582650 Information-Theoretic Modeling (Fall 2014)
Homework 5 (due 9 October)

1. Markov chains. A Markov chain of order m is a model for sequences where each symbol in the sequence, X_i, is conditionally independent of all its other predecessors, $X_{i-1}, X_{i-2}, \ldots$ except $X_{i-1}, \ldots, X_{i-m}$:

$$p(x_i \mid x_{x-i}, x_{i-2}, \ldots) = P(x_i \mid x_{i-1}, \ldots, x_{i-m}) \ .$$

   For example, an zero-order Markov chain is the familiar independence model where each symbol is independent of all the others. We assume that the model is also homogeneous, i.e., that the conditional distributions given by the above expression do not depend on the index i.

   Consider once again the binary alphabet, $x_i \in \{0, 1\}$. The zero-order (independence) model is defined by a single parameter, p, by $p(X_i = 1) = p$. The first order model is defined by two parameters, p_0 and p_1, by $p(X_i = 1 \mid X_{i-1} = x) = p_x$. The second order model would be defined by four parameters, and so on.

   You will probably want to implement the following computations by programming rather than by pencil and paper.

   (a) Consider the sequence

$$1111001001111011111101110011111111111111101101110$$
$$1011111111100010101111100111111101111010110111110$$

   of length 100. Compute the code-length of this sequence under the zero-order model using a mixture code with prior Beta$(0.5, 0.5)$.

   (b) Compute the code-length of the same sequence under the first order model. Use a mixture code where the two parameters are taken to be independent with Beta$(0.5, 0.5)$ priors.

   Note that the problem reduces to applying a zero-order model to the symbols that follow a zero ($01011101\ldots$) and another, independent zero-order model to the symbols that follow a one ($11100111\ldots$). You can let the first symbol, which has no predecessor, be uniformly distributed, $p(X_1 = 1) = 0.5$.

   (c) Apply the same idea to compute the code-length under the second order Markov model where you will just need to break down the sequence into four subsequences, one for symbols following 00, another one for symbols following 01, and so on. Again, use independent zero-order models with Beta$(0.5, 0.5)$ priors to compute the code-lengths of each of these subsequences and let the code-length of the first two symbols be uniform, i.e., two bits.

   Which of the models achieves the shortest code-length? What does this suggest about the sequence?

2. Continuation of the Exercise 1: Apply the NML universal model to the problem in the previous exercise. To obtain the normalizing constant $C = C^{(n)}$ for (sub)sequences of lenght n, use the formula

$$C^{(n)} = \sum_{k=0}^{n} \binom{n}{k} \left(\frac{k}{n}\right)^k \left(\frac{n-k}{n}\right)^{n-k}$$

   rather than summing over all sequences of length n.

3.–4. 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`. Feel free to use any other tools such as `Matlab`, `python`, or `R`[1]

The data you are asked to analyse is given on the course web page in file `data200.txt`. It contains 200 lines, each containing a data point `x y`. The x values are in $[0.01, 2.0]$, and the y values are generated as $y = f^*(x) + \epsilon$ where f^* is some unknown function and ϵ is noise with a zero-mean Gaussian distribution with unknown variance $\sigma^2 > 0$. (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.

We haven't discussed encoding continuous valued data so far but in practice you don't really need to care about it: we will simply evaluate the density value, $q(x)$, and use $-\log_2 q(x)$ as the code-length.[2]

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.

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

```
gnuplot> plot 'data200.txt'
```

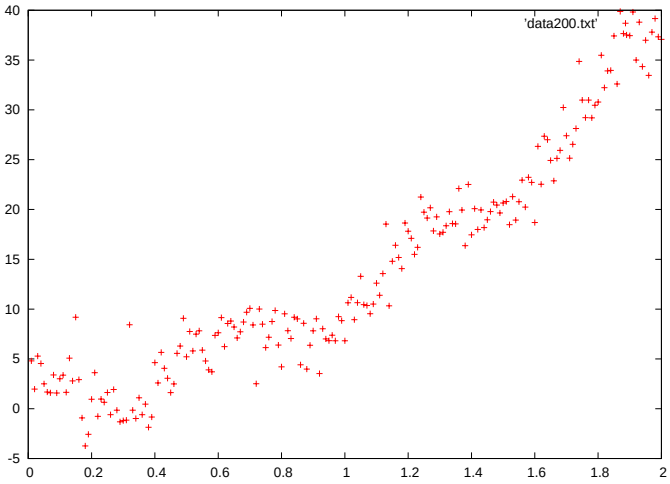

Figure 1: scatter plot of `data200.txt`

Now, let's fit some functions to the data using `gnuplot`'s `fit` (non-linear least-squares) procedure:

```
gnuplot> f(x) = a + b*x
```

[1] Good old `gnuplot`, which many of you probably have never heard of, is actually very handy for this purpose. I tried giving instructions in `R` and `python` but both turned out to be much more clumsy than `gnuplot`. For doing basically the same thing in a number of different environments and languages, see `http://www.walkingrandomly.com/?p=5254`.

[2] In theory, this is not a valid code-length as it will not satisfy the Kraft inequality. To make it a valid code-length, we would need to discretize the observations to finite precision. The used precision would naturally affect the code-length. For practical purposes, the crucial thing is that as the precision is increased, the term depending on the precision typically becomes a constant that doesn't depend on the model class and we don't have to care about it when comparing different model classes.

```
gnuplot> fit f(x) 'data200.txt' via a,b
[output omitted]
gnuplot> plot 'data200.txt', f(x) lt 1 lc 3
```

(The line type `lt` and color `lc` were changed just to make the line show better.)

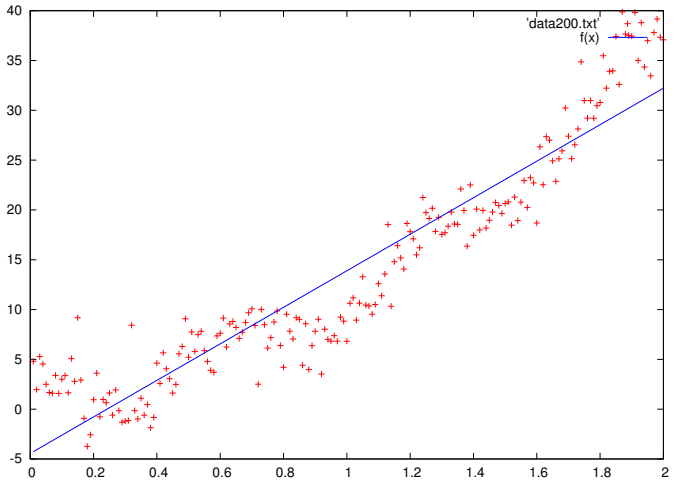

Figure 2: A linear function $f(x) = a + bx$ fitted to the data.

The `fit` command produces some output, of which we need just one line:

```
final sum of squares of residuals : 3080.48
```

This is the residual sum of squares, RSS $= \sum_{i=1}^{n}(y - \hat{f}(x))^2$, where $\hat{f}$ is the fitted function. The RSS values will determine the code-length of the data given the hypothesis (in this case, the linear fit) under the Gaussian error assumption.

Your task is to try to fit different formulae to the data. You can try not only functions that are linear in the parameters — note that any polynomial, e.g., $a + bx + cx^2 + dx^3$ is linear in the parameters (coefficients). You do try, for example, the following non-linear model

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

which produces the following RSS

```
final sum of squares of residuals : 1368.18
```

We use the following simple approximation of a two-part code-length:

(a) for the parameters: $\frac{k}{2}\log_2 n$ bits, and

(b) for the data: a negative logarithm of a Gaussian density fitted to the residuals.

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 linear fit above, the variance is given by $\hat{\sigma}^2 = 3080.48/200 \approx 15.4$.

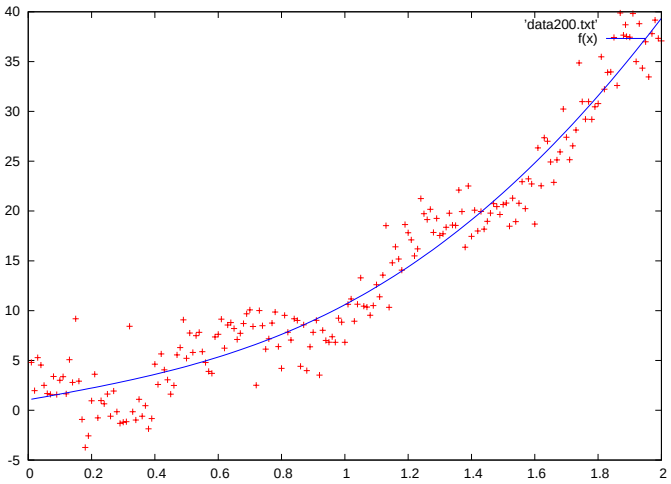

Figure 3: A non-linear function $f(x) = a + bx + c\sin(dx)$ fitted to the data.

The code-length of the second part becomes then

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

Taking care not to get mixed up with the base-2 logarithm and the exponent function logarithm[3], etc, we re-write this as

$$\frac{n}{2}\log_2(2\pi\hat{\sigma}^2)+\log_2(e)\frac{\sum_{i=1}^{n}(\hat{f}(x)-y)^2}{2\hat{\sigma}^2}=\frac{n}{2}\log_2(2\pi\hat{\sigma}^2)-\frac{n}{2}\log_2 e=\frac{n}{2}\log_2(2e\pi\hat{\sigma}^2)\quad.$$

In fact, we can leave out the terms that are constant wrt. the model class when comparing different model classes (but the same data and hence, the same n), and we can simply use the formula

$$\frac{n}{2}\log_2\mathrm{RSS}$$

as the second-part code-length.

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

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

where k is given by the number of fitted parameters in the model *including the variance*, so in the linear case, for example, $k = 3$ (a, b, and σ).

To give an example, in the case of the linear model, the value of the criterion is given by

$$\frac{200}{2}\log_2 3080.48+\frac{3}{2}\log_2 200\approx 1158.89+11.47=1170.36\text{ (bits)}.$$

We can now compare this with the more complex non-linear model with $k = 5$ that yields the code-length

$$\frac{200}{2}\log_2 1368.18+\frac{5}{2}\log_2 200\approx 1041.80+19.11=1060.91<1170.36.$$

Thus, in this case, even though the more complex model had a bigger complexity penalty ($19.11 > 11.47$), the better fit (smaller residuals) results in a shorter total code-length.

[3]It is helpful to note that $\log_2\exp x=\log_2(e)\,x$.

5. Q & A. We have created a discussion forum on *Piazza*. You should have received a link to register from Jussi (if not, please contact him). Register and take a look at the discussions that are going on there.

   (a) Post at least one question on the forum. It should be possible to post anonymously as well.

   (b) Respond to at least one question (other than your own). Note that you need not give a totally correct and complete answer. The whole point is to encourage discussion.

   You get one exercise point by doing the above by the exercise session. Be prepared to indicate which questions and responses are yours if you make them anonymous.