P-Index on Big Data
Technical report

Miha Stajdohar
Bioinformatics Laboratory
Faculty of Computer and Information Science
University of Ljubljana

October 27, 2014

1 Projection Pursuit

Projection Pursuit is a linear mapping algorithm that seeks good projections of multivariate data into a low-dimensional space. The algorithm uses a continuous index that measures the "usefulness" of a projection—the P-Index. The index is defined to maximize the concentration of points into clusters while, at the same time, separate the clusters.

The P-index that expresses such properties of a projection axis k can be written as a product of two functions

$$I(k) = s(k)d(k) \tag{1}$$

where $s(k)$ measures the spread of the data, and $d(k)$ describes the "local density" of the points after projection onto k. For $d(k)$, we take the trimmed standard deviation of the data from the mean as projected onto k:

$$s(k) = [\sum_{i=pN}^{(1-pN)} (X_i \cdot k - \bar{X}_k)^2/(1-2p)N]^{1/2} \tag{2}$$

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 318633

Here N is the total number of data points, and $X_i (i = 1, N)$ are the multivariate vectors representing each of the data points, ordered according to their projections $X_i \cdot k$. A small fraction p of the points that lie at each of the extremes of the projection are omitted from both sums. Thus, extreme values of $X_i \cdot k$ do not contribute to $s(k)$, which is robust against extreme outliers. For $d(k)$, an average nearness function is used

$$d(k) = \sum_{i=1}^{N} \sum_{j=1}^{N} 1(R - r_{ij}) \tag{3}$$

where

$$r_{ij} = |X_i \cdot k - X_j \cdot k| \tag{4}$$

and $1(q)$ is unity for positive-valued arguments and zero for negative values.

For moderate to large sample size N, the cutoff radius R is usually chosen so that the average number of points contained within the window.

For projections in 2 dimensions (k and l) we generalize the equations to

$$s(k, l) = s(k)s(l) \tag{5}$$

and

$$r_{ij} = [(X_i \cdot k - X_j \cdot k)^2 + (X_i \cdot l - X_j \cdot l)^2]^{1/2}. \tag{6}$$

2 Optimization for Big Data

Instead of searching for a general projection we can use the P-Index to score a fixed projection. We can use it to find the best combination of attributes in visualizations like scatter plots. This allows us to extend the method to work with big data. Instead of considering all the points, we can discretize the data into bins and use contingency table to represent the distribution of points. This would resemble drawing a grid on the scatterplot and coloring the cells to represent the cell density. The density of the grid is a parameter of our method and defines how well will the optimized P-Index resemble the original one.

From the contingency table, we can compute the spread of the data as

$$s = \text{std}(\text{sum}_r(C))\,\text{std}(\text{sum}_c(C)) \tag{7}$$

where $C \in \mathbb{N}^{n_i \times n_j}$ is a contingency table of two attributes discretized into n_i and n_j bins, $\text{sum}_r(C)$ the vector corresponding to the row sums of C and $\text{sum}_c(C)$ the vector of column sums.

We compute the local density as

$$d = \sum_{\substack{0 \le i,i' < n_i \\ 0 \le j,j' < n_j \\ (i',j') < (i,j)}} C_{i,j} C_{i',j'} \left(\sqrt{2} - \sqrt{\left(\frac{i-i'}{n_i - 1}\right)^2 + \left(\frac{j-j'}{n_j - 1}\right)^2} \right) \tag{8}$$

and a normalization factor as

$$t = \frac{n(n-1)}{2}. \tag{9}$$

where $n = n_i n_j$ is the number of cells in the contingency table.

The P-Index is computed as

$$I = s\frac{d}{t} \tag{10}$$

3 Analysis and Use Cases

We implemented the method in the graphical data exploration toolkit Orange 3. Figure 1 displays an analysis of the Wine UCI data set. We can observe how the second best Scatterplot projection (attributes Proline and OD280/OD315) neatly separates the three clusters of points.

The computational complexity of P-Index is $O(n \log n)$. We can assume we compute the P-Index of a contingency table for a given granularity in constant time. The time complexity of the optimized P-Index then equals the time complexity to compute the contingency table, which is linear with regard to the number of examples.

We compared the the performance of the two algorithms on two data sets—Wine and WDBC—in Figures 2 and 3.

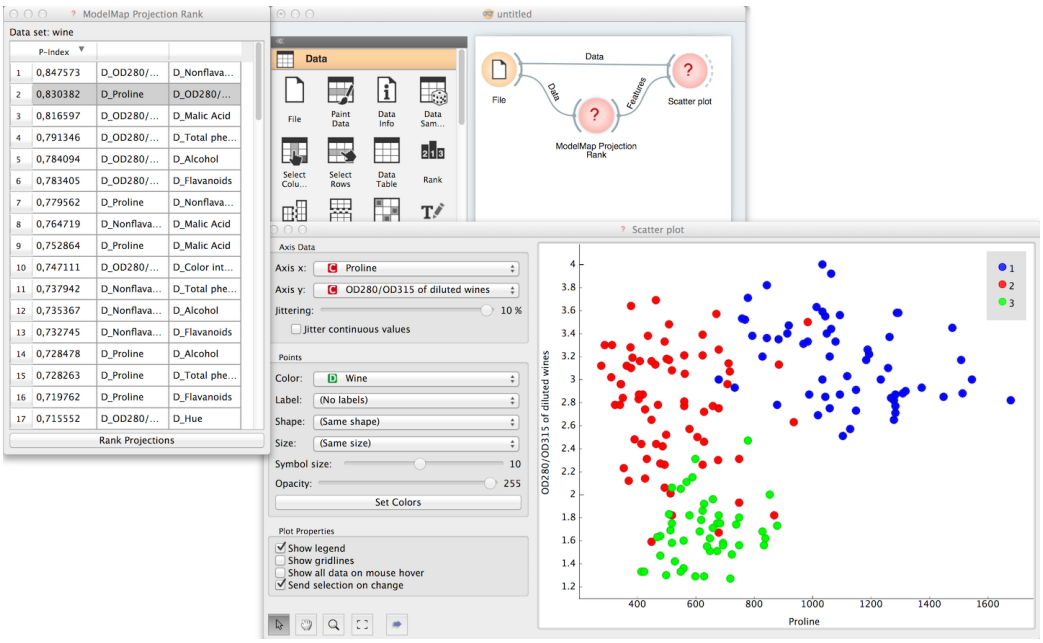

Figure 1: Orange schema and Model Map Projection Rank widget example.

4 Conclusions

We implemented the P-Index based on contingency table in the ModelMap Projection Rank widget in Orange 3. The code that compares the the original and optimized P-Index is open source and available in the Github repository (https://github.com/mstajdohar/orange3-modelmaps).

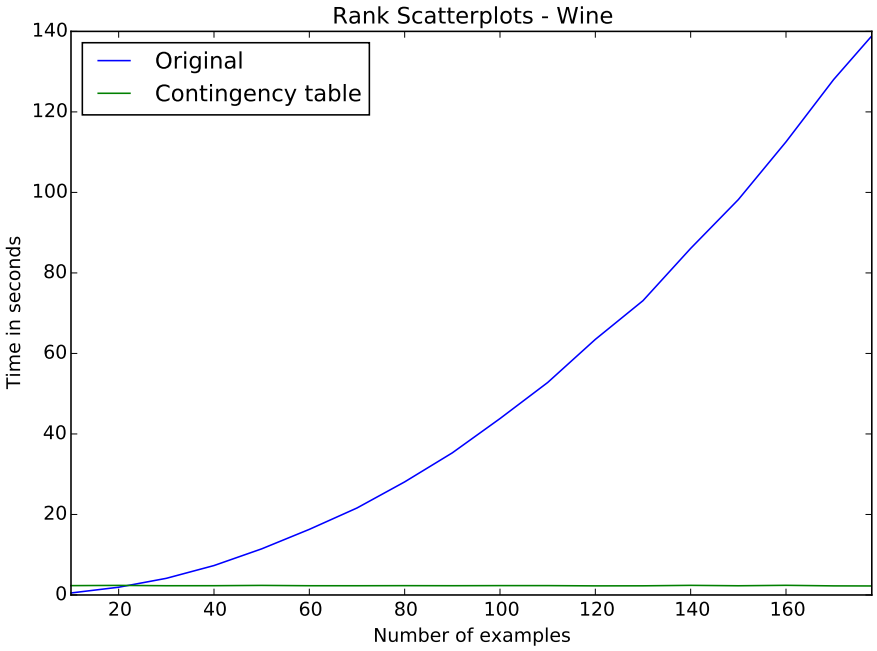

Figure 2: Computation of P-Index with classical and optimized method on the Wine data set.

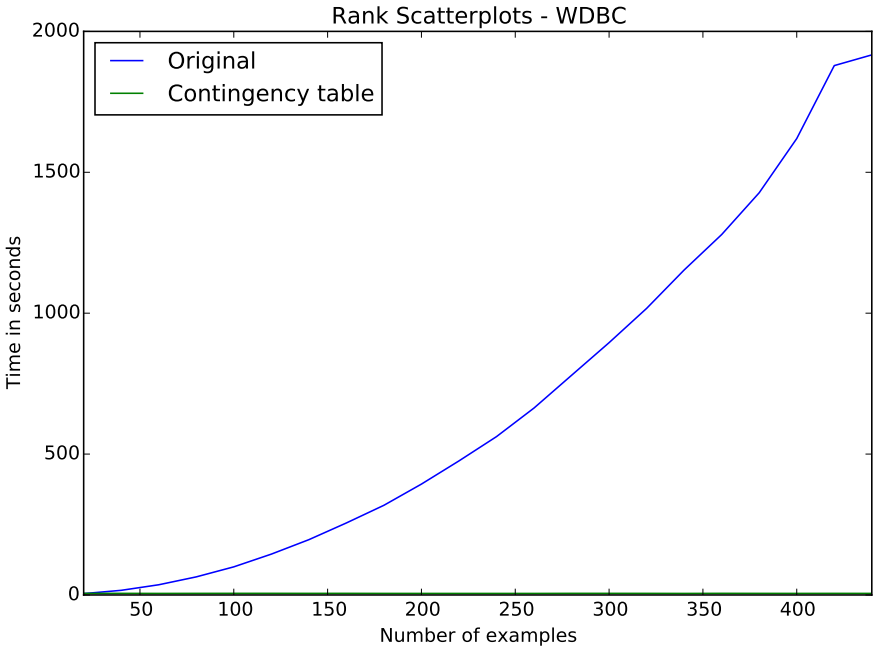

Figure 3: Computation of P-Index with classical and optimized method on the WDBC data set.