NBTree: A Naive Bayes/Decision-Tree Hybrid

Darin Morrison

April 17, 2007

Outline

1. Motivation
   - Problem and Solutions
2. Consideration of Existing Solutions
   - Naive-Bayes Classifiers
   - Decision-Trees
   - Learning Curves
3. New Solution: NBTree
   - Definition of NBTree
   - Performance
4. Summary

Outline

1. Motivation
   - Problem and Solutions
2. Consideration of Existing Solutions
   - Naive-Bayes Classifiers
   - Decision-Trees
   - Learning Curves
3. New Solution: NBTree
   - Definition of NBTree
   - Performance
4. Summary

Automatic Induction of Classifiers

Problem How can we generate a classifier from an arbitrarily sized database of labeled instances, where attributes are not necessarily independent?

Solutions

1. Naive-Bayes Classifiers
2. Decision-Trees (C4.5)
3. ?

Automatic Induction of Classifiers

Problem How can we generate a classifier from an arbitrarily sized database of labeled instances, where attributes are not necessarily independent?

Solutions

1. Naive-Bayes Classifiers
2. Decision-Trees (C4.5)
3. ?

Automatic Induction of Classifiers

Problem How can we generate a classifier from an arbitrarily sized database of labeled instances, where attributes are not necessarily independent?

Solutions

1. Naive-Bayes Classifiers
2. Decision-Trees (C4.5)
3. ?

Automatic Induction of Classifiers

Problem How can we generate a classifier from an arbitrarily sized database of labeled instances, where attributes are not necessarily independent?

Solutions

1. Naive-Bayes Classifiers
2. Decision-Trees (C4.5)
3. ?

Outline

1. Motivation
   - Problem and Solutions
2. Consideration of Existing Solutions
   - Naive-Bayes Classifiers
   - Decision-Trees
   - Learning Curves
3. New Solution: NBTree
   - Definition of NBTree
   - Performance
4. Summary

Naive-Bayes Classifiers

Pros

1. Fast
2. Induced classifiers are easy to interpret
3. Robust to irrelevant attributes
4. Uses evidence from many attributes

Cons

1. Assumes independence of attributes
2. Low performance ceiling on large databases

Outline

1 Motivation
- Problem and Solutions

2 Consideration of Existing Solutions
- Naive-Bayes Classifiers
- Decision-Trees
- Learning Curves

3 New Solution: NBTree
- Definition of NBTree
- Performance

4 Summary

Decision-Trees

Pros

1. Fast
2. Segmentation of data

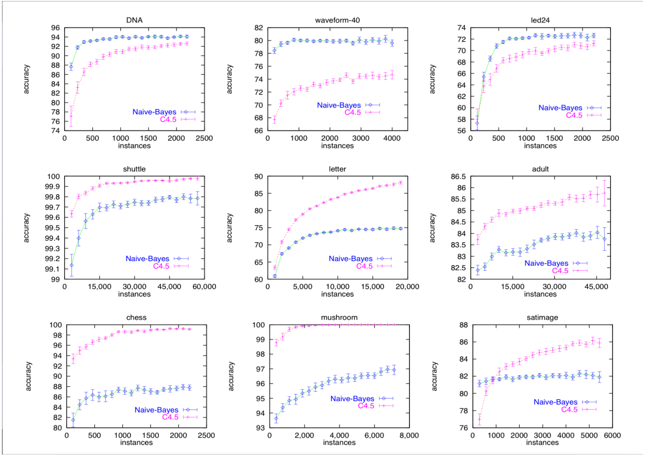
Cons

1. Fragmentation as number of splits becomes large
2. Interpretability goes down as number of splits increase

Outline

1. Motivation
   - Problem and Solutions
2. Consideration of Existing Solutions
   - Naive-Bayes Classifiers
   - Decision-Trees
   - Learning Curves
3. New Solution: NBTree
   - Definition of NBTree
   - Performance
4. Summary

Comparison between NB and C4.5 Learning Curves



Error bars represent 95% confidence intervals on accuracy

Outline

1. Motivation
   - Problem and Solutions
2. Consideration of Existing Solutions
   - Naive-Bayes Classifiers
   - Decision-Trees
   - Learning Curves
3. New Solution: NBTree
   - Definition of NBTree
   - Performance
4. Summary

Algorithm

- $dom(makeNBTree) \triangleq Set\ LabeledInstance$
- $cod(makeNBTree) \triangleq Tree\ Split\ NBC$

1. For each attribute X_i, evaluate the utility, $u(X_i)$, of a split on attribute X_i. For continuous attributes, a threshold is also found at this stage.
2. Let $j = argmax_i(u_i)$, *i.e.*, the attribute with the highest utility.
3. If u_j is not significantly better than the utility of the current node, create a Naive-Bayes classifier for the current node and return.
4. Partition the set of instances T according to the test on X_j. If X_j is continuous, a threshold split is used; if X_j is discrete, a multi-way split is made for all possible values.
5. For each child, call the algorithm recursively on the portion of T that matches the test leading to the child.

Algorithm

- $dom(makeNBTree) \triangleq Set\ LabeledInstance$
- $cod(makeNBTree) \triangleq Tree\ Split\ NBC$

1. For each attribute X_i, evaluate the utility, $u(X_i)$, of a split on attribute X_i. For continuous attributes, a threshold is also found at this stage.
2. Let $j = argmax_i(u_i)$, *i.e.*, the attribute with the highest utility.
3. If u_j is not significantly better than the utility of the current node, create a Naive-Bayes classifier for the current node and return.
4. Partition the set of instances T according to the test on X_j. If X_j is continuous, a threshold split is used; if X_j is discrete, a multi-way split is made for all possible values.
5. For each child, call the algorithm recursively on the portion of T that matches the test leading to the child.

Algorithm

- $dom(makeNBTree) \triangleq Set\ LabeledInstance$
- $cod(makeNBTree) \triangleq Tree\ Split\ NBC$

1. For each attribute X_i, evaluate the utility, $u(X_i)$, of a split on attribute X_i. For continuous attributes, a threshold is also found at this stage.
2. Let $j = argmax_i(u_i)$, *i.e.*, the attribute with the highest utility.
3. If u_j is not significantly better than the utility of the current node, create a Naive-Bayes classifier for the current node and return.
4. Partition the set of instances T according to the test on X_j. If X_j is continuous, a threshold split is used; if X_j is discrete, a multi-way split is made for all possible values.
5. For each child, call the algorithm recursively on the portion of T that matches the test leading to the child.

Algorithm

- $dom(makeNBTree) \triangleq Set\ LabeledInstance$
- $cod(makeNBTree) \triangleq Tree\ Split\ NBC$

1. For each attribute X_i, evaluate the utility, $u(X_i)$, of a split on attribute X_i. For continuous attributes, a threshold is also found at this stage.
2. Let $j = argmax_i(u_i)$, *i.e.*, the attribute with the highest utility.
3. If u_j is not significantly better than the utility of the current node, create a Naive-Bayes classifier for the current node and return.
4. Partition the set of instances T according to the test on X_j. If X_j is continuous, a threshold split is used; if X_j is discrete, a multi-way split is made for all possible values.
5. For each child, call the algorithm recursively on the portion of T that matches the test leading to the child.

Algorithm

- $dom(makeNBTree) \triangleq Set\ LabeledInstance$
- $cod(makeNBTree) \triangleq Tree\ Split\ NBC$

1. For each attribute X_i, evaluate the utility, $u(X_i)$, of a split on attribute X_i. For continuous attributes, a threshold is also found at this stage.
2. Let $j = argmax_i(u_i)$, *i.e.*, the attribute with the highest utility.
3. If u_j is not significantly better than the utility of the current node, create a Naive-Bayes classifier for the current node and return.
4. Partition the set of instances T according to the test on X_j. If X_j is continuous, a threshold split is used; if X_j is discrete, a multi-way split is made for all possible values.
5. For each child, call the algorithm recursively on the portion of T that matches the test leading to the child.

Algorithm

- $dom(makeNBTree) \triangleq Set\ LabeledInstance$
- $cod(makeNBTree) \triangleq Tree\ Split\ NBC$

1. For each attribute X_i, evaluate the utility, $u(X_i)$, of a split on attribute X_i. For continuous attributes, a threshold is also found at this stage.
2. Let $j = argmax_i(u_i)$, *i.e.*, the attribute with the highest utility.
3. If u_j is not significantly better than the utility of the current node, create a Naive-Bayes classifier for the current node and return.
4. Partition the set of instances T according to the test on X_j. If X_j is continuous, a threshold split is used; if X_j is discrete, a multi-way split is made for all possible values.
5. For each child, call the algorithm recursively on the portion of T that matches the test leading to the child.

Utility

Utility of Node
: Computed by discretizing the data and computing 5-fold cross-validation accuracy estimate of using NBC at node

Utility of Split
: Computed by weighted sum of utility of nodes, where weight given to node is proportional to num of instances that reach node

Significance
: Split is *significant* iff the relative reduction in error is greater than 5% and there are at least 30 instances in node

Intuition

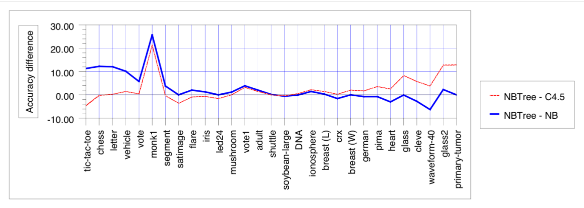
- Attempt to approximate whether generalization accuracy for NBC at each leaf is higher than single NBC at current node

Outline

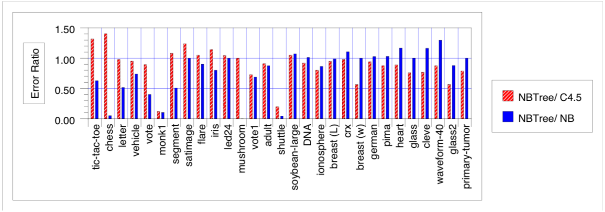
1. Motivation
   - Problem and Solutions
2. Consideration of Existing Solutions
   - Naive-Bayes Classifiers
   - Decision-Trees
   - Learning Curves
3. New Solution: NBTree
   - Definition of NBTree
   - Performance
4. Summary

Graphs

Dataset	No attrs	Train size	Test size	Dataset	No attrs	Train size	Test size	Dataset	No attrs	Train size	Test size
adult	14	30,162	15,060	breast (L)	9	277	CV-10	breast (W)	10	683	CV-10
chess	36	2,130	1,066	cleve	13	296	CV-10	crx	15	653	CV-10
DNA	180	2,000	1,186	flare	10	1,066	CV-10	german	20	1,000	CV-10
glass	9	214	CV-10	glass2	9	163	CV-10	heart	13	270	CV-10
ionosphere	34	351	CV-10	iris	4	150	CV-10	led24	24	200	3000
letter	16	15,000	5,000	monk1	6	124	432	mushroom	22	5,644	3,803
pima	8	768	CV-10	primary-tumor	17	132	CV-10	satimage	36	4,435	2,000
segment	19	2,310	CV-10	shuttle	9	43,500	14,500	soybean-large	35	562	CV-10
tic-tac-toe	9	958	CV-10	vehicle	18	846	CV-10	vote	16	435	CV-10
vote1	15	435	CV-10	waveform-40	40	300	4,700				



Graphs


Error Ratio
1.50
1.00
0.50
0.00
tic-tac-toe
chess
letter
vehicle
vote
monk1
segment
satimage
flare
iris
led24
mushroom
vote1
adult
shuttle
soybean-large
DNA
ionosphere
breast (L)
crx
breast (w)
german
pima
heart
glass
cleve
waveform-40
glass2
primary-tumor
NBTree/ C4.5
NBTree/ NB

Statistics

Average Accuracy

C4.5	81.91%
Naive-Bayes	81.69%
NBTree	84.47%

Number of Nodes per Tree

	C4.5	**NBTree**
letter	2109	251
adult	2213	137
DNA	131	3
led24	49	1

Summary

- NBTree appears to be a viable approach to inducing classifiers, where:
  - Many attributes are relevant for classification
  - Attributes are not necessarily independent
  - Database is large
  - Interpretability of classifier is important
- In practice, NBTrees are shown to scale to large databases and, in general, outperform Decision Trees and NBCs alone

References

R. Kohavi.
Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid
Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996.