Beta Survival Models

David Hubbard DHUBBARD@NETFLIX.COM
Benoit Rostykus BROSTYKUS@NETFLIX.COM
Yves Raimond YRAIMOND@NETFLIX.COM
Tony Jebara TONYJ@SPOTIFY.COM

Abstract

Survival modeling is an important area of study, and has been used widely in many applications including clinical research, online advertising, manufacturing, etc. There are many methods to consider when analyzing survival problems, however these techniques generally focus on either estimating the uncertainty of different risk factors (cox-proportional hazards, etc), or predicting the time to event in a non-parametric way (e.g. tree based methods), or forecasting the survival beyond an observed horizon (parametric techniques such as exponential). In this work, we introduce efficient estimation methods for linear, tree, and neural network versions of the Beta-Logistic model - a classical extension of the logistic function into the discrete survival setting. The Beta-Logistic allows for recovery of the underlying beta distribution as well as having the advantages of non-linear or tree based techniques while still allowing for projecting beyond an observed horizon. Empirical results using simulated data as well as large-scale data-sets across three use-cases (online conversions, retention modeling in a subscription service, and survival of democracies and dictatorships), demonstrate the competitiveness of the method at these tasks. The simplicity of the method and its ability to capture skew in the data makes it a viable alternative to standard techniques particularly when we are interested in forecasting time to event beyond our observed horizon and when the underlying probabilities are heterogeneous.

1. Introduction

Survival modeling, customer lifetime value estimation (Gupta et al., 2006) and product ranking (Rudin et al., 2012; Chang et al., 2012) are of practical interest when we want to estimate time until a specific event occurs or rank items to estimate which will encounter the event first. Traditionally leveraged in medical applications, today survival regression is extensively used in large-scale business settings such as predicting time to conversion in online advertising and predicting retention (or churn) in subscription services. Standard survival regression involves a maximum likelihood estimation problem over a specified continuous distribution of the time until event (exponential for the Accelerated Failure Time model Kalbfleisch and Prentice (2011)) or of the hazard function (in the case of Cox Proportional Hazards Cox (1972)). In practice, time to event problems are often converted to classification problems by choosing a fixed time horizon which is appropriate for the application at hand. One then has to balance training the model on recent data against a longer labeling horizon which might be more desirable. Survival models avoid this trade-off by relying on right-censoring. This maps to a missing data problem where not all events are observed due to the horizon over which the data is collected.

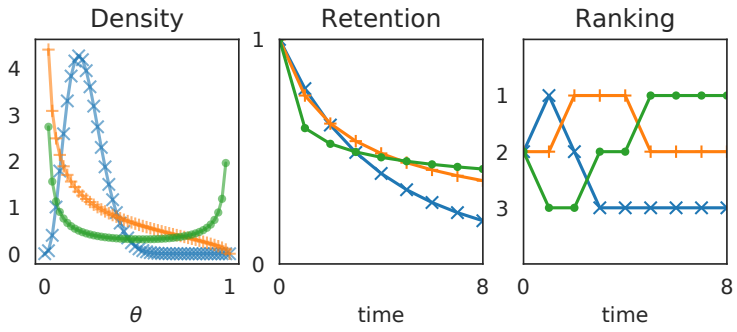

Figure 1: Heterogeneity gives rise to different survival distributions and rankings.

There is evidence (which will be further demonstrated in this article) of the importance of heterogeneity in a variety of real-world time to event data-sets. Heterogeneity indicates that items in a data-set have different survival means and variances. For instance, heterogeneity in a churn modeling context would be that as time increases the customers with the highest probability to retain are the ones which still remain in the data-set. Without considering this effect, it might appear that the baseline churn probability has decreased over time when in fact the first order effect is that there is a mover/stayer bias. Thus methods which don't consider multiple decision points can fail to adequately account for this effect and thus fall victim to the so called ruse of heterogeneity (W. Vaupel and Yashin, 1985).

Consider the following example inspired by Ben-Porath (1973) where we have 2 groups of customers, one in which the customers have a retention probability of 0.5 and in the other the customers are uniformly split between retention probabilities of either 1.0 or 0.0. In this case after having observed only one decision point we would find the retention probabilities of the two groups to be identical. However, if we consider multiple decision points it becomes clear that the latter population has a much higher long term retention rate because some customers therein retain to infinity. In order to capture this unobserved heterogeneity we need a distribution that is flexible enough to capture these dynamics and ideally as simple as possible. To that end we posit a beta prior on our instantaneous event probabilities. The beta has only 2 parameters, yet is flexible enough to capture right/left skewed, U-shaped, or normal distributions. In Figure 1 we have an example data-set that contains three heterogeneous items (green dots, orange plus and blue cross). These items can each be characterized by beta distributions (left panel). At each time period, each item samples a Bernoulli distributed coin from its beta distribution and flips it to determine if the item will retain. In the middle panel, we see the retention of the items over time and in the right-most panel we see the ranking of the items over time. Even though the items are sampling from fixed beta distributions, the ranking of which item is most at risk over time changes. Thus, a stationary set of beta distributions leads to non-stationary survival and rankings. Such nuance cannot be captured by summarizing each item with only a point-estimate of survival (as opposed to a 2-parameter beta distribution).

Due to the discrete and repeat nature of the decision problems over time, we leverage a geometric hypothesis to recover survival distributions. We estimate the parameters of this

model via an empirical Bayes method which can be efficiently implemented through the use of analytical solutions to the underlying integrals. This model termed the beta-logistic was first introduced by Heckman and Willis (1977), and was also studied by Fader and Hardie (2007). We find that in practice this model fits the discrete decision data quite well, and that it allows for accurate projections of future decision points.

We extend the beta-logistic model to the case of large-scale trees or neural-network models that adjust the beta distribution given input covariates. These leverage the use of recurrence relationships to efficiently compute the gradient. Through the beta prior underpinning the model, we show empirically that the beta-logistic is able to model a wide range of heterogeneous behaviors that other survival or binary classification models fail to capture, especially in the presence of skew. As we will see, the beta-logistic model outperforms a typical binary logistic regression in real-world examples, and provides tighter estimated posteriors compared to a typical Laplace approximation.

This paper is organized as follows. We first show the beta-logistic derivation as well as reference the recursion formulas which make the computation efficient. We also make brief remarks about convexity and observe that in practice we rarely encounter convergence issues. We then present simulated examples to help motivate the discussion. This is followed by an empirical performance evaluation of the various models across three large real-world data-sets: a sparse online conversion data-set, a retention modeling data-set from a popular subscription video service, and data on democracies and dictatorships. In many of the examples, the beta-logistic outperforms other baseline methods.

2. The Beta-Logistic for Survival Regression

2.1. Model derivation

Denote by $(x_i, t_i, c_i) \in \mathbb{R}^d \times \mathbb{N} \times \{0, 1\}$ a data-set where x_i are covariates, t_i is the discrete time to event for an observed (i.e. uncensored) datapoint ($c_i = 0$) and t_i is the right-censoring time for a datapoint for which the event of interest hasn't happened yet ($c_i = 1$). A traditional survival model would posit a parametric distribution $p(T|x)$ and then try to maximize the following empirical likelihood over a class of functions f:

$$L = \prod_{\forall i, c_i=0} \mathbb{P}\left(T = t_i | f(x_i)\right) \prod_{\forall i, c_i=1} \mathbb{P}\left(T > t_i | f(x_i)\right).$$

Unfortunately, unless constrained to a few popular distributions such as the exponential one, the maximization of such a quantity is usually intractable for most classes of functions $f(x)$.

Let us instead assume that at each discrete decision point, a customer decides to retain with some (point-estimate) probability $1 - \theta$ where θ is some function of the covariates x. Then we further assume that the instantaneous event probability at decision point t is characterized by a shifted geometric distribution as follows:

$$\mathbb{P}(T = t|\theta) = \theta(1-\theta)^{t-1}, \ \textit{where } \theta \in [0, 1].$$

This then gives the following survival equation:

$$\mathbb{P}(T > t|\theta) = 1 - \sum_{i=1}^{t} \mathbb{P}(T = i|\theta). \tag{1}$$

This geometric assumption follows from the discrete nature of the decisions customers need to make in a subscription service, when continuing to next episodes of a show, etc. It admits a simple and straightforward survival estimate that we can also use to project beyond our observed time horizon. Now in order to capture the heterogeneity in the data, we can instead assume that θ follows a conditional beta prior ($\mathbb{B}$) as opposed to being a point-estimate as follows:

$$f(\theta|\alpha(x),\beta(x)) = \frac{\theta^{\alpha(x)-1}(1-\theta)^{\beta(x)-1}}{B(\alpha(x),\beta(x))}$$

where $\alpha(x)$ and $\beta(x)$ are some arbitrary positive functions of covariates (e.g. measurements that characterize a specific customer or a specific item within the population).

Consider the Empirical Bayes method of Gelman et al. (2013) (also called Type-II Maximum Likelihood Estimation in Berger (2013)) as an estimator for $\alpha(x)$ and $\beta(x)$ (α and β for brevity) given the data:

$$\max_{\alpha,\beta} L(\alpha,\beta)$$

where

$$L(\alpha,\beta) = \prod_{\forall i, c_i=0} \mathbb{P}\left(T = t_i|\alpha,\beta\right) \prod_{\forall i, c_i=1} \mathbb{P}\left(T > t_i|\alpha,\beta\right). \tag{2}$$

Using the marginal likelihood function we obtain:

$$\mathbb{P}(T|\alpha,\beta) = \int_0^1 f(\theta|\alpha,\beta)\mathbb{P}(T|\theta)d\theta.$$

As we will see in the next section, a key property of the beta-logistic model is that it makes the maximization of Equation 2 tractable.

Since α and β have to be positive to define valid beta-distributions, we use an exponential reparameterization and aim to estimate functions $a(x)$ and $b(x)$ such that:

$$\alpha = e^{a(x)} \text{ and } \beta = e^{b(x)}.$$

Throughout the paper, we will also assume that a and b are twice-differentiable.

The name **beta-logistic** for such a model has been coined by Heckman and Willis (1977) and studied when the predictors $a(x) = \gamma_a \cdot x$ and $b(x) = \gamma_b \cdot x$ are linear functions. In this case, at $T = 1$ observe that if we want to estimate the mean this reduces to an over-parameterized logistic regression:

$$\mathbb{P}(T = 1|\alpha,\beta) = \frac{\alpha}{\alpha+\beta} = \frac{1}{1+e^{(\gamma_b-\gamma_a)^\top x}}.$$

2.2. Algorithm

We will now consider the general case where $a(x)$ and $b(x)$ are nonlinear functions and could be defined by the last layer of a neural network. Alternatively, they could be generated by a vectored-output Gradient Boosted Regression Tree (GBRT) model. Using properties of the beta function (see A.1), one can show that:

$$\mathbb{P}(T = 1|\alpha,\beta) = \frac{\alpha}{\alpha+\beta}$$

and

$$\mathbb{P}(T > 1|\alpha, \beta) = \frac{\beta}{\alpha + \beta}.$$

Further, the following recursion formulas hold for $t > 1$:

$$\mathbb{P}(T = t|\alpha, \beta) = \left(\frac{\beta + t - 2}{\alpha + \beta + t - 1}\right) \mathbb{P}(T = t - 1|\alpha, \beta) \tag{3}$$

and

$$\mathbb{P}(T > t|\alpha, \beta) = \left(\frac{\beta + t - 1}{\alpha + \beta + t - 1}\right) \mathbb{P}(T > t - 1|\alpha, \beta). \tag{4}$$

If we denote $\ell = -\log L$ as the function we wish to minimize, Equation 3 and Equation 4 allow us to derive (see Appendix A.2) recurrence relationships for individual terms of $\frac{\partial \ell}{\partial \cdot}$ and $\frac{\partial^2 \ell}{\partial \cdot^2}$. This makes it possible for example to implement a custom loss gradient and Hessian callbacks in popular GBRT libraries such as XGBoost (Chen and Guestrin, 2016) and lightGBM (Ke et al., 2017). In this case, the GBRT models have "vector-output" and predict for every row both $a = \log(\alpha)$ and $b = \log(\beta)$ jointly from a set of covariates, similarly to how the multinomial logit loss is implemented in these libraries. More precisely, choosing a squared loss for the split criterion in each node as is customary, the model will equally weight how well the boosting residuals (gradients) with respect to a and b are regressed on.

Note that because of the inherent discrete nature of the beta-logistic model, the computational complexity of evaluating its gradient on a given datapoint is proportional to the average value of t. Therefore, a reasonable time step discretization value needs to be chosen to properly capture the survival dynamics while allowing fast inference. One can similarly implement this loss in deep learning frameworks. One would typically explicitly pad the label vectors t_i with zeros up to the max censoring horizon (which would bring average computation per row to $O(\max_i t_i)$ for the mini-batch) so that computation can be expressed through vectorized operations, or via frameworks such as Vectorflow (Rostykus and Raimond, 2018) or Tensorflow (Abadi et al., 2015) ragged tensors that allow for variable-length inputs to bring the computational cost back to $O(\text{avg}_i t_i)$.

2.3. Convexity

For brevity, define $\alpha_i = \alpha(x_i)$ and $\beta_i = \beta(x_i)$. In the special case where $a(x) = \gamma_a \cdot x$ and $b(x) = \gamma_b \cdot x$ are linear functions, their second derivative is null and the Hessian of the log-likelihood of Equation 2 is diagonal:

$$\frac{\partial^2 \ell}{\partial \gamma_{a,j}^2} = \gamma_{a,j}^2 \alpha_i \sum_i \left[\frac{\beta_i}{(\alpha_i + \beta_i)^2} + \sum_{u=2}^{t_i} \frac{\beta_i + u - 1}{(\alpha_i + \beta_i + u - 1)^2}\right]$$

and

$$\frac{\partial^2 \ell}{\partial \gamma_{b,j}^2} = \gamma_{b,j}^2 \beta_i \sum_i \left[\frac{\alpha_i}{(\alpha_i + \beta_i)^2} + \sum_{u=2}^{t_i} k_i\right] \tag{5}$$

Table 1: Prediction accuracy (*C-index*) of linear and GBRT versions of the Beta-Logistic compared to common baselines on 3 different data-sets: Criteo Online Conversions, Subscriptions, and Democracies and Dictatorships. For each data-set we created 10 bootstrap samples of 80% of train and test data to estimate the standard errors of .002, .002, and .02 respectively with 95% confidence.

models		*Conversions*	*Subscriptions*	*Democracies*
(1) Logistic Short Window		0.610	0.785	0.67
(2) Logistic Long Window		0.842	0.849	0.61
(3) Weibull		0.744	0.818	0.74
(4) Exponential		0.773	0.847	0.69
(5) Deep Survival Machines		0.860	0.862	0.72
(6) Beta-Logistic		0.843	0.849	0.73
(7) Beta-Logistic GBRT		**0.898**	**0.876**	**0.77**
data-set properties	# examples	1 mil	4 mil	1.8k
	# observations	220k	1.88 mil	1.4k
	# dimensionality	102k	257k	3.8k

where

$$k_i = \begin{cases} (\alpha_i + 1)\frac{\beta_i^2 - (u-2)(\alpha_i+u-1)}{(\beta_i+u-2)^2(\alpha_i+\beta_i+u-1)^2} & \text{if } c_i = 0 \\ \alpha_i \frac{\beta_i^2 - (u-1)(\alpha_i+u-1)}{(\beta_i+u-1)^2(\alpha_i+\beta_i+u-1)^2} & \text{otherwise.} \end{cases}$$

We see that the log-likelihood of the shifted-beta geometric model is always convex in α when a is linear. Further we can see that when all points are observed (no censoring), and the maximum horizon is $T = 2$ then Equation 5 is also convex in b.

Subsequent terms are not convex, however, but despite that in practice we do not encounter significant convexity issues (e.g. local minima and saddle points). It seems likely that in practice the convex terms of the likelihood dominate the non-convex terms. Note once again that there is generally no global convexity of the objective function.

3. Experiments

3.1. Model Details

All conditional linear models are implemented as sparse linear models in Vectorflow (Rostykus and Raimond, 2018) and trained through stochastic gradient descent. We compare our model with 2 other survival models: **Exponential** and **Weibull**. Both are parametric models where the parameters of the distributions are estimated through an exponential reparametrization (since they require positivity in their parameters). We also add as baselines 2 logistic models: one trained at a short horizon (**Logistic Short Window**), and one trained at a long horizon (**Logistic Long Window**) using the maximum window or point of right censoring as these represent windows used with classification models found commonly in industry. For completeness we also baselined against **Deep Survival Machines** (Nagpal et al., 2021) where we tuned learning rate and number of iterations as well

as running a brute force sweep of distribution, number of mixtures in {4,6,8}, and hidden layers of size {50,100,200}.

We also include the beta-logistic trained via lightGBM (Ke et al., 2017) (**Beta-Logistic GBRT**) for comparison to the above sparse linear models. It is worth noting that with a small amount of modification this model can be implemented through lightGBM with a custom loss in python, however it is several orders of magnitude faster to program it directly within the lightGBM package. We have added python implementations for the likelihood, gradient, and Hessian in the reproducibility portion of the appendix, and will provide the code[1] for this project as well to further facilitate reproducibility.

For each data-set shown in the sections to follow, 10 bootstrap samples of 80% of the data are created to train and test the models where half of the sample is used in training and the other half used for evaluation. We then use these samples to compute the mean *C-index* and the standard errors shown in Table 1.

3.2. Synthetic Data

We present simulation results for the beta-logistic, and compare them to the logistic model. We also show that the beta-logistic successfully recovers the posterior for skewed distributions. In our first simulation we have 3 beta distributions which have very different shapes (see table 2 below), but with the same mean (this example is inspired by Fader et al. (2018)). Here, each simulated customer draws a coin from one of these distributions, and then flips that coin repeatedly until they have an event or we reach a censoring horizon (in this particular case we considered 4 decision points).

shape	α	β	μ
normal	4.75	14.25	0.25
right skewed	0.5	1.50	0.25
u shaped	$0.08\bar{3}$	0.25	0.25

Table 2: Heterogeneous beta distributions with identical means.

It is trivial to show that the logistic model will do no better than random in this case, because it is not observing the dynamics of the survival distribution which reveal the differing levels of heterogeneity underlying the 3 populations. If we allow the beta-logistic model to have a dummy variable for each of these cases then it can recover the posterior of each (see Figure 2). This illustrates an important property of the beta-logistic: it recovers posterior estimates even when the data is very heterogeneous and allows us to fit survival distributions well.

3.3. Online Conversions Data

Survival modeling

We now evaluate the performance of the beta-logistic model on a large-scale sparse data-set. We use the Criteo online conversions data-set published alongside Chapelle (2014) and

1. Scripts available upon request from the authors

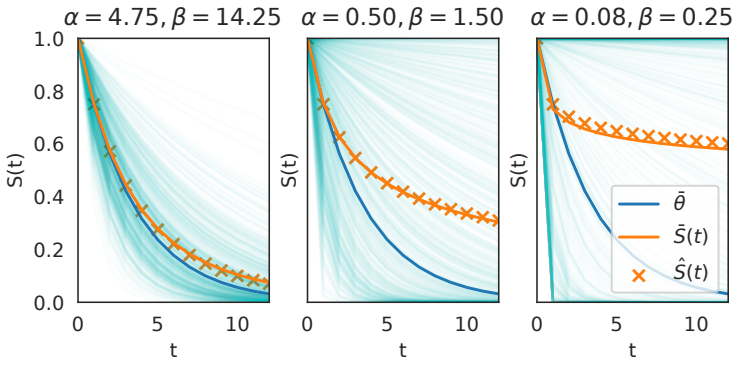

Figure 2: Survival distributions as a function of time as well as an estimate of $\hat{S}(t)$ from the beta-logistic. Using a point-estimate of the mean $\bar{\theta}$ (as in the logistic model) fails to recover the heterogeneity.

publicly available for download[2]. We consider the problem of modeling the distribution of the time between a click event and a conversion event. We will consider a censoring window of 24 hours. As noted in Chapelle (2014), the exponential distribution fits the data reasonably well so we will compare the beta-logistic model against the exponential distribution (1 parameter) and the Weibull distribution (2 parameters). Since the temporal integration of the beta-logistic model is intrinsically discrete, we consider a time-discretization of 5 minute steps. We also add as baselines 2 logistic models: one trained at a horizon of 5 minutes (the shortest interval), and one trained at a horizon of 24 hours (the largest window). Censored events are down-sampled by a factor of 10x. We use 1M rows where we hold out 50% for training and 50% for testing. The total (covariate) dimensionality of the problem is 102K after one-hot-encoding. Note that covariates are sparse and the overall sparsity of the problem is over 99.98%. Results are presented in Table 1.

The beta-logistic survival model outperforms other baselines at all horizons considered in this example. Deep Survival Machines do perform quite well here also, and we suspect that there are rich interactions in the data which these models can exploit. Even though it is a 2-parameter distribution, the Weibull model is interestingly performing worse than the exponential survival model and the binary logistic classifier. We hypothesize that this is due to the poor conditioning of its loss function as well as the numerical instabilities during gradient and expectation computation (the latter requires function calls to the gamma function which is numerically difficult to estimate for moderately small values and for large values).

Posterior size comparison

We next consider the problem as a binary classification task (did a conversion happen within the specified time window?). It is interesting to compare the confidence interval sizes of various models. For the conditional beta-logistic model, the prediction variance on

2. http://labs.criteo.com/2013/12/conversion-logs-dataset/

datapoint x is given by:

$$\mathrm{Var}(x) = \frac{\alpha(x)\beta(x)}{(\alpha(x)+\beta(x))^2\,(\alpha(x)+\beta(x)+1)}.$$

For a logistic model parameterized by $\theta \in \mathbb{R}^d$, a standard way to estimate the confidence of a prediction is through the Laplace approximation of its posterior (MacKay, 2003). In the high-dimensional setting, estimating the Hessian or its inverse become impractical tasks (storage cost is $O(d^2)$ and matrix inversion requires $O(d^3)$ computation). In this scenario, it is customary to assume independence of the posterior coordinates and restrict the estimation to the diagonal of the Hessian $h = \dfrac{1}{\sigma^2} \in \mathbf{R}^d$, which reduces both storage and computation costs to $O(d)$. Hence under this assumption, for a given datapoint x the distribution of possible values for the random variable $Y = \theta^T x$ is also Gaussian with parameters:

$$\mathcal{N}\left(\sum_i \theta_i x_i, \sum_i \sigma_i^2 x_i^2\right).$$

If the full Hessian inverse H^{-1} is estimated, then Y is Gaussian with parameters:

$$\mathcal{N}\left(\theta \cdot x, x^T \cdot H^{-1} x)\right).$$

When Y is Gaussian, the logistic model prediction

$$\mathbb{P}(T = 1|x, \theta) = \frac{1}{1+\exp(-Y)}$$

has a distribution for which the variance v can be conveniently approximated. See Li et al. (2012) for various suggested approximations schemes. We chose to apply the following approximation

$$v = \Phi\left(\frac{\pi\mu/\sqrt{8}-1}{\sqrt{\pi-1+\pi^2\sigma^2/8}}\right) - \left(1+\exp(-\mu/\sqrt{1+\pi\sigma^2/8})\right)^{-2}.$$

Armed with this estimate for the logistic regression posterior variance, we run the following experiment: we random-project (using Gaussian vectors) the original high-dimensional data into a 50-dimensional space, in which we train beta-logistic classifiers and logistic classifiers at various horizons, using 50k training samples every time. We then compare the average posterior size variance on a held-out data-set containing 50k samples. Holdout AUCs were comparable in this case for both models at all horizons. Two posterior approximations are reported for the logistic model: one using the full Hessian inverse and the other one using only the diagonalized Hessian. Results are reported in Figure 3.

Note that the beta-logistic model produces much smaller uncertainty estimates (between 20% and 45% smaller) than the logistic model with Laplace approximation. Furthermore, the growth rate as a function of the horizon of the binary classifier is also smaller for the beta-logistic approach. Also note that the Laplace posterior with diagonal Hessian approximation underestimates the posterior obtained using the full Hessian. Gaussian posteriors are obviously unable to appropriately model data skew.

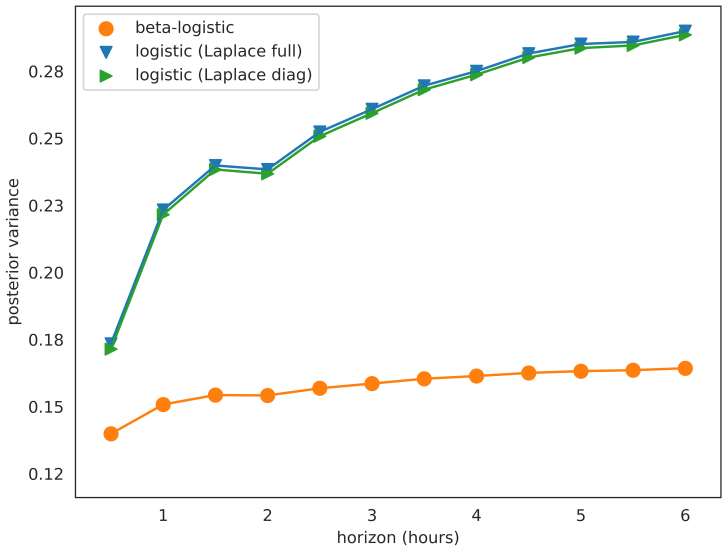

Figure 3: The posterior variance of beta-logistic binary classifiers as well as logistic regressions trained on binary labels data-sets with increasing censoring windows.

This empirical result is arguably clear evidence of the superiority of the posteriors generated by the beta-logistic model over a standard Laplace approximation estimate layered onto a logistic regression model posterior. The beta-logistic posterior is also much cheaper to recover in terms of computational and storage costs. This also suggests that the beta-logistic model could be a viable alternative to standard techniques of explore-exploit models in binary classification settings.

3.4. Subscription Video Streaming Data

We now study the problem of modeling customer retention for a subscription business. We leverage a proprietary data-set from a popular video streaming service. In a subscription setting when a customer chooses not to renew the service, the outcome is explicitly observed and logged. From a practical perspective, it is obviously preferable and meaningful when a customer's tenure with the service is longer rather than shorter. It is also valuable to be able to estimate and project that tenure accurately across different cohorts of customers in order to estimate future revenue, or as input to other estimations where retention is the outcome.

In this particular data-set the cohorts are highly heterogeneous. In Figure 4 we demonstrate this by examining the fitted posterior for 3 different cohorts where you can see that we find large differences in the underlying distributions.

In this example the data set has more than 4M rows with a total dimensionality of around 208k. The data contains a mix of continuous and categorical features and is rich representation of the customer experience. We used 4 discrete decision points to fit the models and samples were taken from multiple points of time so that there is a varying

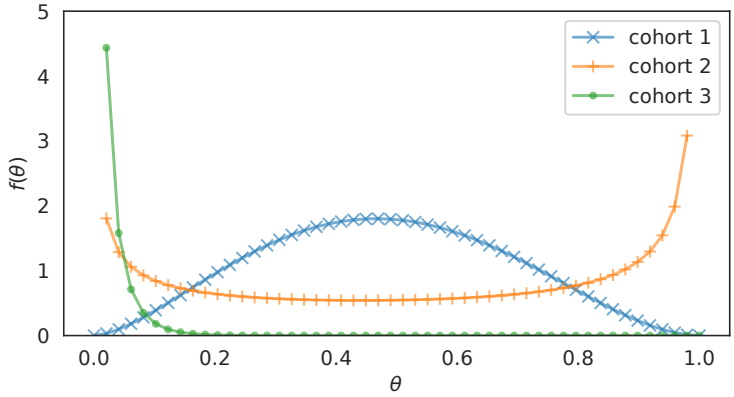

Figure 4: Estimated churn probabilities for 3 different cohorts. The large variations in the shape of the fitted distributions motivate the use of a beta prior on the conditional churn probability.

amount of right censoring as you would find in a typical production setting where models are recomputed over time and data is added into the data set as it is updated. Evaluation was done on a subset of 2M rows which was held out from the model training. In Table 1, we show that the beta-logistic performs quite well as a baseline, but not significantly better than the logistic for ranking when using the longest window as the binary outcome. Both Deep Survival Machines, and the Beta-Logistic GBRT outperform the linear baselines, so we suspect as in the conversion data that there are rich interactions that these models can exploit. It can also easily be shown that although the ranking is quite good for the logistic with long window, the model does not generalize well outside of the sampled time windows e.g. if one were tempted to use the output probabilities and extend to survival using a geometric hypothesis. Using only 1 month as an outcome as in the logistic short window leads to much poorer results as demonstrated here as well as in our simulations, as the model can do no more than compute the mean within local neighborhoods of different cohorts. This illustrates that there is a high level of heterogeneity within even our richest data-sets where one might be tempted to assume that there is enough information to find homogeneous subgroups.

3.5. Democracies and Dictatorships Data

Here we explore the performance of the beta-logistic on the democracies and dictatorships data-set published by Cheibub and Vreeland (2010) and as processed and provided by the lifelines package (Davidson-Pilon, 2019). This data contains country, region, continent, and leader names as well as the type of regime, the start year, and finally classification of democracies as parliamentary, semi-presidential (mixed) and presidential and classification of dictatorships as military, civilian and royal. There is coverage of 202 countries from 1946 until 2008 and the outcome is the duration of these political regimes. Here again we have sampled half of the data for training and the other for testing. From Table 1 we can

see that in this case the Weibull is our best performing sparse linear model. Because the Weibull did so well, we expected that the Deep Survival Machines would do even better, but we think perhaps the limited size of this dataset does not allow DSM to perform as well as we would expect. Again in this example the non-linear beta-logistic does significantly better than all the other baselines.

4. Conclusion

We extended the beta-logistic and its maximum likelihood estimation to linear, tree, and neural models as well as characterized the convexity and properties of the learning problem. Empirical results demonstrate that the beta-logistic is an effective model in discrete time to event problems, and improves over common baselines in the examples given. It seems that in practice regardless of how many attributes are considered there are still unobserved variations between individuals that influence the time to event. Further we demonstrated that we can recover posteriors effectively even when the data is very heterogeneous, and due to the speed and ease of implementation we argue that the beta-logistic is a baseline that should be considered in time to event problems in practice.

In future work, we plan to study the potential use of the beta-logistic in explore-exploit scenarios and as a viable option in reinforcement learning to model long-term consequences from near-term decisions and observations. We would also like to add additional evaluation datasets, and to evaluate outcomes that are outside of the observation window to show the power of the models in extrapolating beyond their observed horizon.

5. Acknowledgments

The authors would like to thank Nikos Vlassis for his helpful comments during the development of this work, and Harald Steck for his thoughtful review and guidance. We would also like to thank the reviewers for their helpful comments and feedback.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Yoram Ben-Porath. Labor-force participation rates and the supply of labor. *Journal of Political economy*, 81(3):697–704, 1973.

James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

Allison Chang, Cynthia Rudin, Michael Cavaretta, Robert Thomas, and Gloria Chou. How to reverse-engineer quality rankings. *Machine Learning*, 88:369–398, September 2012.

Olivier Chapelle. Modeling delayed feedback in display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1097–1105. ACM, 2014.

Jennifer Gandhi Cheibub, José Antonio and James Raymond Vreeland. Democracy and dictatorship revisited. *Public Choice*, 143(2-1):67–101, 2010.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

Cameron Davidson-Pilon. lifelines: survival analysis in python. *Journal of Open Source Software*, 4(40):1317, 2019. doi: 10.21105/joss.01317. URL https://doi.org/10.21105/joss.01317.

Peter S Fader and Bruce GS Hardie. How to project customer retention. *Journal of Interactive Marketing*, 21(1):76–90, 2007.

Peter S Fader, Bruce GS Hardie, Yuzhou Liu, Joseph Davin, and Thomas Steenburgh. "how to project customer retention" revisited: The role of duration dependence. *Journal of Interactive Marketing*, 43:1–16, 2018.

Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

Sunil Gupta, Dominique Hanssens, Bruce Hardie, William Kahn, V. Kumar, Nathaniel Lin, Nalini Ravishanker, and S. Sriram. Modeling customer lifetime value. *Journal of Service Research*, 9(2), 2006.

James J Heckman and Robert J Willis. A beta-logistic model for the analysis of sequential labor force participation by married women. *Journal of Political Economy*, 85(1):27–58, 1977.

John D Kalbfleisch and Ross L Prentice. *The statistical analysis of failure time data*, volume 360. John Wiley & Sons, 2011.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

Lihong Li, Wei Chu, John Langford, Taesup Moon, and Xuanhui Wang. An unbiased offline evaluation of contextual bandit algorithms with generalized linear models. In *Proceedings of the Workshop on On-line Trading of Exploration and Exploitation 2*, pages 19–36, 2012.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Chirag Nagpal, Xinyu Li, and Artur Dubrawski. Deep survival machines: Fully parametric survival regression and representation learning for censored data with competing risks, 2021.

Benoît Rostykus and Yves Raimond. vectorflow: a minimalist neural-network library. *SysML*, 2018.

Cynthia Rudin, David Waltz, Roger N. Anderson, Albert Boulanger, Ansaf Salleb-Aouissi, Maggie Chow, Haimonti Dutta, Philip Gross, Bert Huang, Steve Ierome, Delfina Isaac, Arthur Kressner, Rebecca J. Passonneau, Axinia Radeva, and Leon Wu. Machine learning for the New York City power grid. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):328–345, February 2012.

James W. Vaupel and Anatoliy Yashin. Heterogeneity's ruses: Some surprising effects of selection on population dynamics. *The American statistician*, 39:176–85, 09 1985. doi: 10.1080/00031305.1985.10479424.

Appendix A. Beta Logistic Formulas

A.1. Recurrence derivation

This derivation is taken from Fader and Hardie (2007) where they use it as a cohort model (also called the shifted beta geometric model) that is not conditional on a covariate vector x.

We do not observe θ, but its expectation given the beta prior (also called marginal likelihood) is given by:

$$\begin{aligned}\mathbb{P}(T=t|\alpha,\beta) &= \int_0^1 \theta(1-\theta)^{t-1}\frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)}d\theta \\ &= \frac{B(\alpha+1,\beta+t-1)}{B(\alpha,\beta)}\end{aligned}$$

We can write the above as:

$$\mathbb{P}(T=t|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)*\Gamma(\alpha+1)*\Gamma(\beta+t-1)}{\Gamma(\alpha)*\Gamma(\beta)*\Gamma(\alpha+\beta+t)}.$$

Using the property $\Gamma(z+1)=z\Gamma(z)$ leads to equations (3) and (4), and at $t=1$ we have

$$\begin{aligned}\mathbb{P}(T=1|\alpha,\beta) &= \frac{\Gamma(\alpha+\beta)*\Gamma(\alpha+1)*\Gamma(\beta)}{\Gamma(\alpha)*\Gamma(\beta)*\Gamma(\alpha+\beta+1)} \\ \mathbb{P}(T=1|\alpha,\beta) &= \frac{\alpha}{\alpha+\beta}\end{aligned}$$

A.2. Gradients

Note that for machine learning libraries that do not offer symbolic computation and auto-differentiation, taking the $-log$ of equations (3) and (4) and differentiating leads to the following recurrence formulas for the gradient of the loss function on a given data point with respect to the output parameters a_i and b_i of the model considered. Let $\gamma = \alpha + \beta$, then we have:

$$\frac{\partial \log(\mathbb{P}(T=1))}{\partial a_i} = \frac{\partial a}{\partial a_i}\left(\frac{\beta}{\gamma}\right)$$
$$\frac{\partial \log(\mathbb{P}(T=1))}{\partial b_i} = -\frac{\partial b}{\partial b_i}\left(\frac{\beta}{\gamma}\right)$$

These derivatives expand as follows:

$$\begin{aligned}\frac{\partial \log(\mathbb{P}(T=t))}{\partial a_i} &= \frac{\partial \log(\mathbb{P}(T=t-1))}{\partial a_i} \\ &\quad - \frac{\partial a}{\partial a_i}\left(\frac{\alpha}{\gamma+t-1}\right) \\ \frac{\partial \log(\mathbb{P}(T=t))}{\partial b_i} &= \frac{\partial \log(\mathbb{P}(T=t-1))}{\partial b_i} \\ &\quad + \frac{\partial b}{\partial b_i}\left(\frac{(\alpha+1)\beta}{(\beta+t-2)(\gamma+t-1)}\right)\end{aligned}$$

We can get a similar recursion for the survival function:

$$\frac{\partial \log(\mathbb{P}(T>1))}{\partial a_i} = -\frac{\partial a}{\partial a_i}\left(\frac{\alpha}{\gamma}\right)$$
$$\frac{\partial \log(\mathbb{P}(T>1))}{\partial b_i} = \frac{\partial b}{\partial b_i}\left(\frac{\alpha}{\gamma}\right)$$

$$\begin{aligned}\frac{\partial \log(\mathbb{P}(T>t))}{\partial a_i} &= \frac{\partial \log(\mathbb{P}(T>t-1))}{\partial a_i} \\ &\quad - \frac{\partial a}{\partial a_i}\left(\frac{\alpha}{\gamma+t-1}\right) \\ \frac{\partial \log(\mathbb{P}(T>t))}{\partial b_i} &= \frac{\partial \log(\mathbb{P}(T>t-1))}{\partial b_i} \\ &\quad + \frac{\partial b}{\partial b_i}\left(\frac{\alpha\beta}{(\beta+t-1)(\gamma+t-1)}\right)\end{aligned}$$

Appendix B. Alternative Derivation

Another intuitive derivation of the single-step beta-logistic is obtained by starting from the likelihood for a logistic model and modeling the probabilities with a beta distribution:

$$\begin{aligned} L &= \prod_i \mathbb{P}(y_i = 1|\alpha,\beta)^{y_i}(1-\mathbb{P}(y_i=1|\alpha,\beta))^{y_i-1} \\ &= \prod_{\forall y_i=1} \mathbb{P}(y_i=1|\alpha,\beta) \prod_{\forall y_i=0} (1-\mathbb{P}(y_i=1|\alpha,\beta)) \\ &= \prod_{uncensored} \mathbb{P}(T=1|\alpha,\beta) \prod_{censored} \mathbb{P}(t >= 1|\alpha,\beta). \end{aligned}$$

This is exactly the survival likelihood for a 1 step beta logistic model.

3. Reproducibility

We include simple python implementations of the gradient callbacks that can be passed to XgBoost or lightGBM. Note that efficient implementations of these callbacks in C++ are possible and yield orders of magnitude speedups.

```
def grad_BL(alpha,beta,t,is_censored):
    """
    This function computes the gradient of the beta logistic objective.
    Since it is vectorized in practice for performance reasons, here we write
    the non-vectorized version for readability:
    """
    N = len(alpha)
    g = np.zeros((N, 2))
    for j in range(0,N):
        if (not is_censored[j]):
            #failed
            g[j,0] = beta[j]/(alpha[j] + beta[j])
            g[j,1] = -g[j,0]
            for i in range(2,int(t[j] + 1)):
                g[j,0] += -(alpha[j]/(alpha[j] + beta[j] + i - 1))
                g[j,1] += beta[j]/(beta[j] + i - 2) - beta[j]/(alpha[j] + beta[j
        else:
            #survived
            g[j,:] = -alpha[j]/(beta[j] + alpha[j])
            g[j,1] = -g[j,0]
            for i in range(2,int(t[j] + 1)):
                g[j,0] += -(alpha[j]/(alpha[j] + beta[j] + i - 1))
                g[j,1] += beta[j]/(beta[j] + i - 1) - beta[j]/(alpha[j] + beta[j
    return g

def hess_BL(alpha,beta,t,is_censored):
    """
    This function computes the diagonal of the Hessian of the beta logistic obje
    """
    N = len(alpha)
    h = np.zeros((N,2))
    for j in range(0,N):
        h[j:] = -alpha[j]*beta[j]/((alpha[j] + beta[j])**2)
        if (not is_censored[j]):
            #failed
            for i in range(2,int(t[j] + 1)):
                h[j,0] += -alpha[j]*((beta[j] + i - 1)/(alpha[j] + beta[j] + i -
                d = (beta[j] + i - 2)**2)*(alpha[j] + beta[j] + i - 1)**2)
                h[j,1] += beta[j]*((alpha[j]+1)*(beta[j]**2-(i-2)*(alpha[j]+i-1)
        else:
```

```
            #survived
            for i in range(2,int(t[j] + 1)):
                h[j,0] += -alpha[j]*((beta[j] + i - 1)/(alpha[j] + beta[j] + i -
                d = (beta[j] + i - 2)**2)*(alpha[j] + beta[j] + i - 1)**2)
                h[j,1] += beta[j]*((alpha[j])*(beta[j]**2-(i-1)*(alpha[j]+i-1)/d

    h.shape = (N*2)
    return h

def likelihood_BL(alpha,beta,t,is_censored):
    """
    This function computes beta logistic objective (likelihood : higher = better
    Since it is heavily vectorized in practice for performance reasons, we write
    here the non-vectorized version for readability:
    """
    p = alpha / (alpha + beta)
    s = 1 - p
    for j in range(0,len(alpha)):
        for i in range(2,int(t[j]+1)):
            p[j] = p[j] * (beta[j] + i - 2)/(alpha[j] + beta[j] + i - 1)
            s[j] = s[j] - p[j]
    return p * (1.0 - is_censored) + s * is_censored
```