in Y. Since there is variability associated with the returns on these two assets, we wish to choose α to minimize the total risk, or variance, of our investment. In other words, we want to minimize $Var(\alpha X + (1-\alpha)Y)$. One can show that the value that minimizes the risk is given by

$$\alpha = \frac{{\sigma^2}_Y - \sigma_{XY}}{{\sigma^2}_X + {\sigma^2}_Y - 2\sigma_{XY}} \tag{5}$$

Where ${\sigma^2}_X = Var(X)$, ${\sigma^2}_Y = Var(Y)$, and $\sigma_{XY} = Cov(X,Y)$. In reality, the quantities ${\sigma^2}_X, {\sigma^2}_Y$, and σ_{XY} are unknown. We can compute estimates for these quantities, ${\hat{\sigma}^2}_X, {\hat{\sigma}^2}_Y$, and $\hat{\sigma}_{XY}$, using a data set that contains past measurements for X and Y. We can then estimate the value of α that minimizes the variance of our investment using

$$\alpha = \frac{{\hat{\sigma}^2}_Y - \hat{\sigma}_{XY}}{{\hat{\sigma}^2}_X + {\hat{\sigma}^2}_Y - 2\hat{\sigma}_{XY}} \tag{6}$$

Figure 8 illustrates this approach for estimating α on a simulated data set. In each panel, we simulated 100 pairs of returns for the investments X and Y. We used these returns to estimate ${\sigma^2}_X, {\sigma^2}_Y$, and σ_{XY}, which we then substituted into Eq. (6) in order to obtain estimates for α. The value of $\hat{\alpha}$ resulting from each simulated data set ranges from 0.532 to 0.657.

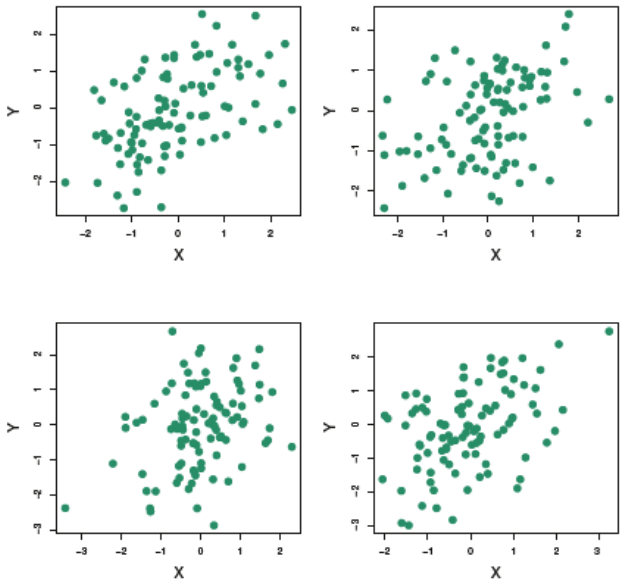

FIGURE 8. Each panel displays 100 simulated returns for investments X and Y. From left to right and top to bottom, the resulting estimates for α are 0.576, 0.532, 0.657, and 0.651.

It is natural to wish to quantify the accuracy of our estimate of α. To estimate the standard deviation of $\hat{\alpha}$, we repeated the process of simulating 100 paired observations of X and Y, and estimating α

using (5.7), 1,000 times. We thereby obtained 1,000 estimates for α, which we can call $\hat{\alpha}_1, \hat{\alpha}_2 \ldots \hat{\alpha}_{1000}$. The left-hand panel of Figure 9 displays a histogram of the resulting estimates.

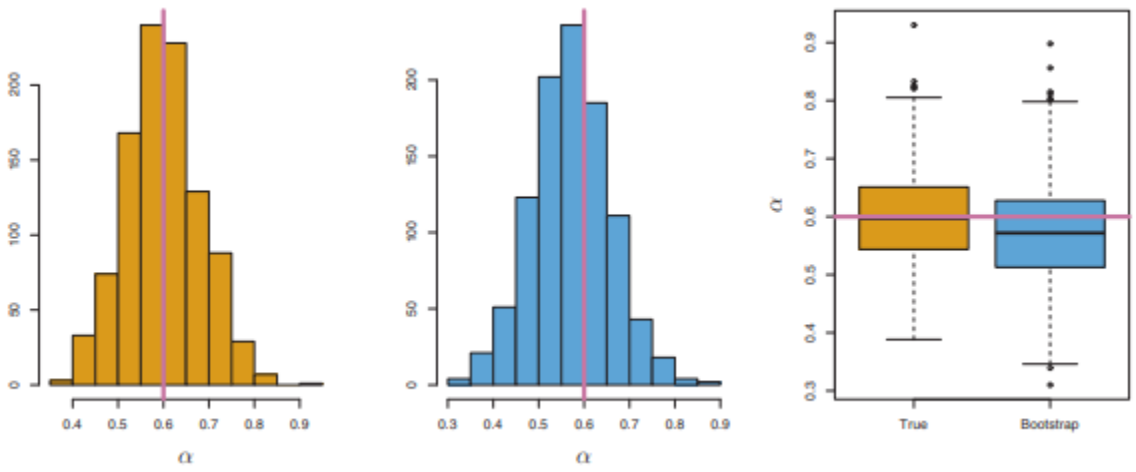

FIGURE 9. Left: A histogram of the estimates of α obtained by generating 1,000 simulated data sets from the true population. Center: A histogram of the estimates of α obtained from 1,000 bootstrap samples from a single data set. Right: The estimates of α displayed in the left and center panels are shown as boxplots. In each panel, the pink line indicates the true value of α.

The mean over all 1,000 estimates for α is

$$\bar{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}^r = 0.5996,$$

very close to $\alpha = 0.6$, and the standard deviation of the estimates is

$$\sqrt{\frac{1}{1000-1} \sum_{r=1}^{1000} (\widehat{\alpha_r} - \bar{\alpha})^2} = 0.083$$

This gives us a very good idea of the accuracy of $\hat{\alpha}$: $SE(\hat{\alpha}) \approx 0.083$. So roughly speaking, for a random sample from the population, we would expect $\hat{\alpha}$ to differ from α by approximately 0.08, on average.

In practice, however, the procedure for estimating SE($\hat{\alpha}$) outlined above cannot be applied, because for real data we cannot generate new samples from the original population. However, the bootstrap approach allows us to use a computer to emulate the process of obtaining new sample sets, so that we can estimate the variability of $\hat{\alpha}$ without generating additional samples. Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set.

This approach is illustrated in Figure 10 on a simple data set, which we call Z, that contains only $n = 3$ observations. We randomly select n observations from the data set in order to produce a bootstrap data set, Z^{*1}. The sampling is performed with replacement, which means that the same observation can occur more than once in the bootstrap data set. In this example, Z^{*1} contains the third observation twice, the first observation once, and no instances of the second observation. Note that if an observation is contained in Z^{*1}, then both its X and Y values are included. We can use Z^{*1} to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^{*1}$. This procedure is repeated B times for some large value of B, in order to produce B different bootstrap data sets, $Z^{*1}, Z^{*2} \ldots Z^{*B}$, and B corresponding α estimates, $\hat{\alpha}^{*1}, \hat{\alpha}^{*2} \ldots \hat{\alpha}^{*B}$. We can compute the standard error of these bootstrap estimates using the formula

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}(\hat{\alpha}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}^{*r'})^2}$$

This serves as an estimate of the standard error of $\hat{\alpha}$ estimated from the original data set.

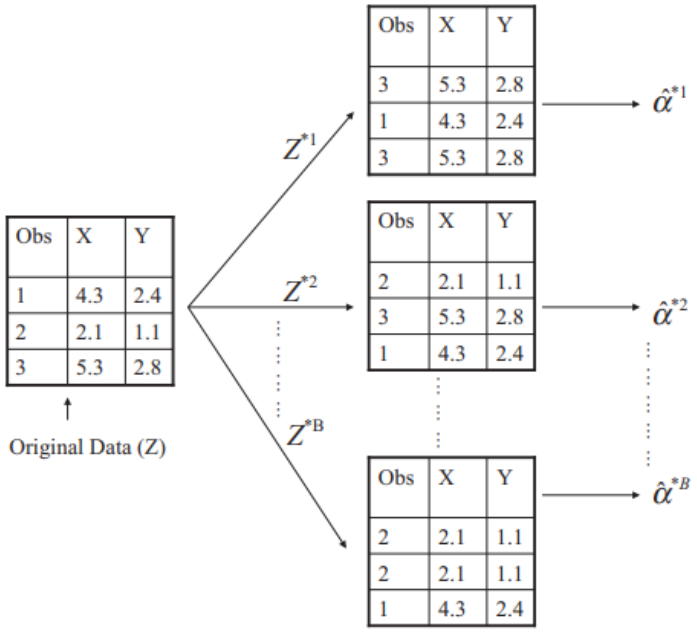

FIGURE 10. A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ =observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α.

The bootstrap approach is illustrated in the center panel of Figure 10, which displays a histogram of 1,000 bootstrap estimates of α, each computed using a distinct bootstrap data set. This panel was constructed on the basis of a single data set, and hence could be created using real data.

Note that the histogram looks very similar to the left-hand panel which displays the idealized histogram of the estimates of α obtained by generating 1,000 simulated data sets from the true population. In particular the bootstrap estimate ($SE(\hat{\alpha})$ from (5.8) is 0.087, very close to the estimate of 0.083 obtained using 1,000 simulated data sets.

Case Study: Microtubule Dynamic Instability Modeling

Microtubules (MTs), cylindrical polymers of $\alpha\beta$-tubulin dimers, are fundamental to cell structure and function. They exhibit dynamic instability, spontaneously switching between four distinct kinetic states: growth, shortening (shrinkage), rescue (switching from shortening to growth), and catastrophe (switching from growth to shortening). While experiments can track MT length changes over time, they struggle to simultaneously and with sufficient temporal resolution reveal the underlying structural and energetic molecular features that govern these transitions. This limited view makes it difficult to answer central questions, such as what specifically triggers catastrophe or rescue, and if the determinants of these states change under different cellular conditions.

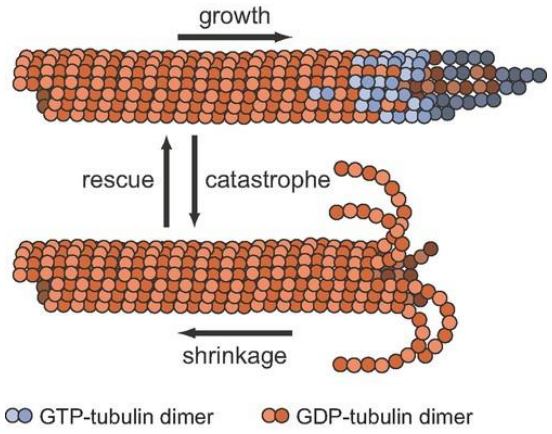

FIGURE 11. the 4 kinetic states of microtubule: growth, rescue, shrinkage (shortening), and catastrophe [1].

The Modeling Approach

To overcome these experimental limitations, Kliuchnikov et al. [2] developed the Microtubule Assembly and Disassembly DYnamics (MADDY) computational model. This model was rigorously parameterized and validated against experimental measurements of MT growth rates, shortening rates, and force generation, ensuring the simulations captured realistic MT behavior. The authors created six case studies to simulate:

- Case Studies 1–3 simulate *physiological* conditions (normal tubulin concentration, 25 μM) but with increasing GTP hydrolysis rates — testing how chemical energy turnover affects stability.
- Case Studies 4–6 simulate *high-tubulin* conditions (250 μM) where microtubule growth is faster, but also more dynamic, again across increasing hydrolysis rates.

Utilizing GPU acceleration, the researchers generated rich time-series data, tracking both MT length and molecular structure as the system spontaneously transitioned between the four kinetic states.

The output from these simulations was used to construct a supervised learning problem. Trajectories were manually labeled into one of the four kinetic states based on the change in length over time. Simultaneously, fourteen quantitative features were extracted at each time point, encompassing both structural and energetic properties.

- **Structural features:** MT length (L), Number of hydrolyzed GDP-bound dimers (n_{hyd}), MT tip length (l) and width (w), Average longitudinal curvature (κ_{long}), Average lateral curvature (κ_{lat}), Number of longitudinal and lateral interactions (n_{long}, κ_{lat})
- **Energetic features:** Total lateral and longitudinal interaction energies in the lattice (U_{lat}, U_{long}), Interaction energies at the MT tip (u_{lat}, u_{long}), Energies required to complete the MT to a full cylinder ($U_{lat}^{add}, U_{long}^{add}$)

The challenge was to correctly classify the kinetic state (label) based solely on the 14 feature measurements at that instant.

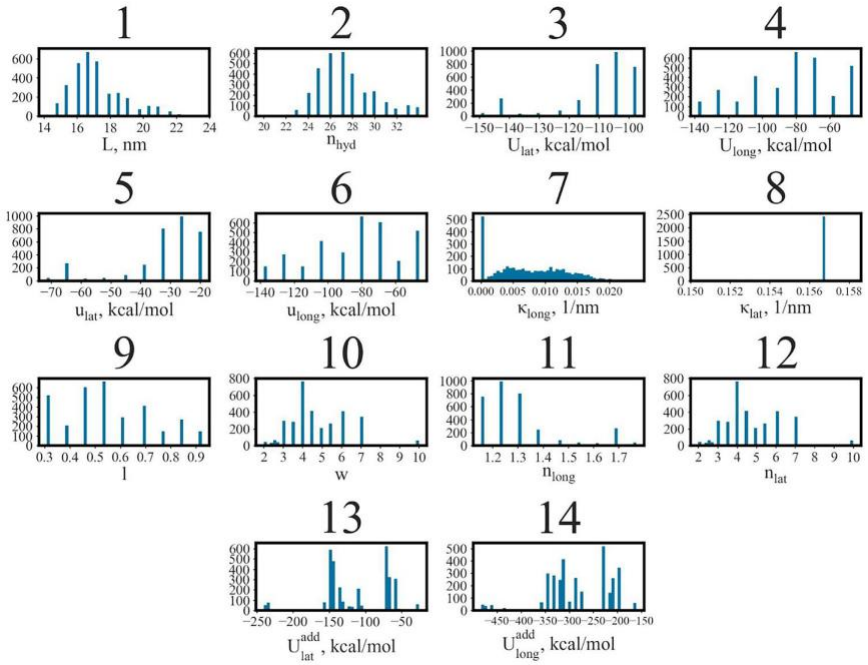

FIGURE 12. Fourteen quantitative features used for machine learning classification: (1) MT length (L), (2) number of hydrolyzed αβ-tubulin dimers in the lattice, (3) total lateral interaction energy, (4) total longitudinal interaction energy, (5) lateral interaction energy at the tip, (6) longitudinal interaction energy at the tip, (7) average longitudinal curvature, (8) average lateral curvature, (9) MT tip length, (10) MT tip width, (11) average number of longitudinal interactions per protofilament, (12) average number of lateral interactions per helical pitch, (13) lateral energy required to complete the cylinder, and (14) longitudinal energy required to

complete the cylinder. These features were selected based on current hypotheses about molecular control of MT growth and shortening.

Robust Assessment and Classifier Performance

This approach is crucial because: 1) it addresses the validation set variability. A single train-test split can give misleading results depending on which observations happen to be in the test set. 2) It maximizes data usage. Unlike a single validation set approach, all data are used for both training and testing. 3) It provides uncertainty estimates. The variance across the 15 trials indicates how stable the model performance is.

Multiple classification algorithms were evaluated:

- Single classifiers: Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), Logistic Regression (LR), K-Nearest Neighbors (KNN)
- Ensemble methods: Bagging Classifier (BAG), Random Forest (RF), Extra Trees (ET), XGBoost (XGB)

The researchers also compared their results to Leave-One-Out Cross-Validation (LOOCV) for selected cases. While LOOCV gave similar accuracy estimates, it was computationally prohibitive given the dataset size and number of models being compared. The 5-fold CV provided nearly the same information at much lower computational cost.

To evaluate classification performance, the authors trained several algorithms to distinguish between the four kinetic states of MT dynamics—assembly, catastrophe, shortening, and rescue—based on the 14 structural and energetic features extracted from MADDY simulation outputs. The baseline accuracy, computed using a Dummy Classifier, ranged from 33–59% across six case studies (1–6), depending on which kinetic state dominated the dataset. Because growth and catastrophe were much more frequent than rescue or shortening, the authors applied the Synthetic Minority Oversampling Technique (SMOTE) to balance the classes by replicating underrepresented examples.

Among traditional classifiers, Logistic Regression (LR) without SMOTE performed the worst (43.1–65.1% accuracy), while Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) with SMOTE achieved the highest single-model performance (64.2–84.1%). Ensemble approaches—Bagging (BAG), Random Forest (RF), and Extreme Gradient Boosting (XGB)—significantly outperformed all single classifiers, achieving accuracies between 74.0% and 87.3% across all six case studies. These results are summarized in Table 2 of the original paper, which identified RF and XGB as the top-performing models for the majority of cases.

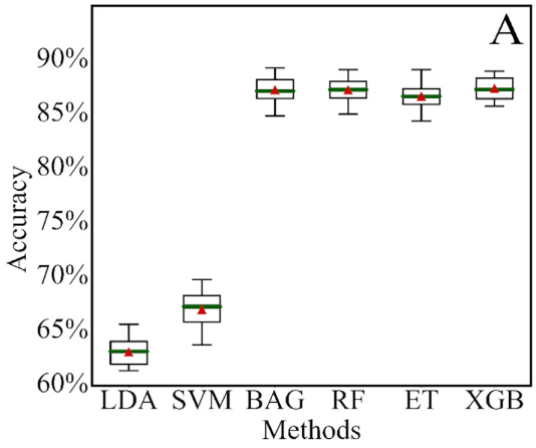

FIGURE 13. Box and whisker plots comparing prediction accuracy for various classification and ensemble methods (LDA, SVM, BAG, RF, ET, and XGB). The green line indicates the median for each method, while red arrowheads indicate the mean values.

For example, Case Study 1 (25 μM tubulin, $k_h yd = 2\ s^{-1}$) achieved 74% accuracy using RF, while Case Study 2 (25 μM, $k_h yd = 4\ s^{-1}$) reached 87.1% using RF/XGB. Under high-concentration conditions (250 μM tubulin, $k_h yd = 5–10\ s^{-1}$, Case Studies 4–6), ensemble accuracy remained robust (77–87%), confirming that these models generalize well across biochemical conditions.

Confusion matrix analyses revealed that growth and catastrophe were predicted with the highest precision, typically exceeding 90–95%, while rescue and shortening were more difficult to classify due to overlapping feature distributions. Prediction accuracy was positively correlated with class population size—states with more samples (growth, catastrophe) were consistently better classified than rare states (rescue, shortening). XGB in particular achieved more consistent classification across all states than RF, suggesting stronger sensitivity to minority-state patterns.

To further quantify performance, the authors used Precision-Recall (PR) and Receiver Operating Characteristic (ROC) curves. Because the dataset was imbalanced, PR curves were prioritized as a more informative metric. The Area Under the PR Curve (AUC) values confirmed the high discriminative ability of ensemble methods:

- Growth: AUC 0.86–0.96
- Catastrophe: AUC 0.91–0.95
- Shortening: AUC 0.60–0.86
- Rescue: AUC 0.63–0.91

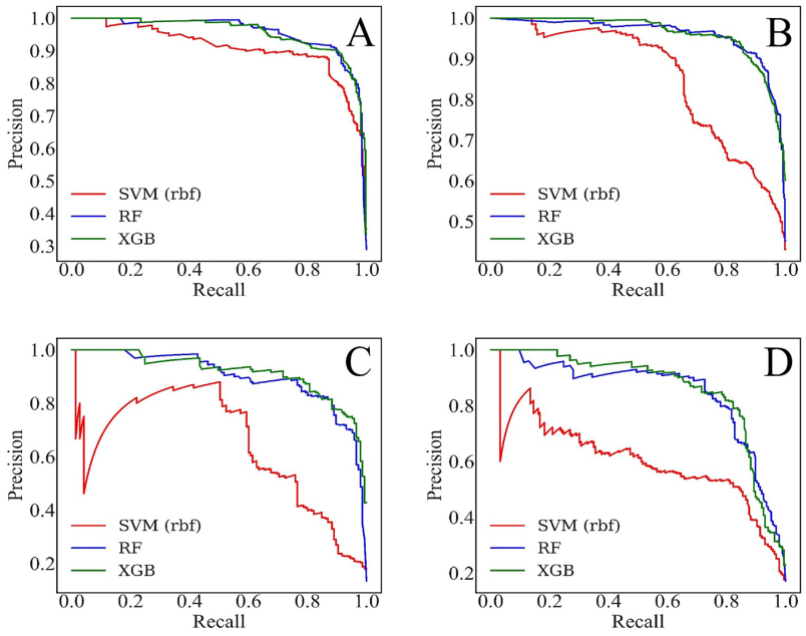

FIGURE 14. Classification quality analysis with Precision-Recall curves: PR curves comparing the overall quality value of classification attained using SVM with RBF kernel (red curves), Random Forest (blue curves), and XGBoost (green curves) for MT catastrophe (panel A), MT growth (panel B), MT rescue (panel C), and MT shortening (panel D). The closer the curve is to the upper right-hand corner, the better is the model prediction.

RF and XGB outperformed SVM in nearly all cases, with XGB yielding slightly higher AUC values overall. For example, in Case Study 2, both RF and XGB achieved AUCs of 0.96 for growth and 0.95 for catastrophe, while SVM lagged at 0.86 and 0.91, respectively.

Lastly, the authors used SHapley Additive exPlanations (SHAP) analysis to interpret the predictions and assign an importance value to each feature. The most significant finding was a concentration-dependent shift in feature importance: the molecular features controlling dynamic instability at a physiological tubulin concentration (25 μM) were drastically different from those at a high concentration (250 μM) (Figure 15).

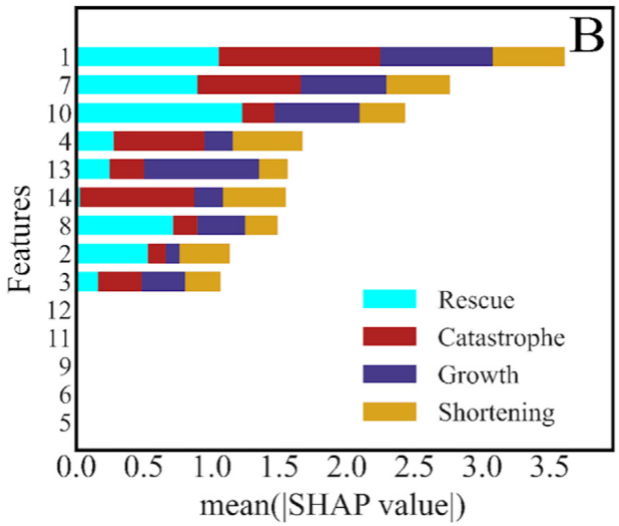

FIGURE 15. Feature importance plots obtained with the XGB method, ranking the features by their mean absolute SHAP values — higher SHAP values correspond to stronger influence on the predicted kinetic state.

This insight—that the molecular determinants of dynamic instability are condition-dependent—was a novel discovery that demonstrates the power of combining physically realistic computational models with interpretable machine learning.

This work serves as a powerful example of how machine learning can guide experimental design, advising biophysicists to focus their efforts on 1) Measuring MT tip width and lattice structure at low tubulin concentrations, and/or 2) Tracking GTP/GDP composition and lateral interactions at high concentrations.

References

1. Bowne-Anderson H, Zanic M, Kauer M, Howard J. Microtubule dynamic instability: a new model with coupled GTP hydrolysis and multistep catastrophe. Bioessays. 2013 May;35(5):452-61. doi: 10.1002/bies.201200131. Epub 2013 Mar 27. Erratum in: Bioessays. 2013 Jun;35(6):579. PMID: 23532586; PMCID: PMC3677417.
2. Kliuchnikov E, Klyshko E, Kelly MS, Zhmurov A, Dima RI, Marx KA, Barsegov V. Microtubule assembly and disassembly dynamics model: Exploring dynamic instability and identifying features of Microtubules' Growth, Catastrophe, Shortening, and Rescue. Comput Struct Biotechnol J. 2022 Jan 31;20:953-974. doi: 10.1016/j.csbj.2022.01.028. PMID: 35242287; PMCID: PMC8861655.