

METHODS PAPER  

Probabilistic Prediction of Oceanographic Velocities with Multivariate Gaussian Natural Gradient Boosting

Michael O'Malley¹ , Adam M. Sykulski² , Rick Lumpkin³ and Alejandro Schuler⁴

¹STOR-i Centre for Doctoral Training, Department of Mathematics and Statistics, Lancaster University, Lancaster, United Kingdom

²Department of Mathematics, Imperial College London, London, United Kingdom

³Physical Oceanography Division, NOAA/Atlantic Oceanographic and Meteorological Laboratory, Miami, FL, USA

⁴Center for Targeted Machine Learning, University of California, Berkeley, Berkeley, CA, USA

Corresponding author: Michael O'Malley; Email: michael.omalley1011@gmail.com

Received: 22 December 2021; **Revised:** 10 March 2023; **Accepted:** 22 March 2023



Keywords: Conditional density estimation; multi-output regression; multivariate Gaussian; ocean currents

Abstract

Many single-output regression problems require estimates of uncertainty along with the point predictions. For this purpose, there exists a class of regression algorithms that predict a conditional distribution rather than a point estimate. The off-the-shelf options are much more limited, however, when the prediction output is multivariate and a joint measure of uncertainty is required. In this paper, we predict a distribution around a multivariate random vector of dimension P , such that the joint uncertainty would quantify the probability of any vector in P -dimensional space. This is more expressive than providing separate uncertainties in each dimension. To enable joint probabilistic regression, we propose a natural gradient boosting approach based on nonparametrically modeling the conditional parameters of the multivariate predictive distribution, where we focus on the multivariate Gaussian distribution. Our method is robust, can be easily trained without extensive tuning, and performs competitively in comparison to existing approaches. The motivating application of our methodology is to predict two-dimensional oceanographic currents measured by freely floating Global Drifter Program drifters using remotely sensed data. We also demonstrate the method's performance on simulated data. We find this method excels when strong correlation between output dimensions is present. As part of this work, we have added the model to the open source package at github.com/stanfordmlgroup/ngboost.

Impact Statement

This paper develops a general multivariate probabilistic regression algorithm to quantify joint uncertainty in multi-output regression. The usefulness of this method comes up in many areas of environmental data science such as oceanography, weather modeling, and environmental epidemiology. This paper demonstrates the performance of our algorithm on an application where the goal is to predict two-dimensional velocities measured by freely floating ocean drifters using remotely sensed wind and geostrophic velocity data. We show that this method is particularly beneficial when strong correlation between the prediction output dimensions is present. We provide the method in a Python package for ease of use in other applications.

  This research article was awarded Open Data and Open Materials badges for transparent practices. See the Data Availability Statement for details.

© The Author(s), 2023. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

1. Introduction

The motivation for this paper is to predict two-dimensional velocity outputs, commonly studied in environmental applications such as ocean currents (Maximenko et al., 2009; Sinha and Abernathey, 2021) and wind velocities (Lei et al., 2009). In such applications, independence between the dimensions being predicted is often assumed, but will seldom be the case in practice, and this independence assumption becomes even more unrealistic when uncertainty estimates are given along with point predictions. Motivated by these challenges, in this paper, we propose a methodology for multivariate prediction in any dimension that allows for dependence between each output dimension, and we provide a real-world large scale example of our method by predicting two-dimensional ocean currents, and their associated uncertainty, using remotely sensed satellite data. Ocean current models are used in practice for applications such as predicting the locations of debris, plastics, and oil spills. By modeling the joint two-dimensional stochasticity of directional currents more accurately, this will translate to better forecasts, decisions, and better measures of associated uncertainty.

The standard regression problem is to predict the value of a continuous random variable \mathbf{Y} based on the values of an observed set of features \mathbf{X} . Under mean squared error loss, this is equivalent to estimating the conditional mean $\mathbb{E}[\mathbf{Y}|\mathbf{X}]$. Often, however, the user is also interested in a measure of predictive uncertainty about that prediction, or even the probability of observing any particular value of the output. In other words, one seeks to estimate the conditional distribution $p(\mathbf{Y}|\mathbf{X})$. When we predict the distribution rather than the point estimate this is called *probabilistic* regression (Duan et al., 2020). In this work, we focus on probabilistic regression where the output \mathbf{Y} is vector valued.

A common approach for flexible probabilistic regression is to use a standard multi-output regression model to parameterize a probability distribution function, then optimize the log-likelihood. Commonly this approach uses a neural network as the multi-output model (Williams, 1996; Sützle and Hrycej, 2005; Rasp and Lerch, 2018). Recently, Duan et al. (2020) proposed an algorithm called natural gradient boosting (NGBoost) which uses a gradient boosting-based learning algorithm to fit each of the parameters in a distribution using an ensemble of weak learners. A notable advantage of this approach is that it does not require extensive tuning to attain state-of-the-art performance. As such, NGBoost works out-of-the-box and is accessible without machine-learning expertise.

Our contribution in this paper is to construct and test a similar method that works for multivariate probabilistic regression. We will achieve this by extending the NGBoost Duan et al. (2020) framework to fit the parameterization of the multivariate Gaussian used by Williams (1996). A common approach for multi-output regression is to either fit each dimension entirely separately with a univariate model or assume that the residual errors in both output dimensions are independent (Segal and Xiao, 2011; Borchani et al., 2015), allowing the practitioner to factor the objective function. In reality, however, output dimensions are often highly correlated if they can be predicted using the same input data. When the task requires a measure of uncertainty, we must therefore model the *joint* uncertainty in the predictions.

In summary, in this paper, we present an NGBoost algorithm that allows for multivariate probabilistic regression. Our key innovation is to place multivariate probabilistic regression in the NGBoost framework, which allows us to model all parameters of a *joint* distribution with flexible regression models. At the same time, we inherit the ease-of-use and performance of the NGBoost framework. In particular, we demonstrate the use of a NGBoost to predict a multivariate Gaussian distribution, and we evaluate the performance in a simulation study and in the oceanographic use case described above. The results show that: (a) joint multivariate probabilistic regression is more effective than naively assuming independence between the outcomes, and (b) our multivariate NGBoost approach outperforms a state-of-the-art neural network approach. We also provide a detailed analysis in the oceanographic use case where we compare a model that assumes independence in the output, with the method we have developed in this work. In this analysis, we find that the new NGBoost algorithm excels in data dense regions where a strong correlation exists between output dimensions. The code developed to produce the results we present in this paper is freely available as part of the `ngboost` Python package.

1.1. Notation

Let the input space be denoted as $\mathcal{X} = \mathbb{R}^d$, and let the output space be denoted as $\mathcal{Y} = \mathbb{R}^P$. The aim of this paper is to learn a conditional distribution from the training dataset $\mathcal{D} = \{\mathbf{X}_i, \mathbf{Y}_i | 1 \leq i \leq N, \mathbf{X}_i \in \mathcal{X}, \mathbf{Y}_i \in \mathcal{Y}\}$. Let $p(\mathbf{Y}|\boldsymbol{\theta})$ be a probability density of the output $\mathbf{Y} \in \mathcal{Y}$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^M$, where M is the dimension of the parameters. In this work, we aim to learn a mapping such that the parameter vector varies for each data point, that is, a function $f : \mathcal{X} \rightarrow \mathbb{R}^M$, inducing the distribution function of \mathbf{Y} , $p(\mathbf{Y}|\boldsymbol{\theta} = f(\mathbf{X}))$. For convenience, we shall henceforth adopt a shorter notation: $p(\mathbf{Y}|\boldsymbol{\theta} = f(\mathbf{X})) = p(\mathbf{Y}|\mathbf{X})$.

2. Background

2.1. Natural gradient boosting

NGBoost (Duan et al., 2020) is a method for probabilistic regression. The approach is based on the well-proven approach of boosting (Friedman, 2001) with two main differences. First, rather than attempting to produce a single-point estimate for $\mathbb{E}[\mathbf{Y}|\mathbf{X}]$, NGBoost aims to fit parameters for a pre-specified probability distribution function with M unknown parameters, in turn producing a prediction for $p(\mathbf{Y}|\mathbf{X})$. Second, in place of the ordinary gradient, the natural gradient is used instead (Amari, 1998).

NGBoost learns this relationship by fitting a base learner $f^{(b)} : \mathcal{X} \rightarrow \mathbb{R}^M$ at each boosting iteration. The weighted sum of these base learners constitutes the function f . Each output of f is used to parameterize the pre-specified distribution. After $B \in \mathbb{N}$ boosting iterations, then the final sum of base learners specifies a conditional distribution:

$$p\left(\mathbf{Y}|\boldsymbol{\theta}(\mathbf{X}) = \left\{ \boldsymbol{\theta}^{(0)} - \eta \sum_{b=1}^B \rho^{(b)} f^{(b)}(\mathbf{X}) \right\}\right),$$

where $\rho^{(b)} \in \mathbb{R}, b \in \{1, \dots, B\}$ are scalings chosen by a line search, $\boldsymbol{\theta}^{(0)}$ is the initialization of the parameters, and η is the fixed learning rate.

For the base learner f , the requirement is that it is a function which maps the features \mathbf{X} to a value in \mathbb{R}^M . In NGBoost, the standard base learner is a regression tree which is what we use here. We fit M independent decisions trees, each tree being one of the entries in the vector-valued output.

The base learner at each iteration is fit to approximate the functional gradient of a scoring rule (the probabilistic regression equivalent of a loss function). Throughout this paper, we use the log score which is more commonly referred to as the log-likelihood, hence the optimization procedure is maximum likelihood estimation; we note that alternatives exist such as those introduced in Gneiting and Raftery (2007). For shorthand in the following explanation, we denote the log-likelihood of \mathbf{Y} evaluated at $\boldsymbol{\theta}$ as $l(\boldsymbol{\theta}) (= \log p(\mathbf{Y}|\boldsymbol{\theta}))$. Multi-parameter boosting aims to form a prediction of the value of the functional gradient $\nabla l(\boldsymbol{\theta})$ from the features \mathbf{X}_i using a standard regression algorithm. In effect, each base learner represents a single gradient descent step. However, this approach by itself is not sufficient to attain good performance in practice. To solve problems with poor training dynamics, NGBoost uses the natural gradient (Amari, 1998) in place of the ordinary gradient. This is particularly advantageous for fitting probability distributions as discussed in Duan et al. (2020).

To calculate the natural gradient, we must pre-multiply the gradient of the log-likelihood by the inverse of the expected Fisher information $\mathcal{I}(\boldsymbol{\theta})$ evaluated at the current point $\boldsymbol{\theta}$ in the parameter space. The expected Fisher information is computed as the expectation of the Hessian matrix:

$$\mathcal{I}(\boldsymbol{\theta})_{ij} = -\mathbb{E} \left[\frac{\partial^2 l}{\partial \theta_i \partial \theta_j} \right], \quad i, j \in \{1, \dots, M\},$$

which is valid when the likelihood l is twice differentiable and under certain regularity conditions. The natural gradient is calculated as

$$\tilde{\nabla}l(\boldsymbol{\theta}) = \mathcal{I}(\boldsymbol{\theta})^{-1} \nabla l(\boldsymbol{\theta}). \quad (1)$$

One of the contributions of this paper is that we provide the calculations $\mathcal{I}(\boldsymbol{\theta})$ for a specific parameterization of the multivariate Gaussian which will be discussed in Section 3.1. For reference, we give pseudo-code of NGBoost in Algorithm 1.

Algorithm 1. NGBoost for probabilistic prediction. Adapted from Duan et al. (2020).

Data: Dataset $\mathcal{D} = \{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^N$.

Input: Boosting iterations B , Learning rate η , Probability distribution with parameter $\boldsymbol{\theta}$, Base learner f .

Output: Scalings and base learners $\{\rho^{(b)}, f^{(b)}\}_{b=1}^B$.

$\boldsymbol{\theta}_i^{(0)} \leftarrow \arg \min_{\boldsymbol{\theta}} \sum_{k=1}^N p(\mathbf{Y}_k | \boldsymbol{\theta})$ {Initialize to marginal. Initial value is constant for all i .}

for $b \leftarrow 1, \dots, B$ **do**

 {Store the $N \times M$ values of natural gradients.}

for $i \leftarrow 1, \dots, N$ **do**

 | $g_i^{(b)} \leftarrow \tilde{\nabla} \log p(\mathbf{Y}_i | \boldsymbol{\theta}_i^{(b-1)})$

end

 {Fit the base learner to predict the natural gradient $g_i^{(b)}$.}

$f^{(b)} \leftarrow \left(\text{fit base learner} \left\{ f(\mathbf{X}_i), g_i^{(b)} \right\}_{i=1}^N \right)$

 {Find the optimal step size.}

$\rho^{(b)} \leftarrow \arg \min_{\rho} \sum_{i=1}^N p\left(\mathbf{Y}_i | \boldsymbol{\theta}_i^{(b-1)} - \rho \cdot f^{(b)}(\mathbf{X}_i)\right)$

 {Finally update $\boldsymbol{\theta}$.}

for $i \leftarrow 1, \dots, N$ **do**

 | $\boldsymbol{\theta}_i^{(b)} \leftarrow \boldsymbol{\theta}_i^{(b-1)} - \eta (\rho^{(b)} \cdot f^{(b)}(\mathbf{X}_i))$

end

end

2.2. Related works

Probabilistic regression has been approached in multiple ways: for example, deep distribution regression (Li et al., 2021), quantile regression forests (Meinshausen and Ridgeway, 2006), and generalized additive models for *location, shape, scale* (GAMLSS; Rigby, 2019). All the listed methods focus on producing some form of outcome distribution. Both Li et al. (2021) and Meinshausen and Ridgeway (2006) rely on binning the output space, an approach which will work poorly in higher output dimensions.

NGBoost is a method closer to Rigby (2019), where we prespecify a form for the desired distribution, and then fit a model to predict the parameters which quantify this distribution. Such a parametric approach allows one to specify a full multivariate distribution while keeping the number of outputs which the learning algorithm needs to predict relatively small. This is in contrast to deep distribution regression (Li et al., 2021), for example, where the number of outputs which a learning algorithm needs to predict becomes infeasibly large as the dimension of \mathbf{Y} grows. This highlights the trade-off between the approaches: by specifying a distribution priori as in GAMLSS and NGBoost, we make the problem tractable; however, this specification comes at a cost of flexibility in the distribution we predict.

Williams (1996) proposes a neural network-based estimating the conditional density of multi-output problems using a multivariate Gaussian distribution. This neural network method is directly comparable to what we aim to achieve here and motivates the approach we will be taking in Section 3.1. The model is specified such that the output of the neural network is the M parameters which are used to specify a multivariate Gaussian distribution, that is, $\theta(\mathbf{X})$ in Section 2.1 is a neural network rather than a weighted sum of regression trees. One minor difference is that methods which use neural networks for conditional density estimation (Bishop, 1994; Williams, 1996) generally fit a single model with M outputs; whereas in NGBoost, the model for $\theta(\mathbf{X})$ could easily be factorized to be written as M independent models.

Gaussian processes (Williams and Rasmussen, 2006) may appear at first glance very similar to the approach which we will take here. In the context of Gaussian processes, multiple output regression has been studied extensively (Álvarez et al., 2010; Alvarez et al., 2012; Liu et al., 2018). Multi-output Gaussian processes, however, are not comparable to what we aim to achieve in this paper. In particular, we want the property that the covariance between output dimensions is heterogeneous, that is, it changes over the input space \mathbf{X} . As an example, the linear model of a coregionalization multi-output approach would be to model:

$$\mathbf{Y} \sim W\mathbf{f}(\mathbf{X}),$$

where W is a $P \times L$ shaped matrix of mixing weights and $\mathbf{f}(\mathbf{X})$ is an $L \times 1$ vector, each coordinate being an independent Gaussian process. The covariance is modeled through the mixture W , which is not a function of \mathbf{X} . Therefore, the covariance between output dimensions is not modeled as a function of \mathbf{X} . Whereas the approach we propose would allow the correlation between output dimensions to vary over space.

Developments in multi-output Gaussian processes also naturally work with data where for some rows i , only a subset of \mathbf{Y}_i needs to be observed. Moreno-Muñoz et al. (2018) focuses on heterogeneous modeling, in the sense that dimensions do not have the same data type (e.g., $\mathbf{Y}_{i,0}$ could follow a Gaussian distribution and $\mathbf{Y}_{i,1}$ could follow a Bernoulli distribution). This would not easily be handled by the NGBoost framework which we use here.

There are also a number of methods in Bayesian uncertainty quantification which aim to quantify uncertainty (Abdar et al., 2021), such as Monte Carlo Dropout (Gal and Ghahramani, 2016), the aforementioned Gaussian processes (Williams and Rasmussen, 2006; Alvarez et al., 2012; Moreno-Muñoz et al., 2018), and Bayesian neural networks (Blundell et al., 2015; Kendall and Gal, 2017). These approaches aim to quantify the posterior distribution or posterior predictive distribution, whereas probabilistic regression generally focuses on fitting a conditional distribution, without a measure of uncertainty on the fitted conditional distribution. These two approaches to uncertainty have trade-offs, generally the barrier to Bayesian uncertainty quantification is the computational cost and complex inference procedures. As an example, compare the fitting of Bayesian additive regression trees (Chipman et al., 2010), to fitting a similar model with gradient boosting (Friedman, 2001), the latter is considerable quicker and easier to implement. Here, we focus on non-Bayesian probabilistic regression as in Rigby (2019), Duan et al. (2020), and Li et al. (2021).

3. Multivariate Gaussian NGBoost

The original NGBoost paper of Duan et al. (2020) briefly noted that NGBoost could be used to jointly model multivariate outcomes, but did not provide details. Here, we show how NGBoost extends to multivariate outcomes and provide a detailed investigation of one useful parametrization, namely, the multivariate Gaussian.

In univariate NGBoost ($P=1, Y \in \mathbb{R}$), the predicted distribution is parametrized with $p(Y|\{\theta_m = f_m(\mathbf{X})\})$, where Y is a univariate outcome. For example, $Y|\mathbf{X}$ may be assumed to follow a univariate Gaussian distribution where μ and $\log \sigma$ are taken to be the parameter vector θ (i.e., the output of NGBoost is $\theta(\mathbf{X}) = (\mu(\mathbf{X}), \log \sigma(\mathbf{X}))$). Therefore, to model a *multivariate* outcome, all that is necessary is to specify a parametric distribution that has multivariate support, as we shall now show.

3.1. Multivariate Gaussian

The multivariate Gaussian is a commonly used generalization of the univariate Gaussian. This generalization allows us to define a joint distribution on vector-valued data. The multivariate Gaussian distribution is commonly written in the *moment* parameterization as

$$\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where \mathbf{Y} is a $P \times 1$ vector, $\boldsymbol{\Sigma}$ is the covariance matrix (a $P \times P$ positive semidefinite matrix), and $\boldsymbol{\mu}$ is the mean vector (a $P \times 1$ vector). The off-diagonal elements, $\boldsymbol{\Sigma}_{k,q}$, $k \neq q$ measures the degree to which dimensions k and q of \mathbf{Y} vary with each other. Often this quantity is reported as a *correlation coefficient*:

$$\rho_{kq} = \frac{\boldsymbol{\Sigma}_{k,q}}{\sqrt{\boldsymbol{\Sigma}_{k,k}\boldsymbol{\Sigma}_{q,q}}}.$$

By putting the multivariate Gaussian within the scope of NGBoost, we enable the modeling of multivariate data where this correlation between dimensions can be predicted as a function of \mathbf{X} .

In particular, this choice of distribution is motivated by the core application of this paper. Ocean currents are often reported and modeled in standard axes—longitudinal and latitudinal flows. In practice, the flow components in each direction are highly correlated and that correlation changes over space. Modeling the correlation between these axes is achievable with the multivariate Gaussian distribution hence our choice.

The rest of this paper is focused on the development and evaluation of a probabilistic regression algorithm using the multivariate Gaussian. We achieve this by placing the multivariate Gaussian within the NGBoost framework introduced in Section 2.1. For the application study in this paper, we are primarily interested in $P = 2$, however, all derivations are given keeping P general so this work can be used for any output dimension. The only constraint is that all output dimensions must be real valued.

3.2. Implementation details

Note that we only consider positive definite matrices for $\boldsymbol{\Sigma}$ to ensure the inverse exists. We write the probability density function as

$$p(\mathbf{Y}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{|\boldsymbol{\Sigma}|^{-\frac{1}{2}}}{(2\pi)^{\frac{P}{2}}} \exp\left[-\frac{(\mathbf{Y}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{Y}_i - \boldsymbol{\mu})}{2}\right]. \tag{2}$$

We fit the parameters of this distribution conditional on the corresponding training data \mathbf{X}_i such that $p(\mathbf{Y}_i|\mathbf{X}_i) = p(\mathbf{Y}_i|\boldsymbol{\mu}(\mathbf{X}_i), \boldsymbol{\Sigma}(\mathbf{X}_i))$. To perform unconstrained gradient-based optimization for any distribution, we must have a parameterization for the multivariate Gaussian distribution where all parameters lie on the real line. The mean vector $\boldsymbol{\mu}$ already satisfies this. However, the covariance matrix does not; it lies in the space of positive definite matrices. We shall model the inverse covariance matrix which is also constrained to be positive definite. We leverage the fact that every positive definite matrix can be factorized using the Cholesky decomposition with positive entries on the diagonals (Banerjee and Roy, 2014).

We opt to use an upper triangular representation of the square root of the inverse covariance matrix $\boldsymbol{\Sigma}^{-1} = \mathbf{U}^T \mathbf{U}$, as used by Williams (1996), where the diagonal is transformed using an exponential to force the diagonal to be positive. As an example, in the two-dimensional case, we have

$$\mathbf{U} = \begin{bmatrix} \exp(a_{11}) & a_{12} \\ 0 & \exp(a_{22}) \end{bmatrix}, \tag{3}$$

which yields the inverse covariance matrix

$$\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \exp(a_{11})^2 & \exp(a_{11})a_{12} \\ \exp(a_{11})a_{12} & a_{12}^2 + \exp(a_{22})^2 \end{bmatrix}.$$

This parameterization for Σ^{-1} ensures that the resulting covariance matrix $\Sigma = (\mathbf{U}^T \mathbf{U})^{-1}$ is positive definite for all $a_{ij} \in \mathbb{R}$. Therefore, we can fit the multivariate Gaussian in an unconstrained fashion using the parameter vector $\theta = (\mu_1, \mu_2, a_{11}, a_{22}, a_{12})$ as the output in the two-dimensional case. Note that the number of parameters grows quadratically with the dimension of the data. Specifically, the relation between M , the dimension of $\theta \in \mathbb{R}^M$, and P , the dimension of \mathbf{Y}_i is

$$M = \frac{P^2 + 3P}{2}. \tag{4}$$

For NGBoost using the log-likelihood scoring rule, we require both the gradient and the Fisher information. The gradient calculations are given in Williams (1996), and the derivations for the Fisher information are given in Appendix B. We have used these derivations to add the multivariate Gaussian distribution to the open-source Python package ngboost as part of the contributions of this paper. Note that the natural gradient is particularly advantageous for multivariate problems such as these. This is because, for example, there are multiple equivalent parameterizations for the multivariate Gaussian distribution (Sützle and Hrycej, 2005; Malagò and Pistone, 2015; Salimbeni et al., 2018), but the choice of parameterization has been shown to be less important when using natural gradients than with classical gradients (see, e.g., Salimbeni et al., 2018).

3.3. Practical adjustment for singular inverse covariance matrices

A small adjustment is made to \mathbf{U} to improve numerical stability in the implementation. In particular, if the diagonal entries of \mathbf{U} from equation (3) ($\exp(a_{ii})$) get close to zero the resulting matrix Σ^{-1} has a determinant close to zero, causing numerical problems to occur when inverting Σ^{-1} . The fix we propose for this is to add a small perturbation of 10^{-6} to the diagonal elements of \mathbf{U} . For most datasets, once the model is fitted, the predicted value for the diagonal elements will be much larger than 10^{-6} so this slight adjustment has a negligible effect on the predictions. This small perturbation is fixed as constant and used throughout this work.

This practical adjustment will become an issue in cases where the determinant of the covariance matrix is very large. As an extreme example, suppose the predicted covariance without the adaptation is $\Sigma = \mathbb{I}_2 \times 10^{12}$, where \mathbb{I}_2 is the identity matrix. The practical adjustment results in a marginal variance four times larger in this case, which is clearly not negligible. Problems like this can typically be mitigated by scaling the output data \mathbf{Y} (e.g., using a min-max scaling strategy).

4. Simulation

We now demonstrate the effectiveness of our multivariate Gaussian NGBoost algorithm in simulation. Specifically, we show that (a) it outperforms a naive baseline where each dimension of \mathbf{Y} is modeled independently, (b) it outperforms a state-of-the art neural network approach, and (c) the natural gradient is a key component in the effective training of distributional boosting models in the multivariate setting.

We simulate the data similarly to Williams (1996). The nature of the simulation tests each algorithm’s ability to uncover nonlinearities in each of the distributional parameter’s relationship with the input. Specifically, we use a one-dimensional input and two-dimensional output, allowing us to illustrate the fundamental benefits of our approach in even the simplest multivariate extension. The data are simulated as follows:

$$\begin{aligned} \mathbf{X}_i &\stackrel{\text{iid}}{\sim} \text{Uniform}(0, \pi) \quad i \in \{1, \dots, N\} \\ \mathbf{Y}_i | \mathbf{X}_i &\sim \mathbb{N} \left(\begin{bmatrix} \mu_1(\mathbf{X}_i) \\ \mu_2(\mathbf{X}_i) \end{bmatrix}, \begin{bmatrix} \sigma_1^2(\mathbf{X}_i) & \sigma_1(\mathbf{X}_i)\sigma_2(\mathbf{X}_i)\rho(\mathbf{X}_i) \\ \sigma_1(\mathbf{X}_i)\sigma_2(\mathbf{X}_i)\rho(\mathbf{X}_i) & \sigma_2^2(\mathbf{X}_i) \end{bmatrix} \right) \end{aligned}$$

with the following functions:

$$\begin{aligned}
 \mu_1(x) &= \sin(2.5x) \sin(1.5x) + x, \\
 \mu_2(x) &= \cos(3.5x) \cos(0.5x) - x^2, \\
 \sigma_1^2(x) &= 0.01 + 0.25[1 - \sin(2.5x)]^2, \\
 \sigma_2^2(x) &= 0.01 + 0.25[1 - \cos(3.5x)]^2, \\
 \rho(x) &= \sin(2.5x) \cos(0.5x).
 \end{aligned}
 \tag{5}$$

Simulated data alongside the true parameters are shown in Figure 1.

We compare multiple methods, all of which predict a multivariate Gaussian. For all boosting models, we use scikit-learn decision trees (Pedregosa et al., 2011) as the base learner. Specifically, we consider five different comparison methods:

- *NGB*: The method proposed in this paper: NGBoost to fit a multivariate Gaussian distribution.
- *Indep NGB*: Independent NGBoost where a univariate Gaussian model is fitted for the two dimensions separately, that is, $\mathbf{Y}_i \sim (\mathcal{N}(\mu_1, \sigma_2), \mathcal{N}(\mu_2, \sigma_2))$. Early stopping allows the number of trees used to predict each dimension to differ.
- *skGB*: Scikit-learn's (Pedregosa et al., 2011) implementation of gradient boosting (skGB) is used as a point prediction approach. To turn the skGB predictions into a multivariate Gaussian, we estimate a constant diagonal covariance matrix based on the residuals of the training data. For metric computation, we assume that each \mathbf{Y}_i follows a multivariate Gaussian with mean from a model fit for each dimension, and constant covariance matrix. Similar to *Indep NGB*, we allow a different number of trees for each dimension. This is the only model which does not have a covariance matrix varying over \mathbf{X} .

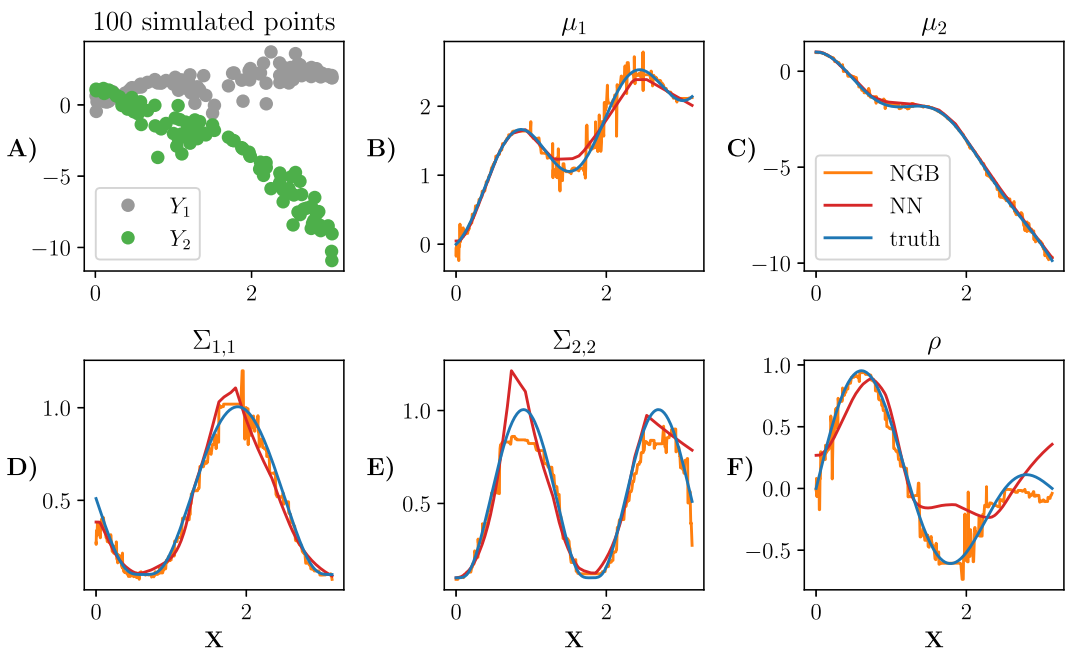


Figure 1. Simulated data from equation (5). A sample of 100 points is shown in panel (a). We plot each parameter in plots (b) to (f). The true parameters of the distribution are shown in blue, the NGBoost fit is shown in orange, and the neural network fit with one hidden layer (100 neurons in the hidden layer) is shown in red. Both model fits are trained on $N = 5,000$ training points.

- *GB*: A multi-parameter gradient boosting algorithm, where the only change from NGB is that the gradients are not multiplied by $\mathcal{I}(\theta)^{-1}$.
- *NN*: The neural network approach from Williams (1996) discussed in Section 2.2. We fit the model using tensorflow (Abadi et al., 2016). More details about the model and the grid search carried out on the network structure are explained in the [Supplementary Material](#).

To prevent overfitting, we employ early stopping for all methods on the held-out validation set, where we use a patience of 50 for all methods. For all boosting approaches, we use the estimators up to the best iteration that was found, and for neural networks we restore the weights back to the best epoch. The base learner for all boosting approaches are run using the default parameters specified in each package. For each N considered in the experiment, the neural network structure with the best log-likelihood metric averaged over all replications is chosen from the grid search. A learning rate of 0.01 is used for all methods, for NN, we use the Adam optimizer (Kingma and Ba, 2015).

We train the models used in the experiment with $N \in \{500, 1000, 3000, 5000, 8000, 10000\}$ with 50 replications for each value of N . For each simulation, N values were simulated as the training dataset. In addition to the N values, 300 values were simulated as the validation set for early stopping, and another 1000 values were used as the test set. We then use the Kullback–Leibler (KL) divergence from the true distribution at the test-set locations as the main comparison metric, however, we report extended results with extra metrics in the [Supplementary Material](#).

The per-model results on the test data points are shown in Table 1. The average KL divergence from the predicted distribution to true distribution is used as a metric. The results show that the NGB method is best for all N despite not being tuned in any way besides early stopping. The KL divergence converges quickly to 0 as N grows, showing that NGB gets better at learning the distribution as N increases. NN performs worse than NGB with higher KL divergence for all values of N . The KL divergence for Indep NGB seems to converge to a nonzero quantity as N increases, likely because the true distribution in equation (5) cannot be captured by a multivariate normal with a diagonal covariance matrix. skGB has large KL divergences which get worse as N increases, likely because of the homogeneous variance fit. We note that GB performs significantly worse than NGB showing that the natural gradient is necessary to fit the multivariate Gaussian effectively with boosting. Further metrics can be seen in the [Supplementary Material](#), in particular, they show that GB fits the mean μ very poorly, which is then compensated for by a large covariance estimate, explaining the large KL divergence.

The modification we made from the simulation in Williams (1996) is that we added x and $-x^2$ terms to $\mu_1(x)$ and $\mu_2(x)$, respectively, in equation (5). This modification is to highlight where our method excels and shows how robust the natural gradient methods are. We also ran the original simulation without the modification, where the results are given in the [Supplementary Material](#). As a summary, we note the two major differences: (a) The gaps between NGB, GB, and NN are much smaller without the modification,

Table 1. Average KL divergence in the test set (to three decimal places) from the predicted distribution to the true distribution as the number of training data points N varies.

N	NGB	Indep NGB	skGB	GB	NN
500	0.564 ± 0.016	1.633 ± 0.043	17.194 ± 0.301	126.227 ± 2.579	1.285 ± 0.547
1000	0.257 ± 0.004	1.150 ± 0.020	17.963 ± 0.270	114.113 ± 1.622	0.320 ± 0.018
3000	0.106 ± 0.002	0.884 ± 0.015	19.609 ± 0.248	97.682 ± 1.397	0.149 ± 0.004
5000	0.081 ± 0.008	0.878 ± 0.015	20.308 ± 0.169	90.101 ± 1.291	0.128 ± 0.005
8000	0.053 ± 0.004	0.866 ± 0.013	20.614 ± 0.170	79.006 ± 1.168	0.103 ± 0.004
10000	0.043 ± 0.001	0.831 ± 0.010	20.554 ± 0.150	74.799 ± 1.191	0.130 ± 0.004

Note. Standard error estimated from 50 replications reported after \pm to three decimal places. Lower values are better, the result with the lowest mean is bolded in each row. An extended table showing additional metrics is shown in the [Supplementary Material](#).

implying that the addition of x and $-x^2$ terms causes the GB method to perform much worse. (b) Without the modification, the NN method does best for $N \in \{500, 1000, 3000\}$, whereas NGB continues to perform best for $N \in \{5000, 8000, 10000\}$.

In simulation results in the [Supplementary Material](#), where x and $-x^2$ are not included in [equation \(5\)](#), GB achieves the lowest RMSE. This suggests that GB only has difficulty learning the mean when adaptation of x and $-x^2$ is included which explains the poor result from GB in [Table 1](#). As noted the only difference between NGB and GB is that we do not pre-multiply by the inverse Fisher information, hence, we can attribute this difference in performance to the natural gradient.

5. Ocean Drifters Application

The motivating application of this paper is to predict two-dimensional oceanographic velocities from satellite data on a large spatial scale. Typically, velocity prediction is done through physics-inspired parametric models fitted with least-squares or similar metrics, treating the directional errors as independent (Mulet et al., 2021). In this section, we shall instead focus on a data-driven approach utilizing a number of observational datasets.

5.1. Data

Here, we introduce the datasets used which shall define $\mathbf{Y}_i \in \mathbb{R}^2$, $\mathbf{X}_i \in \mathbb{R}^9$ ($d=9, P=2$). The data for all sources are available from 1992 to 2019 inclusive. All raw data sources are publicly available online, however, we also provide the processed ready to use dataset for future comparisons at <https://doi.org/10.5281/zenodo.5644972>.

For the model output \mathbf{Y}_i , we seek to predict two-dimensional oceanic near-surface velocities as functions of global remotely sensed climate data. The dataset used to train, validate and test our model comes from the Global Drifter Program, which contains transmitted locations of freely drifting buoys (known as drifters) as they drift in the ocean and measure near-surface ocean flow.

The quality controlled 6-hourly product is used (Lumpkin and Centurioni, 2019) to construct the velocity data points which will be the outputs we aim to predict. We drop drifter observations which have a high location noise estimate, and we apply a low-pass filter to the velocities at 1.5 times the inertial period (a timescale determined by the Coriolis effect), this preprocessing follows previous similar works (Laurindo et al., 2017). We only use data from drifters which still have a holey sock drogue, which is a drogue attached by a tether and is designed such that when attached the drifter will follow ocean near-surface flow. We use the inferred two-dimensional velocities of these drifters as our outputs \mathbf{Y}_i .

The longitude–latitude locations of these observations and the time of year (percentage of 365) are used as three of the nine features in \mathbf{X}_i to account for spatial and seasonal effects. For the remaining six features in \mathbf{X}_i , we use two-dimensional longitude–latitude measurements of *geostrophic velocity* (ms^{-1}), *surface wind stress* (Pa), and *wind speed* (ms^{-1}). These variables are used as they jointly capture geophysical effects known as geostrophic currents, Ekman currents, and wind forcing, which are known to drive oceanic near-surface velocities. We obtain geostrophic velocity and wind measurements from data products Thematic Assembly Centers (2020a) and Thematic Assembly Centers (2020b) respectively, where the data products are interpolated to the longitude–latitude locations of interest. We further preprocess the data as explained in [Appendix A](#).

To allow us to run multiple model fits, we subset the data to only include data points which are spatially located in the North Atlantic Ocean as defined by IHO marine regions (Flanders Marine Institute, 2018) and between $83^\circ W$ and $40^\circ W$ longitude. We also down-sample the data to daily intervals. Ultimately, this results in 414697 observations which we will use to compare the probabilistic regression models.

To give an insight into this dataset in [Figure 2](#), we show the five longest trajectories used in this analysis. In particular, one of the features to notice is that there are areas where the sampled points are

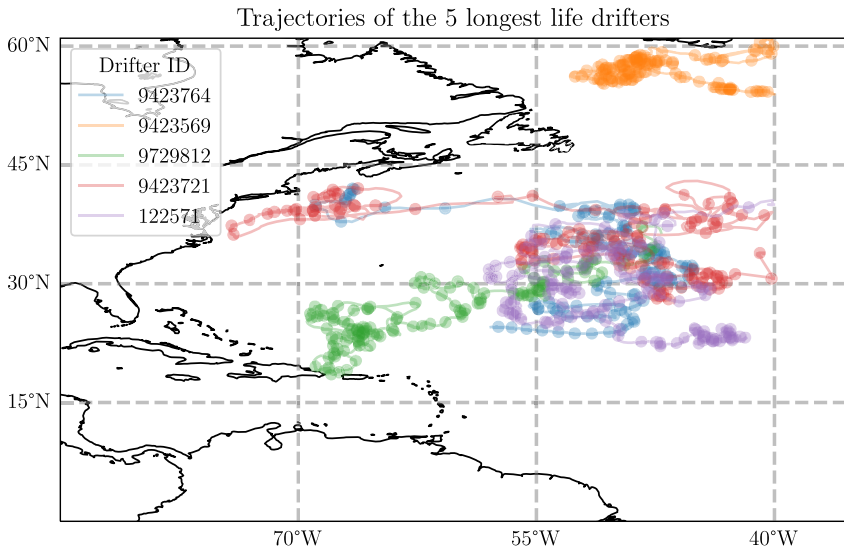


Figure 2. A plot showing the five longest trajectories in the dataset described in detail in [Appendix A](#), where each drifter has a lifetime of between 1100 and 1378 days. A point is plotted every 10 days.

quite close together. For example, even though the northernmost trajectory is almost as long as the rest it seems to travel a much smaller distance overall. The drifters that start in the west, however, travel very long distances overall. This motivates the use of probabilistic regression algorithms, as the velocity measurement will have very different magnitudes depending on where in space they are and other factors. Additionally, we may suspect that the prediction error will be larger when the underlying velocity is larger.

In [Figure 3](#), we display a subset of the predictor variables \mathbf{X} for the first 50 days of the same drifters shown in [Figure 2](#). In general, the geostrophic variables show strongest correlation to the \mathbf{Y}_i data for the respective dimensions. We do not show three of the nine dimensions in \mathbf{X} in this plot: longitude, latitude or days since January 1st. These are not shown in the plot as, in general, we do not expect these relationships to be linear, hence the need for more complicated models.

As a note, we do not exploit the grouped time-series structure of the drifter data in this work. When fitting the models, we ignore the drifter identification (ID) and just concatenate all data together. We learn a purely spatiotemporal model. If the goal was to forecast the location or velocity of the drifter at time $t + 1$ and we knew the history up to time t then exploiting information such as the velocity at time t would likely be an informative feature to use.

5.2. Metrics

Unlike in the simulation, here, there is no ground truth, hence, we cannot use KL divergence as in [Section 4](#) because the true distributions of these data are unknown. We shall instead use held-out data validation. We use a series of performance metrics that diagnose model fit which we introduce now.

Negative log-likelihood (NLL): All methods are effectively minimizing the negative log-likelihood; therefore, we use the negative log-likelihood as one of our metrics:

$$\frac{1}{N} \sum_{i=1}^N \log p(\mathbf{Y}_i | \mathbf{X}_i).$$

RMSE: To compare the point prediction performance of the models, we also report an average root mean squared error (RMSE):

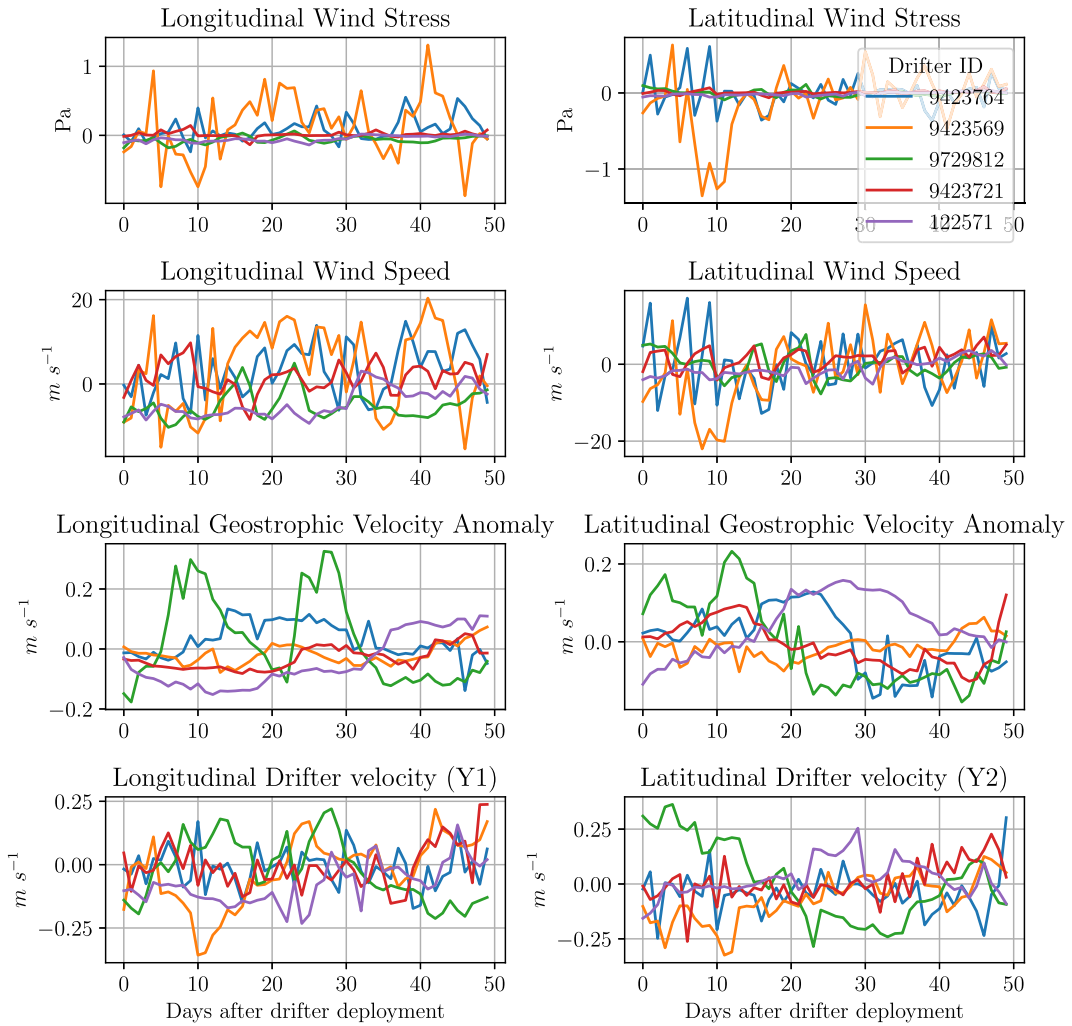


Figure 3. The values of the prediction feature \mathbf{X} for the sample of drifters shown in Figure 2. We only show the first 50 days to allow the reader to see the more granular patterns.

$$\left[\frac{1}{NP} \sum_{i=1}^N \sum_{j=1}^P \left(\mathbf{Y}_{ij} - \hat{\mathbf{Y}}_{ij} \right)^2 \right]^{1/2}, \text{ where } \hat{\mathbf{Y}}_i = \mathbb{E}[\mathbf{Y}|\mathbf{X}_i].$$

Region coverage and area: As this paper focuses on a measure of probabilistic prediction, we also report metrics related to the prediction region. The prediction region is the multivariate generalization of the one-dimensional prediction interval. The two related summaries of interest are: (a) the percentage of \mathbf{Y}_i covered by the $\alpha\%$ prediction region, and (b) the area of the prediction region.

A $\alpha\%$ prediction region can be defined for the multivariate Gaussian distribution as the set of values \mathbf{Y} which satisfy the following inequality:

$$(\mathbf{Y} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{Y} - \boldsymbol{\mu})^T \leq \chi_{P,\alpha}^2, \tag{6}$$

where $\chi_{P,\alpha}^2$ is the quantile function of the χ^2 distribution with P degrees of freedom evaluated at $\alpha\%$. This prediction region forms a hyper-ellipse which has an area given by

Table 2. Average test set metrics defined in Section 5.2.

Metric	NGB	Indep NGB	skGB	GB	NN
NLL	7.73 ± 0.02	7.74 ± 0.02	8.17 ± 0.02	8.79 ± 0.01	7.81 ± 0.02
RMSE (cm s ⁻¹)	14.53 ± 0.14	14.45 ± 0.12	14.31 ± 0.13	24.14 ± 0.36	16.63 ± 0.19
90% PR cov	0.87 ± 0.00	0.87 ± 0.00	0.89 ± 0.00	0.94 ± 0.00	0.89 ± 0.00
90% PR area (cm ² s ⁻²)	2482 ± 51	2568 ± 41	2714 ± 8	8070 ± 22	3670 ± 75

Note. Standard error estimated from 10 replications reported after ±. PR stands for prediction region. Coverage has been shortened to cov. All numbers rounded to two decimal places, aside from the 90% PR area row which is rounded to the nearest integer. Lower values are better for NLL and RMSE, best value is bolded in both rows.

$$\frac{(2\pi)^{P/2}}{P\Gamma(\frac{P}{2})}(\chi^2_{P,\alpha})^{P/2}|\Sigma|^{1/2}. \tag{7}$$

To use this as a metric, we report coverage as the percentage of data points which satisfy equation (6), and we report the average area of the prediction region over all points in the dataset. These metrics will be used in Table 2 in the next section.

5.3. Results

We compare the same five models considered in Section 4. To compare the models, we randomly split the dataset, keeping each individual drifter record within the same set. We put 10% of records into the test set, 9% of records into the validation set, and 81% into the training set. The model is fitted to the training set with access to the validation set for early stopping, and then the metrics from Section 5.2 are evaluated on the test set. This procedure is repeated 10 times. The aggregated metrics are shown in Table 2.

For this example, we also use a grid search for all the boosting approaches, in addition to the neural network approach. The reason we do this is because we found that with the default parameters used in Section 4 early stopping did not occur, suggesting that the base learner may not be flexible enough. For each boosting method, a grid search is carried out on the number of leaves and minimum data in leaves, as outlined in the Supplementary Material. The best hyperparameters from the grid search for each method are selected by evaluating the test set negative log-likelihood.

The aggregated results of the model fits are shown in Table 2. NGB and Indep NGB perform very similarly in terms of NLL, RMSE, and 90% PR coverage in this example. However, NGB provides a smaller 90% PR area, which is expected as correlation will reduce |Σ| in equation (7). To further highlight the differences, we show the spatial differences in negative log-likelihood between these two methods in Figure 4b. In Figure 4c,d, we show the averaged held out spatial predictions for ρ and μ from the NGB model. The results shown in Figure 4b,c suggest that using a combination of NGB and Indep NGB may be suitable for this application. For example, we could use the NGB model in geographic areas with high anticipated correlation; and otherwise we could use Indep NGB.

In Table 2, we see that the NN and GB approaches generally perform poorly. Both methods have a large root mean squared error, which is compensated for by a larger prediction region on average, as can be concluded from the large average PR area. This behavior agrees with the univariate example given in Figure 4 of the NGBoost paper (Duan et al., 2020), and the behavior of GB in the simulation of Section 4.

One of the novelties of this work is that we model the dependence between the two output dimensions. Any region where ρ is close to 0 in figure 4c likely means that this additional modeling of the dependence will not make a significant difference in the predictions. In contrast, any region where the absolute value of ρ is large implies that NGB predictions will be significantly different from Indep NGB. In particular, NGB predicts large values of |ρ| around the South Equatorial and Gulf Stream currents.

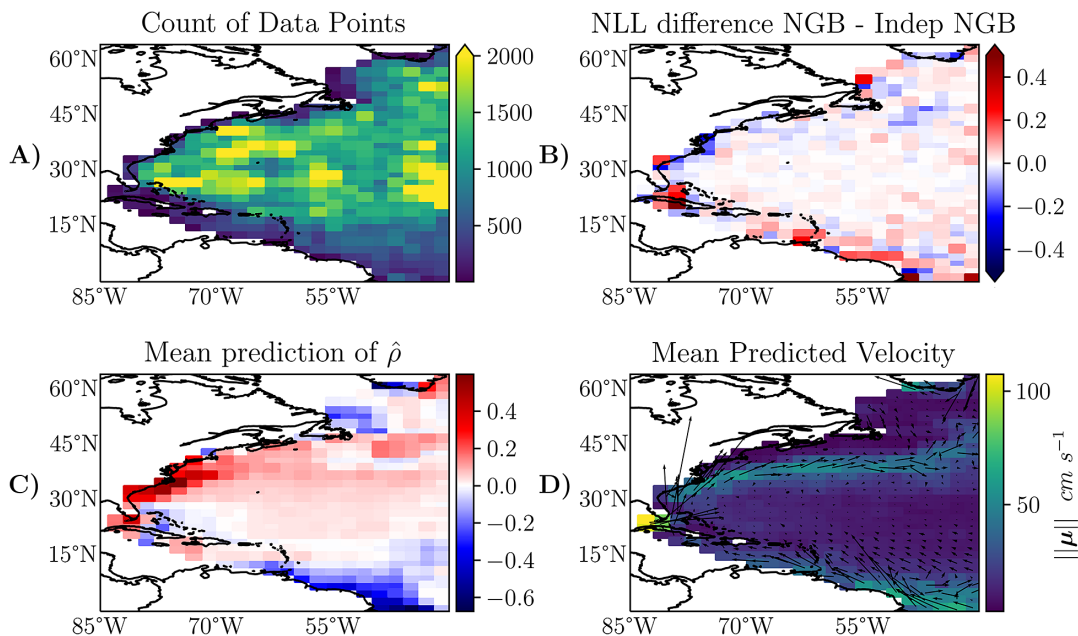


Figure 4. Summaries of test set results within $2^\circ \times 2^\circ$ latitude–longitude bins for the North Atlantic Ocean application. Panel (a) shows the spatial distribution of where the data are sampled. Panel (b) shows the difference between the negative log-likelihood spatially between NGB and Indep NGB; negative values (blue) implying NGB is better than Indep NGB (with vice versa in red). Panel (c) shows the average prediction of ρ , where $\rho = \Sigma_{0,1} / \sqrt{\Sigma_{0,0}\Sigma_{1,1}}$ is extracted from the predicted covariance matrix in the held out set from NGB. Panel (d) shows the mean currents estimated by NGB. All major ocean features are captured by the model (Lumpkin and Johnson, 2013).

In Figure 5, we show a sample trajectory on the Gulf Stream alongside the predicted means and the 70% prediction region from both Indep NGB and NGB. In Figure 5a, the main factor to note is that the elliptical prediction regions of NGB take various orientations on the trajectory, most of which have the major axis aligned with the flow. In contrast, the predictions from Indep NGB in Figure 5b show the ellipses' major axes are always either aligned with the longitudinal or latitudinal axis, as must be the case. This difference is particularly evident in the latter half of the shown trajectory, when traveling north–east, where Indep NGB's predictions show a larger minor axis in the ellipses in comparison to NGB predictions. This is probably the key reason why NGB performs better on average around the Gulf Stream, as seen in Figure 4b.

5.4. Investigation into multivariate NGBBoost against independent NGBBoost

As is clear in the metrics shown in Table 2, the NGB method we have developed is only marginally better than the Indep NGB in terms of negative log-likelihood. This is despite the fact that the model that predicts a multivariate Gaussian seems to be better suited to the problem. The distribution where each dimension is just a univariate Gaussian is a special case of the multivariate Gaussian.

One potential reason may be related to overfitting. In the learning, we allow early stopping to alter the number of boosting rounds. In NGB, the way model fitting works is that we require every parameter to be predicted by B base learners, whereas with Indep NGB, we allow the two models (one for longitudinal and one for latitudinal) to have different numbers of base learners B . The longitudinal model generally chose to use more decision trees than the latitudinal direction. Thus, the early stopping used by Indep NGB has

Drifter ID 54386 Trajectory

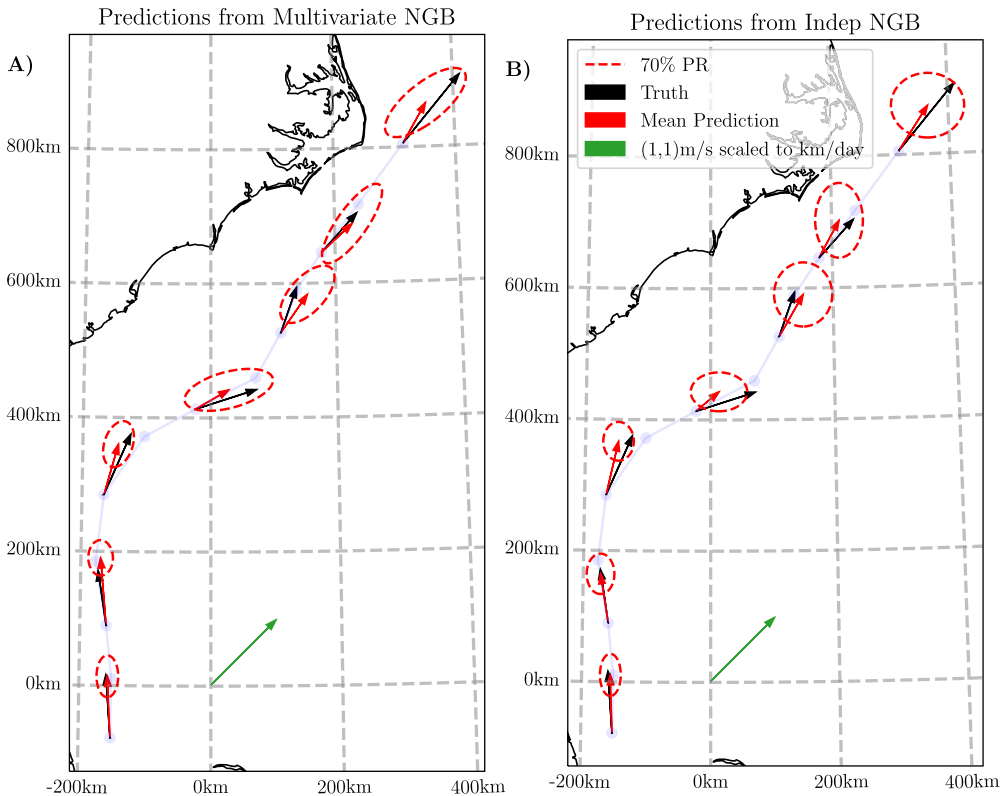


Figure 5. Plots showing the predictions from both NGB (a) and Indep NGB (b) for the first 12 days of drifter ID 54386 starting from September 23, 2005. The velocity predictions are plotted every 2 days for visualization purposes. The velocity measurements are translated to where the drifter would end up after 1 day if it continued at the constant predicted velocity for visualization purposes. The model used for the prediction did not see this trajectory when trained. The faded blue line shows the trajectory of the drifter with a plotted point every day. The 70% prediction region is the boundary from equation (6). The 70% level is just chosen for visualization purposes to prevent the ellipses from overlapping in the plot. Conversion to easting-northing computed using a transverse Mercator projection centered at -78° latitude and 28° longitude.

extra power in comparison to NGB. Further developments in the learning algorithm or fine-tuning of parameters of the base learner may correct this.

Another potential reason may be due to sampling, an argument which will be examined here. We find that NGB does better in areas with strong non-axes aligned currents like the Gulf Stream. For example, if we filter the test data used to calculate the entries in Table 2 to an area which is primarily made up of the Gulf Stream, we find that the improvement is more noticeable; consider the box with corners $(80^{\circ}\text{W}, 31^{\circ}\text{N})$, $(74^{\circ}\text{W}, 35^{\circ}\text{N})$ which is a section largely made up of the Gulf Stream. This area is shown in Figure 6. The average likelihood is 9.46 for NGB and 9.52 for Indep NGB on this subset of the data. That is a larger difference in likelihood than what we observed in the unfiltered dataset in Table 2. We show this difference in NLL across this smaller region in Figure 6b; we note that in some areas, NGB is doing much better, particularly in boxes with stronger currents, which can be identified using Figure 6d).

Another observation that can be seen is where NGB is doing worse than Indep NGB on average. In particular, the spatial boxes near the coast around $(78^{\circ}\text{W}, 33.5^{\circ}\text{N})$ in Figure 6b, where Indep NGB is

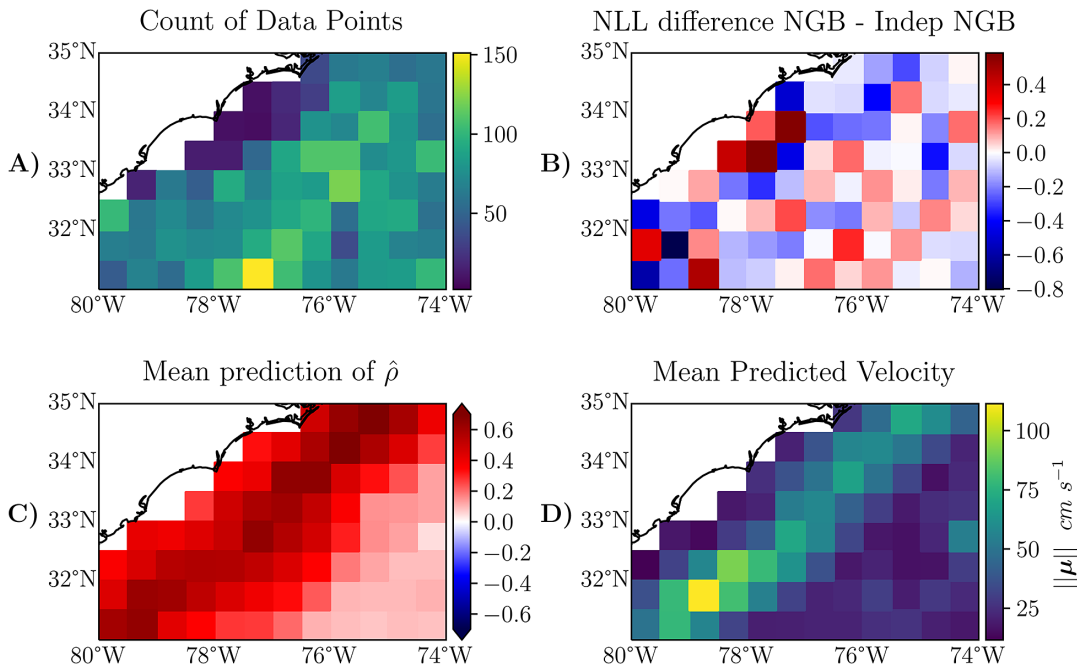


Figure 6. The same metrics as shown in Figure 4, just zoomed in on a smaller section which is analyzed in Section 5.4. The plot has a finer granularity than Figure 4, the resolution here is a $0.5^\circ \times 0.5^\circ$ bin granularity. The definition of the plotted metrics in panels A-D) can be found in the caption of Figure 4.

doing better (positive difference). By looking at the same boxes in Figure 6a,c, we can observe that these areas have few data points and returns a high predicted average correlation from the NGB method. A potential reason for Indep NGB doing better in this specific region is that Indep NGB is benefiting from the independence assumption. There is not enough data for NGB to learn that correlation is not strong in this specific region so NGB extrapolates information from the nearby regions. We can make a similar observation for the region around the bottom right corner of Figure 4b where Indep NGB is doing better in multiple boxes, and there are few data points, but NGB predicts a strong negative correlation.

A final observation is that areas such as the Gulf Stream have very large velocities. The sampling to create this dataset is carried out temporally, where we sampled one point for every day. To examine the significance of this, we reuse the example shown in Figure 5; this trajectory follows the Gulf Stream and it travels the distance shown over 12 days; contributing only 12 points to the dataset even though it covers around 1073 km. On the other hand, consider drifter 122571 shown in Figure 2, to travel 1073 km this drifter took approximately 113 days, hence for the same distance, drifter 11257 contributes almost 10 times more to the evaluation used in Table 2. On average, over the trajectory of drifter 11257, Indep NGB achieved a lower NLL. In summary, due to the nature of the data sampling procedure used here, in the evaluation we give more weight to regions with lower velocities. From Figure 4b,d, we can infer that in lower velocity regions Indep NGB usually has a lower NLL. Hence, we are likely giving too much weight in the objective to drifters in areas with weaker currents; this likely explains the relatively small differences in Table 3.

In summary, the conclusions from our study are that NGB shows improvements over Indep NGB in regions of strong currents where flow is correlated across dimensions, and in other regions of weak flow and correlation, the two methods perform similarly to each other. The exception to this rule is when data sparsity is present, for example, in the case around (78°W, 33.5°N), where Indep NGB can sometimes

Table 3. Summary of data used in the application.

Name	Unit	Resolution (spatial, temporal)
Drifter speed	u-v component ($cm\ s^{-1}$)	Irregular, 6 hourly
Wind speed	u-v component ($m\ s^{-1}$)	0.25 degrees, hourly
Surface wind stress	u-v component (Pa)	0.25 degrees, hourly
Geostrophic velocity anomaly	u-v component ($m\ s^{-1}$)	0.25 degrees, daily
Position	Location-longitude (degrees)	Irregular, 6 hourly
Day of year	Days	6 hourly

Note. Drifter speed is used to define the two-dimensional response Y . The rest of the variables listed defined the nine features used for X .

outperform NGB. Overall, this suggests that NGB can be robustly applied across wide spatial regions, as well as specifically applied in regions where Indep NGB is expected to fail. If there is no correlation present, then Indep NGB will likely do better as the assumption of independence between output dimensions is then correct in a Gaussian setting.

6. Conclusions

This paper has demonstrated the accuracy of our NGBBoost method when focusing on bivariate outcomes with a multivariate Gaussian distribution. The derivations of the natural gradient and implementation in NGBBoost are supplied for any dimensionality, but empirical proof of performance in higher dimensions is left to future work. Due to the quadratic relationship between P and the number of parameters used to parameterize the multivariate Gaussian, the complexity of the learning algorithm increases greatly with higher values of P . Investigating a reduced rank form of the covariance matrix may be of interest for higher P to reduce the number of parameters that must be learned.

Furthermore, the results shown here add to the collection of works that show large improvements in gradient-based learning when using natural gradients (Hoffman et al., 2013; Salimbeni et al., 2018; Duan et al., 2020). In the simulation of Section 4, we showed how crucial the natural gradient is to fitting this model, as the “GB” approach essentially failed to learn. The same exact comparison is shown in the original NGBBoost paper Duan et al. (2020) where GB performs significantly worse than the natural gradient approach. The simulations highlight the robustness of the natural gradient approaches in contrast to the ordinary gradient approaches, as we have shown how making a fairly simple adjustment ($+x$ and $-x^2$ to the mean terms) can yield a large difference in results.

The difference between NGB and Indep NGB was overall relatively small in our real-world application across the entire spatial domain studied, but NGB showed significant improvements around major currents such as the Gulf Stream where directional currents (that are not aligned with the x/y axis) are most prominent. This improvement was corroborated in simulations where NGB performed relatively much better than Indep NGB overall. Generally, one should expect NGB to excel in cases with high correlation between the outcomes, whereas a method assuming independence should suffice when that assumption is warranted. Deep-learning approaches are warranted in cases where a neural network naturally handles the input space such as images, speech, or text.

The analysis in Section 5.4, raises interesting questions about how we should sample this dataset. What would the difference in model fit and results be if we chose to sample a point every 50 km rather than daily when forming our dataset? Such a change in data processing would end up giving more weight to regions where strong currents exist; and in calmer regions where velocities are very low, it could take days to obtain a single sample in the dataset. Alternatively, we could also attempt to combine NGB and Indep NGB using model averaging, and this is reserved for future work.

6.1. Final remarks

In this paper, we have proposed a technique for performing probabilistic regression on vector-valued outputs with NGBoost. We have implemented software in the ngboost Python package which makes joint probabilistic regression easy to do with just a few lines of code and little tuning. Our simulation shows multivariate NGBoost exceeds the performance of existing methods for multivariate probabilistic regression. We have demonstrated the value of modeling the covariance between dimensions on a novel oceanographic problem using Global Drifter Program and remotely sensed satellite data. Specifically, we found that multivariate NGBoost excels in predicting ocean velocities in spatial areas where data sparsity is not an issue and correlation between the output dimensions is present.

Acknowledgments. We would like to thank the anonymous reviewers for taking the time and effort necessary to review the manuscript. We received valuable comments and suggestions that helped us improve the quality of the manuscript. Additionally, we thank Ryan Wolbeck for their work maintaining NGBoost.

Author Contribution. Conceptualization: all authors; Data curation: M.O'M., R.L.; Data visualization: M.O'M.; Methodology: M.O'M., A.M.S., A.S.; Writing original draft: M.O'M., A.M.S. All authors approved the final submitted draft.

Competing Interest. The authors have no competing interests.

Data Availability Statement. Replication data for the oceanographic application from Section 4 can be found at <https://doi.org/10.5281/zenodo.5644972>. The code to reproduce the results and simulation can be found at https://github.com/MikeOMa/MV_Prediction (see release v0.2 for source code used throughout this paper), however, for practical use, one should use the ngboost Python package, a sample code snippet is shown in the Supplementary Material. The package is available from the Python package index and version 0.3.10 was used for this paper.

Ethics Statement. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

Funding Statement. This work was funded by the Engineering and Physical Sciences Research Council (Grant Nos. EP/L015692/1 to M.O'M. and EP/R01860X/1 to A.M.S.).

Supplementary Material. The supplementary material for this article can be found at <https://doi.org/10.1017/eds.2023.4>.

References

- Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray DG, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y and Zheng X (2016) Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Berkeley: USENIX Association, pp. 265–283. <https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf>.
- Abdar M, Pourpanah F, Hussain S, Rezadegan D, Liu L, Ghavamzadeh M, Fieguth P, Cao X, Khosravi A, Acharya UR, Makarek V and Nahavandi S (2021) A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion* 76, 243–297.
- Álvarez M, Luengo D, Titsias M and Lawrence ND (2010) Efficient multioutput Gaussian processes through variational inducing kernels. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy: PMLR 9, pp. 25–32.
- Alvarez MA, Rosasco L and Lawrence ND (2012) Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning* 4(3), 195–266.
- Amari S-I (1998) Natural gradient works efficiently in learning. *Neural Computation* 10(2), 251–276.
- Banerjee S and Roy A (2014) *Linear Algebra and Matrix Analysis for Statistics*. Chapman and Hall/CRC, London.
- Bishop CM (1994) *Mixture Density Networks*. Birmingham: Aston University.
- Blundell C, Cornebise J, Kavukcuoglu K and Wierstra D (2015) Weight uncertainty in neural network. In Bach F and Blei D (eds), *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France: PMLR 37 pp. 1613–1622.
- Borchani H, Varando G, Bielza C and Larranaga P (2015) A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5(5), 216–233.
- Chipman HA, George EI and McCulloch RE (2010) Bart: Bayesian additive regression trees. *Annals of Applied Statistics* 4(1), 266–298.
- Duan T, Anand A, Ding DY, Thai KK, Basu S, Ng AY and Schuler A (2020) NGBoost: Natural gradient boosting for probabilistic prediction. In Daumé H and Singh A (eds), *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*. PMLR 119, pp. 2690–2700.

- Flanders Marine Institute** (2018) IHO sea areas, version 3. Dataset. Available at <https://www.marineregions.org/> (accessed 5 December 2020).
- Friedman JH** (2001) Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29, 1189–1232.
- Gal Y and Ghahramani Z** (2016) Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML '16)*, New York, USA: PMLR 48, pp. 1050–1059.
- Gneiting T and Raftery AE** (2007) Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association* 102(477), 359–378.
- Hoffman MD, Blei DM, Wang C and Paisley J** (2013) Stochastic variational inference. *Journal of Machine Learning Research* 14(5), 1303–1347.
- Kendall A and Gal Y** (2017) What uncertainties do we need in Bayesian deep learning for computer vision?. In von Luxburg U, Guyon I, Bengio S, Wallach H and Fergus R (eds), *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. Red Hook: Curran Associates Inc., pp. 5580–5590.
- Kingma DP and Ba J** (2015) Adam: A method for stochastic optimization. In Bengio Y and LeCun Y (eds), *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, May 7–9. <http://arxiv.org/abs/1412.6980>.
- Laurindo LC, Mariano AJ and Lumpkin R** (2017) An improved near-surface velocity climatology for the global ocean from drifter observations. *Deep Sea Research Part I: Oceanographic Research Papers* 124, 73–92.
- Lei M, Shiyang L, Chuanwen J, Hongling L and Yan Z** (2009) A review on the forecasting of wind speed and generated power. *Renewable and Sustainable Energy Reviews* 13(4), 915–920.
- Li R, Reich BJ and Bondell HD** (2021) Deep distribution regression. *Computational Statistics & Data Analysis* 159, 107203.
- Liu H, Cai J and Ong Y-S** (2018) Remarks on multi-output Gaussian process regression. *Knowledge-Based Systems* 144, 102–121.
- Lumpkin R and Centurioni L** (2019) Global Drifter Program quality-controlled 6-hour interpolated data from ocean surface drifting buoys. NOAA National Centers for Environmental Information. Dataset. Available at <https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.nodc:AOML-GDP> (accessed 15 December 2020).
- Lumpkin R and Johnson GC** (2013) Global ocean surface velocities from drifters: Mean, variance, El Niño–southern oscillation response, and seasonal cycle. *Journal of Geophysical Research: Oceans* 118(6), 2992–3006.
- Malagò L and Pistone G** (2015) Information geometry of the Gaussian distribution in view of stochastic optimization. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*. New York: Association for Computing Machinery, pp. 150–162.
- Maximenko N, Niiler P, Centurioni L, Rio M-H, Melnichenko O, Chambers D, Zlotnicki V and Galperin B** (2009) Mean dynamic topography of the ocean derived from satellite and drifting buoy data using three different techniques. *Journal of Atmospheric and Oceanic Technology* 26(9), 1910–1919.
- Meinshausen N and Ridgeway G** (2006) Quantile regression forests. *Journal of Machine Learning Research* 7(6), 983–999.
- Moreno-Muñoz P, Artés-Rodríguez A and Álvarez MA** (2018) Heterogeneous multi-output Gaussian process prediction. In Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N and Garnett R (eds), *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*. Montréal: Neural Information Processing Systems Foundation, pp. 6711–6720.
- Mulet S, Rio M-H, Etienne H, Artana C, Cancet M, Dibarbouré G, Feng H, Husson R, Picot N, Provost C and Strub PT** (2021) The new CNES-CLS18 global mean dynamic topography. *Ocean Sci.*, 17, 789–808, <https://doi.org/10.5194/os-17-789-2021>.
- 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 É** (2011) Scikit-learn: Machine learning in python. *Journal of Machine Learning Research* 12, 2825–2830.
- Rasp S and Lerch S** (2018) Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review* 146(11), 3885–3900.
- Rigby RA** (2019) *Distributions for Modeling Location, Scale, and Shape: Using GAMLSS in R*. Milton: CRC Press LLC.
- Salimbeni H, Eleftheriadis S and Hensman J** (2018) Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. In *International Conference on Artificial Intelligence and Statistics*, Lanzarote, Canary Islands: AISTATS 2018. PMLR 84, pp. 689–697.
- Segal M and Xiao Y** (2011) Multivariate random forests. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1(1), 80–87.
- Sinha A and Abernathy R** (2021) Estimating ocean surface currents with machine learning. *Frontiers in Marine Science* 8 <https://doi.org/10.3389/fmars.2021.672477>.
- Sützle EA and Hrycej T** (2005) Numerical method for estimating multivariate conditional distributions. *Computational Statistics* 20(1), 151–176.
- Thematic Assembly Centers** (2020a) Global ocean gridded L4 sea surface heights and derived variables reprocessed. E.U. Copernicus Marine Service Information. Dataset. Available at https://resources.marine.copernicus.eu/?option=com_csw&view=details&product_id=SEALEVEL_GLO_PHY_L4_REP_OBSERVATIONS_008_047 (accessed 2 December 2020).
- Thematic Assembly Centers** (2020b) Global ocean wind L4 reprocessed 6 hourly observations. E.U. Copernicus Marine Service Information. Dataset. Available at https://resources.marine.copernicus.eu/?option=com_csw&view=details&product_id=WIND_GLO_WIND_L4_REP_OBSERVATIONS_012_006 (accessed 2 December 2020).
- Williams CKI and Rasmussen CE** (2006) *Gaussian Processes for Machine Learning*, vol. 2. Cambridge, MA: MIT Press.
- Williams PM** (1996) Using neural networks to model conditional multivariate densities. *Neural Computation* 8(4), 843–854.

A. Appendix: Drifter Data Processing

For reproducibility, we give instructions on how the application dataset used in Section 4 is created. In its raw form, the drifter dataset is irregularly sampled in time; however, the product used here is processed to be supplied on a 6-hourly scale (Lumpkin and Centurioni, 2019) and includes location uncertainties. A high uncertainty would be caused by a large gap in the sampling of raw data or a large satellite positioning error. In our study, we dropped all drifter observations with a positional error greater than 0.5° in longitude or latitudinal coordinates.

The six features that we used relate to wind stress, geostrophic velocity, and wind speed, and are all available on longitude–latitude grids, with spatial and temporal resolution specified in Table 3; we refer to these as gridded products. Before using these products to predict drifter observations, we spatially interpolated¹ the gridded products to the drifter locations. To interpolate the gridded products, an inverse distance weighting interpolation was used. We only used the values at the $n = 4$ corners that define the spatial box containing the longitude–latitude location of the drifter location of interest, where the following estimate is used:

$$\frac{\sum_{i=1}^n w_i g_i}{\sum_{i=1}^n w_i}, \tag{A.1}$$

where g_i is the gridded value for the i th corner (e.g., a 0.5 ms^{-1} east–west geostrophic velocity at 30° longitude and 25° latitude), w_i is the inverse of the Haversine distance² between the drifter’s longitude–latitude and the longitude–latitude location of the gridded value g_i .

If two or more of the grid corners that are being interpolated from did not have a value recorded in the gridded product (e.g., if two corners were on a coastline in the case of wind stress and geostrophic velocity), we did not interpolate and treated that point as missing. If only one corner was missing, we use $n = 3$ in equation (A.1), omitting the missing point.

We low-pass filtered the drifter velocities, wind speed, and wind stress series to remove effects caused by inertial oscillations and tides; this process is similar to previous works (Laurindo et al., 2017). A critical frequency of 1.5 times the inertial period (the inertial period is a function of latitude) was used. Due to missing or previously dropped data due to preprocessing steps, some time series have gaps in time larger than 6 hr. In such cases, we split that time series into individual continuous segments. We applied a fifth-order Butterworth low-pass filter to each continuous segment. If any of these segments were shorter than 18 observations (4.5 days), the whole segment was treated as missing from the dataset. The Butterworth filter was applied in a rolling fashion to account for the changing fashion of the inertial period, which defines the critical frequency.

The geostrophic data are available on a daily scale; therefore, we decimated all data to daily, only keeping observations at 00:00. No further preprocessing was performed on the geostrophic velocities (after interpolation) as inertial and tidal motions are not present in the geostrophic velocity product.

Finally, to remove poorly sampled regions from the dataset, we partitioned the domain into 1° × 1° longitude–latitude non-overlapping grid boxes. We counted the number of daily observations contained in each box; if this count was less than 25, then the observations within that box were removed from the dataset. A link to download the processed data can be found at <https://doi.org/10.5281/zenodo.5644972>.

We only considered complete collections for $\mathbf{X}_i, \mathbf{Y}_i$; If any of the data were recorded as missing in the process explained above, that daily observation was removed from the dataset. Before fitting the models, we scaled the training data \mathbf{X}_i to be in the range [0, 1] for all variables, this step was taken to make the neural network fitting more stable, this step will have no effect on the gradient boosting approaches.

B. Appendix: Multivariate Normal Natural Gradient Derivations

To fit the NGBoost model, we require three elements, the log-likelihood, the derivative of the log-likelihood, and the Fisher information matrix. We use the parameterization for the multivariate Gaussian given in Section 2.2.

We shall derive results for the general case where P is the dimension of the data $\mathbf{Y} \in \mathbb{R}^P$. We use $Y_i \in \mathbb{R}, i \in \{1, \dots, P\}$ to denote the value of the i th dimension of \mathbf{Y} in this section; a similar notation is used for $\boldsymbol{\mu}$ and μ_i . The probability density function can be written as

$$p(\mathbf{Y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{P}{2}} |\boldsymbol{\Sigma}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{Y} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{Y} - \boldsymbol{\mu})\right]. \tag{A.2}$$

As mentioned in Section 2, the optimization of $\boldsymbol{\mu}$ is relatively straightforward, as it lives entirely on the unconstrained real line. $\boldsymbol{\Sigma}$ is an $P \times P$ positive definite matrix, which is a difficult constraint. Therefore, optimizing this directly is a challenge. Instead, if we consider the Cholesky decomposition of $\boldsymbol{\Sigma}^{-1} = \mathbf{U}^T \mathbf{U}$, where \mathbf{U} is an upper triangular matrix, we only require that the diagonal be positive to ensure that $\boldsymbol{\Sigma}$ is positive definite. We choose to model the inverse of $\boldsymbol{\Sigma}$ as in Williams (1996).

Therefore, we form an unconstrained representation for \mathbf{U} , where we denote a_{ij} as the element in the i th row and the j th column as follows:

$$a_{ij} = \begin{cases} \exp(v_{ij}), & \text{if } i = j, \\ v_{ij}, & \text{if } i < j, \\ 0, & \text{otherwise,} \end{cases} \quad \text{where } i, j \in \{1, \dots, P\}.$$

¹ The temporal resolutions already match.

² The Haversine distance is the greater circle distance between two longitude–latitude pairs.

Hence, we can fit \mathbf{U} by doing gradient-based optimization on $\{v_{ij} \in \mathbb{R} : 1 \leq i \leq j \leq P\}$ as all values live on the real line. As an example, for $P=2$, we can fit a multivariate Gaussian using unconstrained optimization through the parameter vector $\theta = (\mu_1, \mu_2, v_{11}, v_{12}, v_{22}) \in \mathbb{R}^M$.

As in Williams (1996), we simplify the parameters to

$$\begin{aligned} z_i &= \mu_i - Y_i & i \in \{1, \dots, P\}, \\ \eta_i &= (\mathbf{Uz})_i = \sum_{j=1}^P a_{ij} z_j & i \in \{1, \dots, P\}, \end{aligned}$$

where, we have denoted \mathbf{z} as the column vector of $z_i \in \mathbb{R}$. Throughout the following derivations, we will interchangeably use $\{\Sigma, \mathbf{U}, a_{ij}, v_{ij}, \boldsymbol{\mu}, \mathbf{z}, z_i, \eta_i\}$ without noting that these parameters have a mapping between them (e.g., $a_{11} = \exp(v_{11})$).

Noting that $\log |\Sigma| = \log |\mathbf{U}^T \mathbf{U}|^{-1} = -2 \log |\mathbf{U}| = -2 \log (\prod_{i=1}^P a_{ii})$, the negative log-likelihood can be simplified as follows:

$$\begin{aligned} -\log p(\mathbf{Y}|\theta = (\boldsymbol{\mu}, \mathbf{v})) &= -\frac{P}{2} \log(2\pi) + \frac{1}{2} \times \log |\Sigma| + \frac{1}{2} (\mathbf{Y} - \boldsymbol{\mu}) \Sigma^{-1} (\mathbf{Y} - \boldsymbol{\mu}) \\ &= c - \sum_{i=1}^P \log a_{ii} + \frac{1}{2} \mathbf{z}^T \mathbf{U}^T \mathbf{U} \mathbf{z} \\ &= \sum_{i=1}^P \left\{ \frac{1}{2} \eta_i^2 - v_{ii} \right\} + c, \end{aligned}$$

where c is a constant independent of $\boldsymbol{\mu}$ and v_{ij} . For a shorter notation, we will write $l = -\log p(\mathbf{Y}|\theta = (\boldsymbol{\mu}, \mathbf{v}))$.

The first derivatives are stated in Williams (1996) and we also state them here for reference:

$$\frac{dl}{d\mu_i} = \sum_{j=1}^i \eta_j a_{ji} \quad i \in \{1, \dots, P\} \tag{A.3}$$

$$\frac{dl}{dv_{ii}} = \eta_i z_i a_{ii} - 1 \quad i \in \{1, \dots, P\} \tag{A.4}$$

$$\frac{dl}{dv_{ij}} = \eta_i z_j \quad 1 \leq i < j \leq P. \tag{A.5}$$

The final element that we need for NGBoost to work is the $M \times M$ Fisher information matrix. This was not needed in Williams (1996), so we derive the Fisher information here. We start all calculations from the following definition:

$$\mathcal{I}_{ij} = \mathbb{E} \left[\frac{d^2 l}{d\theta_i d\theta_j} \right], \quad i, j \in \{1, \dots, M\}.$$

Note that \mathcal{I}_{ij} is a symmetric matrix, such that $\mathcal{I}_{ij} = \mathcal{I}_{ji}$. For convenience, we denote the entries of the matrix with the subscripts of the parameter symbols, for example, $\mathcal{I}_{\mu_i, v_{kq}}$. We use the letters $i, j, k, q \in \{1, \dots, P\} \subset \mathbb{N}$ to index the variables. The following two expectations are frequently used: $\mathbb{E}[z_i z_j] = \Sigma_{ij}$ and $\mathbb{E}[z_i] = 0$.

The derivatives of η_j with respect to the other parameters are repeatedly used, so we give them here:

$$\begin{aligned} \frac{d\eta_k}{d\mu_i} &= \begin{cases} 0, & \text{if } k > i, \\ a_{ki}, & \text{if } k \leq i, \end{cases} & i, k \in \{1, \dots, P\} \\ \frac{d\eta_i}{dv_{kq}} &= \frac{d}{dv_{kq}} \sum_{j=1}^P a_{ij} z_j & 1 \leq i \leq P, \quad 1 \leq k \leq q \leq P \\ &= \begin{cases} 0, & \text{if } i \neq k, \\ z_q, & \text{else if } q > k, \\ a_{ii} z_i, & \text{else if } q = k. \end{cases} \end{aligned}$$

We start with the Fisher information for μ_i, μ_j , using the existing derivation of $\frac{dl}{d\mu_k}$ in equation (A.3):

$$\begin{aligned} \frac{d^2 l}{d\mu_i d\mu_k} &= \frac{d}{d\mu_i} \sum_{j=1}^k \eta_j a_{jk} & i, k \in \{1, \dots, P\} \\ &= \sum_{j=1}^k a_{ji} a_{jk} \\ &= [\mathbf{U}^T \mathbf{U}]_{ik}. \end{aligned}$$

We therefore have that

$$\mathcal{I}_{\mu_i, \mu_j} = \Sigma_{ij} \quad i, j \in \{1, \dots, P\}. \tag{A.6}$$

Next, we consider the mean differentiated with respect to v_{kq} again starting from equation (A.3):

$$\begin{aligned} \frac{d^2 l}{d\mu_i dv_{kq}} &= \frac{d}{dv_{kq}} \sum_{j=1}^i \eta_j a_{ji} & i \in \{1, \dots, P\}, 1 \leq k \leq q \leq P \\ &= \sum_{j=1}^k \left[\eta_j \frac{d}{dv_{kq}} a_{ji} + a_{ji} \frac{d}{dv_{kq}} \eta_j \right]. \end{aligned}$$

As the expectation of both terms inside the sum is zero for every valid combination of i, k, q , we conclude that

$$\mathcal{I}_{\mu_i, v_{kq}} = 0 \quad i \in \{1, \dots, P\}, 1 \leq k \leq q \leq P. \tag{A.7}$$

Finally, the last elements that we require for the Fisher information matrix are the entries for v_{ij}, v_{kq} . First, we consider diagonals ($i = j$) w.r.t. all v_{kq} with $k \leq q$. We start by using Equation (A.4) for $\frac{dl}{dv_i}$:

$$\begin{aligned} \frac{d^2 l}{dv_{ii} dv_{kq}} &= \frac{d}{dv_{kq}} \eta_i z_i a_{ii} & i \in \{1, \dots, P\}, 1 \leq k \leq q \leq P \\ &= a_{ii} z_i \frac{d}{dv_{kq}} \eta_i + \eta_i z_i \frac{d}{dv_{kq}} a_{ii} \\ &= \begin{cases} a_{ii} z_i z_q, & \text{if } k = i \text{ and } k < q, \\ a_{ii}^2 z_i z_i, & \text{if } i = k = q, \\ 0, & \text{if } i \neq k, \end{cases} \\ &+ \begin{cases} z_i \sum_{j=i}^P a_{ij} z_j a_{ii}, & \text{if } i = k = q, \\ 0, & \text{Otherwise.} \end{cases} \end{aligned}$$

Taking the expectation, we get

$$\mathcal{I}_{v_{ii}, v_{kq}} = \begin{cases} a_{ii} \Sigma_{iq}, & \text{if } k = i \text{ and } q > k, \\ a_{ii}^2 \Sigma_{ii} + a_{ii} \sum_{j=i}^P a_{ij} \Sigma_{ij}, & \text{if } i = k = q, \\ 0, & \text{if } i \neq k. \end{cases} \quad i \in \{1, \dots, P\}, 1 \leq k \leq q \leq P \tag{A.8}$$

Finally, we derive the Fisher information for the off diagonals with respect to the off diagonals ($i < j$ and $k < q$). We start from the expression for $\frac{dl}{dv_{ij}}$ in Equation (A.5):

$$\begin{aligned} \frac{d^2 l}{dv_{kq} dv_{ij}} &= \frac{d}{dv_{kq}} \eta_i z_j & 1 \leq i < j \leq P, 1 \leq k < q \leq P \\ &= z_j \frac{d}{dv_{kq}} \eta_i \\ &= \begin{cases} z_j z_q, & \text{if } k = i, \\ 0, & \text{if } i \neq k. \end{cases} \end{aligned}$$

Hence,

$$\mathcal{I}_{v_{ij}, v_{kq}} = \begin{cases} \Sigma_{jq}, & \text{if } k = i, \\ 0, & \text{if } k \neq i, \end{cases} \quad 1 \leq i < j \leq P, 1 \leq k < q \leq P. \tag{A.9}$$

Equations (A.6)–(A.9) give the full specification of the Fisher information.