# NGBoost: Natural Gradient Boosting for Probabilistic Prediction

**Tony Duan** [*1] **Anand Avati** [*1] **Daisy Yi Ding** [1] **Khanh K. Thai** [2] **Sanjay Basu** [3] **Andrew Ng** [1]
**Alejandro Schuler** [2]

## Abstract

We present Natural Gradient Boosting (NGBoost), an algorithm for generic probabilistic prediction via gradient boosting. Typical regression models return a point estimate, conditional on covariates, but probabilistic regression models output a full probability distribution over the outcome space, conditional on the covariates. This allows for predictive uncertainty estimation — crucial in applications like healthcare and weather forecasting. NGBoost generalizes gradient boosting to probabilistic regression by treating the parameters of the conditional distribution as targets for a multiparameter boosting algorithm. Furthermore, we show how the *Natural Gradient* is required to correct the training dynamics of our multiparameter boosting approach. NGBoost can be used with any base learner, any family of distributions with continuous parameters, and any scoring rule. NGBoost matches or exceeds the performance of existing methods for probabilistic prediction while offering additional benefits in flexibility, scalability, and usability. An open-source implementation is available at github.com/stanfordmlgroup/ngboost.

## 1. Introduction

Many important supervised machine learning problems are regression problems. Weather forecasting (predicting temperature of the next day based on today's atmospheric variables (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.

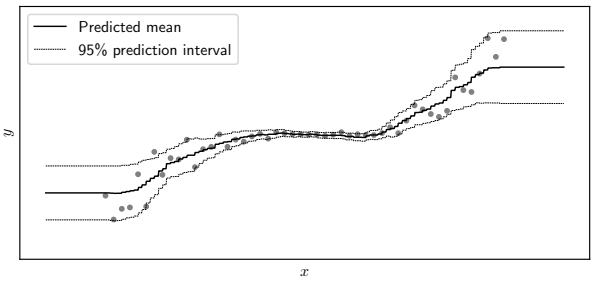


*Figure 1.* Prediction intervals for a toy 1-dimensional probabilistic regression problem, fit via NGBoost. The dots represent data points. The thick black line is the predicted mean after fitting the model. The thin gray lines are the upper and lower quantiles covering 95% of the prediction distribution.

Most machine learning methods tackle this problem with point prediction, returning a single "best guess" prediction (e.g. the temperature tomorrow will be 16°C). However, in these fields it is often important to be able to quantify uncertainty in the prediction or be able to answer multiple questions on the fly (e.g. what's the probability it will be between 18°C and 20°C? What about <15°C?) (Kruchten, 2016).

In order to answer arbitrary questions about the probability of events conditional on covariates, we must estimate the conditional probability distribution $P(y|x)$ for each value of $x$ instead of producing a point estimate like $\mathbb{E}[y|x]$. This is called *probabilistic regression*. Probabilistic regression is increasingly being used in fields like meteorology and healthcare (Gneiting and Raftery, 2007; Avati et al., 2019).

Probabilistic estimation is already the norm in *classification* problems. Although some classifiers (e.g. standard support vector machines) only return a predicted class label, most are capable of returning estimated probabilities for each class; effectively, a conditional probability mass function.

However, existing methods for probabilistic *regression* are either inflexible, slow, or inaccessible to non-experts. Any

[*]Equal contribution [1]Stanford University, Stanford, California, United States [2]Unlearn.ai, San Francisco, California, United States [3]Harvard Medical School, Cambridge, Massachusetts, United States. Correspondence to: Anand Avati <avati@cs.stanford.edu>, Tony Duan <tonyduan@cs.stanford.edu>, Alejandro Schuler <alejandro.schuler@gmail.com>.

*Proceedings of the $37^{th}$ International Conference on Machine Learning*, Online, PMLR 119, 2020. 

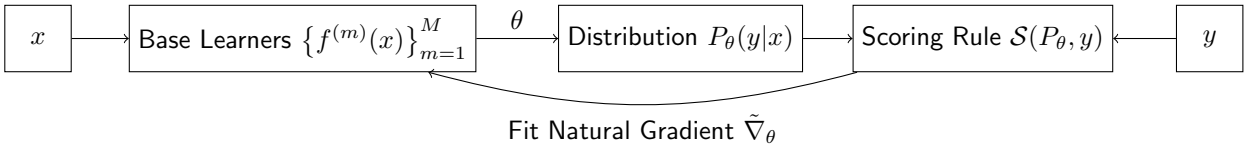


*Figure 2.* NGBoost is modular with respect to choice of base learner, distribution, and scoring rule.

mean-estimating regression method can be made probabilistic by assuming homoscedasticity and estimating an unconditional noise model, but homoscedasticity is a strong assumption and the process requires some statistical know-how. Generalized Additive Models for Shape, Scale, and Location (GAMLSS) allow heteroscedasticity but are restricted to a pre-specified model form (Stasinopoulos et al., 2007). Bayesian methods naturally generate predictive uncertainty estimates by integrating predictions over the posterior, but 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. Inference 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. Bayesian approaches often also scale poorly to large datasets (Rasmussen and Williams, 2005). Bayesian Deep Learning is gaining popularity (Graves, 2011; Blundell et al., 2015; Hernández-Lobato and Adams, 2015) but, while neural networks have empirically excelled at perception tasks (such as with visual and audio input), they usually perform only on par with traditional methods when data are limited in size or tabular. Extensive hyperparameter tuning and informative prior specification are also challenges for Bayesian Deep Learning which make it difficult to use out-of-the-box.

Meanwhile, Gradient Boosting Machines (GBMs) (Friedman, 2001; Chen and Guestrin, 2016) are a set of highly-modular methods that work out-of-the-box and perform well on structured input data, even with relatively small datasets. This can be seen in their empirical success on Kaggle and other data science 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 distributions need to have at least two degrees of freedom (two parameters) to effectively convey both the magnitude and the uncertainty of the predictions, 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 a multiparameter boosting approach and the use of natural gradients (Amari, 1998).

## 2. Summary of Contributions

i. We present Natural Gradient Boosting, a modular algorithm for probabilistic regression (section 3.4) which uses multiparameter boosting and natural gradients to integrate any choice of:
   - Base learner (e.g. Regression 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 (section 3.2).

iii. We demonstrate empirically that NGBoost performs competitively relative to other models in its predictive uncertainty estimates, as well as in its point estimates (section 4).

## 3. Natural Gradient Boosting

In standard prediction settings, the object of interest is an estimate of a scalar function like $\mathbb{E}[y|x]$, where $x$ is a vector of observed features and $y$ is the prediction target. In our setting we are interested in producing a probability distribution $P_\theta(y|x)$ (with CDF $F_\theta$). Our approach is to assume $P_\theta(y|x)$ is of a specified parametric form, then estimate the $p$ parameters $\theta \in \mathbb{R}^p$ of the distribution as functions of $x$.

### 3.1. Proper Scoring Rules

To begin, we need a learning objective. In point prediction, the predictions are compared to the observed data with a loss function. The analogue in probabilistic regression is a *scoring rule*, which compares the estimated probability distribution to the observed data.

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

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

where $Q$ represents the true distribution of outcomes $y$, and $P$ is any other distribution (such as the probabilistic forecast from a model). Since we are working with parametric distributions, we can identify each distribution with its parameters and write the score as $\mathcal{S}(\theta,y)$.

The most commonly used proper scoring rule is the logarithmic score $\mathcal{L}$, which, when minimized, gives the MLE:

$$\mathcal{L}(\theta,y) = -\log P_\theta(y). \qquad (2)$$

Another example is CRPS, which is generally considered a robust alternative to MLE (Gebetsberger et al., 2018). The CRPS (denoted $\mathcal{C}$) is defined as

$$\mathcal{C}(\theta,y) = \int_{-\infty}^{y} F_\theta(z)^2 dz + \int_{y}^{\infty} (1-F_\theta(z))^2 dz. \qquad (3)$$

### 3.2. The Generalized Natural Gradient

We take a standard gradient descent approach to find the parameters that minimize the scoring rule by descending along the negative gradient of the score relative to the parameters at each point $x$. The (ordinary) gradient of a scoring rule $\mathcal{S}$ over a parameterized probability distribution $P_\theta$ with parameter $\theta$ 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,

$$\nabla\mathcal{S}(\theta,y) \propto \lim_{\epsilon\to 0} \operatorname*{arg\,max}_{d:\|d\|=\epsilon} \mathcal{S}(\theta+d,y). \qquad (4)$$

This gradient is *not* invariant to reparameterization. Consider reparameterizing $P_\theta$ to $P_{z(\theta)}(y)$ so $P_\theta(y\in A) = P_\psi(y\in A)$ for all events $A$ when $\psi = z(\theta)$. If the gradient is calculated relative to $\theta$ and an infinitesimal step is taken in that direction, say from $\theta$ to $\theta + d\theta$ the resulting distribution will be different than if the gradient had been calculated relative to $\psi$ and a step was taken from $\psi$ to $\psi + d\psi$. In other words, $P_{\theta+d\theta}(y\in A) \neq P_{\psi+d\psi}(y\in A)$. Thus the choice of parameterization can drastically impact the training dynamics, even though the minima are unchanged.

The problem is that "distance" between two parameter values does not correspond to an appropriate "distance" between the distributions that those parameters identify. This motivates the natural gradient (denoted $\tilde{\nabla}$), which originated in information geometry (Amari, 1998).

**Divergences.** Every proper scoring rule induces a *divergence* that can serve as local distance metric in the space of distributions. A proper scoring rule by definition satisfies the inequality of Eqn 1. The excess score of the right hand side over the left is the divergence induced by that scoring rule (Dawid and Musio, 2014):

$$D_{\mathcal{S}}(Q\|P) = \mathbb{E}_{y\sim Q}[\mathcal{S}(P,y)] - \mathbb{E}_{y\sim Q}[\mathcal{S}(Q,y)], \qquad (5)$$

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 $D_{KL}$), while CRPS induces the $L^2$ divergence (Dawid, 2007; Machete, 2013).

The divergences $D_{KL}$ and $D_{L^2}$ are invariant to how $Q$ and $P$ are parameterized. Though divergences in general are not symmetric, for small changes of the parameters they are almost symmetric and can serve as a local distance metric. When used as such, a divergence induces a statistical manifold where each point in the manifold corresponds to a probability distribution (Dawid and Musio, 2014).

**Natural Gradient.** The generalized natural gradient is the direction of steepest ascent in Riemannian space, which is invariant to parametrization, and is defined as:

$$\tilde{\nabla}\mathcal{S}(\theta,y) \propto \lim_{\epsilon\to 0} \operatorname*{arg\,max}_{d:D_{\mathcal{S}}(P_\theta\|P_{\theta+d})=\epsilon} \mathcal{S}(\theta+d,y). \qquad (6)$$

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

$$\tilde{\nabla}\mathcal{S}(\theta,y) \quad \propto \quad \mathcal{I}_{\mathcal{S}}(\theta)^{-1}\nabla\mathcal{S}(\theta,y) \qquad (7)$$

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

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

$$\tilde{\nabla}\mathcal{L}(\theta,y) \quad \propto \quad \mathcal{I}_{\mathcal{L}}(\theta)^{-1}\nabla\mathcal{L}(\theta,y) \qquad (8)$$

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

$$\mathcal{I}_{\mathcal{L}}(\theta) = \mathbb{E}_{y\sim P_\theta}\left[\nabla_\theta\mathcal{L}(\theta,y)\nabla_\theta\mathcal{L}(\theta,y)^T\right] \qquad (9)$$

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

$$\tilde{\nabla}\mathcal{C}(\theta,y) \quad \propto \quad \mathcal{I}_{\mathcal{C}}(\theta)^{-1}\nabla\mathcal{C}(\theta,y) \qquad (10)$$

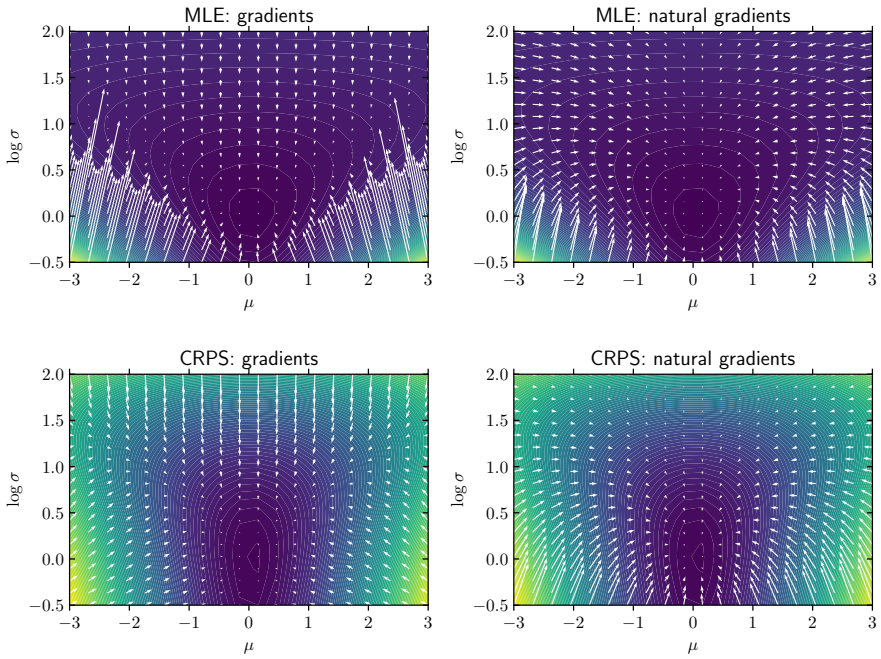


*Figure 3.* Proper scoring rules and corresponding gradients for fitting a Normal distribution on samples $\sim N(0, 1)$. For each scoring rule, the landscape of the score (colors and contours) is identical, but the gradient fields (arrows) are markedly different depending on which kind of gradient is used.

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

$$\mathcal{I}_{\mathcal{C}}(\theta) = 2 \int_{-\infty}^{\infty} \nabla_\theta F_\theta(z) \nabla_\theta F_\theta(z)^T dz. \tag{11}$$

Using the natural gradient for learning the parameters makes the optimization problem invariant to parametrization and leads to more efficient and stable learning dynamics (Amari, 1998). Figure 3 shows the vector field of gradients and natural gradients for $\mathcal{L}$ and $\mathcal{C}$ on the parameter space of a Normal distribution parameterized by $\mu$ (mean) and $\log \sigma$ (logarithm of the standard deviation).

### 3.3. 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.

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 algorithm 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 response variables are residuals, and 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 $\theta$ in the probabilistic regression context.

### 3.4. NGBoost: Natural Gradient Boosting

The NGBoost algorithm is a supervised learning method for probabilistic prediction that uses boosting to estimate the parameters of a conditional probability distribution $P(y|x)$ as functions 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 on real valued outputs, though all of our methods are applicable to other modalities

such as classification and survival prediction.

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

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

**Algorithm 1** NGBoost for probabilistic prediction

**Data:** Dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$.
**Input:** Boosting iterations $M$, Learning rate $\eta$, Probability distribution with parameter $\theta$, 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**

A prediction $y|x$ on a new input $x$ is made in the form of a conditional distribution $P_\theta$, whose parameters $\theta$ 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 $\theta$ 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 $\theta$ 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 $\mu$ 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 stage-specific scaling factors $\rho^{(m)}$, and a common learning rate $\eta$:

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

Each 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 $\theta$. 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 $\rho$ (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 negative scaled projected gradient for $i$, $\rho^{(m)} \cdot f^{(m)}(x_i)$ which is further scaled by a small learning rate $\eta$ (typically 0.1 or 0.01).

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.

### 3.5. Analysis and Discussion

**Boosting for Probabilistic Prediction.** Our boosting approach generalizes gradient boosting to predict conditional distributions. For instance, if the user specifies the conditional distribution to be a Normal distribution with a *fixed* variance and uses the logarithmic scoring rule, our approach recovers the standard boosting algorithm with MSE loss (modulo the per-leaf line search). The advantage of NGBoost is that users are also free to specify any other family of distributions identified by a set of real-valued parameters and allow all of those parameters to vary over the covariates, not just the mean. NGBoost thus trivially extends to a variety of use cases, such as negative binomial boosting (for counts), Gamma or Weibull boosting (for survival prediction, with or without right-censored data), etc.

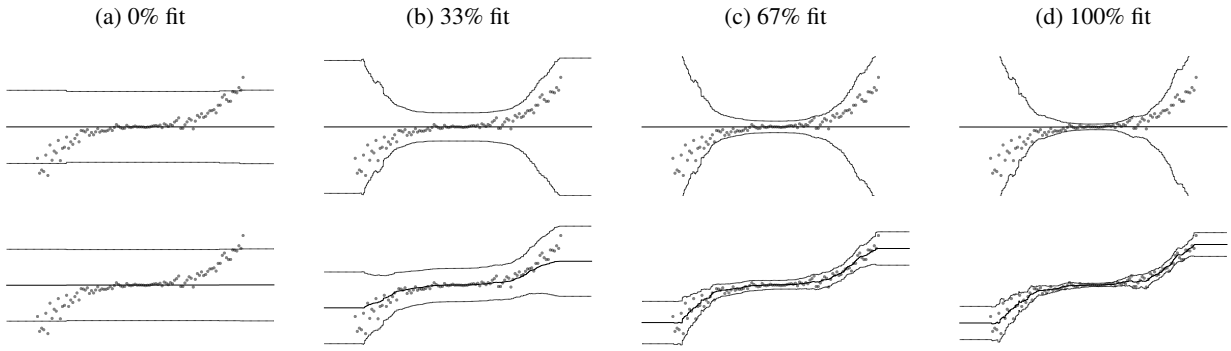


*Figure 4.* Contrasting the learning dynamics between using the ordinary gradient (top row) vs. the natural gradient (bottom row) for the purpose of gradient boosting the parameters of a Normal distribution on a toy data set. With ordinary gradients, we observe that "lucky" examples that are accidentally close to the initial predicted mean dominate the learning. This is because, under the ordinary gradient, the variances of those examples that have the correct mean get 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.

**Multiparameter Boosting.** These wide-ranging extensions of gradient boosting are made possible by turning the distributional prediction problem into a problem of jointly estimating $p$ functions of $x$, one per parameter, according to the scoring rule objective. In this setting, an overall line search for the stage multiplier (as opposed to per-leaf line search) is an inevitable consequence. However, our 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, variances, and "*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.

**Parameterization.** When the probability distribution is in the exponential family and the choice of parameterization is the natural parameters of that family, then a Newton-Raphson step is equivalent to a natural gradient descent step. However, in other parameterizations 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 parameterization 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 ($\mu$) and log-scale ($\log \sigma$) parameterization for both ease of implementation and modeling convenience (to disentangle magnitude of predictions from uncertainty estimates). Since natural gradients are invariant to parameterization this does not pose a problem, whereas the Newton-Raphson method would fail as the problem is no longer convex in this parameterization.

**Computational Complexity.** There are two computational differences between our algorithm and a standard boosting algorithm which contribute to complexity. The first is that a series of learners must be fit for *each* parameter in NGBoost, whereas standard boosting fits only one series of learners. The relative increase in computational cost is thus linear in the number of distributional parameters ($p$). The other difference is that we must compute the natural gradient per observation, which requires as many inversions of a $p \times p$ matrix $\mathcal{I}_S$ as there are observations ($N$). The cost of doing so scales with $p^3$, and linearly with $N$. In practice, both costs are minimal because most commonly used distributions have only one or two parameters and distributions with more than five are exceedingly rare. Scaling in terms of $p$ is therefore not a significant concern. However, even though these costs are otherwise linear in $N$, it may be prudent to avoid inverting a large number of small matrices by sub-sampling mini-batches of data in each boosting iteration. This is done in most implementations of boosting algorithms. All in all, NGBoost scales exactly like other boosting algorithms in terms of $N$ but with larger "constants" that depend on $p \approx 10^0$.

## 4. 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 valida-

*Table 1.* Comparison of probabilistic regression performance on regression benchmark UCI datasets as measured by NLL. Results for MC dropout, Deep Ensembles, and Concrete Dropout are reported from Gal and Ghahramani (2016); Lakshminarayanan et al. (2017); Gal et al. (2017) respectively. NGBoost offers competitive performance in terms of NLL, especially on smaller datasets. The best method for each dataset is bolded, as are those with standard errors that overlap with the best method.

| Dataset | $N$ | NGBoost | MC dropout | Deep Ensembles | Concrete Dropout | Gaussian Process | GAMLSS | DistForest |
|---|---|---|---|---|---|---|---|---|
| Boston | 506 | **2.43 ± 0.15** | **2.46 ± 0.25** | **2.41 ± 0.25** | 2.72 ± 0.01 | **2.37 ± 0.24** | 2.73 ± 0.56 | 2.67 ± 0.08 |
| Concrete | 1030 | **3.04 ± 0.17** | **3.04 ± 0.09** | **3.06 ± 0.18** | 3.51 ± 0.00 | **3.03 ± 0.11** | 3.24 ± 0.08 | 3.38 ± 0.05 |
| Energy | 768 | **0.60 ± 0.45** | 1.99 ± 0.09 | 1.38 ± 0.22 | 2.30 ± 0.00 | **0.66 ± 0.17** | 1.24 ± 0.86 | 1.53 ± 0.14 |
| Kin8nm | 8192 | -0.49 ± 0.02 | -0.95 ± 0.03 | **-1.20 ± 0.02** | -0.65 ± 0.00 | -1.11 ± 0.03 | -0.26 ± 0.02 | -0.40 ± 0.01 |
| Naval | 11934 | -5.34± 0.04 | -3.80 ± 0.05 | -5.63 ± 0.05 | **-5.87 ± 0.05** | -4.98 ± 0.02 | -5.56 ± 0.07 | -4.84 ± 0.01 |
| Power | 9568 | 2.79 ± 0.11 | 2.80 ± 0.05 | 2.79 ± 0.04 | 2.75 ± 0.01 | 2.81 ± 0.05 | 2.86 ± 0.04 | **2.68 ± 0.05** |
| Protein | 45730 | 2.81 ± 0.03 | 2.89 ± 0.01 | 2.83 ± 0.02 | 2.81 ± 0.00 | 2.89 ± 0.02 | 3.00 ± 0.01 | **2.59 ± 0.04** |
| Wine | 1588 | **0.91 ± 0.06** | **0.93 ± 0.06** | **0.94 ± 0.12** | 1.70 ± 0.00 | **0.95 ± 0.06** | **0.97 ± 0.09** | 1.05 ± 0.15 |
| Yacht | 308 | **0.20 ± 0.26** | 1.55 ± 0.12 | 1.18 ± 0.21 | 1.75 ± 0.00 | **0.10 ± 0.26** | 0.80 ± 0.56 | 2.94 ± 0.09 |
| Year MSD | 515345 | 3.43 ± NA | 3.59 ± NA | **3.35 ± NA** | NA± NA | NA ± NA | NA ± NA | NA ± NA |

*Table 2.* Comparison of probabilistic regression performance on regression benchmark UCI datasets as measured by NLL while ablating key components of NGBoost. Multiparameter boosting must be used in tandem with the natural gradient to increase performance. Bolding is as in Table 1.

| Dataset | N | NGBoost | 2nd-Order | Multiparameter | Homoscedastic |
|---|---|---|---|---|---|
| Boston | 506 | **2.43 ± 0.15** | 3.57 ± 0.20 | 3.17 ± 0.13 | **2.79 ± 0.42** |
| Concrete | 1030 | **3.04 ± 0.17** | 4.21 ± 0.05 | 3.94 ± 0.09 | **3.22 ± 0.29** |
| Energy | 768 | **0.60 ± 0.45** | 3.64 ± 0.06 | 3.24 ± 0.09 | **0.68 ± 0.25** |
| Kin8nm | 8192 | -0.49 ± 0.02 | 0.10 ± 0.07 | **-0.52 ± 0.03** | -0.37 ± 0.05 |
| Naval | 11934 | **-5.34± 0.04** | -2.80 ± 0.01 | -3.46 ± 0.00 | -4.35 ± 0.07 |
| Power | 9568 | 2.79 ± 0.11 | 4.11 ± 0.03 | 3.79 ± 0.13 | **2.66 ± 0.11** |
| Protein | 45730 | **2.81 ± 0.03** | 3.23 ± 0.00 | 3.04 ± 0.02 | 2.86 ± 0.01 |
| Wine | 1588 | **0.91 ± 0.06** | 1.21 ± 0.09 | 0.93 ± 0.07 | **1.34 ± 0.67** |
| Yacht | 308 | **0.20 ± 0.26** | 4.11 ± 0.17 | 3.29 ± 0.20 | 2.02 ± 0.21 |
| Year MSD | 515345 | **3.43 ± NA** | 3.80 ± 0.00 | 3.60 ± NA | 3.63 ± NA |

tion set to select $M$ (the number of boosting stages) that gives the best log-likelihood, and then retrain on the entire 90% using the chosen $M$. The retrained 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.

For all experiments, NGBoost was configured with the Normal distribution, decision tree base learner with a maximum depth of three levels, and log scoring rule. The Year MSD dataset, being extremely large relative to the rest, was fit using a learning rate $\eta$ 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. For the Year MSD dataset we use a mini-batch size of 10%, for all other datasets we use 100%.

### 4.1. Probabilistic Regression.

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.

Our comparison in this task is against other probabilistic prediction methods. Namely:

**MC dropout** fits a neural network to the dataset and interprets Bernoulli dropout as a variational approximation for Bayesian inference, obtaining predictive uncertainty by integrating over Monte Carlo samples (Gal and Ghahramani, 2016). We use the results from Gal and Ghahramani (2016) as our benchmark.

**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 (Lakshminarayanan et al., 2017). We use the results from Lakshminarayanan et al. (2017) as our benchmark.

**Concrete Dropout** improves upon MC dropout by employing a continuous relaxation of the Bernoulli distribution to automatically tune the dropout probability (Gal et al., 2017). We use the results from Gal et al. (2017) as our benchmark.

**Gaussian Processes** are a nonparametric Bayesian method where the response is interpreted as a multivariate Gaussian

*Table 3.* Comparison of point-estimation performance on regression benchmark UCI datasets as measured by RMSE. Although not optimized for point estimation, NGBoost still offers competitive performance. Bolding is as in Table 1.

| Dataset | $N$ | NGBoost | Elastic Net | Random Forest | Gradient Boosting | GAMLSS | Distributional Forest |
|---|---|---|---|---|---|---|---|
| Boston | 506 | **2.94 ± 0.53** | 4.08 ± 0.16 | 2.97 ± 0.30 | **2.46 ± 0.32** | 4.32 ± 1.40 | 3.99 ± 1.13 |
| Concrete | 1030 | **5.06 ± 0.61** | 12.1 ± 0.05 | 5.29 ± 0.16 | **4.46 ± 0.29** | 6.72 ± 0.59 | 6.61 ± 0.83 |
| Energy | 768 | 0.46 ± 0.06 | 2.75 ± 0.03 | 0.52 ± 0.09 | **0.39 ± 0.02** | 1.43 ± 0.32 | 1.11 ± 0.27 |
| Kin8nm | 8192 | 0.16 ± 0.00 | 0.20 ± 0.00 | 0.15 ± 0.00 | **0.14 ± 0.00** | 0.20 ± 0.01 | 0.16 ± 0.00 |
| Naval | 11934 | **0.00 ± 0.00** | **0.00 ± 0.00** | **0.00 ± 0.00** | **0.00 ± 0.00** | **0.00 ± 0.00** | **0.00 ± 0.00** |
| Power | 9568 | 3.79 ± 0.18 | 4.42 ± 0.00 | 3.26 ± 0.03 | **3.01 ± 0.10** | 4.25 ± 0.19 | 3.64 ± 0.24 |
| Protein | 45730 | 4.33 ± 0.03 | 5.20 ± 0.00 | **3.60 ± 0.00** | 3.95 ± 0.00 | 5.04 ± 0.04 | 3.89 ± 0.04 |
| Wine | 1588 | 0.63 ± 0.04 | 0.58 ± 0.00 | **0.50 ± 0.01** | 0.53 ± 0.02 | 0.64 ± 0.04 | 0.67 ± 0.05 |
| Yacht | 308 | **0.50 ± 0.20** | 7.65 ± 0.21 | 0.61 ± 0.08 | **0.42 ± 0.09** | 8.29 ± 2.56 | 4.19± 0.92 |
| Year MSD | 515345 | 8.94 ± NA | 9.49 ± NA | 9.05 ± NA | **8.73 ± NA** | NA ± NA | NA ± NA |

distribution with covariance given by some kernel between covariates (Rasmussen and Williams, 2005). Our experiments used an automatic relevance detection kernel fit via gradient-based optimization of the marginal log-likelihood. Datasets with $N > 2000$ employed 1000 inducing inputs randomly chosen from the training set, with inducing points fit with variational inference as in Titsias (2009). All features and labels were standardized to zero-mean and unit variance for pre-processing. The standardized noise level was tuned via grid search for each dataset, with values ranging between 0.01 and 0.1.

**GAMLSS** uses generalized (parametric) linear models to fit each distributional parameter instead of boosting (Stasinopoulos et al., 2007). Responses were parameterized as Normal distributions $N(\mu, \sigma^2)$. The mean $\mu$ and log-std $\log \sigma$ were independently modeled as linear combinations of natural cubic splines of the covariates. No interaction terms were included. All features and labels were standardized to zero-mean and unit variance for pre-processing.

**Distributional Forests** use trees to estimate distributional parameters in each leaf, which are then averaged across the model (Schlosser et al., 2019). Responses were parameterized as Normal distributions $N(\mu, \sigma^2)$. The mean $\mu$ and log-std $\log \sigma$ were independently modeled using forests consisting of 200 trees and default hyper-parameters ($\sqrt{d}$ covariates sampled per split, minimum 20 examples for a split node, minimum 7 examples in a terminal node). All features and labels were standardized to zero-mean and unit variance for pre-processing.

Our probabilistic regression results are summarized in Table 1. Results for the Year MSD dataset are unavailable either because they were not reported or because the necessary computations for gradient-based optimization of hyper-parameters did not fit in memory.

### 4.2. Ablation

We compare the NLL of our full NGBoost algorithm on these data versus that of the following comparators, each tuned in the same fashion:

**2nd-Order** boosting is NGBoost using 2nd-order gradient descent instead of the natural gradient. This tests the added benefit of using the natural gradient vis-a-vis 2nd-order methods. Recent work has argued that the natural gradient improves training dynamics by approximating the Hessian used in 2nd-order methods (Martens, 2014). We use the "saddle-free" Newton-Raphson method of Dauphin et al. (2014) in our implementation of 2nd-order multiparameter boosting to provide a strong baseline.

**Multiparameter** boosting is NGBoost using the ordinary gradient. This tests the added benefit of using the natural gradient vis-a-vis the standard gradient, but still allows for all of the parameters of the distribution to vary across $x$.

**Homoscedastic** boosting is NGBoost assuming a homoscedastic variance $\sigma^2(x) = \sigma^2 = \widehat{\text{Var}[r]}$ where $r$ are the training set residuals from a single-parameter (mean) boosting model. This tests the added benefit of allowing parameters other than the conditional mean to vary across $x$. Note that the natural gradient plays no meaningful role when there is only a single parameter estimated with NGBoost.

Our ablation results are summarized in Table 2.

### 4.3. Point estimation

Although NGBoost is not specifically designed for point estimation, it is easy to extract point estimates of expectations $\hat{\mathbb{E}}[y|x]$ from the estimated distributions $\hat{P}_\theta(y|x)$. We use this approach in a third evaluation to compare the same NGBoost models as above to the Scikit-Learn implementations of random forests, standard gradient boosting, and elastic net regression (Pedregosa et al., 2011). Predictive performance in this evaluation is captured by the root mean

squared-error (RMSE) of the predictions on the test set. We performed hyperparameter tuning for each of the comparator methods using the same validation procedure as described above, although optimizing for RMSE instead of NLL for all methods except NGBoost. We compare to:

**Elastic Net**: We used the Scikit-Learn implementation of elastic net regularized linear models. We tuned over lasso-ridge mixture parameters of 0.01, 0.7, and 0.99 and over a range of regularization parameters between 0.00005 and 0.01. All other parameters were left to their default values.

**Random Forest**: We used the Scikit-Learn implementation of random forests. We set the number of trees to 500 and left all other parameters at their default values.

**Gradient Boosting**: We used the Scikit-Learn implementation of gradient boosted trees. We tuned over learning rates of 0.01, 0.05, and 0.1, tree depths of 3 and 4, and number of boosting iterations between 0 and 1000. All other parameters were left to their default values.

Our point prediction results are summarized in Table 3.

## 5. Conclusions

NGBoost is a method for probabilistic prediction with competitive 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 performs as well as existing methods for probabilistic regression but retains major advantages: NGBoost is flexible, scalable, and easy-to-use. We have not rigorously quantified these advantages in this paper (since they would be largely irrelevant without first establishing performance), but many of the benefits are self-evident. Unlike problem-specific approaches, NGBoost handles classification, regression, survival problems, etc. using the same software package and interface. NGBoost scales to large numbers of features or observations with the same favorable complexity of traditional boosting algorithms. No expert knowledge of deep learning, Bayesian statistics, or Monte Carlo methods is required to use NGBoost. It works out of the box.

Our ablation experiments demonstrate that multiparameter boosting and the natural gradient work together to improve performance. Assuming a uniform variance across all covariates works reasonably well for some datasets, as would be expected, but this is not always the case. However, using multiparameter boosting to relax the homoscedasticity assumption most often results in worse performance, likely due to poor training dynamics. 2nd-order methods result in even worse performance. NGBoost employs the natural gradient to correct the training dynamics of multiparameter boosting. The superiority to 2nd-order methods demonstrates that this is due to exploiting the curvature of the score in distributional space, not the curvature of the score in parameter space. The result is performance that is almost always better than assuming homoscedasticity, sometimes by a large margin.

Furthermore, the advantages of probabilistic regression come almost "for free". On point estimation tasks NGBoost performs better than elastic net, about on par with random forests, and within striking distance of gradient boosting. This is despite the fact that the NGBoost models were (a) optimized for NLL, not to minimize RMSE and (b) less aggressively tuned. Thus, although point prediction will always be best with a dedicated model for that purpose, the loss in RMSE is not substantial if NGBoost is used in order to support probabilistic regression instead.

There are many avenues for future work. This paper is focused on regression problems for clarity of exposition, but NGBoost is also applicable to classification and to survival problems with right-censored data (using the censored likelihood as a scoring rule). NGBoost could also be used for *joint* prediction: by modeling two outcomes $z$ and $y$ with a jointly parameterized conditional distribution $P_\theta(z, y|x)$, a single NGBoost model could answer any question like "what is the probability that it rains more than 4 inches *and* the temperature is greater than 17°C tomorrow?".

Some further technical innovations are also worth exploring. 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 also likely to improve performance, especially in terms of scaling to large datasets.

Although we have shown empirically that NGBoost is useful for probabilistic prediction, it remains to be seen whether it is useful for inference problems and under what assumptions. For instance, if we assume that $y|x \sim \mathcal{D}(\theta(x))$ for some distribution $\mathcal{D}$ with parameters $\theta$ and we estimate $\hat{\theta}_{\text{ngb}}(x)$ using NGBoost, under what conditions do we have that $\hat{\theta}_{\text{ngb}}(x) \to \theta(x)$ as sample size increases? Are there conditions where the convergence is uniform in $x$? If the model is misspecified (i.e. $\mathcal{D}$ is not correct), are there conditions under which moment estimates from the model are still consistent? Addressing these questions and others like them would be of significant value.

### Acknowledgements

This work was funded in part by the National Institutes of Health. We thank anonymous reviewers for feedback.

## 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 Calibrated 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.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight Uncertainty in Neural Network. In *International Conference on Machine Learning*, pages 1613–1622.

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.

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.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2933–2941. Curran Associates, Inc.

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.

Gal, Y., Hron, J., and Kendall, A. (2017). Concrete Dropout. 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 3581–3590. Curran Associates, Inc.

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

Graves, A. (2011). Practical Variational Inference for Neural Networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc.

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.

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.

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

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.

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.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P.,

Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Schlosser, L., Hothorn, T., Stauffer, R., and Zeileis, A. (2019). Distributional regression forests for probabilistic precipitation forecasting in complex terrain. *The Annals of Applied Statistics*, 13(3):1564–1589. Publisher: Institute of Mathematical Statistics.

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

Stasinopoulos, D. M., Rigby, R. A., et al. (2007). Generalized additive models for location scale and shape (gamlss) in r. *Journal of Statistical Software*, 23(7):1–46.

Titsias, M. (2009). Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574.