Sample solutions to Homework 5, Information-Theoretic Modeling (Fall 2014)

Jussi Määttä

October 9, 2014

Question 1

(a)

First, let's try to figure out how the math really works here[1]. (You may skip to the end of the next page if you're not interested.)

We have a zero-order Markov chain model with a Beta(α, β) prior. Suppose we observe a sequence D consisting of n bits, k of which are 0's. The probability of D under the mixture model is

$$p(D) = \int_0^1 p_\theta(D)\, w(\theta)\, \mathrm{d}\theta$$

where

$$w(\theta) = \frac{\theta^{\alpha-1}\,(1-\theta)^{\beta-1}}{B(\alpha,\beta)}$$

is the probability density function (pdf) of the Beta distribution with parameters α and β. For the zero-order model, we have $p_\theta(D) = \theta^k\,(1-\theta)^{n-k}$,

[1] Again, there are simpler solutions (see Q3/HW4), but this way is easier to understand (at least after you've understood it).

so

$$\begin{aligned}
p(D) &= \int_0^1 \theta^k \, (1-\theta)^{n-k} \, \frac{\theta^{\alpha-1} \, (1-\theta)^{\beta-1}}{B(\alpha,\beta)} \, \mathrm{d}\theta \\
&= \frac{1}{B(\alpha,\beta)} \int_0^1 \theta^{(k+\alpha)-1} \, (1-\theta)^{(n-k+\beta)-1} \, \mathrm{d}\theta \\
&= \frac{B(k+\alpha, n-k+\beta)}{B(\alpha,\beta)} \underbrace{\int_0^1 \frac{\theta^{(k+\alpha)-1} \, (1-\theta)^{(n-k+\beta)-1}}{B(k+\alpha, n-k+\beta)} \, \mathrm{d}\theta}_{=1} \\
&= \frac{B(k+\alpha, n-k+\beta)}{B(\alpha,\beta)}
\end{aligned}$$

(the integral of a pdf is always 1).

Suppose then, for example, that we observe the sequence 0011. The above may be expanded as

$$\begin{aligned}
p(0011) &= \frac{B(2+\alpha, 2+\beta)}{B(\alpha,\beta)} \\
&= \frac{B(1+\alpha,\beta)}{B(\alpha,\beta)} \cdot \frac{B(2+\alpha,\beta)}{B(1+\alpha,\beta)} \cdot \frac{B(2+\alpha,1+\beta)}{B(2+\alpha,\beta)} \cdot \frac{B(2+\alpha,2+\beta)}{B(2+\alpha,1+\beta)}.
\end{aligned}$$

Since the beta function has the nice properties

$$\frac{B(x+1,y)}{B(x,y)} = \frac{x}{x+y}, \qquad \frac{B(x,y+1)}{B(x,y)} = \frac{y}{x+y},$$

we may further write the above as

$$p(0011) = \frac{\alpha}{\alpha+\beta} \cdot \frac{\alpha+1}{\alpha+\beta+1} \cdot \frac{\beta}{\alpha+\beta+2} \cdot \frac{\beta+1}{\alpha+\beta+3}.$$

This corresponds to the predictive distribution

$$\begin{aligned}
p(x_{n+1} = 0 \mid x_1, x_2, \ldots, x_n, \alpha, \beta) &= \frac{\alpha+k}{\alpha+\beta+n}, \\
p(x_{n+1} = 1 \mid x_1, x_2, \ldots, x_n, \alpha, \beta) &= 1 - \frac{\alpha+k}{\alpha+\beta+n} = \frac{\beta+(n-k)}{\alpha+\beta+n}
\end{aligned}$$

which was given in the lecture notes[2].

[2]Note that here k is the number of 0's in the first n digits of the sequence.

Note that we do not need to know the final length of the sequence beforehand. This is different from NML (see Question 2 below).

Now, for the 100-digit sequence given in the problem statement, we may execute the following algorithm:

1. Let $\ell \leftarrow 0$, $n \leftarrow 0$, $k \leftarrow 0$, $\alpha \leftarrow 0.5$, $\beta \leftarrow 0.5$.
2. If there are bits left, let $x \in \{0, 1\}$ be the next one; otherwise, stop.
3. If $x = 0$, let $p \leftarrow (\alpha + k)/(\alpha + \beta + n)$ and let $k \leftarrow k + 1$.
   If $x = 1$, let $p \leftarrow (\beta + n - k)/(\alpha + \beta + n)$.
4. Let $\ell \leftarrow \ell + \log_2(1/p)$ and $n \leftarrow n + 1$.
5. Goto 2.

In the end, ℓ will contain the final code-length. For our particular case, the attached Matlab program `q1_demo.m` computes $\ell \approx 83.16$.

(b)

Now things are getting interesting—we will try a first-order Markov chain. There are two parameters and we take them to be independent with Beta$(0.5, 0.5)$ priors. Moreover, we let the first symbol be uniformly distributed, so encoding it will take a single bit.

As suggested in the problem statement, the problem reduces to two "parallel" zero-order Markov chains. Why? Suppose that we've seen $i - 1$ bits of the sequence. If the last bit we've seen is $X_{i-1} = 0$, then our prediction for the i'th bit will depend only on the parameter $p_0 = p(X_i = 1 \mid X_{i-1} = 0)$—and similarly for $X_{i-1} = 1$. So when predicting the i'th bit, we are always using one of two zero-order models.

What we end up doing is essentially the same that we did in (a), but now we have to maintain the values n and k for two different zero-order models. The following algorithm does the trick:

1. Let $\ell \leftarrow 1$, $n_0 \leftarrow 0$, $k_0 \leftarrow 0$, $n_1 \leftarrow 0$, $k_1 \leftarrow 0$, $\alpha \leftarrow 0.5$, $\beta \leftarrow 0.5$.
2. Let x_{prev} be the first input bit.
3. If there are bits left, let $x \in \{0, 1\}$ be the next one; otherwise, stop.

4. If $x = 0$, let $p \leftarrow (\alpha + k_{x_{\text{prev}}})/(\alpha + \beta + n_{x_{\text{prev}}})$ and let $k \leftarrow k + 1$.
   If $x = 1$, let $p \leftarrow (\beta + n_{x_{\text{prev}}} - k_{x_{\text{prev}}})/(\alpha + \beta + n_{x_{\text{prev}}})$.
5. Let $\ell \leftarrow \ell + \log_2(1/p)$ and $n_{x_{\text{prev}}} \leftarrow n_{x_{\text{prev}}} + 1$.
6. Let $x_{\text{prev}} \leftarrow x$.
7. Goto 3.

The attached Matlab program `q1_demo.m` computes $\ell \approx 86.13$.

Using the result we derived above, we could also computed

$$\ell = -\log_2 \left(\frac{1}{2} \cdot \frac{B(6 + 0.5, 23 - 6 + 0.5)}{B(0.5, 0.5)} \cdot \frac{B(18 + 0.5, 76 - 18 + 0.5)}{B(0.5, 0.5)} \right)$$

which gives the same answer (6 out of 23 digits following a zero are zeros, 18 out of 76 digits following a one are zeros).

(c)

The solution is similar to the previous part except that we now update four different zero-order models. It is straightforward to adapt the above algorithm. The codelength, as computed by `q1_demo.m`, is $\ell \approx 89.90$.

Let's recap: $\ell_{(a)} \approx 83.16$, $\ell_{(b)} \approx 86.13$ and $\ell_{(c)} \approx 89.90$. The zero-order Markov chain model achieves the shortest code-length when we use the mixture code for the parameters of the Markov chains. This suggests that of the three models, the zero-order model is the best description of the data.

(But of course, we cannot be certain; the program `q1_demo.m` also generates a random sequence from a second order Markov chain and computes the code-lengths for the sequence. Most of the time the second order model gets the shortest code-length, but not always. In this case, the longer sequence we generate, the more likely it is that the correct model has the shortest code-length.)

Note carefully that our results above do *not* imply that the data really comes from any zero-order Markov chain—merely that it seems to be a better explanation than the other two we've tried.

Question 2

Since the given 100-digits sequence has 24 zeros, the ML parameter for the zero-order model is $\hat{\theta} = 24/100$. The NML normalizing constant is $C^{(100)} \approx 13.2$. Hence, applying the NML universal code to the zero-order Markov chain gives the code-length

$$\begin{aligned}
\ell &= -\log_2 \left(\frac{p_{\hat{\theta}}(D)}{C} \right) \\
&= -\log_2 \left(\hat{\theta}^{24} \, (1-\hat{\theta})^{100-24} \right) + \log_2 C \\
&= -24 \log_2 \hat{\theta} - (100-24) \log_2 (1-\hat{\theta}) + \log_2 C \\
&\approx 83.23.
\end{aligned}$$

What about the first order model? As we counted above, there are 23 digits that follow a zero, 6 of which are zeros; and there are 76 digits that follow a one, 18 of which are zeros. Hence, the ML parameters are $\hat{\theta}_{(0)} = 6/23$ and $\hat{\theta}_{(1)} = 18/76$, and we get the code-length

$$\begin{aligned}
\ell &= -\log_2 \left(\frac{1}{2} \cdot \frac{(6/23)^6 \, (1-6/23)^{23-6}}{C^{(23)}} \cdot \frac{(18/76)^{18} \, (1-18/76)^{76-18}}{C^{(76)}} \right) \\
&\approx 86.35.
\end{aligned}$$

For the second order model, we need to count the number of zeros following 00, the number of zeros following 01, and so on. All in all, we get the code-length

$$\begin{aligned}
\ell = -\log_2 \Bigg(& \frac{1}{4} \cdot \frac{\left(\frac{1}{6}\right)^1 \left(1-\frac{1}{6}\right)^{6-1}}{C^{(6)}} \cdot \frac{\left(\frac{5}{17}\right)^5 \left(1-\frac{5}{17}\right)^{17-5}}{C^{(17)}} \cdot \frac{\left(\frac{5}{17}\right)^5 \left(1-\frac{5}{17}\right)^{17-5}}{C^{(17)}} \\
& \cdot \frac{\left(\frac{13}{58}\right)^{13} \left(1-\frac{13}{58}\right)^{58-13}}{C^{(58)}} \Bigg) \\
\approx 90.51.
\end{aligned}$$

All these numbers are calculated with the attached Matlab program `q2_demo.m`.

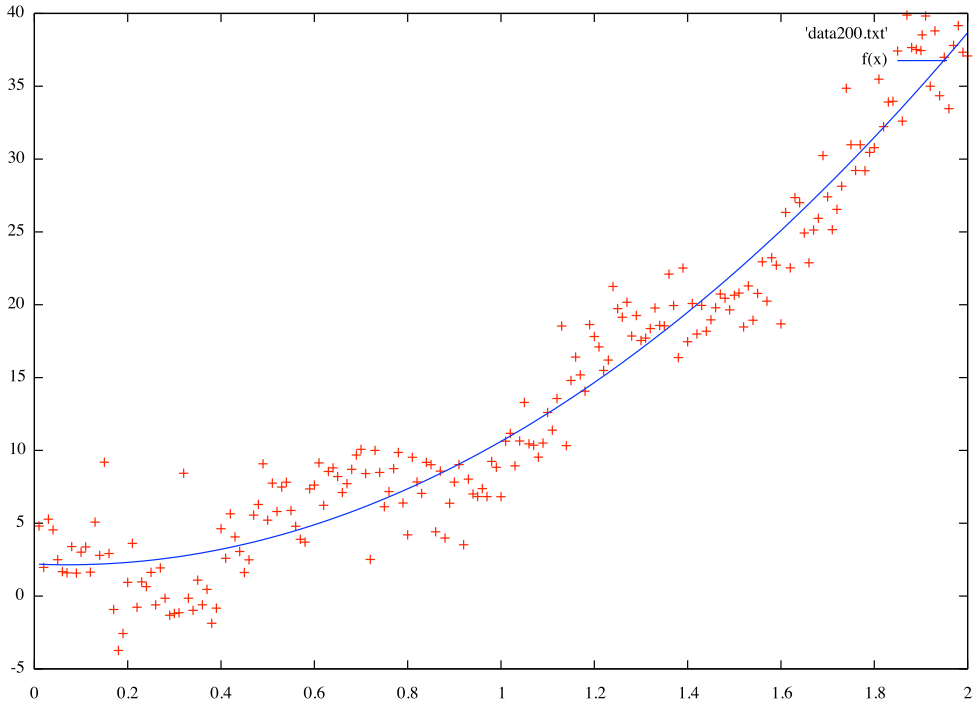

Figure 1: $f_1(x) = a + bx + cx^2$

Questions 3–4

Looking at Figure 2 in the assignment, we see that the linear function $f_0(x) = a + bx$ is a good start. Recall that this function gives an MDL score of about 1170 bits.

The obvious next step is to try a quadratic function: $f_1(x) = a + bx + cx^2$. The fit is shown in Figure 1. Gnuplot gives RSS = 1367.67, so

$$\text{MDL} = \frac{200}{2} \log_2 1367.67 + \frac{4}{2} \log_2 200 \approx 1057.$$

An improvement of more than 100 bits!

A quick test then shows that using a cubic polynomial wouldn't help—the fit looks almost the same as that for the quadratic polynomial.

However, it seems that there is some sort of oscillation in the data that our quadratic polynomial cannot handle. Roughly speaking, our curve is above

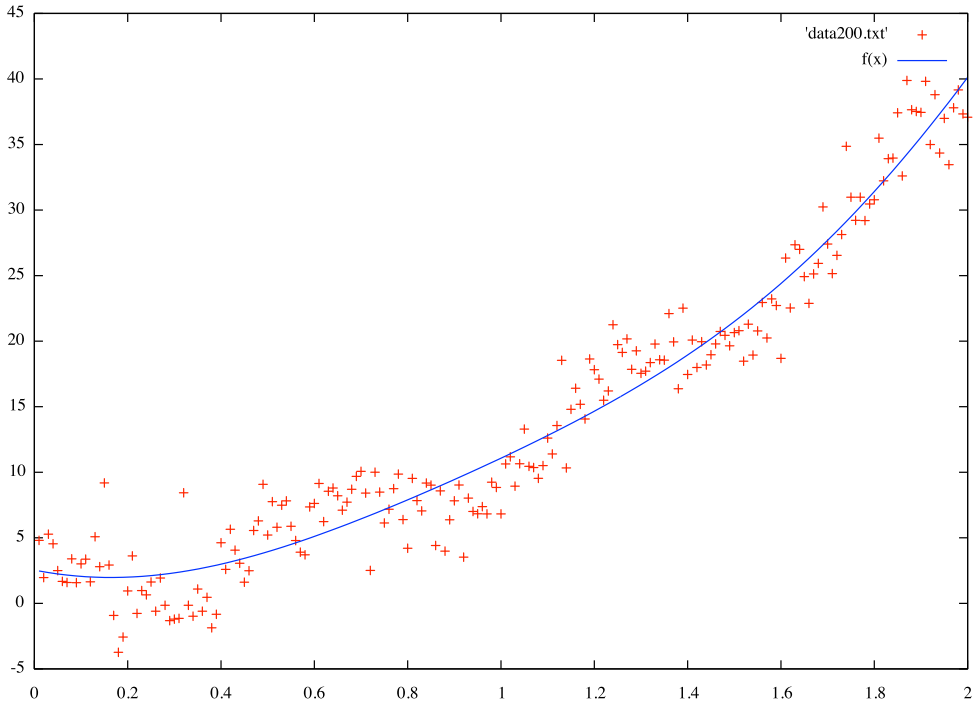

Figure 2: $f_2(x) = a + bx + cx^2 + \alpha \sin(\beta x + \gamma)$

the data points at $x \in [0.2, 0.4]$, below the data points at $x \in [0.4, 0.8]$, above the data at $x \approx 1.2$ and below again at $x \approx 1.5$.

The obvious way to model oscillation is to use trigonometric functions. Let's work on the assumption that the data comes from the sum of a quadratic function and some sort of trigonometric function. We'll try the function $f_2(x) = a + bx + cx^2 + \alpha \sin(\beta x + \gamma)$.

The result is shown in Figure 2. This looks a bit disappointing! The change from the quadratic function isn't that big—neither visually nor in terms of the residues: we have RSS $= 1324.61$ and hence MDL ≈ 1064.

But let's not give up so easily! We're trying to fit a nonlinear function of six parameters to our data. We should help the algorithm by giving sensible initial guesses. Let's look at the residues that arise from using the quadratic function—that is, we take the data points (x, y) and convert them to $(x, y - f_1(x))$ where f_1 is the quadratic polynomial with the fit parameters given by Gnuplot ($a = 2.19824$, $b = -1.39479$, $c = 9.81619$). The resulting

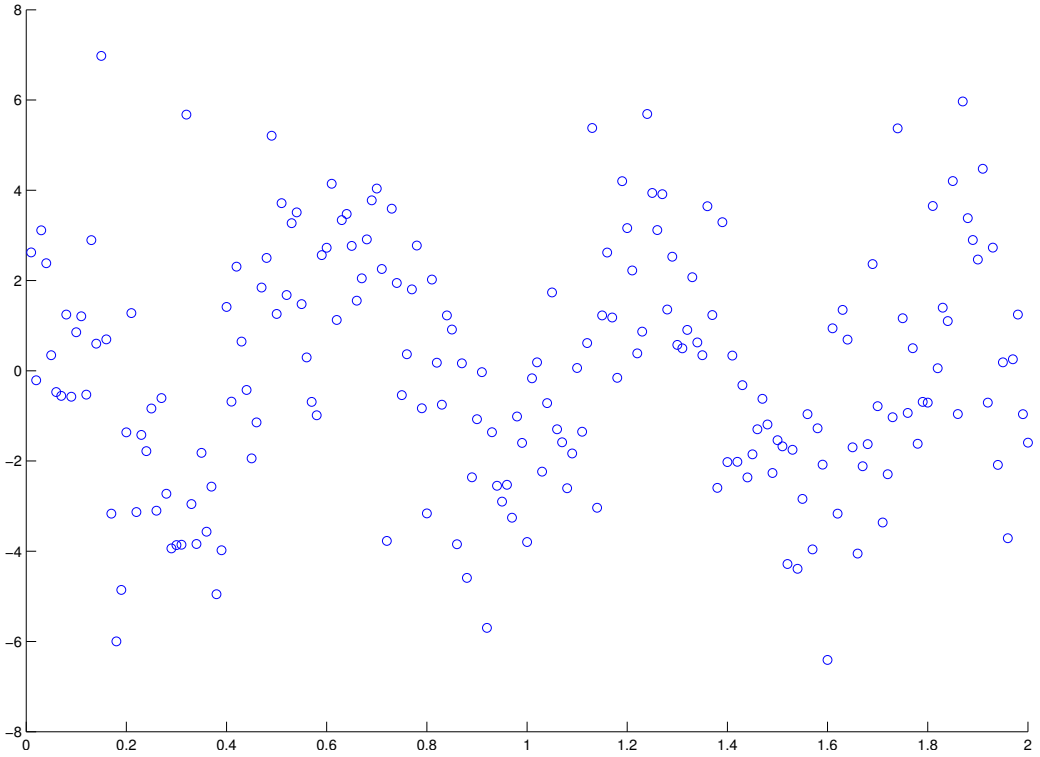

Figure 3: Scatter plot of the points $(x, y - f_1(x))$, where $f_1(\cdot)$ is the quadratic polynomial that gives smallest RSS for the points (x, y).

scatter plot is shown in Figure 3.

By looking at Figure 3, we may estimate the parameters for the term $\alpha \sin(\beta x + \gamma)$. The values $y - f_1(x)$ seem to lie mostly within the interval $[-3, 3]$, so let's guess $\alpha = 3$. And it seems that the x distance between upper (or lower) peaks seems to be about 0.6, so we set $\beta = 2\pi/0.6 \approx 10.5$ to indicate that we would like the term $\alpha \sin(\beta x + \gamma)$ to give the same value for all $x = 0.6 + 2\pi m$, $m \in \mathbb{Z}$. Finally, since it seems that the residue is about zero at around $x \approx 1$, we set $\gamma = -10.5$ so that at $x = 1$, $\alpha \sin(\beta x + \gamma) = 4 \sin(0) = 0$.

We can give this guesses to Gnuplot as follows:

```
gnuplot> alpha = 3
gnuplot> beta = 10.5
gnuplot> gamma = -10.5
```

Then we can try the fitting again:

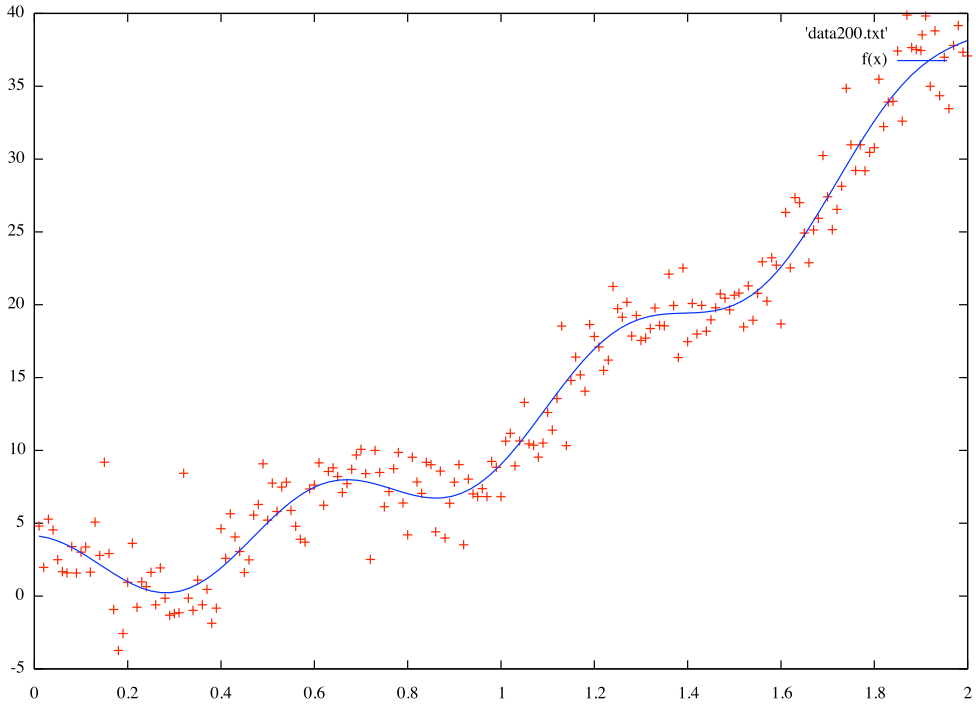

Figure 4: $f_2(x) = a + bx + cx^2 + \alpha \sin(\beta x + \gamma)$, after helping Gnuplot with our guessed values for α, β and γ

```
gnuplot> f(x) = a + b*x + c*x*x + alpha*sin(beta*x+gamma)
gnuplot> fit f(x) 'data200.txt' via a,b,c,alpha,beta,gamma
```

Now we got RSS = 824.858. A huge improvement! With this, the MDL score becomes about 996 bits. The result is shown in Figure 4.

At this point, we decide that further complications to our function would be unlikely to bring much improvement. We could, of course, look at the new residues, but let's not be that ambitious.

Instead, let's look at the values of our six parameters. They are: $a = 1.76219$, $b = 0.341124$, $c = 8.76778$, $\alpha = 2.3906$, $\beta = 10.0323$ and $\gamma = -10.9062$. Do we really need them all?

The constant term a seems necessary, as does the coefficient for the quadratic term. But the value of b seems quite insignificant. Could we do without it? Let's try!

```
gnuplot> f(x) = a + c*x*x + alpha*sin(beta*x+gamma)
gnuplot> fit f(x) 'data200.txt' via a,c,alpha,beta,gamma
```

This gives RSS = 825.278, which is slightly bigger than before (as it must be). But we lost one parameter. This gives MDL ≈ 992. A new record! Apparently it was worth it to throw away the bx term.

The new parameter values are $a = 1.89584$, $c = 8.92553$, $\alpha = 2.38134$, $\beta = 10.0186$ and $\gamma = -10.8867$. Now, recall that $\sin(x + 2\pi) = \sin x$ and $\sin(x+\pi/2) = \cos x$ for all real numbers x. And $\gamma+2\pi+2\pi-\pi/2 \approx 0.09$, which is pretty close to zero. So let's try the function $f_3(x) = a + cx^2 + \alpha \cos \beta x$.

Gnuplot gives RSS = 826.807, again a bit higher than before. But we've lost yet another parameter. The MDL score is about 988 bits, which is better than before.

Our answer, then, is that the function $f_3(x) = a + cx^2 + \alpha \cos \beta x$ is a good match to the data, because it gives the smallest MDL score we can find. (We might, of course, have other ideas for a function to try, but this result is nice enough.) The parameters given by Gnuplot are $a = 1.86379$, $c = 8.93988$, $\alpha = 2.37209$ and $\beta = 10.1006$. The end result is shown in Figure 5.

Is it not impressive how MDL guided us in finding this description of the data? (Sure, we had to use some imagination and help the non-linear least squares fitting algorithm, but still!)

If you find a function that gives a smaller MDL score for this data, please send an email to the lecturer and the teaching assistant!

Note 1. One interesting thing to look at is approximation by trigonometric functions solely. Let

$$f(x) = \sum_{i=1}^{n} a_i \sin(b_i x + c_i) \tag{1}$$

for, say, $n = 4$. Again the nonlinear least squares fitting algorithm may have trouble finding good values, but I was able to use Matlab's interactive fitting tool to find quite nice solutions. (Not as good as the $f_3(x)$ above, though.)

Note 2. Another thing to note is that, of course, *high*-order polynomials can do well. Recall that we may, for instance, use the infinite series representation

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

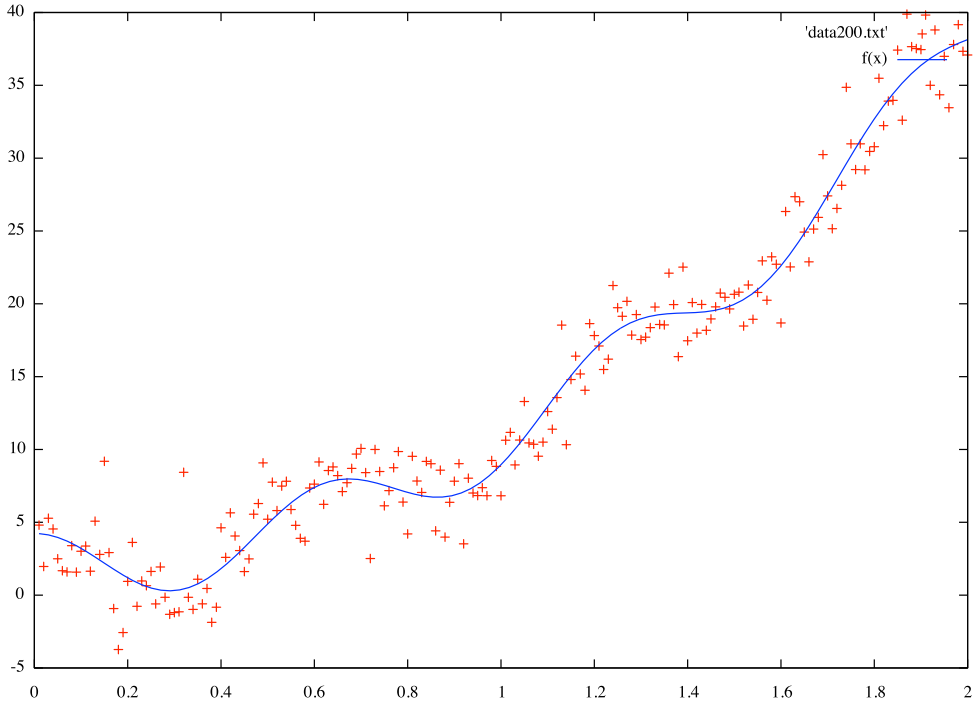

Figure 5: $f_3(x) = a + cx^2 + \alpha \cos \beta x$ (see the text for the values of the parameters)

so it's no wonder that if you take a polynomial of order (say) 32, things may start to look rather nice. (But MDL penalizes the large number of parameters, of course.)

Note 3. We haven't said anything about whether we'd like to predict what happens if $x \notin [0, 2]$. In real life, this would be a crucial question as well (unless you have reason to expect that x never goes outside that range). One way to try to cope with this is to do cross-validation (probably covered e.g. on the course Introduction to Machine Learning).

Note 4. We haven't really specified what sort of functions we'd like to consider for $f(x)$. It was implicitly assumed at least that no numerical constants are allowed (they are treated as parameters to be optimized). Had we been more strict, we could have e.g. specified that we are looking for a function that is built using elementary functions (polynomials, trigonometric functions, e^x, possibly power functions and hyperbolic functions...), has no numerical constants in its definition and fits on a A4 sheet with font size 12. Or, in a more realistic setting, we might e.g. consider all polynomials or all functions of the form (1) above.