Molly Margaret Davies* and Mark J. van der Laan

Optimal Spatial Prediction Using Ensemble Machine Learning

DOI 10.1515/ijb-2014-0060

Abstract: Spatial prediction is an important problem in many scientific disciplines. Super Learner is an ensemble prediction approach related to stacked generalization that uses cross-validation to search for the optimal predictor amongst all convex combinations of a heterogeneous candidate set. It has been applied to non-spatial data, where theoretical results demonstrate it will perform asymptotically at least as well as the best candidate under consideration. We review these optimality properties and discuss the assumptions required in order for them to hold for spatial prediction problems. We present results of a simulation study confirming Super Learner works well in practice under a variety of sample sizes, sampling designs, and data-generating functions. We also apply Super Learner to a real world dataset.

Keywords: cross-validation, spatial interpolation, generalized stacking, oracle inequality, Super Learner

1 Introduction

Optimal prediction of a spatially indexed variable is a crucial task in many scientific disciplines. For example, environmental health applications concerning air pollution often involve predicting the spatial distribution of pollutants of interest, and many agricultural studies rely heavily on interpolated maps of various soil properties.

Numerous algorithmic approaches to spatial prediction have been proposed (see Cressie [1] and Schabenberger and Gotway [2] for reviews), but selecting the best approach for a given data set remains a difficult statistical problem. One particularly challenging aspect of spatial prediction is that location is often used as a surrogate for large sets of unmeasured spatially indexed covariates. In such instances, effective prediction algorithms capable of capturing local variation must make strong, mostly untestable assumptions about the underlying spatial structure of the sampled surface and can be prone to overfitting. Ensemble predictors that combine the output of multiple predictors can be a useful approach in these contexts, allowing one to consider multiple aggressive predictors. There have been some recent examples of the use of ensemble approaches in the spatial and spatiotemporal literature. For example, Zaier et al. [3] used ensembles of artificial neural networks to estimate the ice thickness of lakes and Chen and Wang [4] used stacked generalization to combine support vector machines classifying land-cover types in hyperspectral imagery. Ensembling techniques have also been used to make spatially indexed risk maps. For example, Rossi et al. [5] used logistic regression to combine a library of four base learners trained on a subset of the observed data to obtain landslide susceptibility forecasts for the central Umbrian region of Italy. Kleiber et al. [6] have developed a Bayesian model averaging technique for obtaining locally calibrated probabilistic precipitation forecasts by combining output from multiple deterministic models.

The Super Learner prediction algorithm is an ensemble approach that combines a user-supplied library of heterogeneous candidate learners in such a way as to minimize v-fold cross-validated risk [7]. It is a generalization of the stacking algorithm first introduced by Wolpert [8] within the context of neural networks and later adapted by Breiman [9] to the context of variable subset regression. LeBlanc and Tibshirani [10] discuss stacking and its relationship to the model-mix algorithm of Stone [11] and the predictive sample-reuse method of Geisser [12]. The library on which Super Learner trains can include

***Corresponding author: Molly Margaret Davies,** Group in Biostatistics, University of California, Berkeley, Berkeley, CA, USA, E-mail: mollymdavies@gmail.com
Mark J. van der Laan, Group in Biostatistics, University of California, Berkeley, Berkeley, CA, USA

parametric and nonparametric models as well as mathematical models and other ensemble learners. These learners are then combined in an optimal way in the sense that the Super Learner predictor will perform asymptotically as well as or better than any single prediction algorithm in the library under consideration. Super Learner has been used successfully in nonspatial prediction (see for example Polley et al. [13]). In this paper, we review its optimality properties and discuss the assumptions necessary for these optimality properties to hold within the context of spatial prediction. We also present the results of a simulation study, demonstrating that Super Learner works well in practice under a variety of spatial sampling schemes and data-generating distributions. In addition, we apply Super Learner to a real world dataset, predicting water acidity for a set of 112 lakes in the Southeastern United States. We show Super Learner is a practical, data-driven, theoretically supported way to build an optimal spatial prediction algorithm from a large, heterogeneous set of predictors, protecting against both model misspecification and over-fitting.

2 Problem formulation

Consider a random spatial process indexed by location over a fixed, continuous, d-dimensional domain, $\{Y(s): s \in D \subset \mathbb{R}^d\}$. For a particular set of distinct sampling points $\{S_1, \ldots, S_n\} \subset D$, We observe $\{(\mathbf{S}_i, Y_i^\star): i = 1, \ldots, n\}$, where $Y^\star = Y(\mathbf{S}_i) + \epsilon_i$ and ϵ_i represents measurement error for the i^{th} observation. For all i, we assume $\mathbb{E}[Y_i^\star | \mathbf{S}_i = \mathbf{s}] = Y(\mathbf{s})$. Our objective is to predict $Y(\mathbf{s}')$ for unobserved locations $\mathbf{s}' \subset D$. Thus, our parameter of interest is the spatial process itself. We do not make any assumptions about the functional form of the spatial process. We do, however, assume that one of the following is true: for all i, either

(1) $(\mathbf{S}_i, Y_i^\star)$ are independently and identically distributed (i.i.d.), or
(2) $(\mathbf{S}_i, Y_i^\star)$ are independent but not identically distributed, or
(3) $Y_i^\star$ are independent given $\mathbf{S}_1, \ldots, \mathbf{S}_n$; and $\mathbb{E}[Y_i^\star | \mathbf{S}_1, \ldots, \mathbf{S}_n] = \mathbb{E}[Y_i^\star | \mathbf{S}_i] = Y(\mathbf{S}_i)$. This corresponds to a fixed design.

Each of these sets of assumptions implies that any measurement error is mean zero conditional on $\mathbf{S}_i$, or in the case of fixed design, conditional on $\mathbf{S}_1, \ldots, \mathbf{S}_n$. It is important to note that S could consist of both location and some additional covariates $\mathbf{W}$, i. e. $\mathbf{S} = (\mathbf{X}, \mathbf{W})$, where $\mathbf{X}$ refers to location. In such cases, it may be that measurement error is mean zero conditional on location and covariates, but not on location alone.

While these are reasonable assumptions for many spatial prediction problems, they are nontrivial and may not always be appropriate. For instance, instrumentation and calibration error within sensor networks can result in spatially structured measurement error that is not mean zero given $\mathbf{S}_1, \ldots, \mathbf{S}_n$. There has been an effort on the part of researchers to develop ways to adapt the cross-validation procedure so as to minimize the effects of this kind of measurement error when choosing parameters such as bandwidth in local linear regression or smoothing parameters for splines. Interested readers should consult Opsomer et al. [14] and Francisco-Fernandez and Opsomer [15] for overviews.

3 The Super Learner algorithm

Suppose we have observed n copies of the random variable O with true data-generating distribution $P_0 \in \mathcal{M}$, where the statistical model $\mathcal{M}$ contains all possible data generating distributions for O. The empirical distribution for our sample is denoted P_n. Define a parameter $\Psi: \mathcal{M} \to \mathbf{\Psi} \equiv \{\Psi[P]: P \in \mathcal{M}\}$ in terms of a risk function R as follows: $\Psi[P] = \arg\min_{\psi \in \mathbf{\Psi}} R(\psi, P)$ In this paper, we will limit our discussion to so-called linear risk functions, where $R(\psi, P) = PL(\psi) = \int L(\psi)(o) dP(o)$ for some loss function L. For a discussion of nonlinear risk functions, see van der Laan and Dudoit [16]. We write our parameter of interest as $\psi_0 = \mathbf{\Psi}[P_0] = \arg\min_\psi R(\psi, P_0)$, a function of the true data generating distribution P_0. For many spatial prediction applications, the Mean-Squared Error (MSE) is an appropriate choice for the risk function R, but this needn't necessarily be the case.

Define a library of J base learners of the parameter of interest ψ_0, denoted $\{\hat{\Psi}_j : P_n \to \hat{\Psi}_j[P_n]\}_{j=1}^J$. We make no restrictions on the functional form of the base learners. For example, within the context of spatial prediction, a library could consist of various Kriging and smoothing splines algorithms, Bayesian hierarchical models, mathematical models, machine learning algorithms, and other ensemble algorithms. We make a minimal assumption about the size of the library: it must be at most polynomial in sample size. Given this library of base learners, we consider a family of combining algorithms $\{\hat{\Psi}_\alpha = f(\{\hat{\Psi}_j : j\}, \alpha) : \alpha\}$ indexed by a Euclidean vector α for some function f. One possible choice of combining family is the family of linear combinations, $\hat{\Psi}_\alpha = \sum_{j=1}^J \alpha(j)\hat{\Psi}_j$. If it is known that $\psi_0 \in [0,1]$, one might instead consider the logistic family, $\log[\hat{\Psi}_\alpha/(1-\hat{\Psi}_\alpha)] = \sum_{j=1}^J \alpha(j)\log[\hat{\Psi}_\alpha/(1-\hat{\Psi}_\alpha)]$. In either of these families, one can also constrain the values α can take. In this paper, we constrain ourselves to convex combinations, i. e. for all j, $\alpha(j) \geq 0$ and $\sum_j \alpha(j) = 1$.

Let $\{\mathbf{B}_n\}$be a collection of length n binary vectors that define a random partition of the observed data into a training set $\{O_i : B_n(i) = 0\}$ and a validation set $\{O_i : B_n(i) = 1\}$. The empirical probability distributions for the training and validation sets are denoted $P^0_{n,\mathbf{B}_n}$ and $P^1_{n,\mathbf{B}_n}$, respectively. The estimated risk of a particular estimator $\hat{\Psi} : P_n \to \hat{\Psi}[P_n]$ obtained via cross-validation is defined as

$$\begin{aligned}\mathbb{E}_{\mathbf{B}_n}\left[R\left(\hat{\Psi}\left[P^0_{n,\mathbf{B}_n}\right], P^1_{n,\mathbf{B}_n}\right)\right] &= \mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n} L\left(\hat{\Psi}\left[P^0_{n,\mathbf{B}_n}\right]\right)\right] \\ &= \mathbb{E}_{\mathbf{B}_n}\left[\int L\left(\hat{\Psi}\left[P^0_{n,\mathbf{B}_n}\right], y\right) dP^1_{n,\mathbf{B}_n}(y)\right].\end{aligned}$$

Given a particular class of candidate estimators indexed by α the cross-validation selector selects the candidate which minimizes the cross-validated risk under the empirical distribution P_n,

$$\alpha_n \equiv \arg\min_\alpha \left\{\mathbb{E}_{\mathbf{B}_n}\left[R\left(\hat{\Psi}_\alpha\left[P^0_{n,\mathbf{B}_n}\right], P^1_{n,\mathbf{B}_n}\right)\right]\right\}.$$

The Super Learner estimate of ψ_0 is denoted $\hat{\Psi}_{\alpha_n}[P_n]$.

3.1 Key theoretical results

Super Learner's aggressive use of cross-validation is informed by a series of theoretical results originally presented in van der Laan and Dudoit [16] and expanded upon in van der Vaart et al. [17].We provide a summary of these results below. For details and proofs, the reader is referred to these papers.

First, we define a benchmark procedure called the oracle selector, which selects the candidate estimator that minimizes the cross-validated risk under the true data generating distribution P_0. We denote the oracle selector for estimators based on cross-validation training sets of size $n(1-p)$, where p is the proportion of observations in the validation set, as

$$\tilde{\alpha}_n \equiv \arg\min_\alpha \left\{\mathbb{E}_{\mathbf{B}_n}\left[R\left(\hat{\Psi}_\alpha\left[P^0_{n,\mathbf{B}_n}\right], P_0\right)\right]\right\}.$$

van der Laan and Dudoit [16] present an oracle inequality for the cross-validation selector α_n in the case of random design regression. Let $L(\cdot)$ be a uniformly bounded loss function with

$$M_1 \equiv \sup_{\psi, O} |L(\psi)[O] - L(\psi_0)[O]| < \infty.$$

Let $d_n(\psi, \psi_0) = P_0[L(\psi) - L(\psi_0)]$ be a loss-function based risk dissimilarity between an arbitrary predictor ψ and the parameter of interest ψ_0, where the risk dissimilarity $d_n(\cdot)$ is quadratic in the difference between ψ and ψ_0, i.e. $P_0[L(\psi) - L(\psi_0)]^2 \leq M_2 P_0[L(\psi) - L(\psi_0)]$. Suppose the cross-validation selector α_n defined above is a minimizer over a grid of K_n different α-indexed candidate estimators. Then for any real-valued $\delta > 0$,

$$\mathbb{E}\left[d_n\left(\hat{\Psi}_{\alpha_n}\left[P^0_{n,\mathbf{B}_n}\right],\psi_0\right)\right] \le (1+2\delta)\mathbb{E}\left[\min_{\alpha}\mathbb{E}_{\mathbf{B}_n}d_n\left(\hat{\Psi}_{\alpha}\left[P^0_{n,\mathbf{B}_n}\right],\psi_0\right)\right] + C(M_1,M_2,\delta)\frac{\log K_n}{n}, \tag{1}$$

where $C(\cdot)$ is a constant defined in van der Vaart et al. [17] (see also Appendix A for a definition within the context of fixed regression). Thus if the proportion of observations in the validation set, p, goes to zero as $n \to \infty$, and

$$\frac{1}{n\log n}\mathbb{E}\left[\min_{\alpha}\mathbb{E}_{\mathbf{B}_n}d_n\left(\hat{\Psi}_{\alpha}\left[P^0_{n,\mathbf{B}_n}\right],\psi_0\right)\right] \overset{n\to\infty}{\longrightarrow} 0,$$

it follows that $\hat{\Psi}_{\alpha_n}$, the estimator selected by the cross-validation selector, is asymptotically equivalent to the estimator selected by the oracle, $\hat{\Psi}_{\tilde{\alpha}_n}$, when applied to training samples of size $n(1-p)$, in the sense that

$$\frac{\mathbb{E}_{\mathbf{B}_n}\left[d\left(\hat{\Psi}_{\alpha_n}\left[P^0_{n,\mathbf{B}_n}\right],\psi_0\right)\right]}{\mathbb{E}_{\mathbf{B}_n}\left[d\left(\hat{\Psi}_{\tilde{\alpha}_n}\left[P^0_{n,\mathbf{B}_n}\right],\psi_0\right)\right]} \overset{n\to\infty}{\longrightarrow} 1.$$

The oracle inequality as presented in eq. (1) shows us that if none of the base learners in the library are a correctly specified parametric model and therefore do not converge at a parametric rate, the cross-validation selector performs as well in terms of expected risk dissimilarity from the truth as the oracle selector, up to a typically second order term bounded by $(\log K_n)/n$. If one of the base learners *is* a correctly specified parametric model and thus achieves a parametric rate of convergence, the cross-validation selector converges (with respect to expected risk dissimilarity) at an almost parametric rate of $(\log K_n)/n$.

For the special case where $Y^* = Y$ and the dimension of $\mathbf{S}$ is two, the cross-validation selector performs asymptotically as well as the oracle selector up until a constant factor of $(\log K_n)/n$. When $Y^* = Y$ and the dimension of $\mathbf{S}$, d, is greater than two, the rates of convergence of the base learners will be $n^{-1/d}$. This is slower than $n^{-1/2}$, the rate for a correctly specified parametric model, so the asymptotic equivalence of the cross-validation selector with the oracle selector applies.

The original work of van der Laan and Dudoit [16] used a random regression formulation. Spatial prediction problems where we have assumed either (2) or (3) in Section 2 above require a fixed design regression formulation. We provide a proof of the oracle inequality for the fixed design regression case in Appendix A.

The key message is that Super Learner is a data-driven, theoretically supported way to build the best possible prediction algorithm from a large, heterogeneous set of predictors. It will perform asymptotically as well as or better than the best candidate prediction algorithm under consideration. Expanding the search space to include all convex combinations of the candidates can be an important advantage in spatial prediction problems, where location is often used as a surrogate for unmeasured spatially indexed covariates. Super Learner allows one to consider sufficiently complex, flexible functions while providing protection against overfitting.

4 Cross-validation and spatial data

The theoretical results outlined above depend on the training and validation sets being independent. When this is not the case, there are generally no developed theoretical guarantees of the asymptotic performance of any cross-validation procedure [18]. Bernstein's inequality, which van der Laan and Dudoit [16] use in developing their proof of the oracle inequality, has been extended to accommodate certain weak dependency structures, so it may be that there are ways to justify certain optimality properties of v-fold cross-validation in these cases. There have also been some extensions to potentially useful fundamental theorems

that accommodate other specific dependency structures. Lumley [19] proved an empirical process limit theorem for sparsely correlated data which can be extended to the multidimensional case. Jiang [20] provided probability bounds for uniform deviations in data with certain kinds of exponentially decaying one-dimensional dependence, although it is unclear how to extend these results to multidimensional dependency structures where sampling may be irregular. Neither of these extensions is immediately applicable to the general spatial case, where sampling may or may not be regular and the extent of spatial correlation cannot necessarily be assumed to be sparse. There has been some attention in the spatial literature to the use of cross-validation within the context of Kriging and selecting the best estimates for the parameters in a covariance function, most of it urging cautious and exploratory use [1, 21]. Todini [22] has investigated methods to provide accurate estimates of model-based Kriging error when the covariance structure has been selected via leave-one-out cross-validation, although this remains an open problem.

Recall from Section 2 above that our parameter of interest is the spatial process $Y(\mathbf{s})$ and we have assumed $\mathbb{E}[Y^\star|\mathbf{S}=\mathbf{s}] = Y(\mathbf{s})$. Even if $Y(\mathbf{s})$ is a spatially dependent stochastic process such as a Gaussian random field, the true parameter of interest in most cases is not the full stochastic process, but rather the particular realization from which we have sampled. Conditioning on this realization removes all randomness associated with the stochastic process, and any remaining randomness comes from the sampling design and measurement error. So long as the data conform to one of the statistical models outlined above in Section 2, the optimality properties outlined above will apply.

5 Simulation study

We applied the Super Learner prediction algorithm to six data sets with known data generating distributions simulated on a grid of $128 \times 128 = 16,384$ points in $[0,1]^2 \subset \mathbb{R}^2$. Each spatial process was simulated once, hence samples of stochastic processes were taken from a common realization. All simulated processes were scaled to $[-4, 4]$ before sampling. See Figure 1 for an illustration.

The function $f_1(\cdot)$ is a mean zero stationary Gaussian random field (GRF) with Matérn covariance function [23]

$$C(h,\theta) = \sigma^2\left[\frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{h}{\phi}\right)^{\nu} K_\nu\left(\frac{h}{\phi}\right)\right] + \tau^2,$$

$$\theta = (\sigma^2 = 5,\ \phi = 0.5,\ \nu = 0.5,\ \tau^2 = 0),$$

where h is a distance magnitude between two spatial locations, σ^2 is a scaling parameter, $\phi > 0$ is a range parameter influencing the spatial extent of the covariance function and τ^2 is a parameter capturing micro-scale variation and/or measurement error. $K_\nu(\cdot)$ is a modified Bessel function of the third order and $\nu > 0$ parametrizes the smoothness of the spatial covariation. Learners were given spatial location as covariates.

The function $f_2(\cdot)$ is a smooth sinusoidal surface used as a test function in both Huang and Chen [24] and Gu [25], $f_2(\mathbf{s}) = 1 + 3\sin(2\pi[s_1 - s_2] - \pi)$. Learners were given spatial location as covariates.

The function $f_3(\cdot)$ is a weighted nonlinear function of a spatiotemporal cyclone GRF and an exponential decay function of distances to a set of randomly chosen points in $[-0.5, 1.5]^2 \subset \mathbb{R}^2$. In addition to spatial location, learners were given the distance to the nearest point as a covariate.

The function $f_4(\cdot)$ is defined by the piecewise function

$$f_4(\mathbf{s}, w) = \{|s_1 - s_2| + w\}I(s_1 < s_2) + \{3s_1 \sin(5\pi[s_1 - s_2]) + w\}I(s_1 \geq s_2),$$

where w is Beta distributed with non-centrality parameter 3 and shape parameters 4 and 1.5. Learners were given spatial location and w as covariates.

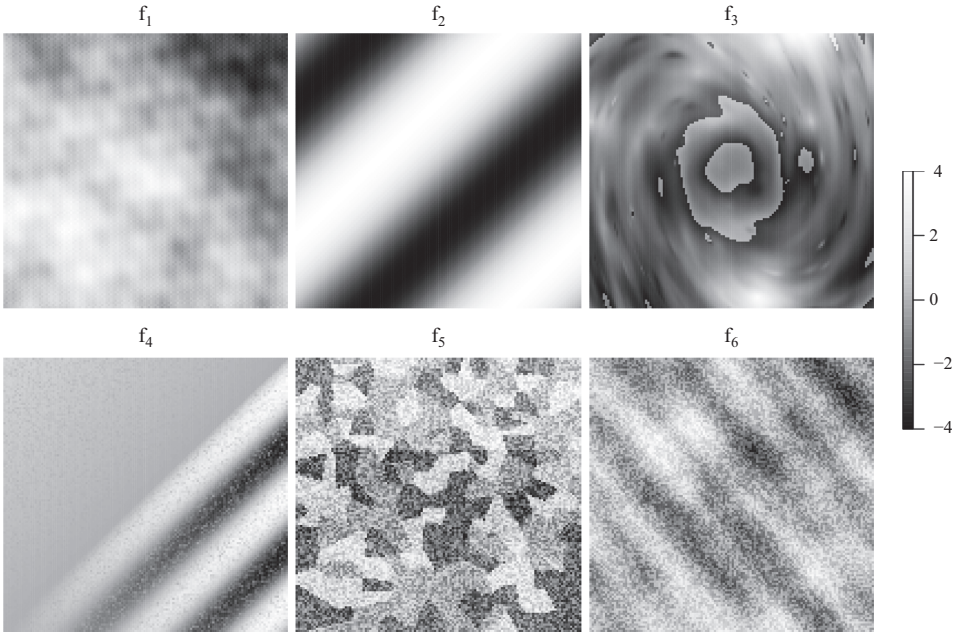

Figure 1: The six spatial processes used in the simulation study. All surfaces were simulated once on the domain $[0, 1]^2$. Process values for all surfaces were scaled to $[-4, 4] \subset \mathbb{R}$.

The function $f_5(\cdot)$ is a sum of several surfaces on $[0, 1] \subset \mathbb{R}^2$; a nonlinear function of a random partition of $[0, 1]^2$; a piecewise smooth function; and $w_2 \sim uniform(-1, 1)$. Learners were given spatial location, partition membership (w_1) and w_2 as covariates.

The function $f_6(\cdot)$ is a weighted sum of a spatiotemporal GRF with five time-points, a distance decay function of a random set of points in $[0, 1]^2$, and a beta-distributed random variable with non-centrality parameter 0 and shape parameters both equal to 0.5. Learners were given spatial location, the five GRFs and the beta-distributed random variable as covariates.

5.1 Spatial Prediction Library

The library provided to Super Learner consisted of either 83 (number of covariates = 2) or 85 (number of covariates > 2) base learners from 13 general classes of prediction algorithms. We provide a brief description of each, and list the parameter values used in the libraries. All algorithms were implemented in R Development Core Team [26]. The names of the R packages used are listed in Table 1.

Table 1: A list of R packages used to build the Super Learner library for spatial prediction.

Algorithm class	R library	Reference(s)
DSA	DSA	Neugebauer and Bullard [27]
GAM	GAM	Hastie [28]
GP	kernlab	Karatzoglou et al. [29]
GBM	GBM	Ridgeway [30]
GLMnet	glmnet	Friedman et al. [31]
KNNreg	FNN	Li [32]
Kriging	geoR	PJ and Ribeiro [33], PJ Ribeiro and Diggle [34]
Polymars	polspline	Kooperberg [35]
Random Forest	randomForest	Liaw and Wiener [36]
SVM	kernlab	Karatzoglou et al. [29]
TPS	fields	Furrer et al. [37]

Deletion/Substitution/Addition (**DSA**) performs data-adaptive polynomial regression using $v-$fold cross-validation and the L_2 loss [38]. Both the number of folds in the algorithm's internal cross-validation and the maximum number of terms allowed in the model (excluding the intercept) were fixed to five. The maximum order of interactions was $k \in \{3, 4\}$, and the maximum sum of powers of any single term in the model was $p \in \{5, 10\}$.

Generalized Additive Models (**GAM**) assume the data are generated by a model of the form $\mathbb{E}[Y|X_1, \ldots, X_p] = \alpha + \sum_{i=1}^{p} \phi_i(X_i)$, where Y is the outcome, $(X_1, \ldots, X_p)$ are covariates and each $\phi_i(\cdot)$ is a smooth nonparametric function [39]. In this simulation study, the $\phi(\cdot)$ are cubic smoothing spline functions parametrized by desired equivalent number of degrees of freedom, $df \in \{2, 3, 4, 5, 6\}$. To achieve a uniformly bounded loss function, predicted values were truncated to the range of the sampled data, plus or minus one.

Gaussian Processes (**GP**) assume the observed data are normally distributed with a covariance structure that can be represented as a kernel matrix [40]. Various implementations of the Bessel, Gaussian radial basis, linear and polynomial kernels were used. See Table 2 for details about the kernel functions and parameter values. Predicted values were truncated to the range of the observed data, plus or minus one, to achieve a uniformly bounded loss function.

Table 2: Kernels implemented in the simulation library. $\langle \mathrm{x}, \mathrm{x}' \rangle$ is an inner product.

Kernel	Function $\mathbf{k}(\mathrm{x}, \mathrm{x}')$	Parameter values
Bessel	$\frac{J_{v+1}(\sigma\|\|\mathrm{x}-\mathrm{x}'\|\|)}{(\|\|\mathrm{x}-\mathrm{x}'\|\|)^{-d(v+1)}}$	J_{v+1} is a Bessel function of 1^{st} kind, $(\sigma, v, d) \in \{1\} \times \{0.5, 1, 2\} \times \{2\}$
Radial Basis Function (RBF)	$\exp(-\sigma\|\|\mathrm{x}-\mathrm{x}'\|\|^2)$	Inverse kernel width σ estimated from data.
Linear	$\langle \mathrm{x}, \mathrm{x}' \rangle$	None
Polynomial	$(\alpha\langle \mathrm{x}, \mathrm{x}' \rangle + c)^d$	$(\sigma, \alpha, d) \in \{1, 3\} \times \{0.001, 0.1, 1\} \times \{1\}$
hyberbolic tangent	$\tanh(\alpha\langle \mathrm{x}, \mathrm{x}' \rangle + c)$	$(\alpha, c) \in \{0.005, 0.002, 0.01\} \times \{0.25, 1\}$

Generalized Boosted Modeling (**GBM**) combines regression trees, which model the relationship between an outcome and predictors by recursive binary splits, and boosting, an adaptive method for combining many weak predictors into a single prediction ensemble [41]. The GBM predictor can be thought of as an additive regression model fitted in a forward stage-wise fashion, where each term in the model is a simple tree. We used the following parameter values: number of trees = 10, 000; shrinkage parameter $\lambda = 0.001$; bag fraction (subsampling rate) = 0.5; minimum number of observations in the terminal nodes of each tree = 10; interaction depth $d \in \{1, 2, 3, 4, 5, 6\}$, where an interaction depth of d implies a model with up to d-way interactions.

GLMnet is a GLM fitted via penalized maximum likelihood with elastic-net mixing parameter $\alpha \in \{1/4, 1/2, 3/4\}$ [31].

K-Nearest Neighbor Regression (**KNNreg**) assumes the unobserved spatial process at a prediction point $\mathbf{s}'$ can be well-approximated by an average of the observed spatial process values at the k nearest sampled locations to $\mathbf{s}'$, $k \in \{1, 5, 10, 20\}$. When $k = 1$ and $\mathbf{S}$ are spatial locations only, this is essentially equivalent to Thiessen Polygons.

Kriging is perhaps the most commonly used spatial prediction approach. A general formulation of the spatial model assumed by Kriging can be written as $Y(s) = \mu(s) + \delta(s)$, $\delta(s) \sim N(0, C(\theta))$. The first term represents the large-scale mean trend, assumed to be deterministic and continuous. The second term is a Gaussian random function with mean zero and positive semi-definite covariance function $C(\theta)$ satisfying a stationarity assumption. The Kriging predictor is given as a linear combination of the observed data, $\hat{\Psi}(\mathbf{s}') = \sum_{i=1}^{n} w_i(\mathbf{s}')Y^*(\mathbf{s}_i)$. The weights $\{w_i\}_{i=1}^{n}$ are chosen so that Var $\left[\hat{\Psi}(\mathbf{s}') - Y(\mathbf{s}')\right]$ is minimized, subject to the constraint that the predictions are unbiased. Thus, given a parametric covariance function with known parameters θ and a known mean structure, a Kriging predictor computes the best linear unbiased

predictor of $Y(\mathbf{s}')$. For the Kriging base learners, the parametric covariance function was assumed to be spherical,

$$C(h,\theta)=\tau^2+\sigma^2\left[1-\frac{2}{\pi}\left(\sin^{-1}\left(\frac{h}{\phi}\right)+\frac{h}{\phi}\sqrt{1-\left(\frac{h}{\phi}\right)^2}\right)\right]\mathbf{I}(h<\phi).$$

The nugget τ^2, scale σ^2, and range ϕ were estimated using Restricted Maximum Likelihood (for details about REML, see for example Gelfand et al. [42], chapter 4, pp 48–49). The trend was assumed to be one of the following: Constant (traditional Ordinary Kriging, OK); a first order polynomial of the locations (traditional Universal Kriging, UK); a weighted linear combination of non-location covariates only (if any); a weighted linear combination of both locations and non-location covariates (if any). All libraries contained the first and second Kriging algorithms. Libraries for simulated processes with additional covariates contained the third and fourth algorithms as well.

Multivariate adaptive polynomial spline regression (**Polymars**) is an adaptive regression procedure using piecewise linear splines to model the spatial process, and is parametrized by the maximum size $m=\min\{6n^{1/3}, n/4, 100\}$, where n is sample size [43].

The **Random Forest** algorithm proposed by Breiman [44] is an ensemble approach that averages together the predictions of many regression trees constructed by drawing B bootstrap samples and for each sample, growing an unpruned regression tree where at each node, the best split among a subset of q randomly selected covariates is chosen. In our implementation, B was set to 1000, the minimum size of the terminal nodes was 5, and the number of randomly sampled variables at each split was $\lfloor\sqrt{p}\rfloor$, where p was the number of covariates.

The library contained a number of Support Vector Machines (**SVM**), each implementing one of two types of regression (epsilon regression, $\epsilon=0.1$; or nu regression, $\nu=0.2$), and one of five kernels: Bessel, Gaussian radial basis, linear, polynomial, and hyperbolic tangent. The kernels are described in Table 2. Predicted values were truncated to plus or minus one the range of the observed data to ensure a bounded loss, and the cost of constraints violation was fixed at 1.

Thin-plate splines (**TPS**) is another common approach to spatial prediction. The observed data are presumed to be generated by a deterministic process $Y(\mathbf{s})=g(\mathbf{s})$, where $g(\cdot)$ is an m times differentiable deterministic function with $m>d/2$ and $\dim(\mathbf{s})=d$. The estimator of $g(\cdot)$ is the minimizer of a penalized sum of squares,

$$\hat{g}=\underset{g\in\mathcal{G}}{\operatorname{argmin}}\sum_{i=1}^{n}(Y_i-g(\mathbf{s}_i))^2+\lambda J_m(g), \tag{2}$$

With d-dimensional roughness penalty

$$J_m(g)=\int_{\mathbb{R}^d}\sum_{\{(\upsilon_1,\ \ldots,\ \upsilon_d)\}}\binom{m}{\upsilon_1,\ \ldots,\ \upsilon_d}\left(\frac{\partial^m g(\mathbf{s})}{\partial\mathbf{s}_1^{\upsilon_1}\ldots\partial\mathbf{s}_d^{\upsilon_d}}\right)^2 d\mathbf{s},$$

where the sum in (5.1) is taken over all nonnegative integers $(\upsilon_1,\ \ldots,\ \upsilon_d)$ such that $\Sigma_{i=1}^{d}\upsilon_i=m$ [45]. The tuning parameter $\lambda\in[0,\infty)$ in (2) controls the permitted degree of roughness for $\hat{g}$. As λ tends to zero, the predicted surface approaches one that exactly interpolates the observed data. Larger values of λ allow the roughness penalty term to dominate, and as λ approaches infinity, $\hat{g}$ tends toward a multivariate least squares estimator. In our library, the smoothing parameter was either fixed to $\lambda\in\{0,\ 0.0001,\ 0.001,\ 0.01,\ 0.1\}$ or estimated data-adaptively using Generalized Cross-validation (GCV) (see Craven and Wahba [46] for a description of the GCV procedure). Predicted values were truncated to plus or minus one of the range of the observed data to ensure a bounded loss.

The library also contained a main terms Generalized Linear Model (**GLM**) and a simple empirical mean function.

5.2 Simulation procedure

Our simulation study examined the effect of sample size ($n \in \{64, 100, 529\}$), signal-to-noise ratio (SNR), and sampling scheme. SNR was defined as the ratio of the sample variance of the spatial process and the variance of additive zero-mean normally distributed noise representing measurement error. Processes were simulated with either no added noise or with noise added to achieve a SNR of 4. We examined three sampling schemes: simple random sampling (SRS), random regular sampling (RRS), and stratified sampling (SS). Random regular samples were regularly spaced subsets of the 16, 384 point grid with the initial point selected at random. Stratified random samples were taken by first dividing the domain $[0, 1]^2$ into n equal-area bins and then randomly selecting a single point from each bin.

The following procedure was repeated 100 times for each combination of spatial process, sample size, SNR level, and sampling design, giving a total of 10,800 simulations:

1. Sample n locations and any associated covariates and process values from the grid of 16, 384 points in $[0, 1]^2 \subset \mathbb{R}^2$ according to one of the three sampling designs described above.
2. For those simulations with SNR = 4, draw n i.i.d. samples of the random variable $\varepsilon \sim N(0, \sigma_\varepsilon^2)$ and add them to the $n$sampled process values $\{Y_1,..., Y_n\}$, where σ_ε^2 has been calculated to achieve an SNR of 4.
3. Pass the sampled values to Super Learner, along with a library of base learners on which to train. The number of folds v used in the cross-validation procedure depended on n: if $n = 64$, then $v = 64$; if $n = 100$, then $v = 20$; if $n = 529$, then $v = 10$. Super learner uses cross-validation and the L_2 loss function to estimate the risk of each candidate predictor and returns an estimate of the optimal convex combination of the predictions made by all base learners according to their cross-validated risk.
4. For each base learner in the library and for the trained Super Learner, predict the spatial process under consideration at all unsampled points. Calculate mean squared errors (MSEs) and then divide these by the variance of the spatial process. We refer to this measure of performance as the Fraction of Variance Unexplained (FVU). This makes it reasonable to compare prediction performances across different spatial processes.

5.3 Simulation results

Table 5 in Appendix B lists the average performance for each individual base learner in the library, and Table 3 summarizes prediction performance for each algorithm class in the library and for Super Learner itself. Super learner was clearly the best predictor overall when comparing across broad classes, with an average FVU of 0.24 (SD = 0.22). The next best performing algorithmic class was thin-plate splines using GCV to choose the roughness penalty, with an average FVU of 0.42 (SD = 0.36). Universal Kriging (FVU = 0.44), random forest (FVU = 0.35), and Ordinary Kriging (FVU = 0.45) all performed similarly, which was slightly less well than TPS (GCV). Super Learner was also the best performer across noise conditions, sampling designs, and sample sizes, with performance improving markedly as sample size increased.

Table 4 breaks algorithmic class performance down by simulated surface. f_1 was a mean-zero GRF, something we would expect both Kriging and thin-plate splines algorithms to predict well. TPS (GCV) and Super Learner were the best performers, with nearly identical average FVUs of 0.11 (SD = 0.06). The other TPS algorithms and Universal Kriging faired slightly less well, with an average FVU of 0.15. Ordinary Kriging had an average FVU of 0.26, which was actually greater than the average FVUs for Random Forest (0.16), K-nearest neighbors regression (0.19), GBM (0.22), GAM (0.24), and DSA (0.25).

f_2 was a simple sinusoidal surface, another functional form where we would expect thin-plate splines to excel, provided the samples properly captured the periodicity of the process. TPS (GCV) had the best overall performance, with an average FVU of 0.07 (SD = 0.09). Super Learner performed only slightly less well, with an average FVU of 0.09 (SD = 0.11). The other TPS algorithms (0.23), Ordinary Kriging (0.24) and Universal Kriging (0.25) performed substantially less well on average.

Table 3: Average FVUs (standard deviations in parentheses) from the simulation study for each algorithm class. FVUs were calculated from predictions made on all unsampled points at each iteration. Algorithms are ordered according to overall performance.

Algorithm Class		Sample Size			SNR		Sampling Design		
	Overall	64	100	529	None	4	SRS	RRS	SS
Super Learner	0.24	0.40	0.25	0.07	0.22	0.27	0.26	0.25	0.22
	(0.22)	(0.26)	(0.15)	(0.06)	(0.22)	(0.22)	(0.23)	(0.24)	(0.18)
TPS (GCV)	0.42	0.58	0.44	0.24	0.40	0.45	0.46	0.41	0.40
	(0.36)	(0.39)	(0.35)	(0.25)	(0.37)	(0.35)	(0.38)	(0.37)	(0.34)
Krige (UK)	0.44	0.59	0.51	0.21	0.42	0.46	0.42	0.53	0.36
	(0.30)	(0.28)	(0.27)	(0.20)	(0.31)	(0.29)	(0.30)	(0.31)	(0.28)
Random Forest	0.45	0.56	0.49	0.29	0.44	0.46	0.48	0.42	0.45
	(0.26)	(0.24)	(0.25)	(0.21)	(0.26)	(0.26)	(0.27)	(0.24)	(0.26)
Krige (OK)	0.45	0.62	0.53	0.21	0.43	0.47	0.41	0.59	0.36
	(0.32)	(0.29)	(0.28)	(0.20)	(0.32)	(0.31)	(0.29)	(0.33)	(0.28)
KNNreg	0.50	0.67	0.56	0.27	0.47	0.53	0.53	0.48	0.49
	(0.34)	(0.34)	(0.33)	(0.21)	(0.35)	(0.33)	(0.35)	(0.33)	(0.34)
TPS	0.53	0.64	0.56	0.37	0.49	0.56	0.58	0.49	0.52
	(0.37)	(0.40)	(0.37)	(0.30)	(0.38)	(0.37)	(0.40)	(0.35)	(0.37)
GBM	0.54	0.69	0.57	0.36	0.53	0.55	0.55	0.54	0.54
	(0.30)	(0.26)	(0.25)	(0.29)	(0.30)	(0.30)	(0.30)	(0.30)	(0.30)
DSA	0.61	0.68	0.62	0.54	0.60	0.63	0.64	0.60	0.60
	(0.28)	(0.31)	(0.26)	(0.23)	(0.26)	(0.29)	(0.31)	(0.26)	(0.26)
GAM	0.65	0.70	0.65	0.60	0.64	0.66	0.68	0.63	0.64
	(0.30)	(0.31)	(0.30)	(0.29)	(0.30)	(0.31)	(0.32)	(0.29)	(0.30)
GLMnet	0.69	0.71	0.69	0.67	0.69	0.69	0.70	0.69	0.69
	(0.25)	(0.24)	(0.25)	(0.24)	(0.25)	(0.25)	(0.25)	(0.24)	(0.24)
GLM	0.69	0.71	0.69	0.67	0.69	0.69	0.70	0.68	0.69
	(0.25)	(0.25)	(0.25)	(0.24)	(0.25)	(0.25)	(0.25)	(0.24)	(0.24)
Polymars	0.73	0.84	0.78	0.56	0.71	0.74	0.76	0.70	0.71
	(0.36)	(0.40)	(0.33)	(0.29)	(0.34)	(0.38)	(0.40)	(0.34)	(0.34)
SVM	0.76	0.83	0.80	0.66	0.76	0.77	0.78	0.76	0.76
	(0.30)	(0.30)	(0.31)	(0.27)	(0.30)	(0.30)	(0.31)	(0.30)	(0.30)
GP	0.77	0.89	0.80	0.61	0.74	0.80	0.80	0.76	0.76
	(0.67)	(0.68)	(0.60)	(0.69)	(0.62)	(0.71)	(0.67)	(0.68)	(0.66)
Mean	1.01	1.01	1.01	1.00	1.01	1.01	1.01	1.00	1.00
	(0.01)	(0.02)	(0.01)	(0.00)	(0.01)	(0.01)	(0.02)	(0.01)	(0.01)

f_3 was a relatively complex function involving a "cyclone" Gaussian random field and a distance decay function of randomly selected points. Once again, the average performances of TPS (GCV) and Super Learner were nearly identical (FVU = 0.30, sd- 0.11).

f_4 was a smooth, heterogeneous process. TPS (GCV) (average FVU = 0.42), Super Learner (0.43), Ordinary Kriging (0.45), and Universal Kriging (0.46) all performed similarly.

f_5 was a clustered, rough surface we would expect to be well-suited to K nearest neighbors, GBM, and Random Forest. In fact, all three of these algorithmic classes had nearly identical performances, with an average FVU of 0.47. Super Learner, however, had an average FVU of $0.22\,(\mathrm{SD} = 0.14)$, which was dramatically better than any of the other algorithmic classes. The Ordinary (average FVU = 0.70) and Universal (0.68) Kriging algorithms had similar average performances to GAM (0.62), GLM (0.67), GLMnet (0.67), and DSA (0.68). Not surprisingly, TPS (GCV) and TPS with fixed λ did poorly, with average FVUs of 0.91 and 1.01, respectively.

f_6 was a somewhat rough surface constructed from a Gaussian random field and point-source distance decay functions. As expected, Kriging with trend $w_1, \ldots, w_6$ had the best performance on average, with an FVU of $0.25\,(\mathrm{SD} = 0.14)$, closely followed by Kriging with trend $\mathbf{s}, w_1, \ldots, w_6$ (average FVU = 0.26, sd = 0.15).

Table 4: Average FVU (standard deviation in parentheses) by spatial process.

Algorithm Class	Average FVU					
	f_1	f_2	f_3	f_4	f_5	f_6
Super Learner	0.11	0.09	0.30	0.43	0.22	0.31
	(0.06)	(0.11)	(0.11)	(0.36)	(0.14)	(0.19)
TPS (GCV)	0.11	0.07	0.30	0.42	0.91	0.72
	(0.06)	(0.09)	(0.11)	(0.36)	(0.17)	(0.23)
Krige (UK)	0.15	0.25	0.37	0.46	0.68	0.47
	(0.11)	(0.33)	(0.20)	(0.32)	(0.23)	(0.28)
Random Forest	0.16	0.31	0.41	0.89	0.47	0.46
	(0.06)	(0.18)	(0.12)	(0.15)	(0.14)	(0.09)
Krige (OK)	0.26	0.24	0.39	0.45	0.70	0.47
	(0.31)	(0.33)	(0.24)	(0.32)	(0.23)	(0.28)
KNNreg	0.19	0.29	0.44	0.92	0.47	0.70
	(0.10)	(0.26)	(0.16)	(0.29)	(0.34)	(0.19)
TPS	0.15	0.23	0.38	0.60	1.01	0.78
	(0.07)	(0.24)	(0.14)	(0.35)	(0.23)	(0.19)
GBM	0.22	0.65	0.49	0.97	0.47	0.46
	(0.07)	(0.36)	(0.13)	(0.08)	(0.24)	(0.08)
DSA	0.25	0.72	0.53	1.03	0.68	0.48
	(0.05)	(0.25)	(0.08)	(0.15)	(0.11)	(0.08)
GAM	0.24	1.05	0.49	1.02	0.62	0.49
	(0.02)	(0.08)	(0.04)	(0.09)	(0.08)	(0.12)
GLMnet	0.37	1.01	0.67	0.99	0.67	0.44
	(0.01)	(0.02)	(0.03)	(0.03)	(0.03)	(0.03)
GLM	0.37	1.02	0.67	0.98	0.67	0.44
	(0.01)	(0.03)	(0.02)	(0.03)	(0.03)	(0.03)
Polymars	0.28	0.94	0.60	1.11	0.78	0.64
	(0.10)	(0.30)	(0.19)	(0.25)	(0.20)	(0.34)
SVM	0.49	0.87	0.71	1.05	0.80	0.66
	(0.28)	(0.27)	(0.20)	(0.15)	(0.19)	(0.33)
GP	0.28	0.64	0.57	1.31	1.01	0.81
	(0.10)	(0.42)	(0.20)	(0.61)	(0.63)	(1.02)
Mean	1.00	1.01	1.01	1.00	1.01	1.01
	(0.01)	(0.01)	(0.01)	(0.01)	(0.01)	(0.01)

Super Learner had the next best average performance, with an average FVU of $0.31\,(\mathrm{SD} = 0.19)$. GLM, GLMnet, GBM, Random Forest, the Ordinary and Universal Kriging algorithms, and DSA all performed similarly slightly less well, with average FVUs from 0.44 to 0.48. The TPS (GCV) and TPS with fixed λ were at a disadvantage given the roughness of the surface, with average FVUs of 0.72 and 0.78, respectively.

These simulation results clearly illustrate some of the chief advantages of Super Learner as a spatial predictor. For surfaces that were perfectly suited for one or more base learners in the library, Super Learner either performed almost as well as the best base learner, or it outperformed its library. For more complex, rougher surfaces, Super Learner performed significantly better than any single base learner in the library. It had the best overall performance even at the smallest sample size, and appeared to be relatively insensitive to sampling strategy.

6 Practical example: predicting lake acidity

We applied Super Learner to a lake acidity data set previously analyzed by Gu [25] and Huang and Chen [24]. Increases in water acidity are known to have a deleterious effect on lake ecology. Having an accurate

estimate of the spatial distribution of lake acidity is an essential first step toward crafting effective regulatory interventions to control it. The data were sampled by the U.S. Environmental Protection Agency during the Fall of 1984 in the Blue Ridge region of the Southeastern United States [47], and consist of longitudes and latitudes (in degrees), calcium ion concentrations (in milligrams per liter), and pH values. The EPA used a systematic stratified sampling design which we treated as fixed here. Because only one sample per lake was collected, we assume some measurement error that is independent of lake pH, calcium ion concentration, and spatial location. The data are freely available in the R package gss [48]. We used the same nearly equal area projection as Gu [25] and Huang and Chen [24],

$$x_1 = \cos((\pi x_{lat})/180)\sin(\pi(x_{lon} - \bar{x}_{lon})/180)$$
$$x_2 = \sin(\pi(x_{lat} - \bar{x}_{lat})/180),$$

where $\bar{x}_{lat}$ and $\bar{x}_{lon}$ are the midpoints of the latitude and longitude ranges, respectively.

Let $\mathbf{x}_i = (x_{i,1}, x_{i,2})$ denote the i^{th} sampling location; w_i denote the calcium ion concentration observed at the i^{th} sampling location; and $Y_i^\star$ be the pH value observed at the i^{th} sampling location. We assume that $\mathbf{E}[Y_i^\star | \mathbf{S}_i = \mathbf{s}] = Y(\mathbf{s})$, where $\mathbf{S}_i = (\mathrm{x}_i, w_i)$. Our objective is to learn the lake pH spatial process from the data.

The library used to predict lake acidity was similar in composition to the simulation library described in Subsection 5.1, with some important differences. We reduced the number of parameterizations for some of the algorithm classes in the library. We used one DSA learner, which used 10-fold cross-validation and considered polynomials of up to five terms $(m = 5)$, each term being at most a two-way interaction $(k = 2)$ with a maximum sum of powers $p = 3$. We used a reduced number of parameterizations of GAM, GBM, TPS, GP, and SVM learners, as well. We also included screening algorithms that allowed us to train learners on specific subsets of covariates: $\mathbf{x}$, w, $\log w$, $(\mathbf{x}, w)$, and $(\mathbf{x}, \log w)$. We considered the L_2 loss function, and the predictions from all base learners were truncated to the observed pH range in order to ensure a uniformly bounded loss.

Table 6 in Appendix B provides a detailed list of the library and shows performance results for each base learner as well as Super Learner itself. Figure 2 provides graphical representations of Super Learner's pH predictions. The Super Learner predictions are a slightly smoothed version of the observed data, with attenuated predictions for the highest and lowest observations. We also cross-validated Super Learner $(V = 5)$ to get an unbiased estimate of its risk. Super Learner had nearly identical performance in terms of cross-validated risk to GAM (4 degrees, using $\times$ and $\log w$) and both versions of Random Forest. The GBM predictors and OK (using $\log w$) had slightly smaller cross-validated risks.

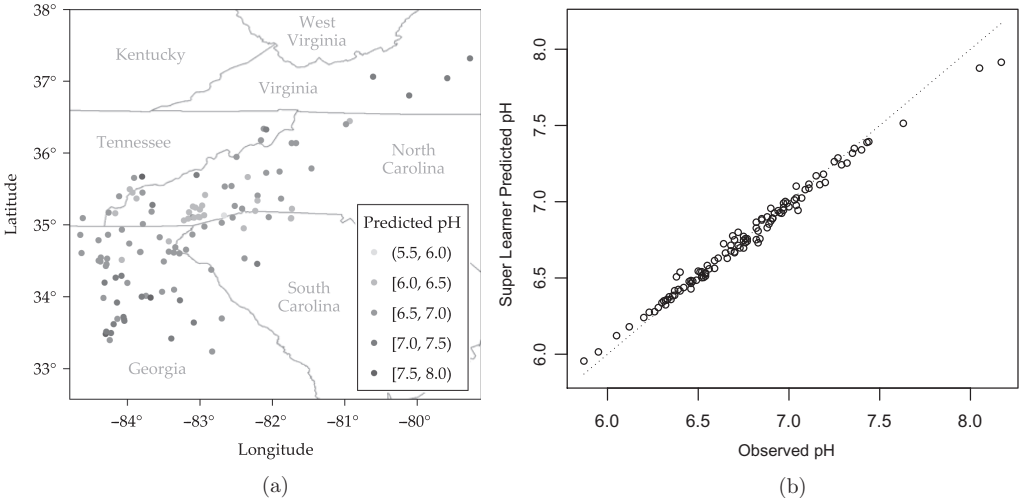

Figure 2: (a) A map of Super Learner's pH predictions, and (b) a plot of Super Learner's predictions as a function of the observed data. Super Learner mildly attenuated the pH values at either end of the range, but otherwise provided a fairly close fit to the data.

Many of the algorithms in the library performed better when given log w as opposed to w, but for those algorithms like GBM and Random Forest that were not attempting to fit some kind of polynomial trend, logging the calcium ion concentration made little difference in performance. This is not surprising, given the relationship between the activities of Ca^{2+} and H^{+} As expected, most algorithms had cross-validated risk estimates that were worse than their empirical risk estimates calculated from predictions made after training on the full data set. The Kriging algorithms, for instance, were all exact interpolators when trained on the full data, and thus had estimated empirical MSEs of 0, whereas their MSEs estimated via cross-validation ranged from 0.07 (FVU = 0.46) to 0.11 (FVU = 0.72). The Gaussian processes with RBF kernel had the most pronounced differences between the two risk estimates. For example, GP (RBF) trained on the covariates ($\mathbf{x}$, w) had an empirical MSE of 0.01 (FVU = 0.08) and a cross-validated MSE of 0.22 (FVU = 1.46).

One should use caution in general when interpreting the weights returned by the Super Learner algorithm. It can be tempting to use them to try to draw conclusions about the underlying data generating process. However, these weights are not necessarily stable, and we would expect them to differ given a different sample, or a different partitioning during cross-validation. Performance guarantees are with respect to the *ensemble*, and not the ranking of individual algorithms whose predictions are being averaged.

7 Conclusion/discussion

In this article, we have demonstrated the use of an ensemble learner for spatial prediction that uses cross-validation to optimally combine the predictions from multiple, heterogeneous base learners. We have reviewed important theoretical results giving performance bounds that imply Super Learner will perform asymptotically at least as well as the best candidate in the library. We discussed the assumptions required for these optimality properties hold. These assumptions are reasonable for many measurement error scenarios and commonly implemented spatial sampling designs, including various forms of stratified and random regular sampling.

Our simulations and practical data analysis used comparatively modest sample sizes. With the increased availability of massive environmental sensor networks and extremely large remotely sensed imagery, scalability of a spatial prediction method is often an important practical concern. The cross-validation step of Super Learner is a so-called 'embarrassingly parallel' problem, and there are implementations of Super Learner that parallelize this step (see for instance the function CV.SuperLearner in the SuperLearner R package). There are also a number of versions of the Super Learner algorithm under active development that are intended specifically to cope with very large data sets. One is called h2oEnsemble [49], and is a stand-alone R package that makes use of a Java-based machine learning platform that can be used for massive cluster computing. Another version of Super Learner has been modified to work with streaming data [50].

In this paper, we have not addressed dependent sampling designs, where sampling at one point changes the probability of sampling at another point. This is an important area for future research. We also limited our scope to the case where measurement error is at least conditionally mean-zero. Spatially structured measurement error that is not conditionally mean zero is a common problem in many spatial prediction applications, and there have been a number of attempts to alter the cross-validation procedure to accommodate it [51, 15]. These proposed techniques generally require one to estimate the error correlation structure from the data or to know it a priori. How well these algorithms perform if the correlation extent is substantially underestimated is unknown. Ideally, it would be best to have a stronger theoretical understanding of how the degree of dependence between training and validation sets affects cross-validated risk estimates both asymptotically and in finite samples. This is an important future area for research.

APPENDIX A. Oracle Inequality for Independent, Nonidentical Experiments and Quadratic Loss

Let $\mathbf{O}^n = (O_1, \ldots, O_n) \sim P_0^n$ be a vector of independent, nonidentical observations, where each $O_i = (\mathbf{X}_i, Y_i)$ consists of two components: a d-dimensional covariate vector $\mathbf{X}_i \in \mathbb{R}^d$, and a univariate outcome $Y_i \in \mathbb{R}$. We associate with each O_i an index $s_i \in S$. The true unknown data generating distribution for each O_i is denoted $P_{0,O_i}(O) = P_{0,O|S}(O|s_i) \in \{P_{0,O|S} : s \in S\}$: Let $P_{s,O}$ be the joint distribution of (S, O), defined by a degenerate marginal distribution of S, $I(S = s)$, and the conditional distribution of O given $S = s$, $P_{O|S}$. We can formulate $\mathbf{O}^n$ as n independent draws $(S_i, O_i) \sim P_{0,(s_i,O_i)}$, $i = 1, \ldots, n$, with empirical probability distribution P_n. Let $\mathcal{M}$ be a set of possible probability distributions of $P_{O|S}$. Define a parameter $\Psi : \mathcal{M} \to \boldsymbol{\Psi}$, and let $\psi_0 = \Psi(P_{0,O|S})$ be the true value of that parameter. Let $\mathbf{B}_n \in \{0, 1\}^n$ be a random vector indicating splits into a training sample, $\{i : \mathbf{B}_n(i) = 0\}$, and validation sample, $\{i : \mathbf{B}_n(i) = 1\}$. Let $p = 1/n \sum_{i=1}^n \mathbf{B}_n(i)$ be the proportion of observations in the validation sample, and let $P^0_{n,\mathbf{B}_n}$ and $P^1_{n,\mathbf{B}_n}$ be the empirical distributions of the training and validation samples, respectively. Define an average joint distribution: $\bar{P}^1_{0,\mathbf{B}_n} = (np)^{-1} \sum_{i:\mathbf{B}_n(i)=1} P_{0,(s_i,O_i)}$. Let $L(\psi)(S, O)$ be a loss function such that for all i, $P_{0,(s_i,O)} L(\psi_0) = \min_{\psi \in \boldsymbol{\Psi}} P_{0,(s_i,O_i)} L(\psi)$. Let $\{\hat{\Psi}_k(P_n) : k = 1, \ldots, K_n\}$ be a set of K_n estimators of ψ_0. Assume $\mathbb{P}(\hat{\Psi}_k(P_n) \in \Psi) = 1$ for all $k = 1, \ldots, K_n$. We write the true cross-validated risk of ψ_0 as $\tilde{\Theta}_{opt} = \mathbb{E}_{\mathbf{B}_n}[\bar{P}^1_{0,\mathbf{B}_n} L(\psi_0)]$. We denote the true conditional cross-validated risk of any estimator $\hat{\Psi}_k$ as

$$\begin{aligned}\tilde{\Theta}_{n(1-p)}(k) &\equiv \mathbb{E}_{\mathbf{B}_n}\left[\bar{P}^1_{0,\mathbf{B}_n} L\left(\hat{\Psi}_k[P^0_{n,\mathbf{B}_n}]\right)\right] \\ &= \mathbb{E}_{\mathbf{B}_n}\left[\frac{1}{np} \sum_{i:\mathbf{B}_n(i)=1} P_{0,(O_i|s_i)} L\left(\hat{\Psi}_k[P^0_{n,\mathbf{B}_n}]\right)(s_i, O_i)\right],\end{aligned}$$

and a benchmark (oracle) selector as $\tilde{k}_{n(1-p)} = \operatorname{argmin}_k \tilde{\Theta}_{n(1-p)}(k)$.

We denote the cross-validated risk of any estimator $\hat{\Psi}_k$ as

$$\tilde{\Theta}_{n(1-p)}(k) \equiv \mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n} L\left(\hat{\Psi}[P^0_{n,\mathbf{B}_n}]\right)\right] = \mathbb{E}_{\mathbf{B}_n}\left[\frac{1}{np} \sum_{i:\mathbf{B}_n(i)=1} L\left(\hat{\Psi}[P^0_{n,\mathbf{B}_n}]\right)(s_i, O_i)\right],$$

and the cross-validation selector as $k_n = \operatorname{argmin}_k \tilde{\Theta}_{n(1-p)}(k)$. Finally, we define a loss-based dissimilarity $d_n(\psi, \psi_0) \equiv \mathbb{E}_{\mathbf{B}_n}\left[\bar{P}^1_{0,\mathbf{B}_n}(L[\psi] - L[\psi_0])\right]$.

Assumptions.

A1. There exists a real-valued $M_1^* < \infty$ such that $\sup_{\psi \in \boldsymbol{\Psi}}\{\sup_{i,s_i,O_i} |L(\psi)(s_i, O_i) - L(\psi_0)(s_i, O_i)|\} \le M_1^*$, where the supremum over O_i is taken over the support of the distribution $P_{0,O_i|s_i}$ of O_i.

A2. There exists a real-valued $M_2 < \infty$ such that

$$\sup_{i,\psi \in \boldsymbol{\Psi}} \left\{\frac{\operatorname{Var}_{P_{0,(s_i,O_i)}}[L(\psi) - L(\psi_0)](S, O)}{\mathbb{E}_{P_{0,(s_i,O_i)}}[L(\psi) - L(\psi_0)](S, O)}\right\} \le M_2.$$

Definitions. We define the following constants.

$$M_1 = 2M_1^*$$
$$C(M_1, M_2, \delta) \equiv 2(1 + \delta)^2 \left(\frac{M_1}{3} + \frac{M_2}{\delta}\right)$$

Finite sample result. For any $\delta > 0$, we have

$$\begin{aligned}\mathbb{E}\left[d_n\left(\hat{\Psi}_{k_n}[P^0_{n,\mathbf{B}_n}], \psi_0\right)\right] &\le (1 + 2\delta)\ \mathbb{E}\left[d_n\left(\hat{\Psi}_{k_{n(1-p)}}[P^0_{n,\mathbf{B}_n}], \psi_0\right)\right] \\ &\quad + 2C(M_1, M_2, \delta)\ \frac{1 + \log K_n}{np}.\end{aligned} \tag{3}$$

Asymptotitic implications. Equation (3) has the following asymptotic implications:

$$\begin{aligned}
&\frac{\log K_n}{np\,\mathbb{E}\left[\tilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right)-\tilde{\Theta}_{opt}\right]} \xrightarrow{n\to\infty} 0 \Rightarrow \frac{\mathbb{E}\left[\tilde{\Theta}_{n(1-p)}(k_n)-\tilde{\Theta}_{opt}\right]}{\mathbb{E}\left[\tilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right)-\tilde{\Theta}_{opt}\right]} \xrightarrow{n\to\infty} 1.\\
&\frac{\log K_n}{np\left(\tilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right)-\tilde{\Theta}_{opt}\right)} \xrightarrow{p} 0 \Rightarrow \frac{\tilde{\Theta}_{n(1-p)}(k_n)-\tilde{\Theta}_{opt}}{\tilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right)-\tilde{\Theta}_{opt}} \xrightarrow{p} 1.
\end{aligned} \tag{4}$$

Equation (4) follows from the fact that, given a sequence of random variables X_1, X_2,..., and a positive function $g(n)$, $\mathbb{E}|X_n| = O\{g(n)\}$ implies $X_n = O_P\{g(n)\}$. This is a direct consequence of Markov's inequality.

Proof of theorem. We have

$$\begin{aligned}
0 &\le \tilde{\Theta}_{n(1-p)}(k_n) - \tilde{\Theta}_{opt}\\
&= \mathbb{E}_{\mathbf{B}_n}\left[\bar{P}^1_{0,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{k_n}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right]\\
&\quad -(1+\delta)\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{k_n}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right] \qquad (5)\\
&\quad +(1+\delta)\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{k_n}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right]\\
&\le \mathbb{E}_{\mathbf{B}_n}\left[\bar{P}^1_{0,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{k_n}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right]\\
&\quad -(1+\delta)\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{k_n}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right] \qquad (6)\\
&\quad +(1+\delta)\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{\tilde{k}_{n(1-p)}}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right]\\
&= \mathbb{E}_{\mathbf{B}_n}\left[\bar{P}^1_{0,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{k_n}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right] \qquad (7)\\
&\quad -(1+\delta)\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{k_n}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right] \qquad (8)\\
&\quad +(1+\delta)\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{\tilde{k}_{n(1-p)}}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right] \qquad (9)\\
&\quad -(1+2\delta)\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{\tilde{k}_{n(1-p)}}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right] \qquad (10)\\
&\quad +(1+2\delta)\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_{\tilde{k}_{n(1-p)}}[P^0_{n,\mathbf{B}_n}]\right)-L(\psi_0)\right\}\right] \qquad (11)
\end{aligned}$$

Equation (5) follows from the definition of $\tilde{\Theta}_{opt}$. Equation (6) follows from the definition of the cross-validation selector k_n, such that for all k, $\tilde{\Theta}_{n(1-p)}(k_n) \le \hat{\Theta}_{n(1-p)}(k)$. Let R_{n,k_n} represent the first two terms in the last expression, (7) and (8). Let $T_{n,\tilde{k}_{n(1-p)}}$ represent the second two terms of the last expression, (9) and (10). The last term, (11), is the benchmark and can be written as $(1+2\delta)\left[\tilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right)-\tilde{\Theta}_{opt}\right]$. Hence,

$$\begin{aligned}
0 \le \tilde{\Theta}_{n(1-p)}(k_n) - \tilde{\Theta}_{opt} \le (1+2\delta)&\left[\tilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right)-\tilde{\Theta}_{opt}\right]\\
&+ R_{n,k_n} + T_{n,\tilde{k}_{n(1-p)}}.
\end{aligned} \tag{12}$$

We now show that $\mathbb{E}R_{n,k_n} + \mathbb{E}T_{n,\tilde{k}_{n(1-p)}} \le 2C(M_1, M_2, \delta)\,(1+\log K_n)/(np)$. We introduce the following notation:

$$\hat{H}_k \equiv P^1_{n,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_k\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}$$

$$\tilde{H}_k \equiv \overline{P}^1_{0,\mathbf{B}_n}\left\{L\left(\hat{\Psi}_k\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}$$

$$R_{n,k}(\mathbf{B}_n) \equiv (1+\delta)\left[\tilde{H}_k - \hat{H}_k\right] - \delta\tilde{H}_k$$

$$T_{n,k}(\mathbf{B}_n) \equiv (1+\delta)\left[\hat{H}_k - \tilde{H}_k\right] - \delta\tilde{H}_k$$

Note that $R_{n,k} = \mathbb{E}_{\mathbf{B}_n}[R_{n,k}(\mathbf{B}_n)]$; $T_{n,k} = \mathbb{E}_{\mathbf{B}_n}[T_{n,k}(\mathbf{B}_n)]$; and that by definition of ψ_0, $\tilde{H}_k \geq 0$ for all k. Note also that given an arbitrary $k \in \{1, \ldots K_n\}$,

$$\mathbb{P}\left[R_{n,k_n}(\mathbf{B}_n) > s | P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] = \mathbb{P}\left[\tilde{H}_{k_n} - H_{k_n} > \frac{s+\delta\tilde{H}_{k_n}}{1+\delta}\middle| P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right]$$
$$\leq K_n \max_k \mathbb{P}\left[\tilde{H}_k - \hat{H}_k > \frac{s+\delta\tilde{H}_k}{1+\delta}\middle| P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right].$$

Similarly for $T_{n,\tilde{k}_{n(1-p)}}(\mathbf{B}_n)$,

$$\mathbb{P}\left[T_{n,\tilde{k}_{n(1-p)}}(\mathbf{B}_n) > s | P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] = K_n \max_k \mathbb{P}\left[\hat{H}_k - \tilde{H}_k > \frac{s+\delta\tilde{H}_k}{1+\delta}\middle| P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right].$$

Conditional on $P^0_{n,\mathbf{B}_n}$ and $\mathbf{B}_n$, consider the np random variables for which $\mathbf{B}_n(i) = 1$, $Z_{k,i} \equiv \left\{L\left(\hat{\Psi}_k\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}(s_i, O_i)$. We can rewrite $\hat{H}_k$ and $\tilde{H}_k$ in terms of $Z_{\mathrm{k,i}}$,

$$\hat{H}_k = \frac{1}{np}\sum_{i=1}^{np} Z_{k,i},$$

$$\tilde{H}_k = \frac{1}{np}\sum_{i=1}^{np} \mathbb{E}\left[Z_{k,i}\middle| P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right].$$

Then $\tilde{H}_k - \hat{H}_k$ is the sum of np mean zero centered random variables. By assumption **A1** above, the random variables $Z_{i,k}$ are bounded, with $|Z_{i,k}| \leq M_1$ a.s. By assumption **A2**, we also have

$$\sigma^2_{k,i} \equiv \mathrm{Var}\left[Z_{k,i}\middle| P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] \leq M_2\,\mathbb{E}\left[Z_{k,i}\middle| P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right],$$

which implies

$$\sigma^2_k \equiv \frac{1}{np}\sum_{i=1}^{np}\sigma^2_{k,i} \leq M_2\frac{1}{np}\sum_{i=1}^{np}\mathbb{E}\left[Z_{k,i}\middle| P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] = M_2\tilde{H}_k.$$

We will apply Bernstein's inequality to the centered empirical mean $\tilde{H}_k - \hat{H}_k$ and obtain a tail probability bounded by $\exp\{-npq/c\}$, where c is a finite, real-valued constant. This will show that the risk dissimilarities converge at a rate of $(\log K_n)/np$. We state Bernstein's inequality for ease of reference. A proof is given in Lemma A.2 on page 594 in Györfi, Kohler, and Walk [52].

Lemma 1 *Bernstein's inequality.*
Let Z_i, $i = 1, \ldots, n$ be independent, real valued random variables such that $Z_i \in [a, b]$ with probability one. Let $0 < \sum_{i=1}^n Var(Z_i)/n \leq \sigma^2$. Then, for all $\epsilon > 0$,

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n}(Z_i-\mathbb{E}Z_i)>\epsilon\right)\leq \exp\left\{\frac{-n\epsilon^2}{2(\sigma^2+\epsilon(b-a)/3)}\right\}.$$

This implies

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n}(Z_i-\mathbb{E}Z_i)>\epsilon\right|\right)\leq 2\exp\left\{\frac{-n\epsilon^2}{2(\sigma^2+\epsilon(b-a)/3)}\right\}.$$

By Bernstein's lemma, for $q>0$,

$$\begin{aligned}\mathbb{P}\left[R_{n,k}(\mathbf{B}_n)>q\middle|P^0_{n,\mathbf{B}_n},\mathbf{B}_n\right]&=\mathbb{P}\left[\tilde{H}_k-\hat{H}_k>\frac{1}{1+\delta}\left(q+\delta\tilde{H}_k\right)\middle|P^0_{n,\mathbf{B}_n},\mathbf{B}_n\right]\\&\leq\mathbb{P}\left[\tilde{H}_k-\hat{H}_k>\frac{1}{1+\delta}\left(q+\frac{\delta\sigma_k^2}{M_2}\right)\middle|P^0_{n,\mathbf{B}_n},\mathbf{B}_n\right]\\&\leq\exp\left\{-\left(\frac{np}{2[1+\delta]^2}\right)\left(\frac{\left[q+\delta\sigma_k^2/M_2\right]^2}{\sigma_k^2+\frac{M_1}{3(1+\delta)}\left[q+\delta\sigma_k^2/M_2\right]}\right)\right\}.\end{aligned}$$

Note that

$$\frac{\left[q+\delta\sigma_k^2/M_2\right]^2}{\sigma_k^2+\frac{M_1}{3(1+\delta)}\left[q+\delta\sigma_k^2/M_2\right]}=\frac{\left[s+\delta\sigma_k^2/M_2\right]^2}{\frac{\sigma_k^2}{q+\sigma_k^2/M_2}+\frac{M_1}{3(1+\delta)}}\geq\frac{\left[s+\delta\sigma_k^2/M_2\right]^2}{\frac{M_2}{\delta}+\frac{M_1}{3}}\geq\frac{s}{\frac{M_2}{\delta}+\frac{M_1}{3}}.$$

This shows that for $q>0$,

$$\mathbb{P}\left[R_{n,k_n}(\mathbf{B}_n)>q\middle|P^0_{n,\mathbf{B}_n},\mathbf{B}_n\right]\leq K_n\exp\{(-npq)/C(M_1,M_2,\delta)\}.$$

In particular, this provides us with a bound for the marginal probability of $R_{n,k_n}(\mathbf{B}_n)$,

$$\mathbb{P}\left[R_{n,k_n}(\mathbf{B}_n)>q\right]\leq K_n\exp\{(-npq)/C(M_1,M_2,\delta)\}.$$

As in the proof of theorem 1 in van der Laan, Dudoit, and Keles [53] and Dudoit and van der Laan [54], for each $u>0$, we have

$$\mathbb{E}[R_{n,k_n}]\leq u+\int_u^\infty K_n\exp\{(-npq)/C(M_1,M_2,\delta)\}\,\mathrm{d}q.$$

The minimum is attained at $u_n=C(M_1,M_2,\delta)\log K_n/np$ and is given by $C(M_1,M_2,\delta)(\log K_n+1)/np$. Thus $\mathbb{E}R_{n,k_n}\leq C(M_1,M_2,\delta)(1+\log K_n)(np)$. The same applies for $\mathbb{E}T_{n,\tilde{k}_{n(1-p)}}$.

Taking the expected values of the quantities in (A.4) yields the following finite sample result:

$$\begin{aligned}0&\leq\mathbb{E}\left[\tilde{\Theta}_{n(1-p)}(k_n)\right]-\tilde{\Theta}_{opt}\\&\leq(1+2\delta)\left(\mathbb{E}\left[\tilde{\Theta}_{n(1-p)}\right]-\tilde{\Theta}_{opt}\right)+2C(M_1,M_2,\delta)\left[\frac{1+\log K_n}{np}\right].\end{aligned}$$

This completes the proof. ■

APPENDIX B. Tables

Table 5: Simulation results for full library. For each algorithm, average Fraction of Variance Unexplained, (Avg FVU, standard deviation in parentheses) is the FVU averaged over all spatial processes, sample sizes, sampling designs, and noise conditions. At each iteration, MSEs were calculated using all unsamped locations. Note that of the eight Kriging algorithms, only two were used to predict all spatial processes.

Algorithm	Parameters	Avg FVU
Super Learner		0.24 (0.22)
DSA (v, m, k, p)	(5, 5, 3, 10)	0.62 (0.29)
	(5, 5, 3, 5)	0.61 (0.26)
	(5, 5, 4, 10)	0.62 (0.28)
	(5, 5, 4, 5)	0.61 (0.26)
GAM (degree)	2	0.65 (0.28)
	3	0.64 (0.29)
	4	0.65 (0.30)
	5	0.65 (0.31)
	6	0.66 (0.32)
GBM (degree)	1	0.64 (0.28)
	2	0.56 (0.28)
	3	0.53 (0.30)
	4	0.52 (0.30)
	5	0.51 (0.31)
	6	0.50 (0.31)
GLM		0.69 (0.25)
GLMnet (α)	0.25	0.69 (0.25)
	0.5	0.69 (0.25)
	0.75	0.69 (0.25)
GP (Bessel)	(1, 0.5, 2)	1.12 (1.52)
	(1, 1, 2)	0.81 (0.81)
	(1, 2, 2)	0.74 (0.67)
(linear)		0.69 (0.25)
(poly)	(1, 0.001, 1)	0.69 (0.25)
	(1, 0.01, 1)	0.69 (0.25)
	(1, 0.1, 1)	0.69 (0.25)
	(1, 1, 1)	0.69 (0.25)
	(3, 0.001, 1)	0.66 (0.27)
	(3, 0.01, 1)	0.65 (0.29)
	(3, 0.1, 1)	0.83 (0.65)
	(3, 1, 1)	0.84 (0.68)
(RBF)		0.92 (0.90)
KNNreg (k)	1	0.53 (0.38)
	5	0.40 (0.31)
	10	0.48 (0.32)
	20	0.60 (0.33)
Kriging (trend)	s	0.46 (0.34)
f_3, f_4 only	s,w_1	0.41 (0.26)
f_5 only	s,w_1,w_2	0.51 (0.15)
f_6 only	s,w_1, ...,w_6	0.26 (0.15)
	none	0.49 (0.35)
f_3, f_4 only	w_1	0.42 (0.28)
f_5 only	w_1,w_2	0.52 (0.16)
f_6 only	w_1, ...,w_6	0.25 (0.14)

(*continued*)

Table 5: (*continued*)

Algorithm	Parameters	Avg FVU
Mean		1.01 (0.01)
Polymars		0.73 (0.36)
Random Forest		0.45 (0.26)
SVM (Bessel; eps)	(1, 1, 1)	0.65 (0.27)
	(1, 1, 2)	0.57 (0.28)
	(1, 2, 1)	0.66 (0.27)
	(1, 2, 2)	0.59 (0.27)
(Bessel; nu)	(1, 1, 1)	0.67 (0.27)
	(1, 1, 2)	0.62 (0.27)
	(1, 2, 1)	0.68 (0.27)
	(1, 2, 2)	0.63 (0.27)
(linear; eps)		0.71 (0.25)
(linear; nu)		0.72 (0.28)
(poly; eps)	(1, 0.001, 1)	0.92 (0.12)
	(1, 0.1, 1)	0.71 (0.25)
	(1, 1, 1)	0.71 (0.25)
	(3, 0.001, 1)	0.85 (0.17)
	(3, 0.1, 1)	0.64 (0.28)
	(3, 1, 1)	0.83 (0.61)
(poly; nu)	(1, 0.001, 1)	0.97 (0.08)
	(1, 0.1, 1)	0.71 (0.25)
	(1, 1, 1)	0.72 (0.28)
	(3, 0.001, 1)	0.91 (0.14)
	(3, 0.1, 1)	0.66 (0.29)
	(3, 1, 1)	0.92 (0.71)
(RBF; eps)		0.48 (0.34)
(RBF; nu)		0.50 (0.32)
(tanh; eps)	(0.01, 0.25)	0.76 (0.21)
	(0.01, 1)	0.82 (0.18)
	(0.005, 0.25)	0.81 (0.19)
	(0.005, 1)	0.87 (0.16)
	(0.002, 0.25)	0.88 (0.15)
	(0.002, 1)	0.93 (0.11)
(tanh; nu)	(0.01, 0.25)	0.82 (0.19)
	(0.01, 1)	0.88 (0.16)
	(0.005, 0.25)	0.88 (0.16)
	(0.005, 1)	0.93 (0.12)
	(0.002, 0.25)	0.94 (0.11)
	(0.002, 1)	0.98 (0.07)
TPS (λ)	(GCV)	0.42 (0.36)
	0	0.52 (0.45)
	0.0001	0.44 (0.39)
	0.001	0.44 (0.35)
	0.01	0.54 (0.33)
	0.1	0.69 (0.28)

Table 6: Lake acidity results for full library. **S** denotes the variable subset each algorithm was given. Risks were estimated via cross-validation (CV) or on the full dataset (Full). β are the convex weights assigned to each algorithm in the Super Learner predictor.

Algorithm	$\widehat{\mathbf{MSE}}\,(\widehat{\mathbf{FVU}})$			
	S	**CV**	**Full**	β
Super Learner		0.08 (0.50)	0.00 (0.03)	
DSA				
	x,w	0.13 (0.85)	0.12 (0.80)	0
	x,w_ℓ	0.09 (0.57)	0.07 (0.46)	0
GAM (degree)				
2	x,w	0.09 (0.58)	0.07 (0.49)	0
	x,w_ℓ	0.08 (0.51)	0.07 (0.45)	0
3	x,w	0.08 (0.54)	0.07 (0.43)	0
	x,w_ℓ	0.08 (0.51)	0.06 (0.41)	0
4	x,w	0.08 (0.53)	0.06 (0.40)	0
	x,w_ℓ	0.08 (0.50)	0.06 (0.39)	0
GBM (degree)				
2	x,w	0.07 (0.49)	0.05 (0.32)	0
	x,w_ℓ	0.07 (0.49)	0.05 (0.32)	0
4	x,w	0.08 (0.50)	0.04 (0.29)	0
	x,w_ℓ	0.07 (0.49)	0.05 (0.32)	0
6	x,w	0.07 (0.49)	0.04 (0.28)	0.12
	x,w_ℓ	0.07 (0.49)	0.04 (0.29)	0
GLM				
	x,w	0.11 (0.74)	0.10 (0.67)	0
	x,w_ℓ	0.08 (0.54)	0.08 (0.50)	0
GLMnet (α)				
0.25	x,w	0.12 (0.79)	0.10 (0.67)	0
	x,w_ℓ	0.08 (0.55)	0.08 (0.50)	0
0.5	x,w	0.11 (0.73)	0.10 (0.67)	0
	x,w_ℓ	0.08 (0.54)	0.08 (0.50)	0
0.75	x,w	0.11 (0.75)	0.10 (0.67)	0
	x,w_ℓ	0.08 (0.54)	0.08 (0.50)	0
GP (Bessel)				
(1, 0.5, 2)	x,w	0.13 (0.83)	0.04 (0.25)	0
	x,w_ℓ	0.16 (1.03)	0.03 (0.21)	0
(1, 1, 2)	x,w	0.14 (0.90)	0.04 (0.27)	0
	x,w_ℓ	0.15 (0.97)	0.03 (0.22)	0
(1, 2, 2)	x,w	0.16 (1.08)	0.04 (0.29)	0
	x,w_ℓ	0.17 (1.10)	0.04 (0.25)	0
GP (linear)				
	x,w	0.11 (0.74)	0.10 (0.67)	0
	x,w_ℓ	0.08 (0.54)	0.08 (0.50)	0
GP (RBF)				
	x,w	0.22 (1.45)	0.02 (0.16)	0
	x,w_ℓ	0.22 (1.46)	0.01 (0.08)	0
GP (poly.)				
(1, 0.001, 1)	x,w	0.11 (0.74)	0.10 (0.67)	0
	x,w_ℓ	0.08 (0.54)	0.08 (0.50)	0
(1, 0.01, 1)	x,w	0.11 (0.74)	0.10 (0.67)	0
	x,w_ℓ	0.08 (0.54)	0.08 (0.50)	0

(*continued*)

Table 6: (*continued*)

Algorithm	$\widehat{\text{MSE}}$ ($\widehat{\text{FVU}}$)			
	S	**CV**	**Full**	β
(1, 0.1, 1)	x,w	0.11 (0.74)	0.10 (0.67)	0
	x,w_ℓ	0.08 (0.54)	0.08 (0.50)	0
(1, 1, 1)	x,w	0.11 (0.74)	0.10 (0.67)	0
	x,w_ℓ	0.08 (0.54)	0.08 (0.50)	0
KNNreg (k)				
1	x	0.17 (1.12)	0.00 (0.00)	0.02
	x,w	0.11 (0.73)	0.00 (0.00)	0.08
5	x	0.12 (0.76)	0.08 (0.52)	0
	x,w	0.08 (0.55)	0.06 (0.38)	0.04
10	x	0.11 (0.73)	0.09 (0.61)	0
	x,w	0.08 (0.53)	0.07 (0.43)	0
20	x	0.11 (0.72)	0.10 (0.66)	0.03
	x,w	0.09 (0.56)	0.08 (0.50)	0
Kriging				
(OK)		0.11 (0.71)	0.00 (0.00)	0
	w	0.09 (0.56)	0.00 (0.00)	0
	w_ℓ	0.07 (0.46)	0.00 (0.00)	0.58
(UK)	x	0.11 (0.72)	0.00 (0.00)	0
	x,w	0.09 (0.60)	0.00 (0.00)	0
	x,w_ℓ	0.08 (0.51)	0.00 (0.00)	0
Mean		0.15 (1.00)	0.15 (1.00)	0
Polymars				
	x,w	0.10 (0.63)	0.04 (0.27)	0
	x,w_ℓ	0.08 (0.56)	0.05 (0.36)	0
RF				
	x,w	0.08 (0.50)	0.02 (0.11)	0.06
	x,w_ℓ	0.08 (0.50)	0.02 (0.12)	0
SVM (Bessel; eps)				
(1, 1, 2)	x,w	0.09 (0.57)	0.06 (0.43)	0
	x,w_ℓ	0.08 (0.55)	0.06 (0.42)	0
(1, 2, 1)	x,w	0.09 (0.56)	0.08 (0.51)	0
	x,w_ℓ	0.08 (0.52)	0.07 (0.46)	0
(1, 2, 2)	x,w	0.09 (0.56)	0.07 (0.45)	0
	x,w_ℓ	0.08 (0.55)	0.07 (0.44)	0
SVM (Bessel; nu)				
(1, 1, 2)	x,w	0.09 (0.57)	0.07 (0.48)	0
	x,w_ℓ	0.09 (0.61)	0.07 (0.46)	0
(1, 2, 1)	x,w	0.1 (0.64)	0.08 (0.56)	0
	x,w_ℓ	0.08 (0.56)	0.07 (0.48)	0
(1, 2, 2)	x,w	0.09 (0.59)	0.07 (0.49)	0
	x,w_ℓ	0.09 (0.58)	0.07 (0.47)	0
SVM (poly, eps)				
(1, 0.001, 1)	x,w	0.15 (0.97)	0.14 (0.96)	0
	x,w_ℓ	0.14 (0.94)	0.14 (0.92)	0
(1, 0.1, 1)	x,w	0.12 (0.78)	0.10 (0.69)	0
	x,w_ℓ	0.08 (0.52)	0.08 (0.50)	0
(1, 1, 1)	x,w	0.12 (0.78)	0.11 (0.69)	0
	x,w_ℓ	0.08 (0.53)	0.08 (0.50)	0.08
(3, 0.001, 1)	x,w	0.14 (0.92)	0.13 (0.89)	0
	x,w_ℓ	0.12 (0.81)	0.12 (0.78)	0

(*continued*)

Table 6: (*continued*)

Algorithm	$\widehat{\text{MSE}}$ ($\widehat{\text{FVU}}$)			
	S	CV	Full	β
SVM (poly, nu)				
(1, 0.001, 1)	x,w	0.15 (0.98)	0.15 (0.97)	0
	x,w_ℓ	0.15 (0.97)	0.14 (0.95)	0
(1, 0.1, 1)	x,w	0.11 (0.73)	0.11 (0.70)	0
	x,w_ℓ	0.08 (0.55)	0.08 (0.53)	0
(1, 1, 1)	x,w	0.11 (0.74)	0.11 (0.70)	0
	x,w_ℓ	0.08 (0.54)	0.08 (0.52)	0
(3, 0.001, 1)	x,w	0.14 (0.94)	0.14 (0.92)	0
	x,w_ℓ	0.13 (0.89)	0.13 (0.87)	0
TPS (GCV)	x	0.11 (0.71)	0.08 (0.53)	0

References

1. Cressie N. *Statistics for spatial data*, revised ed. Wiley Series in probability and mathematical statistics. New York: John Wiley and Sons, Inc, 1993.
2. Schabenberger O, Gotway C. *Statistical methods for spatial data analysis*. Texts in Statistical Science. Boca Raton: Chapman & Hall, CRC, 2005.
3. Zaier I, Shu C, Ouarda T, Seidou O, Chebana F. Estimation of ice thickness on lakes using artificial neural network ensembles. J Hydrol 2010;383:330–40.
4. Chen J, Wang C. Using stacked generalization to combine SVMs in magnitude and shape feature spaces for classification of hyperspectral data. IEEE Trans Geosci Remote Sensing 2009;47:2193–205.
5. Rossi M, Guzzetti F, Reichenbach P, Mondini AC, Peruccacci S. Optimal landslide susceptibility zonation based on multiple forecasts. Geomorphology 2010;114:129–42.
6. Kleiber W, Raftery A, Gneiting T. Geostatistical model averaging for locally calibrated probabilistic quantitative precipitation forecasting. J Am Stat Assoc 2011;106:1291–303.
7. Polley E, van der Laan M. Super learner in prediction U.C. Berkeley Division of Biostatistics Working Paper Series, 2010.
8. Wolpert D. Stacked generalization. Neural Networks 1992;5:241–59.
9. Breiman L. Stacked regressions. Machine Learning 1996;24:49–64.
10. LeBlanc M, Tibshirani R. Combining estimates in regression and classification. J Am Stat Assoc 1996;91:1641–50.
11. Stone M. Cross-validatory choice and assessment of statistical procedures. J R Stat Soc Ser B 1974;36:111–47.
12. Geisser S. The predictive sample reuse method with applications. J Am Stat Assoc 1975;70:320–8.
13. Polley E, Rose S, van der Laan M. Targeted learning: casual inference for observational and experimental data. New York: Springer, chapter 3: Super Learning, 43–65, 2011.
14. Opsomer J, Wang Y, Yang Y. Nonparametric regression with correlated errors. Stat Sci 2001;16:134–53.
15. Francisco-Fernandez M, Opsomer J. Smoothing Parameter Selection Methods for Nonparametric Regression with Spatially Correlated Errors. The Canadian Journal of Statistics 2005;33:279–95.
16. van der Laan M, Dudoit S. Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples U.C. Berkeley Division of Biostatistics Working Paper Series, 2003.
17. van der Vaart A, Dudoit S, van der Laan M. Oracle inequalities for multi-fold cross validation. Stat Decisions 2006;24:351–71.
18. Arlot S, Celisse A. A survey of cross-validation procedures for model selection. Stat Surv 2010;4:44–79.
19. Lumley T. An empirical process limit theorem for sparsely correlated data UW Biostatistics Working Paper Series, 2005.
20. Jiang W. On uniform deviations of general empirical risks with unboundedness, dependence, and high dimensionality. J Mach Learning Res 2009;10:977–96.
21. Davis B. Uses and abuses of cross-validation in geostatistics. Math Geol 1987;19:241–8.
22. Todini E. Influence of parameter estimation on uncertainty in Kriging: Part 1 – theoretical development. Hydrol Earth Syst Sci 2001;5:215–23.
23. Matérn B. Spatial variation, 2nd ed. New York: Springer, 1986.
24. Huang H, Chen C. Optimal geostatistical model selection. J Am Stat Assoc 2007;102:1009–24.
25. Gu C. Smoothing spline ANOVA models. New York: Springer, 2002.

26. R Development Core Team. R: A language and environment for statistical computing, r foundation for statistical computing, Vienna, Austria. Available at: http://www.R-project.org/, 2012.
27. Neugebauer R, Bullard J. DSA: Deletion/Substitution/Addition algorithm. Available at: http://www.stat.berkeley.edu/laan/Software/, r package version 3.1.4, 2010.
28. Hastie T. gam: generalized additive models. Available at: http://CRAN.R-project.org/package=gam, r package version 1.04.1, 2011.
29. Karatzoglou A, Smola A, Hornik K, Zeileis A. Kernlab – an S4 Package for Kernel Methods in R. J Stat Software 2004;11:1–20. Available at: http://www.jstatsoft.org/v11/i09/.
30. Ridgeway G. gbm: Generalized boosted regression models. Available at: http://CRAN.R-project.org/package=gbm, r package version 1.6-3.1, 2010.
31. Friedman J, Hastie T, Tibshirani R. Regularization paths for generalized linear models via coordinate descent. J Stat Software 2010;33. Available at: http://www.jstatsoft.org/v33/i01/.
32. Li S. FNN: fast nearest neighbor search algorithms and applications. Available at: http://CRAN.R-project.org/package=FNN , r package version 0.6-3, 2012.
33. PJ, PD, Ribeiro J. Model based geostatistics. New York: Springer, 2007.
34. PJ Ribeiro J, Diggle P. 2001. geoR: a package for geostatistical analysis. R News 2001;1:14–18. Available at: http://CRAN.R-project.org/doc/Rnews/.
35. Kooperberg C. polspline: polynomial spline routines. Available at: http://CRAN.R-project.org/package=polspline, r package version 1.1.5, 2010.
36. Liaw A, Wiener M. Classification and regression by randomforest. R News 2002;2:18–22. Available at: http://CRAN.R-project.org/doc/Rnews/.
37. Furrer R, Nychka D, Sain S. Fields: tools for spatial data. Available at: http://CRAN.R-project.org/package=fields, r package version 6.6.1, 2011.
38. Sinisi S, van der Laan M. The deletion/substitution/addition algorithm in loss function based estimation. J Stat Methods Mol Biol 2004;3:Article 18.
39. Hastie T. Statistical models in S, Wadsworth and Brooks/Cole, chapter 7: Generalized Additive Models, 1991.
40. Williams C. Learning in graphical models. Cambridge, MA: The MIT Press, chapter Prediction with Gaussian processes: from linear regression to linear prediction and beyond, 599–621, 1999.
41. Friedman J. Greedy function approximation: a gradient boosting machine. Ann Stat 2001;29:367–78.
42. Gelfand A, Diggle P, Fuentes M, Guttorp P, editors. Handbook of spatial statistics. Boca Raton: CRC Press, 2010.
43. Stone C, Hansom M, Kooperberg C, Truong Y. The use of polynomial splines and their tensor products in extended linear modeling (with discussion). Ann Stat 1997;25:1371–470.
44. Breiman L. Random forests. Machine Learning 2001;45:5–32.
45. Green P, Silverman B. Nonparametric regression and generalized linear models. number 58 in Monographs on Statistics and Applied Probability. Boca Raton: Chapman & Hall/CRC, 1994.
46. Craven P, Wahba G. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. Numer Math 1979;31:377–403.
47. Ellers J, Landers D, Brakke D. Chemical and Physical Characteristics of Lakes in the Southeastern United States. Environmental Science and Technology 1988;22:172–7.
48. Gu C. gss: general smoothing splines. Available at: http://CRAN.R-project.org/package=gss, r package version 2.0-11, 2012.
49. Ledell E. h2oEnsemble. Available at: http://www.stat.berkeley.edu/~ledell/software.html, 2015.
50. Lendle S. OnlineSuperLearner. Available at: https://github.com/lendle/OnlineSuperLearner.jl, 2015.
51. Carmack P, Schucany W, Spence J, Gunst R, Lin Q, Haley R. Far casting cross-validation. J Comput Graph Stat 2009;18:879–93.
52. Györfi L, Kohler M, Walk H. A distribution-free theory of nonparametric regression. Springer Series in Statistics. New York: Springer, 2002.
53. van der Laan M, Dudoit S, Keles S. Asymptotic optimality of likelihood based cross-validation. Stat Appl Genet Mol Biol 2004;3:Article 4.
54. Dudoit S, van der Laan M. Asymptotics of cross-validated risk estimation in estimator selection and performance assessment. Stat Methodol 2005;2:131–54.