NaNs

UPDATE: Loo score works now with Logistic Horseshoe_regularised & real data to predict depression!

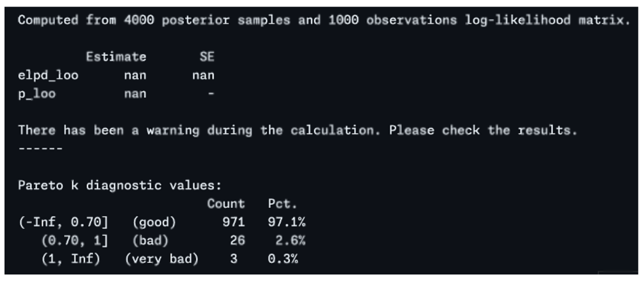
Model: Laplace

Scenario: 11, mixed coefficient sizes, 1000 samples, 5 active, 5 inactive features

Experiments:

1. Different priors for tau
   a.

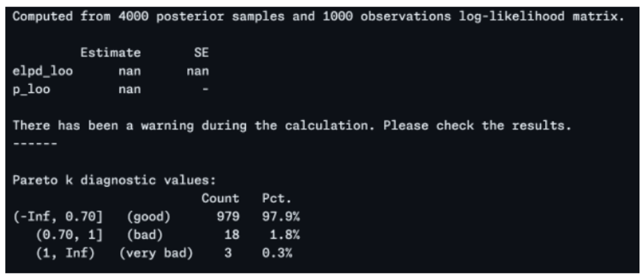
```
beta ~ double_exponential(0, tau);
alpha ~ normal(0, 5);
tau ~ cauchy(0, 0.1);
```



```
Computed from 4000 posterior samples and 1000 observations log-likelihood matrix.

         Estimate       SE
elpd_loo      nan      nan
p_loo         nan        -

There has been a warning during the calculation. Please check the results.
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.70]   (good)      971   97.1%
   (0.70, 1]   (bad)        26    2.6%
   (1, Inf)   (very bad)    3    0.3%
```

   b.

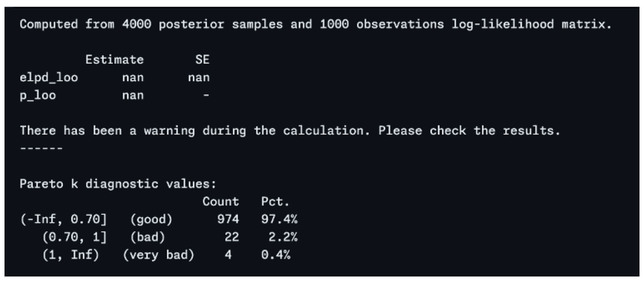
```
beta ~ double_exponential(0, tau);
alpha ~ normal(0, 5);
tau ~ cauchy(0, 0.5);
```



```
Computed from 4000 posterior samples and 1000 observations log-likelihood matrix.

         Estimate       SE
elpd_loo      nan      nan
p_loo         nan        -

There has been a warning during the calculation. Please check the results.
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.70]   (good)      979   97.9%
   (0.70, 1]   (bad)        18    1.8%
   (1, Inf)   (very bad)    3    0.3%
```

   c.

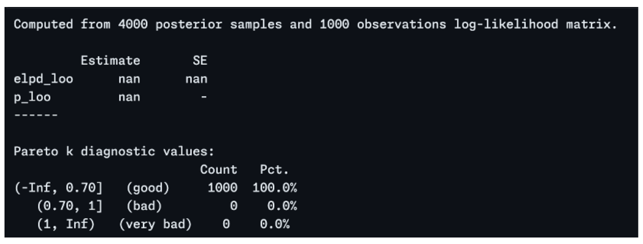
```
beta ~ double_exponential(0, tau);
alpha ~ normal(0, 5);
tau ~ cauchy(0, 1);
```



```
Computed from 4000 posterior samples and 1000 observations log-likelihood matrix.

         Estimate       SE
elpd_loo      nan      nan
p_loo         nan        -

There has been a warning during the calculation. Please check the results.
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.70]   (good)      974   97.4%
   (0.70, 1]   (bad)        22    2.2%
   (1, Inf)   (very bad)    4    0.4%
```

   d.

```
beta ~ double_exponential(0, tau);
alpha ~ normal(0, 5);
tau ~ cauchy(0, 10);
```



```
Computed from 4000 posterior samples and 1000 observations log-likelihood matrix.

         Estimate       SE
elpd_loo      nan      nan
p_loo         nan        -
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.70]   (good)     1000  100.0%
   (0.70, 1]   (bad)         0    0.0%
   (1, Inf)   (very bad)    0    0.0%
```

2. Tried increasing number of samples from 1000 to 4000, as I thought it might give me more reliable posterior distribution
3. Tried increasing number of chains from 4 to 8

Result: still get NaNs all the time.

Reason:

```
(1) RuntimeWarning: divide by zero encountered in divide
  b_ary /= prior_bs * ary[int(n / 4 + 0.5) - 1]
```

```
(2) RuntimeWarning: invalid value encountered in multiply
  k_ary = np.log1p(-b_ary[:, None] * ary).mean(axis=1)  # pylint: disable=no-member
(3) RuntimeWarning: invalid value encountered in scalar divide
  sigma = -k_post / b_post
```

This is from the `_gpdfit` method that is used by `loo()` to Estimate the parameters for the Generalized Pareto Distribution (GPD) given the data.

If I understand correctly, `ary` from the warning above contains log likelihood values from my Laplace model. So if those values are really small, the multiplication in warning (1) might be very close to 0 ⇒ division by 0 ⇒ b_ary becomes nan

And then in (2) given that b_ary is nan, it can't be multiplied…

4. Tried adding a small value to each log_likelihood, still get nans…

    ```
    generated quantities {
      array[N] int y_new;
      y_new = bernoulli_logit_rng(X * beta + alpha);

      array[N] real log_lik;
      for (n in 1:N) {
        real log_prob = bernoulli_logit_lpmf(y[n] | dot_product(X[n], beta) + alpha);
        log_lik[n] = log_prob + 1e-10;
      }
    }
    ```

This is my whole Laplace Model:

```
laplace_sparse_logistic_regression_model = """
data {
  int<lower=1> N;          // number of observations
  int<lower=0> K;          // number of features
  matrix[N, K] X;          // predictor matrix
  array[N] int y;  // binary outcome variable
}
parameters {
  vector[K] beta;          // coefficients for predictors
  real alpha;              // intercept

  real<lower=0> tau;       // scale parameter for Laplace distribution. Smaller T => stronger
}
model {
  // Priors
  beta ~ double_exponential(0, tau); // Laplace prior for coefficients
  alpha ~ normal(0, 5);
  tau ~ cauchy(0, 10);

  // Likelihood
  y ~ bernoulli_logit(X * beta + alpha);
}
generated quantities {
  array[N] int y_new;
  y_new = bernoulli_logit_rng(X * beta + alpha);

  array[N] real log_lik;
  for (n in 1:N) {
```

```
    log_lik[n] = bernoulli_logit_lpmf(y[n] | dot_product(X[n], beta) + alpha);
  }
}
"""
```

5. I looked into the log_lik values, and indeed many values are really really small. So I tried replacing them with a small value instead:

```
idata = az.from_pystan(posterior=fit)

# Replace super small values with a small value
log_lik = idata.log_likelihood['log_lik']
log_lik = log_lik.where((log_lik > 0) & (log_lik < 1e-10), 1e-10)
log_lik = log_lik.where((log_lik > -1e-10) & (log_lik < 0), -1e-10)
idata.log_likelihood['log_lik'] = log_lik

# Calculate as usual
loo_result = az.loo(idata, pointwise=True)
```

And it did solve my nan issue! but it's a bad solution 😅🤪😩

```
Computed from 4000 posterior samples and 1000 observations log-likelihood matrix.

          Estimate       SE
elpd_loo     -0.00     0.00
p_loo         0.00        -

There has been a warning during the calculation. Please check the results.
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.70]   (good)        0    0.0%
   (0.70, 1]   (bad)         0    0.0%
   (1, Inf)   (very bad) 1000  100.0%
```

Model: Horseshoe_regularised

Scenario: C, real data, binary values recoded from 1/2 to 0/1, numerical features standardised, categorical features from misc questionnaire one-hot-encoded, features from other questionnaires left as they are, imputed values rounded

Experiments:

a.

```
'scale_icept': 10,  # (paper) prior std for the intercept: larger => intercept can take more va
'scale_global':  (1 / (X.shape[1] - 1)) * (2 / np.sqrt(X.shape[0])), #(formula: (1/(d-1)) * (2/
 
'nu_global': 1, # >1 !dof for the half-t prior for tau: smaller => heavier tails, more deviatio
'nu_local': 1, # smaller -> heavier tails
'slab_scale': 5, # (paper value) regularisation to prevent extremely large coefficients. larger
'slab_df': 4  #(paper value) lower => heavier tails and larger coefficients
```

```
Computed from 4000 posterior samples and 1080 observations log-likelihood matrix.

         Estimate       SE
elpd_loo      nan      nan
p_loo         nan        -

There has been a warning during the calculation. Please check the results.
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.70]   (good)       63    5.8%
   (0.70, 1]   (bad)       186   17.2%
   (1, Inf)   (very bad)  831   76.9%
```

values outside small zone [-1e-10: 1e-10]: 2503780
values inside small zone: 1816220

b.

```
'scale_icept': 10,
'scale_global':  (1 / (X.shape[1] - 1)) * (2 / np.sqrt(X.shape[0])),

'nu_global': 10,
'nu_local': 10,
'slab_scale': 5,
'slab_df': 4
```

```
Computed from 4000 posterior samples and 1080 observations log-likelihood matrix.

         Estimate       SE
elpd_loo  -478.85    15.48
p_loo      112.47        -

There has been a warning during the calculation. Please check the results.
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.70]   (good)      895   82.9%
   (0.70, 1]   (bad)        55    5.1%
   (1, Inf)   (very bad)  130   12.0%
```

values outside small zone [-1e-10: 1e-10]: 4230836
values inside small zone: 89164

🥳

P.S: Laplace and other basic models probably struggle to work with this real data for now just because the models are quite weak themselves, but i will take a look more