Model Evaluation and Information Criteria

Felipe José Bravo Márquez

September 30, 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**: moderls that learn too little from the data, which also leads to poor prediction.
- We will study two common families of approaches to tackle these problems.
  - **Regularization**: a mechanism to tell our models not to get too excited by the data.
  - **Information criteria**: a scoring device 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.

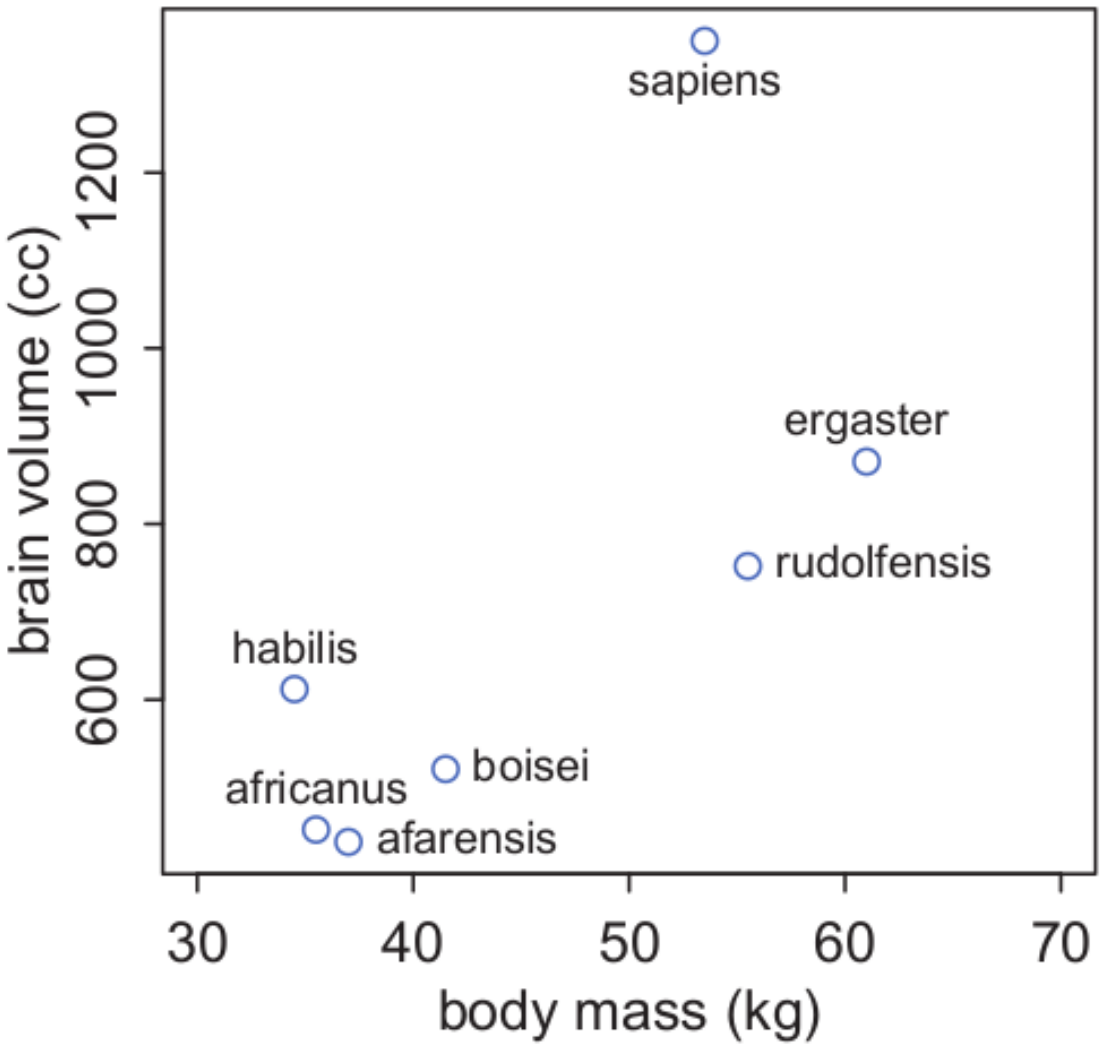
The problem with parameters

- 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 )
```

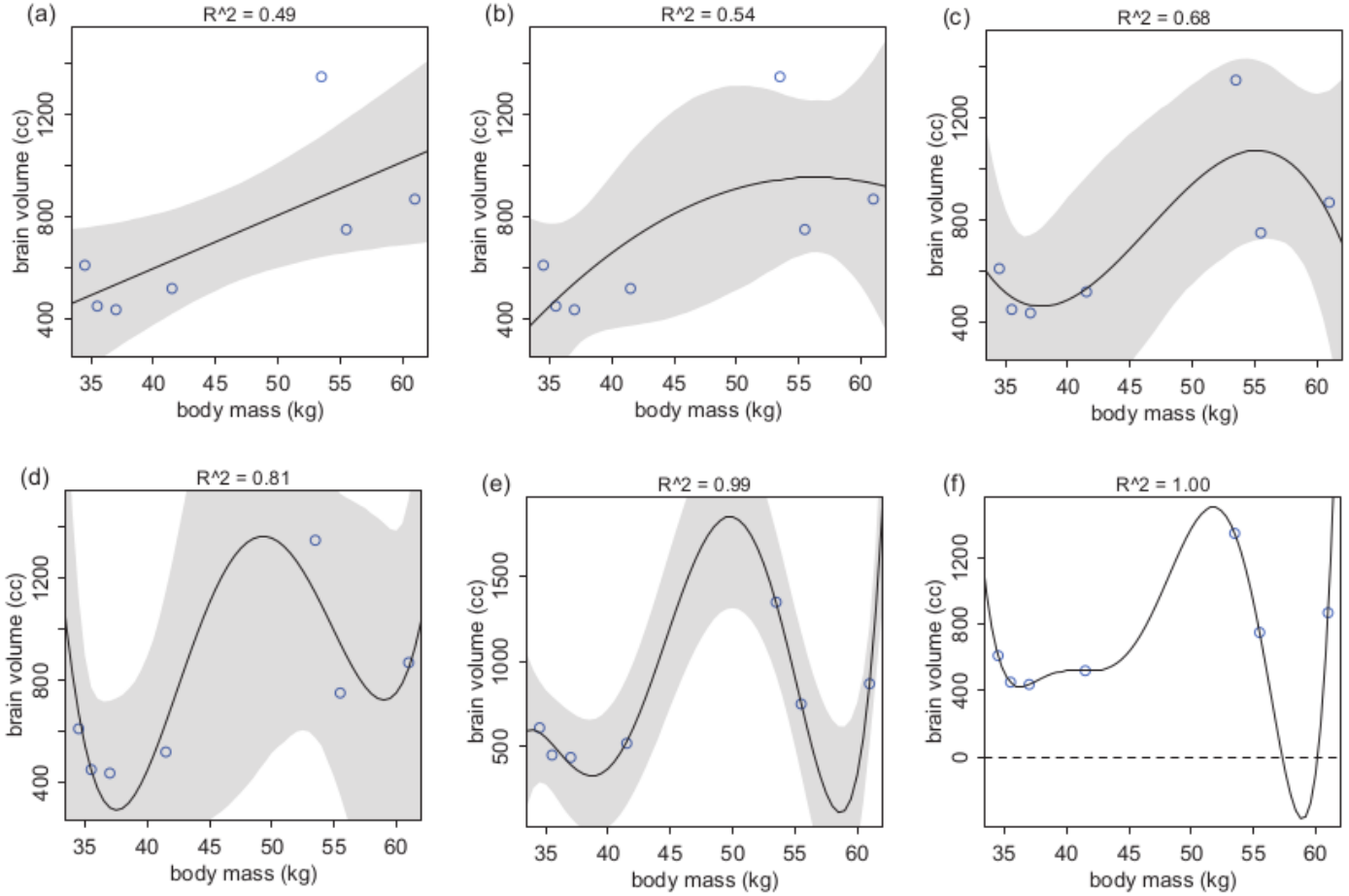
The problem with parameters

- 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:

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

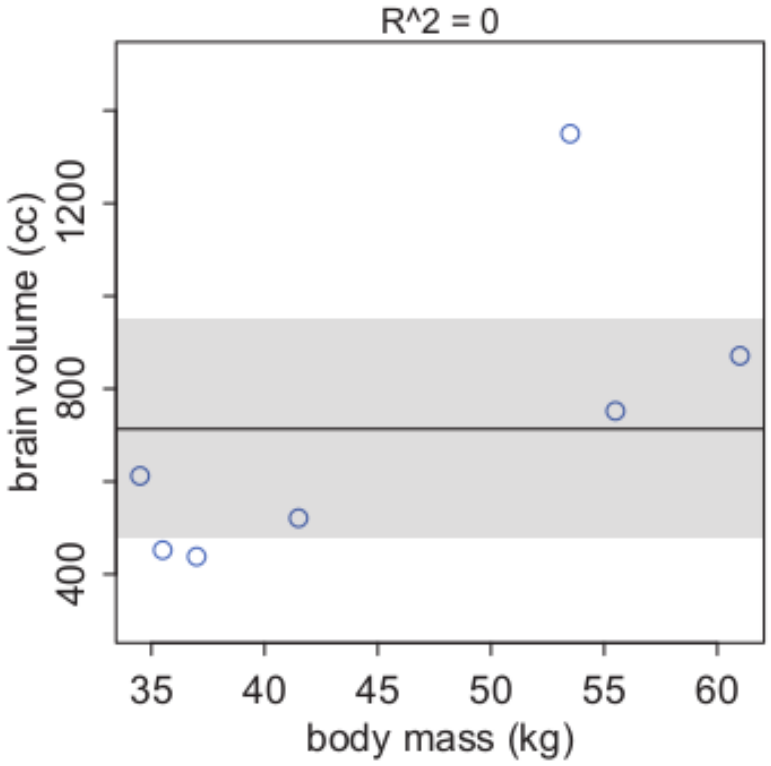
- There are no predictor variables here, just the intercept β_0.
- We can fit this model as follows:

```
> 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 cretierion we will se how both regularization and information criteria help to improve and estimate the out-of-sample deviance of a model.
- As usual, we will introduce these concepts 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

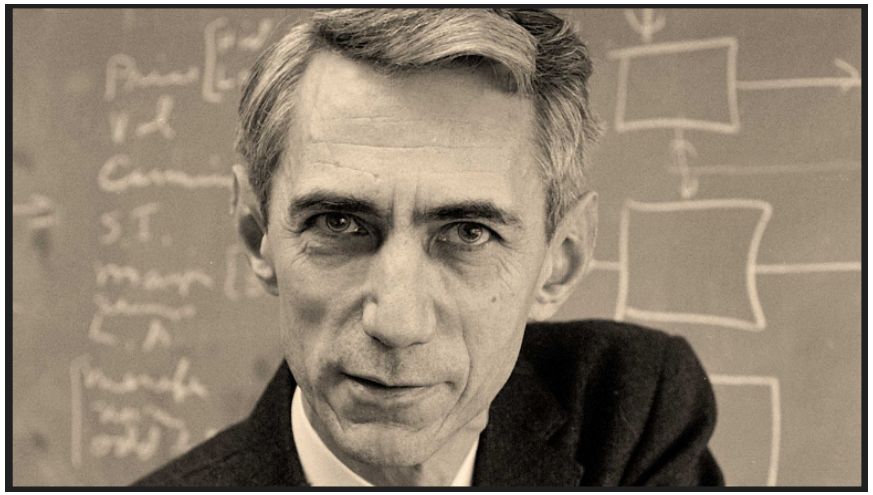
- So we want to use the log probability of the data to score the accuracy of competing models.
- The next problem is how to measure distance from perfect prediction.
- A perfect prediction would just report the true probabilities of rain on each day.
- So when either weatherperson provides a prediction 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 much is our uncertainty reduced by learning an outcome?
- To answer this question we need a way to 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:

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

```
> 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]:

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

- 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.
- In that case:

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

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

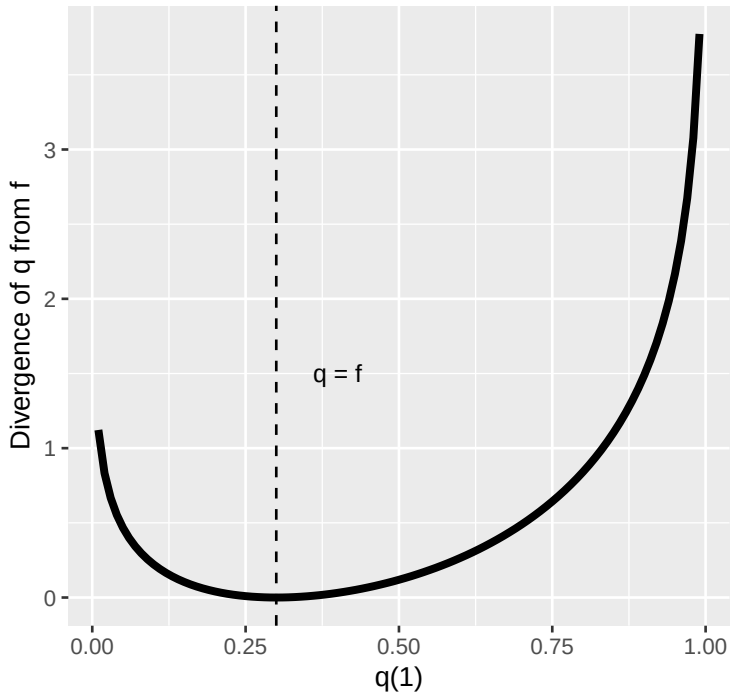
- But more importantly, as q grows more different from f, the divergence D_{KL} also grows.
- Suppose the true target distribution is $f = \{0.3, 0.7\}$.
- Suppose the approximating distribution q can be anything from $q = \{0.01, 0.99\}$ to $q = \{0.99, 0.01\}$.
- Let's build a plot with $q(1)$ on the 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 of this also means that 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.
- Indeed, just summing the log-probabilities of each observed case provides an approximation of $\mathbb{E}(\log q(x))$ and $\mathbb{E}(\log r(x))$.
- This also means that the absolute magnitude of these values will not be interpretable.
- Only the difference $\mathbb{E}(\log q(x)) - \mathbb{E}(\log r(x))$ informs us about the divergence of each model from the target f.

Estimating divergence

- All of this delivers us to a very common measure of relative model fit, one that also turns out to be an approximation of K-L divergence.
- 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.
- 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)
```

Deviance

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

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

- 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:

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

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

- 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.

Regularization

- Blabla

Information criteria

- Blabla

Using information criteria

- Blabla

Conclusions

- Blabla

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.