NGBoost: Natural Gradient Boosting for Probabilistic Prediction

Tony Duan*
tonyduan@cs.stanford.edu

Anand Avati*
avati@cs.stanford.edu

Daisy Yi Ding
dingd@stanford.edu

Sanjay Basu
sanjay_basu@hms.harvard.edu

Andrew Y. Ng
ang@cs.stanford.edu

Alejandro Schuler
alejandro.schuler@gmail.com

Abstract

We present Natural Gradient Boosting (NG-Boost), an algorithm which brings probabilistic prediction capability to gradient boosting in a generic way. Predictive uncertainty estimation is crucial in many applications such as healthcare and weather forecasting. Probabilistic prediction, which is the approach where the model outputs a full probability distribution over the entire outcome space, is a natural way to quantify those uncertainties. Gradient Boosting Machines have been widely successful in prediction tasks on structured input data, but a simple boosting solution for probabilistic prediction of real valued outputs is yet to be made. NGBoost is a gradient boosting approach which uses the *Natural Gradient* to address technical challenges that makes generic probabilistic prediction hard with existing gradient boosting methods. Our approach is modular with respect to the choice of base learner, probability distribution, and scoring rule. We show empirically on several regression datasets that NG-Boost provides competitive predictive performance of both uncertainty estimates and traditional metrics.

1 Introduction

Many real-world supervised machine learning problems have tabular features and real-valued targets. Weather forecasting (predicting temperature of the next day based on today's atmospheric variables
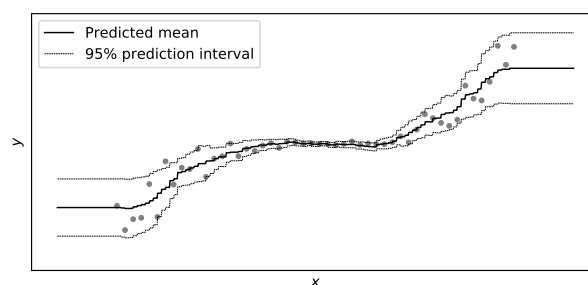
---

*Equal contribution.

Figure 1: Prediction intervals for a toy 1-dimensional probabilistic regression problem, fit via NGBoost. The dots represent data points. The thick black line is the predicted mean after fitting the model. The thin gray lines are the upper and lower quantiles covering 95% of the prediction distribution. NGBoost enables probabilistic prediction of real values with gradient boosting.

(Gneiting and Katzfuss, 2014)) and clinical prediction (predicting time to mortality with survival prediction on structured medical records of the patient (Avati et al., 2018)) are important examples. But rarely should the model be absolutely confident about a prediction. In such tasks it is crucial to estimate the uncertainty in predictions. This is especially the case when the predictions are directly linked to automated decision making, as probabilistic uncertainty estimates are important in determining manual fall-back alternatives in the workflow (Kruchten, 2016).

Bayesian methods naturally generate predictive uncertainty estimates by integrating predictions over the posterior, but they also have practical shortcomings when one is only interested in predictive uncertainty (as opposed to inference over parameters of interest). Exact solutions to Bayesian models are limited to simple models, and calculating the posterior distribution of more powerful models such as Neural Networks (NN) (Neal, 1996) and Bayesian Additive Regression Trees (BART) (Chipman et al., 2010) is difficult. In-
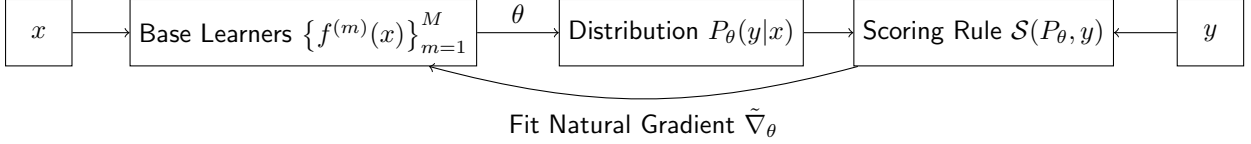
Figure 2: NGBoost is modular with respect to choice of base learner, distribution, and scoring rule.

ference in these models requires computationally expensive approximation via, for example, MCMC sampling. Moreover, sampling-based inference requires some statistical expertise and thus limits the ease-of-use of Bayesian methods. Among non-parametric Bayesian approaches, scalability to very large datasets can be a challenge (Rasmussen and Williams, 2005). Bayesian Deep Learning is gaining popularity (Gal and Ghahramani, 2016; Lakshminarayanan et al., 2017), but, while neural networks have empirically excelled at perception tasks (such as with visual and audio input), they perform on par with traditional methods when data are tabular. Small training datasets and informative prior specification are also challenges for Bayesian neural nets.

Alternatively, meteorology has adopted *probabilistic forecasting* as the preferred methodology for weather prediction. In this setting, the output of the model, given the observed features, is a probability distribution over the entire outcome space. The models are trained to maximize sharpness, subject to calibration, by optimizing scoring rules such as Maximum Likelihood Estimation (MLE) or the more robust Continuous Ranked Probability Score (CRPS) (Gneiting and Raftery, 2005, 2007). This yields calibrated uncertainty estimates. Outside of meteorology, sharp and calibrated probabilistic prediction of time-to-event outcomes (such as mortality) has recently been explored in healthcare (Avati et al., 2019).

Meanwhile, Gradient Boosting Machines (GBMs) (Chen and Guestrin, 2016; Friedman, 2001) are a set of highly-modular methods that work well on structured input data, even with relatively small datasets. This can be seen in their empirical success on Kaggle and other competitions (Chen and Guestrin, 2016). In classification tasks, their predictions are probabilistic by default (by use of the sigmoid or softmax link function). But in regression tasks, they output only a scalar value. Under a squared-error loss these scalars can be interpreted as the mean of a conditional Gaussian distribution with some (unknown) constant variance. However, such probabilistic interpretations have little use if the variance is assumed constant. The predicted distribution needs to have at least two degrees of freedom (two parameters) to effectively convey both the magnitude and the uncertainty of the prediction,

as illustrated in Figure 1. It is precisely this problem of simultaneous boosting of multiple parameters from the base learners which makes probabilistic forecasting with GBMs a challenge, and NGBoost addresses this with the use of natural gradients (Amari, 1998).

Summary of Contributions

i. We present Natural Gradient Boosting, a modular boosting algorithm for probabilistic forecasting (sec 2.4) which uses natural gradients to integrate any choice of:
   - Base learner (e.g. Decision Tree),
   - Parametric probability distribution (Normal, Laplace, etc.), and
   - Scoring rule (MLE, CRPS, etc.).

ii. We present a generalization of the Natural Gradient to other scoring rules such as CRPS (sec 2.2).

iii. We demonstrate empirically that NGBoost performs competitively relative to other models in its predictive uncertainty estimates as well as on traditional metrics (sec 3).

2 Natural Gradient Boosting

In standard prediction settings, the object of interest is an estimate of the scalar function $\mathbb{E}[y|x]$, where x is a vector of observed features and y is the prediction target. In our setting we are interested in producing a probabilistic forecast with probability density $P_\theta(y|x)$, by predicting parameters $\theta \in \mathbb{R}^p$. We denote the corresponding cumulative densities as F_θ.

2.1 Proper Scoring Rules

We start with an overview of proper scoring rules and their corresponding induced divergences, which lays the groundwork for describing the natural gradient.

A proper scoring rule $\mathcal{S}$ takes as input a forecasted probability distribution P and one observation y (outcome), and assigns a score $\mathcal{S}(P, y)$ to the forecast such that the true distribution of the outcomes gets the best score in expectation (Gneiting and Raftery, 2007). In mathematical notation, a scoring rule $\mathcal{S}$ is a proper scoring rule if and only if it satisfies

$$\mathbb{E}_{y \sim Q}[\mathcal{S}(Q, y)] \quad \leq \quad \mathbb{E}_{y \sim Q}[\mathcal{S}(P, y)] \quad \forall P, Q, \quad (1)$$

where Q represents the true distribution of outcomes y, and P is any other distribution (such as the probabilistic forecast from a model). Proper scoring rules encourage the model to output calibrated probabilities when used as loss functions during training.

We limit ourselves to a parametric family of probability distributions, and identify a particular distribution by its parameters θ.

The most commonly used proper scoring rule is the log score $\mathcal{L}$, also known as MLE:

$$\mathcal{L}(\theta, y) \quad = \quad -\log P_\theta(y).$$

The CRPS is another proper scoring rule, which is generally considered a robust alternative to MLE. The CRPS applies only to real valued probability distributions and outcomes. While the MLE enjoys better asymptotic properties in the well specified case, empirically the CRPS tends to produce sharper prediction distributions when the noise model is mis-specified (Gebetsberger et al., 2018). The CRPS (denoted $\mathcal{C}$) is defined as

$$\mathcal{C}(\theta, y) = \int_{-\infty}^{y} F_\theta(z)^2 dz + \int_{y}^{\infty} (1 - F_\theta(z))^2 dz,$$

where F_θ is the cumulative distribution function of P_θ.

Divergences. Every proper scoring rule satisfies the inequality of Eqn 1. The excess score of the right hand side over the left is the divergence induced by that proper scoring rule (Dawid and Musio, 2014):

$$D_\mathcal{S}(Q\|P) = \mathbb{E}_{y\sim Q}[\mathcal{S}(P, y)] - \mathbb{E}_{y\sim Q}[\mathcal{S}(Q, y)],$$

which is necessarily non-negative, and can be interpreted as a measure of difference from one distribution Q to another P.

The MLE scoring rule induces the Kullback-Leibler divergence (KL divergence, or D_{KL}):

$$D_\mathcal{L}(Q\|P) = \mathbb{E}_{y\sim Q}[\mathcal{L}(P, y)] - \mathbb{E}_{y\sim Q}[\mathcal{L}(Q, y)]$$
$$= \mathbb{E}_{y\sim Q}\left[\log \frac{Q(y)}{P(y)}\right]$$
$$=: D_{KL}(Q\|P),$$

while CRPS induces the L^2 divergence (Dawid, 2007):

$$D_\mathcal{C}(Q\|P) = \mathbb{E}_{y\sim Q}[\mathcal{C}(P, y)] - \mathbb{E}_{y\sim Q}[\mathcal{C}(Q, y)]$$
$$= \int_{-\infty}^{\infty} (F_Q(z) - F_P(z))^2 \, dz$$
$$=: D_{L^2}(Q\|P),$$

where F_Q and F_P are the CDFs of the two distributions. A more detailed derivation of the above can be found in (Machete, 2013).

The divergences D_{KL} and D_{L^2} are invariant to the choice of parametrization. Though divergences in general are not symmetric (and hence not a measure of *distance*), at small changes of parameter they are almost symmetric and can serve as a distance measure locally. When used as local measure of distance, the divergences induce a statistical manifold where each point in the manifold corresponds to a probability distribution (Dawid and Musio, 2014).

2.2 The Generalized Natural Gradient

The (ordinary) gradient of a scoring rule $\mathcal{S}$ over a parameterized probability distribution P_θ with parameter θ and outcome y with respect to the parameters is denoted $\nabla\mathcal{S}(\theta, y)$. It is the direction of steepest ascent, such that moving the parameters an infinitesimally small amount in that direction of the gradient (as opposed to any other direction) will increase the scoring rule the most. That is,

$$\nabla\mathcal{S}(\theta, y) \propto \lim_{\epsilon \to 0} \underset{d:\|d\|=\epsilon}{\arg\max} \mathcal{S}(\theta + d, y).$$

It should be noted that this gradient is *not* invariant to reparametrization. Consider reparameterizing P_θ to $P_{z(\theta)}(y)$ so $P_\theta(y) = P_\psi(y)$ for all y when $\psi = z(\theta)$. If the gradient is calculated relative to θ and an infinitesimal step is taken in that direction, say from θ to $\theta + d\theta$ the resulting distribution will be different than if the gradient had been calculated relative to ψ and a step was taken from ψ to $\psi + d\psi$. In other words, $P_{\theta+d\theta}(y) \neq P_{\psi+d\psi}(y)$. Thus the choice of parametrization can drastically impact the training dynamics, even though the minima are unchanged. Because of this, it behooves us to instead update the parameters in a way that reflects how we are moving in the space of distributions, which is ultimately what is of interest.

This motivates the natural gradient (denoted $\tilde{\nabla}$), whose origins can be traced to the field of information geometry (Amari, 1998). While the natural gradient was originally defined for the statistical manifold with the distance measure induced by D_{KL} (Martens, 2014), we provide a more general treatment here that applies to any divergence that corresponds to some proper scoring rule. The generalized natural gradient is the direction of steepest ascent in Riemannian space, which is invariant to parametrization, and is defined:

$$\tilde{\nabla}\mathcal{S}(\theta, y) \propto \lim_{\epsilon \to 0} \underset{d:D_\mathcal{S}(P_\theta\|P_{\theta+d})=\epsilon}{\arg\max} \mathcal{S}(\theta + d, y).$$

If we solve the corresponding the optimization problem, we obtain the natural gradient of the form

$$\tilde{\nabla}\mathcal{S}(\theta, y) \quad \propto \quad \mathcal{I}_\mathcal{S}(\theta)^{-1}\nabla\mathcal{S}(\theta, y)$$
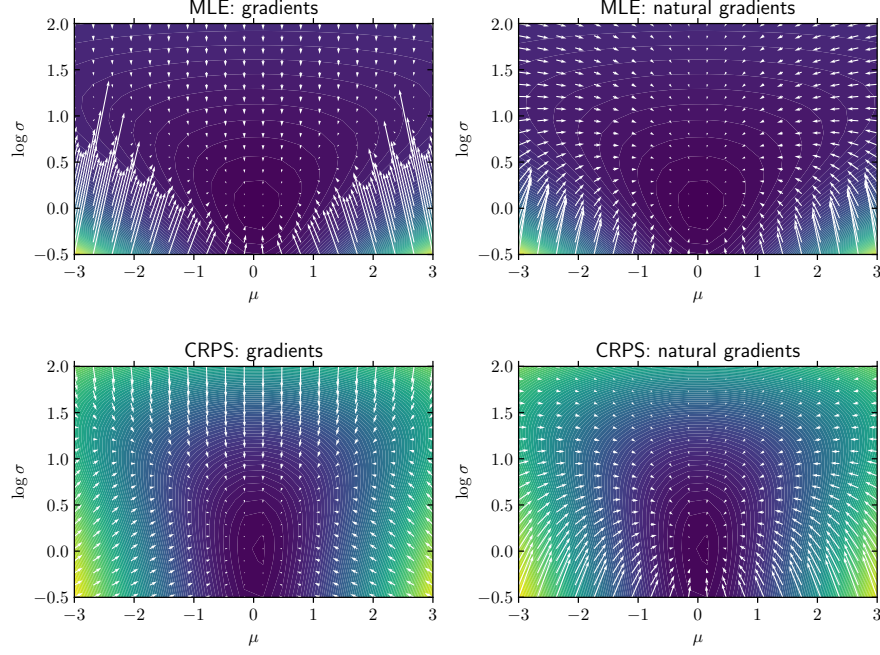
Figure 3: Proper scoring rule loss functions and corresponding gradients for fitting a Normal distribution on samples $\sim N(0, 1)$. For each scoring rule, the landscape of the loss (colors and contours) is identical, but the gradient fields (arrows) are markedly different depending on which kind of gradient is used.

where $\mathcal{I}_\mathcal{S}(\theta)$ is the Riemannian metric of the statistical manifold at θ, which is induced by the scoring rule $\mathcal{S}$.

By choosing $\mathcal{S} = \mathcal{L}$ (i.e. MLE) and solving the above optimization problem, we get:

$$\tilde{\nabla}\mathcal{L}(\theta, y) \quad \propto \quad \mathcal{I}_\mathcal{L}(\theta)^{-1}\nabla\mathcal{L}(\theta, y)$$

where $\mathcal{I}_\mathcal{L}(\theta)$ is the Fisher Information carried by an observation about P_θ, which is defined as:

$$\mathcal{I}_\mathcal{L}(\theta) = \mathbb{E}_{y \sim P_\theta}\left[\nabla_\theta\mathcal{L}(\theta, y)\nabla_\theta\mathcal{L}(\theta, y)^T\right]$$
$$= \mathbb{E}_{y \sim P_\theta}\left[-\nabla_\theta^2\mathcal{L}(\theta, y)\right].$$

Similarly, by choosing $\mathcal{S} = \mathcal{C}$ (i.e. CRPS) and solving the above optimization problem, we get:

$$\tilde{\nabla}\mathcal{C}(\theta, y) \quad \propto \quad \mathcal{I}_\mathcal{C}(\theta)^{-1}\nabla\mathcal{C}(\theta, y)$$

where $\mathcal{I}_\mathcal{C}(\theta)$ is the Riemannian metric of the statistical manifold that uses D_{L^2} as the local distance measure, given by (Dawid, 2007):

$$\mathcal{I}_\mathcal{C}(\theta) = 2\int_{-\infty}^\infty \nabla_\theta F_\theta(z)\nabla_\theta F_\theta(z)^T dz.$$

Using the natural gradient for learning the parameters makes the optimization problem invariant to parametrization, and tends to have a much more efficient and stable learning dynamics than using just

the gradients (Amari, 1998). Figure 3 shows the vector field of gradients and natural gradients for both MLE and CRPS on the parameter space of a Normal distribution parameterized by μ (mean) and $\log\sigma$ (logarithm of the standard deviation).

2.3 Gradient Boosting

Gradient boosting (Friedman, 2001) is a supervised learning technique where several weak learners (or base learners) are combined in an additive ensemble. The model is learnt sequentially, where the next base learner is fit against the training objective residual of the current ensemble. The output of the fitted base learner is then scaled by a learning rate and added into the ensemble.

Gradient boosting is effectively a *functional gradient descent* algorithm. The residual at each stage is the functional gradient of the loss function with respect to the current model. The gradient is then projected onto the range of the base learner class by fitting the learner to the gradient.

The boosting framework can be generalized to any choice of base learner but most popular implementations use shallow decision trees because they work well in practice (Chen and Guestrin, 2016; Ke et al., 2017).

When fitting a decision tree to the gradient, the algo-

rithm partitions the data into axis-aligned slices. Each slice of the partition is associated with a leaf node of the tree, and is made as homogeneous in its response variable (the gradients at that set of data points) as possible. The criterion of homogeneity is typically the sample variance. The prediction value of the leaf node (which is common to all the examples ending up in the leaf node) is then set to be the additive component to the predictions that minimizes the loss the most. This is equivalent to doing a "line search" in the functional optimization problem for each leaf node, and, for some losses, closed form solutions are available. For example, for squared error, the result of the line search will yield the sample mean of the response variables in the leaf.

We now consider adapting gradient boosting for prediction of parameters θ in the probabilistic forecasting context. We highlight two opportunities for improvements.

i. *Splitting mismatch.* Even when the loss chosen to be optimized is not the squared error, the splitting criterion of the decision tree algorithm is generally the sample variance of the gradients for efficiency of implementation. But the following linesearch at each leaf node is then performed to minimize the chosen loss. Two examples are the algorithms described in Friedman (2001) for least absolute deviation regression and second-order likelihood classification. Grouping training examples based on similarity of gradient while eventually assigning them a common curvature adjusted gradient can be suboptimal. Ideally we seek a partition where the objective for the linesearch and the splitting criteria in the decision trees are consistent, and yet retains the computational efficiency of the mean squared error criterion.

ii. *Multi-parameter boosting.* It is a challenge to implement gradient boosting to estimate multiple parameters $\theta(x)$ instead of a single conditional mean $\mathbb{E}[y|x]$. Using a single tree per stage with multiple parameter outputs per leaf node would not be ideal since the splitting criteria based on the gradient of one parameter might be suboptimal with respect to the gradient of another parameter. We thus require one tree per parameter at each boosting stage. But with multiple trees, the differences between the resulting partitions would make per-leaf line searches impossible.

Gradient boosting for k-class classification effectively estimates multiple parameters at each stage. Since the parameters in this case are all symmetric, most implementations minimize a second order approximation of the loss function with a di-

agonal Hessian. This breaks down the line search problem into k independent 1-parameter subproblems, where a separate line search can be implemented at each leaf of each of the k trees, thereby sidestepping the issues of multiparameter boosting. In a more general setting, like boosting for the two parameters of a Normal distribution, such an approximation would not work well.

While addressing the splitting mismatch is an incremental improvement (and has also been addressed by Chen and Guestrin (2016); Ke et al. (2017)), allowing for multi-parameter boosting is a radical innovation that enables boosting-based probabilistic prediction in a generic way (including probabilistic regression and survival prediction). Our approach provides both of these improvements, as we describe next.

2.4 NGBoost: Natural Gradient Boosting

The NGBoost algorithm is a supervised learning method for probabilistic forecasting that approaches boosting from the point of view of predicting parameters of the conditional probability distribution $y|x$ as a function of x. Here y could be one of several types ($\{\pm 1\}$, $\mathbb{R}$, $\{1, \ldots, K\}$, $\mathbb{R}_+$, $\mathbb{N}$, etc.) and x is a vector in $\mathbb{R}^d$. In our experiments we focus mostly on real valued outputs, though all of our methods are applicable to other modalities such as classification and time to event prediction.

The algorithm has three modular components, which are chosen upfront as a configuration:

- Base learner (f),
- Parametric probability distribution (P_θ), and
- Proper scoring rule ($\mathcal{S}$).

A prediction $y|x$ on a new input x is made in the form of a conditional distribution P_θ, whose parameters θ are obtained by an additive combination of M base learner outputs (corresponding to the M gradient boosting stages) and an initial $\theta^{(0)}$. Note that θ can be a vector of parameters (not limited to be scalar valued), and they completely determine the probabilistic prediction $y|x$. For example, when using the Normal distribution, $\theta = (\mu, \log\sigma)$ in our experiments. To obtain the predicted parameter θ for some x, each of the base learners $f^{(m)}$ take x as their input. Here $f^{(m)}$ collectively refers to the set of base learners, one per parameter, of stage m. For example, for a normal distribution with parameters μ and $\log\sigma$, there will be two base learners, $f_\mu^{(m)}$ and $f_{\log\sigma}^{(m)}$ per stage, collectively denoted as $f^{(m)} = \left(f_\mu^{(m)}, f_{\log\sigma}^{(m)}\right)$. The predicted outputs are scaled with a stage-specific scaling factor

---

Algorithm 1 NGBoost for probabilistic prediction

Data: Dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$.
Input: Boosting iterations M, Learning rate η, Probability distribution with parameter θ, Proper scoring rule $\mathcal{S}$, Base learner f.
Output: Scalings and base learners $\{\rho^{(m)}, f^{(m)}\}_{m=1}^M$.

$\theta^{(0)} \leftarrow \arg\min_\theta \sum_{i=1}^n \mathcal{S}(\theta, y_i)$ ▷ initialize to marginal
for $m \leftarrow 1, \ldots, M$ **do**
    for $i \leftarrow 1, \ldots, n$ **do**
      $g_i^{(m)} \leftarrow \mathcal{I}_{\mathcal{S}}\left(\theta_i^{(m-1)}\right)^{-1} \nabla_\theta \mathcal{S}\left(\theta_i^{(m-1)}, y_i\right)$
    end
    $f^{(m)} \leftarrow \mathsf{fit}\left(\left\{x_i, g_i^{(m)}\right\}_{i=1}^n\right)$
    $\rho^{(m)} \leftarrow \arg\min_\rho \sum_{i=1}^n \mathcal{S}\left(\theta_i^{(m-1)} - \rho \cdot f^{(m)}(x_i), y_i\right)$
    for $i \leftarrow 1, \ldots, n$ **do**
      $\theta_i^{(m)} \leftarrow \theta_i^{(m-1)} - \eta \left(\rho^{(m)} \cdot f^{(m)}(x_i)\right)$
    end
end

$\rho^{(m)}$, and a common learning rate η:

$$y|x \sim P_\theta(x), \qquad \theta = \theta^{(0)} - \eta \sum_{m=1}^M \rho^{(m)} \cdot f^{(m)}(x).$$

The scaling factor $\rho^{(m)}$ is a single scalar, even if the distribution has multiple parameters. The model is learnt sequentially, a set of base learners $f^{(m)}$ and a scaling factor $\rho^{(m)}$ per stage. The learning algorithm starts by first estimating a common $\theta^{(0)}$ such that it minimizes the sum of the scoring rule $\mathcal{S}$ over the response variables from all training examples, essentially fitting the marginal distribution of y. This becomes the initial predicted parameter $\theta^{(0)}$ for all examples.

In each iteration m, the algorithm calculates, for each example i, the natural gradients $g_i^{(m)}$ of the scoring rule $\mathcal{S}$ with respect to the predicted parameters of that example up to that stage, $\theta_i^{(m-1)}$. Note that $g_i^{(m)}$ has the same dimension as θ. A set of base learners for that iteration $f^{(m)}$ are fit to predict the corresponding components of the natural gradients $g_i^{(m)}$ of each example x_i.

The output of the fitted base learner is the projection of the natural gradient on to the range of the base learner class. This projected gradient is then scaled by a scaling factor $\rho^{(m)}$ since local approximations might not hold true very far away from the current parameter position. The scaling factor is chosen to minimize the overall true scoring rule loss along the direction of the projected gradient in the form of a line search. In practice, we found that implementing this line search

by successive halving of ρ (starting with $\rho = 1$) until the scaled gradient update results in a lower overall loss relative to the previous iteration works reasonably well and is easy to implement.

Once the scaling factor $\rho^{(m)}$ is determined, the predicted per-example parameters are updated to $\theta_i^{(m)}$ by adding to each $\theta_i^{(m-1)}$ the scaled projected gradient $\rho^{(m)} \cdot f^{(m)}(x_i)$ which is further scaled by a small learning rate η (typically 0.1 or 0.01). The small learning rate helps fight overfitting by not stepping too far along the direction of the projected gradient in any particular iteration.

The pseudo-code is presented in Algorithm 1. For very large datasets computational performance can be easily improved by simply randomly sub-sampling mini-batches within the fit() operation.

2.5 Qualitative Analysis and Discussion

Splitting mismatch. The splitting mismatch phenomenon can occur when using decision trees as the base learner. NGBoost uses the natural gradients as the response variable to fit the base learner. This is in contrast with Friedman (2001) where the ordinary gradient is used, and with Chen and Guestrin (2016) where a second order Taylor approximation is minimized in a Newton-Raphson step. When NGBoost uses regression trees as the base learner, the sample variance of the natural gradient serves as the splitting criteria and the sample mean in each leaf node as the prediction. This is another way of making the splitting and linesearch criteria consistent, and comes "for free" as a consequence of using the natural gradient in a somewhat less decision tree-specific way. This is a crucial aspect of NGBoost's modularity.

Parametrization. When the probability distribution is in the exponential family and the parameters are the natural parameters of that family, then a Newton-Raphson step is equivalent to a natural gradient descent step. However, in other parametrizations and distributions, the equivalence need not hold. This is especially important in the boosting context because, depending on the inductive biases of the base learners, certain parametrization choices may result in more suitable model spaces than others. For example, one setting we are particularly interested in is the two-parameter Normal distribution. Though it is in the exponential family, we use a mean (μ) and log-scale ($\log \sigma$) parametrization for both ease of implementation and modeling convenience (to disentangle magnitude of predictions from uncertainty estimates). Since natural gradients are invariant to parametrization this does not pose a problem, whereas

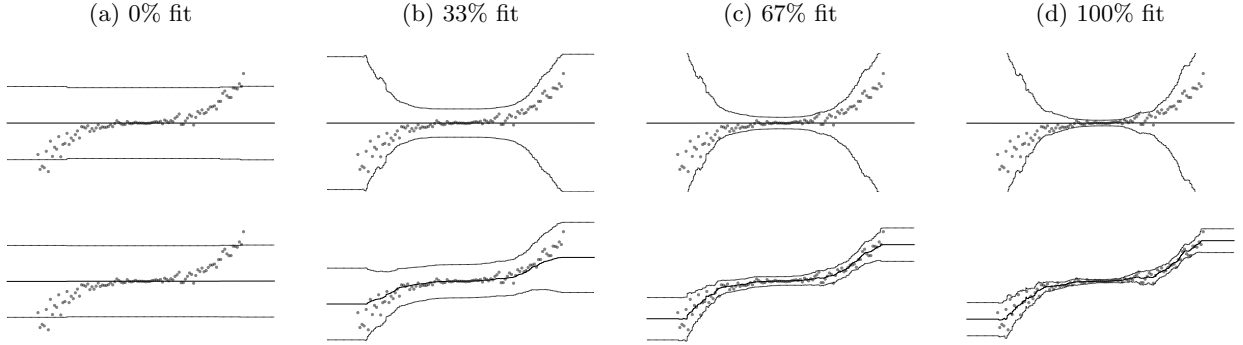| (a) 0% fit | (b) 33% fit | (c) 67% fit | (d) 100% fit |

Figure 4: Contrasting the learning dynamics between using the ordinary gradient (top row) vs. the natural gradient (bottom row) for the purpose of gradient boosting the parameters of a Normal distribution on a toy data set. With ordinary gradients, we observe that "lucky" examples that are accidentally close to the initial predicted mean dominate the learning. This is because, under the ordinary gradient, the variance of those examples that have the correct mean gets adjusted much more aggressively than the wrong means of the "unlucky" examples. This results in simultaneous overfitting of the "lucky" examples in the middle and underfitting of the "unlucky" examples at the ends. Under the natural gradient, all the updates are better balanced.

the Newton-Raphson method would fail as the problem is no longer convex in this parametrization.

Multi-parameter boosting. For predicting multiple parameters of a probability distribution, overall line search (as opposed to per-leaf line search) is an inevitable consequence. NGBoost's use of natural gradient makes this less of a problem as the gradients of all the examples come "optimally pre-scaled" (in both the relative magnitude between parameters, and across examples) due to the inverse Fisher Information factor. The use of ordinary gradients instead would be sub-optimal, as shown in Figure 4. With the natural gradient the parameters converge at approximately the same rate despite different conditional means and variances and different "*distances*" from the initial marginal distribution, even while being subjected to a common scaling factor $\rho^{(m)}$ in each iteration. We attribute this stability to the "optimal pre-scaling" property of the natural gradient.

3 Experiments

Our experiments use datasets from the UCI Machine Learning Repository, and follow the protocol first proposed in Hernández-Lobato and Adams (2015). For all datasets, we hold out a random 10% of the examples as a test set. From the other 90% we initially hold out 20% as a validation set to select M (the number of boosting stages) that gives the best log-likelihood, and then re-fit the entire 90% using the chosen M. The re-fit model is then made to predict on the held-out 10% test set. This entire process is repeated 20 times for all datasets except Protein and Year MSD, for which

it is repeated 5 times and 1 time respectively. The Year MSD dataset, being extremely large relative to the rest, was fit using a learning rate η of 0.1 while the rest of the datasets were fit with a learning rate of 0.01. In general we recommend small learning rates, subject to computational feasibility.

Traditional predictive performance is captured by the root mean squared-error (RMSE) of the forecasted means (i.e. $\hat{\mathbb{E}}[y|x]$) as estimated on the test set. The quality of predictive uncertainty is captured in the average negative log-likelihood (NLL) (i.e. $\log \hat{P}_\theta(y|x)$) as measured on the test set.

Though our methods are most similar to gradient boosting approaches, the problem we are trying to address is probabilistic prediction. Hence, our empirical results are on probabilistic prediction tasks and datasets, and likewise our comparison is against other probabilistic prediction methods. We compare our results against MC dropout (Gal and Ghahramani, 2016) and Deep Ensembles (Lakshminarayanan et al., 2017), as they are the most comparable in simplicity and approach. MC dropout fits a neural network to the dataset and interprets Bernoulli dropout as a variational approximation, obtaining predictive uncertainty by integrating over Monte Carlo samples. Deep Ensembles fit an ensemble of neural networks to the dataset and obtain predictive uncertainty by making an approximation to the Gaussian mixture arising out of the ensemble.

Our results are summarized in Table 1. For all results, NGBoost was configured with the Normal distribution, decision tree base learner with a maximum depth of three levels, and MLE scoring rule.

Dataset	N	RMSE			NLL		
		MC dropout	Deep Ensembles	NGBoost	MC dropout	Deep Ensembles	NGBoost
Boston	506	**2.97 ± 0.85**	**3.28 ± 1.00**	**2.94 ± 0.53**	**2.46 ± 0.25**	**2.41 ± 0.25**	**2.43 ± 0.15**
Concrete	1030	**5.23 ± 0.53**	**6.03 ± 0.58**	**5.06 ± 0.61**	**3.04 ± 0.09**	**3.06 ± 0.18**	**3.04 ± 0.17**
Energy	768	1.66 ± 0.19	2.09 ± 0.29	**0.46 ± 0.06**	1.99 ± 0.09	1.38 ± 0.22	**0.60 ± 0.45**
Kin8nm	8192	0.10 ± 0.00	**0.09 ± 0.00**	0.16 ± 0.00	-0.95 ± 0.03	**-1.20 ± 0.02**	-0.49 ± 0.02
Naval	11934	0.01 ± 0.00	**0.00 ± 0.00**	**0.00 ± 0.00**	-3.80 ± 0.05	**-5.63 ± 0.05**	-5.34 ± 0.04
Power	9568	**4.02 ± 0.18**	4.11 ± 0.17	**3.79 ± 0.18**	**2.80 ± 0.05**	**2.79 ± 0.04**	**2.79 ± 0.11**
Protein	45730	**4.36 ± 0.04**	4.71 ± 0.06	**4.33 ± 0.03**	2.89 ± 0.01	**2.83 ± 0.02**	**2.81 ± 0.03**
Wine	1588	**0.62 ± 0.04**	0.64 ± 0.04	**0.63 ± 0.04**	**0.93 ± 0.06**	0.94 ± 0.12	**0.91 ± 0.06**
Yacht	308	1.11 ± 0.38	1.58 ± 0.48	**0.50 ± 0.20**	1.55 ± 0.12	1.18 ± 0.21	**0.20 ± 0.26**
Year MSD	515345	**8.85 ± NA**	8.89 ± NA	8.94 ± NA	3.59 ± NA	**3.35 ± NA**	3.43 ± NA

Table 1: Comparison of performance on regression benchmark UCI datasets. Results for MC dropout and Deep Ensembles are reported from Gal and Ghahramani (2016) and Lakshminarayanan et al. (2017) respectively. NGBoost offers competitive performance in terms of RMSE and NLL, especially on smaller datasets.

4 Related Work

Calibrated uncertainty estimation. Approaches to probabilistic prediction can broadly be distinguished as Bayesian or non-Bayesian. Bayesian approaches (which include a prior and perform posterior inference) that leverage decision trees for tabular datasets include Chipman et al. (2010) and Lakshminarayanan et al. (2016). A non-Bayesian approach similar to our work is Lakshminarayanan et al. (2017) which takes a parametric approach to training heteroskedastic uncertainty models. Such a heteroskedastic approach to capturing uncertainty has also been called *aleatoric* uncertainty estimation (Kendall and Gal, 2017). Uncertainty that arises due to dataset shift or out-of-distribution inputs (Ovadia et al., 2019) is not in the scope of our work. Post-hoc calibration techniques such as Platt scalaing have also been proposed (Guo et al., 2017; Kuleshov et al., 2018; Kumar et al., 2019), though we focus on learning models that are naturally calibrated. However, we note that such post-hoc calibration techniques are not incompabile with our proposed methods.

Proper scoring rules. Gneiting and Raftery (2007) proposed the paradigm of maximizing sharpness subject to calibration in probabilistic forecasting, and introduced the CRPS scoring rule for meteorology. Recently Avati et al. (2019) extended the CRPS to the time-to-event setting and Gebetsberger et al. (2018) empirically examined its tradeoffs relative to MLE. We extend this line of work by introducing the natural gradient to CRPS and making practical recommendations for regression tasks.

Gradient Boosting. Friedman (2001) proposed the gradient boosting framework, though its use has primarily been in homoskedastic regression as opposed to probabilistic forecasting. We are motivated in part by the empirical success of decision tree base learners, which have favorable inductive biases for tabular datasets (such as Kaggle competitions and electronic health records). Popular scalable implementations of tree-based boosting methods include Chen and Guestrin (2016); Ke et al. (2017).

5 Conclusions

We propose a method for probabilistic prediction (NGBoost) and demonstrate state-of-the-art performance on a variety of datasets. NGBoost combines a multiparameter boosting algorithm with the natural gradient to efficiently estimate how parameters of the presumed outcome distribution vary with the observed features. NGBoost is flexible, modular, easy-to-use, and fast relative to existing methods for probabilistic prediction.

There are many avenues for future work. The natural gradient loses its invariance property with finite step sizes, which we can address with differential equation solvers for higher-order invariance (Song et al., 2018). Better tree-based base learners and regularization (e.g. Chen and Guestrin (2016); Ke et al. (2017)) are worth exploring. Many time-to-event predictions are made from tabular datasets, and we expect NGBoost to perform well in such settings as well (Schmid and Hothorn, 2008).

Acknowledgements

References

Amari, S.-i. (1998). Natural Gradient Works Efficiently in Learning. *Neural Computation*, page 29.

Avati, A., Duan, T., Jung, K., Shah, N. H., and Ng, A. (2019). Countdown Regression: Sharp and Cal-

ibrated Survival Predictions. In *Uncertainty in Artificial Intelligence*. arXiv: 1806.08324.

Avati, A., Jung, K., Harman, S., Downing, L., Ng, A., and Shah, N. H. (2018). Improving palliative care with deep learning. *BMC Medical Informatics and Decision Making*, 18(4):122.

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM. event-place: San Francisco, California, USA.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298.

Dawid, A. P. (2007). The geometry of proper scoring rules. *Annals of the Institute of Statistical Mathematics*, 59(1):77–93.

Dawid, A. P. and Musio, M. (2014). Theory and Applications of Proper Scoring Rules. *METRON*, 72(2):169–183. arXiv: 1401.0398.

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232.

Gal, Y. and Ghahramani, Z. (2016). Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, ICML'16, pages 1050–1059. JMLR.org. event-place: New York, NY, USA.

Gebetsberger, M., Messner, J. W., Mayr, G. J., and Zeileis, A. (2018). Estimation Methods for Nonhomogeneous Regression Models: Minimum Continuous Ranked Probability Score versus Maximum Likelihood. *Monthly Weather Review*, 146(12):4323–4338.

Gneiting, T. and Katzfuss, M. (2014). Probabilistic Forecasting. *Annual Review of Statistics and Its Application*, 1(1):125–151.

Gneiting, T. and Raftery, A. E. (2005). Weather Forecasting with Ensemble Methods. *Science*, 310(5746):248–249.

Gneiting, T. and Raftery, A. E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks. In *International Conference on Machine Learning*, ICML'17, pages 1321–1330. JMLR.org.

Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *International*

Conference on Machine Learning, ICML'15, pages 1861–1869. JMLR.org. event-place: Lille, France.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc.

Kendall, A. and Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc.

Kruchten, N. (2016). Machine learning meets economics.

Kuleshov, V., Fenner, N., and Ermon, S. (2018). Accurate Uncertainties for Deep Learning Using Calibrated Regression. In *International Conference on Machine Learning*, pages 2796–2804.

Kumar, A., Poole, B., and Murphy, K. (2019). Learning Generative Samplers using Relaxed Injective Flow. Technical report.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc.

Lakshminarayanan, B., Roy, D. M., and Teh, Y. W. (2016). Mondrian Forests for Large-Scale Regression when Uncertainty Matters. In *Artificial Intelligence and Statistics*, pages 1478–1487.

Machete, R. L. (2013). Contrasting probabilistic scoring rules. *Journal of Statistical Planning and Inference*, 143(10):1781–1790.

Martens, J. (2014). New insights and perspectives on the natural gradient method. Technical report. arXiv: 1412.1193.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. (2019). Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. Technical report.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adap-*

tive Computation and Machine Learning). The MIT Press.

Schmid, M. and Hothorn, T. (2008). Flexible boosting of accelerated failure time models. *BMC Bioinformatics*, 9:269.

Song, Y., Song, J., and Ermon, S. (2018). Accelerating Natural Gradient with Higher-Order Invariance. In *International Conference on Machine Learning*, pages 4713–4722.