Model Evaluation and Information Criteria

Felipe José Bravo Márquez

November 9, 2021

Model Evaluation and Information Criteria

- In this class we will introduce various concepts for evaluating statistical models.
- According to [McElreath, 2020], there are two fundamental kinds of statistical error:
  - **Overfitting**: models that learn too much from the data leading to poor prediction.
  - **Underfitting**: models that learn too little from the data, which also leads to poor prediction.
- We will study the following approaches to tackle these problems.
  - **Regularization**: a mechanism to tell our models not to get too excited by the data.
  - **Information criteria** and **Cross-validation**: scoring devices to estimate predictive accuracy of our models.
- In order to introduce information criteria, this class must also introduce some concepts of **information theory**.

The problem with parameters

- In the class of linear regression we learned that including more attributes can lead to a more accurate model.
- However, we also learned that adding more variables almost always improves the fit of the model to the data, as measured by the coefficient of determination R^2.
- This is true even when the variables we add to a model have no relation to the outcome.
- So it's no good to choose among models using only fit to the data.

The problem with parameters

- While more complex models fit the data better, they often predict new data worse.
- This means that a complex model will be very sensitive to the exact sample used to fit it.
- This will lead to potentially large mistakes when future data is not exactly like the past data.
- But simple models, with too few parameters, tend instead to underfit, systematically over-predicting or under-predicting the data.
- Regardless of how well future data resemble past data.
- So we can't always favor either simple models or complex models.
- Let's examine both of these issues in the context of a simple data example.

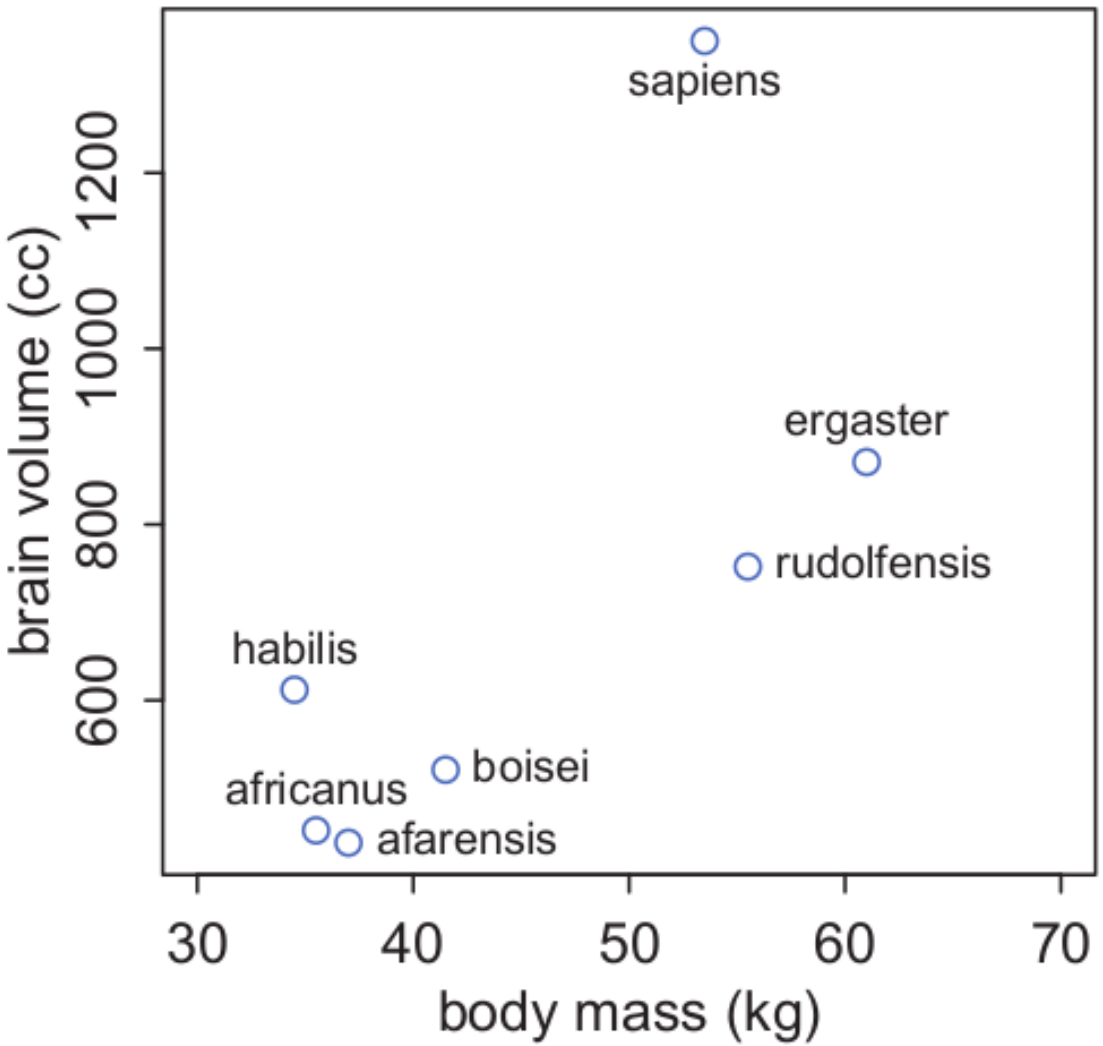
The problem with parameters

- We are going to create a data.frame containing average brain volumes and body masses for seven hominin species.

```
sppnames <- c( "afarensis","africanus","habilis",
               "boisei", "rudolfensis","ergaster",
               "sapiens")
brainvolcc <- c( 438 , 452 , 612, 521, 752, 871,
                 1350 )
masskg <- c( 37.0 , 35.5 , 34.5 , 41.5 , 55.5 ,
             61.0 , 53.5 )
d <- data.frame( species=sppnames , brain=brainvolcc,
                 mass=masskg )
```

- It's not unusual for data like this to be highly correlated.
- Brain size is correlated with body size, across species.

The problem with parameters



The problem with parameters

- We will model brain size as a function of body size.
- We will fit a series of increasingly complex model families and see which function fits the data best.
- Each of these models will just be a polynomial of higher degree.

```
reg.ev.1 <- lm( brain ~ mass , data=d )
reg.ev.2 <- lm( brain ~ mass + I(mass^2)
                , data=d )
reg.ev.3 <- lm( brain ~ mass + I(mass^2)
                + I(mass^3),data=d )
reg.ev.4 <- lm( brain ~ mass + I(mass^2)
                + I(mass^3) + I(mass^4),data=d )
reg.ev.5 <- lm( brain ~ mass + I(mass^2)
                + I(mass^3) + I(mass^4)
                + I(mass^5),data=d )
reg.ev.6 <- lm( brain ~ mass + I(mass^2)
                + I(mass^3) + I(mass^4)+
                  I(mass^5)+ I(mass^6),data=d )
```

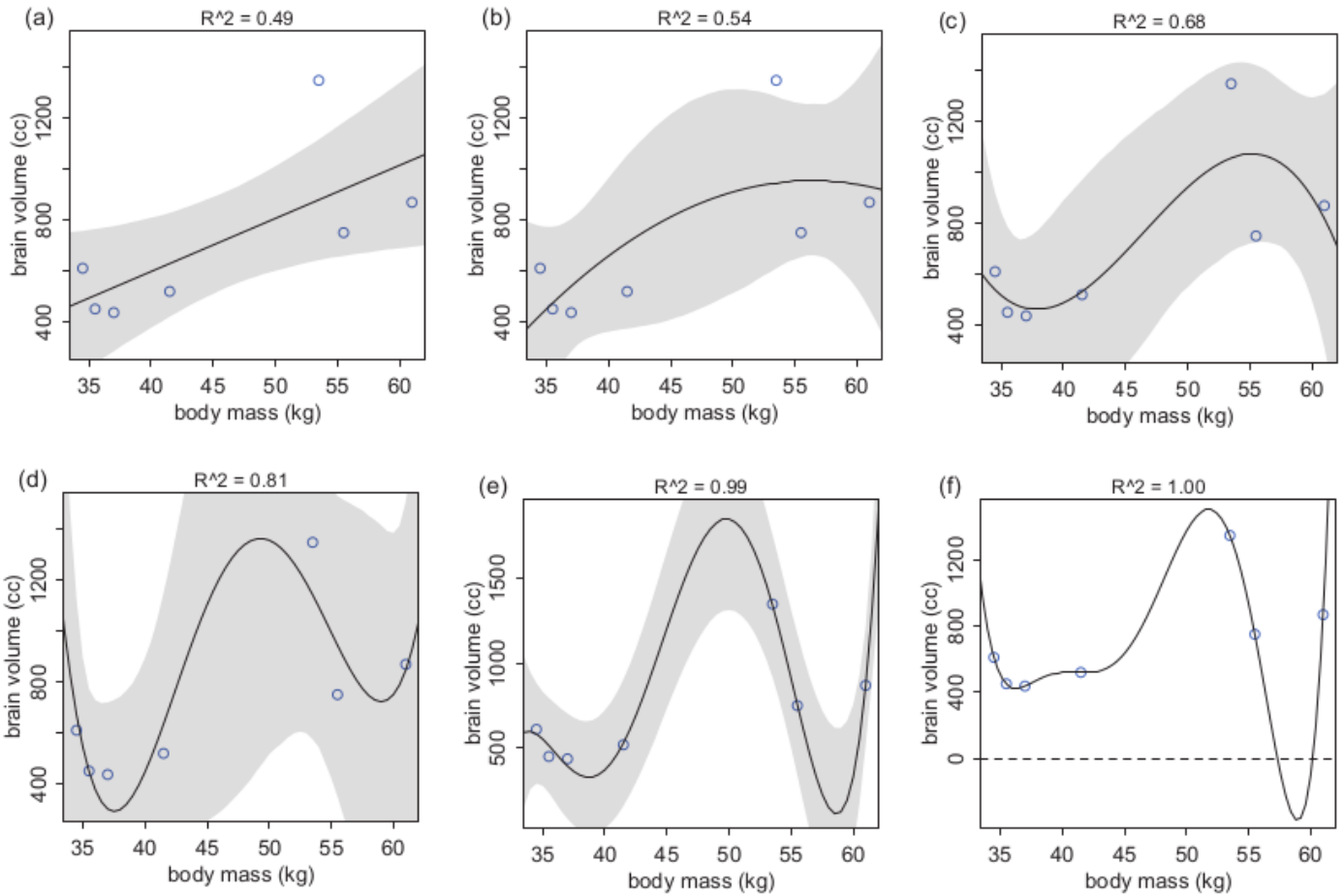
The problem with parameters

- Let's calculate R^2 for each of these models:

```
> summary(reg.ev.1)$r.squared
[1] 0.490158
> summary(reg.ev.2)$r.squared
[1] 0.5359967
> summary(reg.ev.3)$r.squared
[1] 0.6797736
> summary(reg.ev.4)$r.squared
[1] 0.8144339
> summary(reg.ev.5)$r.squared
[1] 0.988854
> summary(reg.ev.6)$r.squared
[1] 1
```

- As the degree of the polynomial defining the mean increases, the fit always improves.
- The sixth-degree polynomial actually has a perfect fit, $R^2 = 1$.

The problem with parameters



Polynomial linear models of increasing degree, fit to the hominin data. Each plot shows the predicted mean in black, with 89% interval of the mean shaded. R^2, is displayed above each plot. (a) First-degree polynomial. (b) Second-degree. (c) Third-degree. (d) Fourth-degree. (e) Fifth-degree. (f) Sixth-degree. Source: [McElreath, 2020].

The problem with parameters

- We can see from looking at the paths of the predicted means that the higher-degree polynomials are increasingly absurd.
- For example, **reg.ev.6** the most complex model makes a perfect fit, but the model is ridiculous.
- Notice that there is a gap in the body mass data, because there are no fossil hominins with body mass between 55 kg and about 60 kg.
- In this region, the models has nothing to predict, so it pays no price for swinging around wildly in this interval.
- The swing is so extreme that at around 58 kg, the model predicts a negative brain size!
- The model pays no price (yet) for this absurdity, because there are no cases in the data with body mass near 58 kg.

The problem with parameters

- Why does the sixth-degree polynomial fit perfectly?
- Because it has enough parameters to assign one to each point of data.
- The model's equation for the mean has 7 parameters:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \beta_5 x_i^5 + \beta_6 x_i^6 + \epsilon_i \quad \forall i$$

  and there are 7 species to predict brain sizes for.
- So effectively, this model assigns a unique parameter to reiterate each observed brain size.
- This is a general phenomenon: If you adopt a model family with enough parameters, you can fit the data exactly.
- But such a model will make rather absurd predictions for yet-to-be-observed cases.

Too few parameters hurts, too

- The overfit polynomial models manage to fit the data extremely well.
- But they suffer for this within-sample accuracy by making nonsensical out-of-sample predictions.
- In contrast, underfitting produces models that are inaccurate both within and out of sample.
- For example, consider this model of brain volume:

$$y_i = \beta_0 + \epsilon_i \quad \forall i$$

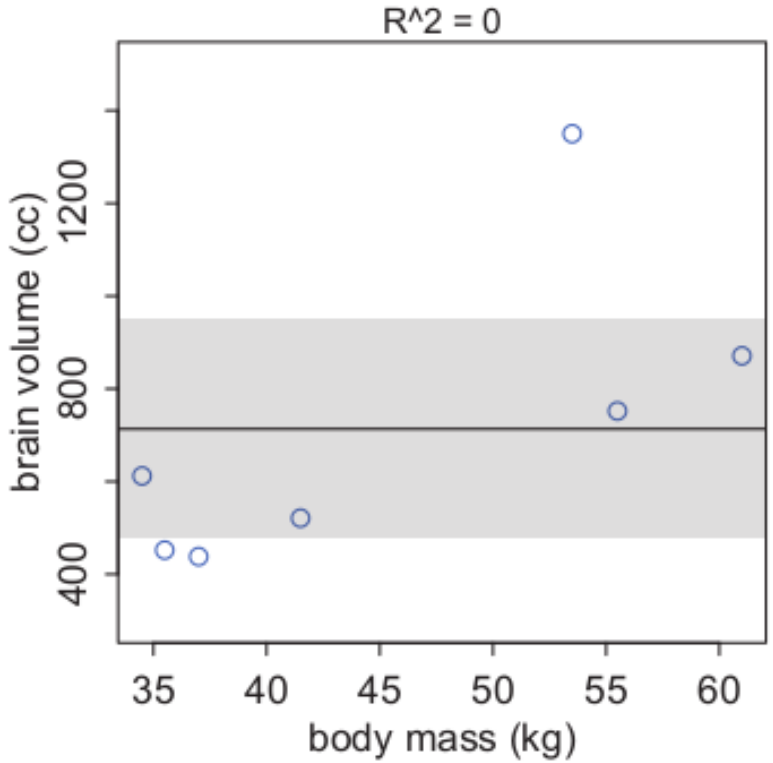
- There are no predictor variables here, just the intercept β_0.
- We can fit this model as follows:

```
> reg.ev.0 <- lm( brain ~ 1 , data=d )
> summary(reg.ev.0)$r.squared
[1] 0
```

- The value of R^2 is 0.

Too few parameters hurts, too

- This model estimates the mean brain volume, ignoring body mass.



- As a result, the regression line is perfectly horizontal and poorly fits both smaller and larger brain volumes.
- Such a model not only fails to describe the sample.
- It would also do a poor job for new data.

Roadmap

- The first thing we need to navigate between overfitting and underfitting problems is a criterion of model performance.
- We will see how **information theory** provides a useful criterion for model evaluation: the **out-of-sample deviance**.
- Once we learn about this criterion we will see how **cross-validation** and **information criteria** can help to estimate the out-of-sample deviance of a model.
- We will also learn how to use **regularization** to improve the out-of-sample deviance of a model.
- As usual, we will start with an example.

The Weatherperson

- Suppose in a certain city, a certain weatherperson issues uncertain predictions for rain or shine on each day of the year.
- The predictions are in the form of probabilities of rain.
- The currently employed weatherperson predicted these chances of rain over a 10-day sequence:

Day	1	2	3	4	5	6	7	8	9	10
Prediction	1	1	1	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Observed										

- A newcomer rolls into town, and this newcomer boasts that he can best the current weatherperson, by always predicting sunshine.
- Over the same 10 day period, the newcomer's record would be:

Day	1	2	3	4	5	6	7	8	9	10
Prediction	0	0	0	0	0	0	0	0	0	0
Observed										

The Weatherperson

- Define **hit rate** as the average chance of a correct prediction.
- So for the current weatherperson, she gets $3 \times 1 + 7 \times 0.4 = 5.8$ hits in 10 days, for a rate of $5.8/10 = 0.58$ correct predictions per day.
- In contrast, the newcomer gets $3 \times 0 + 7 \times 1 = 7$, for $7/10 = 0.7$ hits per day.
- The newcomer wins.
- Let's compare now the two predictions using another metric, the **joint likelihood**: $\prod f(y_i; \theta)$ for a frequentist model or $\prod f(y_i|\theta)$ for a Bayesian one.
- The joint likelihood corresponds to the joint probability of correctly predicting the observed sequence.
- To calculate it, we must first compute the probability of a correct prediction for each day.
- Then multiply all of these probabilities together to get the joint probability of correctly predicting the observed sequence.

The Weatherperson

- The probability for the current weather person is $1^3 \times 0.4^7 \approx 0.005$.
- For the newcomer, it's $0^3 \times 1^7 = 0$.
- So the newcomer has zero probability of getting the sequence correct.
- This is because the newcomer's predictions never expect rain.
- So even though the newcomer has a high average probability of being correct (hit rate), he has a terrible joint probability (likelihood) of being correct.
- And the joint likelihood is the measure we want.
- Because it is the unique measure that correctly counts up the relative number of ways each event (sequence of rain and shine) could happen.
- In the statistics literature, this measure is sometimes called the **log scoring rule**, because typically we compute the logarithm of the joint probability and report that.

Information and uncertainty

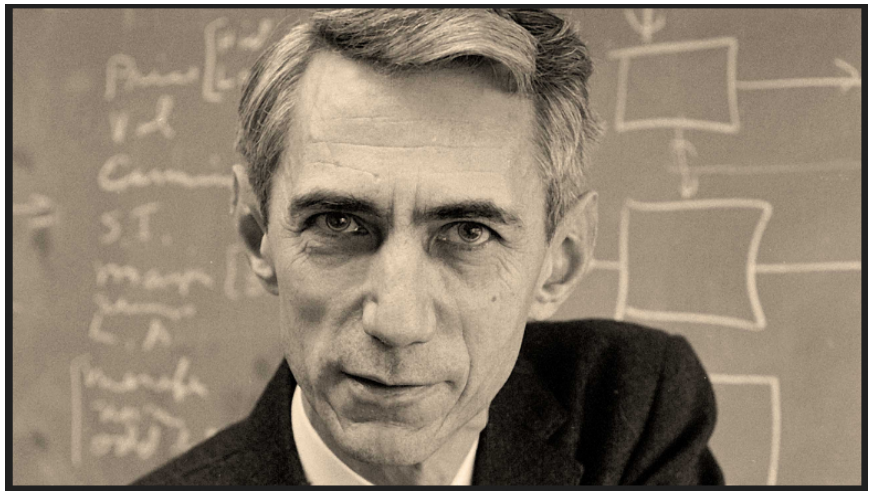
- So we want to use the log probability of the data to score the accuracy of competing models.
- The next problem is how to measure distance from perfect prediction.
- A perfect prediction would just report the true probabilities of rain on each day.
- So when either weatherperson provides a prediction that differs from the target, we can measure the distance of the prediction from the target.

Information and uncertainty

- What kind of distance should we adopt?
- Getting to the answer depends upon appreciating what an accuracy metric needs to do.
- It should appreciate that some targets are just easier to hit than other targets.
- For example, suppose we extend the weather forecast into the winter. Now there are three types of days: rain, sun, and snow.
- Now there are three ways to be wrong, instead of just two.
- This has to be reflected in any reasonable measure of distance from the target, because by adding another type of event, the target has gotten harder to hit.
- Before presenting a distance metric that satisfies the properties described above, we must introduce some concepts from information theory.

Information Theory

- The field of information theory, with Claude Shannon as one of its pioneering figures, was originally applied to problems of message communication, such as the telegraph.



- The basic insight is to ask: How can we quantify the uncertainty inherent in a probability distribution?

Entropy

- Information entropy is a function that measures the uncertainty on probability functions.
- Let X be a discrete random variable of m different possible events, with a probaility mass function f, the entropy of X is defnined as follows:

$$H(X) = -\mathbb{E}(\log f(x)) = -\sum_{i=1}^{m} f(x_i) \log f(x_i) \tag{1}$$

- Usually we use log base 2, in which case the entropy units are called **bits** [Murphy, 2021].
- If X is continuous with density function f, the entropy $H(X)$ takes the following form:

$$H(X) = -\mathbb{E}(\log f(x)) = -\int f(x) \log f(x) dx \tag{2}$$

- In plainer words: “The uncertainty contained in a probability distribution is the negative average log-probability of an event”.

Entropy

- An example will help to demystify the function $H(X)$.
- To compute the information entropy for the weather, suppose the true probabilities of rain ($X = 1$) and shine ($X = 2$) are $f(1) = \mathbb{P}(X = 1) = 0.3$ and $f(2) = \mathbb{P}(X = 2) = 0.7$, respectively.
- Then:

$$H(X) = -(f(1) \log f(1) + (f(2) \log f(2)) \approx 0.88$$

- As an R calculation:

```
> f <- c( 0.3 , 0.7 )
> -sum( f*log2(f) )
[1] 0.8812909
```

Entropy

- Suppose instead we live in Abu Dhabi.
- Then the probabilities of rain and shine might be more like $f(1) = 0.01$ and $f(2) = 0.99$.

```
> f <- c( 0.01 , 0.99 )
> -sum( f*log2(f) )
[1] 0.08079314
```

- Now the entropy is about 0.08.
- Why has the uncertainty decreased?
- Because in Abu Dhabi it hardly ever rains.
- Therefore there's much less uncertainty about any given day, compared to a place in which it rains 30% of the time.
- These entropy values by themselves don't mean much to us, though.
- Instead we can use them to build a measure of accuracy. That comes next.

Divergence

- How can we use information entropy to say how far a model is from the target?
- The key lies in divergence.
- Divergence: The additional uncertainty induced by using probabilities from one distribution to describe another distribution.
- This is often known as **Kullback-Leibler divergence** or simply K-L divergence, named after the people who introduced it for this purpose.
- Suppose for example that the true distribution of events is encoded by function f: $f(1) = 0.3$, $f(2) = 0.7$.
- Now, suppose we believe that these events happen according to another function q: $q(1) = 0.25$, $q(2) = 0.75$.
- How much additional uncertainty have we introduced, as a consequence of using q to approximate f?
- The answer is the the K-L divergence $D_{KL}(f, q)$.

Divergence

- If f and q are probability mass functions, the K-L divergence is defined as follows:

$$D_{KL}(f,q) = \sum_{i=1}^{m} f(x_i)(\log f(x_i) - \log q(x_i)) = \sum_{i=1}^{m} f(x_i) \log \left(\frac{f(x_i)}{q(x_i)} \right) \quad (3)$$

```
> f<-c(0.3,0.7)
> q<-c(0.25,0.75)
> sum(f* log2(f/ q))
[1] 0.00923535
```

- This naturally extends to continuous density functions as well [Murphy, 2021]:

$$D_{KL}(f,q) = \int f(x) \log \left(\frac{f(x)}{q(x)} \right) dx \quad (4)$$

- In plainer language, the divergence is the average difference in log probability between the target (f) and model (q).
- This divergence is just the difference between two entropies.
- The entropy of the target distribution f and the cross entropy arising from using q to predict f.

Cross entropy and divergence

- When we use a probability distribution q to predict events from another distribution f, this defines something known as cross entropy $H(f, q)$:

$$H(f, q) = -\sum_{i=1}^{m} f(x_i) \log q(x_i) \tag{5}$$

- The notion is that events arise according to f, but they are expected according to the q, so the entropy is inflated, depending upon how different f and q are.
- Divergence is defined as the additional entropy induced by using q.
- So it is the difference between $H(f)$, the actual entropy of events, and $H(f, q)$:

$$D_{KL}(f, q) = H(f, q) - H(f) \tag{6}$$

- So divergence really is measuring how far q is from the target f, in units of entropy.
- Notice that which is the target matters: $H(f, q)$ does not in general equal $H(q, f)$.

Divergence

- When $f = q$, we know the actual probabilities of the events and the divergence becomes zero:

$$D_{KL}(f, q) = D_{KL}(f, f) = 0$$

```
> q <- f
> sum(f*log2(f/ q))
[1] 0
```

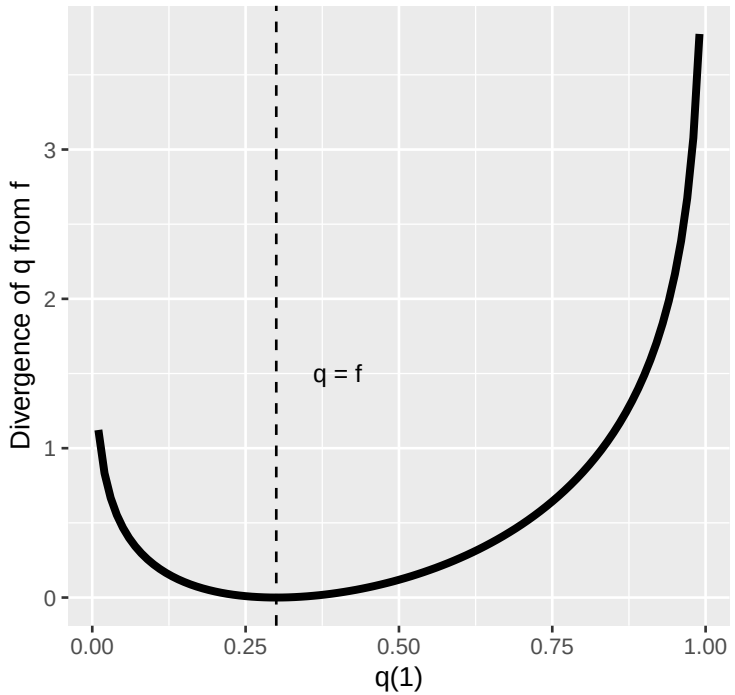
- As q grows more different from f, the divergence D_{KL} also grows.
- Suppose the true target distribution is $f = \{0.3, 0.7\}$.
- Suppose the approximating distribution q can be anything from $q = \{0.01, 0.99\}$ to $q = \{0.99, 0.01\}$.
- Let's build a plot with $q(1)$ on the horizontal axis and the divergence $D_{KL}(f, q)$ on vertical one.

Divergence

```
t <-
  tibble(f_1  = .3,
         f_2  = .7,
         q_1  = seq(from = .01, to = .99, by = .01)) %>%
  mutate(q_2  = 1 - q_1) %>%
  mutate(d_kl = (f_1 * log2(f_1 / q_1)) + (f_2 * log2(f_2 / q_2)))

t %>%
  ggplot(aes(x = q_1, y = d_kl)) +
  geom_vline(xintercept = .3,  linetype = 2) +
  geom_line(size = 1.5) +
  annotate(geom = "text", x = .4, y = 1.5, label = "q = f",
           size = 3.5) +
  labs(x = "q(1)",
       y = "Divergence of q from f")
```

Divergence



- Only exactly where $q = f$, at $q(1) = 0.3$, does the divergence achieve a value of zero. Everyplace else, it grows.
- Since predictive models specify probabilities of events (observations), we can use divergence to compare the accuracy of models.

Estimating divergence

- To use D_{KL} to compare models, it seems like we would have to know f, the target probability distribution.
- In all of the examples so far, I've just assumed that f is known.
- But when we want to find a model q that is the best approximation to f, the "truth," there is usually no way to access f directly.
- We wouldn't be doing statistical inference, if we already knew f.
- But there's an amazing way out of this predicament.
- It helps that we are only interested in comparing the divergences of different candidates, say q and r.
- In that case, most of f just subtracts out, because there is a $\mathbb{E}(\log f(x))$ term in the divergence of both q and r.
- This term has no effect on the distance of q and r from one another.

Estimating divergence

- So while we don't know where f is, we can estimate how far apart q and r are, and which is closer to the target.
- All we need to know is a model's average log-probability: $\mathbb{E}(\log q(x))$ for q and $\mathbb{E}(\log r(x))$ for r.
- To approximate the relative value of $\mathbb{E}(\log q(x))$, we can use the log-probability score of the model:

$$S(q) = \sum_{i=1}^{m} \log q(x_i) \tag{7}$$

- This sore is an estimate of $\mathbb{E}(\log q(x))$, just without the final step of dividing by the number of observations.
- It is also an approximation of the K-L divergence of the model relative to the unknown target model.

Calculating Log-Likelihood

- Most of the standard model fitting functions in R support “logLik” , which computes the sum of log-probabilities, usually known as the log-likelihood of the data.

```
> logLik(reg.ev.1)
'log Lik.' -47.46249 (df=3)
> logLik(reg.ev.2)
'log Lik.' -47.13276 (df=4)
```

- Recall that the absolute magnitude of these values are not interpretable.
- Only the difference $S(q) - S(r)$ informs us about the divergence of each model from the unknown target f.
- We can calculate the log-likelihood of a model by hand using the following code:

```
b0<-reg.ev.1$coefficients[1]
b1<-reg.ev.1$coefficients[2]

logLik1 <- sum(log(dnorm(
  d$brain ,
  mean=b0+b1*d$mass ,
  sd=sigma(reg.ev.1))
  ))
> logLik1
[1] -47.64015
```

Calculating Log-Likelihood

- Note that this log-likelihood differs from the one obtained with `logLik`.
- This is because `sigma(reg.ev.1)` returns an unbiased estimator of σ:

$$\hat{\sigma} = \sqrt{\frac{\sum \epsilon_i^2}{df}}$$

where $df = n - 2 = 7 - 2 = 5$.

```
> sigma(reg.ev.1)
[1] 252.0567
> reg.ev.1$df.residual
[1] 5
> sigma.LM <-sqrt(sum(reg.ev.1$residuals^2)/
+                     reg.ev.1$df.residual)
> sigma.LM
[1] 252.0567
```

Calculating Log-Likelihood

- On the other hand, the function `logLik` uses the maximum likelihood estimator of σ:

$$\hat{\sigma}_{ML} = \sqrt{\frac{\sum \epsilon_i^2}{n}}$$

```
sigma.ML<-sqrt(sum(reg.ev.1$residuals^2)
                /dim(d)[1])

> sigma.ML
[1] 213.0268

logLik1.1 <- sum(log(dnorm(
  d$brain ,
  mean=b0+b1*d$mass ,
  sd=sigma.ML)
))
> logLik1.1
[1] -47.46249
```

Calculating Log-Likelihood

- We can obtain the same results using MAP estimates from a Bayesian model with flat priors:

```
library(rethinking)
b.reg.ev.1<- quap(
  alist(
    brain ~ dnorm( mu , sigma ) ,
    mu <- b0 + b1*mass
  ) ,
  data=d ,
  start=list(b0=mean(d$brain),b1=0,sigma=sd(d$brain)) ,
  method="Nelder-Mead" )

theta <- coef(b.reg.ev.1)
logLik1.2 <- sum(log(dnorm(
  d$brain ,
  mean=theta[1]+theta[2]*d$mass ,
  sd=theta[3])
  ))

> logLik1.2
[1] -47.46249
```

Deviance

- It is also quite common to see something called the **deviance**, which is $S(q)$ multiplied by -2.

$$D(q) = -2 \times S(q) = -2 \sum_{i=1}^{n} \log q(x_i) \tag{8}$$

- The 2 is there for historical reasons.
- When comparing the deviance, smaller values are better.

```
> -2*logLik(reg.ev.1)
'log Lik.' 94.92499 (df=3)
> -2*logLik(reg.ev.2)
'log Lik.' 94.26553 (df=4)
> -2*logLik(reg.ev.3)
'log Lik.' 91.66948 (df=5)
> -2*logLik(reg.ev.4)
'log Lik.' 87.85016 (df=6)
```

From deviance to out-of-sample

- Deviance has the same flaw as R^2: It always improves as the model gets more complex.
- It is really the deviance on new data that interests us.
- When we usually have data and use it to fit a statistical model, the data comprise a **training sample**.
- Parameters are estimated from it.
- Then we can imagine using those estimates to predict outcomes in a new sample, called the **test sample**.
- We can compute the deviance on the test sample to obtain the **out-of-sample deviance**.
- Let's explore it with an example.

From deviance to out-of-sample

- Suppose we have the following data generation process and we fit linear regressions with between 1 and 5 free parameters to the data:

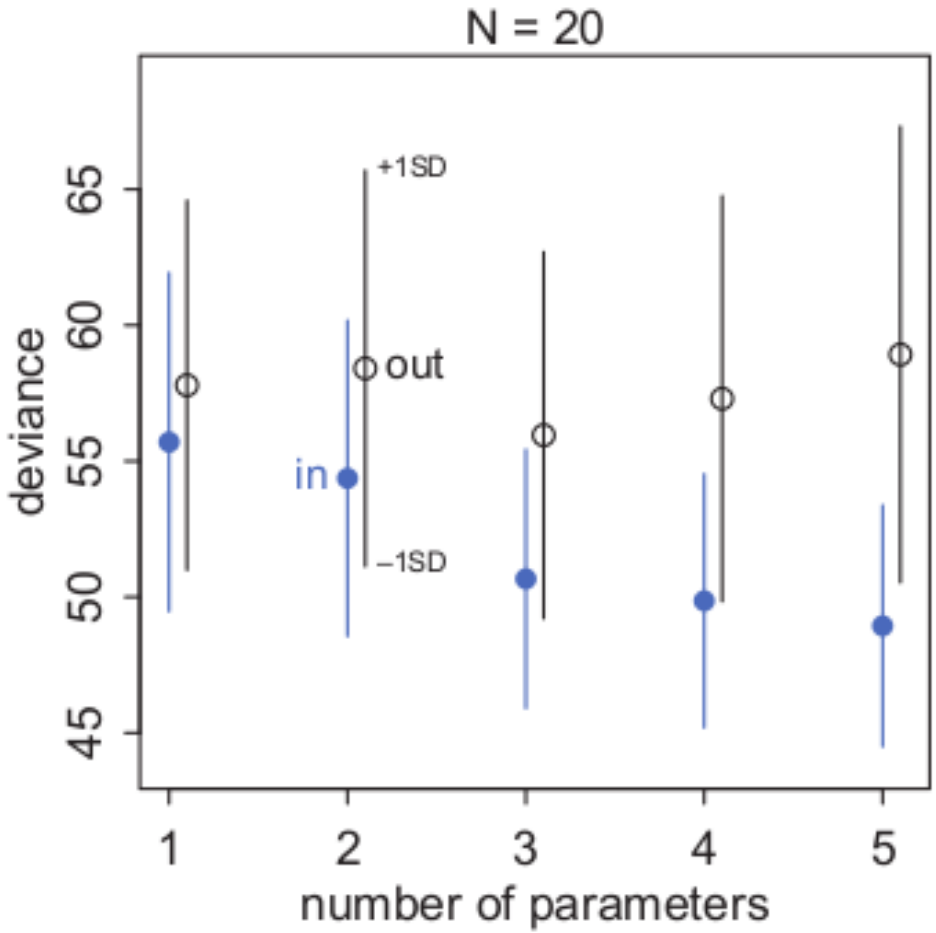
$$\text{Data generating model:} \quad y_i \sim \text{Normal}(\mu_i, 1)$$
$$\mu_i = (0.15)x_{1,i} - (0.4)x_{2,i}$$

$$\text{Models fit to data:} \quad \mu_i = \alpha$$
$$\mu_i = \alpha + \beta_1 x_{1,i}$$
$$\mu_i = \alpha + \beta_1 x_{1,i} + \beta_2 x_{2,i}$$
$$\mu_i = \alpha + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \beta_3 x_{3,i}$$
$$\mu_i = \alpha + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \beta_3 x_{3,i} + \beta_4 x_{4,i}$$

- Since the "true" model has non-zero coefficients for only the first two predictors, we can say that the true model has 3 parameters.
- In other words, x_3 and x_4 as uncorrelated with y.

From deviance to out-of-sample

- The following diagram shows in-sample and out-of-sample deviance scores for $10,000$ simulations of 20 training and 20 testing data points from our data generation process.



From deviance to out-of-sample

- Models with different numbers of predictor variables are shown on the horizontal axis.
- Deviance across 10,000 simulations is shown on the vertical.
- Blue shows deviance in-sample, the training data.
- Black shows deviance out-of-sample, the test data.
- Points show means, and the line segments show $\pm$ 1 standard deviation.
- A smaller deviance means a better fit.

From deviance to out-of-sample

- Regarding the in-sample deviance we see that with increasing numbers of parameters, the average deviance declines.
- This is the same phenomenon we saw earlier with R^2.
- On the other hand, the out-of-sample deviance is smallest on average for 3 parameters, which is the data-generating model
- And it gets worse (increases) with the addition of each parameter after the third.
- The point of this thought experiment is to demonstrate how deviance behaves, in theory.
- While deviance on training data always improves with additional predictor variables, deviance on future data may or may not.

Regularization

- The root of overfitting is a model's tendency to get overexcited by the training sample.
- One way to prevent a linear model to overfit the training data is regularization.
- From a frequentist point of view, the most popular regularization technique is **ridge regression**.
- In this approach we penalize large parameter values β in the likelihood function.
- We add a penalization term to the target function SSE to keep the squares of the parameter values low:

$$\text{argmin}_\beta \, SSE(\beta) + \lambda \sum_{j=0}^{m} \beta_j^2 \tag{9}$$

- The value of λ is an hyper-parameter that controls the ammount of overfitting.

Regularization

- If λ is zero, we are just doing least mean squares.
- If it is too large, on the other hand, we risk underfitting.
- This value is usually "tuned" on an independent partition of data referred to as "validation set".
- The function lm.ridge built into R's MASS library implements the ridge regression method.

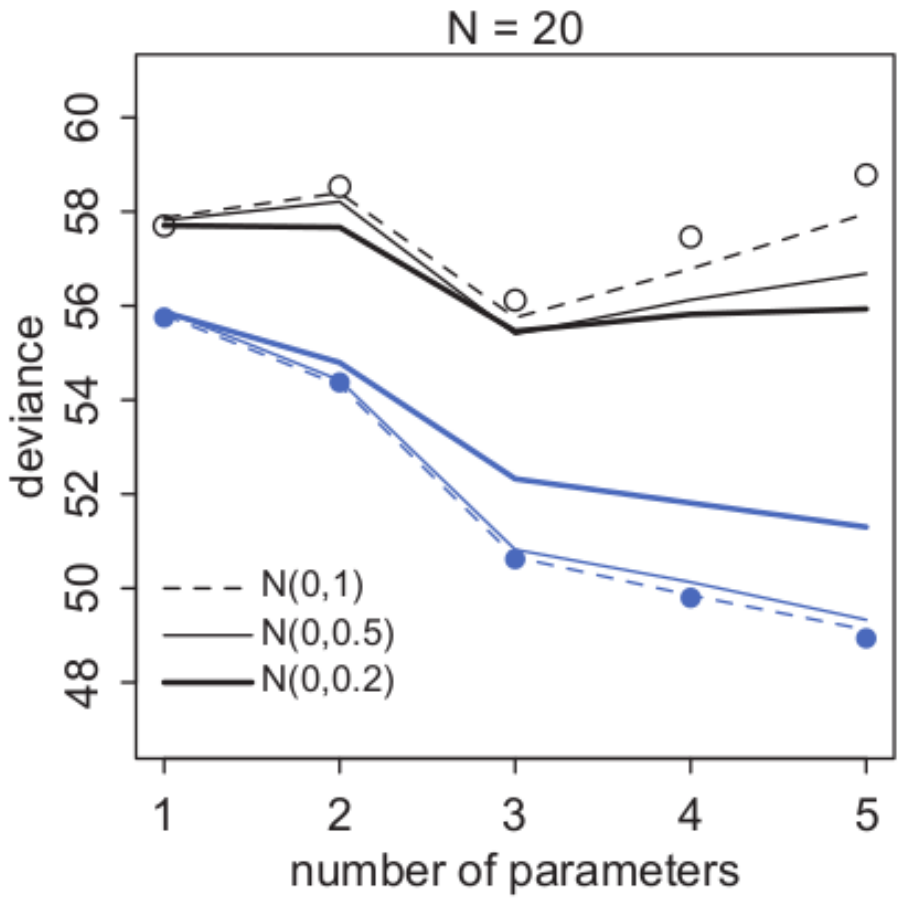
```
library(MASS)
rid.ev.4<- lm.ridge(brain ~ mass + I(mass^2)
          + I(mass^3) + I(mass^4),data=d,lambda = 0.1)

> rid.ev.4
                          mass      I(mass^2)
 6.725187e+01   9.751407e+00   7.235780e-02
    I(mass^3)      I(mass^4)
 4.874209e-04 -1.289110e-06
```

Regularization

- Ridge regression is an example of how a statistical procedure can be understood from both Bayesian and non-Bayesian perspectives.
- From a Bayesian point of view, ridge regression is equivalent to setting a Gaussian prior centered at zero for each β coefficient.
- In this setting, the value of σ controls the ammount of regularization.
- For example, a prior $\beta \sim$Normal$(0, 1)$ says that, before seeing the data, the machine should be very skeptical of values above 2 and below -2.
- In the next slide, we repeat the experiment of computing the in-sample and out-of-sample deviance for models with different number of attributes using different regularization priors.
- The data generation process is the same as before.

Regularization



- The points in both plots are the same as in previous figure (no regularization).
- The lines show training (blue) and testing (black) deviance for three regularizing priors.
- Dashed: Each beta-coefficient is given a Normal(0, 1) prior. Thin solid: Normal(0, 0.5). Thick solid: Normal(0, 0.2).

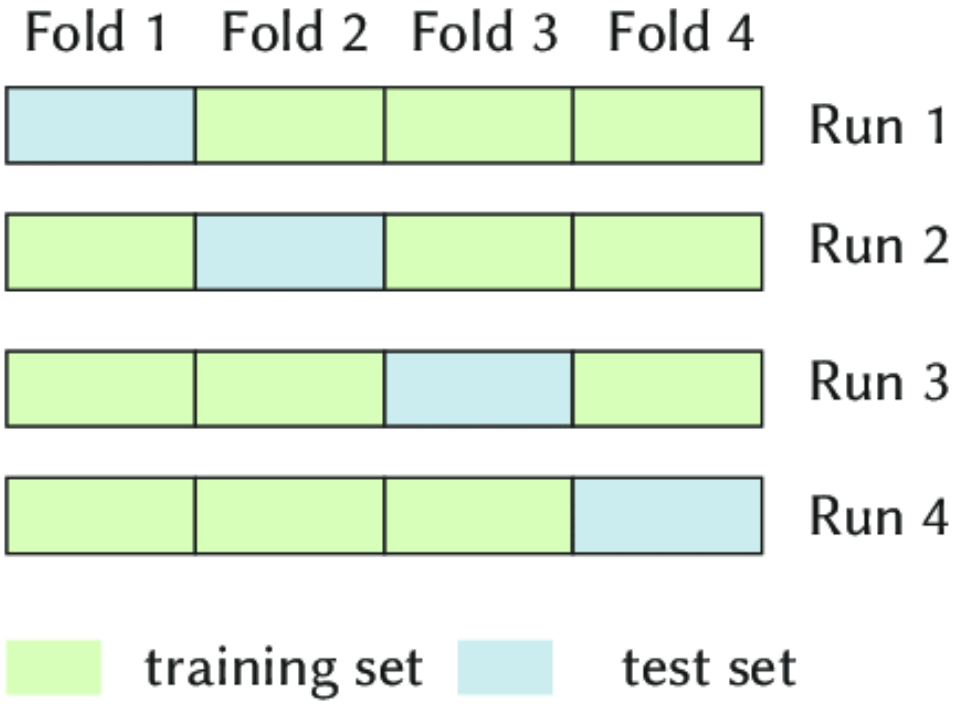
Regularization

- The training deviance always increases (gets worse) with tighter priors.
- The thick blue trend is substantially larger than the others.
- This is because the skeptical prior prevents the model from adapting completely to the sample.
- But the test deviances, out-of-sample, improve (get smaller) with the tighter priors.
- The model with three parameters is still the best model out-of-sample.
- The regularizing priors have little impact on its deviance.
- But also notice that as the prior gets more skeptical, the harm done by an overly complex model is greatly reduced.
- For the Normal(0, 0.2) prior (thick line), the models with 4 and 5 parameters are barely worse than the correct model with 3 parameters.
- If we can tune the regularizing prior right, then overfitting can be greatly reduced.

Cross Validation

- We just showed how to estimate the out-of-sample deviance by evaluating the model on observations that were not used to fit the model.
- However, in many cases, we do not want to waste a portion of our data just for evaluation.
- So what is usually done is **cross-validation**: to divide the sample in a number of chunks, called “folds”.
- Then the model is asked to predict each fold, after training on all the others.
- We then average over the score (e.g., deviance) for each fold to get an estimate of out-of-sample performance.
- The minimum number of folds is 2.
- At the other extreme, you could make each point observation a fold and fit as many models as you have individual observations.
- This is called leave-one-out cross-validation (often abbreviated as LOOCV).
- The key trouble with leave-one-out cross-validation is that, if we have 1000 observations, that means computing 1000 models.
- That can be time consuming.

Cross Validation



Information criteria

- Information criteria are a group of scoring devices that construct a theoretical estimate of the relative out-of-sample deviance using just the training data.
- They are a cheaper alternative to cross-validation.
- The most known information criterion is the **Akaike information criterion**, abbreviated AIC.
- AIC provides a surprisingly simple estimate of the average out-of-sample deviance:

$$AIC = D_{\text{train}} + 2p \tag{10}$$

  where D_{train} is the in-sample deviance, and p is the number of free parameters to be estimated in the model.
- In a linear regression model, p equals to the number of β coefficients (including the intercept) plus 1, to account to the model's standard deviation σ.

Akaike information criterion

- We can calculate AIC for the “reg.ev.1” model as follows:

```
> -2*logLik(reg.ev.1)+2*3
'log Lik.' 100.925 (df=3)
```

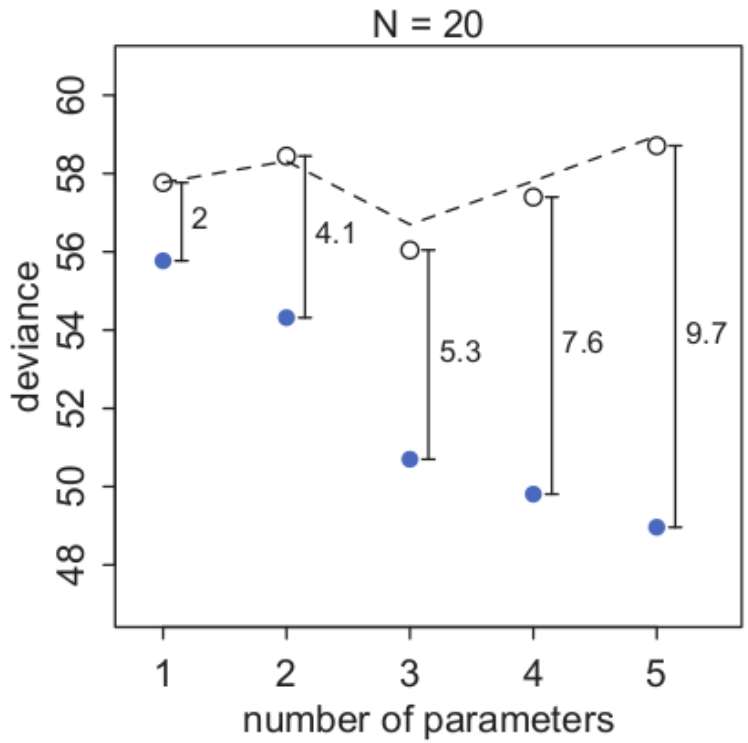
- In this model $p = 3$, two β coefficients (β_0, β_1) and σ.
- AIC can be computed directly in R using the “AIC” function:

```
> AIC(reg.ev.1)
[1] 100.925
> AIC(reg.ev.2)
[1] 102.2655
> AIC(reg.ev.3)
[1] 101.6695
> AIC(reg.ev.4)
[1] 99.85016
> AIC(reg.ev.5)
[1] 82.16384
```

- In AIC, the lower the better, so the criterion is still preferring overfitted models in this example.

Akaike information criterion

- Let's try to understand with an example where the AIC formula comes from.[1]
- The figure below shows again the in-sample and out-of-sample deviances for the simulated data.



[1] A more detailed derivation can be found here:
https://barumpark.com/blog/2018/aic-and-bic/

Akaike information criterion

- The vertical line segments measure the average distance between training deviance (blue) and test deviance (open black).
- Each distance is nearly twice the number of parameters, as labeled on the horizontal axis.
- The dashed lines show exactly the blue points plus twice the number of parameters, tracing closely along the average out-of-sample deviance for each model.
- These lines therefore show AIC for each model, an approximation of the out-of-sample deviance.

Information criteria

- AIC provides an approximation of predictive accuracy, as measured by out-of-sample deviance.
- All information criteria aim at this same target, but are derived under more and less general assumptions.
- AIC is just the oldest and most restrictive.
- AIC is commonly used in frequentist models fitted with maximum likelihood estimation.
- When employed in a Bayesian setting, it assumes that the priors are flat and the posterior is approximately a multivariate Gaussian.
- Two more-general criteria are the Deviance Information Criterion (DIC) and the Widely Applicable Information Criterion (WAIC).
- DIC accommodates informative priors, but still assumes that the posterior is multivariate Gaussian.
- WAIC is more general yet, making no assumption about the shape of the posterior.
- We will only study DIC in this class.

Deviance Information Criterion (DIC)

- The Deviance Information Criterion (DIC) is a widely used and easy to compute Bayesian information criterion.
- DIC is essentially a version of AIC that is aware of informative priors.
- Like AIC, it assumes a multivariate Gaussian posterior distribution.
- This means if any parameter in the posterior is substantially skewed, and also has a substantial effect on prediction, then DIC like AIC can go horribly wrong.

Deviance Information Criterion (DIC)

- DIC is calculated from the posterior distribution of the training deviance.
- What does it mean for the deviance to have a posterior distribution?
- Since the parameters have a posterior distribution, and since the deviance is computed from the parameters, deviance must also have a posterior distribution.
- Classical “deviance” is defined at the MAP values.
- But in principle the posterior distribution provides information about predictive uncertainty.
- That uncertainty turns out to help us estimate out-of-sample deviance.

Deviance Information Criterion (DIC)

- So define D now as the posterior distribution of deviance.
- This means we compute deviance (on the training sample) for each set of sampled parameter values in the posterior distribution.
- So if we draw $10,000$ samples from the posterior, we compute 10,000 deviance values.
- Let $\overline{D}$ indicate the average of D.
- Also define $\hat{D}$ as the deviance calculated at the posterior mean.
- This means we compute the average of each parameter in the posterior distribution.
- Then we plug those averages into the deviance formula to get $\hat{D}$ out.
- Once you have $\overline{D}$ and $\hat{D}$, DIC is calculated as:

$$DIC = \overline{D} + (\overline{D} - \hat{D}) = \overline{D} + p_D \tag{11}$$

Deviance Information Criterion (DIC)

- The difference $\overline{D} - \hat{D} = p_D$ is analogous to the number of parameters used in computing AIC.
- It is an "effective" number of parameters that measures how flexible the model is in fitting the training sample.
- More flexible models entail greater risk of overfitting.
- So this p_D term is sometimes called a penalty term.
- It is just the expected distance between the deviance in-sample and the deviance out-of-sample.
- In the case of flat priors, DIC reduces directly to AIC, because the expected distance is just the number of parameters.
- But more generally, p_D will be some fraction of the number of parameters, because regularizing priors constrain a model's flexibility.
- The function "DIC" in the "rethinking" package will compute DIC for a model fit with "quap" or "ulam".

Deviance Information Criterion (DIC)

- Let's compare DIC for three Bayesian linear models for the Howell data.
- b.reg1: a simple linear regression of height using weight as a predictor.
- b.reg2: a multivariate linear regression of height using weight and age as predictors.
- b.reg3: a polynomial regression of height using weight and weight square as predictors.

```
library(rethinking)
data(Howell1)
d <- Howell1
d2 <- d[ d$age >= 18 , ]

b.reg1 <- quap(
  alist(
    height ~ dnorm( mu, sigma ),
    mu <- b0 + b1*weight,
    b0 ~ dnorm( 150 , 50 ) ,
    b1 ~ dnorm( 0 , 1) ,
    sigma ~ dunif( 0 , 50 )
  ) , data=d2 )
```

Deviance Information Criterion (DIC)

```
b.reg2 <- quap(
  alist(
    height ~ dnorm( mu, sigma ),
    mu <- b0 + b1*weight+b2*age,
    b0 ~ dnorm( 150 , 50 ) ,
    b1 ~ dnorm( 0 , 1) ,
    b2 ~ dnorm( 0 , 1) ,
    sigma ~ dunif( 0 , 50 )
  ) , data=d2 )

b.reg3 <- quap(
  alist(
    height ~ dnorm( mu, sigma ),
    mu <- b0 + b1*weight+b2*weight^2,
    b0 ~ dnorm( 150 , 50 ) ,
    b1 ~ dnorm( 0 , 1) ,
    b2 ~ dnorm( 0 , 1) ,
    sigma ~ dunif( 0 , 50 )
  ) , data=d2 )
```

Deviance Information Criterion (DIC)

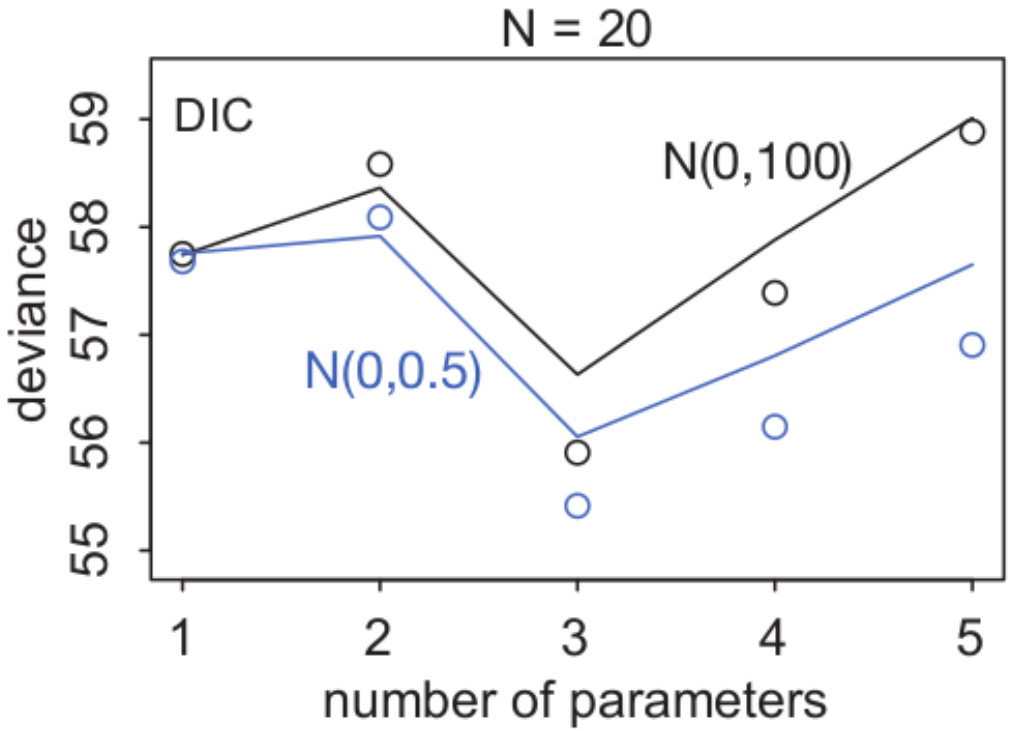
- Let's compute DIC for each model:

```
> DIC(b.reg1)
[1] 2148.409
attr(,"pD")
[1] 3.190277
> DIC(b.reg2)
[1] 2149.218
attr(,"pD")
[1] 3.941273
> DIC(b.reg3)
[1] 2149.858
attr(,"pD")
[1] 3.805133
```

- In this case the lowest (better) score in achieved by the simplest model.
- What we have just done is to use the DIC information criterion for model comparison.

Deviance Information Criterion (DIC)

- With the definition of DIC in hand, let's review one more simulation exercise.



- The figure above shows the results of 10,000 simulations each for the five familiar models with between 1 and 5 parameters.

Deviance Information Criterion (DIC)

- The plot displays only out-of-sample deviance now, for simplicity.
- The black points are average out-of-sample deviance resulting from simulations with nearly flat priors.
- The blue points result from simulations using regularizing Normal(0, 0.5) priors.
- The black and blue lines show the estimated out-of-sample deviance from DIC, with colors corresponding to groups of points.
- DIC is an accurate estimate of out-of-sample deviance, being within 1 point of deviance of the actual average in most cases.

Deviance Information Criterion (DIC)

- Also notice that the regularizing prior (blue) still helps, and that DIC tracks this help.
- This suggests that using both regularization and information criteria will always beat using only one or the other alone.
- Regularization, as long as it's not too strong, reduces overfitting for any particular model.
- Information criteria instead help us measure overfitting across models fit to the same data.
- These are complementary functions.
- And since both are very easy to use and widely available, there's no reason to shy away from their use.

Conclusions

- This class has been a marathon.
- It began with the problem of overfitting, a universal phenomenon by which models with more parameters fit a sample better, even when the additional parameters are meaningless.
- Theree common tools were introduced to address over-fitting: regularization, cross-validation, and information criteria.
- Regularization reduces overfitting during estimation, and both cross-validation and information criteria help estimate the degree of overfitting.
- In all cases, keep in mind that these tools are heuristic.
- They provide no guarantees.
- No statistical procedure will ever substitute for iterative scientific investigation.

References I


McElreath, R. (2020).
Statistical rethinking: A Bayesian course with examples in R and Stan.
CRC press.


Murphy, K. P. (2021).
Probabilistic Machine Learning: An introduction.
MIT Press.