582650 Information-Theoretic Modeling (Autumn 2012)
Homework 5/Solutions

1. Consider binary sequences $x^{15} = (x_1, x_2, \ldots, x_{15}) \in \{0, 1\}^{15}$ of length $n = 15$. Let $\mathcal{M} = \{p_\theta : \theta \in [0, 1]\}$ be a model class consisting of i.i.d. Bernoulli distributions—hence, the probability of sequence $x^{15}$ is given by $\theta^s (1 - \theta)^{n-s}$, where $s = \sum_{i=1}^{15} x_i$ is the number of ones and $n - s$ the number of zeros in $x^{15}$.

We quantize the parameter space $\Theta = [0, 1]$ by choosing 11 points at even intervals, letting the possible quantized parameters be $\theta^q \in \Theta^q = \{0.0, 0.1, 0.2, \ldots, 1.0\}$.

(a) What is the two-part code-length (ignoring the integer requirement) for data sequence $x^{15} = 001000100000001$? Since we are not using the optimal quantization, we need to evaluate the two-part code-length as

$$\min_{\theta^q \in \Theta^q} \left[ \ell(\theta^q) + \log_2 \frac{1}{p_{\hat{\theta}^q}(D)} \right].$$

Use the uniform code for $\theta^q$ which implies $\ell(\theta^q) = \log_2 11$ for all $\theta^q \in \Theta^q$.

**Solution** Since the maximum likelihood parameter $\hat{\theta}(x^{15}) = \frac{s}{15} = \frac{3}{15} = 0.2$ actually appears in the quantization, it is clear that the minimum is achieved for $\theta^q = \hat{\theta}(x^{15}) = 0.2$.

We have

$$\min_{\theta^q \in \Theta^q} \left[ \ell(\theta^q) + \log_2 \frac{1}{p_{\hat{\theta}^q}(D)} \right] = \log_2 11 + \log_2 \frac{1}{p_{0.2}(D)} = \log_2 11 - 3 \log_2 0.2 - 12 \log_2 0.8 \approx 14.29$$

(b) Compute the mixture code-length,

$$\log_2 \sum_{\theta^q \in \Theta^q} \frac{1}{p_{\theta^q}(x^{15}) w(\theta^q)},$$

with the uniform prior $w(\theta^q) = \frac{1}{11}$ for all $\theta^q \in \Theta^q$.

Compare these code-lengths. **Optional:** Does the order of the code-lengths depend on the actual sequence $x^{15}$?

**Solution** The mixture code-length is

$$- \log_2 \sum_{\theta^q \in \Theta^q} 2^{-\ell(\theta^q) - \log_2 \frac{1}{p_{\theta^q}(x^{15})}} = \log_2 11 - \log_2 \sum_{\theta^q \in \Theta^q} \theta^q^3 (1 - \theta^q)^{12} \approx 12.96 < 14.29$$

Trying for other sequences, with different numbers of ones, we get the same order. That is, the mixture code always yields shorter code-length than the two-part code. This is clear, since for the corresponding (sub-) probabilities we have

$$P_{mix}(x^{15}) = \sum_{\theta^q \in \Theta^q} \frac{1}{11} p_{\theta^q}(x^{15}) > \max_{\theta^q \in \Theta^q} \frac{1}{11} p_{\theta^q}(x^{15}) = P_{2-part}(x^{15}).$$

This holds for all sequences $x^{15}$, take the minus-log to get code-lengths (cf. Homework 4, Bonus Problem).
2. Continuation of the first exercise: Compute the normalized maximum likelihood code-length,
\[
\log_2 \frac{1}{p_\hat{\theta}(x^{15}) / C},
\]
where \( C = \sum_{y^{15} \in \{0,1\}^{15}} p_\theta(y^{15}) \),
where the sum is over all the possible 15 bit sequences. Note that each term \( p_\theta(y^{15}) \) in the sum involves the parameters maximizing the probability of sequence \( y^{15} \). The maximizing \( \theta \) for \( y^{15} \) is given by \( \hat{\theta} = \frac{\sum y_i}{n} \). By these observations, we obtain
\[
p_\theta(y^{15}) = \left( \frac{\sum y_i}{n} \right)^{\sum y_i} \left( 1 - \frac{\sum y_i}{n} \right)^{n - \sum y_i}
\]
Optional: Can you figure out a way to compute the sum faster than by enumerating all the \( 2^{15} \) possible binary sequences?

Solution In order to simplify computation observe that the maximum likelihood only depends on the number of ones \( s \) in the sequence. As we have \( \binom{15}{s} \) different sequences with \( s \) ones we get
\[
C = \sum_{y^{15} \in \{0,1\}^{15}} p_\theta(y^{15}) = \sum_{s=0}^{15} \binom{15}{s} \left( \frac{s}{15} \right)^s \left( \frac{15 - s}{15} \right)^{15-s} \approx 5.55
\]
The NML codelength then is
\[
\log_2 C - \log_2 p_\hat{\theta}(x^{15}) \approx 13.30
\]
Both mixture code and NML code are complete (tight Kraft inequality), such that neither can give shorter codelength for all data. While NML gives shorter code for sequences with maximum likelihood parameter close to either zero or one, the mixture code is shorter inbetween these extremes.

3.–4. Again, this problem counts as two because it requires a bit of work (but the actual amount of work is not as large as you might think based on this lengthy problem description).

You’ll need some tool to fit parametric functions to data. Below the problem is explained assuming you use gnuplot, which is easy to use, freely available and sufficient for our needs here. Feel free to use any other mathematics package (probably more powerful) if you wish. A good tutorial for gnuplot is available at [http://www.duke.edu/~hpgavin/gnuplot.html](http://www.duke.edu/~hpgavin/gnuplot.html). In particular, it explains how to save your plots as PostScript files (other formats are also supported; say help set terminal in gnuplot).

The data you are asked to analyse is given on the course web page in file noisy_50.txt. It contains 50 lines, each containing a data point \( x \ y \). The \( x \) values are in \( \{0,2,4,\ldots,98\} \), and \( y \) values are generated as \( y = f(x) + \eta \) where \( f \) is some unknown function and \( \eta \) is noise from some unknown i.i.d. source. (In an actual application you might not know even this much.) You task is to see how well you can recover the unknown \( f \) from this noisy sample using MDL to choose a model class.

On the course web page there is also the file clean_all.txt, which contains the “true” values \( f(x) \) for \( x = 0.0,0.1,0.2,\ldots,100 \) in the format explained above. Please do not look into this file until you’ve done all your curve fitting and decided upon your best estimate for \( f \). You can then plot your estimate against clean_all.txt to see how successful you were in uncovering the “ground truth.” This will be much more interesting if you don’t spoil yourself by looking at the clean data in advance.
For grading purposes, you can think that Problem 3 is understanding the task, setting yourself up and getting at least some kind of estimate with MDL. Problem 4 would then be to do more experiments, explain what you are doing and evaluate the end result. Your answer should contain an explanation of what you did, a couple of sample plots, and any conclusions you can make. You are not graded on how good fit to “ground truth” you get, as long as your method is sound. (This is a fairly simple artificial data set, so you can get a quite accurate estimate if you make some lucky guesses in choosing your model classes.)

First, let’s take a look at the data (Figure 1):

```
gnuplot> plot 'noisy_50.txt'
```

![Figure 1: noisy_50.txt](image)

Now, let’s fit a quadratic function, i.e., a second order polynomial, to the data using gnuplot’s fit procedure:

```
gnuplot> p2(x)=a+b*x+c*x**2
gnuplot> fit p2(x) 'noisy_50.txt' via a,b,c
gnuplot> plot 'noisy_50.txt', p2(x)
```

The fit command will produce quite a bit of output, of which we need just one line:

```
final sum of squares of residuals : 53.4983
```

This is the residual sum of squares (RSS) we need for computing the MDL.

You can of course fit also higher-order polynomials (which actually is a good way to get started). You are not even restricted to polynomials: you can fit any function that can be written in gnuplot, including exponentials, logarithms, trigonometric functions, etc. Your task is to find a function for which the MDL criterion gives as small a value as possible. Below is a more detailed explanation of how to use (two-part) MDL for this particular problem.

If we want to encode the data using a this kind of a model, we need to encode
Figure 2: A quadratic function \( f(x) = a + bx + cx^2 \) fitted to the data.

(a) the parameters: we use the asymptotic formula \( \frac{k}{2} \log_2 n \) as the code-length for this part,
(b) the data: we use a Gaussian distribution.

In the Gaussian density fitted to the data, the mean is given by the fitted curve and the variance is given by the residual sum of squares divided by the sample size: \( \hat{\sigma}^2 = \text{RSS}/n \). For instance, in the quadratic case above, the variance is given by \( \hat{\sigma}^2 = 53.4983/50 \approx 1.07 \).

The fact that the Gaussian distribution is defined as a density, not a probability mass function, is actually of no concern—this will be explained on Friday’s lecture. The code-length of the second part becomes then

\[
\log_2 \left( \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} e^{\frac{(f(x) - y)^2}{2\hat{\sigma}^2}} \right)^{-1},
\]

where \( f(x) \) is the fitted function. This can be re-written as

\[
\frac{n}{2} \log_2(2\pi\hat{\sigma}^2) + \sum_{i=1}^{n} \frac{(f(x) - y)^2}{(2\ln 2)\hat{\sigma}^2},
\]

where the sum of squared residuals and the ML estimate of the variance \( \hat{\sigma}^2 \) cancel each other, and the second term becomes a constant (see Lecture 10). We can thus write the code-length as

\[
\frac{n}{2} \log_2 \text{RSS} + \text{constant},
\]

where constant doesn’t depend on the data or the function we are fitting, and can be ignored.

The total code-length which gives the final MDL criterion is therefore

\[
\frac{n}{2} \log_2 \text{RSS} + \frac{k}{2} \log_2 n,
\]
where \( k \) is given by the number of parameters in the model plus one for the variance parameter. To give an example, in the case of the quadratic model, the value of the criterion is given by

\[
\frac{50}{2} \log_2 53.4983 + \frac{4}{2} \log_2 50 \approx 154.82.
\]

(There are three parameters \( a, b, c \), so \( k = 3 + 1 = 4 \).) As a comparison, we can fit a simpler linear model:

```plaintext
gnuplot> p1(x)=a+b*x
gnuplot> fit p1(x) 'noisy_50.txt' via a,b
```

This gives an RSS of 57.6132, so the MDL criterion becomes

\[
\frac{50}{2} \log_2 57.6132 + \frac{3}{2} \log_2 50 \approx 154.67 < 154.82.
\]

Thus, in this case MDL gives a slight preference for the simpler linear model compared to the quadratic model with a better accuracy but one more parameter.

**Solution** Of course, you were allowed to fit any function of your choice, but the problem setting did suggest polynomials of varying degree. This is what you get when fitting polynomials of degree 0..10:

<table>
<thead>
<tr>
<th>degree</th>
<th>RSS</th>
<th>( \frac{n}{2} \log_2 \text{RSS} )</th>
<th>( \frac{k}{2} \log_2 n )</th>
<th>MDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>72.923</td>
<td>154.71</td>
<td>5.64</td>
<td>161.35</td>
</tr>
<tr>
<td>1</td>
<td>57.6132</td>
<td>146.21</td>
<td>8.47</td>
<td>154.67</td>
</tr>
<tr>
<td>2</td>
<td>53.4983</td>
<td>143.54</td>
<td>11.29</td>
<td>154.82</td>
</tr>
<tr>
<td>3</td>
<td>48.8888</td>
<td>140.29</td>
<td>14.11</td>
<td>154.40</td>
</tr>
<tr>
<td>4</td>
<td>46.7604</td>
<td>138.68</td>
<td>16.93</td>
<td>155.61</td>
</tr>
<tr>
<td>5</td>
<td>39.8494</td>
<td>132.91</td>
<td>19.75</td>
<td>152.67</td>
</tr>
<tr>
<td>6</td>
<td>33.6704</td>
<td>126.84</td>
<td>22.58</td>
<td>149.41</td>
</tr>
<tr>
<td>7</td>
<td>32.7131</td>
<td>125.79</td>
<td>25.40</td>
<td>151.19</td>
</tr>
<tr>
<td>8</td>
<td>32.1407</td>
<td>125.16</td>
<td>28.22</td>
<td>153.38</td>
</tr>
<tr>
<td>9</td>
<td>30.8205</td>
<td>123.56</td>
<td>31.04</td>
<td>154.69</td>
</tr>
<tr>
<td>10</td>
<td>30.7832</td>
<td>123.60</td>
<td>33.86</td>
<td>157.46</td>
</tr>
</tbody>
</table>

The RSS drops monotonically with growing degree of the fitted polynomial. On the other hand the model complexity grows linearly. The MDL criterion trades off the two and achieves its minimum at degree 6. The noiseless data is also a polynomial of degree 6, and the estimate based on the noisy data gets reasonably close, much closer than the observed noisy data points. In this sense we have denoised the data.
Figure 3: A polynomial of degree 6 (MDL-optimal solution) fitted to the data. Noiseless data for comparison.

**Bonus Problem** Generate your own data from some parametric function $f(x)$, and see if that function is identified correctly by the MDL criterion. Try adding noise and using different sample sizes.

**Solution** Whatever you have tried, points for interesting experiments and results!