Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

Cross-Validated Kernel Ensemble: Robust Hypothesis Test for Nonlinear Effect with Gaussian Processes

Wenying Deng
Harvard University

Jeremiah Zhe Liu
Harvard University

Brent Coull
Harvard University

Abstract

The R package **CVEK** introduces a robust hypothesis test for nonlinear effect with gaussian processes. CVEK is an ensemble-based estimator that adaptively learns the form of the main-effect kernel from data, and constructs an companion variance component test. Package **CVEK** implements the estimator for two testing procedures, namely asymptotic test and bootstrap test. Additionally, it implements a variety of tuning parameter criteria, including Akaike Information Criteria, Generalized Cross Validation, Generalized Maximum Profile Marginal Likelihood and leave-one-out Cross Validation. Moreover, there are three kinds of ensemble strategies to create the ultimate ensemble kernel: Simple Averaging, Empirical Risk Minimization and Exponential Weighting. The null distribution of the test statistic can be approximated using a scaled chi-square distribution, and therefore statistical inference based on the results of this package, such as hypothesis testing can be performed. Extensive simulations demonstrate the robustness and correct implementation of the estimator.

Keywords: robust hypothesis test, nonlinear effect, gaussian processes, **CVEK**, R.

1. Introduction

In recent years, kernel machine-based hypothesis tests (e.g. SKAT) for high-dimensional, nonlinear effects has seen widespread application in GWAS and gene-environment interaction studies. However, constructing a test for the interaction between groups of continuous features (for example, interaction between groups of air pollutants and multi-category nutrition intake) remains difficult in practice. The main challenges root from (1) constructing an appropriate main-effect kernel that induces unbiased estimator for the null model, and (2) constructing an appropriate interaction-effect kernel describing only the effect of between-groups interaction, which is necessary for building a valid test statistic. Recently, (Liu and Coull 2017) addressed the first challenge by proposing Cross-Validated Ensemble of Kernels (CVEK), an ensemble-based estimator that adaptively learns the form of the main-effect kernel from data, and constructs an companion variance component test. While interesting, the null distribution of CVEK is constructed using asymptotic approximation, and requires the interaction-kernel to be fixed a priori, therefore calling into question the validity of the test in limited sample, and prevents practitioners from deploying flexible methods to learn the interaction kernel from data. In this work, we seek to address these shortcomings by proposing a bootstrap test for CVEK. We conduct comprehensive simulation study to evaluate the validity (i.e. Type I error) and power of the proposed test using diverse choices of modeling strategy and under a wide range of data-generation mechanisms. Our simulation results revealed valuable insight on the impact of choice of estimation strategy (i.e. choices of tuning-parameter selection criteria and ensemble strategy) on the performance of the resulting hypothesis test.

2. Models and software

2.1. Cross-Validated Ensemble of Kernels

Overview

Cross-Validated Ensemble of Kernels (CVEK) (Liu and Coull 2017) is an ensemble-based method for an unknown data-generating function $h : \mathbb{R}^p \to \mathbb{R}$. Traditional practices for estimating h, for example Gaussian Process (GP) regression using a single kernel function, tends to impose *a priori* assumption on the mathematical property of h by specifying the reproducing kernel function k for $h \in \mathcal{H}$, thereby risking inducing biased estimation and incorrect inference. To circumvent this issue, CVEK proposes estimating h using the ensemble of GP predictions generated from a library of (fixed) base kernel functions $\{k_d\}_{d=1}^D$:

$$\hat{h}(\mathbf{x}) = \sum_{d=1}^{D} u_d \hat{h}_d(\mathbf{x}), \quad \mathbf{u} \in \Delta = \{\mathbf{u} \mid \mathbf{u} \geq 0, \parallel \mathbf{u} \parallel_1 = 1\}, \tag{2.1.1}$$

where $\hat{h}_d$ is the kernel predictor generated by d^{th} base kernel k_d.

To be more specific, for each given basis kernel $\{k_d\}_{d=1}^D$, CVEK first estimates $\hat{\mathbf{h}}_d = \mathbf{K}_d(\mathbf{K}_d + \hat{\lambda}_d \mathbf{I})^{-1}\mathbf{y}$, the prediction based on d^{th} kernel, where the tuning parameter $\hat{\lambda}_d$ is selected by minimizing certain criteria (see section 2.1.2). After which, CVEK combines the individual base kernel predictors $\{\hat{h}_d\}_{d=1}^D$ according to a certain ensemble strategy (see section 2.1.3). In addition to producing ensemble prediction $\hat{h}$, CVEK also produces an ensemble kernel matrix $\hat{\mathbf{K}}$ which describes the mathematical property of the ensemble RKHS. $\hat{\mathbf{K}}$ is estimated by solving:

$$\hat{\mathbf{K}}(\hat{\mathbf{K}} + \lambda \mathbf{I})^{-1} = \hat{\mathbf{A}}s.$$

In fact, $\hat{\mathbf{K}}$ can be computed in closed form. If we denote $\mathbf{U}_A$ and $\{\delta_{A,k}\}_{k=1}^n$ the eigenvectors and eigenvalues of $\hat{\mathbf{A}}$, then:

$$\hat{\mathbf{K}} = \mathbf{U}_A \text{diag}(\frac{\delta_{A,k}}{1 - \delta_{A,k}}) \mathbf{U}_A^\top.$$

A complete summary of the proposed procedure (using LooCV for tuning-parameter selection and empirical risk minimization for ensemble strategy) is available in Algorithm 1.

Tuning Parameter Selection

Models may provide a good fit to the training data, but it will not fit sufficiently well to the test data. Tuning parameter could be chosen to address this problem. Here we define four objective functions in terms of tuning parameter $\lambda \in \Lambda$ to be minimized. Denote

$$\mathbf{A}_\lambda = \mathbf{K}(\mathbf{X},\mathbf{X})[\mathbf{K}(\mathbf{X},\mathbf{X}) + \lambda\mathbf{I}]^{-1}. \tag{2.1.2}$$

In this way, $tr(\mathbf{A}_\lambda)$ is the effective number of model parameters, excluding μ and σ^2. It decreases monotonically with $\lambda > 0$. Additionally, we denote $\mathbf{y}^\star$ as the centered $\mathbf{y}$: $\mathbf{y}^\star = \mathbf{y} - \hat{\boldsymbol{\mu}}$, where $\hat{\mu} = \frac{1}{n}\sum_{i=1}^n y_i$.

K-Fold and Leave-one-out Cross Validation

Cross validation is probably the simplest and most widely used method for estimating prediction error. Suppose we do a K-fold cross-validation, which partitions observations into K groups, $\kappa(1), ..., \kappa(K)$, and calculates $\mathbf{A}_\lambda$ K times, each time leaving out group $\kappa(i)$, to get $\mathbf{A}_\lambda^{-\kappa(1)}, \mathbf{A}_\lambda^{-\kappa(2)}$, etc. For $\mathbf{A}_\lambda^{-\kappa(i)}$, cross-validated residuals are calculated on the observations in $\kappa(i)$, which did not contribute to estimating $\mathbf{A}$. The objective function estimates prediction error and is the sum of the squared cross-validated residuals:

$$\lambda_{K-CV} = \underset{\lambda\in\Lambda}{\text{argmin}} \Big\{\log \sum_{i=1}^{K} [\mathbf{y}^\star_{\kappa(i)} - \mathbf{A}_\lambda^{-\kappa(i)}\mathbf{y}^\star_{\kappa(i)}]^\top [\mathbf{y}^\star_{\kappa(i)} - \mathbf{A}_\lambda^{-\kappa(i)}\mathbf{y}^\star_{\kappa(i)}]\Big\}. \tag{2.1.3}$$

LooCV is the situation when $K = n$. In this case, we can write our objective function as (Golub, Heath, and Wahba 1979):

$$\lambda_{n-CV} = \underset{\lambda\in\Lambda}{\text{argmin}} \Big\{\log \mathbf{y}^{\star T}[\mathbf{I} - \text{diag}(\mathbf{A}_\lambda) - \frac{1}{n}\mathbf{I}]^{-1}(\mathbf{I} - \mathbf{A}_\lambda)^2[\mathbf{I} - \text{diag}(\mathbf{A}_\lambda) - \frac{1}{n}\mathbf{I}]^{-1}\mathbf{y}^\star\Big\}. \tag{2.1.4}$$

The value K influences bias and variance of cross-validation. With $K = n$, the cross-validation estimator is approximately unbiased for the true (expected) prediction error because it almost use all the data in each training set. Therefore, it can have high variance because n training sets are so similar to one another. Additionally, the computational burden is also considerable, requiring n applications of the learning method. On the other hand, with larger K such as 5 (Hastie, Tibshirani, and Friedman 2009) or 10, cross-validation will have lower variance, but making bias a problem.

Akaike Information Criteria

Based on the idea of "model fit + model complexity", Akaike's Information Criterion (AIC) (Akaike 1998) chooses λ by minimizing,

$$\begin{aligned} AIC =& 2(p+2) - 2\log(\hat{L}) \\ =& 2(p+2) - 2[-\frac{n}{2}\log(\hat{\sigma}^2) - \frac{n}{2}\log(2\pi) - \frac{1}{2\hat{\sigma}^2}\mathbf{y}^{\star\top}(\mathbf{I} - \mathbf{A}_\lambda)^2\mathbf{y}^\star] \\ =& 2(p+2) + n\log[\frac{1}{n}\mathbf{y}^{\star\top}(\mathbf{I} - \mathbf{A}_\lambda)^2\mathbf{y}^\star] + n + n\log(2\pi). \end{aligned}$$

Drop the constant n and divide it by n, we obtain our objective function:

$$\lambda_{AIC} = \underset{\lambda \in \Lambda}{\text{argmin}} \Big\{ \log \mathbf{y}^{\star\top}(\mathbf{I} - \mathbf{A}_\lambda)^2 \mathbf{y}^\star + \frac{2[\text{tr}(\mathbf{A}_\lambda) + 2]}{n} \Big\}. \tag{2.1.5}$$

When n is small, extreme overfitting is possible, giving small bias/ large variance estimates. The small-sample correction of AIC (Hurvich and Tsai 1989; Hurvich Clifford M., Simonoff Jeffrey S., and Tsai Chih-Ling 2002) is derived by minimizing minus 2 times expected log likelihood, where we plug in $\mathbf{A}_\lambda$ and $\hat{\sigma}^2$. In this case, we obtain our small-sample size objective function AICc:

$$\lambda_{AICc} = \underset{\lambda \in \Lambda}{\text{argmin}} \Big\{ \log \mathbf{y}^{\star\top}(\mathbf{I} - \mathbf{A}_\lambda)^2 \mathbf{y}^\star + \frac{2[\text{tr}(\mathbf{A}_\lambda) + 2]}{n - \text{tr}(\mathbf{A}_\lambda) - 3} \Big\}. \tag{2.1.6}$$

Compare (2.1.5) and (2.1.6) , it is easy to tell that AICc considers more of the model complexity since $\frac{n}{n - \text{tr}(\mathbf{A}_\lambda) - 3} > 1$. It makes sense intuitively because it need to shrink more to prevent small bias/ large variance estimates.

Generalized Cross Validation

In (2.1.4), if we approximate each $A_{\lambda[ii]}$ with their mean $\frac{\text{tr}(\mathbf{A}_\lambda)}{n}$, in a sense that we give equal weight to all observations. We get the Generalized Cross Validation (GCV) objective function:

$$\lambda_{GCV} = \underset{\lambda \in \Lambda}{\text{argmin}} \Big\{ \log \mathbf{y}^{\star\top}(\mathbf{I} - \mathbf{A}_\lambda)^2 \mathbf{y}^\star - 2\text{log}[1 - \frac{\text{tr}(\mathbf{A}_\lambda)}{n} - \frac{1}{n}] \Big\}. \tag{2.1.7}$$

The "$-\frac{1}{n}$" terms in (2.1.7) is because GCV counts μ as part of model complexity, but not σ^2. This motivates the proposed small-sample correction to GCV (Boonstra, Mukherjee, and Taylor 2015), which does count σ^2 as a parameter:

$$\lambda_{GCVc} = \underset{\lambda \in \Lambda}{\text{argmin}} \Big\{ \log \mathbf{y}^{\star\top}(\mathbf{I} - \mathbf{A}_\lambda)^2 \mathbf{y}^\star - 2\text{log}[1 - \frac{\text{tr}(\mathbf{A}_\lambda)}{n} - \frac{2}{n}]_+ \Big\}. \tag{2.1.8}$$

Under this situation, perfect fit of the observations to the predictions, given by $\lambda = 0$, cannot occur.

Generalized Maximum Profile Marginal Likelihood

Suppose we relate Y and $\mathbf{X}$ by a linear model, $Y = \mu + \phi(\mathbf{X})^\top \boldsymbol{\beta} + \epsilon$, with $\epsilon \sim \text{N}(0, \sigma^2)$, where $\phi(\mathbf{x})$ is a function mapping a D-dimensional input vector $\mathbf{x}$ into an p-dimensional feature space. If we assume $\boldsymbol{\beta}$ are jointly and independently normal with mean zero and variance σ^2/λ, the penalty term matches the negative normal log-density, up to a normalizing constant not depending on $\boldsymbol{\beta}$:

$$p_\lambda(\boldsymbol{\beta}, \sigma^2) = \frac{\lambda}{2\sigma^2} \boldsymbol{\beta}^\top \boldsymbol{\beta} - \frac{p}{2}\text{log}(\lambda) + \frac{p}{2}\text{log}(\sigma^2).$$

One can consider a marginal likelihood, where λ is interpreted as the variance component of a mixed-effects model:

$$\begin{aligned} m(\lambda, \sigma^2) =& \text{log} \int_{\boldsymbol{\beta}} \exp\{l(\boldsymbol{\beta}, \sigma^2) - p_\lambda(\boldsymbol{\beta}, \sigma^2)\} d\boldsymbol{\beta} \\ =& -\frac{1}{2\sigma^2} \mathbf{y}^{\star\top}(\mathbf{I} - \mathbf{A}_\lambda)\mathbf{y}^\star - \frac{n}{2}\text{log}(\sigma^2) + \frac{1}{2}\text{log}|\mathbf{I} - \mathbf{A}_\lambda|. \end{aligned}$$

From this, $\mathbf{y}^\star \mid \lambda, \sigma^2$ is multivariate normal with mean $\mathbf{0}_n$ and covariance $\sigma^2(\mathbf{I}-\mathbf{A}_\lambda)^{-1}$. The maximum profile marginal likelihood (MPML) estimate, originally proposed for smoothing spline (Wecker and Ansley 1983), profiles $m(\lambda, \sigma^2)$ over σ^2, replacing each instance with $\hat{\sigma}^2_\lambda = \mathbf{y}^{\star\top}(\mathbf{I}-\mathbf{A}_\lambda)\mathbf{y}^\star/n$, and maximized the "concentrated" log-likelihood, $m(\lambda, \hat{\sigma}^2_\lambda)$:

$$\lambda_{MPML} = \underset{\lambda\in\Lambda}{\operatorname{argmin}}\Big\{log\ \mathbf{y}^{\star\top}(\mathbf{I}-\mathbf{A}_\lambda)\mathbf{y}^\star - \frac{1}{n}\log|\mathbf{I}-\mathbf{A}_\lambda|\Big\}.$$

Closely related is the generalized/ restricted MPML (Harville 1977; Wahba 1985), which adjusts the penalty to account for estimation of regression parameter β_0 that is not marginalized, resulting in one degree of freedom:

$$\lambda_{GMPML} = \underset{\lambda\in\Lambda}{\operatorname{argmin}}\Big\{log\ \mathbf{y}^{\star\top}(\mathbf{I}-\mathbf{A}_\lambda)\mathbf{y}^\star - \frac{1}{n-1}\log|\mathbf{I}-\mathbf{A}_\lambda|\Big\}. \tag{2.1.9}$$

Ensemble Strategy

Empirical Risk Minimization

After obtaining the estimated errors $\{\hat{\boldsymbol{\epsilon}}_d\}_{d=1}^D$, we estimate the ensemble weights $\mathbf{u} = \{u_d\}_{d=1}^D$ such that it minimizes the overall error (Liu and Coull 2017):

$$\hat{\mathbf{u}} = \underset{\mathbf{u}\in\Delta}{\operatorname{argmin}} \parallel \sum_{d=1}^D u_d \hat{\boldsymbol{\epsilon}}_d \parallel^2 \quad \text{where } \Delta = \{\mathbf{u} \mid \mathbf{u} \geq 0, \parallel \mathbf{u} \parallel_1 = 1\}.$$

Then we produce the final ensemble prediction:

$$\hat{\mathbf{h}} = \sum_{d=1}^D \hat{u}_d \mathbf{h}_d = \sum_{d=1}^D \hat{u}_d \mathbf{A}_{d,\hat{\lambda}_d}\mathbf{y}^\star = \hat{\mathbf{A}}\mathbf{y}^\star,$$

where $\hat{\mathbf{A}} = \sum_{d=1}^D \hat{u}_d \mathbf{A}_{d,\hat{\lambda}_d}$ is the ensemble matrix.

Simple Averaging

Motivated by existing literature in omnibus kernel (Zhan, Plantinga, Zhao, and Wu 2017), we propose another way to obtain the ensemble matrix by simply choosing unsupervised weights $u_d = 1/D$ for $d = 1, 2, ...D$.

Exponential Weighting

Additionally, (Dalalyan and Tsybakov 2007) gives a new strategy to calculate weights based on the estimated errors $\{\hat{\boldsymbol{\epsilon}}_d\}_{d=1}^D$.

$$u_d(\beta) = \frac{\exp(- \parallel \hat{\boldsymbol{\epsilon}}_d \parallel_2^2 / \beta)}{\sum_{d=1}^D \exp(- \parallel \hat{\boldsymbol{\epsilon}}_d \parallel_2^2 / \beta)}$$

Kernel Choice

In this section (Press 2006) we give introductions to some commonly-used kernel functions, including two stationary covariance functions (gaussian rbf, mat$\acute{e}$rn and rational quadratic), as well as non-stationary covariance functions (polynomial and neural network). For convenience, we define $r = |\mathbf{x} - \mathbf{x}'|$.

Gaussian RBF Kernels

The Gaussian radial basis function (RBF), also known as *squared exponential* (SE) kernel function has the form

$$k_{SE}(r) = \exp\Big(-\frac{r^2}{2l^2}\Big),$$

with parameter l defining the *characteristic length-scale.* This covariance function is infinitely differentiable, which means that the GP with this covariance function has mean square derivatives of all orders, and is thus very smooth. The spectral density of SE covariance function is $S(s) = (2\pi l^2)^{D/2}\exp(-2\pi^2 l^2 s^2)$. (Stein 1999) argues that such strong smoothness assumptions are unrealistic for modeling many physical processes, and recommends the Matérn class. However, the squared exponential is probably the most widely-used kernel within the kernel machines field. The SE kernel is *infinitely divisible* in that $(k(r))^\top$ is a valid kernel for all $t > 0$; the effect of raising k to the power of t is simply to rescale l. The eigenvalues are subject to exponential decay: $\lambda_i = O(e^{-\beta i})$ for $\beta > 0$.

Matérn Kernels

The *Matérn class* of covariance functions is given by

$$k_{Matern}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\frac{\sqrt{2\nu r}}{l}\Big)^{\nu} K_\nu\Big(\frac{\sqrt{2\nu r}}{l}\Big),$$

with positive parameters ν and l, where K_ν is a modified Bessel function (Abramowitz 1974). This covariance function has a spectral density

$$S(s) = \frac{2^D\pi^{D/2}\Gamma(\nu + D/2)(2\nu)^\nu}{\Gamma(\nu)l^{2\nu}}\Big(\frac{2\nu}{l^2} + 4\pi^2 s^2\Big)^{-(\nu+D/2)}$$

in D dimensions. For the Matérn class the process $f(\mathbf{x})$ is k-times MS differentiable is and only if $\nu > k$. The Matérn covariance functions become especially simple when ν is half-integer: $\nu = p + 1/2$, where p is a non-negative integer. In this case the covariance function is a product of an exponential and a polynomial of order p, the general expression can be derived from (Abramowitz 1974), giving

$$k_{\nu=p+1/2}(r) = \exp\Big(-\frac{\sqrt{2\nu r}}{l}\Big)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!(p-i)!}\Big(\frac{\sqrt{8\nu r}}{l}\Big)^{p-i}.$$

It is possible that the most interesting cases for machine learning are $\nu = 3/2$ and $\nu = 5/2$, since for $\nu = 1/2$ the process becomes very rough and for $\nu \geq 7/2$, in the absence of explicit prior knowledge about the existence of higher order derivatives, it is probably very hard from finite noisy training examples to distinguish between $\nu \geq 7/2$. The eigenvalues are subject to polynomial decay: $\lambda_i = O(i^{-\alpha})$ for $\alpha > 1$.

Rational Quadratic Kernels

The *rational quadratic* (RQ) covariance function

$$k_{RQ}(r) = \Big(1 + \frac{r^2}{2\alpha l^2}\Big)^{-\alpha},$$

with α, $l > 0$ can be seen as a *scale mixture* (an infinite sum) of squared exponential (SE) covariance functions with different characteristic length-scales (sum of covariance functions are also a valid covariance). Parameterizing now in terms of inverse squared length scales, $\tau = l^{-2}$, and putting a gamma distribution on $p(\tau \mid \alpha, \beta) \propto \tau^{\alpha-1} exp(-\alpha\tau/\beta)$, we can add up the contributions through the following integral

$$k_{RQ}(r) = \int p(\tau \mid \alpha, \beta) k_{SE}(r \mid \tau) d\tau \propto \int \tau^{\alpha-1} exp\Big(-\frac{\alpha\tau}{\beta}\Big) exp\Big(-\frac{\tau r^2}{2}\Big) d\tau \propto \Big(1 + \frac{r^2}{2\alpha l^2}\Big)^{-\alpha},$$

where we have set $\beta^{-1} = l^2$. The limit of the RQ covariance for $\alpha \to \infty$ is the SE covariance function with characteristic length-scale l.

Polynomial Kernels

It is also interesting to show an explicit feature space construction for the *polynomial covariance function.* We consider the homogeneous polynomial case as the inhomogeneous case can simply be obtained by considering $\mathbf{x}$ to be extended by concatenating a constant. We write

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') =& (\mathbf{x} \cdot \mathbf{x}')^p = (\sum_{d=1}^{D} x_d x'_d)^p = (\sum_{d_1=1}^{D} x_{d_1} x'_{d_1}) ... (\sum_{d_p=1}^{D} x_{d_p} x'_{d_p}) \\ =& \sum_{d_1=1}^{D} ... \sum_{d_p=1}^{D} (x_{d_1} ... x_{d_p})(x'_{d_1} ... x'_{d_p}) \triangleq \phi(\mathbf{x}) \cdot \phi(\mathbf{x}'). \end{aligned}$$

Notice that this sum apparently contains D^p terms but in fact it is less than this as the order of the indices in the monomial $x_{d_1} ... x_{d_p}$ is unimportant, e.g. for $p = 2$, $x_1 x_2$ and $x_2 x_1$ are the same monomial. We can remove the redundancy by defining a vector $\mathbf{m}$ whose entry m_d specifies the number of times index d appears in the monomial, under the constraint that $\sum_{i=1}^{D} m_i = p$. Thus $\phi_{\mathbf{m}}(\mathbf{x})$, the feature corresponding to vector $\mathbf{m}$ is proportional to the monomial $x_1^{m_1} ... x_D^{m_D}$. The degeneracy of $\phi_{\mathbf{m}}(\mathbf{x})$ is $\frac{p!}{m_1! ... m_D!}$, giving the feature map

$$\phi_{\mathbf{m}}(\mathbf{x}) = \sqrt{\frac{p!}{m_1! ... m_D!}} x_1^{m_1} ... x_D^{m_D}.$$

For example, for $p = 2$ in $D = 2$, we have $\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2)^\top$. Additionally, we have intercept kernel when $p = 0$, and linear kernel when $p = 1$.

Neural Network Kernels

Consider a network which takes an input $\mathbf{x}$, has one hidden layer with N_H units and then linearly combines the outputs of the hidden units with a bias b to obtain $f(\mathbf{x})$. The mapping can be written

$$f(\mathbf{x}) = b + \sum_{j=1}^{N_H} v_j h(\mathbf{x}; \mathbf{u}_j),$$

where the v_j's are the hidden-to-output weights and $h(\mathbf{x}; \mathbf{u})$ is the hidden unit transfer function (which we shall assume is bounded) which depends on the input-to-hidden weights $\mathbf{u}$. Let b and the v's have independent zero-mean distributions of variance σ_b^2 and σ_v^2, respectively, and let the weights $\mathbf{u}_j$ for each hidden unit be independently and identically distributed. Denoting all weights by $\mathbf{w}$, we obtain

$$\begin{aligned}\mathrm{E}_{\mathbf{w}}[f(\mathbf{x})] =&0,\\ \mathrm{E}_{\mathbf{w}}[f(\mathbf{x})f(\mathbf{x}')] =&\sigma_b^2 + \sum_j \sigma_v^2 \mathrm{E}_{\mathbf{u}}[h(\mathbf{x};\mathbf{u}_j)h(\mathbf{x}';\mathbf{u}_j)] \qquad (2.1.10)\\ =&\sigma_b^2 + N_H \sigma_v^2 \mathrm{E}_{\mathbf{u}}[h(\mathbf{x};\mathbf{u})h(\mathbf{x}';\mathbf{u})], \qquad (2.1.11)\end{aligned}$$

where (2.1.11) follows because all of the hidden units are identically distributed. The sum in (2.1.10) is over N_H identically and independently distributed random variables. As the transfer function is bounded, all moments of the distribution will be bounded and hence the central limit theorem can be applied, showing that the stochastic process will converge to a Gaussian process in the limit as $N_H \to \infty$.

By evaluating $\mathrm{E}_{\mathbf{u}}[h(\mathbf{x};\mathbf{u})h(\mathbf{x}';\mathbf{u})]$ we can obtain the covariance function of the neutral network. For example if we choose the error function $h(z) = \mathrm{erf}(z) = 2/\sqrt{\pi}\int_0^z e^{-t^2}dt$ as the transfer function, let $h(\mathbf{x};\mathbf{u}) = \mathrm{erf}\Big(u_0 + \sum_{j=1}^D u_j x_j\Big)$ and choose $\mathbf{u} \sim \mathrm{N}(0, \Sigma)$ then we obtain

$$k_{NN}(\mathbf{x},\mathbf{x}') = \frac{2}{\pi}\sin^{-1}\Big(\frac{2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^\top \Sigma \tilde{\mathbf{x}}')}}\Big),$$

where $\tilde{\mathbf{x}} = (1, x_1, ..., x_d)^\top$ is an augmented input vector.

2.2. Hypothesis Test

Asymptotic Test

We use the classical variance component test (Lin 1997) to construct a testing procedure for the hypothesis about Gaussian process function:

$$H_0 : h \in \mathcal{H}_0. \qquad (2.2.1)$$

We first translate above hypothesis into a hypothesis in terms of model parameters. The key of our approach is to assume that h lies in a RKHS generated by a *garrote kernel function* $k_\delta(\mathbf{z},\mathbf{z}')$ (Maity and Lin 2011), which is constructed by including an extra *garrote parameter* δ to a given kernel function. When $\delta = 0$, the garrote kernel function $k_0(\mathbf{x},\mathbf{x}') = k_\delta(\mathbf{x},\mathbf{x}') \mid_{\delta=0}$ generates exactly $\mathcal{H}_0$, the space of functions under the null hypothesis. In order to adapt this general hypothesis to their hypothesis of interest, practitioners need only to specify the form of the garrote kernel so that $\mathcal{H}_0$ corresponds to the null hypothesis. For example, if $k_\delta(\mathbf{x}) = k(\delta * x_1, x_2, ..., x_p), \delta = 0$ corresponds to the null hypothesis $H_0 : h(\mathbf{x}) = h(x_2, ..., x_p)$, i.e. the function $h(\mathbf{x})$ does not depend on x_1. As a result, the general hypothesis is equivalent to:

$$H_0 : \delta = 0. \qquad (2.2.2)$$

We now construct a test statistic $\hat{T}_0$ for (2.2.2) by noticing that the garrote parameter δ can be treated as a variance component parameter in the linear mixed model. This is because the Gaussian process under garrote kernel can be formulated into below LMM:

$$\mathbf{y} = \boldsymbol{\mu} + \mathbf{h} + \boldsymbol{\epsilon}, \quad \text{where} \quad \mathbf{h} \sim \mathrm{N}(\mathbf{0}, \tau\mathbf{K}_\delta) \quad \boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \sigma^2\mathbf{I}), \qquad (2.2.3)$$

where $\mathbf{K}_\delta$ is the kernel matrix generated by $k_\delta(\mathbf{z}, \mathbf{z}')$. Consequently, we can derive a variance component test for H_0 by calculating the square derivative of L_{REML} with respect to δ under H_0 (Lin 1997):

$$\hat{T}_0 = \hat{\tau} * (\mathbf{y} - \hat{\boldsymbol{\mu}})^\top \mathbf{V}_0^{-1}[\partial \mathbf{K}_0]\mathbf{V}_0^{-1}(\mathbf{y} - \hat{\boldsymbol{\mu}}), \tag{2.2.4}$$

where $\mathbf{V}_0 = \hat{\sigma}^2\mathbf{I} + \hat{\tau}\mathbf{K}_0$. In this expression, $\mathbf{K}_0 = \mathbf{K}_\delta \mid_{\delta=0}$, and $\partial\mathbf{K}_0$ is the null derivative kernel matrix whose $(i,j)^{th}$ entry is $\frac{\partial}{\partial\delta}k_\delta(\mathbf{x}, \mathbf{x}') \mid_{\delta=0}$.

The null distribution of $\hat{T}$ can be approximated using a scaled chi-square distribution $\kappa\chi^2_\nu$ using Satterthwaite method by matching the first two moments of T:

$$\kappa * \nu = \mathrm{E}(T) = \hat{\tau} * \mathrm{tr}(\mathbf{V}_0 \partial\mathbf{K}_0) \quad 2 * \kappa^2 * \nu = Var(T) = \hat{\mathbf{I}}_{\delta\delta},$$

with solution:

$$\hat{\kappa} = \hat{\mathbf{I}}_{\delta\delta}/[\hat{\tau} * \mathrm{tr}(\mathbf{V}_0^{-1}\partial\mathbf{K}_0)] \quad \hat{\nu} = [\hat{\tau} * \mathrm{tr}(\mathbf{V}_0^{-1}\partial\mathbf{K}_0)]^2/(2 * \hat{\mathbf{I}}_{\delta\theta}),$$

where $\hat{\mathbf{I}}_{\delta\delta} = \mathbf{I}_{n,\delta\delta} - \mathbf{I}_{\delta\theta}^\top \mathbf{I}_{\theta\theta}^{-1}\mathbf{I}_{\delta\theta}$ is the efficient information of δ under REML. $\mathbf{I}_{\delta\delta}$, $\mathbf{I}_{\theta\theta}$ and $\mathbf{I}_{\delta\theta}$ are submatrices of the REML information matrix. Numerically more accurate, but computationally less efficient approximation methods are also available.

Finally, the p-value of this test is calculated by examining the tail probability of $\hat{\kappa}\chi^2_{\hat{\nu}}$:

$$p = P(\hat{\kappa}\chi^2_{\hat{\nu}} > \hat{T}) = P(\chi^2_{\hat{\nu}} > \hat{T}/\hat{\kappa}).$$

A complete summary of the proposed testing procedure is available in Algorithm 2.

Bootstrap Test

In practice, the sample size of the collected data is always small. To make valid inferences about a population from the sample, we need to perform resampling. Commonly used method in small sample size is bootstrap, which can give valid tests with moderate sample sizes.

Testing in a regression model framework requires computing the distribution of the test statistic under sampling from the null-hypothesis model. A good approximation to the distribution of the test statistic under sampling from the true null-hypothesis model is the distribution of the test statistic under sampling from the fitted null-hypothesis model. For instance, when testing (2.2.2), we first fit the model under the null,

$$\mathrm{E}(\mathbf{y}^\star) = \mathbf{K}_0(\mathbf{K}_0 + \lambda\mathbf{I})^{-1}\mathbf{y} = \mathbf{A}_0\mathbf{y}. \tag{2.2.5}$$

and generate $\mathbf{Y}^\star$ for each individuals with a random noise, whose variance is also estimated. We then compute the test statistic for this simulated sample, and repeat this process B times. The empirical distribution these provide is an estimate of the test statistic's distribution under the null. Correspondingly, p-values are calculated as the proportion of simulated test statistics that are most extreme than the observed value.

If the distribution of the test statistic depends smoothly on the regression parameter values, which is true in all standard examples, this 'parametric bootstrap' approach gives an asymptotically valid test (Davison & Hinkley 1997, 4.2.3). Like the classical bootstrap, it samples from a distribution based on the observed data, but the simulations are from a fitted parametric model rather than the empirical distribution. To obtain a valid test, the fitted parametric model is chosen so that the null hypothesis is satisfied. A complete summary of the proposed testing procedure is available in Algorithm 3.

3. Illustrations

Using a library of base kernels, **CVEK** learns a proper generating function from data by directly minimizing the ensemble model's error, and tests whether the data is generated by the RKHS under the null hypothesis.

CVEK is composed of three parts: model definition, estimation and testing. Given data and the setting of two groups of variables, model definition returns the responses and two matrices indicating two groups of variables we need to test, and generates the expected kernel library. Estimation conducts gaussian process regression based on the estimated ensemble kernel matrix. And testing executes the hypothesis test and returns p-value representing whether there is interaction effect between these two groups of variables.

3.1. Model Definition

`define_model` function has four returns with four parameters.

```
R> define_model

 function (formula, label_names, data, kern_par)
 {
     Y <- data[, as.character(attr(terms(formula), "variables"))[2]]
     re <- generate_formula(formula, label_names)
     generic_formula0 <- re$generic_formula
     len <- re$length_main
     X <- model.matrix(generic_formula0, data)[, -1]
     n <- nrow(X)
     Xm <- colMeans(X)
     p <- ncol(X)
     X <- X - rep(Xm, rep(n, p))
     Xscale <- drop(rep(1/n, n) %*% X^2)^0.5
     X <- X/rep(Xscale, rep(n, p))
     X1 <- X[, c(1:length(label_names[[1]]))]
     X2 <- X[, c((length(label_names[[1]]) + 1):len)]
     kern_list <- list()
     for (d in 1:nrow(kern_par)) {
         kern_list[[d]] <- generate_kernel(kern_par[d, ]$method,
             kern_par[d, ]$Sigma, kern_par[d, ]$l, kern_par[d,
                 ]$p)
     }
     list(Y = Y, X1 = X1, X2 = X2, kern_list = kern_list)
 }
```

Note that:

- All four parameters are mandatory for management.
- Users can generate `kern_par` with the function `generate_kernel`.
- If users want to test data with given and known interaction strength, they can generate data with the function `generate_data`.

For example, we have a dataset whose `n <-100, label_names <- list(X1 = c("x1", "x2"), X2 = c("x3", "x4")), int_effect <- 0.3, method <- "rbf", l <- 1, eps <- 0.01`:

```
R> label_names <- list(X1 = c("x1", "x2"), X2 = c("x3", "x4"))
R> data <- generate_data(n = 100, label_names, method = "rbf",
+    int_effect = .3, l = 1, eps = .01)
```

We want our kernel library contains three kernels: `method <- "rbf", l <- 0.5`, `method <- "polynomial", p <- 2` and `method <- "matern", l <- 1.5, p <- 3`:

```
R> kern_par <- data.frame(method = c("rbf", "polynomial", "matern"),
+    Sigma = rep(0, 3), l = c(.5, 1, 1.5), p = 1:3)
R> kern_par$method <- as.character(kern_par$method)
```

and the null model is `Y ~ X1 + X2`:

```
R> formula <- Y ~ X1 + X2
```

With all these parameters specified, we can define our model:

```
R> fit <- define_model(formula, label_names, data, kern_par)
```

3.2. Estimation

After defining the model, we can apply `estimation` function to conduct gaussian process regression based on the estimated ensemble kernel matrix.

`estimation` function has five returns with eight parameters.

```
R> estimation

function(
  Y, X1, X2, kern_list, mode = "loocv", strategy = "erm",
  beta = 1, lambda_list = exp(seq(-10, 5, .5))) {
  n <- length(Y)
  kern_size <- length(kern_list)
  base_est <- estimate_base(n, kern_size, Y, X1, X2, kern_list,
                            mode, lambda_list)
  P_K_hat <- base_est$P_K_hat
  error_mat <- base_est$error_mat
  ens_res <- ensemble(n, kern_size, strategy, beta, error_mat, P_K_hat)
  K_ens <- ensemble_kernel_matrix(ens_res$A_est)
  lambda_ens <- tuning(Y, K_ens, mode, lambda_list)
  ens_est <- estimate_ridge(X = matrix(1, nrow = n, ncol = 1),
                            K = K_ens, Y = Y, lambda = lambda_ens)
  list(lambda = lambda_ens,
       beta = ens_est$beta,
       alpha = ens_est$alpha, K = K_ens,
       u_hat = ens_res$u_hat, base_est = base_est)
}
```

Note that:

- The first four parameters are mandatory for management.
- For the last four parameters, users can substitute alternatives for the default ones.

Continuing with our example, we want to use `erm` to ensemble our base kernels and `loocv` to select tuning parameter whose range is `lambda_list <- exp(seq(-5, 5))`:

```
R> mode <- "loocv"
R> strategy <- "erm"
R> lambda_list <- exp(seq(-5, 5))
```

Then we can find the solution with the help of `estimation`:

```
R> sol <- estimation(fit$Y, fit$X1, fit$X2, fit$kern_list, mode, strategy, lambda_list)
```

3.3. Testing

Finally, here comes the testing procedure.
`testing` function has one return with several parameters.

```
R> testing

function(formula_int, label_names, Y, X1, X2, kern_list,
                    mode = "loocv", strategy = "erm", beta = 1,
                    test = "boot", lambda_list = exp(seq(5, 5)),
                    B = 100) {
  re <- generate_formula(formula_int, label_names)
  generic_formula0 <- re$generic_formula
  len <- re$length_main
  data <- as.data.frame(cbind(Y, X1, X2))
  colnames(data) <- c("Y", label_names[[1]], label_names[[2]])
  X <- model.matrix(generic_formula0, data)[, -1]
  X12 <- X[, c((len + 1):dim(X)[2])]
  n <- length(Y)
  result <- estimation(Y, X1, X2, kern_list, mode, strategy, beta, lambda_list)
  lambda <- result$lambda
  beta0 <- result$beta[1, 1]
  alpha0 <- result$alpha
  K_gpr <- result$K
  u_weight <- result$u_hat
  sigma2_hat <- estimate_noise(Y, lambda, beta0, alpha0, K_gpr)
  tau_hat <- sigma2_hat / lambda
  test <- match.arg(test, c("asym", "boot"))
  func_name <- paste0("test_", test)
  pvalue <- do.call(func_name, list(n = n, Y = Y, X12 = X12,
                                    beta0 = beta0, alpha0 = alpha0,
                                    K_gpr = K_gpr, sigma2_hat = sigma2_hat,
                                    tau_hat = tau_hat, B = B))
  list(pvalue = pvalue, u_weight = u_weight)
}
```

Note that `formula_int` is the alternative model with interaction.

Now, we want to conduct score test with `test <- "boot", B <- 100` since the sample size is small (`n <- 100`).

```
R> formula_int <- Y ~ X1 * X2
R> test <- "boot"
R> B <- 100
R> pvalue <- testing(formula_int, label_names, fit$Y, fit$X1, fit$X2, fit$kern_list,
+    mode, strategy, beta = 1, test, lambda, B)
R> pvalue

[1] 0
```

4. Summary and discussion

We evaluated the finite-sample performance of the proposed interaction test in a simulation study that is analogous to a real nutrition-environment interaction study. We generate two groups of input features $(\mathbf{x}_{i,1}, \mathbf{x}_{i,2}) \in \mathbb{R}^{p_1} \times \mathbb{R}^{p_2}$ independently from standard Gaussian distribution, representing normalized data representing subject's level of exposure to p_1 environmental pollutants and the levels of a subject's intake of p_2 nutrients during the study. Throughout the simulation scenarios, we keep $n = 100$, and $p_1 = p_2 = 2$. We generate the outcome y_i as:

$$y_i = h_1(\mathbf{x}_{i,1}) + h_2(\mathbf{x}_{i,2}) + \delta * h_1(\mathbf{x}_{i,1}) * h_2(\mathbf{x}_{i,2}) + \epsilon_i,$$

where h_1, h_2 are sampled from RKHSs $\mathcal{H}_1$, $\mathcal{H}_2$, generated using a ground-truth kernel k_{true}. We standardize all sampled functions to have unit form, so that δ represents the strength of interaction relative to the main effect.

For each simulation scenario, we first generated data using δ and k_{true} as above, then selected a k_{model} to estimate the null model and obtain p-value using Algorithm 2 and 3 repsectively.

We repeated each scenario 200 times, and evaluate the test performance using the empirical probability $\hat{P}(p \leq 0.05)$ estimates the test's Type I error, and should be smaller or equal to the significance level 0.05. Under alternative hypothesis $H_a : \delta > 0$, $\hat{P}(p \leq 0.05)$ estimates the test's power, and should ideally approach 1 quickly as the strength of interaction δ increases.

In this study, we varied k_{true} to produce data-generating functions $h_\delta(\mathbf{x}_{i,1}, \mathbf{x}_{i,2})$ with different smoothness and complexity properties, and varied k_{model} to reflect different common modeling strategies for the null model in addition to using CVEK. We then evaluated how these two aspects impact the hypothesis test's Type I error and power.

In terms of data-generating mechanism, we consider 12 combinations: 3 polynomial kernels ($p = 1, 2, 3$) representing finite-dimensional, parametric functions of different degree of nonlinearity. 3 Gaussian RBF kernels ($l = 0.5, 1, 1.5$) representing smooth functions of different frequency, and also 6 Matern kernels, with $l \in \{0.5, 1, 1.5\}$ and $\nu \in \{\frac{3}{2}, \frac{5}{2}\}$, representing functions with different frequency and differentiability. For modeling strategies, we consider 4 types of kernel libraries: (1) 3 polynomial kernels with degree ($p = 1, 2, 3$, Figure 2 & 7); (2) 3 RBF kernels with wavelength ($l = 0.6, 1, 2$, Figure 3 & 8); (3) 3 polynomial kernels ($p = 1, 2, 3$) and 3 RBF kernels ($l = 0.6, 1, 2$, Figure 4 & 9), and (4) 3 $l = 1$ Matern kernels ($\nu = 1/2, 3/2, 5/2$) and 3 RBF kernels ($l = 0.6, 1, 2$, Figure 5 & 10). For each combination of kernel library and data-generating mechanism, we estimate the null model using four methods for tuning-parameter selection (LooCV, AICc, GCVc, GMPML) and the three methods for ensemble strategy (ERM, simple averaging, exponential weighting), resulting in 12 null model estimates for each of the $12 * 4 * 2 = 96$ combinations.

Specifically, for exponential weighting, we also consider four different values of β: a fixed value 1, $\min\{RSS\}_{d=1}^{D}/10$, $\text{median}\{RSS\}_{d=1}^{D}$ and $\max\{RSS\}_{d=1}^{D} * 2$. Here $\{RSS\}_{d=1}^{D}$ are the set of residual sum of squares of D base kernels. Figure 1 to 10 are showing $\beta = \min\{RSS\}_{d=1}^{D}/10$ in exponential weighting.

The simulation results are presented graphically from Figure 1 to 10 (the first five are asymptotic tests and the last five bootstrap tests). They show the estimated $\hat{P}(p < 0.05)$ (y-axis) as a function of Interaction Strength $\delta \in \{0, 0.1, 0.2, 0.3\}$ (x-axis). For each figure, the combination of top margin and right margin indicate the data-generating mechanism: Polynomial, RBF, Matern $\nu = 3/2$, and Matern $\nu = 5/2$; $l = 0.5, 1, 1.5$ (represent $p = 1, 2, 3$ for Polynomial). And the lines refer to the different combination of tuning parameter selection (colors) and ensemble strategy (line types).

Generally, bootstrap is capable of achieving satisfying performance (i.e. correct Type I error and decent power compared to the true kernel) when tuning parameter and ensemble method are chosen carefully, although the power of asymptotic test is greater than our bootstrap test. Specifically, the exact performance of the test depends on the choice of tuning parameter selection, ensemble strategy, and the relation between model and data.

For asymptotic test, we observed clear differences in test performance between different tuning-parameter criteria and different ensemble strategies. For tuning-parameter selection, LooCV and GMPML generally guarantee correct Type I error except when the base kernels are strictly simpler than the truths (Figure 2), but GMPML potentially leads to low power under the alternative(Figure 3/ 4). Moreover, AICc can guarantee correct Type I error only when the kernels in the library are smoother than the data-generating kernel (Figure

3, column 2-4). For ensemble strategies, ERM leads to better power (when Type I error is guaranteed) if base kernels are flexible and infinite-dimensional (Figure 3/ 5). In the more complex scenario when the library consists of a mixture of finite- and infinite-dimensional kernels (Figure 4), simple averaging performs better only if data-generating kernels are of low frequency and infinite-dimension (row 3, column 3-4), otherwise ERM performs better (row 2). EXP is quite similar to but not as good as ERM since its power is lower than ERM under the alternative.

In terms of bootstrap test, we also observed differences based on different tuning-parameter criteria and different ensemble strategies. For tuning-parameter selection, LooCV generally guarantees correct Type I error, but potentially leads to low power under the alternative (Figure 10). On the other hand, AICc is more powerful if kernels in the library are as or more complex (in terms of smoothness) than the true kernel (Figure 8, column 2; Figure 10, column 2, row 3 & column 3-4). Moreover, GMPML is more powerful if kernels in the library are smoother than the data-generating kernel (Figure 7; Figure 8/ 9, column 3-4). For ensemble strategies, simple averaging provides better Type I error and power if base kernels are simple and finite-dimensional (Figure 7). However, ERM leads to better power (when Type I error is guaranteed) if base kernels are flexible and infinite-dimensional (Figure 8/ 10). In the more complex scenario when the library consists of a mixture of finite- and infinite-dimensional kernels (Figure 9), simple averaging performs better if data-generating kernels are of low frequency (row 3), otherwise ERM performs better (row 2). As for EXP (discussed in Appendix in detail), it has fairly greater power under the alternative while guarantees correct Type I error under the null at the same time (Figure 8, 10, column 2-4s; Figure 9). Otherwise when the data-generating kernel is strictly simpler than kernels in the library, EXP potentially cannot guarantee correct Type I error (Figure 8, 10, column 1s). And GMPML is not a good partner of it (Figure 8-9, column 2-3). However, the case is different for GMPML as GMPML-ERM strategy is always (and sometimes substantially) more powerful than GMPML-AVG. And in this case, GMPML is not a good partner for EXP as it cannot guarantee correct Type I error.

Additionally, for our bootstrap test, there are two interesting observations that we'd like to point out: if the kernel library contains only very flexible kernels, model estimates will overfit the data and produce test with very low power, even if the kernels in the library are exactly the data-generation kernels (Figure 6, row 1 & column 2-4). On the other hand, if the kernel library contains kernels that are strictly simpler than the truths, even the proposed LooCV-ERM will inevitably produce slight-to-moderate level of Type I error inflation (e.g. using polynomial kernels to fit RBF data (Figure 7, column 2), or using too few RBF kernels to fit cubic polynomial (Figure 8/ 10, row 3 & column 1). In the latter case, the range of RBF length-scale parameters is not sufficient to cover the spectral density of the cubic function). Finally, in some cases, we observed powers decrease (LooCV in Figure 6/ 7, row 3 & column 1; LooCV with ERM in Figure 8/ 9, row 3 & column 1). We will investigate the mechanism behind such phenomenon in future work.

Moreover, Figure 2 & 7 (column 2-4) see the apparent advantages of our proposed bootstrap-LooCV: when the base kernels are strictly simpler than the truths, only bootstrap-LooCV can guarantee correct Type I error. Also, under the circumstance that base kernels are strictly more flexible than the truths (Figure 5/ 10, column 1), ERM performs better in asymptotic test while simple averaging better in bootstrap test.

Algorithm 1 Cross Validated Ensemble of Kernels (CVEK)

```
procedure CVEK
Input: A library of kernels $\{k_d\}_{d=1}^D$, Data $(\mathbf{y}, \mathbf{x})$
Output: Ensemble Kernel Matrix $\widehat{\mathbf{K}}$
# Stage 1: Estimate $\lambda$ and CV error for each kernel
    for $d = 1$ to $D$ do
        $\mathbf{K}_d = \mathbf{K}_d / tr(\mathbf{K}_d)$
        $\widehat{\lambda}_d = argmin$ LOOCV$\Big(\lambda \mid \mathbf{K}_d\Big)$
        $\widehat{\epsilon}_d =$ CV$\Big(\widehat{\lambda}_d \mid \mathbf{K}_d\Big)$
    end for
# Stage 2: Estimate ensemble weights $\mathbf{u}_{D\times 1} = \{u_1, \dots, u_D\}$
    $\hat{\mathbf{u}} = \underset{\mathbf{u}\in\Delta}{argmin} \parallel \sum_{d=1}^D u_d \widehat{\epsilon}_d \parallel^2$    where $\Delta = \{\mathbf{u} \mid \mathbf{u} \geq 0, \parallel \mathbf{u} \parallel_1 = 1\}$
# Stage 3: Assemble the ensemble kernel matrix $\widehat{\mathbf{K}}_{ens}$
    $\widehat{\mathbf{A}} = \sum_{d=1}^D \widehat{\mu}_d \mathbf{A}_{\widehat{\lambda}_d, k_d}$
    $\mathbf{U}_A, \boldsymbol{\delta}_A =$ spectral_decomp$(\widehat{\mathbf{A}})$
    $\lambda_{\mathbf{K}} = min\Big(1, (\sum_{k=1}^n \frac{\delta_{A,k}}{1-\delta_{A,k}})^{-1}, min(\{\widehat{\lambda}_d\}_{d=1}^D)\Big)$
    $\widehat{\mathbf{K}} = \lambda_{\mathbf{K}} * \widehat{\mathbf{U}}_A \; diag\Big(\frac{\delta_{A,k}}{1-\delta_{A,k}}\Big) \; \widehat{\mathbf{U}}_A^\top$
end procedure
```

Algorithm 2 Variance Component Test for $h \in \mathcal{H}_0$

```
procedure VCT FOR INTERACTION
Input: Null Kernel Matrix $\mathbf{K}_0$, Derivative Kernel Matrix $\partial\mathbf{K}_0$, Data $(y, \mathbf{x})$
Output: Hypothesis Test p-value $p$
# Stage 1: Estimate Null Model using REML
    $(\hat{\boldsymbol{\mu}}, \hat{\tau}, \hat{\sigma}^2) = argmax L_{REML}(\boldsymbol{\mu}, \tau, \sigma^2 \mid \mathbf{K}_0)$
# Stage 2: Compute Test Statistic and Null Distribution Parameters
    $\hat{T}_0 = \hat{\tau} * (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top \mathbf{V}_0^{-1} \partial\mathbf{K}_0 \mathbf{V}_0^{-1} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})$
    $\hat{\kappa} = \hat{\mathbf{I}}_{\delta\delta} / [\hat{\tau} * tr(\mathbf{V}_0^{-1} \partial\mathbf{K}_0)]$,    $\hat{\nu} = [\hat{\tau} * tr(\mathbf{V}_0^{-1} \partial\mathbf{K}_0)]^2 / (2 * \hat{\mathbf{I}}_{\delta\theta})$
# Stage 3: Compute p-value and reach conclusion
    $p = P(\hat{\kappa}\chi^2_{\hat{\nu}} > \hat{T}) = P(\chi^2_{\hat{\nu}} > \hat{T}/\hat{\kappa})$
end procedure
```

Algorithm 3 Parametric Bootstrap Test

```
procedure PARAMETRIC BOOTSTRAP TEST
Input: Null Kernel Matrix $\mathbf{K}_0$, Derivative Kernel Matrix $\partial\mathbf{K}_0$, Data $(\mathbf{y}, \mathbf{x})$
Output: Hypothesis Test p-value $p$
# Stage 1:  Estimate Null Model using Gaussian Process Regression
    $\hat{\boldsymbol{\mu}} = \mathbf{A}_0\mathbf{y}, \quad \hat{\sigma}^2 = \frac{\mathbf{y}^\top(\mathbf{I}-\mathbf{A}_0)\mathbf{y}}{n-tr(\mathbf{A}_0)}, \quad \hat{\tau}$
# Stage 2:  Sample response from the fitted model obtain in Step 1
# and compute the test statistic based on fitting the alternative
# model, repeat for $B$ times
    for $b = 1$ to $B$ do
        $\mathbf{y}^\star = \hat{\boldsymbol{\mu}} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathrm{N}(0, \hat{\sigma}^2)$
        $\hat{T}_{0b} = \hat{\tau} * (\mathbf{y}^\star - \hat{\boldsymbol{\mu}})^\top \mathbf{V}_0^{-1} \partial\mathbf{K}_0 \mathbf{V}_0^{-1} (\mathbf{y}^\star - \hat{\boldsymbol{\mu}})$
    end for
# Stage 3:  Compute the test statistic for the original data, based
# on fitting the alternative hypothesis model
    $\hat{T}_0 = \hat{\tau} * (\mathbf{y} - \hat{\boldsymbol{\mu}})^\top \mathbf{V}_0^{-1} \partial\mathbf{K}_0 \mathbf{V}_0^{-1} (\mathbf{y} - \hat{\boldsymbol{\mu}})$
# Stage 4:  Compute p-value and reach conclusion
    $p = \frac{1}{B} \sum_{b=1}^{B} I(\hat{T}_{0b} > \hat{T}_0)$
end procedure
```

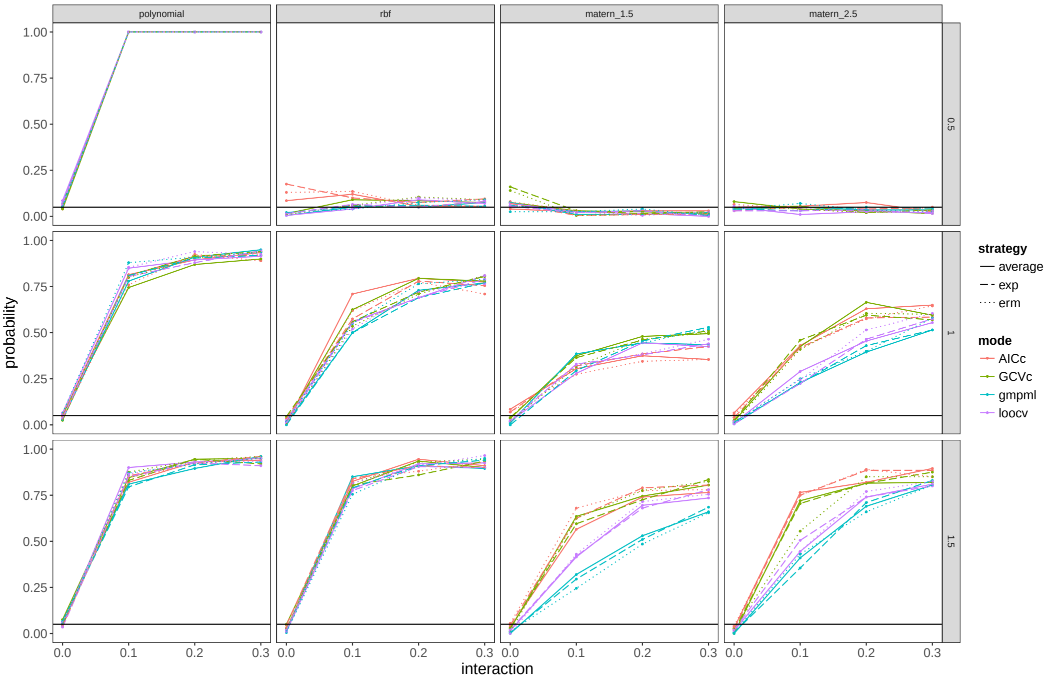

Figure 1: Asym, True kernel only

Acknowledgments

The authors are grateful to the Department of Biostatistics at the Harvard Chan School, with special thanks to Dr. Erin Lake for offering insightful instructions and helpful resources.

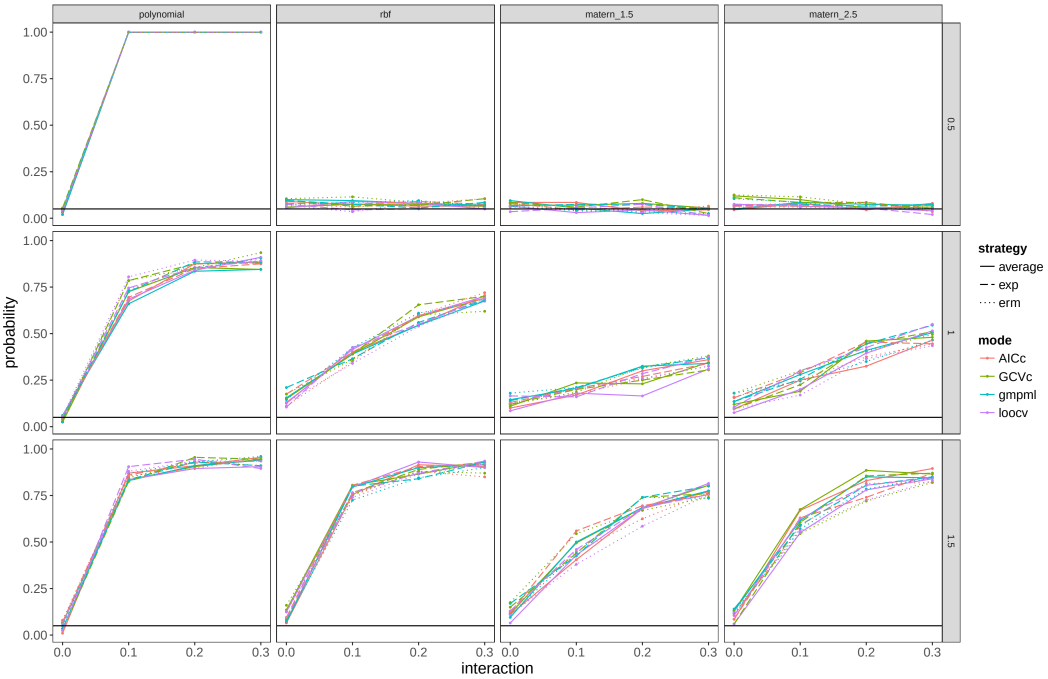

Figure 2: Asym, 3 Polynomial kernels

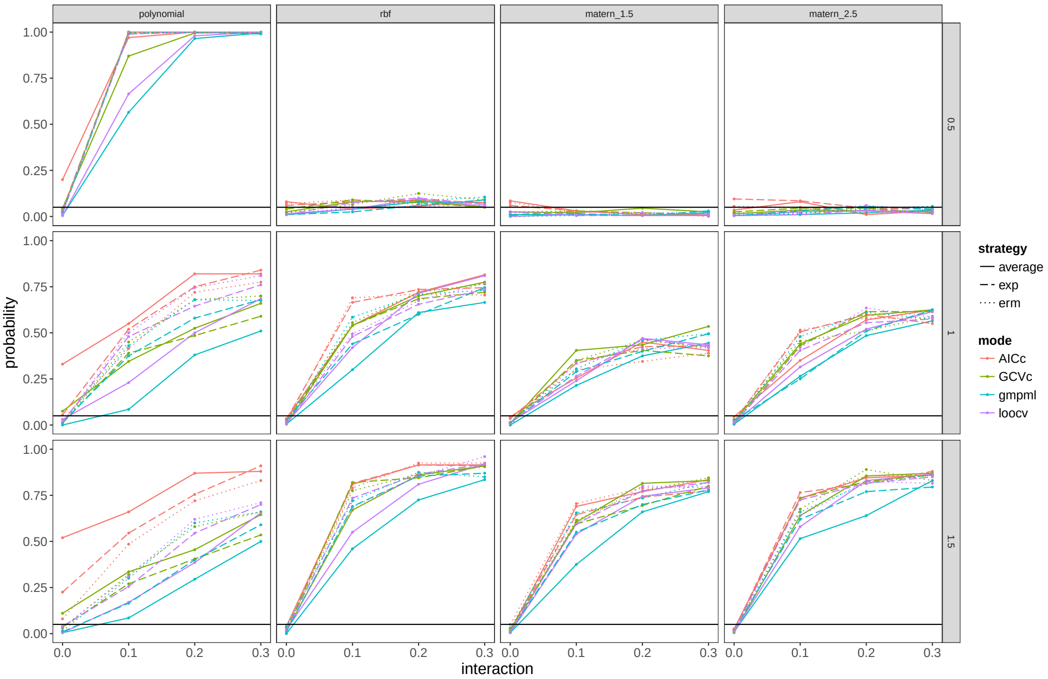

Figure 3: Asym, 3 RBF kernels

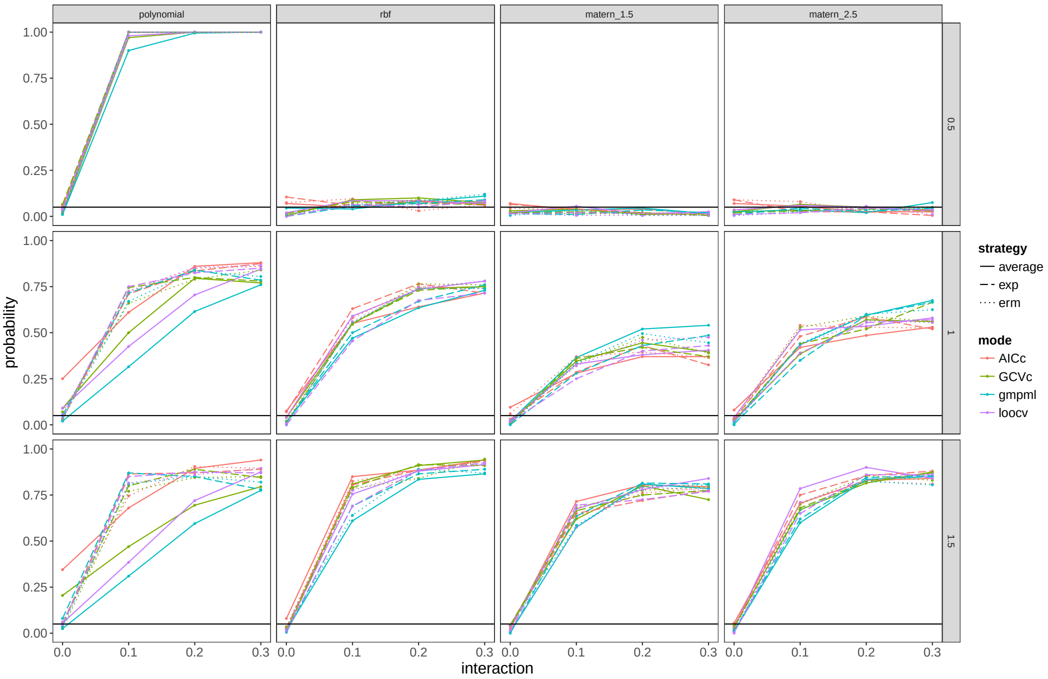

Figure 4: Asym, 3 Polynomial kernels and 3 RBF kernels

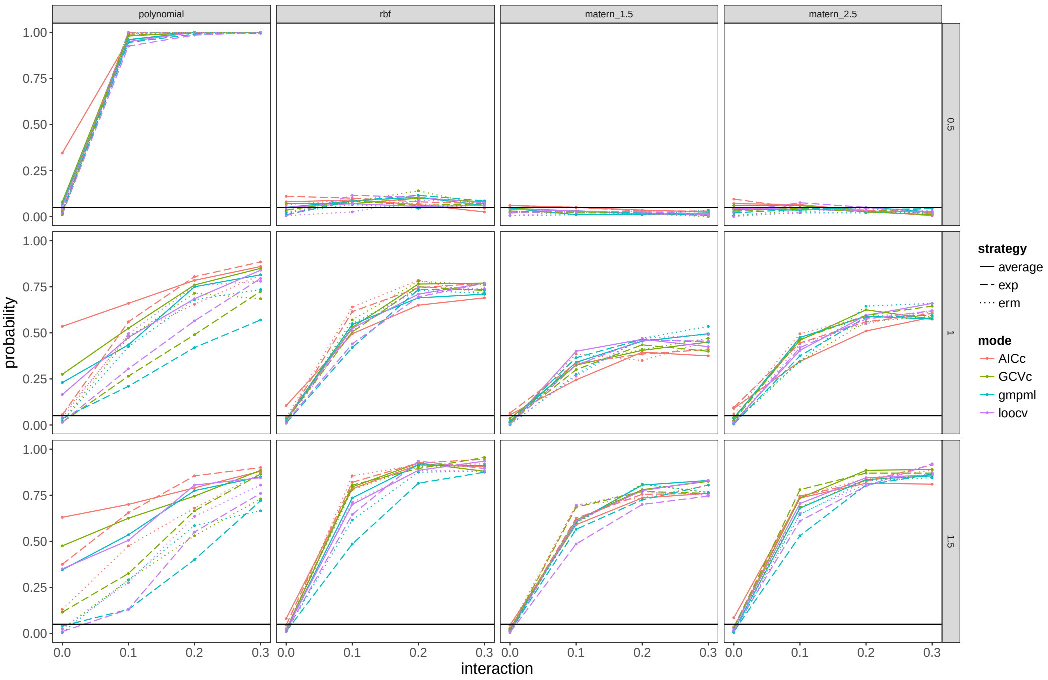

Figure 5: Asym, 3 Matern kernels and 3 RBF kernels

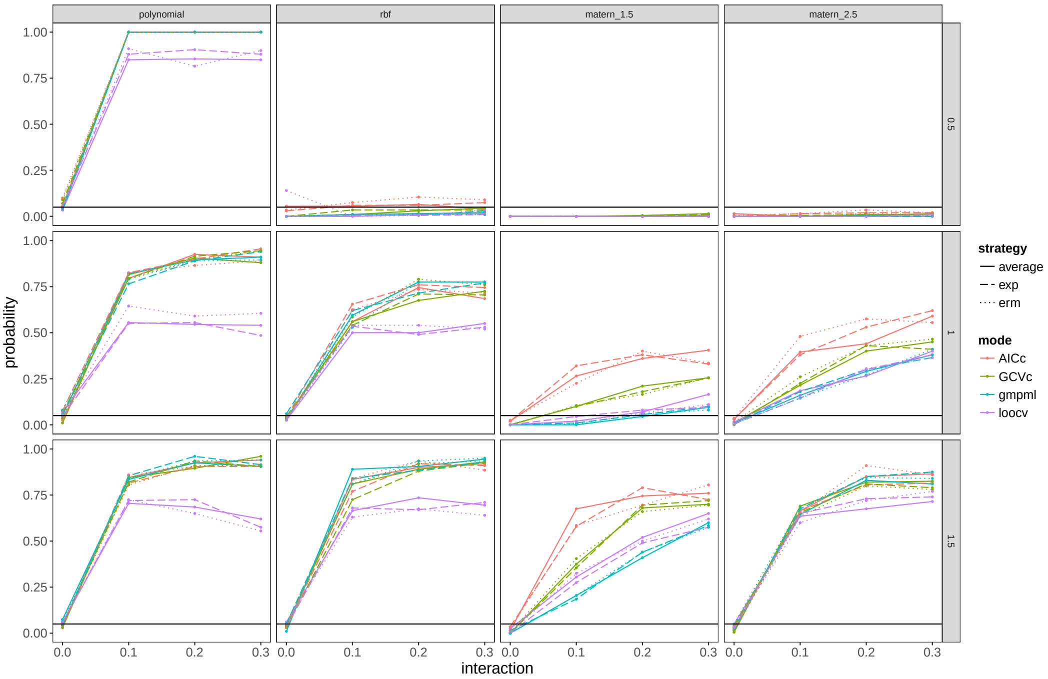

Figure 6: Boot, True kernel only

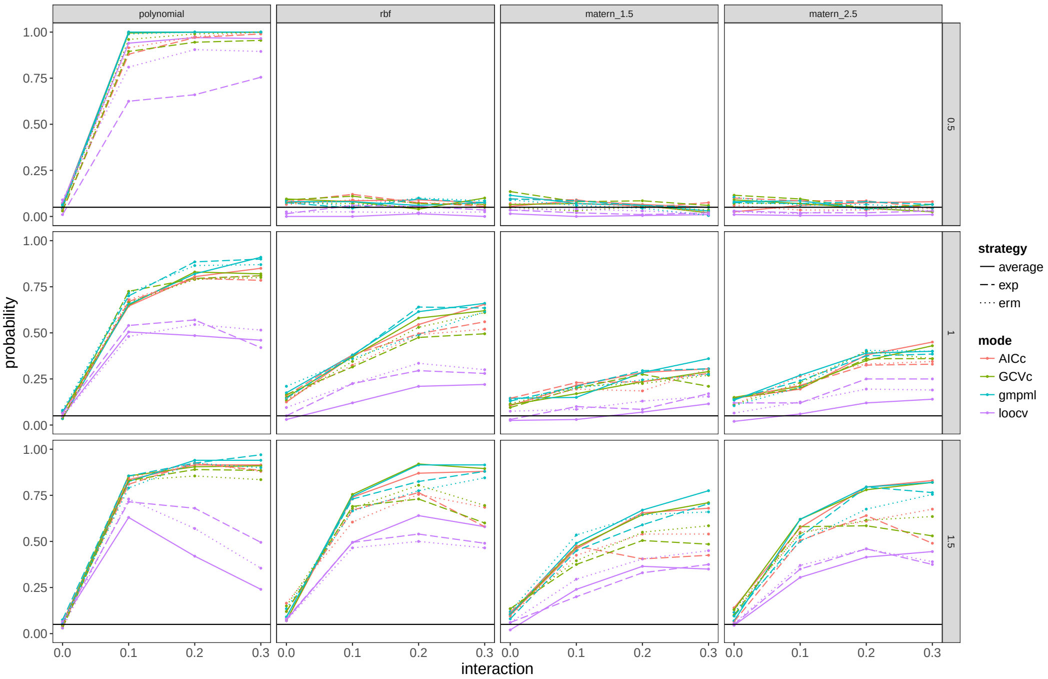

Figure 7: Boot, 3 Polynomial kernels

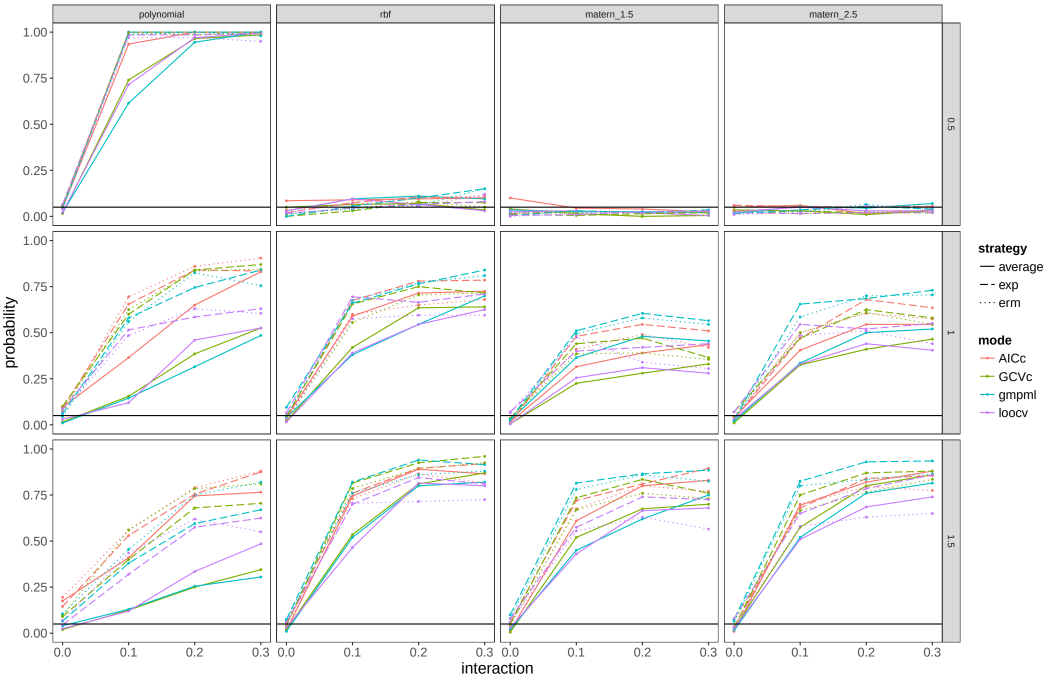

Figure 8: Boot, 3 RBF kernels

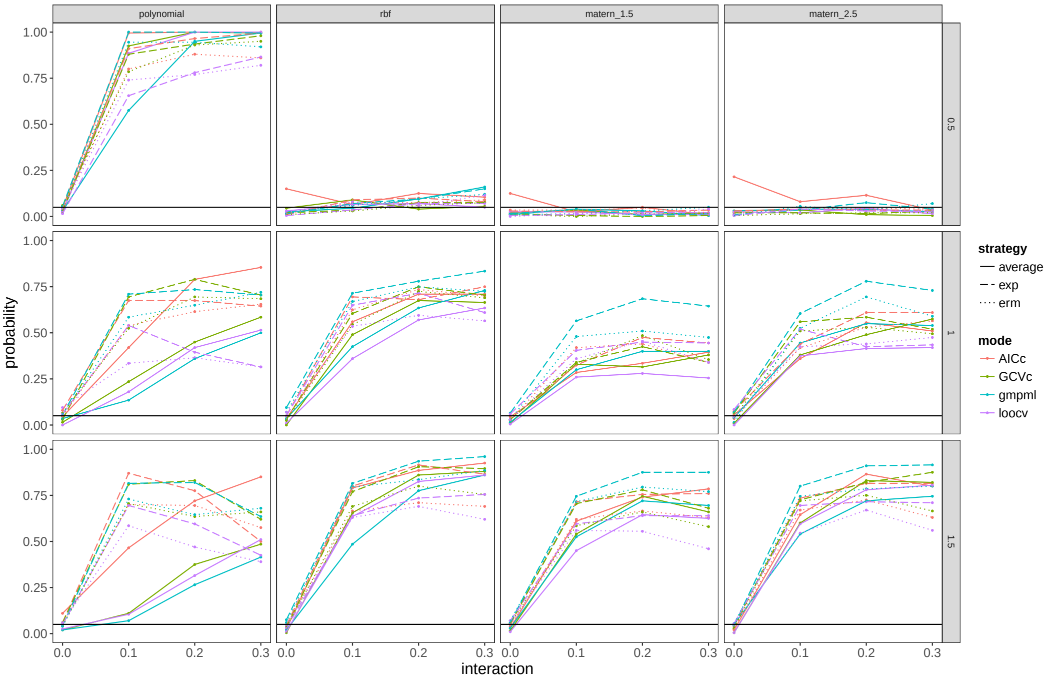

Figure 9: Boot, 3 Polynomial kernels and 3 RBF kernels

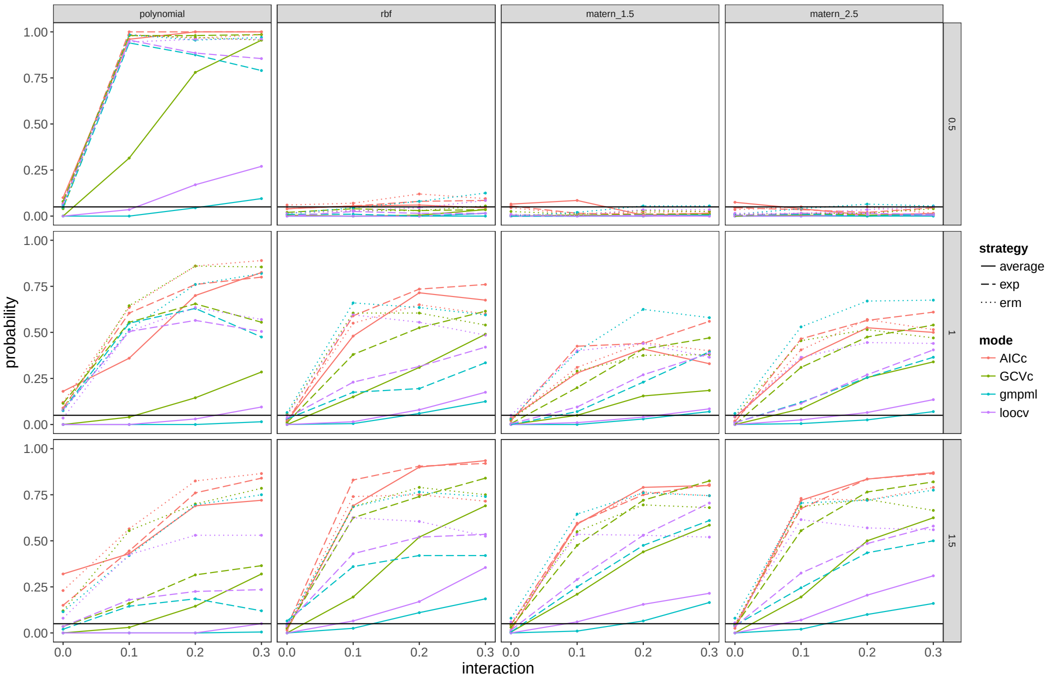

Figure 10: Boot, 3 Matern kernels and 3 RBF kernels

References

Abramowitz M (1974). *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables,*. Dover Publications, Inc., New York, NY, USA. ISBN 978-0-486-61272-0.

Akaike H (1998). "Information Theory and an Extension of the Maximum Likelihood Principle." In *Selected Papers of Hirotugu Akaike*, Springer Series in Statistics, pp. 199–213. Springer, New York, NY. ISBN 978-1-4612-7248-9 978-1-4612-1694-0.

Boonstra PS, Mukherjee B, Taylor JMG (2015). "A Small-Sample Choice of the Tuning Parameter in Ridge Regression." *Statistica Sinica*, **25**(3), 1185–1206. ISSN 1017-0405. doi:10.5705/ss.2013.284.

Bůžková P, Lumley T, Rice K (2011). "Permutation and parametric bootstrap tests for gene-gene and gene-environment interactions." *Annals of Human Genetics*, **75**(1), 36–45. ISSN 1469-1809. doi:10.1111/j.1469-1809.2010.00572.x.

Craven P, Wahba G (1979). "Smoothing noisy data with spline functions | SpringerLink." URL https://link.springer.com/article/10.1007/BF01404567.

Dalalyan AS, Tsybakov AB (2007). "Aggregation by Exponential Weighting and Sharp Oracle Inequalities." In *Learning Theory*, Lecture Notes in Computer Science, pp. 97–111. Springer, Berlin, Heidelberg. ISBN 978-3-540-72925-9 978-3-540-72927-3. doi:10.1007/978-3-540-72927-3_9. URL https://link.springer.com/chapter/10.1007/978-3-540-72927-3_9.

Golub GH, Heath M, Wahba G (1979). "Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter." *Technometrics*, **21**(2), 215–223. ISSN 0040-1706. doi:10.1080/00401706.1979.10489751. URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1979.10489751.

Harville DA (1977). "Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems." *Journal of the American Statistical Association*, **72**(358), 320–338. ISSN 0162-1459. doi:10.2307/2286796. URL http://www.jstor.org/stable/2286796.

Hastie T, Tibshirani R, Friedman J (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics, 2 edition. Springer-Verlag, New York. ISBN 978-0-387-84857-0. URL //www.springer.com/fr/book/9780387848570.

Hurvich CM, Tsai CL (1989). "Regression and time series model selection in small samples." *Biometrika*, **76**(2), 297–307. ISSN 0006-3444. doi:10.1093/biomet/76.2.297. URL https://academic.oup.com/biomet/article/76/2/297/265326.

Hurvich Clifford M, Simonoff Jeffrey S, Tsai Chih-Ling (2002). "Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **60**(2), 271–293. ISSN 1369-7412. doi:10.1111/1467-9868.00125. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00125.

Lee, Nelder, Pawitan (2006). "Generalized Linear Models with Random Effects: Unified Analysis via H-likelihood."

Lin X (1997). "Variance component testing in generalised linear models with random effects." *Biometrika*, **84**(2), 309–326. ISSN 0006-3444. doi:10.1093/biomet/84.2.309. URL https://academic.oup.com/biomet/article/84/2/309/233889.

Lin X, Zhang D (1999). "Inference in generalized additive mixed modelsby using smoothing splines." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **61**(2), 381–400. ISSN 1467-9868. doi:10.1111/1467-9868.00183. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00183.

Lindstrom MJ, Bates DM (1988). "Newton—Raphson and EM Algorithms for Linear Mixed-Effects Models for Repeated-Measures Data." *Journal of the American Statistical Association*, **83**(404), 1014–1022. ISSN 0162-1459. doi:10.1080/01621459.1988.10478693. URL https://doi.org/10.1080/01621459.1988.10478693.

Liu D, Lin X, Ghosh D (2007). "Semiparametric regression of multidimensional genetic pathway data: least-squares kernel machines and linear mixed models." *Biometrics*, **63**(4), 1079–1088. ISSN 0006-341X. doi:10.1111/j.1541-0420.2007.00799.x.

Liu JZ, Coull B (2017). "Robust Hypothesis Test for Nonlinear Effect with Gaussian Processes." *arXiv:1710.01406 [stat]*. URL http://arxiv.org/abs/1710.01406.

Maity A, Lin X (2011). "Powerful tests for detecting a gene effect in the presence of possible gene-gene interactions using garrote kernel machines." *Biometrics*, **67**(4), 1271–1284. ISSN 1541-0420. doi:10.1111/j.1541-0420.2011.01598.x.

Pawitan (2001). *In All Likelihood: Statistical Modelling and Inference Using Likelihood.* Oxford University Press, Oxford, New York. ISBN 978-0-19-850765-9.

Press TM (2006). "Gaussian Processes for Machine Learning." URL https://mitpress.mit.edu/books/gaussian-processes-machine-learning.

Reiss PT, Ogden RT (2009). "Smoothing Parameter Selection for a Class of Semiparametric Linear Models." *JRSS-B*, **71**(2).

Stein ML (1999). *Interpolation of Spatial Data: Some Theory for Kriging.* Springer Series in Statistics. Springer-Verlag, New York. ISBN 978-0-387-98629-6. URL //www.springer.com/us/book/9780387986296.

Wahba G (1985). "A Comparison of GCV and GML for Choosing the Smoothing Parameter in the Generalized Spline Smoothing Problem." *The Annals of Statistics*, **13**(4), 1378–1402. ISSN 0090-5364, 2168-8966. doi:10.1214/aos/1176349743. URL https://projecteuclid.org/euclid.aos/1176349743.

Wahba G (1990). *Spline Models for Observational Data.* CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics. ISBN 978-0-89871-244-5. doi:10.1137/1.9781611970128. URL https://epubs.siam.org/doi/book/10.1137/1.9781611970128.

Wecker WE, Ansley CF (1983). "The Signal Extraction Approach to Nonlinear Regression and Spline Smoothing." *Journal of the American Statistical Association*, **78**(381), 81–89. ISSN 0162-1459. doi:10.1080/01621459.1983.10477935. URL https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1983.10477935.

Zhan X, Plantinga A, Zhao N, Wu MC (2017). "A fast small-sample kernel independence test for microbiome community-level association analysis." *Biometrics*, **73**(4), 1453–1463. ISSN 0006-341X. doi:10.1111/biom.12684. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5592124/.

A. Derivation of Derivative wrt lambda

Corresponding to (2.2.3),

$$\mathbf{y} = \boldsymbol{\mu} + \mathbf{h} + \boldsymbol{\epsilon} \quad \text{where} \quad \mathbf{h} \sim \mathrm{N}(\mathbf{0}, \tau \mathbf{K}_\delta) \quad \boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \sigma^2 \mathbf{I}),$$

where $\mathbf{K}_\delta$ is the kernel matrix generated by $k_\delta(\mathbf{z}, \mathbf{z}')$.

The general model may be expressed in matrix form as (Reiss and Ogden 2009):

$$\mathbf{y} = \boldsymbol{\mu} + \Phi(\mathbf{X})^\top \boldsymbol{\beta} + \boldsymbol{\epsilon} \quad \text{where } \Phi(\mathbf{X})^\top \text{ is } n \times p, \tag{App.1}$$

where $\Phi(\mathbf{X})$ is the aggregation of columns $\phi(\mathbf{x})$ for all cases in the training set, and $\phi(\mathbf{x})$ is a function mapping a D-dimensional input vector $\mathbf{x}$ into an p-dimensional feature space. This model is fitted by penalized least squares, i.e., our estimate is

$$(\hat{\mu}, \hat{\boldsymbol{\beta}}) = \underset{\mu, \boldsymbol{\beta}}{\operatorname{argmin}} \, (\parallel \mathbf{y} - \boldsymbol{\mu} - \Phi(\mathbf{X})^\top \boldsymbol{\beta} \parallel^2 + \lambda \boldsymbol{\beta}^\top \boldsymbol{\beta}). \tag{App.2}$$

The development that follows depends on the following Assumptions:

- $\mathbf{1}^\top \boldsymbol{\Phi}(\mathbf{X})^\top = \mathbf{0}$,
- $\mathbf{y}$ is not in the column space of $\mathbf{1}$,

where $\mathbf{1}$ is a $n \times 1$ vector.

A.1. Derivative of REML

As our choice of matrix notation suggests, model (App.1) can be seen as equivalent to a linear mixed model, in the following sense. The criterion in (App.2) is proportional to the log likelihood for the partly observed "data" $(\mathbf{y}, \boldsymbol{\beta})$ with respect to the unknowns $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$, i.e., the best linear unbiased prediction (BLUP) criterion, for the mixed model

$$\mathbf{y}|\boldsymbol{\beta} \sim \mathrm{N}(\boldsymbol{\mu} + \Phi(\mathbf{X})^\top \boldsymbol{\beta}, \sigma^2 \mathbf{I}), \quad \boldsymbol{\beta} \sim \mathrm{N}(0, (\sigma^2/\lambda)\mathbf{I}).$$

Under this model, $Var(\mathbf{y}) = \sigma^2 \mathbf{V}_\lambda$ where

$$\mathbf{V}_\lambda = \mathbf{I} + \lambda^{-1} \Phi(\mathbf{X})^\top \Phi(\mathbf{X}) = \mathbf{I} + \lambda^{-1} \mathbf{K}_\delta. \tag{App.3}$$

The mixed model formulation motivates treating λ as a variance parameter to be estimated by maximizing the log likelihood

$$l(\mu, \lambda, \sigma^2|\mathbf{y}) = -\frac{1}{2}\Big[\log|\sigma^2 \mathbf{V}_\lambda| + (\mathbf{y} - \boldsymbol{\mu})^\top (\sigma^2 \mathbf{V}_\lambda)^{-1} (\mathbf{y} - \boldsymbol{\mu})\Big]$$

Maximizing this log likelihood results in estimating σ^2 with a downward bias, which is removed if we instead maximize the restricted log likelihood

$$l_R(\mu, \lambda, \sigma^2|\mathbf{y}) = -\frac{1}{2}\Big[\log|\sigma^2 \mathbf{V}_\lambda| + (\mathbf{y} - \boldsymbol{\mu})^\top (\sigma^2 \mathbf{V}_\lambda)^{-1} (\mathbf{y} - \boldsymbol{\mu}) + \log|\sigma^{-2} \mathbf{1}^\top \mathbf{V}_\lambda^{-1} \mathbf{1}|\Big]. \tag{App.4}$$

We shall refer to the resulting estimate of λ as the REML choice of the parameter.
For given μ and λ, the value of σ^2 maximizing the restricted log likelihood (App.4) is

$$\hat{\sigma}^2_{\mu,\lambda} = (\mathbf{y}-\boldsymbol{\mu})^\top \mathbf{V}_\lambda^{-1}(\mathbf{y}-\boldsymbol{\mu})/(n-1). \tag{App.5}$$

Substituting in this value and ignoring an additive constant leads to the profile restricted log likelihood

$$l_R(\mu,\lambda|\mathbf{y}) = -\frac{1}{2}\Big[\log|\mathbf{V}_\lambda| + \log|\mathbf{1}^\top\mathbf{V}_\lambda^{-1}\mathbf{1}| + (n-1)\log\{(\mathbf{y}-\boldsymbol{\mu})^\top\mathbf{V}_\lambda^{-1}(\mathbf{y}-\boldsymbol{\mu})\}\Big]. \tag{App.6}$$

For given λ, the value of μ maximizing this last expression is the generalized least square fit $\hat{\mu}_\lambda = (\mathbf{1}^\top\mathbf{V}_\lambda^{-1}\mathbf{1})^{-1}\mathbf{1}^\top\mathbf{V}_\lambda^{-1}\mathbf{y}$.
Using the readily verified equality $\mathbf{V}_\lambda^{-1} = \mathbf{I} - \mathbf{A}_\lambda$, the following key facts about $\mathbf{P}_\lambda$ can be shown to hold under Assumptions 1-2:

$$\mathbf{P}_\lambda = \mathbf{I} - \mathbf{H}_\lambda, \tag{App.7}$$

where $\mathbf{H}_\lambda$ is the hat matrix defined by $\hat{\mathbf{y}} = \mathbf{H}_\lambda\mathbf{y}$ and given by

$$\mathbf{H}_\lambda = \mathbf{1}(\mathbf{1}^\top\mathbf{1})^{-1}\mathbf{1}^\top + \mathbf{A}_\lambda, \tag{App.8}$$

$$\mathbf{V}_\lambda^{-1}\mathbf{1} = \mathbf{1}, \tag{App.9}$$

$$\mathbf{P}_\lambda^k = \mathbf{V}_\lambda^{-k} - \mathbf{1}(\mathbf{1}^\top\mathbf{1})^{-1}\mathbf{1}^\top. \text{ for } k = 1,\ 2, \ldots \tag{App.10}$$

Under Assumptions 1-2, repeated application of (App.9) gives $\mathbf{y} - \hat{\boldsymbol{\mu}}_\lambda = [\mathbf{I} - \mathbf{1}(\mathbf{1}^\top\mathbf{1})^{-1}\mathbf{1}^\top]\mathbf{y}$, and hence

$$(\mathbf{y}-\hat{\boldsymbol{\mu}}_\lambda)^\top\mathbf{V}_\lambda^{-1}(\mathbf{y}-\hat{\boldsymbol{\mu}}_\lambda) = \mathbf{y}^\top\mathbf{P}_\lambda\mathbf{y}. \tag{App.11}$$

Substituting (App.11) into (App.6) yields the profile restricted log likelihood for λ alone:

$$l_R(\lambda|\mathbf{y}) = -\frac{1}{2}\Big[\log|\mathbf{V}_\lambda| + \log|\mathbf{1}^\top\mathbf{V}_\lambda^{-1}\mathbf{1}| + (n-1)\log(\mathbf{y}^\top\mathbf{P}_\lambda\mathbf{y})\Big] \tag{App.12}$$

Setting the derivative of (App.12) with respect of λ to zero will yield an equation for the REML estimate of λ. By (App.9) again, $\log|\mathbf{1}^\top\mathbf{V}_\lambda^{-1}\mathbf{1}| = \log|\mathbf{1}^\top\mathbf{1}|$, which does not depend on λ, so the differentiation reduces to finding the derivatives of $\log|\mathbf{V}_\lambda|$ and $\log(\mathbf{y}^\top\mathbf{P}_\lambda\mathbf{y})$. To that end we shall need the (component-wise) derivatives of $\mathbf{V}_\lambda$ and $\mathbf{P}_\lambda$ with respect to λ. These can be shown to be:

$$\frac{\partial\mathbf{V}_\lambda}{\partial\lambda} = \lambda^{-1}(\mathbf{I} - \mathbf{V}_\lambda), \tag{App.13}$$

$$\frac{\partial\mathbf{P}_\lambda}{\partial\lambda} = \lambda^{-1}(\mathbf{P}_\lambda - \mathbf{P}_\lambda^2). \tag{App.14}$$

A formula in (Lindstrom and Bates 1988)(p. 1016), together with (App.13), leads to

$$\frac{\partial}{\partial\lambda}\log|\mathbf{V}_\lambda| = \lambda^{-1}\mathrm{tr}(\mathbf{V}_\lambda^{-1} - \mathbf{I}).$$

By (App.10), $\mathrm{tr}(\mathbf{V}_\lambda^{-1}) = \mathrm{tr}(\mathbf{P}_\lambda) + \mathrm{tr}[\mathbf{I} - \mathbf{1}(\mathbf{1}^\top\mathbf{1})^{-1}\mathbf{1}^\top] = \mathrm{tr}(\mathbf{P}_\lambda) + 1$, so we conclude that

$$\frac{\partial}{\partial\lambda}\log|\mathbf{V}_\lambda| = \lambda^{-1}[\mathrm{tr}(\mathbf{P}_\lambda) - (n-1)]. \tag{App.15}$$

By Assumption 2, $\mathbf{y}^\top\mathbf{P}_\lambda\mathbf{y} > 0$. Thus, using (App.14), we obtain

$$\frac{\partial}{\partial\lambda}\log(\mathbf{y}^\top\mathbf{P}_\lambda\mathbf{y}) = \lambda^{-1}\Big[1 - \frac{\mathbf{y}^\top\mathbf{P}_\lambda^2\mathbf{y}}{\mathbf{y}^\top\mathbf{P}_\lambda\mathbf{y}}\Big]. \tag{App.16}$$

Under our Assumptions, the matrix

$$\mathbf{P}_\lambda = \mathbf{V}_\lambda^{-1} - \mathbf{V}_\lambda^{-1}\mathbf{1}(\mathbf{1}^\top\mathbf{V}_\lambda^{-1}\mathbf{1})^{-1}\mathbf{1}^\top\mathbf{V}_\lambda^{-1}$$

which plays a role in some treatments of mixed model theory, turns out to be important for both the REML and the GCV approach to choosing λ.

By (App.12), (App.15) and (App.16), we obtain

$$\frac{\partial l_R(\lambda|\mathbf{y})}{\partial\lambda} = \frac{1}{2\lambda}\Big[(n-1)\frac{\mathbf{y}^\top\mathbf{P}_\lambda^2\mathbf{y}}{\mathbf{y}^\top\mathbf{P}_\lambda\mathbf{y}} - \mathrm{tr}(\mathbf{P}_\lambda)\Big]. \tag{App.17}$$

Thus by (App.7), (App.11) and (App.17), $\frac{\partial l_R(\lambda|\mathbf{y})}{\partial\lambda} = 0$ implies

$$\frac{(\mathbf{y} - \hat{\boldsymbol{\mu}}_\lambda)^\top\mathbf{V}_\lambda^{-1}(\mathbf{y} - \hat{\boldsymbol{\mu}}_\lambda)}{n-1} = \frac{\mathbf{y}^\top(\mathbf{I} - \mathbf{H}_\lambda)^2\mathbf{y}}{\mathrm{tr}(\mathbf{I} - \mathbf{H}_\lambda)}. \tag{App.18}$$

which is also

$$\frac{\mathbf{y}^\top\mathbf{P}_\lambda\mathbf{y}}{n-1} = \frac{\mathbf{y}^\top\mathbf{P}_\lambda^2\mathbf{y}}{\mathrm{tr}(\mathbf{P}_\lambda)} \quad \text{or} \quad \frac{\mathbf{y}^\top\mathbf{P}_\lambda\mathbf{y}}{\mathbf{y}^\top\mathbf{P}_\lambda^2\mathbf{y}} = \frac{n-1}{\mathrm{tr}(\mathbf{P}_\lambda)}, \tag{App.19}$$

where $\hat{\mu}_\lambda$ and $\mathbf{H}_\lambda$ are the parameter estimate and hat matrix, respectively, obtained with smoothing parameter value λ. The left side of (App.18) is the REML estimate of σ^2 (Wahba 1990). The right side equals $\| \mathbf{y} - \hat{\mathbf{y}} \|^2 / [n - \mathrm{tr}(\mathbf{H}_\lambda)]$, an estimate of σ^2 based on viewing $tr(\mathbf{H}_\lambda)$ as the degrees of freedom of the smoother (Pawitan 2001)(p. 487) and (Lee, Nelder, and Pawitan 2006)(p. 279). In other words, when λ is estimated by REML, the REML error variance estimate agrees with the "smoothing-theoretic" variance estimate.

A.2. Derivative of GCV

The GCV criterion is given by

$$GCV(\lambda) = \frac{\| \mathbf{y} - \hat{\mathbf{y}} \|^2}{[1 - \mathrm{tr}(\mathbf{H}_\lambda)/n]^2} = \frac{\mathbf{y}^\top(\mathbf{I} - \mathbf{H}_\lambda)^2\mathbf{y}}{[\mathrm{tr}(\mathbf{I} - \mathbf{H}_\lambda)]^2} = \frac{\mathbf{y}^\top\mathbf{P}_\lambda^2\mathbf{y}}{[\mathrm{tr}(\mathbf{P}_\lambda)]^2},$$

with the last equality following from (App.7). This criterion, originally proposed by (Craven and Wahba 1979), is an approximation to $\frac{1}{n}\sum_{i=1}^n \frac{(y_i - \hat{y}_i)^2}{(1 - h_{\lambda[ii]})^2}$, where $h_{\lambda[11]}, ..., h_{\lambda[nn]}$ are the diagonal elements of $\mathbf{H}_\lambda$. The latter expression can be shown (at least in some smoothing problems) to be equal to the leave-one-out cross-validation criterion, but lacks an invariance-under-reparametrization property that is gained by instead using GCV (Wahba 1990)(pp. 52-53). Using (App.14), we can obtain

$$\frac{\partial GCV(\lambda)}{\partial\lambda} = \frac{2}{\lambda[\text{tr}(\mathbf{P}_\lambda)]^3}\Big[\text{tr}(\mathbf{P}_\lambda^2)\mathbf{y}^\top P_\lambda^2\mathbf{y} - \text{tr}(\mathbf{P}_\lambda)\mathbf{y}^\top P_\lambda^3\mathbf{y}\Big]. \qquad \text{(App.20)}$$

Thus at the GCV-minimizing λ we have

$$\frac{\mathbf{y}^\top\mathbf{P}_\lambda^3\mathbf{y}}{\text{tr}(\mathbf{P}_\lambda^2)} = \frac{\mathbf{y}^\top\mathbf{P}_\lambda^2\mathbf{y}}{\text{tr}(\mathbf{P}_\lambda)} \quad \text{or} \quad \frac{\mathbf{y}^\top\mathbf{P}_\lambda^3\mathbf{y}}{\mathbf{y}^\top\mathbf{P}_\lambda^2\mathbf{y}} = \frac{\text{tr}(\mathbf{P}_\lambda^2)}{\text{tr}(\mathbf{P}_\lambda)}$$

A.3. Derivative of AIC

The AIC criterion is given by

$$AIC(\lambda) = \log(\parallel \mathbf{y}-\hat{\mathbf{y}} \parallel^2) + \frac{2}{n}[\text{tr}(\mathbf{H}_\lambda)+1] = \log(\mathbf{y}^\top\mathbf{P}_\lambda^2\mathbf{y}) + \frac{2}{n}\text{tr}(\mathbf{I}-\mathbf{P}_\lambda) + \frac{2}{n}.$$

Using (App.14), we can obtain

$$\frac{\partial AIC(\lambda)}{\partial\lambda} = \frac{2}{\lambda}\Big[1 - \frac{\mathbf{y}^\top\mathbf{P}_\lambda^3\mathbf{y}}{\mathbf{y}^\top\mathbf{P}_\lambda^2\mathbf{y}} - \frac{1}{n}\text{tr}(\mathbf{P}_\lambda - \mathbf{P}_\lambda^2)\Big]. \qquad \text{(App.21)}$$

Thus at the AIC-minimizing λ we have

$$\frac{\mathbf{y}^\top\mathbf{P}_\lambda^3\mathbf{y}}{\mathbf{y}^\top\mathbf{P}_\lambda^2\mathbf{y}} = \frac{\text{tr}(\mathbf{I}-\mathbf{P}_\lambda+\mathbf{P}_\lambda^2)}{n}.$$

B. Derivation of the REML based Test Statistic

B.1. Derivation of the Score Test Statistic

In this section, we derive the score test statistic based on REML (Maity and Lin 2011). Denote $\mathbf{V}(\boldsymbol{\theta}) = \sigma^2\mathbf{V}_\lambda = \sigma^2\mathbf{I}+\tau\mathbf{K}_\delta$, where $\boldsymbol{\theta} = (\delta,\tau,\sigma^2)$. The REML given in (App.4) can be rewritten as

$$l_R = -\frac{1}{2}\Big[\log|\mathbf{V}(\boldsymbol{\theta})| + \log|\mathbf{1}^\top\mathbf{V}(\boldsymbol{\theta})^{-1}\mathbf{1}| + (\mathbf{y}-\boldsymbol{\mu})^\top\mathbf{V}(\boldsymbol{\theta})^{-1}(\mathbf{y}-\boldsymbol{\mu})\Big]. \qquad \text{(App.22)}$$

Under $H_0:\delta=0$ (2.2.2), we set $\boldsymbol{\theta}_0 = (0,\tau,\sigma^2)$ and

$$\mathbf{P}_0(\boldsymbol{\theta}_0) = \mathbf{V}(\boldsymbol{\theta}_0)^{-1} - \mathbf{V}(\boldsymbol{\theta}_0)^{-1}\mathbf{1}[\mathbf{1}^\top\mathbf{V}(\boldsymbol{\theta}_0)^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top\mathbf{V}(\boldsymbol{\theta}_0)^{-1}.$$

Take the derivative of (App.22) with respect to δ,

$$\begin{aligned}\frac{\partial l_R}{\partial \delta} =& -\frac{1}{2}\Big[\frac{\partial \log|\mathbf{V}(\boldsymbol{\theta})|}{\partial \delta} + \frac{\partial \log|\mathbf{1}^\top \mathbf{V}(\boldsymbol{\theta})^{-1}\mathbf{1}|}{\partial \delta} + \frac{\partial (\mathbf{y}-\boldsymbol{\mu})^\top \mathbf{V}(\boldsymbol{\theta})^{-1}(\mathbf{y}-\boldsymbol{\mu})}{\partial \delta}\Big] \\ =& -\frac{1}{2}\Big[\mathrm{tr}(\mathbf{V}(\boldsymbol{\theta})^{-1}\frac{\partial \mathbf{V}(\boldsymbol{\theta})}{\partial \delta}) + \mathrm{tr}([\mathbf{1}^\top \mathbf{V}(\boldsymbol{\theta})^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top \frac{\partial \mathbf{V}(\boldsymbol{\theta})^{-1}}{\partial \delta}\mathbf{1}) \\ &+ (\mathbf{y}-\boldsymbol{\mu})^\top \frac{\partial \mathbf{V}(\boldsymbol{\theta})^{-1}}{\partial \delta}(\mathbf{y}-\boldsymbol{\mu})\Big] \\ =& -\frac{1}{2}\Big[\mathrm{tr}(\mathbf{V}(\boldsymbol{\theta})^{-1}\tau(\partial \mathbf{K}_\delta)) - \mathrm{tr}(\tau(\partial \mathbf{K}_\delta)\mathbf{V}(\boldsymbol{\theta})^{-1}\mathbf{1}[\mathbf{1}^\top \mathbf{V}(\boldsymbol{\theta})^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top \mathbf{V}(\boldsymbol{\theta})^{-1}) \\ &- (\mathbf{y}-\boldsymbol{\mu})^\top \mathbf{V}(\boldsymbol{\theta})^{-1}\tau(\partial \mathbf{K}_\delta)\mathbf{V}(\boldsymbol{\theta})^{-1}(\mathbf{y}-\boldsymbol{\mu})\Big] \\ =& \frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\top \mathbf{V}(\boldsymbol{\theta})^{-1}\tau(\partial \mathbf{K}_\delta)\mathbf{V}(\boldsymbol{\theta})^{-1}(\mathbf{y}-\boldsymbol{\mu}) \\ &- \frac{1}{2}\mathrm{tr}\Big[\tau(\partial \mathbf{K}_\delta)\big[\mathbf{V}(\boldsymbol{\theta})^{-1} - \mathbf{V}(\boldsymbol{\theta})^{-1}\mathbf{1}[\mathbf{1}^\top \mathbf{V}(\boldsymbol{\theta})^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top \mathbf{V}(\boldsymbol{\theta})^{-1}\big]\Big], \qquad \text{(App.23)}\end{aligned}$$

where $\partial \mathbf{K}_\delta$ is the derivative kernel matrix whose $(i,j)^{th}$ entry is $\frac{\partial k_\delta(\mathbf{x},\mathbf{x}')}{\partial \delta}$. If we further denote $\mathbf{K}_0 = \mathbf{K}_\delta|_{\delta=0}$ and $\partial \mathbf{K}_0 = (\partial \mathbf{K}_\delta)|_{\delta=0}$, we get the REML based score function of δ evaluated at H_0

$$S_{\delta=0} = \frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\top \mathbf{V}(\boldsymbol{\theta}_0)^{-1}\tau(\partial \mathbf{K}_0)\mathbf{V}(\boldsymbol{\theta}_0)^{-1}(\mathbf{y}-\boldsymbol{\mu}) - \frac{1}{2}\mathrm{tr}[\tau(\partial \mathbf{K}_0)\mathbf{P}_0].$$

To test for $H_0 : \delta = 0$, we propose to use the score-based test statistic

$$\hat{T}_0 = \hat{\tau}(\mathbf{y}-\hat{\boldsymbol{\mu}})^\top \mathbf{V}_0^{-1}(\partial \mathbf{K}_0)\mathbf{V}_0^{-1}(\mathbf{y}-\hat{\boldsymbol{\mu}}), \qquad \text{(App.24)}$$

where $\mathbf{V}_0 = \hat{\sigma}^2\mathbf{I} + \hat{\tau}\mathbf{K}_0$.

B.2. The Null Distribution of the Test Statistic

For simplicity, we denote

$$\mathbf{V} = \mathbf{V}(\boldsymbol{\theta}),$$
$$\mathbf{P} = \mathbf{P}(\boldsymbol{\theta}) = \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{1}[\mathbf{1}^\top \mathbf{V}^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top \mathbf{V}^{-1}.$$

With similar derivation as (App.23), for each $\theta_i \in \boldsymbol{\theta} = (\delta, \tau, \sigma^2)$, we have

$$\frac{\partial l_R}{\partial \theta_i} = -\frac{1}{2}\Big[\mathrm{tr}(\mathbf{P}\frac{\partial \mathbf{V}}{\partial \theta_i}) - (\mathbf{y}-\boldsymbol{\mu})^\top \mathbf{V}^{-1}(\frac{\partial \mathbf{V}}{\partial \theta_i})\mathbf{V}^{-1}(\mathbf{y}-\boldsymbol{\mu})\Big]. \qquad \text{(App.25)}$$

From (Liu, Lin, and Ghosh 2007) we know $\hat{\boldsymbol{\mu}} = [\mathbf{1}^\top \mathbf{V}^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top \mathbf{V}^{-1}\mathbf{y}$, plug it in (Lin and Zhang 1999), we obtain

$$(\mathbf{y}-\boldsymbol{\mu})^\top \mathbf{V}^{-1} = \mathbf{y}^\top(\mathbf{I} - \mathbf{1}[\mathbf{1}^\top \mathbf{V}^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top \mathbf{V}^{-1})^\top \mathbf{V}^{-1} = \mathbf{y}^\top.\mathbf{P}$$

(App.27) becomes

$$\frac{\partial l_R}{\partial \theta_i} = -\frac{1}{2}\Big[\mathrm{tr}(\mathbf{P}\frac{\partial \mathbf{V}}{\partial \theta_i}) - \mathbf{y}^\top \mathbf{P}(\frac{\partial \mathbf{V}}{\partial \theta_i})\mathbf{P}\mathbf{y}\Big].$$

The second-order partial derivatives with respect to θ_i and θ_j is

$$\begin{aligned}\frac{\partial^2 l_R}{\partial\theta_i\partial\theta_j} =& -\frac{1}{2}\Big[\mathrm{tr}(\frac{\partial\mathbf{P}}{\partial\theta_j}\frac{\partial\mathbf{V}}{\partial\theta_i}) + \mathrm{tr}(\mathbf{P}\frac{\partial^2\mathbf{V}}{\partial\theta_i\partial\theta_j}) + \mathbf{y}^\top\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_i})\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_j})\mathbf{P}\mathbf{y}\\ &+ \mathbf{y}^\top\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_j})\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_i})\mathbf{P}\mathbf{y} - \mathbf{y}^\top\mathbf{P}\frac{\partial^2\mathbf{V}}{\partial\theta_i\partial\theta_j}\mathbf{P}\mathbf{y}\Big],\end{aligned} \tag{App.26}$$

where we have used the fact that

$$\begin{aligned}\frac{\partial\mathbf{P}}{\partial\theta_j} =& -\mathbf{V}^{-1}\frac{\partial\mathbf{V}}{\partial\theta_j}\mathbf{V}^{-1} + \mathbf{V}^{-1}\frac{\partial\mathbf{V}}{\partial\theta_j}\mathbf{V}^{-1}\mathbf{1}[\mathbf{1}^\top\mathbf{V}^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top\mathbf{V}^{-1}\\ &+ \mathbf{V}^{-1}\mathbf{1}[\mathbf{1}^\top\mathbf{V}^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top\mathbf{V}^{-1}\frac{\partial\mathbf{V}}{\partial\theta_j}\mathbf{V}^{-1}\\ &- \mathbf{V}^{-1}\mathbf{1}([\mathbf{1}^\top\mathbf{V}^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top\mathbf{V}^{-1}\frac{\partial\mathbf{V}}{\partial\theta_j}\mathbf{V}^{-1}\mathbf{1}[\mathbf{1}^\top\mathbf{V}^{-1}\mathbf{1}]^{-1})\mathbf{1}^\top\mathbf{V}^{-1}\\ =& -\mathbf{P}\frac{\partial\mathbf{V}}{\partial\theta_j}\mathbf{P}.\end{aligned}$$

Then (App.26) turns into

$$\begin{aligned}\frac{\partial^2 l_R}{\partial\theta_i\partial\theta_j} =& -\frac{1}{2}\Big[-\mathrm{tr}(\mathbf{P}\frac{\partial\mathbf{V}}{\partial\theta_j}\mathbf{P}\frac{\partial\mathbf{V}}{\partial\theta_i}) + \mathrm{tr}(\mathbf{P}\frac{\partial^2\mathbf{V}}{\partial\theta_i\partial\theta_j}) + \mathbf{y}^\top\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_i})\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_j})\mathbf{P}\mathbf{y}\\ &+ \mathbf{y}^\top\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_j})\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_i})\mathbf{P}\mathbf{y} - \mathbf{y}^\top\mathbf{P}\frac{\partial^2\mathbf{V}}{\partial\theta_i\partial\theta_j}\mathbf{P}\mathbf{y}\Big].\end{aligned} \tag{App.27}$$

Since

$$\begin{aligned}\mathrm{E}(\mathbf{P}\mathbf{y}\mathbf{y}^\top) &= \mathbf{P}[\mathrm{Var}(\mathbf{y}) + (\mathrm{E}\mathbf{y})(\mathrm{E}\mathbf{y})^\top] = \mathbf{P}[\mathbf{V} + \boldsymbol{\mu}\boldsymbol{\mu}^\top] = \mathbf{P}\mathbf{V},\\ \mathbf{P}\mathbf{V}\mathbf{P} &= \mathbf{P}[\mathbf{I} - \mathbf{1}[\mathbf{1}^\top\mathbf{V}^{-1}\mathbf{1}]^{-1}\mathbf{1}^\top\mathbf{V}^{-1}] = \mathbf{P},\end{aligned}$$

we get

$$\begin{aligned}\mathrm{E}\Big[\mathbf{y}^\top\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_j})\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_i})\mathbf{P}\mathbf{y}\Big] =& \mathrm{tr}\Big(\mathrm{E}\Big[\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_j})\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_i})\mathbf{P}\mathbf{y}\mathbf{y}^\top\Big]\Big)\\ =& \mathrm{tr}\Big(\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_j})\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_i})\mathbf{P}\mathbf{V}\Big)\\ =& \mathrm{tr}\Big(\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_j})\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_i})\Big),\\ \mathrm{E}\Big[\mathbf{y}^\top\mathbf{P}\frac{\partial^2\mathbf{V}}{\partial\theta_i\partial\theta_j}\mathbf{P}\mathbf{y}\Big] =& \mathrm{tr}\Big(\mathbf{P}\frac{\partial^2\mathbf{V}}{\partial\theta_i\partial\theta_j}\Big).\end{aligned}$$

Therefore,

$$\mathbf{I}_{\theta_i,\theta_j} = -\mathrm{E}\Big[\frac{\partial^2 l_R}{\partial\theta_i\partial\theta_j}\Big] = \frac{1}{2}\mathrm{tr}\Big(\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_j})\mathbf{P}(\frac{\partial\mathbf{V}}{\partial\theta_i})\Big).$$

C. Figures for 4 Different beta's of Exponential Weighting

Below shows the performances under four different βs of exponential weighting: a fixed value 1, $\min\{RSS\}_{d=1}^{D}/10$, $\text{median}\{RSS\}_{d=1}^{D}$ and $\max\{RSS\}_{d=1}^{D} * 2$. Here $\{RSS\}_{d=1}^{D}$ are the set of residual sum of squares of D base kernels. Lines refer to the different combination of tuning parameter selection (colors) and βs (line types).

Generally speaking, the differences of different βs in bootstrap test are more obvious than in asymptotic test. For asymptotic test, min can guarantee correct Type I error and maintain better power under the alternative (Figure 3-5, column 1s) while the other three βs are similar. In terms of bootstrap test, fixed, med and max also have similar performances, but fixed works better if base kernels are simple and finite-dimensional (Figure 7, column 2). In terms of small β (min), it has fairly greater power under the alternative while guarantees correct Type I error under the null at the same time (Figure 8, 10, column 2-4s; Figure 9). Otherwise when the data-generating kernel is strictly simpler than kernels in the library, min potentially cannot guarantee correct Type I error (Figure 8, 10, column 1s). And GMPML is not a good partner of it (Figure 8-9, column 2-3).

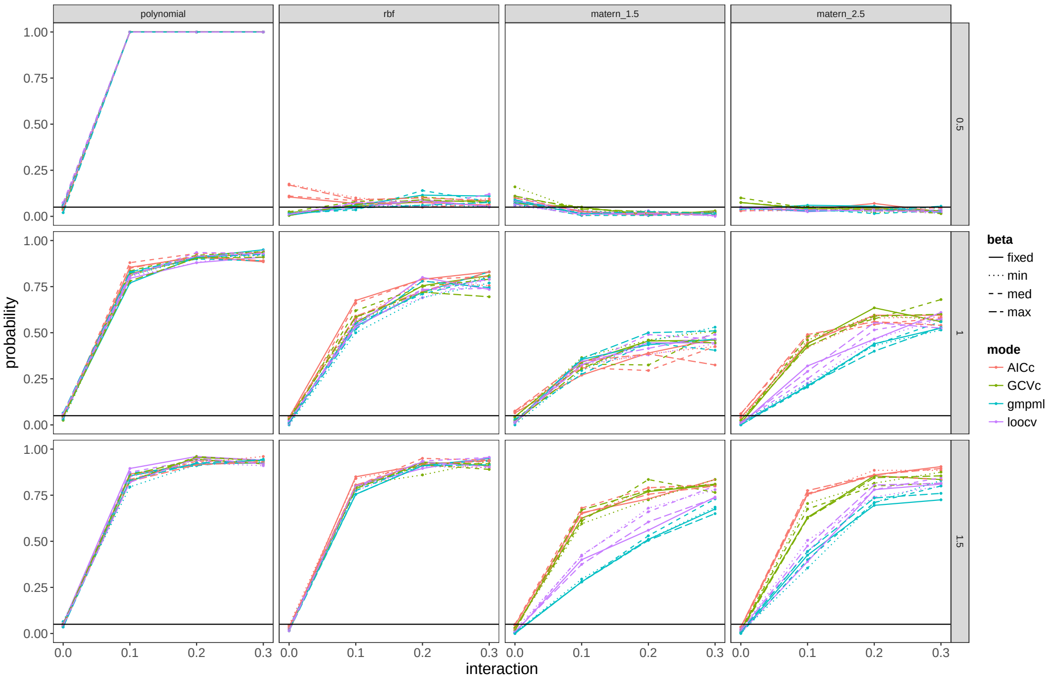

Figure 11: Asym, True kernel only

Affiliation:

Wenying Deng, Jeremiah Zhe Liu, Brent Coull
Department of Biostatistics
Harvard University
Cambridge, MA 02138 USA
E-mail: {wdeng@hsph, zhl112@mail, bcoull@hsph}.harvard.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd

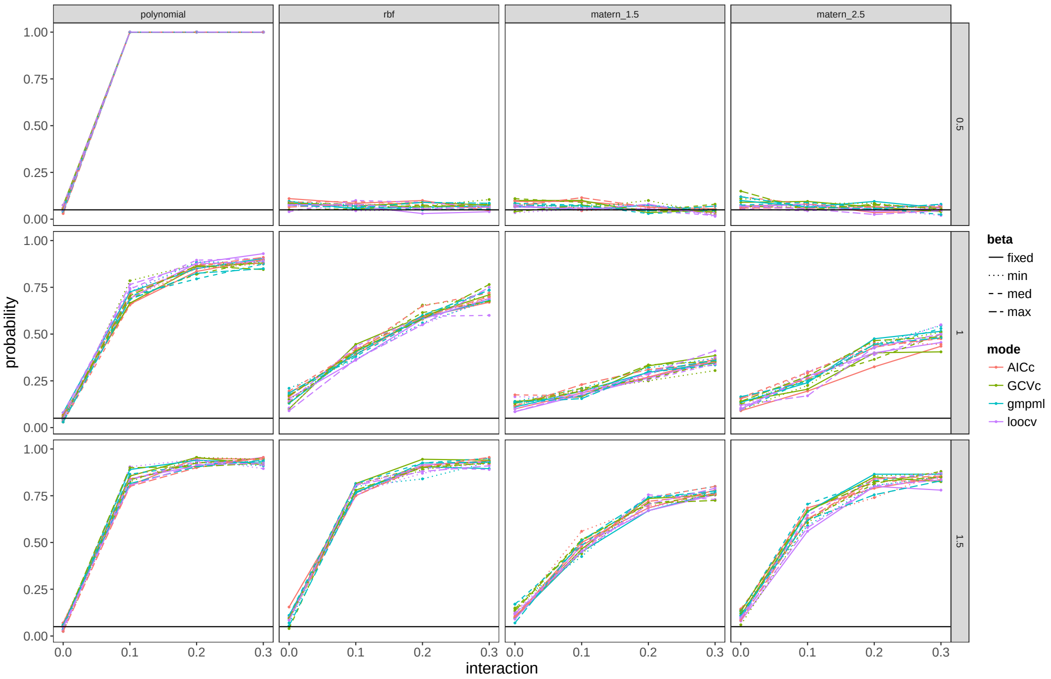

Figure 12: Asym, 3 Polynomial kernels

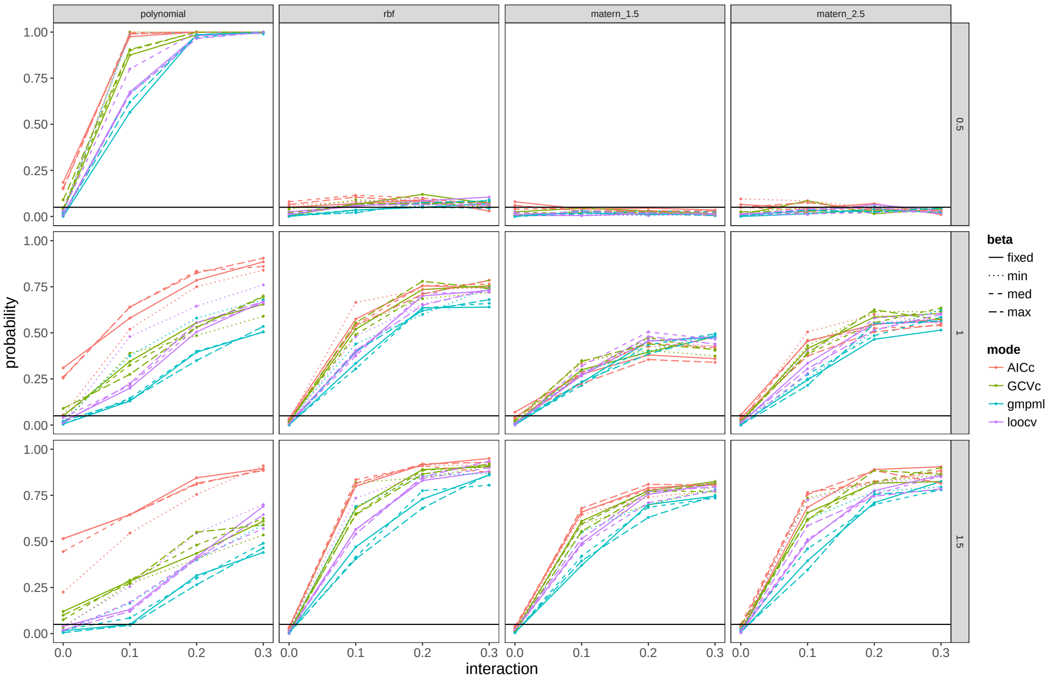

Figure 13: Asym, 3 RBF kernels

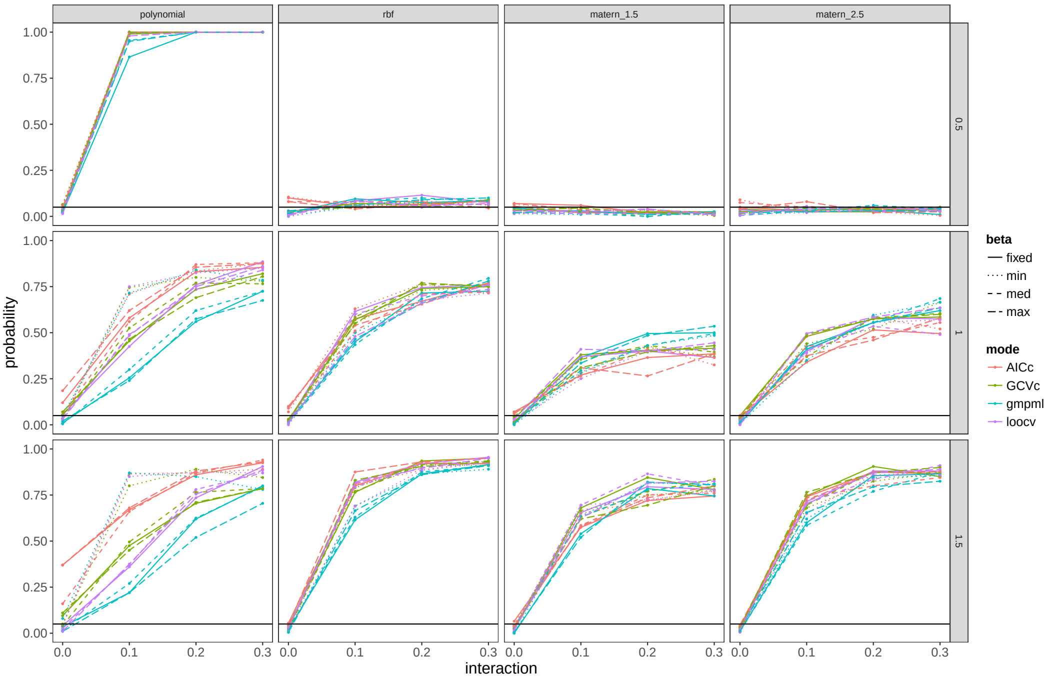

Figure 14: Asym, 3 Polynomial kernels and 3 RBF kernels

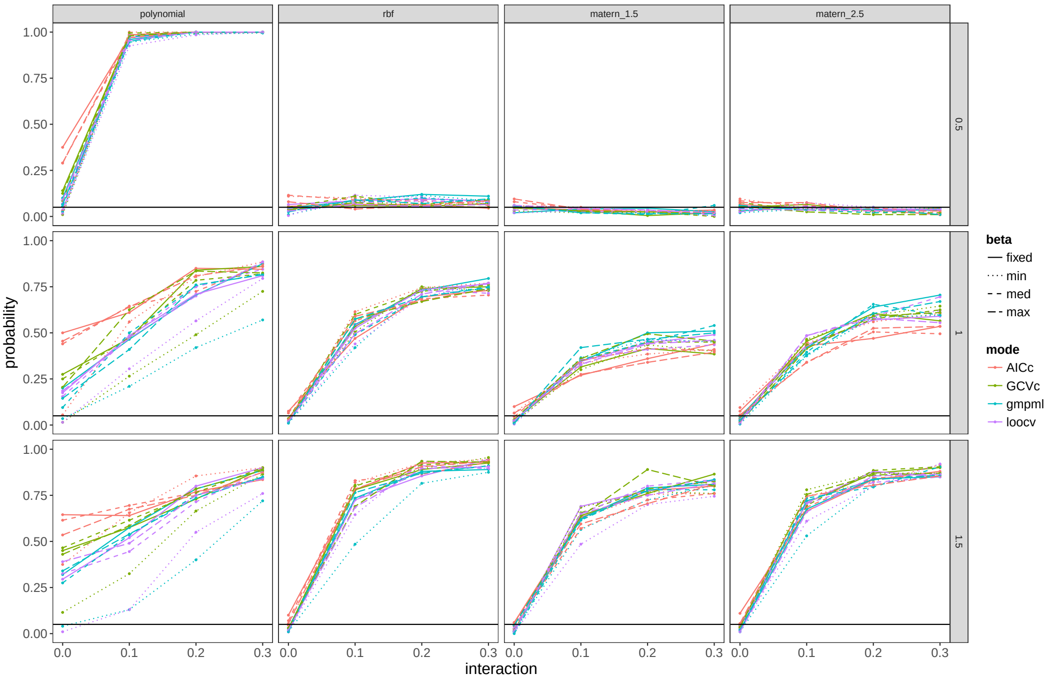

Figure 15: Asym, 3 Matern kernels and 3 RBF kernels

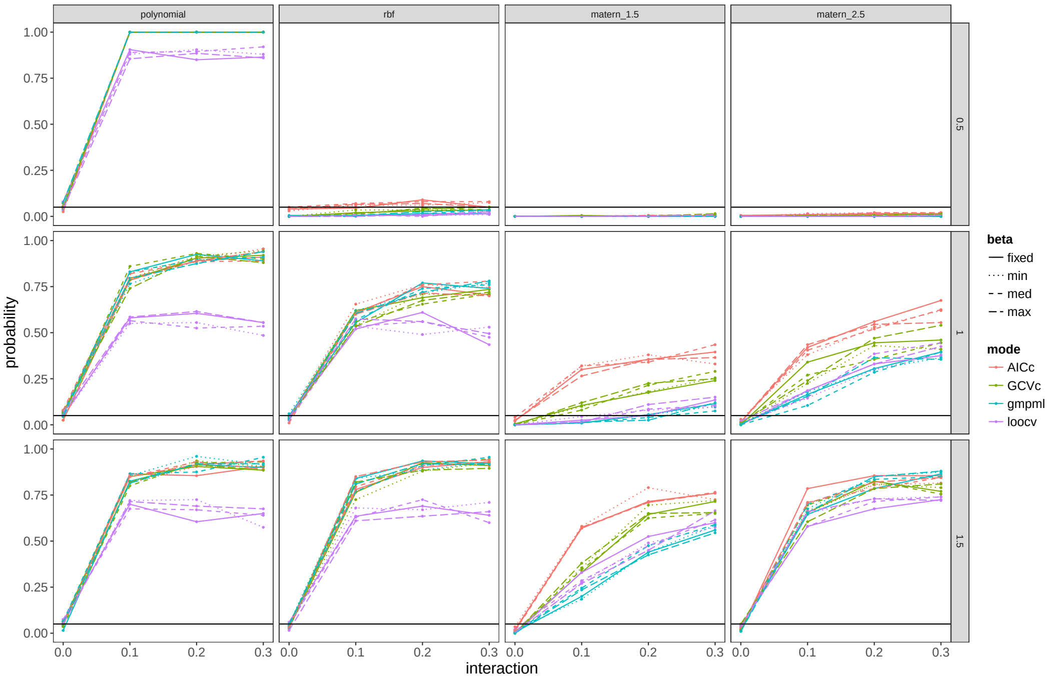

Figure 16: Boot, True kernel only

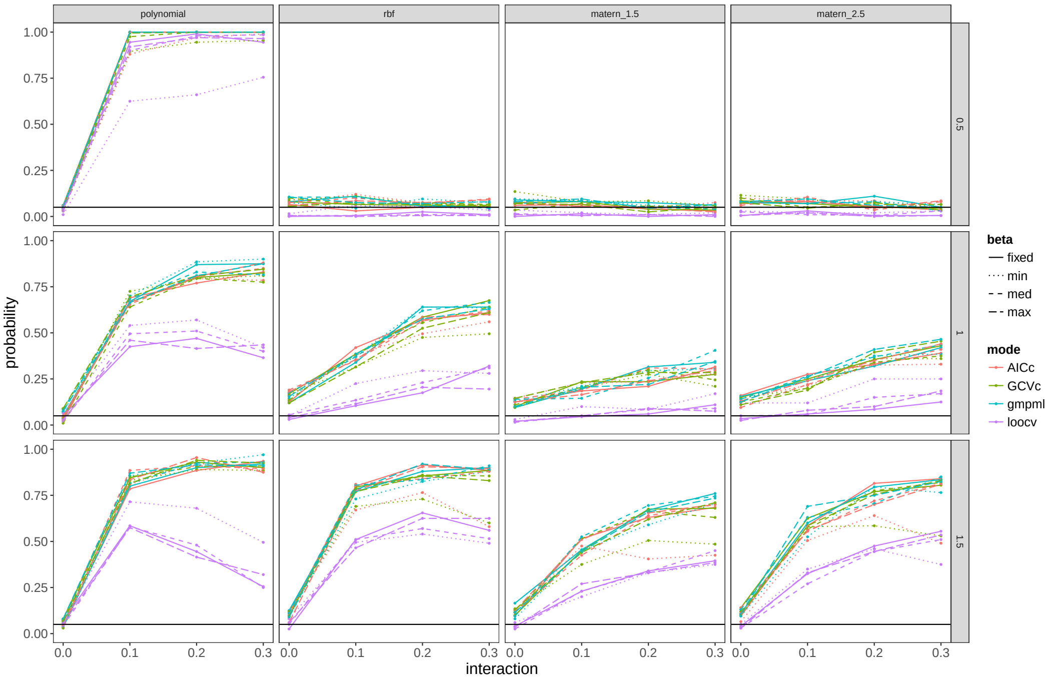

Figure 17: Boot, 3 Polynomial kernels

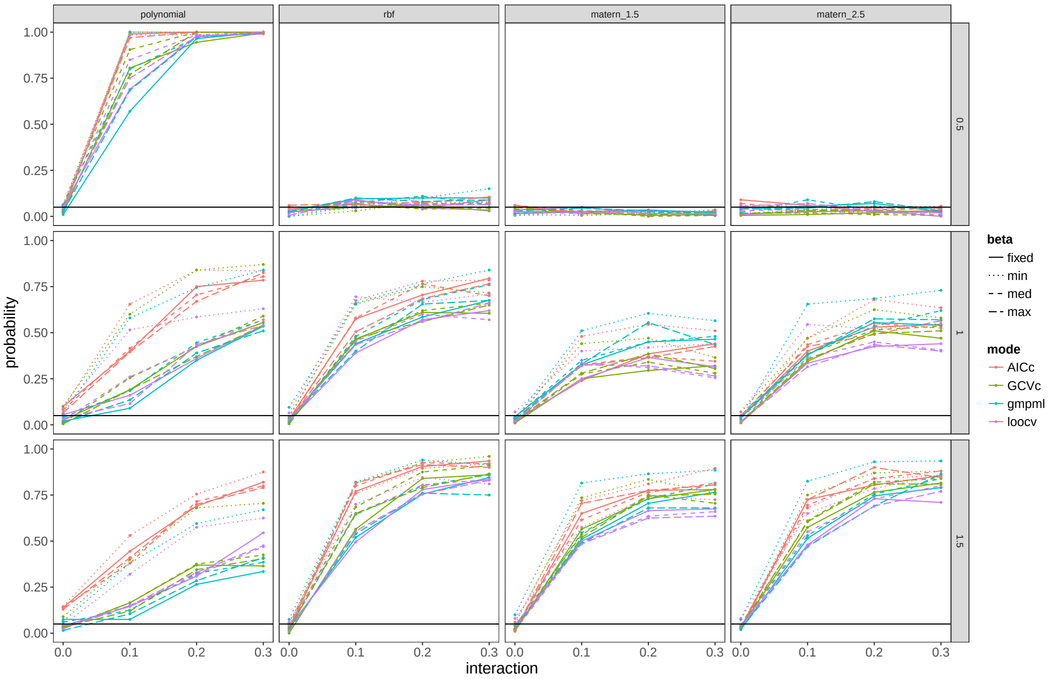

Figure 18: Boot, 3 RBF kernels

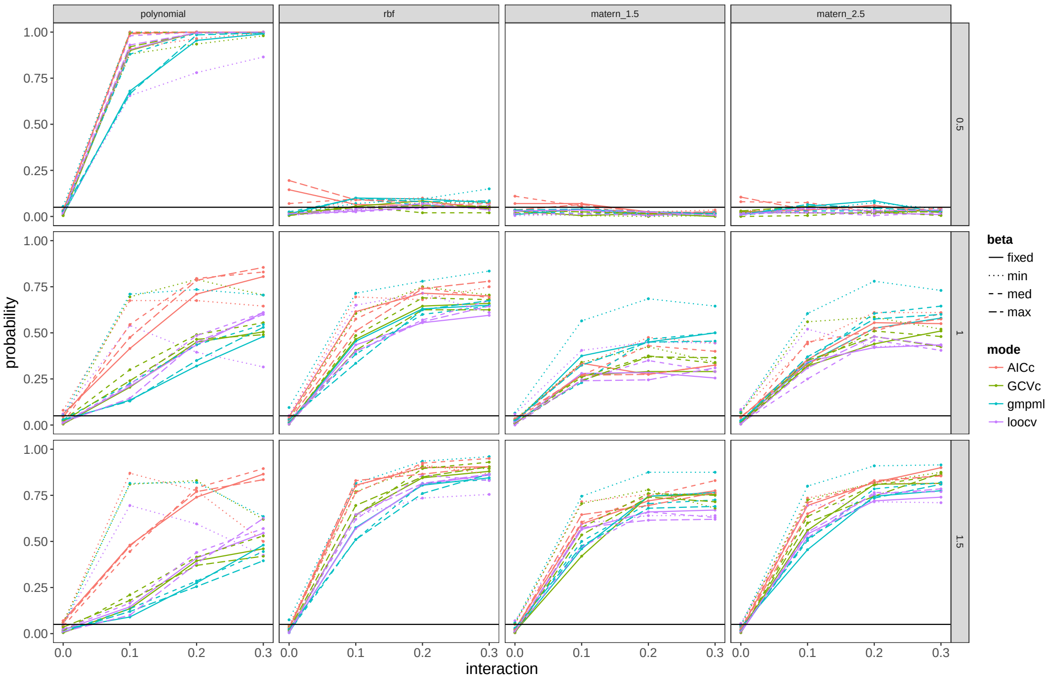

Figure 19: Boot, 3 Polynomial kernels and 3 RBF kernels

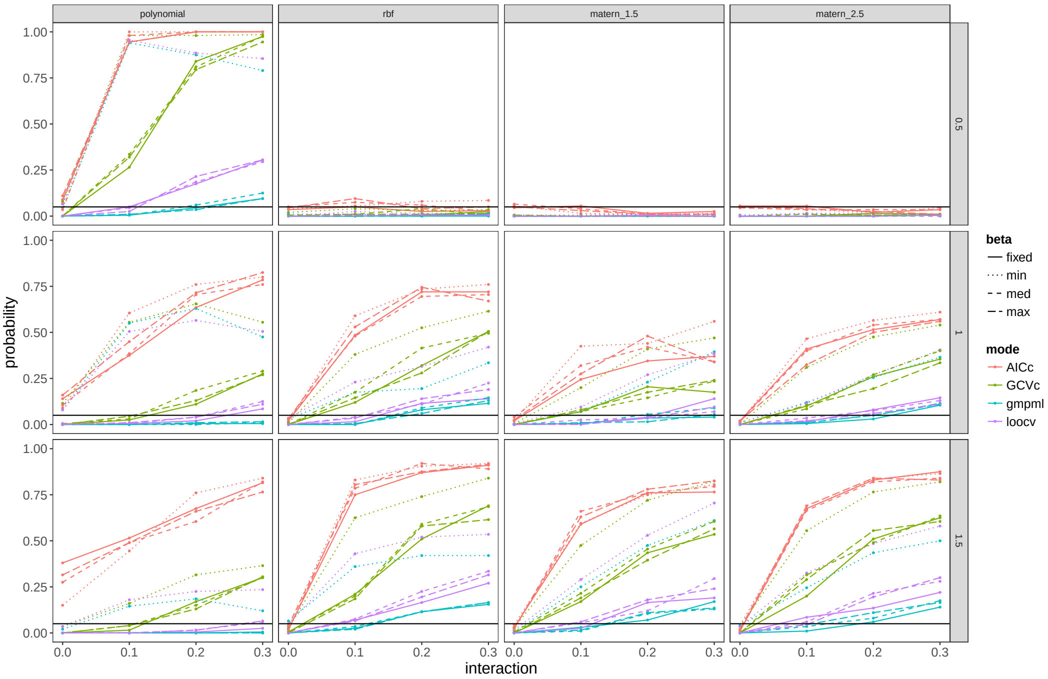

Figure 20: Boot, 3 Matern kernels and 3 RBF kernels