Projection predictive variable selection using Stan+R

Juho Piironen[*] and Aki Vehtari

June 24, 2015

Abstract

This document is additional material to our previous study comparing several strategies for variable subset selection (Piironen and Vehtari, 2015). Our recommended approach was to fit the full model with all the candidate variables and best possible prior information, and perform the variable selection using the projection predictive framework. Here we give an example of performing such an analysis, using Stan for fitting the model, and R for the variable selection.

1 Introduction

Identifying relevant explanatory variables is often of interest in applied statistical analysis. In this context, the relevance of a variable is usually determined by its predictive power on the target variable of interest. The gain from successful variable selection is typically improved model interpretability, but depending on the problem, also reduced measurement costs and computational savings may be obtained.

In our recect study (Piironen and Vehtari, 2015), we discussed and compared several strategies for performing variable selection in practical problems and concluded that often best results are obtained by the following strategy: fit the full model with all the candidate variables and best possible prior information, and perform the variable selection using the projection predictive method (Goutis and Robert, 1998; Dupuis and Robert, 2003). Our comparison showed that this approach generally outperforms the selection of the most probable variables or variable combination, methods which are often used for performing Bayesian variable selection.

This document serves as additional material to our study, the purpose being to provide an example of how to carry out the model fitting with Stan and the subsequent variable selection using R. Sampling the full model under the popular spike-and-slab prior (Mitchell and Beauchamp, 1988) is infeasible with Stan, but the prior assumptions about the sparsity can be conveniently formulated using the hierarchical shrinkage priors, such as the horseshoe (Carvalho et al., 2009, 2010), which allow convenient computation using Stan. All the relevant codes are provided in the Appendix.

The document is organized as follows. Section 2 reviews the hierarchical shrinkage in the context of linear regression using half-Student-t priors for the weight scales, and

[*] first.last@aalto.fi

discusses the horseshoe prior as a special case. Section 3 shortly reviews the projection predictive variable selection and discusses the computations in the example case of a linear Gaussian model. Finally, in Section 4 we provide an illustrative numerical example.

2 Hierarchical shrinkage

Consider the single output linear Gaussian regression model with several input variables, given by

$$\begin{aligned} f_i &= \mathbf{w}^\mathsf{T}\mathbf{x}_i \\ y_i &= f_i + \varepsilon_i, \quad \varepsilon \sim \mathrm{N}\big(0, \sigma^2\big), \quad i = 1, \dots, n \end{aligned} \tag{1}$$

where $\mathbf{x}$ is the m-dimensional vector of inputs, $\mathbf{w}$ contains the corresponding weights and σ^2 is the noise variance. A hierarchical shrinkage (HS) prior for the regression weights $\mathbf{w} = (w_1, \dots, w_m)$ can be obtained as

$$\begin{aligned} w_i \,|\, \lambda_i, \tau &\sim \mathrm{N}\big(0, \lambda_i^2\tau^2\big) \\ \lambda_i &\sim t_\nu^+(0, 1)\,. \end{aligned} \tag{2}$$

where t_ν^+ denotes the half-Student-t prior with ν degrees of freedom. We will refer to (2) by the acronym HS-t_ν. The horsehoe prior (Carvalho et al., 2009, 2010) is obtained by setting $\nu = 1$, that is, by introducing half-Cauchy priors for the local scale parameters λ_i. The horseshoe prior has been shown to possess desirable theoretical properties and good performance in practice (Carvalho et al., 2009, 2010; Datta and Ghosh, 2013; van der Pas et al., 2014). Intuitively, we expect the local variance parameters λ_i^2 to be large for those inputs that have high relevance, and small for those with negligible relevance, while the global variance term τ^2 adjusts the overall sparsity level. The shrinkage coefficient $\kappa_i = 1/(1 + \lambda_i^2)$ describes the amount of shrinkage for the weight w_i, so that $\kappa_i = 0$ means no shrinkage (λ_i^2 large) and $\kappa_i = 1$ complete shrinkage (λ_i^2 small).

Figure 1 shows the priors on κ_i implied by different choices of ν. In all the cases, the prior encourages shrinkage ($\kappa_i \approx 1$) which is due to the fact that the density of the half-t prior evaluates to a positive constant near the origin, and thus contains mass near $\lambda_i = 0$. Moreover, the long tails of the Cauchy distribution ($\nu = 1$) allow some of the λ_i to be very large, leaving those weights essentially unshrunk ($\kappa_i = 0$). The reason why the horseshoe yields results closely similar to the spike-and-slab prior (Mitchell and Beauchamp, 1988) is due to this dual nature (complete shrinkage or no-shrinkage). When the value of ν is increased, the prior still allows strong shrinkage but leaves none of the weights completely unshrunk.

Recently, an extended version of the horseshoe prior, called the horseshoe+ was proposed by Bhadra et al. (2015). Consider adding another level of local scale parameters to (2) as

$$\begin{aligned} w_i \,|\, \lambda_i, \tau &\sim \mathrm{N}\big(0, \lambda_i^2\eta_i^2\tau^2\big) \\ \lambda_i &\sim t_\nu^+(0, 1), \\ \eta_i &\sim t_\nu^+(0, 1). \end{aligned} \tag{3}$$

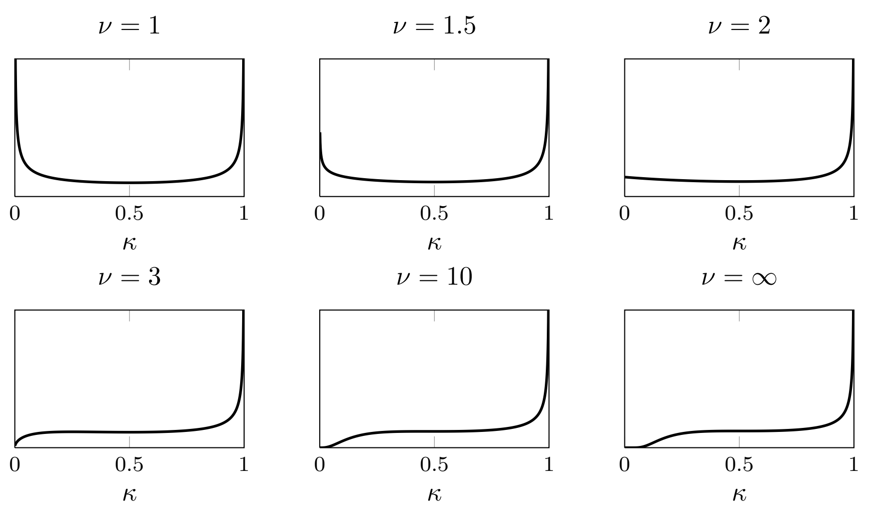

Figure 1: The shrinkage profile of the prior (2), that is, the prior distribution of the shrinkage coefficient $\kappa_i = 1/(1 + \lambda_i^2)$, for different values of ν. The case $\nu = 1$ corresponds to the horseshoe. $\kappa_i = 0$ means no shrinkage and $\kappa_i = 1$ complete shrinkage.

We shall refer to the prior (3) as HS-t_ν+. The horseshoe+ is then obtained by using half-Cauchy distributions, that is, setting $\nu = 1$ (note that the above notation differs slightly from the original paper). The authors argue that the horseshoe+ obtains an improved behaviour compared to the original horseshoe in ultra-sparse problems, both theoretically and empirically.

The hierarchical priors (2) and (3) are straightforward to implement and use in Stan. However, in practice we observe that for the particular case of $\nu = 1$ (horseshoe and horseshoe+) NUTS produces a lot of divergent transitions even after the warm-up period[1]. Experimentally we find that the number of divergent transitions can be reduced by larger value of ν, that is, by shortening the tails of the priors for the local scale terms (see Section 4.1). As depicted in Figure 1, this affects the shrinkage properties of the prior but as will be demonstrated in Section 4.1, the effect regarding the predictions may be negligible.

3 Projection predictive variable selection

This section shortly reviews the idea of the projection predictive method for variable selection. Section 3.1 presents the general idea and Section 3.2 discusses the linear Gaussian regression model as a special case. Our discussion is concise, for more information about the assessment of the method, see our previous paper (Piironen and Vehtari, 2015), and for more on the theoretical considerations and related concepts, see the review by Vehtari and Ojanen (2012).

[1]See this thread for outlining the problem `https://groups.google.com/d/msg/stan-dev/pt1NNytNVUI/4PHO9ekefBYJ`

3.1 General framework

The idea in the projection approach of Goutis and Robert (1998) and Dupuis and Robert (2003) is to simplify the full model M_* by projecting the information in the posterior onto the submodels so that the predictions change as little as possible. Given the parameters of the full model $\boldsymbol{\theta}^*$, the projected parameters $\boldsymbol{\theta}^\perp$ in the parameter space of submodel $M_\perp$ are defined as

$$\boldsymbol{\theta}^\perp = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \mathrm{KL}(p(\tilde{y} \,|\, \mathbf{x}_i, \boldsymbol{\theta}^*, M_*) \,\|\, p(\tilde{y} \,|\, \mathbf{x}_i, \boldsymbol{\theta}, M_\perp)) \,. \tag{4}$$

The discrepancy between the full model and the submodel is then defined to be the expectation of this divergence over the posterior of the full model

$$\delta(M_* \| M_\perp) = \frac{1}{n} \sum_{i=1}^{n} \mathrm{E}_{\boldsymbol{\theta}^* | D, M_*} \Big[\mathrm{KL}\Big(p(\tilde{y} \,|\, \mathbf{x}_i, \boldsymbol{\theta}^*, M_*) \,\|\, p(\tilde{y} \,|\, \mathbf{x}_i, \boldsymbol{\theta}^\perp, M_\perp) \Big) \Big] \,. \tag{5}$$

The posterior expectation in (5) is in general not available analytically. Dupuis and Robert (2003) proposed calculating the discrepancy by drawing samples $\{\boldsymbol{\theta}^*_s\}_{s=1}^S$ from the posterior of the reference model, calculating the projected parameters $\{\boldsymbol{\theta}^\perp_s\}_{s=1}^S$ individually according to (4), and then approximating (5) as

$$\delta(M_* \| M_\perp) \approx \frac{1}{nS} \sum_{i=1}^{n} \sum_{s=1}^{S} \mathrm{KL}\Big(p(\tilde{y} \mid \mathbf{x}_i, \boldsymbol{\theta}^*_s, M_*) \,\|\, p(\tilde{y} \mid \mathbf{x}_i, \boldsymbol{\theta}^\perp_s, M_\perp) \Big) \,. \tag{6}$$

This approximation makes the computations feasible as it requires only ability to draw samples from the full model and a routine for solving the optimization problem (4). In general case the optimization problem can be solved numerically (using e.g. Newton's method) but for the simplest models such as the linear Gaussian case, the minimization can be carried out analytically (see the next section).

In model selection, we seek for submodels $M_\perp$ which have small discrepancy from the full model (6). The final assessment of how much the full model can be simplified, can be performed using cross-validation (see Section 4.2).

3.2 Example: linear Gaussian model

Consider the linear Gaussian model (1). For easier notation, we set the first term of $\mathbf{w}$ to be the intercept w_0 and the first input to be fixed $x_0 = 1$. For this model, the projected parameters (4) can be calculated analytically. Given a sample $(\mathbf{w}, \sigma^2)$ from the posterior of the full model, the projected parameters are given by (see Appendix A)

$$\mathbf{w}_\perp = (\mathbf{X}_\perp{}^\mathsf{T} \mathbf{X}_\perp)^{-1} \mathbf{X}_\perp{}^\mathsf{T} \, \mathbf{X}\mathbf{w} \tag{7}$$

$$\sigma_\perp^2 = \sigma^2 + \frac{1}{n} (\mathbf{X}\mathbf{w} - \mathbf{X}_\perp \mathbf{w}_\perp)^\mathsf{T} (\mathbf{X}\mathbf{w} - \mathbf{X}_\perp \mathbf{w}_\perp), \tag{8}$$

and the associated KL-divergence (for this particular sample) is

$$d(\mathbf{w}, \sigma^2) = \frac{1}{2} \log \frac{\sigma_\perp^2}{\sigma^2} . \tag{9}$$

Here $\mathbf{X} = (\mathbf{x}_1^\mathsf{T}, \dots, \mathbf{x}_n^\mathsf{T})$ denotes the $n \times m$ predictor matrix of the full model, and $\mathbf{X}_\perp$ the contains those columns of $\mathbf{X}$ that correspond to the submodel we are projecting onto. The projection equations (7) and (8) have a nice interpretation. The projected weights are determined by the maximum likelihood solution with the observations $\mathbf{y}$ replaced by the fit of the full model $\mathbf{f} = \mathbf{X}\mathbf{w}$. The projected noise variance is the noise level of the full model plus the mismatch between the fits of the full and the projected model.

As discussed in the previous section, we draw a sample $\{\mathbf{w}_s, \sigma_s^2\}_{s=1}^S$ from the posterior of the full model, compute the projected parameters and associated KL-divergences according to Equations (7), (8) and (9), and then estimate the discrepancy between the full and submodel as

$$\delta(M_* \| M_\perp) = \frac{1}{S} \sum_{s=1}^{S} d(\mathbf{w}_s, \sigma_s^2)\,. \tag{10}$$

This procedure will produce a parsimonious model with exactly zero weights for the variables that are left-out.

4 Numerical example: Crime dataset

This section presents an example of fitting a regression model under the HS-t_ν and HS-t_ν+ priors using Stan (Section 4.1), and demonstrates how to perform the subsequent variable selection using R (Section 4.2). All the codes can be found online[2], and the most relevant parts are also included in the Appendix. We use the Crime dataset[3] used in our earlier study. After removing the features and instances with missing values, the data consist of 1992 instances with $d = 102$ predictor variables. We normalized all the input variables to have zero mean and unit variance, and log normalized the original target variable (total number of crimes per population) to get a real valued and more Gaussian output. For illustrational purposes, we split the data randomly into two so that $n = 1000$ points are used for model training and variable selection, and the remaining $\tilde{n} = 992$ points are used for testing.

4.1 Model fitting

We fit the linear Gaussian regression model (1) with all the $p = 102$ predictors under the HS-t_ν and HS-t_ν+ priors with $\nu = 1$ and $\nu = 3$ (see Section 2). Note that we use these hierarchical priors only for the weights of the nonconstant inputs. For the intercept term we use a weakly informative prior

$$w_0 \sim \mathrm{N}\big(0, 5^2\big).$$

For the global scale parameter τ and for the noise variance σ^2, we use the following uninformative priors

$$\begin{aligned} \tau &\sim \mathrm{C}^+(0,1) \\ \sigma^2 &\propto 1\,. \end{aligned}$$

[2]`https://github.com/jtpi/rstan-varsel`

[3]`https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime`

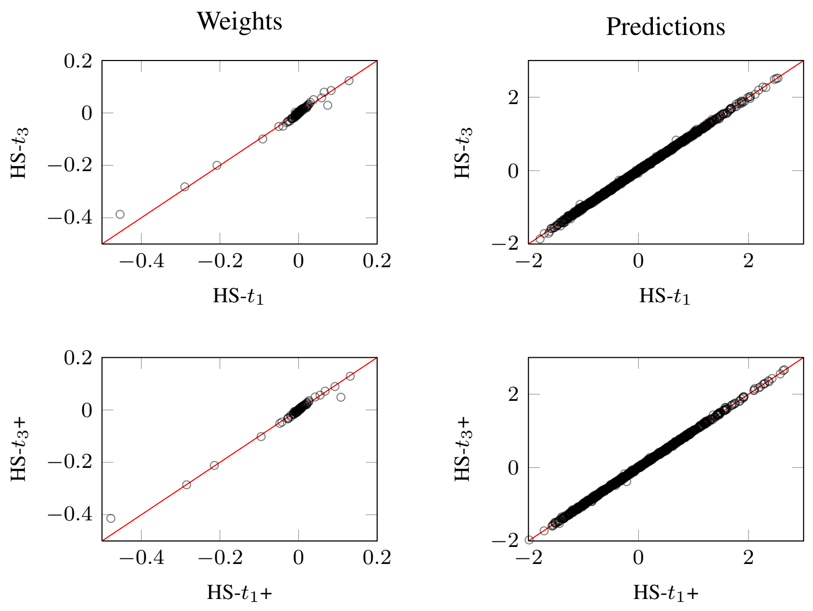

Figure 2: Left column: Posterior means for the weights with different priors, HS-t_1 vs. HS-t_3 (top) and HS-t_1+ vs. HS-t_3+ (bottom). HS-t_1 and HS-t_1+ correspond to horseshoe and horsehoe+ (see Section 2). Right column: the same but for the predictive means on the test set.

The Stan codes for fitting the models are given in Appendix B.

Using the training data, we sample 4 chains each having 1000 samples, and from each chain we remove the first half as a warmup. Figure 2 shows the effect of the degrees of freedom ν in (2) and (3) on the posterior means of the weights and predictions on the test data. In both cases, two of the weights are shrunk more heavily towards zero under $\nu = 3$, while the predictions remain practically the same. The percentage of divergent transitions under the posterior simulations were 3.4% and 5.2% for HS-t_1 and HS-t_1+, whereas only 0.0% and 0.1% for HS-t_3 and HS-t_3+. Given the more robust sampling and negligible effect on the predictive performance, we proceed to the variable selection using $\nu = 3$. However, we emphasize the tentative nature of this experiment and do not recommend this choice to be used uncritically. Thorough understanding of the effects of ν needs more research, but we do not focus on it in this report.

4.2 Variable selection

After fitting the model (previous section) with all the variables we proceed to the variable selection. Here we use the HS-t_3 prior (Section 2) for the full model and the projection predictive variable selection strategy (Section 3.2). As a search heuristic, we use forward searching, that is, starting from the empty model, we add variables one at a time, each time choosing the variable that decreases the KL-divergence (10) the most. As discussed in our study (Piironen and Vehtari, 2015), the effect of number of chosen variables on the predictive ability can be assessed reliably using cross-validation. In

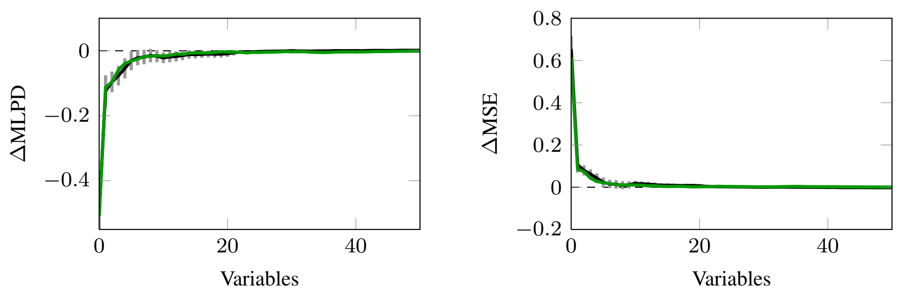

Figure 3: Difference in the mean log predictive density (MLPD) and mean squared error (MSE) between the submodel and the full model as a function of number of chosen variables up to 50 variables. Black is the average over the $K = 10$ cross-validated searches, grey bars denoting the 95% credible interval, and green is the test performance when the search is done using all the training data.

other words, we repeat the model fitting and selection K times each time leaving n/K points out for validation, and evaluate the performance of the full model and the found submodels using these left-out data. The relevant R codes are provided in Appendix C.

Figure 3 shows the difference in the mean log predictive density and mean squared error between the submodel and the full model as a function of number of added variables up to 50 variables. The black line is the average over the $K = 10$ cross-validation folds and the green line shows the result when the fitting and searching is performed using all the training data and the performance is evaluated on the test data. As expected, there is a good correspondence between the cross-validated and test performance. For this dataset, most of the predictive ability of the full model is captured with about 5 variables, and 20 variables are enough for getting predictions indistinguishable from the full model for all practical purposes.

References

Bhadra, A., Datta, J., Polson, N. G., and Willard, B. (2015). The horseshoe+ estimator of ultra-sparse signals. *arXiv:1502.00560*.

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2009). Handling sparsity via the horseshoe. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 73–80.

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika*, 97(2):465–480.

Datta, J. and Ghosh, J. K. (2013). Asymptotic properties of Bayes risk for the horseshoe prior. *Bayesian Analysis*, 8(1):111–132.

Dupuis, J. A. and Robert, C. P. (2003). Variable selection in qualitative models via an entropic explanatory power. *Journal of Statistical Planning and Inference*, 111(1-2):77–94.

Goutis, C. and Robert, C. P. (1998). Model choice in generalised linear models: a Bayesian approach via Kullback–Leibler projections. *Biometrika*, 85(1):29–37.

Mitchell, T. J. and Beauchamp, J. J. (1988). Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1036.

Piironen, J. and Vehtari, A. (2015). Comparison of Bayesian predictive methods for model selection. *arXiv:1503.08650.*

van der Pas, S. L., Kleijn, B. J. K., and van der Vaart, A. W. (2014). The horseshoe estimator: posterior concentration around nearly black vectors. *Electronic Journal of Statistics*, 8(2):2585–2618.

Vehtari, A. and Ojanen, J. (2012). A survey of Bayesian predictive methods for model assessment, selection and comparison. *Statistics Surveys*, 6:142–228.

Acknowledgments

We thank Tomi Peltola for sharing his Stan-codes which helped implementing the models in this document.

A Projection for linear Gaussian model

Let $f_i^\perp = \mathbf{w}_\perp{}^\mathsf{T}\mathbf{x}_i^\perp$ denote the fit of the submodel, where $\mathbf{w}_\perp$ is the projected weight vector and $\mathbf{x}_i^\perp$ the input vector corresponding to the submodel on which we are projecting. Moreover, let $\sigma_\perp^2$ denote the projected noise variance. The cost function in (4) can be written as

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n} \mathrm{KL}\big(\mathrm{N}(f_i,\sigma^2) \,\|\, \mathrm{N}(f_i^\perp,\sigma_\perp^2)\big) \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(\log\frac{\sigma_\perp^2}{\sigma^2} + \frac{\sigma^2 + (f_i - f_i^\perp)^2}{\sigma_\perp^2} - 1\right) \\
&= \frac{1}{2}\left(\log\frac{\sigma_\perp^2}{\sigma^2} + \frac{1}{\sigma_\perp^2}\left(\sigma^2 + \frac{1}{n}\sum_{i=1}^{n}(f_i - f_i^\perp)^2\right) - 1\right) \\
&= \frac{1}{2}\left(\log\frac{\sigma_\perp^2}{\sigma^2} + \frac{1}{\sigma_\perp^2}\left(\sigma^2 + \frac{1}{n}(\mathbf{f} - \mathbf{f}_\perp)^\mathsf{T}(\mathbf{f} - \mathbf{f}_\perp)\right) - 1\right), \\
&= \frac{1}{2}\left(\log\frac{\sigma_\perp^2}{\sigma^2} + \frac{1}{\sigma_\perp^2}\left(\sigma^2 + \frac{1}{n}(\mathbf{X}\mathbf{w} - \mathbf{X}_\perp\mathbf{w}_\perp)^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{X}_\perp\mathbf{w}_\perp)\right) - 1\right), \qquad \text{(A.1)}
\end{aligned}$$

where $\mathbf{f} = (f_1,\dots,f_n)$, $\mathbf{f}_\perp = (f_1^\perp,\dots,f_n^\perp)$, and $\mathbf{X}$ and $\mathbf{X}_\perp$ denote the predictor matrices of the full and submodel, respectively. Setting the gradient of (A.1) with respect to $\mathbf{w}_\perp$ to zero, we obtain

$$\mathbf{w}_\perp = (\mathbf{X}_\perp{}^\mathsf{T}\mathbf{X}_\perp)^{-1}\mathbf{X}_\perp{}^\mathsf{T}\,\mathbf{X}\mathbf{w}. \qquad \text{(A.2)}$$

Accordingly, minimizing (A.1) with respect to $\sigma_\perp^2$ gives us

$$\sigma_\perp^2 = \sigma^2 + \frac{1}{n}(\mathbf{X}\mathbf{w} - \mathbf{X}_\perp\mathbf{w}_\perp)^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{X}_\perp\mathbf{w}_\perp). \qquad \text{(A.3)}$$

Plugging the expressions (A.2) and (A.3) into (A.1) gives us the divergence introduced by the projection of these particular parameters $(\mathbf{w}, \sigma^2)$

$$
\begin{aligned}
d(\mathbf{w}, \sigma^2) &= \frac{1}{n} \sum_{i=1}^{n} \mathrm{KL}\big(\mathrm{N}\big(f_i, \sigma^2\big) \parallel \mathrm{N}\big(f_i^{\perp}, \sigma_{\perp}^2\big)\big) \\
&= \frac{1}{2} \log \frac{\sigma^2}{\sigma_{\perp}^2}. \qquad \text{(A.4)}
\end{aligned}
$$

B Stan codes

Linear regression with Gaussian noise and HS-t_ν prior (2) on the weights. The case $\nu = 1$ corresponds to the horseshoe.

```
/* lg_t.stan */

functions {
    // square root of a vector (elementwise)
    vector sqrt_vec(vector x) {
        vector[dims(x)[1]] res;

        for (m in 1:dims(x)[1]){
            res[m] <- sqrt(x[m]);
        }
        return res;
    }
}

data {
    int<lower=0> n; // number of observations
    int<lower=0> d; // number of predictors
    vector[n] y;    // outputs
    matrix[n,d] X;  // inputs
    real<lower=1> nu; // degrees of freedom for the half t-priors
}

parameters {

    // intercept and noise std
    real w0;
    real<lower=0> sigma;

    // auxiliary variables for the variance parameters
    vector[d] z;
    real<lower=0> r1_global;
    real<lower=0> r2_global;
    vector<lower=0>[d] r1_local;
    vector<lower=0>[d] r2_local;
}

transformed parameters {

    // global and local variance parameters, and the input weights
    real<lower=0> tau;
    vector<lower=0>[d] lambda;
    vector[d] w;

    tau <- r1_global * sqrt(r2_global);
    lambda <- r1_local .* sqrt_vec(r2_local);
    w <- z .* lambda*tau;
}

model {

    // observation model
    y ~ normal(w0 + X*w, sigma);

    // half t-priors for lambdas (nu = 1 corresponds to horseshoe)
    z ~ normal(0, 1);
    r1_local ~ normal(0.0, 1.0);
    r2_local ~ inv_gamma(0.5*nu, 0.5*nu);

    // half cauchy for tau
    r1_global ~ normal(0.0, 1.0);
    r2_global ~ inv_gamma(0.5, 0.5);

    // weakly informative prior for the intercept
    w0 ~ normal(0,5);

    // using uniform prior on the noise variance
}
```

The same model as above, but with HS-t_ν+ prior (3) on the weights. The case $\nu = 1$ corresponds to the horseshoe+.

```
/* lg_tplus.stan */

functions {
    // square root of a vector (elementwise)
    vector sqrt_vec(vector x) {
        vector[dims(x)[1]] res;

        for (m in 1:dims(x)[1]){
            res[m] <- sqrt(x[m]);
        }
        return res;
    }
}

data {
    int<lower=0> n; // number of observations
    int<lower=0> d; // number of predictors
    vector[n] y;    // outputs
    matrix[n,d] X;  // inputs
    real<lower=1> nu; // degrees of freedom for the half t-priors
}

parameters {

    // intercept and noise std
    real w0;
    real<lower=0> sigma;

    // auxiliary variables for the variance parameters
    vector[d] z;
    real<lower=0> r1_global;
    real<lower=0> r2_global;
    vector<lower=0>[d] r1_local;
    vector<lower=0>[d] r2_local;
    vector<lower=0>[d] r1_localplus;
    vector<lower=0>[d] r2_localplus;
}

transformed parameters {

    // global and local variance parameters, and the input weights
    real<lower=0> tau;
    vector<lower=0>[d] lambda;
    vector<lower=0>[d] lambdaplus;
    vector[d] w;

    tau <- r1_global * sqrt(r2_global);
    lambda <- r1_local .* sqrt_vec(r2_local);
    lambdaplus <- r1_localplus .* sqrt_vec(r2_localplus);
    w <- z .* lambda .* lambdaplus * tau;
}

model {

    // observation model
    y ~ normal(w0 + X*w, sigma);

    // half t-priors for all the lambdas (nu = 1 corresponds to horseshoe+)
    z ~ normal(0, 1);
    r1_local ~ normal(0.0, 1.0);
    r2_local ~ inv_gamma(0.5*nu, 0.5*nu);
    r1_localplus ~ normal(0.0, 1.0);
    r2_localplus ~ inv_gamma(0.5*nu, 0.5*nu);

    // half cauchy for tau
    r1_global ~ normal(0.0, 1.0);
    r2_global ~ inv_gamma(0.5, 0.5);

    // weakly informative prior for the intercept
    w0 ~ normal(0,5);

    // using uniform prior on the noise variance
}
```

C R codes

Code for performing the projection onto a submodel for linear Gaussian model, given a sample from the full model:

```
lm_proj <- function(w,sigma2,x,indproj) {

    # assume the intercept term is stacked into w, and x contains
    # a corresponding vector of ones. returns the projected samples
    # and estimated kl-divergence.

    # pick the columns of x that form the projection subspace
    n <- dim(x)[1]
    xp <- x[,indproj]

    # solve the projection equations
    fit <- x %*% w # fit of the full model
    wp <- solve(t(xp) %*% xp, t(xp) %*% fit)
    sigma2p <- sigma2 + colMeans((fit - xp %*% wp)^2)

    # this is the estimated kl-divergence between the full and projected model
    kl <- mean(0.5*log(sigma2p/sigma2))

    # reshape wp so that it has same dimensionality as x, and zeros for
    # those variables that are not included in the projected model
    d <- dim(w)[1]
    S <- dim(w)[2]
    wptemp <- matrix(0, d, S)
    wptemp[indproj,] <- wp
    wp <- wptemp

    return(list(w=wp, sigma2=sigma2p, kl=kl))
}
```

Forward variable searching by minimizing the associated KL-divergence in the projection:

```
lm_fprojsel <- function(w, sigma2, x) {

    # forward variable selection using the projection. returns the
    # indices of the variables chosen (in order) and accociated kl-divergences.

    d = dim(x)[2]
    chosen <- 1 # chosen variables, start from the model with the intercept only
    notchosen <- setdiff(1:d, chosen)

    # start from the model having only the intercept term
    kl <- rep(0,d)
    kl[1] <- lm_proj(w,sigma2,x,1)$kl

    # start adding variables one at a time
    for (k in 2:d) {

        nleft <- length(notchosen)
        val <- rep(0, nleft)

        for (i in 1:nleft) {
            ind <- sort( c(chosen, notchosen[i]) )
            proj <- lm_proj(w,sigma2,x,ind)
            val[i] <- proj$kl
        }

        # find the variable that minimizes the kl
        imin <- which.min(val)
        chosen <- c(chosen, notchosen[imin])
        notchosen <- setdiff(1:d, chosen)

        kl[k] <- val[imin]
    }
    return(list(chosen=chosen, kl=kl))
}
```

Example of sampling the full model and performing the variable selection using the projection:

```
# load the data
n <- 1000 # number of training points
data <- as.matrix(read.table('crimedata.csv', sep=','))
dims <- dim(data)
ntotal <- dims[1]
nt <- ntotal - n # number of test points
d <- dims[2]-1

# training data
y <- data[1:n, 1]
x <- data[1:n, 2:(d+1)]

# test data
yt <- data[(n+1):ntotal, 1]
xt <- data[(n+1):ntotal, 2:(d+1)]

# fit the full model
fit <- stan("lg_t.stan", data=list(X=x,y=y,d=d,n=n,nu=3.0), iter=100, chains=1)
e <- extract(fit)

w <- rbind(e$w0, t(e$w)) # stack the intercept and weights
sigma2 <- e$sigma^2
x <- cbind(rep(1,n), x) # add a vector of ones to the predictor matrix
xt <- cbind(rep(1,ntest), xt)

# perform the variable selection
spath <- lm_fprojsel(w, sigma2, x)

# compute the predictions on the test set using the projected parameters
mlpd <- rep(0, d+1)
mse <- rep(0, d+1)
for (k in 1:(d+1)) {

    # projected parameters
    submodel <- lm_proj(w, sigma2, x, spath$chosen[1:k])
    wp <- submodel$w
    sigma2p <- submodel$sigma2

    # mean squared error
    ypred <- rowMeans(xt %*% wp)
    mse[k] <- mean((yt-ypred)^2)

    # mean log predictive density
    pd <- dnorm(yt, xt %*% wp, sqrt(sigma2p))
    mlpd[k] <- mean(log(rowMeans(pd)))
}
```