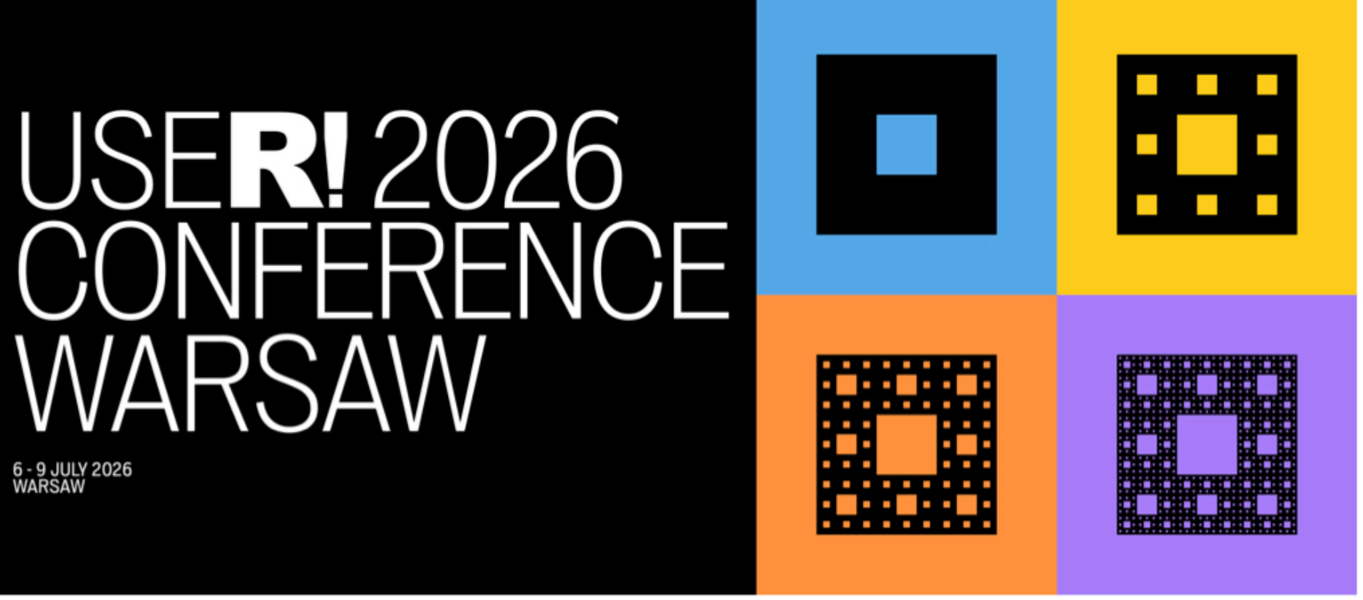


An Enhanced Projection Pursuit Tree Classifier with Visual Methods for Assessing Algorithmic Improvements

Natalia da Silva
Universidad de la República

Dianne Cook
Monash University

Eun-Kyung Lee
Ewha Womans University

Motivation

- `PPtree` is a Projection Pursuit Tree method for classification problems, but has some limitations we want to address.
- Extends `PPtree` by allowing more splits and more flexible class groupings in the projection pursuit computation.
- Includes visualization tools to understand the new method and compare it with others.
- Implemented in `PPtreeExt` package
- 

Available on CRAN

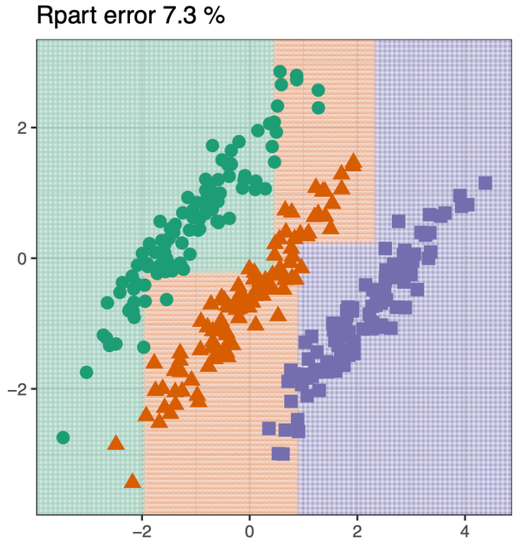
CART vs PPtree

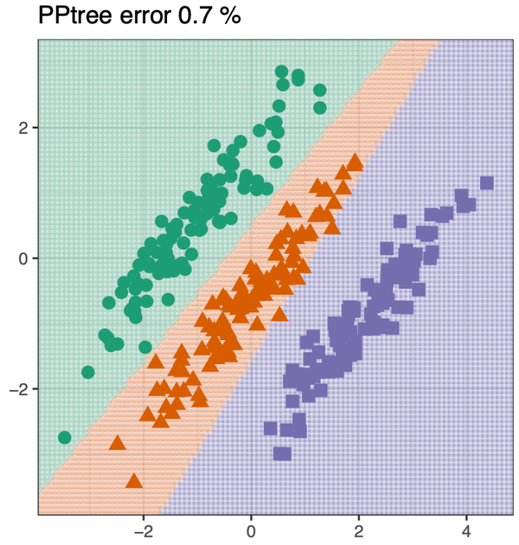
CART classifiers

- Split on a single variable at a time
- Rectangular, axis-aligned decision boundaries
- Need many splits to approximate an oblique boundary

PPtree classifiers

- Split on linear combinations of variables optimizing a PP index
- Define oblique boundaries

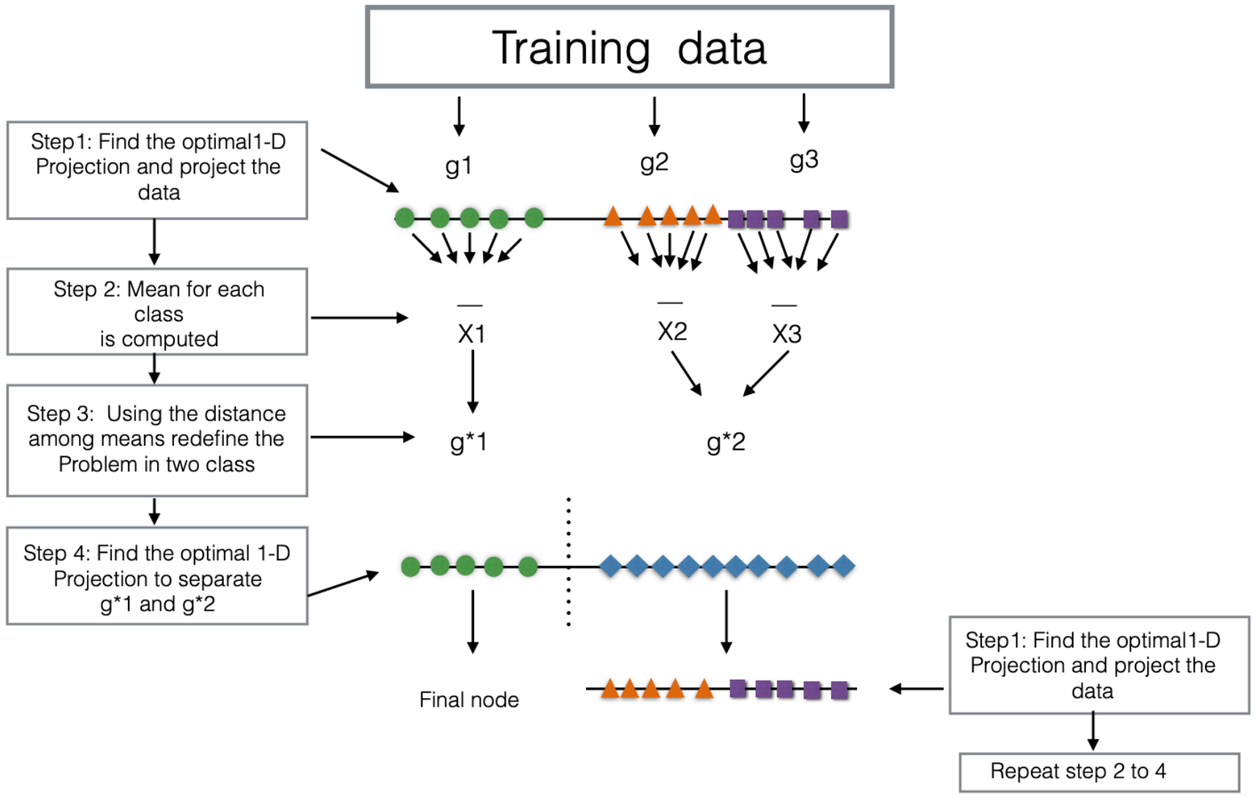




Projection Pursuit Classification Tree

- At each node, PPtree finds a **1-D linear projection** optimizing a **PP index** (e.g. LDA or PDA) that best separates the classes: $z_i = \alpha^T x_i$.

PPtree Algorithm Diagram

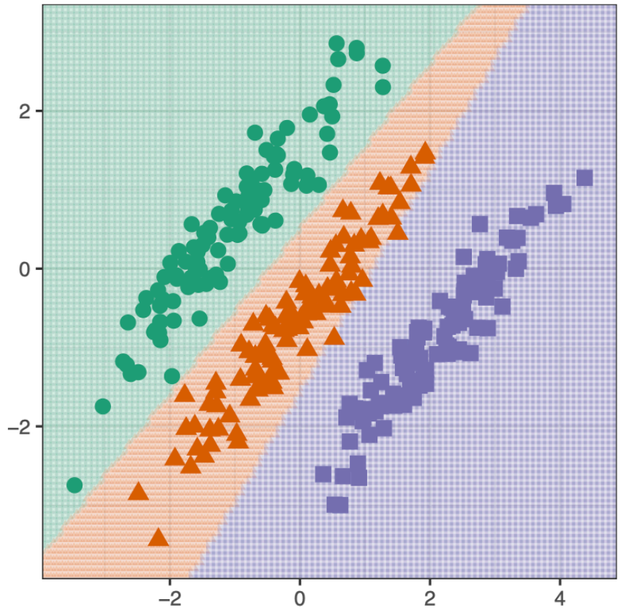


PPtree limitations

- **Problem 1 – Unequal variances**

Combining classes into a super-group creates unequal variance-covariance structure, boundaries are pulled towards one group

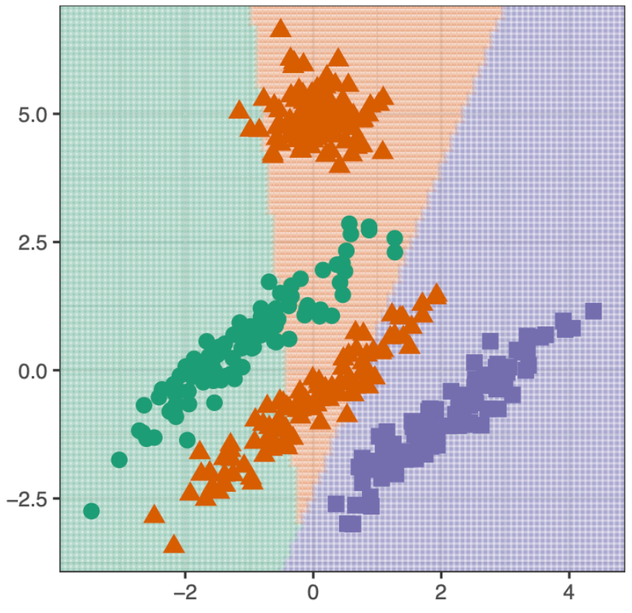
PPtree error 0.7 %



- **Problem 2 – Multiple clusters per class**

PPtree defines a rigid tree structure where the depth of the tree is at most $G - 1$. Cannot handle classes that have non-elliptical or multi-cluster shapes

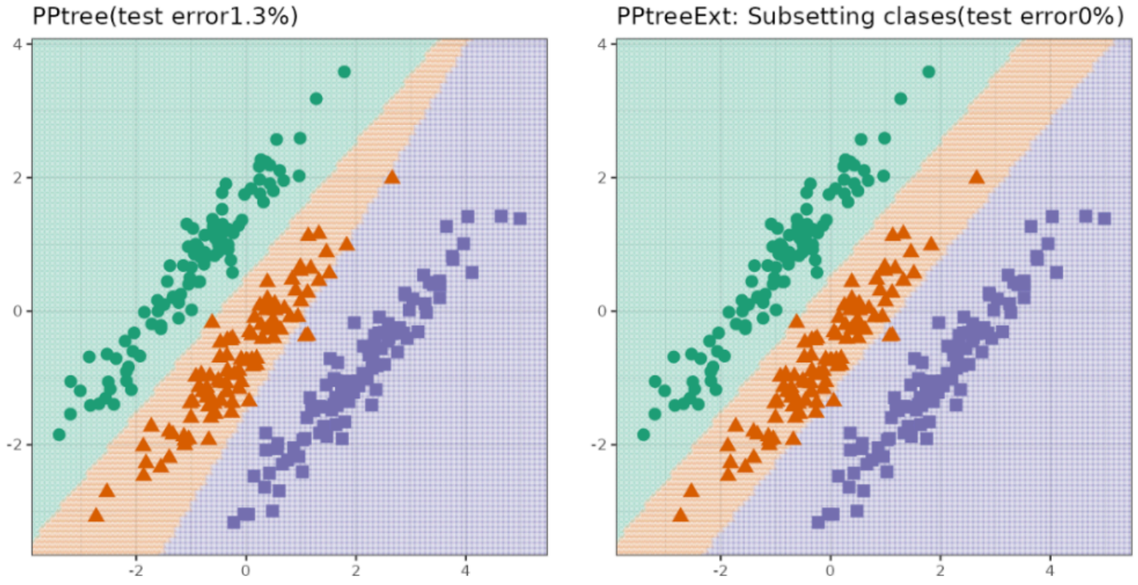
PPtree error 23.8 %

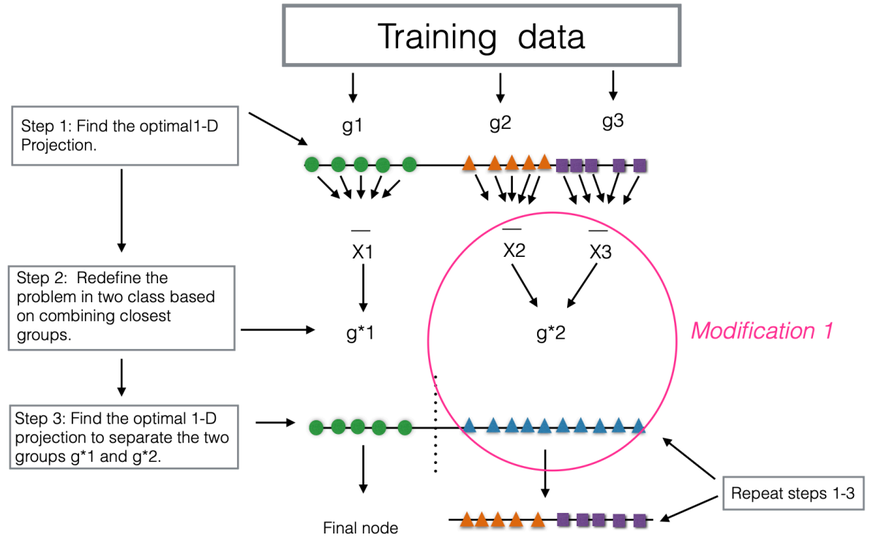


`PPtreeExt` address these limitations

Modification 1: Subsetting Classes

- **Original Step 2:** all remaining classes are combined into super-class g_2^*
- **Modified Step 2*:** only the **two closest classes** (smallest projected mean difference) are used to compute the next projection and split point. The class that was dropped is reintroduced in the next split.





> Subtle boundary change, but more principled and symmetric between class clusters

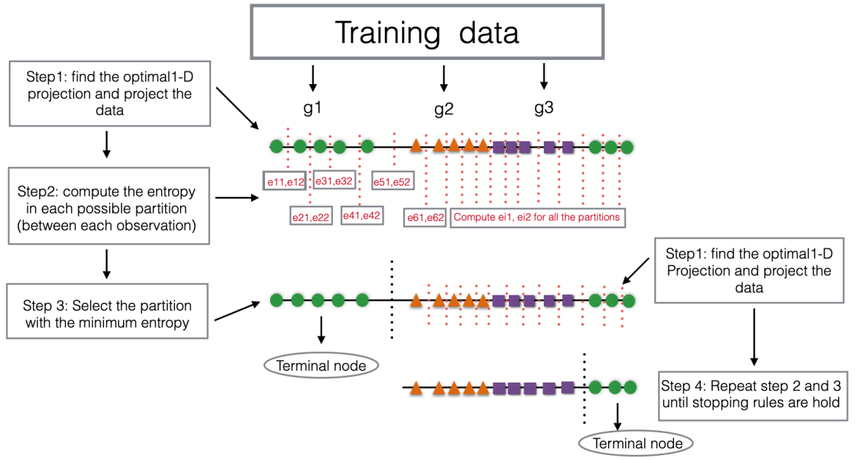
Modification 2: Multiple Splits via Entropy

Replaces the mean-based split with an **entropy minimization** criterion (like CART), applied on the **1-D projected data**:

$$E(s) = -\sum_{j=1}^{G} p_{js} \log(p_{js})$$

The best split minimizes the weighted impurity across left/right children:

$$E(s_L, s_R) = \frac{n_L}{n_L + n_R} E(s_L) + \frac{n_R}{n_L + n_R} E(s_R)$$



Interactive Shiny App for Debugging

`PPtreeExt` includes a **Shiny app** for side-by-side comparison of decision boundaries:

Three data generation methods:

- `Basic_Sim` – user-controlled class means and correlations
- `SIM_Outlier` – adds a separated cluster to one class
- `MixSim` – Gaussian mixtures with K components

We can explore the boundaries for `PPtreeExt` and compare with `rpart` and `PPtree` in different simulation scenarios.

```
# Run locally from the package
library(PPtreeExt)
explorapp()

# Or visit the hosted version:
# https://natyds.shinyapps.io/explorapp/
```

This app has been a valuable tool for debugging and evaluating the algorithmic behavior

and was essential for **developing** the modifications, not just presenting them.

Shiny app

Basic-Sim | SIM-Outliers | MixSim

Rule: 1

Modification: Multiple splits

Group means: -1, 0.6, 0, -0.6, 2,-1

Correlations: 0.95, 0.5, 0.75

Group sample: 100, 100, 100

Add outliers to class: 2

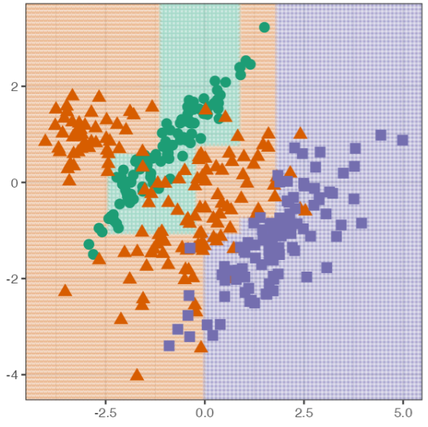
Out. X1, X2 means: -3, 1

Out. X1, X2 sd: .7,.4

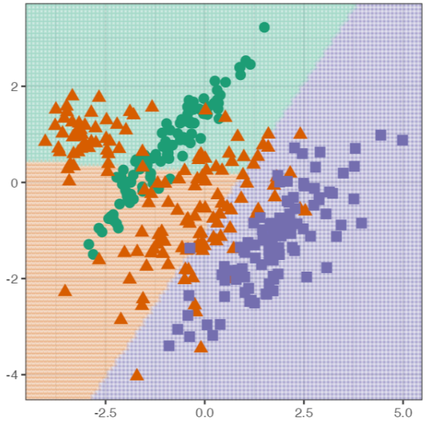
Out. sample size: 50

OK

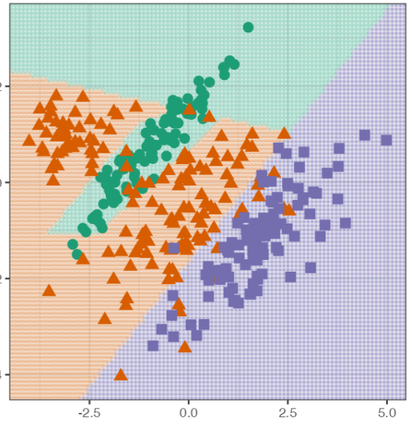
Rpart(test error25.3%)



PPtree(test error39.1%)



PPtreeExt: Multiple splits(test error11.5%)



Benchmark Study: 94 Datasets

Setup:

- 94 benchmark classification datasets
- Compared: CART, PPtree (LDA, PDA), Random Forest, PPforest, SVM, **PPtreeExt**
- 200 stratified train/test splits (2/3 – 1/3)
- Performance: average test error with 95% CI

Results (modification 2):

- Outperforms **all tree methods** in **18 of 94** datasets
- In **3 of those 18**, also beats ensemble methods (RF, PPforest, SVM)
- Strongest gains when:
  - Classes separate on linear *combinations* of variables
  - Classes have unequal variances or multiple clusters
  - High predictor correlation

Why Does It Win?

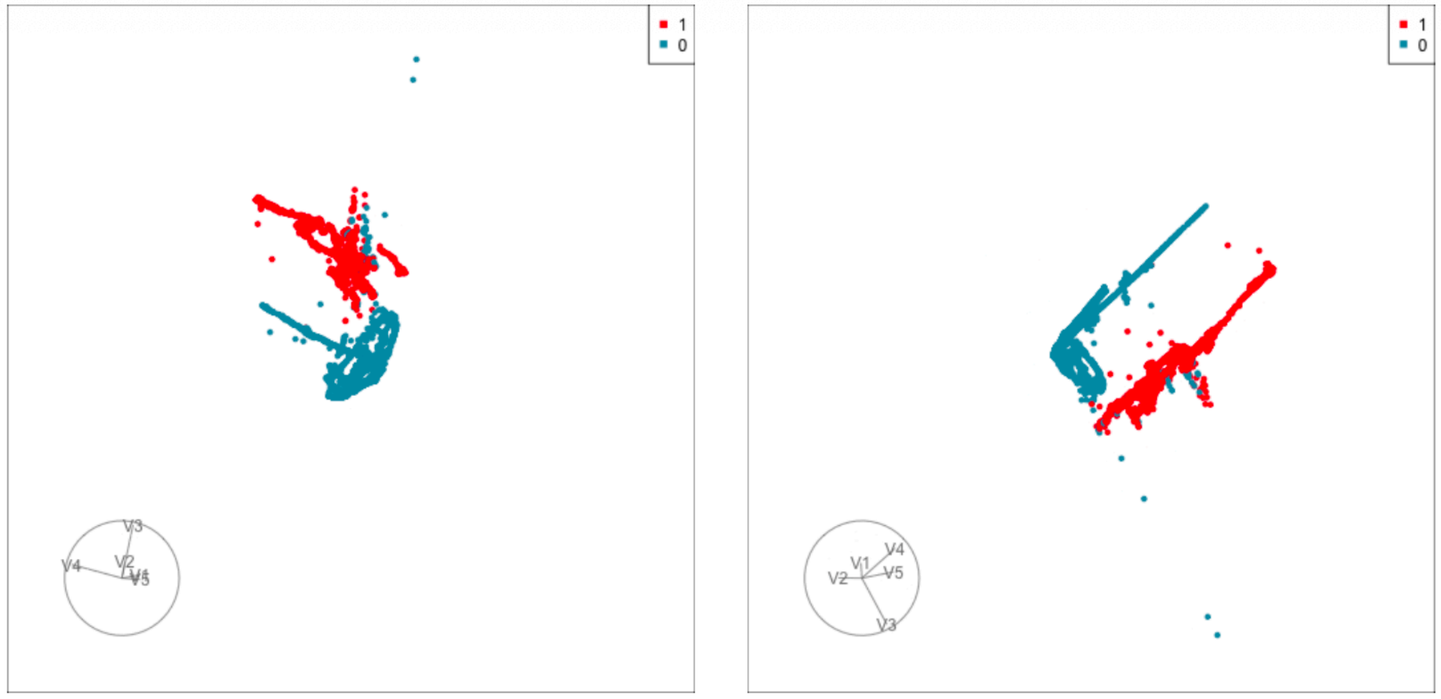
We would expect the modified PPtree algorithm to

- outperform rpart when the best separation between groups is in oblique directions
- and outperform PPtree when class variances are different, or when there are multiple clusters for any class.

Why Does It Win? High-D Visualization

data20-3 (5 predictors, 2 classes): grand tour reveals why

- Class 0 is split into **two clusters on either side** of class 1 → original PPtree is not optimal
- Separation is primarily V3 with small contributions from other variables → `rpart` misses the combinations
- PPtreeExt (mod 2) handles *both* issues simultaneously

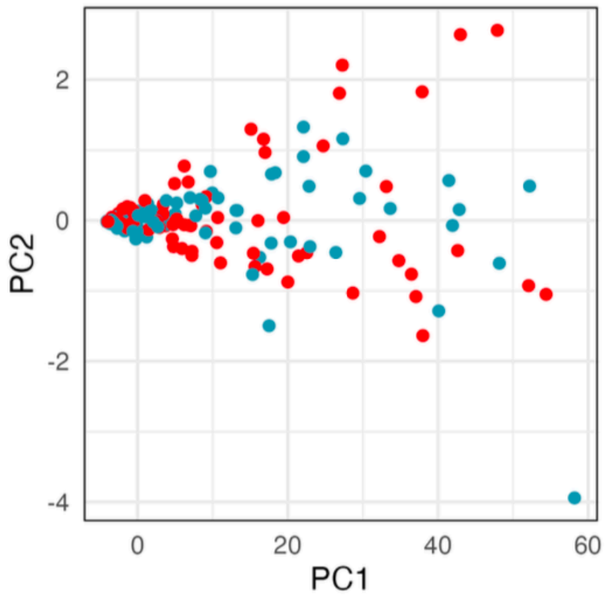


(a) projection 1 (b) projection 2

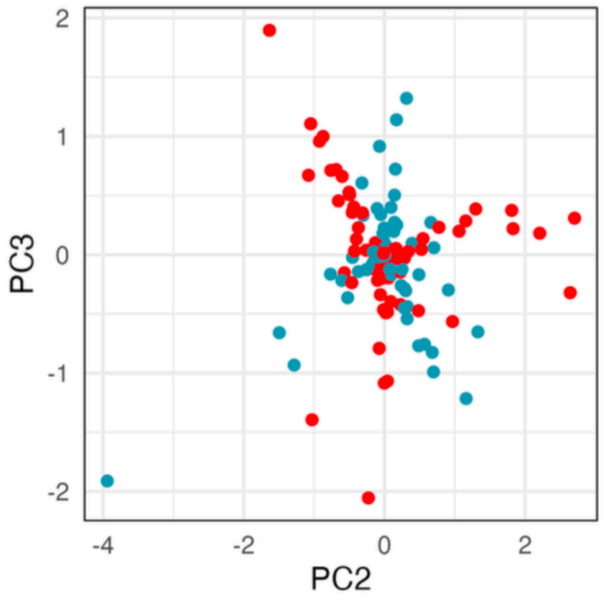
Why Does It Win? High-D Visualization

data49-2 (100 predictors, 2 classes):

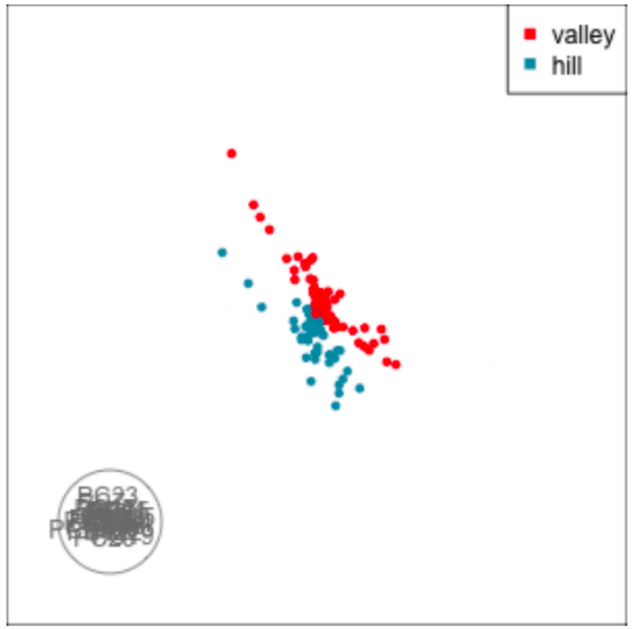
1. PCA → PC1 explains 99.8% of variance but it is not useful for separating the classes
2. PC2 vs PC3 shows an interesting class differences, each class appears to have three linear pieces pointing in different directions
3. A guided projection pursuit tour on the first 26 principal components shows that these classes could be separated with a linear combination. `PPtreeExt` handles multiple oblique linear cuts; `PPtree` only gets one



(a) PC1 vs PC2



(b) PC2 vs PC3

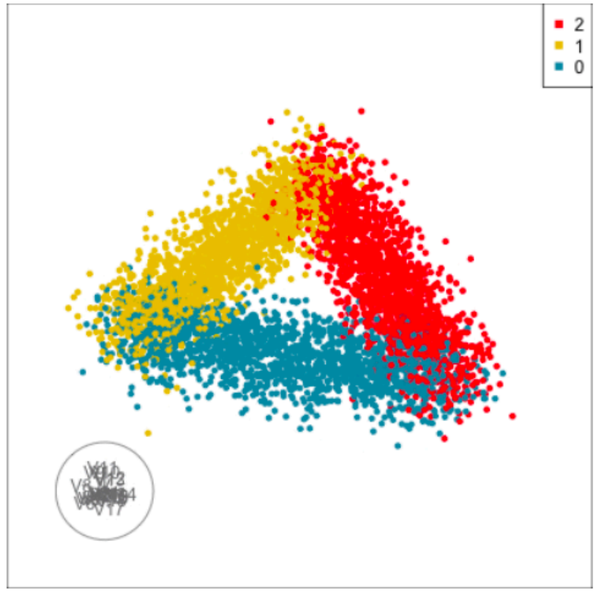


(c) optimal projection

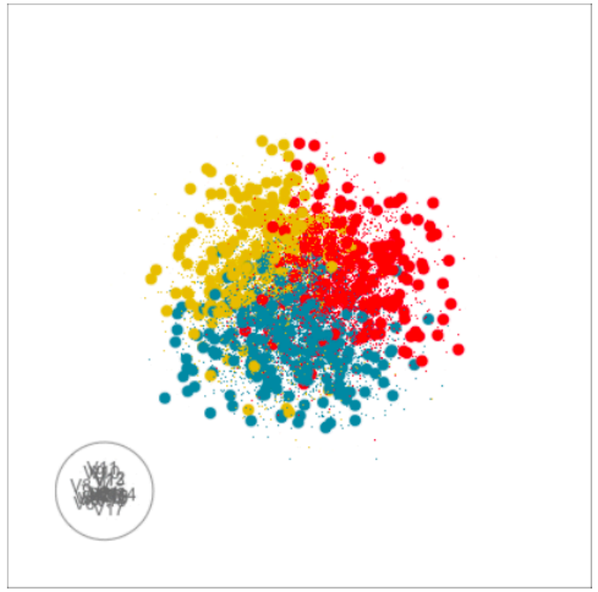
Slice Tour: Visualizing Boundaries in High-D

data69-1 (21 predictors, 3 classes)

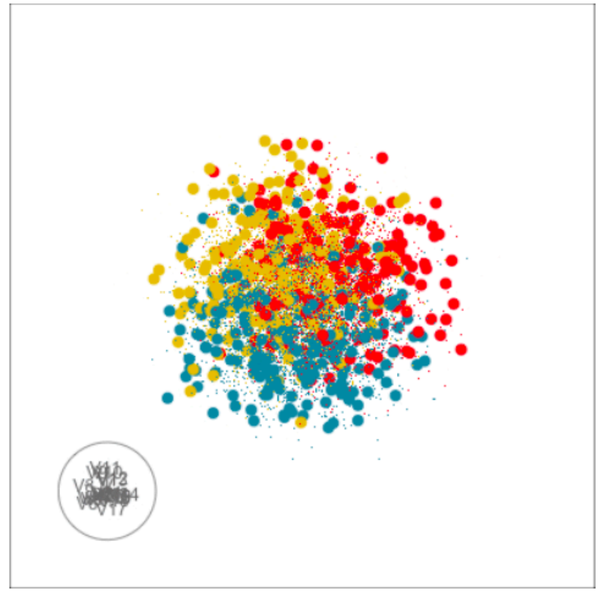
- Strategy: predict a grid of uniform random points in the 21-D space → keep points near the boundary → view with a slice tour using the same projection for all methods
- The boundary for `PPtreeExt` (mod 2) mirrors what we learn from the data but the `rpart` boundary is very messy. This data needs oblique cuts, which `PPtreeExt` provides but `rpart` does not. The modified algorithm handles the variance differences better than the original `PPtree`.



(a) data



(b) PPtreeExt



(c) rpart

Summary

Two modifications to PPtree, implemented in `PPtreeExt`:

- **Subsetting classes** → better split boundaries by using only the two closest classes in each projection step
- **Entropy-based multiple splits** → flexible tree depth, handles multimodal classes

Two visual diagnostic tools:

- **Shiny app** – interactive 2-D boundary comparison (development *and* communication tool)
- **Tours** – high-dimensional examination of data and model fits

Try it:

```
install.packages("PPtreeExt")
library(PPtreeExt)

# Run the Shiny explorer
run_app()

# Fit the extended classifier
PPtreeExtclass()
```

Shiny app:
natyds.shinyapps.io/explorapp/

Working paper
https://arxiv.org/abs/2602.21130

References

- Lee, Y.D., Cook, D., Park, J., Lee, E.-K. (2013). PPtree: Projection Pursuit Classification Tree. *Electronic Journal of Statistics*, 7, 1369–1386.
- Lee, E.-K. (2018), PPtreeViz: An R Package for Visualizing Projection Pursuit Classification Trees, Journal of Statistical Software, 83, 1–30. https://doi.org/10.18637/jss.v083.i08.
- da Silva, N., Cook, D., Lee, E.-K. (2021). A Projection Pursuit Forest Algorithm for Supervised Classification. *JCGS*, 30, 1168–1180.
- da Silva, N., Cook, D., Lee, E.-K. (2026). **PPtreeExt**: Projection Pursuit Classification Tree Extensions. CRAN.
- da Silva, N., Cook, D., Lee, E.-K. (2026). An Enhanced Projection Pursuit Tree Classifier with Visual Methods for Assessing Algorithmic Improvements.https://arxiv.org/abs/2602.21130
- Wickham, H. (2022), classifly: Explore Classification Models in High Dimensions. https://doi.org/10.32614/CRAN.package.classifly.
- Wickham, H., Cook, D., Hofmann, H. (2015). Visualizing Statistical Models: Removing the Blindfold. *Statistical Analysis and Data Mining*, 8, 203–225.
- Wickham, H., Cook, D., Hofmann, H., Buja, A. (2011). Tourr: An R Package for Exploring Multivariate Data with Projections. *JSS*, 40, 1–18.

Thank You!