Visual diagnostics for constrained optimisation with application to guided tours

Author 1 *
Department of YYY, University of XXX
and
Author 2
Department of ZZZ, University of WWW

October 13, 2020

Abstract

Friedman & Tukey commented on their initial paper on projection pursuit in 1974 that "the technique used for maximising the projection index strongly influences both the statistical and the computational aspects of the procedure." While many projection pursuit indices have been proposed in the literature, few concerns the optimisation procedures. In this paper, we illustrates the diagnostics of optimisers for projection pursuit guided tour via the construction of a data object and producing diagnostic plots. These diagnostics workflow can be applied generally on optimisers to assess its performance. An R package, ferrn, has been created to implement the diagnostics.

Keywords: optimisation, projection pursuit, guided tourr, visual, diagnostics, R

*The authors gratefully acknowledge ...

1 Introduction

Visualisation has been widely used in exploratory data analysis. Presenting information in a graphical format often allows people to see information they would otherwise not see. This motivates our work of creating plots to diagnose optimisation algorithms with the aim to understand and compare features of different algorithms.

In an optimization problem the goal is to find the best solution within the space of all feasible solutions, which typically is represented by a set of constraints. The problem consists in optimizing an objective function $f : S \to \mathbb{R}$ with $S \in \mathbb{R}^n$ in a reduced space given by the problem constraints to either minimize or maximize a function.

The problem context of the optimisation that has been worked in this paper is projection pursuit guided tour, which are exploratory data analysis tools that detect interesting structure of high dimensional data through projection on low dimensional space. Optimisation is applied here to search for the low dimensional space that finds the most interesting projection.

The remainder of the paper is organised as follows. Section 2 provides a literature review of optimisation methods, specifically the line search methods. Section 3 reviews projection pursuit guided tour, forms the optimisation problem and its features, and introduces three main existing algorithms. Section 4 presents new visual diagnostics designs. Data object is first defined to pinned down the data structure, several diagnostic plots are then defined with sample examples given. Section 5 shows the application of how the diagnostic plots designed previously can be used to understand and compare different algorithms and how they contribute to modifications that improve the algorithms. Finally, Section 6 describes the R package: ferrn, that implements all the visual diagnostics above.

2 Optimisation Methods

Given an optimisation problem, two broad approaches find the optimum based on different thinking and hence have its own strength and deficits. An analytical approach aims to find the optimal solution in a finite number of steps, but the closed-form solution may not be available when the problem starts to become complex. An iterative approach, on the

other hand, finds the optimum based on the idea of making progressive improvement to the current solution and researchers can decide when to stop if a desirable accuracy has been achieved. However, a potential issue is that it may end up finding a local optimum. In this paper, the main focus is on a particular iterative method: *line search method* (Fletcher 2013). In a simple one-dimensional problem of finding the x that minimises $f(x)$, line search conducts its search via an iterative up as in Equation 1.

$$x^{(j+1)} = x^{(j)} + \alpha_k * d^{(j)} \tag{1}$$

where $d^{(j)}$ is the searching direction in iteration j, and α_j is the step-size. Theoretical results have demonstrated the global convergence of the algorithm when α_k is chosen by minimising $f(x^{(j)} + \alpha * d^{(j)})$ (Curry 1944). However, in practice, this is rarely implemented due to its computational demanding or sometimes the existence of such a minimisation(Fletcher 2013). A modified approach is to impose a mandatory decrease in the objective function for each iteration: $f^{(j+1)} > f^{(j)}$ and this approach turns out to be efficient in practical problems despite the lost of guarantee on global convergence.

3 Projection pursuit guided tour

Modern development of the line search methods focuses on proposing different computation on the searching direction: $d^{(j)}$ and various approximations on the step size: α_j catered for practical optimisation problems. The specific problem context discussed in this paper is projection pursuit guided tour. Projection pursuit and guided tour are two separate methods in exploratory data analysis focusing on different aspects: Projection pursuit, coined by Friedman & Tukey (1974), detects interesting structures (i.e. clustering, outliers and skewness) in multivariate data via low dimensions projection whilst guided tour is one particular variation in a broader class of data visualisation method, tour.

Let $\mathbf{X}_{n\times p}$ be the data matrix, an n-d projection can be seen as a linear transformation $T : \mathbb{R}^p \mapsto \mathbb{R}^d$ defined by $\mathbf{Y} = \mathbf{X} \cdot \mathbf{A}$, where $\mathbf{Y}_{n\times d}$ is the projected data and $\mathbf{A}_{p\times d}$ is the projection basis. Define $f : \mathbb{R}^{n\times d} \mapsto \mathbb{R}$ to be an index function that maps the projection basis $\mathbf{A}$ onto an index value I, this function is commonly known as the projection pursuit

index (PPI) function, or the index function and is used to measure the "interestingness" of a projection. A number of index functions have been proposed in the literature to detect different data structures, including Legendre index (Friedman & Tukey 1974), Hermite index (Hall et al. 1989), natural Hermite index (Cook et al. 1993), chi-square index (Posse 1995), LDA index (Lee et al. 2005) and PDA index (Lee & Cook 2010).

As a general visualisation method, A tour produces animated visualisation of the high dimensional data via rotating low dimension planes. Different tours choose the low dimensional planes differently, for example, a grand tour(Cook et al. 2008) selects the planes randomly in the high dimensional space and manual tour(Cook & Buja 1997) chooses a sequence of bases that gradually phase in and out one variable to view the contribution of that variable in the visualisation. Guided tour, the main interest of this paper, chooses its projection with the aid of projection pursuit. Given a random start, projection pursuit iteratively finds bases with higher index value and guided tour constructs geodesic interpolation between two consecutive to form a tour path. Mathematical details of the geodesic interpolation can be found in Buja et al. (2005). Figure 1 shows a sketch of the tour path consisting of the blue frames produced by the projection pursuit optimisation algorithm iteratively and the white frames, the interpolations between two blue frames. The tour method has been implemented in the `tourr` package in R, available on the Comprehensive R Archive Network at `https://cran.r-project.org/web/packages/tourr/` (Wickham et al. 2011).

3.1 Optimisation in the tour

The optimisation problem is stated as follows. Given a randomly generated starting basis $\mathbf{A}_1$, projection pursuit finds the final projection basis $\mathbf{A}_T$ that satisfies the following optimisation problem:

$$\arg\max_{\mathbf{A}\in\mathcal{A}} f(\mathbf{X}\cdot\mathbf{A}) \tag{2}$$

$$s.t. \mathbf{A}'\mathbf{A} = I_d \tag{3}$$

where I_d is the d-dimensional identity matrix and the constraint requires the projection

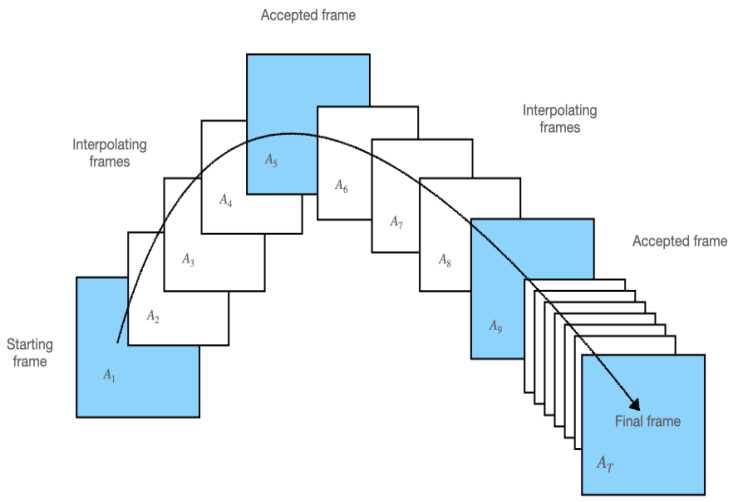

Figure 1: Each square (frame) represents the projected data with a corresponding basis. Blue frames are found by an optimisation algorithm iteratively whilst the white frames are constructed between two blue frames by geodesic interpolation.

bases $\mathbf{A}$ to be an orthogonal matrices.

Several features of this optimisation are worth noticing. First of all, this is a constraint optimisation problem as the decision variables form the entries of a projection basis, which is required to be orthonormal. It is also likely that the objective function may not be differentiable for a constructed index function and in these cases, gradient-based methods may not work well. Although finding the global maximum is the goal of an optimisation problem, it is also interesting to inspect local maximum in projection pursuit since it could present unexpected interesting projections. Lastly, there is also one computational consideration: the optimisation procedure needs to be fast to compute since the tour animation is played in real-time.

3.2 Existing algorithms

Below we introduce three possible algorithms: `search_better`, `search_better_random`, and `search_geodesic`. The first two are derivative free methods that sample candidate bases in the neighbourhood whilst `search_geodesic` is an analogue of gradient ascent on the projection basis space.

Algorithm 1: random search

input : $\mathbf{A}_{\text{cur}}$, f, α, $l_{\max}$

output: $\mathbf{A}_l$

initialisation;
Set $l = 1$;
while $l < l_{\max}$ **do**
    Generate $\mathbf{A}_l = (1-\alpha)\mathbf{A}_{\text{cur}} + \alpha\mathbf{A}_{\text{rand}}$ and orthogonalise $\mathbf{A}_l$;
    Compute $I_l = f(\mathbf{A}_l)$;
    if $I_l > I_{cur}$ **then**
        return $\mathbf{A}_l$;
    end
    $l = l + 1$;
end

`search_better` is a random search device 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 a randomly generated matrix with the same dimension as $\mathbf{A}_{\text{cur}}$. $\mathbf{A}_l$ is then orthogonalised to ensure the orthonormal constraint is fulfilled. When a basis is found with index value higher than the current basis $\mathbf{A}_{\text{cur}}$, the search terminates and outputs the basis for guided tour to construct an interpolation path. The next iteration of search begins after adjusting α by a cooling parameter: $\alpha_{j+1} = \alpha_j * \text{cooling}$. The termination condition is when the maximum number of iteration $l_{\max}$ is reached. The algorithm of `search_better` is summarised in Algorithm 1. A slightly different cooling scheme has been proposed by Posse (1995) to include a halving parameter c. Rather than reducing the radius of the searching neighbourhood, α, at each iteration, Posse's design only adjust α if the last search takes more than c times to find an accepted basis to avoid the searching space being reduced too fast.

Simulated annealing (`search_better_random`) (Kirkpatrick et al. 1983, Bertsimas et al. (1993)) uses the same sampling process as `search_better` but allow a probabilistic acceptance of a basis with lower index value based on the annealing $T(l)$. Given an initial T_0,

Algorithm 2: simulated annealing

1 Compute $I_l = f(\mathbf{A}_l)$ and $T(l) = \frac{T_0}{\log(l+1)}$;
2 **if** $I_l > I_{cur}$ **then**
3 **return** $\mathbf{A}_l$;
4 **else**
5 Compute $P = \min\left\{\exp\left[-\frac{I_{\text{cur}} - I_l}{T(l)}\right], 1\right\}$;
6 Draw U from a uniform distribution: $U \sim \text{Unif}(0, 1)$;
7 **if** $P > U$ **then**
8 **return** $\mathbf{A}_l$;
9 **end**
10 **end**

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, simulated annealing gives it a second chance to be accepted with probability

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

where $I_{(\cdot)}$ denotes the index value of a given basis. This implementation allows the algorithm to jump out of a local maximum and enables a more holistic search of the whole parameter space. This feature is particularly useful when local maximum is presented. The algorithm can be written as replacing line 5-8 of Algorithm 1 with Algorithm 2.

Cook et al. (1995) used a gradient ascent algorithm on the space of the projection bases. In gradient ascent, one first find the direction for improvement via computing the gradient information. In `search_geodesic`, $2n$ bases are first generated in a tiny neighbourhood of the current basis, controlled by the neighbourhood parameter δ. A geodesic is then constructed using the current basis and the one in $2n$ bases with the highest index value. If the neighbourhood parameter δ is tiny, the geodesic constructed is an analogue of the gradient information in the curved space and works as the searching direction. The next step in gradient ascent is to conduct a line search to find the best improvement along the gradient direction and in `search_geodesic`, this is replaced by optimising the index value along the geodesic direction over an 90 degree angle from $-\pi/4$ to $\pi/4$. The optima

Algorithm 3: search geodesic

input : $\mathbf{A}_{\text{cur}}$, f, $l_{\max}$, $n = 5$, δ

output: $\mathbf{A}_{**}$

initialisation;

Set $l = 1$;

while $l < l_{\max}$ **do**

  Generate $2n$ bases in a small neighbourhood, δ, of $\mathbf{A}_{\text{cur}}$ and ensure orthogonality ;

  Find the one with the largest index value: $\mathbf{A}_{*}$;

  Construct the geodesic $\mathcal{G}$ from $\mathbf{A}_{\text{cur}}$ to $\mathbf{A}_{*}$;

  Optimise the index value on the geodesic $\mathcal{G}$ over a 90 degree window to produce the optima $\mathbf{A}_{**}$;

  Compute $I_{**} = f(\mathbf{A}_{**})$, $p_{\text{diff}} = (I_{**} - I_{\text{cur}})/I_{**}$;

  if $p_{diff} > 0.001$ **then**

    return $\mathbf{A}_{**}$;

  end

  $l = l + 1$;

end

$\mathbf{A}_{**}$ is returned for the current iteration if it meets the termination condition on percentage improvement. The procedure will also terminate if $l_{\max}$ is reached. Algorithm 3 summarises the steps in geodesic search.

4 Visual diagnostics

To be able to make diagnostics on the optimisers, the algorithms need to populate a data structure with key elements of the algorithm. When the algorithms run, key information regarding the decision variable, objective function and hyper-parameters, needs to be recorded and stored as a data object so that it is ready to be supplied to the plotting functions for diagnostics.

4.1 Data structure for diagnostics

In the optimisation algorithms for projection pursuit, the three main elements to record are 1) projection bases: $\mathbf{A}$, 2) index values: I, and 3) State: S, which labels the observation with detailed stage in the optimisation. Possible values for `search_better` and `search_better_random` includes `random_search`, `new_basis`, and `interpolation`. `search_geodesic` has a wider variety that includes `new_basis`, `direction_search`, `best_direction_search`, `best_line_search`, and `interpolation`.

Multiple iterators are also needed to index the data collected at different levels. t is a unique identifier that prescribes the natural ordering of each observation; j is the counter for each search-and-interpolate iteration, which remains the same within one round and has an increment of one once a new round starts. l is the counter for each search/interpolation, which provides the information of how many basis the algorithm has searched before finding one to return. There are other parameters of interest, depends on the particular problem content and they are denoted as V_p. Two most common examples include V_1 = method, which tags the name of the algorithm used, and V_2 = alpha, the neighbourhood parameter that controls the size in sampling candidate bases. A matrix notation of the data structure is presented in Equation 4.

$$
\left[\begin{array}{c|c|cc|cc|cc}
t & \mathbf{A} & I & S & j & l & V_1 & V_2 \\
\hline
1 & \mathbf{A}_1 & I_1 & S_1 & 1 & 1 & V_{11} & V_{12} \\
\hline
2 & \mathbf{A}_2 & I_2 & S_2 & 2 & 1 & V_{21} & V_{22} \\
3 & \mathbf{A}_3 & I_3 & S_3 & 2 & 2 & V_{31} & V_{32} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & l_2 & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & 2 & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & 2 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 2 & k_2 & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & J & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
T & \mathbf{A}_T & I_T & S_T & J & l_J & V_{T1} & V_{T2} \\
\hline
\vdots & \vdots & \vdots & \vdots & J & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & J & k_J & \vdots & \vdots \\
\hline
\vdots & \vdots & \vdots & \vdots & J+1 & 1 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
T' & \mathbf{A}_{T'} & I_{T'} & S_{T'} & J+1 & l_{J+1} & V_{T'1} & V_{T'2}
\end{array}\right]
=
\left[\begin{array}{c}
\text{column name} \\
\hline
\text{search (start basis)} \\
\hline
\text{search} \\
\text{search} \\
\vdots \\
\text{search (accepted basis)} \\
\hline
\text{interpolate} \\
\text{interpolate} \\
\vdots \\
\text{interpolate} \\
\hline
\vdots \\
\hline
\text{search} \\
\vdots \\
\text{search (final basis)} \\
\hline
\text{interpolate} \\
\vdots \\
\text{interpolate} \\
\hline
\text{search (no output)} \\
\vdots \\
\text{search (no output)}
\end{array}\right]
\tag{4}
$$

where $T' = T + k_J + l_{J+1}$. Note that there is no output in iteration $J + 1$ since the optimiser can't find a better basis and the algorithm terminates. In this notation, final basis found is A_T with the highest index value I_T.

The data structure constructed above meets the tidy data principle (Wickham et al. 2014) that requires each observation forms a row and each variable forms a column. With tidy data structure, data wrangling and visualisation have been significantly simplified by the well-developed packages like dplyr(Wickham et al. 2020) and ggplot2(Wickham 2016).

The construction of diagnostic plots uses the concept of grammar of graphic (Wickham 2010) in ggplot2. In grammar of graphic, plots are not defined by its appearance (i.e. box-

plot, histogram, scatter plot etc) but by stacked layers. In the construction of diagnostic plots, there are multiple elements one may wants to emphasize and there's no single plot name that would meet the need, the stacked layers concept, on the other hand, allow information to be overlaid on the each other and one can build the plot from scratch as long as the variables have been stored in a dataset.

4.2 Check how hard the optimiser is working

A primary interest of diagnosing an optimiser is to study how it progressively finds its optimum. Directly plotting the index value across its natural order will cause the graph to be disproportional to the iteration since it usually takes longer for an optimiser to find a better basis towards the end. Another option is to use summarisation for each iteration. Boxplot is a suitable candidate that can provide five points summary of each iteration. Other additional information not presented in the boxplot and then be added with new layers, for example text information on the number of points can be added at the bottom of each iteration and the position of basis returned by projection pursuit can be highlighted in point. Further, an option to switch between displaying points and boxplot geometry is helpful when the number of observation is small in one iteration and this is achieved via a `cutoff` parameter.

Figure 2 shows a sample of the plot constructed using `search_better` with different value on the parameter `max_tries`. Comparing the returned index value of each iteration shows that the termination at `max_tries = 25` is not sufficient for `search_better` to explore the parameter space and a value of 500 is preferred over 25 in this context.

4.3 Examining the optimisation progress

Points on the interpolation path are another interest in tour since the projection on these bases will be played by the tour animation. Figure 3 presents the interpolation of three different tour paths each with different curvature. The leftmost plot shows an interpolation where the index value increases progressively and monotonically. The middle path has an increase-then-decreases pattern in the last two iterations and the rightmost path shows even a decrease in the index value at iteration three. The middle situation can be avoid via

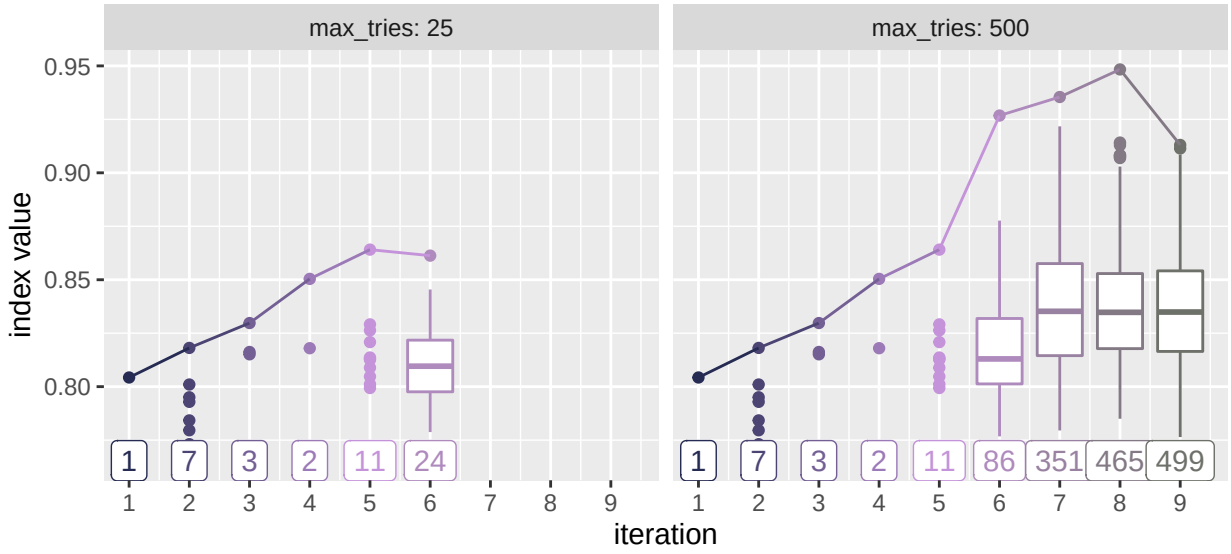

Figure 2: A comparison of `search_better` with different parameter value on `max_tries`. A six variable dataset `boa6` is used with `holes` index on a 2D problem. A `max_tries = 500` since it allows the optimiser to find better basis with higher index value at iteration six.

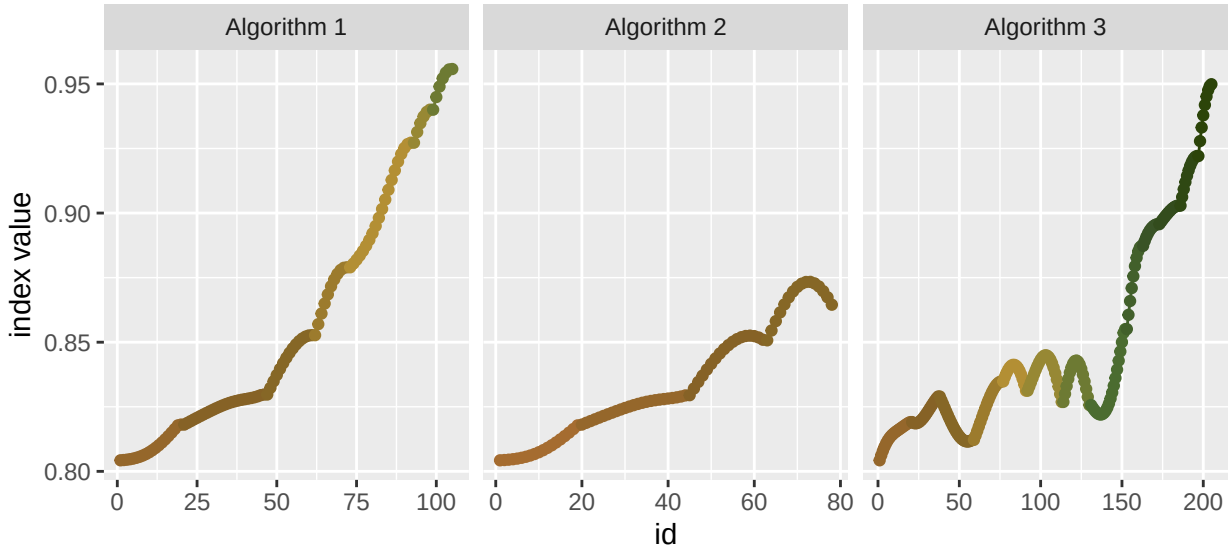

Figure 3: The resulting trace plot on the interpolated points has been plotted when using three different algorithms to optimise the index. The color represents the number of iteration. It can be observed that each algorithm differs in length in the optimisation and the curvature of the improvement for each algorithm also varies.

Figure 4: 1D projection on the 5 variable dataset `boa5` with two optimisers: `search_better` and `search_geodesic`. The yellow point corresponds to the theoretical best basis [0, 1, 0, 0, 0] with `V2` being the only non-normal variable in the dataset. The underlying grey points are randomly generated on the 5D space and reduced to 2D via PCA along with all the search points presented. The enlarged color points that colored as the interpolation are the starting points of each algorithm. They are initially simulated with the same starting points but all the bases in `search_geodesic` has been flipped positive to ensure that bases with same projected images but a sign difference are represented by the same point in the plot.

a construction of the interruption, which will be detailed in section 5.2 and the rightmost cases is a deliberate construction of `search_better_random` where an inferior basis can be accepted in a probability manner so as to avoid trapping in the local maximum.

4.4 Understanding the optimiser's coverage of the search space

Apart from checking the progression of an optimiser, another interesting aspect is to visualise how the search looks like in its parameter space. Given the orthonormality constraint, the projection bases $\mathbf{A}_{p\times d}$ lives on the surface of a $p \times d$ dimension sphere, where the dimension can easily go over two or three. While visualising the search paths on the original high

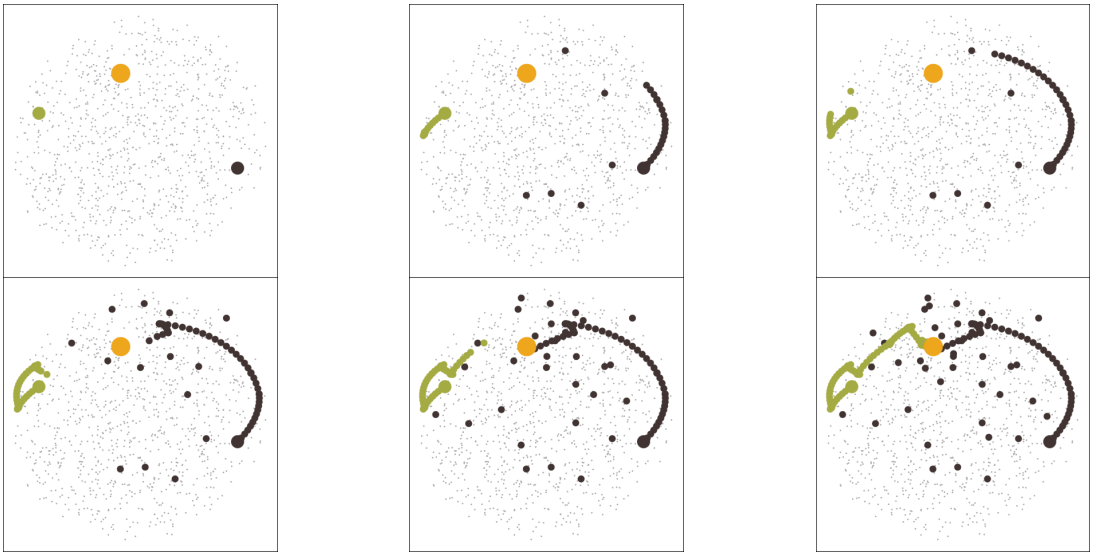

Figure 5: A selected number of frames from the animated PCA plot. With animation, it is easier to track the progression from the start to finish in each algorithm.

dimensional sphere would require skills for the viewers to perceive rotation of geometry in higher dimensional space (d $>$ 3), an easier alternative is to view the reduced space via some dimension reduction methods, i.e. principal component analysis. To better perceive the the search path as an embedding of a hollow sphere, random points on the high dimensional sphere is generated using the package `geozoo` and PCA is conducted on both the bases and the points on the surface of the sphere.

Figure 4 plots the first two principal components of two search paths, one using `search_better` and another uses `search_geodesic`. The search in the space reduced by PCA matches with the optimiser description before where the random sampling in `search_better` is broader, controlled by `alpha` parameter, which is default to 0.5 while the directional search in `search_geodesic`, controlled by `delta` with a default of 0.01, is so tiny that it can barely seen.

4.5 Animating the diagnostic plots

Animated plots can be informative in diagnostics, especially in the case of PCA plot when the starting and ending of the search is not clear. Figure 5 shows six frames of an animated

Figure 6: A selected number of frames from the tour animation for viewing the 5D space of all the projection bases. The second frame on the top row view the space from a direction that is close to the one in PCA plot. The tour animation allows for a more holistic view of the full space in high dimensions from different angles.

version of Figure 4 and this time, it shows that `search_better` finds the optimum quicker than `search_geodesic`.

4.6 The tour looking at itself

Viewing the bases on the reduced space via PCA shed some lights on the space the optimisers have explored, the visualisation on the original $p \times d$ dimension enables a more holistic stereoscopical view of the search. To view a high dimensional ($d \geq 3$) object on a screen, an approach is to play the rotation of the object in animation and this can be done via a regular grand tour. Compare to the PCA plot, the animated rotation (tour) display in Figure 6 gives a more well-rounded view of the search and one can view the curved region of the tour path from different angles, which may not be presented in the PCA plot. Also the grand tour animation encompasses the PCA projection since the rotation from PCA is just one angle that maximises the variance of the bases and grand tour produces a sequence of angle that view the search from different direction. As an evidence, the last frame in

Figure 6 is a frame select from the tour animation that is close to the PCA angle and the projection looks similar to one in Figure 4.

5 Diagnosing an optimiser

For a particular index function, the best algorithm to optimise relates to the character of the index and the data. If the index function is smooth and has single maximum, all of the three algorithms introduced above can find the maximum. When multiple optima are presented, `search_better` may stuck at the local maximum and in the case where the index function is non-smooth, `search_geodesic` may even fail to find the maximum. In this section, examples will be presented to outline how the diagnostic plots can be used to compare the performance of optimisers in different scenarios.

5.1 Simulation setup

Random variables with different structures has been simulated and the distribution of each is presented in Equation 5 to 11. Variable `x1`, `x8`, `x9` and `x10` are normal distributed with zero mean and unit variance and `x2` to `x7` are mixtures of normal distributions with varied weights and locations. The mixture variables have been scaled to have an overall unit variance before running the projection pursuit.

$$x_1 \stackrel{d}{=} x_8 \stackrel{d}{=} x_9 \stackrel{d}{=} x_{10} \sim \mathcal{N}(0,1) \tag{5}$$

$$x_2 \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1) \tag{6}$$

$$\Pr(x_3) = \begin{cases} 0.5 & \text{if } x_3 = -1 \text{ or } 1 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$x_4 \sim 0.25\mathcal{N}(-3,1) + 0.75\mathcal{N}(3,1) \tag{8}$$

$$x_5 \sim \frac{1}{3}\mathcal{N}(-5,1) + \frac{1}{3}\mathcal{N}(0,1) + \frac{1}{3}\mathcal{N}(5,1) \tag{9}$$

$$x_6 \sim 0.45\mathcal{N}(-5,1) + 0.1\mathcal{N}(0,1) + 0.45\mathcal{N}(5,1) \tag{10}$$

$$x_7 \sim 0.5\mathcal{N}(-5,1) + 0.5\mathcal{N}(5,1) \tag{11}$$

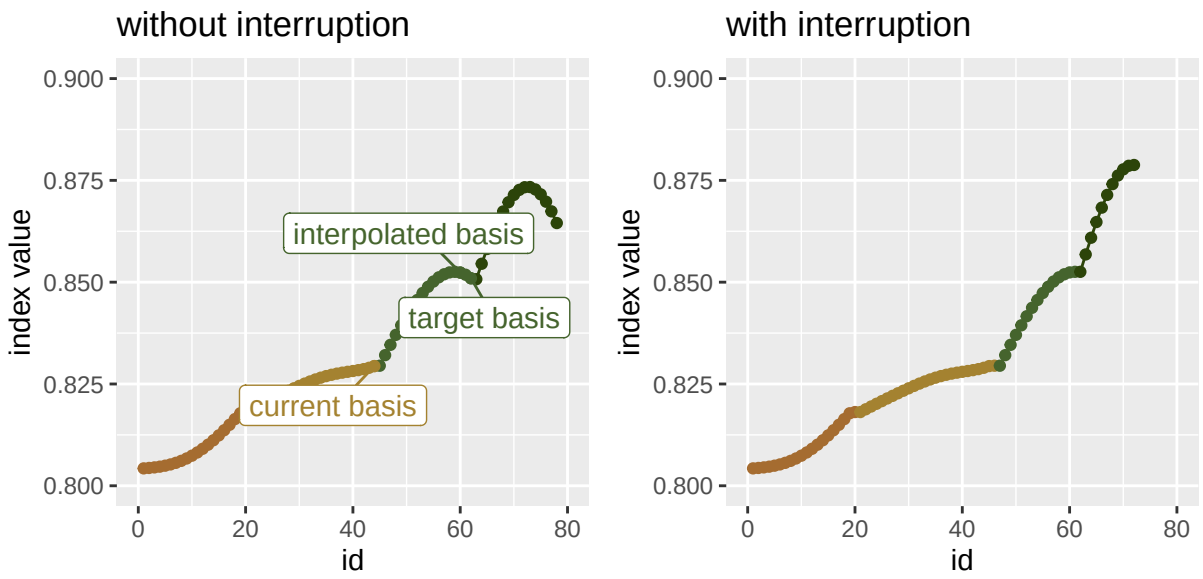

Figure 7: Two-D projection on `boa6` data with holes index optimised by `search_better`. A comparison of the trace plot with and without the interruption. When the interruption does not take place, the index value of the target basis can be smaller than the interpolated basis. The interruption forces the interpolation to finish at the highest interpolated basis on the tour path.

5.2 A problem of not monotonic

In section 4.3, an interpolation with increase-then-decrease pattern has been presented. This pattern is undesirable since the optimiser could have start the next iteration from the highest basis on the tour path, as annotated as the interpolated basis in the plot, but instead, it is forced to start from the target basis. This motivates the design of an interruption to check the index value on the tour path so that the interpolating bases is accepted only up to the one with the largest index value. After implementing this interruption, the search finds higher final index value with fewer steps as shown in the right panel of Figure 7.

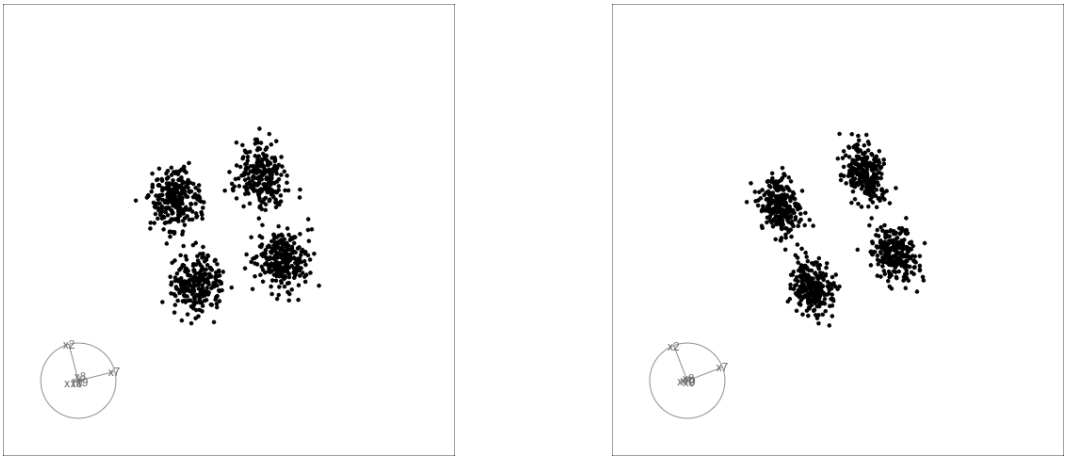

Figure 8: Two-D projection on `boa6` data with holes index optimised by `search_geodesic`. The left panel shows the final projected data before polish and the right panel shows the one after. The separation of the clusters on the y axis becomes sharper after the polish.

5.3 Close but not close enough

Once the final basis has been found by an algorithm, one may want to push further to investigate whether there's an even better basis in the close neighbourhood. This motivates the polish search where the final basis is supplied as the start of a new guided tour to search for any local breakthrough.

Similar to `search_better` as a stochastic random search, `search_polish` has a different scheme of reducing the search neighbourhood. In each search-interpolation iteration, `search_better` has a fixed neighbourhood parameter alpha and this alpha is reduced by the cooling parameter only after an iteration finishes. On the contrary, `search_polish` allows alpha to be reduced during each iteration to exploit the search in the neighbourhood. Further, to avoid the case where alpha becomes too small and the further search is meaningless, three more stopping criteria have been added, on top of the original `max.tries` limit. These include:

1) the distance between the candidate basis 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 need to be larger than 0.01

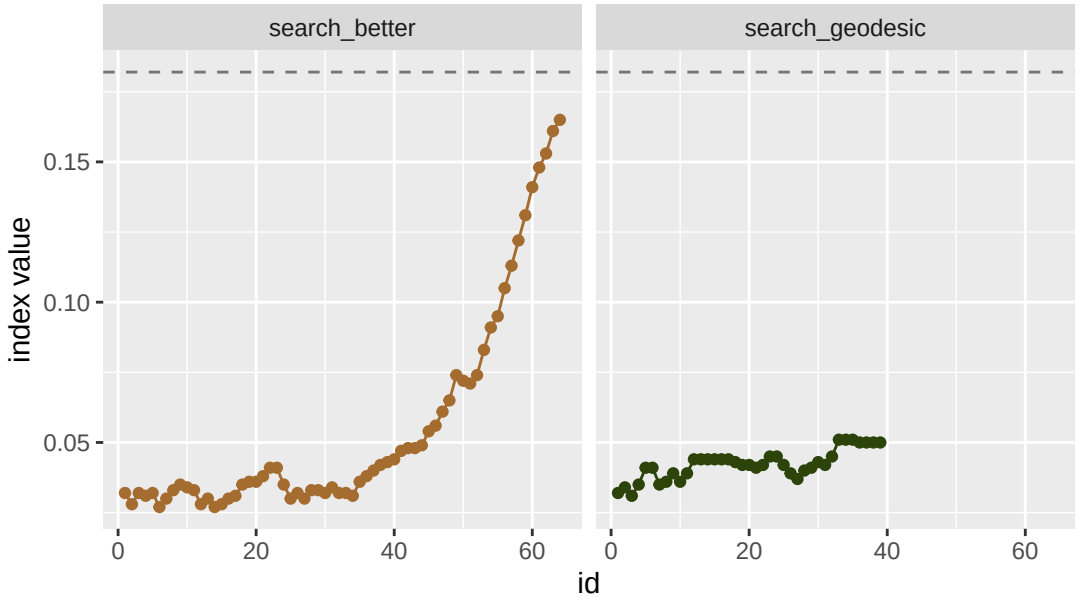

Figure 9: One-D projection on `boa5` data with noisy index `norm_kol` optimised by `search_geodesic` and `search_better`. The grey dashed line represents the index value of the theoretical best basis. `search_geodesic` fails to optimise the noisy index while `search_better` has made reasonable improvements to reach the index value close to the one of theoretical best basis.

Figure 8 presents the final projections found before and after applying `search_polish` on `search_geodesic`. Polish search improves the index value from 0.9618 to 0.9627 with reduction of weights on the non-informative variables. In terms of the projected data as in Figure 8, polish works to sharpen the edges of each cluster.

5.4 Seeing the signal in the noise

The index function, up until this point, are all smooth, while this is not the case for all the index functions. `norm_kol`, a 1D projection function based on the Kolmogorov test, compares the difference between the 1D projected data, $\mathbf{Y}_{n\times 1}$ and a randomly generated normal distribution, y_n based on the empirical cumulated distribution function (ECDF). Denotes the ECDF function as $F_{\cdot}(u)$ with the subscript indicating the projection or the random normal variable, the `norm_kol` index is defined by

$$\max\left[F_{\mathbf{P}}(u) - F_y(u)\right]$$

Figure 9 compares the tracing plot of two optimisers: `search_geodesic` and `search_better`. This time, the interpolated path is no longer smooth when using either algorithm and `search_geodesic` fails to optimise this index with barely improvement of the index value. On the other hand, `search_better` is doing a relatively well. With theoretical best basis [0, 1, 0, 0, 0] produces an index value of 0.182, `search_better` finds the final basis [0.0376, -0.9916, -0.0581, -0.0831, 0.0716] with an index value of 0.165. A further polish step will give a marginal improvement of index value to 0.175 with a basis of [0.0223, -0.9965, -0.0352, -0.0591, 0.0418]. At this stage, the difference between the theoretical best and what has been found is likely due to simulation error since the best possible basis for a simulated data will be slightly off the theoretical best basis, which is derived based on the distributional assumption in Equation 5 to 11.

The second experiment with the noisy index is to understand how the optimisers perform when the local maximum is presented. The dataset used is `boa6` where `x2` and `x7` are informative. The two theoretical best bases are [0, 1, 0, 0, 0, 0] and [0, 0, 1, 0, 0, 0] with index value 0.182 and 0.235, respectively. Hence, the global maximum happens when variable `x7` is found. Simulation is done with 20 randomly generated seeds on two optimisers: `search_better` (Figure 10) and `search_better_random` (Figure 11 and Figure 12). Compared to Figure 10 and 11, Figure 12 further increases the neighbourhood parameter `alpha` from 0.5 to 0.7 to enlarge the search space. The data object is collected for each simulation and computation is done to reduce the bases from its original 6D to a 2D points by principal component analysis so as to view the search paths.

In Figure 10, the starting point largely affects whether `search_better` will find the global maximum. The three seeds on row three: seed 9145, seed 2511, and seed 9209, and three on row four: seed 2888, seed 9334, and seed 9819 easily find `V7` because they are born in Rome. On the other end, the ones on the first two rows can only find `V2`, largely due to the fact that it doesn't even have chances to search near `V7`.

A comparison between `search_better` and `search_better_random` shows all four combinations where both, neither, or one of the two algorithms find the global maximum. Two

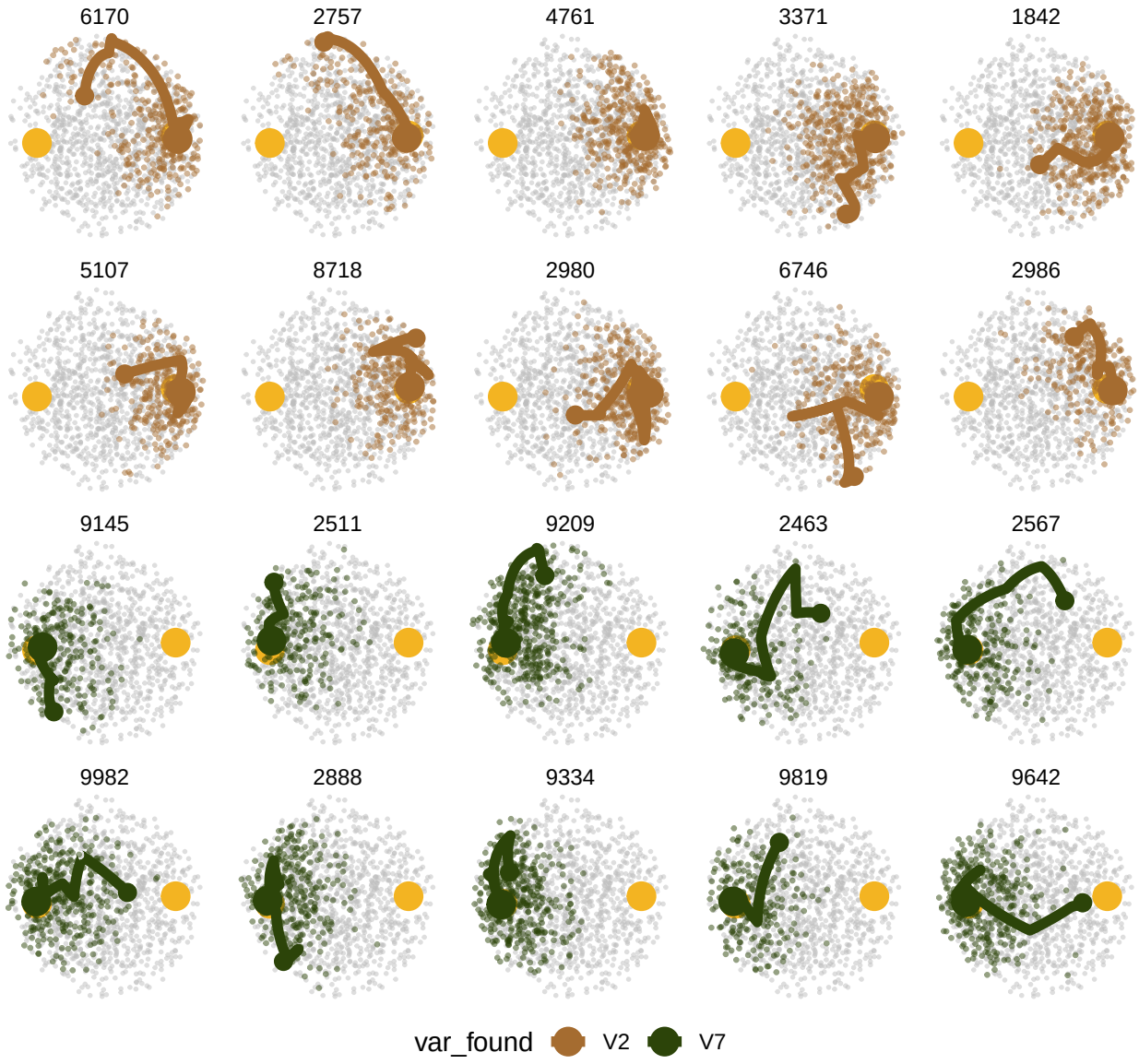

Figure 10: One-D projection of `norm_kol` index on `boa6` data optimised by `search_better` with 20 randomly generated seeds. Each point represents a basis in the original 6D space, reduced to 2D by PCA. The grey points are random bases generated from 6-D hollow sphere and the two yellow points represent the local maximum corresponds to basis when `V2` is found and the global maximum where `V7` is found. The points produced by the search algorithm is colored by whether the global or local maximum is found with the interpolated bases highlighted as a path and the final basis as an amplified point.

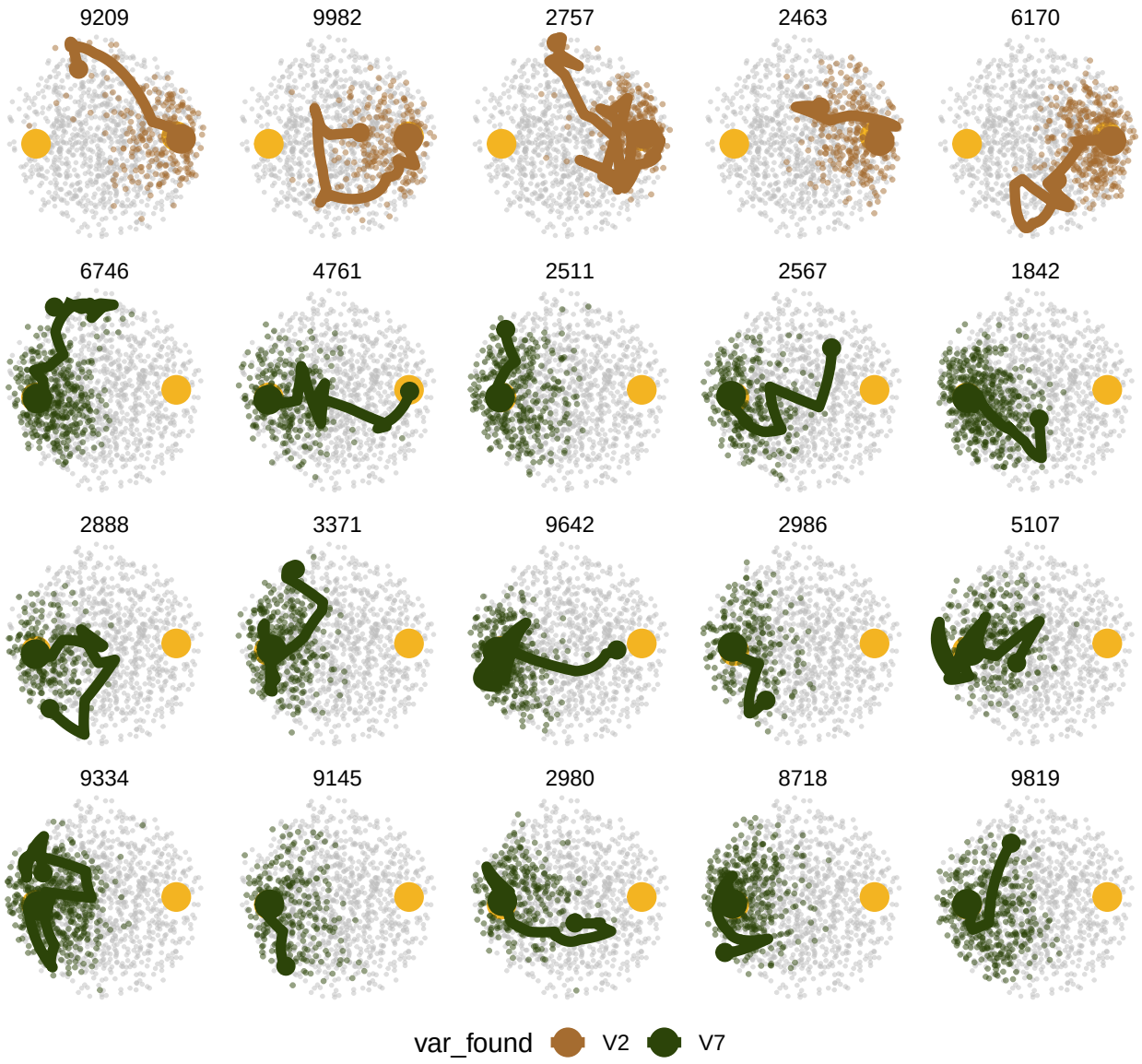

Figure 11: One-D projection of `norm_kol` index on `boa6` data optimised by `search_better_random` with the same seeds. `search_better_random` has a probabilistic acceptance implementation that would also a basis with lower index value. This design allows the optimiser to jump out of the local maximum and hence more incidents find the global maximum.

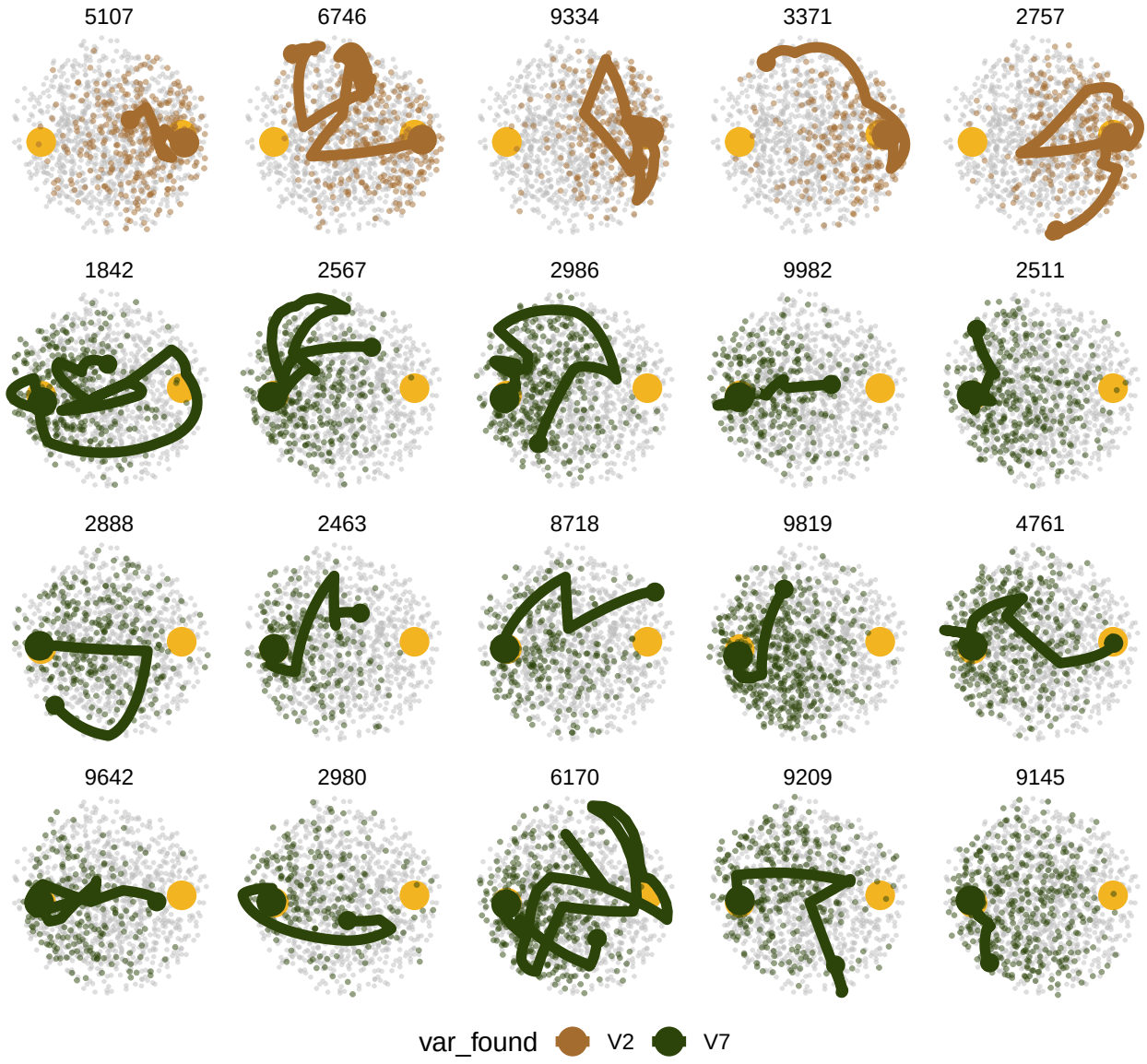

Figure 12: One-D projection of `norm_kol` index on `boa6` data optimised by `search_better_random` with an larger searching neighbourhood of 0.7. Further tuning of the parameter has been taken place to adjust the neighbourhood parameter `alpha` from 0.5 to 0.7.

iconic cases that shows how `search_better_random` improves `search_better` are seeds 4761 (position (1,4) in Figure 10 and (2,2) in Figure 11) and seed 1842 ((2,2) in Figure 10 and (4,5) in Figure 11). In both cases, `search_better_random` accepts inferior points at the initial iteration and this acceptance has later been proved to change the direction of the search and hence allows the optimiser to explore near the global optimum and in the end, find the global optimum. However, this change of search direction may sometimes have an adverse effect on `search_better_random` and change the search to the neighbourhood of `V2`. This happens on seed 9209 and seed 9982 (position (1,1) and (1,2) in Figure 11). The two cases that neither the algorithm finds the global optimum happens on seed 6170 and seed 2757 (position (1,1) and (1,2) in Figure 10 and position (1,5) and (1,3) in Figure 11). This is largely due to the fact that the searching neighbourhood is not large enough to allow the space near `V7` to be adequately explored.

In Figure **??**, the neighbourhood parameter `alpha` is increased from a 0.5 default to 0.7 to solve the insufficiency of search neighbourhood. Remember that a candidate basis is generated via a linear combination of the current basis and a randomly generated basis on the surface of the sphere, an increase of `alpha` gives more weights on the randomly generated basis and hence allows a wider search. Seed 9982 ((2,4) in Figure 12) and seed 2463 ((3,2) in Figure 12) benefit from this and find the global maximum. A noticeable feature of this enlarged search space is that the interpolation paths start to jump around the space but its usefulness is arguable. In seed 2888 (position (3,1)), seed 2986 (position (2,3)), seed 9209 (position (4,4)), just to name a few, the allows a near `V2` bases to switch to a near `V7` bases and results these cases to find the global maximum. While in the case of seed 6746 (position (1,2)) and seed 2757 (position (1,5)), the jump happens from near `V7` basis to near `V2` basis and it doesn't come back and leave those simulations in the local maximum.

The conclusion from this experiment is that the usage of `search_better_random` and increase the search space are methods that can avoid trapping in local maximum but the best solution will depends on the starting points of the simulation and the seed used.

6 Implementation

The implementation of this projection has been divided into two packages: the data collection object is implemented in the existing `tourr` package while the optimiser diagnostics have been implemented in a new package, `ferrn`. When a guided tour is run, the users can choose if the data from optimisation should be collected via the `verbose` argument. Once the data object has been obtained, the package, `ferrn`, can provide four diagnostic plots as shown in Section 4. The structure of package functionality has been listed below.

- `explore_trace_search()`: produces summary plots, as shown in Figure 2
- `explore_trace_interp()`: produces trace plots for the interpolation points, as shown in Figure 3
- `explore_space_pca()`: produces plots of projection basis on the reduced space by PCA, as shown 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, as shown in Figure 6.
- `get_*()` extracts and manipulates certain components from the existing data object.
  - `get_best()`: extracts the best basis found in the data object
  - `get_start()`: extracts the starting basis
  - `get_interp()`: extracts the observations in the interpolation
  - `get_search_count()`: produces the summary table of the number of observation in each iteration
  - `get_basis_matrix()`: flattens all the bases into a matrix
- `bind_*()` incorporates additional information outside the tour optimisation into the data object.
  - `bind_theoretical()`: incorporates the best possible basis to the existing data object with the supply of the index function and original data for producing the index value.

  - `bind_random()`: generates 1000 points on the high dimensional surface of a sphere and binds it to the existing data object and output as a tibble object. `bind_random_matrix()` binds the points to the basis matrix.

- Color

  - `botanical_palettes`: a collection of color palettes from Australian native plants. Quantitative palettes include daisy, banksia and cherry and sequential palettes contain fern and acacia.
  - `botanical_pal()`: a color interpolator
  - `scale_color_botanical()`: a ggplot construction for using botanical palettes.

7 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 orthornormality 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` package; 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 & Cook 2020) and understand how the optimisers behave for index functions with different structures.

References

Bertsimas, D., Tsitsiklis, J. et al. (1993), 'Simulated annealing', *Statistical science* **8**(1), 10–15.

Buja, A., Cook, D., Asimov, D. & Hurley, C. (2005), 'Computational methods for high-dimensional rotations in data visualization', *Handbook of statistics* **24**, 391–413.

Cook, D. & Buja, A. (1997), 'Manual controls for high-dimensional data projections', *Journal of computational and Graphical Statistics* **6**(4), 464–480.

Cook, D., Buja, A. & Cabrera, J. (1993), 'Projection pursuit indexes based on orthonormal function expansions', *Journal of Computational and Graphical Statistics* **2**(3), 225–250.

Cook, D., Buja, A., Cabrera, J. & Hurley, C. (1995), 'Grand tour and projection pursuit', *Journal of Computational and Graphical Statistics* **4**(3), 155–172.

Cook, D., Buja, A., Lee, E.-K. & Wickham, H. (2008), Grand tours, projection pursuit guided tours, and manual controls, *in* 'Handbook of data visualization', Springer, pp. 295–314.

Curry, H. B. (1944), 'The method of steepest descent for non-linear minimization problems', *Quarterly of Applied Mathematics* **2**(3), 258–261.

Fletcher, R. (2013), *Practical methods of optimization*, John Wiley & Sons.

Friedman, J. H. & Tukey, J. W. (1974), 'A projection pursuit algorithm for exploratory data analysis', *IEEE Transactions on computers* **100**(9), 881–890.

Hall, P. et al. (1989), 'On polynomial-based projection indices for exploratory projection pursuit', *The Annals of Statistics* **17**(2), 589–605.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983), 'Optimization by simulated annealing', *science* **220**(4598), 671–680.

Laa, U. & Cook, D. (2020), 'Using tours to visually investigate properties of new projection pursuit indexes with application to problems in physics', *Computational Statistics* pp. 1–35.

Lee, E., Cook, D., Klinke, S. & Lumley, T. (2005), 'Projection pursuit for exploratory supervised classification', *Journal of Computational and graphical Statistics* **14**(4), 831–846.

Lee, E.-K. & Cook, D. (2010), 'A projection pursuit index for large p small n data', *Statistics and Computing* **20**(3), 381–392.

Posse, C. (1995), 'Projection pursuit exploratory data analysis', *Computational Statistics & data analysis* **20**(6), 669–687.

Wickham, H. (2010), 'A layered grammar of graphics', *Journal of Computational and Graphical Statistics* **19**(1), 3–28.

Wickham, H. (2016), *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York.
URL: *https://ggplot2.tidyverse.org*

Wickham, H., Cook, D., Hofmann, H. & Buja, A. (2011), 'tourr: An R package for exploring multivariate data with projections', *Journal of Statistical Software* **40**(2), 1–18.
URL: *http://www.jstatsoft.org/v40/i02/*

Wickham, H., François, R., Henry, L. & Müller, K. (2020), *dplyr: A Grammar of Data Manipulation*. R package version 1.0.0.
URL: *https://CRAN.R-project.org/package=dplyr*

Wickham, H. et al. (2014), 'Tidy data', *Journal of Statistical Software* **59**(10), 1–23.