Applied Probabilistic Programming & Bayesian Machine Learning

§3-2: Bayesian Linear Regression

MIT IAP 2017 Max W. Shen

At this point, you should be comfortable using Stan as a tool for Bayesian inference. In this section, we dive into some theoretical considerations regarding assumptions that are implicitly made during Bayesian modeling, particularly with hierarchical models and regression.

An understanding of these topics will allow you to appreciate the potential and limits of Bayesian modeling, as well as recognize scenarios where Bayesian modeling is applicable and where it's not.

1 Bayesian regression

The ordinary linear regression setup assumes the following model:

$$Y \sim N(XW, \sigma^2 I) \tag{1}$$

For any Y_i, $E[Y_i] = X_i W$ - the mean of Y_i depends linearly on its associated features X_i. The observations Y_i are normally distributed about this mean, with variance σ^2.

X is an N-by-D matrix of features, W is a D-by-1 weight vector, and Y is an N-by-1 vector of observations.

1.1 Comparison to Common Approaches

Linear regression is an extraordinarily common machine learning method and has implementations far and wide across many libraries and languages. We take this opportunity to compare to and remotivate the Bayesian approach.

Analytically, model 1 can be solved trivially.

$$arg\min_W ||Y - XW||^2 \tag{2}$$

The optimization objective 2 is equivalent to model 1. Expanding the square, we get $Y^TY - W^TX^TY - Y^TXW + W^TX^TXW$. Since Y^TXW has dimension 1x1, it's equal to its transpose, so we have:

$$arg\min_W Y^TY - 2W^TX^TY + W^TX^TXW \tag{3}$$

First-order optimization via differentiation gives:

$$\begin{aligned} -X^TY + (X^TX)W &= 0 \\ W &= (X^TX)^{-1}X^TY \end{aligned} \tag{4}$$

Libraries such as scikit-learn for Python provide easy-to-use and fast functions that perform this optimization.

However, this optimization setup gives us only a point estimate of final weight vector. The advantage of the Bayesian approach is obtaining a full posterior distribution, from which we can obtain confidence intervals and infinitely simulate outcomes of interest. In addition, Stan makes it easy to augment our model with additional complexity, such as the 2-layer regression example studied in section 1-2, or add complex prior knowledge.

1.2 Identifiability

If two features are collinear, such that columns i and j of X have $w_i x_{n,i} + w_j x_{n,j} = (w_i + w_j) x_{n,i}$ for any $n \in \{1, ..., N\}$, then there are infinite values of w_i, w_j for any particular value of $(w_i + w_j)$. This will lead to identifiability issues and poor inference.

1.3 Conjugate Priors as Pseudo-Observations

Under the ordinary regression setting with $E[Y] = XW$ and normally distributed errors, the act of placing normal priors over the values of the weights has an equivalent correspondence to pseudo-observations. This reflects the fact that the normal distribution is a conjugate prior to itself.

Consider a prior distribution on a particular weight:

$$w_j \sim N(\mu_{w_j}, \sigma^2_{w_j}) \tag{5}$$

where $\mu_{w_j}, \sigma^2_{w_j}$ are known. This is equivalent to an additional observation μ_{w_j} that is predicted using only feature x_j, and the prediction task has a variance of $\sigma^2_{w_j}$.

Specifically, append an additional observation μ_{w_j} to y, append a row of to X with all zeros but a 1 at the j-th column of X, and expand the covariance matrix with a diagonal element of $\sigma^2_{w_j}$ with zeros for the rest of the row and column.

Considering two extremes is useful:

In the limit of no prior information, with $\sigma^2_{w_j} \to \infty$, an additional observation is made with infinite variance which puts no constraint on w_j.

In the limit of perfect prior information, with $\sigma^2_{w_j} = 0$, an additional observation is made with zero variance (infinite confidence), fixing $w_j = \mu_{w_j}$.

2 Extending Bayesian Regression

Under the ordinary Bayesian regression setting (as in model 6), the mean of outcome Y_i is assumed to depend linearly on features X_i, and errors are assumed to be normally distributed.

$$Y \sim N(XW, \sigma^2 I) \tag{6}$$

Various relaxations of this ordinary regression model lead to different fields of Bayesian analysis, each of them well-studied.

2.1 Generalized Linear Models

Generalized linear models (GLMs) assume the following setup:

$$Y \sim N(g^{-1}(XW), \sigma^2 I) \tag{7}$$

such that $g(E[Y_i]) = X_i W$ where $g()$ is called the *link function*. The mean no longer depends linearly on $X_i W$, instead depending linearly on $g^{-1}(X_i W)$.

In standard linear regression, the weight vector W has a straightforward interpretation. In GLMs, W's interpretation becomes more obscured.

A vector of changes in features ΔX_i leads to a change in Y_i as $g^{-1}(g(Y_i) \pm (\Delta X_i)W)$.

Information criteria should not be used to compare different link functions.

2.1.1 Common Link Functions: Linear

Let $E[Y_i] = \mu$.

$$g(\mu) = \mu \tag{8}$$

Link function 8 gives standard linear regression: $g(\mu) = \mu = XW$.

2.1.2 Common Link Functions: Binary / Binomial

Suppose $y_i \sim Bin(n_i, \mu_i)$. A common preprocessing step is to ignore n_i by modeling y_i/n_i instead, which fall between 0 and 1. Let the link function be the *logit transformation*:

$$g(\mu_i) = \log\left(\frac{\mu_i}{1-\mu_i}\right) \tag{9}$$

Link function 9 gives logistic regression. Note that the inverse-logit transforms real numbers produced by XW into the range of 0 and 1. An alternative, common in econometrics, is the probit link:

$$\begin{aligned} g(\mu_i) &= \Phi^{-1}(\mu) \\ \Phi(s) &= \int_{-\infty}^{s} Gaussian(0,1) \end{aligned} \tag{10}$$

Figure 1 compare the logit and probit link functions; they are very similar. One advantage of the probit link function is the ability to replace the Gaussian distribution in the definition of $\Phi()$ with

a t-distribution, making the model less confident by driving all predictions closer towards 0.50. This can act as a form of regularization to improve generalization. This idea is discussed in greater depth in Lecture 2's section on robust models.

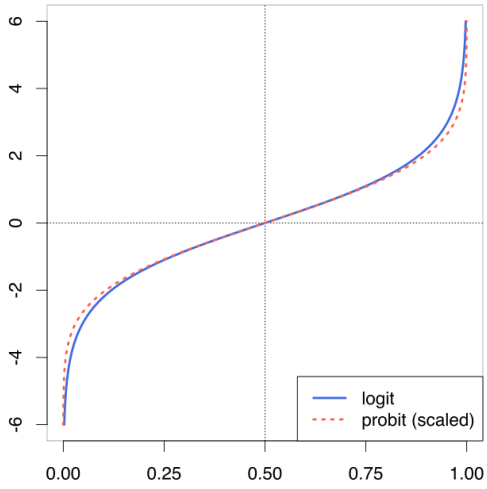

Figure 1: Comparison of Logit and Probit (Wikipedia)

2.1.3 Common Link Functions: Poisson

In the case of counted data, it's natural to use a poisson distribution with mean μ and variance μ. A common link function for Poisson data is:

$$g(\mu) = \log(\mu) \tag{11}$$

Such that $\log(\mu) = XW$.

Exercise 1

Consider the following relation. The second version is preferred because addition is more computationally stable than multiplication.

$$\begin{aligned} y_i &= x_{i1}^{w_1} \times x_{i2}^{w_2} \times ... \times x_{ik}^{w_k} \\ \log(y_i) &= w_1 \log(x_{i1}) + w_2 \log(x_{i2}) + ... + w_k \log(x_{ik}) \end{aligned} \tag{12}$$

What is $g()$?

Exercise 2

Your friend attempted to write model for a GLM with an exponential link function. What is wrong with this model? Your friend says the model's predictions are consistently biased across the input space. Would you expect the model's predictions to be consistently biased over or under the observed outcomes?

```
data {
  int N;
  int D;
  matrix[N, D] X;
  vector<lower=0>[N] Y;
}
parameters {
  vector[D] W;
  real<lower=0> w_sigma;
  real<lower=0> sigma;
}
model {
  log(Y) ~ normal(X*W, sigma);
  W ~ double_exponential(0, w_sigma);
  sigma ~ cauchy(0, 1);
  w_sigma ~ cauchy(0, 1);
}
```

2.2 Non-Linear Models

A straightforward way to handle non-linear models is simply adding non-linear features to X.

For instance, all possible second-degree polynomials are linear to model 13.

$$y \sim N(w_0 + w_1 x + w_2 x^2, \sigma) \tag{13}$$

This approach can become prohibitively expensive when considering nonlinear combinations of multiple variables. For instance, including all pairs of N variables gives us $O(N^2)$ terms because there are $\frac{N(N-1)}{2}$ pairs. In these situations, Bayesian variable selection and shrinkage priors become key for effective generalization, discussed in lecture 4 and 1 respectively.

3 Ordered Logistic Regression in Stan

When data is categorical and ordered, such as a survey question with responses ranging from 1 (strongly disagree) to 7 (strongly agree), we may wish to predict the categorical, ordered responses using a regression setup to include features we believe are predictive. This extends the logistic regression problem from 2 classes to any number of classes.

We provide sample code model_3-2.py and model_3-2.stan to show you how this is done in Stan.

First, we introduce our simulation assumptions:

```
counts = np.array( [15, 20, 10, 30, 25, 35] )
X_means = [-3, -2, -1, 0, 1, 2]
X_std = 0.1
N = sum(counts)
NUM_CLASSES = len(counts)
```

Here, we assume 6 distinct classes, with associated counts for each class. Each class has a single feature, X, which is drawn from a normal distribution with a particular mean and standard deviation. We have N observations in total for the 6 distinct classes.

As an example, the first class would have 15 observations, with 15 values for X drawn from a $Normal(-3, 0.1)$ distribution. The rest of the simulation code creates the variables X and Y in this manner.

3.1 The Ordered Logistic Model

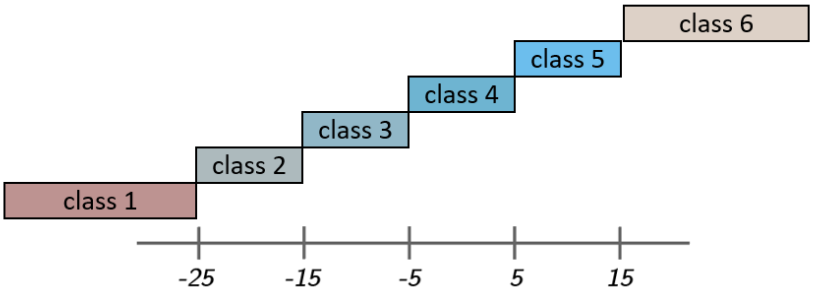

Figure 2: Classifying Samples in our example model

A natural way to set the problem up is to infer intercept values that divide the real number line into subsections correspond to classes. In our case, we have 6 distinct classes, so we want 5 different intercepts. This allows us to work with real numbers, the natural domain of regression, to solve a categorical problem.

In our case, our regression is simply wx_i, so we can find the probability of class 5 using 2 as:

$$p(y_i = \text{class } 5) = p(5 \leq wx_i \leq 15) \tag{14}$$

In general, with $1, ..., K$ classes, we can label the intercepts $1, ..., K-1$, allowing us to rewrite equation 14 as:

$$p(y_i = \text{class } k) = p(wx_i \leq \text{intercept } k) - p(wx_i \leq \text{intercept } k-1) \tag{15}$$

The first and last classes are slightly different:

$$\begin{aligned} p(y_i = \text{class } 1) &= p(wx_i \leq \text{intercept } 1) - 0 \\ p(y_i = \text{class } K) &= 1 - p(wx_i \leq \text{intercept } K-1) \end{aligned} \tag{16}$$

3.2 Stan Modeling

Over in model_3-2.stan, we have:

```
parameters {
  ordered[NUM_CLASSES - 1] intercepts;
  real w;
}

transformed parameters {
  vector[N] phi;
  phi = X * w;
}

model {
  ...
  for (n in 1:N) {
    Y[n] ~ ordered_logistic(phi[n], intercepts);
  }
}
```

Here, we initialize 5 intercepts, mandating that they are increasing in order (which is required for our model to make sense), and our single weight w which is for our single feature x.

The ordered_logistic function encodes the likelihood given intercepts and phi, which we use to store $X * w$.

3.3 Interpreting the Posterior

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
intercepts[0]	-26.0	0.17	3.54	-33.73	-28.12	-25.53	-23.64	-19.72	422.0	1.0
intercepts[1]	-15.0	0.09	2.39	-20.16	-16.46	-14.82	-13.35	-10.87	711.0	1.0
intercepts[2]	-5.81	0.05	1.69	-9.24	-6.92	-5.66	-4.6	-2.9	1012.0	1.0
intercepts[3]	5.52	0.04	1.33	3.33	4.55	5.38	6.39	8.33	1289.0	1.0
intercepts[4]	15.79	0.1	2.34	11.63	14.16	15.54	17.19	21.1	541.0	1.0
w	10.63	0.07	1.42	8.13	9.7	10.51	11.44	13.84	448.0	1.0

We note that our X was centered at $[-3, -2, -1, 0, 1, 2]$, so our samples' ordered-logit scores lay at -30, -20, -10, 0, 10, and 20. Note how the intercepts lay between the clusters of ordered-logit scores, at -25, -15, -5, 5, and 15. Sample classification follows exactly by figure 2 above, where the x-tick marks correspond to the posterior intercept values.

4 Bibliography and Additional Resources

Chapter 14 of BDA3 discusses Bayesian regression.

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

See this paper for more information on the theory of exchangeability.

`http://www.uv.es/~bernardo/Exchangeability.pdf`