TWIX

Trees W1th eXtra splits

Sergej Potapov
Martin Theus
Simon Urbanek
Motivation

• Where the classical CART algorithm fails
  ∵ Greedy algorithms never go for a (locally) second best solution, which would result in a better overall (global) solution.

Example: XOR-data
Motivation

• Where the classical CART algorithm fails
  – Greedy algorithms never go for a (locally) second best solution, which would result in a better overall (global) solution.

Example: XOR-data
Motivation

• Where the classical CART algorithm fails
  – Greedy algorithms never go for a (locally) second best solution, which would result in a better overall (global) solution.

Example: XOR-data
Motivation

• Where the classical CART algorithm fails
  – Greedy algorithms never go for a (locally) second best solution, which would result in a better overall (global) solution.
Trees: More Problems

- Non-orthogonal splitting directions …
Trees: More Problems

- Non-orthogonal splitting directions ...
Trees: More Problems

• Non-orthogonal splitting directions …

… can not be handled by single trees, no matter how we split
Bagging Revisited

Bagging = Bootstrap Aggregation, tries to simulate an infinite sample by bootstrapping, i.e. sampling from the original sample with replacement.

Repeat $N$ times:

1. Generate a bootstrap sample $D_i$ of size $n$.
2. Fit model $\hat{f}_{D_i}$.

Depending on the problem the $N$ results are aggregated:

- Classification: $g(x) = \arg\max_{c \in C} \sum_{i=1}^{N} I(f_{D_i}(x) = c)$

- Regression: $g(x) = \frac{1}{N} \sum_{i=1}^{N} f_{D_i}(x)$
Ensembles

- **General Idea**
  Use many “different” classifier and combine them to get more accurate results.

- Bagging: Instability of trees yields different models

- Random Forests: Restrict input space randomly to get wider range of models

- Boosting: Iterate to up-weight “bad” points
Ensembles

• **General Idea**
  Use many “different” classifier and combine them to get more accurate results.

• Bagging: Instability of trees yields different models

• Random Forests: Restrict input space randomly to get wider range of models

• Boosting: Iterate to up-weight “bad” points

**Question:**
Why use randomly generated (sub-optimal) models?
Tree Mechanics

- CART is a recursive partitioning algorithm
- Each node is split according to the maximum gain in the loss function
- Mountain plots show the loss function for a variable for all possible split points
Tree Mechanics

- CART is a recursive partitioning algorithm
- Each node is split according to the maximum gain in the loss function
- Mountain plots shows the loss function for a variable for all possible split points
Tree Mechanics

- CART is a recursive partitioning algorithm
- Each node is split according to the maximum gain in the loss function
- Mountain plots shows the loss function for a variable for all possible split points
Tree Mechanics

- CART is a recursive partitioning algorithm
- Each node is split according to the maximum gain in the loss function
- Mountain plots shows the loss function for a variable for all possible split points

Loss Functions

Mountain Plot
Tree Mechanics

- CART is a recursive partitioning algorithm
- Each node is split according to the maximum gain in the loss function
- Mountain plots shows the loss function for a variable for all possible split points
**Idea behind TWIX**

- Since the greedy CART algorithm not necessarily finds the “optimal” tree, try second best splits.
- Use these forests for aggregation
- Expect better results for both single trees and aggregations

**Problems**

- How to find “good” candidates for second best splits?
- Number of inner nodes grows exponentially with the number of levels in the tree
  \[ \Rightarrow \text{so does the number of alternative trees} \]
Second Best Splits: South African Heart Data

- sbp
- tobacco
- ldi
- adiposity
- typea
- obesity
- alcohol
- age
Second Best Splits: South African Heart Data
Second Best Splits: South African Heart Data
Second Best Splits: South African Heart Data
Second Best Splits: Global vs. Local

- When searching for a “best” split point, we can either look for
  - all top n greatest deviance gains, or
  - only look for local maxima

- Example
  Top 6 splits
Second Best Splits: Global vs. Local

- When searching for a “best” split point, we can either look for
  - all top n greatest deviance gains, or
  - only look for local maxima

- Example
  Top 6 splits
Second Best Splits: Global vs. Local

- When searching for a “best” split point, we can either look for
  - all top n greatest deviance gains, or
  - only look for local maxima

- Example
  Top 6 splits

![Graph showing age distribution with split points identified]
Second Best Splits: Forcing Variables

- Often a single variable dominates the potential deviance gain, and shadows all other variables
  ⇒ Many probably good split points are lost.

- Solution:
  Force a minimum number of split points for each variable.

- Example: top 6 vs. top 3
Second Best Splits: Forcing Variables

- Often a single variable dominates the potential deviance gain, and shadows all other variables
  ⇒ Many probably good split points are lost.

- Solution:
  Force a minimum number of split points for each variable.

- Example: top 6 vs. top 3
Second Best Splits: Forcing Variables

• Often a single variable dominates the potential deviance gain, and shadows all other variables
  ⇒ Many probably good split points are lost.

• Solution:
  Force a minimum number of split points for each variable.

• Example: top 6 vs. top 3
Second Best Splits: Forcing Variables

- Often a single variable dominates the potential deviance gain, and shadows all other variables
  ⇒ Many probably good split points are lost.

- Solution:
  Force a minimum number of split points for each variable.

- Example: top 6 vs. top 3
Second Best Splits: Grid Search

- In some situations good split points might not even be associated with some (local) maximum in deviance gain. (Remember the XOR Example)

- Grid searches are most exhaustive, but also most expensive.
Second Best Splits: Grid Search

- In some situations good split points might not even be associated with some (local) maximum in deviance gain. (Remember the XOR Example)

- Grid searches are most exhaustive, but also most expensive.
Implementation: The Grid

- If we allow $s_j$ splits per node on level $j$ of the tree, we get a maximum of

$$S = \prod_{i=1}^{k} s_i^{2^{i-1}}$$

trees for a tree with no more than $k$ levels. Example:

$$s = (7, 4, 2) \Rightarrow S = 7^2 \cdot 4^1 \cdot 2^2 = 7 \cdot 16 \cdot 16 = 1792$$

⇒ Work on a grid of computers
Implementation: The Grid

- If we allow $s_j$ splits per node on level $j$ of the tree, we get a maximum of
  \[
  S = \prod_{i=1}^{k} s_i^{2^{i-1}}
  \]
trees for a tree with no more than $k$ levels. Example:

  \[
  s = (7, 4, 2) \Rightarrow S = 7^0 \cdot 4^1 \cdot 2^2 = 7 \cdot 16 \cdot 16 = 1792
  \]

  \[\Rightarrow\text{ Work on a grid of computers}\]
Using the R Package

• The most important tuning parameters are

  - **method**
    Which split points will be used? This can be "deviance" (default), "grid" or "local". If the method is set to: local the program uses the local maxima of the split function (entropy), deviance all values of the entropy, grid grid points.

  - **topn.method**
    one of "complete" (default) or "single". A specification of the consideration of the split points. If set to "complete" it uses split points from all variables, else it uses split points per variable.

  - **topN**
    integer vector. How many splits will be selected and at which level? If length 1, the same size of splits will be selected at each level. If length > 1, for example topN=c(3,2), 3 splits will be chosen at first level, 2 splits at second level and for all next levels 1 split.

  - **level**
    maximum depth of the trees. If level set to 1, trees consist of root node.

  - **Stopping Rules:**
    - **minsplit**
      the minimum number of observations that must exist in a node.
    - **minbucket**
      the minimum number of observations in any terminal <leaf> node.
    - **Devmin**
      the minimum improvement on entropy by splitting.
South African Heart Disease Data cont.
South African Heart Disease Data cont.

• To get a “fair”, i.e. generalizable and not too overfitted classifier, we usually split the data into 3 chunks:

   Training          Validation          Test
South African Heart Disease Data cont.

- To get a “fair”, i.e. generalizable and not too overfitted classifier, we usually split the data into 3 chunks:

  - **Training**
    All models are trained using the training data
South African Heart Disease Data cont.

- To get a “fair”, i.e. generalizable and not too overfitted classifier, we usually split the data into 3 chunks:

  - **Training**
    All models are trained using the training data
  
  - **Validation**
    The “best” model is selected using the validation data
    (The chosen model is then estimated with training+validation)
South African Heart Disease Data cont.

- To get a “fair”, i.e. generalizable and not too overfitted classifier, we usually split the data into 3 chunks:

  - **Training**
    All models are trained using the training data

  - **Validation**
    The “best” model is selected using the validation data
    (The chosen model is then estimated with training+validation)

  - **Test**
    The performance is then assessed with the test data
South African Heart Disease Data cont.

- To get a “fair”, i.e. generalizable and not too overfitted classifier, we usually split the data into 3 chunks:

  Training  Validation  Test

  - **Training**
    All models are trained using the training data

  - **Validation**
    The “best” model is selected using the validation data
    (The chosen model is then estimated with training+validation)

  - **Test**
    The performance is then assessed with the test data

**What about trees?**

Model Structure = Model parameters
The Dataset
The Dataset

- 10 Variables, 462 Observations
The Dataset

- 10 Variables, 462 Observations
- Target: **Coronary Heart Disease (chd)**, 34.63% = 160 cases
The Dataset

- 10 Variables, 462 Observations

- Target: **Coronary Heart Disease (chd)**, 34.63% = 160 cases

- Inputs:
  - continuous
  - sbp  systolic blood pressure
  - tobacco  cumulative tobacco (kg)
  - ldl  low density lipoprotein cholesterol
  - adiposity
  - typea  type-A behavior
  - obesity
  - alcohol  current alcohol consumption
  - age  age at onset
The Dataset

- 10 Variables, 462 Observations
- Target: **Coronary Heart Disease (chd)**, 34,63% = 160 cases
- Inputs:
  - **continuous**
    - sbp    systolic blood pressure
    - tobacco cumulative tobacco (kg)
    - ldl    low density lipoprotein cholesterol
    - adiposity
    - typea  type-A behavior
    - obesity
    - alcohol current alcohol consumption
    - age    age at onset
  - **discrete**
    - famhist family history of heart disease (Present, Absent)
The Dataset: Univariate

sbp
The Dataset: Univariate

sbp
The Dataset: Univariate

sbp
The Dataset: Univariate

<table>
<thead>
<tr>
<th>sbp</th>
<th>tobacco</th>
<th>idl</th>
<th>adiposity</th>
</tr>
</thead>
<tbody>
<tr>
<td>typea</td>
<td>obesity</td>
<td>alcohol</td>
<td>age</td>
</tr>
</tbody>
</table>

### Variables
- sbp
- tobacco
- idl
- adiposity
- typea
- obesity
- alcohol
- age
The Dataset: Bivariate
The Dataset: Multivariate
The Dataset: Multivariate

sbp
tobacco
ldl
adiposity
obesity
alcohol
typea
alcohol
age

The Dataset: Multivariate
TWIX: Diagnostics

• For a given “Multitree” we can compare deviance and classification rate on training and test/validation data.

Example:
TWIX: Tree Selection
TWIX: Tree Selection

- The CCR (Correct Classification Rate) of the top TWIX trees are better than those of greedy trees and many other classification methods.

**Quest:**
How to find the “best” trees from the validation data?
TWIX: Tree Selection

• The CCR (Correct Classification Rate) of the top TWIX trees are better than those of greedy trees and many other classif. methods

**Quest:**
How to find the “best” trees from the validation data?

• Several approaches:
  (a) Sort trees according to:
    ■ training deviance,
    ■ validation deviance,
    ■ validation CCR,
    and pick the best!
  (b) Avoid extreme trees, i.e. forget trees having the worst deviances or CCRs and repeat (a).
  (c) Look for structural properties like balance of tree, purity and size of leaves
  (d) Identify clusters among the trees and avoid selecting trees from a “bad” cluster
  (e) Use a mixture from (a) – (d) …
Looking at Tree Clusters

- Metric: Jaccard Coefficient \( d_{jacc}(B_i, B_j) := \frac{1}{|V_i \cup V_j|} \cdot \sum_{k=1}^{n} |\gamma_{i,k} - \gamma_{j,k}| \)

- Create groups via MDS and hierarchical clustering
Looking at Forests

- Traceplots show a tree ensemble in a single framework

Cluster 1
Looking at Forests

• Traceplots show a tree ensemble in a single framework

Cluster 1

Cluster 3
The Competitors: On 100 random samples

- Logistic Regression
  
  \[ \text{glm(response~., data=dataTrain, family="binomial")} \]

- Traditional CART (from rpart)
  
  \[ \text{rpart(response ~ ., data=dataTrain,} \]
  
  \[ \text{ parms=list(split='information'))} \]

- Bagging (from ipred)
  
  \[ \text{bagging(response~., data=dataTrain)} \]

- SVM (from e1070)
  
  \[ \text{svm(response~.,data=dataTrain)} \]

!! None of the methods has been fine-tuned !!
TWIX: Results

- 100 runs of a (20,3) TWIX tree, local maxima
TWIX: Results

- 100 runs of a (20,3) TWIX tree, local maxima
TWIX: Results

- 100 runs of a (20,3) TWIX tree, local maxima
TWIX: Results

- 100 runs of a (20,3) TWIX tree, local maxima
**TWIX: Results**

- 100 runs of a (20,3) TWIX tree, local maxima
TWIX: Results

- 100 runs of a (20,3) TWIX tree, local maxima
TWIX: Results

- 100 runs of a (20,3) TWIX tree, local maxima
TWIX Results: Parallel Coordinates

- rpart (test)
- Bagging (test)
- TWIX (test)
- TWIX agg (test)
- svm (test)
- logistic (test)
- TWIX (training)
- svm (training)
- rpart (training)
TWIX Results: Parallel Coordinates
TWIX Results: Parallel Coordinates

- rpart (test)
- Bagging (test)
- TWIX (test)
- TWIX agg (test)
- svm (test)
- logistic (test)
- TWIX (training)
- svm (training)
- rpart (training)
TWIX Results: Parallel Coordinates
How does TWIX compare?

- Are the resulting ensembles similar to other tree-based ensemble methods?
How does TWIX compare?

- Are the resulting ensembles similar to other tree-based ensemble methods?

rForests \rightarrow TWIX

Force the use of splits in all variables
How does TWIX compare?

- Are the resulting ensembles similar to other tree-based ensemble methods?

- TWIX
  - many splits close to greedy split

- Bagging
  - Force the use of splits in all variables

- rForests
  - Force the use of splits in all variables
Conclusion
Conclusion

• For the South-African-Heart Data
  – Single TWIX-trees out-perform traditional trees and usually bagged trees
  – Aggregated TWIX beats bagged trees and reaches top performance
  – TWIX gives good single alternative tree models
Conclusion

• For the South-African-Heart Data
  – Single TWIX-trees out-perform traditional trees and usually bagged trees
  – Aggregated TWIX beats bagged trees and reaches top performance
  – TWIX gives good single alternative tree models

• Still room for performance improvement
  – Better (more stable) tree selection
  – Improved selection of “second best” splits
  – Improved aggregation of the trees (weights, boosting, …)
  – More tests on more datasets (mlbench, “report78”)
  – Better understanding of the tree-families
Conclusion

• For the South-African-Heart Data
  – Single TWIX-trees out-perform traditional trees and usually bagged trees
  – Aggregated TWIX beats bagged trees and reaches top performance
  – TWIX gives good single alternative tree models

• Still room for performance improvement
  – Better (more stable) tree selection
  – Improved selection of “second best” splits
  – Improved aggregation of the trees (weights, boosting, …)
  – More tests on more datasets (mlbench, “report78”)
  – Better understanding of the tree-families

• Computational effort can be high, but parallel computing helps
Conclusion

• For the South-African-Heart Data
  – Single TWIX-trees out-perform traditional trees and usually bagged trees
  – Aggregated TWIX beats bagged trees and reaches top performance
  – TWIX gives good single alternative tree models

• Still room for performance improvement
  – Better (more stable) tree selection
  – Improved selection of “second best” splits
  – Improved aggregation of the trees (weights, boosting, …)
  – More tests on more datasets (mlbench, “report78”)
  – Better understanding of the tree-families

• Computational effort can be high, but parallel computing helps
• Understanding of tree ensembles still poor, classical metrics fail
Conclusion

• For the South-African-Heart Data
  – Single TWIX-trees out-perform traditional trees and usually bagged trees
  – Aggregated TWIX beats bagged trees and reaches top performance
  – TWIX gives good single alternative tree models

• Still room for performance improvement
  – Better (more stable) tree selection
  – Improved selection of “second best” splits
  – Improved aggregation of the trees (weights, boosting, …)
  – More tests on more datasets (mlbench, “report78”)
  – Better understanding of the tree-families

• Computational effort can be high, but parallel computing helps

• Understanding of tree ensembles still poor, classical metrics fail

• Complex methods are hard to implement and hard to test, importance of “reproducible research” cannot be underestimated!