Applied Probabilistic Programming & Bayesian Machine Learning

§3-4: Correlation

MIT IAP 2017 Max W. Shen

1 Correlation among Predictors: Identifiability Part 2

In Section 1-4, we discussed identifiability issues in model 1:

$$\begin{aligned} y_i &\sim N(\mu_0 + \mu_1, \sigma) \\ \mu_0 &\sim \text{weakly-informative}() \\ \mu_1 &\sim \text{weakly-informative}() \end{aligned} \tag{1}$$

In regression, we have an N-by-D feature matrix with N observations, each with values for D features. If two features are highly correlated or identical across all N observations, then these two feature vectors are going to have identifiability issues. When two features are completely identical, with flat priors on the weights, then our model is completely non-identifiable, and our posterior standard deviations are going to be infinite. If two features are correlated but not identical, then our model is weakly-identifiable, and our posterior standard deviations are going to be high.

Figure 1 simulates two feature vectors with varying correlation (x-axis) and plots the corresponding posterior standard deviation. We can see that above a correlation of 0.90, our posterior standard deviation starts to curve sharply upwards, approaching infinity as the correlation approaches 1.

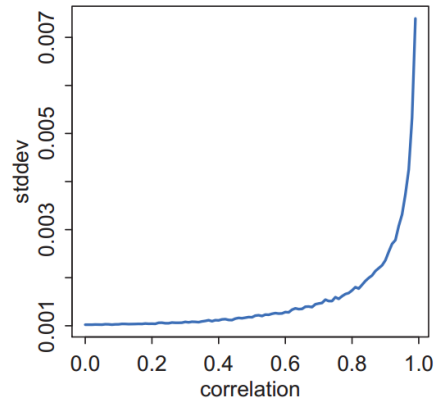

FIGURE 5.10. The effect of correlated predictor variables on the narrowness of the posterior distribution. The vertical axis shows the standard deviation of the posterior distribution of the slope. The horizontal axis is the correlation between the predictor of interest and another predictor added to the model. As the correlation increases, the standard deviation inflates.

Figure 1: Statistical Rethinking Ch 5.3 — pg. 149

1.1 In Practice

In practice, it is useful to start data analysis by plotting the pairwise correlation between all features. This can give you a sense for potential collinearity and identifiability issues.

However, high correlation between two features does not necessarily mean your model will have identifiability issues, since regression takes into account the rest of your features as well. **Identifiability issues arise when one feature, given all the rest of your features, provides little to no additional information.**

In some fields, running your data through principal component analysis (PCA) or some other factorization method that outputs orthogonal, or statistically independent, feature vectors is a useful way to avoid collinearity issues. However, some scientific fields frown upon PCA preprocessing for producing feature vectors that are not directly interpretable, and for introducing more assumptions into your model.

In practice, if you have identified a cluster of highly collinear features, you can just keep one to use in your model for more stable inference. If you can show that no matter which one you use, you get similar predictions, then you can argue that each collinear feature is really providing the same information.

Exercise 1

In the "milk" toy dataset, the "perc.fat" and "perc.lactose" features are highly correlated. Find their correlation.

Incorporate both into a regression model to predict the energy content of milk. What do you observe as you incorporate or exclude other features?

2 Correlation among Parameters: Modeling with Covariance

The root problem in non-identifiable and weakly-identifiable models is correlation between parameters, causing issues in inference. The ideal case for inference is for all parameters to be completely independent from all other parameters, producing a more "compact" posterior distribution enabling faster inference.

If we have reason to believe that some parameters are correlated, a good approach is to jointly model the parameters using a multivariate distribution that explicitly includes a covariance matrix between the parameters.

Consider a single-variable regression model where we believe the intercept c and the slope w are correlated:

$$\begin{aligned} y_i &\sim Normal(\mu_i, \sigma) \\ \mu_i &= c + wx_i \\ \begin{bmatrix} c \\ w \end{bmatrix} &\sim MVNormal\Big(\begin{bmatrix} \mu_c \\ \mu_w \end{bmatrix}, \mathbf{S} \Big) \\ \mathbf{S} &= \begin{pmatrix} \sigma_c & 0 \\ 0 & \sigma_w \end{pmatrix} \mathbf{R} \begin{pmatrix} \sigma_c & 0 \\ 0 & \sigma_w \end{pmatrix} \\ \mu_c, \mu_w, \sigma, \sigma_c, \sigma_w &\sim \text{weakly-informative}() \\ \mathbf{R} &\sim LKJcorr(1) \end{aligned} \tag{2}$$

Here, $\mathbf{S}$ is our covariance matrix, which we decompose into:

$$\mathbf{S} = \begin{pmatrix} \sigma_c & 0 \\ 0 & \sigma_w \end{pmatrix} \mathbf{R} \begin{pmatrix} \sigma_c & 0 \\ 0 & \sigma_w \end{pmatrix} \tag{3}$$

σ_c and σ_w are each variable's typical standard deviation. $\mathbf{R}$ is a correlation matrix, with the general form given by equation 4, where ρ is the correlation between our two variables.

$$\mathbf{R} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \tag{4}$$

Working with $\mathbf{R}$ is useful because we can conveniently express prior knowledge directly on ρ using an LKJCorr prior distribution, which has a single parameter η. Figure 2 depicts the impact of eta on the shape of the distribution. $\eta = 1$ provides a uniform distribution over correlation matrices, while $\eta > 1$ provides a peak at $\rho = 0$ correlation, and $\eta < 1$ produces a trough at $\rho = 0$ correlation.

The LKJCorr probability density function, for Σ which is a positive-definite symmetric matrix with unit diagonals, is given in 5.

$$LkjCorr(\Sigma|\eta) \propto \det(\Sigma)^{(\eta-1)} \tag{5}$$

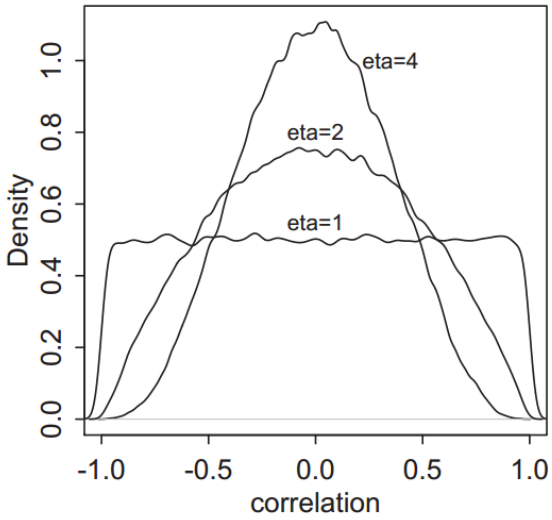

Figure 2: LKJ's Eta parameter. Statistical Rethinking, 13.1 — pg. 394

2.1 Stan Implementation

In this folder, you will find "model_3-4.stan", which implements model 2. You can run the model using python or R.

Here, we provide a description of the simulated data, and some comments on new Stan functionality we use for the model.

2.1.1 Simulated Data

The setup: We have $L = 2$ weights in our regression: An intercept c and a weight w. We have an N-by-D feature matrix. In this case, X is binary. Our output matrix, Y, is the same dimensions as X, where

$$Y_{i,j} \sim Normal(c + w * X_{i,j}, \sigma) \tag{6}$$

Here, we have $N = 20$ draws from a multivariate normal for c and w, producing N different values for c and w.

Our feature matrix X is N-by-D, representing $D = 10$ observations of this particular binary feature. In this case, the first 5 values of X are 0 and the last 5 values are 1, same for all N.

The true values used for simulation are:

```
c_mu = -5               (in stan: mus[0])
w_mu = 5                (in stan: mus[1])
corr = 0.75             (in stan: R[1,0] and R[0,1])
c_std = 1               (in stan: cw_sigmas[0])
w_std = 1               (in stan: cw_sigmas[1])
sigma = 0.02    (in stan: sigma)
```

An example posterior:

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
cw_mus[0]	-5.39	0.01	0.26	-5.9	-5.57	-5.4	-5.21	-4.9	324.0	1.0
cw_mus[1]	4.65	0.01	0.21	4.25	4.51	4.64	4.77	5.06	193.0	1.0
.....										
R[0,0]	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	500.0	nan
R[1,0]	0.69	6.4e-3	0.12	0.43	0.63	0.71	0.78	0.87	343.0	1.0
R[0,1]	0.69	6.4e-3	0.12	0.43	0.63	0.71	0.78	0.87	343.0	1.0
R[1,1]	1.0	2.1e-184	8e-17	1.0	1.0	1.0	1.0	1.0	500.0	nan
.....										
cw_sigmas[0]	1.06	9.1e-3	0.17	0.79	0.94	1.04	1.15	1.42	352.0	1.0
cw_sigmas[1]	0.83	7.0e-3	0.13	0.61	0.73	0.82	0.91	1.11	344.0	1.0
sigma	0.02	5.1e-5	1.1e-3	0.02	0.02	0.02	0.02	0.02	500.0	1.01

2.1.2 Stan Code

First, we discuss the cws parameter:

```
parameters {
  vector[N] c;
  vector[N] w;
  ...
}
```

```
transformed parameters {
  vector[L] cws[N];     # N-by-L matrix
  ...
  # Change data type for multi_normal in model block
  for (n in 1:N) {
    cws[n,1] = c[n];
    cws[n,2] = w[n];
  }
}

model {
  ...
  cws ~ multi_normal(cw_mus, S);
  ...
}
```

We define "cws" in the transformed parameters block as an N-by-L matrix which simply grabs values from the "c" and "w" parameters defined in the original parameter block. The reason for this is because the L-dimensional "multi_normal" distribution in the model block must output a vector of size L. This is just a transformation of the data type.

Parameters "c" and "w" don't have explicit priors on them in the model block, because they are distributed according to the multivariate normal through "cws".

```
transformed parameters {
  ...
  # Create covariance matrix, S
  # performs: diagonalize sigmas_cw * R * diagonalize sigmas_cw
  S = quad_form_diag(R, cw_sigmas);
}
```

The "quad_form_diag" function is described on page 379 of the Stan reference:

```
matrix quad_form_diag(matrix m, vector v)
        The quadratic form using the column vector v as a diagonal matrix, i.e.,
        diag_matrix(v) * m * diag_matrix(v).
```

This calculates:

$$\mathbf{S} = \begin{pmatrix} \sigma_c & 0 \\ 0 & \sigma_w \end{pmatrix} \mathbf{R} \begin{pmatrix} \sigma_c & 0 \\ 0 & \sigma_w \end{pmatrix} \tag{7}$$

Exercise 2

Your friend comes to you with the same model, with one line different in the parameters block:

```
vector[L] sigmas;
```

When they run inference on the model, they observe that their inferred correlation values nearly always don't converge. You reduce the number of chains to one and observe the following:

cw_sigmas[0]	-1.07	9.4e-3	0.17	-1.44	-1.17	-1.06	-0.95	-0.8	338.0	1.0
cw_sigmas[1]	-0.85	7.7e-3	0.14	-1.23	-0.94	-0.82	-0.75	-0.64	335.0	1.01
...										
R[0,0]	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	500.0	nan
R[1,0]	0.69	6.4e-3	0.12	0.43	0.63	0.71	0.78	0.87	343.0	1.0
R[0,1]	0.69	6.4e-3	0.12	0.43	0.63	0.71	0.78	0.87	343.0	1.0
R[1,1]	1.0	2.1e-184	8e-17	1.0	1.0	1.0	1.0	1.0	500.0	nan

Your friend tells you that usually, Stan throws errors with negative standard deviations, even if you don't explicitly place a lower bound at 0 on the standard deviation parameters, so he/she didn't bother placing a lower bound on sigmas in this model.

Why do you observe a converged model with negative sigmas, and what's the fix?

3 Bibliography and Additional Resources

Chapter 5 of Statistical Rethinking discusses collinearity.

The Stan reference is available here: http://mc-stan.org/documentation/

Section 21 in the Stan reference also discusses collinearity.