This is a C++ implementation of the localized sliced inverse regression method. The goal is to do supervised dimension reduction, that is, given the predictor/input matrix X (of size $n \times p$ where $n$ is the sample size and $p$ is the number of predictors) and the response/output vector Y (of size $n$), find linear combinations of the predictors (i.e., FEATURES) such that those features are predictive.

For instance, if the response is only correlated with the second predictor, then the function should be able to return a coefficient vector close to $[0 \ 1 \ 0 \ 0 \ ...]'$ (with dimension $p$) — post-multiplying it with $X$ yields the feature matrix. Another example: if the response is say the square of the second predictor plus the square of the fifth predictor plus noise, then the function should be able to return a coefficient vector close to $[0 \ 1 \ 0 \ 0 \ ...]'$ and a coefficient vector close to $[0 \ 0 \ 0 \ 1 \ 0 \ 0 \ ...]'$. These coefficient vectors are called e.d.r (effective dimension reduction) directions.

There are four important parameters that need to specify each time. They are:

- **xy_type[]**: if =$'c'$: Y is discrete (classification type); if =$'r'$: Y is continuous (regression type).
- **d**: How many coefficient vectors (e.d.r. directions) should be returned?
- **H**: Number of slices/bins. If xy_type[]='$c'$” this will be reset to the number of classes so an arbitrary number would be fine; if xy_type[]='$r'$” the default value is 5 and this needs to be tuned.
- **numNN**: Number of nearest neighbors in each slice/bin. Could set it to be around one third of $n/H$, however it had better be tuned.

***************

Below is an outline of the algorithm for the localized sliced inverse regression method:

1. Suppose there are samples $(x_i, y_i), i = 1, \cdots, n$. Each $x_i$ is a $p$-vector representing there are $p$ predictors.

2. Divide the sample into $H$ groups $G_1, \cdots, G_H$ in terms of the response, that is, if $y_i$’s are discrete then the samples are naturally grouped: each class is a group; if $y_i$’s are continuous then sort the $y_i$’s first and group the sample according to the sorted response: the “floor($n/H$)” samples with the smallest response form $G_1$, so on.

3. For each sample $(x_i, y_i)$ compute the local mean

$$m_{i,loc} = \frac{1}{k} \sum_{j \in s_i} x_j$$
where

\[ s_i = \{ j : x_j \text{ is in the same group of } x_i \text{ and belongs to the } k \text{ nearest neighbors of } x_i \} \]

Euclidean distance between vectors is used. In the code “numNN” replaces “k”.

4. Compute the between-variation matrix

\[ \Gamma_B = \frac{1}{n} \sum_{i=1}^{n} (m_{i,\text{loc}} - m)(m_{i,\text{loc}} - m)' \]

where \( m \) is the mean of \( x_i \)'s (the grand mean); Compute the within-variation matrix

\[ \Gamma_W = \frac{1}{n} \sum_{i=1}^{n} (x_i - m)(x_i - m)' \]

Both are \( p \times p \) matrices.

5. Solve the generalized eigen-value problem:

\[ \Gamma_B \beta = \lambda \Gamma_W \beta \]

The columns of \( \beta \) give the e.d.r. directions. In practice can return the first \( d \) columns.

For details please refer to the original paper (downloadable at http://stat.duke.edu/ km68/lsir.htm, see the first line on that page.)