Lazy Rule Learning
Nikolaus Korfhage
Introduction

Lazy Rule Learning Algorithm

Possible Improvements

Improved Lazy Rule Learning Algorithm

Implementation

Evaluation and Results
Rule Learning

- 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

- Training data utilized by each query instance individually
- Classify instances simultaneously
- More time for classification
\( k\text{-NN} \)

\[ W_i = \frac{1}{d(\text{TestInstance}, x_i)} \]
Lazy Rule Learning

- 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**

LazyRule (Instance, Examples)

InitialRule = ∅

BestRule = InitialRule

for Class ∈ Classes

    Conditions ← POSSIBLECONDITIONS(Instance)

    NewRule = REFINERULE (Instance, Conditions, InitialRule, Class)

    if NewRule > BestRule

        BestRule = NewRule

return BestRule
POSSIBLE CONDITIONS

POSSIBLE CONDITIONS (Instance)

Conditions ← ∅

for Attribute ∈ Attributes
    Value = ATTRIBUTE VALUE (Attribute, Instance)
    if Value ≠ ∅
        Conditions = Conditions ∪ {(Attribute = Value)}

return Conditions
**RefineRule**

**RefineRule** (*Instance*, *Conditions*, *Rule*, *Class*)

```plaintext
if *Conditions* ≠ ∅

  *BestRule* = *Rule*
  *BestCondition* = BESTCONDITION (*Rule*, *Conditions*)
  *Refinement* = *Rule* ⨆ *BestCondition*
  *Evaluation* = EVALUATERULE (*Refinement*)
  *NewRule* = <*Evaluation*, *Refinement*>

if *NewRule* > *BestRule*

  *BestRule* = *NewRule*

  **RefineRule** (*Instance*, *Conditions* \ *BestCondition*, *NewRule*, *Class*)

**return** *BestRule*
```
Numeric Attributes

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

▶ Condition $outlook = sunny$
  → covers some training examples

▶ but condition $temperature = 85$
  → covers no training example

▶ Solution
  → infer two conditions, e.g. $temperature \geq 80 \land temperature < 90$
Numeric Attributes

![Diagram showing distributions for different classes: child, adolescent, adult. The diagram includes measures on a scale from 80 to 200.

The diagram illustrates the distribution of heights in three categories: child, adolescent, and adult. Each category is represented with different colored dots aligned along a scale from 80 to 200. The child class has blue dots, the adolescent class has red dots, and the adult class has green dots. The distribution patterns vary across the classes, indicating different growth stages and average heights.]
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 `REFINE_RULE`: $c \cdot a \cdot t$
- 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 → 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
  → faster
- Consider only useful instances to learn rule on
  → higher accuracy
Computation Time
## Accuracy

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

significance level: 0.01

- improvement
- degradation
Learned Rules

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

<table>
<thead>
<tr>
<th></th>
<th>LAZYRULE</th>
<th>LAZYRULENN $\geq, k = 5$</th>
<th>LAZYRULENN $\geq, k = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>75.41</td>
<td>82.86</td>
<td>82.86</td>
</tr>
<tr>
<td>Average Rule Length</td>
<td>2.88</td>
<td>0.89</td>
<td>19.56</td>
</tr>
<tr>
<td>Empty Rules (%)</td>
<td>0.01</td>
<td>54.41</td>
<td>1.78</td>
</tr>
</tbody>
</table>
Implementation

- Based on SECO-framework
  - Rules
  - Heuristics

- Weka:
  - Evaluation
  - Interface
  - kNN
Weka Interface

- hillclimbing
- beam
- all
- k nearest neighbors
- % nearest neighbors
- k random instances
- % random instances
Evaluation

- 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

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

#### Average accuracy

<table>
<thead>
<tr>
<th>Method</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
<th>$k = 5$</th>
<th>$k = 10$</th>
<th>$k = 15$</th>
<th>$k = 25$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lazy RuleNN</td>
<td>83.31</td>
<td>83.02</td>
<td>84.00</td>
<td>83.90</td>
<td>82.94</td>
<td>82.47</td>
<td>82.09</td>
</tr>
<tr>
<td>kNN</td>
<td>83.35</td>
<td>82.75</td>
<td>83.73</td>
<td>83.47</td>
<td>81.73</td>
<td>80.23</td>
<td>78.18</td>
</tr>
<tr>
<td>kNN, weighted</td>
<td>83.35</td>
<td>83.29</td>
<td>84.16</td>
<td>84.27</td>
<td>83.69</td>
<td>83.29</td>
<td>82.14</td>
</tr>
<tr>
<td>JRip</td>
<td>83.09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J48</td>
<td>83.37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results
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)