Visual Diagnostics for Constrained Optimisation with Application to Guided Tours

by H.Sherry Zhang, Dianne Cook, Ursula Laa, Nicolas Langrené, and Patricia Menéndez

Abstract Guided tour searches for interesting low-dimensional views of high-dimensional data via optimising a projection pursuit index function. The first paper of projection pursuit by Friedman and Tukey (1974) stated that "the technique used for maximising the projection index strongly influences both the statistical and the computational aspects of the procedure." While much work has been done in proposing indices in the literature, less has been done on evaluating the performance of the optimisers. In this paper, we implement a data collection object in the optimisation of projection pursuit guided tour and introduce visual diagnostics based on the data object collected. These diagnostics and this workflow can be applied to a broad class of optimisers, to assess their performance. An R package, **ferrn**, has been created to implement the diagnostics.

keywords: graphics, multivariate, optimization

Introduction

Visualisation is widely used in exploratory data analysis (Tukey, 1977; Unwin, 2015; Healy, 2018; Wilke, 2019). Presenting information in graphics often unveils information that would otherwise not be discovered and provides a more comprehensive understanding of the problem at hand. Task specific tools such as Li et al. (2020) show how visualisation can be used to understand, for instance, the behaviour of neural network on classification models. However, no general visualisation tool is available for diagnosing optimisation procedures. The work presented in this paper brings visualization tools into optimisation problems with the aim to better understand the performance of optimisers in practice.

The goal of continuous optimisation is to find the best solution within the space of all feasible solutions where typically the best solution is decided by an objective function. Broadly speaking, optimization can be unconstrained or constrained (Kelley, 1999). The unconstrained problem can be formulated as a minimization (or maximization) problem such as $\min_x f(x)$ where $f : \mathbb{R}^n \to \mathbb{R}$ is an objective function with certain properties defined in an L^p space. In this case, solutions rely on gradient descent or ascent methods. In the constrained optimization problem additional restrictions are introduced via a set of functions that can be convex or non-convex: $g_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \dots k$ and hence the problem can be written as $\min_x f(x)$ *subject to* $g_i(x) \leq 0$. Here methods such as multipliers and convex optimization methods including linear and quadratic programming can be used.

The focus of this paper is on the optimisation problem arising in the projection pursuit guided tour (Buja et al., 2005), an exploratory data analysis tool used for detecting interesting structures in high-dimensional data through a set of lower-dimensional projections (Cook et al., 2008). The goal of the optimisation is to identify the projection, characterised by the projection basis, that gives the most interesting low-dimensional view. A view is interesting if it can show some unexpected structure of the data, e.g., bi-modality, clustering and outliers.

The optimization challenges encountered in the projection pursuit guided tour problem are common to those of optimization in general. Examples of those include the existence of multiple optima (local and global), the trade off between computational burden and proximity to the optima, dealing with noisy objective functions that might be non-smooth and non-differentiable (Jones et al., 1998). Those are not unique to this context and therefore the visualization tools and optimization methods presented in this paper can be easily applied to any other optimization problems.

The remainder of the paper is organised as follows. Section 2.2 provides an overview of optimisation methods, specifically line search methods. Section 2.3 reviews projection pursuit guided tour, defines the optimisation problem and outlines three existing algorithms. Section 2.4 presents the new visual diagnostics. A data structure is defined to capture information during the optimisation, and different diagnostic plots are designed with the data collected. Section 2.5 shows applications of how these diagnostic plots can be used to discover interesting aspects of different optimisers and guide improvement to the existing algorithms. Finally, Section 2.6 describes the R package: **ferrn**, that implements the visual diagnostics.

Optimisation methods

Optimisation problems are ubiquitous in many areas of study. While in some cases analytical solutions can be found, the majority of problems rely on numerical methods to find the optimal solution. These numerical methods follow iterative approaches that aim at finding the optimum by progressively improving the current solution until a desirable accuracy is achieved. Although this principle seems uncomplicated, a number of challenges arise such as the possible existence of multiple maxima (local and global), constraints and noisy objective function, and the trade-off between desirable accuracy and computational burden. In addition, the optimization results might depend on the algorithm starting values, affecting the consistency of results.

The optimisation problem of our interest is on constrained optimization (Bertsekas, 2014) as defined in the introduction section, and assuming it is not possible to find a solution to the problem in the way of a closed-form. That is, the problem consists of finding the minimum or maximum of a function $f \in L^p$ in the constrained $\mathcal{A}$ space.

A large class of methods utilises the gradient information of the objective function, with the most notable one being the gradient ascent (descent) method. Although gradient optimization methods are popular, they rely on the availability of the objective function derivatives and on the complexity of the constraints. As will be shown in the next section, the independent variable, in our optimisation problem, is a matrix of projection basis and the computational time required to perform differentation on a matrix could impede the render of tour animation. Hence, gradient-based methods are not the focus of this paper and considerations will be given to derivative-free methods. Derivative-free methods, which do not rely on the knowledge of the gradient, are more generally applicable. Derivative-free methods have been developed over the years, where the emphasis is on finding, in most cases, a near optimal solution. Below will introduce two random search methods: creeping random search and simulated annealing, and a pseudo derivative search.

A common search scheme utilised by both derivative-free methods and gradient methods is line search. In line search methods, users are required to provide an initial estimate x_1 and, at each iteration, a search direction S_k and a step size α_k are generated. Then one moves on to the next point following $x_{k+1} = x_k + \alpha_k S_k$ and the process is repeated until the desire convergence is reached. While gradient-based methods choose the search direction by the gradient, derivative-free methods uses local information of the objective function to determine the search direction. The choice of step size also needs considerations, as inadequate step sizes might prevent the optimisation method to converge to an optimum. An ideal step size can be chosen via finding the value of $\alpha_k \in \mathbb{R}$ that maximises $f(x_k + \alpha_k S_k)$ with respect to α_k at each iteration.

Several R implementations address optimization problems with both general purpose as well as task specifics solvers. The most prominent one within the general solvers is `optim()` in the **stats** (R Core Team, 2020) package, which provides both gradient-based and derivative-free optimisation functions. Another general solver specialised in non-linear optimisation is **nloptr** (Johnson, 2020). Specific solvers for simulated annealing include `optim(...,method "SANN")` and package **GenSA** (Xiang et al., 2013) that deals with more complicated objective functions. For other task specific solvers, readers are recommended to visit the relevant sections in CRAN task review on *optimisation and mathematical programming* (Theussl et al., 2020).

Projection pursuit guided tour

Projection pursuit guided tour combines two different methods (projection pursuit and guided tour) in exploratory data analysis. Projection pursuit, coined by Friedman and Tukey (1974), detects interesting structures (e.g. clustering, outliers and skewness) in multivariate data via low-dimensional projections. Guided tour is one variation of a broader class of data visualisation methods, tour, which displays high-dimensional data through a series of animated projections.

Let $\mathbf{X}_{n\times p}$ be the data matrix with n observations in p dimensions. A d-dimensional projection is a linear transformation from $\mathbb{R}^p$ into $\mathbb{R}^d$ defined as $\mathbf{Y} = \mathbf{X} \cdot \mathbf{A}$, where $\mathbf{Y}_{n\times d}$ is the projected data and $\mathbf{A}_{p\times d}$ is the projection basis matrix. We define $f : \mathbb{R}^{n\times d} \mapsto \mathbb{R}$ to be an index function that maps the projected data $\mathbf{Y}$ onto a scalar value. This is commonly known as the projection pursuit index function, or just index function, and is used to measure the "interestingness" of a given projection. An interesting projection shows structures that are non-normal and this has been argued for two reasons: Theoretical proofs from Diaconis and Freedman (1984) show that projections tend to be normal as n and p approaching infinity under certain conditions. Huber (1985) illustrates that, for indices based on entropy measures, [...]. Index functions proposed in the litearture include early indices that can be categorised by measuring the L^2 distance between the projection and a normal distribution.(Legendre index (Friedman and Tukey, 1974), Hermite index (Hall et al., 1989), and natural Hermite index (Cook

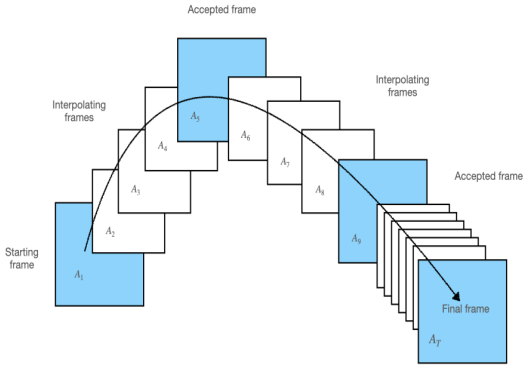

Figure 1: An illustration for demonstrating the frames in a tour path. Each square (frame) represents the projected data with a corresponding basis. Blue frames are returned by the projection pursuit optimisation and white frames are constructed between two blue frames by geodesic interpolation.

et al., 1993)); chi-square index (Posse, 1995) for detecting spiral structure; kurtosis (Loperfido, 2020) and skewness (Loperfido, 2018) indices for detecting outliers in financial time series; and most recently, scagnostic indices (Laa and Cook, 2020) for summarising structures in scatterplot matrices based on eight scagnostic measures.

As a general visualisation method, tour produces animations of high dimensional data via rotations of low-dimensional planes. There are different tour implementations depending on how the high dimensional space is investigated: grand tour (Cook et al., 2008) selects the planes randomly to provide a general overview; manual tour (Cook and Buja, 1997) gradually phases in and out one variable to understand the contribution of that variable in the projection. Guided tour, the main interest of this paper, chooses the planes with the aid of projection pursuit, to gradually reveal the most interesting projection. Given a random start, projection pursuit iteratively finds bases with higher index values and the guided tour constructs a geodesic interpolation between these planes to form a tour path. Figure 1 shows a sketch of the tour path where the blue frames are produced by the projection pursuit optimisation and the white frames are interpolations between the blue frames. Mathematical details of the geodesic interpolation can be found in Buja et al. (2005). The aforementioned tour method has been implemented in the R package **tourr** (Wickham et al., 2011).

Optimisation in the tour

In projection pursuit the optimisation problem aims at finding the global and local maxima that give interesting projections according to an index function. That is, it starts with a given randomly selected basis $\mathbf{A}_1$ and aims at finding an optimal final projection basis $\mathbf{A}_T$ that satisfies the following optimisation problem:

$$\arg\max_{\mathbf{A}\in\mathcal{A}} f(\mathbf{X}\cdot\mathbf{A}) \quad s.t. \quad \mathbf{A}'\mathbf{A} = I_d \tag{1}$$

where f and $\mathbf{X}$ are defined as in the previous section, $\mathcal{A}$ is the set of all p-dimensional projection bases, I_d is the d-dimensional identity matrix, and the constraint ensures the projection bases, $\mathbf{A}$, to be orthonormal. It is worth noticing the following: 1) The optimisation is constrained and the orthonormality constraint imposes a geometrical structure on the bases space: it forms a Stiefel manifold. 2) There may be index functions for which the objective function might not be differentiable. 3) While finding the global optimum is the goal of the optimisation problem, interesting projections may also appear in the local optimum. 4) The optimisation should be computationally agile since the tour animation is viewed by the users during the optimisation.

Existing algorithms

Three optimisers have been implemented in the **tourr** (Wickham et al., 2011) package: Creeping random search (CRS), simulated annealing (SA), and pseudo derivative (PD). Creeping random search (CRS) is a random search optimiser that samples a candidate basis $\mathbf{A}_l$ in the neighbourhood of the current basis $\mathbf{A}_{\text{cur}}$ by $\mathbf{A}_l = (1-\alpha)\mathbf{A}_{\text{cur}} + \alpha\mathbf{A}_{\text{rand}}$ where α controls the radius of the sampling neighbourhood and $\mathbf{A}_{\text{rand}}$ is generated randomly. $\mathbf{A}_l$ is then orthonormalised to fulfil the basis

constraint. If $\mathbf{A}_l$ has an index value higher than the current basis $\mathbf{A}_{\text{cur}}$, the optimiser outputs $\mathbf{A}_l$ for guided tour to construct an interpolation path. The neighbourhood parameter α is adjusted by a cooling parameter: $\alpha_{j+1} = \alpha_j * \text{cooling}$ before the next iteration starts. The optimiser terminates when the maximum number of iteration l_{max} is reached before a better basis can be found. The algorithm of SA is summarised in Algorithm 1 in the appendix. Posse (1995) has proposed a slightly different cooling scheme by introducing a halving parameter c. In his proposal, the α is only adjusted if the last iteration takes more than c times to find a better basis.

Simulated annealing (SA) (Kirkpatrick et al., 1983; Bertsimas et al., 1993) uses the same sampling process as CRS but allows a probabilistic acceptance of a basis with lower index value than the current. Given an initial value of T_0, the temperature at iteration l is defined as $T(l) = \frac{T_0}{\log(l+1)}$. When a candidate basis fails to have an index value larger than the current basis, SAJO gives it a second chance to be accepted with probability

$$P = \min\left\{\exp\left[-\frac{|\ I_{\text{cur}} - I_l\ |}{T(l)}\right], 1\right\}$$

where $I_{(\cdot)}$ denotes the index value of a given basis. This implementation allows the optimiser to make a move and explore the basis space even the landing basis does have a higher index value and hence enable the optimiser to jump out of a local optimum. The algorithm 2 in the appendix highlights how SAJO differs from SA in the inner loop.

Pseudo derivative (PD) search (Cook et al., 1995) uses a different strategy than SA and SAJO. Rather than randomly sample the basis space, PD first computes a search direction by evaluating vases close to the current basis. A step size is then chosen along the geodesic direction by another optimisation over an 90 degree angle from $-\pi/4$ to $\pi/4$. The resulting candidate basis $\mathbf{A}_{**}$ is returned for the current iteration if it has a higher index value than the current one. Algorithm 3 in the appendix summarises the inner loop of the PD.

Visual diagnostics

A data structure for diagnosing optimisers in projection pursuit guided tour is first defined. With this data structure, four types of diagnostic plots are presented. In the section, examples will be given to show how these plot can be used to reveal issues in the projection pursuit optimisation.

Data structure for diagnostics

Three main pieces of information are recorded for the projection pursuit optimisers: 1) projection bases $\mathbf{A}$, 2) index values I, and 3) state S. For CRS and SA, possible states include `random_search`, `new_basis`, and `interpolation`. Pseudo derivative (PD) has a wider variety of states including `new_basis`, `direction_search`, `best_direction_search`, `best_line_search`, and `interpolation`. Multiple iterators index the information collected at different levels: t is a unique identifier prescribing the natural ordering of each observation; j and l are the counter of the outer and inner loop respectively. Other parameters of interest recorded include `method` that tags the name of the optimiser, and `alpha` that indicates the sampling neighbourhood size for searching observations. A matrix notation of the data structure is presented in equation 2.

$$
\left[\begin{array}{cccccccc}
t & \mathbf{A} & I & S & j & l & V_1 & V_2 \\
\hline
1 & \mathbf{A}_1 & I_1 & S_1 & 1 & 1 & V_{11} & V_{12} \\
\hline
2 & \mathbf{A}_2 & I_2 & S_2 & 2 & 1 & V_{21} & V_{22} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & l_2 & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & 2 & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & k_2 & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & J & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
T & \mathbf{A}_T & I_T & S_T & J & l_J & V_{T1} & V_{T2} \\
\hline
\vdots & \vdots & \vdots & \vdots & J & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & J & k_J & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & J+1 & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
T' & \mathbf{A}_{T'} & I_{T'} & S_{T'} & J+1 & l_{J+1} & V_{T'1} & V_{T'2}
\end{array}\right]
=
\left[\begin{array}{c}
\text{column name} \\
\hline
\text{search (start basis)} \\
\hline
\text{search} \\
\vdots \\
\text{search (accepted basis)} \\
\hline
\text{interpolate} \\
\vdots \\
\text{interpolate} \\
\hline
\vdots \\
\hline
\text{search} \\
\vdots \\
\text{search (final basis)} \\
\hline
\text{interpolate} \\
\vdots \\
\text{interpolate} \\
\hline
\text{search (no output)} \\
\vdots \\
\text{search (no output)}
\end{array}\right]
\tag{2}
$$

where $T' = T + k_J + l_{J+1}$. Note that there is no output in iteration $J+1$ since the optimiser does not find a better basis in the last iteration and terminates. The final basis found is A_T with index value I_T.

The data structure constructed above meets the tidy data principle (Wickham et al., 2014) that requires each observation to form a row and each variable to form a column. With tidy data structure, data wrangling and visualisation can be significantly simplified by well-developed packages such as **dplyr** (Wickham et al., 2020) and **ggplot2** (Wickham, 2016).

The construction of diagnostic plots adopts the core concept of grammar of graphics (Wickham, 2010) in ggplot2. In grammar of graphics, plots are not produced by calling the commands, named by the appearance of the plot, i.e., boxplot and histogram, but by the concept of stacked layers. Seeing plots as stacked layers gives analysts the freedom to compose plots with multiple elements on hand.

Diagnostic 1: Checking how hard the optimiser is working

A starting point of diagnosing an optimiser is to understand how many searches an optimiser has conducted. One may want to simply plot the index value of the search points across its natural order, but a point geometry may work well if each iteration has a similar number of points. When some iterations have considerably more points than others, using a point geometry over-emphasizes the iterations that have more search points since these iterations will occupy the mast majority of the plot space. An alternative is to summarise the search in each iteration using boxplots and each iteration will then be spaced out equally. Occasionally, one may still want to switch back to a point geometry if the number of points is small in a particular iteration and this can be achieved via the `cutoff` argument in the search plot function. Additional annotations are added to facilitate better reading of the plot and these include

1) the number of points searched in each iteration can be added as text label at the bottom of each

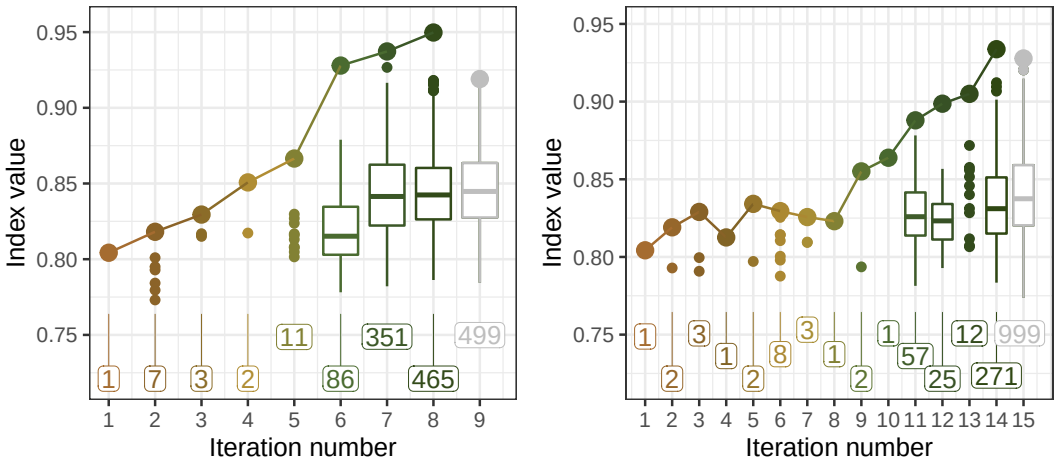

Figure 2: A comparison of the searches by two optimisers: CRS (left) and SA (right) on a 2D projection problem of a six-variable dataset, `boa6` using holes index. Both optimisers reach the final basis with a similar index value while it takes SA longer to find the final basis.

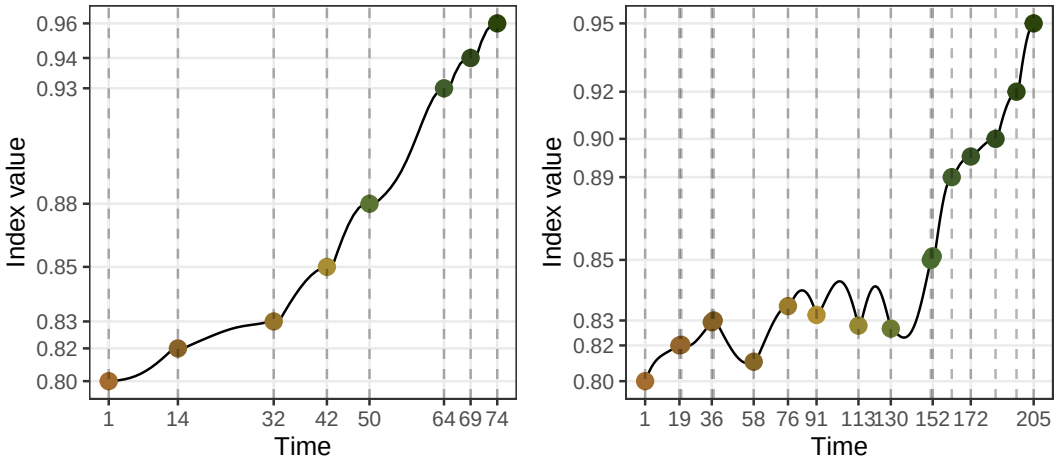

Figure 3: An inspection of the index values as the optimisation progress for two optimisers: CRS (left) and SA (right). The holes index is optimised for a 2D projection problem on the six-variable dataset `boa6`. Lines indicate the interpolation and dots indicate new target bases generated by the optimisers. Interpolation in both optimisation is smooth while SA is observed to first pass by some bases with higher index value before reach the target bases in time 76-130.

iteration; 2) the anchor bases to interpolate are connected and highlighted in a larger size; and 3) the colour of the last iteration is in a grey scale to indicate no better basis found in this iteration.

Figure 2 shows an example of the search plot for CRS (left) and SA (right). Both optimisers quickly find better bases in the first few iterations and then take longer to find one in the later iterations. The anchor bases, the ones found with the highest index value in each iteration, always have an increased index value in the optimiser CRS while this is not the case for SA. This feature gives SA an advantage in this simple example to quickly find the optimum.

Diagnostic 2: Examining the optimisation progress

Another interesting diagnostic is to examine how the index value improves between interpolating bases since the projection on these bases is played by the tour animation. Trace plots are created by plotting the index value against time. Figure 3 presents the trace plot of the same optimisers as Figure 2 and one can observe that the trace is smooth in both cases. It may seem bizarre at the first sight that the interpolation in the left panel first passes some bases with higher index value before decreases to a lower target. This happens because on the one hand, the probabilistic acceptance in SA implies that some worse bases will be accepted by the optimiser. On the other hand, guided tour interpolate between the current and target basis to provide a smooth transition between projections. These two facts indicate that a non-monotonic interpolation can't be avoided for SA. Later, in section 2.5.2, there

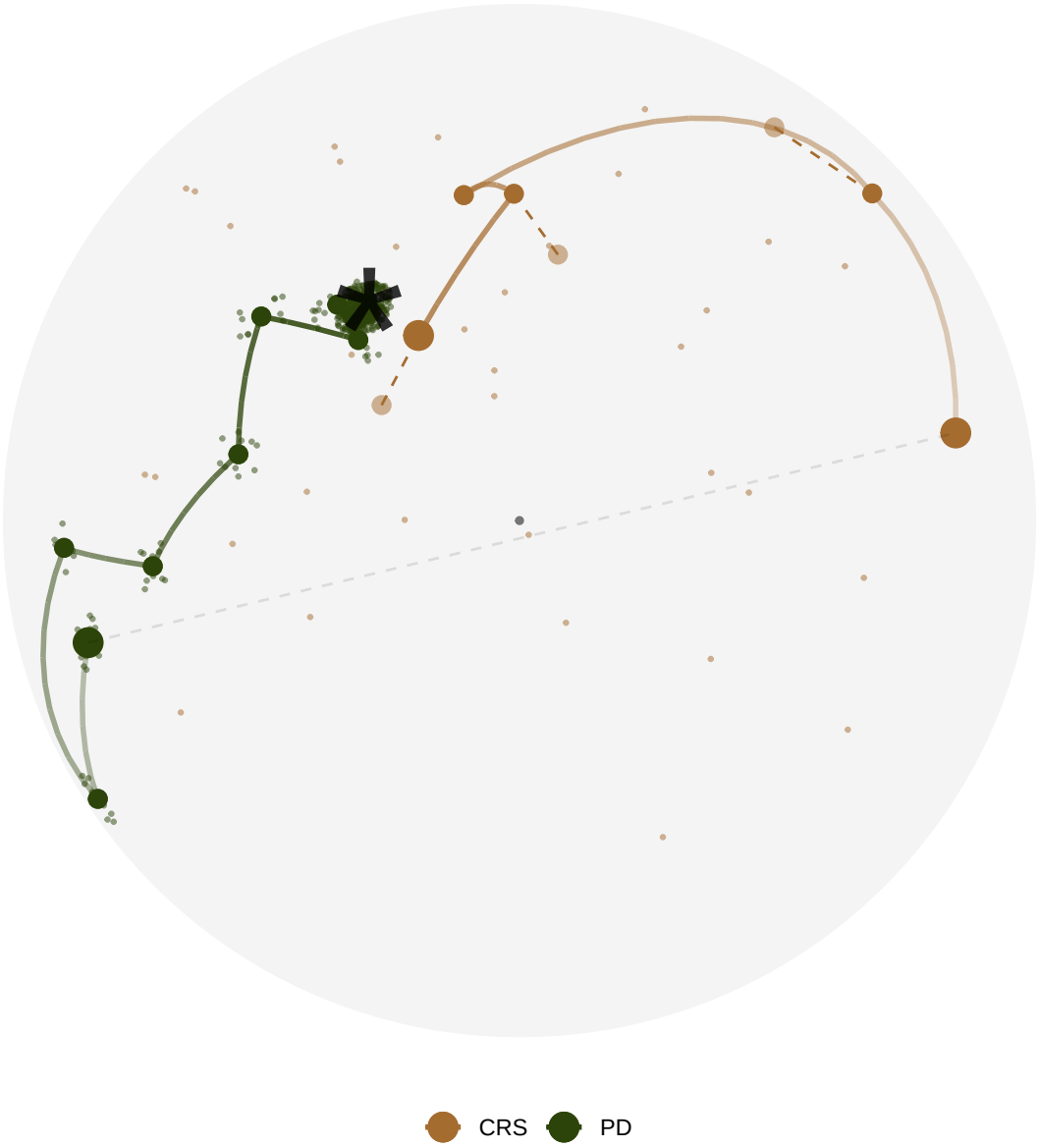

Figure 4: Search paths of CRS (green) and PD (brown) in the PCA-reduced basis space for 1D projection problem on the five-variable dataset, `boa5` using holes index. The basis space, a 5D unit sphere, is projected onto a 2D circle by PCA. All the bases in PD has been flipped for easier comparison of the final bases and a grey dashed line has been annotated to indicate the symmetry of the two start bases.

will be a discussion on improving the non-monotonic interpolation for CRS.

Diagnostic 3a: Understanding the optimiser's coverage of the search space

Apart from checking the search and progression of an optimiser, looking at where the bases are positioned in the basis space is another interest. Given the orthonormality constraint, the space of projection bases $\mathbf{A}_{p\times d}$ is a $p \times d$-dimensional sphere. A dimensionality reduction method, e.g. principal component analysis is applied to first project all the bases onto a 2D space. In a projection pursuit guided tour optimisation, there are various types of bases involved: 1) The starting basis; 2) The search bases that the optimiser evaluated to produce the anchor bases; 3) The anchor bases that have the highest index value in each iteration; 4) The interpolating bases on the interpolation path; and finally 5) the end basis. The importance of these bases differs and the most important ones are the starting, interpolating and end bases. The anchor and search bases can be turned on with argument `details = TRUE`. Sometimes, two optimisers can start with the same basis but finish with bases of opposite sign. This happens because the projection is invariant to the sign difference of the bases and so does the index value, however, this creates difficulties for comparing the optimisers since the end bases will be symmetric with respect to the origin. A sign flipping step is conducted to flip the signs of all the bases

Figure 5: Six frames selected from the animated version of the previous plot. With animation, the progression of the search paths from start to finish are better identified. CRS (green) finishes the optimisation quicker than PD (brown) since there is no further movement for CRS in the sixth frame. The full video of the animation can be found at https://vimeo.com/504242845.

in one routine if different optimisations finish at opposite places.

Several annotations have been made to help understanding this plot. As mentioned previously, the original basis space is a high-dimensional sphere and random bases on the sphere can be generated via the CRAN package **geozoo** (Schloerke, 2016). Along with the bases recorded during the optimisation and a zero basis, the parameters (centre and radius) of the 2D space can be estimated. PCA is performed to get the first two PC coordinates of all the bases and the centre of the circle is the PC coordinates of the zero matrix, and radius is estimated as the largest distance from the centre to the basis. The theoretical best basis is known for simulated data and can be labelled to compare how close the end basis found by the optimisers is to the theoretical best. Various aesthetics, i.e. size, alpha and colour, are applicable to emphasize critical elements and adjust for the presentation. For example, anchor points and search points are less important and hence a smaller size and alpha is used. Alpha can also be applied on the interpolation paths to show the start to finish from transparent to opaque.

Figure 4 shows the PCA plot of CRS and PD for a 1D projection problem. Both optimisers find the optimum but PD gets closer. With PCA plot, one can visually appreciate the nature of these two optimisers: PD first evaluates points in a small neighbourhood for a promising direction, while CRS evaluates points randomly in the search space to search for the next target. There are dashed lines annotated for CRS and it describes the interruption of the interpolation, which will be discussed in details in Section 2.5.2.

Diagnostic 3b: Animating the diagnostic plots

Animation is another type of display to show how the search progresses from start to finish in the space. An `animate = TRUE` argument is used to enable the animation in the PCA plot. Figure 5 shows six frames from the animation of the PCA plot in Figure 4. An additional piece of information one can learn from this animation is that CRS finds its end basis quicker than PD since CRS finishes its search in the 5th frame while PD is still making more progress.

Diagnostic 4a: The tour looking at itself

As mentioned previously, the original $p \times d$ dimension space can be simulated via randomly generated bases in the **geozoo** (Schloerke, 2016) package. While the PCA plot projects the bases from the direction that maximises the variance, the tour plot displays the original high-dimensional space from various directions using animation. Figure 6 shows some frames from the tour plot of the same two optimisations in its original 5D space.

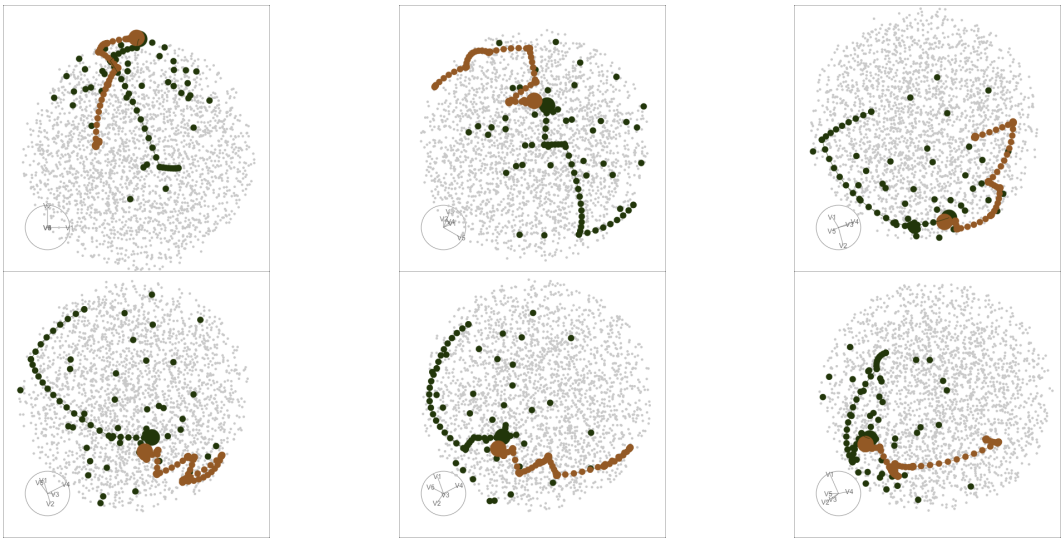

Figure 6: Six frames selected from rotating the high dimensional basis space, along with the same two search paths from Figure 4 and 5. The basis space in this example is a 5D unit sphere, on which points (grey) are randomly generated via the CRAN package **geozoo**. The full video of the animation can be found at https://vimeo.com/512885379.

Diagnostic 4b: Forming a torus

While the previous few examples have looked at the space of 1D bases in a unit sphere, this section visualises the space of 2D bases. Recall that the columns in a 2D basis are orthogonal to each other, the space of $p \times 2$ bases is a p-D torus. If take $p = 3$, one would see a classical 3D torus shape as shown by the grey points in Figure 7. The two circles of the torus can be observed to be perpendicular to each other and this can be linked back to the orthogonality of the columns in the bases. Two paths optimised by CRS and PD are plotted on top of the torus and coloured in green and brown respectively to match the previous few 1D space plots. The final basis found by PD and CRS are shown in a larger shape and printed below respectively:

```
#>              [,1]        [,2]
#> [1,]  0.001196285  0.03273881
#> [2,] -0.242432715  0.96965761
#> [3,] -0.970167484 -0.24226493
```

```
#>             [,1]         [,2]
#> [1,]  0.05707994 -0.007220138
#> [2,] -0.40196202 -0.915510160
#> [3,] -0.91387549  0.402230054
```

Both optimisers have found the third variable in the first column, but the second variable in the second column differs by a sign.One would expect to see this in the torus plot as the final bases match each other when projected onto one torus circle (due to the same sign in the first column) and be symmetric when projected onto the other (due the the sign difference in the second column). In Figure 7, this can be seen most clearly in frame 5 where the two circles are rotated into a line from our view.

Diagnosing an optimiser

In this section, several examples will be presented to show how the diagnostic plots discover something unexpected in projection pursuit optimisation, and guide the implementation of new features.

Simulation setup

Random variables with different distributions have been simulated and the distributional form of each variable is presented in equations 3 to 9. Variables `x1`, `x8`, `x9` and `x10` are normal noise with zero mean and unit variance and `x2` to `x7` are normal mixtures with varied weights and locations. All the variables have been scaled to have an overall unit variance before projection pursuit.

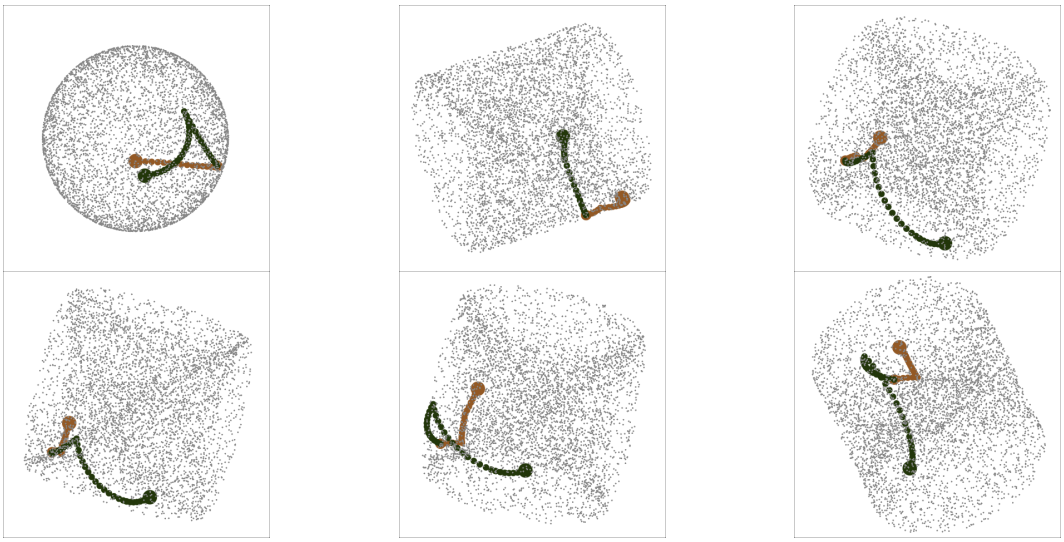

Figure 7: Six frames selected from rotating the 2D basis space along with two search paths optimised by PD (brown) and CRS (green). The projection problem is a 2D projection with three variables using holes index. The grey points are randomly generated 2D projection bases in the space and it can be observed that these points form a torus. The full video of the animation can be found at https://vimeo.com/512882784.

$$x_1 \overset{d}{=} x_8 \overset{d}{=} x_9 \overset{d}{=} x_{10} \sim \mathcal{N}(0,1) \tag{3}$$

$$x_2 \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1) \tag{4}$$

$$\Pr(x_3) = \begin{cases} 0.5 & \text{if } x_3 = -1 \text{ or } 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$x_4 \sim 0.25\mathcal{N}(-3,1) + 0.75\mathcal{N}(3,1) \tag{6}$$

$$x_5 \sim \frac{1}{3}\mathcal{N}(-5,1) + \frac{1}{3}\mathcal{N}(0,1) + \frac{1}{3}\mathcal{N}(5,1) \tag{7}$$

$$x_6 \sim 0.45\mathcal{N}(-5,1) + 0.1\mathcal{N}(0,1) + 0.45\mathcal{N}(5,1) \tag{8}$$

$$x_7 \sim 0.5\mathcal{N}(-5,1) + 0.5\mathcal{N}(5,1) \tag{9}$$

A problem of non-monotonicity

An example of non-monotonic interpolation has been given in Figure 3: a path that passes bases with higher index value than the target one. For SA, a non-monotonic interpolation is justified since target bases do not necessarily have a higher index value than the current one, while this is not the case for CRS. The original trace plot for a 2D projection problem, optimised by CRS, is shown on the left panel of Figure 8 and one can observe clearly that the non-monotonic interpolation has undermined the optimiser to realise its full potential. Hence, an interruption is implemented to stop at the best basis found in the interpolation. The right panel of Figure 8 shows the trace plot after implementing the interruption and while the first two interpolations are identical, the basis at Time 61 has a higher index value than the target in the third interpolation. Rather than starting the next iteration from the target basis on Time 65, CRS starts the next iteration at Time 61 on the right panel and reaches a better final basis.

Close but not close enough

Once the final basis has been found by an optimiser, one may want to push further in the close neighbourhood to see if an even better basis can be found. A polish search takes the final basis of an optimiser as the start of a new guided tour to search for local breakthrough. The polish algorithm is similar to the CRS but with three distinctions: 1) a hundred rather than one candidate bases are generated each time in the inner loop; 2) the neighbourhood size is reduced in the inner loop, rather than in the outer loop; and 3) three more termination conditions have been added to ensure the new

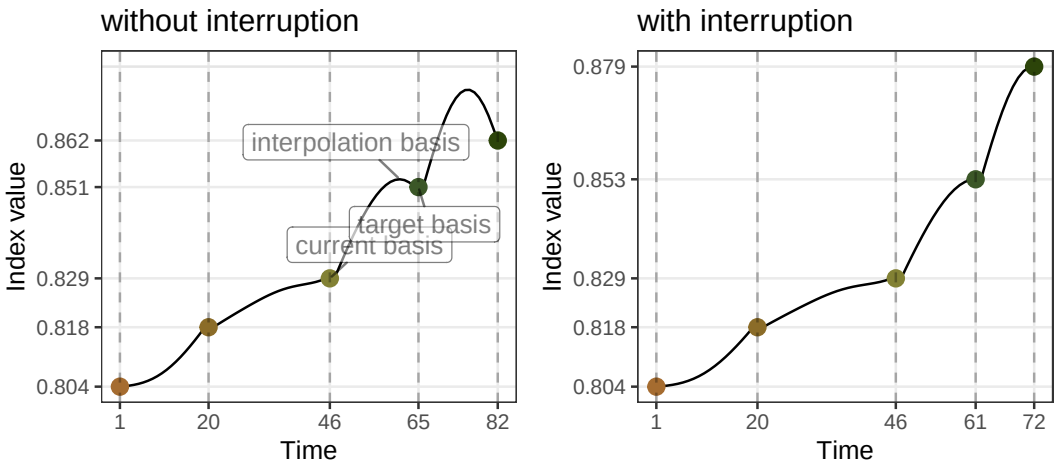

Figure 8: Comparison of the interpolation before and after implementing the interruption for the 2D projection problem on boa6 data using holes index, optimised by CRS. On the left panel, basis with higher index value is found during the interpolation but not used. On the right panel, the interruption stops the interpolation at the basis with the highest index value for each iteration and results in a final basis with higher index value, as shown on the right panel.

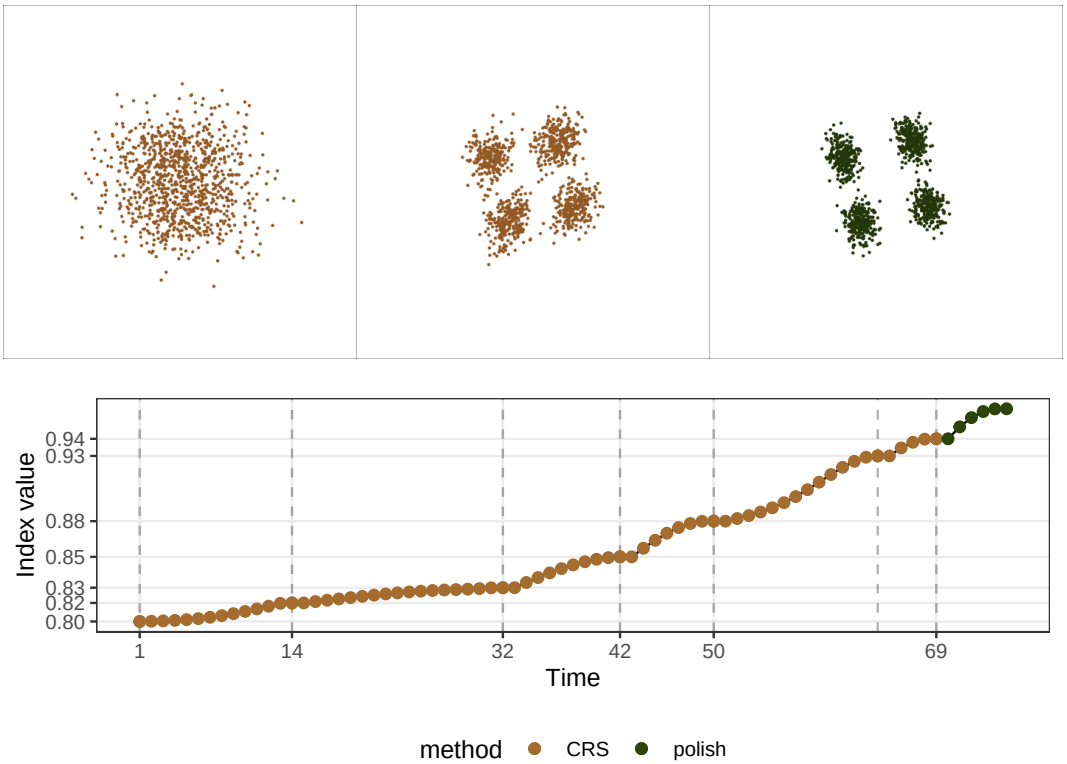

Figure 9: Comparison of the projected data before and after using polishing for a 2D projection problem on boa6 data using holes index. Top row shows the intial projected data and the final views after CRS and polish search and the second row traces the index value. The clustering structure in the data is detected by CRS (top middle panel) but the polish step improves the index value and produces clearer boundaries of the clusters (top right panel), especially along the vertial axis.

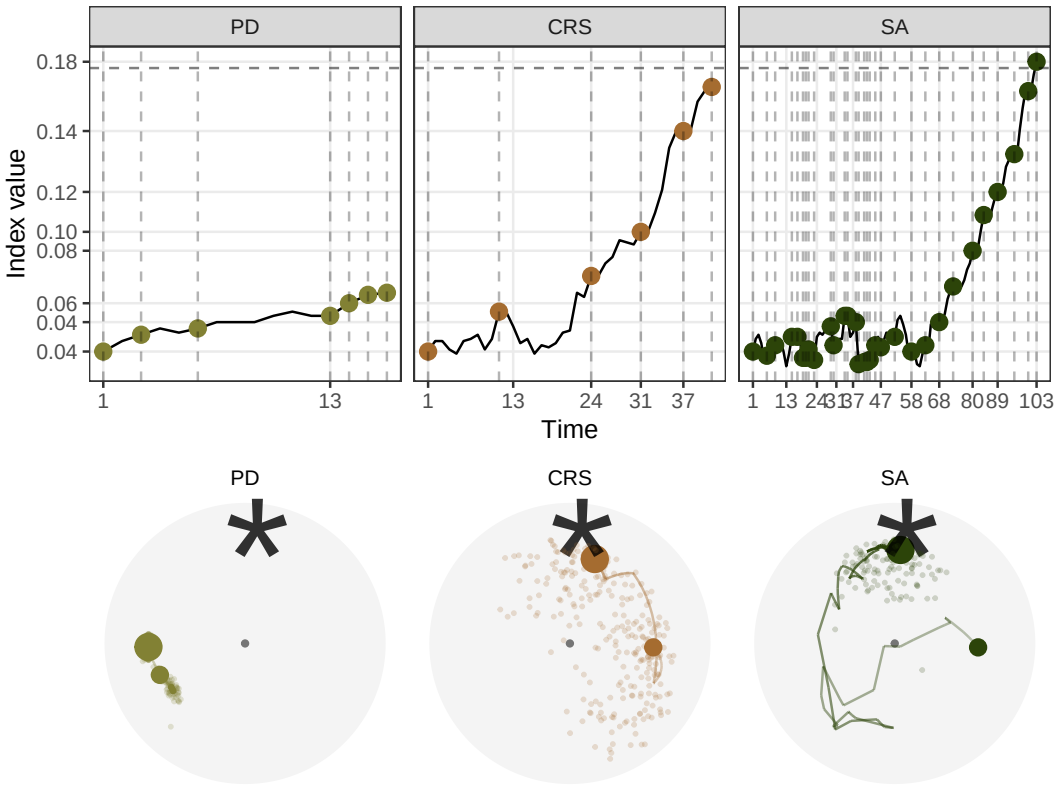

Figure 10: Comparison of the three optimisers in optimising $I^{nk}(n)$ index for a 1D projection problem on a five-variable dataset, `boa5`. Both CRS and SA succeeds in the optimisation, PD fails to optimise this non-smooth index. Further, SA takes much longer than CRS to finish the optimisation, it finishes off closer to the theoretical best.

basis generated is distinguishable from the current one in terms of the distance in the space, percentage change in the index value, and neighbourhood size:

1) the distance between the basis found and the current basis needs to be larger than 1e-3;
2) the percentage change of the index value need to be larger than 1e-5; and
3) the alpha parameter on itself needs to be larger than 0.01

Figure 9 presents the projectd data and trace plot of a 2D projection, optimised by CRS and followed by the polish. In the top row shows the intial projection, the final projection after CRS, and the final projection after polish, repsectively. The end basis found by CRS reveals the four clusters in the data, but the edges of each cluster are not clean-cut. Polish works with this end basis and further pushes the index value to produce much clear edges of the cluster, especially along the vertical axis.

Seeing the signal in the noise

The holes index function used for all the examples before this section produces a smooth interpolation, while this is not the case for all the indices. A 1D projection function, $I^K(n)$, compar,es the projected data, $\mathbf{Y}_{n\times 1}$, to a randomly generated normal distribution, $\mathcal{N}_{n\times 1}$, based on the Kolmogorov test. Let $F_{\cdot}(n)$ be the ECDF function, with two possible subscripts Y and $\mathcal{N}$ representing the projected and randomly generated data, and n denoting the number of observation, the Normal Kolmogorov index, $I^{nk}(n)$, is defined as:

$$I^K(n) = \max\left[F_Y(n) - F_{\mathcal{N}}(n)\right]$$

With a non-smooth index function, two research questions are raised:

1) whether any optimiser fails to optimise this non-smooth index; and
2) whether the optimisers can find the global optimum despite the presence of a local optimum

Figure 10 presents the trace and PCA plots of all three optimisers and as expected, none of the interpolated path is smooth. There is barely any improvement made by PD, indicating its failure in

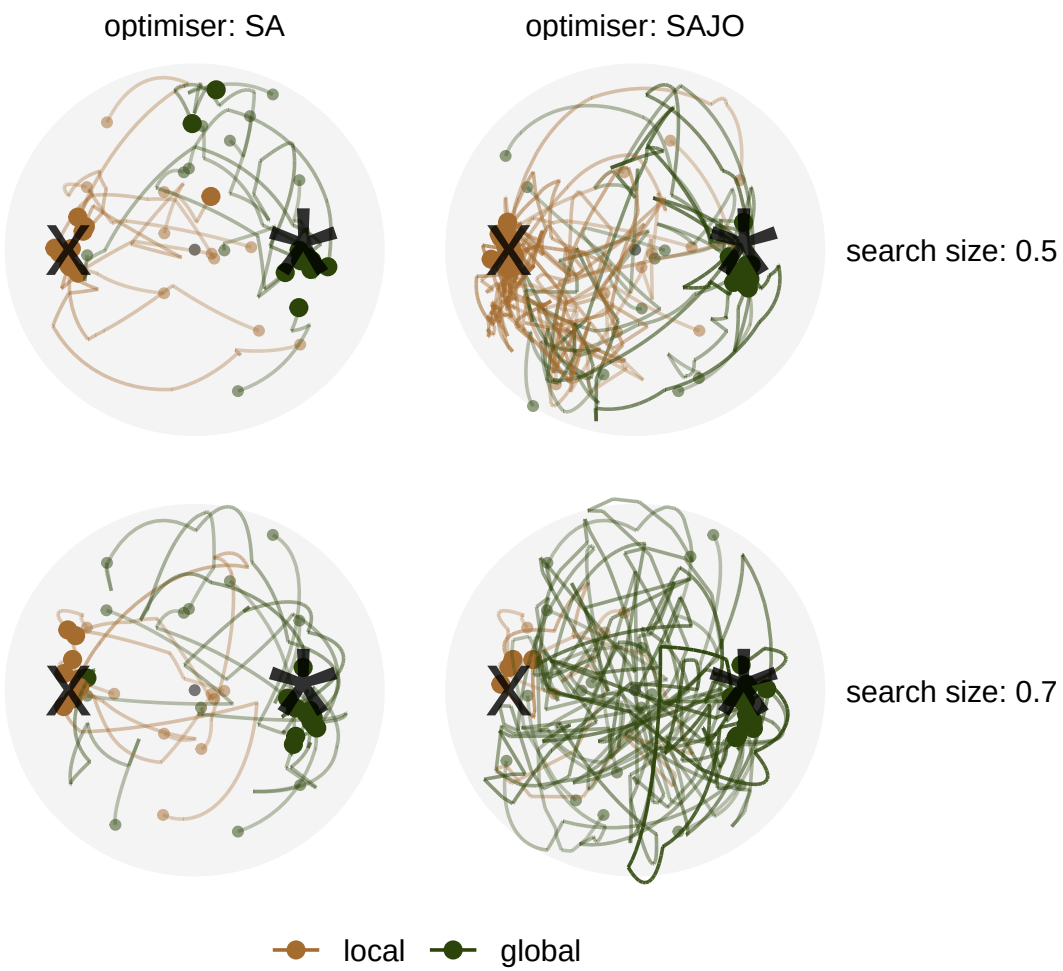

Figure 11: Comparing 20 search paths in the PCA-projected basis space facetted by two optimisers: CRS and SA, and two search sizes: 0.5 and 0.7. The optimisation is on the 1D projecton index, $I^{nk}(n)$, for boa6 data, where a local optimum, annotated by the cross (x), is presented in this experiment, along with the global optimum (*).

optimising non-smooth index. While CRS and SA have managed to get close to the index value of the theoretical best, trace plot shows that it takes SA much longer to interpolate towards the final basis. This long interpolation path is partially due to the fluctuation in the early iterations, where the SA tends to generously accept inferior bases before concentrating near the optimum. The PCA plot shows the interpolation path and search points excluding the last termination iteration. Pseudo Derivative (PD) quickly gets stuck near the starting position. Comparing the amount of random search done by CRS and SA, it is surprising that SA does not carry as many samples as CRS. Combining the insights from both the trace and PCA plot, one can learn the two different search strategies by CRS and SA: CRS frequently samples in the space and only make a move when an improvement is guaranteed to be made, while SA first broadly accepts bases in the space and then starts the extensive sampling in a narrowed subspace. The specific decision of which optimiser to use will depend on the index curve in the basis space but if the basis space is non-smooth, accepting inferior bases at first, as what SA has done, can lead to a more efficient search, in terms of the overall number of points evaluated.

The next experiment compares the performance of CRS and SA when a local maximum exists. Two search neighbourhood sizes: 0.5 and 0.7 are compared to understand how a large search neighbourhood would affect the final basis and the length of the search. Figure 11 shows 80 paths simulated using 20 seeds in the PCA plot, faceted by optimiser and search size. With CRS and a search size of 0.5, despite being the simplest and fastest, the optimiser fails in three instances where the final basis lands neither near the local nor the global optimum. With a larger search size of 0.7, more seeds have found the global maximum. Comparing CRS and SA for a search size of 0.5, SA does not seem to improve the final basis found, despite having longer interpolation paths. However, the denser paths near the local maximum is an indicator that SA is working hard to examine if there is any other optimum in the basis space but the relative small search size has diminished its ability to reach the global maximum. With a larger search size, almost all the seeds (16 out of 20) have found the global maximum and some

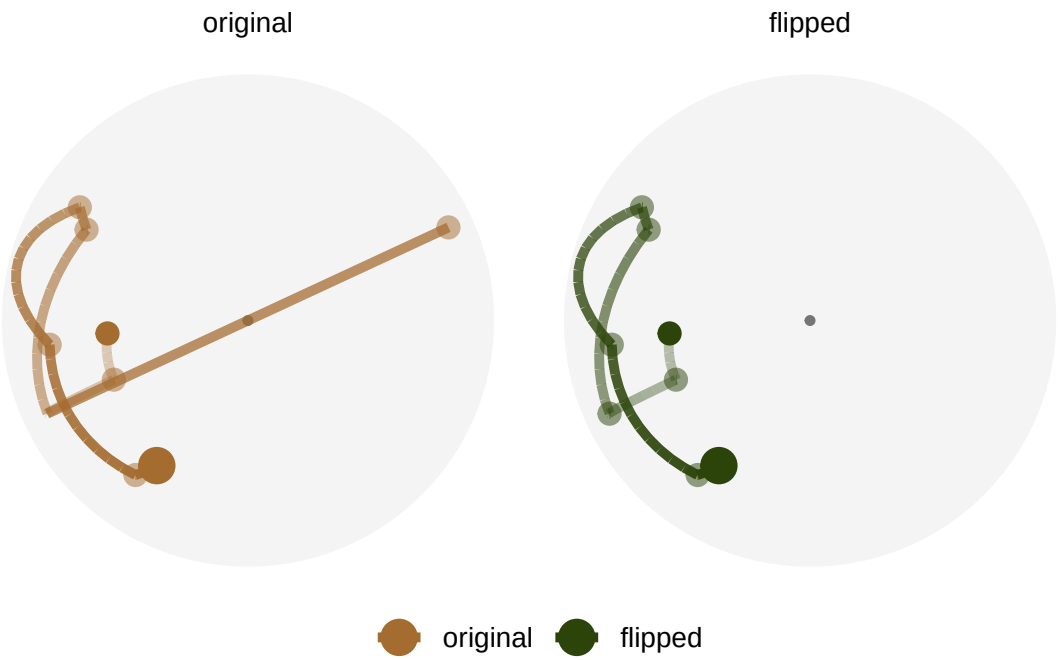

Figure 12: Comparison of the interpolation in the PCA-projected basis space before and after reconciling the orientation of the target basis. Optimisation is on the 1D projection index, $I^{nk}(n)$, for boa6 data using CRS with seed 2463. The dots represent the target basis in each iteration and the path shows the interpolation. On the left panel, one target basis is generated with an opposite orientation to the current basis (hence appear on the other side of the basis space) and the interpolator crosses the origin to perform the interpolation. The right panel shows the same interpolation after implementing an orientation check and the undesirable interpolation disappears.

final bases are much closer to the theoretical best, as compared to the three other cases. This indicates that SA, with a reasonable large search size, is able to overcome the local optimum and optimise close towards the global optimum.

Reconciling the orientation

One interesting situation observed in the previous examples is that, for some simulations as shown on the left panel of Figure 12, the target basis is generated on the other half of the basis space and the interpolator seems to draw a straight line to interpolate. As mentioned previously, bases with opposite signs do not affect the projection and index value, but clearly, we prefer the target to have the same orientation as the current basis. The orientation of two bases can be checked via calculating the determinant and a negative value indicates opposite direction. Hence an orientation check is carried out once a new target basis is generated and the sign in the target basis will be flipped if a negative determinant is obtained. The interpolation after implementing the orientation check is shown on the right panel of Figure 12 where the unsatisfactory interpolation no longer exist.

Implementation

The implementation of this projection has been divided into two packages: the data collection object is implemented in the existing CRAN package **tourr** (Wickham et al., 2011) while the optimiser diagnostics have been implemented in a new package, **ferrn**. When a guided tour is run, the users can choose to collect the data by binding `animate_*()` to a variable. Once the data object has been obtained, the **ferrn** package provides diagnostic plots shown in Section 2.4. The main functions of the **ferrn** package functionality is listed below.

- Main plotting functions:
  - `explore_trace_search()` produces summary plots in Figure 2
  - `explore_trace_interp()`produces trace plots for the interpolation points in Figure 3
  - `explore_space_pca()` produces the PCA plot of projection bases on the reduced space in Figure 4. Animated version in Figure 5 can be turned on via the argument `animate =`

TRUE.
  - `explore_space_tour()` produces animated tour view on the full space of the projection bases in Figure 6.

- `get_*()` extracts and manipulates certain components from the existing data object.
  - `get_anchor()` extracts target observations
  - `get_basis_matrix()` flattens all the bases into a matrix
  - `get_best()` extracts the observation with the highest index value in the data object
  - `get_dir_search()` extracts directional search observations for PD search
  - `get_interp()` extracts interpolated observations
  - `get_interp_last()` extracts the ending interpolated observations in each iteration
  - `get_interrupt()` extracts the ending interpolated observations and the target observations if the interpolation is interrupted
  - `get_search()` extracts search observations
  - `get_search_count()` extracts the count of search observations
  - `get_space_param()` produces the coordinates of the centre and radius of the basis space
  - `get_start()` extracts the starting observation
  - `get_theo()` extracts the theoretical best observations, if presented

- `bind_*()` incorporates additional information outside the tour optimisation into the data object.
  - `bind_theoretical()` binds the theoretical best observation in simulated experiment
  - `bind_random()` binds randomly generated bases in the projection bases space to the data object
  - `bind_random_matrix()` binds randomly generated bases and outputs in a matrix format

- `add_*()` create ggprotos for different components for the PCA plot
  - `add_anchor()` is a wrapper for plotting anchor bases
  - `add_anno()` is a wrapper for annotating the symmetry of start bases
  - `add_dir_search()` is a wrapper for plotting the directional search bases with magnified distance
  - `add_end()` is a wrapper for plotting end bases
  - `add_interp()` is a wrapper for plotting the interpolation path
  - `add_interp_last()` is a wrapper for plotting the last interpolation bases for comparing with target bases when interruption is carried
  - `add_interrupt()` is a wrapper for linking the last interpolation bases with target ones when interruption is carried
  - `add_search()` is a wrapper for plotting search bases
  - `add_space()` is a wrapper for plotting the circular space
  - `add_start()` is a wrapper for plotting start bases
  - `add_theo()` is a wrapper for plotting theoretical best bases, if applicable

- Utilities
  - `theme_fern()` and `format_label()` for better display of the grid lines and axis formatting
  - `clean_method()` to clean up the name of the optimisers
  - `botanical_palettes` is a collection of colour palettes from Australian native plants. Quantitative palettes include daisy, banksia and cherry and sequential palettes contain fern and acacia
  - `botanical_pal()` as the colour interpolator
  - `scale_color_*()` and `scale_fill_*()` for scaling the colour and fill of the plot

Conclusion

This paper has illustrated setting up a data object that can be used for diagnosing a complex optimisation procedure. The ideas were illustrated using the optimisers available for projection pursuit guided tour. Here the constraint is the orthonormality condition of the projection bases. The approach used here could be broadly applied to understand other constrained optimisers.

Four diagnostic plots have been introduced to investigate the progression and the projection space of an optimiser. The implementation of these visualisations is designed to be easy-to-use with each plot can be produced with a simple supply of the data object. More advanced users may decide to modify on top of the basic plots or even build their own.

Most of the work in this project has been translated into code in two packages: the collection of the data object is implemented in the existing **tourr** (Wickham et al., 2011) package; while the manipulation and visualisation of the data object are implemented in the new **ferrn** package. Equipped with handy tools to diagnose the performance of optimisers, future work can extend the diagnostics to a wider range of index functions, i.e. scagnostics, association, and information index (Laa and Cook, 2020) and understand how the optimisers behave for index functions with different structures.

Acknowledgements

This article is created using **knitr** (Xie, 2015) and **rmarkdown** (Xie et al., 2018) in R. The source code for reproducing this paper can be found at: https://github.com/huizezhang-sherry/paper-tour-vis.

Bibliography

D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014. URL https://doi.org/10.1016/C2013-0-10366-2. [p2]

D. Bertsimas, J. Tsitsiklis, et al. Simulated annealing. *Statistical Science*, 8(1):10–15, 1993. URL https://doi.org/10.1214/ss/1177011077. [p4]

A. Buja, D. Cook, D. Asimov, and C. Hurley. Computational methods for high-dimensional rotations in data visualization. *Handbook of Statistics*, 24:391–413, 2005. URL https://doi.org/10.1016/S0169-7161(04)24014-7. [p1, 3]

D. Cook and A. Buja. Manual controls for high-dimensional data projections. *Journal of Computational and Graphical Statistics*, 6(4):464–480, 1997. URL https://doi.org/10.2307/1390747. [p3]

D. Cook, A. Buja, and J. Cabrera. Projection pursuit indexes based on orthonormal function expansions. *Journal of Computational and Graphical Statistics*, 2(3):225–250, 1993. URL https://doi.org/10.2307/1390644. [p2]

D. Cook, A. Buja, J. Cabrera, and C. Hurley. Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics*, 4(3):155–172, 1995. URL https://doi.org/10.1080/10618600.1995.10474674. [p4]

D. Cook, A. Buja, E.-K. Lee, and H. Wickham. Grand tours, projection pursuit guided tours, and manual controls. In *Handbook of Data Visualization*, pages 295–314. Springer, 2008. URL https://doi.org/10.1007/978-3-540-33037-0_13. [p1, 3]

P. Diaconis and D. Freedman. Asymptotics of graphical projection pursuit. *The annals of statistics*, pages 793–815, 1984. [p2]

J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 100(9):881–890, 1974. URL https://doi.org/10.1109/T-C.1974.224051. [p1, 2]

P. Hall et al. On polynomial-based projection indices for exploratory projection pursuit. *The Annals of Statistics*, 17(2):589–605, 1989. URL https://doi.org/10.1214/aos/1176347127. [p2]

K. Healy. *Data visualization: a practical introduction*. Princeton University Press, 2018. ISBN 978-0691181622. URL https://socviz.co/. [p1]

P. J. Huber. Projection pursuit. *The annals of Statistics*, pages 435–475, 1985. [p2]

S. G. Johnson. *The NLopt nonlinear-optimization package*, 2020. URL https://github.com/stevengj/nlopt. [p2]

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998. URL https://doi.org/10.1023/A:1008306431147. [p1]

C. T. Kelley. *Iterative methods for optimization*. SIAM, 1999. URL https://doi.org/10.1137/1.9781611970920. [p1]

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. URL https://doi.org/10.1126/science.220.4598.671. [p4]

U. Laa and D. Cook. Using tours to visually investigate properties of new projection pursuit indexes with application to problems in physics. *Computational Statistics*, pages 1–35, 2020. URL https://doi.org/10.1007/s00180-020-00954-8. [p3, 16]

M. Li, Z. Zhao, and C. Scheidegger. Visualizing neural networks with the grand tour. *Distill*, 2020. URL https://doi.org/10.23915/distill.00025. [p1]

N. Loperfido. Skewness-based projection pursuit: A computational approach. *Computational Statistics and Data Analysis*, 120(C):42–57, 2018. [p3]

N. Loperfido. Kurtosis-based projection pursuit for outlier detection in financial time series. *The European Journal of Finance*, 26(2-3):142–164, 2020. [p3]

C. Posse. Projection pursuit exploratory data analysis. *Computational Statistics & Data Analysis*, 20(6): 669–687, 1995. URL https://doi.org/10.1016/0167-9473(95)00002-8. [p3, 4]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL https://www.R-project.org/. [p2]

B. Schloerke. *geozoo: Zoo of Geometric Objects*, 2016. URL https://CRAN.R-project.org/package=geozoo. R package version 0.5.1. [p8]

S. Theussl, F. Schwendinger, and H. W. Borchers. *CRAN Task View: Optimization and Mathematical Programming*, 2020. URL https://cran.r-project.org/web/views/Optimization.html. [p2]

J. W. Tukey. *Exploratory data analysis*, volume 2. Reading, MA, 1977. [p1]

A. Unwin. *Graphical data analysis with R*, volume 27. CRC Press, 2015. URL https://doi.org/10.1201/9781315370088. [p1]

H. Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1): 3–28, 2010. URL https://doi.org/10.1198/jcgs.2009.07098. [p5]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. URL https://doi.org/10.1007/978-0-387-98141-3. [p5]

H. Wickham, D. Cook, H. Hofmann, and A. Buja. tourr: An R package for exploring multivariate data with projections. *Journal of Statistical Software*, 40(2):1–18, 2011. URL http://doi.org/10.18637/jss.v040.i02. [p3, 14, 16]

H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2020. URL https://CRAN.R-project.org/package=dplyr. R package version 1.0.2. [p5]

H. Wickham et al. Tidy data. *Journal of Statistical Software*, 59(10):1–23, 2014. URL https://doi.org/10.18637/jss.v059.i10. [p5]

C. O. Wilke. *Fundamentals of data visualization: a primer on making informative and compelling figures*. O'Reilly Media, 2019. ISBN 978-1492031086. URL https://clauswilke.com/dataviz/. [p1]

Y. Xiang, S. Gubian, B. Suomela, and J. Hoeng. Generalized Simulated Annealing for Global Optimization: The GenSA Package. *The R Journal*, 5(1):13–28, 2013. URL https://doi.org/10.32614/RJ-2013-002. [p2]

Y. Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. URL https://yihui.name/knitr/. ISBN 978-1498716963. [p16]

Y. Xie, J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018. URL https://bookdown.org/yihui/rmarkdown. ISBN 978-1138359338. [p16]

Appendix

Three algorithms (Creeping random search, simulated annealing, and pseudo derivative) used in projection pursuit guided tour optimisation.

Algorithm 1: Creeping random search (CRS)

input : $f(.)$, α_1, $l_{\max}$, cooling
output: $\mathbf{A}_l$

Generate random start $\mathbf{A}_1$ and set $\mathbf{A}_{\text{cur}} \coloneqq \mathbf{A}_1$, $I_{\text{cur}} = f(\mathbf{A}_{\text{cur}})$, $j = 1$;
repeat
  Set $l = 1$;
  repeat
    Generate $\mathbf{A}_l = (1 - \alpha_j)\mathbf{A}_{\text{cur}} + \alpha_j \mathbf{A}_{\text{rand}}$ and orthogonalise $\mathbf{A}_l$;
    Compute $I_l = f(\mathbf{A}_l)$;
    Update $l = l + 1$;
  until $l > l_{\max}$ *or* $I_l > I_{cur}$;
  Update $\alpha_{j+1} = \alpha_j *$ cooling;
  Construct the geodesic interpolation between $\mathbf{A}_{\text{cur}}$ and $\mathbf{A}_l$;
  Update $\mathbf{A}_{\text{cur}} = \mathbf{A}_l$ and $j = j + 1$;
until $\mathbf{A}_l$ *is too close to* $\mathbf{A}_{cur}$ *in terms of geodesic distance*;

Algorithm 2: Simulated annealing (SA)

repeat
  Generate $\mathbf{A}_l = (1 - \alpha_j)\mathbf{A}_{\text{cur}} + \alpha_j \mathbf{A}_{\text{rand}}$ and orthogonalise $\mathbf{A}_l$;
  Compute $I_l = f(\mathbf{A}_l)$, $T(l) = \frac{T_0}{\log(l+1)}$ and $P = \min\left\{\exp\left[-\frac{I_{\text{cur}} - I_l}{T(l)}\right], 1\right\}$;
  Draw U from a uniform distribution: $U \sim$ Unif(0, 1);
  Update $l = l + 1$;
until $l > l_{\max}$ *or* $I_l > I_{cur}$ *or* $P > U$;

Algorithm 3: Pseudo derivative (PD)

repeat
  Generate n random directions $\mathbf{A}_{\text{rand}}$;
  Compute $2n$ candidate bases deviate from $\mathbf{A}_{\text{cur}}$ by an angle of δ while ensuring orthogonality;
  Compute the corresponding index value for each candidate bases;
  Determine the search direction as from $\mathbf{A}_{\text{cur}}$ to the candidate bases with the largest index value;
  Determine the step size via optimising the index value on the search direction over a 90 degree window;
  Find the optimum $\mathbf{A}_{**}$ and compute $I_{**} = f(\mathbf{A}_{**})$, $p_{\text{diff}} = (I_{**} - I_{\text{cur}})/I_{**}$;
  Update $l = l + 1$;
until $l > l_{\max}$ *or* $p_{diff} > 0.001$;

H.Sherry Zhang
Monash University
Department of Econometrics and Business Statistics

huize.zhang@monash.edu

Dianne Cook
Monash University
Department of Econometrics and Business Statistics

dicook@monash.edu

Ursula Laa
University of Natural Resources and Life Sciences
Institute of Statistics

ursula.laa@boku.ac.at

Nicolas Langrené
CSIRO Data61
34 Village Street, Docklands VIC 3008 Australia

nicolas.langrene@csiro.au

Patricia Menéndez
Monash University
Department of Econometrics and Business Statistics

patricia.menendez@monash.edu