PRIORS AND REGULARIZATION

ETHAN LEVIEN

Contents

1. Priors

Models encode our assumptions about how data was generate. Sometimes we want to incorporate vague information, such as “the function describing my data is very smooth and changes roughly on a time-scale of 5 hours". Or, we might want to include many predictors, but to avoid overfitting penalize high values of the predictors. For example, we might believe there is an interaction term, but suspect it is much smaller than the additive terms. This motivates a different approach to statistics, one which treats parameters as themselves random variables.

To see what these means, let’s consider our linear regression model with a single-predictor:

$$Y|X \sim \mathrm{Normal}(aX+b,\sigma_{\epsilon}). \tag{1}$$

This is how we would specify our model previously. a, b and σ_{ϵ} are a fixed numbers, which is “classical" statistics, we pretend we know absolutely nothing about. It’s easy to imagine a situation however, where these are strongly contained, for example, in an autoregressive model we might know that $-1<a<1$. It seems like fitting the regression model without imposing this constraint is not going to get the most out of our data. The remedy in situations like this is to add **priors**. Priors are simply another layer to our model, they tell us the distribution from which the parameters are drawn. In general, for a linear regression model we can write

$$Y|X,a,b,\sigma_{\epsilon} \sim \mathrm{Normal}(aX+b,\sigma_{\epsilon}) \tag{2}$$
$$a,b,\sigma_{\epsilon} \sim \text{Prior-Distribution}(\mathrm{hyperparameters}) \tag{3}$$

The prior distribution can in principle be anything, and we might impose priors on some parameters and not other. The **hyperparameters** are the parameters in the prior distribution. With priors, we can make predictions BEFORE we see the data. These are called **prior predictions**.

At this point there are a couple of questions we need to answer:

- How do we select the prior distribution?
- How do we incorporate priors into our inference?

The answer to the first question is context dependent, but the following exercise will illustrate how this might play out in a concrete example.

Example 1. *Specifying priors for election model*

Exercise 1: *Specifying priors for human height*

Date: April 2022.

2. Bayesian inference

We now address the questions: How do priors play into our inference? That is, how do we incorporate the prior knowledge of the parameters into our *posterior* knowledge. Bayes' theorem tell us exactly how to do this (in theory). If D is a our data and θ is our parameter(s) Bayes' theorem tells us that

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \tag{4}$$

The distributions appearing in Equation (4) have the following names and interpretations:

- $P(\theta|D)$ is the posterior.
- $P(D|\theta)$ is the likelihood
- $P(\theta)$ is the prior distribution
- $P(D)$ is the evidence. It represents the chance we observe the data *unconditional* on the parameters. This can be obtained by marginalizing over the priors.

While in classical statistics, the objective is to determine (estimate) a parameter θ, in Bayesian statistics our objective is to compute the posteior distribution. Once we have this, we can obtain so-called **point estimates**, for example, by taking the average of θ. The posterior is analogous to the sample distribution.

3. Computing the posterior

3.1. **An analytically tractable example.**

In the general, the posterior is difficult or impossible to find a formula for. Instead, we need to relay on numerical methods. However, for the linear regression it turns out we can select the priors so that the posterior is analytically tractable. To see how this works, we start with the inference for a Normal distribution

$$Y \sim \text{Normal}(\mu,\sigma) \tag{5}$$

(we can think of this as a linear regression model with $a = 0$ and $\mu = b$). Recall that the distribution of the data $D = \{(Y_1, X_1), \dots, (Y_N, X_N)\}$ is

$$p(D|\theta) = \prod_i \frac{1}{\sqrt{2\sigma^2\pi}} e^{-(Y_i-\mu)^2/2\sigma^2} \tag{6}$$

$$= \frac{1}{(2\sigma^2\pi)^{N/2}} e^{-\sum_i (Y_i-\mu)^2/2\sigma^2} \tag{7}$$

We can notice that if the priors are very flat, then this is also proportional to the posterior distribution. Let's start by pretending that σ_ϵ^2 is known and take our priors on μ to be Normal as well

$$\mu \sim \text{Normal}(\mu_0, \sigma_\mu) \tag{8}$$

Then the posterior is *proportional to*

$$p(D|\theta)p(\theta) = \frac{1}{(2\sigma^2\pi)^{N/2}} e^{-\sum_i (Y_i-\mu)^2/2\sigma^2} \frac{1}{\sqrt{2\pi\sigma_\mu^2}} e^{-(\mu-\mu_0)^2/2\sigma_\mu^2} \tag{9}$$

$$= \frac{1}{(2\sigma^2\pi)^{N/2}} \frac{1}{\sqrt{2\pi\sigma_\mu^2}} e^{-\sum_i (Y_i-\mu)^2/2\sigma^2 - (\mu-\mu_0)^2/2\sigma_\mu^2} \tag{10}$$

On this surface this looks a bit complicated as a function of μ, but there is a trick: Notice that all the dependence on μ comes from the exponent. We can rewrite this as

$$A\mu^2 + B\mu + C \tag{11}$$

where

$$A = \frac{1}{2}\left(\frac{N}{\sigma^2} + \frac{1}{\sigma_\mu^2}\right) \tag{12}$$

$$B = -\frac{\sum_i Y_i}{\sigma^2} - \frac{\mu_0}{\sigma_\mu^2} \tag{13}$$

$$C = \frac{\sum_i Y_i^2}{2\sigma^2} + \frac{\mu_0^2}{\sigma_\mu^2} \tag{14}$$

If we factor this quadratic equation, we find that is can be written

$$A(\mu - B/2A)^2 + \text{const.} \tag{15}$$

We don't care what the constant terms is, since the it is the first term which tells us the mean and standard deviation of μ. Now observe that

$$-\frac{B}{2A} = \bar{Y}\frac{\sigma_\mu^2}{\sigma_\mu^2 + \sigma^2/N} + \mu_0\frac{\sigma^2/N}{\sigma_\mu^2 + \sigma^2/N} \equiv \mu_b \tag{16}$$

and

$$\frac{1}{A} = 2\frac{\sigma_\mu^2\sigma^2/N}{\sigma_\mu^2 + \sigma^2/N} \equiv 2\sigma_b^2 \tag{17}$$

In particular, we can deduce that

$$\mu|D \sim \text{Normal}\left(\mu_b, \sigma_b\right) \tag{18}$$

where μ_b and σ_b^2 defined above are the mean and variance of the posterior distribution. Despite all the messiness, these actually have very clear interpretations which will give us an intuition above how Bayesian statistics really works. To understand these, think about the following questions:

(1) What happens as $N \to \infty$?
(2) What happens as $\sigma^2 \to 0$ or ∞?
(3) What happens as $\sigma_\mu \to 0$ or ∞?

Example 2. *Relationship between posterior and sample distribution*

3.2. **Markov chain Monte Carlo.** How do we draw samples from the posterior if we can't find a formula? This is actually tricky, and doing it efficiently turns out to be an entire field in applied mathematics. The important thing to recognize is that is is usually easy to find a function that the posterior is proportional to, since from Equation (4),

$$P(\theta|D) \propto P(D|\theta)P(\theta). \tag{19}$$

The challenging part is to find $P(D)$, since this requires us to sum (or integrate) over all possible parameters. So, we can reframe the question of how to generate samples from the posterior as: if we have a function $f(x)$ which is NOT a probability distribution, how can we draw samples from the probability distribution (or density) which is proportional to f? A class of algorithms called Markov Chain Monte Carlo provide an answer to this question.

4. Probabilistic programming in pymc

The “probabilistic programming" package pymc allows us to specify and perform MCMC simulations without worrying (too much) about the underlying details of these algorithms. Here I'll go through an example of inferring whether a coin is biased to give you a sense of the code structure.

```
> # we build a model for the coin
> # we start by simply defining a model using the Model() constructor
> coin_flipping = Model()
>
> # right now coin_flipping is just an "empty" model
> # it has no variables or paramaters.
> # model specifications in pymc3 are wrapped in a with-statement
> # don't overthink this statement, basically within with is where we tell pymc what to do
> with coin_flipping: #
>     # first we define out priors
>     # the model has one parameter p (the probability to get heads)
>     q = Uniform("q", lower=0, upper=1)
>
>     # now we tell pymc3 what the probability distribution of our data is
>     # this is generally what we think of as a our "model" or likelihood
>     y = Bernoulli("y", p=q, observed=flips)
```

Now that we've given pymc3 everything it needs to know about our statistical model.

NOTE: When we specify distributions in pymc we do not use the “np.random“ functions. Those generate samples from a random distribution, while here we are just telling pymc3 what the distributions of parameters and data are.

The next step is to sample from the Posterior distribution using Markov chain Monte Carlo simulations

```
>   # anything that we do with the model happens inside the with statement
>   with coin_flipping:
>     # this will generate samples from the posterior distribution
>     # it may take a minute or so
>     trace = sample(4000,cores =3)
```

The structure of the pymc inference code is:

(1) Tell pymc you are making a model by writing

```
> name_of_model = Model()
```

(2) In the indented code below a with statement tell pymc what you the distributions of your parameters and variables are. These can either be parameters or variables in our model we have data for. This looks like:

```
> with name_of_model:
>   # this defines out prior
>   name_of_parameter = NameOfDistribution("name of variable",parameters)
>   .
>   .
>   .
>   # this defines our likelihood (our model)
>   name_of_variable = NameOfDistribution(args)
```

Generally the distribution functions that specify what distributions for pymc3 to use can take arrays for the parameters and observed data. For example, we can make the mean of the normal distribution a vector and the ith entry will be used as the mean for the ith entry of y. In particuar, for a regression model we can write:

```
> y = Normal("y",mu = a*x_data + b,sigma = sigma,observed = y_data)
```

(3) In the indented code below a with statement tell pymc to generate samples from the posterior distribution. The pymc function for doing this is called sample.

```
> with name_of_model:
>   trace = sample(number of samples)
```

Some things to remember:

- In general, don't do anything inside the with statement besides these things!
- do things in this order
- if you change something in your model or are making a new model do the first step agian

Trace is a python object which saves all the information about the posterior distribution. perhaps most importantly, trace stores a list of values of p. each number in this list is a random sample from the posterior distribution

```
> q_samples = trace["q"]
```

5. Bayesian p-values

6. Least squares and relationship to regulation view

In classical statistics, there is a way incorporate prior information which is conceptually different, but (in many cases) mathematically equivalent to the use of priors. Let's recall that in the context of linear regression, the coefficient estimates $\hat{a}_i$ we obtain minimize the squared residuals. That is, when we fit a linear regression model we are minimizing L where

$$L(\hat{a}) = \sum_i \left(Y_i - \sum \hat{a}_i X_i\right)^2. \tag{20}$$

We call L our loss function. The idea of **regularization** is to impose some additional structure on the $\hat{a}$. For example, we might them not to be too large, and thus, we want $\sum \hat{a}_i^2$ to be small. To achieve this we would add the regularization term

$$L(\hat{a}) = \sum_i \left(Y_i - \sum_j \hat{a}_j X_{i,j}\right)^2 + \lambda \sum \hat{a}_i^2 \tag{21}$$

where $X_{i,j}$ is the jth predictor that goes with the ith data point. The term λ controls how heavily we weight the constraint that $\hat{a}$ should be close to zero. Do you see the connection to regularization and the equation for the posterior of a Normal distribution given by Equation (9)?