Feature Extraction from Discrete Attributes

Olcay Taner Yıldız

Department of Computer Engineering
Işık University
Istanbul, Turkey

ICPR 2010

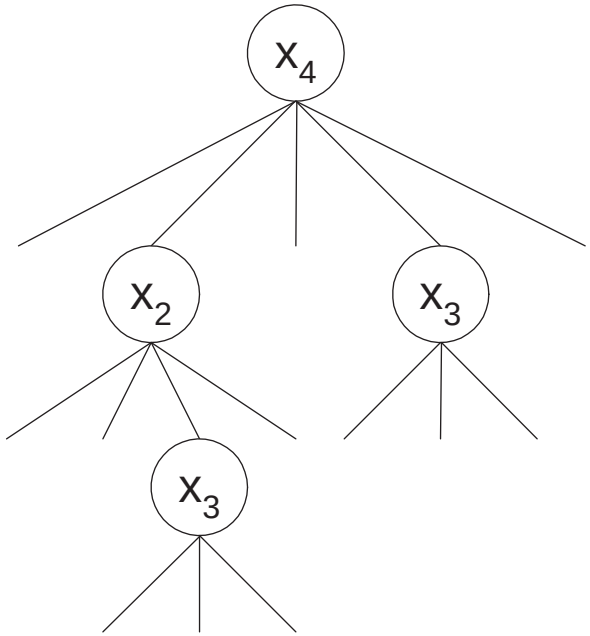
Outline

Decision Trees

- Tree-based structures
- Each internal node implements a decision function, $f_m(\boldsymbol{x})$
- Univariate decision tree, $f_m(\boldsymbol{x})$ uses only one attribute
- If attribute is discrete, there will be L children of each internal node

Drawbacks of L-ary splits

- Training examples are separated into small subsets; poor predictor for the unseen test instances
- Single possible split for each discrete attribute



Proposed Solution

- Increase the number of distinct splits
- k-ordering: For each subset (size k) of the attributes, generate all possible orderings of the values of those attributes exhaustively
- Apply the usual univariate decision tree algorithm using these orderings as the new attributes
- Number of splits: $\Pi_{i=1}^{k} n_{s_i}$

Definition

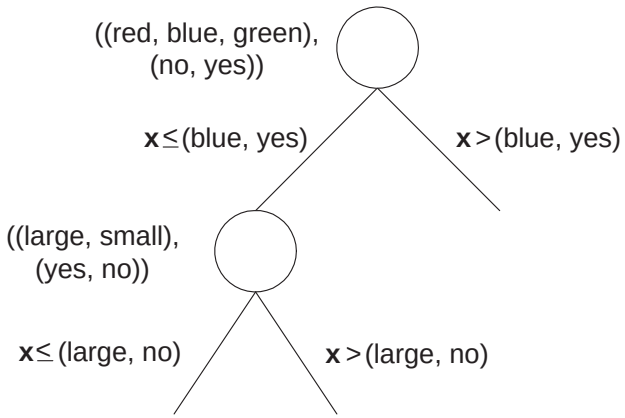
- Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_d$ be d discrete attributes of a dataset D
- Each attribute $\boldsymbol{x}_i$ can have n_i distinct values represented as $v_{i1}, v_{i2}, \ldots, v_{in_i}$
- For each k attributes $\boldsymbol{x}_{s_1}, \boldsymbol{x}_{s_2}, \ldots, \boldsymbol{x}_{s_k}$, k-ordering is defined as the permutation list $(p_1, p_2, \ldots, p_k)$, where p_i is a permutation of values of the feature s_i (a permutation of $v_{s_i1}, v_{s_i2}, \ldots, v_{s_in_{s_i}}$).
- ((red, blue, green), (yes, no), (large, extralarge, small, medium)) is a 3-ordering

k-ordering

- One can use categorical and/or nominal discrete attributes
- There are $\sum_{s_1,s_2,\ldots,s_k \in 1,2,\ldots,d} n_{s_1}! n_{s_2}! \ldots n_{s_k}!$ distinct k-orderings

Relational Operator

- Let $(x_1, x_2, \ldots, x_d)$ and $(y_1, y_2, \ldots, y_d)$ be two instances
- $(x_{s_1}, x_{s_2}, \ldots, x_{s_k}) \prec (y_{s_1}, y_{s_2}, \ldots, y_{s_k})$ if and only if $x_{s_i} = y_{s_i}$ for $i = 1, \ldots, t \geq 1$, and $x_{s_{t+1}}$ comes before $y_{s_{t+1}}$ in permutation p_{t+1}
- Given the 2-ordering ((red, blue, green), (no, yes)), (red, no) $\prec$ (red, yes) $\prec$ (blue, no) $\prec$ (blue, yes) $\prec$ (green, no) $\prec$ (green, yes)

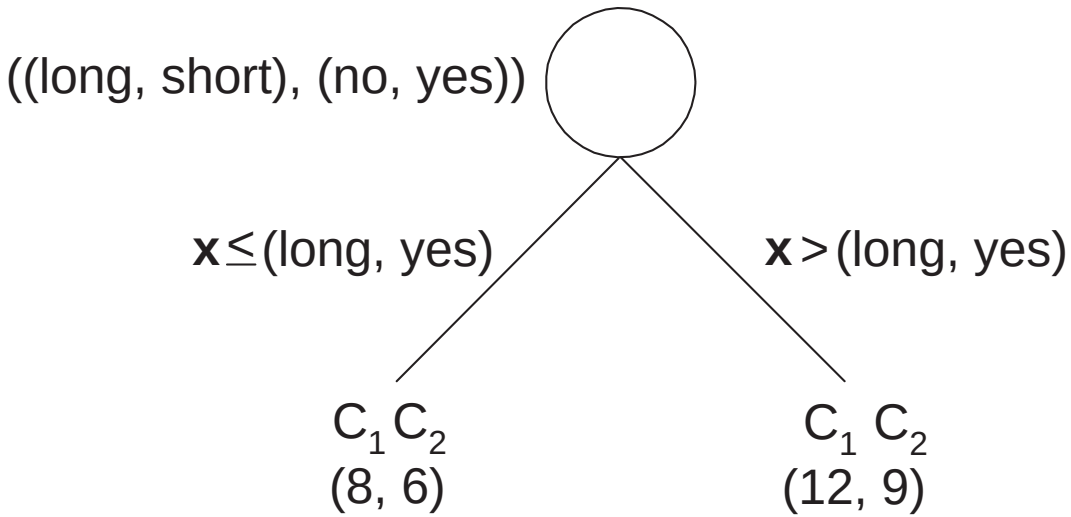


Tree construction: Idea

- At each decision node, we generate all possible k-orderings
- For each new attribute there are $n_{s_1} \ldots n_{s_k}$ distinct split points
- For all attributes and for all split points of those attributes find the best split with the best information gain

Example

	C_1	C_2
(long, no)	5	2
(long, yes)	3	4
(short, no)	2	5
(short, yes)	10	4



Pseudocode of the algorithm

```
Best-k-OrderingandSplit(m, k)
1   bestImpurity = ∞
2   bestSplit = bestOrdering = nil
3   for each feature permutation (s_1, ..., s_k)
4       foreach permutation p_1 of values of x_{s_1}
5           ...
6           foreach permutation p_k of values of x_{s_k}
7               foreach split point sp = (v_1, ..., v_k)
8                   if Impurity(sp, m) < bestImpurity
9                       bestImpurity = Impurity(sp, m)
10                      bestSplit = (v_1, ..., v_k)
11                      bestOrdering = (p_1, ..., p_k)
12  return bestSplit, bestOrdering
```

Datasets

Table: Details of the datasets. d: Number of attributes, C: Number of classes, N: Sample size, n/v: n attributes of the dataset has v distinct values

Dataset	d	C	N	n/v
acceptors	88	2	3889	88/4
artificial	10	2	320	10/2
balance	4	3	625	4/5
car	6	4	1728	3/3, 2/4, 1/5
connect4	42	3	67557	42/3
donors	13	2	6246	13/4
hayesroth	4	3	160	1/3, 3/4
krvskp	36	2	3196	35/2, 1/3
monks	6	2	432	1/2, 3/3, 1/4
nursery	8	5	12960	1/2, 4/3, 2/4, 1/5
promoters	57	2	106	57/4
spect	22	2	267	22/2
splice	60	3	3175	60/4
tictactoe	9	2	958	9/3
titanic	3	2	2201	2/2, 1/4
vote	16	2	435	16/2

Error Rate

Table: The average and standard deviations of error rates of C4.5 and K-tree algorithms with k = 1, 2, 3, 4.

Dataset	C4.5	1-tree	2-tree	3-tree	4-tree
acceptors	15.66±1.74	12.65±0.82	12.04±0.94	-	-
artificial	0.74±1.56	0.74±1.56	0.74±1.56	0.74±1.56	0.00±0.00
balance	40.53±2.70	27.13±3.39	**24.88±2.32**	-	-
car	12.46±1.92	3.97±0.55	**2.36±0.54**	3.14±1.15	-
connect4	26.01±0.45	24.88±0.42	23.66±0.37	-	-
donors	7.76±0.69	6.22±0.34	6.68±0.43	-	-
hayesroth	26.73±1.50	21.45±4.09	27.82±4.78	25.82±4.83	28.00±5.30
krvskp	1.23±0.40	0.96±0.37	0.94±0.41	**0.46±0.20**	0.60±0.25
monks	14.72±6.12	12.71±5.79	**0.00±0.00**	0.00±0.00	0.00±0.00
nursery	5.50±0.45	1.66±0.25	**0.43±0.23**	0.60±0.17	-
promoters	24.44±10.29	27.22±12.20	20.56±3.97	-	-
spect	20.33±2.46	20.33±2.46	20.89±0.70	20.11±2.43	19.78±2.86
splice	9.76±0.87	7.11±0.83	6.11±0.56	-	-
tictactoe	22.78±1.65	10.53±3.67	8.97±2.67	**4.59±2.47**	5.56±3.19
titanic	21.80±0.54	21.72±0.53	21.32±0.41	21.29±0.44	-
vote	5.10±0.36	5.10±0.36	5.03±0.33	5.38±0.29	5.38±1.16

Tree Complexity

Table: The average and standard deviations of number of nodes of C4.5 and K-tree algorithms with k = 1, 2, 3, 4.

Dataset	C4.5	1-tree	2-tree	3-tree	4-tree
acceptors	31.60±20.73	9.80±4.83	**8.20±3.97**	-	-
artificial	5.60±0.84	5.60±0.84	3.80±0.42	2.80±0.42	3.00±0.00
balance	9.80±7.00	13.80±4.57	11.60±2.63	-	-
car	62.70±8.55	32.60±3.34	19.50±2.12	**16.20±0.92**	-
connect4	887.00±115.99	575.70±65.82	314.10±58.83	-	-
donors	73.60±19.12	20.30±6.36	**9.50±4.55**	-	-
hayesroth	16.60±2.37	8.90±1.37	5.50±0.71	5.30±1.57	4.40±0.70
krvskp	29.80±4.44	26.90±3.14	17.20±1.69	12.00±1.33	**8.70±1.34**
monks	31.10±7.03	22.80±7.89	**5.00±0.00**	5.00±0.00	5.00±0.00
nursery	243.90±20.79	122.90±6.14	**35.30±2.50**	36.60±2.32	-
promoters	5.80±3.22	2.20±0.92	2.50±0.71	-	-
spect	1.80±2.53	1.80±2.53	1.50±1.58	1.80±1.32	2.10±1.85
splice	64.90±7.49	16.90±3.11	**9.20±1.81**	-	-
tictactoe	48.20±9.44	24.70±3.89	15.50±4.25	12.20±1.87	**12.10±1.10**
titanic	7.70±1.64	5.80±1.03	3.50±1.43	3.30±0.95	-
vote	2.90±1.20	2.90±1.20	3.00±1.76	2.20±0.63	2.00±0.00

Summary

- Propose a new schema to order a subset of k discrete attributes
- Using all orderings of values of those k attributes as new extracted features, propose a novel decision tree classifier.
- Our proposed algorithm performs better than C4.5 in terms of error rate and tree complexity

Future Work

- Same idea can also be applied to C4.5Rules and Ripper