Kriging

June 04, 2016 *Weitian Tong*

1 Ideas

All interpolation algorithms, including inverse distance squared, splines, radial basis functions, triangulation, estimate the value at a given point as a weighted sum of data values at surrounding points. Almost all these weight functions are decreasing with increasing of separation "distance" [1].

Assume that $Z(\boldsymbol{x})$ can be decomposed into a deterministic, *i.e.* nonrandom trend function $\mu(x)$, and a real-valued residual random function $Y(\boldsymbol{x})$, such that

$$Z(\boldsymbol{x}) = \mu(\boldsymbol{x}) + R(\boldsymbol{x}).$$

$Z(\boldsymbol{x})$ is treated as a random field with a trend component, $\mu(\boldsymbol{x})$, and a residual component, $R(\boldsymbol{x})$. Kriging estimates residual at $\boldsymbol{x}$ as weighted sum of residuals at surrounding data points. Kriging weights are derived from covariance function or semivariogram, which should characterize residual component.

There are several versions of Kriging based on different assumptions. For more details about Kriging, please refer to the attached note on Kriging. For the details on varigram, please refer to the wikipedia. We will concentrate on the Ordinary Kriging first.

KRIGING
Input: $\mathrm{PM}_{2.5}$ data set $D = \{\boldsymbol{x}_i, i = 1, \ldots, n\}$ and an interpolation point $\boldsymbol{x}_0$
Output: Predict the $\mathrm{PM}_{2.5}$ value of $\boldsymbol{x}_0$

1. Calculate the empirical variogram
2. Fit the empirical variogram with some well-known varigram function
3. Calculate corresponding matrix and vector by the varigram function
4. Estimate the point by a simple calculation of matrix multiplication

The above is the framework for the Kriging method. Here are my ideas.

- For Step 2, instead of fitting the empirical variogram with some well-known varigram function, we learn the varigram.
- For Step 4, find a quick way to implement the matrix multiplication.

Please implement the classic Original Kriging first. Then we will discuss how to learn the varigram and speed up the matrix multiplication.

[1] Here we need to define *distance* appropriately