Inductive Rule Learning in a Nutshell

Johannes Fürnkranz

- Introduction
  - Learning Rule Sets
  - Terminology
  - Coverage Spaces

- Separate-and-Conquer Rule Learning
  - Covering algorithm
  - Top-Down Hill-Climbing
  - Rule Learning Heuristics
  - Overfitting and Pruning
Inductive Machine Learning algorithms induce a classifier from labeled training examples. The classifier generalizes the training examples, i.e., it is able to assign labels to new cases.

An inductive learning algorithm searches in a given family of hypotheses (e.g., decision trees, neural networks) for a member that optimizes given quality criteria (e.g., estimated predictive accuracy or misclassification costs).
Concept Learning

- **Given:**
  - Positive Examples $E^+$
  - examples for the concept to learn (e.g., days with golf)
  - Negative Examples $E^-$
  - counter-examples for the concept (e.g., days without golf)
  - Hypothesis Space $H$
  - a (possibly infinite) set of candidate hypotheses
  - e.g., rules, rule sets, decision trees, linear functions, neural networks, ...

- **Find:**
  - Find the target hypothesis $h \in H$
  - the target hypothesis is the concept that was used (or could have been used) to generate the training examples
Conjunctive Rule

\[
\text{if } (\text{att}_i = \text{val}_{il}) \text{ and } (\text{att}_j = \text{val}_{jJ}) \quad \text{then } +
\]

Body of the rule (IF-part)
- contains a conjunction of conditions
- a condition typically consists of comparison of attribute values

Head of the rule (THEN-part)
- contains a prediction
- typically + if object belongs to concept, – otherwise

- Coverage
  - A rule is said to \textit{cover} an example if the example satisfies the conditions of the rule.

- Prediction
  - If a rule covers an example, the rule's head is predicted for this example.
A sample task

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Outlook</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play Golf?</th>
</tr>
</thead>
<tbody>
<tr>
<td>hot</td>
<td>sunny</td>
<td>high</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>hot</td>
<td>sunny</td>
<td>high</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>hot</td>
<td>overcast</td>
<td>high</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>cool</td>
<td>rain</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>cool</td>
<td>overcast</td>
<td>normal</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>mild</td>
<td>sunny</td>
<td>high</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>cool</td>
<td>sunny</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>mild</td>
<td>rain</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>mild</td>
<td>sunny</td>
<td>normal</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>mild</td>
<td>overcast</td>
<td>high</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>hot</td>
<td>overcast</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>mild</td>
<td>rain</td>
<td>high</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>cool</td>
<td>rain</td>
<td>normal</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>mild</td>
<td>rain</td>
<td>high</td>
<td>false</td>
<td>yes</td>
</tr>
</tbody>
</table>
A Simple Solution

- The solution is
  - a set of rules
    - that is complete and consistent on the training examples
  → it must be part of the version space
- but it does not generalize to new examples!
A Better Solution

IF Outlook = overcast THEN yes
IF Humidity = normal AND Outlook = sunny THEN yes
IF Outlook = rainy AND Windy = false THEN yes
Separate-and-Conquer Rule Learning

- Learn a set of rules, one by one

1. Start with an empty theory $T$ and training set $E$
2. Learn a single (consistent) rule $R$ from $E$ and add it to $T$
3. If $T$ is satisfactory (complete), return $T$
4. Else:
   - Separate: Remove examples explained by $R$ from $E$
   - Conquer: goto 2.

- One of the oldest family of learning algorithms
  - goes back AQ (Michalski, 60s)
  - FRINGE, PRISM and CN2: relation to decision trees (80s)
  - popularized in ILP (FOIL and PROGOL, 90s)
  - RIPPER brought in good noise-handling

- Different learners differ in how they find a single rule
Relaxing Completeness and Consistency

- So far we have always required a learner to learn a complete and consistent theory
  - e.g., one rule that covers all positive and no negative examples
  - This is not always a good idea (→ overfitting)

- Example:
  - Training set with 200 examples, 100 positive and 100 negative
  - **Theory A** consists of 100 complex rules, each covering a single positive example and no negatives
    → Theory A is complete and consistent on the training set
  - **Theory B** consists of a single rule, covering 99 positive and 1 negative example
    → Theory B is incomplete and inconsistent on the training set

- Which one will generalize better to unseen examples?
Separate-and-Conquer Rule Learning

(i) Original Data

(iv) Step 3

Terminology

- **training examples**
  - \( P \): total number of positive examples
  - \( N \): total number of negative examples

- **examples covered by the rule (predicted positive)**
  - **true positives** \( p \): positive examples covered by the rule
  - **false positives** \( n \): negative examples covered by the rule

- **examples not covered the rule (predicted negative)**
  - **false negatives** \( P-p \): positive examples not covered by the rule
  - **true negatives** \( N-n \): negative examples not covered by the rule

<table>
<thead>
<tr>
<th></th>
<th>predicted +</th>
<th>predicted -</th>
</tr>
</thead>
<tbody>
<tr>
<td>class +</td>
<td>( p ) (true positives)</td>
<td>( P-p ) (false negatives)</td>
</tr>
<tr>
<td>class -</td>
<td>( n ) (false positives)</td>
<td>( N-n ) (true negatives)</td>
</tr>
<tr>
<td></td>
<td>( p + n )</td>
<td>( P+N - (p+n) )</td>
</tr>
</tbody>
</table>
Coverage Spaces

- good tool for visualizing properties of covering algorithms
- each point is a theory covering $p$ positive and $n$ negative examples

- **perfect theory**: all positive and no negative examples are covered
- **universal theory**: all examples are covered
- **random theory**: maintain $P/(P+N)\%$ positive and $N/(P+N)\%$ negative examples
- **iso-accuracy**: cover same amount of positive and negative examples
- **empty theory**: no examples are covered
- **opposite theory**: all negative and no positive examples are covered

\[ \text{covered positive examples} \]
\[ \text{covered negative examples} \]

\[ P \]
\[ N \]
Covering Strategy

- **Covering or Separate-and-Conquer** rule learning algorithms learn one rule at a time.
- This corresponds to a path in coverage space:
  - The **empty theory** $R_0$ (no rules) corresponds to (0,0).
  - Adding one rule **never decreases** $p$ or $n$ because adding a rule covers more examples (generalization).
  - The **universal theory** $R^+$ (all examples are positive) corresponds to (N,P).
Top-Down Hill-Climbing

- successively extends a rule by adding conditions

- This corresponds to a path in coverage space:
  - The rule $p: \neg \text{true}$ covers all examples (universal theory)
  - Adding a condition never increases $p$ or $n$ (specialization)
  - The rule $p: \neg \text{false}$ covers no examples (empty theory)

- which conditions are selected depends on a *heuristic function* that estimates the quality of the rule
Rule Learning Heuristics

- Adding a rule should
  - increase the number of covered negative examples as little as possible (do not decrease consistency)
  - increase the number of covered positive examples as much as possible (increase completeness)

- An evaluation heuristic should therefore trade off these two extremes
  - Example: Laplace heuristic \( h_{Lap} = \frac{p+1}{p+n+2} \)
    - grows with \( p \to \infty \)
    - grows with \( n \to 0 \)
  - Example: Precision \( h_{prec} = \frac{p}{p+n} \)
    - is not a good heuristic. Why?
### Example

<table>
<thead>
<tr>
<th>Condition</th>
<th>p</th>
<th>n</th>
<th>Precision</th>
<th>Laplace</th>
<th>p-n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hot</td>
<td>2</td>
<td>2</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0</td>
</tr>
<tr>
<td>Mild</td>
<td>3</td>
<td>1</td>
<td>0.7500</td>
<td>0.6667</td>
<td>2</td>
</tr>
<tr>
<td>Cold</td>
<td>4</td>
<td>2</td>
<td>0.6667</td>
<td>0.6250</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outlook =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sunny</td>
<td>2</td>
<td>3</td>
<td>0.4000</td>
<td>0.4286</td>
<td>-1</td>
</tr>
<tr>
<td>Overcast</td>
<td>4</td>
<td>0</td>
<td>1.0000</td>
<td>0.8333</td>
<td>4</td>
</tr>
<tr>
<td>Rain</td>
<td>3</td>
<td>2</td>
<td>0.6000</td>
<td>0.5714</td>
<td>1</td>
</tr>
<tr>
<td>Humidity =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>3</td>
<td>4</td>
<td>0.4286</td>
<td>0.4444</td>
<td>-1</td>
</tr>
<tr>
<td>Normal</td>
<td>6</td>
<td>1</td>
<td>0.8571</td>
<td>0.7778</td>
<td>5</td>
</tr>
<tr>
<td>Windy =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>3</td>
<td>3</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0</td>
</tr>
<tr>
<td>False</td>
<td>6</td>
<td>2</td>
<td>0.7500</td>
<td>0.7000</td>
<td>4</td>
</tr>
</tbody>
</table>

- **Heuristics Precision and Laplace**
  - add the condition Outlook= Overcast to the (empty) rule
  - stop and try to learn the next rule
- **Heuristic Accuracy / \( p - n \)**
  - adds Humidity = Normal
  - continue to refine the rule (until no covered negative)
Isometrics in Coverage Space

- Isometrics are lines that connect points for which a function in $p$ and $n$ has equal values.

**Examples:**
Isometrics for heuristics $h_p = p$ and $h_n = -n$
Top-Down Hill-Climbing

- **Top-Down Strategy**: A rule is successively *specialized*

1. Start with the universal rule R that covers all examples
2. Evaluate all possible ways to add a condition to R
3. Choose the best one (according to some heuristic)
4. If R is satisfactory, return it
5. Else goto 2.

- Almost all greedy s&c rule learning systems use this strategy
Precision (Confidence)

- **basic idea:**
  percentage of positive examples among covered examples

- **effects:**
  - rotation around origin (0,0)
  - all rules with same angle equivalent
  - in particular, all rules on $P/N$ axes are equivalent

\[ h_{\text{Prec}} = \frac{p}{p+n} \]
Accuracy

- **basic idea:**
  percentage of correct classifications
  *(covered positives plus uncovered negatives)*

- **effects:**
  - isometrics are parallel to 45° line
  - covering one positive example is as good as not covering one negative example

\[
h_{\text{Acc}} = \frac{p + (N - n)}{P + N} \approx p - n
\]

Why are they equivalent?

\[
h_{\text{Acc}} = \frac{P}{P + N}
\]

\[
h_{\text{Acc}} = \frac{1}{2}
\]

\[
h_{\text{Acc}} = \frac{N}{P + N}
\]
Weighted Relative Accuracy

- **basic idea:**
  normalize accuracy with the class distribution

- **effects:**
  - isometrics are parallel to diagonal
  - covering $x\%$ of the positive examples is considered to be as good as not covering $x\%$ of the negative examples

\[
h_{WRA} = \frac{p+n}{P+N} \left( \frac{p}{p+n} - \frac{P}{P+N} \right) \approx \frac{p}{P} - \frac{n}{N}
\]
Correlation

- **basic idea:** measure correlation coefficient of predictions with target
- **effects:**
  - non-linear isometrics
  - in comparison to WRA
  - prefers rules near the edges
  - steepness of connection of intersections with edges increases
  - equivalent to $\chi^2$
Overfitting

- Overfitting
  - Given
    - a fairly general model class
    - enough degrees of freedom
  - you can always find a model that explains the data
    - even if the data contains error (noise in the data)
    - in rule learning: each example is a rule

- Such concepts do not generalize well!
  → Pruning
Overfitting - Illustration

Polynomial degree 1
(linear function)

Polynomial degree 4
(n-1 degrees can always fit n points)

Prediction for this value of $x$?

or here?

here
Overfitting Avoidance

- A perfect fit to the data is not always a good idea
  - data could be imprecise
    - e.g., random noise
  - the hypothesis space may be inadequate
    - a perfect fit to the data might not even be possible
    - or it may be possible but with bad generalization properties
      (e.g., generating one rule for each training example)

- Thus it is often a good idea to avoid a perfect fit of the data
  - fitting polynomials so that
    - not all points are exactly on the curve
  - learning concepts so that
    - not all positive examples have to be covered by the theory
    - some negative examples may be covered by the theory
Overfitting Avoidance

- learning concepts so that
  - not all positive examples have to be covered by the theory
  - some negative examples may be covered by the theory
Complexity of Concepts

- For simpler concepts there is less danger that they are able to overfit the data
  - for a polynomial of degree \( n \) one can choose \( n+1 \) parameters in order to fit the data points

→ many learning algorithms focus on learning simple concepts
  - a short rule that covers many positive examples (but possibly also a few negatives) is often better than a long rule that covers only a few positive examples

- **Pruning**: Complex rules will be simplified
  - **Pre-Pruning**: during learning
  - **Post-Pruning**: after learning
Pre-Pruning

- keep a theory simple *while* it is learned
  - decide when to **stop adding conditions** to a rule *(relax consistency constraint)*
  - decide when to **stop adding rules** to a theory *(relax completeness constraint)*
- efficient but not accurate
Post Pruning

... Literals

... Post-Pruning Decisions

... Pre-Pruning Decisions
Post-Pruning: Example

<table>
<thead>
<tr>
<th>Rule</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF T=hot AND H=high AND O=sunny AND W=false THEN no</td>
<td></td>
</tr>
<tr>
<td>IF T=hot AND H=high AND O=sunny AND W=true THEN no</td>
<td></td>
</tr>
<tr>
<td>IF T=hot AND H=high AND O=overcast AND W=false THEN yes</td>
<td></td>
</tr>
<tr>
<td>IF T=cool AND H=normal AND O=rain AND W=false THEN yes</td>
<td></td>
</tr>
<tr>
<td>IF T=cool AND H=normal AND O=overcast AND W=true THEN yes</td>
<td></td>
</tr>
<tr>
<td>IF T=mild AND H=high AND O=sunny AND W=false THEN no</td>
<td></td>
</tr>
<tr>
<td>IF T=cool AND H=normal AND O=overcast AND W=true THEN yes</td>
<td></td>
</tr>
<tr>
<td>IF T=mild AND H=normal AND O=rain AND W=false THEN yes</td>
<td></td>
</tr>
<tr>
<td>IF T=mild AND H=high AND O=overcast AND W=true THEN yes</td>
<td></td>
</tr>
<tr>
<td>IF T=hot AND H=normal AND O=overcast AND W=false THEN yes</td>
<td></td>
</tr>
<tr>
<td>IF T=mild AND H=high AND O=rain AND W=true THEN no</td>
<td></td>
</tr>
<tr>
<td>IF T=cool AND H=normal AND O=rain AND W=true THEN no</td>
<td></td>
</tr>
<tr>
<td>IF T=mild AND H=high AND O=rain AND W=false THEN yes</td>
<td></td>
</tr>
</tbody>
</table>
Post-Pruning: Example

IF $H=\text{high}$ AND $O=\text{sunny}$ THEN no

IF $O=\text{rain}$ AND $W=\text{true}$ THEN no

ELSE yes
Incremental Reduced Error Pruning

Pre-Pruning

Post-Pruning

Integrating Pre- and Post-Pruning

I-REP tries to combine the advantages of pre- and post-pruning
Multi-class problems

- **GOAL:** discriminate \( c \) classes from each other

- **PROBLEM:** many learning algorithms are only suitable for binary (2-class) problems

- **SOLUTION:** "Class binarization": Transform an \( c \)-class problem into a series of 2-class problems
One-against-all binarization

Treat each class as a separate concept:

- $c$ binary problems, one for each class
- label examples of one class positive, all others negative
Pairwise Classification

- $c(c-1)/2$ problems
- each class against each other class

- ✔ smaller training sets
- ✔ simpler decision boundaries
- ✔ larger margins
### Accuracy

- **error rates** on 20 datasets with 4 or more classes
  - 10 significantly better ($p > 0.99$, McNemar)
  - 2 significantly better ($p > 0.95$)
  - 8 equal
  - never (significantly) worse

<table>
<thead>
<tr>
<th>dataset</th>
<th>Ripper unord.</th>
<th>ordered</th>
<th>$R^2$</th>
<th>ratio</th>
<th>$&lt;$</th>
</tr>
</thead>
<tbody>
<tr>
<td>abalone</td>
<td>81.03</td>
<td>82.18</td>
<td>72.99</td>
<td>0.888</td>
<td>++</td>
</tr>
<tr>
<td>covertype</td>
<td>35.37</td>
<td>38.50</td>
<td>33.20</td>
<td>0.862</td>
<td>++</td>
</tr>
<tr>
<td>letter</td>
<td>15.22</td>
<td>15.75</td>
<td>7.85</td>
<td>0.498</td>
<td>++</td>
</tr>
<tr>
<td>sat</td>
<td>14.25</td>
<td>17.05</td>
<td>11.15</td>
<td>0.654</td>
<td>++</td>
</tr>
<tr>
<td>shuttle</td>
<td>0.03</td>
<td>0.06</td>
<td>0.02</td>
<td>0.375</td>
<td>=</td>
</tr>
<tr>
<td>vowel</td>
<td>64.94</td>
<td>53.25</td>
<td>53.46</td>
<td>1.004</td>
<td>=</td>
</tr>
<tr>
<td>car</td>
<td>5.79</td>
<td>12.15</td>
<td>2.26</td>
<td>0.186</td>
<td>++</td>
</tr>
<tr>
<td>glass</td>
<td>35.51</td>
<td>34.58</td>
<td>25.70</td>
<td>0.743</td>
<td>++</td>
</tr>
<tr>
<td>image</td>
<td>4.15</td>
<td>4.29</td>
<td>3.46</td>
<td>0.808</td>
<td>+</td>
</tr>
<tr>
<td>lr spectrometer</td>
<td>64.22</td>
<td>61.39</td>
<td>53.11</td>
<td>0.865</td>
<td>++</td>
</tr>
<tr>
<td>optical</td>
<td>7.79</td>
<td>9.48</td>
<td>3.74</td>
<td>0.394</td>
<td>++</td>
</tr>
<tr>
<td>page-blocks</td>
<td>2.85</td>
<td>3.38</td>
<td>2.76</td>
<td>0.816</td>
<td>++</td>
</tr>
<tr>
<td>solar flares (c)</td>
<td>15.91</td>
<td>15.91</td>
<td>15.77</td>
<td>0.991</td>
<td>=</td>
</tr>
<tr>
<td>solar flares (m)</td>
<td>4.90</td>
<td>5.47</td>
<td>5.04</td>
<td>0.921</td>
<td>=</td>
</tr>
<tr>
<td>soybean</td>
<td>8.79</td>
<td>8.79</td>
<td>6.30</td>
<td>0.717</td>
<td>++</td>
</tr>
<tr>
<td>thyroid (hyper)</td>
<td>1.25</td>
<td>1.49</td>
<td>1.11</td>
<td>0.749</td>
<td>+</td>
</tr>
<tr>
<td>thyroid (hypo)</td>
<td>0.64</td>
<td>0.56</td>
<td>0.53</td>
<td>0.955</td>
<td>=</td>
</tr>
<tr>
<td>thyroid (repl.)</td>
<td>1.17</td>
<td>0.98</td>
<td>1.01</td>
<td>1.026</td>
<td>=</td>
</tr>
<tr>
<td>vehicle</td>
<td>28.25</td>
<td>30.38</td>
<td>29.08</td>
<td>0.957</td>
<td>=</td>
</tr>
<tr>
<td>yeast</td>
<td>44.00</td>
<td>42.39</td>
<td>41.78</td>
<td>0.986</td>
<td>=</td>
</tr>
<tr>
<td>average</td>
<td>21.80</td>
<td>21.90</td>
<td>18.52</td>
<td>0.770</td>
<td>=</td>
</tr>
</tbody>
</table>
Summary

- **Rules** can be learned via top-down hill-climbing
  - add one condition at a time until the rule covers no more negative exs.
- **Heuristics** are needed for guiding the search
  - can be visualize through isometrics in coverage space
- **Rule Sets** can be learned one rule at a time
  - using the covering or separate-and conquer strategy
- **Overfitting** is a serious problem for all machine learning algorithms
  - too close a fit to the training data may result in bad generalizations
- **Pruning** can be used to fight overfitting
  - Pre-pruning and post-pruning can be efficiently integrated
- **Multi-class problems** can be addressed by multiple rule sets
  - one-against-all classification or pairwise classification