Information-Theoretic Modeling

Lecture 9: The MDL Principle

Teemu Roos

Department of Computer Science, University of Helsinki

Fall 2014

Lecture 9: MDL Principle

Jorma Rissanen (left) receiving the IEEE Information Theory Society Best Paper Award from Claude Shannon in 1986.

IEEE Golden Jubilee Award for Technological Innovation (for the invention of arithmetic coding) 1998; **IEEE Richard W. Hamming Medal** (for fundamental contribution to information theory, statistical inference, control theory, and the theory of complexity) 1993; **Kolmogorov Medal** 2006; **IBM Outstanding Innovation Award** (for work in statistical inference, information theory, and the theory of complexity) 1988; **IEEE Claude E. Shannon Award** 2009; ...

Prediction Game

Suppose we have an urn (a population) of red and blue balls.

We don't know the ratio of red balls vs blue balls in the urn.

We start picking balls from the urn one by one and return each ball back into the urn after recording its color.

What is the probability that the first ball is red?

Prediction Game

Suppose we have an urn (a population) of red and blue balls.

We don't know the ratio of red balls vs blue balls in the urn.

We start picking balls from the urn one by one and return each ball back into the urn after recording its color.

What about the probability that the second ball is red given that the first ball was red?

Prediction Game

Since the drawing of the balls from the urn is i.i.d. there is some fixed probability $\Pr[\text{red}] = P$.

If we knew P (which we don't), we'd predict $\Pr[\text{red} \mid \text{red}] = P$, etc.

Bayesian solution: If we were uncertain about P but had a prior distribution (pdf) $w(p)$, then we could use a mixture model:

$$\Pr[\text{red} \mid \text{red}] = \int p\, w(p \mid \text{red}) dp \ ,$$

where $w(p \mid \text{red})$ is the posterior distribution given by $w(p \mid \text{red}) = p\, w(p) / \Pr[\text{red}]$.

If the prior distribution is expressed as a Beta density $w(p) = \mathrm{Beta}(p\,;\, \alpha, \beta)$, then the posterior is given by $w(p \mid \text{red}) = \mathrm{Beta}(p\,;\, \alpha, \beta + 1)$.

Prediction Game

The general rule is
$w(p \mid x_1, \ldots, x_n) = \mathrm{Beta}(p\,;\, \alpha + n_{\text{red}}, \beta + n_{\text{blue}})$.

The predictive distribution becomes

$$\Pr[\textit{red} \mid x_1, \ldots, x_n] = \frac{\beta + n_{\text{red}}}{n + \alpha + \beta} \quad .$$

As $n \to \infty$, the effect of the prior vanishes and we end up predicting $\Pr[\text{red} \mid x_1, \ldots, x_n] = n_{\text{red}}/n + o(1)$.

This is the ML estimate: **optimal, if we knew it in advance!**

The *hyperparameters* α and β act as imaginary observations:
– big values make us ssllooww to follow the ML,
– small values are risky (can you see why?)

No Such Things as an Uninformative Prior

Where do we get the prior from?

We don't really have reason to believe that any particular value $0 \leq p \leq 1$ would be more likely than another.

Therefore, we should perhaps assign a uniform prior probability on P: $w(p) = \mathrm{Beta}(p\,;\,1,1) = 1$.

What about the square of the probability, P^2? By the same argument, we should also assign a uniform prior on P^2!

In general, it is very hard or impossible to represent complete ignorance in terms of an "uninformative" prior.

Prediction Game with Utility Function

Now suppose that for each picked ball, you'll get € $\log_2 2q$ where q is the probability you quoted for the actual outcome.

This makes it possible to evaluate our prediction rule by summing up the money at the end of the game:

$$\sum_{i=1}^{n} \log_2 2q(x_i \mid x_1, \ldots, x_{i-1}) = 2 \log_2 \prod_{i=1}^{n} q(x_i \mid x_1, \ldots, x_{i-1})$$
$$= 2 \log_2 q(x_1, \ldots, x_n) \ .$$

Important: Sum of log scores equals log of joint probability.

Some priors are better for some sequences, other priors for other sequences. Which sequence will occur? $\Rightarrow$ Circular argument.

Prediction Game: Information-Theoretic Approach

The log score also equals minus the code-length: $\ell = -\log_2 q$.

The **redundancy** is defined as the difference between the expected code-length using q and the code-length that is based on the (unknown) true p:

$$\text{redundancy}(q, p) = \text{KL}(p \,||\, q) \ .$$

The **regret** is defined as the difference between the actual code-length using q and the shortest possible code-length using whatever (fixed) p:

$$\text{regret}(q, x_1, \ldots, x_n) = -\log_2 q(x_1, \ldots, x_n) - \min_{0 \le p \le 1} [-\log_2 p^{n_{\text{red}}} (1-p)^{n_{\text{blue}}}]$$

Prediction Game: Information-Theoretic Approach

An information-theoretic approach is to minimize the **worst-case** regret or redundancy.

For any universal model, q, the worst-case per-symbol regret or redundancy goes to zero.

The rate of convergence to zero depends on the prior.

$\Rightarrow$ An objective criterion for choosing priors.

Of course, q need not be a mixture universal model: *optimal* universal model is NML (but need to know the length of the game, n).

Prediction Game: More Flexibility

Suppose now that every other ball is drawn from a different urn.

The proportions of red balls may or may not be the same.

We can model odd and even draws using two independent universal models, or we can use the same universal model for both.

Prediction Game: More Flexibility

Exercise: Check that

1. if the proportions are the same, it is better to use a single universal model, $q(x_1, x_2, \ldots)$,
2. if the proportions are different, it is *much* better to use two independent universal models,
   $q_{\text{two}}(x_1, x_2, \ldots) = q_{\text{odd}}(x_1, x_3, \ldots) \times q_{\text{even}}(x_2, x_4, \ldots)$.

This can be used to **test** whether the proportions are the same!

Prediction Game: More Flexibility

More generally, we can build more and more flexible universal models by introducing more parameters.

For example:

- We can model multivariate data (vectors) by allowing no interactions, two-way interactions, three-way interactions, etc.
- Or for time series data, we can build Markov chains of increasing order.

The price we have to pay for flexibility is that the redundancy and the regret grow: we lose more compared to the ML model.

For smooth parametric models, asymptotically the price (code-length) is $\frac{1}{2} \log_2 n$ bits for each free parameter.

House

House

Brandon has

1. cough,
2. severe abdominal pain,
3. nausea,
4. low blood pressure,
5. fever.

1. pneumonia,common cold,
2. appendicitis,gout medicine,
3. food poisoning,gout medicine,
4. hemorrhage,gout medicine,
5. meningitis.common cold.

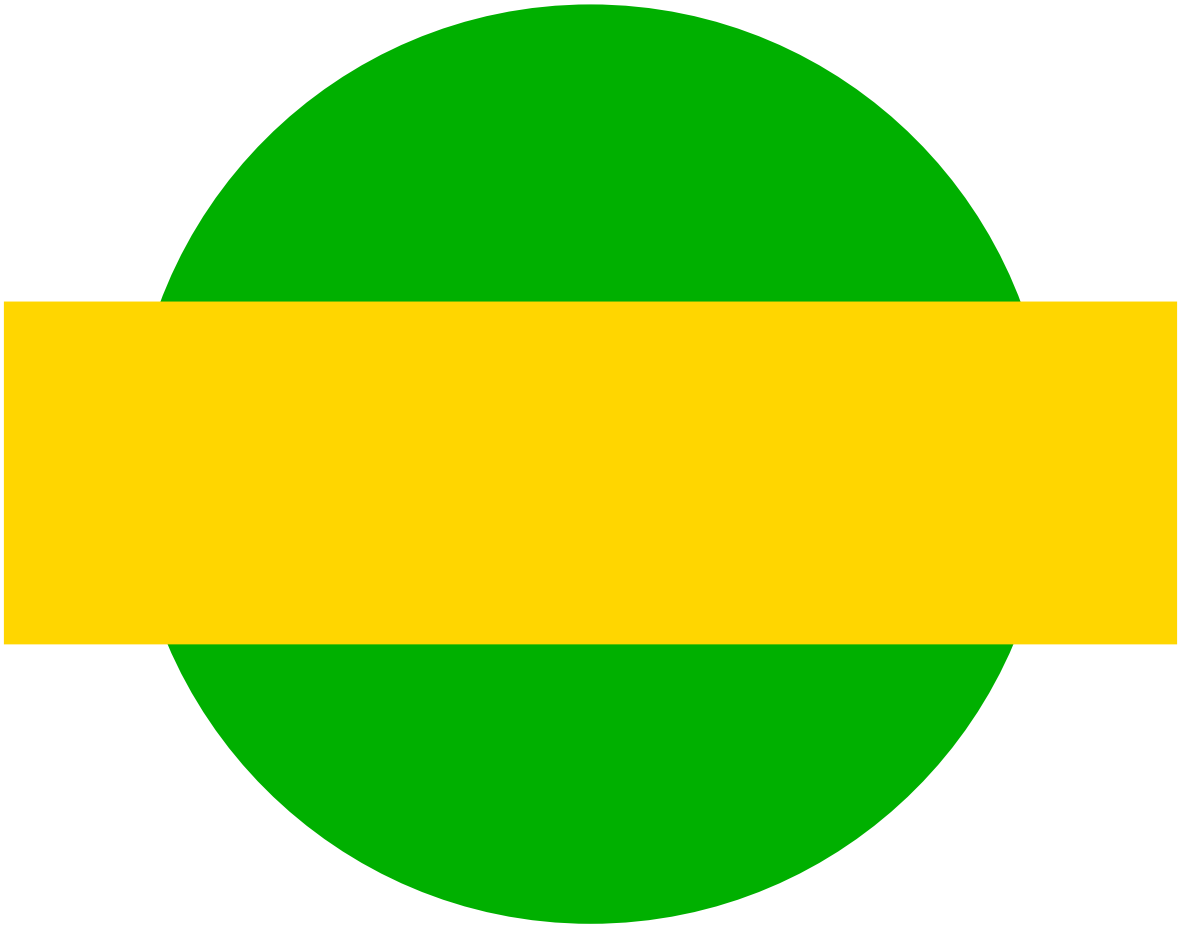
No single disease causes all of these.

Each symptom can be caused by *some* (possibly different) disease...

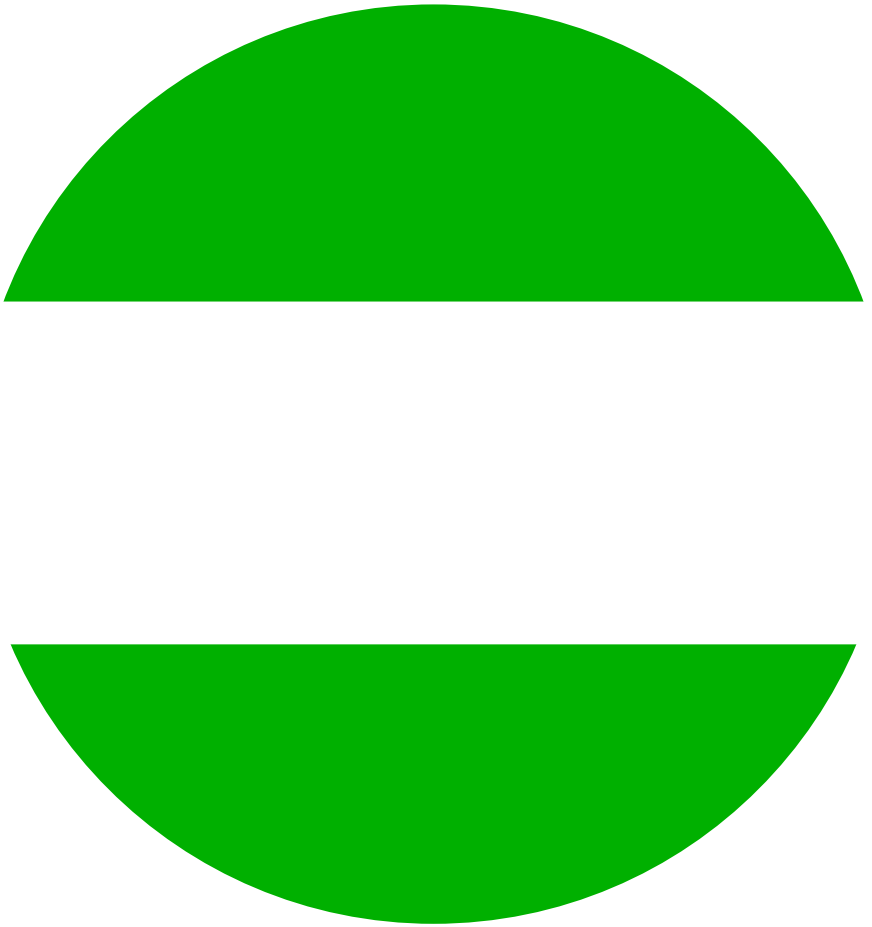
Dr. House explains the symptoms with two simple causes:

1. common cold, causing the cough and fever,
2. pharmacy error: cough medicine replaced by gout medicine.

Visual Recognition

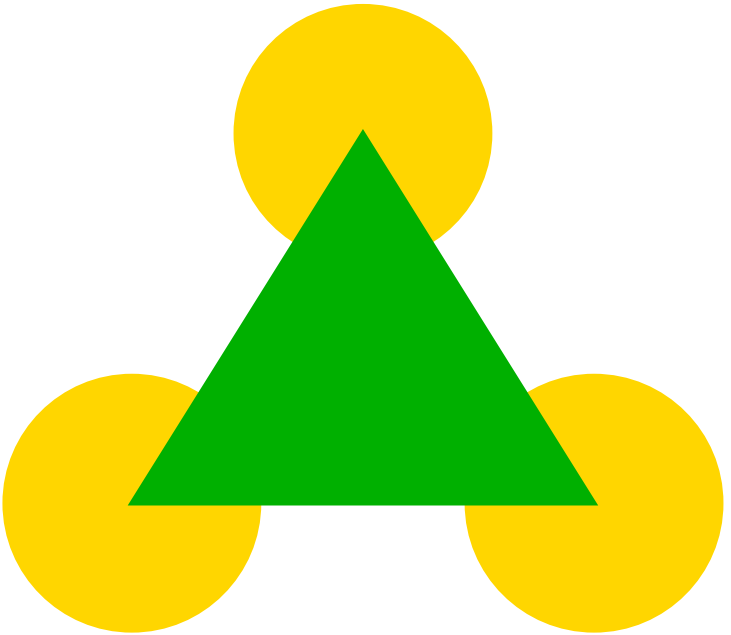


Visual Recognition

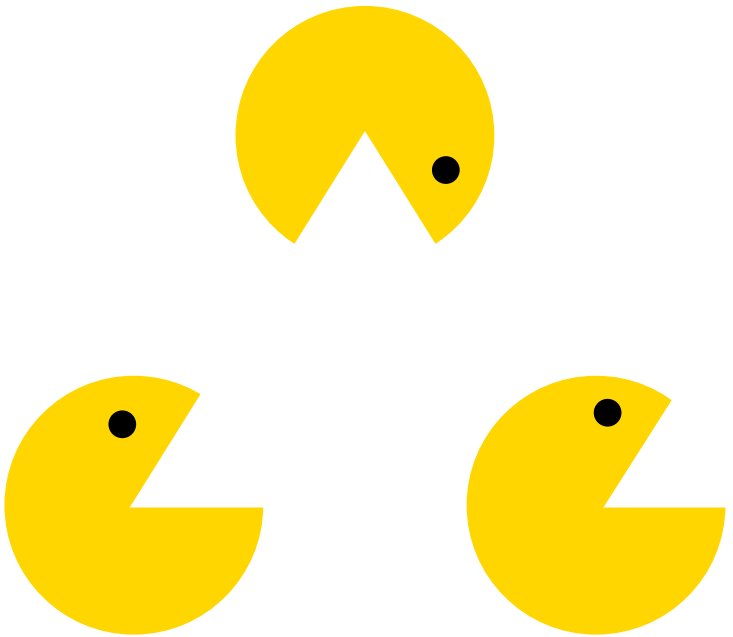


Visual Recognition

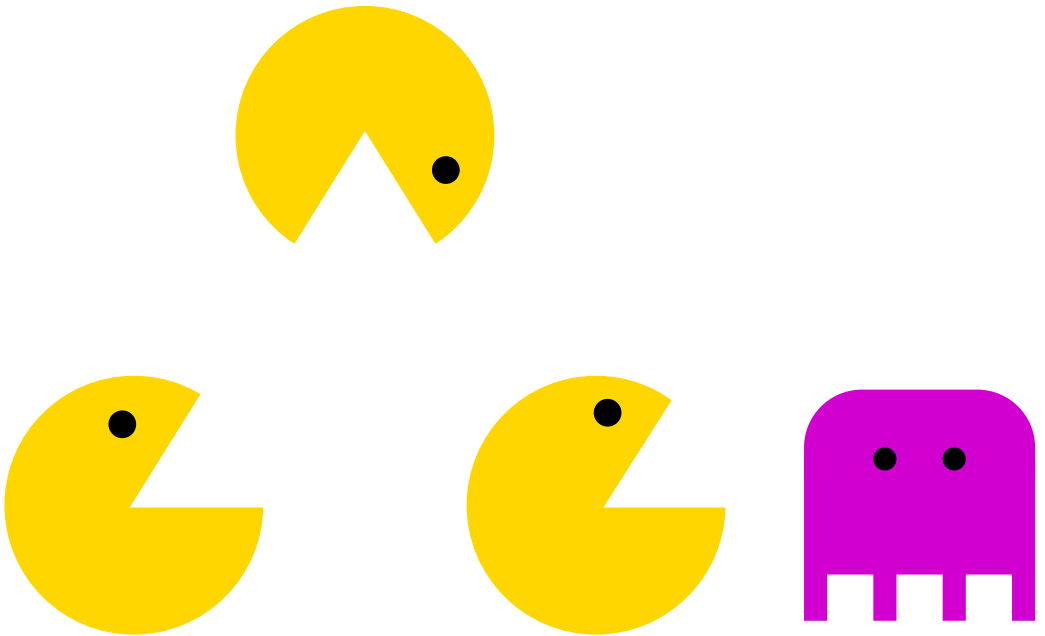
Visual Recognition



Visual Recognition



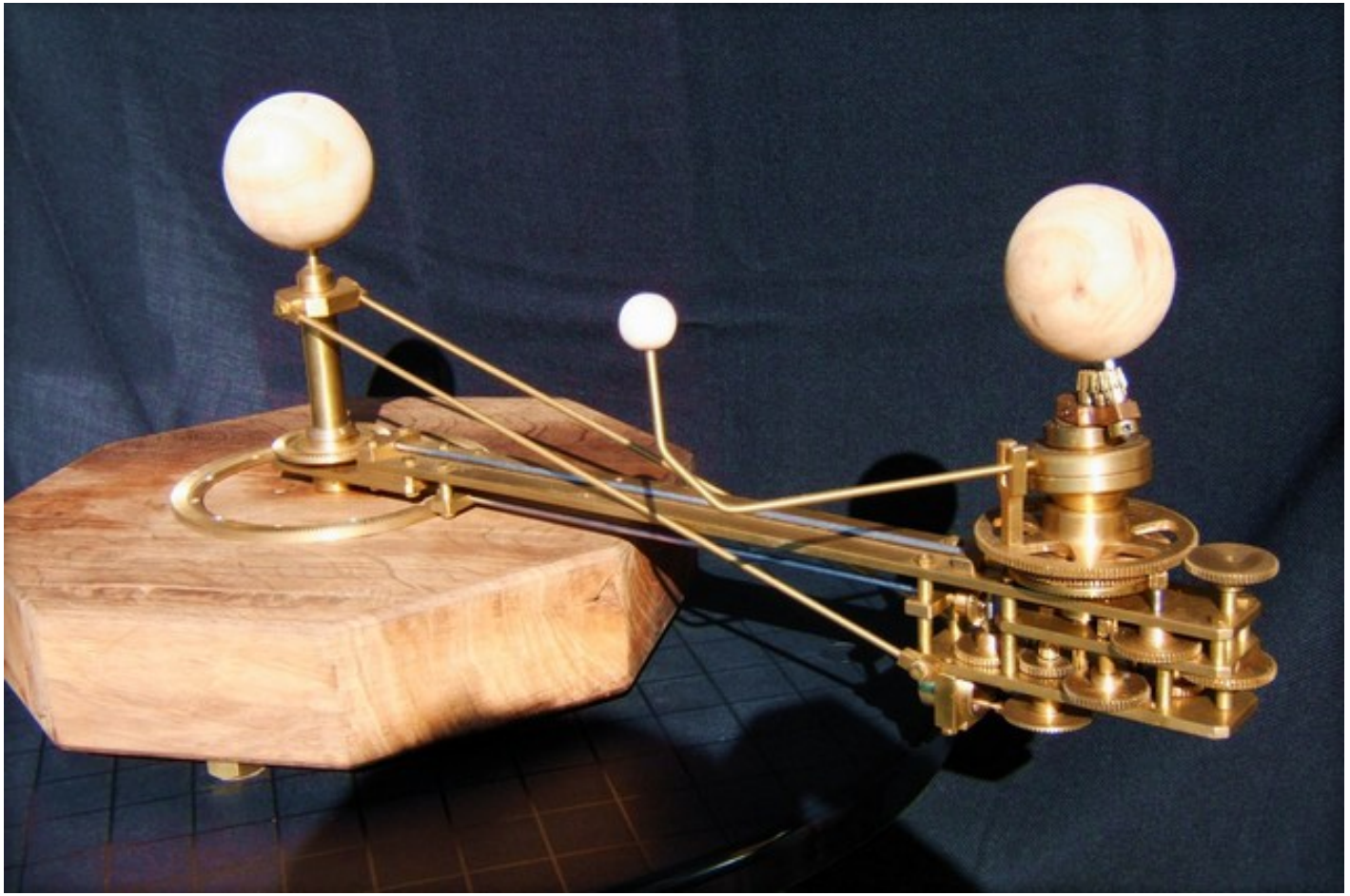
Visual Recognition



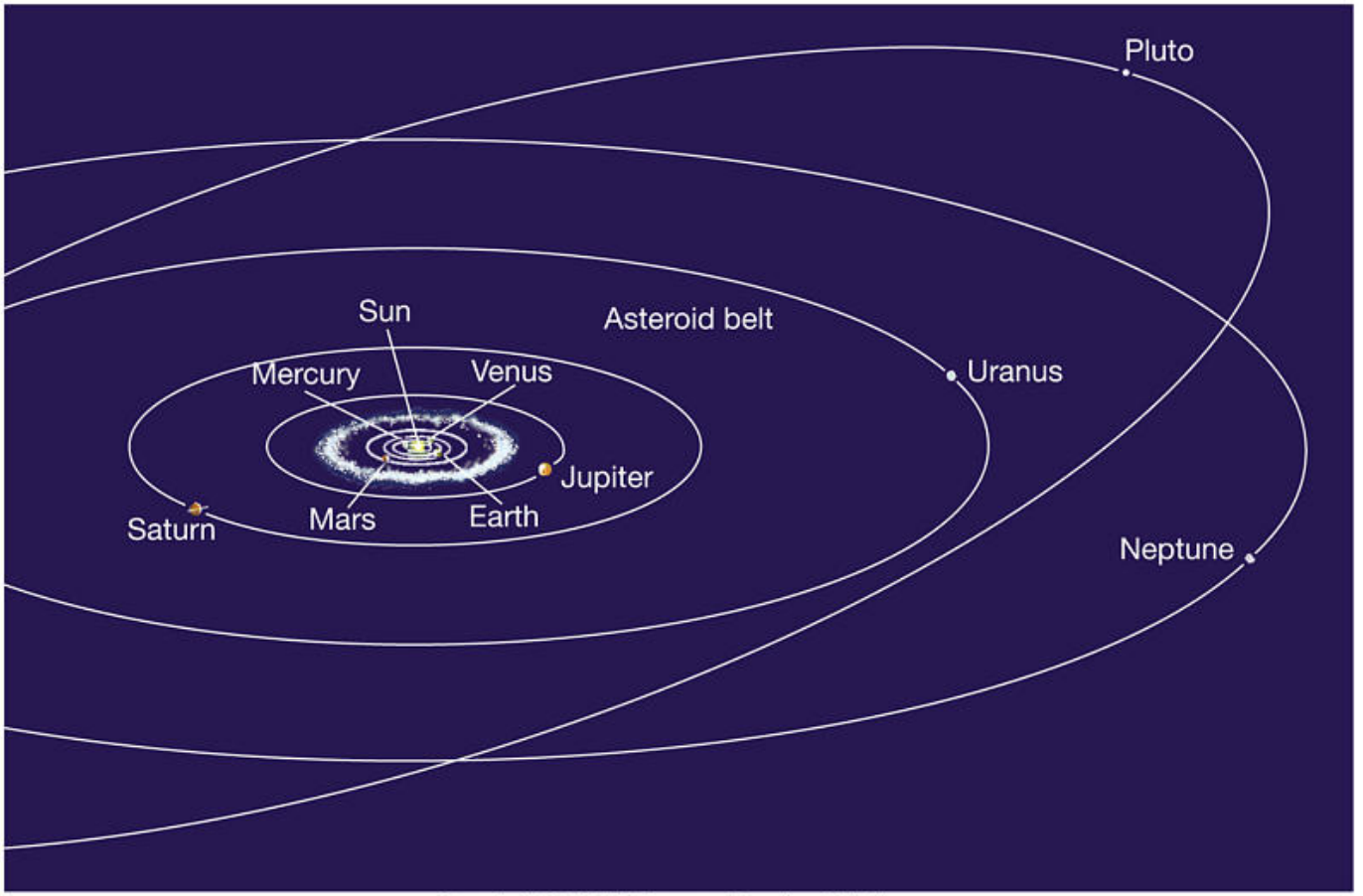
Astronomy


Schema huius præmissæ diuisionis Sphærarum.
EMPIREVM
HABITACVLVM
DEI
ET
OMNIVM
ELECTORVM
COELVM
Primum Mobile
Nonum Coelum
Octauum
Firmamentum
Crystallinum
SATVRNI
IOVIS
MARTIS
SOLIS
VENERIS
MERCVRII
LVNA

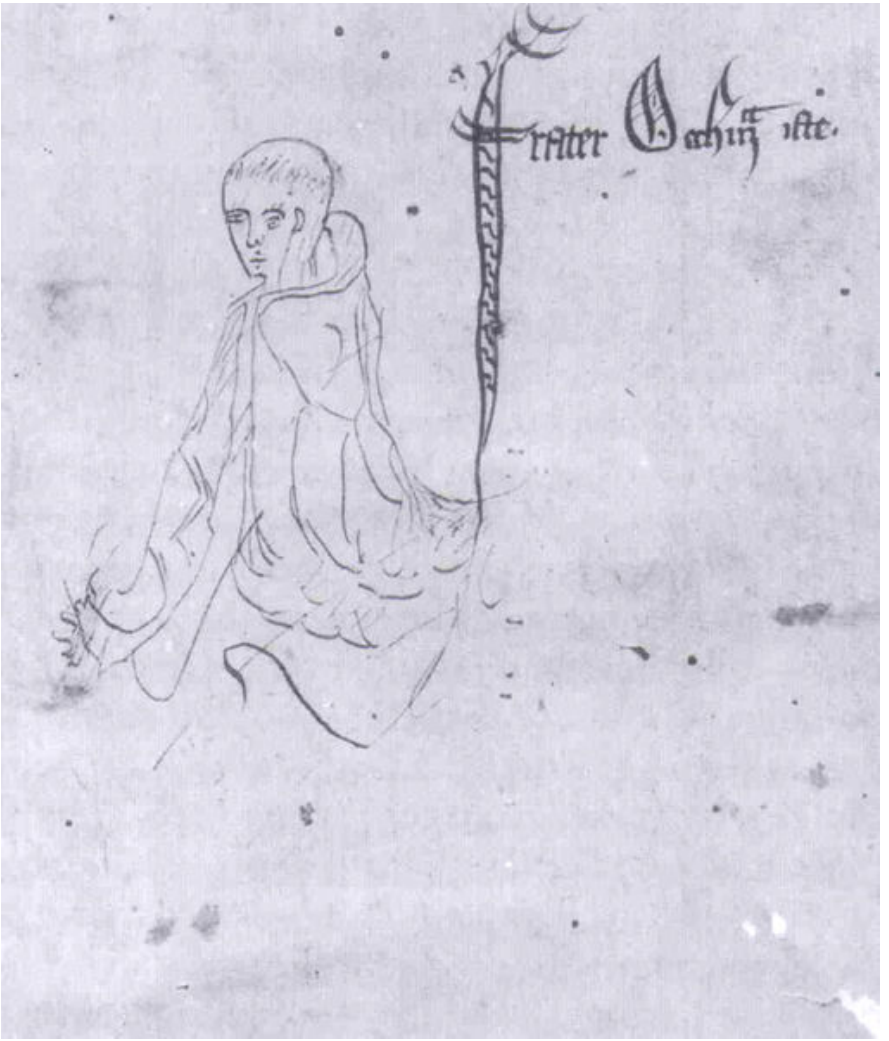
Astronomy



Astronomy



William of Ockham (c. 1288–1348)



Occam's Razor

Occam's Razor

Entities should not be multiplied beyond necessity.

Isaac Newton: "We are to admit no more causes of natural things than such as are both true and sufficient to explain their appearances."

Diagnostic parsimony: Find the fewest possible causes that explain the symptoms.

(**Hickam's dictum:** "Patients can have as many diseases as they damn well please.")

MDL Principle

Minimum Description Length (MDL) Principle (2-part)

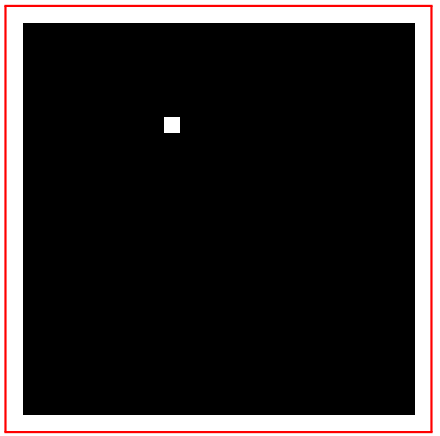
Choose the hypothesis which minimizes the sum of

1. the codelength of the hypothesis, and
2. the codelength of the data with the help of the hypothesis.

How to encode data *with the help of a hypothesis*?

Encoding Data: Rules & Exceptions

Idea 1: Hypothesis = rule; encode exceptions.



Black box of size $25 \times 25 = 625$, white dots at $(x_1, y_1), \ldots, (x_k, y_k)$.

For image of size $n = 625$, there are 2^n different images, and

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

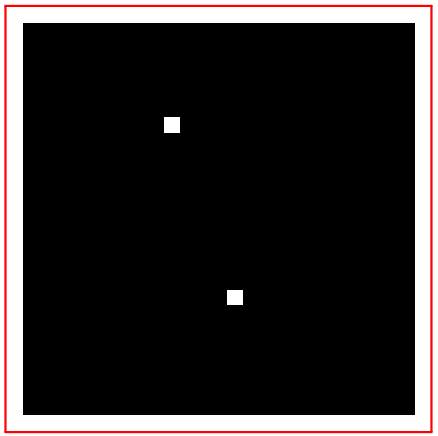
different groups of k exceptions.

$k = 1 : \binom{n}{1} = 625 \ll 2^{625} \approx 1.4 \cdot 10^{188}$.

Codelength $\log_2(n+1) + \log_2 \binom{n}{k} \approx 19$ vs. $\log_2 2^{625} = 625$

Encoding Data: Rules & Exceptions

Idea 1: Hypothesis = rule; encode exceptions.



Black box of size $25 \times 25 = 625$, white dots at $(x_1, y_1), \ldots, (x_k, y_k)$.

For image of size $n = 625$, there are 2^n different images, and

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

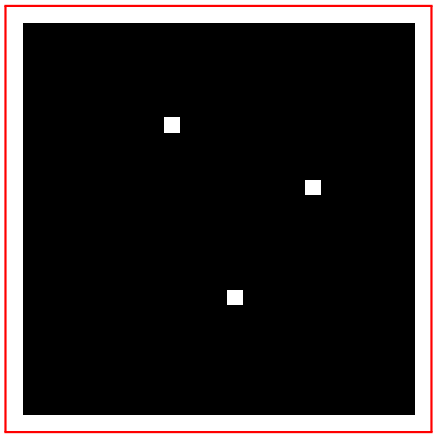
different groups of k exceptions.

$k = 2 : \binom{n}{2} = 195\,000 \ll 2^{625} \approx 1.4 \cdot 10^{188}.$

Codelength $\log_2(n+1) + \log_2 \binom{n}{k} \approx 27$ vs. $\log_2 2^{625} = 625$

Encoding Data: Rules & Exceptions

Idea 1: Hypothesis = rule; encode exceptions.



Black box of size $25 \times 25 = 625$, white dots at $(x_1, y_1), \ldots, (x_k, y_k)$.

For image of size $n = 625$, there are 2^n different images, and

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

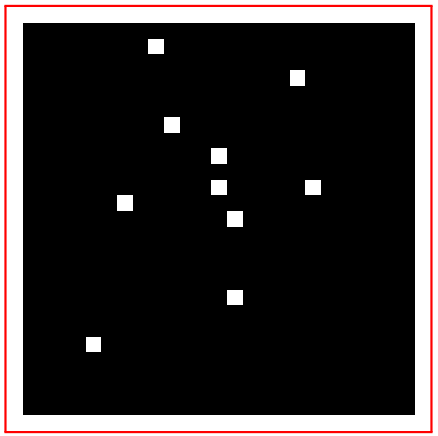
different groups of k exceptions.

$k = 3 : \binom{n}{3} = 40\,495\,000 \ll 2^{625} \approx 1.4 \cdot 10^{188}.$

Codelength $\log_2(n+1) + \log_2 \binom{n}{k} \approx 35$ vs. $\log_2 2^{625} = 625$

Encoding Data: Rules & Exceptions

Idea 1: Hypothesis = rule; encode exceptions.



Black box of size $25 \times 25 = 625$, white dots at $(x_1, y_1), \ldots, (x_k, y_k)$.

For image of size $n = 625$, there are 2^n different images, and

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

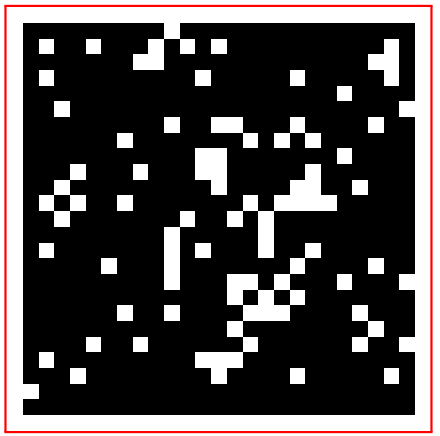
different groups of k exceptions.

$$k = 10 : \binom{n}{10} = 2\,331\,354\,000\,000\,000\,000\,000 \ll 2^{625}.$$

Codelength $\log_2(n+1) + \log_2 \binom{n}{k} \approx 80$ vs. $\log_2 2^{625} = 625$

Encoding Data: Rules & Exceptions

Idea 1: Hypothesis = rule; encode exceptions.



Black box of size $25 \times 25 = 625$, white dots at $(x_1, y_1), \ldots, (x_k, y_k)$.

For image of size $n = 625$, there are 2^n different images, and

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

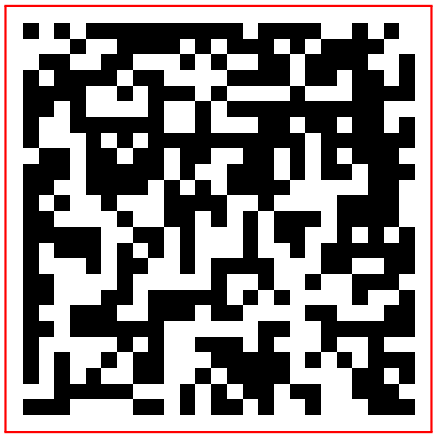
different groups of k exceptions.

$$k = 100 : \binom{n}{100} \approx 9.5 \cdot 10^{117} \ll 2^{625} \approx 1.4 \cdot 10^{188}.$$

Codelength $\log_2(n+1) + \log_2 \binom{n}{k} \approx 401$ vs. $\log_2 2^{625} = 625$

Encoding Data: Rules & Exceptions

Idea 1: Hypothesis = rule; encode exceptions.



Black box of size $25 \times 25 = 625$, white dots at $(x_1, y_1), \ldots, (x_k, y_k)$.

For image of size $n = 625$, there are 2^n different images, and

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

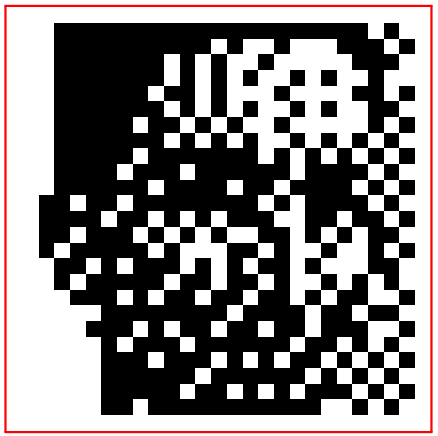
different groups of k exceptions.

$k = 300 : \binom{n}{300} \approx 2.7 \cdot 10^{186} < 2^{625} \approx 1.4 \cdot 10^{188}.$

Codelength $\log_2(n+1) + \log_2 \binom{n}{k} \approx 629$ vs. $\log_2 2^{625} = 625$

Encoding Data: Rules & Exceptions

Idea 1: Hypothesis = rule; encode exceptions.



Black box of size $25 \times 25 = 625$, white dots at $(x_1, y_1), \ldots, (x_k, y_k)$.

For image of size $n = 625$, there are 2^n different images, and

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

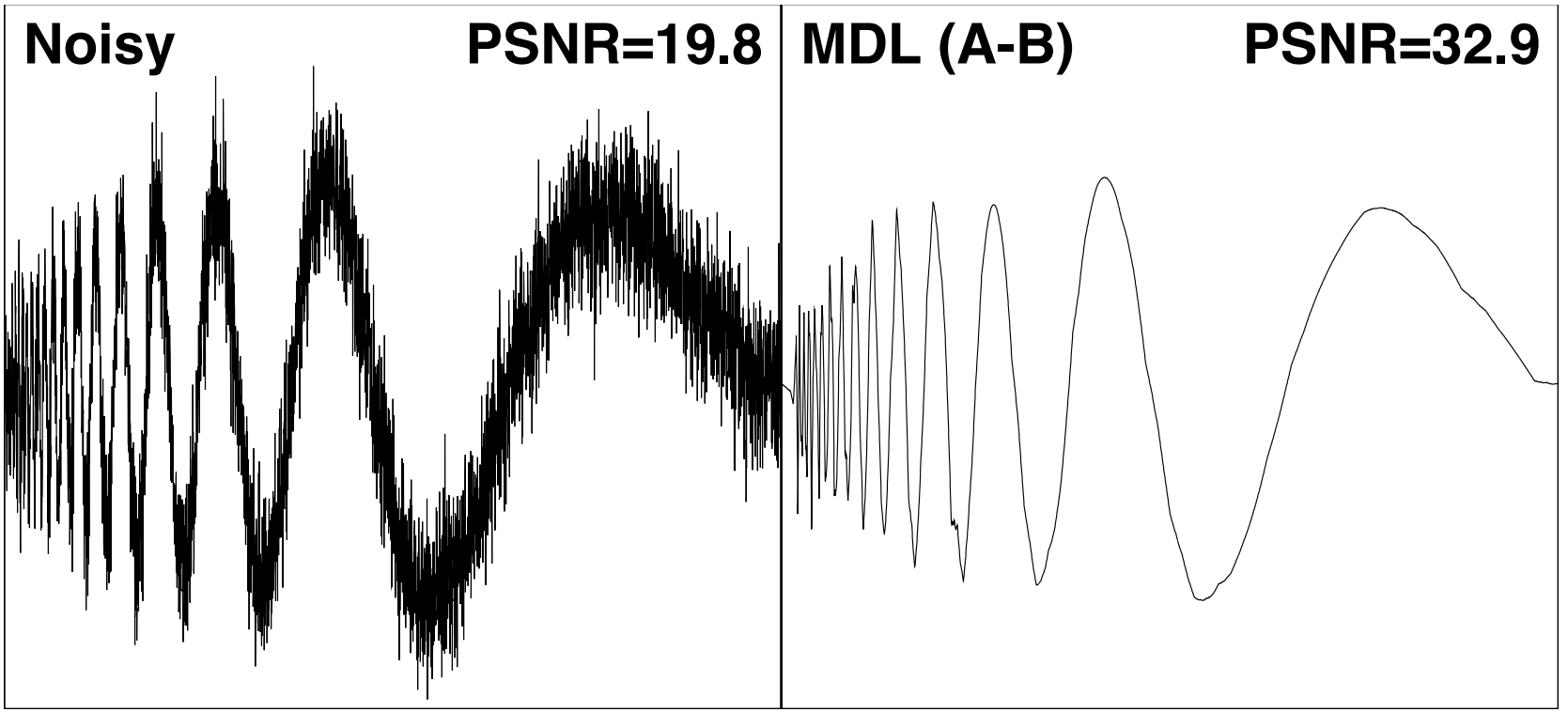
different groups of k exceptions.

$$k = 372 : \binom{n}{372} \approx 5.1 \cdot 10^{181} \ll 2^{625} \approx 1.4 \cdot 10^{188}.$$

Codelength $\log_2(n+1) + \log_2 \binom{n}{k} \approx 613$ vs. $\log_2 2^{625} = 625$

Encoding Data: Probabilistic Models

Idea 2: Hypothesis = probability distribution.



Rissanen & Shannon: $\log_2 \frac{1}{p_{\hat{\theta}}(D)} + \frac{k}{2} \log_2 n.$

Polynomials

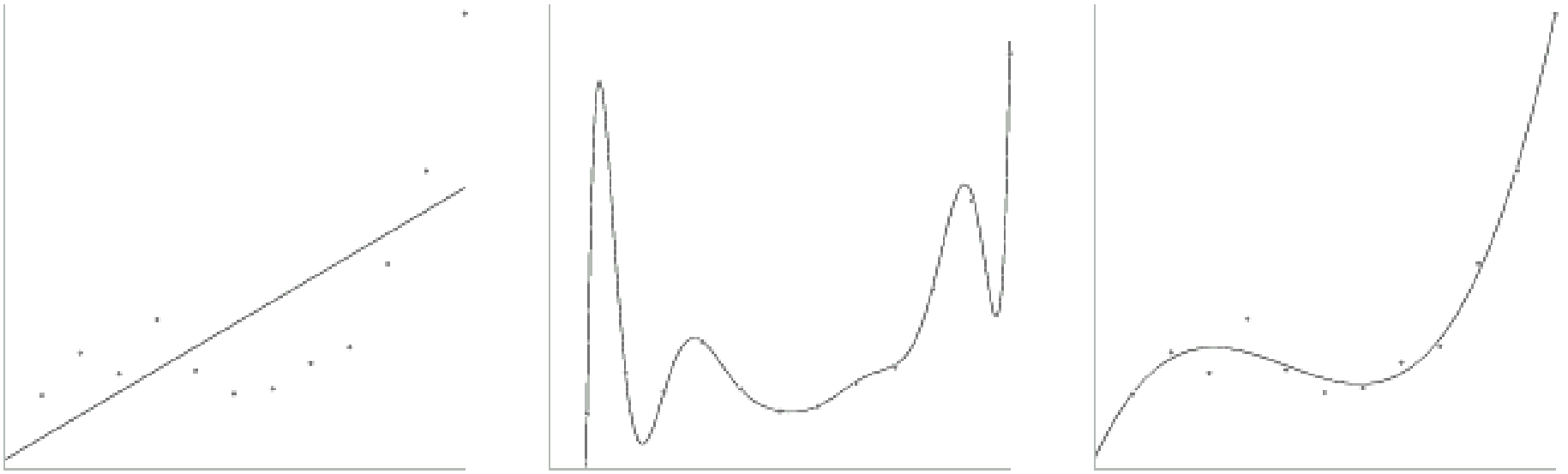


Figure 1: A simple (1.1), complex (1.2) and a trade-off (3rd degree) polynomial.

From P. Grünwald

Old-Style MDL

With the precision $\frac{1}{\sqrt{n}}$ the codelength for data is almost optimal:

$$\min_{\theta^q \in \{\theta^{(1)}, \theta^{(2)}, \ldots\}} \ell_{\theta^q}(D) \approx \min_{\theta \in \Theta} \ell_\theta(D) = \log_2 \frac{1}{p_{\hat{\theta}}(D)} \ .$$

This gives the total codelength formula:

"Steam MDL"

$$\ell_{\theta^q}(D) + \ell(\theta^q) \approx \log_2 \frac{1}{p_{\hat{\theta}}(D)} + \frac{k}{2} \log_2 n \ .$$

Old-Style MDL

The $\frac{k}{2} \log_2 n$ formula is only a rough approximation, and works well only for very large samples.

MDL in the 21st century:

- More advanced codes: mixtures, normalized maximum likelihood, etc.

MDL Model Selection

Perhaps the best known use for MDL is for *model class selection* (which is often called just model selection).

Suppose we have a family of model classes $\mathcal{M}_1, \ldots, \mathcal{M}_m$, each with their own parameter set $\Theta_1, \ldots, \Theta_m$.

We want to pick the one that seems best suited for our data D.

Typically $\mathcal{M}_1 \subset \cdots \subset \mathcal{M}_m$, so $\mathcal{M}_m$ always achieves the best fit to data, but may be *overfitting* (see *Introduction to machine learning*).

Therefore we use MDL and pick $\mathcal{M}_i$ that minimizes the total code length.

MDL Model Selection

Actually here we have a "three-part" code:

1. Encoding of the model class: $\ell(\mathcal{M}_i),\ i \in \{1, \ldots, m\}$.
2. Encoding of the parameter (vector): $\ell_1(\theta),\ \theta \in \Theta_i$.
3. Encoding of the data: $\log_2 \frac{1}{p_\theta(D)},\ D \in \mathcal{D}$.

If we have a finite family of m model classes, we can do Part 1 with $\ell(\mathcal{M}_i) = \log_2 m$ for all i.

This can be generalized to infinite families of model classes, in which case we often to pick p which is "as uniform as possible" over $\mathbb{N}$ and $\ell(\mathcal{M}_i) = \log_2(1/p(i))$.

MDL Model Selection

If we are interested in choosing a model class (and not the parameters), we can improve parts 2 & 3 by combining them into a better universal code than two-part:

1. Encoding of the model class index: $\ell(\mathcal{M}_i),\ i \in \mathbb{N}$.
2. Encoding of the data: $\ell_{\mathcal{M}_i}(D),\ D \in \mathcal{D}$, where $\ell_{\mathcal{M}_i}$ is a universal code-length (e.g., mixture, NML) based on model class $\mathcal{M}_i$.

MDL Model Selection

MDL Explanation of MDL

The success in extracting the structure from data can be measured by the codelength.

In practice, we only find the structure that is "visible" to the used model class(es). For instance, the Bernoulli (coin flipping) model only sees the number of 1s.

MDL & Bayes

The MDL model selection criterion

$$\text{minimize } \ell(\theta) + \ell_\theta(D)$$

can be interpreted (via $p = 2^{-\ell}$) as

$$\text{maximize } p(\theta) \cdot p_\theta(D) \ .$$

In Bayesian probability, this is equivalent to **maximization of posterior probability**:

$$p(\theta \mid D) = \frac{p(\theta)\, p(D \mid \theta)}{p(D)} \ ,$$

where the term $p(D)$ (the marginal probability of D) is constant wrt. θ and doesn't affect model selection.

$\Rightarrow$ **Probabilistic Modelling**

Coming next

- examples of MDL in applications.
- *Kolmogorov complexity*.