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

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

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

(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 $x^{15}$?

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

$$p_{\hat{\theta}}(y^{15}) = \left( \frac{\sum y_i}{n} \right)^{\sum y_i} \left( 1 - \frac{\sum y_i}{n} \right)^{n - \sum y_i}.$$

Optional: Can you figure out a way to compute the sum faster than by enumerating all the $2^{15}$ possible binary sequences?
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 to 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. 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.) Your 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
gnuplot> plot 'noisy_50.txt'
```

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

```gnuplot
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

(a) the parameters: we use the asymptotic formula $k \frac{1}{2} \log_2 n$ 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: $\hat{\sigma}^2 = \frac{RSS}{n}$. For instance, in the quadratic case above, the variance is given by $\hat{\sigma}^2 = \frac{53.4983}{50} \approx 1.07$.

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

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

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

$$\frac{n}{2} \log_2 RSS + \frac{k}{2} \log_2 n,$$

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:

```
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.

5. Go to [https://ilmo.cs.helsinki.fi/kurssit/servlet/Valinta?kieli=en](https://ilmo.cs.helsinki.fi/kurssit/servlet/Valinta?kieli=en) and fill in the course feedback form. You may wish to wait until the exam to do so; you get credit for this problem if you promise to fill in the form no later than one week after the exam.

**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.