Lazy Rule Learning

Nikolaus Korfhage

TECHNISCHE UNIVERSITÄT DARMSTADT

Introduction

Lazy Rule Learning Algorithm

Possible Improvements

Improved Lazy Rule Learning Algorithm

Implementation

Evaluation and Results

Rule Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

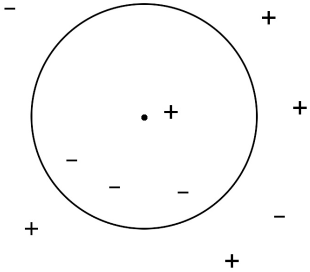
- Learns classifier once on the training data to classify test instances
- Classifier $\rightarrow$ Rule set
- Separate-and-conquer
  - add rules that cover many positive examples

Lazy Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

- Training data utilized by each query instance individually
- Classify instances simultaneously
- More time for classification

k-NN



$$w_i = \frac{1}{d(TestInstance, x_i)}$$

Lazy Rule Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

- Combine lazy learning and rule learning
  - produces many context-less rules
- Learn **one** rule
- Rule consists of conditions from test instance
- Rule should classify test instance correctly
- Example:
  - Test instance: <rainy, 68, 80, FALSE>
  - Rule: play = yes :- windy = FALSE.
- # rules = # instances to classify

LazyRule

TECHNISCHE UNIVERSITÄT DARMSTADT

LazyRule (*Instance*, *Examples*)
InitialRule = $\emptyset$
BestRule = *InitialRule*
for *Class* $\in$ *Classes*
    Conditions $\leftarrow$ PossibleConditions(*Instance*)
    NewRule = RefineRule (*Instance*, *Conditions*, *InitialRule*, *Class*)
    if *NewRule* $>$ *BestRule*
        BestRule = *NewRule*
return *BestRule*

PossibleConditions

PossibleConditions (*Instance*)
Conditions $\leftarrow \emptyset$
for *Attribute* $\in$ *Attributes*
    Value = AttributeValue (*Attribute*, *Instance*)
    if *Value* $\neq \emptyset$
        Conditions = *Conditions* $\cup$ $\{$(*Attribute* = *Value*)$\}$
return *Conditions*

REFINERULE

TECHNISCHE
UNIVERSITÄT
DARMSTADT

REFINERULE (*Instance*, *Conditions*, *Rule*, *Class*)
if *Conditions* $\neq \emptyset$
    BestRule = *Rule*
    BestCondtion = BESTCONDITION (*Rule*, *Conditions*)
    Refinement = *Rule* $\cup$ *BestCondtion*
    Evaluation = EVALUATERULE (*Refinement*)
    NewRule = $<$*Evaluation*, *Refinement*$>$
    if *NewRule* $>$ *BestRule*
        BestRule = *NewRule*
        REFINERULE (*Instance*, *Conditions* $\setminus$ *BestCondition*, *NewRule*, *Class*)
return *BestRule*

Numeric Attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

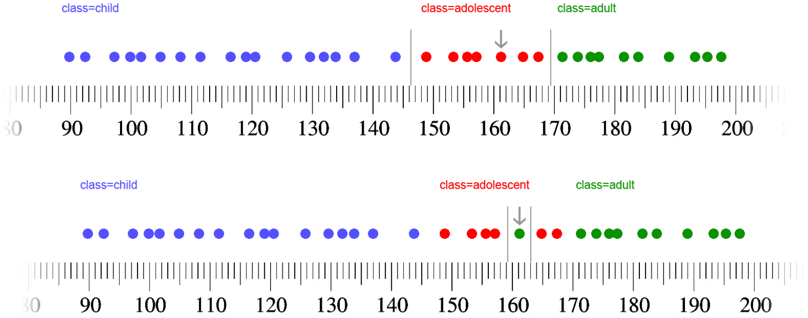
- Test instance:
  <sunny, 85 ,85, FALSE>

- Condition *outlook* = *sunny*
  $\rightarrow$ covers some training examples

- but condition *temperature* = 85
  $\rightarrow$ covers **no** training example

- Solution
  $\rightarrow$ infer two conditions, e.g. $temperature \geq 80 \wedge temperature < 90$

Numeric Attributes



Example of Learned Rules

play = yes :- humidity < 88, humidity >= 70, windy = FALSE.

play = yes :- temperature >= 72, temperature < 84.

play = yes :- outlook = overcast.

play = yes :- humidity >= 70, humidity < 82.5, windy = FALSE.

play = no :- outlook = sunny, temperature >= 70.5, temperature < 80.5.

Heuristics

- LazyRule evaluated with heuristics available in SeCo
- Laplace significantly better on most datasets
- Results:
  - Laplace 76.17
  - Linear Regression 72.44
  - *F*-Measure 70.62
  - Linear Cost 69.17
  - *m*-Estimate 68.81
  - Foil Gain 65.99
  - ...

Complexity

- # rules to check for one test instance: $O(c \cdot a^2)$
- # rules all instances: $O(c \cdot a^2 \cdot d)$
- # instances to check on first call RefineRule: $c \cdot a \cdot t$
- Decrease a or t

c : # classes
a : # attributes
d : # instances to classify
t : # training instances

Possible Improvements

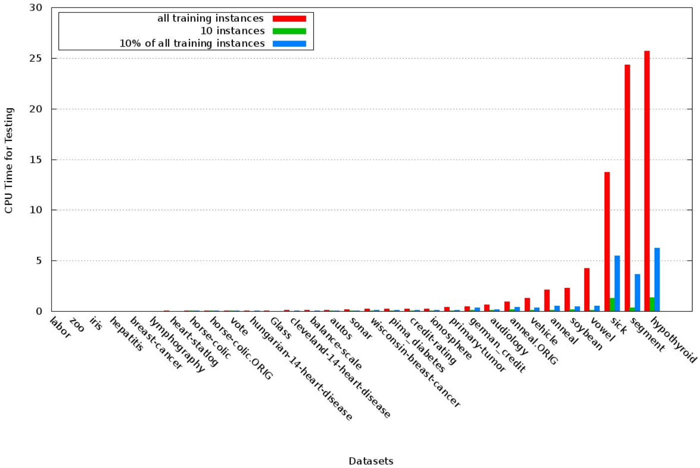
- Increase accuracy
  - Beam search
- Reduce execution time
  - Consider less data $\rightarrow$ random subset of training data
  - Preselect attributes
- Increase accuracy and decrease execution time
  - Learn rules on k-nearest neighbors

LazyRuleNN

- Learn rule on *k*-nearest neighbors
- Less training data to learn rule on
  $\rightarrow$ faster
- Consider only useful instances to learn rule on
  $\rightarrow$ higher accuracy

Computation Time

TECHNISCHE UNIVERSITÄT DARMSTADT



Accuracy

Dataset	LazyRule	LazyRuleNN, $k = 5$
iris	94.27	95.40
labor	85.67	88.20
balance-scale	85.44	86.16
heart-statlog	70.63	79.33 ∘
zoo	77.10	96.35 ∘
hepatitis	79.70	84.91
Glass	61.10	66.77
wisconsin-breast-cancer	91.47	96.81 ∘
lymphography	76.16	84.11
breast-cancer	72.15	73.37
autos	65.45	68.81
hungarian-14-heart-disease	80.41	82.52
primary-tumor	41.18	43.45
credit-rating	84.55	86.17
cleveland-14-heart-disease	75.59	83.08 ∘
pima-diabetes	69.91	73.84
vote	94.05	93.38
horse-colic.ORIG	71.92	63.02 •
audiology	48.30	66.02 ∘
vehicle	52.74	70.56 ∘
horse-colic	81.79	82.15
ionosphere	91.91	85.36 •
anneal.ORIG	89.21	94.42 ∘
vowel	29.42	93.67 ∘
sonar	64.14	82.42 ∘
anneal	90.75	98.24 ∘
german-credit	70.84	73.11
soybean	80.08	91.07 ∘
sick	93.88	96.28 ∘
segment	73.89	95.68 ∘
hypothyroid	92.86	93.43
Average	75.37	82.84

significance level: 0.01

improvement

degradation

Learned Rules

TECHNISCHE UNIVERSITÄT DARMSTADT

- Shorter rules for small *k*
- More empty rules

	LazyRule	LazyRuleNN $>$, k = 5	LazyRuleNN $\geq$, k = 5
Accuracy (%)	75.41	82.86	82.86
Average Rule Length	2.88	0.89	19.56
Empty Rules (%)	0.01	54.41	1.78

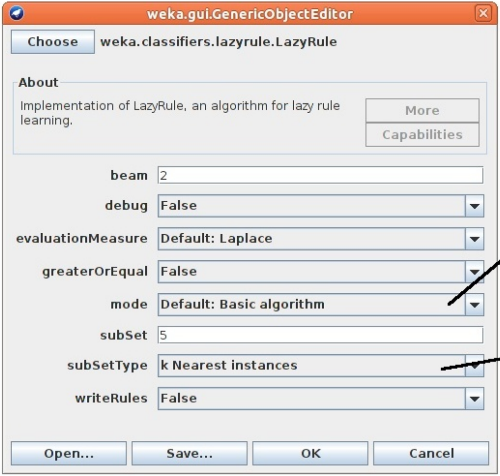
Implementation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Based on SECo-framework
  - Rules
  - Heuristics
- Weka:
  - Evaluation
  - Interface
  - kNN

Weka Interface

TECHNISCHE
UNIVERSITÄT
DARMSTADT



- hillclimbing
- beam
- all

- k nearest neighbors
- % nearest neighbors
- k random instances
- % random instances

Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- 37 datasets
- Evaluating possible improvements:
  - Weka: ten-fold CV
  - Corrected paired Student's t-Test
  - Leave-one-out cross-validation
- Comparing algorithms:
  - Weka: ten-fold CV
  - Friedmann test with post-hoc Nemenyi test

LazyRuleNN and other algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

Compared to:

- Decision tree algorithm J48 (C4.5)
- Separate-and-conquer rule learning algorithm JRip (RIPPER)
- k-nearest neighbor
- Weighted k-nearest neighbor
- $k = 1, 2, 3, 5, 10, 15, 25$

Results

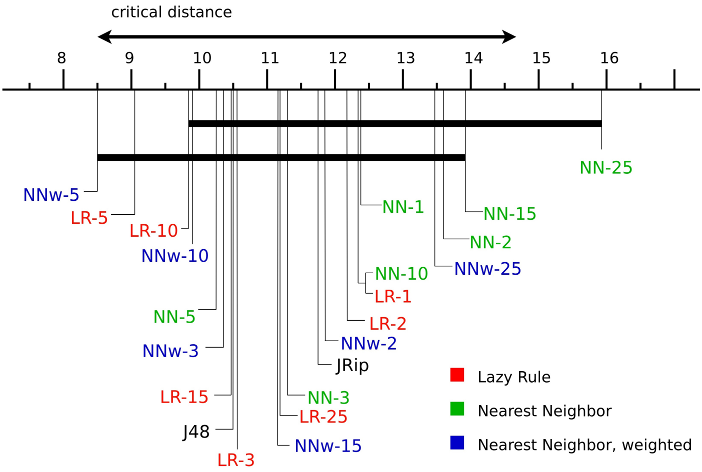
TECHNISCHE UNIVERSITÄT DARMSTADT

- Average accuracy

	$k = 1$	$k = 2$	$k = 3$	$k = 5$	$k = 10$	$k = 15$	$k = 25$
LazyRuleNN	83.31	83.02	84.00	83.90	82.94	82.47	82.09
kNN	83.35	82.75	83.73	83.47	81.73	80.23	78.18
kNN, weighted	83.35	83.29	84.16	84.27	83.69	83.29	82.14
JRip	83.09						
J48	83.37						

Results

TECHNISCHE UNIVERSITÄT DARMSTADT



Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Combines lazy learning and rule learning
- Improved lazy rule learning algorithm uses kNN
- Not significantly worse than considered learning algorithms
- Learns many context-free rules (one for each instance)
- May be useful for other projects (e.g. Learn-a-LOD)