Non-Parametric Regression Modeling (PE & I version)

Gary D. Knott, Ph.D.
Civilized Software Inc.
12109 Heritage Park Circle
Silver Spring, MD 20906 USA
Email: csi@civilized.com
Phone: (301) 962-3711

When you can't find an adequate model, nonparametric regression modeling does the job.

Gary D. Knott, Ph.D. Civilized Software Inc.

Gary D. Knott, Ph.D. is CEO of Civilized Software Inc. (Silver Spring, MD (301)962-3711, home-page: http://www.civilized.com), a firm that specializes in mathematical and statistical software development. Civilized Software Inc. is the developer of the mathematical and statistical modeling program MLAB.

In many cases, you run into data which has an understood physical basis and you can come up with a model–a set of equations that describe it–to describe or predict its behavior. The modeling job then becomes one of finding the best model parameters, which you do by applying a method such as least-squares curve fitting.

Matters get more complicated if you can't find a model from theory that adequately describes the data. Then you may try an alternative approach to describing the data that doesn't use a data-independent equation. Such a description is called a nonparametric model. Situations where nonparametric models are needed often arise with economic time-series data or other phenomenological data. Indeed, when working with such data, using a formulaic model is often a poor approach because the selected model is almost certainly logically wrong.

There are a variety of methods for obtaining a non-parametric model for given data. These methods include kernel estimation, polynomial fitting with moving-weights, smoothing splines, moving-means and moving-medians. Here we will concentrate on kernel estimation and moving-means; moving-weight polynomial fitting and smoothing splines have comparable performance properties, but they are far more mathematically-elaborate methods.

Let us focus on the 2-dimensional case where we have a collection of data points $(x_1, y_1), \ldots, (x_n, y_n)$ for which a model $f(x)$ is desired. In this context, kernel non-parametric regression estimation of f entails computing f as a weighted sum of the values $y_1, \ldots, y_n$ where the weights are normalized peaks defined by some kernel form with each peak centered at x_i so as to represent the contribution of the data point (x_i, y_i). Thus using kernel estimation defines f as

$$f(x) = \sum_{1 \le i \le n} y_i [K((x - x_i)/u) / \sum_{1 \le j \le n} K((x - x_j)/u)].$$

The kernel function K can be chosen in a variety of ways. One reasonable choice is as a truncated gaussian density function: $K(v) =$ if $|v| > 6$ then 0 else $g(v)$, where $g(v) = \frac{1}{2\pi} e^{-v^2}$. Acutually, as you can see, the $1/(2\pi)$ factor is not necessary.

The parameter u is called the *kernel width*; it determines the spread of the kernel function. The larger u is, the more unimportant the differences in the individual y_i-values become. As u tends to ∞, $f(x)$ tends to a constant value independent of x. [What happens as $u \to 0$?] A reasonable choice for u is $[(\max_{1 \le i \le n} x_i - \min_{1 \le i \le n} x_i)/(2n/5)]^{1/2}$. A more elaborate approach might use dynamically-varying values of u depending upon the x-values.

We may look at an example of non-parametric regression via kernel estimation using the mathematical and statistical modeling system *MLAB*. The *MLAB* computer program was originally developed at the National Institutes of Health and includes curve-fitting, differential equation-solving, statistics and graphics as some of its major capabilities. *MLAB* is a tool for researchers in science and engineering. *MLAB* is an ideal tool for solving simulation and parameter-estimation problems such as chemical kinetics, or neurophysiological models.

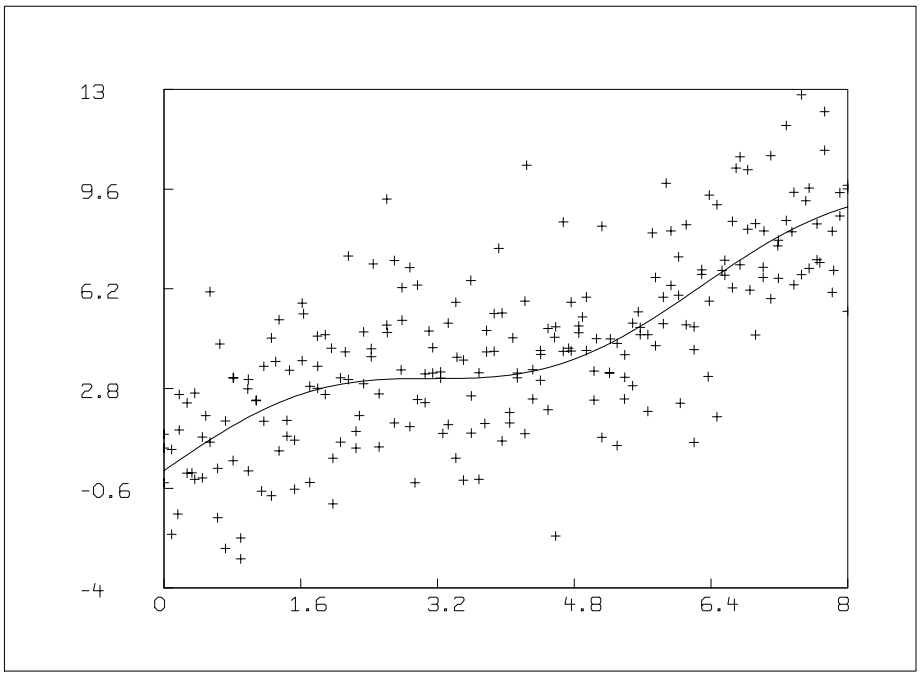
We'll actually use *MLAB* to generate our example data as well as to compute and display the kernel-based model function; this shows the ability of *MLAB* to do discrete simulation studies, in addition to its central focus of doing continuous simulation via solving systems of differential equations. The text below shown in "typerwriter" font presents the *MLAB* commands used to compute and draw the exhibited results.

We will use the "trend" curve $sin(x) + x$ and add normal random noise to obtain our "data". We will arrange to generate some data points with duplicate x-values in order to subsequently demonstrate the handling of such data. The expression `a:b!k` denotes a column vector of `k` equally-spaced

values starting with `a` and ending with `b`. The operator `&` denotes row-concatenation and the operator `&'` denotes column-concatenation. The expression `points(f,((0:8!90)r)&(0:8!50))` denotes the 230-row, 2-column matrix whose first column contains the values `0:8!90` repeated twice followed by the values `0:8!50` and whose second column contains the `f`-values corresponding to the x-values in the first column.

```
fct f(x)=g(x)+2*normran(0)
fct g(x)=sin(x)+x
m=sort(points(f,((0:8!90)^^2)&(0:8!50)))

draw m lt none pt crosspt ptsize .01
xv=0:8!120
draw trend = points(g,xv)
view
```




```
delete trend
```

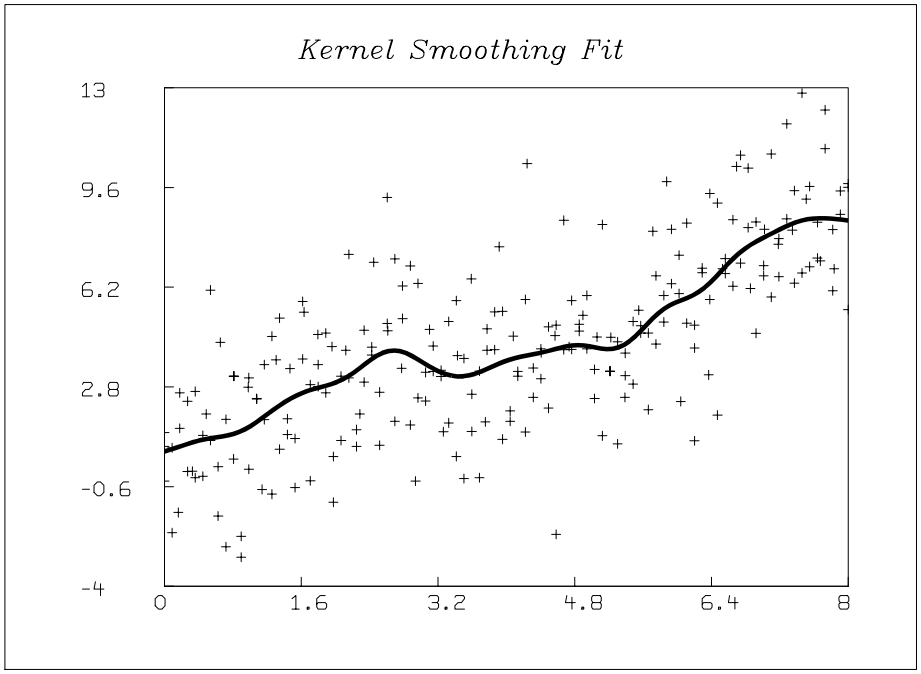
Now we will compute the kernel-based non-parametric estimate for the data points in `m`. In order to be more efficient, we use embedded assignment operator (`_`) in the definition of the kernel estimator function sf. This allows us to accumulate the required denominator sum in the variable `ks` without repetitive computation.

```
n=nrows(m)
x= m col 1; y=m col 2;

fct sf(x)=(ks_0)+sum(j,1,n,[ks_ks+[kv_Kr((x-m[j,1])/u)]]*0+kv*m[j,2])/ks
fct Kr(x)=if abs(x)>6 then 0 else gaussd(x)

u = sqrt((maxv(x)-minv(x))/(2*n/5))
s=points(sf,xv)

draw kernel = s color orange lt (1,0,0,0,0,.005,0)
top title "Kernel Smoothing Fit" color red font 17
view
```




```
delete kernel
```

Note that the resulting curve is relatively smooth, but not as smooth as the underlying trend curve. The degree of "oscillation" is controlled by the kernel width parameter u. The greater u is, the smoother the kernel-estimator is and the less it tracks local features in the data.

Another very common approach to non-parametric regression is to use a moving-average method. The basic idea is that the value at x_i should be the

weighted average of the nearby data points. This has the effect of smoothing the variation in the data to yield less extreme and less oscillatory data points. Indeed all non-parametric regression schemes can be considered to be forms of data smoothing. Thus, for example, we may define our moving-average model as:

$$m(x_i) = \sum_{1 \le j \le k} w_j y_{i-1-\lfloor k/2 \rfloor + j}$$

Here k is the size of the moving data "window" and $w_1, \ldots, w_k$ are weights which sum to 1. Generally these weights should taper down to 0 at each end of the sequence; otherwise the moving-average model will be non-continuous. The moving-average method is a certain form of kernel estimation method restricted to estimating values only at the given ordinal locations $x_1, \ldots, x_n$. We can extend the moving-average method by using an interpolating function such as a cubic spline interpolating function of the points $(x_1, m(x_1)), \ldots, (x_n, m(x_n))$ produced by the moving-average computation.

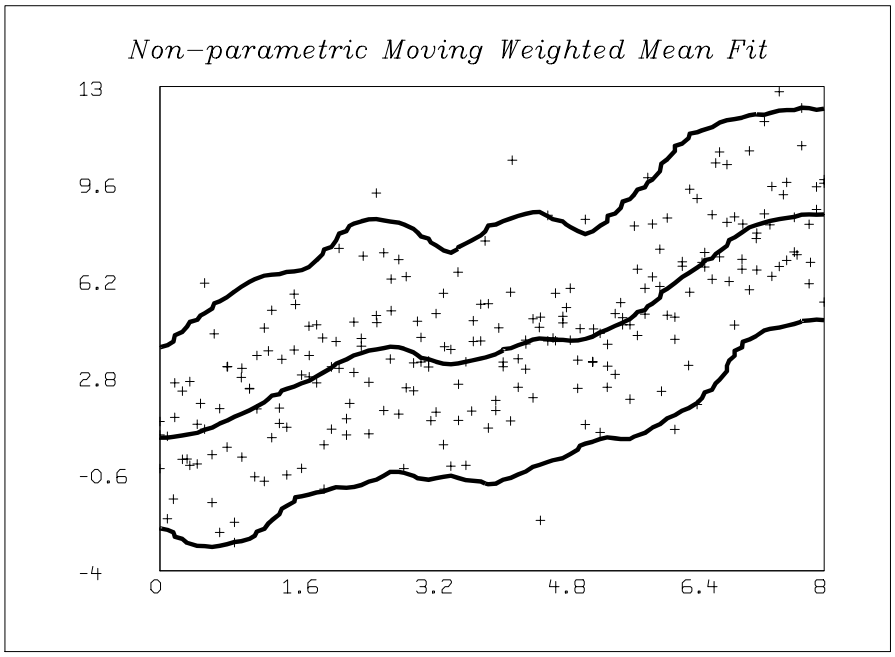
MLAB contains a built-in moving-average operation together with various other weighted moving window computations. We can demonstrate the moving-average non-parametric model for the data exhibited above using *MLAB*.

Below we show the weighted moving-average non-parametric model curve and an associated 2σ standard-deviation error band. These curves are all computed using the corresponding builtin functions within *MLAB* in order to produce the non-parametric model curves and associated error bands for the data that we generated above. Note that the data has multiple points with the same x-values, which is easily handled by *MLAB*. In this case we will use a moving window of 45 points which are weighted with the weight-values specified in the vector `wm`.

```
* wm=(0:1!15)&(1^^15)&(1:0!15)
* a=mmean(m col 2,45,0,4,wm)
* s=mstddev(m col 2,45,0,4,wm)

* draw m lt none pt crosspt ptsize .01
* za=(m col 1)&'a&'s; za=rdup(za); tx = za col 1;
* draw za col (1,2) color red lt (1,0,0,0,0,.005,0)
* draw tx &'((za col 2)+1.96*(za col 3)) color brown lt (1,0,0,0,0,.005,0)
```

```
* draw tx &'((za col 2)-1.96*(za col 3)) color brown lt (1,0,0,0,0,.005,0)
* top title "Non-parametric Moving Weighted Mean Fit" font 17
* view
```

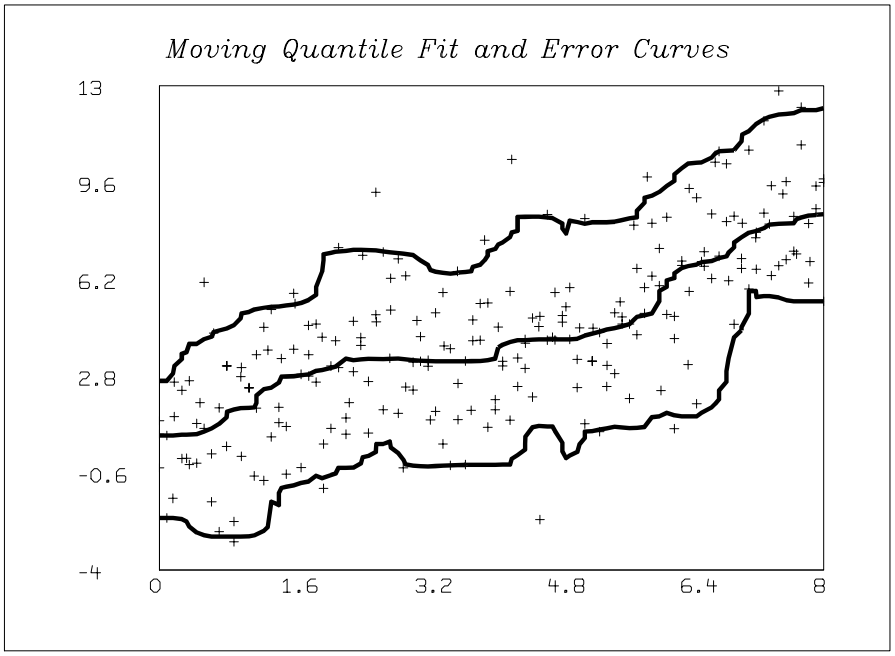


Again the degree of "oscillation" is governed by the size of the moving window and the weights that are used.

When the data has large variations or outliers, a more robust method of estimating a central trend function is to use a moving-median non-parametric method. An example is shown below using the built-in weighted moving-quantile function `mquantile` found in *MLAB*.

```
* a=mquantile(m col 2,.5,45,0,4,wm)
* q1=mquantile(m col 2,.05,45,0,4,wm)
* q2=mquantile(m col 2,.95,45,0,4,wm)

* draw m lt none pt crosspt ptsize .01
* za=(m col 1)&'a&'q1&'q2; za=rdup(za);
* draw za col (1,2) color red lt (1,0,0,0,0,.005,0)
* draw za col (1,3) color brown lt (1,0,0,0,0,.005,0)
* draw za col (1,4) color brown lt (1,0,0,0,0,.005,0)
* top title "Moving Quantile Fit and Error Curves" font 17
* view
```



In this situation the estimated curve is noticablly "rougher", but less influenced by outlier points. One sometimes useful-strategum is to use .05 and .95 quantile curves to determine a data band region, discard those points outside the band, and then fit the remainning points with a moving mean curve.