Lecture 9.2: Regularization and information criteria

** This lecture is based on chapter 6 of Statistical Rethinking by Richard McElreath. To reproduce some of it, you will need to install the 'rethinking' package from github. The code is below:*

```
install.packages(c('devtools','coda','mvtnorm','loo'))
library(devtools)
install_github("rmcelreath/rethinking")
```

```
library(rstan)
library(shinystan)
library(car)
library(mvtnorm)
library(rethinking)
library(MASS)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
source("../utilityFunctions.R")
```

Last time we discussed over- and underfitting models and began to discuss information theory, divergence, and deviance.

As a reminder, we want to estimate the divergence of our model Q from the true model P (i.e., estimate the deviance). But this would seemingly require that we know the true model P.

- But if we know the true model why do statistical inference in the first place?

Luckily, we only need be concerned with comparing the divergences of different candidates, say Q and R.

- While we don't know P, we can estimate how far apart Q and R are by comparing the average log-probability from each model to get an estimate of the relative distances of each model from the target.
- The absolute magnitude of the values is not interpretable—neither $E \log q_i$ or $E \log r_i$ suggest a good or bad model by themselves.
- Only the difference $E \log q_i - E \log r_i$ informs us about the divergence of each model from the target P.

An approximation of the K-L divergence (and thus *relative* model fit) is the *deviance*:

$$D(Q) = -2 \sum_i \log q_i$$

where q_i is the likelihood of observation i.The -2 is for historical reasons because, in many cases, the distribution of two deviances has a χ^2-distribution.

Because we are in Bayes land, there is uncertainty about parameters; thus there will also be uncertainty about the deviance of a model.

- For any specific parameter values, there is a defined deviance. But a posterior distribution of parameter values begets a posterior distribution of deviances.

Deviance to out-of-sample

Deviance is a nice way to measure the distance of our model from the target. But deviance has the same problem as R^2—it always improves as the model becomes more complex (this will be relaxed when we revisit multi-level modeling).

- Deviance, like R^2, is a measure of retrodictive accuracy, not predictive accuracy.
- What we want is the deviance on new data.

Enter in vs. out of sample: Imagine we have data and use that data to fit a statistical model.

- The data comprise the *training sample*. Parameters are estimated, and then we can imagine using those estimates to predict new outcomes from a new sample, called a *test sample*.

Here is an outline of a thought exercise to explore the distinction between deviance in and out of sample:

1. We have a training sample of size N.
2. Using the training sample, we fit a statistical model and compute the deviance—D_{train}.
3. Now imagine we have another sample of size N from the same population/biological process—the test sample.
4. We will now compute the deviance on the test sample (D_{test}) using our parameter estimates from step 2.

To visualize the results, we will conduct the thought experiment 500 times for five different regression models. The generating model is

$$\begin{aligned} y_i &\sim \text{Normal}(\mu_i, 1) \\ \mu_i &= 0 + 0.15x_{1,i} - 0.4x_{2,i}. \end{aligned}$$

This is a Gaussian outocome y with an intercept $\alpha = 0$ and slopes for two predictors equaling $\beta_1 = 0.15$ and $\beta_2 = -0.4$, respectively.

- The first model will have one free parameter, a linear regression with an unknown mean and $\sigma = 1$.
- Each parameter added adds a predictor variable and beta coefficient.

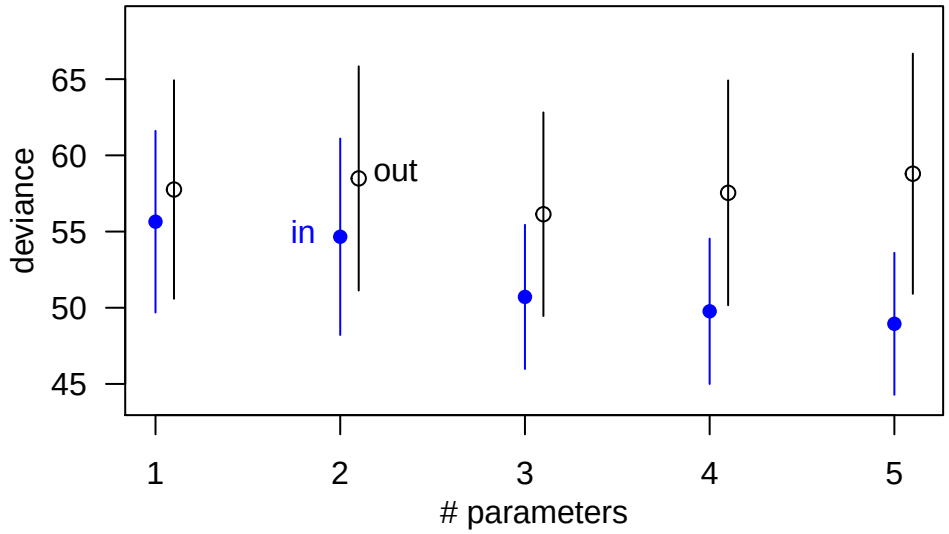
- The "true" model has non-zero coefficients for only the first two predictors, so the true model has 3 parameters total.

We will run these models using the `sim.train.test` function from the rethinking package. Note this takes a few minutes.

- I tried to streamline the code and add it to our `utility.functions.R` script but it took too long.

```
N <- 20
kseq <- 1:5
nSims <- 500

# If you have a mac set this
cores <- 4

dev <- sapply(kseq, function (k) {
#  print(k);
# If you have a mac use this
  r <- mcreplicate( nSims, sim.train.test(N=N, k=k), mc.cores=cores)
# If you don't have a mac use this
#  r <- replicate(nSims, sim.train.test( N=N, k=k ));
  c( mean(r[1,]), mean(r[2,]), sd(r[1,]), sd(r[2,]) )
} )
```

```
par(mar=c(3,3.2,0.1,0.5))
plot(1:5, dev[1,], ylim=c(min(dev[1:2,])-5, max(dev[1:2,]+10)),
  xlim=c(1,5.1), las=1, ann=FALSE, pch=16, col="blue")

mtext(text = "# parameters", side=1, line = 2, cex=1)
mtext(text = "deviance", side=2, line = 2.2, cex=1)

points((1:5)+0.1, dev[2,])
for (i in kseq) {
  IN <- dev[1,i] + c(-1,1)*dev[3,i]
  OUT <- dev[2,i] + c(-1,1)*dev[4,i]
  lines(c(i,i), IN, col="blue")
  lines(c(i,i)+0.1, OUT)
}

text(1.8, 55, "in", col="blue")
text(2.3, 59, "out")
```



In the plot, the blue points/lines show the mean $\pm 1SD$ of the deviance calculated on the training data. Increasing the number of parameters decreases the average deviance.

- Smaller deviances mean a better fit.

But look at the open points and black lines—the out-of-sample deviance.

- While the training deviance gets better with additional parameters, the average test deviance is smallest for three parameters, i.e., the data generating model.
  - as parameters are added, the deviance gets worse.

Note that there is no guarantee that the "true" model will have the smallest average out-of-sample deviance.

- This is illustrated by the 2 parameter model, which does worse on average than the one-parameter model. This is because there is uncertainty with only 20 samples.

Deviance is a measure of predictive accuracy, not truth. The true model is not guaranteed to produce the best predictions, and a false model is not guaranteed to produce bad predictions.

Regularization

So how can we prevent overfitting? One way is to prevent a model from getting too excited about the "training" sample.

- When priors are flat or nearly flat, the likelihood (i.e., the data) run the show because the prior suggests every value (or nearly every value) is equally plausible
- Any particular value therefore has very little prior weight.

We can avoid this by using more "skeptical" priors that slow the rate of learning from the sample.

- The most common way to do this is a *regularizing* or *shrinkage* prior on β coefficients.

- But too much skepticism is bad as well if we underfit the model. So we need to tune the prior so it is mildly skeptical.

Consider the following model:

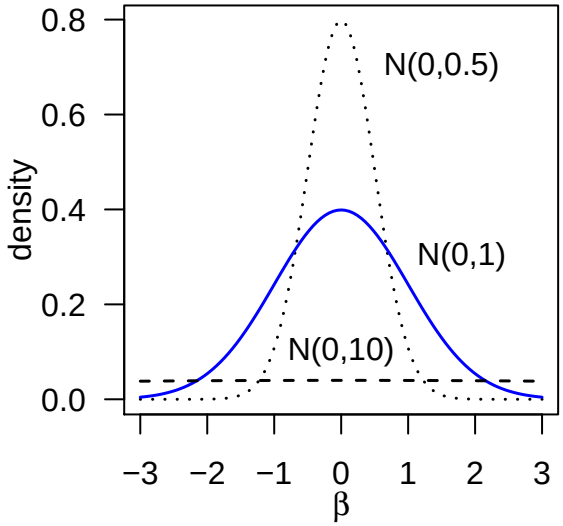
$$\begin{aligned} y_i &\sim \text{Normal}(\mu_i, \sigma) \\ \mu_i &= \alpha + \beta x_i \\ \alpha &\sim \text{Normal}(0, 20) \\ \beta &\sim \text{Normal}(0, 1) \\ \sigma &\sim \text{Cauchy}^+(0, 5) \end{aligned}$$

Assume that the predictor x has been standardized so that it's mean is 0 and SD is 1.

- The prior on α will depend on the scale of y but should be given enough leeway to swing around both positively and negatively so that it has little impact on inference.

The prior on β, however, is narrower and is meant to shrink. The prior $\beta \sim \text{Normal}(0, 1)$ suggests that, before observing the data, our model should be skeptical of values below and above [-2, 2].

- This is because a normal distribution with a standard deviation of 1 assigns only 5% plausability to values above/below 2 SD.
- can interpret as saying that a change in 1 SD in x is unlikely ot produce 2 units change in y.



Here is a figure of three different priors. The blue is $\text{N}(0, 1)$. An even more skeptical prior $\text{N}(0, 0.5)$ is also shown (dotted line), as is an essentially flat prior ($\text{N}(0, 10)$).

How strong to make the prior will depend on the data at hand and the sample size. With more data, even a very strong regularizing prior will be overcome by the likelihood, which is a good thing.

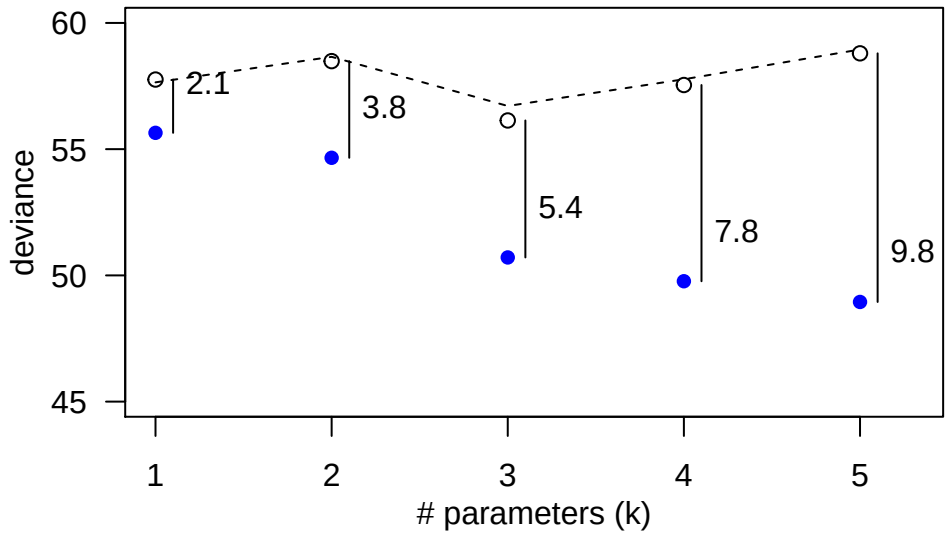
- As more information is added, we need to be less and less skeptical about modeling the noise vs. modeling the process.

- Could also repeatedly *cross-validate*, holding out part of the data, fitting the model, and then using that model on the saved data. But often we don't have enough data to do this.

What we might want as well is a way to use all of our data in a single model and forcast a model's out-of-sample deviance.

Information criteria

```
par(mar=c(3,3.2,0.1,0.5))
plot(1:5, dev[1,], ylim=c(45,60),
  xlim=c(1,5.3), las=1, ann=FALSE, pch=16, col="blue")

mtext(text = "# parameters (k)", side=1, line = 2, cex=1)
mtext(text = "deviance", side=2, line = 2.2, cex=1)

points((1:5), dev[2,])
lines(kseq, dev[1,] + 2*kseq, lty=2)
for (i in kseq) {
  lines(c(i,i)+0.1, c(dev[1,i], dev[2,i]))
  text(i+0.3, dev[2,i] - (dev[2,i]-dev[1,i])+2,
    labels = eval(round(dev[2,i]-dev[1,i],1)))
}
```



Here is a plot of the in-sample (blue) vs. out-of-sample (black) deviances for the simulations we did earlier.

- The lines measure the average distance between the training deviance and test deviance. Notice that each distance is approximately 2x the number of parameters labeled on the x-axis.

- The dashed line shows exactly the blue points + 2x the number of parameters.

This is the phenomenon behind *information criteria*. The most widely known IC is the *Akaike Information Criteria (AIC)*, which provides a simple measure of out-of-sample deviance:

$$\text{AIC} = D_{train} + 2k$$

where k is the number of free parameters estimated. In the above figure, the dashed lines trace out the AIC values of the models.

All information critera aim to provide an approximation of predictive accuracy measured by out-of-sample deviance. AIC is the oldest and most restrictive, being reliable only when:

1. The priors are flat or completely driven by the likelihood;
2. The posterior is approximately multivariate normal;
3. The sample size N is much greater than the number of parameters, k.

Flat priors are not usually the best, as discussed before, so we want something a bit better.

Deviance Information Criterion (DIC)

DIC is quite popular in Bayesian statistics. DIC is essentially a relaxed version of AIC that allows for informative priors.

- However, DIC requires a multivariate normal posterior, and so if the posterior is skewed, then DIC can give misleading inferences.

DIC is calculated from the posterior distribution of training deviance. Given that the parameters have a posterior, now so does the deviance.

We will define D now as the posterior distribution of deviance.

1. Compute deviance for each set of parameter values in the posterior distribution.
    - If we have 4,000 posterior iterations, we would have 4,000 deviance values.
2. Let $\bar{D}$ = the average of D.
3. Let $\hat{D}$ = the deviance calculated at the posterior mean.
    - i.e., calculate the mean of each parameter value then plug those averages into the deviance formula

DIC is calculated as

$$\text{DIC} = \bar{D} + (\bar{D} - \hat{D}) = \bar{D} + p_d.$$

$\bar{D} - \hat{D} = p_d$ is equivalent to the number of parameters used in calculating AIC.

- It is an "effective" number of parameters that measures how flexible the model is in fitting the training sample.
    - More flexible models are more prone to overfitting

- p_d, often called the penalty term, is the expected distance between the in-sample deviance and out-of-sample deviance.

With flat priors, DIC reduces to AIC, but usually p_d is a fraction of the actual number of parameters because of regularizing priors that keep model overfitting in check.

WAIC

WAIC—the Widely Applicable or Watanabe Information Criterion—is another IC that we will use a lot. WAIC is more flexible than DIC because it doesn't require a multivariate Gaussian posterior, and it is often more accurate.

WAIC is *pointwise*. The prediction uncertainty is considered sample-by-sample or case-by-case.

- WAIC handles uncertainty where it matters, for each independent observation, then sums up across observations
- This is useful because often models do a better job estimating some data more than others.

We will define $\Pr(y_i)$ as the average likelihood of observation i.

- Do this by computing the likelihood of y_i for each set of parameters in each iteration of the posterior.
- Then we average the likelihoods for each observation i and sum over all observations. This gives us the *log-pointwise-predictive-density (lppd)*:

$$\text{lppd} = \sum_{i=1}^{N} \log \Pr(y_i)$$

that is, *the log-pointwise-predictive-density is the total, across observations, of the logarithm of the average likelihood of each observation.*

- The lppd is a pointwise analog of deviance averaged over the posterior. If we multiply it by -2, it would be on the same scale of deviance.

Next we need the effective number of parameters p_{WAIC}. Call $var(y_i)$ the variance in log-likelihood for each observation i in the training sample,

$$p_{WAIC} = \sum_{i=1}^{N} var(y_i)$$

Now WAIC is defined as

$$\text{WAIC} = -2(\text{lppd} - p_{WAIC})$$

Also note that because WAIC is pointwise, we can get a WAIC value for each observation. Not only do we get a WAIC value for each observation, we can get the standard error of the WAIC values; thus we get uncertainty in our model comparison estimates.