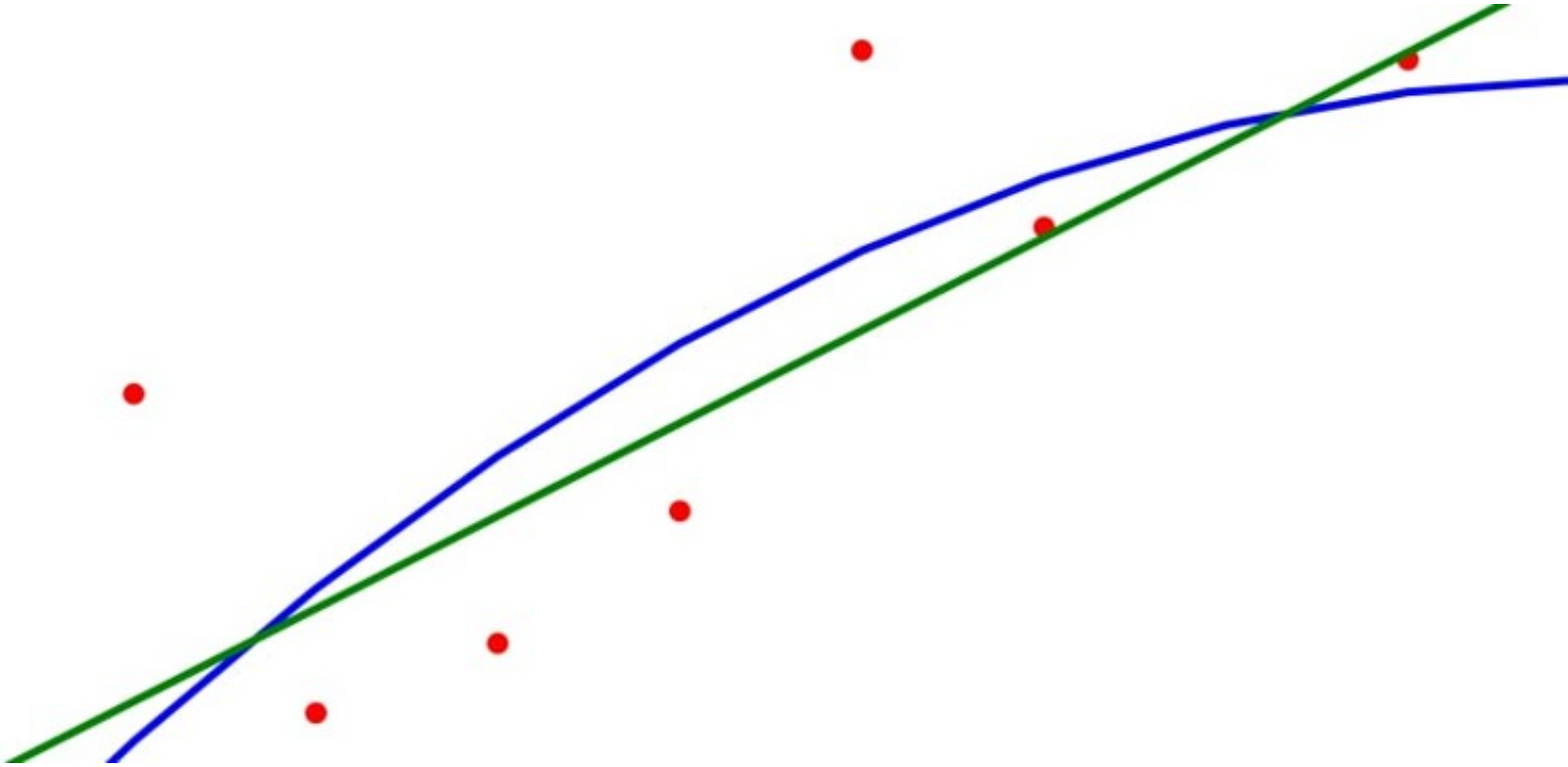


Model Comparison

1. Very Short Introduction to Bayesian Model Comparison
2. Wolf's Dice
3. Wolf's Dice II: What Asymmetry?
4. Laplace Approximation
5. From Laplace to BIC
6. Bayesian Model Testing Comparisons
7. Cross-Validation vs Bayesian Model Comparison
8. Very Short Introduction to Bayesian Model Comparison
9. Wolf's Dice
10. Wolf's Dice II: What Asymmetry?
11. Laplace Approximation
12. From Laplace to BIC
13. Bayesian Model Testing Comparisons
14. Cross-Validation vs Bayesian Model Comparison

Very Short Introduction to Bayesian Model Comparison

At least within Bayesian probability, there is a single unique unambiguously-correct answer to "how should we penalize for model complexity?": calculate the probability of each model, given the data. This is Hard to compute in general, which is why there's a whole slew of of other numbers which approximate it in various ways.

Here's how it works. Want to know whether model 1 or model 2 is more consistent with the data? Then compute $P[model_1|data]$ and $P[model_2|data]$. Using Bayes' rule:

$$P[model_i|data] = \frac{1}{Z} P[data|model_i] P[model_i]$$

where Z is the normalizer. If we're just comparing two models, then we can get rid of that annoying Z by computing odds for the two models:

$$\frac{P[model_1|data]}{P[model_2|data]} = \frac{P[data|model_1]}{P[data|model_2]} \frac{P[model_1]}{P[model_2]}$$

In English: posterior relative odds of the two models is equal to prior odds times the ratio of likelihoods. That likelihood ratio $\frac{P[data|model_1]}{P[data|model_2]}$ is the Bayes factor: it directly describes the update in the relative odds of the two models, due to the data. Calculating the Bayes factor - i.e. $P[data|model_i]$ for each model - is the main challenge of Bayesian model comparison.

Example

20 coin flips yield 16 heads and 4 tails. Is the coin biased?

Here we have two models:

- Model 1: coin unbiased
- Model 2: coin has some unknown probability θ of coming up heads (we'll use a uniform prior on θ for simplicity)

The second model has one free parameter (the bias) which we can use to fit the data, but it's more complex and prone to over-fitting. When we integrate over that free parameter, it will fit the data poorly over most of the parameter space - thus the "penalty" associated with free parameters in general.

In this example, the integral is exactly tractable (it's a dirichlet-multinomial model), and we get:

- $P[data|model_1] = \binom{20}{16} (\frac{1}{2})^{16} (\frac{1}{2})^4 = 0.0046$

- $P[\text{data}|\text{model}_2] = \int_\theta (20)(\theta)^{16}(1-\theta)^4 d\theta = 0.048$

So the Bayes factor is (.048)/(.0046) ~ 10, in favor of a biased coin. In practice, I'd say unbiased coins are at least 10x more likely than biased coins in day-to-day life a priori, so we might still think the coin is unbiased. But if we were genuinely unsure to start with, then this would be pretty decent evidence in favor.

Wolf's Dice

Around the mid-19th century, Swiss astronomer Rudolf Wolf rolled a pair of dice 20000 times, recording each outcome. Here is his data:

		White Die						
		1	2	3	4	5	6	Total
Red Die	1	547	587	500	462	621	690	3407
	2	609	655	497	535	651	684	3631
	3	514	540	468	438	587	629	3176
	4	462	507	414	413	509	611	2916
	5	551	562	499	506	658	672	3448
	6	563	598	519	487	609	646	3422
	Total	3246	3449	2897	2841	3635	3932	20000

I've taken the data from Jaynes, who uses it as an example for maximum-entropy methods. We'll use it for several examples, including some of Jaynes' models.

The most interesting fact about Wolf's dice data is that some faces come up significantly more often than others. As a quick back-of-the-envelope check, we can see this by calculating the expected number of times a face should come up on one die ($20000 * \frac{1}{6} \approx 3333.3$), and the standard deviation of this number ($\sqrt{20000 * (\frac{1}{6}) * (1 - \frac{1}{6})} \approx 52.7$). A quick glance at the data shows that the column- and row-totals differ from their expected value by roughly 2 to 6 standard deviations, an error much larger than we'd expect based on random noise.

That, however, is an ad-hoc test. The point of this sequence is to test our hypotheses from first principles, specifically the principles of probability theory. If we want to know which of two or more models is correct, then we calculate the probability of each model given the data.

White Die: Biased or Not?

Let's start with just Wolf's white die. Our two models are:

- Model 1: all faces equally probable
- Model 2: each face i has its own probability p_i (we'll use a uniform prior on p_i for now, to keep things simple)

Our data is the bottom row in the table above, and our goal is to calculate $P[\text{model}_i|\text{data}]$ for each model.

The first step is Bayes' rule:

$$P[\text{model}_i|\text{data}] = \frac{1}{Z}P[\text{data}|\text{model}_i]P[\text{model}_i]$$

Here's what those pieces each mean:

- $P[\text{model}_i]$ is our prior probability for each model - for simplicity, let's say the two are equally likely a priori.

- Z is the normalizer, chosen so that the posterior probabilities sum to one: $P[\text{model}_1|\text{data}] + P[\text{model}_2|\text{data}] = 1$ (implicitly, we assume one of the two models is "correct").
- $P[\text{data}|\text{model}_i]$ is computed using model_i

... so the actual work is in calculating $P[\text{data}|\text{model}_i]$.

Model 1: Unbiased

For the first model, $P[\text{data}|\text{model}_1]$ is a standard probability problem: given $n = 20000$ unbiased die rolls, what's the probability of $n_1 = 3246$ 1's, $n_2 = 3449$ 2's, etc? This is a multinomial distribution; the answer is

$$P[n_1 \dots n_6|\text{model}_1] = \frac{n!}{n_1! \dots n_6!} p_1^{n_1} \dots p_6^{n_6} = \frac{20000!}{3246! \dots 3312!} \left(\frac{1}{6}\right)^{20000} \approx 6.3 * 10^{-70}$$

Note the symmetry factor with the factorials: we're computing the probability of the observed *counts*, not the probability of a particular string of outcomes, so we have to add up probabilities of all the outcomes with the same counts. The same factor will appear in model 2 as well, for the same reason.

Model 2: Biased

The second model is a bit more complicated. If we knew the probabilities for each face $p_1 \dots p_6$ then we could use the multinomial distribution formula, same as model 1. But since we don't know the p's, we need to integrate over possible values:

$$P[\text{data}|\text{model}_2] = \int_p P[\text{data}|p] dP[p] = \int_p \frac{n!}{n_1! \dots n_6!} p_1^{n_1} \dots p_6^{n_6} dP[p]$$

Here $dP[p]$ is our prior distribution on p - in this case uniform, i.e. a dirichlet distribution with parameter $\alpha = 1$. This is a somewhat nontrivial integral: the constraint $\sum_i p_i = 1$ means that we're integrating over a five-dimensional surface in six dimensions. Fortunately, other people have already solved this integral in general: it's the very handy dirichlet-multinomial distribution. With $\alpha = 1$ the result is particularly simple; the integral comes out to:

$$\frac{n!}{n_1! \dots n_k!} \int_p p_1^{n_1} \dots p_k^{n_k} dP[p|\alpha = 1] = \frac{n!(k-1)!}{(n+k-1)!}$$

We'll use this formula quite a bit in the next post. For now, we're using $k = 6$ outcomes, so we get

$P[data|model_2] = \int_p P[data|p]dP[p] = \frac{20000!}{20005!} \approx 3.7 * 10^{-20}$

So the data is around 10^{50} times more likely under this model than under the unbiased-die model.

Probability of Bias

Last step: let's go back to where we started and compute the posterior probabilities of each model. We plug $P[data|model_i]$ back into Bayes' rule. Each model had a prior probability of 0.5, so we compute the normalizer Z as:

$Z = P[data|model_1]P[model_1] + P[data|model_2]P[model_2] \approx 1.9 * 10^{-20}$

so

$P[model_1|data] = \frac{1}{Z} P[data|model_1]P[model_1] \approx 1.7 * 10^{-50}$

$P[model_2|data] \approx 1.0 - 1.7 * 10^{-50}$

A few comments on this result...

First, we have pretty conclusively confirmed that the faces are not equally probable, given this data.

Second, the numbers involved are REALLY BIG - ratios on the order of 10^{50}. This is par for the course: since independent probabilities multiply, probabilities tend to be roughly exponential in the number of data points. One side effect is that, as long as we have a reasonable amount of data, the priors don't matter much. Even if we'd thought that an unbiased die was a thousand times more likely than a biased die a priori, those huge exponents would have completely swamped our prior.

Playing around with α will reveal that the same applies to our prior distribution for p: the result is not very sensitive to changes in the prior. A word of caution, however: priors over unknown parameters become more important in high-dimensional problems.

Third, notice that there was no maximization and no extra parameters to twiddle. Once the models were specified, we had zero choices to make, zero degrees of freedom to play with. Any "free parameters" - i.e. p - have a prior over which we integrate. That integral is the hard part: as the models get larger and more complicated, we need to evaluate hairy high-dimensional integrals. The problem is not just NP-complete, it's #P-complete. In practice, approximations are used instead, including the entire range of hypothesis tests and model comparison tests - that's where maximization enters the picture. We'll talk about some of those later, especially about when they're likely to work or not work.

Finally, note that we compared a model which is conceptually "compatible" with the data (i.e. capable of generating roughly the observed frequencies), to one which is conceptually "incompatible" (i.e. the observed frequencies are way outside the expected range). A more interesting case is to consider two models which are both "compatible". In that case, we'd want to use some kind of complexity penalty to say that the more complex model is less

likely - otherwise we'd expect overfit. In the next post, we'll revisit Wolf's dice with a couple models from Jaynes, and see how P[data|model] "penalizes" overly-complicated models.

Wolf's Dice II: What Asymmetry?

In the previous post, we looked at Rudolph Wolf's data on 20000 rolls of a pair of dice. Specifically, we looked at the data on the white die, and found that it was definitely biased. This raises an interesting question: what biases, specifically, were present? In particular, can we say anything about the physical asymmetry of the die? Jaynes addressed this exact question; we will test some of his models here.

Elongated Cube Models

Jaynes suggests that, if the die were machined, then it would be pretty easy to first cut an even square along two dimensions. But the cut in the third dimension would be more difficult; getting the length to match the other two dimensions would be tricky. Based on this, we'd expect to see an asymmetry which gives two opposite faces (1 & 6, 2 & 5, or 3 & 4) different probabilities from all the other faces.

Here's what the model looks like for the 1 & 6 pair:

- 1 & 6 each have the same probability $\frac{p}{2}$
- 2, 3, 4 & 5 each have the same probability $\frac{1-p}{4}$
- Uniform prior on p (i.e. dirichlet with $\alpha = 1$)

Let's call this model$_{1,6}$.

I will omit the details of calculations in this post; readers are welcome to use them as exercises. (All the integrals can be evaluated using the dirichlet-multinomial $\alpha=1$ formula from the previous post.) In this case, we find

$$P[data|model_{1,6}] = \frac{n!}{(n+1)!} \left(\frac{1}{2}\right)^{n_1+n_6} \left(\frac{1}{4}\right)^{n_2+n_3+n_4+n_5} \approx 2.2 * 10^{-59}$$

For the other two opposite face pairs, we get:

- 2,5: $1.4 * 10^{-63}$
- 3,4: $8.5 * 10^{-29}$

... sure enough, an asymmetry on the 3,4 axis goes a very long way toward explaining this data.

Recall from the previous post that the unbiased model gave a marginal likelihood P[data|model] around 10^{-70}, and the biased model with separate probabilities for each face gave around 10^{-20}. So based on the data, our 3,4 model is still about a billion times less probable than the full biased model (assuming comparable prior probabilities for the two models), but it's getting relatively close - probabilities naturally live on a log scale. It looks like the 3-4 asymmetry is the main asymmetry in the data, but some other smaller asymmetry must also be significant.

Just for kicks, I tried a model with a different probability for each pair of faces, again with uniform prior on the p's. That one came out to $1.7 * 10^{-30}$ - somewhat worse than the 3,4 model. If you're used to traditional statistics, this may come as a surprise: how can a strictly more general model have *lower* marginal likelihood P[data|model]? The answer is that, in traditional statistics, we'd be looking for the unobserved parameter values p with the *maximum* likelihood P[data|model, p] - of course a strictly more general model will have a maximum likelihood value at least as high. But when computing P[data|model], we're integrating over the unobserved parameters p. A more general model has more ways to be wrong; unless it's capturing some important phenomenon, a smaller fraction of the parameter space will have high P[data|model, p]. We'll come back to this again later in the sequence.

Pip Asymmetry Model

Jaynes' other main suggestion was that the pips on the die are asymmetric - i.e. there's less mass near the 6 face than the 1 face, because more pips have been dug out of the 6 face.

As a first approximation to this, let's consider just the asymmetry between 1 and 6 - the pair with the highest pip difference. We'll also keep all the structure from the 3,4 model, since that seems to be the main asymmetry. Here's the model:

- 3 & 4 have the same probability $\frac{p}{2}$, as before
- 2 & 5 have the same probability $\frac{1-p}{4}$, as before
- 1 & 6 *together* have probability $\frac{1-p}{2}$, same as 3 and 5 together, but their individual probabilities may be different. Conditional on rolling either a 1 or 6, 1 comes up with probability p' and 6 with probability $(1-p')$
- Both p and p' have uniform priors

The conditional parameterization for 1 & 6 is chosen to make the math clean.

Let's call this $\text{model}_{3,4+\text{pip}}$. Marginal likelihood:

$$P[\text{data}|\text{model}_{3,4+\text{pip}}] = \frac{(n_1+1)!}{(n_1+n_6+1)!}\left(\frac{(n_3+n_4)!}{n_3!n_4!}\left(\frac{1}{2}\right)^{n_3+n_4}\right) *$$

$$\left(\frac{(n_1+n_2+n_5+n_6)!}{n_1!n_2!n_5!n_6!}\left(\frac{1}{2}\right)^{n_1+n_6}\left(\frac{1}{4}\right)^{n_2+n_5}\right)\left(\frac{(n_1+n_6+1)!}{(n_1+n_6+1)!}\right) \approx 2.3 * 10^{-16}$$

... and now we have a model which solidly beats separate probabilities for each face!

(I also tried a pip model by itself, without the 3,4 asymmetry. That one wound up at $2.1 * 10^{-70}$ - almost as bad as the full unbiased model.)

We can also go one step further, and assume that the pip difference also causes 2 and 5 to have slightly different probabilities. This model gives $P[\text{data}|\text{model}] \approx 3.9 * 10^{-17}$ - a bit lower than the model above, but close enough that it still gets significant posterior probability (about $\frac{3.9*10^{-17}}{3.9*10^{-17}+2.3*10^{-16}} = 14\%$ assuming equal priors; all the other models we've seen have near-zero posterior assuming equal priors). So based on the data, the model with just the 1-6 pip difference is a bit better, but we're not entirely sure. My guess is that a fancier model could significantly beat both of these by predicting that the effect of a pip difference scales with the number of pips, rather than just using whole separate parameters for the 1-6 and 2-5 differences. But that would get into hairier math, so I'm not going to do it here.

To recap, here's what $\text{model}_{3,4+\text{pip}}$ says:

- 3 and 4 have the same probability, but that probability may be different from everything else
- 2 and 5 have the same probability, and 1 and 6 together have the same probability as 2 and 5, but 1 and 6 have different probabilities.

That's it; just two "free parameters". Note that the full biased model, with different probabilities for each face, is strictly more general than this - any face probabilities p which are compatible with $\text{model}_{3,4+\text{pip}}$ are also compatible with the full biased model. But the full biased model is compatible with *any* face probabilities p; $\text{model}_{3,4+\text{pip}}$ is not compatible with all possible p's. So if we see data which matches the p's compatible with $\text{model}_{3,4+\text{pip}}$, then that must push up our posterior for $\text{model}_{3,4+\text{pip}}$ relative to the full unbiased model - $\text{model}_{3,4+\text{pip}}$ makes a stronger prediction, so it gets more credit when it's right. The result: less flexible models which are consistent with the data will get higher posterior probability. The "complexity penalty" is not explicit, but implicit: it's just a natural consequence of conservation of expected evidence.

Next post we'll talk about approximation methods for hairy integrals, and then we'll connect all this to some common methods for scoring models.

Laplace Approximation

The last couple posts compared some specific models for 20000 rolls of a die. This post will step back, and talk about more general theory for Bayesian model comparison.

The main problem is to calculate P[data|model] for some model. The model will typically give the probability of observed data x (e.g. die rolls) based on some unobserved parameter values θ (e.g. the p's in the last two posts), along with a prior distribution over θ. We then need to compute

$$P[\text{data}|\text{model}] = \int_\theta P[\text{data}|\theta] dP[\theta]$$

which will be a hairy high-dimensional integral.

Some special model structures allow us to simplify the problem, typically by factoring the integral into a product of one-dimensional integrals. But in general, we need some method for approximating these integrals.

The two most common approximation methods used in practice are Laplace approximation around the maximum-likelihood point, and MCMC (see e.g. here for application of MCMC to Bayes factors). We'll mainly talk about Laplace approximation here - in practice MCMC mostly works well in the same cases, assuming the unobserved parameters are continuous.

Laplace Approximation

Here's the idea of Laplace approximation. First, posterior distributions tend to be very pointy. This is mainly because independent probabilities multiply, so probabilities tend to scale exponentially with the number of data points. Think of the probabilities we calculated in the last two posts, with values like 10^{-70} or 10^{-20} - that's the typical case. If we're integrating over a function with values like that, we can basically just pay attention to the region around the highest value - other regions will have exponentially small weight.

Laplace' trick is to use a second-order approximation within that high-valued region. Specifically, since probabilities naturally live on a log scale, we'll take a second order-approximation of the log likelihood around its maximum point. Thus:

$$\int_\theta e^{\ln P[\text{data}|\theta]} dP[\theta] \approx \int_\theta e^{\ln P[\text{data}|\theta_{max}] + \frac{1}{2}(\theta-\theta_{max})^T \left(\frac{d^2 \ln P}{d\theta^2}\Big|_{\theta_{max}}\right)(\theta-\theta_{max})} dP[\theta]$$

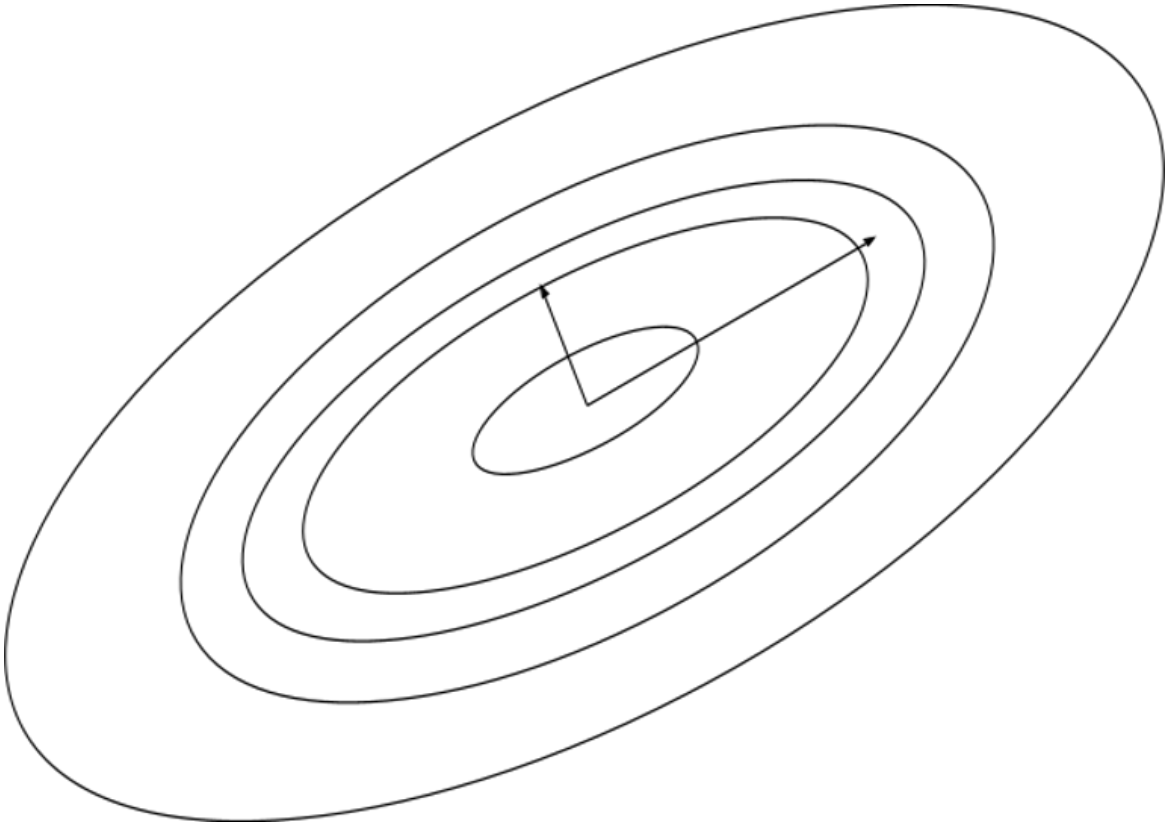
If we assume that the prior dP[θ] is uniform (i.e. dP[θ] = dθ), then this looks like a normal distribution on θ with mean θ_{max} and variance given by the inverse Hessian matrix of the log-likelihood. (It turns out that, even for non-uniform dP[θ], we can just transform θ so that the prior looks uniform near θ_{max}, and transform it back when we're done.) The result:

$$\int_\theta e^{\ln P[\text{data}|\theta]} dP[\theta] \approx P[\text{data}|\theta_{max}] p[\theta_{max}] (2\pi)^{\frac{k}{2}} \det\left(-\frac{d^2 \ln P}{d\theta^2}\Big|_{\theta_{max}}\right)^{-\frac{1}{2}}$$

Let's walk through each of those pieces:

- $P[\text{data}|\theta_{max}]$ is the usual maximum likelihood: the largest probability assigned to the data by any particular value of θ.
- $p[\theta_{max}]$ is the prior probability density of the maximum-likelihood θ point.
- $(2\pi)^{\frac{k}{2}}$ is that annoying constant factor which shows up whenever we deal with normal distributions; k is the dimension of θ.
- $\det(-\frac{d^2 \ln p}{d\theta^2}|_{\theta_{max}})$ is the determinant of the "Fisher information matrix"; it quantifies how wide or skinny the peak is.

A bit more detail on that last piece: intuitively, each eigenvalue of the Fisher information matrix tells us the approximate width of the peak in a particular direction. Since the matrix is the inverse variance (in one dimension $\frac{1}{\sigma^2}$) of our approximate normal distribution, and "width" of the peak of a normal distribution corresponds to the standard deviation σ, we use an inverse square root (i.e. the power of $-\frac{1}{2}$) to extract a width from each eigenvalue. Then, to find how much volume the peak covers, we multiply together the widths along each direction - thus the determinant, which is the product of eigenvalues.



Why do we need eigenvalues? The diagram above shows the general idea: for the function shown, the two arrows would be eigenvectors of the Hessian $\frac{d^2 \ln p}{d\theta^2}$ at the peak. Under a second-order approximation, these are principal axes of the function's level sets (the ellipses

in the diagram). They are the natural directions along which to measure the width. The eigenvalue associated with each eigenvector tells us the width, and then taking their product (via the determinant) gives a volume. In the picture above, the determinant would be proportional to the volume of any of the ellipses.

Altogether, then, the Laplace approximation takes the height of the peak (i.e. $P[data|\theta_{max}]p[\theta_{max}]$) and multiplies by the volume of θ-space which the peak occupies, based on a second-order approximation of the likelihood around its peak.

Laplace Complexity Penalty

The Laplace approximation contains our first example of an explicit complexity penalty.

The idea of a complexity penalty is that we first find the maximum log likelihood $\ln P[data|\theta_{max}]$, maybe add a term for our θ-prior $\ln p[\theta_{max}]$, and that's the "score" of our model. But more general models, with more free parameters, will always score higher, leading to overfit. To counterbalance that, we calculate some numerical penalty which is larger for more complex models (i.e. those with more free parameters) and subtract that penalty from the raw score.

In the case of Laplace approximation, a natural complexity penalty drops out as soon as we take the log of the approximation formula:

$$\ln P[data|model] \approx \ln P[data|\theta_{max}] + \ln p[\theta_{max}] + \frac{k}{2}\ln(2\pi) - \frac{1}{2}\ln\det\left(-\frac{d^2 \ln P}{d\theta^2}\Big|_{\theta_{max}}\right)$$

The last two terms are the complexity penalty. As we saw above, they give the (log) volume of the likelihood peak in θ-space. The wider the peak, the larger the chunk of θ-space which actually predicts the observed data.

There are two main problems with this complexity penalty:

- First, there's the usual issues with approximating a posterior distribution by looking at a single point. Multimodal distributions are a problem, insufficiently-pointy distributions are a problem. These problems apply to basically any complexity penalty method.
- Second, although the log determinant of the Hessian can be computed via backpropagation and linear algebra, that computation takes $O(k^3)$. That's a lot better than the exponential time required for high-dimensional integrals, but still too slow to be practical for large-scale models with millions of parameters.

Historically, a third issue was the math/coding work involved in calculating a Hessian, but modern backprop tools like Tensorflow or autograd make that pretty easy; I expect in the next few years we'll see a lot more people using a Laplace-based complexity penalty directly.

The $O(k^3)$ runtime remains a serious problem for large-scale models, however, and that problem is unlikely to be solved any time soon: a linear-time method for computing the Hessian log determinant would yield an $O(n^2)$ matrix multiplication algorithm.

From Laplace to BIC

The previous post outlined Laplace approximation, one of the most common tools used to approximate hairy probability integrals. In this post, we'll use Laplace approximation to derive the Bayesian Information Criterion (BIC), a popular complexity penalty method for comparing models with more free parameters to models with fewer free parameters.

The BIC is pretty simple:

- Start with the maximum log likelihood $\ln P[\text{data}|\theta_{max}]$
- Subtract $\frac{k}{2}\ln N$ to penalize for complexity, where N is the number of data points and k is the number of free parameters (i.e. dimension of θ).

Thus: $\ln P[\text{data}|\theta_{max}] - \frac{k}{2}\ln N$. Using this magic number, we compare any two models we like.

Let's derive that.

BIC Derivation

As usual, we'll start from P[data|model]. (Caution: don't forget that what we really care about is P[model|data]; we can jump to P[data|model] only as long as our priors are close enough to be swamped by the evidence.) This time, we'll assume that we have N independent data points x_i, all with the same unobserved parameters - e.g. N die rolls with the same unobserved biases. In that case, we have

$$P[\text{data}|\text{model}] = \int_\theta P[\text{data}|\theta]dP[\theta] = \int_\theta \prod_{i=1}^{N} P[x_i|\theta]p[\theta]d\theta$$

Next, apply Laplace approximation and take the log.

$$\ln P[\text{data}|\text{model}] \approx \sum_i \ln P[x_i|\theta_{max}] + \ln p[\theta_{max}] + \frac{k}{2}\ln(2\pi) - \frac{1}{2}\ln\det(H)$$

where the Hessian matrix H is given by

$$H = \frac{d^2}{d\theta^2}\ln P[\text{data}|\theta]|_{\theta_{max}} = \sum_i \frac{d^2}{d\theta^2}\ln P[x_i|\theta]|_{\theta_{max}}$$

Now for the main trick: how does each term scale as the number of data points N increases?

- The max log likelihood $\sum_i P[x_i|\theta_{max}]$ is a sum over data points, so it should scale roughly proportionally to N.
- The prior density and the $\frac{k}{2}\ln(2\pi)$ are constant with respect to N.
- H is another sum over data points, so it should also scale roughly proportionally to N.

Let's go ahead and write H as $N * (\frac{1}{N}H)$, to pull out the N-dependence. Then, if we can remember how determinants scale:

$$\ln\det(N * (\tfrac{1}{N}H)) = \ln\det(\tfrac{1}{N}H) + k * \ln N$$

so we can re-write our Laplace approximation as

$$\ln P[\text{data}|\text{model}] \approx \sum_i \ln P[x_i|\theta_{max}] + p[\theta_{max}] + \tfrac{k}{2}\ln(2\pi) - \tfrac{1}{2}\ln\det(\tfrac{1}{N}H) - \tfrac{k}{2}\ln N$$

$$= \ln P[\text{data}|\theta_{max}] - \tfrac{k}{2}\ln N + O(1)$$

where O(1) contains all the terms which are roughly constant with respect to N. The first two terms are the BIC.

In other words, the BIC is just the Laplace approximation, but ignoring all the terms which don't scale up as the number of data points increases.

When Does BIC Work?

What conditions need to hold for BIC to work? Let's go back through the derivation and list out the key assumptions behind our approximations.

- First, in order to jump from P[model|data] to P[data|model], our models should have roughly similar prior probabilities P[model] - i.e. within a few orders of magnitude.
- Second, in order for any point approximation to work, the posterior parameter distribution needs to be pointy and unimodal - most of the posterior probability mass must be near θ_{max}. In other words, we need enough data to get a precise estimate of the unobserved parameters.
- Third, we must have N large enough that $\frac{k}{2}\ln N$ (the smallest term we're keeping) is much larger than the O(1) terms we're throwing away.

That last condition is the big one. BIC is a large-N approximation, so N needs to be large for it to work. How large? That depends how big $\ln\det(\frac{1}{N}H)$ is - N needs to be exponentially larger than that. We'll see an example in the next post.

Next post will talk more about relative advantages of BIC, Laplace, and exact calculation for comparing models. We'll see a concrete example of when BIC works/fails.

Bayesian Model Testing Comparisons

We've now seen three different methods for model comparison:

- Exact calculation of posterior probabilities of each model via P[data|model]
- Laplace approximation
- Bayesian Information Criterion (BIC)

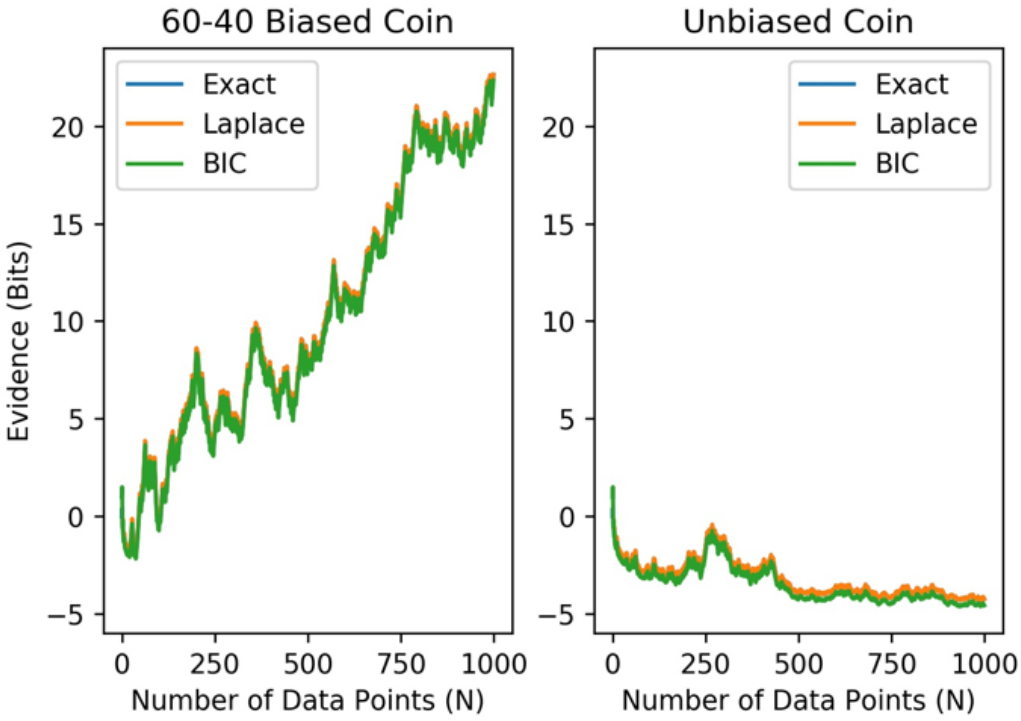
Based on the derivations, we can make some predictions:

- Laplace is a second-order approximation of the exact integral around the maximum-likelihood estimate. It should work well when the integral is dominated by one very pointy peak. In English, it should work well when we have enough data to get a precise estimate of the unobserved parameters.
- BIC is an approximation of Laplace as the number of data points $N \to \infty$. It ignores terms which don't scale with N (but do scale with number of parameters k), so it will do best when N is large and k is small.

Let's test these predictions out with some dice-rolling simulations.

The graphs below show the difference in bits of evidence (i.e. lnP[model|data] or its approximations) between a biased model and an unbiased model, as a function of the number of data points in each simulation, for each of the three methods. A positive difference indicates that the method favors the biased model; a negative difference indicates that the method favors the unbiased model.

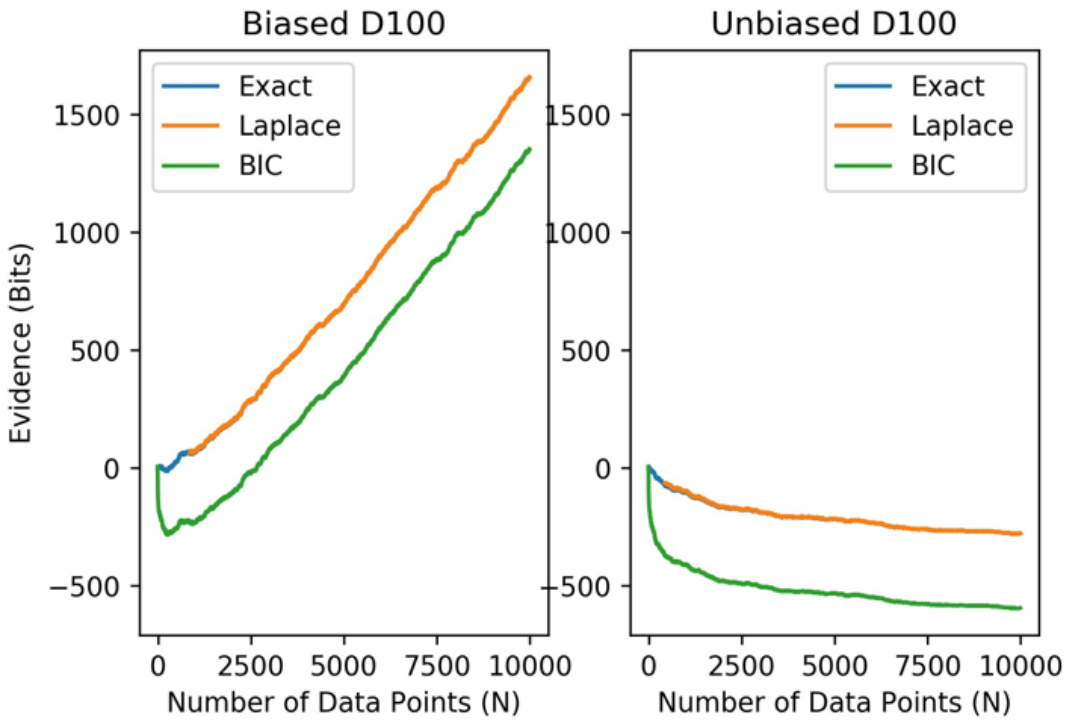
First up, let's compare a simulation with a 60-40 biased coin to a simulation with an unbiased coin. Hopefully, our methods will favor the biased model in the biased coin simulation and the unbiased model in the unbiased coin simulation.



Here's what's going on in those plots:

- All three methods agree very well. We can't see the exact method's line at all, because it's under the Laplace line. BIC does have a visible bias - it's usually a little below exact & Laplace - but the difference is small.
- All three methods generally assign higher evidence to the biased model (line above zero) in the biased coin simulation, and higher evidence to the unbiased model (line below zero) in the unbiased simulation.
- As the number of data points N increases, the evidence in favor of the biased model grows roughly linearly in the biased simulation. But in the unbiased simulation, the evidence in favor of the unbiased model grows much more slowly - logarithmically, in theory.

Because the BIC is a large-N approximation, ignoring terms which scale with k but not N, we'd expect BIC to perform worse as we crank up the number of parameters k. Let's try that: here's another pair of simulations with a 100-sided die (the biased die has 1/200 weight on half the faces and 3/200 on the other half).



This time, the BIC has a very large error - hundreds of bits of evidence in favor of an unbiased model, regardless of whether the coin is biased or not. That said, after the first few data points, the BIC's error mostly stays constant; recall that the terms ignored by the BIC are all roughly constant with respect to N. Meanwhile, the Laplace approximation agrees wonderfully with the exact calculation. (However, note that the Laplace approximation is absent in the leftmost part of each plot - for these models, it isn't well-defined until we've seen at least one of each outcome.)

Finally, notice that the exact calculation itself gives pretty reasonable probabilities in general, and in particular for small N. When the number of data points is small, it's always pretty close to zero, i.e. roughly indifferent between the models. In the high-k simulations, the exact solution gave reliably correct answers after a few hundred data points, and was roughly indifferent before that. Compare that to the BIC, which gave a very confident wrong answer in the biased case and only worked its way back to the correct answer after around 3000 data points. The moral of this story is: precise Bayesian calculations are more important when N is smaller and k is larger. We'll come back to that theme later.

Next post will add cross-validation into the picture, reusing the simulations above.

Cross-Validation vs Bayesian Model Comparison

Suppose we need to predict the outcomes of simulated dice rolls as a project for a machine learning class. We have two models: an "unbiased" model, which assigns equal probability to all outcomes, and a "biased" model, which learns each outcome frequency from the data.

In a machine learning class, how would we compare the performance of these two models?

We'd probably use a procedure like this:

- Train both models on the first 80% of the data (although training is trivial for the first model).
- Run both models on the remaining 20%, and keep whichever one performs better.

This method is called cross-validation (along with its generalizations). It's simple, it's intuitive, and it's widely-used.

So far in this sequence, we've talked about Bayesian model comparison: to compare two models, calculate the posterior P[model|data] for each model. How do cross-validation and Bayesian model comparison differ?

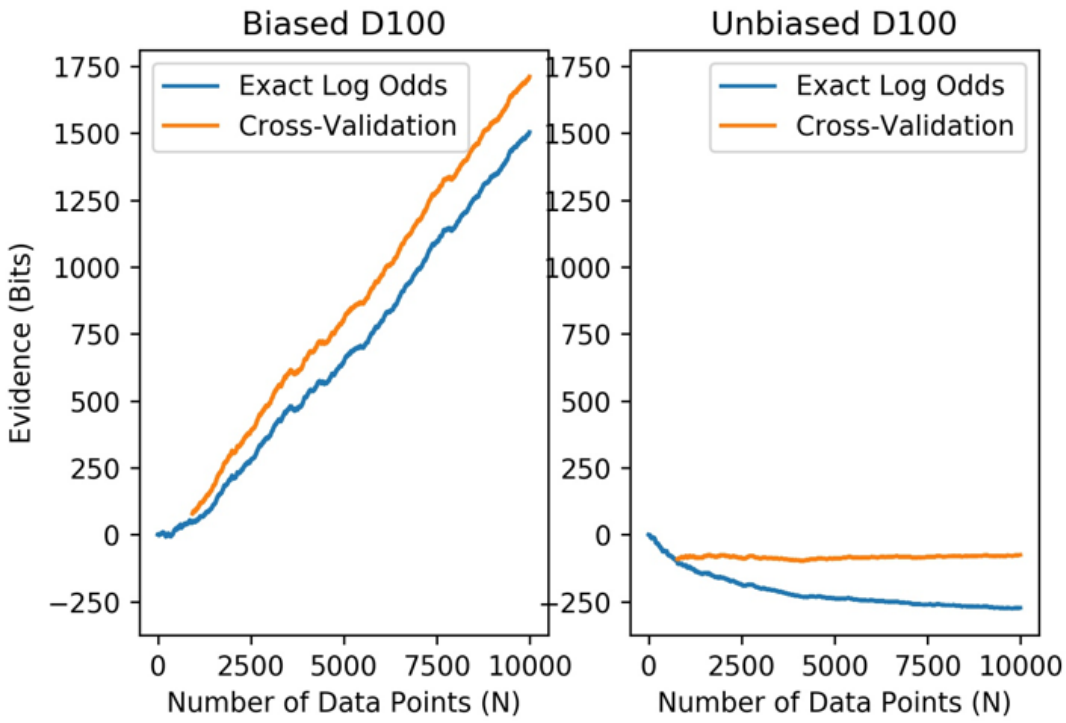
Biased/Unbiased Die Simulation

Let's run a simulation. We'll roll a 100-sided die N times, using both a biased die and an unbiased die. We'll apply both cross-validation and Bayesian model comparison to the data, and see which model each one picks. Specifics:

- We'll use N-fold cross-validation with log likelihood loss: for each data point x_i, we learn the maximum-likelihood parameters p_i^* based on all data except x_i. To get the a final metric, we then sum log likelihood over all points: $\sum_i \ln P[x_i|p_i^*]$
- For Bayesian model comparison, we'll compute lnP[model|data] for an unbiased model, and a model with uniform prior on the biases, just like we did for Wolf's Dice
- The simulated biased die has probability 1/200 on half the faces and 3/200 on the other half

We'll plot the difference in score/evidence/whatever-you-want-to-call-it assigned to each model by each method, as the number of data points N ranges from 1 up to 10000.

Here are the results from one run:



First and most important: both cross-validation and Bayesian model comparison assign more evidence to the biased model (i.e. line above zero) when the die is biased, and more evidence to the unbiased model (i.e. line below zero) when the die is unbiased.

The most striking difference between the methods is in the case of an unbiased die: Bayesian model comparison assigns lower and lower probability to the biased model, whereas cross-validation is basically flat. In theory, as $N \to \infty$, the cross-validation metric will be random with roughly zero mean.

Why the difference? Because cross-validation and Bayesian model comparison answer different questions.

Different Questions

Compare:

- Cross-validation: how accurately will this model predict future data gathered the same way?
- Bayesian: How likely is this model given the data? Or equivalently, via Bayes' rule: how well does this model predict the data we've seen?

So one is asking how well the model can predict future data, while the other is asking how well the model predicted past data. To see the difference, think about the interesting case from the simulation: biased vs unbiased model running on data from an unbiased die. As $N \to \infty$, the biased model learns the true (unbiased) frequencies, so the two models will make the same predictions going forward. With the same predictions, cross-validation is indifferent.

For cross-validation purposes, a model which gave wrong answers early but eventually learned the correct answers is just as good as a model which gave correct answers from the start.

If all we care about is predicting future data, then we don't really care whether a model made correct predictions from the start or took a while to learn. In that case, cross-validation works great (and it's certainly much computationally easier than the Bayesian method). On the other hand, if one model made correct predictions from the start and the other took a long time to learn, then that's Bayesian evidence in favor of the model which was correct from the start.

That difference becomes important in cases like Wolf's Dice II, where we wanted to deduce the physical asymmetries of a die. In that case, the fully-general model and our final model both make the same predictions about future rolls once they have enough data. But they differ on predictions about what the world looks like aside from the data itself - for instance, they make different predictions about what we would find if we took out some calipers and actually measured the dimensions of Wolf's white die.

Prediction vs Understanding

Years ago, while working in the lab of a computational biologist, he and I got into an argument about the objective of "understanding". I argued that, once some data can be predicted, there is nothing else left to understand about it. Whether it's being predicted by a detailed physical simulation or a simple abstract model or a neural network is not relevant.

Today, I no longer believe that.

Wolf's Dice II is an excellent counter-example which highlights the problem. If two models always make the same predictions about everything, then sure, there's no important difference between them. But don't confuse "make the same predictions about everything" with "make the same predictions *about the data*" or "make the same predictions about future data of this form". Even if two models eventually come to the exact same conclusion about the outcome distribution from rolls of a particular die, they can still make different predictions about the physical properties of the die itself.

If two models make different predictions about something out in the world, then it can be useful to evaluate the probabilities of the two models - even if they make the same predictions about future data of the same form as the training data.

Physical properties of a die are one example, but we can extend this to e.g. generalization problems. If we have models which make similar predictions about future data from the training distribution, but make different predictions more generally, then we can apply Bayesian model comparison to (hopefully) avoid generalization error. Of course, Bayesian model comparison is not a guarantee against generalization problems - even in principle it can only work if there's any generalization-relevant evidence in the data at all. But it should work in almost any case where cross-validation is sufficient, and many other cases as well. (I'm hedging a bit with "almost any"; it is possible for cross-validation to "get lucky" and outperform sometimes, but that should be rare as long as our priors are reasonably accurate.)

Conclusion?

In summary:

- Cross-validation tells us how well a model will predict future data of the same form as the training data. If that's all you need to know, then use cross-validation; it's much

easier computationally than Bayesian model comparison.
- Bayesian model comparison tells us how well a model predicted past data, and thus the probability of the model given the data. If want to evaluate models which make different predictions about the world even if they converge to similar predictions about future data, then use Bayesian model comparison.

One final word of caution unrelated to the main point of this post. One practical danger of cross-validation is that it will overfit if we try to compare too many different models. As an extreme example, imagine using one model for every possible bias of a coin - a whole continuum of models. Bayesian model comparison, in that case, would simply yield the posterior distribution of the bias; the maximum-posterior model would likely be overfit (depending on the prior), but the full distribution would be correct. This is an inherent danger of maximization as an epistemic technique: it's always an approximation to the posterior, so it will fail whenever the posterior isn't dominated by the maximal point.

Very Short Introduction to Bayesian Model Comparison

At least within Bayesian probability, there is a single unique unambiguously-correct answer to "how should we penalize for model complexity?": calculate the probability of each model, given the data. This is Hard to compute in general, which is why there's a whole slew of of other numbers which approximate it in various ways.

Here's how it works. Want to know whether model 1 or model 2 is more consistent with the data? Then compute $P[model_1|data]$ and $P[model_2|data]$. Using Bayes' rule:

$$P[model_i|data] = \frac{1}{Z} P[data|model_i] P[model_i]$$

where Z is the normalizer. If we're just comparing two models, then we can get rid of that annoying Z by computing odds for the two models:

$$\frac{P[model_1|data]}{P[model_2|data]} = \frac{P[data|model_1]}{P[data|model_2]} \frac{P[model_1]}{P[model_2]}$$

In English: posterior relative odds of the two models is equal to prior odds times the ratio of likelihoods. That likelihood ratio $\frac{P[data|model_1]}{P[data|model_2]}$ is the Bayes factor: it directly describes the update in the relative odds of the two models, due to the data. Calculating the Bayes factor - i.e. $P[data|model_i]$ for each model - is the main challenge of Bayesian model comparison.

Example

20 coin flips yield 16 heads and 4 tails. Is the coin biased?

Here we have two models:

- Model 1: coin unbiased
- Model 2: coin has some unknown probability θ of coming up heads (we'll use a uniform prior on θ for simplicity)

The second model has one free parameter (the bias) which we can use to fit the data, but it's more complex and prone to over-fitting. When we integrate over that free parameter, it will fit the data poorly over most of the parameter space - thus the "penalty" associated with free parameters in general.

In this example, the integral is exactly tractable (it's a dirichlet-multinomial model), and we get:

- $P[data|model_1] = \binom{20}{16} (\frac{1}{2})^{16} (\frac{1}{2})^4 = 0.0046$

- $P[\text{data}|\text{model}_2] = \int_\theta (20)(\theta)^{16}(1-\theta)^4 d\theta = 0.048$

So the Bayes factor is (.048)/(.0046) ~ 10, in favor of a biased coin. In practice, I'd say unbiased coins are at least 10x more likely than biased coins in day-to-day life a priori, so we might still think the coin is unbiased. But if we were genuinely unsure to start with, then this would be pretty decent evidence in favor.

Wolf's Dice

Around the mid-19th century, Swiss astronomer Rudolf Wolf rolled a pair of dice 20000 times, recording each outcome. Here is his data:

		White Die						
		1	2	3	4	5	6	Total
Red Die	1	547	587	500	462	621	690	3407
	2	609	655	497	535	651	684	3631
	3	514	540	468	438	587	629	3176
	4	462	507	414	413	509	611	2916
	5	551	562	499	506	658	672	3448
	6	563	598	519	487	609	646	3422
	Total	3246	3449	2897	2841	3635	3932	20000

I've taken the data from Jaynes, who uses it as an example for maximum-entropy methods. We'll use it for several examples, including some of Jaynes' models.

The most interesting fact about Wolf's dice data is that some faces come up significantly more often than others. As a quick back-of-the-envelope check, we can see this by calculating the expected number of times a face should come up on one die ($20000 * \frac{1}{6} \approx 3333.3$), and the standard deviation of this number ($\sqrt{20000 * (\frac{1}{6}) * (1 - \frac{1}{6})} \approx 52.7$). A quick glance at the data shows that the column- and row-totals differ from their expected value by roughly 2 to 6 standard deviations, an error much larger than we'd expect based on random noise.

That, however, is an ad-hoc test. The point of this sequence is to test our hypotheses from first principles, specifically the principles of probability theory. If we want to know which of two or more models is correct, then we calculate the probability of each model given the data.

White Die: Biased or Not?

Let's start with just Wolf's white die. Our two models are:

- Model 1: all faces equally probable
- Model 2: each face i has its own probability p_i (we'll use a uniform prior on p_i for now, to keep things simple)

Our data is the bottom row in the table above, and our goal is to calculate $P[\text{model}_i|\text{data}]$ for each model.

The first step is Bayes' rule:

$$P[\text{model}_i|\text{data}] = \frac{1}{Z}P[\text{data}|\text{model}_i]P[\text{model}_i]$$

Here's what those pieces each mean:

- $P[\text{model}_i]$ is our prior probability for each model - for simplicity, let's say the two are equally likely a priori.

- Z is the normalizer, chosen so that the posterior probabilities sum to one: $P[\text{model}_1|\text{data}] + P[\text{model}_2|\text{data}] = 1$ (implicitly, we assume one of the two models is "correct").
- $P[\text{data}|\text{model}_i]$ is computed using model_i

... so the actual work is in calculating $P[\text{data}|\text{model}_i]$.

Model 1: Unbiased

For the first model, $P[\text{data}|\text{model}_1]$ is a standard probability problem: given $n = 20000$ unbiased die rolls, what's the probability of $n_1 = 3246$ 1's, $n_2 = 3449$ 2's, etc? This is a multinomial distribution; the answer is

$$P[n_1 \dots n_6|\text{model}_1] = \frac{n!}{n_1! \dots n_6!} p_1^{n_1} \dots p_6^{n_6} = \frac{20000!}{3246! \dots 3312!} \left(\frac{1}{6}\right)^{20000} \approx 6.3 * 10^{-70}$$

Note the symmetry factor with the factorials: we're computing the probability of the observed *counts*, not the probability of a particular string of outcomes, so we have to add up probabilities of all the outcomes with the same counts. The same factor will appear in model 2 as well, for the same reason.

Model 2: Biased

The second model is a bit more complicated. If we knew the probabilities for each face $p_1 \dots p_6$ then we could use the multinomial distribution formula, same as model 1. But since we don't know the p's, we need to integrate over possible values:

$$P[\text{data}|\text{model}_2] = \int_p P[\text{data}|p] dP[p] = \int_p \frac{n!}{n_1! \dots n_6!} p_1^{n_1} \dots p_6^{n_6} dP[p]$$

Here $dP[p]$ is our prior distribution on p - in this case uniform, i.e. a dirichlet distribution with parameter $\alpha = 1$. This is a somewhat nontrivial integral: the constraint $\sum_i p_i = 1$ means that we're integrating over a five-dimensional surface in six dimensions. Fortunately, other people have already solved this integral in general: it's the very handy dirichlet-multinomial distribution. With $\alpha = 1$ the result is particularly simple; the integral comes out to:

$$\frac{n!}{n_1! \dots n_k!} \int_p p_1^{n_1} \dots p_k^{n_k} dP[p|\alpha = 1] = \frac{n!(k-1)!}{(n+k-1)!}$$

We'll use this formula quite a bit in the next post. For now, we're using $k = 6$ outcomes, so we get

$P[data|model_2] = \int_p P[data|p]dP[p] = \frac{20000!}{20005!} \approx 3.7 * 10^{-20}$

So the data is around 10^{50} times more likely under this model than under the unbiased-die model.

Probability of Bias

Last step: let's go back to where we started and compute the posterior probabilities of each model. We plug $P[data|model_i]$ back into Bayes' rule. Each model had a prior probability of 0.5, so we compute the normalizer Z as:

$Z = P[data|model_1]P[model_1] + P[data|model_2]P[model_2] \approx 1.9 * 10^{-20}$

so

$P[model_1|data] = \frac{1}{Z} P[data|model_1]P[model_1] \approx 1.7 * 10^{-50}$

$P[model_2|data] \approx 1.0 - 1.7 * 10^{-50}$

A few comments on this result...

First, we have pretty conclusively confirmed that the faces are not equally probable, given this data.

Second, the numbers involved are REALLY BIG - ratios on the order of 10^{50}. This is par for the course: since independent probabilities multiply, probabilities tend to be roughly exponential in the number of data points. One side effect is that, as long as we have a reasonable amount of data, the priors don't matter much. Even if we'd thought that an unbiased die was a thousand times more likely than a biased die a priori, those huge exponents would have completely swamped our prior.

Playing around with α will reveal that the same applies to our prior distribution for p: the result is not very sensitive to changes in the prior. A word of caution, however: priors over unknown parameters become more important in high-dimensional problems.

Third, notice that there was no maximization and no extra parameters to twiddle. Once the models were specified, we had zero choices to make, zero degrees of freedom to play with. Any "free parameters" - i.e. p - have a prior over which we integrate. That integral is the hard part: as the models get larger and more complicated, we need to evaluate hairy high-dimensional integrals. The problem is not just NP-complete, it's #P-complete. In practice, approximations are used instead, including the entire range of hypothesis tests and model comparison tests - that's where maximization enters the picture. We'll talk about some of those later, especially about when they're likely to work or not work.

Finally, note that we compared a model which is conceptually "compatible" with the data (i.e. capable of generating roughly the observed frequencies), to one which is conceptually "incompatible" (i.e. the observed frequencies are way outside the expected range). A more interesting case is to consider two models which are both "compatible". In that case, we'd want to use some kind of complexity penalty to say that the more complex model is less

likely - otherwise we'd expect overfit. In the next post, we'll revisit Wolf's dice with a couple models from Jaynes, and see how P[data|model] "penalizes" overly-complicated models.

Wolf's Dice II: What Asymmetry?

In the previous post, we looked at Rudolph Wolf's data on 20000 rolls of a pair of dice. Specifically, we looked at the data on the white die, and found that it was definitely biased. This raises an interesting question: what biases, specifically, were present? In particular, can we say anything about the physical asymmetry of the die? Jaynes addressed this exact question; we will test some of his models here.

Elongated Cube Models

Jaynes suggests that, if the die were machined, then it would be pretty easy to first cut an even square along two dimensions. But the cut in the third dimension would be more difficult; getting the length to match the other two dimensions would be tricky. Based on this, we'd expect to see an asymmetry which gives two opposite faces (1 & 6, 2 & 5, or 3 & 4) different probabilities from all the other faces.

Here's what the model looks like for the 1 & 6 pair:

- 1 & 6 each have the same probability $\frac{p}{2}$
- 2, 3, 4 & 5 each have the same probability $\frac{1-p}{4}$
- Uniform prior on p (i.e. dirichlet with $\alpha = 1$)

Let's call this model$_{1,6}$.

I will omit the details of calculations in this post; readers are welcome to use them as exercises. (All the integrals can be evaluated using the dirichlet-multinomial $\alpha=1$ formula from the previous post.) In this case, we find

$$P[data|model_{1,6}] = \frac{n!}{(n+1)!} \left(\frac{1}{2}\right)^{n_1+n_6} \left(\frac{1}{4}\right)^{n_2+n_3+n_4+n_5} \approx 2.2 * 10^{-59}$$

For the other two opposite face pairs, we get:

- 2,5: $1.4 * 10^{-63}$
- 3,4: $8.5 * 10^{-29}$

... sure enough, an asymmetry on the 3,4 axis goes a very long way toward explaining this data.

Recall from the previous post that the unbiased model gave a marginal likelihood P[data|model] around 10^{-70}, and the biased model with separate probabilities for each face gave around 10^{-20}. So based on the data, our 3,4 model is still about a billion times less probable than the full biased model (assuming comparable prior probabilities for the two models), but it's getting relatively close - probabilities naturally live on a log scale. It looks like the 3-4 asymmetry is the main asymmetry in the data, but some other smaller asymmetry must also be significant.

Just for kicks, I tried a model with a different probability for each pair of faces, again with uniform prior on the p's. That one came out to $1.7 * 10^{-30}$ - somewhat worse than the 3,4 model. If you're used to traditional statistics, this may come as a surprise: how can a strictly more general model have *lower* marginal likelihood P[data|model]? The answer is that, in traditional statistics, we'd be looking for the unobserved parameter values p with the *maximum* likelihood P[data|model, p] - of course a strictly more general model will have a maximum likelihood value at least as high. But when computing P[data|model], we're integrating over the unobserved parameters p. A more general model has more ways to be wrong; unless it's capturing some important phenomenon, a smaller fraction of the parameter space will have high P[data|model, p]. We'll come back to this again later in the sequence.

Pip Asymmetry Model

Jaynes' other main suggestion was that the pips on the die are asymmetric - i.e. there's less mass near the 6 face than the 1 face, because more pips have been dug out of the 6 face.

As a first approximation to this, let's consider just the asymmetry between 1 and 6 - the pair with the highest pip difference. We'll also keep all the structure from the 3,4 model, since that seems to be the main asymmetry. Here's the model:

- 3 & 4 have the same probability $\frac{p}{2}$, as before
- 2 & 5 have the same probability $\frac{1-p}{4}$, as before
- 1 & 6 *together* have probability $\frac{1-p}{2}$, same as 3 and 5 together, but their individual probabilities may be different. Conditional on rolling either a 1 or 6, 1 comes up with probability p' and 6 with probability $(1-p')$
- Both p and p' have uniform priors

The conditional parameterization for 1 & 6 is chosen to make the math clean.

Let's call this $\text{model}_{3,4+\text{pip}}$. Marginal likelihood:

$$P[\text{data}|\text{model}_{3,4+\text{pip}}] = \frac{(n_1+1)!}{(n_1+n_6+1)!}\left(\frac{(n_3+n_4)!}{n_3!n_4!}\left(\frac{1}{2}\right)^{n_3+n_4}\right) *$$

$$\left(\frac{(n_1+n_2+n_5+n_6)!}{n_1!n_2!n_5!n_6!}\left(\frac{1}{2}\right)^{n_1+n_6}\left(\frac{1}{4}\right)^{n_2+n_5}\right)\left(\frac{(n_1+n_6+1)!}{(n_1+n_6+1)!}\right) \approx 2.3 * 10^{-16}$$

... and now we have a model which solidly beats separate probabilities for each face!

(I also tried a pip model by itself, without the 3,4 asymmetry. That one wound up at $2.1 * 10^{-70}$ - almost as bad as the full unbiased model.)

We can also go one step further, and assume that the pip difference also causes 2 and 5 to have slightly different probabilities. This model gives $P[\text{data}|\text{model}] \approx 3.9 * 10^{-17}$ - a bit lower than the model above, but close enough that it still gets significant posterior probability (about $\frac{3.9*10^{-17}}{3.9*10^{-17}+2.3*10^{-16}} = 14\%$ assuming equal priors; all the other models we've seen have near-zero posterior assuming equal priors). So based on the data, the model with just the 1-6 pip difference is a bit better, but we're not entirely sure. My guess is that a fancier model could significantly beat both of these by predicting that the effect of a pip difference scales with the number of pips, rather than just using whole separate parameters for the 1-6 and 2-5 differences. But that would get into hairier math, so I'm not going to do it here.

To recap, here's what $\text{model}_{3,4+\text{pip}}$ says:

- 3 and 4 have the same probability, but that probability may be different from everything else
- 2 and 5 have the same probability, and 1 and 6 together have the same probability as 2 and 5, but 1 and 6 have different probabilities.

That's it; just two "free parameters". Note that the full biased model, with different probabilities for each face, is strictly more general than this - any face probabilities p which are compatible with $\text{model}_{3,4+\text{pip}}$ are also compatible with the full biased model. But the full biased model is compatible with *any* face probabilities p; $\text{model}_{3,4+\text{pip}}$ is not compatible with all possible p's. So if we see data which matches the p's compatible with $\text{model}_{3,4+\text{pip}}$, then that must push up our posterior for $\text{model}_{3,4+\text{pip}}$ relative to the full unbiased model - $\text{model}_{3,4+\text{pip}}$ makes a stronger prediction, so it gets more credit when it's right. The result: less flexible models which are consistent with the data will get higher posterior probability. The "complexity penalty" is not explicit, but implicit: it's just a natural consequence of conservation of expected evidence.

Next post we'll talk about approximation methods for hairy integrals, and then we'll connect all this to some common methods for scoring models.

Laplace Approximation

The last couple posts compared some specific models for 20000 rolls of a die. This post will step back, and talk about more general theory for Bayesian model comparison.

The main problem is to calculate P[data|model] for some model. The model will typically give the probability of observed data x (e.g. die rolls) based on some unobserved parameter values θ (e.g. the p's in the last two posts), along with a prior distribution over θ. We then need to compute

$$P[\text{data}|\text{model}] = \int_\theta P[\text{data}|\theta] dP[\theta]$$

which will be a hairy high-dimensional integral.

Some special model structures allow us to simplify the problem, typically by factoring the integral into a product of one-dimensional integrals. But in general, we need some method for approximating these integrals.

The two most common approximation methods used in practice are Laplace approximation around the maximum-likelihood point, and MCMC (see e.g. here for application of MCMC to Bayes factors). We'll mainly talk about Laplace approximation here - in practice MCMC mostly works well in the same cases, assuming the unobserved parameters are continuous.

Laplace Approximation

Here's the idea of Laplace approximation. First, posterior distributions tend to be very pointy. This is mainly because independent probabilities multiply, so probabilities tend to scale exponentially with the number of data points. Think of the probabilities we calculated in the last two posts, with values like 10^{-70} or 10^{-20} - that's the typical case. If we're integrating over a function with values like that, we can basically just pay attention to the region around the highest value - other regions will have exponentially small weight.

Laplace' trick is to use a second-order approximation within that high-valued region. Specifically, since probabilities naturally live on a log scale, we'll take a second order-approximation of the log likelihood around its maximum point. Thus:

$$\int_\theta e^{\ln P[\text{data}|\theta]} dP[\theta] \approx \int_\theta e^{\ln P[\text{data}|\theta_{max}] + \frac{1}{2}(\theta-\theta_{max})^T \left(\frac{d^2 \ln P}{d\theta^2}\Big|_{\theta_{max}}\right)(\theta-\theta_{max})} dP[\theta]$$

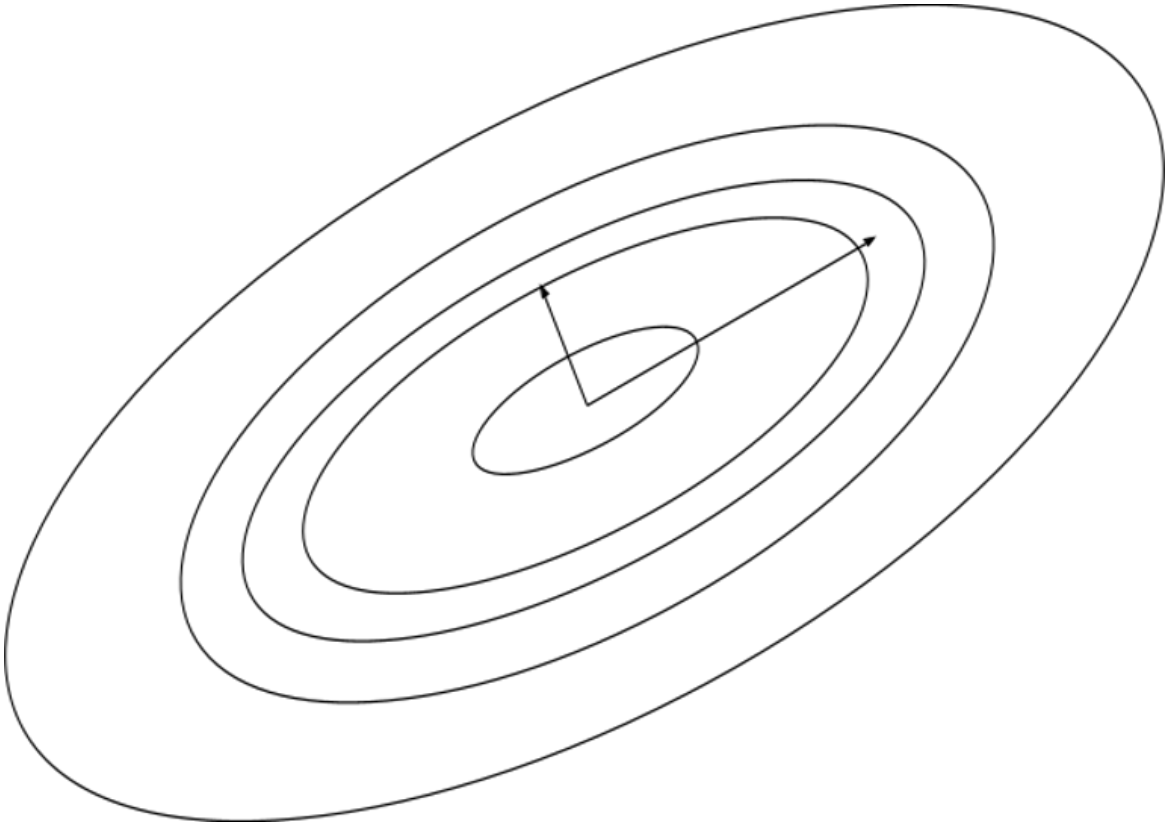
If we assume that the prior dP[θ] is uniform (i.e. dP[θ] = dθ), then this looks like a normal distribution on θ with mean θ_{max} and variance given by the inverse Hessian matrix of the log-likelihood. (It turns out that, even for non-uniform dP[θ], we can just transform θ so that the prior looks uniform near θ_{max}, and transform it back when we're done.) The result:

$$\int_\theta e^{\ln P[\text{data}|\theta]} dP[\theta] \approx P[\text{data}|\theta_{max}] p[\theta_{max}] (2\pi)^{\frac{k}{2}} \det\left(-\frac{d^2 \ln P}{d\theta^2}\Big|_{\theta_{max}}\right)^{-\frac{1}{2}}$$

Let's walk through each of those pieces:

- $P[\text{data}|\theta_{max}]$ is the usual maximum likelihood: the largest probability assigned to the data by any particular value of θ.
- $p[\theta_{max}]$ is the prior probability density of the maximum-likelihood θ point.
- $(2\pi)^{\frac{k}{2}}$ is that annoying constant factor which shows up whenever we deal with normal distributions; k is the dimension of θ.
- $\det(-\frac{d^2 \ln p}{d\theta^2}|_{\theta_{max}})$ is the determinant of the "Fisher information matrix"; it quantifies how wide or skinny the peak is.

A bit more detail on that last piece: intuitively, each eigenvalue of the Fisher information matrix tells us the approximate width of the peak in a particular direction. Since the matrix is the inverse variance (in one dimension $\frac{1}{\sigma^2}$) of our approximate normal distribution, and "width" of the peak of a normal distribution corresponds to the standard deviation σ, we use an inverse square root (i.e. the power of $-\frac{1}{2}$) to extract a width from each eigenvalue. Then, to find how much volume the peak covers, we multiply together the widths along each direction - thus the determinant, which is the product of eigenvalues.



Why do we need eigenvalues? The diagram above shows the general idea: for the function shown, the two arrows would be eigenvectors of the Hessian $\frac{d^2 \ln p}{d\theta^2}$ at the peak. Under a second-order approximation, these are principal axes of the function's level sets (the ellipses

in the diagram). They are the natural directions along which to measure the width. The eigenvalue associated with each eigenvector tells us the width, and then taking their product (via the determinant) gives a volume. In the picture above, the determinant would be proportional to the volume of any of the ellipses.

Altogether, then, the Laplace approximation takes the height of the peak (i.e. $P[data|\theta_{max}]p[\theta_{max}]$) and multiplies by the volume of θ-space which the peak occupies, based on a second-order approximation of the likelihood around its peak.

Laplace Complexity Penalty

The Laplace approximation contains our first example of an explicit complexity penalty.

The idea of a complexity penalty is that we first find the maximum log likelihood $\ln P[data|\theta_{max}]$, maybe add a term for our θ-prior $\ln p[\theta_{max}]$, and that's the "score" of our model. But more general models, with more free parameters, will always score higher, leading to overfit. To counterbalance that, we calculate some numerical penalty which is larger for more complex models (i.e. those with more free parameters) and subtract that penalty from the raw score.

In the case of Laplace approximation, a natural complexity penalty drops out as soon as we take the log of the approximation formula:

$$\ln P[data|model] \approx \ln P[data|\theta_{max}] + \ln p[\theta_{max}] + \frac{k}{2}\ln(2\pi) - \frac{1}{2}\ln\det\left(-\frac{d^2 \ln P}{d\theta^2}\Big|_{\theta_{max}}\right)$$

The last two terms are the complexity penalty. As we saw above, they give the (log) volume of the likelihood peak in θ-space. The wider the peak, the larger the chunk of θ-space which actually predicts the observed data.

There are two main problems with this complexity penalty:

- First, there's the usual issues with approximating a posterior distribution by looking at a single point. Multimodal distributions are a problem, insufficiently-pointy distributions are a problem. These problems apply to basically any complexity penalty method.
- Second, although the log determinant of the Hessian can be computed via backpropagation and linear algebra, that computation takes $O(k^3)$. That's a lot better than the exponential time required for high-dimensional integrals, but still too slow to be practical for large-scale models with millions of parameters.

Historically, a third issue was the math/coding work involved in calculating a Hessian, but modern backprop tools like Tensorflow or autograd make that pretty easy; I expect in the next few years we'll see a lot more people using a Laplace-based complexity penalty directly.

The $O(k^3)$ runtime remains a serious problem for large-scale models, however, and that problem is unlikely to be solved any time soon: a linear-time method for computing the Hessian log determinant would yield an $O(n^2)$ matrix multiplication algorithm.

From Laplace to BIC

The previous post outlined Laplace approximation, one of the most common tools used to approximate hairy probability integrals. In this post, we'll use Laplace approximation to derive the Bayesian Information Criterion (BIC), a popular complexity penalty method for comparing models with more free parameters to models with fewer free parameters.

The BIC is pretty simple:

- Start with the maximum log likelihood $\ln P[\text{data}|\theta_{max}]$
- Subtract $\frac{k}{2}\ln N$ to penalize for complexity, where N is the number of data points and k is the number of free parameters (i.e. dimension of θ).

Thus: $\ln P[\text{data}|\theta_{max}] - \frac{k}{2}\ln N$. Using this magic number, we compare any two models we like.

Let's derive that.

BIC Derivation

As usual, we'll start from P[data|model]. (Caution: don't forget that what we really care about is P[model|data]; we can jump to P[data|model] only as long as our priors are close enough to be swamped by the evidence.) This time, we'll assume that we have N independent data points x_i, all with the same unobserved parameters - e.g. N die rolls with the same unobserved biases. In that case, we have

$$P[\text{data}|\text{model}] = \int_\theta P[\text{data}|\theta]dP[\theta] = \int_\theta \prod_{i=1}^{N} P[x_i|\theta]p[\theta]d\theta$$

Next, apply Laplace approximation and take the log.

$$\ln P[\text{data}|\text{model}] \approx \sum_i \ln P[x_i|\theta_{max}] + \ln p[\theta_{max}] + \frac{k}{2}\ln(2\pi) - \frac{1}{2}\ln\det(H)$$

where the Hessian matrix H is given by

$$H = \frac{d^2}{d\theta^2}\ln P[\text{data}|\theta]|_{\theta_{max}} = \sum_i \frac{d^2}{d\theta^2}\ln P[x_i|\theta]|_{\theta_{max}}$$

Now for the main trick: how does each term scale as the number of data points N increases?

- The max log likelihood $\sum_i P[x_i|\theta_{max}]$ is a sum over data points, so it should scale roughly proportionally to N.
- The prior density and the $\frac{k}{2}\ln(2\pi)$ are constant with respect to N.
- H is another sum over data points, so it should also scale roughly proportionally to N.

Let's go ahead and write H as $N * (\frac{1}{N}H)$, to pull out the N-dependence. Then, if we can remember how determinants scale:

$$\ln\det(N * (\tfrac{1}{N}H)) = \ln\det(\tfrac{1}{N}H) + k * \ln N$$

so we can re-write our Laplace approximation as

$$\ln P[\text{data}|\text{model}] \approx \sum_i \ln P[x_i|\theta_{max}] + p[\theta_{max}] + \tfrac{k}{2}\ln(2\pi) - \tfrac{1}{2}\ln\det(\tfrac{1}{N}H) - \tfrac{k}{2}\ln N$$

$$= \ln P[\text{data}|\theta_{max}] - \tfrac{k}{2}\ln N + O(1)$$

where O(1) contains all the terms which are roughly constant with respect to N. The first two terms are the BIC.

In other words, the BIC is just the Laplace approximation, but ignoring all the terms which don't scale up as the number of data points increases.

When Does BIC Work?

What conditions need to hold for BIC to work? Let's go back through the derivation and list out the key assumptions behind our approximations.

- First, in order to jump from P[model|data] to P[data|model], our models should have roughly similar prior probabilities P[model] - i.e. within a few orders of magnitude.
- Second, in order for any point approximation to work, the posterior parameter distribution needs to be pointy and unimodal - most of the posterior probability mass must be near θ_{max}. In other words, we need enough data to get a precise estimate of the unobserved parameters.
- Third, we must have N large enough that $\frac{k}{2}\ln N$ (the smallest term we're keeping) is much larger than the O(1) terms we're throwing away.

That last condition is the big one. BIC is a large-N approximation, so N needs to be large for it to work. How large? That depends how big $\ln\det(\frac{1}{N}H)$ is - N needs to be exponentially larger than that. We'll see an example in the next post.

Next post will talk more about relative advantages of BIC, Laplace, and exact calculation for comparing models. We'll see a concrete example of when BIC works/fails.

Bayesian Model Testing Comparisons

We've now seen three different methods for model comparison:

- Exact calculation of posterior probabilities of each model via P[data|model]
- Laplace approximation
- Bayesian Information Criterion (BIC)

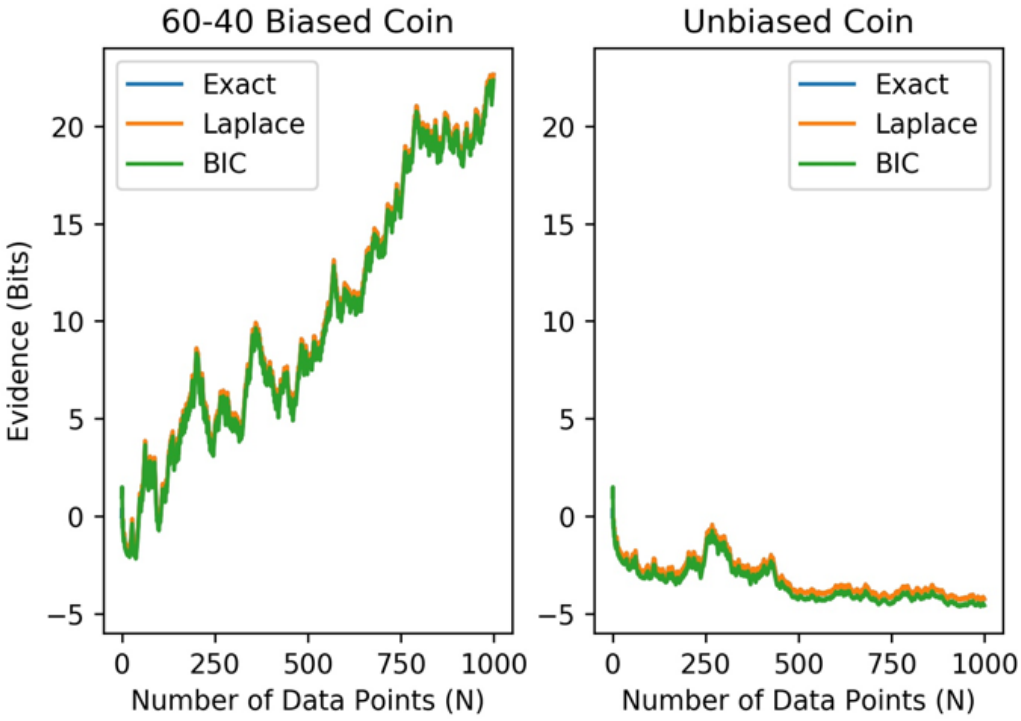
Based on the derivations, we can make some predictions:

- Laplace is a second-order approximation of the exact integral around the maximum-likelihood estimate. It should work well when the integral is dominated by one very pointy peak. In English, it should work well when we have enough data to get a precise estimate of the unobserved parameters.
- BIC is an approximation of Laplace as the number of data points $N \to \infty$. It ignores terms which don't scale with N (but do scale with number of parameters k), so it will do best when N is large and k is small.

Let's test these predictions out with some dice-rolling simulations.

The graphs below show the difference in bits of evidence (i.e. lnP[model|data] or its approximations) between a biased model and an unbiased model, as a function of the number of data points in each simulation, for each of the three methods. A positive difference indicates that the method favors the biased model; a negative difference indicates that the method favors the unbiased model.

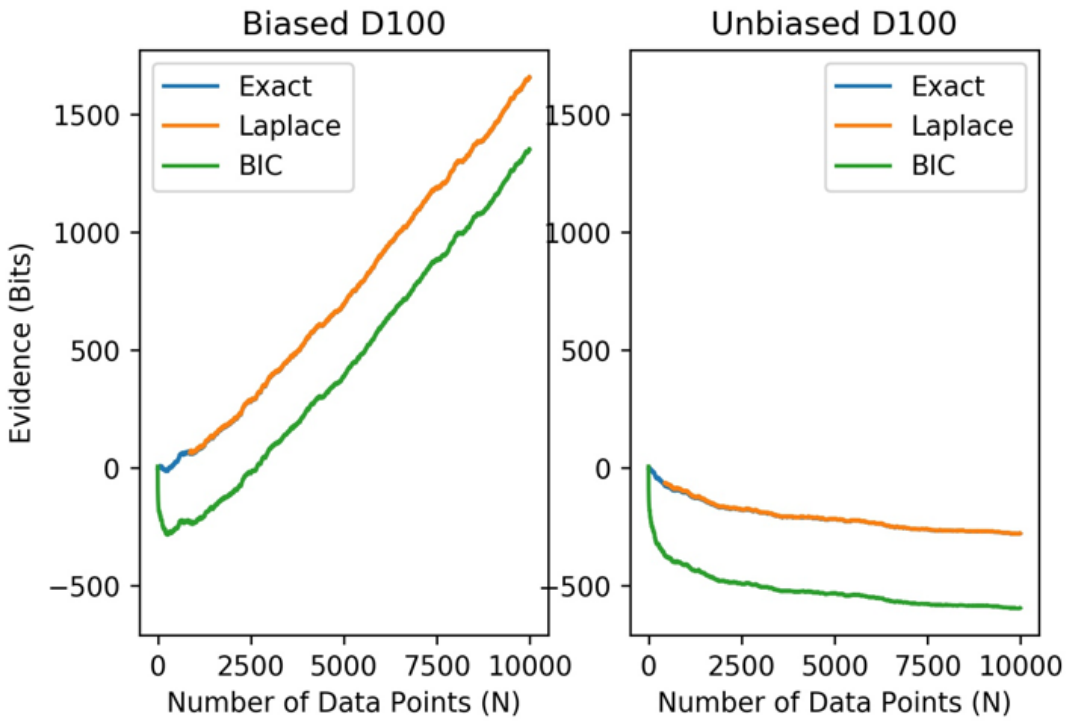
First up, let's compare a simulation with a 60-40 biased coin to a simulation with an unbiased coin. Hopefully, our methods will favor the biased model in the biased coin simulation and the unbiased model in the unbiased coin simulation.



Here's what's going on in those plots:

- All three methods agree very well. We can't see the exact method's line at all, because it's under the Laplace line. BIC does have a visible bias - it's usually a little below exact & Laplace - but the difference is small.
- All three methods generally assign higher evidence to the biased model (line above zero) in the biased coin simulation, and higher evidence to the unbiased model (line below zero) in the unbiased simulation.
- As the number of data points N increases, the evidence in favor of the biased model grows roughly linearly in the biased simulation. But in the unbiased simulation, the evidence in favor of the unbiased model grows much more slowly - logarithmically, in theory.

Because the BIC is a large-N approximation, ignoring terms which scale with k but not N, we'd expect BIC to perform worse as we crank up the number of parameters k. Let's try that: here's another pair of simulations with a 100-sided die (the biased die has 1/200 weight on half the faces and 3/200 on the other half).



This time, the BIC has a very large error - hundreds of bits of evidence in favor of an unbiased model, regardless of whether the coin is biased or not. That said, after the first few data points, the BIC's error mostly stays constant; recall that the terms ignored by the BIC are all roughly constant with respect to N. Meanwhile, the Laplace approximation agrees wonderfully with the exact calculation. (However, note that the Laplace approximation is absent in the leftmost part of each plot - for these models, it isn't well-defined until we've seen at least one of each outcome.)

Finally, notice that the exact calculation itself gives pretty reasonable probabilities in general, and in particular for small N. When the number of data points is small, it's always pretty close to zero, i.e. roughly indifferent between the models. In the high-k simulations, the exact solution gave reliably correct answers after a few hundred data points, and was roughly indifferent before that. Compare that to the BIC, which gave a very confident wrong answer in the biased case and only worked its way back to the correct answer after around 3000 data points. The moral of this story is: precise Bayesian calculations are more important when N is smaller and k is larger. We'll come back to that theme later.

Next post will add cross-validation into the picture, reusing the simulations above.

Cross-Validation vs Bayesian Model Comparison

Suppose we need to predict the outcomes of simulated dice rolls as a project for a machine learning class. We have two models: an "unbiased" model, which assigns equal probability to all outcomes, and a "biased" model, which learns each outcome frequency from the data.

In a machine learning class, how would we compare the performance of these two models?

We'd probably use a procedure like this:

- Train both models on the first 80% of the data (although training is trivial for the first model).
- Run both models on the remaining 20%, and keep whichever one performs better.

This method is called cross-validation (along with its generalizations). It's simple, it's intuitive, and it's widely-used.

So far in this sequence, we've talked about Bayesian model comparison: to compare two models, calculate the posterior P[model|data] for each model. How do cross-validation and Bayesian model comparison differ?

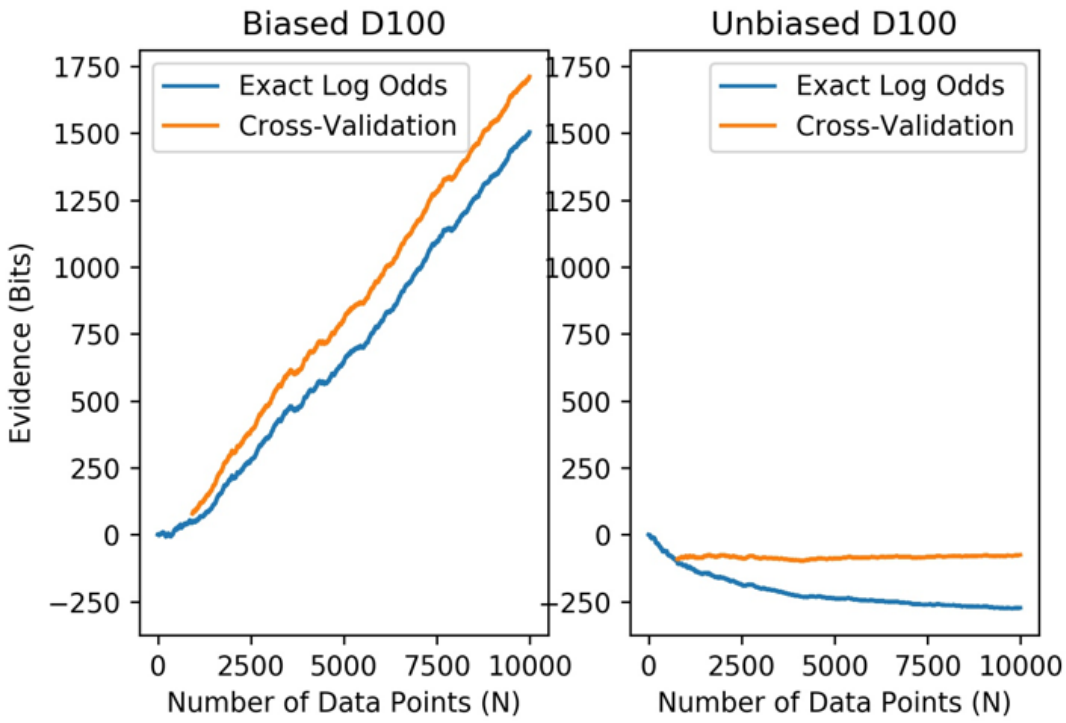
Biased/Unbiased Die Simulation

Let's run a simulation. We'll roll a 100-sided die N times, using both a biased die and an unbiased die. We'll apply both cross-validation and Bayesian model comparison to the data, and see which model each one picks. Specifics:

- We'll use N-fold cross-validation with log likelihood loss: for each data point x_i, we learn the maximum-likelihood parameters p_i^* based on all data except x_i. To get the a final metric, we then sum log likelihood over all points: $\sum_i \ln P[x_i|p_i^*]$
- For Bayesian model comparison, we'll compute lnP[model|data] for an unbiased model, and a model with uniform prior on the biases, just like we did for Wolf's Dice
- The simulated biased die has probability 1/200 on half the faces and 3/200 on the other half

We'll plot the difference in score/evidence/whatever-you-want-to-call-it assigned to each model by each method, as the number of data points N ranges from 1 up to 10000.

Here are the results from one run:



First and most important: both cross-validation and Bayesian model comparison assign more evidence to the biased model (i.e. line above zero) when the die is biased, and more evidence to the unbiased model (i.e. line below zero) when the die is unbiased.

The most striking difference between the methods is in the case of an unbiased die: Bayesian model comparison assigns lower and lower probability to the biased model, whereas cross-validation is basically flat. In theory, as $N \to \infty$, the cross-validation metric will be random with roughly zero mean.

Why the difference? Because cross-validation and Bayesian model comparison answer different questions.

Different Questions

Compare:

- Cross-validation: how accurately will this model predict future data gathered the same way?
- Bayesian: How likely is this model given the data? Or equivalently, via Bayes' rule: how well does this model predict the data we've seen?

So one is asking how well the model can predict future data, while the other is asking how well the model predicted past data. To see the difference, think about the interesting case from the simulation: biased vs unbiased model running on data from an unbiased die. As $N \to \infty$, the biased model learns the true (unbiased) frequencies, so the two models will make the same predictions going forward. With the same predictions, cross-validation is indifferent.

For cross-validation purposes, a model which gave wrong answers early but eventually learned the correct answers is just as good as a model which gave correct answers from the start.

If all we care about is predicting future data, then we don't really care whether a model made correct predictions from the start or took a while to learn. In that case, cross-validation works great (and it's certainly much computationally easier than the Bayesian method). On the other hand, if one model made correct predictions from the start and the other took a long time to learn, then that's Bayesian evidence in favor of the model which was correct from the start.

That difference becomes important in cases like Wolf's Dice II, where we wanted to deduce the physical asymmetries of a die. In that case, the fully-general model and our final model both make the same predictions about future rolls once they have enough data. But they differ on predictions about what the world looks like aside from the data itself - for instance, they make different predictions about what we would find if we took out some calipers and actually measured the dimensions of Wolf's white die.

Prediction vs Understanding

Years ago, while working in the lab of a computational biologist, he and I got into an argument about the objective of "understanding". I argued that, once some data can be predicted, there is nothing else left to understand about it. Whether it's being predicted by a detailed physical simulation or a simple abstract model or a neural network is not relevant.

Today, I no longer believe that.

Wolf's Dice II is an excellent counter-example which highlights the problem. If two models always make the same predictions about everything, then sure, there's no important difference between them. But don't confuse "make the same predictions about everything" with "make the same predictions *about the data*" or "make the same predictions about future data of this form". Even if two models eventually come to the exact same conclusion about the outcome distribution from rolls of a particular die, they can still make different predictions about the physical properties of the die itself.

If two models make different predictions about something out in the world, then it can be useful to evaluate the probabilities of the two models - even if they make the same predictions about future data of the same form as the training data.

Physical properties of a die are one example, but we can extend this to e.g. generalization problems. If we have models which make similar predictions about future data from the training distribution, but make different predictions more generally, then we can apply Bayesian model comparison to (hopefully) avoid generalization error. Of course, Bayesian model comparison is not a guarantee against generalization problems - even in principle it can only work if there's any generalization-relevant evidence in the data at all. But it should work in almost any case where cross-validation is sufficient, and many other cases as well. (I'm hedging a bit with "almost any"; it is possible for cross-validation to "get lucky" and outperform sometimes, but that should be rare as long as our priors are reasonably accurate.)

Conclusion?

In summary:

- Cross-validation tells us how well a model will predict future data of the same form as the training data. If that's all you need to know, then use cross-validation; it's much

easier computationally than Bayesian model comparison.
- Bayesian model comparison tells us how well a model predicted past data, and thus the probability of the model given the data. If want to evaluate models which make different predictions about the world even if they converge to similar predictions about future data, then use Bayesian model comparison.

One final word of caution unrelated to the main point of this post. One practical danger of cross-validation is that it will overfit if we try to compare too many different models. As an extreme example, imagine using one model for every possible bias of a coin - a whole continuum of models. Bayesian model comparison, in that case, would simply yield the posterior distribution of the bias; the maximum-posterior model would likely be overfit (depending on the prior), but the full distribution would be correct. This is an inherent danger of maximization as an epistemic technique: it's always an approximation to the posterior, so it will fail whenever the posterior isn't dominated by the maximal point.