Applied Probabilistic Programming & Bayesian Machine Learning

§2-2: Information Criteria & Cross-Validation

MIT IAP 2017 Max W. Shen

1 Information Criteria: Introduction

In machine learning, one of the most important questions we have after building a model is: how well does this model generalize to future or unseen data? In other words, has our model learned something *true*? To address this, it is common to split the data into training set, used to train the model, and a test set, used to measure the accuracy of the model.

In general, our model's accuracy on testing data cannot be higher than our model's accuracy on training data. If it is, usually that points to issues in the model, or issues with splitting data into training and testing data (the two sets must be completely statistically independent!). This is known as cross-validation.

Information criteria (IC), such as AIC (Akaike IC), BIC (Bayesian IC), DIC (Deviance IC), and WAIC (Watanabe-Akaike IC), are mathematically convenient formula that play the same role as cross-validation. In fact, all of the information criteria are theoretically motivated by being asymptotically equivalent to leave-one-out cross-validation.

It can be computationally expensive to perform cross-validation since it typically requires retraining your model many many times. Information criteria act as computationally quick estimates of out-of-sample predictive accuracy.

Since information criteria play the same role as cross-validation, we can use information criteria without splitting data into training and testing sets. Consider some data y and some estimated parameters $\hat{\theta}$. We can use the log likelihood of the data, as in equation 1 as a proxy to "accuracy".

$$\log p(y|\hat{\theta}) \tag{1}$$

Equation 1, equivalent to within-sample predictive accuracy, is known as the "Naive information criterion", because it overestimates out-of-sample predictive accuracy.

The popular information criteria take the form of an adjusted within-sample predictive accuracy:

$$\log p(y|\hat{\theta}) - p_{method} \tag{2}$$

In practice, the popular information criteria multiply this quantity by -2 to equate this quantity with something called "deviance", used for historical reasons. **Since we originally desired to maximize the log likelihood of data (explaining the data better), we now wish to minimize the deviance.**

$$-2\log p(y|\hat{\theta}) + 2p_{method} \tag{3}$$

Next, we briefly discuss each of the popular information criteria. **However, our key takeaway for practical purposes is simply to use WAIC as a fast alternative to leave-one-out cross validation.**

1.1 In Practice...

In practice, information criteria are useful for **model comparison**, providing a quantitative means of deciding which model best explains the data. Information criteria can also be useful for combining models together, with the information criterion score acting as a "weight" on how much you should trust one model over another.

Information criteria for model comparison requires that each model was trained on exactly the same data. Removing data will, in general, improve your information criteria scores.

1.2 Akaike Information Criterion (AIC)

$$AIC = -2 \log p(y|\hat{\theta}_{mle}) + 2k \tag{4}$$

where k is the number of parameters in the model. The correction of k adjusts for the expected improvement in the log likelihood using k parameters by chance alone.

AIC uses a point estimate of the parameters, $\hat{\theta}_{mle}$ - the posterior mode - so it is not truly Bayesian.

There exists an adjusted form of AIC (1989) which corrects a bias in AIC when the number of parameters k is similar to the number of datapoints, notated as N. It takes the form:

$$AIC_C = -2 \log p(y|\hat{\theta}_{mle}) + 2k\Big(\frac{N}{N-k+1}\Big) \tag{5}$$

2 Effective Parameters

Parameters are quantities that we attempt to update given some data via inference. Intuitively, some parameters are more meaningful than others. If a parameter is fully constrained by prior information, then our belief about that parameter doesn't change regardless of seeing data. On the other extreme, a parameter can be completely unconstrained by prior knowledge, and determined completely by data.

We can count a parameter as "0" if it's fully constrained, predetermined by the prior, and as "1" if the parameter is fully unconstrained and completely determined by data.

Beyond linear models with flat priors, the number of effective parameters is *less than* the total number of parameters. Since informative priors and hierarchical models (which insert their own kind of prior knowledge) are common in practice, AIC often tends to over-penalize and underestimate the true out-of-sample predictive accuracy.

As an example, consider the model:

$$\begin{aligned} y_i, ..., y_n &\sim N(\theta, 1) \\ \theta &\sim uniform(0, \infty) \end{aligned} \tag{6}$$

Our uniform prior is truncated at 0, so it is informative.

In the case where y is large and positive, our effective number of parameters will be 1, since we estimate θ completely from data.

In the case where y is close to 0, the effective number of parameters will approach 1/2, since half the posterior information comes from data, and half from the prior truncated at 0. In the extreme, $y_i = 0$ for all i, and there would be symmetric and equal posterior probability of θ being positive or negative under a flat prior, but the truncated prior adds information regarding exactly half of the posterior probability.

It turns out the number of effective parameters is a value that is estimated by more advanced information criteria, particularly, DIC and WAIC.

2.1 Deviance Information Criterion (DIC)

$$\begin{aligned} DIC &= -2 \log p(y|\hat{\theta}_{Bayes}) + p_{DIC} \\ p_{DIC} &= 2\Big(\log p(y|\hat{\theta}_{Bayes}) - E(\log p(y|\theta)) \Big) \end{aligned} \tag{7}$$

Here, $\hat{\theta}_{Bayes} = E(\theta|y)$, the posterior mean. Again, this is a point estimate, so DIC is not fully Bayesian.

From the mathematical definition, p_{DIC} can be interpreted as a sort of variance, where the expectation in $-E(\log p(y|\theta))$ is taken over the entire posterior distribution of θ. In practice, since this expectation is infeasible, we simply simulate from the posterior S times and combine to get an expectation:

$$\text{computed } p_{DIC} = 2\Big(\log p(y|\hat{\theta}_{Bayes}) - \frac{1}{S} \sum_{s=1}^{S} (\log p(y|\theta^s)) \Big) \tag{8}$$

The original p_{DIC} is more numerically stable, but it can be negative, so sometimes an alternate definition is used that is guaranteed to be non-negative:

$$\text{alternate } p_{DIC} = 2 \times \text{variance}(\log p(y|\theta)) \tag{9}$$

p_{DIC} can also be interpreted as the effective number of parameters. As an intuitive example, if a parameter is completely constrained by the priors, then over simulations it will retain the same value, and contribute 0 to the variance of the log likelihood across simulations of θ. As expected, this parameter should count as 0, and it does.

2.2 Watanabe-Akaike Information Criteria (WAIC), (2013)

$$\begin{aligned} WAIC &= -2\log p(y) + p_{WAIC} \\ p_{WAIC} &= 2\sum_{i=1}^{n}\Big(\log E(p(y_i|\theta)) - E(\log p(y_i|\theta))\Big) \end{aligned} \tag{10}$$

WAIC uses the predictive probability density for the data $p(y)$, which we can marginalize to incorporate θ in a fully Bayesian manner:

$$WAIC = -2\sum_{i=1}^{n}\log p(y_i|\theta)p(\theta) + p_{WAIC} \tag{11}$$

Similar to DIC, the expectations in p_{WAIC} are taken over all of θ, preserving WAIC's fully Bayesian status. In practice, you would run simulations to calculate the expectation, just like for p_{DIC} in equation 8.

Also similar to DIC, p_{WAIC} can also be interpreted as the variance in the log likelihood over simulations of θ:

$$\text{alternate } p_{WAIC} = \sum_{i=1}^{n}\text{variance}(\log p(y_i|\theta)) \tag{12}$$

p_{WAIC} can also be interpreted as the number of effective parameters in your model.

In practice, WAIC is the current state-of-the-art information criterion, so there isn't much of a reason to use any other criterion if you have the choice.

We note that WAIC is also sometimes written as:

$$\begin{aligned} WAIC &= elpd_{WAIC} + p_{WAIC} \\ elpd_{WAIC} &= -2\log p(y) \end{aligned} \tag{13}$$

You can confirm that this is a simple rewriting of equation 10.

3 Leave-One-Out Cross-Validation (LOO-CV)

In practice, leave-one-out cross-validation is the most preferred manner of assessing out-of-sample predictive accuracy, since you are actually working with an out-of-sample data point. k-fold cross validation is a useful approximation to LOO-CV that can be much more computationall inexpensive than LOO-CV, since LOO-CV, in the worst case, requires you to retrain your model N times for a dataset of size N.

It turns out LOO-CV/LOO can also be used to estimate the number of effective parameters in a model. We defer discussion on this topic and point to this paper for more details:

`http://www.stat.columbia.edu/~gelman/research/unpublished/loo_stan.pdf`

4 "Bayesian" Information Criterion (BIC)

BIC is defined as $-2 \log p(y|\hat{\theta} + k \log n$, for a model with k parameters and a dataset of size n.

BIC has a different goal than AIC, DIC, and WAIC, so we do not suggest using it in practice. Specifically, BIC attempts to estimate $p(y)$ instead of estimating out-of-sample predictive accuracy. Recall that we have been estimating $p(y)$ through simulation, as a quantity we further adjust to get to out-of-sample predictive accuracy.

It is possible for a complex model to predict well and have a low (good) AIC, DIC, and WAIC value, but have a high (poor) BIC score.

5 Example: Tadpole Data

In this dataset, provided as "data_tadpole.csv", studied 48 different "tanks" each containing some density of tadpoles, represented as integers in the **density** column. The main measured quantity is the density of surviving tadpoles after some event, represented as integers in the **surv** column.

Using "str(data)" in R gives:

```
'data.frame': 48 obs. of 5 variables:
$ density : int 10 10 10 10 10 10 10 10 10 10 ...
$ pred : Factor w/ 2 levels "no","pred": 1 1 1 1 1 1 1 1 2 2 ...
$ size : Factor w/ 2 levels "big","small": 1 1 1 1 2 2 2 2 1 1 ...
$ surv : int 9 10 7 10 9 9 10 9 4 9 ...
$ propsurv: num 0.9 1 0.7 1 0.9 0.9 1 0.9 0.4 0.9 ...
```

For now, we'll only focus on the density and surv columns, ignoring the rest. We provide the dataset for you to explore on your own if you wish.

We consider two models for this data, and we'll compare their WAIC. In this folder, you'll find model_2-2.R, model1.stan (describing model 14), and model2.stan (describing model 15). Apologies to Python users! We use a library in R to help calculate WAIC, and to our knowledge no similar library exists yet in Python, so this example is only in R.

Model 1:

$$\begin{aligned} surv_i &\sim binomial(density_i, p_i) \\ logit(p_i) &= \alpha_i \\ \alpha_i &\sim cauchy(0, 1) \end{aligned} \tag{14}$$

Model 2:

$$
\begin{aligned}
surv_i &\sim binomial(density_i, p_i) \\
logit(p_i) &= \alpha_i \\
\alpha_i &\sim normal(\mu, \sigma) \\
\mu &\sim cauchy(0, 1) \\
\sigma &\sim \text{Half-cauchy}(0, 1)
\end{aligned} \tag{15}
$$

In both models, we're using the logit function (the namesake of logistic regression) to convert α_i, a real number, to p_i, a quantity bound between 0 and 1. The inverse-logit function is depicted in figure 1. We do this to ensure that the p_i parameter in the binomial distribution has the correct range. We can interpret α_i as the log-odds of survival, since $logit(0.5) = 0$, so positive α_i indicates a likelihood of surviving above 50%, while negative α_i indicates a likelihood of surviving below 50%. One reason we prefer working with α allows us to use natural prior distributions over the entire number line, such as the normal distribution and Cauchy distribution.

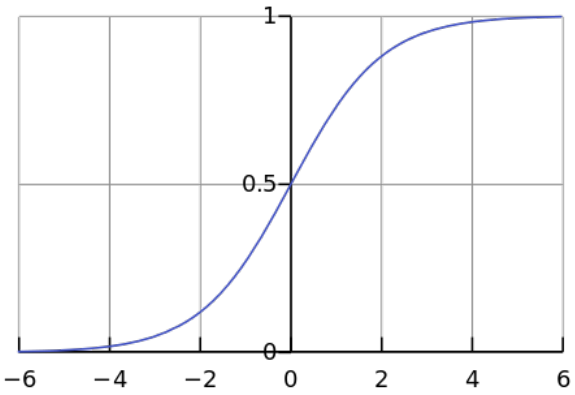

Figure 1: Inverse-Logit Function (Wikipedia)

Exercise 1

The data block has some new notation:

```
data {
  int N;
  int density[N];
  int surv[N];
}
```

We feel sort of bad to ask you to do this, but not enough to stop us. Search in the Stan reference to find what the new notation (int var[N];) means. We point you to chapter 3, "Data Types", on pg. 35.

We use this notation to avoid using a for-loop with the $\sim binomial()$ statement.

5.1 The "Generated Quantities" Block in Stan

The "generated quantities" block is new. We use it to calculate WAIC in a convenient manner.

```
generated quantities {
  vector[N] log_lik;
  for (n in 1:N)
    log_lik[n] = binomial_lpmf(surv[n] | density[n], p[n]);
}
```

Here, we instantiate a vector named "log_lik". The "binomial_lpmf" function is the "log probability mass function" of the observation surv[n] from a binomial distribution parameterized by density[n] (N) and p[n] (θ) - see the Stan reference (pg. 410) for the exact function description.

Since our likelihood function is a binomial distribution, "binomial_lpmf" is used to find the log likelihood of each individual datapoint y_i.

You should recall that this is precisely the term used in WAIC:

$$WAIC = -2\sum_{i=1}^{n} \log p(y_i|\theta)p(\theta) + p_{WAIC} \tag{16}$$

In general, Stan has '..._lpmf' functions for all of its typical probability distributions you have come to know, making it fairly easy to generate a similar "log_lik" object for any other model.

5.2 Handling log_lik in R

All variables defined in the generated quantities block are also returned to python/R through the "fit" object.

Within model_2-2.R, we extract "log_lik" using:

```
fe1 <- extract(fit1);
log_lik1 <- fe1$log_lik
```

Notice here that we do not use "tail(... ,n=1);" - we extract the full matrix instead of just the last iteration's values.

Stan generates a version of "log_lik" (and all other variables defined in the generated quantities block) at each iteration of inference. This is convenient for calculating WAIC, since after warmup our HMC chain is directly sampling from the true posterior distribution. This means we have thousands of iterations of random samples from the true posterior distribution on our parameters. During each iteration the "generated quantities" code runs and provides us with an estimate of our point-wise log likelihood.

As a result, the object "log_lik" will contain thousands of values of our point-wise log likelihood, precisely what is needed to calculate WAIC.

We use the R library 'loo' to assist us with the rest of the calculations. It supports functions waic(), which we discussed, and loo(), which we did not discuss, and computes scores and effective number of parameters for both information criteria given a log_lik object.

```
library('loo');
...
waic(log_lik1);
waic(log_lik2);

loo(log_lik1);
loo(log_lik2);
```

5.3 Interpreting Information Criteria: WAIC

If you run model_2-2.R, you will see some output like this:

```
> waic(log_lik1);
Computed from 4000 by 48 log-likelihood matrix

          Estimate  SE
elpd_waic   -104.5 3.6
p_waic        25.6 1.0
waic         209.1 7.2
Warning message:
37 (77.1%) p_waic estimates greater than 0.4.
We recommend trying loo() instead.

> waic(log_lik2);
Computed from 16000 by 48 log-likelihood matrix

          Estimate  SE
elpd_waic    -99.5 3.7
p_waic        20.5 0.8
waic         199.0 7.3
Warning message:
25 (52.1%) p_waic estimates greater than 0.4.
We recommend trying loo() instead.
```

The rewriting of WAIC we briefly mentioned above explains the names elpd_waic and p_waic:

$$\begin{aligned} WAIC &= elpd_{WAIC} + p_{WAIC} \\ elpd_{WAIC} &= -2 \log p(y) \end{aligned} \tag{17}$$

Model 1 has 48 parameters, while model 2 has 50 parameters. However, we notice that model 1's p_waic is 25.6, while model 2's p_waic is 20.5, even though we added 2 parameters to the model. This directly reflects the additional prior information we added by making model 2 hierarchical. Finally, SE (standard error) is roughly 1, which suggests this difference (25.6 vs. 20.5) can be called "significant".

Model 2's total WAIC is lower at 199.0 compared to model 1's 209.1, indicating that model 2 explains the data better. However, the standard error is roughly 7, so this difference is not that

strong.

Exercise 2

Interpret the output of loo().

You may notice that the inferred number of parameters, "p_loo", is quite different from "p_waic". In practice, it is common for different information criteria to claim a different number of effective parameters. As a result, we suggest interpreting the number of effective parameters in a relative manner (i.e., model2 has fewer effective parameters than model1) rather than an absolute manner.

Exercise 3

Augment some existing Stan code you have with a "generated quantity" block which creates a "log_lik" object. Use R to find the WAIC score and number of effective parameters for your model.

For example, you can use a model that you wrote in Section 1.

6 The Full Circle of Bayesian Modeling

At this point, we have covered the main topics of all three steps below. With information criteria, you now have a quantitative tool for comparing different models you build during model checking. Comparing models to other models can be very useful, though remember to also check your models individually against the data via posterior predictive checks to ensure that each model has learned something reasonable.

1. Write a model
2. Obtain posteriors by running inference
3. Check the model, and repeat from step 1 if needed

If, in the process of checking your model, you realize that your model is misspecified, or is too sensitive to certain assumptions, the next step is to write another model and repeat from step 1.

There are many risks to such an iterative procedure, but unfortunately in practice it lies at the heart of most data scientists.

One approach for avoiding overfitting your data is to retain an extra "second" held-out test set, to be used only with your "final final model" after iterating the 3 steps several times. The extra held-out test set is in addition to your "first" held-out test set which is used during any particular iteration of the 3 steps to assess WAIC, for example.

Similarly, in applied sciences, more data can be gathered to "validate" your model, which for many journals acts as a strong argument that your model is "publication-worthy" - at that point, the rest is up to your specific scientific finding!

7 Model Interpretation

Bayesian models can be appealing for interpretive machine learning.

Any Bayesian model that is composed, in any manner, of well-studied probability distributions (Normal, Binomial, Exponential, etc ... this is the domain we consider in this class) provides a direct, step-by-step method for simulating data from the model. This step-by-step method can be interpreted causally, and gives a process that can be plotted on a timeline. But unfortunately, there is no general reason why this simulation process may correspond to the true underlying process that generated the data.

In addition, nearly all of the historically well-studied probability distributions have interpretations. A bernoulli random variable parameterized by θ corresponds to a coin-flip with probability θ, represents a particularly interpretable case. The Normal distribution, a bit more abstract, still has manageable interpretations: the Central Limit Theorem informs us that sums of independent random variables (no matter their distribution) will become a normal distribution in the limit, and the perspective of information entropy tells us that the normal distribution is the probability distribution that makes the least additional assumptions when only the mean and variance are known for some data.

Fortunately, expressions of model uncertainty via confidence intervals are easily and clearly interpreted.

8 Bibliography and Additional Resources

This paper discusses LOO-CV and WAIC in Stan.

`http://www.stat.columbia.edu/~gelman/research/unpublished/loo_stan.pdf`

Chapter 6 in Statistical Rethinking discusses information criteria.

Chapters 6 and 7 of Bayesian Data Analysis 3 discuss model checking and information criterion.

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