INSTITUTE FOR DEFENSE ANALYSES

Quantifying and Visualizing Forecast Uncertainty with the FIFE

Evan Miyakawa
John W. Dennis
James Bishop
Alan Gelder

December 2021
Approved for public release;
distribution unlimited
IDA Paper NS P-31857
Log: H 21-000409

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

About this Publication
The work was conducted by the Institute for Defense Analyses (IDA) under CRP C6523.

For More Information:
Dr. Alan B. Gelder, Project Leader
agelder@ida.org, 703-845-6879
ADM John C. Harvey, Jr., USN (Ret), Director, SFRD, IDA
jharvey@ida.org, 703-575-4530

INSTITUTE FOR DEFENSE ANALYSES

IDA Paper NS P-31857

Quantifying and Visualizing Forecast Uncertainty with the FIFE

Evan Miyakawa
John W. Dennis
James Bishop
Alan Gelder

Executive Summary

Survival analysis can be a useful tool for modeling the attrition of service members, particularly when it comes to forecasting future states of survival for those members. Government sponsors are often interested in predicting these attrition rates at future time points. The Institute for Defense Analyses (IDA) has developed a tool for this purpose: the Finite Interval Forecasting Engine (FIFE) (Institute for Defense Analyses 2021). FIFE is a forecasting tool that produces predictions through the use of various modeling frameworks, including deep neural networks and gradient boosted trees. FIFE combines methods from both survival analysis and multivariate time series analysis to predict future states of survival, along with total counts of attrition, for service members at various future points in time.

We discuss methods for quantifying uncertainty in these survival forecasts, both for individual probabilities of exit, and aggregated total exits. While FIFE currently uses advanced approaches for maximizing forecasting performance, through the use of LightGBM for gradient-boosted trees, and Keras for neural networks, there are little-to-no implemented methods for measuring uncertainty in these predictions. Point estimates for future values of interest can be close to the truth, but they are never correct. In some classification prediction problems, future realized data that occurs far from the training data can be classified incorrectly with high probability. Use of prediction intervals around those point estimates leads to appropriate understanding of the accuracy of these predictions. Having a probability distribution associated with a future value, instead of just a point estimate, facilitates understanding of the uncertainty associated with forecasts.

We define relevant terminology in relation to prediction uncertainty and address how these terms differ based on field of study, and we discuss the properties of prediction intervals and describe the specifics of our objective in adding methods to FIFE that can produce these intervals. Our literature review investigates differing approaches to quantifying forecast uncertainty, which includes generic methods and learner-specific methods. We then implement a few of these approaches in FIFE and discuss their performance.

This page is intentionally blank.

Contents

This page is intentionally blank.

1. Introduction

Survival analysis can be a useful tool for modeling the attrition of service members, particularly when it comes to forecasting future states of survival for those members. Government sponsors are often interested in predicting these attrition rates at future time points. The Institute for Defense Analyses (IDA) has developed a tool for this purpose: the Finite Interval Forecasting Engine (FIFE) (Institute for Defense Analyses 2021). FIFE is a forecasting tool that produces predictions through the use of various modeling frameworks, including deep neural networks and gradient boosted trees. FIFE combines methods from both survival analysis and multivariate time series analysis to predict future states of survival, along with total counts of attrition, for service members at various future points in time.

We discuss methods for quantifying uncertainty in these survival forecasts, both for individual probabilities of exit, and aggregated total exits. While FIFE currently uses advanced approaches for maximizing forecasting performance, through the use of LightGBM for gradient-boosted trees, and Keras for neural networks, there are little-to-no implemented methods for measuring uncertainty in these predictions. Point estimates for future values of interest can be close to the truth, but they are never correct. In some classification prediction problems, future realized data that occurs far from the training data can be classified incorrectly with high probability. Use of prediction intervals around those point estimates leads to appropriate understanding of the accuracy of these predictions. Having a probability distribution associated with a future value, instead of just a point estimate, facilitates understanding of the uncertainty associated with forecasts.

We can imagine a situation where a service needs to forecast attrition to make decisions regarding force planning. A point prediction simply informs the individual of the expected attrition rate, and says nothing regarding how close one can expect the actual attrition rate to be to the forecast. An interval forecast, on the other hand, can inform the planner that the actual attrition rate is expected to fall into some range with some specified probability. Two methods can produce the same point forecast but different interval forecasts. A shorter interval with good properties would indicate that we are fairly certain regarding the actual future attrition rate compared to a comparable interval that is wider. Understanding the properties of these intervals is necessary to understand how well they capture uncertainty.

Further, certain situations may result in the need to budget for situations other than the expected mean or median attrition rate. For example, forecasting an upper bound on attrition with some probability may be more appropriate for a critical occupation. Forecast intervals can facilitate understanding of such bounds.[1]

The paper will proceed as follows: We define relevant terminology in relation to prediction uncertainty and address how these terms differ based on field of study. We will then discuss the important properties of prediction intervals and describe the specifics of our objective in adding methods to FIFE that can produce these intervals. Our literature review investigates differing approaches to quantifying forecast uncertainty, which includes generic methods and learner-specific methods. We then implement a few of these approaches in FIFE and discuss their performance.

Note that our proposed methods are not producing any mechanism to explain our predictions or make any kind of causal inference. Our methods measure uncertainty surrounding predictions that FIFE makes; they do not analyze any of the underlying covariates or suggesting correlation between any variables.

A. Terminology

First, we define a few terms that are important to the content of this paper. We should note the distinction between *prediction interval* and *confidence interval* in the literature, and also address how these terms are used different in statistics versus econometrics. Chatfield (1993) describes the difference: "The term *confidence interval* is usually applied to estimates of (fixed but unknown) parameters. In contrast, a prediction interval is an estimate of an unknown future value that can be regarded as a random variable at the time the forecast is made." In general, prediction intervals account for more sources of uncertainty than confidence intervals do, because of the unknown factors that will affect future outcomes. These sources of uncertainty include things like model misspecification and noise from random error in the data, and, as such, lead to intervals that have wider ranges than confidence intervals would, given the same chosen confidence level (Khosravi et al. 2011). In econometrics, CIs and PIs are sometimes used interchangeably, which can lead to some confusion. In some cases, a confidence interval for an unknown parameter, such as a mean of a random variable, can be used as the prediction interval for future realizations, which may cause these intervals to be too tight and have coverage probability much below the desired level (Hyndman 2013). In this paper, we focus on prediction intervals, which we will sometimes also interchangeably call *forecast intervals*, in order to note that the intervals are quantifying uncertainty related to forecasted values.

[1] Other methods also exist to help with this problem, but they will not be discussed here.

B. Prediction Interval Properties

There are two main properties we desire for quality prediction intervals: coverage probability and interval width. When we construct a prediction interval (or confidence interval) with a given confidence level (95% for example), we are indicating that the interval will ideally have at least a 95% chance of containing the future unrealized value of the random variable of interest. In practice, this is called the PI coverage probability (PICP), which is measured as

$$PICP = \frac{1}{n}\sum_{i=1}^{n} c_i,$$

where $c_i = 1$ when the interval contains the true value, and $c_i = 0$ otherwise. Since an actual future realization only occurs once, we can use simulation techniques to obtain the PICP for a model by treating observed data as training data.

As Khosravi et al. (2011) notes, if all we care about is achieving at least nominal coverage probability, we could make every interval infinitely wide, which would give us no new information about the accuracy of our predictions. Along with achieving the desired coverage level, we also want our intervals to have the smallest width possible. This is measured by the mean prediction interval width (MPIW):

$$MPIW = \frac{1}{n}\sum_{i=1}^{n}\big(U(X_i) - L(X_i)\big),$$

where $U(X_i)$ is the upper bound and $L(X_i)$ is the lower bound of the PI for the ith sample.

As we compare prediction interval techniques, we will use these measures to determine which approaches lead to the highest quality PIs.

C. Types of Uncertainty

First, we briefly discuss the various types of uncertainty that are present when building models and making forecasts, some of which we can quantify, and some we cannot. The literature generally distinguishes three categories of uncertainty with differing methods to quantify each: approximation uncertainty, model (or epistemic) uncertainty, and data (or aleatoric) uncertainty (Hüllermeier and Waegeman 2021).

Approximation uncertainty exists because we cannot always pick the model from all evaluated candidate models that will perform the best in practice. The signal that a model is trying to find is often confused with noise and random error in our training data. Having a larger number of candidate models that are evaluated leads to a higher chance of the chosen model, picked based on some type of performance score, having simply exploited the noise the best, without actually being best suited to model future unrealized data. Approximation uncertainty can be lowered by trying to have training and testing data that is as similar as possible to the future real data, but it can never be eliminated entirely.

Model uncertainty (also known as epistemic uncertainty) arises from the fact that we cannot evaluate all possible models, and may not have the true model in our set of candidate models that are evaluated. Our lack of knowledge about the true structure of the model leads to this uncertainty (Chatfield 1995). Estimates of model uncertainty are only accurate if our set of candidate models is representative of all possible models, which practically is challenging given that we can only ever evaluate a finite number of models. Specific methods for quantifying epistemic uncertainty depend on the type of modeling framework used, such as a neural network or gradient boosted tree modeler.

Data uncertainty (also known as aleatoric uncertainty), is an irreducible type of uncertainty that occurs because of the stochastic nature of all random variables we would be trying to predict. Simply put, a forecasted probability or value will almost never equal the actual observed outcome. Aleatoric uncertainty is accounting for the randomness of actual data, describing the variance of the conditional distribution of our target variable, due to unmeasured variables, or things like measurement error (Tagasovska and Lopez-Paz 2019).

The following example illustrates these three types of uncertainty: Suppose we are trying to predict the future probability p that a non-traditional die will roll a 1. Let X represent the face that die lands on when it is rolled. Suppose we choose a set of candidate models with the probability mass function of $p(x) = \frac{1}{n}$, where n is the total number of sides on that dice. If we have a set of data from previous dice rolls, or our own manufactured training data, approximation uncertainty occurs when one candidate model is perceived to be the best in relation to our training data ($p(x) = \frac{1}{5}$ for example), when in practice a different model will give us the best predictions (say $p(x) = \frac{1}{4}$). Model (epistemic) uncertainty represents the fact that we may not know enough about the true structure of the model. Suppose that the true model is actually of the form $p(x) = \frac{c}{n}$, where c is the number of sides on the dice that have a 1 on them. Our set of candidate models is not representative of the true model, since we do not consider the fact that a dice could have a number of sides that have the same integer on them. In this case, quantifying this model uncertainty can correctly demonstrate our lack of knowledge about the true model. Finally, data (aleatoric) uncertainty comes from the fact that the proportion of times we observed the desired outcome (the proportion of times we roll a 1) will often not equal our predicted probability of that outcome, due to the randomness of the actual dice roll. In a perfect world, we could greatly reduce this uncertainty by knowing variables like the launching point of the dice and the physics of its movement, but accounting for all factors in practice is impossible. We can only aim to have a model that comes as close as possible to predicting the true outcomes.

The methods that we explore in this paper make attempts to measure some of these types of uncertainty through prediction intervals. However, just because we measure some of the uncertainty in these intervals, we should not disregard the rest as if they do not exist. It can be dangerous to give a measurement for some of the unknown that we have in our predictions, only to lead others into a false sense of security that nothing else could be outside of our scope of knowledge. Educating others on uncertainty that a method cannot quantify is just as important as noting what it can.

D. Task Specifics

In this paper, we explore methods for measuring uncertainty, particularly in relation to the forecasts generated by the Finite Interval Forecasting Engine from IDA. FIFE has two different approaches available for predicting the future probability of survival for individuals in the service at future time points: a deep neural network modeler through Keras[2] and a gradient boosted tree modeler through LightGBM.[3] Our aim is to explore possible methods for adding prediction intervals in relation to these forecasts, and implementing some of the most promising ones as FIFE methods. Adding these measures of uncertainty quantification will enhance the ability of those who are using these predictions to make proper decisions based on how certain these forecasts are likely to be. We aim to compute prediction intervals, both for the individual probability of survival for a given time point, as well as the overall count of exits at a time point for a number of individuals.

[2] Keras documentation: https://keras.io/

[3] LightGBM documentation: https://lightgbm.readthedocs.io/en/latest/

This page is intentionally blank.

2. Methods

There are two categories of prediction interval methods that we explore in relation to quantifying uncertainty in survival data predictions: generic methods, and learner-specific methods. A generic PI method indicates that the strategy does not need to know anything about the type of training method used in creating the predictions, but instead can be applied to any type of survival data time series predictions. These methods are useful in that they can be applied in many scenarios, but also tend to be more conservative in their prediction interval estimation than others.

Chernoff bounds are prediction intervals derived theoretically for the sum of independent Bernoulli trials, each with its own independent probability p_i. Through the use of Chebyshev's inequality, a nice expression is obtained for a $(1-\alpha)\%$ prediction interval, conditional on just the sum of predicted probabilities of success $\sum_{i=1}^{n} p_i$. These bounds tend to be very conservative by nature, giving wider intervals than obtained elsewhere (Goemans 2015).

Techniques for computing exact coverage probability of any set of existing prediction intervals derived theoretically (with a formula) are detailed by Wang (2008). By using simulation techniques to compute the coverage probability for a supposed value of p, given a fixed sample size, a *minimum coverage probability* and *average coverage probability* can be calculated by getting these coverage probabilities across a fine grid of possible p values. Wang further suggests a method for altering existing PIs based on the average coverage probability, in order to ensure that the final PI will have average coverage probability equal to the nominal coverage desired.

A clustering technique using K-Nearest-Neighbors regression (Huang and Perry 2016) provides an interesting way of obtaining prediction intervals, in that similar scenarios from previous historical data are used to form a probabilistic distribution of forecasting error, for any random variable to have a future value predicted. For a given individual who has a future value predicted, a search for previous observations in the historical data who shared similar covariates is conducted using KNN regression, and the realized outcomes for those similar observations are combined to create a distribution of possible forecast errors, from which PIs can be derived. Similarity in this method is quantified by a measure of Euclidean distance in a hyperspace with dimensions defined by a number of selected variables. Huang suggests having time be represented as a covariate as well, in order to give appropriate lesser weight to training data that occurred further in the past.

Next, we turn to learner-specific methods, which use the underlying modeling framework to construct appropriate prediction intervals for forecasts. Since FIFE uses neural network and gradient boosted tree strategies, we will focus our literature review on these two specific learner types. There are multiple ways of constructing PIs using theoretical or asymptotic formulations, such as Hwang and Ding's (1997) method, which uses an asymptotic result to form prediction intervals for single layer neural network predictions. However, the method assumes that noise is normal and homogeneous, which is not guaranteed in most scenarios. Many of these theoretical PIs either make assumptions that are often violated, or may not reach nominal coverage probability in practice.

Bayesian methods for constructing prediction intervals using a posterior predictive distribution for a random variable of interest are very common, and often highly accurate. In cases of large data (as is often the case with FIFE datasets), the computational cost of implementing these methods is not as feasible. The same can be said for bootstrap-type PI methods for neural networks, in that they are also very computationally expensive in big data settings, although they are simple and easy to implement.

Gal and Ghahramani (2016) uses an ensemble of neural network models with dropout to approximate a Bayesian posterior distribution when constructing prediction intervals, yielding similar accuracy to Bayesian methods without being too costly. The method, called MC Dropout, can be used with any deep neural network model that has dropout applied between each hidden layer, and utilizes predictions from independent models to form a probability distribution for future outcomes. The paper demonstrates how the predictions from these independent NN models approximate the true Bayesian posterior predictive distribution. MC Dropout is very easy to use, and is almost as fast to run as the original neural network model if parallelization is possible.

Another promising NN ensemble method is the one provided by Lakshminarayanan, Pritzel, and Blundell (2017), which also uses independent NN models to approximate a Bayesian posterior. This method includes a measurement for quantifying *out-of-distribution shift*.[4] Well-calibrated predictions using this approach are robust to model misspecification and out-of-distribution shift. However, implementing this method is a little more challenging than Gal's MC Dropout method.

While the previous methods are all based on fitting prediction intervals to existing neural network forecasts training using a prediction error loss function, Khosravi et al. (2011) introduces a method called Lower-Upper Bound Estimation (LUBE), that trains deep neural networks on a loss function constructed to yield the highest quality prediction intervals. If the ultimate goal is to generate accurate predictions, then building your neural network model to output the tightest prediction intervals, with the desired coverage probability, makes a lot of sense. Khosravi et. al gives a strategy for constructing a prediction interval quality score, that values both coverage probability and interval width, which can be used as the loss function for the NN training. Each

[4] Out-of-distribution shift, or data shift, occurs when certain properties of the dataset used for training are different from those of the dataset used for other purposes, such as testing or forecasting. A neural network that is making predictions for a dataset that is different in this respect from the training data should have a much higher level of predictive uncertainty.

prediction will then yield two values, a lower and upper bound for the prediction interval, instead of just a single point estimate.

There are two prediction interval methods for stochastic gradient boosted trees that we find intriguing for use in FIFE. The first, proposed by Malinin, Prokhorenkova, and Ustimenko (2021), constructs an ensemble of independent SGBT models to get a sense of the uncertainty in our predictions. While there is no guarantee in how well a distribution of predictions from usual independent stochastic gradient boosted tree models will approximate the true Bayesian posterior predictive distribution for a random variable, Malinin uses Langevin dynamics (Ustimenko and Prokhorenkova 2021) introduced in Stochastic Gradient Langevin Boosting (SGLB) to ensure that ensembles of predictions do approximate the true posterior. These SGLB models inject Gaussian noise into the gradients and use L2 regularization in a sequential boosting algorithm, in order to enable proper sampling from the posterior.

Malinin also proposes a method for cutting down the computation time of these prediction intervals by using just one SGLB model, and forming a virtual ensemble of models by taking sequential subsets of trees within the single model to approximate a full independent ensemble. While this technique does yield lower quality prediction intervals, compared to an independent ensemble of SGLB models, it outperforms any estimate of prediction uncertainty that would otherwise be gained through a traditional single stochastic gradient boosted tree model. These methods also include a measure for distinguishing epistemic and aleatoric uncertainty in the model predictions, which is a very nice feature. The main drawback with these approaches in relation to FIFE is that they require set choices of learning rate and L2 regularization in order to theoretically approximate the true posterior predictive distribution, which doesn't allow for any set of hyperoptimized choice of gradient boosted tree parameters to be used.

The Natural Gradient Boosting (NGBoost) algorithm (Duan et al. 2020) is another good choice for constructing prediction intervals for gradient boosted trees. NGBoost allows for any choice of base learner (such as a gradient boosted tree), parametric probability distribution, and scoring rule / loss function (such as the MLE), to generalize gradient boosting to probabilistic regression and compute prediction intervals for random variables.

We next examine several of the aforementioned approaches, each of which we implement in some form into FIFE, and examine the performance of each.

A. Chernoff Bounds

Chernoff bounds are probability statements about random variables that give a probability of a random variable being greater than or less than a certain quantity, derived using Chebyshev's inequality.[5] There are different forms to Chernoff bounds, but we focus on the bounds for a sum

[5] Chebyshev's Inequality is $P(|X - E(X)| \geq a) \leq \frac{Var(X)}{a^2}$. For a sum of iid random variables, this can be reworked as $P\left(\left|\frac{\Sigma X_i}{n} - \mu\right| \geq \epsilon\right) \leq \frac{\sigma^2}{n\epsilon^2}$.

of independent Bernoulli trials, each with its own independent probability of success, p_i. We can use these to form prediction intervals around the count of exits from service for a group of individuals at a given future time point.

Let $X = \sum_{i=1}^{n} X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = E(X) = \sum p_i$. Then, for a chosen value of δ, the Chernoff upper bound is given by

$$P(X \geq (1 + \delta_U)\mu) \leq \left(\frac{e^{\delta_U}}{(1 + \delta_U)^{1+\delta_U}}\right)^{\mu} \text{ for all } \delta_U > 0, \tag{1}$$

and the Chernoff lower bound is given by

$$P(X \leq (1 - \delta_L)\mu) \leq \left(\frac{1}{e^{\delta_L}(1 - \delta_L)^{1-\delta_L}}\right)^{\mu} \text{ for all } 0 < \delta_L < 1. \tag{2}$$

By strategically choosing the value of δ, one can create bounds that have a specified level of confidence, such as 95% confidence.

Sometimes, a somewhat looser, but more convenient set of bounds are used in practice. The conservative upper bound is expressed as

$$P(X \geq (1 + \delta_U)\mu) \leq \exp\left(-\frac{\delta_U^2}{2 + \delta_U}\mu\right) \text{ for all } \delta_U > 0, \tag{3}$$

and the conservative lower bound is expressed as

$$P(X \geq (1 - \delta_L)\mu) \leq \exp\left(-\frac{\delta_L^2}{2}\mu\right) \text{ for all } 0 < \delta_L < 1. \tag{4}$$

By strategically choosing the value of δ, one can create bounds that have a specified level of confidence, such as 95% confidence (Goemans 2015).

1. Chernoff Bounds in FIFE

A previous implementation existed in FIFE to calculate the Chernoff bounds around a random variable that represents the sum of iid Bernoulli trials, with a given level of significance, α. However, the bounds use the more conservative form of the Chernoff bounds, given that the formulation simplifies the expressions for the bounds in order to analytically calculate the desired δ for a given significance level α. The function that calculates these bounds is the 'compute_aggregation_uncertainty()' function, inside 'utils.py'.

With this current formulation, we can solve for the specific δ needed to get the exact bounds that lead to the desired significance level α. For example, if $\alpha = 0.025$, we can set the upper tail bound $\exp\left(-\frac{\delta_U^2}{2+\delta_U}\mu\right)$ to be equal to 0.025, and solve for δ_U.

However, these formulations are more conservative than the original bounds in (1) and (2), and they produce wider intervals than necessary; we will use the original expressions to produce tighter intervals. This requires an optimization function, since there isn't always an analytical solution for δ in this case.

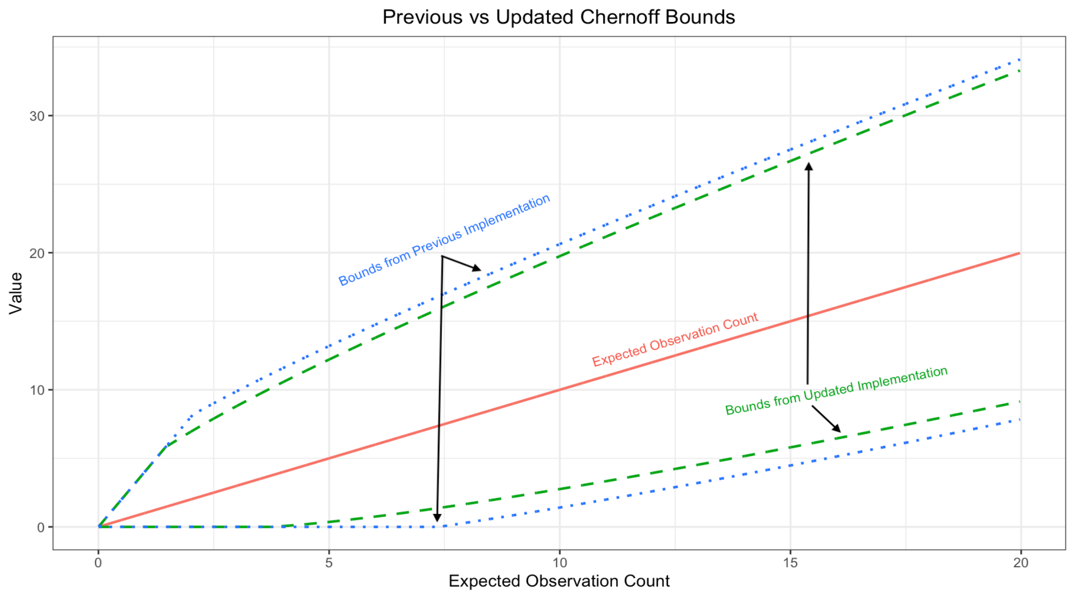
Our goal is to set α, our significance level, and then solve for δ, where $\alpha = \left(\frac{e^{\delta_U}}{(1+\delta_U)^{1+\delta_U}}\right)^{\mu}$. We can use an optimizer to find the single value of δ_U, the Chernoff upper bound, that minimizes the expression below:

$$\left[\left(\frac{e^{\delta_U}}{(1+\delta_U)^{1+\delta_U}}\right)^{\mu} - \alpha\right]^2. \tag{5}$$

Similarly, to find the lower bound, we minimize the expression:

$$\left[\left(\frac{1}{e^{\delta_L}(1-\delta_L)^{1-\delta_L}}\right)^{\mu} - \alpha\right]^2. \tag{6}$$

Using this approach in experimentation always leads to single solutions for δ. Even if this method were to fail, we could always fall back on the simplified version of the Chernoff Bounds, in (3) and (4), that can be calculated analytically. Figure 1 illustrates the tighter intervals that the original Chernoff Bounds give, compared to the simplified version currently implemented in FIFE.

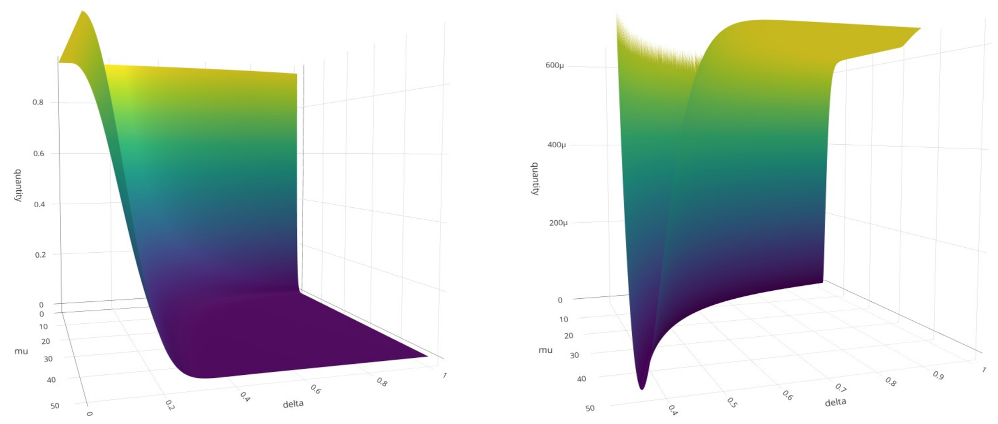


Note: A comparison of the conservative confidence bounds, (3) and (4), to the tighter confidence bounds, (1) and (2).

Figure 1. Comparison of Chernoff Bounds

2. 3D Plots of Minimization Function

Figure 2 provides a 3D plot of the function that is minimized in order to find the optimal δ values that lead to our calculated upper and lower bounds. The left plot shows the value of the function to minimize for the upper bound, given a grid of μ and δ values, for a fixed $\alpha = 0.025$. It is difficult to see where the function is minimized in the plot on the left, but we can gain more clarity by looking at the plot on the right, which enlarges the figure to focus on the areas where the function quantity is under 0.000625. The right pane indicates that the function has a global minimum for any given value of μ, leading to our unique solution for δ:



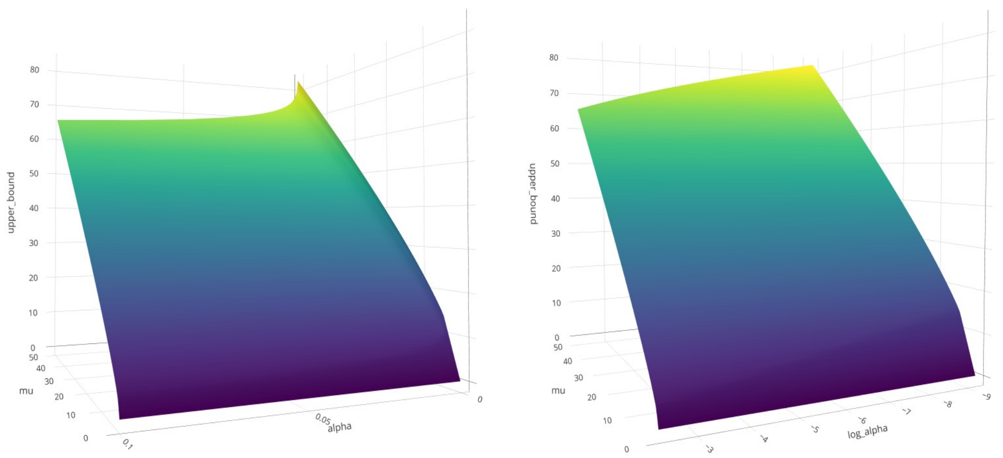
Note: These plots show the values of the function to be minimized for any given μ and δ values when calculating the Chernoff upper bound. The plot on the right is an enlarged version of the plot on the left. For any μ, there appears to be a global minimum, leading to a unique solution for δ.

Figure 2. Chernoff Bounds Objective Function

The plot of the minimization function for the lower bound is similar, so we won't include it here.

3. 3D Plots of Optimized Bounds

Figure 3 provides 3D plots of calculated Chernoff upper bound values for a grid of possible μ values (expected sum of independent Bernoulli trials) and α values (significance levels). The 3D plots for the lower bound values show similar behavior, but with the calculated bound values dipping lower towards small α values, not rising like in the plots above for the upper bounds.



Note: These plots show the calculated Chernoff upper bounds for a grid of μ and α values. The plot on the right is the log-scale version of the plot on the left.

Figure 3. Chernoff Bounds Calculations

B. MC Dropout for Deep Neural Networks

Bayesian methods for estimating prediction uncertainty in neural network (NN) models are usually accurate but computationally expensive; Gal and Ghahramani (2016) proposes a deep neural network model ensemble technique to approximate Bayesian methods without the additional cost, which addresses this issue. Gal proves that a collection of independent deep neural network model predictions, with dropout applied in each hidden layer, can be aggregated to approximate the true posterior predictive distribution of a random variable to be predicted. The proposed method, called MC Dropout, can be applied to any existing deep NN model that uses a constant dropout rate between each hidden layer. Neural network models in FIFE frequently use dropout, which makes MC Dropout a very enticing approach to obtaining prediction intervals in our case.

Dropout neural network models sample binary variables for every input point and every hidden unit inside each layer (except the last), with each variable taking value 1 with probability p_i and value 0 if not. A value of 0 corresponds to that unit being dropped for that instance. This technique is successfully used to prevent model over-fitting. Gal proves that the model outputs resulting from the addition of random dropout approximate the probabilistic deep Gaussian process, with no simplifying assumptions needed.[6]

1. Obtaining Model Uncertainty

Gal shows that the predicted values from each outcome of T independent deep NN models produce a posterior predictive distribution for each of those predicted random variables. In particular, for a chosen random variable, y_i^*, to be predicted, we can estimate the first two moments of the predictive distribution as follows:

$$E(y_i^*) \approx \frac{1}{T}\sum_{t=1}^{T} \hat{y}_{it}^*,$$

and

$$Var(y_i^*) \approx \tau^{-1} + \frac{1}{T}\sum_{t=1}^{T}\left(\hat{y}_{it}^* - E(y_i^*)\right)^2,$$

where τ is the model precision parameter. These estimates for the first two moments can be used to construct desired prediction intervals for random variables of interest.

According to Gal, while using a large number of independent NN models T can be helpful in getting an accurate estimate of the prediction intervals, a smaller number of models, such as 10, may be enough in many cases to get a usable estimate. Furthermore, if parallelization is possible,

[6] Additional technical details are provided in the appendix.

independent forward passes through the NN model can be done concurrently, leading to almost identical computation time as compared to just a single model run.

2. Choosing hyperparameters

There are several hyperparameters needed in MC Dropout in order to derive the model precision, τ, the inverse of which is added on to the variance of the NN predictions to obtain the total predictive variance of the model. The model precision τ is calculated as

$$\tau = \frac{p\ell^2}{2N\lambda}, \tag{7}$$

where p is the rate at which units are *not* being dropped, ℓ is the prior-length scale, N is the number of observations in the data, and λ is the weight-decay, functioning as the L2 regularization parameter in (13). The choice of ℓ greatly impacts τ, as the Gaussian process prior depends on ℓ, set to be shorter for high-frequency data and longer for low-frequency data. Unfortunately, there is very little instruction on how to properly pick this parameter.

One of Gal's examples mentions setting the prior length-scale to be 10^{-2} for most datasets based on the range of the data, but this choice is not feasible for predictive variance for survival probabilities. In this example, they use Bayesian optimization to find the optimal τ, which leads to their choice of λ, which seems to suggest that using an optimization strategy to directly choose τ is actually the preferable route, rather than using (7) to calculate it based on an imprecise choice of ℓ. The model precision itself is just an estimate of the inverse of the assumed observation noise (Gal 2015). The fact that little guidance is given on strategies to find the optimal τ makes using MC Dropout in practice much less straightforward than desired.

The choice of activation function for the NN hidden layers also impacts the behavior of the predictive uncertainty estimates. Activation functions that saturate, such as the sigmoid activation function used by default in FIFE, will behave differently than those that do not, such as a ReLU activation function. Gal notes some anecdotal evidence that use of the ReLU activation function increases the predictive uncertainty for estimates far from the original data, while the sigmoid function does not.

3. Usage in FIFE

The MC Dropout method for obtaining prediction intervals for forecasts is available as the 'compute_model_uncertainty()' method for an object of class 'TFModeler'. The function creates an ensemble of forecasts from independent neural network models (with dropout required between each layer), which are aggregated to form an asymptotic posterior distribution for the future values or classification probabilities of any observation of interest, for a number of future time points. The user can choose the number of independent models that will be used. As noted in Gal et al., while the quality of prediction intervals will be more robust with a larger number of models (50 or 100 for example), a good estimate of the amount of uncertainty can be generated with a smaller number of models. We would recommend using a minimum of 10 independent dropout models. In addition, the user can specify other hyperparameters for the model using the ‘params’ argument, a specific dropout rate, if different than previously specified in ‘params’, and the confidence level with which the prediction intervals will be computed (95% confident by default).

Estimated forecasts, and their corresponding prediction intervals, are plotted using the ‘plot_forecast_prediction_intervals()’ method for ‘TF_Modeler’, which requires the output of ‘compute_model_uncertainty()’. Individual observation forecasts, as well as aggregated sums of individuals are shown using this function. Note that prediction intervals for the sum of exit counts are for the expected value of counts, not the actual realized count number.

Additionally, unbalanced panel data can be generated for use with these methods using the ‘fabricate_data()’ function in ‘tests_performance/Data_Fabrication.py’ in FIFE, which now includes an argument, ‘covariates_affect_outcome’, which, if True, allows for individual covariates to have an affect on outcome probability, along with the survival probability for each person diminishing over time, following a Weibull distribution for the log hazard rate.

C. Stochastic Gradient Langevin Boosting

The use of ensembles of independent models is a valuable technique for quantifying predictive uncertainty in stochastic gradient boosted trees. One can use a model ensemble method to obtain probability distribution for a future outcome value. An ensemble of independent stochastic gradient boosted trees (with different seeds) can be used to obtain these distributions (and corresponding prediction intervals), but there are no guarantees as to how well this will approximate the true predictive distribution, usually leading to distorted intervals.

Malinin, Prokhorenkova, and Ustimenko (2021) introduces a method to approximate a Bayesian predictive posterior distribution by combining stochastic gradient boosting (SGB) with Langevin Dynamics[7] (Ustimenko and Prokhorenkova 2021). Stochastic Gradient Langevin Boosting (SGLB) uses a special form of the Langevin diffusion equation[8] specifically designed for gradient boosting. This enables sampling from the desired predictive distribution through the use of SGLB ensembles. Estimates of uncertainty through this method also are able to disentangle model (epistemic) uncertainty and data (aleatoric) uncertainty for the predictions.

Traditional Stochastic Gradient Boosting uses a random sub-sample of the training data, rather than the whole dataset, to update the sequential model and train the next tree. This technique introduces a small amount of noise into the model in order to prevent model overfitting. Given a dataset $D = \left\{\mathrm{x}^{(i)}, y^{(i)}\right\}_{i=1}^{N}$ and loss function L, with a chosen SGB learning rate ϵ, the model updates at each iteration t using the following formula:

[7] Langevin dynamics mathematically are commonly used to model the dynamics of molecular systems.

[8] A stochastic differential equation used in Langevin Dynamics (van Kampen 2007).

$$F^{(t)}(\mathbf{x}) = F^{(t-1)}(\mathbf{x}) + \epsilon h^{(t)}(\mathbf{x}),$$

where F^{t-1} is the previously constructed model at the last iteration, and $h^{(t)}(\mathrm{x})$ is the function that approximates the negative gradient $-g^{(t)}(\mathrm{x}, y) := -\left.\frac{dL(y,s)}{ds}\right|_{s=F^{(t-1)}(\mathrm{x})}$. The function $h^{(t)}(\mathrm{x}) \in H$ is chosen from some family of functions H, in order to minimize the expected difference between the negative gradient and the learner, as seen in the following:

$$h^{(t)} = \underset{h \in H}{\operatorname{argmin}}\, E_D\left[\left(-g^{(t)}(\mathbf{x}, y) - h\left(\mathbf{x}, \boldsymbol{\phi}^{(t)}\right)\right)^2\right], \tag{8}$$

where $\boldsymbol{\phi}^{(\mathbf{t})}$ are the parameters of the function $h^{(t)}$.

While analyzing an ensemble of independent SGB model predictions can provide some sense of the predictive uncertainty, there is no theoretical underpinning that leads to any kind of convergence towards a desired Bayesian posterior distribution.

Stochastic Gradient Langevin Boosting (SGLB) models make two changes to the traditional SGB modeling process. First, Gaussian noise is explicitly injected into the gradients that are used to train the model, allowing for proper exploration of the solution space to find the global optimum. Equation (8) is replaced by

$$h^{(t)} = \underset{h \in H}{\operatorname{argmin}}\, E_D\left[\left(-g^{(t)}(\mathbf{x}, y) - h(\mathbf{x}) + \nu\right)^2\right], \ \ \nu \sim \mathcal{N}\left(0, \frac{2}{\beta\epsilon} I_N\right), \tag{9}$$

for some choice of β. The Gaussian noise introduced by ν allows the SGLB model to properly sample the posterior distribution for our parameter space. Additionally, a regularization parameter, γ, is introduced into the model updating algorithm leading to the following formula:

$$F^{(t)}(\mathrm{x}) = (1 - \gamma\epsilon)F^{(t-1)}(\mathrm{x}) + \epsilon h^{(t)}\left(\mathrm{x}, \boldsymbol{\phi}^{(t)}\right). \tag{10}$$

Ustimenko and Prokhorenkova (2021) show that introducing γ as a regularizer in this fashion is equivalent to standard L2 regularization in the form of $L_N(F, \gamma) = L_N(F) + \frac{\gamma}{2}\left|\left|F\right|\right|_2^2$.

The SGLB model parameters $\boldsymbol{\theta}^{(t)}$ at each iteration form a Markov chain that weakly converges to the stationary distribution:

$$p_\beta^*(\boldsymbol{\theta}) \propto \exp(-\beta L(\boldsymbol{\theta}|D) - \beta\gamma||\Gamma\boldsymbol{\theta}||_2^2),$$

where Γ is an implicitly defined regularization matrix, based on the choice of tree algorithm (Malinin, Prokhorenkova, and Ustimenko 2021). In order to enable sampling from the true posterior distribution, Malinin sets $\beta = N$ and $\gamma = \frac{1}{2N}$ in order for the stationary distribution, using a negative log-likelihood loss function, to be proportional to the true posterior $p(\boldsymbol{\theta}|D)$:

$$p_\beta^*(\boldsymbol{\theta}) \propto \exp(\log p\left(D \left|\boldsymbol{\theta}\right) - \frac{1}{2}\right| |\Gamma\boldsymbol{\theta}||_2^2\Big) \propto p(D|\boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{11}$$

under Gaussian prior distribution $p(\boldsymbol{\theta}) = \mathcal{N}(0, \Gamma)$.

Predictions from an ensemble of independent SGLB models can be aggregated to form a predictive distribution for any random variables to be forecast in the future. Prediction intervals constructed in this fashion vastly outperform PIs made from normal SGB models.

1. Virtual SGLB

In certain cases, producing an ensemble of SGLB models can be computationally expensive, especially if the dataset is large. Malinin, Prokhorenkova, and Ustimenko (2021) proposes an alternative method that outputs prediction intervals at a much lower cost, in the form of a Virtual SGLB, which enables generating a virtual ensemble from only one single SGLB model instance. Due to the sequential nature of gradient boosted trees, a simple random sampling of trees isn't appropriate, since each tree is dependent upon the previous tree. Instead, a truncated version of an SGLB model, containing all trees from $F^{(1)}(\mathrm{x})$ up to some $F^{(k)}(\mathrm{x})$, can be used as one of many individual models in the virtual ensemble. For instance, if a single SGLB model contains 1000 trees, a virtual ensemble containing 10 sub-models can be obtained by the first sub-model having trees 1 through 500, the second sub-model having trees 1 through 550, and so on up to the last sub-model, containing trees 1 through 1000. Predictions from each of these sub-models can be used in the same way as in a normal SGLB ensemble to form predictive distributions. Given that there will be a high degree of correlation between the models in the virtual ensemble, predictive uncertainty estimates perform worse than when using a usual SGLB ensemble, but this technique still provides much better prediction intervals than any traditional PIs constructed from one single SGB or SGLB model. In a sense, a virtual SGLB provides a much higher quality set of PIs than a single model, without any added computational cost.

2. Differentiating Model and Data Uncertainty

There are a few steps to calculating the total uncertainty in predictions using SGLB ensembles, as well as differentiating that uncertainty into data uncertainty and model uncertainty. We define $\{P(y|\mathrm{x};\boldsymbol{\theta}^{(m)})\}_{m=1}^{M}$ as an ensemble of probabilistic models sampled from the posterior $p(\boldsymbol{\theta}|D)$, where $P(y|\mathrm{x},\boldsymbol{\theta}^{(m)})$ is a single model from that ensemble. Malinin notes that the predictive posterior distribution of the ensemble is obtained by taking the expectation with respect to the models in the ensemble:

$$P(y|\mathrm{x},D) = E_{p(\boldsymbol{\theta}|D)}[P(y|\mathrm{x},\boldsymbol{\theta})] \approx \frac{1}{M}\sum_{m=1}^{M} P\left(y|\mathrm{x};\boldsymbol{\theta}^{(m)}\right),\ \ \boldsymbol{\theta}^{(m)} \sim p(\boldsymbol{\theta}|D),$$

where M is the number of models in the ensemble.

The entropy of the predictive posterior distribution estimates the total uncertainty in predictions:

$$\mathcal{H}[P(y|\mathrm{x}, D)] = E_{P(y|\mathrm{x},D)}[-\log P(y|\mathrm{x}, D)]$$
$$\approx \frac{1}{N}\sum_{i=1}^{N} -\log\left[\frac{1}{M}\sum_{i=1}^{M} P\left(y|\mathrm{x};\boldsymbol{\theta}^{(m)}\right)\right].$$

This measure of total uncertainty includes both data and model uncertainty. In order to estimate data (aleatoric) uncertainty by itself, we obtain the expected entropy of each of the M models in the ensemble, $E\big[\mathcal{H}[P(y|\mathrm{x},\boldsymbol{\theta})]\big]$. By subtracting the expected data uncertainty from the total uncertainty, we can obtain $\mathcal{I}[y,\boldsymbol{\theta}|\mathrm{x}, D]$, the estimated model (epistemic) uncertainty:

$$\mathcal{I}[y,\boldsymbol{\theta}|\mathrm{x}, D] = \mathcal{H}[P(y|\mathrm{x}, D)] - E\big[\mathcal{H}[P(y|\mathrm{x};\boldsymbol{\theta})]\big]$$
$$\approx \mathcal{H}\left[\frac{1}{M}\sum_{m=1}^{M} P\left(y|\mathrm{x};\boldsymbol{\theta}^{(m)}\right)\right] - \frac{1}{M}\sum_{m=1}^{M} \mathcal{H}\left[P\big(y|\mathrm{x};\boldsymbol{\theta}^{(m)}\big)\right]$$
$$\approx \frac{1}{N}\sum_{i=1}^{N} -\log\left[\frac{1}{M}\sum_{i=1}^{M} P\left(y_i|\mathrm{x}_i;\boldsymbol{\theta}^{(m)}\right)\right]$$
$$-\frac{1}{M}\sum_{i=1}^{M}\left[\frac{1}{N}\sum_{i=1}^{N} -\log\big[P\big(y_i|\mathrm{x}_i;\boldsymbol{\theta}^{(m)}\big)\big]\right].$$

Essentially, model uncertainty is being measured as the level of spread or disagreement with respect to the models in the ensemble, whereas data uncertainty is being measured as the level of spread within each model.

3. Usage in FIFE

The Stochastic Gradient Langevin Boosthing method for obtaining prediction intervals for gradient boosting forecasts is available as the 'compute_model_uncertainty()' method for an object of class 'LGBModeler'. The function creates an ensemble of forecasts from independent gradient boosting models, which are aggregated to form an asymptotic posterior distribution for the future values or classification probabilities of any observation of interest, for a number of future time horizons. The user can choose the number of independent models that will be used. Similar to the neural network prediction intervals, while the quality of prediction intervals will be more robust with a larger number of models (50 or 100 for example), a good estimate of the amount of uncertainty can be generated with a smaller number of models. We would recommend using a minimum of 10 independent dropout models.

In addition, the user can specify other hyperparameters for the model using the 'params' argument, like the confidence level with which the prediction intervals will be computed (95% confident by default). The 'langevin_variance' parameter of 'compute_model_uncertainty()' is the choice of σ^2 in our adaptation of Equation (9), written as follows:

$$h^{(t)} = \operatorname*{argmin}_{h\in H} E_D\left[\left(-g^{(t)}(\mathbf{x}, y) - h(\mathbf{x}) + \nu\right)^2\right],\ \ \nu \sim \mathcal{N}(0, \sigma^2 I_N), \qquad (12)$$

We set $\sigma^2 = \frac{2}{\beta\epsilon}$ to adjust the variance of the Gaussian noise inserted into the gradient as a whole. By default, σ^2 is set to zero in 'compute_model_uncertainty()', although this is not recommended. Without using a proper strategy to tune this parameter, resulting prediction intervals will be distorted, especially if $\sigma^2 = 0$. We talk about strategies for choosing this parameter in section 2.4.

Estimated forecasts, and their corresponding prediction intervals, are plotted using the 'plot_forecast_prediction_intervals()' method for 'LGB_Modeler', which requires the output of 'compute_model_uncertainty()'. Individual observation forecasts, as well as aggregated sums of individuals are shown using this function. Note that prediction intervals for the sum of exit counts are for the expected value of counts, $E[\sum_{i=1}^{N} y_i]$, not the actual realized count number, $[\sum_{i=1}^{N} y_i]$, where y_i is 1 if individual i survives, and 0 if individual i exits.

In order to allow for noise to be injected into the gradients for FIFE LightGBM models, we use a custom objective function, which adds noise with a defined variance to the original gradients based on the binary log loss function used in classification for binary outcomes. When defining custom objective functions in LightGBM, outputs are given in logit form, so a sigmoid function is needed to transform the values into the correct survival probabilities. Currently, the resulting predictions using our custom loss function[9] (with no added Gaussian noise) are close to predictions using the default loss function but not identical. As a temporary fix, we have adjusted all forecasts using the custom loss function up or down based on the difference between mean forecasts from the default function and mean forecasts using the custom function.

D. Restrictions for MC Dropout and Stochastic Gradient Langevin Boosting

There is a notable restriction that prevents us from using the MC Dropout and SGLB methods right out-of-the-box: both require specific choices of regularization parameters, such as λ in Equation (7) and γ in Equation (10). Adding or changing these hyperparameters for our neural network or gradient boosting models would not be advantageous, because any previous hyperoptimization used to fine-tune these parameters would be undone, and the predictions themselves would lose accuracy.

In the case of MC Dropout, any choice of L2 regularization parameter may be used, but this type of regularization has to exist in the dropout loss function, and no other type of regularization seems to be allowed. Additionally, in order to calculate the model precision τ, Equation (7) requires an arbitrary choice of prior length-scale ℓ, which is common for Gaussian process models. However, little advice is given in Gal for choosing this, and even minor tweaks to this parameter can dramatically alter the resulting model precision τ. The paper even suggests to optimize the choice of τ, rather than calculating it directly.

[9] The custom loss function is called 'bce_loss()' within FIFE.

For Stochastic Gradient Langevin Boosting, Malinin, Prokhorenkova, and Ustimenko (2021) sets L2 regularization parameter $\gamma = \frac{1}{2N}$, in order to derive Equation (11). It does not seem that any other regularization parameters are allowed in the gradient boosted tree model.

MC Dropout and Stochastic Gradient Langevin Boosting both provide a similar process to quantifying forecast uncertainty. First, they get the base predictive variance from ensembles of predictions. Next, they employ some additional step to inflate that variance to account for all of the uncertainty. We want to use this process to compute prediction intervals for our models, without needing to meet the specific constraints of MC Dropout and SGLB by changing the original model structure.

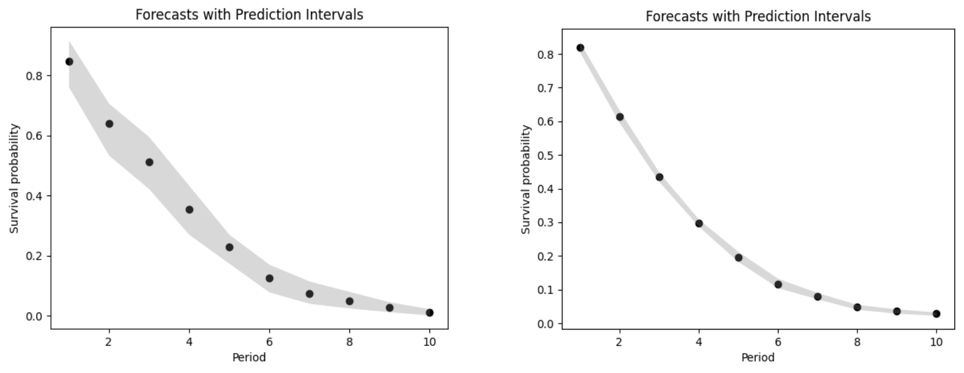
Our solution is to design a strategy to optimize the inflated variance parameters for each of these methods in order to produce resulting prediction intervals that have the desired out-of-sample coverage probability. For neural network models, we can tune the choice of model precision τ to accomplish this. A single ensemble of NN models can be trained, resulting in a predictive variance for each random variable to be predicted, and then the choice of τ can be adjusted such that the resulting variance added to the original prediction intervals meets the desired width. We propose a method for tuning τ later.

For gradient boosted tree models, we will use Equation (12), so that we can directly optimize the choice of variance, σ^2, in order to achieve out-of-sample nominal coverage probability. Unlike our neural network approach, a new ensemble of predictions must be constructed each time we choose a new value for σ^2.

1. Parameter Tuning for Simulated Data

In order to test a possible strategy for tuning the variance inflation parameters for both the neural network and gradient boosting models, we simulated panel data using FIFE's 'fabricate_data()' function, inside 'tests_performance/Data_Fabrication.py'. We simulated a dataset of 2000 individuals, having 50 total time periods, with each person having around a 40% chance of exiting at any time point on average. However, by setting the argument 'covariates_affect_outcome=True', each observation's probability of exit depends on the individual covariates, as well as the length of time periods the observation has been in the simulation, with the probability of exit increasing over time, using a Weibull distribution for the log hazard rate of survival. We used the first two-thirds of the dataset (periods 1 through 33) as the training data, and reserved the final two-thirds as the test set, used to calculate out-of-sample PI coverage probability.

After hyperoptimizing parameters for both the neural network and gradient boosting modelers, we constructed an ensemble of 50 independent predictions for both, using the 'compute_model_uncertainty()' function, without adding any variance inflation. Figure 4 illustrates an example of the predicted survival probability for an observation in the test dataset, for both modelers. Each point shows the predicted survival probability at a future time horizon, and the edges of the gray area are the lower and upper bounds for the 95% prediction intervals constructed using the ensemble predictions.



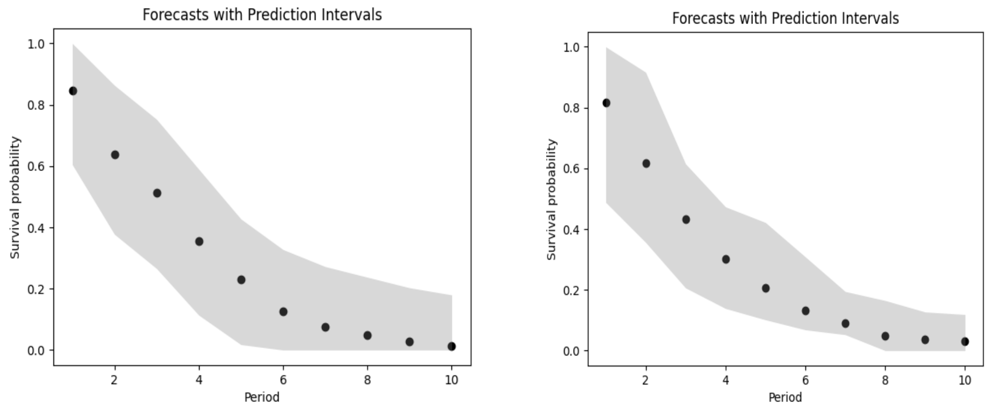
Note: These plots show the predicted survival probability, with corresponding naive 95% prediction intervals, for an example observation in the simulated dataset. These naive 95% PIs are distorted; that is, they claim to have 95% coverage probability, but their actual coverage probabilities are much lower: 48% for the intervals in the left panel and 2% for the intervals in the right panel.

Figure 4. Predicted Survival Probability with Naive Bounds

While these prediction intervals seem very informative, their actual out-of-sample coverage probability is terrible: 48% for the neural network, and 2% for the gradient boosted tree. The predictions from the ensembles themselves do not have enough variance to account for the amount of uncertainty needed.

To widen the prediction intervals for the neural network, we will optimize our choice of model precision τ. The inverse, $1/\tau$, is the variance that is added to the existing variance of the prediction intervals, in order to form wider PIs. We can choose the value of τ such that the new prediction intervals have an out-of-sample coverage probability of exactly 95%, averaged across all future time horizons.

For the gradient boosted tree, we can choose an optimal value of σ^2, the variance of the Gaussian noise injected into the gradient, such that the resulting PIs have at least nominal coverage. Each time we choose a new value of σ^2, a new ensemble of predictions must be computed using this choice, until we find the value that leads to around 95% out-of-sample coverage probability, averaged across all time horizons. We can't get a choice of σ^2 that leads to the exact nominal coverage, because the predictions themselves will change each time due to the stochastic nature of the gradient boosting. Figure 5 shows the new prediction intervals that result from tuning these parameters to achieve 95% coverage, averaged across all time horizons.



Note: These plots show the predicted survival probability, with corresponding 95% prediction intervals, for an example observation in the simulated dataset after optimizing the choice of variance inflation parameters to achieve nominal coverage.

Figure 5. Predicted Survival Probability with Proposed Bounds

2. Parameter Tuning for Real Data

Calculating out-of-sample PI coverage probability for real data with a continuous output would be simple, but given that our outputs are categorical, we cannot use the same strategy used in our simulation study to tune the variance inflation parameters, because we only observe the actual outcomes in real data rather than the underlying categorical probabilities. There are several different potential solutions to this that we have considered.

One option is to calculate an empirical survival probability for each observation in the test set by grouping observations with similar covariates into clusters, and then using the actual survival proportion for a cluster at a time point as the empirical survival probability. These empirical probabilities would then be used to assess out-of-sample PI coverage. Some possible clustering methods include K-Means Clustering, and K-Nearest-Neighbors, the latter being implemented in Huang and Perry (2016), which could be of use here. The biggest issue with this approach is that datasets with categorical variables are difficult to use in these clustering approaches. While Preud'homme et al. (2021) compare the computational feasibility of these approaches with large datasets and high numbers of categorical variables is uncertain. Another empirical option is to simply bin observations in the test set together that have similar predicted survival probabilities at the same time horizon. The observed survival proportion for all observations in each bin can be used as the empirical survival probability, which is used to assess out-of-sample PICP. This may not be a reasonable approach since this method only groups observations together based on the predicted output, without consideration of any of the covariates.

The approaches above group observations together to determine the mean empirical frequency conditional on the groupings. An alternative approach involves modeling the mean conditional on covariates *in-sample* on the test set. In principle, we could train the same FIFE modeler on the test data to predict the mean of the empirical frequencies conditional on covariates in the test set. These survival probabilities can be treated as the future unobserved probabilities and then used to compute the out-of-sample PICP for the intervals originally constructed.

This page is intentionally blank

.

3. Discussion

Many of the best existing methods for constructing prediction intervals for forecasts for neural networks and gradient boosted trees use ensembles of independent models to form approximate posterior predictive distributions for forecasted random variables. Both MC Dropout for neural networks (Gal and Ghahramani 2016), and Stochastic Gradient Langevin Boosting (Malinin, Prokhorenkova, and Ustimenko 2021) take additional measures to add to the existing predictive variance to account for more predictive uncertainty and generate PIs that should have nominal coverage probability. MC Dropout uses a global model precision parameter τ to add variance to the existing ensemble PIs. SGLB adds Gaussian noise into the gradients to randomize the parameter movement and mimic sampling from the predictive posterior.

However, both of these methods require specific choices of model tuning parameters, such as L2 regularization, in order for the predictive distributions to approximate a Bayesian posterior. We want to use these methods of variance inflation without changing any of the existing model hyperparameters, such as choice of L2 regularization, that have already been optimized to achieve the best predictive performance.

Our proposed solution is to tune the parameters that control the amount of variance inflation by monitoring out-of-sample prediction interval coverage probability, until we achieve nominal coverage. This is simple enough for simulated data where we know the underlying survival probabilities used to generate the observed outcomes, but this is more challenging when dealing with real data where the probabilities are not visible. We have proposed several possible strategies for computing empirical survival probabilities for the out-of-sample data, so that we can still calculate the PI coverage probabilities for our test data and tune the variance inflation parameters accordingly.

A. Future Work

The main objective for possible future work is to implement a method for tuning the variance inflation parameters used in MC Dropout and Stochastic Gradient Langevin Boosting when using real classification data. Currently, both PI methods are implemented in FIFE, but they will likely output distorted intervals because variance inflation parameters are not tuned by default. Additionally, it may be worth tuning these parameters specifically to each time horizon, since the amount of uncertainty in forecasts will vary across time.

We are aware of additional intriguing prediction interval methods for both neural networks and gradient boosting that we did not have time to explore in depth, such as Lower-Upper Bound

Estimation for NNs (Khosravi et al. 2011). This method trains a neural network to output the highest quality prediction intervals, instead of single point estimates. More detail on this approach can be found in Appendix B.

Finally, an additional approach to capturing model uncertainty that we did not discuss involves quantile regression (Koenker 2005). Forecasting the lower $\alpha/2$th and corresponding upper $1 - \alpha/2$th quantiles of the distribution provides a natural method for capturing uncertainty in a form that is analogous to a prediction interval. Future work can make use of analytical developments in quantile regression for survival data, binary responses, and non-parametric quantile regression. While Koenker (2005) provides a textbook level treatment of these topics, recent work illuminates more recent developments (Koenker et al. 2017).

Appendix A. Additional Detail for MC Dropout

Typically, a deep neural network with dropout has a minimization objective function, such as a squared loss, including some L_2 regularization of the model weights. We will denote $\hat{\mathrm{y}}$ as the output of a NN model with L layers, N observations, and loss function $E(\mathrm{y}_i, \hat{\mathrm{y}}_i)$. The dropout minimization objective function often takes the following form:

$$L_{dropout} = \frac{1}{N}\sum_{i=1}^{N} E\left(\mathrm{y}_i, \hat{\mathrm{y}}_i\right) + \lambda \sum_{i=1}^{L}\left(\left||\mathrm{W}_i|\right|_2^2 + \left||\mathrm{b}_i|\right|_2^2\right), \qquad (13)$$

where W_i is the NN's matrix of weights for a given layer i, and b_i is the bias vector for that layer.

Gal shows that the dropout objective minimizes the Kullback-Leibler Divergence (KL Divergence) between an approximate distribution and the posterior distribution of a deep Gaussian process, which will enable sampling from the approximate posterior predictive distribution using an ensemble of NN model predictions.

Let $\boldsymbol{\omega}$ be the set of finite rank covariance function parameters for the deep GP model. Given input and output sets X and Y, the predictive probability of the model can be written as

$$p(\mathrm{y}|\mathrm{x},\mathrm{X},\mathrm{Y}) = \int p(\mathrm{y}|\mathrm{x},\boldsymbol{\omega})p(\boldsymbol{\omega}|\mathrm{X},\mathrm{Y})d\boldsymbol{\omega},$$

where $p(\mathrm{y}|\mathrm{x},\boldsymbol{\omega}) \sim \mathcal{N}(\hat{\mathrm{y}}(\mathrm{x},\boldsymbol{\omega}), \tau^{-1}\mathrm{I}_D)$ and τ is the model precision parameter. Though the posterior distribution $p(\boldsymbol{\omega}|\mathrm{X},\mathrm{Y})$ is intractable, an approximate distribution $q(\boldsymbol{\omega})$ is constructed as a distribution over matrices whose columns are randomly set to zero, just like how dropout is applied in the deep NN model. Given some probabilities p_i and matrices M_i, Gal defines $q(\boldsymbol{\omega})$ as:

$$\mathrm{W}_i = \mathrm{M}_i \cdot \mathrm{diag}\left(\left[\mathrm{z}_{i,j}\right]_{j=1}^{K_i}\right)$$
$$\mathrm{z}_{i,j} \sim \mathrm{Bernoulli}(p_i), \text{for } i = 1, \ldots, L, j = 1, \ldots, K_{i-1}.$$

When $\mathrm{z}_{i,j} = 0$, this indicates that unit j in layer $i - 1$ is being dropped out as an input to layer i.

The Gaussian process objective function in this particular case takes the following form:

$$L_{GP-MC} = \frac{1}{N}\sum_{i=1}^{N} \frac{-\log p\left(\mathrm{y}_n|\mathrm{x}_n, \widehat{\boldsymbol{\omega}}_n\right)}{\tau} + \sum_{i=1}^{L}\left(\frac{p_i\ell^2}{2\tau N}\left||\mathrm{M}_i|\right|_2^2 + \frac{\ell^2}{2\tau N}\left||\mathrm{m}_i|\right|_2^2\right),$$

where $\widehat{\boldsymbol{\omega}}_n$ is a matrix of Bernoulli random variables from a single sample, meant to approximate the dropout matrix, τ is the model precision parameter, p_i is 1 minus the dropout rate, and ℓ is some prior length-scale chosen based on the frequency of the data (more on this later). By setting the NN loss function to be $E\left(\mathrm{y}_n, \hat{\mathrm{y}}(\mathrm{x}_n, \widehat{\boldsymbol{\omega}}_n)\right) = -\log p\left(\mathrm{y}_n | \mathrm{x}_n, \widehat{\boldsymbol{\omega}}_n\right)/\tau$, the GP minimization function L_{GP-MC} matches the original dropout minimization function $L_{dropout}$ in (13). Gal states, "The sampled $\widehat{\omega}_n$ result in realizations from the Bernoulli distribution $z_{i,j}^n$, equivalent to binary variables in the dropout case" (Gal and Ghahramani 2016).

Appendix B. Lower-Upper Bound Estimation for Deep Neural Network Predictions

Lower-Upper Bound Estimation (LUBE) is a technique made for constructing and training deep neural networks to produce the highest quality prediction intervals for future predictions (Khosravi et al. 2011). While we did not have time to explore this method for use in FIFE, we think it could be a useful alternative to constructive high quality prediction intervals.

Traditionally, prediction uncertainty quantification methods are applied after a neural network model is trained and has produced point estimates for predicted quantities of interest. While point estimates produced this way might be accurate, the resulting intervals around those points are often either too conservative or do not reach nominal coverage in practice. Khosravi recommends training neural networks with the goal of obtaining the best prediction intervals, instead of obtaining the best point estimates, if obtaining high quality prediction intervals is the end goal. In the LUBE method, neural network models are trained with a loss function specifically constructed to obtain prediction intervals with nominal coverage and smaller interval width, through the use of a prediction interval quality score.

Khosravi's neural network development procedure directly addresses all the desired characteristics of prediction intervals, beyond just coverage probability, which is only one part of a high quality interval. A prediction interval that is built only to have maximum coverage can be unreasonably wide, if it is not trained to simultaneously have more precise ranges. The proposed prediction interval quality score addresses all of these properties in measuring how good an interval is. Model training proceeds by improving the quality of the prediction intervals. The method leads to more accurate predictions by making narrower prediction intervals through training, while maintaining nominal coverage.

While we have not been able to implement the LUBE method in FIFE, we recommend further investigation of this method for use in FIFE.

This page is intentionally blank.

Appendix C. Illustrations

List of Figures

This page is intentionally blank.

Appendix D. References

Chatfield, Chris. 1993. “Calculating Interval Forecasts.” *Journal of Business & Economic Statistics* 11 (2): 121–35. https://doi.org/10.2307/1391361.

———. 1995. “Model Uncertainty, Data Mining and Statistical Inference.” *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 158 (3). https://rss.onlinelibrary.wiley.com/doi/abs/10.2307/2983440.

Duan, Tony, Avati Anand, Daisy Yi Ding, Khanh K. Thai, Sanjay Basu, Andrew Ng, and Alejandro Schuler. 2020. “NGBoost: Natural Gradient Boosting for Probabilistic Prediction.” In *Proceedings of the 37th International Conference on Machine Learning*, edited by Hal Daumé III and Aarti Singh, 119:2690–700. Proceedings of Machine Learning Research. PMLR. http://proceedings.mlr.press/v119/duan20a.html.

Gal, Yarin. 2015. “What My Deep Model Doesn’t Know... Yarin Gal - Blog Cambridge Machine Learning Group.” http://mlg.eng.cam.ac.uk/yarin/blog_3d801aa532c1ce.html.

Gal, Yarin, and Zoubin Ghahramani. 2016. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.” In *International Conference on Machine Learning*, 1050–59. PMLR. http://proceedings.mlr.press/v48/gal16.html.

Goemans, Michel. 2015. “Chernoff Bounds, and Some Applications.” *MIT 18.310 Lecture Notes*. https://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf.

Huang, Jing, and Matthew Perry. 2016. “A Semi-Empirical Approach Using Gradient Boosting and k-Nearest Neighbors Regression for GEFCom2014 Probabilistic Solar Power Forecasting.” *International Journal of Forecasting* 32 (3): 1081–86. https://doi.org/10.1016/j.ijforecast.2015.11.002.

Hüllermeier, Eyke, and Willem Waegeman. 2021. “Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods.” *Machine Learning* 110 (3): 457–506. https://doi.org/10.1007/s10994-021-05946-3.

Hwang, J. T. Gene, and A. Adam Ding. 1997. “Prediction Intervals for Artificial Neural Networks.” *Journal of the American Statistical Association* 92 (438): 748–57. https://doi.org/10.1080/01621459.1997.10474027.

Hyndman, Rob. 2013. “The Difference Between Prediction Intervals and Confidence Intervals.” Accessed July 12, 2021. https://robjhyndman.com/hyndsight/intervals/.

Institute for Defense Analyses. 2021. "FIFE: Finite-Interval Forecasting Engine [software]." https://github.com/IDA-HumanCapital/fife. Version 1.4.

Khosravi, Abbas, Saeid Nahavandi, Doug Creighton, and Amir F. Atiya. 2011. "Lower Upper Bound Estimation Method for Construction of Neural Network-Based Prediction Intervals." *IEEE Transactions on Neural Networks* 22 (3): 337–46. https://doi.org/10.1109/TNN.2010.2096824.

Koenker, R., V. Chernozhukov, X. He, and L. Peng. 2017. *Handbook of Quantile Regression*. Edited by R. Koenker, V. Chernozhukov, X. He, and L. Peng. Chapman and Hall/CRC. https://doi.org/10.1201/9781315120256.

Koenker, Roger. 2005. *Quantile Regression*. Econometric Society Monographs. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511754098.

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. 2017. "Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles." *arXiv:1612.01474 [Cs, Stat]*. http://arxiv.org/abs/1612.01474.

Malinin, Andrey, Liudmila Prokhorenkova, and Aleksei Ustimenko. 2021. "Uncertainty in Gradient Boosting via Ensembles." *arXiv:2006.10562 [Cs, Stat]*. http://arxiv.org/abs/2006.10562.

Preud'homme, Gregoire, Kevin Duarte, Kevin Dalleau, Claire Lacomblez, Emmanuel Bresso, Malika Smaïl-Tabbone, Miguel Couceiro, et al. 2021. "Head-to-Head Comparison of Clustering Methods for Heterogeneous Data: A Simulation-Driven Benchmark." *Scientific Reports* 11 (1): 4202. https://doi.org/10.1038/s41598-021-83340-8.

Tagasovska, Natasa, and David Lopez-Paz. 2019. "Single-Model Uncertainties for Deep Learning." https://arxiv.org/abs/1811.00908v3.

Ustimenko, Aleksei, and Liudmila Prokhorenkova. 2021. "SGLB: Stochastic Gradient Langevin Boosting." *arXiv:2001.07248 [Cs, Stat]*. http://arxiv.org/abs/2001.07248.

van Kampen, N. G. 2007. "Chapter IX - The Langevin Approach." In *Stochastic Processes in Physics and Chemistry,* Third Edition, edited by N. G. van Kampen, 219–43. North-Holland Personal Library. Amsterdam: Elsevier. https://doi.org/10.1016/B978-044452965-7/50012-X.

Wang, Hsiuying. 2008. "Coverage Probability of Prediction Intervals for Discrete Random Variables." *Computational Statistics & Data Analysis* 53 (1): 17–26. https://doi.org/10.1016/j.csda.2008.07.017.

Appendix E. Abbreviations

Term	Definition
CI	Confidence Interval
FIFE	Finite Interval Forecasting Engine
GBM	Gradient Boosted Machine
IDA	Institute for Defense Analyses
KNN	K-Nearest Neighbors
LUBE	Lower-Upper Bound Estimation
NN	Neural Network
PI	Prediction Interval

This page is intentionally blank.