LinVer Manual

July 14, 2016

LinVer is a reference Matlab implementation of a verification framework for Bayesian inference algorithms outlined in [1]. It is based on a linear regression problem for which analytical or semi-analytical solutions are known. It provides a rigorous means of testing output chains of Markov chain Monte Carlo (MCMC) algorithms used for Bayesian inference are distributed correctly via an implementation of a hypothesis test for equal distributions based on the energy distance statistic [2]. While the main goal of the code is as a reference for those interested in implementing the framework in other verification software, it is also useable as-is as a basic verification tool. Mathematical details of the framework can be found in Appendix A of [1] and will be described further in a forthcoming paper.

Be aware that LinVer is in development and may contain some bugs. The implementation of the energy statistic test is not completely verified. The calculation of the true posteriors is believed to be correct, however, for the basic cases.

Chapter 1

Parameters of the Linear Equation

The verification framework is based on Bayesian inference for the linear equation defined by

$$y = G\beta + \varepsilon(\lambda, \phi).$$

The variables making up this equation are:

y :	A vector of N observations.
G :	The $N \times N_\beta$ design matrix.
β :	The $N_\beta \times 1$ column vector $\begin{bmatrix}\beta_1 & \cdots & \beta_{N_\beta}\end{bmatrix}^T$ of regression parameters.
ε :	The $N \times 1$ column vector of observation noise.
λ :	The precision parameter for the noise.
ϕ :	The correlation parameter for the noise covariance.

In this implementation of the framework, the first column of the design matrix G is all one's, so the first regression parameter β_1 is a bias term. The remaining entries of G (if any) are drawn from a standard normal distribution (i.e., mean 0 and variance 1). This choice of G is not crucial to the framework as-implemented and can be substituted.

The observation noise is zero-mean and normally distributed so that $\varepsilon \sim N(0, C)$ with the $N \times N$ covariance matrix $C(\lambda, \phi)$ depending on the parameters λ, ϕ. The λ, ϕ dependence is expressed by

$$C(\lambda, \phi) = \frac{1}{\lambda} R(\phi),$$

where $R(\phi)$ is the correlation function which depends on ϕ. The framework includes three possible choices of correlation:

$$\text{No correlation: } R(\phi) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$\text{Equicorrelation: } R(\phi) = \begin{bmatrix} 1 & \phi & \phi & \cdots & \phi & \phi \\ \phi & 1 & \phi & \cdots & \phi & \phi \\ \phi & \phi & 1 & \cdots & \phi & \phi \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi & \phi & \phi & \cdots & 1 & \phi \\ \phi & \phi & \phi & \cdots & \phi & 1 \end{bmatrix}$$

$$\text{AR(1) correlation: } R(\phi) = \begin{bmatrix} 1 & \phi & \phi^2 & \cdots & \phi^{N-2} & \phi^{N-1} \\ \phi & 1 & \phi & \cdots & \phi^{N-3} & \phi^{N-2} \\ \phi^2 & \phi & 1 & \cdots & \phi^{N-4} & \phi^{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi^{N-2} & \phi^{N-3} & \phi^{N-4} & \cdots & 1 & \phi \\ \phi^{N-1} & \phi^{N-2} & \phi^{N-3} & \cdots & \phi & 1 \end{bmatrix}$$

For each of the correlation cases, the correlation parameter ϕ is restricted to a different domain:

$$\begin{aligned} \text{No correlation: } & \phi \in \emptyset \\ \text{Equicorrelation: } & \phi \in [0, 1) \\ \text{AR(1) correlation: } & \phi \in (-1, 1) \end{aligned}$$

Note that we have defined the domain for ϕ in the no correlation case to be the empty set since the corresponding correlation matrix in this case has no ϕ dependence. The reason for using these choices of covariance functions is that these choices each have known analytical forms for the inverse and determinant. This facilitates computation of the exact solution to the Bayesian inverse problems described below, as the inverse and determinant of the covariance matrices appear in the analytical and semi-analytical solutions for the parameter distributions.

Chapter 2

Bayesian Inference

The purpose of the framework is to provide a means of verifying Bayesian inference algorithms using the equations and parameters in Chapter 1. The framework has three different cases of unknown parameters depending on which of the parameters β, λ, and ϕ are treated as unknown. The cases used in the framework are summarized in the following table.

Case	Known	Unknown	Calibration Parameters
1	λ, ϕ	β	$\hat{\beta}$
2	ϕ	β, λ	$\hat{\beta}, \hat{\lambda}$
3	None	β, λ, ϕ	$\hat{\beta}, \hat{\lambda}, \hat{\phi}$

Bayesian inference relies on Bayes rule

$$p\left(\hat{\theta}\middle| y\right) = \frac{p\left(y\middle|\hat{\theta}\right) p_0\left(\hat{\theta}\right)}{p\left(y\right)},$$

where $\hat{\theta}$ is the inferred parameter, y is the data, and the $p(\cdot)$ functions refer to the following probability density functions:

$$\begin{aligned}
p\left(\hat{\theta}\middle| y\right) &: \text{ Posterior of parameter value } \hat{\theta} \text{ given data } y \\
p\left(y\middle|\hat{\theta}\right) &: \text{ Likelihood of data } y \text{ given parameter value } \hat{\theta} \\
p_0\left(\hat{\theta}\right) &: \text{ Prior for inferred parameter value } \hat{\theta} \\
p\left(y\right) &: \text{ Marginal likelihood of the data } y \text{ given by } \int_{D(\hat{\theta})} p\left(y\middle|\hat{\theta}\right) p_0\left(\hat{\theta}\right)\, d\hat{\theta}
\end{aligned}$$

As can be determined from the above table and Bayes rule above, the verification framework computes the following marginal posteriors according to which set of parameters are chosen to be calibrated.

$$\begin{aligned}
&\textbf{Case 1}: p\left(\hat{\beta}\middle| y\right) \\
&\textbf{Case 2}: p\left(\hat{\beta}\middle| y\right),\ \ p\left(\hat{\lambda}\middle| y\right) \\
&\textbf{Case 3}: p\left(\hat{\beta}\middle| y\right),\ \ p\left(\hat{\lambda}\middle| y\right),\ \ p\left(\hat{\phi}\middle| y\right)
\end{aligned}$$

Case 1 and Case 2 are computed exactly, in the sense that an exact parametric distribution for each of the marginal posteriors can be derived which can be computed using standard routines for calculating the associated special functions. The posterior for Case 3 is an integral expression involving special functions and is computed using numerical integration. Note that Case 3 requires the use of the equicorrelation or AR(1) correlation function, since there is no ϕ dependence for uncorrelated observation error.

For each of the three cases of unknown parameters, there are two choices of prior. These are labeled the "Non-informative" and "Gaussian" priors, according to whether the marginal prior for $\hat{\beta}$ is uniform or Gaussian. The following describes the priors for each of the calibrated variables.

Non-informative

$$\begin{aligned}
\textbf{Case 1}: \; & p_0\left(\hat{\beta}\right) \propto 1 \\
\textbf{Case 2}: \; & p_0\left(\hat{\beta}\middle|\hat{\lambda}\right) \propto 1 \\
& p_0\left(\hat{\lambda}\right) \propto \frac{1}{\hat{\lambda}} \\
\textbf{Case 3}: \; & p_0\left(\hat{\beta}\middle|\hat{\lambda},\hat{\phi}\right) \propto 1 \\
& p_0\left(\hat{\lambda}\middle|\hat{\phi}\right) \propto \frac{1}{\hat{\lambda}} \\
& p_0\left(\hat{\phi}\right) = \frac{1}{\phi_H - \phi_L}
\end{aligned}$$

The priors which are described as a proportional expression are improper priors. The parameters ϕ_H and ϕ_L are the high and low values of the domain of ϕ, so that $\phi \in [\phi_L, \phi_H]$. These are configurable parameters in the framework and will depend on the choice of correlation function. These are the only parameters needed for the non-informative prior, and only when ϕ is an unknown parameter (i.e., only in Case 3).

Gaussian

$$\begin{aligned}
\textbf{Case 1}: \; & p_0\left(\hat{\beta}\right) \text{ is } N\left(\mu_0, \frac{1}{\lambda}\Sigma_0\right) \\
\textbf{Case 2}: \; & p_0\left(\hat{\beta}\middle|\hat{\lambda}\right) \text{ is } N\left(\mu_0, \frac{1}{\hat{\lambda}}\Sigma_0\right) \\
& p_0\left(\hat{\lambda}\right) \propto \frac{1}{\hat{\lambda}} \\
\textbf{Case 3}: \; & p_0\left(\hat{\beta}\middle|\hat{\lambda},\hat{\phi}\right) \text{ is } N\left(\mu_0, \frac{1}{\hat{\lambda}}\Sigma_0\right) \\
& p_0\left(\hat{\lambda}\middle|\hat{\phi}\right) \propto \frac{1}{\hat{\lambda}} \\
& p_0\left(\hat{\phi}\right) = \frac{1}{\phi_H - \phi_L}
\end{aligned}$$

The only difference between the non-informative and Gaussian cases is the prior on $\hat{\beta}$ is Gaussian with a prescribed $N_\beta \times 1$ mean vector μ_0 and unscaled $N_\beta \times N_\beta$ covariance Σ_0. The framework uses diagonal Σ_0, although this is not strictly necessary. Make note that the prior distribution in Case 1 has a covariance matrix scaled by $1/\lambda$, the known value of λ. In the other cases λ is unknown so this scaling is replaced by $1/\hat{\lambda}$, the calibrated variable.

Chapter 3

Using the Verification Framework

The following is an outline of the steps involved in using the verification framework.

1. Choose the size of the problem: N, the number of observations and N_β, the number of regression parameters.
2. Choose the values of the true parameters β, λ, ϕ. These are the values that the Bayesian inference algorithm will attempt to learn from the data.
3. Set the design matrix G. The default recommendation (and what is used in examples) is to set all entries in the first column to ones and fill the remaining entries (if any) with data drawn from a standard normal distribution.
4. Use the true parameter β to determine the error-free model output y_0 according to $y_0 = G\beta$.
5. Choose a correlation function $R(\phi)$: no correlation, equicorrelation, or AR(1) correlation.
6. Use the chosen correlation function and the true parameters λ, ϕ to generate the observation error ε according to the distribution $N(0, C)$ where $C = R(\phi)/\lambda$.
7. Add the observation error ε to the error-free model output y_0 to obtain the calibration data $y = y_0 + \varepsilon$.
8. Choose which parameters to calibrate (i.e., which are treated as unknowns): Case 1: $\hat{\beta}$, Case 2: $\hat{\beta}, \hat{\lambda}$, or Case 3: $\hat{\beta}, \hat{\lambda}, \hat{\phi}$. If using Case 3, choose the parameters ϕ_L, ϕ_H restricting the domain of ϕ. These need to be chosen in agreement with the domain associated with the correlation function.
9. Choose whether to use the non-informative prior or the Gaussian prior for the regression parameters $\hat{\beta}$. If using the Gaussian prior, choose the prior mean μ_0 and prior un-scaled covariance Σ_0.
10. Using the calibration routine to be verified, sample from the posterior of the unknown parameters. It is important that this be done using the calibration data y generated above, using the same design matrix G set above.
11. Use the verification framework to calculate the true posterior using all the chosen parameters as input.
12. Compare the true posterior computed by the verification framework with that sampled by the routine being verified.

This outline leaves open the specific ways the comparison between the true posterior and the output of the calibration routine are to be done. There are multiple ways to do this, some being more appropriate for different types of calibration routines. We specifically mention the case of verifying Markov chain Monte Carlo (MCMC) output, as we have included a method aimed at this common case. Here we recommend a statistical hypothesis test based on the energy statistic, and the Matlab implementation includes an implementation of this.

Chapter 4

Using the Matlab Implementation

The Matlab scripts `demo_case1.m, demo_case2.m, demo_case3.m` demonstrate the use of the Matlab implementation for Case 1 (β unknown), Case 2 (β, λ unknown), and Case 3 (β, λ, ϕ unknown) respectively.

The main function used in computing the posterior is `eval_posterior`. This function takes a structure `param` as input which defines all the parameters of the problem. The fields of this structure are as follows.

`param.N`:	Number of observations.
`param.Nbeta`:	Number of regression parameters.
`param.G`:	`N` by `Nbeta` design matrix.
`param.prior`:	Structure defining the prior. See below for more information.
`param.beta`:	`Nbeta` by 1 vector of the true regression parameters β
`param.lambda`:	Scalar value of the true precision parameter λ.
`param.phi`:	Scalar value of the true correlation parameter ϕ.
`param.corrfunc`:	String defining the correlation function type: `"none", "equal", "ar"`.
`param.y`:	`N` by 1 vector of observations.
`param.unknowns`:	String defining which parameters are unknown. See below.
`param.betarange`:	`Nbeta` by 2 matrix. Left column is lower limit of β, right column is upper limit.
`param.lambdarange`:	1 by 2 matrix. Left value is lower limit of λ, right column is upper limit.
`param.phirange`:	1 by 2 matrix. Left value is lower limit of ϕ, right column is upper limit.

Prior structure. Let the prior structure be named `prior`. It always has the field `prior.type` which is a string variable with two possible values: `"noninformative"` or `"gaussian"`. If the `prior.type` field is set to `"noninformative"`, no other fields need to be defined. If it is set to `"gaussian"`, the field `prior.sigma0` must be the `Nbeta` by `Nbeta` unscaled covariance and `prior.mu0` is the `Nbeta` by 1 mean vector of the prior.

Unknowns strings. The `prior.unknowns` field can take one of the string values: `"beta"`, `"beta_lambda"`, `"beta_lambda_phi"`. These correspond to whether the unknowns are Case 1: β, Case 2: β, λ, or Case 3: β, λ, ϕ.

Calibration data. The function `eval_noise` takes the same `param` structure above as input, except `param.y` need not be defined. It returns a `N` by 1 vector of unscaled observation errors. This vector can be divided by $\sqrt{\lambda}$ and added to the error-free model output to provide calibration data.

Range parameters. The `param.betarange`, `param.lambdarange`, and `param.phirange` fields are defined for practical purposes to bound the range of the parameters. In the ideal problem, each component of β can be any component in $\mathbb{R}$ and λ ranges from $(0, \infty)$. The ranges for these parameters are used only by the Metropolis-Hastings algorithm provided as an example routine for verification and should not affect the true posterior calculation. The range for ϕ, however, is different, as this defines the range of the uniform prior. It is necessary to choose this prior so that the range is smaller than the range of the actual correlation parameter ($[0, 1)$ for equicorrelation and $(-1, 1)$ for AR(1) correlation) as the posterior for ϕ involves integrals which do not converge if every point in the domain of ϕ is weighted by a uniform prior.

Posterior structure. The function `eval_posterior` returns a structure which we call `post`. The fields of this structure depend on which parameters are unknown. The posterior structure always includes a field defining a function for the marginal posterior density of each unknown. These are given by `post.pbeta`, `post.plambda`, and `post.pphi`. These functions are used by passing a vector argument of values to evaluate the posterior at. For example, the code

```
beta = linspace(-1,1)'; y = post.pbeta(beta);
```

evaluates the posterior for a scalar-valued β at 100 evenly spaced points between -1 and 1 and stores the result in the variable `y`. If $N_\beta > 1$ then the argument should be an $M \times N_\beta$ vector, where M is the number of points in parameter space to evaluate the posterior of β at. The functions for the posterior of λ and ϕ work similarly, except there is no need to consider a vector-valued case since these are always scalar-valued random variables.

In addition to these functions, in the cases where the posteriors are known parametric distributions, relevant parameters are also returned. In Case 1, β is unknown and its posterior is Gaussian with mean μ_2 and covariance Σ_2/λ (where λ is the known scale parameter). These are returned in the posterior as `post.mu2` and `post.sigma2`. In Case 2, β and λ are both unknown. The posterior of λ is a Gamma distribution with shape parameter a_1 and scale parameter b_1. These are returned as `post.a1` and `post.b1`. Note that there are multiple conventions for the parameterization of the Gamma distribution. The convention used here is compatible with that used in Matlab if `1 / post.b1` is used for the scale parameter. The posterior for β is a multivariate t-distribution with a scale matrix, location vector, and degrees-of-freedom parameter. These are returned as `post.scl`, `post.loc`, and `post.dof`, respectively. The marginals for Case 3 are not known parametric distributions, so we will not discuss the other fields defined.

Energy statistic test for equal distributions. An implementation of a non-parametric test of distribution equality for two samples is included in the Matlab code. This provides means of comparing a random sample drawn from the posterior with a given sample computed from a Markov chain Monte Carlo code, for example. The function is

```
results = do_energy_test(sample, param, post, alpha, numtests, N)
```

where the arguments are:

`sample`: Sample to compare to the true posterior. This should be drawn from the code being verified. It should be a $M \times N_{param}$ matrix of M observations of the N_{param}-dimensional parameter vector. That is, each row is an observation of the parameter vector, whose size N_{param} depends on which parameters are unknown. For Case 1 $N_{param} = N_\beta$ and each observation is the regression parameters. For Case 2 $N_{param} = N_\beta + 1$ and each observation is the N_β regression parameter observations followed by the λ observation. For Case 3 $N_{param} = N_\beta + 2$ and each observation is the N_β regression parameters, followed by the λ observation, followed by the ϕ observation.

`param`: This is the parameter structure described above.

`post`: This is the posterior structure returned by `eval_posterior`.

`alpha`: This is the confidence level of the test, between 0 and 1. Lower values represent stricter confidences, but will lead to more computationally involved tests. Defaults to 0.05.

`numtests`: This is the number of hypothesis tests to run. Each test draws a separate set of realizations from the posterior to use in the hypothesis test which checks if `sample` is drawn from the same distribution. If the number of failed tests significantly exceeds `numtests * alpha`, then it is reasonable to expect that `sample` comes from a different distribution and the code that generated it failed to produce the correct output at the given signficance level. Defaults to 100.

`N`: Number of samples to draw from the true posterior to compare with `sample`. Defaults to 100.

If the arguments after `post` are omitted, the defaults described above will be used. The `results` structure has the following fields.

`results.exactsamp`: This is a `N * numtests` by `Nparam` matrix containing the samples drawn from the true posterior for each test.

`results.fail_ratio`: This is the ratio of tests run which were rejected. If this value exceeds `alpha` significantly, the correctness of `sample` is questionable.

`results.fail_pvalue`: This is the probability of observing at least as many rejections as occurred given the number of tests and the confidence level `alpha`.

Sampling from the true posterior. The function `draw_posterior_sample(param, post, N)` can be used to draw N samples from the posterior. The arguments are similar to those described above.

Efficient correlation matrix operations. The structure of the correlation functions used in the framework can be exploited to derive efficient recursive algorithms for matrix-vector multiplication, matrix inversion, and multiplication by the Cholesky factors of these matrices. These algorithms can make much larger data sizes feasible. While these algorithms are not used in the main functions of the Matlab implementation of the framework, reference code is provided in `fastchol_ar1.m`, `fastchol_eq.m`, `fastmv_ar1.m`, `fastmv_eq.m`, `fastmv_ar1inv.m`, and `fastmv_eqinv.m`. See the comments in those files for more information.

Alternative priors for ϕ. The only prior implemented for ϕ is the uniform prior above. This is computed using Gaussian quadrature rules which are implemented in the file `gauss_quadrature.m`. That file contains additional quadrature rules for placing a Gaussian-distribution prior on ϕ and a Beta-distribution prior. While that code has been tested and appears to correctly implement the quadratures associated with those priors, the option to uses those priors is not currently included in the framework. This could be changed, if desired, by making appropriate changes to the function `eval_beta_lambda_phi_params` in `eval_posterior.m`.

Other information. The above provides a basic outline of the verification framework. Additional information on other optional features can be found by reading the code and its comments.

Bibliography

[1] Brian M Adams, Laura Painton Swiler, Russell Hooper, Allison Lewis, Jerry A McMahan, Ralph C Smith, and Brian Williams. User guidelines and best practices for CASL VUQ analysis using DAKOTA. Technical report, Sandia National Laboratories, Albuquerque, New Mexico, 2014.

[2] Gábor J Székely and Maria L Rizzo. Testing for equal distributions in high dimension. *InterStat*, 5:1–6, 2004.