Effect of decision tree quality on tree-based optimization results

Dr. Vladimir Fishman and Dr. Anatoly Reynberg

Credit Scoring and Credit Control XI Conference August 2009

Topics of Presentation

- New tree-based method of solving large scale record level optimization problem
- Importance of decision tree quality for this approach, and measures of the tree quality
- Improved enumeration procedure for nodes splitting/merging
- Combined (hybrid) tree building algorithm
- Role of the criterion choice in the automated tree building. Some advanced criteria (integral, global)
- Practical examples of optimization gain resulting from the improved tree quality
- Conclusions

Tree-based Optimization. Algorithm

- Goal: to build a tree-based solution to the original record-level constraint optimization problem, which is as close as possible to the optimal solution
- **Step 1**. Solve original optimization problem
- **Step 2**. Build automated regression tree, using the decision options, assign by the optimal solution to each record, as the multi-dimensional dependent variable
- Tree based assignments can serve as the proxy for the optimal solution, but there is no guarantee that the constraints will be satisfied. So the following step is necessary
- **Step 3**. Obtain the optimal solution to the smaller scale tree-based segment level constraint optimization, considering each segment as one record

Tree-based Optimization. Advantages

- The knowledge of the original record-level optimal solution allows for “supervised”, and therefore more accurate segmentation
- Utility function of optimization solution of tree based problem is generally degraded in comparison with the original large scale problem, but the degradation is normally very small.
- The degradation may become larger in some complex cases
- Using improved tree-building routine can decrease the gap between the values of the objective function of the original solution and the tree-based one
- Tree-based solution is much easier interpretable, which often helps to overcome the “black-box” fear

Tree Quality Measures

- Commonly accepted, based on an impurity measure. Example:

  Initial (root) impurity $I_0 = 1 - \sum_{j=1}^{k} \frac{n_{.j}^2}{n_{..}^2}$ Tree impurity $I_t = \sum_{i=1}^{m} (1 - \sum_{j=1}^{k} \frac{n_{ij}^2}{n_{i.}^2}) \frac{n_{i.}}{n_{..}}$

  Relative impurity reduction in % $IR = 100\% \frac{I_0 - I_t}{I_0}$

  k-number of decision options, m – number of terminal nodes

  Greater value shows better performing tree

- Advanced, optimization related. Relative gap in the value of the objective function between the optimal solution of the original record-level problem and the optimal solution of the tree-based segmented problem. Lesser value indicates better performing tree-based solution

Automated Tree Building Algorithms

- Any tree building algorithm consists of three major steps: merging, splitting, and stopping.

  Merging combines together small bins up to the stage when meaningful splitting can start. During the splitting an attempt is made to find the grouping of the initial bins, providing the best value of the selected criterion, and split the node according to this grouping

- Examples of most widely used tree building algorithms CHAID (Kass) (Chi-Square), Exhaustive CHAID (Biggs) (Chi-Square), C4.5, C5.0 (Quinlan) (information gain)

Efficient Splitting Is the Key to Better Performance

- Combinatorial nature of the best grouping search

Number of combination to consider

Ordinal predictor

$$N = \frac{(m-1)!}{(g-1)!(m-g)!} = C_{m-1}^{g-1} \qquad (1)$$

Nominal predictor

$$N = \sum_{i=0}^{g-1} (-1)^{i} \frac{(g-i)^{m}}{i!(m-i)!} \qquad (2)$$

(Stirling number of the second kind)

m – initial number of bins, g – desired number of groups

- Complete enumeration of all possible combination in many cases cannot be reached in a reasonable time, especially for nominal attributes

Efficient Splitting Is the Key to Better Performance

- Palliative: Pair-wise grouping

Number of combination to consider

Ordinal predictor

$$N = \frac{(m-g)(m+g-1)}{2} \tag{3}$$

Nominal predictor

$$N = \frac{mg(m-g)}{2} + \frac{(m-g-1)(m-g)(m-g+1)}{6} \tag{4}$$

- Pair-wise grouping requires much lesser (polynomial by m) number of permutations to be analyzed, but leaves a lot of combinations out of consideration

Efficient Splitting –Our Approach

- Our solution – two stage analysis
- First, for the given number N of combinations to be fully analyzed, and for the desired size of the final grouping *g* we determine from (1) and (2) the value of *m*, starting with which we can afford comprehensive enumeration
- **Stage 1.** Starting with the initial number of values/categories *M* in an attribute to analyze, we perform pair-wise grouping to reduce the number of resulting groups to *m*
- **Stage 2.** Exhaustive analysis of all possible combinations of the obtained m groups to the final g groups is performed
- By design this procedure guarantees comprehensive analysis for attributes with relatively small number of values/categories, and generally better than pair-wise merging enumeration for larger number

Efficient Splitting – Our Approach

- Complete enumeration requires analysis of huge amount of combination, especially for nominal attributes. If m=10, g=5, then N = 45525, but for m=15, g=7 it is already greater than 4.08×10^8
- For the special case of the binary dependent variable the required number of combinations to analyze for a nominal attribute can be reduced to the analogous number for the ordinal one.
- Our primary focus is to build trees with several binary dependent variables (decision options), so one of the ways to reduce the computational complexity of splitting with a nominal attribute is to find the dominant binary decision option, and look for the best grouping, using this one binary dependent variable and the efficient way of grouping

Efficient Splitting – Our Approach

- The next step – to let all the described methods to compete
- In the case of an attribute with a large number of values/categories, and large number of final groups we still can be far away from complete enumeration
- If we try to find the best splitting, obtained by each of the method considered, and choose the best of the best, we often end up with the better tree, than the one obtained by each method separately
- This “hybrid” approach requires somewhat greater time, but can produce the real edge

Criterion Choice – Our Approach

- Various criteria evaluate the quality of a split differently. Often the criterion choice determines the type of the tree building method (e.g. CHAID vs. C4.5). At the same time the structure of the most tree building algorithms is quite similar. It is natural to combine different criteria, allowing the researcher to choose the most advantageous one.
- Some commonly accepted criteria implemented

P-value based on $\chi^2 = \sum_{i=1}^{m}\sum_{j=1}^{k} \frac{(n_{ij} - n_{i.} \cdot n_{.j} / n_{..})^2}{n_{i.} \cdot n_{.j} / n_{..}}$ df = (m-1)(k-1)

P-value based on $F = \frac{SS_{BG}(n_{..} - m)}{SS_{WG}(m-1)}$ (continuous dependent)

Gini Index of Diversity $I_0 - I_t$ (see slide 5)

Criterion Choice – Our Approach

Information Gain $EG = \sum_{j=1}^{k} EG_j \cdot \frac{n_{.j}}{n_{..}}$, where

$$EG_j = \ln n_{..} - \frac{1}{n_{..}} \cdot \left[n_{.j} \cdot \ln n_{.j} + (n_{..} - n_{.j}) \cdot \ln(n_{..} - n_{.j}) - \sum_{i=1}^{m} [(n_{i.} - n_{ij}) \cdot \ln(n_{i.} - n_{ij}) + n_{ij} \cdot \ln n_{ij} - n_{i.} \cdot \ln n_{i.}] \right]$$

Twoing
(for binary split only) $\varphi = 0.25 \cdot \frac{n_{1.}}{n_{..}} \frac{n_{2.}}{n_{..}} \left[\sum_{j=1}^{k} \left| \frac{n_{1j}}{n_{1.}} - \frac{n_{2j}}{n_{2.}} \right| \right]^2$

All the criteria are applied only after establishing proper split significance

Criterion Choice – Our Approach

Integral Criterion

Assume, that for a data set with *N* records we know both actual values of the binary dependent variable, taking values 0 and 1, and the score produced by some model. Sort the dependent by the model score in descending order. Let *i* be a record number in the sorted list, *x = i/N*, *R* – sum of all 1's divided by *N* (i.e. average Good Rate, or Response Rate), *R(x)* – rate for the records up to *i*, *R(1-x)* – rate for rest of the records.

The quantity *(R(x) – R(1-x))/R* can be considered as the measure of the model classification ability at the point x. It can be rewritten as *(R(x)-R)/R/(1-x).* Integral measure for the score interval (x_1,x_2) will be

$$\frac{1}{x_2 - x_1}\int_{x_1}^{x_2}\frac{R(\xi) - R}{R(1-\xi)}d\xi$$

This criterion has an ability to measure the model performance within some predefined score interval. It can be converted to a contingency table form, to be used in tree building

Criterion Choice – Our Approach

Global criteria

- Usually any measure of a node splitting quality is calculated using contingency table collected locally, only for the node considered
- However, often more accurate result can be obtained when the measure is calculated using extended contingency table, which combines contingency tables for all final nodes, built up to the moment
- In this case the best splitting choice is dependent on the entire tree, which has been built

Practical Examples of Optimization Gain

Case 1. Mobil Phone Company

Data size – 5126 records, # of attributes – 20, type of attributes – continuous

Optimization task – Profit maximization, Decision options – 9

Global (resource) constraints: 6

Profit for the optimal solution of the record level problem - $52,111

Tree	# of levels	# of final nodes	Max # of children	Criterion	Tree-based profit ($)	Gap (%)
Standard	5	101	5	Chi-Square	45,201	13.3
Advanced	5	72	5	Chi-Square	46,211	11.3
Hybrid	5	76	5	Chi-Square	46,951	9.9

Optimization Gain due to the tree enhancements 3.4%

Practical Examples of Optimization Gain

Case 2. Credit Line Increase

Data size – 154,240 records, # of attributes – 104, type of attributes – mixed

Optimization task – Profit maximization, Decision options – 12

Global (resource) constraints: 4

Initial 2 level 4 final nodes tree is provided by the client

Profit for the optimal solution of the record level problem - $96,683,297

Tree	# of levels	# of final nodes	Max # of children	Criterion	Tree-based profit ($)	Gap (%)
Standard	5	100	5	Chi-Square	87,601,552	9.4
Advanced	5	127	5	Gini Index	89,682,210	7.2
Hybrid	5	97	5	Gini Index	89,956,150	7.0

Optimization Gain due to the tree enhancements 2.4%

Practical Examples of Optimization Gain

Case 3. Credit limit assignment

Data size – 40649 records, # of attributes – 99, type of attributes – mixed

Optimization task – Profit maximization, Decision options – 19

Global (resource) constraints: 3

Initial 2 level 8 final nodes tree is provided by the client

Profit for the optimal solution of the record level problem - $5,127,021

Tree	# of levels	# of final nodes	Max # of children	Criterion	Tree-based profit ($)	Gap (%)
Standard	5	251	5	Chi-Square	4,667,103	9.0
Advanced	5	246	5	Chi-Square	4,699,995	8.3
Hybrid	5	265	5	Chi-Square	4,715,004	8.0

Optimization Gain due to the tree enhancements 1.0%

Conclusions

- Approximating of the decisions, found as a solution of the large scale record level constraint optimization problem, by building the regression tree enables to obtain accurate easily interpretable tree-based solutions
- In this setting the quality of the tree is important. The more complex an optimization scenario is, the better tree approximation may required
- We approach the task of building high quality tree by both finding more efficient ways of enumeration in splitting/merging of the tree nodes, and using numerous criteria of the split quality
- Having the advantage of knowing optimal record-level solution in advance, we can objectively estimate the gap between the optimal and tree-based solution, and, if needed, take additional steps to improving the tree quality