Surrogate Modeling

1 Introduction

Capturing uncertainty in high-fidelity computational models can be costly on many domains or even infeasible. Here we follow the uncertainty quantification literature in building so called *surrogate models*, which are designed to approximate the input-output relation of complex models.

In the following we describe the nonstandard data we are working with and how we can use surrogate models to capture the uncertainties present in the model. We describe various different surrogate models in section 2.

1.1 Uncertainty Propagation

Model Calibration Say we want to analyze a model $\mathcal{M}$ that is parameterized with a parameter vector $\theta \in \Theta \subset \mathbb{R}^d$. Given a specific research question, we are interested in a *quantity of interest* (qoi), that we can compute from the model. We can write this relationship in a functional form, $qoi = \tilde{\mathcal{M}}(\theta)$ –here we use a similar notation as for the model since the functional form, i.e. the mathematical relationship between the quantity of interest and the parameters can be derived from the model, but it does not characterize the model in general. For notational convenience we will write $y = \tilde{\mathcal{M}}(\theta)$ hereinafter.

Further, we assume that we obtained estimates $\hat{\theta}$ for θ, for example by using the maximum-likelihood technique, with corresponding (approximate) covariance matrix $\hat{\Sigma} = var(\hat{\theta})$. We also note that these estimates are calibrated on an observational data set $\{(Y_i, X_i) : i \in \mathcal{I}\}$, where Y_i and X_i denote the observed outcome of interest and observed covariates of individual i, respectively. We emphasize that for all further analyses we only need the observed data through the estimates of the parameter vector $\hat{\theta} = \hat{\theta}(\{(Y_i, X_i) : i \in \mathcal{I}\})$ and its dispersion $\hat{\Sigma} = \hat{\Sigma}(\{(Y_i, X_i) : i \in \mathcal{I}\})$.

Propagation of Uncertainty Given what we have found above, our general goal is to propagate the uncertainty in the model parameters, represented by its approximate distribution $\mathcal{N}(\hat{\theta}, \hat{\Sigma})$, through the model. That is, if we let $\boldsymbol{\theta} \sim \mathcal{N}(\hat{\theta}, \hat{\Sigma})$, we are interested in the distribution of $\tilde{\mathcal{M}}(\boldsymbol{\theta})$. For (very) simple models this distribution can be computed in analytical form, however, for most models an analytical approach is infeasible.

Experimental Design To approximate $\tilde{\mathcal{M}}(\boldsymbol{\theta})$ we use a standard Monte Carlo technique in which we sample $\theta_1, \ldots, \theta_n$ from $\mathcal{N}(\hat{\theta}, \hat{\Sigma})$ for a reasonable large n. For each $i = 1, \ldots, n$, we then compute the quantity of interest using the high-fidelity model, i.e. $y_i = \tilde{\mathcal{M}}(\theta_i)$. The resulting (simulated) data set will be denoted by $\mathcal{D} := \{(y_i, \theta_i) : i = 1, \ldots, n\}$.

If it is possible to create such a data set for a reasonable large n, our analysis is done. We could then approximate the distribution of $\tilde{\mathcal{M}}(\boldsymbol{\theta})$ by the empirical distribution of $\{y_i : i = 1, \ldots, n\}$. However, for many complex models the evaluation of $\tilde{\mathcal{M}}$ at some

point θ can be slow, even on modern computers. Further, since we aim at propagating the uncertainty of θ through the model, we must draw values for θ from all regions where the mass of $\boldsymbol{\theta}$ is non-negligible. With standard models used in Economics the dimension of the parameter vector can easily exceed 50, which means we have to sample from a 50 dimensional space. Therefore, for most models, computation of the quantity of interest using an exhaustive sample is infeasible. This is why we use surrogate models which are designed to be very fast to evaluate.

1.2 Model Validation

Let $f : \mathbb{R}^{\dim \theta} \to \mathbb{R}^{\dim y}$ denote a generic surrogate model. f has to accomplish two main tasks. One, it has to be fast to evaluate. And two, it has to approximate the model well in all regions of interest. That is, $f(\theta) \approx \tilde{\mathcal{M}}(\theta)$ for all θ in the regions of interests. While the first task is problem dependent and hard to formalize, assesing the prediction error is well-defined and common practice among many disciplines.

Since our goal is to evaluate f on new data points, we have to make sure to evaluate the prediction error of f on new data points as well. Common approaches to evaluate model performance are cross-validation or hold-out techniques. We proceed using the hold-out technique, that is, splitting the data set $\mathcal{D}$ into two subsamples $\mathcal{D}^T$ and $\mathcal{D}^V$, used for training (T) and validation (V), respectively. We then *train* our model on the data points in $\mathcal{D}^T$ and *validate* model performance by estimating the out-of-sample mean absolute prediction error on data points in $\mathcal{D}^V$ using

$$\text{MAE}(f, \mathcal{D}^V) := \frac{1}{|\mathcal{D}^V|} \sum_{(y_i, \theta_i) \in \mathcal{D}_S^V} |y_i - f(\theta_i)| \,. \tag{1}$$

In the following section we will present different kind of surrogate models and how they compare with respect to the estimated mean absolute error. For complex models it can already be computationally very costly to generate the data set $\mathcal{D}$ for $n > 1000$. This is why we repeat our analysis for varying training sizes, which stretch from 100 to 75000.

2 Surrogate Models

Here we present various surrogate models which were used to approximate the model. We will write n^τ for a fixed training sample size.

For notational convenience we further write $\boldsymbol{\Omega} = (\theta_1^\top, \ldots, \theta_{n^\tau}^\top)^\top \in \mathbb{R}^{n^\tau \times d}$, where the rows of $\boldsymbol{\Omega}$ denote the different draws from the asymptotic distribution used for training and the columns the different dimensions/ parameters of the model. Further, we write $y = (y_1, \ldots, y_{n^\tau})^\top \in \mathbb{R}^{n^\tau}$.

2.1 Linear Regression

Model:

$$y_i = \beta_0 + \sum_{j=1}^{d} \beta_j \, \theta_i^{(j)} + \epsilon_i \tag{2}$$

$$= \beta_0 + \beta^\top \theta_i + \epsilon_i \tag{3}$$

Estimation:

$$\beta_0^*, \beta^* = \underset{b_0 \in \mathbb{R}, \boldsymbol{b} \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \|y - b_0 - \boldsymbol{\Omega b}\|_2^2 \right\} \tag{4}$$

Prediction:

$$y_{test} = f_{linreg}(\theta_{test}) = \beta_0^* + \beta^* \theta_{test} \tag{5}$$

2.1.1 Results

n	100	200	500	700	1000	2000	5000	7000	10000	20000	50000	70000
mae	0.0093	0.0088	0.0086	0.0086	0.0085	0.0084	0.0083	0.0083	0.0083	0.0083	0.0083	0.0083

Table 1: Linear regression

2.2 Polynomial Regression (2nd degree)

Model:

$$y_i = \beta_0 + \sum_{j=1}^{d} \beta_{1,j}\, \theta_i^{(j)} + \sum_{j=1}^{d} \beta_{2,j}\, (\theta_i^{(j)})^2 + \sum_{j \neq k} \beta_{3,jk}\, \theta_i^{(j)} \theta_i^{(k)} + \epsilon_i \tag{6}$$

Estimation: Write $\phi(\theta)$ for the vector containing linear terms, squared terms and interaction terms. Then, $y_i = \beta_0 + \beta\, \phi(\theta_i) + \epsilon_i$ and

$$\beta_0^*, \beta^* = \underset{b_0 \in \mathbb{R}, \boldsymbol{b} \in \mathbb{R}^{(2d + \binom{d}{2})}}{\operatorname{argmin}} \left\{ \|y - b_0 - \phi(\boldsymbol{\Omega}) \boldsymbol{b}\|_2^2 \right\} \tag{7}$$

Prediction:

$$y_{test} = f_{polreg}(\theta_{test}) = \beta_0^* + \beta^* \phi(\theta_{test}) \tag{8}$$

2.2.1 Results

n	500	700	1000	2000	5000	7000	10000	20000	50000	70000
mae	0.013	0.0084	0.007	0.0061	0.0057	0.0056	0.0055	0.0055	0.0054	0.0054

Table 2: Linear regression with 2nd degree polynomial features

2.3 Ridge Regression (Tikhonov regularization)

The main idea behind using regularization methods is to avoid overfitting and therefore increase prediction accuracy on test data.

Model: Same as equation 2 or equation 6, depending on the degree of the polynomial used.

Estimation: Say we use a polynomial model of degree 1, then

$$\beta_0^*, \beta^*(\alpha) = \underset{b_0 \in \mathbb{R}, \boldsymbol{b} \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \|y - b_0 - \Omega \boldsymbol{b}\|_2^2 + \alpha \|\boldsymbol{b}\|_2^2 \right\}, \tag{9}$$

and $\beta^* = \beta^*(\alpha^*)$, where α^* is chosen via cross validation on the training set.

Prediction: Same as equation 5 or equation 8, depending on the degree of the polynomial used, but using the regularized coefficients.

2.3.1 Results

n	100	200	500	700	1000	2000	5000	7000	10000	20000	50000	70000
mae	0.0261	0.0091	0.0072	0.007	0.0067	0.0061	0.0057	0.0056	0.0055	0.0055	0.0054	0.0054

Table 3: Ridge regression

2.4 Neural Networks

Work in progress.

2.4.1 Results

n	100	200	500	700	1000	2000	5000	7000	10000	20000	50000	70000
mae	0.2749	0.1795	0.0656	0.0387	0.0232	0.0091	0.0078	0.0078	0.008	0.0066	0.0067	0.0064

Table 4: Small neural network

n	100	200	500	700	1000	2000	5000	7000	10000	20000	50000	70000
mae	0.2184	0.1158	0.0299	0.0245	0.0204	0.0125	0.0091	0.0068	0.008	0.0061	0.0062	0.006

Table 5: Large neural network

n	100	200	500	700	1000	2000	5000	7000	10000	20000	50000	70000
mae	0.1878	0.0881	0.0274	0.0245	0.0204	0.013	0.0083	0.0075	0.0083	0.0065	0.0065	0.0056

Table 6: Huge neural network

n	100	200	500	700	1000	2000	5000	7000	10000	20000	50000	70000
mae	0.1498	0.058	0.0286	0.0281	0.0224	0.0152	0.0094	0.0078	0.008	0.0092	0.0058	0.0062

Table 7: Deep neural network

3 Results

Here we compare the performance of the different surrogate models along two different margins that might be important to the applied researcher: the number of observations in the training data n^τ, i.e. the number of draws from the asymptotic distribution, as this can be computationally intensive and along the number of parameters that are included in the prediction.

3.1 Varying n^τ

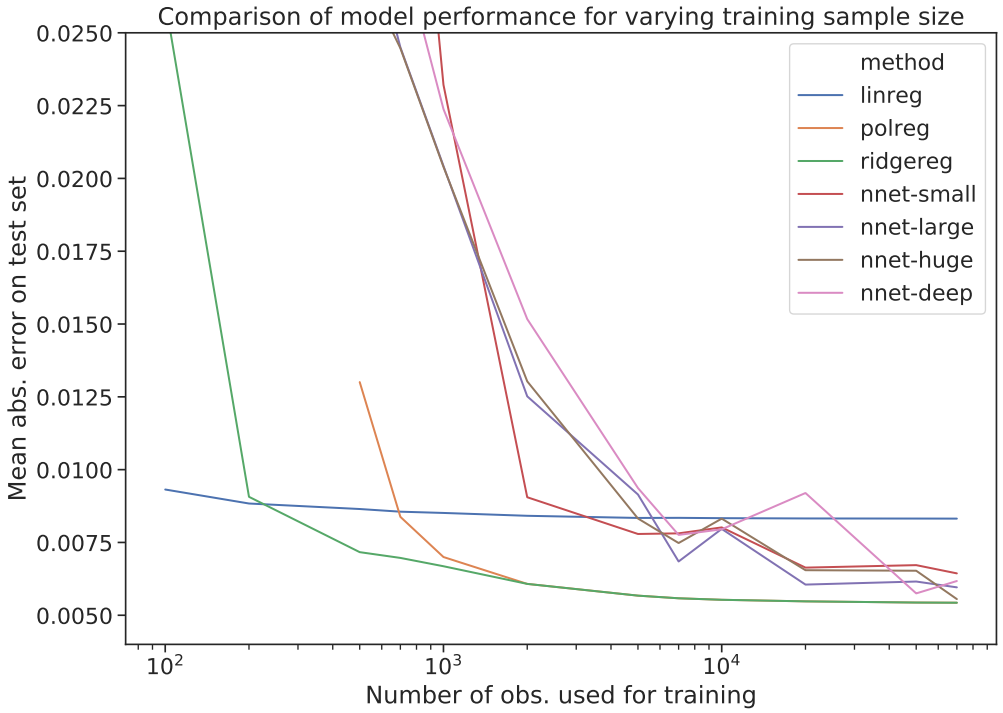

Figure 1: Comparison of model performance for varying training sample size. Number of training observations n^τ are depicted using a logarithmic scale on the x-axis. Mean absolute prediction error on the validation set $\mathcal{D}^V$ is depicted on the y-axis.

For our data we can clearly see that the Ridge regression model using polynomial features and standard linear model using polynomial model outperform all other models. Importantly they already do so for very few training observations.

3.2 Variable selection

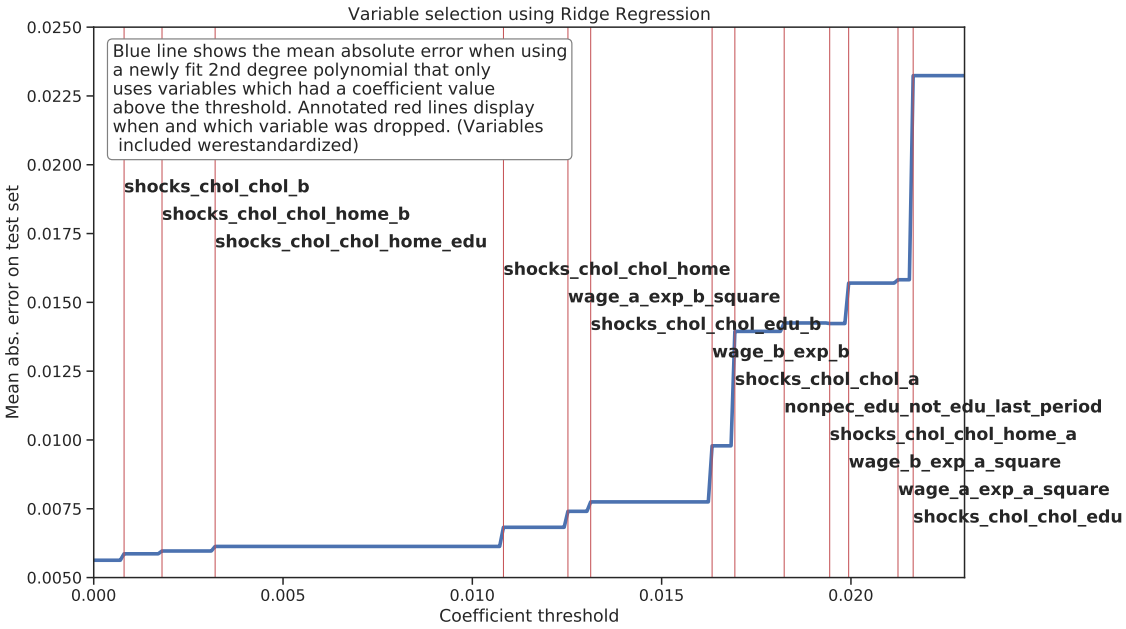

Figure 2: Blue line shows mean absolute error when using a newly fit 2nd degree polynomial that only uses variables which had a coefficient value above the threshold. Annotated red lines display when and which variable was dropped. (Input variables were standardized beforehand to assure comparaibility.)