Adaptive Information Processing

Model complexity and the MDL principle

Tjalling Tjalkens and Bert de Vries
Signal Processing Systems Group

June 9, 2017

Contents

Introduction

This part of the course discusses the issue of selecting a most suitable model from a set of possible models, based on the observed data. We will see that model selection is a very different problem than the parameter determination which was discussed in Part 1. With model selection we consider a choice between various models, or parametrized structures, that may contain different numbers of parameters. Choosing a too simple model will not allow us to describe all training data well, thus increasing interpolation errors or extrapolation errors. On the other hand, choosing an overly complex model will describe the training data well, even too well, but cannot generalize well. A too complex model actually assumes structure in data which is really just randomness, i.e. we not only model the data but also the noise in the data.
Part 2 is again subdivided into three parts as indicated below.

Part 2-A: The Bayesian Information Criterion: We show that model selection is not just another Maximum Likelihood problem and we will take a Bayesian point of view to the selection. The main result here is the application of the Laplace Approximation. We show that this approximation allows us to avoid the overfitting problem and this seems to be due to a *parameter cost* penalty.

Part 2-B: Bayesian model estimation and the Context-tree model selection: We continue the Bayesian approach, now for the case of discrete variables, also known as categorical data. We use this result in an advanced model selection problem setting, namely the selection of Context models. Two new results are given here. First the Bayesian mixture as a substitute for the Laplace Approximation for categorical data. This method also gives an appropriate parameter cost penalty. Secondly, the treatment of the class of Context-Tree models show another issue with model selection problems, namely a computational complexity issue. The CTW algorithm is a particular solution to this complexity problem.

Part 2-C: Descriptive complexity: In the previous discussions, the "cost-factor" $\frac{1}{2}\log N$ repeatedly popped up. We here investigate why this is so and will see that this factor is precisely the information theoretical rate of learning a Bernoulli parameter value. Using more Information Theory we show the similarities between universal source coding and the model selection problem as proposed by the Minimum Description Length approach. Following this approach we describe the Stochastic Complexity as a measure to be used in model selection.

Further reading

Bishop's book contains various chapters and sections that give background material and/or additional information related to the model selection problem. We sometimes list relevant chapters. These are not obligatory reading but might help you understanding the context of the model selection issue.
The book "The Minimum Description Length Principle" by Peter D. Grünwald, MIT Press, 2007, ISBN: 9780262529631 gives an in-depth discussion of this approach but is a very advanced text not required for this course.

Additional reading
Introduction
Bishop §1.2: Probability Theory
Bishop §1.3: Model Selection
Bishop §1.4: The Curse of Dimensionality

Probabilities
Bishop §2.1: Binary Variables
Bishop §2.2: Multinomial Variables

Chapter 2-A

The Bayesian Information Criterion

A.1 Parameter and model estimation

Additional reading
Introduction
Bishop §3.3: Bayesian Linear Regression
Bishop §3.4: Bayesian Model Comparison

A.1.1 Define our variables and notation

Before we consider models, parameters (of these models), and the data we introduce their notation.

Model	$\mathcal{M}_i$	model prior	$p(\mathcal{M}_i)$
Parameters	θ_i	parameter prior	$p(\theta_i \mid \mathcal{M}_i)$
Data	x^N		

The *model* $\mathcal{M}$ describes the structure of the data generating process and the *parameters* θ fill in the values of the parameters in the structure.
E.g. the data is a vector x^N of real numbers, generated by an independent and identically distributed (i.i.d.) Gaussian structure with mean m and variance s. So, for all i, with $1 \leq i \leq N$:

$$p(x_i) = \mathcal{N}(x_i|m, s).$$
$$p(x^N) = \prod_{i=1}^{N} \mathcal{N}(x_i|m, s).$$

In this case we could say that the data generating process structure $\mathcal{M}$ is i.i.d. Gaussian with parameters $\theta = (m, s)$ and then we write

$$p(x_i|\mathcal{M}, \theta) = \mathcal{N}(x_i|m, s).$$
$$p(x^N|\mathcal{M}, \theta) = \prod_{i=1}^{N} \mathcal{N}(x_i|m, s).$$

So, with $\mathcal{M}$ and θ we write the structure and parameters more explicitly.

The advantage of this notation is that we can easily talk about "the probability of the data, given the model structure but not knowing the parameter values", which becomes

$$p(x^N|\mathcal{M}) = \int_\Theta p(\theta|\mathcal{M})p(x^N|\mathcal{M},\theta)\,d\theta.$$

Also we can write the posterior distribution of the parameters, given the data and the model structure as:

$$p(\theta|x^N,\mathcal{M}).$$

We can in the same way describe the posterior distribution of the *model* given the data x^N. Note that in the following we consider $\mathcal{M}$ as a random variable, that can take various *structures* as value. These structure values or possible models as collected in the model set $\mathfrak{M}$. E.g. consider an i.i.d. Gaussian structure and call it $\mathcal{M}_A$, and consider an i.i.d. Gamma structure and call it $\mathcal{M}_B$. The model set is now $\mathfrak{M} = \{\mathcal{M}_A, \mathcal{M}_B\}$. Structure $\mathcal{M}_A$ has a parameter vector $\theta_A = (m, s)$, where m is the mean and s is the variance of the distribution. The Gamma distribution is defined by two parameters, the *shape* parameter α and the *rate* parameter β and is given as

$$p(x) = \frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}.$$

So, structure $\mathcal{M}_B$ is the Gamma structure with parameters $\theta_B = (\alpha, \beta)$. We are given, or assume, a *prior* over the model variable $\mathcal{M}$: $P(\mathcal{M})$. As we will explain in more detail in the next two sections, we can now write

$$p(x^N|\mathcal{M}_A) = \int_{\Theta_A} p(\theta_a|\mathcal{M}_A)p(x^N|\mathcal{M}_A,\theta_A)\,d\theta_A,$$
$$P(\mathcal{M}_A|x^N) = \frac{P(\mathcal{M}_A)p(x^N|\mathcal{M}_A)}{p(x^N)},$$

where

$$p(x^N) = P(\mathcal{M}_A)p(x^N|\mathcal{M}_A) + P(\mathcal{M}_B)p(x^N|\mathcal{M}_B).$$

A.1.2 Point estimates for parameters

As said before, we use "Bayes Rule" that will give us the posterior parameter distribution, i.e. the distribution of the parameters θ_i of the model $\mathcal{M}_i$, given the observed data x^N.

$$p(\theta_i|\mathcal{M}_i, x^N) = \frac{p(\theta_i|\mathcal{M}_i)p(x^N|\mathcal{M}_i,\theta_i)}{p(x^N|\mathcal{M}_i)}$$

In the denominator we marinalize out the value of the parameters, so

$$p(x^N|\mathcal{M}_i) = \int_{\Theta_i} p(\theta_i|\mathcal{M}_i)p(x^N|\mathcal{M}_i,\theta_i)\,d\theta_i$$

We want a point estimate for θ_i (given $\mathcal{M}_i$).
We can use the Maximum A-posteriori Probability (MAP) rule:

$$\hat{\theta}_i = \arg\max_{\theta_i} p(\theta_i|\mathcal{M}_i, x^N).$$

Or, in the case of a uniform prior or when we don't want to use a prior the Maximum Likelihood (ML) rule:

$$\hat{\theta}_i = \arg\max_{\theta_i} p(x^N|\mathcal{M}_i, \theta_i).$$

A.1.3 Point estimates for models

We try to use the "Bayes Rule" approach, that was so successful for finding good parameter values.

$$p(\mathcal{M}_i|x^N) = \frac{p(\mathcal{M}_i)p(x^N|\mathcal{M}_i)}{p(x^N)}$$

$$p(x^N) = \int_{\mathfrak{M}_i} p(\mathcal{M}_i)p(x^N|\mathcal{M}_i)\, d\mathcal{M}_i$$

We want a point estimate for $\mathcal{M}$ (MAP or ML).

$$\hat{\mathcal{M}} = \arg\max_{\mathcal{M}_i} p(\mathcal{M}_i|x^N) = \arg\max_{\mathcal{M}_i} p(x^N|\mathcal{M}_i)$$

Where we assume a uniform prior or want to work without priors.
We need to compute a rather complicated integral

$$p(x^N|\mathcal{M}_i) = \int_{\Theta_i} p(\theta_i|\mathcal{M}_i)p(x^N|\mathcal{M}_i, \theta_i)\, d\theta_i$$

Often $p(\theta_i|\mathcal{M}_i, x^N)$ is sharply peaked and because

$$p(\theta_i|\mathcal{M}_i, x^N) \propto p(\theta_i|\mathcal{M}_i)p(x^N|\mathcal{M}_i, \theta_i),$$

we might be able to approximate the integrand given above.

A.2 Maximum Likelihood and Overfitting

Additional reading
Bishop §1.1: Example: Polynomial Curve Fitting

We approximate the integral by the peak of the integrand, which occurs at θ_i^{MAP}. For convenience we often use θ_i^{ML}, because this is simpler to derive though it might be far off from the optmum.

$$p(\theta_i|\mathcal{M}_i)p(x^N|\mathcal{M}_i, \theta_i) \approx \delta(\theta_i - \theta_i^{\text{ML}})p(\theta_i|\mathcal{M}_i)p(x^N|\mathcal{M}_i, \theta_i)$$

and find

$$p(x^N|\mathcal{M}_i) \propto p(\theta_i^{\text{ML}}|\mathcal{M}_i)p(x^N|\mathcal{M}_i, \theta_i^{\text{ML}})$$

So we end up with (with or without model prior)

$$\mathcal{M}^{\text{MAP}} = \arg\max_{\mathcal{M}_i} p(\mathcal{M}_i)p(\theta_i^{\text{MAP}}|\mathcal{M}_i)p(x^N|\mathcal{M}_i, \theta_i^{\text{MAP}})$$

$$\mathcal{M}^{\text{ML}} = \arg\max_{\mathcal{M}_i} p(x^N|\mathcal{M}_i, \theta_i^{\text{ML}})$$

A.2.1 Example: Linear regression

We will apply the ML approach to a simple model selection problem, namely to determine which parameters actually play a role in a linear regression problem. The regression model has k parameters. One could argue that parameters with a value zero actually do not exist.
We have the following model equations.

$$y_n = \theta^T \underline{x}_n + n_n;$$
$$y_n \in \mathbb{R}; \qquad \theta \in \mathbb{R}^k; \qquad \underline{x}_n \in \mathbb{R}^k; \qquad n_n \sim \mathcal{N}(0, \sigma^2)$$

Observe: $(y_1, \underline{x}_1), (y_2, \underline{x}_2), \ldots, (y_N, \underline{x}_N)$.
A single observation consists of the inputs $\underline{x}$ and the resulting output y. We observe N independent experiments.

ML estimate: $\hat{\theta} = (X^T X)^{-1} X^T \underline{y}$.
This is already explained in Part 1, on Gaussian models.

Matrix: $X = [\underline{x}_1, \underline{x}_2, \ldots \underline{x}_N]^T$.

Models: $\mathcal{M} \subset \{1, 2, \ldots, k\}$. e.g.

$$\mathcal{M} = \{1, 3\}; \qquad y_n = \theta_1 x_{n1} + \theta_3 x_{n3} + n_n$$

So, a parameter index that is not in the model $\mathcal{M}$ is actually forced to zero, and thus isn't really present, it's value doesn't need to be estimated. The model becomes simpler.

First we use 50 data samples: y^{50} given the inputs $(x_{i,1}, x_{i,2}, x_{i,3})_{i=1}^{50}$. The table below lists the outcomes of the experiments. The first column lists the model used, the three next columns give the ML estimates of the parameters θ where '0' denotes an unused parameter. $\hat{\sigma}^2$ gives the estimated variance of the noise n_n, which obviously also depends on the estimated $\hat{\theta}$. P_{ML} is the probability of the resulting noise vector $\underline{n}$ given as

$$P_{ML} = p(y^{50} | \underline{x}^{50}, \hat{\theta}) = \prod_{i=1}^{50} \mathcal{N}(y_i - \hat{\theta}\underline{x}_i | 0, \sigma^2).$$

$$N = 50; \quad \underline{x} \in [0, 1]^3; \quad \theta = (0, 0.6, 0);$$
$$\sigma^2 = 1 \quad \text{actual } \sigma^2 = 0.799$$

$\mathcal{M}$	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\sigma}^2$	$\ln P_{ML}$ $\sigma^2 = 1$	$\ln P_{ML}$ $\sigma^2 = \hat{\sigma}^2$
{}	0	0	0	0.949	−69.675	−69.642
{1}	0.690	0	0	0.804	−66.040	−65.485
{2}	0	0.604	0	0.799	−65.934	−65.352
{3}	0	0	0.307	0.912	−68.738	−68.635
{12}	0.379	0.361	0	0.780	−65.441	−64.728
{13}	1.171	0	−0.522	0.766	−65.099	−64.286
{23}	0	0.970	−0.472	0.766	−65.101	−64.287
{123}	0.908	0.752	−0.940	0.686	−63.097	−61.525

From the table we observe that the largest probability is obtained by model {123}, so the most complex model. This could be an exception, caused by too few data.

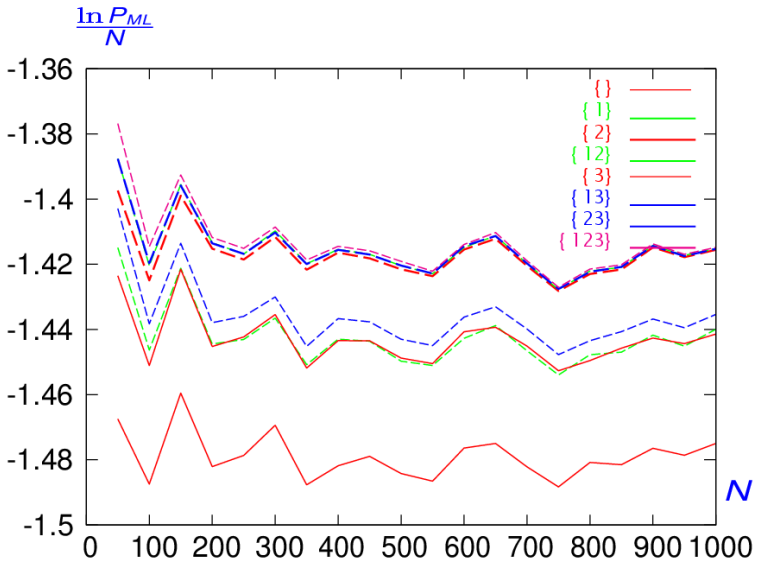
We try again with 1000 data samples: y^{1000} given the inputs $(x_{i,1}, x_{i,2}, x_{i,3})_{i=1}^{1000}$.

$N = 1000; \quad \underline{x} \in [0,1]^3; \quad \theta = (0, 0.6, 0);$

$\sigma^2 = 1 \quad$ actual $\sigma^2 = 1.015$

$\mathcal{M}$	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\sigma}^2$	$\ln P_{ML}$ $\sigma^2 = 1$	$\ln P_{ML}$ $\sigma^2 = \hat{\sigma}^2$
$\{\}$	0	0	0	1.144	−1491	−1486
$\{1\}$	0.435	0	0	1.083	−1460	−1459
$\{2\}$	0	0.619	0	1.015	−1426	−1426
$\{3\}$	0	0	0.507	1.058	−1448	−1447
$\{12\}$	−0.099	0.693	0	1.013	−1425	−1425
$\{13\}$	0.105	0	0.430	1.056	−1447	−1446
$\{23\}$	0	0.549	0.095	1.013	−1426	−1426
$\{123\}$	−0.173	0.622	0.167	1.010	−1424	−1424

Again we see that the most complex model gives the largest probability to the sequence, however, the margin is small.
In the following graph we plot the probability of the sequence of y values, given the inputs $\underline{x}^3$ and the maximum likelihood parameters $\underline{\hat{x}}^3$, normalized to the sequence length N.



From the graph we conclude that the "only noise" model $\mathcal{M} = \{\}$ has the worst performance, and that all models that include the actual parameter θ_2, i.e. $\{\{2\}, \{1,2\}, \{2,3\}, \{1,2,3\}\}$ perform almost the same and the most complex of these, $\{1,2,3\}$, performs the best but is clearly an unwanted over-estimation.

A.2.2 Example: A discrete Markov structure.

Introducing the structure

Consider a binary second order Markov process: $\Pr\{X_i = 1|x^{i-1}\} = \Pr\{X_i = 1|x_{i-2}x_{i-1}\}$. See section **??** for background information on Markov structures.
So, it is actually a set of four i.i.d. sub-sources.
This implies that we can describe the sequence probability as the product of the probabilities of

the four i.i.d. sub-sequences,

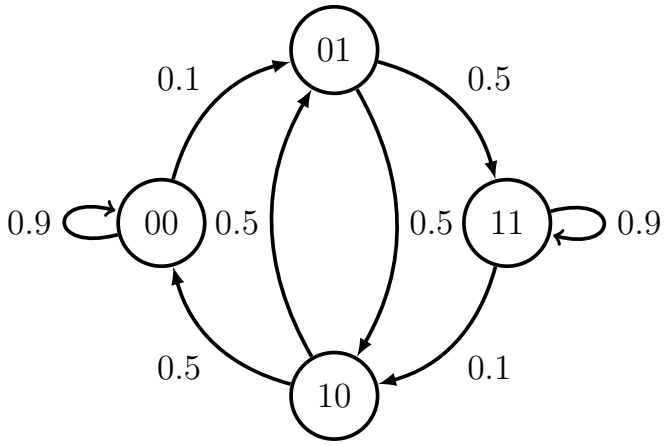
$$P_{ML} = p(x^N|\theta) = \prod_{i=1}^{4} p(\underline{x}[i]|\theta_i),$$

where $\underline{x}[i]$ denotes the i^{th} sub-sequence.
The ML estimate of an i.i.d. binary source is given as

$$n(s|x) = \text{the number of times } s \in \mathcal{X}^* \text{ occurs in } x$$
$$p(x^N|\theta) = (1-\theta)^{n(0|x^N)}\theta^{n(1|x^N)}$$

To find the maximal probability we differentiate

$$\frac{\partial}{\partial\theta}\ln p(x^N|\theta) = \frac{n(1|x^N) - N\theta}{\theta(1-\theta)} = 0$$
$$\hat{\theta} = \frac{n(1|x^N)}{N}$$



Let S be the state variable of an m-th order Markov source, so $S_i = X_{i-m}\dots X_{i-1}$ and $l(S_i) = m$ bits, then

$$\theta_s = \Pr\{X_i = 1|S_i = s\}$$

The Maximum Likelihood estimator is

$$\hat{\theta}_s = \frac{n(s1|x^N)}{n(s0|x^N) + n(s1|x^N)}$$

With this we find the ML probability for x^N (with initial state ς, see next slide)

$$p(x^N|m, \underline{\hat{\theta}}, \varsigma) = \prod_{s\in\{0,1\}^m} \left\{\hat{\theta}_s^{n(s1|x^N)}(1-\hat{\theta}_s)^{n(s0|x^N)}\right\}$$

Intermezzo: Initial state

The state variable of a m-th order Markov source is defined by the most recent m past symbols of the source. Initially, we cannot know the source state because we haven't seen m or more symbols yet.

So we assume knowledge of some initializing symbols that help in defining the first m values of the state variable. We will denote these initial symbols by ς and usually leave them unspecified. This also implies that when we use the count function $n(s0|x^N)$ we imply the use of ς, e.g. let $x^5 = 10110$, $m = 2$, and $\varsigma = 01$. We wish to count the number of ones in state $s = 01$.

$$n(s1|x^5) = 2$$

We consider the concatenation of ς and x^5: 01 101100
and count the number of occurrences of $s1 = 011$ in that string.

s	$n(s0\|x^5)$	$n(s1\|x^5)$
00	0	0
01	0	2
10	0	1
11	2	0

So indeed, we count 2 zeros and 3 ones.

The simulations results

We run a simulation on a sequence of 50 binary symbols, generated by the second-order Markov source with probabilities as given in the figure above. We determine the ML sequence probability assuming a Markov order ranging from 0 up to 4, inclusive. We also run a simulation on 200 symbols.

```
octave:1> mytest(50,[0.1,0.5,0.5,0.9],4)
ML models sequence logprobs:
Order 0:  logpr = -34.657359
Order 1:  logpr = -14.546445
Order 2:  logpr = -14.883390
Order 3:  logpr = -15.185437
Order 4:  logpr = -15.444986

octave:2> mytest(200,[0.1,0.5,0.5,0.9],4)
ML models sequence logprobs:
Order 0:  logpr = -137.416984
Order 1:  logpr = -102.949521
Order 2:  logpr = -87.992931
Order 3:  logpr = -84.732718
Order 4:  logpr = -80.546002
```

For 50 symbols the 1-th order model performs best, possibly because there is not enough data. At 200 symbols we again observe the over-fitting problem, again the most complex model performs best.

A.2.3 Conclusions

Obviously, this method does not work.

- Any model that includes the actual model assigns essentially the same probability to the data.
- We observe that (usually) the higher order models give higher probabilities to the sequence.
- But high order models cannot predict well (too restricted).
- The higher order models are too well tuned.

This is undesirable, the estimated model adapts itself to the noise and the resulting model is an over estimation of the actual model. model.

A.3 Preventing overfitting

Additional reading Laplace Approximation

Bishop §4.4: The Laplace Approximation

A.3.1 Laplace approximation

We approximate the integrand by a Gaussian around the peak. The mean and variance of the Gaussian are determined by the integrand.
This approximation turns out to give more interesting results.
Suppose we have an arbitrary non-negative real function $f(z)$, where z is a k-dimensional vector.
We need an estimate of the normalizing constant Z_f.

$$Z_f = \int f(z)\, dz$$

Let z_0 be a maximum of $f(z)$. Use the Taylor expansion.

$$\ln f(z) \approx \ln f(z_0) - \frac{1}{2}(z - z_0)A(z - z_0)$$
$$A_{ij} = -\frac{\partial^2}{\partial z_i \partial z_j} \ln f(z)\Big|_{z=z_0}$$

Approximate $f(z)$ by the unnormalized Gaussian

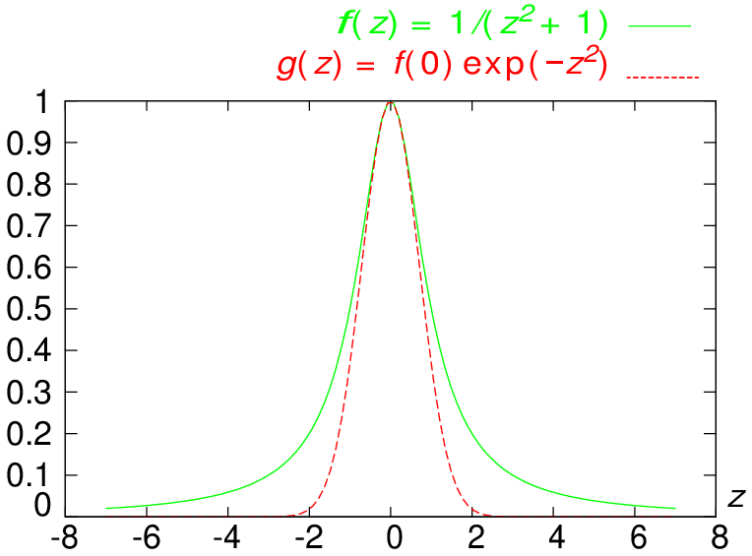
$$g(z) = f(z_0) \exp\left(-\frac{1}{2}(z - z_0)A(z - z_0)\right)$$

A, not necessarily good, approximation of Z_f is

$$Z_f \approx Z_g = \int g(z)\, dz = f(z_0)\sqrt{\frac{(2\pi)^k}{\det A}}$$

Example 1.

$$f(z) = \frac{1}{z^2 + 1} \quad \text{Has maximum at } z_0 = 0.$$
$$Z_f = \pi$$
$$A = -\frac{\partial^2}{\partial z^2} \ln f = -\frac{f''f - f'^2}{f^2}$$
$$f(0) = 1; \quad f'(0) = 0; \quad f''(0) = -2; \text{ so } A = 2$$
$$g(z) = f(0) \exp\left(-\frac{1}{2} zAz\right) = e^{-z^2}$$
$$Z_g = \sqrt{\pi}$$



Consider again $p(x^N|\mathcal{M}_i)$.

$$p(x^N|\mathcal{M}_i) = \int_\Theta p(\theta_i|\mathcal{M}_i)p(x^N|\mathcal{M}_i,\theta_i)\,d\theta_i$$

We again use the fact that

$$p(\theta_i|\mathcal{M}_i,x^N) \propto p(\theta_i|\mathcal{M}_i)p(x^N|\mathcal{M}_i,\theta_i)$$

is often sharply peaked, say at $\hat{\theta}_i$. Using the Laplace approximation we may write

$$p(x^N|\mathcal{M}_i) \approx \sqrt{\frac{(2\pi)^k}{\det A}}p(\hat{\theta}_i|\mathcal{M}_i)p(x^N|\mathcal{M}_i,\hat{\theta}_i)$$

Comparing two models give

$$\frac{p(\mathcal{M}_i|x^N)}{p(\mathcal{M}_j|x^N)} \approx \frac{p(\mathcal{M}_i)}{p(\mathcal{M}_j)} \frac{\sqrt{\frac{(2\pi)^{k_i}}{\det A_i}}p(\hat{\theta}_i|\mathcal{M}_i)}{\sqrt{\frac{(2\pi)^{k_j}}{\det A_j}}p(\hat{\theta}_j|\mathcal{M}_j)} \frac{p(x^N|\mathcal{M}_i,\hat{\theta}_i)}{p(x^N|\mathcal{M}_j,\hat{\theta}_j)}$$

Initial model preference
Cost of (number of) parameters
Likelihood ratio

This is ML model estimation, it works because we consider the model complexity also!

A.3.2 The Bayesian Information Criterion

A more refined approximation (Schwartz criterion or Bayesian Information Criterion) gives

$$\log\frac{p(\mathcal{M}_i|x^N)}{p(\mathcal{M}_j|x^N)} \approx \log\frac{p(\mathcal{M}_i)}{p(\mathcal{M}_j)} + \log\frac{p(x^N|\mathcal{M}_i,\hat{\theta}_i)}{p(x^N|\mathcal{M}_j,\hat{\theta}_j)} - \frac{1}{2}(k_i-k_j)\log N,$$

where k_i resp. k_j gives the number of free parameters in model $\mathcal{M}_i$ or $\mathcal{M}_j$ respectively.

This BIC is widely applied and turned out to be very useful.

What happens when we apply the correction term $\frac{k}{2}\log N$? We shall revisit the two examples.

A.3.3 Example: Linear regression with BIC correction

$N = 50; \quad \underline{x} \in [0,1]^3; \quad \theta = (0, 0.6, 0);$

$\sigma^2 = 1 \quad$ actual $\sigma^2 = 0.852$

$\mathcal{M}$	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\sigma}^2$	$\ln P_{BIC}$
{}	0	0	0	1.068	−72.653
{1}	0.699	0	0	0.909	−70.632
{2}	0	0.773	0	0.841	−68.923
{3}	0	0	0.572	0.944	−71.491
{12}	0.159	0.662	0	0.837	−70.790
{13}	0.553	0	0.172	0.905	−72.478
{23}	0	0.811	−0.050	0.840	−70.869
{123}	0.240	0.728	−0.159	0.834	−72.670

$N = 1000; \quad \underline{x} \in [0,1]^3; \quad \theta = (0, 0.6, 0);$

$\sigma^2 = 1 \quad$ actual $\sigma^2 = 0.977$

$\mathcal{M}$	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\sigma}^2$	$\ln P_{BIC}$
{}	0	0	0	1.077	−1457.2
{1}	0.411	0	0	1.022	−1433.4
{2}	0	0.551	0	0.976	−1410.3
{3}	0	0	0.362	1.034	−1439.3
{12}	−0.017	0.564	0	0.976	−1413.7
{13}	0.315	0	0.128	1.020	−1435.6
{23}	0	0.637	−0.117	0.974	−1412.7
{123}	0.040	0.620	−0.133	0.974	−1416.1

A.3.4 Example: A discrete Markov structure with BIC correction

```
octave:1> mytest(50,[0.1,0.5,0.5,0.9],4)
Parameter scaled ML log probabilities:
Order 0:  logpr = -36.613371
Order 1:  logpr = -18.458468
Order 2:  logpr = -22.707436
Order 3:  logpr = -30.833529
Order 4:  logpr = -46.741170

octave:2> mytest(200,[0.1,0.5,0.5,0.9],4)
Parameter scaled ML log probabilities:
Order 0:  logpr = -140.066143
Order 1:  logpr = -108.247838
Order 2:  logpr = -98.589565
Order 3:  logpr = -105.925987
Order 4:  logpr = -122.932541
```

A.3.5 Conclusion

The examples indicate that the correct model (order) is recovered, basically by using an ML selection criterion with an additional penalty term for the model complexity.

However, this BIC is derived as an approximation to the true Bayesian a-posteriori probability.

A better justification for the BIC should exist!

Chapter 2-B

Bayesian model estimation

B.1 Introduction

In this chapter we develop the Bayesian model estimation, or hierarchical Bayes approach, in more detail.
First we will give an example of a Bayesian averaging to obtain good sequence probabilities when the parameters of a model are unknown a-priori. The resulting sequence probability is *close* to any sequence probability as given by the model structure and any possible value of the parameters, in the sense that the normalized probability ratio converges to 1. The cost of not knowing the parameter value has a very similar form as the Laplace approximation gives.
Secondly we consider applying these probabilities in a Finite order Markov setting, showing that in that setting the same convergence and model cost occurs.
The we introduce a more efficient class, known as *tree models*. These are special cases of finite order Markov models that allow a more efficient description with fewer parameters for many practical data modeling problems. We now formulate a problem where not only the parameter values are unknown but also the actual tree structure of the model is unknown. We develop a hierarchical Bayes approach to solving the efficient sequence probability assignment question in this case.
A major drawback of this approach is its computational complexity and we apply an efficient method known as the *Context-tree-weighting method* to generate the probabilities efficiently.

Additional reading

Bishop §14.1: Bayesian Model Averaging
Bishop §14.4: Tree-based Models

B.2 Additional notation

i^{th} order Markov	$\mathcal{M}_i$	The state is determined by the previous i symbols.
Parameter vector	θ_i	This vector describes all probabilities $p(X_n \mid X_{n-i}, X_{n-i+1}, \ldots, X_{n-1})$.
Parameter element	$\theta_{i,s}$	$\theta_{i,s} = p(X_n \mid X_{n-i}^{n-1} = s)$.
Model state	s	s is a binary sequence of length i.
Initial state	ς	ς is also a binary sequence of length i.

B.3 Beta-weighted probabilities for i.i.d. sequences

Let $\mathcal{M}$, with a single real valued parameter θ, assign probabilities to binary sequences, so it generates sequences x^N, where $x_i \in \{0, 1\}$ for all i, $1 \leq i \leq N$. The parameter $\theta \in [0, 1]$ describes

the probability of the i.i.d. variables X_i, or

$$\Pr\{X_i = 1\} = \theta.$$

So we define a probability function $P(x)$ as,

$$\begin{aligned} P(0) &= 1 - \theta, \\ P(1) &= \theta. \end{aligned}$$

From the i.i.d. assumption follows

$$\begin{aligned} P(x^N|\mathcal{M}, \theta) &= \Pr\{X^N = x^N\}, \\ &= \prod_{i=1}^{N} P(x_i). \end{aligned}$$

In order to stress the fact that we know the model $\mathcal{M}$ (binary i.i.d. generation of symbols) and the value of the parameter θ we write $P(x^N|\mathcal{M}, \theta)$.

We consider the more practical situation where we do know that the data is generated i.i.d., but we do *not* know the probabilities of the symbols, so we ask the following question.

Question: What is a good probability to assign to x^N if θ is not known?
Solution: We will take a Bayesian approach, assuming a prior distribution $p(\theta)$ over θ and calculating the resulting marginal sequence probability,

$$P(x^n|\mathcal{M}) = \int_\Theta p(\theta) P(x^N|\mathcal{M}, \theta)\, d\theta.$$

B.4 Finite order Markov

Example 2 (Revisit first lecture). Let $\mathcal{M}_i$ be the i-th order binary Markov model (source). Then $\theta_i \in \Theta_i = [0,1]^{2^i}$ and $\theta_{i,s} = p(X_n = 1|x_{n-i}^{n-1} = s)$. So in the binary case the parameter vector per state $\theta_{i,s}$ describes the probability of a '1' in that state.
Beta distribution for prior $p(\theta_i|\mathcal{M}_i)$, with $\alpha = \beta = \frac{1}{2}$ (Jeffreys prior).

$$\begin{aligned} p(\theta_i|\mathcal{M}_i, \varsigma) &= \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\right)^{2^i} \prod_{s\in\{0,1\}^i} \theta_{i,s}^{\alpha-1}(1-\theta_{i,s})^{\beta-1} \\ &= \frac{1}{\pi^{2^i}} \prod_{s\in\{0,1\}^i} \theta_{i,s}^{-1/2}(1-\theta_{i,s})^{-1/2} \end{aligned}$$

$$
\begin{aligned}
p(x^N|\mathcal{M}_i,\varsigma) &= \int_{\Theta_i} p(\theta_i|\mathcal{M}_i)p(x^N|\mathcal{M}_i,\theta_i)\,d\theta_i \\
&= \frac{1}{\pi^{2^i}} \\
&\quad \int_{\Theta_i} \prod_{s\in\{0,1\}^i} \theta_{i,s}^{n(s1|x^N)-1/2}(1-\theta_{i,s})^{n(s0|x^N)-1/2}\,d\theta_i \\
&= \frac{1}{\pi^{2^i}} \\
&\quad \prod_{s\in\{0,1\}^i} \int_{[0,1]} \theta_{i,s}^{n(s1|x^N)-1/2}(1-\theta_{i,s})^{n(s0|x^N)-1/2}\,d\theta_{i,s} \\
&= \prod_{s\in\{0,1\}^i} \frac{\Gamma(n(s0|x^N)+\frac{1}{2})\Gamma(n(s1|x^N)+\frac{1}{2})}{\pi\Gamma(n(s0|x^N)+n(s1|x^N)+1)}
\end{aligned}
$$

So we must study the behaviour of

$$
\begin{aligned}
P_e(a,b) &\triangleq \frac{\Gamma(a+\frac{1}{2})\Gamma(b+\frac{1}{2})}{\pi\Gamma(n+1)} \\
a &\triangleq n(0|x^N) \\
b &\triangleq n(1|x^N)
\end{aligned}
$$

It is a memoryless sub-sources of the Markov source. x^N is generated i.i.d. with parameter θ.

The actual probability of x^N under this source is

$$
p(x^N|\mathcal{M},\theta) = (1-\theta)^a\theta^b
$$

We can write

$$
P_e(a,b) = \frac{\frac{1}{2}\frac{3}{2}\cdots(a-\frac{1}{2})\cdot\frac{1}{2}\frac{3}{2}\cdots(b-\frac{1}{2})}{(a+b)!}
$$

We can also consider the conditional probabilities, $P_e(1|a,b)$, the probability of a "1" following a zeros and b ones, and $P_e(0|a,b)$, the probability of a "0" following a zeros and b.

$$
\begin{aligned}
P_e(1|a,b) &= \frac{P_e(a,b+1)}{P_e(a,b)} = \frac{b+\frac{1}{2}}{a+b+1} \\
P_e(0|a,b) &= \frac{P_e(a+1,b)}{P_e(a,b)} = \frac{a+\frac{1}{2}}{a+b+1}
\end{aligned}
$$

e.g. consider the sequence $x^7 = 0110010$. It contains $a=4$ zeros and $b=3$ ones, so (with some abuse of notation)

$$
\begin{aligned}
P_e(x^7) = P_e(4,3) &= \frac{\frac{1}{2}\frac{3}{2}\frac{5}{2}\frac{7}{2}\frac{1}{2}\frac{3}{2}\frac{5}{2}}{7!} \\
&= \frac{5}{2048}. \\
P_e(x^7) &= P_e(x_1)P_e(x_2|x_1)\cdots P_e(x_7|x_1^6) \\
&= P_e(0|0,0)P_e(1|1,0)P_e(1|1,1)P_e(0|1,2) \\
&\quad P_e(0|2,2)P_e(1|3,2)P_e(0|3,3) \\
&= \frac{\frac{1}{2}}{1}\frac{\frac{1}{2}}{2}\frac{\frac{3}{2}}{3}\frac{\frac{3}{2}}{4}\frac{\frac{5}{2}}{5}\frac{\frac{5}{2}}{6}\frac{\frac{7}{2}}{7} = \frac{5}{2048}.
\end{aligned}
$$

Again with the help of Stirling's approximation we can derive, for large a and b the following. (Excercise). Note: $a + b = N$.

$$\log_2 \frac{p(x^N|\mathcal{M}, \theta)}{P_e(a, b)} \leq \frac{1}{2} \log_2 N + \frac{1}{2} \log_2 \frac{\pi}{2}$$

Actually, we can prove that for all $a \geq 0$ and $b \geq 0$

$$\log_2 \frac{p(x^N|\mathcal{M}, \theta)}{P_e(a, b)} \leq \frac{1}{2} \log_2 N + 1$$

Back to the i-th order Markov source.

$$p(x^N|\mathcal{M}_i, \theta_i, \varsigma) = \prod_{s \in \{0,1\}^i} \theta_{i,s}^{n(s1|x^N)} (1 - \theta_{i,s})^{n(s0|x^N)}$$
$$p(x^N|\mathcal{M}_i, \varsigma) = \prod_{s \in \{0,1\}^i} P_e(n(s1|x^N), n(s0|x^N))$$

With the previous bound we find

$$\begin{aligned}
&\log_2 \frac{p(x^N|\mathcal{M}_i, \theta_i, \varsigma)}{p(x^N|\mathcal{M}_i, \varsigma)} = \\
&\quad \log_2 \frac{\prod_{s \in \{0,1\}^i} \theta_{i,s}^{n(s1|x^N)} (1 - \theta_{i,s})^{n(s0|x^N)}}{\prod_{s \in \{0,1\}^i} P_e(n(s1|x^N), n(s0|x^N))} \\
&= \sum_{s \in \{0,1\}^i} \log_2 \frac{\theta_{i,s}^{n(s1|x^N)} (1 - \theta_{i,s})^{n(s0|x^N)}}{P_e(n(s1|x^n), n(s0|x^N))} \\
&\leq \sum_{s \in \{0,1\}^i} \frac{1}{2} \log_2 n(s|x^N) + 1 \overset{*1}{\leq} \frac{2^i}{2} \log_2 \frac{N - i}{2^i} + 2^i
\end{aligned}$$

(*1): Here we use Jensen's inequality.
So we conclude that for any parameter vector θ_i we have (approximately!)
[From now on we do not explicitly write ς anymore]

$$\log_2 p(x^N|\mathcal{M}_i) \approx \log_2 p(x^N|\mathcal{M}_i, \theta_i) - \frac{2^i}{2} \log_2 \frac{N - i}{2^i} - 2^i$$

Maximum Likelihood parameters (and $N \gg \max\{2^i, 2^j\}$)

$$\log_2 \frac{p(\mathcal{M}_i|x^N)}{p(\mathcal{M}_j|x^N)} \approx \log_2 \frac{p(\mathcal{M}_i)}{p(\mathcal{M}_j)} + \log_2 \frac{p(x^N|\mathcal{M}_i, \hat{\theta}_i)}{p(x^N|\mathcal{M}_j, \hat{\theta}_j)} - \frac{2^i - 2^j}{2} \log_2 N$$

So, again we observe the parameter cost!
Recap: Memoryless binary source: one parameter $\theta = \Pr\{X = 1\}$

Recap: Markov order-k: one parameter per state. There are 2^k states. The k symbols $x_{i-k}, \ldots, x_{i-1}$ form the context of the symbol x_i.

Real world models: Length of context depends on its contents. e.g. Natural language (English, Dutch, $\cdots$): if context starts with x_{i-1} ='q' then no more symbols are needed.
A tree source is a nice concept to describe such sources.

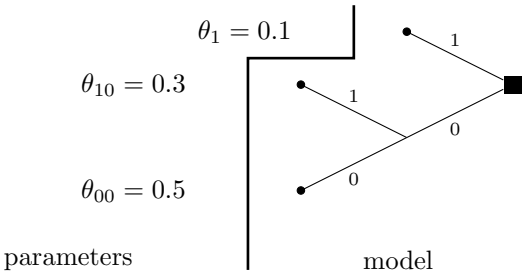
A tree source consists of a set $\mathcal{S}$ of suffixes that together form a tree.
To each suffix (leaf) s in the tree there corresponds a parameter θ_s.

Some more notation: By $x^N_{|s}$ we denote the sub-sequence of symbols from x^N that are preceded by the sequence s.

Example: $x^8 = 01011010$; $s = 01$; then $x^8_{|01} = x_3x_5x_8 = 010$.

Example 3. Let $\mathcal{S} \triangleq \{00, 10, 1\}$ and $\theta_{00} = 0.5, \theta_{10} = 0.3$, and $\theta_1 = 0.1$ then

$$\Pr\{X_i = 1|\cdots, x_{i-2} = 0, x_{i-1} = 0\} = 0.5,$$
$$\Pr\{X_i = 1|\cdots, x_{i-2} = 1, x_{i-1} = 0\} = 0.3,$$
$$\Pr\{X_i = 1|\cdots, x_{i-1} = 1\} = 0.1.$$



Just as before ("Bayesian model estimation") we must estimate the sequence probabilities of the memoryless subsources that correspond to the leaves of the tree (states of the source).
Let the full sequence be x^N and the subsequence for state s be written as $x^N_{|s}$. Before we wrote

$$P_e(a, b) = \frac{\Gamma(a + \frac{1}{2})\Gamma(b + \frac{1}{2})}{\pi\Gamma(a + b + 1)}$$

We shall now use the shorthand notation for the estimated probability of the subsequence generated in state s given the full sequence x^i:

$$P_e(a_s, b_s) = \frac{\Gamma(a_s + \frac{1}{2})\Gamma(b_s + \frac{1}{2})}{\pi\Gamma(a_s + b_s + 1)}$$

$$a_s = n(s0|x^N) = n(0|x^N_{|s})$$
$$b_s = n(s1|x^N) = n(1|x^N_{|s})$$

Let $\mathcal{S} = \{00, 10, 1\}$. Assume we observe $x^7 = 0\,1\,0\,0\,1\,1\,0$ with an initial past ς that ends with $\cdots\,1\,1\,0$.

	ς			x^7						
	$\cdots$	1	1	0	0	1	0	0	1	1	0
			$\cdots 00$			x			x		
past			$\cdots 10$	x				x			
			$\cdots 1$				x			x	x

Now sorting the symbols of x^7 according to the relevant past symbols we get

$$\begin{aligned} P_e(0100110|\cdots 110) &= P_e(0|0,0)P_e(1|0,0)P_e(0|0,0)P_e(0|1,0)P_e(1|0,1)P_e(1|1,0)P_e(1|1,1) \\ &= \underbrace{P_e(00)}_{10}\underbrace{P_e(11)}_{00}\underbrace{P_e(010)}_{1} \\ &= \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{1}{2}\cdot\frac{3}{4}\cdot\frac{1}{2}\cdot\frac{1}{4}\cdot\frac{3}{6} = \frac{9}{1024} \end{aligned}$$

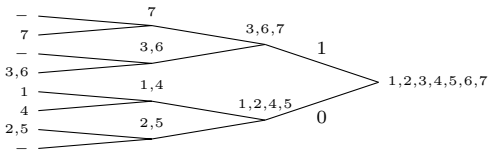
See "Bayesian Estimation"

$$\log_2 \frac{p(x^N|\mathcal{S},\theta)}{\prod_{s\in\mathcal{S}} P_e(a_s,b_s)} \le \frac{|\mathcal{S}|}{2}\log_2\frac{N}{|\mathcal{S}|} + |\mathcal{S}|$$

Problem: We do not know $\mathcal{S}$!

Context tree (of depth D)

	ς			x^7						
$\cdots$	1	1	0	0	1	0	0	1	1	0
index				1	2	3	4	5	6	7



In every node of the tree we write the indices of the symbols that have the corresponding past symbols. In every node s we keep track of the counts $a_s = n(s0|x^N)$ and $b_s = n(s1|x^N)$.
We shall assign a probability to the subsequence $x^N_{|s}$ for every state s in the context tree.

We shall do this in such a way that in the root of the tree we assign a probability to the whole sequence x^N that is a mixture of all possible tree sources.

We use the following observations to build, recursively, this probability.

The probability we build is written as follows

$$P_w^s = P_w(x^N_{|s}),$$

where P_w^s is the shorthand notation we shall use.

Later we return to this and make the notation more precise.
Suppose s is a leaf: All we know are a_s and b_s so we assign the subsequence probability

$$P_w^s = P_e(a_s, b_s).$$

Now if s is an internal node, we have two options for the subsequence probability.

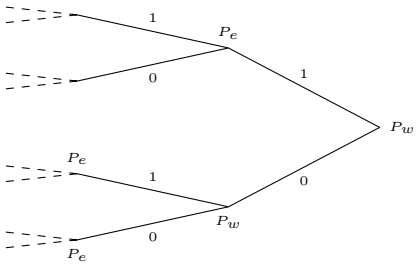
1: $P_e(a_s, b_s)$. 2: $P_w^{0s}P_w^{1s}$.

We must make a choice or better even, mix these options. So we set

$$P_w^s = \frac{P_e(a_s,b_s) + P_w^{0s}P_w^{1s}}{2}.$$

Analysis.

Let $\mathcal{S} = \{00, 10, 1\}$ and we use a context tree with depth $D > 2$. We look at the P_w's for different nodes. For the nodes $s \in \mathcal{S}$ we consider (in the analysis) only the P_e's.



$$P_w^{00} \geq \frac{1}{2} P_e(a_{00}, b_{00})$$
$$P_w^{10} \geq \frac{1}{2} P_e(a_{10}, b_{10})$$
$$P_w^{1} \geq \frac{1}{2} P_e(a_{1}, b_{1})$$

Now we consider nodes nearer to the root and take only the $P_w^{0s} P_w^{1s}$ part.

$$\begin{aligned} P_w^0 &\geq \frac{1}{2} P_w^{00} P_w^{10} \\ &\geq \frac{1}{8} P_e(a_{00}, b_{00}) P_e(a_{10}, b_{10}) \\ P_w^\lambda &\geq \frac{1}{2} P_w^0 P_w^1 \\ &\geq \frac{1}{32} P_e(a_{00}, b_{00}) P_e(a_{10}, b_{10}) P_e(a_1, b_1) \end{aligned}$$

Here λ denotes the root of the tree.
For general trees (or suffix sets) $\mathcal{S}$ we find

$$P_w^\lambda \geq 2^{1-2|\mathcal{S}|} \prod_{s \in \mathcal{S}} P_e(a_s, b_s)$$

So

$$\log_2 P_w^\lambda \geq \log_2 p(x^N | \mathcal{S}, \theta) - \left(2|\mathcal{S}| - 1 + \frac{|\mathcal{S}|}{2} \log_2 N + |\mathcal{S}| \right).$$

Real sequence probability - Cost of describing the tree - Cost of the parameters ~~Contributions to the weighted probability are:~~ Real sequence probability; Cost of describing the tree; Cost of the parameters

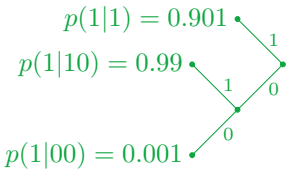
This algorithm achieves the (asymptotically) optimal log-likelihood ratio (not only on the average but also individually for every data sequence).

$$\log \frac{p(x^N | \mathcal{S}, \theta)}{P_w^\lambda} \leq 2|\mathcal{S}| - 1 + \frac{|\mathcal{S}|}{2} \log_2 N + |\mathcal{S}|.$$

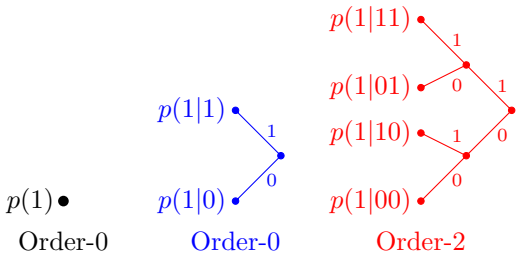
Another essential property of the "Context-Tree Weighting" (CTW) algorithm is its efficient implementation. The number of trees squares with every increment of D and yet the amount of work is at most linear in $D \cdot N$.
We can even write a stronger result when we realise that the method has no knowledge of a "real model".
Let $\mathbf{S}_D$ be the set of all tree models with a maximal depth of atmost D.

$$\log P_w^\lambda \geq \max_{\mathcal{S} \in \mathbf{S}_D} \left\{ \log p(x^N | \mathcal{S}, \theta) - \left(2|\mathcal{S}| - 1 + \frac{|\mathcal{S}|}{2} \log_2 N + |\mathcal{S}| \right) \right\}.$$

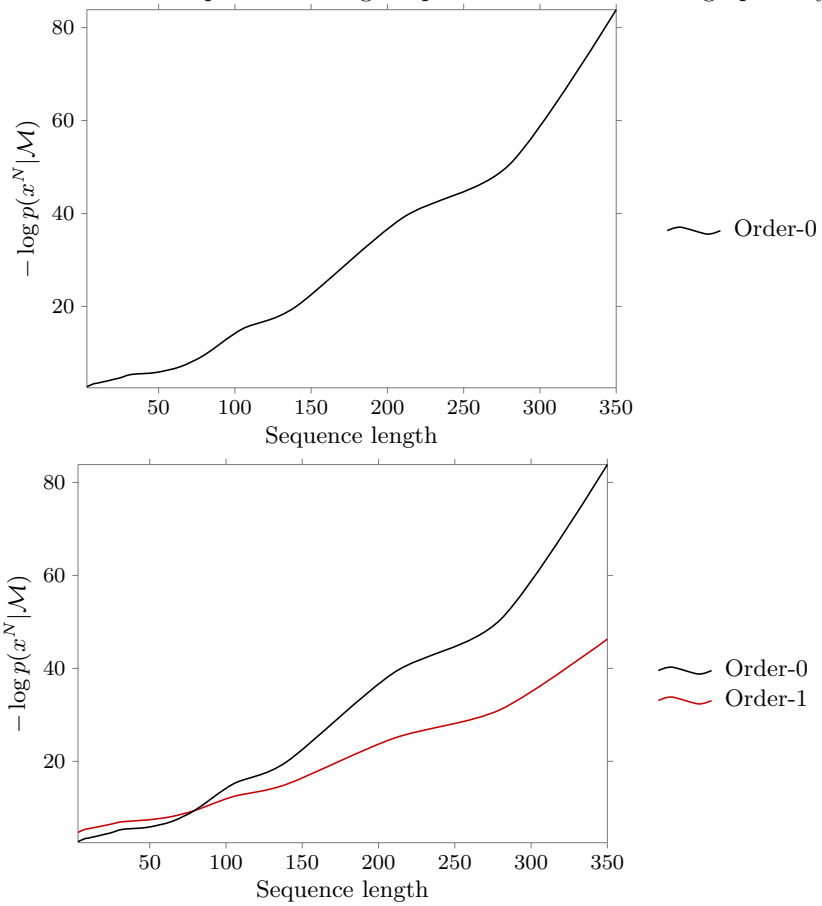
This algorithm is an instantiation of the MDL principle. It finds (in the class $\mathfrak{S}_D$) the model $\mathcal{S}$ that maximizes the sequence probability.
Example: Consider the following actual model.

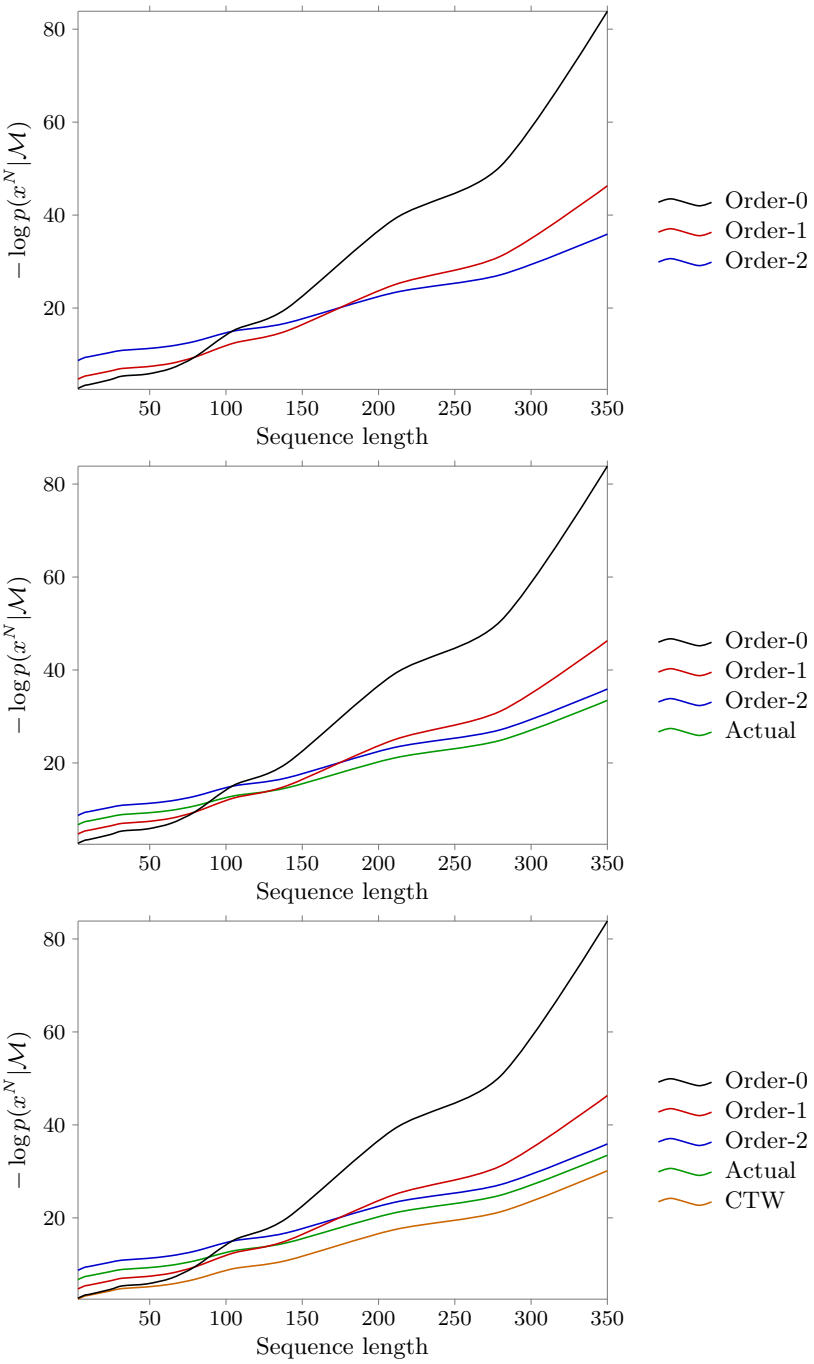


We shall use the following models.



The results for sequences of length upto $N = 350$ are shown graphically.





We see that initially the memoryless (order-0) model performs even better than the actual model.

After about 80 symbols the order-1 model becomes better than both the order-0 and the actual model.

From 120 symbols on the actual model is better than the simpeler models.

The order-2 model is always worse than the actual model. It describes the same probabilities but has too many parameters.

But the CTW model outperforms all models over the whole range of sequence lengths!
We shall now derive an expression, based on the previous method, for the a-posteriori model probability. We consider only binary data but the approach also works for arbitrary alphabets.

First we repeat our notation.

A model is described by a complete suffix set $\mathcal{S}$.

The suffix set can be seen as the set of leaves of a binary tree. Our model class is the set of all complete binary trees whose depth is not more than D, for a given D. We write $\mathfrak{S}_D$ for the model class. So we say that $\mathcal{S} \in \mathfrak{S}_D$.
The depth of a tree is the length of the longest path from the root to a leaf.
Every model $\mathcal{S}$ has a set of parameters θ_s, one for every state $s \in \mathcal{S}$ of the model. θ_s gives the probability of a 1 given that the previous symbols were s.

$$\theta_s = \Pr\{X_t = 1 | X_{t-\ell}^{t-1} = s\}, \text{ where } \ell = |s|$$

The probability of a sequence, given a model $\mathcal{S}$ with parameters θ_s, $s \in \mathcal{S}$ is

$$p(x^N|\mathcal{S}, \theta) = \prod_{s \in \mathcal{S}} p(x_{|s}^N|\theta_s)$$

and

$$p(x_{|s}^N|\theta_s) = (1-\theta_s)^{n(0|x_{|s}^N)} \theta_s^{n(1|x_{|s}^N)}$$

Note (again) that $n(0|x_{|s}^N) = n(s0|x^N)$.
Actually, we silently assume that the first few symbols also have a "context". So we assume that there are some symbols preceding x^N.
We must define some prior distributions. First the prior on the parameters.
We use the beta distribution. (In a non-binary case this generalizes to the Dirichlet distribution.) As done before we select the parameters in the beta distribution to be $\frac{1}{2}$.
So given a model $\mathcal{S}$ then for every $s \in \mathcal{S}$ we take

$$p(\theta_s|\mathcal{S}) = \frac{1}{\pi} \theta_s^{-\frac{1}{2}} (1-\theta_s)^{-\frac{1}{2}}$$

This results in the following sequence probability, first assuming one state s only

$$\begin{aligned} p(x_{|s}^N) &= \int_0^1 p(\theta_s|\mathcal{S}) \theta_s^{n(s1|x^N)} (1-\theta_s)^{n(s0|x^N)} \, d\theta_s \\ &= \frac{\Gamma(n(s0|x^N) + \frac{1}{2}) \Gamma(n(s1|x^N) + \frac{1}{2})}{\pi \Gamma(n(s|x^N) + 1)} \end{aligned}$$

Now for any tree model $\mathcal{S}$ we find

$$\begin{aligned} p(x^N|\mathcal{S}) &= \prod_{s \in \mathcal{S}} \int_0^1 p(\theta_s|\mathcal{S}) p(x_{|s}^N|\theta_s) \, d\theta_s \\ &= \prod_{s \in \mathcal{S}} \frac{\Gamma(n(s0|x^N) + \frac{1}{2}) \Gamma(n(s1|x^N) + \frac{1}{2})}{\pi \Gamma(n(s|x^N) + 1)} \end{aligned}$$

Next we need a prior on the tree models $\mathcal{S}$ in the set $\mathfrak{S}_D$. We wish to use the efficient CTW method of weighting so we choose the corresponding prior.

First define

$$\Delta_D(S) \stackrel{\Delta}{=} 2|\mathcal{S}| - 1 - |\{s \in \mathcal{S} : |s| = D\}|.$$

Then we take the prior

$$p(\mathcal{S}) = 2^{-\Delta_D(\mathcal{S})}$$

We prove that this is a proper prior probability.
Obviously, $p(\mathcal{S}) > 0$ for all $\mathcal{S} \in \mathfrak{S}_D$. We must show that it sums up to one.

We give a proof by induction.

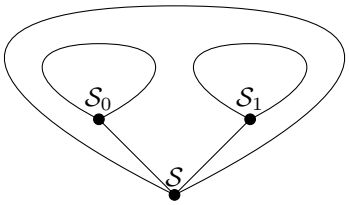
Step 1: $D = 0$: $\mathfrak{S}_0 = \{\lambda\}$, the memoryless source.

$$\Delta_0(\lambda) = 2 \cdot 1 - 1 - 1$$

Where the last -1 comes from the fact that the single state of λ is at level $D = 0$ so $p(\lambda) = 1$.
Induction: Assume it holds for $D \leq D^*$. Now if $\mathcal{S} \in \mathfrak{S}_{D^*+1}$ then

- $\mathcal{S} = \lambda$, i.e. root node only.
- $\mathcal{S}$ contains two trees on level 1, say $\mathcal{S}_0 \in \mathfrak{S}_{D^*}$ and $\mathcal{S}_1 \in \mathfrak{S}_{D^*}$. We have

$$\Delta_{D^*+1}(\mathcal{S}) = 1 + \Delta_{D^*}(\mathcal{S}_0) + \Delta_{D^*}(\mathcal{S}_1)$$



- We repeat: $\mathcal{S}$ contains two trees on level 1, say $\mathcal{S}_0 \in \mathfrak{S}_{D^*}$ and $\mathcal{S}_1 \in \mathfrak{S}_{D^*}$. We have

$$\begin{aligned}
&\Delta_{D^*+1}(\mathcal{S}) = 1 + \Delta_{D^*}(\mathcal{S}_0) + \Delta_{D^*}(\mathcal{S}_1) \\
&\sum_{\mathcal{S} \in \mathfrak{S}_{D^*+1}} 2^{-\Delta_{D^*+1}(\mathcal{S})} = 2^{-1} + \\
&\qquad \sum_{\mathcal{S}_0 \in \mathfrak{S}_{D^*}} \sum_{\mathcal{S}_1 \in \mathfrak{S}_{D^*}} 2^{-1-\Delta_{D^*}(\mathcal{S}_0)-\Delta_{D^*}(\mathcal{S}_1)} \\
&\qquad = 2^{-1} + 2^{-1} \underbrace{\sum_{\mathcal{S}_0 \in \mathfrak{S}_{D^*}} 2^{-\Delta_{D^*}(\mathcal{S}_0)}}_{=1} \underbrace{\sum_{\mathcal{S}_1 \in \mathfrak{S}_{D^*}} 2^{-\Delta_{D^*}(\mathcal{S}_1)}}_{=1} \\
&\qquad = 2^{-1} + 2^{-1} = 1
\end{aligned}$$

We now show that the weighted sequence probability

$$p(x^N) = \sum_{\mathcal{S} \in \mathfrak{S}_D} p(\mathcal{S}) p(x^N|\mathcal{S}),$$

is produced by the weighting procedure of CTW, so

$$p(x^N) = P_w^\lambda.$$

We shall prove this using (mathematical) induction.

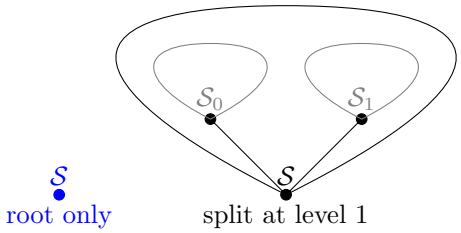
First assume $D = 0$: $\mathfrak{S}_0 = \{\lambda\}$, so the only tree in the set consists of a root only. Therefor $\Delta_0(\lambda) = 0$. So,

$$\begin{aligned} p(x^N) &= p(\lambda)p(x^N|\lambda) \\ &= 2^0 P_e(n(0|x^N), n(1|x^N)) \\ &= P_w^\lambda, \end{aligned}$$

because λ is also a leaf and in a leaf $P_w = P_e$.
Now assume that for all $D \leq D^*$

$$\sum_{\mathcal{S} \in \mathfrak{S}_D} p(\mathcal{S})p(x^N|\mathcal{S}) = P_w^\lambda$$

The tree $\mathcal{S}$ is either the root only or it consists of a root plus two trees, $\mathcal{S}_0$ and $\mathcal{S}_1$, on level one.



$$\begin{aligned} \sum_{\mathcal{S} \in \mathfrak{S}_{D^*+1}} & p(\mathcal{S})p(x^N|\mathcal{S}) = \\ &= 2^{-1} P_e(n(0|x^N), n(1|x^N)) + \\ &\quad \sum_{\mathcal{S} \in \mathfrak{S}_{D^*+1}: \mathcal{S} \neq \lambda} p(\mathcal{S})p(x^N|\mathcal{S}) \end{aligned}$$

$$\begin{aligned} \sum_{\mathcal{S} \in \mathfrak{S}_{D^*+1}: \mathcal{S} \neq \lambda} & p(\mathcal{S})p(x^N|\mathcal{S}) = \\ &\sum_{\mathcal{S}_0 \in \mathfrak{S}_{D^*}} \sum_{\mathcal{S}_1 \in \mathfrak{S}_{D^*}} \frac{1}{2} 2^{-\Delta_{D^*}(\mathcal{S}_0)} 2^{-\Delta_{D^*}(\mathcal{S}_1)} \times \\ &\qquad p(x^N_{|0}|\mathcal{S}_0)p(x^N_{|1}|\mathcal{S}_1) \\ = \frac{1}{2} &\sum_{\mathcal{S}_0 \in \mathfrak{S}_{D^*}} 2^{-\Delta_{D^*}(\mathcal{S}_0)} p(x^N_{|0}|\mathcal{S}_0) \times \\ &\sum_{\mathcal{S}_1 \in \mathfrak{S}_{D^*}} 2^{-\Delta_{D^*}(\mathcal{S}_1)} p(x^N_{|1}|\mathcal{S}_1) \\ = \frac{1}{2} &P_w^0 P_w^1 \end{aligned}$$

And so we find

$$\begin{aligned} \sum_{\mathcal{S} \in \mathfrak{S}_{D^*+1}} & p(\mathcal{S})p(x^N|\mathcal{S}) = \\ &= \frac{1}{2} P_e(n(0|x^N), n(1|x^N)) + \frac{1}{2} P_w^0 P_w^1 \\ &= P_w^\lambda \end{aligned}$$

Thus we can compute the a-posteriori model probability.

$$p(\mathcal{S}|x^N) = \frac{p(\mathcal{S})p(x^N|\mathcal{S})}{p(x^N)}$$

$$p(\mathcal{S}|x^N) = \frac{p(\mathcal{S})p(x^N|\mathcal{S})}{p(x^N)}$$

$$p(\mathcal{S}|x^N) = \frac{2^{-\Delta_D(\mathcal{S})}p(x^N|\mathcal{S})}{p(x^N)}$$

$$p(\mathcal{S}|x^N) = \frac{2^{-\Delta_D(\mathcal{S})}\prod_{s\in\mathcal{S}} P_e(n(s0|x^N), n(s1|x^N))}{p(x^N)}$$

$$p(\mathcal{S}|x^N) = \frac{2^{-\Delta_D(\mathcal{S})}\prod_{s\in\mathcal{S}} P_e(n(s0|x^N), n(s1|x^N))}{P_w^\lambda}$$

$$p(\mathcal{S}|x^N) = \frac{2^{-\Delta_D(\mathcal{S})}\prod_{s\in\mathcal{S}} P_e(n(s0|x^N), n(s1|x^N))}{P_w^\lambda}$$

So, we can use the same computations as in the CTW.

An efficient way to find the Bayesian MAP model exists, but its discussion is not a part of this course.

Part 2-C

Descriptive complexity

How difficult is it to describe this sequence?
0101010101010101010101010101010101010101010101010101 25 repetitions of "01", simple

1011010100000100111100110011001111111100111011111001 The first 50 fractional bits of $1/\sqrt{2}$, simple

0011011100100100010011111100001011001000101100110 50 coin tosses, complex

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

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.

Example 4 (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.

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.

Example 5.

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:	$l_C(x^N) = l(c_1 \dots c_j) = j$

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

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

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.

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

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: $r_N = R_N - h(\theta)$
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 $\theta = \theta_2$.

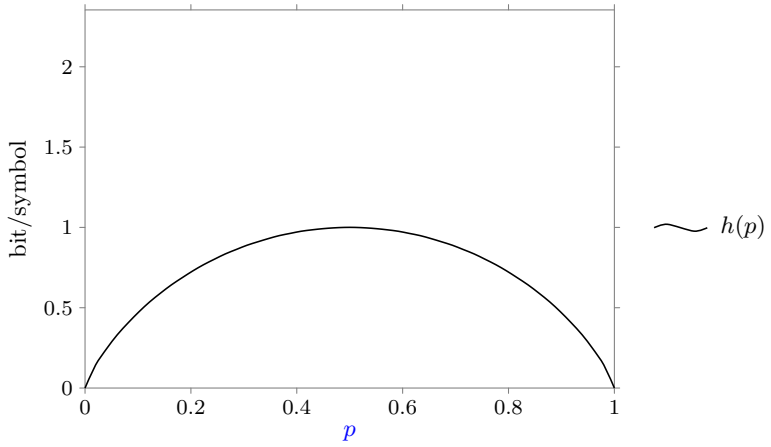
We also create the code C_{12} which uses the smallest code word from C_1 and C_2 with a '0' or '1' prepended to indicate from which code the word comes.

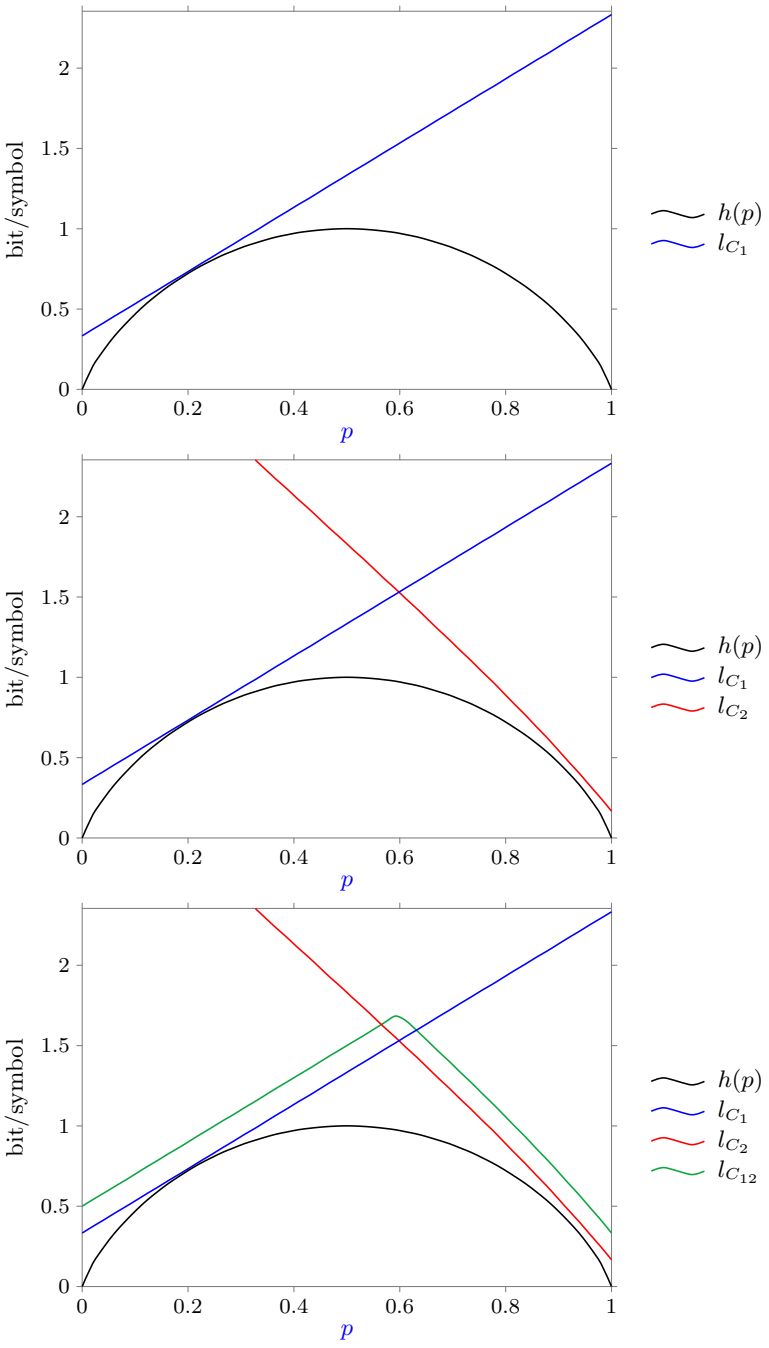
In all cases the code words are created using the ideal code wordlengths $l_C^*(x^N)$.

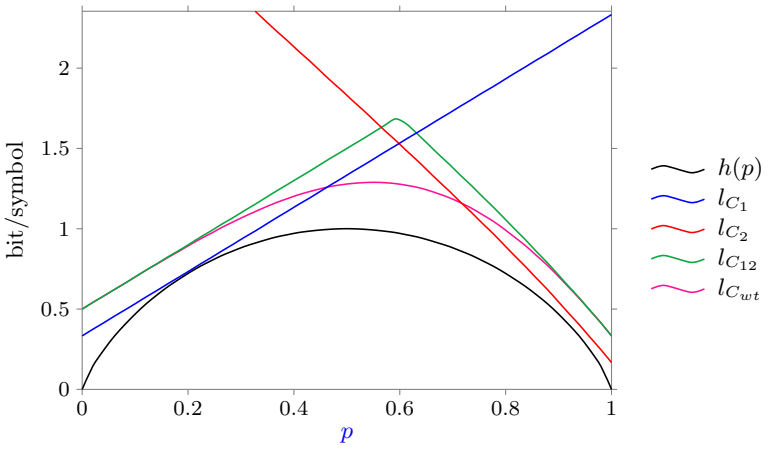
The code C_{mix} is make using the mixed (weighted) probabilities

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

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




bit/symbol
2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1
p
h(p)
l_{C_1}
bit/symbol
2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1
p
h(p)
l_{C_1}
l_{C_2}
bit/symbol
2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1
p
h(p)
l_{C_1}
l_{C_2}
l_{C_{12}}



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.

Theorem 6. *[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.
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.
In the Bayesian Model estimation problem we looked at the log-regret criterion:

$$\log \frac{p(x^N|\mathcal{M}_i, \theta_i)}{p(x^N|\mathcal{M}_i)},$$

regret from not knowing the parameters.

or the criterion

$$\log \frac{p(x^N|\mathcal{M}_i, \theta_i)}{p(x^N)},$$

regret from not knowing the model plus parameters.

Remember

$$p(x^N|\mathcal{M}_i) = \int_{\Theta_i} p(\theta_i|\mathcal{M}_i) p(x^N|\mathcal{M}_i, \theta_i)\, d\theta_i$$

and

$$p(x^N) = \int_{\mathfrak{M}} p(\mathcal{M}) p(x^N|\mathcal{M})\, d\mathcal{M},$$

or more often when the model class is discrete

$$p(x^N) = \sum_{\mathcal{M} \in\, \mathfrak{M}} p(\mathcal{M}) p(x^N|\mathcal{M}).$$

If $\mathcal{M}_i$, θ_i has actually generated x^N then

$$-\log p(x^N|\mathcal{M}_i, \theta_i)$$

is the ideal codeword length.

And

$$-\log p(x^N|\mathcal{M}_i) \text{ resp. } -\log p(x^N)$$

is the actual codeword length of a good code using these 'estimated' probabilities.
Thus

$$\log \frac{p(x^N|\mathcal{M}_i, \theta_i)}{p(x^N|\mathcal{M}_i)} \text{ resp. } \log \frac{p(x^N|\mathcal{M}_i, \theta_i)}{p(x^N)}$$

can be seen as

Data compression: The excess codeword length (individual redundancy).

Machine learning: The individual log-regret.

So now we see that the expected redundancy of a code C on sequences x^N from a source $\mathcal{M}_i$, θ_i, given by

$$r_N(C) = \sum_{x^N \in \mathcal{X}^N} p(x^N|\mathcal{M}_i, \theta_i) \log \frac{p(x^N|\mathcal{M}_i, \theta_i)}{p(x^N|\mathcal{M}_i)}$$

resp.

$$r_N(C) = \sum_{x^N \in \mathcal{X}^N} p(x^N|\mathcal{M}_i, \theta_i) \log \frac{p(x^N|\mathcal{M}_i, \theta_i)}{p(x^N)}$$

can also be seen as the expected log-regret with respect to $\mathcal{M}_i$ and θ_i.
We again take a Bayesian approach.

But we are also dealing with variable-length codes C. We first discuss how we can relate codeword lengths $l_C(x^N)$ to probabilities $p(x^N|\mathcal{M},\theta)$.

The answer is through the ideal codeword length

$$l_C(x^N) \sim -\log_2 p(x^N|\mathcal{M},\theta).$$

Let C be any binary, prefix-free code with K words, where l_i denotes the length of the i^{th} code word, that satisfies the Kraft inequality with equality, i.e.

$$\sum_{i=1}^{K} 2^{-l_i} = 1.$$

We see that 2^{-l_i} plays the role of a probability value, namely

$$\begin{aligned} 2^{-l_i} &> 0, && \text{because } 0 < l_i < \infty, \\ \sum_{i=1}^{K} 2^{-l_i} &= 1, && \text{total probability is 1.} \end{aligned}$$

A probability vector $Q = \{q_1, q_2, \ldots, q_K\}$ is called a dyadic probability vector, if for all i, $1 \le i \le K$, there exists integers n_i such that

$$q_i = 2^{-n_i}.$$

An example
$Q = \left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{32}\right\}$ is dyadic, namely

$$\begin{aligned} q_1 = q_2 = q_3 &= 2^{-2}, & q_4 &= 2^{-3}, \\ q_5 &= 2^{-4}, & q_6 = q_7 &= 2^{-5}, \\ & & \sum_{i=1}^{7} q_i &= 1. \end{aligned}$$

The code that corresponds to this probability vector has codeword lengths

$$\begin{aligned} l_1 = l_2 = l_3 &= 2, & l_4 &= 3 \\ l_5 &= 4 & l_6 = l_7 &= 5. \end{aligned}$$

An example of such a code can have the following 7 words

$$\begin{aligned} c_1 &= 00 & c_2 &= 01 \\ c_3 &= 10 & c_4 &= 110 \\ c_5 &= 1110 & c_6 &= 11110 \\ c_7 &= 11111 \end{aligned}$$

If C is a code that satisfies the Kraft inequality with equality, then we denote the corresponding, unique dyadic probability vector by Q_C.

The set of all dyadic probability vectors of the same length K will be denoted by $\mathcal{Q}_\mathcal{C}$, where the vector length K is not specified explicitely. We also write $\mathcal{Q}$ for the set of all probability vectors of the same length K.
Obviously any $Q_C \in \mathcal{Q}_\mathcal{C}$ is also a member of $\mathcal{Q}$, so

$$\mathcal{Q}_\mathcal{C} \subset \mathcal{Q}.$$

So again, 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 $\Theta = [0, 1]$.
If $Q_C \in \mathcal{Q}_\mathcal{C}$ then the redundancy of of the corresponding code C is given by

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

Maximizing over the parameter values we get the maximum expected redundancy of a given code C.

$$r_N^+(C) = \sup_{\theta \in \Theta} D(p(X^N|\theta) \| Q_C(X^N))$$

Now we can look for the best possible code that minimizes the maximum expected redundancy.

$$r_N^+ = r_N^+(C^*) = \min_C r_N^+(C).$$

So C^* is the code that minimizes the worst-case expected redundancy over all parameter values. r_N^+ is the resulting minimax expected redundancy.

Instead of the worst-case redundancy we can also consider weighted redundancies.
Let $w(\theta)$ be a prior distribution over θ. The Bayes redundancy is given by

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

Because the maximum is never smaller than the average, we have

$$r_N^+(C) \geq \mathcal{D}(w; Q_C),$$

and likewise we obtain for the best possible code

$$r_N^+ \geq \min_C \mathcal{D}(w; Q_C).$$

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

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

and because we can minimize over a larger set

$$r_N^+ \geq \min_Q \mathcal{D}(w; Q).$$

It turns out that the Q^* that realizes the minimum is the $w(\theta)$ weighted probability

$$Q^* = \int_\Theta p(x^N|\theta)w(\theta)\,d\theta.$$

And thus we can observe:

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

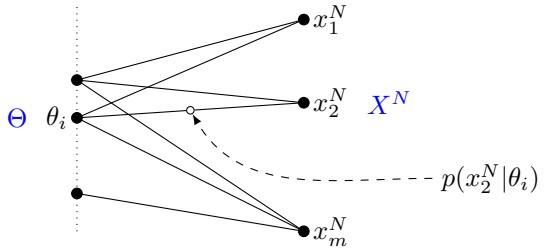
Channel input θ probabilities $w(\theta)$ — Channel transition probabilities $p(x^N|\theta)$ — Channel output x^N probabilities $Q^*(x^N)$

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

Because this last bound is independent of the prior $w(\theta)$, we can tighten the bound by maximizing over all possible priors $w(\theta)$ and find

$$r_N^+ \geq \max_{w(\theta)} I(\theta;X^N) = C_{\theta\to X^N}.$$

So the redundancy is lower bounded by (often 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, because it causes extra codeword length, but it is unavoidable.
- Machine learning: this is what's it about, but we cannot learn faster then the channel capacity.

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

bits of π: Almost zero complexity. The model is easy to describe and can only generate this sequence. So the first part is easy and the second part too because it is easy to compute the 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 so we need a lot of bits to describe what the model did.

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.*

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.

This results in the notion of stochastic complexity

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

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

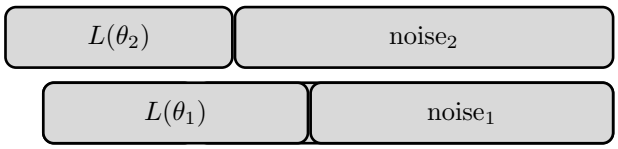
is known as the NML (Normalized Maximum Likelihood). That is must be normalized is reasonable because

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

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

Stochastic complexity ≈ 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:

$L(\text{noise}_1) > L(\text{noise}_2)$ hardly possible because $L(\theta_2) - L(\theta_1)$ cannot be large.
$L(\text{noise}_1) < L(\text{noise}_2)$ very likely.

So, $\mathcal{M}_1$ explains more of the data (less noise)
"Real data model": binary 1$^{\text{th}}$ 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$^{\text{th}}$ 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

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

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.

Where to find information
WebPage: `http://bertdv.github.io/teaching/AIP-5SSB0/`
Prefix for all links below: `https://github.com/bertdv/AIP-5SSB0/blob/master/lessons/`

The lecture notes: `Tjalling/alltext4.pdf?dl=0`
Information Theory: `Tjalling/combined-article.pdf?dl=0`
Markov structures and summary:
`Tjalling/summary-beamer.pdf?dl=0`
Exercises for part 2:
`exercises/5MB20-exercises-part-2.pdf`
Hints for the exercises
`exercises/5MB20-exercises-part-2-hints.pdf`