Ensembles of Decision Rules — General Framework for Rule Induction

Krzysztof Dembczyński, Jerzy Błaszczyński, Wojciech Kotłowski, Roman Słowiński, Marcin Szeląg

Intelligent Decision Support Systems,
Institute of Computing Science,
Poznań University of Technology

October 17th, 2006
Those who ignore Statistics are condemned to reinvent it

Brad Efron
**Decision rule** is a logical expression in the form:

\[
\text{if } [\text{conditions}], \text{ then } [\text{decision}].
\]

If an object satisfies conditions of the rule, then it is **assigned** to the recommended class.

Otherwise the object remains **unassigned**.
Decision rules were common in the early machine learning approaches (AQ, CN2, RIPPER).

The most popular decision rule induction algorithms are based on a sequential covering procedure (also known as separate-and-conquer strategy).

Decision rule models are widely considered in the Rough Set approaches to knowledge discovery and in Logical Analysis of Data where they are called patterns.

Wide interest in decision rules may be explained by their simplicity and ease in interpretation.

However, it seems that decision trees are much more popular in machine leaning approaches.
Ensembles of decision rules described here follow a specific and original approach to decision rule generation:

Single rule is treated as a subsidiary, base classifier in the ensemble that indicates only one of the decision classes.
**Ensemble methods** became a very popular and efficient approach to machine learning problems.

Ensembles consist in forming committees of simple learning and classification procedures often referred to as **base** (or **weak**) **learners** (or **classifiers**).

The ensemble members are applied to a prediction task and their individual outputs are then aggregated to one output of the whole ensemble.

The aggregation is computed as a linear combination of outputs.

The most popular base learners are decision trees.

There are several approaches to construction of the ensemble like **bagging** and **boosting**.

Ensembles are often treated as **off-the-shelf methods-of-choice**.
Ensembles of decision rules

- Ensemble consists of **single decision rules**.
- Variant of **Forward Stagewise Additive Modeling** (by Hastie, Tibshirani and Friedman) is used in construction of the ensemble.
- **Single rule** is created in each iteration of Forward Stagewise Additive Modeling.
- The rules are used in a prediction procedure by linear combination of their outputs.
- The outputs of the rules can be additionally parametrized.
Ensembles of decision rules

- Ensembles of decision rules are **competitive** with other machine learning methods.
- The rules are **easy in interpretation**.
- The algorithm is characterized by **low computational cost**.
- The approach is very **flexible**, for example, one can construct ensembles of **classification** or **regression** rules.
There are some similar approaches:

- **RuleFit** (by Friedman and Popescu): based on Forward Stagewise Additive Modeling; the decision trees are used as base classifiers, and then each node (interior and terminal) of each resulting tree produces a rule; it is setup by the conjunction of conditions associated with all of the edges on the path from the root to that node; rule ensemble is fitted by gradient directed regularization.

- **SLIPPER** (by Cohen and Singer): uses AdaBoost schema that is a specific case of Forward Stagewise Additive Modeling to produce an ensemble of decision rules.

- **Lightweight Rule Induction** (by Weiss and Indurkhya): uses specific reweighing schema and DNF-formulas for single rules.
Ensembles of decision rules can be seen as a connection of three very important issues in machine learning:

- induction of decision rules by sequential covering algorithms,
- boosting weak learners,
- gradient boosting machines.

In our opinion the methodology is original, however, most of theoretical results are based on those achieved by Friedman, Popescu, Hastie, Tibshirani, Schapire and Freund.

The originality comes from the fact that we are applying specific weak learner that is a single decision rule.

Our goal is to present a general framework for rule induction.
1. Problem Statement

2. Ensembles of Decision Rules

3. Experimental Results
The aim is to:

- predict the unknown value of an attribute \( y \) (called *output*, *response variable* or *decision attribute*) of an object
- using the known joint values of other attributes (called *predictors*, *condition attributes* or *independent variables*)

\[
x = (x_1, x_2, \ldots, x_n).
\]

The goal of a learning task is to find a function \( F(x) \) using a set of training examples \( \{y_i, x_i\}_{1}^{N} \) that predicts accurately \( y \).
The optimal prediction procedure is given by:

\[ F^*(x) = \arg \min_{F(x)} E_{yx}L(y, F(x)) \]

where the expected value \( E_{yx} \) is over joint distribution of all attributes \((y, x)\) for the data to be predicted.

\( L(y, F(x)) \) is a loss or cost for predicting \( F(x) \) when the actual value is \( y \).

The learning procedure tries to construct \( F(x) \) to be the best possible approximation of \( F^*(x) \).
We consider **binary classification** problem, in which \( y \in \{-1, 1\} \) and **regression** problem, in which \( y \in \mathbb{R} \).

The typical loss in classification tasks is 0-1 loss:

\[
L(y, F(x)) = \begin{cases} 
0 & y = F(x), \\
1 & y \neq F(x).
\end{cases}
\]

The typical loss in regression tasks is squared loss:

\[
L(y, F(x)) = (y - F(x))^2
\]
Loss functions in classification tasks:

- **Sigmoid loss**
  \[
  L(y, F(x)) = \frac{1}{1 + \exp(\beta \cdot y \cdot F(x))}
  \]

- **SVM loss**
  \[
  L(y, F(x)) = \begin{cases} 
  0, & y \cdot F(x) \geq 1, \\
  1 - y \cdot F(x), & y \cdot F(x) < 1, 
  \end{cases}
  \]

- **Exponential loss**
  \[
  L(y, F(x)) = \exp(-\beta \cdot y \cdot F(x))
  \]

- **Binomial log-likelihood**
  \[
  L(y, F(x)) = \log(1 + \exp(-\beta \cdot y \cdot F(x)))
  \]
Loss
0−1 loss
Sigmoid loss
SVM loss
Exponential loss
Binomial log−likelihood loss
Loss functions in regression tasks:

- Least absolute deviation
  \[ L(y, F(x)) = |y - F(x)| \]

- SVM regression loss
  \[
  L(y, F(x)) = \begin{cases} 
  0, & |y - F(x)| < 1, \\
  |y - F(x)| - 1, & |y - F(x)| \geq 1,
  \end{cases}
  \]

- Huber loss
  \[
  L(y, F(x)) = \begin{cases} 
  \frac{1}{2} (y - F(x))^2, & |y - F(x)| \leq \sigma, \\
  \sigma (|y - F(x)| - \sigma/2), & |y - F(x)| > \sigma.
  \end{cases}
  \]
Squared loss
Least absolute deviance
SVM regression loss
Huber loss
Linear models: Linear models are among the most popular for data fitting:

\[ F(x) = a_0 + \sum_{m=1}^{M} a_m \cdot f_m(x) \]

where \( \{a_m\}_0^M \) are parameters to be estimated and \( \{f_m\}_0^M \) may be the original measured variables and/or selected functions constructed from them.
Linear models:

The parameters of the linear model are estimated through:

\[
\{ \hat{a}_m \}_0^M = \arg \min_{\{a_m\}_0^M} \sum_{i=1}^{N} L(y_i, a_0 + \sum_{m=1}^{M} a_m \cdot f_m(x)) + \lambda \cdot P(\{a_m\}_1^M)
\]

where the first term measures the loss on the training sample, and the second term is used for regularization.

- \( \lambda \) controls the degree of regularization
- commonly employed penalty functions \( P(\{a_m\}_1^M) \) are:

\[
P_1(\{a_m\}_1^M) = \sum_{m=1}^{M} |a_m| \quad P_2(\{a_m\}_1^M) = \sum_{m=1}^{M} |a_m|^2
\]
1 Problem Statement

2 Ensembles of Decision Rules

3 Experimental Results
**Decision rule** is the simplest and the most comprehensive representation of knowledge in the form of logical expression:

\[
\text{if [conditions], then [decision].}
\]

**Example**

\[
\begin{align*}
\text{if } & \text{ duration } \geq 31.5 \\
\text{and savings status } & \neq \text{ no known savings} \\
\text{and savings status } & \not\in (500, 1000) \\
\text{and checking status } & \neq \text{ no checking account} \\
\text{and checking status } & < 200 \\
\text{and employment } & \neq \text{ unemployed} \\
\text{and purpose } & = \text{ furniture/equipment}, \\
\text{then customer } & = \text{ bad}
\end{align*}
\]
**Decision rule:**

**Condition part** of a decision rule is represented by a complex:

\[
\Phi = \phi_1^* \land \phi_2^* \land \ldots \land \phi_t^*,
\]

where \( \phi^* \) is a selector and \( t \) is a number of selectors in the complex (i.e., **length** of the rule).

**Selector** \( \phi^* \) is defined as \( x_j^* \land v_j \), where \( v_j \) is a value or a subset of values from the domain of \( j \)-th attribute, \( \land \) is specified as \( =, \neq, \in, \geq \) or \( \leq \), depending on the type of \( j \)-th attribute.

Objects covered by complex \( \Phi \) are denoted by \( cov(\Phi) \) and referred to as **cover** of a complex \( \Phi \).

**Decision part** of a rule indicates one of the decision classes and is denoted by \( dec = d \), where \( d = 1 \) or \( d = -1 \) in the simplest case.
Decision rule denoted by $r(x, c)$, where $c = (\Phi, dec)$, is defined as:

$$
r(x, c) = \begin{cases} 
d & x \in cov(\Phi), \\
0 & x \notin cov(\Phi).
\end{cases}
$$

In the simplest form, the loss of a single decision rule (0-1-$\ell$ loss) takes the following form:

$$
L(y, r(x, c)) = \begin{cases} 
0 & y \cdot r(x, c) > 0, \\
1 & y \cdot r(x, c) < 0, \\
\ell & r(x, c) = 0,
\end{cases}
$$

where $\ell \in \langle 0, 1 \rangle$ is a penalty for specificity of the rule: the lower the value of $\ell$, the smaller the number of objects covered by the rule from the opposite class.
Ensembles of decision rules

**input**: set of training examples \(\{y_i, x_i\}_1^N\),

\(M\) – number of decision rules.

**output**: ensemble of decision rules \(\{r_m(x)\}_1^M\).

\[
F_0(x) := \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha); \text{ or } F_0(x) := 0;
F_0(x) := \nu \cdot F_0(x);
\]

**for** \(m = 1 \text{ to } M \) **do**

\[
c_m := \arg\min_{c} \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c));
\]

\[
r_m(x) = r(x, c_m);
F_m(x) = F_{m-1}(x) + \nu \cdot r_m(x);
\]

**end**

**ensemble** = \(\{r_m(x)\}_1^M\);

Forward Stagewise Additive Modeling is a general framework suited to simulate ensemble approaches: bagging, random forest, boosting.
Ensembles of decision rules

**input** : set of training examples \( \{y_i, x_i\}_{i=1}^N \),
\( M \) – number of decision rules.

**output** : ensemble of decision rules \( \{r_m(x)\}_{i=1}^M \).

\[
F_0(x) := \arg\min_\alpha \sum_{i=1}^N L(y_i, \alpha); \text{ or } F_0(x) := 0;
F_0(x) := \nu \cdot F_0(x);
\]

\text{for} \ m = 1 \ \text{to} \ M \ \text{do}
\[
c_m := \arg\min_c \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c));
\]

\[
r_m(x) = r(x, c_m);
F_m(x) = F_{m-1}(x) + \nu \cdot r_m(x);
\]
\text{end}

ensemble = \( \{r_m(x)\}_{i=1}^M \);


Ensembles of decision rules

**input**: set of training examples \( \{y_i, x_i\}_{1}^{N} \),

\( M \) – number of decision rules.

**output**: ensemble of decision rules \( \{r_m(x)\}_{1}^{M} \).

\[ F_0(x) := \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha); \text{ or } F_0(x) := 0; \]

\[ F_0(x) := \nu \cdot F_0(x); \]

**for** \( m = 1 \) **to** \( M \) **do**

\[ c_m := \arg\min_{c} \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c)); \]

\[ r_m(x) = r(x, c_m); \]

\[ F_m(x) = F_{m-1}(x) + \nu \cdot r_m(x); \]

**end**

\[ \text{ensemble} = \{r_m(x)\}_{1}^{M}; \]

\( M \) is a number of rules to be generated.
Ensembles of decision rules

**input**: set of training examples \( \{y_i, x_i\}_{1}^{N} \),

\( M \) – number of decision rules.

**output**: ensemble of decision rules \( \{r_m(x)\}_{1}^{M} \).

\[
F_0(x) := \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha); \quad \text{or} \quad F_0(x) := 0;
\]

\[
F_0(x) := \nu \cdot F_0(x);
\]

**for** \( m = 1 \) **to** \( M \) **do**

\[
c_m := \arg\min_c \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c));
\]

\[
r_m(x) = r(x, c_m);
\]

\[
F_m(x) = F_{m-1}(x) + \nu \cdot r_m(x);
\]

**end**

\( ensemble = \{r_m(x)\}_{1}^{M} \);

\( L(y_i, F(x)) \) is a loss function.
Ensembles of decision rules

**input**: set of training examples \( \{y_i, x_i\}_{1}^{N} \),

\( M \) – number of decision rules.

**output**: ensemble of decision rules \( \{r_m(x)\}_{1}^{M} \).

\[
F_0(x) := \arg \min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha); \text{ or } F_0(x) := 0;
\]

\[
F_0(x) := \nu \cdot F_0(x);
\]

**for** \( m = 1 \) **to** \( M \) **do**

\[
c_m := \arg \min_{c} \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c));
\]

\[
r_m(x) = r(x, c_m);
\]

\[
F_m(x) = F_{m-1}(x) + \nu \cdot r_m(x);
\]

**end**

ensemble = \( \{r_m(x)\}_{1}^{M} \);

\( r_m(x, c) \) is a decision rule characterized by a set of parameters \( c \).
Ensembles of decision rules

**input**: set of training examples \( \{y_i, x_i\}_{i=1}^N \),
\( M \) – number of decision rules.

**output**: ensemble of decision rules \( \{r_m(x)\}_{m=1}^M \).

\[
F_0(x) := \arg \min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha); \text{ or } F_0(x) := 0;
\]
\[
F_0(x) := \nu \cdot F_0(x);
\]

**for** \( m = 1 \) **to** \( M \) **do**

\[
c_m := \arg \min_{c} \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c));
\]
\[
r_m(x) = r(x, c_m);
\]
\[
F_m(x) = F_{m-1}(x) + \nu \cdot r_m(x);
\]

**end**

\( ensemble = \{r_m(x)\}_{m=1}^M; \)

\( S_m(\eta) \) represents a different subsample of size \( \eta \ll N \) randomly drawn with or without replacement from the original training data.
Ensembles of decision rules

**input**: set of training examples \( \{y_i, x_i\}_{i=1}^{N} \),
\( M \) – number of decision rules.

**output**: ensemble of decision rules \( \{r_m(x)\}_{m=1}^{M} \).

\[
F_0(x) := \arg\min_\alpha \sum_{i=1}^{N} L(y_i, \alpha); \text{ or } F_0(x) := 0;
\]

\[
F_0(x) := \nu \cdot F_0(x);
\]

for \( m = 1 \) to \( M \) do

\[
c_m := \arg\min_c \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c));
\]

\[
r_m(x) = r(x, c_m);
\]

\[
F_m(x) = F_{m-1}(x) + \nu \cdot r_m(x);
\]

end

ensemble = \( \{r_m(x)\}_{m=1}^{M} \);

\( \nu \in (0, 1) \) is a shrinkage parameter that determines the degree to which previously generated decision rules \( r_k(x), k = 1, \ldots, m \), affect the successive one in the sequence, i.e., \( r_{m+1}(x) \).
Ensembles of decision rules

**input** : set of training examples \( \{y_i, x_i\}_{1}^{N} \),
\( M \) – number of decision rules.

**output** : ensemble of decision rules \( \{r_m(x)\}_{1}^{M} \).

\[
F_0(x) := \arg \min_\alpha \sum_{i=1}^{N} L(y_i, \alpha); \text{ or } F_0(x) := 0; \\
F_0(x) := \nu \cdot F_0(x); \\
\text{for } m = 1 \text{ to } M \text{ do} \\
\quad c_m := \arg \min_c \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c)); \\
\quad r_m(x) = r(x, c_m); \\
\quad F_m(x) = F_{m-1}(x) + \nu \cdot r_m(x); \\
\text{end} \\
ensemble = \{r_m(x)\}_{1}^{M}.
\]

\[
F_0(x) := \arg \min_\alpha \sum_{i=1}^{N} L(y_i, \alpha); \text{ or } F_0(x) := 0; \text{ defines a default rule or there is no default rule.}
\]
Ensembles of decision rules

In each consecutive iteration $m$ we augment the function $F_{m-1}(x)$ by one additional rule $r_m(x)$ weighed by shrinkage parameter $\nu$.

This gives a linear combination of rules $F_m(x)$.

The additional rule $r_m(x) = r(x, c)$ is chosen to minimize

$$\sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c)).$$
Prediction procedure is performed according to:

\[ F(x) = a_0 + \sum_{m=1}^{M} a_m r_m(x). \]

It is a linear classifier in a very high dimensional space of derived decision rules that are highly nonlinear functions of the original predictor variables \( x \).

Parameters \( \{a_m\}_{0}^{M} \) can be obtained in many ways:

- set to fixed values, for example \( a_0=0 \) and \( \{a_m = 1/M\}_{1}^{M} \),
- computed by some optimization techniques,
- fitted in cross-validation experiments,
- estimated in a process of constructing the ensemble (like in AdaBoost).
Prediction procedure is performed according to:

\[ F(x) = a_0 + \sum_{m=1}^{M} a_m r_m(x). \]

In the case of decision rules, parameters \( \{a_m\}_1^M \) can be identified with the output of a single rule \( r_m(x) \), i.e.:

\[ r_m(x) = \begin{cases} a_m \cdot d_m & x \in \text{cov}(\Phi), \\ 0 & x \notin \text{cov}(\Phi). \end{cases} \]

where \( a_m \in \mathbb{R}^+ \), \( d_m \in \{-1, 1\} \) usually, and let

\[ \alpha_m = a_m \cdot d_m, \quad \alpha_m \in \mathbb{R}. \]
**Ensembles of decision rules**

There are two crucial elements of the algorithm to be chosen:
- heuristic constructing a single decision rule,
- loss function.
Greedy heuristic constructing a single decision rule

Search for \( c \) such that:

\[
L_m = \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c))
\]

is minimal.

- At the beginning, there is an empty rule: the complex of the rule is empty (no selectors are specified) and decision is not determined,
- In the next step, a new selector is added to the complex and the decision of the rule is determined. The selector and the decision are chosen to give the minimal value of \( L_m \).
- The above step is repeated until \( L_m \) is minimized.
Greedy heuristic constructing a single decision rule

Search for $c$ such that:

$$L_m = \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(x_i) + r(x_i, c))$$

is minimal.

Minimal value of $L_m$ is a natural stop criterion in building a single rule.

Additionally, another stop criterion can be introduced, for example, length of the rule.
Ensembles of decision rules with different loss functions:

- 0-1-$\ell$ loss
- Sigmoid loss
- Exponential loss
- Squared loss (Regression Rules)
- Binomial log-likelihood loss
0-1-\ell loss

\[ L_{0-1-\ell}(y, F_m(x)) = \begin{cases} 
0 & y \cdot F_m(x) > 0, \\
1 & y \cdot F_m(x) < 0, \\
\ell & y \cdot F_m(x) = 0.
\end{cases} \]

Population minimizer

\[ F^*(x) = \arg \min_{F(x)} \mathbb{E}_{y|x} L_{0-1-\ell}(y, F(x)) = \begin{cases} 
d & \max_{d=\{-1,1\}} P(y = d|x) > 1 - \ell, \\
0 & \max_{d=\{-1,1\}} P(y = d|x) \leq 1 - \ell.
\end{cases} \]
0-1-\ell loss

\[
L_{0-1-\ell}(y, F_m(x)) = \begin{cases} 
0 & y \cdot F_m(x) > 0, \\
1 & y \cdot F_m(x) < 0, \\
\ell & y \cdot F_m(x) = 0.
\end{cases}
\]

Prediction procedure is performed according to:

\[
F(x) = \text{sign}(a_0 + \sum_{m=1}^{M} a_m r_m(x)).
\]
0-1-$\ell$ loss

\[ L_{0-1-\ell}(y, F_m(x)) = \begin{cases} 
0 & y \cdot F_m(x) > 0, \\
1 & y \cdot F_m(x) < 0, \\
\ell & y \cdot F_m(x) = 0.
\end{cases} \]

Classification rules:

\[ r(x, c) = \begin{cases} 
d & x \in \text{cov}(\Phi), \\
0 & x \notin \text{cov}(\Phi),
\end{cases} \]

where \( d \in \{-1, 1\} \).
0-1-\ell loss

\[ L_{0-1-\ell}(y, F_m(x)) = \begin{cases} 
0 & y \cdot F_m(x) > 0, \\
1 & y \cdot F_m(x) < 0, \\
\ell & y \cdot F_m(x) = 0.
\end{cases} \]

In each iteration \( m \), the following expression has to be minimized:

\[
\sum_{y_i \cdot (F_{m-1}(x_i) + r(x_i, c)) < 0} 1 + \ell \cdot \sum_{F_{m-1}(x_i) + r(x_i, c) = 0} 1
\]
Sequential covering (separate-and-conquer)

*Learn a rule that covers a part of the given training examples, remove the covered examples from the training set (the separate part) and recursively learn another rule that covers some of the remaining examples (the conquer part) until no examples remain.*


0-1-$\ell$ loss and sequential covering

0-1-$\ell$ loss gives a **procedure similar to sequential covering**, because loss of training examples covered by one rule is already equal 0, and there is no need to look for another rule covering them, i.e. it *corresponds* to removing them from the set of training examples.
Ensembles of decision rules – simple sequential covering

**input**: set of training examples \( \{y_i, x_i\}^N_1 \),

\( M \) – number of decision rules.

**output**: ensemble of decision rules \( \{r_m(x)\}^M_1 \).

\[ F_0(x) := 0; \]

**for** \( m = 1 \) **to** \( M = N \) **do**

\[ c_m := \arg \min_c \sum_{i \in S_m(\eta)} L_{0-1-\ell}(y_i, F_{m-1}(x_i) + r(x_i, c)); \]

\[ r_m(x) = r(x, c_m); \]

\[ F_m(x) = F_{m-1}(x) + \nu \cdot r_m(x); \]

**end**

**ensemble** = \( \{r_m(x)\}^M_1 \);

- \( L(y_i, F(x)) = L_{0-1-\ell}(y_i, F(x)), \ell < \frac{1}{N}, \)
- Decision of the rules is set to \( d = 1 \) (positive examples),
- \( F_0(x) := 0, M = N, \nu = 1, \eta = N, \)
- \( F(x) = \text{sign}(-0.1 + \sum_{m=1}^M r_m(x)). \)
input : set of training examples $X = \{y_i, x_i\}_1^N$
  set of positive examples $\hat{X} \subset X$ being $P(Cl)$ or $\overline{P}(Cl)$.
output: set of decision rules $\{r_m(x)\}_1^M$.

$F_0(x) = 0; m = 0$;
while $\sum_{i \in \hat{X}} L(y_i, F_m(x_i, c)) \neq 0$ do
  $m = m + 1$;
  $c_m = \arg \min_c \sum_{i \in \hat{X}} L_{0-1-\ell}(y_i, F_{m-1}(x_i) + r(x_i, c))$;
  $r_m(x) = r(x, c_m)$;
  $F_m(x) = F_{m-1}(x) + r_m(x)$;
end

$M = m$; rules = $\{r_m(x)\}_1^M$;

- $L(y_i, F(x)) = L_{0-1-\ell}(y_i, F(x))$, $\ell < \frac{1}{N}$,
- Decision rule covers only positive examples,
- $F(x) = \text{sign}(\sum_{m=1}^M \text{mat}(r_m(x)) \cdot \text{sup}(r_m(x)) \cdot \text{spe}(r_m(x)) \cdot r_m(x))$,
  where $\text{mat}$ is matching, $\text{sup}$ is support and $\text{spe}$ is specificity of the rule $r_m(x)$. 
Ensemble of decision rules – sequential covering II

**input**: set of training examples \( \{y_i, x_i\}_i^N \),

\( M \) – number of decision rules.

**output**: ensemble of decision rules \( \{r_m(x)\}_1^M \).

\[
F_0(x) := 0;
\]

**for** \( m = 1 \) **to** \( M = N \) **do**

\[
\begin{align*}
& c := \arg\min_c \sum_{i \in S_m(\eta)} L_{0-\ell}(y_i, F_{m-1}(x_i) + r(x_i, c)); \\
& r_m(x) = r(x, c); \\
& F_m(x) = F_{m-1}(x) + 2^{M-m} \cdot r_m(x);
\end{align*}
\]

**end**

**ensemble** = \( \{r_m(x)\}_1^M \);

- \( L(y_i, F(x)) = L_{0-\ell}(y_i, F(x)), \ell < \frac{1}{N}, d \in \{-1, 1\} \),
- \( F_0(x) := 0, M = N, \nu = 2^{M-m} \),
- \( F(x) = \text{sign}(a_0 + \sum_{m=1}^M a_m r_m(x)) \),

where \( a_0 \) indicates the majority class and \( a_m = 2^{M-m+1} \).
Sigmoid loss

\[ L_{\text{sigm}}(y, F_m(x)) = \frac{1}{1 + \exp(\beta \cdot y \cdot F_m(x))} \]

Population minimizer

\[ F^*(x) = \arg \min_{F(x)} \mathbb{E}_{y|x} L_{\text{sigm}}(y, F(x)) = \begin{cases} 
+ \inf & P(y = 1|x) > \frac{1}{2}, \\
- \inf & P(y = -1|x) < \frac{1}{2}, \\
default & \text{otherwise.}
\end{cases} \]
Sigmoid loss

\[ L_{\text{sigm}}(y, F_m(x)) = \frac{1}{1 + \exp(\beta \cdot y \cdot F_m(x))} \]

Prediction procedure is performed according to:

\[ F(x) = \text{sign}(a_0 + \sum_{m=1}^{M} a_m r_m(x)). \]
Sigmoid loss

\[ L_{\text{sigm}}(y, F_m(x)) = \frac{1}{1 + \exp(\beta \cdot y \cdot F_m(x))} \]

Classification rules:

\[ r(x, c) = \begin{cases} 
  d & x \in \text{cov}(\Phi), \\
  0 & x \not\in \text{cov}(\Phi), 
\end{cases} \]

where \( d \in \{-1, 1\} \).
Sigmoid loss

\[ L_{\text{sigm}}(y, F_m(x)) = \frac{1}{1 + \exp(\beta \cdot y \cdot F_m(x))} \]

In each iteration \( m \), the following expression has to be minimized:

\[ \sum_{y_i \cdot r(x_i, c) > 0} \frac{1}{1 + \exp(\beta \cdot y_i \cdot (F_{m-1}(x_i) + d))} + \]

\[ \sum_{y_i \cdot r(x_i, c) < 0} \frac{1}{1 + \exp(\beta \cdot y_i \cdot (F_{m-1}(x_i) + d))} + \]

\[ \sum_{r(x_i, c) = 0} \frac{1}{1 + \exp(\beta \cdot y_i \cdot F_{m-1}(x_i))} \]
Sigmoid loss

\[ L_{\text{sigm}}(y, F_m(x)) = \frac{1}{1 + \exp(\beta \cdot y \cdot F_m(x))} \]

- Sigmoid loss is a relaxed form of the 0-1-\( \ell \) loss function,
- Hard statistical interpretation of sigmoid loss,
- Our first (and promising) experiments were performed using sigmoid loss.
Exponential loss

\[ L_{\text{exp}}(y, F_m(x)) = \exp(-\beta \cdot y \cdot F_m(x)) \]

Population minimizer

\[ F^*(x) = \arg\min_{F(x)} E_{y|x} L_{\text{exp}}(y, F(x)) = \frac{1}{2} \log \frac{P(y = 1|x)}{P(y = -1|x)} \]
Exponential loss

\[ L_{\text{exp}}(y, F_m(x)) = \exp(-\beta \cdot y \cdot F_m(x)) \]

Prediction procedure is performed according to:

\[ F(x) = a_0 + \sum_{m=1}^{M} a_m r_m(x), \]

and

\[ P(y = 1|x) = \frac{1}{1 + \exp(-2F(x))}. \]
Exponential loss

\[ L_{\text{exp}}(y, F_m(x)) = \exp(-\beta \cdot y \cdot F_m(x)) \]

Classification rules:

\[ r(x, c) = \begin{cases} 
\alpha & x \in \text{cov}(\Phi), \\
0 & x \notin \text{cov}(\Phi), 
\end{cases} \]

where \( \alpha \in \{-d, d\} \), \( d = \text{const} \) or \( \alpha \in \mathbb{R} \).
Exponential loss

\[ L_{\text{exp}}(y, F_m(x)) = \exp(-\beta \cdot y \cdot F_m(x)) \]

For \( \alpha \in \{-d, d\} \), one has to minimized in each iteration \( m \):

\[
\begin{align*}
\sum_{y_i \cdot r(x_i, c) > 0} \exp(-\beta \cdot (y_i \cdot F_{m-1}(x_1) + d)) &+ \\
\sum_{y \cdot r(x_i, c) < 0} \exp(-\beta \cdot (y_i \cdot F_{m-1}(x_1) - d)) &+ \\
\sum_{r(x_i, c) = 0} \exp(-\beta \cdot y_i \cdot F_{m-1}(x_1)) &,
\end{align*}
\]

where \( \exp(-\beta \cdot y_i \cdot F_{m-1}(x_i)) \) can be treated as \( w_i^{(m)} \) (i.e., weight of \( i \)-th training example in the \( m \)-th iteration).
Exponential loss

\[ L_{\text{exp}}(y, F_m(x)) = \exp(-\beta \cdot y \cdot F_m(x)) \]

The above leads to:

\[
\sum_{y_i \cdot r(x_i, c) < 0} w_i^{(m)} + \ell \cdot \sum_{r(x_i, c) = 0} w_i^{(m)}
\]

where

\[
el = \frac{1 - e^{-d}}{e^d - e^{-d}} \quad d = \log \frac{1 - \ell}{\ell}.
\]

The above can be treated as weighed 0-1-\(\ell\) loss.
Exponential loss

\[ L_{\text{exp}}(y, F_m(x)) = \exp(-\beta \cdot y \cdot F_m(x)) \]

For \( \alpha \in \mathbb{R} \) the heuristic constructing a single rule has to minimize:

\[
2 \sqrt{\sum_{y_i \cdot r(x_i, c) > 0} w_i^{(m)} \cdot \sum_{y_i \cdot r(x_i, c) < 0} w_i^{(m)} + \sum_{r(x_i, c) = 0} w_i^{(m)}}
\]

and the final output of the rule is:

\[
\alpha = \frac{1}{2} \log \frac{\sum_{y_i \cdot r(x_i, c) > 0} w_i^{(m)}}{\sum_{y_i \cdot r(x_i, c) < 0} w_i^{(m)}}
\]
Exponential loss

One can reformulate

\[ \alpha = \frac{1}{2} \log \frac{\sum_{y_i \cdot r(x_i, c) > 0} w_i^{(m)}}{\sum_{y_i \cdot r(x_i, c) < 0} w_i^{(m)}} \]

to the following form:

\[ \alpha = \frac{1}{2} \log \frac{P(y = 1 | r(x, c))}{P(y = -1 | r(x, c))} \]

that is related also to **confirmation measure**:

\[ l(E|H) = \log \frac{P(E|H)}{P(E|\neg H)} = \log \frac{P(H|E) \cdot P(\neg H)}{P(\neg H|E) \cdot P(H)} \]

where \( E \) is evidence, \( H \) is hypothesis, \( P(H) \) and \( P(\neg H) \) are constant in classification tasks.
Ensembles of Decision Rules and AdaBoost

- $F_m(x)$ is updated according to:

$$F_m(x) = F_{m-1}(x) + \alpha \cdot r_m(x)$$

which causes the weights for the next iteration to be:

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{-\alpha y_i r_m(x)}$$

that gives, in fact, the reweighing schema well-known from AdaBoost.

- $\{2\alpha\}_1^M$ parameters are used in AdaBoost to weighing weak learners.

- AdaBoost uses a linear classifier combining weak learners.

- Decision rule is a specific base classifier that exceeds random classifier: the space covered by the rule has small error, uncovered space could be treated as a space of coin throwing.
SLIPPER (by Cohen and Singer)

- Uses AdaBoost schema (i.e., uses exponential loss and minimizes with respect to $\alpha$ and $c$),
- Decision rules are generated only for positive examples,
- There is a default rule,
- Heuristic constructing a single rule applies post-pruning.

Output of decision rules is \( r_m(x) \in \{0, 1\} \),

Uses specific reweighing schema:

\[
  w_i^{(m)} = 1 + \left( \sum_{j=1}^{m-1} I(y_i \neq r_j(x_i, c)) \right)^3
\]

Uses DNF-formulas for single rules,

Decision rules are separately induced for each class (the same number of rules for each class),

Heuristic constructing a single rule minimizes (for $r(x_i, c) \in \{0, 1\}$):

$$\sum_{y_i \cdot r(x_i, c) = -1} w_i^{(m)} + k \cdot \sum_{r(x_i, c) = 0 \land y_i = 1} w_i^{(m)},$$

where $k = 1, 2, 4, \ldots$ is changing during the construction of a rule, $\sum_{y_i \cdot r(x_i, c) = -1} w_i^{(m)}$ are weighed false positives and $\sum_{r(x_i, c) = 0 \land y_i = 1} w_i^{(m)}$ are weighed false negatives.

Ensembles of Decision Rules and Lightweight Rule Induction

For $k = \text{const}$ minimization of

\[
\sum_{y_i \cdot r(x_i, c) = -1} w_i^{(m)} + k \cdot \sum_{r(x_i, c) = 0 \land y_i = 1} w_i^{(m)},
\]

is equivalent to minimization of

\[
\sum_{y_i \cdot r(x_i, c) = -1} w_i^{(m)} + \ell \sum_{r(x_i, c) = 0} w_i^{(m)}
\]

where

\[
\ell = \frac{k}{k + 1}
\]
Ensembles of Decision Rules and Lightweight Rule Induction

Taking several assumptions like
- $k = \text{const}$,
- exponential example reweighing,
- 1-DNF formulas for single decision rules,

then we obtain an algorithm similar to ensembles of decision rules
- with exponential loss
- and $d = \text{const}$.
<table>
<thead>
<tr>
<th>Squared loss (regression rules)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_{\text{squared}}(y, F_m(x)) = (y - F_m(x))^2 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Population minimizer</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F^*(x) = \arg \min_{F(x)} E_{y</td>
</tr>
</tbody>
</table>
Squared loss (regression rules)

\[ L_{squared}(y, F_m(x)) = (y - F_m(x))^2 \]

Prediction procedure is performed according to:

\[ F(x) = a_0 + \sum_{m=1}^{M} a_m r_m(x), \]
Squared loss (regression rules)

\[ L_{squared}(y, F_m(x)) = (y - F_m(x))^2 \]

Regression rules:

\[ r(x, c) = \begin{cases} 
\alpha & x \in \text{cov}(\Phi), \\
0 & x \notin \text{cov}(\Phi), 
\end{cases} \]

where \( \alpha \in \mathbb{R} \).
Squared loss (regression rules)

\[ L_{squared}(y, F_m(x)) = (y - F_m(x))^2 \]

In each iteration \( m \), the following expression has to be minimized:

\[
\sum_{r(x_i, c)=0} (y_i - F_{m-1}(x_i))^2 + \sum_{r(x_i, c)\neq 0} (y_i - F_{m-1}(x_i) - \alpha)^2
\]

where

\[
\alpha = \frac{\sum_{r(x_i, c)\neq 0} (y_i - F_{m-1}(x_i))}{\sum_{r(x_i, c)\neq 0} 1}.
\]
Approximation of $f(x) = 3x^5 - x^4 - x$, model built on 41 observations.
Regression Rules

Approximation of \( f(x) = 3x^5 - x^4 - x \), model built on 41 observations.
Regression Rules

Approximation of $f(x) = 3x^5 - x^4 - x$, model built on 41 observations.
Regression Rules

Approximation of $f(x) = 3x^5 - x^4 - x$, model built on 41 observations.
Regression Rules

Approximation of $f(x) = 3x^5 - x^4 - x$, model built on 41 observations.
Regression Rules

Approximation of \( f(x) = 3x^5 - x^4 - x \), model built on 41 observations.
Approximation of $f(x) = 3x^5 - x^4 - x$, model built on 41 observations.
Regression Rules

Approximation of $f(x) = 3x^5 - x^4 - x$, model built on 41 observations.
Binomial log-likelihood

\[ L_{\log}(y, F_m(x)) = \log(1 + \exp(-\beta \cdot y \cdot F_m(x))) \]

Population minimizer

\[ F^*(x) = \arg \min_{F(x)} E_{y|x} L_{\log}(y, F(x)) = \frac{1}{2} \log \frac{P(y = 1|x)}{P(y = -1|x)} \]
Binomial log-likelihood

\[ L_{log}(y, F_m(x)) = \log(1 + \exp(-\beta \cdot y \cdot F_m(x))) \]

Prediction procedure is performed according to:

\[ F(x) = a_0 + \sum_{m=1}^{M} a_m r_m(x), \]

and

\[ P(y = 1|x) = \frac{1}{1 + \exp(-2F(x))}. \]
Binomial log-likelihood

\[ L_{log}(y, F_m(x)) = \log(1 + \exp(-\beta \cdot y \cdot F_m(x))) \]

Classification rules:

\[ r(x, c) = \begin{cases} 
\alpha & x \in \text{cov}(\Phi), \\
0 & x \notin \text{cov}(\Phi),
\end{cases} \]

where \( \alpha \in \{-d, d\} \), \( d = \text{const} \) or \( \alpha \in \mathbb{R} \).
Binomial log-likelihood

$$L_{log}(y, F_m(x)) = \log(1 + \exp(-\beta \cdot y \cdot F_m(x)))$$

For $\alpha \in \{-d, d\}$, one has to minimized in each iteration $m$:

$$\sum_{r(x_i,c)=0} \log(1 + \exp(-\beta \cdot y_i \cdot F_{m-1}(x_1))) +$$
$$\sum_{y \cdot r(x_i,c) > 0} \log(1 + \exp(-\beta \cdot (y_i \cdot F_{m-1}(x_1) + d))) +$$
$$\sum_{y \cdot r(x_i,c) < 0} \log(1 + \exp(-\beta \cdot (y_i \cdot F_{m-1}(x_1) - d))),$$

where $\exp(-\beta \cdot y_i \cdot F_{m-1}(x_i))$ can be treated as $w_i^{(m)}$ (i.e., weight of $i$-th training example in the $m$-th iteration).
Binomial log-likelihood

\[ L_{log}(y, F_m(x)) = \log(1 + \exp(-\beta \cdot y \cdot F_m(x))) \]

For \( \alpha \in \mathbb{R} \), there is no straight-forward optimization, the solution is to use gradient boosting machines.

Binomial log-likelihood

\[ L_{log}(y, F_m(x)) = \log(1 + \exp(-\beta \cdot y \cdot F_m(x))) \]

In each iteration \( m \) build a **regression rule** minimizing:

\[
\sum_{r(x_i,c)=0} (\bar{y}_i)^2 + \sum_{r(x_i,c) \neq 0} (\bar{y}_i - \bar{\alpha})^2
\]

where \( \bar{y}_i \) is a pseudo-response defined as:

\[
\bar{y}_i = -\left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} = \frac{2y_i}{1 + \exp(2y_iF_{m-1}(x))}.
\]

Final output of the rule is \( \alpha = \sum_{r(x,c) \neq 0} \bar{y}_i / \sum_{r(x_i,c) \neq 0} |\bar{y}_i| (2 - |\bar{y}_i|) \).
Ensembles of decision rules and Gradient Boosting Trees

In each iteration \( m \) of gradient boosting trees one looks for

\[
\left\{ \gamma_{jm}, c_{jm} \right\}_1^J = \arg \min_{\left\{ \gamma_j, c_j \right\}_1^J} \sum_{i=1}^N L(y_i, F_{m-1}(x_i)) + \sum_{j=1}^J \gamma_j \cdot r(x_i, c_j)
\]

where \( \left\{ r(x_i, c_j) \right\}_1^J \) represents all paths from the root to leaves in the tree.

In the ensembles of decision rules, each rule \( r_m(x, c) \) and the output \( \alpha \) is optimized separately:

\[
(\alpha_m, c_m) = \arg \min_{\alpha, c} \sum_{i=1}^N L(y_i, F_{m-1}(x_i)) + \alpha \cdot r(x_i, c)
\]
Conclusions

Several properties of ensembles of decision rules in comparison to decision tree ensembles:

- natural stop criterion in building a single rule,
- decision rule $r_m(x)$ and its output $\alpha$ are optimized individually taking into account all previously generated rules $\{r_m(x)\}^{m-1}_1$.
- parameters $\{a_m\}^M_1$ corresponds to outputs of single rules $\{\alpha_m\}^M_1$ – one parameter to be optimize.

One can treat the ensembles of decision rules as:

- extension of sequential covering
- or ensembles of specific base classifiers.
1 Problem Statement

2 Ensembles of Decision Rules

3 Experimental Results
Software was written and experiments were performed using:

- Weka package: http://www.cs.waikato.ac.nz/ml/weka/
<table>
<thead>
<tr>
<th>Classifier</th>
<th>Abbrev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NaiveBayes</td>
<td>NB</td>
</tr>
<tr>
<td>Logistic</td>
<td>Log</td>
</tr>
<tr>
<td>RBFNetwork</td>
<td>RBF</td>
</tr>
<tr>
<td>SMO</td>
<td>SMO</td>
</tr>
<tr>
<td>IBk</td>
<td>IBL</td>
</tr>
<tr>
<td>AdaBoostM1 -P 100 -S 1 -I 50 -W DecisionStump</td>
<td>AB DS</td>
</tr>
<tr>
<td>AdaBoostM1 -P 100 -S 1 -I 50 -W REPTree-M 2 -V 0.0010 -N 3 -S 1 -L -1</td>
<td>AB RT</td>
</tr>
<tr>
<td>AdaBoostM1 -P 100 -S 1 -I 50 -W J48 -C 0.25 -M 2</td>
<td>AB J48</td>
</tr>
<tr>
<td>AdaBoostM1 -P 100 -S 1 -I 50 -W PART -M 2 -C 0.25 -Q 1</td>
<td>AB PT</td>
</tr>
<tr>
<td>Bagging -P 100 -S 1 -I 50 -W REPTree-M 2 -V 0.0010 -N 3 -S 1 -L -1</td>
<td>B RT</td>
</tr>
<tr>
<td>Bagging -P 100 -S 1 -I 50 -W J48 -C 0.25 -M 2</td>
<td>B J48</td>
</tr>
<tr>
<td>Bagging -P 100 -S 1 -I 50 -W PART -M 2 -C 0.25 -Q 1</td>
<td>B PT</td>
</tr>
<tr>
<td>LogitBoost -P 100 -F 0 -R 1 -L -1.8e308 -H 1.0 -S 1 -I 50 -W DecisionStump</td>
<td>LB DS</td>
</tr>
<tr>
<td>LogitBoost -P 100 -F 0 -R 1 -L -1.8e308 -H 1.0 -S 1 -I 50 -W REPTree-M 2 -V 0.0010 -N 3 -S 1 -L -1</td>
<td>LB RT</td>
</tr>
<tr>
<td>J48 -C 0.25 -M 2</td>
<td>J48</td>
</tr>
<tr>
<td>RandomForest -I 50 -K 0 -S 1</td>
<td>RF</td>
</tr>
<tr>
<td>PART -M 2 -C 0.25 -Q 1</td>
<td>PT</td>
</tr>
<tr>
<td>Ensemble of Decision Rules, $L_{sign}(y, F_m(x)), M = 50$, bootstrap sample $\eta = N, \nu = 0.5, a_0 = F_0(x), {a_m}_M^1 = 1$</td>
<td>EDR</td>
</tr>
</tbody>
</table>
### Data sets included in the experiment

<table>
<thead>
<tr>
<th>Data set</th>
<th>Attributes</th>
<th>Class -1</th>
<th>Class 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>German Credit (credit-g)</td>
<td>21</td>
<td>300</td>
<td>700</td>
</tr>
<tr>
<td>Pima Indians Diabetes (diabetes)</td>
<td>9</td>
<td>268</td>
<td>500</td>
</tr>
<tr>
<td>Heart Statlog (heart-statlog)</td>
<td>14</td>
<td>120</td>
<td>150</td>
</tr>
<tr>
<td>J. Hopkins University Ionosphere (ionosphere)</td>
<td>35</td>
<td>126</td>
<td>225</td>
</tr>
<tr>
<td>King+Rook vs. King+Pawn on a7 (kr-vs-kp)</td>
<td>37</td>
<td>1527</td>
<td>1669</td>
</tr>
<tr>
<td>Sonar, Mines vs. Rocks (sonar)</td>
<td>61</td>
<td>97</td>
<td>111</td>
</tr>
</tbody>
</table>
Credit-g – leave-one-out estimate (accuracy [%])

![Graph showing accuracy for different algorithms.]
Diabetes – leave-one-out estimate (accuracy [%])

- PT
- AB J48
- AB RT
- RBF
- J48
- IBL
- LB DS
- B PT
- RF
- B RT
- LB RT
- EDR
- NB
- AB PT
- B J48
- AB DS
- SMO
- Log

Accuracy [%]: 66 68 70 72 74 76 78
Heart-statlog – leave-one-out estimate (accuracy [%])

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>J48</td>
<td>74</td>
</tr>
<tr>
<td>AB RT</td>
<td>76</td>
</tr>
<tr>
<td>PT</td>
<td>78</td>
</tr>
<tr>
<td>AB J48</td>
<td>80</td>
</tr>
<tr>
<td>LB RT</td>
<td>82</td>
</tr>
<tr>
<td>IBL</td>
<td>84</td>
</tr>
<tr>
<td>B J48</td>
<td>84</td>
</tr>
<tr>
<td>LB DS</td>
<td>84</td>
</tr>
<tr>
<td>RBF</td>
<td>84</td>
</tr>
<tr>
<td>B PT</td>
<td>84</td>
</tr>
<tr>
<td>RF</td>
<td>84</td>
</tr>
<tr>
<td>EDR</td>
<td>76</td>
</tr>
<tr>
<td>AB DS</td>
<td>78</td>
</tr>
<tr>
<td>B RT</td>
<td>80</td>
</tr>
<tr>
<td>NB</td>
<td>82</td>
</tr>
<tr>
<td>Log</td>
<td>82</td>
</tr>
<tr>
<td>SMO</td>
<td>82</td>
</tr>
<tr>
<td>AB PT</td>
<td>84</td>
</tr>
</tbody>
</table>
kr-vs-kp – leave-one-out estimate (accuracy [%])

accuracy [%]
80 85 90 95 100

LB RT RBF NB AB DS EDR SMO LB DS IBL Log PT B RT AB RT B J48 RF B PT J48 AB J48 AB PT
<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>65</td>
</tr>
<tr>
<td>J48</td>
<td>70</td>
</tr>
<tr>
<td>RBF</td>
<td>75</td>
</tr>
<tr>
<td>LB RT</td>
<td>80</td>
</tr>
<tr>
<td>PT</td>
<td>85</td>
</tr>
<tr>
<td>Log</td>
<td>90</td>
</tr>
<tr>
<td>B RT</td>
<td></td>
</tr>
<tr>
<td>SMO</td>
<td></td>
</tr>
<tr>
<td>AB RT</td>
<td></td>
</tr>
<tr>
<td>B J48</td>
<td></td>
</tr>
<tr>
<td>EDR</td>
<td></td>
</tr>
<tr>
<td>IBL</td>
<td></td>
</tr>
<tr>
<td>LB DS</td>
<td></td>
</tr>
<tr>
<td>AB DS</td>
<td></td>
</tr>
<tr>
<td>B PT</td>
<td></td>
</tr>
<tr>
<td>AB J48</td>
<td></td>
</tr>
<tr>
<td>AB PT</td>
<td></td>
</tr>
<tr>
<td>RF</td>
<td></td>
</tr>
</tbody>
</table>
Future plans and conclusions
Future plans and conclusions

Promising first steps in the research on ensembles of decision rules, but still a lot to do:

- exhaustive experimental research,
- missing values,
- interpretation of rules,
- robust loss functions,
- multi-class problem,
- ordinal classification problem,
- ranking problem.

Ensembles of decision rules are flexible, powerful, fast and interpretable.
Future plans and conclusions

Promising first steps in the research on **ensembles of decision rules**, but still a lot to do:

- exhaustive experimental research,
- missing values,
- interpretation of rules,
- robust loss functions,
- multi-class problem,
- ordinal classification problem,
- ranking problem.

Ensembles of decision rules are **flexible, powerful, fast** and **interpretable**.

Yesterday, Today, ..., Tomorrow ...
Thank you :) ...
There is now time for questions and discussion ...