SUPPLEMENTARY MATERIAL

Non-Gaussian Likelihood

Posterior approximation Using the second order Taylor expansion for the log likelihood terms $L_i(f_i, \phi)$ (where $f_i = \boldsymbol{\beta}^\mathsf{T}\mathbf{x}_i$ and ϕ denotes a possible dispersion parameter), we can approximate the posterior as (ch. 16.2, Gelman et al., 2013)

$$\log p(\boldsymbol{\beta} \,|\, \boldsymbol{\Lambda}, \tau, \phi, \mathcal{D}) \approx \log p(\boldsymbol{\beta} \,|\, \boldsymbol{\Lambda}, \tau, \phi) - \sum_{i=1}^{n} \frac{1}{2\tilde{\sigma}_i^2}(\tilde{z}_i - f_i)^2 + \text{const.},$$

where

$$\tilde{z}_i = f_i - \frac{L_i'(f_i,\phi)}{L_i''(f_i,\phi)}, \quad \tilde{\sigma}_i^2 = -\frac{1}{L_i''(f_i,\phi)},$$

denote the location and variance of the Gaussian pseudo-observations. The derivatives are calculated w.r.t. f_i at the posterior mode $\bar{f}_i = \bar{\boldsymbol{\beta}}^\mathsf{T}\mathbf{x}_i$. Using these, the posterior (given the hyperparameters) is approximately

$$p(\boldsymbol{\beta} \,|\, \boldsymbol{\Lambda}, \tau, \phi, \mathcal{D}) \approx \mathrm{N}\big(\boldsymbol{\beta} \,|\, \bar{\boldsymbol{\beta}}, \boldsymbol{\Sigma}\big),$$
$$\bar{\boldsymbol{\beta}} = \tau^2\boldsymbol{\Lambda}\left(\tau^2\boldsymbol{\Lambda} + (\mathbf{X}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{X})^{-1}\right)^{-1}\hat{\boldsymbol{\beta}},$$
$$\boldsymbol{\Sigma} = (\tau^{-2}\boldsymbol{\Lambda}^{-1} + \mathbf{X}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{X})^{-1},$$

where $\tilde{\mathbf{z}} = (\tilde{z}_1, \dots, \tilde{z}_n)$, $\tilde{\boldsymbol{\Sigma}} = \mathrm{diag}(\tilde{\sigma}_1^2, \dots, \tilde{\sigma}_n^2)$ and $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{z}}$ (assuming the first inverse exists).

Logistic regression Consider the logistic regression model

$$p(y_i = 1 \,|\, f_i) = s(f_i) = \frac{1}{1 + \exp(-f_i)}.$$

The second derivative for the ith log-likelihood term is given by

$$L_i''(f_i) = \left(\frac{y_i}{s(f_i)} - \frac{1-y_i}{1-s(f_i)}\right)s''(f_i) - \left(\frac{y_i}{s(f_i)^2} + \frac{1-y_i}{(1-s(f_i))^2}\right)(s'(f_i))^2.$$

If we now plug in the derivatives

$$s'(f_i) = s(f_i)(1 - s(f_i)),$$
$$s''(f_i) = s'(f_i)(1 - 2s(f_i)),$$

after a few lines of straightforward algebra, we are left with

$$L_i''(f_i) = s(f_i)(s(f_i) - 1).$$

This is a strictly negative function with minimum at $s(f_i) = \frac{1}{2}$, which occurs when $f_i = 0$. Thus also $\tilde{\sigma}_i^2 = -1/L_i''(f_i)$ is minimized at $f_i = 0$. In other words, those points that lie on the classification boundary are the most informative ones, and the pseudo-variance for these points is

$$\tilde{\sigma}_i^2 = -\frac{1}{\frac{1}{2}\left(-\frac{1}{2}\right)} = 4.$$

This result serves as a useful reference value as discussed in Section 3.4.

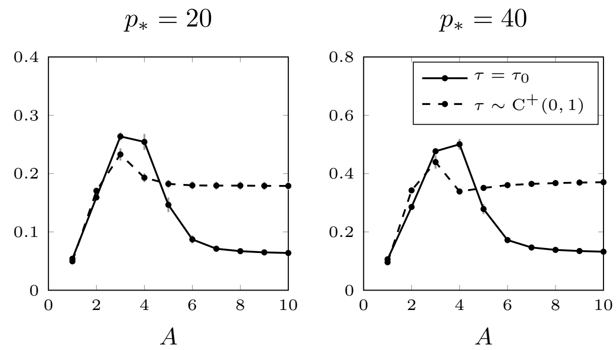

Figure 6: *Synthetic example*: Mean squared error (MSE) between the estimated and the true coefficient vector of length $n = 400$ on average over 100 different data realizations. The true coefficient vector has either $p_* = 20$ or $p_* = 40$ elements with a nonzero value equal to A and the rest of the coefficients are set to zero.

Additional experiments

We present here an additional synthetic experiment that could not fit to the main content of the paper. The example is taken from van der Pas et al. (2014). Consider model (9), where each y_i is generated by adding Gaussian noise with $\sigma^2 = 1$ to the corresponding signal β_i. We generated 100 data realizations with $n = 400$ and the true $\boldsymbol{\beta}_*$ having either $p_* = 20$ or $p_* = 40$ nonzero entries equal to $A = 1, 2, \dots, 10$ with the rest of the entries being zeros. We then computed the mean squared error (MSE) between the estimated posterior mean $\bar{\boldsymbol{\beta}}$ and the true $\boldsymbol{\beta}_*$ for the prior $\tau \sim \mathrm{C}^+(0,1)$ and for $\tau = \tau_0$, where τ_0 is computed from Equation (17) with the oracle prior guess $p_0 = p_*$. The purpose of this setup is to further demonstrate how one could benefit from the prior knowledge about the sparsity of $\boldsymbol{\beta}$ using our framework, provided such prior knowledge exists. Notice though, that also in the latter case τ has a distribution because it depends on σ which is treated as an unknown parameter.

Figure 6 shows the MSE for the two priors for the different values of p_* and A. For both priors the error

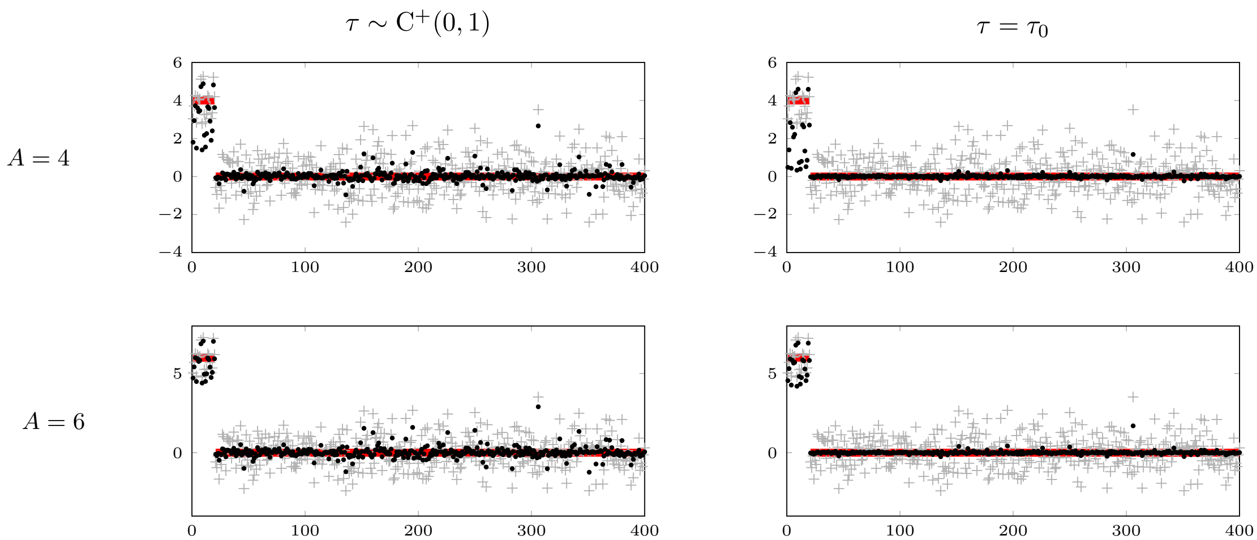

Figure 7: *Synthetic example*: An example data realization $\mathbf{y}$ (crosses), posterior mean $\bar{\boldsymbol{\beta}}$ (dots) and the true signal $\boldsymbol{\beta}_*$ (red lines) for $A = 4$ and $A = 6$ when $p_* = 20$. In both cases the oracle value for τ helps to shrink the zero components in $\boldsymbol{\beta}$ but also overshrinks the actual signals in the case $A = 4$.

is largest around $A \approx 3.5$, which is called the "universal threshold" by van der Pas et al. (2014). Below this threshold the nonzero components in $\boldsymbol{\beta}$ are too small to be detected and are thus shrunk too heavily towards zero which introduces error. For $A = 4$ the oracle prior actually yields worse results due to this overshrinkage (see discussion below), but gives clearly superior results for larger A.

Figure 7 illustrates the data $\mathbf{y}$ and the estimated coefficients $\bar{\boldsymbol{\beta}}$ for one particular data realization when $A = 4$ and $A = 6$. In both cases the oracle choice of τ helps to shrink the zero components in $\boldsymbol{\beta}$ towards zero, but for $A = 4$ also overshrinks the nonzero components. The reason for the overshrinkage is that some observations y_i that correspond to zero signal ($\beta_i = 0$) happen to have similar magnitude to the observations coming from an actual signal ($\beta_i = A$), and thus these irrelevant components "steal" from the limited budget for m_{eff}. For this particular value of A (and p_*) the overshrinkage of the actual signals happens to be worse in terms of MSE than undershrinkage of the zero components, and thus one would get better results by setting p_0 to be slightly above the true p_* (results not shown). For $A = 6$ the actual signals are large enough to be distinguished from zero, and the oracle selection of τ yields substantially better estimate for $\boldsymbol{\beta}$.

Stan codes

The following shows the Stan code for the linear Gaussian model. We use the parametrization proposed by Peltola et al. (2014) (codes at https://github.com/to-mi/stan-survival-shrinkage) as it is more robust for sampling than the literal (3). Even with this parametrization we usually set `adapt_delta` $= 0.99$ when calling Stan, as this can sometimes reduce the number of divergent transitions which can be an issue for the horseshoe prior (see Piironen and Vehtari, 2015).

In the code, both τ and λ_j are given half-t priors with the degrees of freedom and the scale defined by the user (the scale can be adjusted only for τ, the local parameters λ_j have unit scale). Setting `nu_local` $= 1$ corresponds to the horseshoe. `nu_global` $= 1$ gives τ a half-Cauchy prior, whereas fixing `nu_global` to some large value (say 100) would give τ practically a half-normal prior. The scale for τ is `scale_global*sigma`, so if we want to set this to be $\tau_0 = \frac{p_0}{D-p_0}\frac{\sigma}{\sqrt{n}}$ (Eq. (16)), we should set `scale_global` $= \frac{p_0}{(D-p_0)\sqrt{n}}$.

```
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=0> scale_icept;  // prior std for the intercept
    real<lower=0> scale_global; // scale for the half-t prior for tau
                                // (tau0 = scale_global*sigma)
    real<lower=1> nu_global;    // degrees of freedom for the half-t prior for tau
    real<lower=1> nu_local;     // degrees of freedom for the half-t priors for lambdas
                                // (nu_local = 1 corresponds to the horseshoe)
}

parameters {
    real beta0;     // intercept
    real logsigma;  // log of noise std

    // auxiliary variables that define the global and local 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 {
    real<lower=0> tau;          // global shrinkage parameter
    vector<lower=0>[d] lambda;  // local shrinkage parameters
    vector[d] beta;             // regression coefficients
    vector[n] f;                // latent values
    real sigma;                 // noise std

    sigma = exp(logsigma);
    lambda = r1_local .* sqrt(r2_local);
    tau = r1_global * sqrt(r2_global);
    beta = z .* lambda*tau;
    f = beta0 + x*beta;
}

model {
    // half-t priors for lambdas
    z ~ normal(0, 1);
    r1_local ~ normal(0.0, 1.0);
    r2_local ~ inv_gamma(0.5*nu_local, 0.5*nu_local);

    // half-t prior for tau
    r1_global ~ normal(0.0, scale_global*sigma);
    r2_global ~ inv_gamma(0.5*nu_global, 0.5*nu_global);

    // gaussian prior for the intercept
    beta0 ~ normal(0,scale_icept);

    // observation model
    y ~ normal(f, sigma);
}
```

The code for the logistic regression model is very similar, we simply remove the lines related to the noise deviation `sigma`, and change the observation model and the type of the target variable data `y`. The scale for τ is now simply `scale_global`. Thus, to follow our recommendation, we set `scale_global` $= \tau_0 = \frac{p_0}{D-p_0}\frac{\sigma}{\sqrt{n}}$ (Eq. (16)), by plugging in $\sigma = 2$ (Sec. 3.4).

```
data {
    int<lower=0> n;               // number of observations
    int<lower=0> d;               // number of predictors
    int<lower=0,upper=1> y[n];    // outputs
    matrix[n,d] x;                // inputs
    real<lower=0> scale_icept;    // prior std for the intercept
    real<lower=0> scale_global;   // scale for the half-t prior for tau
    real<lower=1> nu_global;      // degrees of freedom for the half-t priors for tau
    real<lower=1> nu_local;       // degrees of freedom for the half-t priors for lambdas
                                  // (nu_local = 1 corresponds to the horseshoe)
}

parameters {
    real beta0; // intercept

    // auxiliary variables that define the global and local 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 {
    real<lower=0> tau;            // global shrinkage parameter
    vector<lower=0>[d] lambda;    // local shrinkage parameter
    vector[d] beta;               // regression coefficients
    vector[n] f;                  // latent values

    lambda = r1_local .* sqrt(r2_local);
    tau = r1_global * sqrt(r2_global);
    beta = z .* lambda*tau;
    f = beta0 + x*beta;
}

model {
    // half-t priors for lambdas
    z ~ normal(0, 1);
    r1_local ~ normal(0.0, 1.0);
    r2_local ~ inv_gamma(0.5*nu_local, 0.5*nu_local);

    // half-t prior for tau
    r1_global ~ normal(0.0, scale_global);
    r2_global ~ inv_gamma(0.5*nu_global, 0.5*nu_global);

    // gaussian prior for the intercept
    beta0 ~ normal(0,scale_icept);

    // observation model
    y ~ bernoulli_logit(f);
}
```