Part C

Descriptive complexity

Descriptive complexity

How difficult is it to describe this sequence?

01010101010101010101010101010101010101010101010101

Descriptive complexity

How difficult is it to describe this sequence?

01010101010101010101010101010101010101010101010101

25 repetitions of "$\mathbf{01}$", simple

10110101000001001111001100110011111110011101111001

Descriptive complexity

How difficult is it to describe this sequence?

01010101010101010101010101010101010101010101010101

25 repetitions of "$\mathbf{01}$", simple

10110101000001001111001100110011111110011101111001

The first 50 fractional bits of $1/\sqrt{2}$, simple

00110111001001000100111111000010110010001011001100

Descriptive complexity

How difficult is it to describe this sequence?

01010101010101010101010101010101010101010101010101
25 repetitions of "01", simple

10110101000001001111001100110011111110011101111001
The first 50 fractional bits of $1/\sqrt{2}$, simple

00110111001001000100111111000010110010001011001100
50 coin tosses, complex

Descriptive complexity

How difficult is it to describe this sequence?

01010101010101010101010101010101010101010101010101
25 repetitions of "01", simple

10110101000001001111001100110011111110011101111001
The first 50 fractional bits of $1/\sqrt{2}$, simple

00110111001001000100111111000010110010001011001100
50 coin tosses, complex

Simple sequences are "easy" to describe, complex ones must be described symbol by symbol.

Shannon complexity

Can the (Shannon) entropy be considered as a measure of complexity?

Yes, but the entropy depends on the probability of a sequence given an underlying source or stochastic data generating process.

Assuming that a source assigns probabilities $\Pr\{X = x\}$ the entropy of the source is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} \Pr\{X = x\} \log_2 \Pr\{X = x\}.$$

This is the expected number of bits needed to represent X.

Shannon complexity

For a sequence x a corresponding notion is the ideal code wordlength given as

$$I(x) = -\log_2 \Pr\{X = x\}.$$

This can be interpreted as the most favorable representation length.

A disadvantage of Shannon's measures seems to be the fact that the complexity of a sequence depends on the probability of the sequence and not on the sequence itself.

Shannon complexity

Example 5: [of the 'unreasonable' interpretation]

Let $\mathcal{X} = \{01101010000010011110, 00110111001001000100\}$ and let the source select between the two sequences with equal probability $(\frac{1}{2}, \frac{1}{2})$.

The entropy of the source is 1 bit per sequence (of 20 symbols)! However, the two strings each appear much more complex than 1 bit!!

The complexity is hidden in the source description, namely in $\mathcal{X}$, which is already known by the receiver. We shall see that universal data compression gives a more fundamental answer to this problem.

Universal data compression

Is it also possible to find a more meaningful measure using Shannon's information measure?

Because we do not know the model and its parameter values, we must consider data compression for parametrized classes of sources.

Universal data compression

Example 6:

Parametrized binary source (I.I.D. source class)

Alphabet:	$\mathcal{X} = \{0, 1\}$;
Sequence:	$x^N = x_1 \dots x_N$;
	(N is the block length)
Probabilities:	$\Pr\{X_i = 1\} = 1 - \Pr\{X_i = 0\} = \theta$.
	$0 \le \theta \le 1$.

Code:	$C : \mathcal{X}^N \to \{0, 1\}^*$
Code word:	$c(x^N) = c_1 \dots c_j \in C$
Length:	$l_C(x^N) = l(c_1 \dots c_j) = j$

Universal data compression

Ideal code wordlength

The best possible code wordlengths come from Huffman's algorithm, but these are hard to compute.

The task: minimize over the choice of lengths $l_C(x^N)$

$$\sum_{x^N \in \mathcal{X}^N} p(x^N) l_C(x^N)$$

where the lengths must satisfy Kraft's inequality

$$\sum_{x^N \in \mathcal{X}^N} 2^{-l_C(x^N)} \le 1$$

Universal data compression

Ideal code wordlength

Ignoring the requirement that code wordlengths are integer, we find that the optimal code wordlengths are

$$l_C(x^N) = -\log_2 p(x^N)$$

The upward rounded version of these lengths still satisfy Kraft's inequality and the resulting code achieves Shannon's upper bound.

We write $l_C^*(x^N)$ for these ideal code wordlengths.

$$\begin{aligned} l_C^*(x^N) &= \left\lceil -\log_2 p(x^N) \right\rceil \\ &< -\log_2 p(x^N) + 1 \end{aligned}$$

Universal data compression

Remember $n(a|x^N)$ is the number of times the symbol a occurs in x^N.

Sequence probability: $p(x^N) = (1-\theta)^{n(0|x^N)}\theta^{n(1|x^N)}$

Expected code word length: $\bar{l_C} = \sum_{x^N \in \mathcal{X}^N} p(x^N) l_C(x^N)$

(Expected) code rate: $R_N = \frac{\bar{l_C}}{N}$

(Expected) code redundancy: $r_N = R_N - h(\theta)$

Universal data compression

First assume that we know that $\theta = \theta_1 = 0.2$ or $\theta = \theta_2 = 0.9$ but we don't know which θ generated x^N.

Universal data compression

First assume that we know that $\theta = \theta_1 = 0.2$ or $\theta = \theta_2 = 0.9$ but we don't know which θ generated x^N.

We design a code C_1 assuming that $\theta = \theta_1$.

Universal data compression

First assume that we know that $\theta = \theta_1 = 0.2$ or $\theta = \theta_2 = 0.9$ but we don't know which θ generated x^N.

We design a code C_1 assuming that $\theta = \theta_1$.

And a code C_2 assuming $\theta = \theta_2$.

Universal data compression

First assume that we know that $\theta = \theta_1 = 0.2$ or $\theta = \theta_2 = 0.9$ but we don't know which θ generated x^N.

We design a code C_1 assuming that $\theta = \theta_1$.

And a code C_2 assuming $\theta = \theta_2$.

We also create the code C_{12} which uses the smallest code word from C_1 and C_2 with a '0' or '1' prepended to indicate from which code the word comes.

Universal data compression

First assume that we know that $\theta = \theta_1 = 0.2$ or $\theta = \theta_2 = 0.9$ but we don't know which θ generated x^N.

We design a code C_1 assuming that $\theta = \theta_1$.

And a code C_2 assuming $\theta = \theta_2$.

We also create the code C_{12} which uses the smallest code word from C_1 and C_2 with a '0' or '1' prepended to indicate from which code the word comes.

In all cases the code words are created using the ideal code wordlengths $l^*_C(x^N)$.

Universal data compression

First assume that we know that $\theta = \theta_1 = 0.2$ or $\theta = \theta_2 = 0.9$ but we don't know which θ generated x^N.

We design a code C_1 assuming that $\theta = \theta_1$.

And a code C_2 assuming $\theta = \theta_2$.

We also create the code C_{12} which uses the smallest code word from C_1 and C_2 with a '0' or '1' prepended to indicate from which code the word comes.

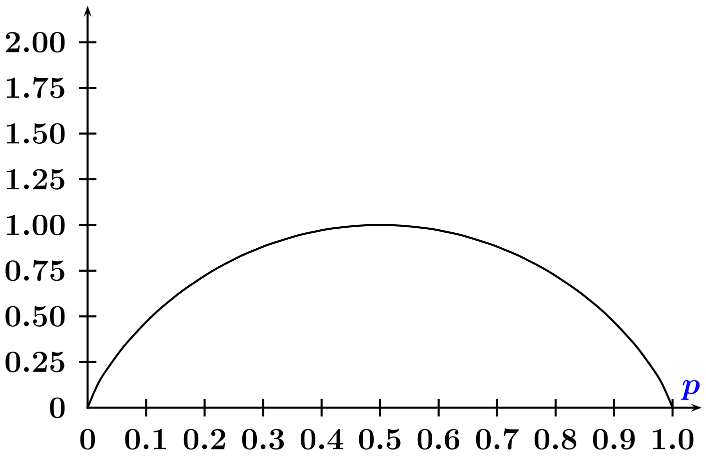
In all cases the code words are created using the ideal code wordlengths $l^*_C(x^N)$.

The code C_{mix} is make using the mixed (weighted) probabilities

$$p_{\text{mix}}(x^N) = \frac{p(x^N|\theta_1) + p(x^N|\theta_2)}{2}$$

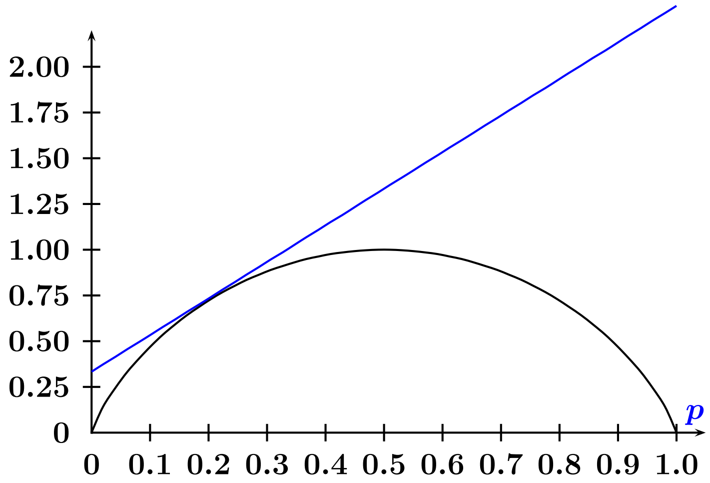
Universal data compression

The results for block length $N = 6$ are shown graphically.



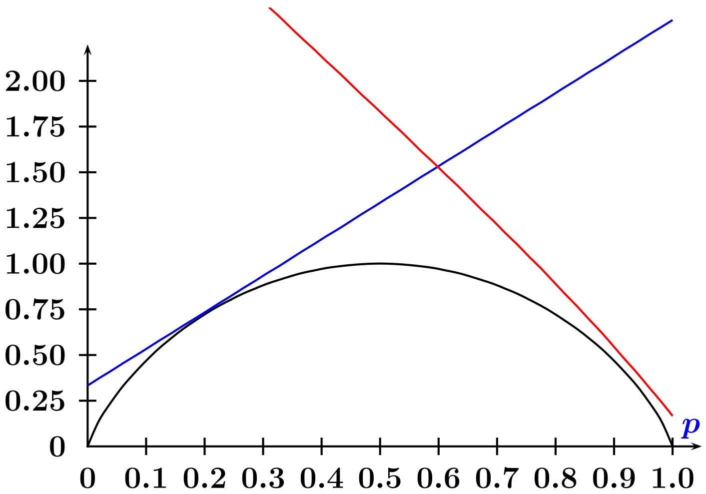
Universal data compression

The results for block length $N = 6$ are shown graphically.



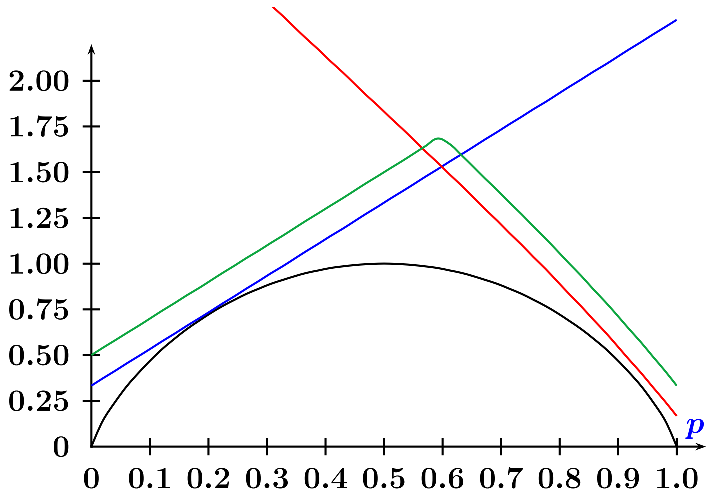
Universal data compression

The results for block length $N = 6$ are shown graphically.



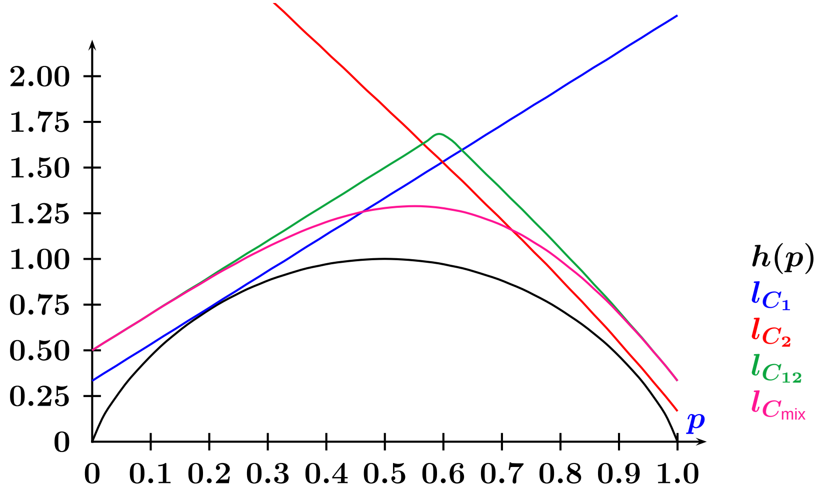
Universal data compression

The results for block length $N = 6$ are shown graphically.



Universal data compression

The results for block length $N = 6$ are shown graphically.



Universal data compression

We conclude that

- Using an ordinary source code only works (well) if we are accurate in predicting the source probabilities.

Universal data compression

We conclude that

- Using an ordinary source code only works (well) if we are accurate in predicting the source probabilities.
- That a two-part code works for more than one source. First part: description of the source (parameters). Second part: the compressed version of the sequence assuming the given source.

Universal data compression

We conclude that

- Using an ordinary source code only works (well) if we are accurate in predicting the source probabilities.
- That a two-part code works for more than one source. First part: description of the source (parameters). Second part: the compressed version of the sequence assuming the given source.
- Mixing (weighting) probabilities works at least as good as the two-part code and can be performed in one run through the data.

Universal data compression

Theorem 1 *[Optimal number of sources] For a sequence x^N generated by an binary i.i.d. source with unknown $\Pr\{X = 1\} = \theta$ the optimal number of alternative sources is of order $\sqrt{N}$ and the achieved redundancy of the resulting code C^*, relative to any i.i.d. source, is bounded as*

$$r_N(C^*) < \frac{\log_2 N}{2N}(1 + \epsilon),$$

and also

$$r_N(C^*) > \frac{\log_2 N}{2N}(1 - \epsilon),$$

for any $\epsilon > 0$ and N sufficiently large.
We shall not prove this theorem here.

Universal data compression

Discussion:

For the binary i.i.d. source which is described by one parameter θ, the optimal redundancy is $\frac{\log_2 N}{2N}$.

Universal data compression

Discussion:

For the binary i.i.d. source which is described by one parameter θ, the optimal redundancy is $\frac{\log_2 N}{2N}$.

This apparently is the cost we must pay for not knowing θ.

Universal data compression

Discussion:

For the binary i.i.d. source which is described by one parameter θ, the optimal redundancy is $\frac{\log_2 N}{2N}$.

This apparently is the cost we must pay for not knowing θ.

It also indicates that the number of discernible sources is roughly $\sqrt{N}$ in this case.

Universal data compression

Discussion:

For the binary i.i.d. source which is described by one parameter θ, the optimal redundancy is $\frac{\log_2 N}{2N}$.

This apparently is the cost we must pay for not knowing θ.

It also indicates that the number of discernible sources is roughly $\sqrt{N}$ in this case.

The next result will explain some of these observations.

Redundancy-capacity theorem

We again take a Bayesian approach.

Let $\mathcal{Q}_\mathcal{C}$ be the set of all dyadic probabilities and $\mathcal{Q}$ be the set of all probabilities.
$\mathcal{S}$ is the set of all sources parametrized by a vector θ that takes values in a parameter space Θ.
We have seen the example of the binary i.i.d. source with a one dimensional parameter $\theta = \Pr\{X = 1\}$ and $\Theta = [0, 1]$.

Redundancy-capacity theorem

If $Q_C \in \mathcal{Q}_\mathcal{C}$ then the redundancy of of the corresponding code C is given by

$$\begin{aligned} r &= \sum_{x^N \in \mathcal{X}^N} p(x^N|\theta) \log_2 \frac{p(x^N|\theta)}{Q_C(x^N)} \\ &= D(p(X^N|\theta) \| Q_C(X^N)) \end{aligned}$$

Let $w(\theta)$ be a prior distribution over θ. The Bayes redundancy is given by

$$\mathcal{D}(w; Q_C) = \int_\Theta D(p(X^N|\theta) \| Q_C(X^N)) w(\theta) \, d\theta$$

Redundancy-capacity theorem

If we allow all probabilities, not only dyadic ones, we obtain:

$$\mathcal{D}(w; Q) = \int_\Theta w(\theta) D(p(X^N|\theta) \| Q(X^N)) \, d\theta$$

Redundancy-capacity theorem

If we allow all probabilities, not only dyadic ones, we obtain:

$$\mathcal{D}(w;Q) = \int_{\Theta} w(\theta) D(p(X^N|\theta)\|Q(X^N))\, d\theta$$
$$= \int_{\Theta} \sum_{x^N \in \mathcal{X}^N} w(\theta) p(x^N|\theta) \log_2 \frac{p(x^N|\theta)}{Q(x^N)}\, d\theta$$

Channel input θ probabilities $w(\theta)$

Redundancy-capacity theorem

If we allow all probabilities, not only dyadic ones, we obtain:

$$\mathcal{D}(w;Q) = \int_{\Theta} w(\theta) D(p(X^N|\theta)\|Q(X^N))\, d\theta$$
$$= \int_{\Theta} \sum_{x^N \in \mathcal{X}^N} w(\theta) p(x^N|\theta) \log_2 \frac{p(x^N|\theta)}{Q(x^N)}\, d\theta$$

Channel transition probabilities $p(x^N|\theta)$

Redundancy-capacity theorem

If we allow all probabilities, not only dyadic ones, we obtain:

$$\mathcal{D}(w;Q) = \int_{\Theta} w(\theta) D(p(X^N|\theta)\|Q(X^N))\, d\theta$$
$$= \int_{\Theta} \sum_{x^N \in \mathcal{X}^N} w(\theta) p(x|\theta) \log_2 \frac{p(x^N|\theta)}{Q(x^N)}\, d\theta$$

Channel output x^N probabilities $Q(x^N)$

Redundancy-capacity theorem

If we allow all probabilities, not only dyadic ones, we obtain:

$$\mathcal{D}(w;Q) = \int_{\Theta} w(\theta) D(p(x^N|\theta)\|Q(X^N))\, d\theta$$
$$= \int_{\Theta} \sum_{x^N \in \mathcal{X}^N} w(\theta) p(x|\theta) \log_2 \frac{p(x^N|\theta)}{Q(x^N)}\, d\theta$$

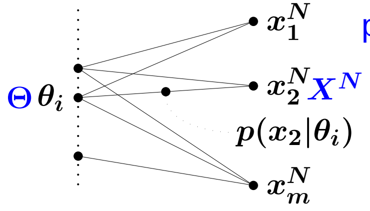
$$= I(\theta; X^N)$$

Redundancy-capacity theorem

We can maximize over all possible priors $w(\theta)$ and find

$$r \geq \max_{w(\theta)} I(\theta; X^N)$$

So the redundancy is lower bounded by (actually it is equal to) the capacity of the channel from the source parameters to the source output sequence x^N.



Redundancy: learning source parameters from data

- Source coding: we don't want this.
- Machine learning: this is what's it about.

The meaning of model information

Efficient description of data can be split into two parts:

- Information about the 'model'

  Universal compression redundancy: The description of the parameters of the data generating process.

- Selection of one of the 'possible' sequences.

  Universal compression: One of the "typical sequences" selected and described with $NH(P_x)$ bits.

The meaning of model information

The first part describes what the model 'can do'.

bits of π: Almost zero complexity. The model is easy to describe and can only generate this sequence. Easy to predict bits.

bits from an i.i.d. source $\theta = \frac{1}{2}$: Highly complex. The model is very simple but the set of possible sequences is large. Hard to predict bits.

The meaning of model information

Occam's razor:

One should not increase, beyond what is necessary, the number of entities required to explain anything.

The most useful statement of the principle for scientists is:

When you have two competing theories which make exactly the same predictions, the one that is simpler is the better.

Universal source coding:

Take the simplest model that describes your data.

Terminology

The two-part description separates model information from random selection

Universal coding: There is a certain unavoidable cost for parameters in a model. It is the price for learning the parameters.

Distinguishable models (parameter values): For a sequence of length N we can use (selection or weighting) about $\sqrt{N}$ distinct values.

Occam's razor: Take the simplest explanation that explains the observations.

Terminology

This results in the notion of stochastic complexity

$$-\log_2 p(x^N|\mathcal{M})$$

$$p(x^N|\mathcal{M}) = \frac{p(x^N|\mathcal{M}, \hat{\theta}(x^N))}{\sum_{x^N \in \mathcal{X}^N} p(x^N|\mathcal{M}, \hat{\theta}(x^N))}$$

is known as the NML (Normalized Maximum Likelihood).
That is must be normalized is reasonable because

$$\sum_{x^N \in \mathcal{X}^N} p(x^N|\mathcal{M}, \hat{\theta}(x^N)) \geq 1$$

And the normalizing constant determines the model cost.
Note that we assume here that the model priors $p(\mathcal{M})$ are all equal!

Terminology

Suppose I have two model classes, $\mathcal{M}_1$ and $\mathcal{M}_2$, for my data x^N and the stochastic complexity $-\log_2 p(x^N|\mathcal{M}_1)$ is smaller than $-\log_2 p(x^N|\mathcal{M}_2)$.
Because the model information part is proportional to $\log_2 N$ and the "noise" part is proportional to N, a smaller complexity means "less noise". So $\mathcal{M}_1$ explains more of the data.
This leads to the Minimum Description Length Principle.

> The best model for the data is is the model that results in the smallest stochastic complexity.

Terminology

Stochastic complexity $\approx$ ideal codeword length.
Coding interpretation:
$L(\theta) = O(\log N); L(\text{noise}) = O(N)$

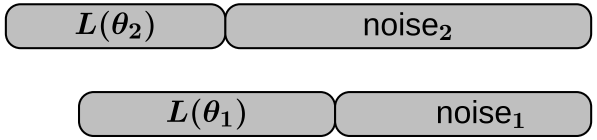
$L(\theta_2)$ | noise$_2$

$L(\theta_1)$ | noise$_1$

Say x^N with $N = 1000$. $L(\text{noise}_2) + L(\theta_2) = 500 + 5k_2$.
With model $\mathcal{M}_1$, x^N has smaller stochastic complexity:
$L(\text{noise}_1) > L(\text{noise}_2)$ hardly possible because $L(\theta_2) - L(\theta_1)$ cannot be large.

Terminology

Stochastic complexity $\approx$ ideal codeword length.
Coding interpretation:
$L(\theta) = O(\log N); L(\text{noise}) = O(N)$



Say x^N with $N = 1000$. $L(\text{noise}_2) + L(\theta_2) = 500 + 5k_2$.

With model $\mathcal{M}_1$, x^N has smaller stochastic complexity:

$L(\text{noise}_1) < L(\text{noise}_2)$ very likely.

So, $\mathcal{M}_1$ explains more of the data (less noise)

Stochastic Complexity (MDL)

"Real data model": binary 1^{th} order Markov,
$\theta_0 = \Pr\{X_i = 1 | x_{i-1} = 0\} = \frac{1}{4}$,
$\theta_1 = \Pr\{X_i = 1 | x_{i-1} = 1\} = \frac{1}{2}$

Then: $\Pr\{X_i = 1\} = \frac{1}{3}$.

$\mathcal{M}_1$ is i.i.d. with $\hat{\theta}_1 \approx \frac{1}{3}$.

$\mathcal{M}_2$ is 1^{th} order Markov with $\hat{\theta}_2 \approx (\frac{1}{4}, \frac{1}{2})$.

$H(X|\mathcal{M}_1, \hat{\theta}_1) = 0.918$ bit.

$H(X|\mathcal{M}_2, \hat{\theta}_2) = 0.874$ bit.

Stochastic Complexity (MDL)

Stochastic complexity

$$S.C._1 \sim \frac{\log_2 N}{2} + 0.918N.$$
$$S.C._2 \sim \log_2 N + 0.874N.$$

For $N < 70 : S.C._1 < S.C._2$ and for
$N > 70 : S.C._1 > S.C._2$.

So if there is not enough data the MDL selects a smaller model than the "true" model.

This is good!

There is not enough data to estimate properly a complex model.