CVEK-boot Report
Wenying Deng

1. Introduction

2. Method

1. CVEK

1.1. main algorithm

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

$$H_0 : h \in \mathcal{H}_0. \tag{1.1.1}$$

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* $k_\delta(\mathbf{z}, \mathbf{z}')$, 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:

$$H_0 : \delta = 0. \tag{1.1.2}$$

We now construct a test statistic $\hat{T}_0$ for (1.1.2) 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:

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

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:

$$\hat{T}_0 = \hat{\tau} * (\mathbf{y} - \hat{\boldsymbol{\mu}})^T \mathbf{V}_0^{-1} [\partial \mathbf{K}_0] \mathbf{V}_0^{-1} (\mathbf{y} - \hat{\boldsymbol{\mu}}) \tag{1.1.3}$$

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 $\frac{\partial}{\partial \delta} k_\delta(\mathbf{x}, \mathbf{x}') \mid_{\delta=0}$. As discussed previously, misspecifying the null kernel function k_0 negatively impacts the performance of the resulting hypothesis test. To better understand the mechanism at play, we express the test statistic $\hat{T}_0$ from (1.1.3) in terms of the model residual $\hat{\boldsymbol{\epsilon}} = \mathbf{y} - \hat{\boldsymbol{\mu}} - \hat{\mathbf{h}}$:

$$\hat{T}_0 = (\frac{\hat{\tau}}{\hat{\sigma}^4}) * \hat{\boldsymbol{\epsilon}}^T [\partial \mathbf{K}_0] \hat{\boldsymbol{\epsilon}}, \tag{1.1.4}$$

Algorithm 1 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 $p$
    #Step 1:  Estimate Null Model using REML
    $(\hat{\boldsymbol{\mu}}, \hat{\tau}, \hat{\sigma}^2) = argminL_{REML}(\boldsymbol{\mu}, \tau, \sigma^2|\mathbf{K}_0)$
    #Step 2:  Compute Test Statistic and Null Distribution Parameters
    $\hat{T}_0 = \hat{\tau} * (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^T \mathbf{V}_0^{-1} \partial\mathbf{K}_0 \mathbf{V}_0^{-1} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})$
    $\hat{\kappa} = \hat{\mathbf{I}}_{\delta\delta}/[\hat{\tau} * tr(\mathbf{V}_0^{-1}\partial\mathbf{K}_0)], \quad \hat{\nu} = [\hat{\tau} * tr(\mathbf{V}_0^{-1}\partial\mathbf{K}_0)]^2/(2 * \hat{\mathbf{I}}_{\delta\delta})$
    #Step 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
```

where we have used the fact $\mathbf{V}_0^{-1}(\mathbf{y} - \hat{\boldsymbol{\mu}}) = (\hat{\sigma}^2)^{-1}(\hat{\boldsymbol{\epsilon}})$. As shown, the test statistic $\hat{T}_0$ is a scaled quadratic-form statistic that is a function of the model residual. If k_0 is too restrictive, model estimates will **underfit** the data even under the null hypothesis, introducing extraneous correlation among the $\hat{\epsilon}_i$'s, therefore leading to overestimated $\hat{T}_0$ and eventually underestimated p-valueunder under the null. In this case, the test procedure will frequently reject the null hypothesis (i.e. suggest the existence of nonlinear interaction) even when there is in fact no interaction, yielding an invalid test due to **inflated Type I error**. On the other hand, if k_0 is too flexible, model estimates will likely **overfit** the data in small samples, producing underestimated residuals, an underestimated test statistic, and overestimated p-values. In this case, the test procedure will too frequently fail to reject the null hypothesis (i.e. suggesting there is no interaction) when there is in fact interaction, yielding an insensitive test with **diminished power**.

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:

$$\kappa * \nu = E(T), \quad 2 * \kappa^2 * \nu = Var(T)$$

with solution:

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

where $\hat{\mathbf{I}}_{\delta\theta}$ and $\hat{\mathbf{I}}_{\delta\theta}$ are the 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 1.

In light of the discussion about model misspecification in Introduction section, we highlight the fact that our proposed test (1.1.3) is robust against model misspecification under the alternative, since the calculation of test statistics do not require detailed parametric assumption about k_δ. However, the test is NOT robust against model misspecification under the null, since the expression of both test statistic $\hat{T}_0$ and the null distribution parameters $(\hat{\kappa}, \hat{\nu})$ still involve the kernel matrices generated by k_0 (see Algorithm 1). We address this problem by proposing a robust estimation procedure for the kernel matrices under the null.

1.2. 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

$$\mathbf{P}_\lambda = \mathbf{K}(\mathbf{X}, \mathbf{X})[\mathbf{K}(\mathbf{X}, \mathbf{X}) + \lambda \mathbf{I}]^{-1} \tag{1.2.1}$$

In this way, $Trace(\mathbf{P}_\lambda)$ is the effective number of model parameters, excluding β_0 and σ^2. It decreases monotonically with $\lambda > 0$.

cross-validation: K fold and loo[1]

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 $\boldsymbol{\beta}_\lambda$ K times, each time leaving out group $\kappa(i)$, to get $\boldsymbol{\beta}_\lambda^{-\kappa(1)}, \boldsymbol{\beta}_\lambda^{-\kappa(2)}$, etc. For $\boldsymbol{\beta}_\lambda^{-\kappa(i)}$, cross-validated residuals are calculated on the observations in $\kappa(i)$, which did not contribute to estimating $\boldsymbol{\beta}$. The objective function estimated prediction error and is the sum of the squared cross-validated residuals:

$$\lambda_{K-CV} = \underset{\lambda \in \Lambda}{argmin} \ ln \sum_{i=1}^{K} (\mathbf{y}_{\kappa(i)} - \Phi(\mathbf{x}_{\kappa(i)})\boldsymbol{\beta}_\lambda^{-\kappa(i)})^T (\mathbf{y}_{\kappa(i)} - \Phi(\mathbf{x}_{\kappa(i)})\boldsymbol{\beta}_\lambda^{-\kappa(i)}) \tag{1.2.2}$$

LooCV is the situation when $K = n$. In this case, we can write our objective function as:

$$\lambda_{n-CV} = \underset{\lambda \in \Delta}{argmin} \ ln \sum_{i=1}^{n} \frac{(Y_i - \phi(X_i)^T \boldsymbol{\beta}_\lambda)^2}{(1 - P_{\lambda[ii]} - \frac{1}{n})^2} \tag{1.2.3}$$

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 or 10, cross-validation will have lower variance, but making bias a problem.

AIC and small sample correction

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

$$\begin{aligned} AIC =& 2(p+2) - 2ln(\hat{L}) \\ =& 2(p+2) - 2[-\frac{n}{2}ln(\hat{\sigma}^2) - \frac{n}{2}ln(2\pi) - \frac{1}{2\hat{\sigma}^2}\mathbf{y}^T(\mathbf{I}_n - \mathbf{P}_\lambda)^2\mathbf{y}] \\ =& 2(p+2) + nln[\frac{1}{n}\mathbf{y}^T(\mathbf{I}_n - \mathbf{P}_\lambda)^2\mathbf{y}] + n + nln(2\pi) \end{aligned}$$

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

$$\lambda_{AIC} = \underset{\lambda \in \Lambda}{argmin}\{ln \ \mathbf{y}^T(\mathbf{I}_n - \mathbf{P}_\lambda)^2\mathbf{y} + \frac{2(Trace(\mathbf{P}_\lambda) + 2))}{n}\} \tag{1.2.4}$$

When n is small, extreme overfitting is possible, giving small bias/ large variance estimates. The small-sample correction of AIC is derived by minimizing minus 2 times expected log likelihood, where we plug in $\mathbf{P}_\lambda$ and $\hat{\sigma}^2$. In this case, we obtain our small-sample size objective function AICc:

$$\lambda_{AICc} = \underset{\lambda \in \Lambda}{\operatorname{argmin}}\{\ln \mathbf{y}^T(\mathbf{I}_n - \mathbf{P}_\lambda)^2\mathbf{y} + \frac{2(Trace(\mathbf{P}_\lambda)+2))}{n - Trace(\mathbf{P}_\lambda) - 3}\} \tag{1.2.5}$$

Compare (1.2.4) and (1.2.5), it is easy to tell that AICc considers more of the model complexity since $\frac{n}{n-Trace(\mathbf{P}_\lambda)-3} > 1$. It makes sense intuitively because it need to shrink more to prevent small bias/ large variance estimates.

GCV and small sample correction

In (1.2.3), if we approximate each $P_{\lambda[ii]}$ with their mean $\frac{Trace(\mathbf{P}_\lambda)}{n}$, in a sense that we give equal weight to all observations. We get our GCV objective function:

$$\lambda_{GCV} = \underset{\lambda \in \Lambda}{\operatorname{argmin}}\{\ln \mathbf{y}^T(\mathbf{I}_n - \mathbf{P}_\lambda)^2\mathbf{y} - 2\ln(1 - \frac{Trace(\mathbf{P}_\lambda)}{n} - \frac{1}{n})\} \tag{1.2.6}$$

The "$-\frac{1}{n}$" terms in (1.2.6) is because GCV counts β_0 as part of model complexity, but not σ^2. This motivates the proposed small-sample correction to GCV, which does count σ^2 as a parameter:

$$\lambda_{GCVc} = \underset{\lambda \in \Lambda}{\operatorname{argmin}}\{\ln \mathbf{y}^T(\mathbf{I}_n - \mathbf{P}_\lambda)^2\mathbf{y} - 2\ln(1 - \frac{Trace(\mathbf{P}_\lambda)}{n} - \frac{2}{n})_+\} \tag{1.2.7}$$

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

GMPML-based selection

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 $\boldsymbol{\beta}$:

$$p_\lambda(\boldsymbol{\beta}, \sigma^2) = \frac{\lambda}{2\sigma^2}\boldsymbol{\beta}^T\boldsymbol{\beta} - \frac{p}{2}\ln(\lambda) + \frac{p}{2}\ln(\sigma^2)$$

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

$$\begin{aligned} m(\lambda, \sigma^2) =& \ln \int_{\boldsymbol{\beta}} \exp\{l(\boldsymbol{\beta}, \sigma^2) - p_\lambda(\boldsymbol{\beta}, \sigma^2)\}d\boldsymbol{\beta} \\ =& -\frac{1}{2\sigma^2}\mathbf{y}^T(\mathbf{I}_n - \mathbf{P}_\lambda)\mathbf{y} - \frac{n}{2}\ln(\sigma^2) + \frac{1}{2}\ln \mid \mathbf{I}_n - \mathbf{P}_\lambda \mid \end{aligned}$$

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

$$\lambda_{MPML} = \underset{\lambda \in \Lambda}{\operatorname{argmin}}\{\ln \mathbf{y}^T(\mathbf{I}_n - \mathbf{P}_\lambda)\mathbf{y} - \frac{1}{n}\ln \mid \mathbf{I}_n - \mathbf{P}_\lambda \mid\}$$

Generalized MPML adjusts the penalty to account for estimation of regression parameter β_0 that is not marginalized, resulting in one degree of freedom:

$$\lambda_{GMPML} = \underset{\lambda \in \Lambda}{\operatorname{argmin}}\{\ln \mathbf{y}^T(\mathbf{I}_n - \mathbf{P}_\lambda)\mathbf{y} - \frac{1}{n-1}\ln \mid \mathbf{I}_n - \mathbf{P}_\lambda \mid\} \tag{1.2.8}$$

Discussion
????

$$\mid \mathbf{I}_n - \mathbf{P}_\lambda \mid^{\frac{1}{n}}, \quad (1 - \frac{\mathrm{Trace}(\mathbf{P}_\lambda)}{n} - \frac{1}{n})^2$$

1.3. ensemble strategy

Observation in (1.1.4) motivates the need for a kernel estimation strategy that is *flexible* so that it does not underfit under the null, yet *stable* so that it does not overfit under the alternative. To this end, we propose estimating h using the ensemble of a library of fixed base kernels $\{k_d\}_{d=1}^D$:

$$\hat{h}(\mathbf{x}) = \sum_{d=1}^D u_d \hat{h}_d(\mathbf{x}), \quad \mathbf{u} \in \Delta = \{\mathbf{u}|\mathbf{u} \geqslant 0, \| \mathbf{u} \|_1 = 1\} \tag{1.3.1}$$

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 $\{k_d\}_{d=1}^D$, we first estimate $\hat{\mathbf{h}}_d = \mathbf{K}_d(\mathbf{K}_d + \hat{\lambda}_d \mathbf{I})^{-1}\mathbf{y}$, the prediction based on d^{th} kernel, where the tuning parameter $\hat{\lambda}_d$ is selected by minimizing one of the four objective functions given in section 1.2. Denote the estimated error for d^{th} kernel as $\hat{\epsilon}_d$ and $\mathbf{A}_{d,\lambda} = \mathbf{K}_d(\mathbf{K}_d + \lambda\mathbf{I})^{-1}$.

cross-validation
???
After obtaining the estimated errors $\{\hat{\epsilon}_d\}_{d=1}^D$, we estimate the ensemble weights $\mathbf{u} = \{u_d\}_{d=1}^D$ such that it minimizes the overall error:

$$\hat{\mathbf{u}} = \underset{\mathbf{u}\in\Delta}{argmin} \| \sum_{d=1}^D u_d \hat{\epsilon}_d \|^2 \quad where \Delta = \{\mathbf{u}|\mathbf{u} \geqslant 0, \| \mathbf{u} \|_1 = 1\}$$

Then produce the final ensemble prediction:

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

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

simple averaging
Another way to obtain the ensemble matrix would be simply choose $u_d = 1_D$ for $d = 1, 2, ...D$.
????

Now that we have the ensemble matrix $\hat{\mathbf{A}}$, we can estimate the ensemble kernel matrix $\hat{\mathbf{K}}$ by solving:

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

Specifically, if we denote $\mathbf{U}_A$ and $\{\delta_{A,k}\}_{k=1}^n$ the eigenvector and eigenvalues of $\hat{\mathbf{A}}$, then $\hat{\mathbf{K}}$ adopts the form:

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

1.4. kernel choice

types of kernel

characterization as a function class

spectral property

2. Hypothesis Test

2.1. Asymptotic Test

2.2. Bootstrap Test

3. Simulation

4. Conclusion