STA 414/2104, Spring 2012 — Assignment #3

Due at the start of class on March 22. Please hand it in on 8 1/2 by 11 inch paper, stapled in the upper left, with no other packaging.

This assignment is to be done by each student individually. You may discuss it in general terms with other students, but the work you hand in should be your own. In particular, you should not leave any discussion with someone else with any written notes (either on paper or in electronic form).

In this assignment, you will use a Gaussian process regression model with a form of covariance function that allows for both additive models and models with interactions between variables, and implement it using the eigendecomposition of the covariance matrix. You will try out your implementation on two artificial data sets I haved provided and on the same ozone data set as you used in Assignment 1.

Recall that in a Gaussian process regression model, the covariance function for observed training responses, $y^{(1)}, \ldots, y^{(n)}$, associated with vectors of p covariates, $x^{(1)}, \ldots, x^{(n)}$, can be written as the sum of a noise free covariance function, $K(x, x')$, and a noise term that is non-zero only for the covariance of a response with itself. That is,

$$\text{Cov}(y^{(i)}, y^{(i')}) \;=\; K(x^{(i)}, x^{(i')}) \;+\; \sigma^2 \delta_{i,i'}$$

where $\delta_{i,i'}$ is zero if $i \neq i'$ and 1 if $i = i'$. In this assignment, you will use the following form for $K(x, x')$:

$$K(x, x') \;=\; \eta^2 \Big[10^2 \;+\; \alpha \exp\Big(-\rho^2 \sum_{j=1}^{p} (x_j - x_j')^2\Big) \;+\; (1-\alpha) \sum_{j=1}^{p} \exp(-\rho^2 (x_j - x_j')^2) \Big]$$

where ρ, α, and η are parameters that will need to be found from the data.

The first exponential term in $K(x, x')$ gives higher covariance when x and x' are close in Euclidean distance, which is invariant with respect to rotation of the coordinate system. In contrast, the later sum of exponential terms gives higher covariance when individual coordinates x_j and x_j' are close, so the coordinate system used matters. Also, the first exponential term goes to zero if x and x' differ greatly in *any* coordinate, which is not the case for the later sum of exponential terms.

Another way of looking at this covariance function is that it describes the prior covariance for a function that can be written as $f(x) = f_0(x) + \sum_j f_j(x_j)$, with $f_0, f_1, \ldots, f_p$ being independent in the prior. This covariance function is the sum of covariance functions for each component function. The first exponential term corresponds to the function f_0, which allows interactions between the covariates, while the other terms correspond to the functions $f_1, \ldots, f_p$, which each look at only one covariate. The parameter α controls the relative importance of these terms. If $\alpha = 1$, the model is purely interactive. If $\alpha = 0$, the model is purely additive. Intermediate values for α would be appropriate when f has an additive component, but also has some interactions.

Gaussian process models are usually implemented using the Cholesky decomposition of the covariance matrix for the training responses. In this assignment, however, you will use the eigendecomposition, since, although findig it is about 15 times slower, it allows for a trick in which the log likelihood can be quickly evaluated for many combinations of values for η and σ (with ρ and α fixed), which will be especially helpful for the unsophisticated search strategy that will be used in this assignment.

Let C be the covariance matrix for the training responses, so that

$$C_{i,i'} \;=\; \text{Cov}(y^{(i)}, y^{(i')}) \;=\; K(x^{(i)}, x^{(i')}) \;+\; \sigma^2\delta_{i,i'} \;=\; \eta^2 B(x^{(i)}, x^{(i')}) \;+\; \sigma^2\delta_{i,i'}$$

where $B(x, x')$ is the sum of terms in square brackets in the expression for $K(x, x')$ above. We can write this as $C = \eta^2 B + \sigma^2 I$.

The eigendecomposition of C, which can be found with R's `eigen` function (use the `symmetric=TRUE` option for best performance), is

$$C \;=\; E\Lambda E^T$$

where E is an $n \times n$ matrix whose columns are the eigenvectors of C, and Λ is a diagonal matrix whose diagonal elements, $\lambda_1, \ldots, \lambda_n$, are the corresponding eigenvalues. Note that the inverse of C has the same eigenvectors as C, but the eigenvalues are $1/\lambda_1, \ldots, 1/\lambda_n$.

The eigendecomposition can be used to compute the log likelihood, which is

$$L(\rho, \alpha, \eta, \sigma) \;=\; -(1/2)\log(|C|) \;-\; (1/2)y^T C^{-1} y$$

where y is the vector of observed training responses. The log of the determinant of C is $\sum_i \log \lambda_i$. The second term can be written as

$$(1/2)y^T C^{-1} y \;=\; (1/2)y^T (E\Lambda E^T)^{-1} y \;=\; (1/2)y^T E\Lambda^{-1} E^T y \;=\; (1/2)u^T \Lambda^{-1} u$$

where $u = E^T y$. Note that Λ^{-1} is diagonal with elements $1/\lambda_1, \ldots, 1/\lambda_n$ on the diagonal. In R, the product of a diagonal matrix with diagonal elements given by the vector `d` times a vector `u` can be computed as `d*u`, which is much faster than the matrix product, `diag(d)%*%u`.

The eigendecomposition can also be used to find the predictive mean for the response in a test case, which is

$$k^T C^{-1} y \;=\; k^T E\Lambda^{-1} E^T y \;=\; k^T E\Lambda^{-1} u$$

where k is the vector of covariances of the test response with the training responses.

The advantage of using the eigendecomposition is that if the eigendecomposition of B is $E\Lambda E^T$, with Λ being diagonal with diagonal elements $\lambda_1, \ldots, \lambda_n$, then the eigendecomposition of $C = \eta^2 B + \sigma^2 I$ is $C = E\Lambda' E^T$, with the same eigenvectors as B, and eigenvalues of $\lambda'_i = \eta^2\lambda_i + \sigma^2$. So after one expensive computation of the eigendecomposition of B, for some values of ρ and α, the eigendecomposition of C can be quickly found for any values of η and σ, as long as the values of ρ and α are unchanged. (Note that it may be necessary to add a slight amount (eg, 0.0001^2) to the diagonal of B before finding its eigendecomposition to avoid numerical problems, which would have a negligible effect on the results.)

You should write an R function called `gp` that implements the Gaussian process model described above, using the eigendecomposition method. The arguments of `gp` should be a matrix of training inputs, `X`, a vector of training responses, `y`, a matrix of test inputs, `Xtst`, and vectors `rho.vals`, `alpha.vals`, `eta.vals`, and `sigma.vals` that contain values for the ρ, α, η, and σ parameters to consider. This function should find the combination of parameter values (from among those to consider) that has the highest likelihood, and then use those parameter values to predict the responses for the test cases. (You need only produce a predictive mean, not a predictive variance.) It should return both these predictions and the maximum likelihood values of the parameters that it found (as a list, which may contain other information too, if you wish).

For this assignment, you should consider the following sets of values for the parameters:

$$\begin{array}{rcl}
\rho & : & 0.10,\ 0.14,\ 0.20,\ 0.28,\ 0.40,\ 0.56,\ 0.80,\ 1.1 \\
\alpha & : & 0,\ 1/3,\ 2/3,\ 1 \\
\eta & : & 0.35,\ 0.50,\ 0.71,\ 1.0,\ 1.4,\ 2.0,\ 2.8,\ 4.0,\ 5.6,\ 8.0 \\
\sigma & : & 0.20,\ 0.28,\ 0.40,\ 0.56,\ 0.80,\ 1.1,\ 1.6,\ 2.2,\ 3.2,\ 4.5
\end{array}$$

On the course web page, I have provided two artificial data sets to try out your program on, along with the same ozone data set as you used for Assignment 1. They are to be read with `read.table` using the `head=TRUE` option. You should standardize the inputs for these data sets to have mean zero and standard deviation one on the training cases (the test cases should of course be rescaled with the same offset and scaling factor as the training cases). Be sure to convert the inputs to a matrix before trying to use them in matrix operations.

You should report the square root of the average squared error for the predictions on test cases for these data sets, along with the values of ρ, α, η, and σ that are chosen by maximum likelihood. You should hand in your `gp` function and the script you use to run it on these data sets. You should also hand in a discussion of the results, for which you may wish to look at other things, such as how well the Gaussian process model performs on these data sets if you force the extreme values of $\alpha = 0$ (purely additive) and $\alpha = 1$ (purely interactive), and how performance on the ozone data set compares to what you found with the methods in Assignment 1.