Short Course:
Introduction to Information-Theoretic Modeling

Fifth Brazilian Conference on Statistical Modelling in Insurance and Finance

April 11–12, 2011

Teemu Roos

Helsinki Institute for Information Technology HIIT, Finland

" La semplicità è la forma più alta della perfezione. "
" Simplicity is the ultimate sophistication. "
Leonardo da Vinci (1452–1519)

Contents

Prelude: Occam's Razor

According to the so called Electronic Banking Conspiracy[1], a secret organization is conspiring to throw down the global economic system and the whole society. The first step in their plot, replacing precious metal-based currency by virtual money, has already mostly taken place. Next, worldwide chaos will be caused by a complete blackout, erasing all electronic records of bank accounts. Following the usual conspiracy theory pattern, anyone denying or contradicting the existence of such a plot is taken to be clearly a member of the organization, pointing at the involvement of several important government and financial bodies. Similar conspiracy theories abound, involving UFOs, the Apollo moon landings, the 9/11 attacks, the Bible, Elvis, global warming, etc.

[1] See e.g. http://en.wikipedia.org/wiki/List_of_conspiracy_theories.

Such conspiracy theories are very hard to debunk. To see why, consider the following analogue. Figure 1 shows a typical IQ test question. In order to continue the sequence, one starts to consider various hypotheses explaining the appearance of the four initial symbols. The answer is not clear until the solution presents itself, after which it is so simple and obvious that no one can doubt its correctness. Likewise, to continue the integer sequence

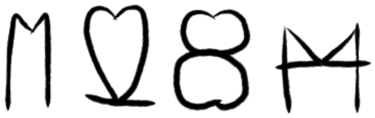
Figure 1: What comes next in the sequence?

Solution: The symbols are integers mirrored along the vertical axis. The next symbol will look something like Ƨ5.

$$1, 2, 4, 8, 16, \ldots \tag{1}$$

one easily summons the exponential sequence $2^0, 2^1, 2^2, 2^3, 2^4, \ldots$ the next integer being $2^5 = 32$. However, how can we be so convinced about the solutions we found? Isn't it always *possible* that the sequence in Fig. 1 consists of the first four symbols of an alphabet used by an alien civilization in outer space? And isn't it possible that the sequence (1) is the 4-Stöhr sequence[2], the following element of which is 31, not 32?

[2] The h-Stöhr sequence is defined as follows: Let $a_1 = 1$, and define for any $n > 1$, a_{n+1} as the least integer greater than a_n that cannot be written as the sum of at most $h \geq 2$ terms among $a_1, \ldots, a_n$; see http://mathworld.wolfram.com/StoehrSequence.html.

So how are the sequence continuation problems at all related to conspiracy theories? Answer: both can be resolved by the application of *Occam's Razor*. The principle of Occam's Razor — also known as *principle of (logical) economy*, or *principle of parsimony* — states that all other things being equal, a simpler explanation is preferred to a more complex one. In the case of IQ test questions, we usually feel more attracted to the simplest solution, and almost invariably, this is also what the designer of the question has had in mind. The alien alphabet hypothesis is an example of an artificial solution that is obviously nonsense exactly for the reason that *it cannot be rejected by any evidence*: one can continue the sequence arbitrarily long, and still it is possible that the aliens are using such symbols, and that the next one will be different. The reason why conspiracy theories cannot be falsified is precisely the same.[3]

In other words, the very fact that neither conspiracy theories nor sequence continuations based on alien alphabets can be *disproved* shows that they are 'non-explanations': they do not rule out any possible events, and hence, they have no explanatory or predictive power.

Of course, none of the above implies that a simple explanation is necessarily true, and a complex one false: if all simple explanations turn out to be wrong, and only complex ones remain, one of them must be true. We should therefore take Occam's Razor as a *heuristic* that can be nor 'right' nor 'wrong', but instead, it can be 'useful' or 'useless' — a matter that can be decided by applying it in various circumstances and seeing whether it leads to good inferences.

[3] According to Karl E. Popper (1902–1994), a theory is *scientific* only if it is *falsifiable* by potentially obtainable empirical evidence. This led him to reject, e.g., psychoanalysis and Marxism on the basis of being non-falsifiable; K. E. Popper, *Logik der Forschung (The Logic of Scientific Discovery)*, Mohr Siebeck, 1934.

In his essay, *Simplicity: Views of Some Nobel Laureates in Economic Science*,[4] Michael McAleer reviews the answers he received from several Nobel laureates to his questions regarding the role of simplicity in their work. One of the respondents, John F. Nash, Jr. (b. 1928, Nobel laureate 1994) wrote:

[4] Arnold Zellner, Hugo A. Keuzenkamp, and Michael McAleer, editors. *Simplicity, Inference and Modelling: Keeping it Sophisticatedly Simple*. Cambridge University Press, Cambridge, UK, 2001

> " Yes, I have definitely had appreciation of principles of simplicity and this is well illustrated in economic theory, in my case [...]
>
> Good examples, in economic or economics-related theory, are my axioms for bargaining or Shapley's axioms giving the 'Shapley value'. [...] "

However, as many of the respondents, Robert M. Solow (b. 1924, Nobel laureate 1987) points out that there are limits to how far we should push the idea of simplicity:

> Obviously, then, I think simplicity is a desirable characteristic of a model. But again a qualification is needed. I am prepared to believe that some things one might like to model are inherently complex, and will not yield to simplicity. In that case it would be foolish to insist on simplicity.

Nash too goes on to add a 'but' when he concludes his response to McAleer:

> " It is certainly true that simplicity has a major function but also it's difficult to think that a simple 'rule of simplicity' can be given so that, by simply using that rule, it would be easy to produce good scientific research! "

The goal of this short course is to explore to which extent such a 'rule of simplicity' can be applied in statistical modeling, and to which extent it is useful. This will take us on a tour in *information theory*, where simplicity is measured in terms of *entropy*, *data compression*, and *computability*. We will discuss the modern version of

Occam's razor, known as the *Minimum Description Length (MDL) Principle*, and relate it to more classical statistical techniques as well as Bayesian methods.

Although many practical examples will demonstrate that information-theoretic principles provide powerful tools for statistical modeling and prediction, we need to stay aware of the inherent complexity of the real world which no principle, no matter how elegant, will take away.

The contents of this short course, as well as these lecture notes, are organized in two main parts. The first part reviews the basic concepts of information theory, focusing in particular on the theory of data compression. The second part will introduce the MDL principle and some of its applications in statistical problems such as model selection and prediction.

Information Theory

The birth of information theory can be dated quite precisely at the publication of Claude E. Shannon's (1916–2001) article *A Mathematical Theory of Communication* in 1948. Related ideas had been around for a while, going back to Ludvig Boltzmann (1844–1906) whose gravestone contains the formula

$$S = k \log W, \tag{2}$$

relating the entropy S of an ideal gas to the number of microstates, W, of the system corresponding to a given macrostate. The constant $k = 1.38062 \times 10^{-23}$ joule/kelvin is related to the thermodynamic significance of the entropy. An important property of Boltzmann's formula is the *logarithmic* relation between the number of states and the entropy.

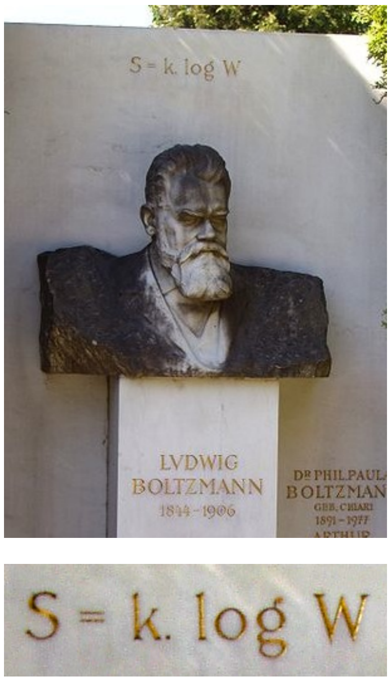

Figure 2: Boltzmann's grave in Zentralfriedhof cemetery in Vienna, and a detail showing the engraved formula. Source: Wikipedia.

Equation (2) is applicable when the microstates of the system can be treated as exchangeable or equally probable. A generalization for the case where each state may have a distinct probability, denoted by p_i, was proposed by J. Willard Gibbs (1839–1903), who defined entropy as

$$S = k \sum_i p_i \log \frac{1}{p_i}, \tag{3}$$

where k is again Boltzmann's constant. Note that when there are W equally probably microstates, Gibbs' formula reduces to Boltzmann's, since then we obtain $p_i = 1/W$ for all i, and by substituting this into Eq. (3) we get

$$k \sum_i \frac{1}{W} \log \frac{1}{1/W} = \frac{kW}{W} \log W = k \log W. \tag{4}$$

Shannon's insight was that the entropy (now defined without the thermodynamic constant),

$$H = \sum_i p_i \log \frac{1}{p_i}, \tag{5}$$

has a fundamental role in the theory of data communication and compression. In order to describe Shannon's results, we need to discuss the basics of coding and introduce a few mathematical notations. For a comprehensive and accessible textbook, see [5].

[5] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, NY, 1991

Basics of coding

A REMARK ON MATHEMATICAL NOTATION: We will use upper-case letters X, Y, Θ, etc., to denote random variables, and lower-case letters, x, y, θ, etc., to denote their values; although we occasionally slip from this convention when (hopefully) there is no risk of confusion. Domains (the sets of possible values) are denoted by calligraphic letters when available, e.g., $\mathcal{X}, \mathcal{Y}, \Theta$.

Letters p, q, etc., are used to denote probability mass functions (pmf) or probability density functions (pdf), with the relevant variable indicated in the subscript, e.g., p_X, whenever it is not clear from the context. Hence, the expression $\Pr[X = x]$, where X is a (discrete) random variable, and x its value, may be written as $p_X(x)$ or even just $p(x)$. Alternatively, we may denote the probabilities by $p_1, p_2, \ldots$ (as above, Eqs. (3)–(5)). The expectation of an expression like $\phi(x)$, involving the random variable X, is denoted by $\mathbb{E}_{X \sim p}[\phi(x)]$, where the subscript indicates the variable over which the expectation is taken and the relevant distribution. Whenever the distribution is clear from the context, it is omitted.

ENCODING DATA CAN BE FORMALIZED as a mapping between source (input) sequences and code (output) sequences. In *data compression* the objective is to map source sequences to code sequences that are as short as possible, and yet enable the reconstruction of the source sequence.

For simplicity, we consider encoding methods where each symbol in the source sequence, $x_1, \ldots, x_n$ is encoded separately. Such codes are called *symbol codes*. As is common, we restrict the code sequences to be binary. Such a code is formally a mapping $C : \mathcal{X} \to \{0,1\}^*$ from the source alphabet $\mathcal{X}$ to the set of finite binary sequences, called *codewords*.

The *extension* of code C is the mapping $C^* : \mathcal{X}^* \to \{0,1\}^*$ obtained by concatenating the codewords $C(x_i)$ for each input symbol

x_i (see Fig. 3):

$$C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n). \tag{6}$$

In Shannon's theory, the performance is characterized by considering a probabilistic source emitting random source symbols, $X_1, X_2, \ldots$. For simplicity, we assume that the symbols are independent and identically distributed, following the probability distribution p. The performance of the code is then measured by in terms of the *expected codeword length*:

$$\mathbb{E}[\ell(C(X))] = \sum_{x \in \mathcal{X}} p(x)\, \ell(C(x)), \tag{7}$$

where $\ell(C(x))$ denotes the length of the codeword.

A symbol code C is said be *decodable* (or *lossless*), if its extension, C^*, is a one-to-one mapping, i.e., iff

$$(x_1, \ldots, x_n) \neq (y_1, \ldots, y_n) \;\Rightarrow\; C^*(x_1, \ldots, x_n) \neq C^*(y_1, \ldots, y_n). \tag{8}$$

For example, consider the following codes:

1. A code with codewords $\{0, 1, 10, 11\}$ is *not* uniquely decodable: 10 could mean either $1, 0$ or 10.

2. A code with codewords $\{00, 01, 10, 11\}$ *is* uniquely decodable: Each pair of bits can be decoded individually.

3. A code with codewords $\{0, 01, 011, 0111\}$ is also uniquely decodable. (Question: What does 0011 mean?)

Code 3 above is decodable but it is somewhat less inconvenient to decode than, for instance, Code 2. Namely, when decoding a Code 3 sequence starting with $0011\ldots$, without yet knowing the continuation, one cannot know whether the last two 1s are related to codeword 011 or 0111. In contrast, Code 2 can be decoded *instantaneously*: every codeword can be identified and decoded as soon as it is received. Such a property is associated with *prefix(-free) codes*. Formally, a code is prefix-free if and only if no codeword is a prefix of another.

The codeword lengths, $l_1, \ldots, l_n$, of a prefix-free code satisfy the important *Kraft inequality*[6]:

$$\sum_{i=1}^{n} 2^{-l_i} \leq 1. \tag{9}$$

For instance, Code 1 above does *not* satisfy the Kraft inequality:

$$2^{-1} + 2^{-1} + 2^{-2} + 2^{-2} = \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} = 1\frac{1}{2} > 1,$$

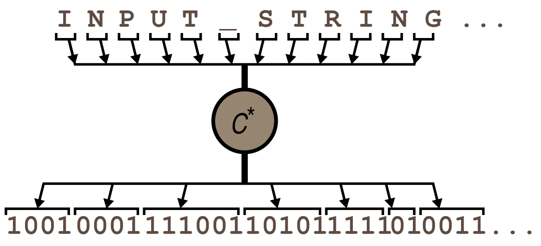

Figure 3: The extension of a symbol code. The codewords of code C are as follows:

I	1001
N	0001
P	111001
U	10101
T	1111
_	01
S	0011
⋮	⋮

[6] Leon G. Kraft. *A Device for Quantizing, Grouping, and Coding Amplitude-Modulated Pulses*. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1949

while Code 3 does:

$$2^{-1}+2^{-2}+2^{-3}+2^{-4}=\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}=\frac{15}{16}\le 1.$$

Conversely, for any set of codeword lengths, $l_1,\ldots,l_n$, that satisfy the Kraft inequality, there exists a prefix code with the given codeword lengths.

The Kraft inequality can be illustrated by means of the following tables. In the leftmost table, the total 'budget' of one unit (the vertical length of the table) is enough to cover the cost of all the codewords, $\sum_i 2^{-l_i} = 1$. In the rightmost table the used codewords are `0`, `1`, `10`, and `11`. The sum of the costs, $1\frac{1}{2}$, exceeds the budget and the Kraft inequality is violated. In the leftmost table the codewords `0`, `10`, `110`, and `111` satisfy the inequality. The code is decodable, and in indeed prefix-free.

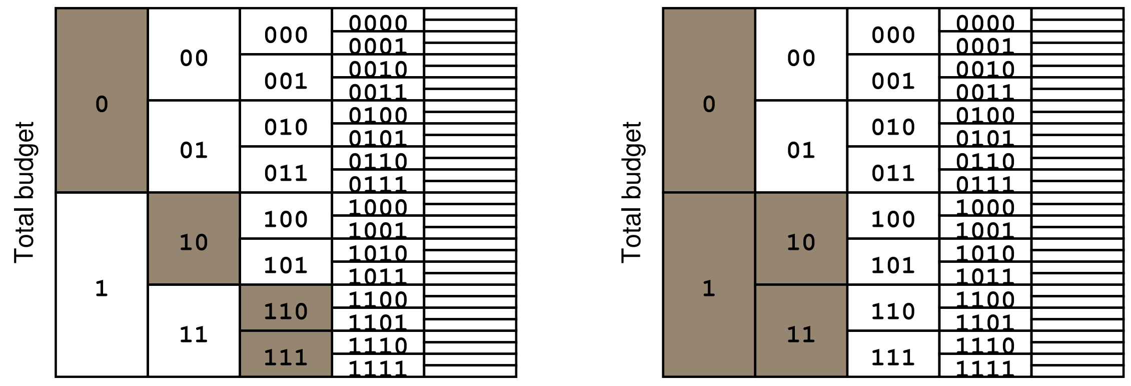

Figure 4: Illustration of the Kraft inequality.

A very useful result known as the *Kraft-McMillan theorem* is that the Kraft inequality applies not only to prefix codes, but to *all* decodable codes.[7] Hence, the restriction to prefix codes has no effect on the achievable compression ratio: any decodable code can be converted to a corresponding prefix code with the same codeword lengths.

The importance of the Kraft-McMillan theorem is two-fold. First, it guarantees that we do not lose anything by restricting the codes to be prefix-free. Secondly, and even more importantly, it enables the unification of (decodable) codes and probability distributions, which provides a way to apply probabilistic concepts such as entropy and mutual information to analyzing the performance of codes.

The unification of codes and probabilities is achieved by defining for a given set of codeword lengths, $\ell(C(x))$, a corresponding (sub-) probability distribution as

$$q(x)=2^{-\ell(C(x))} \quad \Leftrightarrow \quad \ell(C(x))=-\log q(x)=\log\frac{1}{q(x)}, \tag{10}$$

[7] Brockway McMillan. Two inequalities implied by unique decipherability. *IRE Transactions on Information Theory*, 2(4): 115–116, 1956

TABLE 1: IMPORTANT OBSERVATIONS (KRAFT-MCMILLAN THEOREM)

1. All decodable codes satisfy the Kraft inequality:
$$\sum_{i=1}^{n} 2^{-l_i} \leq 1.$$
2. For any decodable code with codeword lengths $l_1, \ldots, l_n$, there exists a *prefix* code with the same codeword lengths.

where C is a decodable code.[8] The fact that the Kraft inequality may be satisfied as a *strict* inequality, i.e., the sum may be less than one, means that the sum of the probabilities $q(x)$ may be less than one, in which case the term *sub*-probability distribution is used. For the sake of simplicity, in what follows, we assume that the sum equals one.

[8] We take all logarithms in base 2 so that, e.g., $\log 2^x = x$.

We can now analyze the performance of a decodable code (assumed to satisfy the Kraft inequality as an equality). The expected codeword length, Eq. (7), can then be written as

$$\mathbb{E}[\ell(C(X))] = \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{q(x)} \tag{11}$$

In order to analyze this quantity, we now introduce some mathematical concepts related to entropy and information.

Entropy and information

GIVEN A DISCRETE RANDOM VARIABLE X with pmf p, we can measure the amount of 'surprise' associated with each outcome $x \in \mathcal{X}$ by the quantity

$$I_p(x) = \log \frac{1}{p(x)}. \tag{12}$$

The intuition is that the less likely the outcome, the more surprised we are to observe it. The entropy of X measures the *expected* amount of 'surprise':

$$H(X) = \mathbb{E}[I_p(X)] = \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)}, \tag{13}$$

which just another way to write the earlier definition, Eq. (5).

For a binary random variable, $X \in \{0, 1\}$, the entropy is maximized — quite intuitively — when the two outcomes are equally probable, see Fig. 5. Naturally, when the outcome is not random at all, $\Pr[X = 1] = 0$ or $\Pr[X = 1] = 1$, there is no surprise, $H(X) = 0$.

As a direct generalization of the entropy of a single variable, we have the *joint entropy* of two random variables, X and Y, given by

$$H(X,Y) = \sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} p(x,y) \log \frac{1}{p(x,y)}, \tag{14}$$

where $p(x,y) = \Pr[X = x, Y = y]$ denotes the joint pmf of the two random variables. Having observed $Y = y$, we can also evaluate the entropy of the conditional distribution of X given the observation $Y = y$ as follows:

$$H(X \mid Y = y) = \sum_{x\in\mathcal{X}} p(x \mid y) log \frac{1}{p(x \mid y)}, \tag{15}$$

which depends on the actual observed value $y \in \mathcal{Y}$. Taking the expectation of this over all the possible values of Y gives the *conditional entropy of X given Y*:

$$H(X \mid Y) = \sum_{y\in\mathcal{Y}} p(y) H(X \mid Y = y). \tag{16}$$

The following table summarizes the intuitive meaning of the above quantities.

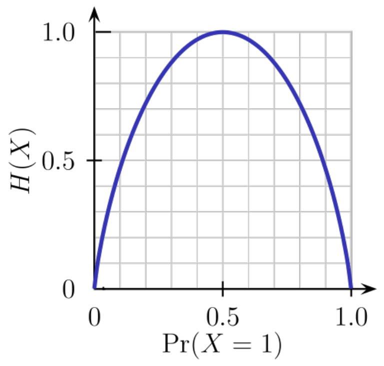

Figure 5: The entropy of a binary random variable as a function of the probability $\Pr[X = 1]$. Source: Wikipedia.

TABLE 2: ENTROPIES

1. The entropy, $H(X)$, measures the uncertainty about the random variable X.

2. The joint entropy, $H(X,Y)$, measures the uncertainty about the pair (X,Y).

3. The entropy of the conditional distribution, $H(X \mid Y = y)$, measures the uncertainty about X when we know that $Y = y$.

4. The conditional entropy, $H(X \mid Y)$, measures the *expected* uncertainty about X when the value of Y is known.

The *chain rule of entropy* relates the different entropies to each other:

$$H(X,Y) = H(Y) + H(X \mid Y). \tag{17}$$

In words, the uncertainty about (X, Y) is equal to the uncertainty related to Y plus the uncertainty related to X when Y is known.

In case the two random variables are *independent*, i.e.,

$$p(x, y) = p(x)p(y) \quad \text{for all } x \in \mathcal{X}, y \in \mathcal{Y}, \tag{18}$$

it is easy to show that $H(X \mid Y) = H(X)$ and $H(Y \mid X) = H(X)$. This implies that the chain rule becomes

$$X \perp\!\!\!\perp Y \quad \Rightarrow \quad H(X, Y) = H(X) + H(Y), \tag{19}$$

where the symbol '$\perp\!\!\!\perp$' denotes independence. Both versions of the chain rule can be easily generalized to more than two random variables.

It is now natural to ask how much (on the average) does knowing one variable reduce the uncertainty about another variable. This is quantified by the *mutual information*:

$$I(X\,;Y) = H(X) - H(X \mid Y). \tag{20}$$

It readily follows from the definition and the basic properties of the entropy that the mutual information is *symmetric*:

$$I(X\,;Y) = I(Y\,;X), \tag{21}$$

which means that the amount of information that X carries about Y is the same as the amount of information that Y carries about X.

Another useful property of mutual information is that it is non-negative

$$I(X\,;Y) \geq 0. \tag{22}$$

Figure 6 illustrates the relationships between the entropies and the mutual information.

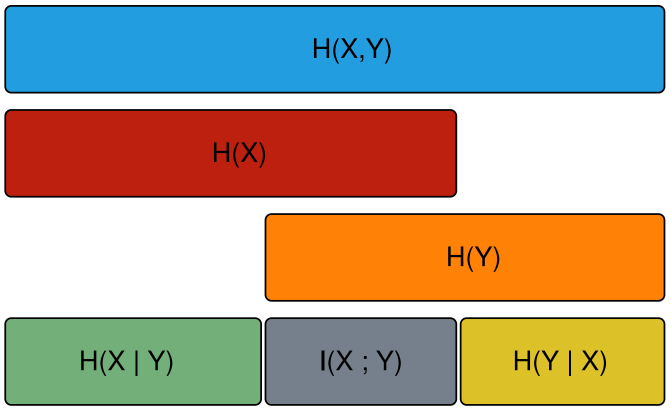

Figure 6: Relationships between the entropies and mutual information:
$$\begin{aligned} H(X,Y) &= H(X) + H(Y \mid X) \\ H(X,Y) &= H(Y) + H(X \mid Y) \\ I(X\,;Y) &= H(X) - H(X \mid Y) \\ I(X\,;Y) &= H(Y) - H(Y \mid X) \end{aligned}$$

THE CRUCIAL OBSERVATION BY SHANNON was to link the above information theoretic concepts to coding and data compression. In order to understand how this can be achieved, we introduce a useful tool called the *Kullback-Leibler (KL) divergence*. The KL-divergence between probability distributions p and q is defined as

$$\mathrm{KL}(p \parallel q) = \mathbb{E}_{X\sim p}\left[\log \frac{p(X)}{q(X)}\right] = \sum_{x\in\mathcal{X}} p(x) \log \frac{p(x)}{q(x)}. \tag{23}$$

An interpretation of the KL-divergence is that it gives the expected difference between the surprise incurred by using distribution q compared to the surprise incurred by using the true distribution p:

$$\begin{aligned}\mathrm{KL}(p \parallel q) &= \mathbb{E}_{X\sim p}[I_q(X) - I_p(X)] \\ &= \mathbb{E}_{X\sim p}\left[\log \frac{1}{q(X)} - \log \frac{1}{p(X)}\right].\end{aligned} \tag{24}$$

Recall now the expected codeword length, Eq. (11), when the codeword lengths are given by $\ell(C(x)) = \log \frac{1}{q(x)}$ and the source symbols are generated by source distribution p. We can rewrite the expectation as

$$\begin{aligned}\mathbb{E}[\ell(C(X))] &= \sum_{x\in\mathcal{X}} p(X) \log \frac{p(x)}{p(x)q(x)} \\ &= \sum_{x\in\mathcal{X}} p(X) \left[\log \frac{1}{p(x)} + \log \frac{p(x)}{q(x)}\right] \\ &= H(X) + \mathrm{KL}(p \parallel q),\end{aligned} \tag{25}$$

where we first multiplied and divided the argument of the logarithm by $p(x)$, then used the fact that $\log ab = \log a + \log b$, and finally, observed that the sum can be rewritten using the entropy and the KL-divergence.

A fundamental result, due to Gibbs, states that the KL-divergence is never negative

$$\mathrm{KL}(p \parallel q) \geq 0, \tag{26}$$

with equality if and only if the two distributions are identical.

The importance of the above is that it suggests a way to choose the codeword lengths in an optimal way. Namely, given a source distribution p, we must try to minimize the term $\mathrm{KL}(p \parallel q)$ since the first term, $H(X)$ depends only on the source. Since we know that $\mathrm{KL}(p \parallel q) \geq 0$, and that it is minimized when $p = q$, it follows that a code with codeword lengths given by

$$\ell(C(x)) = \log \frac{1}{p(x)} \tag{27}$$

is necessarily optimal. Intuitively, this is very appealing: short codewords should be assigned to probable symbols and longer codewords for less probable ones. For instance, a symbol with probability $\frac{1}{2}$

TABLE 3: IMPORTANT OBSERVATIONS (EXPECTED CODEWORD LENGTH VS ENTROPY)

1. Given a prefix code $C(x)$, the expected codeword length under source distribution p is given by

$$\mathbb{E}[\ell(C(X))] = H(X) + \mathrm{KL}(p \parallel q),$$

where q is defined as $q(x) = 2^{-\ell(C(x))}$.

2. Since $\mathrm{KL}(p \parallel q) \geq 0$, this implies

$$\mathbb{E}[\ell(C(X))] \geq H(X),$$

i.e., the expected codeword length is lower bounded by the entropy.

3. The expected code-length is minimized if the codeword lengths are given by

$$\ell(C(x)) = \log \frac{1}{p(x)}.$$

gets a 1-bit codeword (hence, either `0` or `1`), whereas a symbol with probability $\frac{1}{256}$ gets a 8-bit codeword, e.g., `00101100`. Table 3 above summarizes these observations.

However, there are a few caveats in this. First, there is no guarantee that the ideal codeword lengths, $\log \frac{1}{p(x)}$, are integers. (Try to imagine a codeword with length, say, $\frac{3}{4}$ or .123!) Secondly, even if we know the optimal codeword lengths — and even if they happen to be integers — it is not trivial to choose the actual codewords so that the code is prefix-free, or even decodable.

Solving the actual codeword selection problem when the source distribution is given has been studied in great detail. The early codes by Shannon and Robert M. Fano (b. 1917) were nearly optimal and achieved the upper bound

$$\mathbb{E}[\ell(C(X))] \leq H(X) + 1, \tag{28}$$

i.e., expected codeword length within one bit per symbol of the entropy lower bound. Shannon and Fano were, however, unable to figure out how to construct a code that would guarantee that the excess expected code-length was minimized. At an MIT information theory course in the early 1950s, Fano actually posed the problem to a class of graduate students, among whom was David A. Huffman (1925–1999). Huffman was able to solve the problem, and the solution later became known as the Huffman code, still used today in many data compression applications.

Later various enhancements and variations of compression methods have been developed to compress data faster and better. In particular, the integer codeword length requirement can be circumvented by the use of *arithmetic coding*, invented by Jorma Rissanen.[9] Furthermore, special purpose algorithms have been developed for various types of data such as image, sound, video, text, etc. In many of these, the coding is not lossless, i.e., the source signal can be reconstructed from the compressed representation only approximately. For an extensive reference, see [10].

[9] Jorma Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20 (3):198–203, 1976

[10] David Solomon. *Data Compression: The Complete Reference*. Springer, New York, NY, 3rd edition, 2004

Kolmogorov complexity

MEASURING THE COMPLEXITY OF OBJECTS (or their descriptions) by using the entropy and the actual code-length is quite natural. A complex object cannot be compressed as much as a simple one can, etc. However, the fact that both the entropy and the optimal code-length depend on the distribution supposed to govern the source, can be seen as a practical problem. How can we know what the distribution generating, say, a genomic sequence, or an economic time series is like?

Moreover, given a source distribution, the entropy only measures the *expected* code-length. Let us consider for example the uniform distribution over all binary sequence of length 10. The probability of each 20-bit sequence is then $1/2^{20} = 1/1048576 \approx 0.000001$. The entropy is easily seen to be

$$\sum_{x \in \{0,1\}^{20}} \frac{1}{2^{20}} \log \frac{1}{1/2^{20}} = 20. \tag{29}$$

But what if the observed sequence happens to be

```
01010101010101010101,
```

or some other obviously *simple* sequence? If we measure the complexity of the sequence by the entropy, we miss all the features of the particular sequence that was observed that are not determined by the source distribution. This is because the entropy measures the complexity of the sequence only indirectly, through the assumed source distribution. The same holds more or less to the code-length, which in this case would be 20 bits too.

An answer to the above 'lacuna' in Shannon's information theory attracted the attention of the great Russian probabilist Andrey N. Kolmogorov (1903–1987). In his paper *Three Approaches to the*

Quantitative Definition of Information[11], Kolmogorov discussed the following three approaches to the definition of information (or 'complexity'):

11 A. N. Kolmogorov, Three Approaches to the Quantitative Definition of Information, *Problems in Information Transmission*, 1:3–11, 1965.

1. A combinatorial approach based on the logarithm of the number of possible outcomes, where the complexity measure is essentially given by Boltzmann's formula, Eq. (2).
2. The probabilistic approach due to Shannon.
3. An approach based on *computability* as defined earlier by Alan M. Turing (1912–1954).

The third approach was independently discovered almost simultaneously by three researchers: Kolmogorov, Ray Solomonoff (1926–2009), and Gregory J. Chaitin (b. 1947). Due to this, it is sometimes called Solomonoff-Kolmogorov-Chaitin complexity. For a comprehensive textbook, see[12].

12 Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, Berlin, 1993

In order to define Kolmogorov complexity, we introduce the concept of a *Turing machine*. A Turing machine is a formal model that underlies practically all modern computers. It involves a tape that the computer can read and write (a memory), and a set of rules that determines the action to take in a particular state of the computer and when reading a particular symbol on the tape. When inputs are given on the tape in the beginning, and after the Turing machine has been allowed to run until the rules tell it to halt, the output of the machine can be read from the tape.[13] Formally, each Turing machine corresponds to a *partial recursive* (or *computable*) function $U : \{0,1\}^* \to \mathcal{X}^*$ mapping input sequences $\omega \in \{0,1\}^*$ to output sequences $x \in \mathcal{X}^*$. We call the input sequences *programs* since, given a fixed Turing machine, they determine the performed computations. For programs such that the machine never halts, the function is said to be undefined (hence the term *partial* recursive).

13 For some sets of rules and for some inputs, the Turing machine will never halt. The problem of deciding whether a given Turing machine will actually halt for a given input, is surprisingly difficult to solve. It was in fact shown by Turing, in his seminal paper *On Computable Numbers, with an Application to the Entscheidungsproblem*, published in the Proceedings of the London Mathematical Society in 1937, that the problem is *undecidable*, i.e., it cannot be solved even in principle. The consequences of this observation have been subsequently explored to great depth by Chaitin and others.

Despite its simplicity, the Turing machine model is actually very flexible. In particular, it is possible to construct Turing machines that are *universal* in the sense that they can imitate any other Turing machine. Modern computers are examples of practically universal Turing machines. The computable function implemented by a (real) computer depends the particular aspects of the computer as well as the programming language that we use to control it. General purpose programming languages such as *Java*, *Lisp*, *python*, as well as scripting languages used in environments such as *R*, *Matlab*, and *SAS* each define their own computable functions. Each of the above is also universal.

The *Kolmogorov complexity* $K_U(x)$ of (a description of) an object x is defined as the length (in bits) of the shortest program that produces x

Java	R	python
`public static String repeat(String str, int n){`	`paste(rep("01",10),collapse=´´)`	`"01" * 10`
`     StringBuilder ret = new StringBuilder();`		
`     for(int i = 0;i < n;i++) ret.append(str);`		
`     return ret.toString();`		
`}`		
`public static void main(String[] args){`		
`     System.out.println(repeat("01", 10));`		
`}`		

Figure 7: Examples of programs in different languages printing the sequence `01010101010101010101`. Adapted from `http://rosettacode.org/wiki/Repeat_a_string`

when run on a universal Turing machine U:

$$K_U(x) = \min_{\omega \in \{0,1\}^*} \{\ell(\omega) : U(\omega) = x\}. \tag{30}$$

The definition of $K(x)$ depends on the specific Turing machine U. However, as the complexity of x increases, this dependency becomes asymptotically negligible since for any two universal machines, U and V, we have

$$|K_U(x) - K_V(x)| \leq c_{U,V} \quad \text{for all } x, \tag{31}$$

where $c_{U,V}$ is a fixed (but usually unknown) constant. This is called the Invariance Theorem, since it states that the complexity measure is asymptotically invariant under modifications in the underlying universal machine. In practice, we know that, for instance,

$$|K_{\text{Lisp}}(x) - K_{\text{Java}}(x)| \leq c_{\text{Lisp,Java}}, \tag{32}$$

since it is possible to write a Lisp compiler in Java, and vice versa.

Going back to the sequence `01010101010101010101`, we can easily construct a program that, when run on a suitable universal Turing machine, prints out 10 times the bits `01`; see Fig. 7. It is clear that no matter how long the same pattern is continued, we can always produce it using the same technique. This implies that the Kolmogorov complexity of such a sequence is essentially independent of the sequence length.[14] This in stark contrast with the result obtained in Shannon's approach, where (in this particular case) the complexity of the sequence is equal to its length.

[14] In fact we need to also encode the length of the sequence to the Turing machine so that it knows when to stop printing more zeros and ones. This can be achieved with less than $2 \log n$ bits, where n is the length of the sequence. Importantly, this is significantly less than the length of the sequence, n.

Unfortunately, besides the unknown additive constant $c_{U,V}$ in Eq. (31), related to the choice of the universal Turing machine, there is a more serious drawback in Kolmogorov complexity. Namely, it is not *computable*: there is no way to algorithmically calculate the Kolmogorov complexity. In practical applications, it is possible to approximate it by using existing compression techniques, such as the popular Lempel-Ziv algorithm used in tools such as `gzip` and `WinZip`.

Since any of them produces a valid description of the input string, they provide upper bounds on the shortest such description, i.e., the Kolmogorov complexity. This has lead to several interesting results, see e.g. [15].

[15] Stéphane Grumbach and Fariza Tahi. A new challenge for compression algorithms: Genetic sequences. *Journal of Information Processing and Management*, 30(6):875–866, 1994; Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005; and Stephanie Wehner. Analyzing worms and network traffic using compression. *Journal of Computer Security*, 15 (3):303–320, 2007

Exercises

1. Show that the entropy is never negative, $H(X) \geq 0$.
2. Use the fact that $I(X\,;Y) \geq 0$, Eq. (22), to conclude that knowing the value of one random variable, Y, decreases *on the average* the uncertainty about another random variable X:
$$H(X \mid Y) \leq H(X).$$
3. Let the joint distribution of X and Y be given by the following table. The *marginal distribution* of X is then given by
$$p_X(0) = \Pr[X=0, Y=0] + \Pr[X=0, Y=1] = 1/4,$$
$$p_X(1) = \Pr[X=1, Y=0] + \Pr[X=1, Y=1] = 1/4 + 1/2 = 3/4,$$
and similarly, the marginal distribution of Y is given by $p_Y(0) = p_Y(1) = 1/2$. Recall that the conditional distribution of X given Y is determined by
$$p(x) = \frac{p(x,y)}{p(y)}.$$
Now, evaluate the entropy $H(X \mid Y=0)$. Is it true that $H(X \mid Y=0) < H(X)$? Compare this to the statement in Exercise 2.

Ex. 3

		Y	
		0	1
X	0	1/4	0
	1	1/4	1/2

4. Let `DECODABLE` denote the set of uniquely decodable symbol codes, let `KRAFT` denote the set of symbol codes that satisfy Kraft's inequality, let `PREFIX` denote the set of prefix-free symbol codes, and let `ALL` denote the set of all symbol codes. Write the sets in the correct order so that
$$\texttt{SET1} \subset \texttt{SET2} \subset \texttt{SET3} \subset \texttt{SET4}$$
holds. (The symbol '$\subset$' denotes the subset relation, i.e., the left-hand side is a subset of the right-hand side).
For each subset relation, present a code showing that the left-hand side is a *strict* subset of the right-hand side, i.e., that the two sets are not equivalent.
5. Let the source distribution p be given by the table beside. What are the optimal codeword lengths under p. Can you construct the actual codewords so that the code is prefix-free?

Ex. 5

	x				
	A	B	C	D	E
$p(x)$	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$

6. Show that the Kolmogorov complexity of more than *half* of the binary sequences $x \in \{0,1\}^{1000}$ of length 1000 is at least 999 bits. *Hint:* How many programs (binary sequences) there are that are shorter than 1000 bits? Compare this number to the number of all possible strings x of length 1000.

Minimum Description Length Principle

In this second part of the course, we discuss how the above information-theoretic concepts can be used as a basis for statistical inference. In particular, we will describe the modern version of Occam's Razor, namely the so called *Minimum Description Length (MDL) principle*. MDL is a relatively recent approach, compared to classical statistical techniques as well as Bayesian methods. For a recent and comprehensive textbook see [16]. A more concise and perhaps more easily approachable source is Rissanen's classic "little green book." [17]

[16] Peter D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007

[17] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company, New Jersey, 1989

The MDL principle was proposed by Jorma Rissanen (b. 1932), a Finnish information theorist mentioned above in connection to the invention of arithmetic coding.[18] The motivation for MDL derived from obvious difficulties of the frequentist framework to deal with the problem of over-fitting, i.e., choosing overly complex models. Rissanen was inspired by the concept of *universal* description methods in Kolmogorov complexity, as well as the earlier Minimum Message Length (MML) principle.[19]

[18] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5): 465–471, 1978

[19] Chris S. Wallace and David M. Boulton. An information measure for classification. *Computer Journal*, 11(2): 185–194, 1968

The three central concepts in the theory of MDL are *complexity*, *information*, and *noise*. Roughly, their relationship is that the total complexity in an object is the sum of the information and the noise in it:

$$\textit{complexity} \quad \approx \quad \textit{information} \; + \; \textit{noise}. \tag{33}$$

The objective of MDL is to separate the information and the noise that together define the given set of data. In its basic form MDL principle amount to choosing the hypothesis that minimizes the total description length:

$$\min_{h \in \mathcal{H}} \left(\ell(h) + \ell(D\,;h)\right), \tag{34}$$

where $\ell(h)$ denotes the code-length of the hypothesis, and $\ell(D\,;h)$ denotes length of the description of the data given the hypothesis. We elaborate on both of these terms below.

The real advantage of MDL and related information-theoretic approaches is that they provide a principled way to balance the complexity of the hypothesis and the goodness-of-fit. It is clear from the basic formula, Eq. (34), that even if a complex hypothesis achieves a shorter code-length for the data, $\ell(D\,;h_{\text{complex}})$, than a simple hypothesis, the simple hypothesis should be chosen if it achieves a shorter total description length.

Three kinds of MDL

The interesting, and practically important, question is how to define the code-lengths $\ell(h)$ and $\ell(D\,;h)$. We will organize the discussion on this topic in three parts, depending on what kind of hypotheses are being considered.

Case I. Non-stochastic hypotheses: In case the hypothesis is, for instance, a sequence continuation that specifies exactly one particular integer sequence, then the code-length of the data is zero if the observed sequence matches the one specified by the hypothesis, $\ell(D\,;h) = 0$; once the hypothesis is given, the sequence is fully specified. However, if the observed and specified sequences differ, then we consider the code-length to be infinite in accordance with the fact that

$$p(D\,;h) = 0 \quad \Rightarrow \quad \log \frac{1}{p(D\,;h)} = \infty. \tag{35}$$

Given an initial sequence and two alternative hypotheses, each of which specifies one sequence, and both match the initial sequence, the choice based on MDL is made by minimizing the code-length of the hypothesis, $\ell(h)$.

In practice, we often cannot expect that the data either match the hypothesis exactly or not at all. Instead, we can allow some discrepancy and encode the data in terms of the difference between the prediction given by the hypothesis and the observed data. This implies that the code-length of the data given the hypothesis will be non-zero. Assume, for instance, that we are comparing two equally complex hypotheses, $\ell(h_1) = \ell(h_2)$. Assume further that hypothesis h_1 allows only one sequence, the initial symbols of which agree exactly with the observed initial sequence, but that hypothesis h_2, is so flexible that it allows a great deal of variation in the initial sequence — such as the alien alphabet hypothesis. Then the code-length of the data given h_1 is zero but the code-length of the data given h_2 is greater than zero, $\ell(D\,;h_2) > 0$. The MDL criterion then prefers the more specific hypothesis h_1.

Codes based on encoding exceptions are among the most natural ones based on non-stochastic hypotheses. If a binary sequence of length n matches the hypothesized one at k elements but differs at the remaining $n-k$ elements, we can encode the difference by listing the indices of the differing elements. A straightforward code requires $(n-k)\lceil \log n \rceil$ bits[20] ($\lceil \log n \rceil$ bits for each of the $n-k$ exceptions). A more refined method based on observing that the number of ways to choose the $n-k$ exceptions out of the n indices is given by

$$\binom{n}{n-k} = \frac{n!}{k!(n-k)!}, \tag{36}$$

where $n!$ denotes the factorial. This reduces the code-length to

$$\left\lceil \log \binom{n}{k} \right\rceil = \lceil \log n! - \log k! - \log(n-k)! \rceil, \tag{37}$$

[20] The $\lceil \cdot \rceil$ notation refers to the smallest integer greater than or equal to the argument (read: ceiling). Thus, for instance, the number of bits required to express an integer between $0, \dots, 5$ is $\lceil \log 6 \rceil = 3$; the number of alternatives is six, and each can be assigned a unique 3-bit codeword.

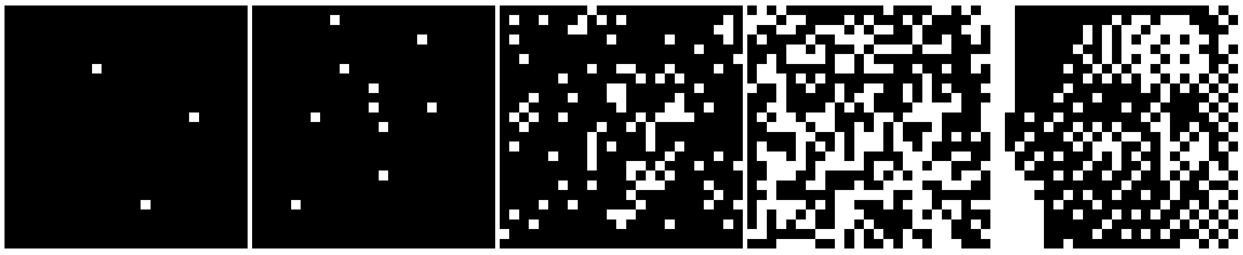

Figure 8: Encoding data under the non-stochastic hypothesis "black square with white dots". Code-length is calculated using Eq. (38), where $n = 25 \times 25 = 625$, and k is the number of white dots.

which is strictly less than $(n-k)\lceil \log n \rceil$.

Figure 8 illustrates the idea of exception-based hypotheses. The hypothesis states that the object to be described is a black square of size 25×25 pixels. The exceptions to the rule are white pixels at different positions of the square, which are encoded by first giving their number, using $\lceil \log(n+1) \rceil$ bits (the number of exceptions can be anything between 0 and n), and then specifying their locations. Using the code of Eq. (37), the total code-length then becomes

$$\ell(D\,;h) = \lceil \log(n+1) \rceil + \left\lceil \binom{n}{k} \right\rceil, \tag{38}$$

where $n = 625$ and k is the number of white pixels. The code-lengths can be compared to the straightforward binary representation where the color of each pixel is given by a single bit (`0` for black, `1` for white, or vice versa), which would always give the code-length 625. Notice that the black-square-with-white-dots hypothesis sometimes gives significantly shorter code-lengths (but not always, see image *d*) — we have in fact created a very crude method for *image compression*; see Fig. 9

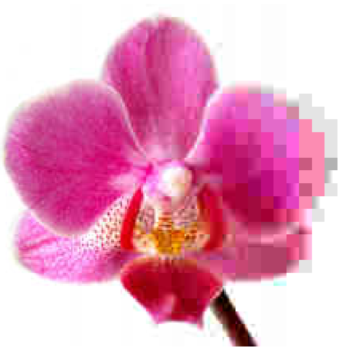

Figure 9: In practice, compressed image formats, such as GIF and PNG exploit the local structure in the pixel colors. Most natural images contain structures such as contiguous regions of the same or similar colors or texture that can be exploited to compress the data significantly more than the our crude code based on only the number of white pixels on a black background. In *lossy* compression, such as the kind used in JPEG format, some loss of information is accepted in return for several orders of magnitude more compact representation. The above image shows the progressive loss of accuracy as the level of compression is increased from left to right. *Source:* Wikipedia (photo by A. Karwath, processing by I. Karonen).

Case II. Stochastic point hypotheses: It is often advantageous to construct codes based on *stochastic* hypotheses, i.e., probability distributions. We first discuss point hypotheses, wherein a single distribution is involved. This is nicely handled by Shannon's framework, and the code-length becomes

$$\ell(D\,;h) = \log \frac{1}{p_h(D)}, \tag{39}$$

where $p_h(D)$ denotes the probability of data D under hypothesis h. The term 'ideal code-length' is often used to refer to the real numbers $\log \frac{1}{p}$ independent of whether they are integers or not. The fact that the actual codewords need never be constructed when applying the MDL principle makes it unnecessary to round the numbers to integers. In the following, we use such ideal code-lengths.

Here too, the code-length of a hypothesis, $\ell(h)$, must be carefully determined. Whenever the hypothesis amounts to a probability distribution, such as, e.g., the Poisson or geometric distribution, that involves a continuous-valued parameter, the parameter must be truncated to finite precision, or otherwise it cannot be encoded with a finite number of bits; see the discussion of two-part codes in the next section.

CASE III. STOCHASTIC COMPOSITE HYPOTHESES: The real challenge for model selection criteria is the composite hypothesis case where each hypothesis corresponds to a *set* of distributions, or a *model class*. Much of the difficulty is related to handling models of varying complexity. For instance, one may need to choose which variables to include in a regression model, or to choose the length of the look-back (lag) in a time series prediction model. Popular solutions to the problem include forward and backward selection procedures, Akaike information criterion (AIC), Bayesian information criterion (BIC), etc.

The inspiration that led Rissanen to formulate the MDL principle was to imitate the construction of a universal computable function (Turing machine) in Kolmogorov complexity. The corresponding MDL concept is a *universal code* that achieves essentially as short a code-length as any of the distributions in the model class. The difference being that the universal code is universal with respect to the distributions in the given model class, whereas the universal computable function is universal with respect to all computable functions. The code-length of the data achieved when using the universal code is called the *stochastic complexity* as its role is similar to that of Kolmogorov complexity. The key observation is that once a universal code is associated to each model class, they can be compared head-to-head in terms of the stochastic complexity of the data. This puts each model class on an equal footing independent of its complexity.

Universal codes

There are three main types of universal codes used to define the stochastic complexity. We briefly describe each one.

1. TWO-PART CODE: Historically, the first type of universal codes is called the *two-part code*, where one first encodes truncated parameter values, and then the data given the truncated parameters. The specifics of the truncation (or *quantization*) procedure may strongly affect the outcome, but under some conditions, the code-length can be asymptotically approximated as

$$\ell_{\text{approx}}(D\,;\,M) = \min_{\theta\in\Theta} \log \frac{1}{p(D\,;\,\theta, M)} + \frac{k}{2}\log n, \qquad (40)$$

where the minimization is over all the parameter values θ in the parameter space Θ corresponding to model M, and k is the number of (free) parameters in the model. It is easy to see that the minimizing parameter values are given by the familiar maximum likelihood parameters, which we denote as $\hat{\theta}(D)$. Hence, the two-part code-length becomes

$$\ell_{\text{approx}}(D\,;\,M) = -\log p(D\,;\,\hat{\theta}(D), M) + \frac{k}{2}\log n, \tag{41}$$

which coincides with the Bayesian information criterion (BIC).

2. Mixture code: The second definition is based on so called mixtures distributions (often called *Bayesian* mixtures) of the form

$$\ell_{\text{mixture}}(D\,;\,M) = -\log\int_{\Theta} p(D\,;\,\theta, M)\, w_M(\theta)\, d\theta, \tag{42}$$

where w_M denotes the parameter prior. Here the prior is used in a technical sense without its Bayesian interpretation.[21]

3. Normalized maximum likelihood code: Finally, the most recent definition of stochastic complexity is based on the Normalized Maximum Likelihood (NML) distribution, originally proposed by Yuri Shtarkov (b. 1935) for data compression in 1987. The NML distribution is defined as

$$p_{\text{nml}}(D\,;\,M) = \frac{p(D\,;\,\hat{\theta}(D), M)}{C_M}, \tag{43}$$

where the normalizing constant C_M is given by

$$C_M = \sum_{D' \in \mathcal{X}^n} p(D'\,;\,\hat{\theta}(D'), M), \tag{44}$$

the sum being over all sequences of the same length, n, as the observed data D.

The stochastic complexity based on the NML distribution is given by

$$\ell_{\text{nml}}(D\,;\,M) = -\log p(D\,;\,\hat{\theta}(D), M) + \log C_M. \tag{45}$$

The term *parametric complexity* is used for $\log C_M$ since it gives the additional code-length incurred because the best parameter value $\hat{\theta}(D)$ is not known in advance.

[21] Quoting Rissanen (*Information Theory and Neural Nets* in Smolensky, Mozer, and Rumelhart (editors): Mathematical Perspectives on Neural Networks, 1996):

> " In the MDL principle for statistical inference there is no need for the awkward Bayesian interpretations of the meaning of the prior probability on the parameters. Rather, we may interpret distributions, such as $[p(D\,;\,M)]$, just as convex linear combinations of the models in the class, whose utility will be assessed on other grounds... "

(paraphrased from Baxter and Oliver, *MDL and MML: Similarities and Differences (Introduction to Minimum Encoding Inference – Part III)*, Technical Report, Monash University, 1994)

We demonstrate each variant of MDL in a simple model selection problem involving stochastic composite hypotheses. Consider two models that we will to illustrate the three kinds of universal codes. Model M_1 involves independent binary-valued variables, X and Y. It corresponds to a model class including distributions of the form

$$p_{M_1}(x,y\,;\theta_x,\theta_y) = \begin{cases} (1-\theta_x)(1-\theta_y), & \text{if } x=0, y=0 \\ (1-\theta_x)\theta_y, & \text{if } x=0, y=1 \\ \theta_x(1-\theta_y), & \text{if } x=1, y=0 \\ \theta_x\theta_y, & \text{if } x=1, y=1, \end{cases} \tag{46}$$

where θ_x and θ_y are two parameters taking values within the range $[0,1]$. The model is essentially a combination of two Bernoulli models with independent parameters.

Model M_2 also involves two binary variables, but this time they are dependent. This corresponds to the model class

$$p_{M_2}(x,y\,;\theta_{00},\theta_{01},\theta_{10},\theta_{11}) = \begin{cases} \theta_{00}, & \text{if } x=0, y=0 \\ \theta_{01}, & \text{if } x=0, y=1 \\ \theta_{10}, & \text{if } x=1, y=0 \\ \theta_{11}, & \text{if } x=1, y=1, \end{cases} \tag{47}$$

where there are now four continuous-valued parameters; although a more parsimonious parameterization exploits the fact that $\theta_{00}+\theta_{01}+\theta_{10}+\theta_{11}=1$, so that the dimensionality (degrees of freedom) of the the model class is actually three. Model M_2 is equivalent to a single multinomial model with $k=4$ possible outcomes, namely the four combinations of the values of x and y.

Extending both models to n independent and identically distributed observations can be achieved by letting

$$p(D\,;\theta,M) = \prod_{i=1}^{n} p(x_i,y_i\,;\theta,M), \tag{48}$$

where $D=((x_1,y_1),\ldots,(x_n,y_n))$ denotes the observation sequence, and θ denotes the set of parameters for the model in question.

For concreteness, we consider the following small data set:

	observation									
	1	2	3	4	5	6	7	8	9	10
x	0	1	1	0	1	0	0	1	0	0
y	0	0	0	1	1	0	1	0	1	1

The code-lengths for each of the three variants are then computed as follows.

1. Two-part code: We approximate the two-part code-length using the approximation of Eq. (41). The required maximum likelihood

parameters are given by the relative frequencies in the data:

$$M_1: \qquad \hat\theta_x = 4/10, \qquad \hat\theta_y = 5/10, \tag{49}$$

$$M_2: \qquad \begin{array}{ll} \hat\theta_{00} = 2/10, & \hat\theta_{01} = 3/10, \\ \hat\theta_{10} = 4/10, & \hat\theta_{11} = 1/10, \end{array} \tag{50}$$

and hence, the code-lengths become

$$\begin{aligned} \ell_{\text{approx}}(D\,;M_1) &= -\log p(D\,;\hat\theta_x,\hat\theta_y,M_1) + \frac{2}{2}\log 10 \\ &= -\log[(1-\hat\theta_x)^6\hat\theta_x^4(1-\hat\theta_y)^5\hat\theta_y^5] + \log 10 \\ &\approx 19.7 + 3.3 = 23.0, \end{aligned} \tag{51}$$

and

$$\begin{aligned} \ell_{\text{approx}}(D\,;M_2) &= -\log p(D\,;\hat\theta_{00},\hat\theta_{01},\hat\theta_{10},\hat\theta_{11},M_2) + \frac{3}{2}\log 10 \\ &= -\log \hat\theta_{00}^2\hat\theta_{01}^3\hat\theta_{10}^4\hat\theta_{11} + \frac{3}{2}\log 10 \\ &\approx 18.5 + 5.0 = 23.4. \end{aligned} \tag{52}$$

By this criterion, we would (just slightly) prefer model M_1, i.e., the model where the two variables are independent of each other. Note that this is due to the greater complexity of the second model (5.0 bits versus 3.3 bits), despite the fact that it leads to a more compact encoding of the data given the model (18.5 bits versus 19.7 bits).

2. Mixture code: The mixture code requires a prior. For model M_1, we adopt a uniform density $\theta \sim \text{Uni}(0,1)$ that assigns the same probability to any equal-length interval within the range $[0,1]$. The same uniform density is used for both of the parameters, θ_x and θ_y, which are considered independent of each other.

   For model M_2, we need to take into account the requirement that the parameters sum to one. In this case, the uniform distribution over all admissible sets of parameter values is equivalent to the Dirichlet prior $\text{Dir}(1,1,1,1)$. The respective mixture code-lengths — well known in Bayesian theory as negative logarithms of so called *Dirichlet-multinomial integrals* — for models M_1 and M_2 are then given by

$$\begin{aligned} \ell_{\text{mixture}}(D\,;M_1) &= -\log\left(\frac{n_{[x=0]}!\,n_{[x=1]}!}{n!}\,\frac{n_{[y=0]}!\,n_{[y=1]}!}{n!}\right) \\ &= -\log\frac{6!4!5!5!}{10!10!} \approx 15.7, \end{aligned} \tag{53}$$

   where $n_{x=0}$ denotes the number of observations with $x = 0$, etc., and $n!$ denotes the factorial of n; and

$$\begin{aligned}\ell_{\text{mixture}}(D\,;\,M_2) &= -\log \frac{n_{[xy=00]}!\,n_{[xy=01]}!\,n_{[xy=10]}!\,n_{[xy=11]}!}{n} \\ &= -\log \frac{2!3!4!1!}{10!} \approx 13.6.\end{aligned} \tag{54}$$

Thus, the mixture code leads to a different decision than the approximate two-part code, i.e., choosing model M_2.

3. Normalized maximum likelihood code: The NML code-length, Eq. (45), consists of two terms, the first of which is the same as the first term in the approximate two-part code, Eqs. (51)–(52). The second term is given by the normalization term that involves a sum over all possible data sequences with length n, the number of which is exponential in n. Fortunately, there exists a remarkably recurrence formula for the multinomial case, which gives the normalization term in linear time[22] (see Exercise 9 below).

   For model M_1, the normalization constant is given by the square of the corresponding constant for a simple Bernoulli model with $n = 10$, the calculation of which is straightforward and omitted, see e.g. [23]:

$$C_{M_1} \approx 4.66^2 = 21.7. \tag{55}$$

   For model M_2, the constant is given by

$$C_{M_2} \approx 38.0. \tag{56}$$

   Thus, the respective NML code-lengths are given by

$$\begin{aligned}\ell_{\text{nml}}(D\,;\,M_1) &= -\log p(D\,;\,\hat{\theta}_x, \hat{\theta}_y, M_1) + \log C_{M_1} \\ &\approx 19.7 + 4.4 = 24.1,\end{aligned} \tag{57}$$

   and

$$\begin{aligned}\ell_{\text{nml}}(D\,;\,M_2) &= -\log p(D\,;\,\hat{\theta}_{00}, \hat{\theta}_{01}, \hat{\theta}_{10}, \hat{\theta}_{11}, M_2) + \log C_{M_2} \\ &\approx 18.5 + 5.2 = 23.7.\end{aligned} \tag{58}$$

   Hence, also the NML code leads to the choice of M_2, although again the margin is slight.

[22] Petri Kontkanen and Petri Myllymäki. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103(6): 227–233, 2007

[23] Peter D. Grünwald. *The Minimum Description Length Principle.* MIT Press, 2007

The above situation is actually typical: the asymptotic two-part formula is known to penalize model complexity more heavily than mixture and NML models. Note that the decision based on the mixture code may have been different, had we chosen different priors for the parameters.

The fact that the different variants sometimes lead to different decisions is somewhat troubling. However, it can be shown that as the size of the data set grows larger, the criteria tend to converge to the same outcome; for details see [24] and references therein. Furthermore, since the mixture version is equivalent to Bayesian model choice using so called Bayes factors, the above implies that model selection by MDL is asymptotically equivalent to Bayesian model selection, at least when the said regularity assumptions hold. In practice, the sample sizes are always finite, and differences remain.

[24] Peter D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007

Exercises

7. Let the model class be given by Bernoulli distributions where each bit in a sequence, $D = x_1, \ldots, x_n$, is independent of the others with probability $\Pr(X_i = 1) = \theta$ for all $1 \le i \le n$, and

$$p(D\,;\theta) = (1-\theta)^{n_{[x=0]}} \theta^{n_{[x=1]}}.$$

The parameter takes values $\theta \in \Theta = \{0.0, 0.5, 1.0\}$, and the parameter prior is given by $w(0.0) = 1/4, w(0.5) = 1/2, w(1.0) = 1/4$.

(a) Evaluate the two-part code-length of sequence $D = 0011$ using the formula

$$\min_{\theta \in \{0.0, 0.5, 1.0\}} \left[\log \frac{1}{p(D\,;\theta)} + \ell(\theta) \right]$$

when the code-length of the parameters is given by $\ell(\theta) = 2^{-w(\theta)}$.

(b) Evaluate the mixture code-length of sequence $D = 0011$ by using the discrete version of Eq. (42):

$$\ell_{\text{mixture}}(D\,; M) = -\log \sum_{\theta \in \Theta} p(D\,;\theta, M)\, w_M(\theta).$$

(c) Evaluate the normalized maximum likelihood code-length of sequence $D = 0011$ using Eqs. (43)–(44).

8. Evaluate the NML normalization term for the Bernoulli model with $\theta \in \Theta = [0,1]$ when $n = 10$, i.e., compute the sum

$$C^{(10)}_{\text{Ber}} = \sum_{D \in \{0,1\}^{10}} p(D\,;\hat{\theta}(D)) = \sum_{D \in \{0,1\}^{10}} (1-\hat{\theta}(D))^{n_{[x=0]}} \hat{\theta}(D)^{n_{[x=1]}},$$

where the maximum likelihood parameter is given by

$$\hat{\theta}(D) = \frac{n_{[x=1]}}{n}.$$

The straightforward algorithm requires the evaluation of a sum with $2^{10} = 1024$ terms. Can you figure out how to manage with only 11 terms by grouping like terms?

9. The linear-time algorithm of Kontkanen and Myllymäki for computing the normalizing constant for NML in multinomial models is based on the recurrence

$$C^{(n)}_k = C^{(n)}_{k-1} + \frac{n}{k-2} C^{(n)}_{k-2},$$

where n is the sample size, and k is the cardinality of the multinomial, i.e., the number of different values the variable can take. The recurrence is initialized using known values of $C^{(n)}_1 = 1$ and $C^{(n)}_k$, the latter of which can be evaluated in linear time using a straightforward formula (see Exercise 8).

Let $n = 10$ and $k = 5$. Evaluate the normalizing constant $C_k^{(n)}$ using the above recurrence and the fact that

$$C_2^{(n)} = C_{\text{Ber}}^{(n)} \approx 4.66.$$

As an intermediate step, you will also confirm the fact $C_{M_2} = C_4^{(n)} \approx 38.0$ as stated in the example on p. 24.

10. Figure 10 shows eight models (polynomials) fitted to the same data.

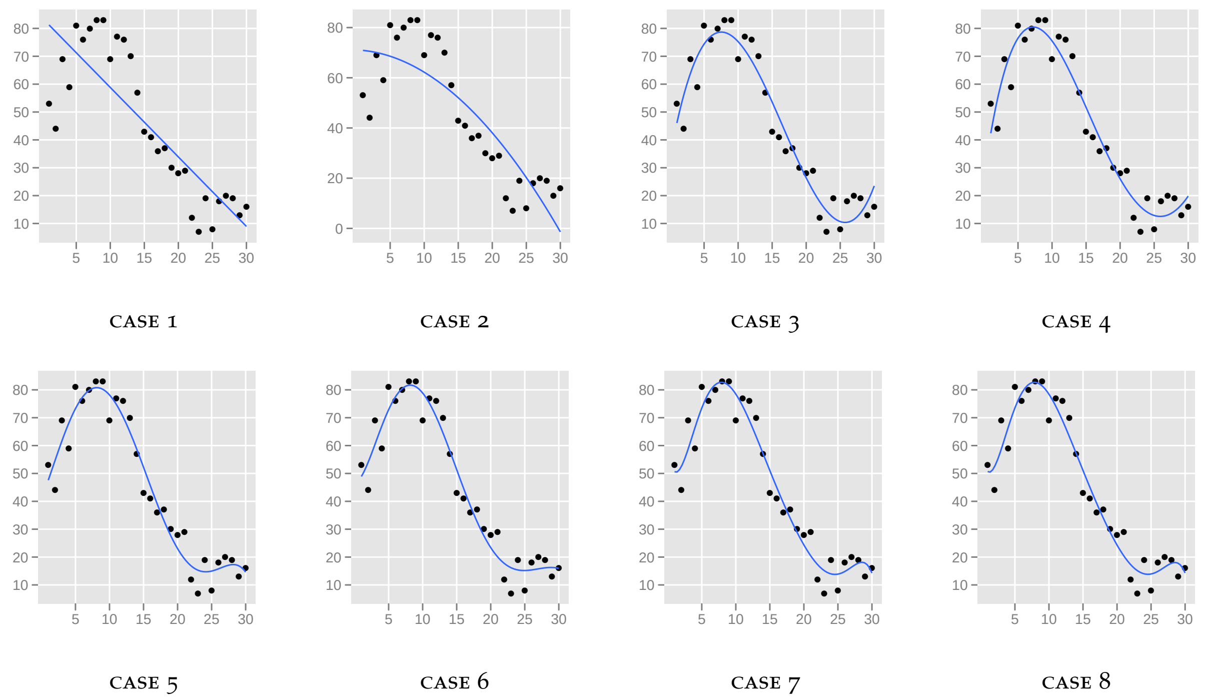

Figure 10: Eight models fitted to the same data; see Exercise 10.

For the eight cases in Fig. 10, the code-lengths $\ell(D\,;\hat{\theta},M)$ and $\frac{k}{2}\log n$ take the following values

	$\ell(D\,;\hat{\theta},M)$	$\frac{k}{2}\log n$
CASE 1	60.99	3.81
CASE 2	57.71	5.71
CASE 3	55.11	7.61
CASE 4	45.45	9.52
CASE 5	44.63	11.42
CASE 6	42.77	13.33
CASE 7	39.44	15.23
CASE 8	37.76	17.13

Which model is chosen by the MDL principle?

Case studies

CASE STUDY I. SIGNAL DENOISING The goal of signal denoising is to remove the *noise* in an observed signal while retaining its intended or *informative* part. Various approaches have been devised. Often, a straightforward smoothing does the job in a satisfactory manner, removing most of the noise, but it has the side effect of blurring sharp edges and peaks in the signal. The blurring phenomenon can be avoided or at least significantly alleviated by methods that adapt to the local smoothness of the signal. A popular approach way to obtain smoothness adaptive methods is to use *wavelets*.[25]

[25] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, CA, 1998; and David L. Donoho and Iain M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995

The idea in wavelet based denoising is to represent the signal as a linear combination of *wavelet basis functions*. A set of Haar basis functions is shown in Eq. (59).

$$\begin{bmatrix} 1/\sqrt{8} & \times & (1, & 1, & 1, & 1, & 1, & 1, & 1, & 1) \\ 1/\sqrt{8} & \times & (1, & 1, & 1, & 1, & -1, & -1, & -1, & -1) \\ 1/2 & \times & (1, & 1, & -1, & -1, & 0, & 0, & 0, & 0) \\ 1/2 & \times & (0, & 0, & 0, & 0, & 1, & 1, & -1, & -1) \\ 1/\sqrt{2} & \times & (1, & -1, & 0, & 0, & 0, & 0, & 0, & 0) \\ 1/\sqrt{2} & \times & (0, & 0, & 1, & -1, & 0, & 0, & 0, & 0) \\ 1/\sqrt{2} & \times & (0, & 0, & 0, & 0, & 1, & -1, & 0, & 0) \\ 1/\sqrt{2} & \times & (0, & 0, & 0, & 0, & 0, & 0, & 1, & -1) \end{bmatrix} \tag{59}$$

The first row represents a constant basis function that can be used to uniformly translate the signal to the desired level. The second row represents the basis function corresponding to the average difference between the first half and the second half of the signal, etc. Note that the basis functions 3–8 are *spatially localized*, i.e., they only affect a part of the signal. Furthermore, the basis functions have variable *frequency*: in the rows nearer the lower end of the matrix, there are sharper changes in the values as one moves from column to column.

In the wavelet regression model, the linear combination of the basis functions, involving certain coefficients, $\beta = (\beta_1, \ldots, \beta_m)$, with which the basis functions are multiplied, determines the mean of the signal

$$y^n = \mathcal{W}^T\beta + \epsilon^n = \sum_{i=1}^{m} w_i\beta_i + \epsilon^n, \tag{60}$$

where $\mathcal{W}$ is a $m \times n$ matrix containing the basis functions, such as (59), w_i is the i'th row of $\mathcal{W}$, and ϵ^n are Gaussian errors. The goal is to choose the coefficients so that $\mathcal{W}^T\beta$ is as close as possible to the informative part of the signal, and the noise is left in the residuals. Typically, one chooses a subset of the coefficients to retain and sets the remaining ones to zero. (In addition, one can shrink the remaining ones as well.)

Choosing the coefficients to retain is a good example of a model selection problem. Hence, MDL provides a princpled solution where the complexity of the model (the number of retained coefficients) is determined automatically. An example of an MDL denoising method is described in [26]. The criterion, obtained by using the NML universal code, for choosing the subset of retained coefficients is given by

[26] Jorma Rissanen. MDL denoising. *IEEE Transactions on Information Theory*, 46(7):2537–2543, 2000; and Teemu Roos, Petri Myllymäki, and Jorma Rissanen. MDL denoising revisited. *IEEE Transactions on Signal Processing*, 101(4):839–849, 2009

$$\frac{n-k}{2}\ln\frac{S(y^n)-S_\gamma(y^n)}{(n-k)^3}+\frac{k}{2}\ln\frac{S_\gamma(y^n)}{k^3}, \qquad (61)$$

where $S(y^n)$ denotes the sum of squares of the maximum likelihood (ML) parameter estimates $\hat{\beta}$; $S_\gamma(y^n)$ denotes the sum of squares of the ML estimates of parameters in subset $\gamma \subseteq \{1,\ldots,m\}$, and k is the number of parameters in γ. Further refinements are obtained by separating the wavelet coefficients into different *levels*, and by applying shrinkage to the retained coefficients. Figure 11 shows the results of the method when applied to a simple simulated example, where the signal consists of a piece-wise constant function.

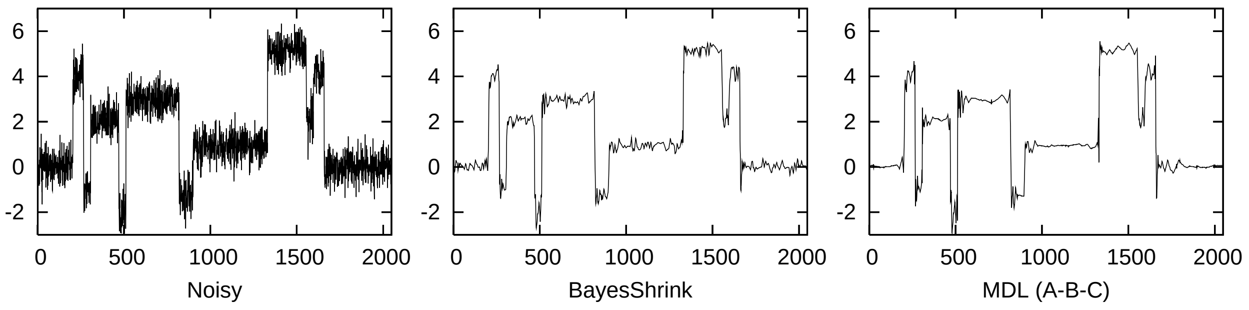

Figure 11: An example of wavelet denoising. The left-most panel shows a simulated noisy signal. The middle panel shows the result obtained by a Bayesian shrinkage method (Chang, Yu, & Vetterli, "Adaptive wavelet thresholding for image denoising and compression", *IEEE Transactions on Image Processing*, 9:1532–1546, 2000). The right-most panel shows the result of MDL denoising. The peak-signal-to-noise ratio (PSNR) is significantly enhanced by both denoising methods, the MDL method being slightly better.

Such denoising methods can be used as a preprocessing step in many types of data analysis, and it can be applied to one-dimensional (as above) as well as two-dimensional data, such as images; see Fig. 12.

Case Study II: Bayesian Network Structure Learning

Bayesian networks are a popular model class for describing dependency structures in multivariate data. To specify a Bayesian network, we first fix a directed acyclic graph (DAG) where *nodes* represent random variables, and directed *edges* represent direct statistical dependency. Sometimes a *causal* interpretation is attached to the dependencies, so that an edge represent a direct *cause–effect* relation. However, it is important to keep in mind that this is not always the case.

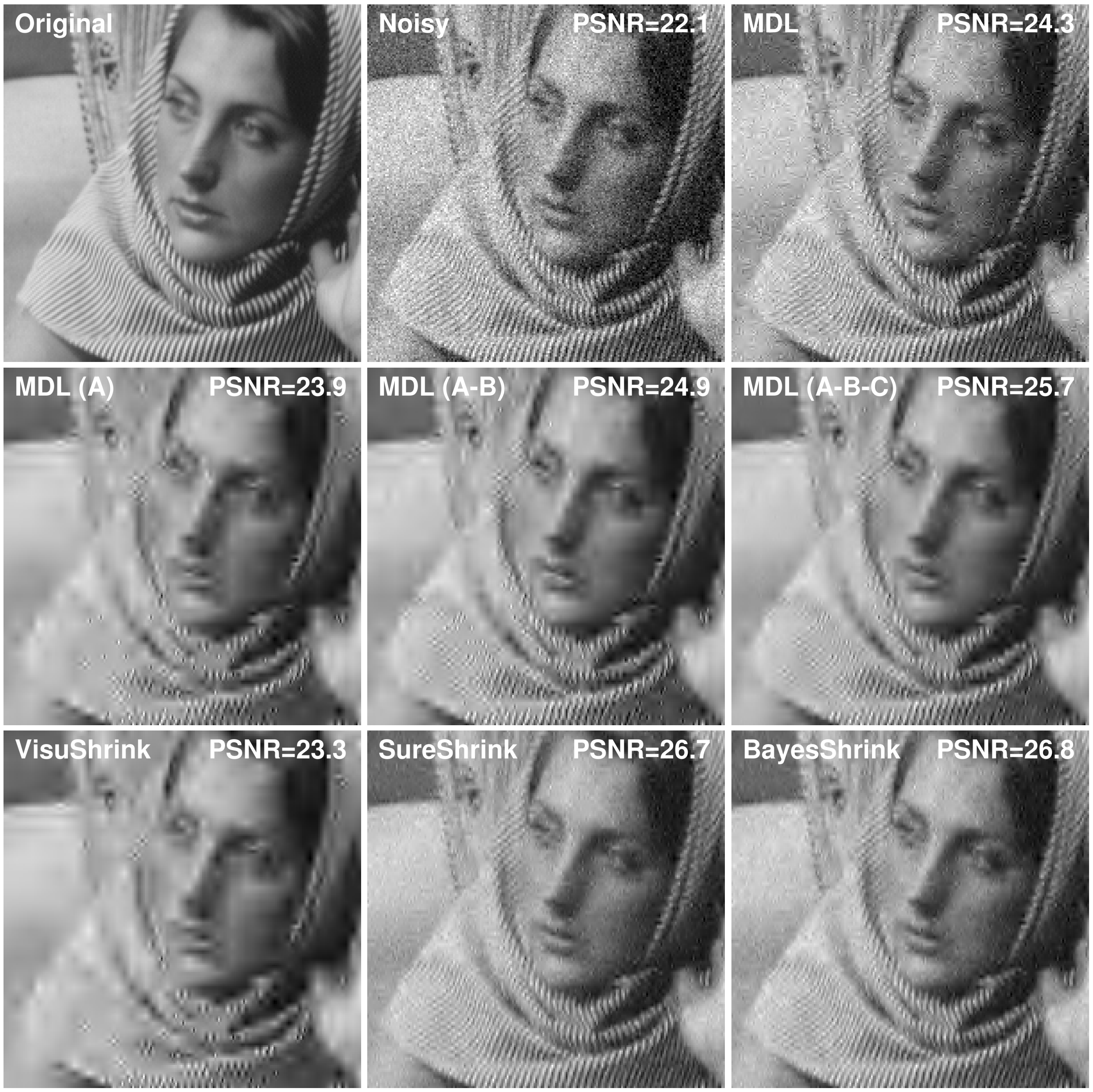

Figure 12: Wavelet denoising of a photographic image. The top-left panel shows the original (all images show a detail of a larger image). The same image with additive Gaussian white noise is shown in the top-middle panel. The results of various denoising methods are shown in the remaining panels. In terms of the peak-signal-to-noise ratio (PSNR) the BayesShrink method (see the caption of Fig. 11) is the best one. For details, see T. Roos, *Statistical and Information-Theoretic Methods for Data Analysis*, PhD dissertation, University of Helsinki, 2007.

To give an example, the DAG in Fig. 13 implies that once the values of *Disease* (the unshaded node in the middle) and *Age* are observed, the *Symptoms* variable is independent of *Climate* and *Occupation*. However, *Age* and *Symptoms* are dependent even given *Disease* and any of the other variables due to the direct edge connecting the former. (*Disclaimer:* We take no clinical responsibility...)

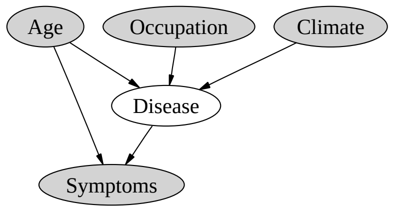

Figure 13: A directed acyclic graph (DAG) reprenting a Bayesian network structure.

In addition to the structure encoded by a DAG, a set of parameters is associated to a Bayesian network. They give the *local conditional distributions* between directly connected variables. If we denote the set of *parents* of a node $X_i \in \{X_1, \ldots, X_m\}$ by $\text{Pa}_i \subseteq \{X_1, \ldots, X_m\} \setminus X_i$, the local conditional distributions are defined as

$$\Pr[X_i = j \mid \text{Pa}_i = k] = \theta_{ijk}, \tag{62}$$

that should be read as "the probability that X_i takes its j'th value given that its parents take their k'th parent configuration (combination of values) equals θ_{ijk}." Given the structure and the parameters, the joint distribution of the variables is uniquely defined by

$$p(x_1, \ldots, x_m) = \prod_{j=1}^{m} \theta_{i x_i \text{pa}_i}, \tag{63}$$

where pa_i denotes the parent configuration corresponding to the observation vector $x_1, \ldots, x_m$.

Given data in the form of a set of observed data vectors $D = x^{(1)}, \ldots, x^{(n)}$, where $x^{(i)} = (x_1^{(i)}, \ldots, x_m^{(i)})$ for each $1 \leq i \leq n$, the structure learning problem is to choose a suitable DAG that describes the (in)dependency structure governing the data. Since a more complex network that includes more edges than a simpler one can represent more distributions, a way of controlling the complexity of the learned network is required — again, a case where MDL is found useful.[27]

[27] Of course, there are various other approaches to complexity penalization, either explicit or implicit in their preference for simpler models. In the context of Bayesian networks, one can, for instance, accept only edges that improve the fit with statistical significance. Bayesian methods are also very popular, see e.g., D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data", *Machine Learning* 20:197–243, 1995.

In order to apply the NML universal code to choosing the network structure, one would need to evaluate the normalizing constant

$$C_n^{\mathcal{B}} = \sum_{D \in \mathcal{D}^n} \prod_{i=1}^{n} p(x^{(i)}\,; \hat{\theta}^{\mathcal{B}}(D), \mathcal{B}), \tag{64}$$

where $\mathcal{B}$ denotes the structure, and $\hat{\theta}^{\mathcal{B}}(D)$ are the corresponding ML parameters for data D. This is, however, computationally infeasible since the number of possible data sets, $D \in \mathcal{D}^n$, is exponential in both the number of variables, m, as well as the sample size, n.

To reduce the computational cost, one can approximate the NML code by the so called *factorized NML* (fNML).[28] In fNML, each variable is encoded separately (but not independently) so that the normalizing term only involves a sum over all the possible observations

[28] Tomi Silander, Teemu Roos, and Petri Myllymäki. Learning locally minimax optimal Bayesian networks. *International Journal of Approximate Reasoning*, 51: 544–557, 2010

of that variable. This can be achieved by first encoding any of the variables that have no parents, say, variable X_1, as if it were the only variable in the model. This step reduces to the multinomial case, for which there is a linear-time algorithm, see p. 24 and Exercise 9. Having encoded all variables that have no parents, we can encode each of the variables whose parent set is included in the already encoded variables, by considering their *conditional* distribution given the parents. Again, the computation reduces to the multinomial case. We then carry on recursively, until all variables have been encoded.

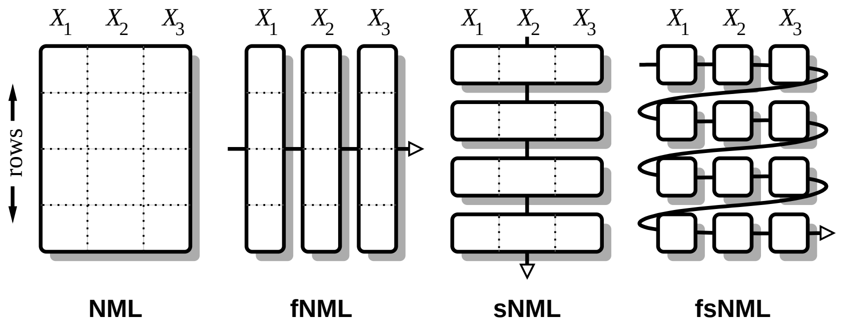

Figure 14: A schematic illustration of different ways to obtain NML-like universal codes for multivariate data. *From left to right: NML*: whole data block is normalized at the same time; *factorized NML*: each variable is encoded separately; *sequential NML*: each observation vector (row) is encoded separately; *factorized-sequential NML*: each component of each observation vector is encoded separately. Note that encoding parts of the data separately does not imply that they would be treated *independently* of each other. *Source:* T. Silander, T. Roos, and P. Myllymäki, "Learning locally minimax optimal Bayesian networks", *International Journal of Approximate Reasoning* 51:544–557, 2010.

The resulting fNML criterion is very efficient and requires no adjustable parameters to be chosen by the user. The lack of adjustable parameters is in general a benefit of the MDL approach compared to, for instance, Bayesian methods where a prior distribution has to be carefully chosen.

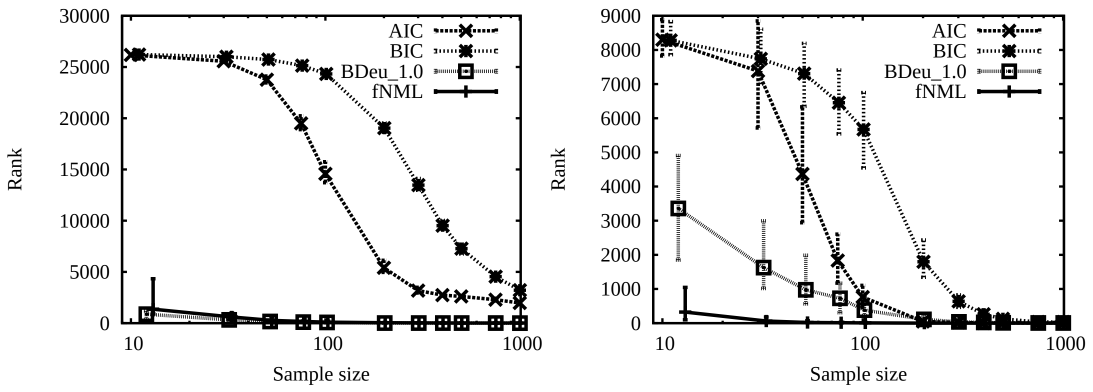

Figure 15: A comparison of four model selection criteria, AIC, BIC, BDe, and fNML. The graphs show the rank of the correct model structure when ordered according to each of the criteria (the lower the rank, the better). The two panels correspond to two different sampling distributions of the parameters of the simulated networks. *Source:* ibid. (see Fig. 14)

The fNML criterion was compared to the state-of-the-art Bayesian criterion, the *Bayesian-Dirichlet criterion* (BDe). For this purpose, 200 different random Bayesian networks with five to seven nodes were generated. Figure 15 shows the *rank* of the true structure when all possible structures were ordered according to four different model selection criteria — for a perfect criterion (an "oracle") that always chooses the true structure, the rank would always be one. It is clearly seen that fNML and BDe criteria outperform the Akaike information criterion (AIC) as well as BIC. Moreover, the two different graphs that correspond to different parameter distributions used in the simulation, show that the BDe criterion is sensitive to the parameter distribution. In the left panel, the parameter distribution matches the $\mathrm{Dir}(1, \ldots, 1)$ distribution assumed in the BDe score which in this case performs optimally. However, in the right panel the distribution is slightly off, which clearly degrades the relative performance of the BDe score. The fNML score is more robust.

Bibliography

Rohan A. Baxter and Jonathan J. Oliver. MDL and MML: Similarities and differences (Introduction to minimum encoding inference — Part III). Technical Report 207, Department of Computer Science, Monash University, Clayton, Vic., 1994.

S. Grace Chang, Bin Yu, and Martin Vetterli. Adaptive wavelet thresholding for image denoising and compression. *IEEE Transactions on Image Processing*, 9(9):1532–1546, 2000.

Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, NY, 1991.

David L. Donoho and Iain M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432): 1200–1224, 1995.

Stéphane Grumbach and Fariza Tahi. A new challenge for compression algorithms: Genetic sequences. *Journal of Information Processing and Management*, 30(6):875–866, 1994.

Peter D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

Petri Kontkanen and Petri Myllymäki. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103 (6):227–233, 2007.

Leon G. Kraft. *A Device for Quantizing, Grouping, and Coding Amplitude-Modulated Pulses*. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1949.

Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, Berlin, 1993.

Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, CA, 1998.

Brockway McMillan. Two inequalities implied by unique decipherability. *IRE Transactions on Information Theory*, 2(4):115–116, 1956.

Karl Popper. *The Logic of Scientific Discovery*. Hutchinson & Co., London, UK, 1st English edition, 1959.

Jorma Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5): 465–471, 1978.

Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company, New Jersey, 1989.

Jorma Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, 1996a.

Jorma Rissanen. Information theory and neural nets. In P. Smolensky, M. C. Mozer, and D. E. Rumelhart, editors, *Mathematical Perspectives on Neural Networks*. Lawrence Erlbaum Associates, 1996b.

Jorma Rissanen. MDL denoising. *IEEE Transactions on Information Theory*, 46 (7):2537–2543, 2000.

Teemu Roos. *Statistical and Information-Theoretic Methods for Data Analysis*. PhD thesis, University of Helsinki, 2007.

Teemu Roos, Petri Myllymäki, and Jorma Rissanen. MDL denoising revisited. *IEEE Transactions on Signal Processing*, 101(4):839–849, 2009.

Yuri M. Shtarkov. Universal sequential coding of single messages. *Problems of Information Transmission*, 23(3):175–186, 1987.

Tomi Silander, Teemu Roos, and Petri Myllymäki. Learning locally minimax optimal Bayesian networks. *International Journal of Approximate Reasoning*, 51:544–557, 2010.

David Solomon. *Data Compression: The Complete Reference*. Springer, New York, NY, 3rd edition, 2004.

Alan M. Turing. On computable numbers, with an application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society*, 42: 230–265, 1937.

Chris S. Wallace and David M. Boulton. An information measure for classification. *Computer Journal*, 11(2):185–194, 1968.

Stephanie Wehner. Analyzing worms and network traffic using compression. *Journal of Computer Security*, 15(3):303–320, 2007.

Arnold Zellner, Hugo A. Keuzenkamp, and Michael McAleer, editors. *Simplicity, Inference and Modelling: Keeping it Sophisticatedly Simple*. Cambridge University Press, Cambridge, UK, 2001.