Trimming the complexity of Ranking by Pairwise Comparison

Samuel Hiard

University of Liège

11th February 2013

Outline

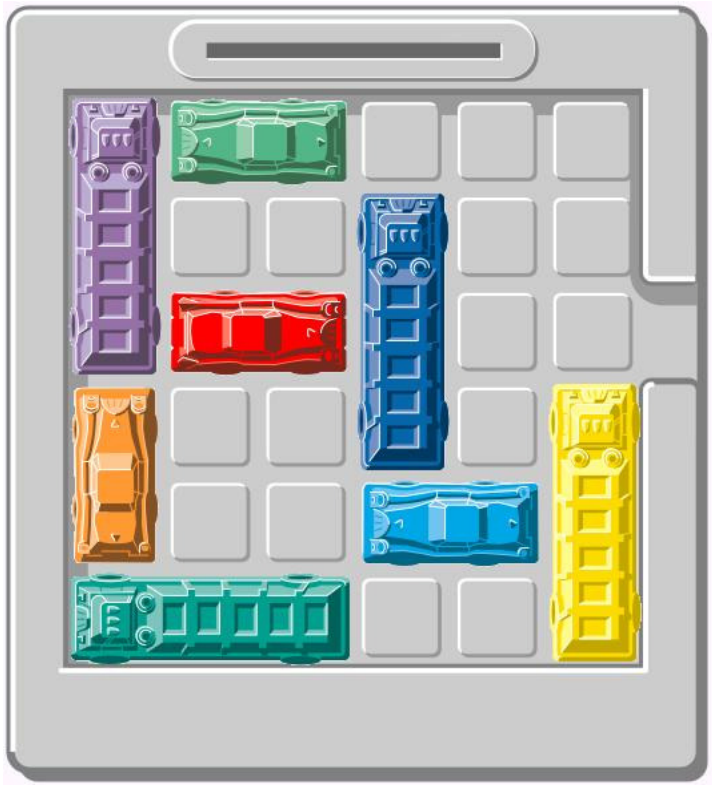
- 1st Part
  - Context and definitions
    - Artificial Intelligence
    - Supervised Learning
    - Label Ranking
  - Problem setting
  - Contribution
- 2nd Part
  - Methods and algorithms
  - Empirical validation
  - Conclusion

Artificial intelligence

- Hard to define because of the subjectivity of the word ``intelligence''.
- Literature:
  - AI produces an intelligent response to a given task (Encyclopedia of Artificial Intelligence)
  - AI simulates the behaviour of a human brain (Ibid)
  - ``*The building of computer programs which perform tasks which are, <u>for the moment</u>, performed in a more satisfactory way by humans because they require high-level mental processes such as: perception learning, memory organization and critical reasoning*'' (Marvin Lee Minsky, ~1956)
- AI should also be ``adaptive'': be able to change its behaviour if the objective or the environment changes

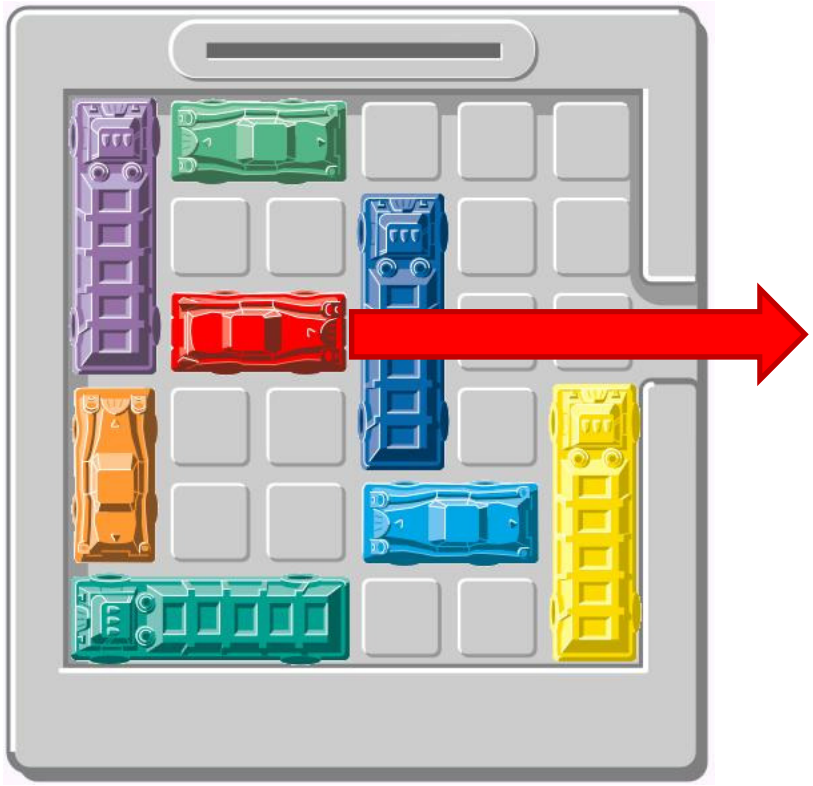
Applications of AI

Problem solving (e.g. Rush Hour)



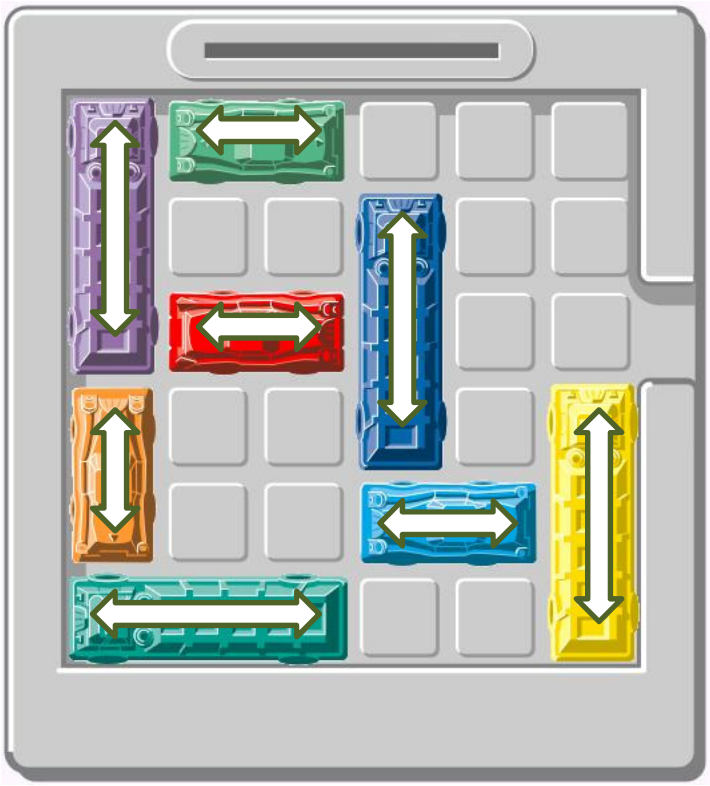
Applications of AI

Problem solving (e.g. Rush Hour)



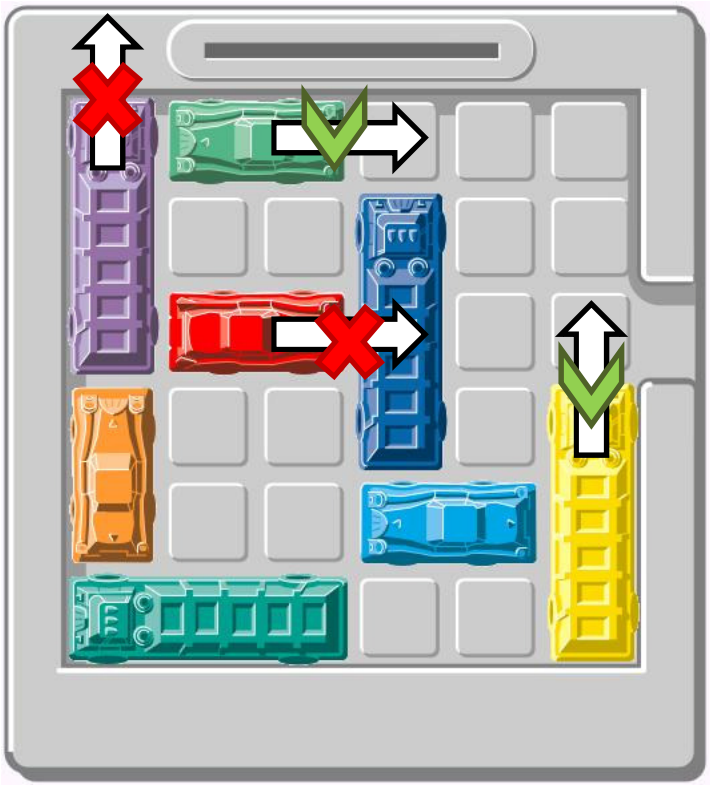
Applications of AI

Problem solving (e.g. Rush Hour)



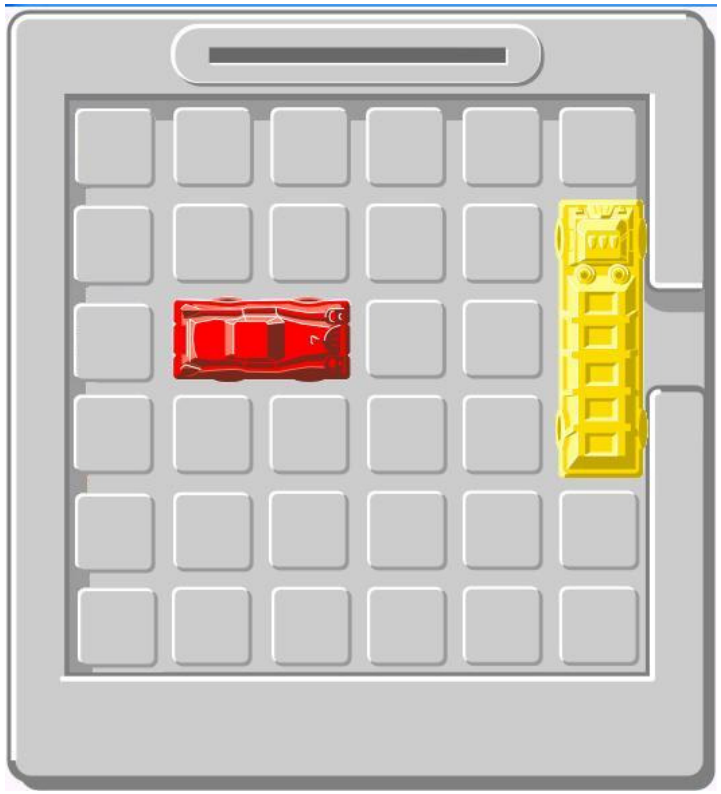
Applications of AI

Problem solving (e.g. Rush Hour)



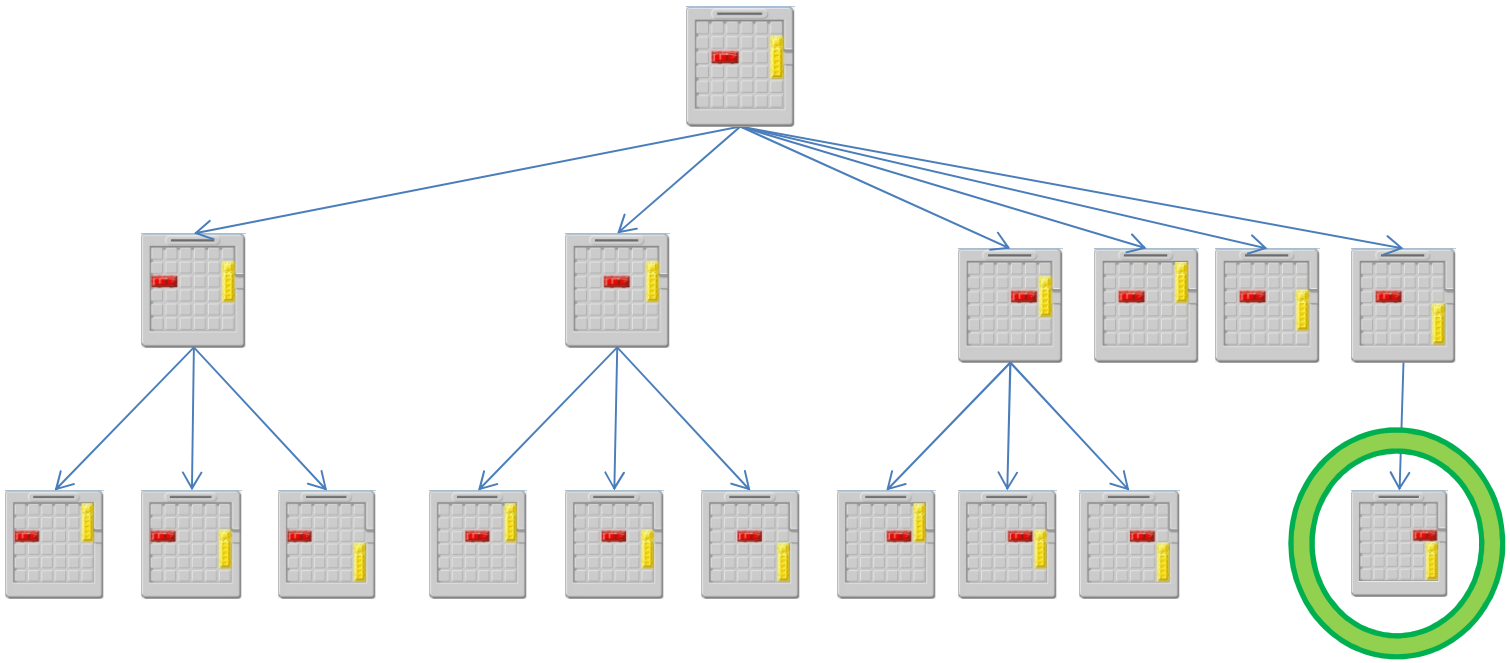
Applications of AI

Problem solving (e.g. Rush Hour)



Applications of AI

Problem solving (e.g. Rush Hour)



Solved by exploring the possibilities

Applications of AI

Problem solving (e.g. Rush Hour)

- Produces an intelligent response
- Is inspired by the human cognitive behaviour

But

- Requires prior knowledge
  - Exploring every state → solution discovery
  - Rules of the game (legal moves, transition graph)
- Many reachable states → long computational time

Applications of AI

Other approach:

- Let good players solve these puzzles
- Observe and memorize their actions

Red car	Blue car	Yellow truck	…	Action
[3,3,HOR]	[5,4,HOR]	[4,6,VERT]	…	[Blue car, left, 2]
[3,4,HOR]	[-,-,-]	[2,6,VERT]	…	[Yellow car, down, 2]
[3,2,HOR]	[2,1,VERT]	[-,-,-]	…	[Red car, right, 1]
[3,2,HOR]	[2,1,VERT]	[-,-,-]	…	[Blue car, up, 1]
…	…	…	…	…

Applications of AI

Solving the problem:

- Find the current situation in the database
- Apply the most frequent move

  → The solver acts like a good player.

  → Trading accuracy for computation reduction

- What if the current situation is *not* in the database?

  → Supervised Learning

Supervised Learning

- Problem: Raw data can be hard to exploit
- Example: Attractivity of long coats

Age	Size	Liking
39	1.75	LIKES
42	1.65	DISLIKES
27	1.82	DISLIKES
37	1.77	LIKES
…	…	…

- Would a 35-years old, 1.73m height man like a long coat?

Supervised Learning

- Problem: Raw data can be hard to exploit
- Example: Attractivity of long coats

Age	Size	Liking (/10)
39	1.75	8
42	1.65	3
27	1.82	2
37	1.77	9
…	…	…

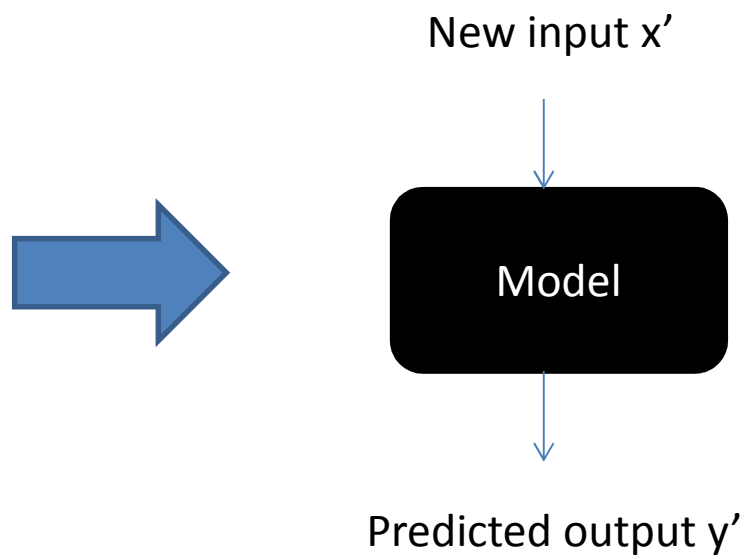
- How much would a 35-years old, 1.73m height man like a long coat?

Supervised Learning

- Objective: Model construction

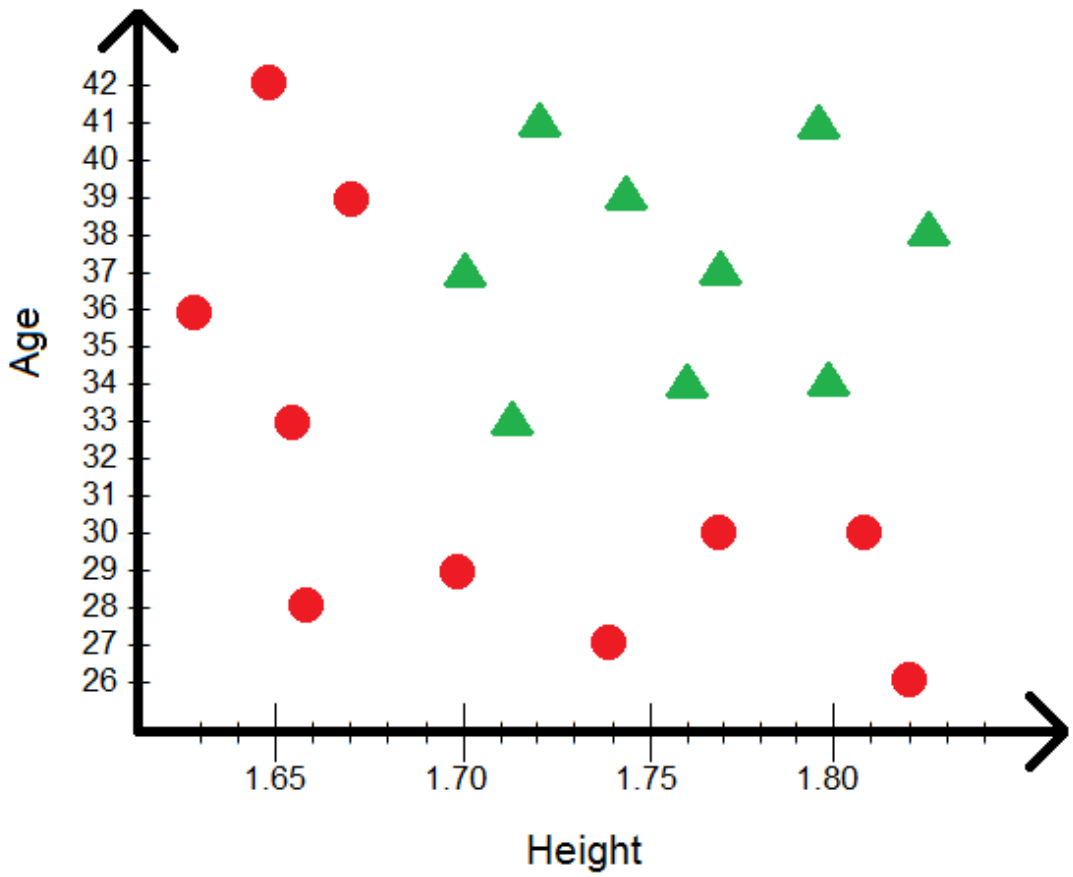
Age	Size	Liking
39	1.75	LIKES
42	1.65	DISLIKES
27	1.82	DISLIKES
37	1.77	LIKES
…	…	…

Inputs X Outputs Y



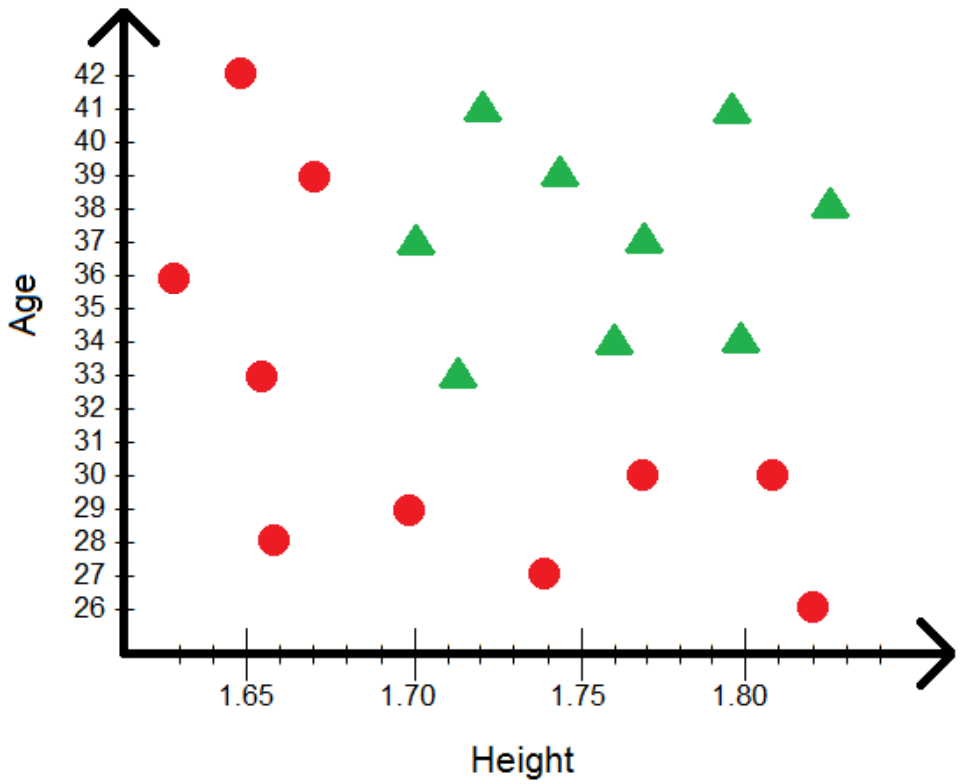
Supervised Learning

- Premise: Objects with the same class tend to cluster



Model construction

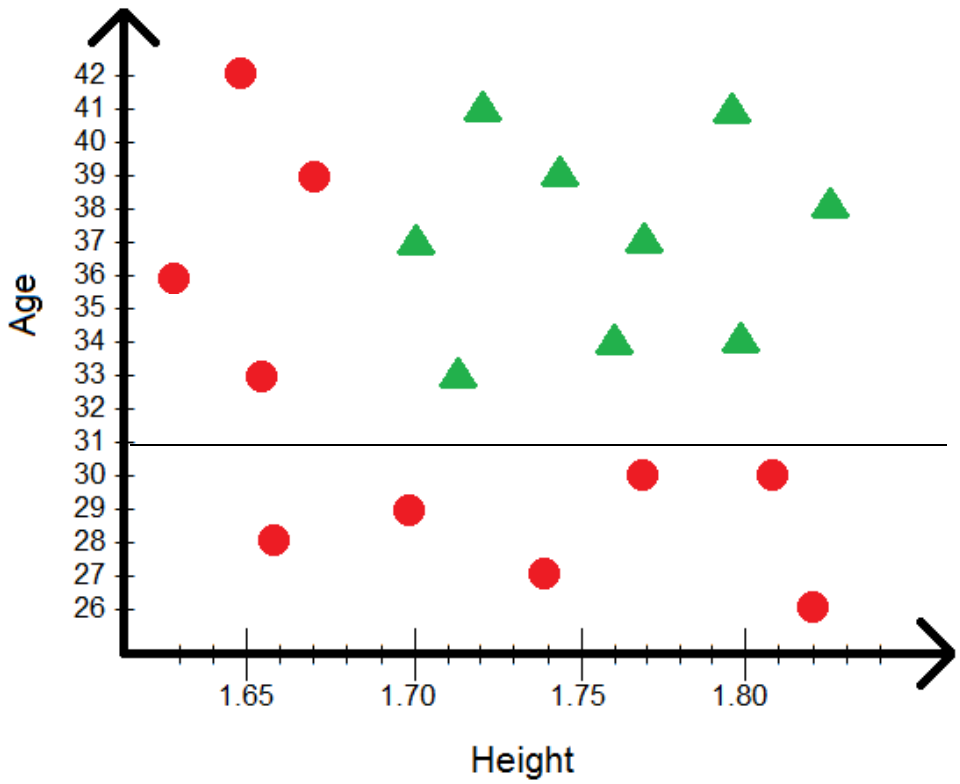
- Example : c4.5 algorithm

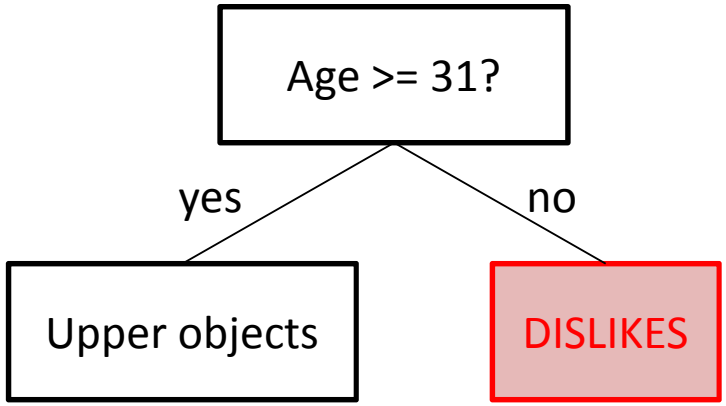


All objects

Model construction

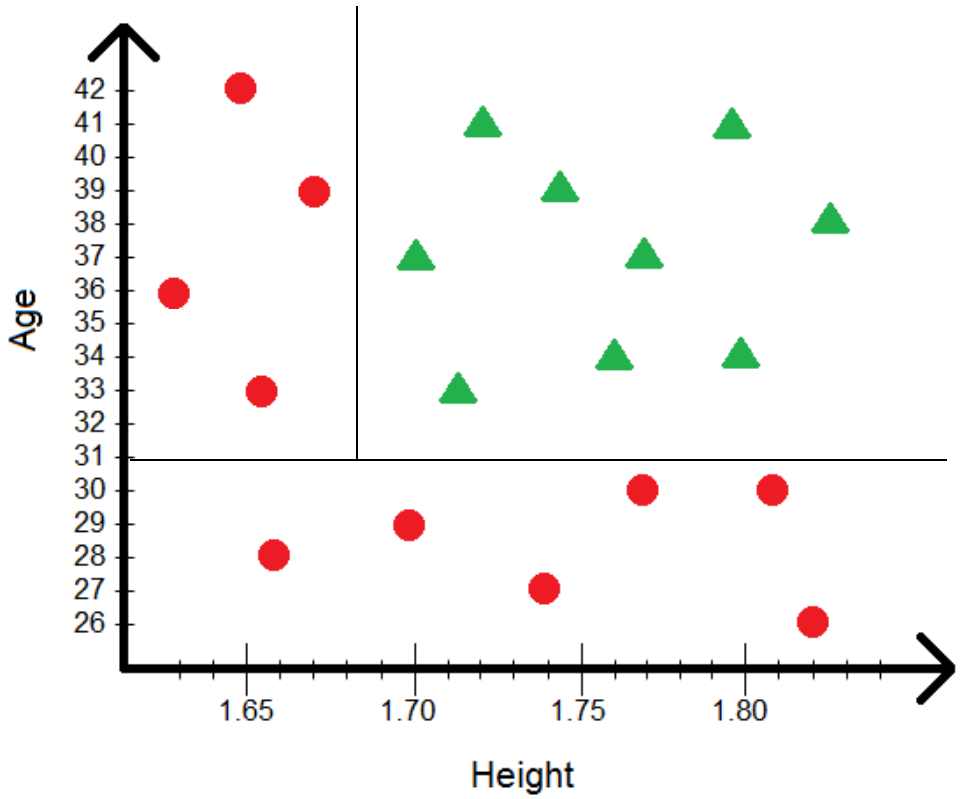
- Example : c4.5 algorithm

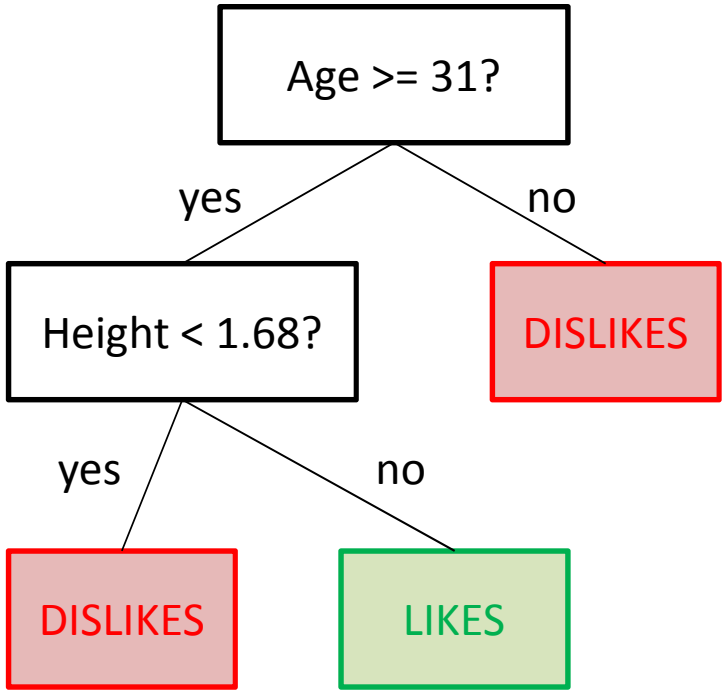




Model construction

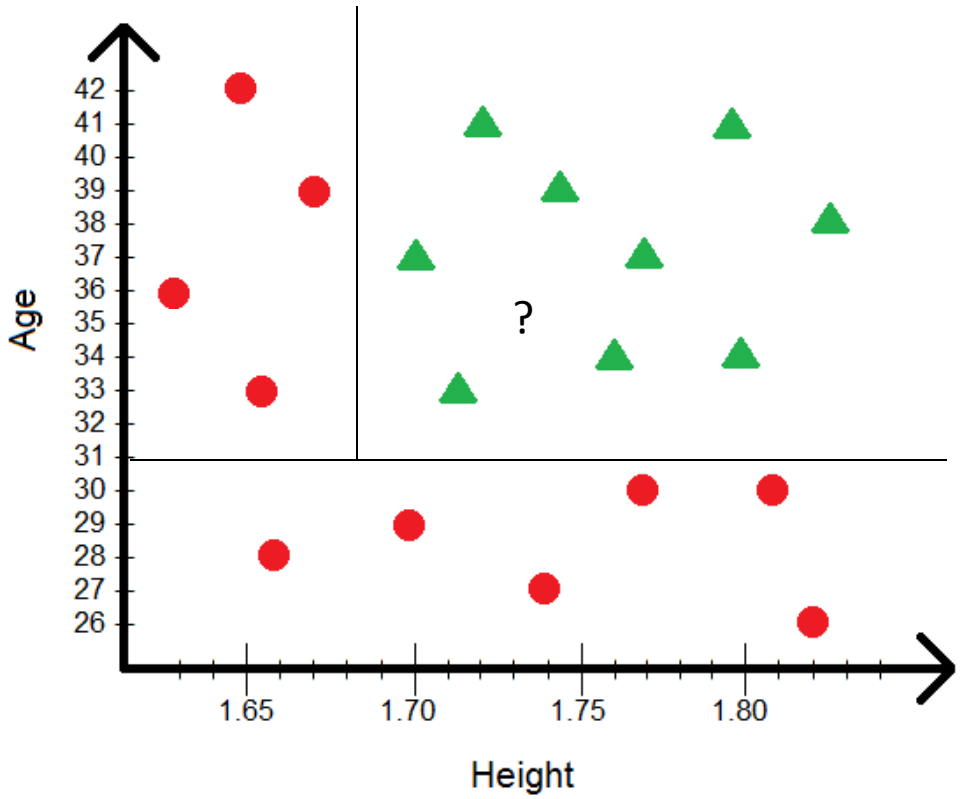
- Example : c4.5 algorithm

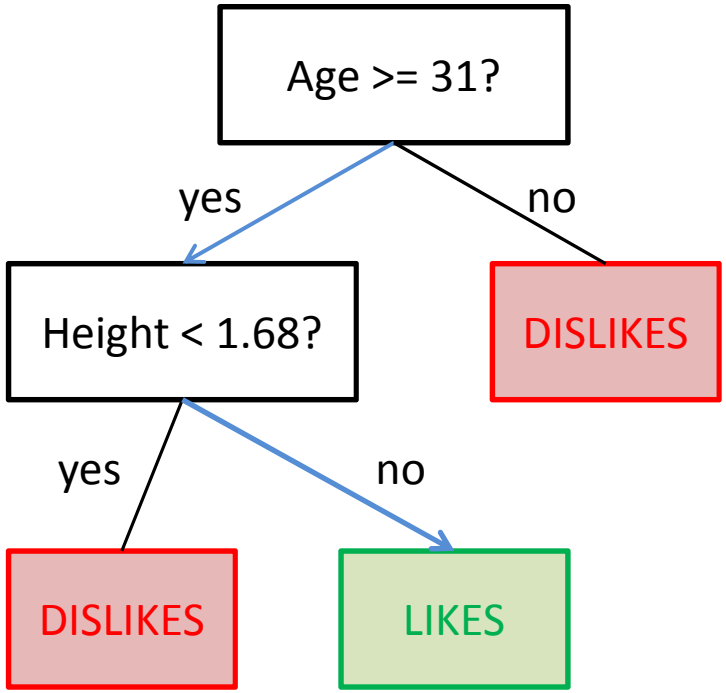




Model prediction

- Prediction for (possibly) unseen objects

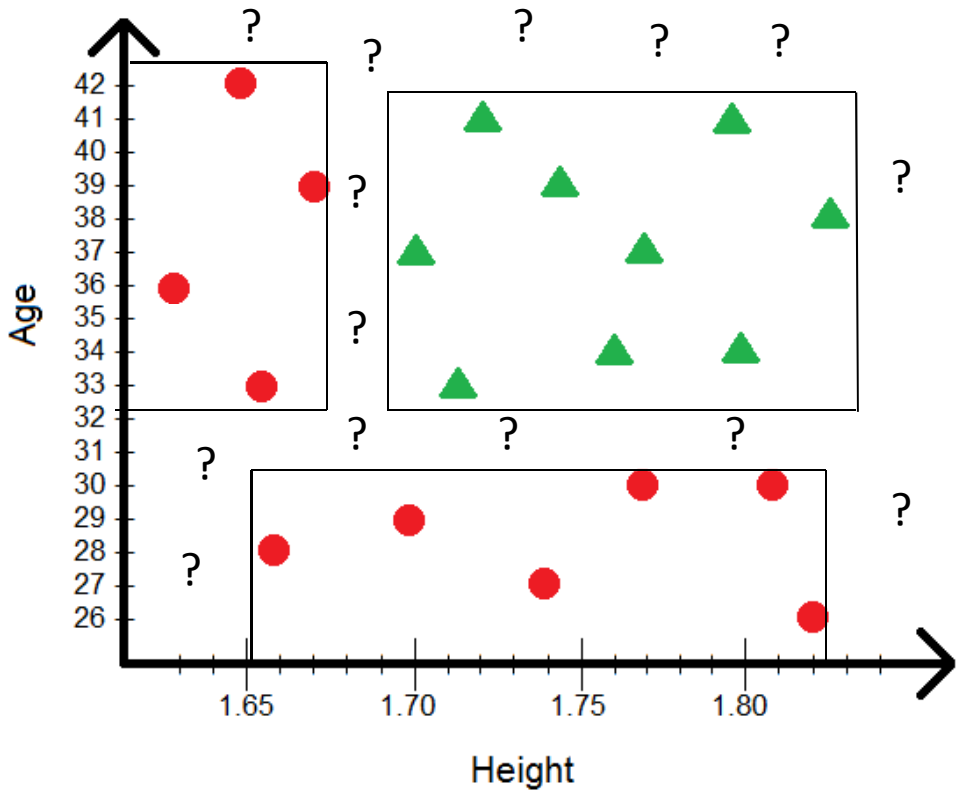




- A 35-year old, 1.73m height man will thus like a long coat

Model prediction

- Prediction for border objects can be problematic



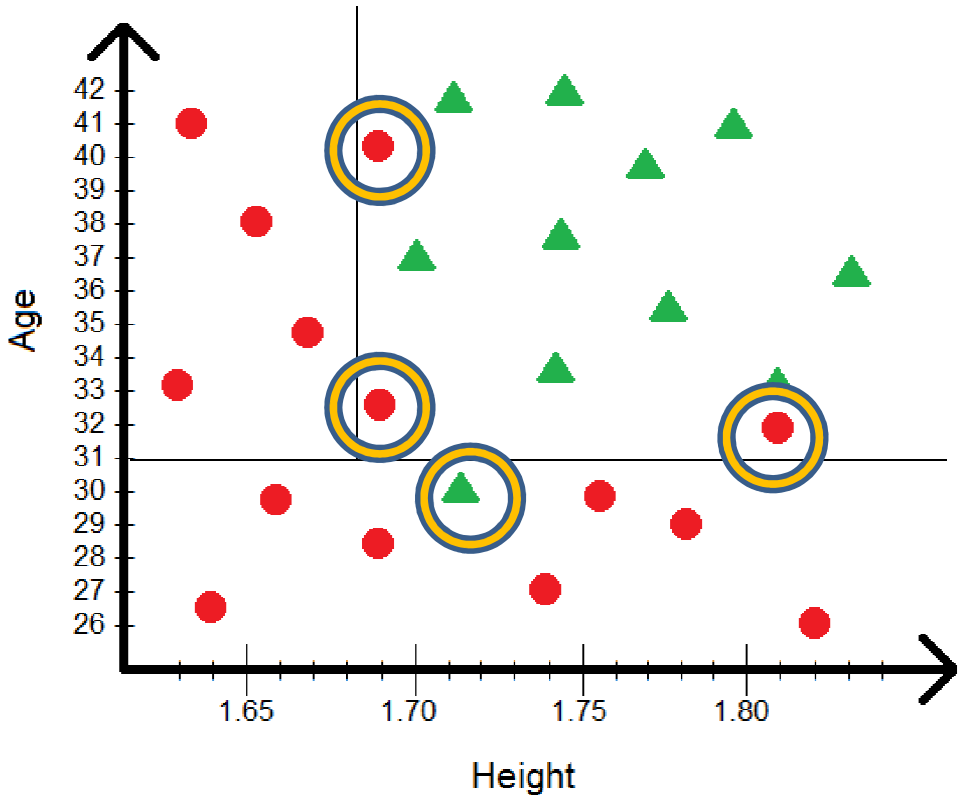
The model will always output something

Fitting the sample ≠ Representing the general rule

- We need to evaluate our model

Model evaluation

- Test on new data (resampling, partitioning)



$$\text{Accuracy} = \frac{\#\,well\ classified}{\#\,total\ predictions}$$

In this example,

accuracy = 21/25 = 84%

Different types of goal

Goal type (y)	Problem type	Example
1 symbolic value	Classification	y = « DISLIKES »
1 numerical value	Regression	y = 12.34
1 image	Image prediction Image completion	y = 
…	…	…
An ordered subset of symbolic values	**Label Ranking**	**y =**  **>**  **>** 

Label Ranking

- Particular case of Supervised Learning
- The output is not a number nor a symbol, but an order between labels (« ranking »), e.g.

A > B > C > D	A > B > C
(total order)	(partial order, more frequent in practice)

- Provides more information than a single label.

  E.g.: A book store (say amazon) can perform recommendations to increase its sales rate.

Recommender systems

ID	Age	Size
12	32	1.68

New user

Is submitted to

User ratings

Are used to build

Model

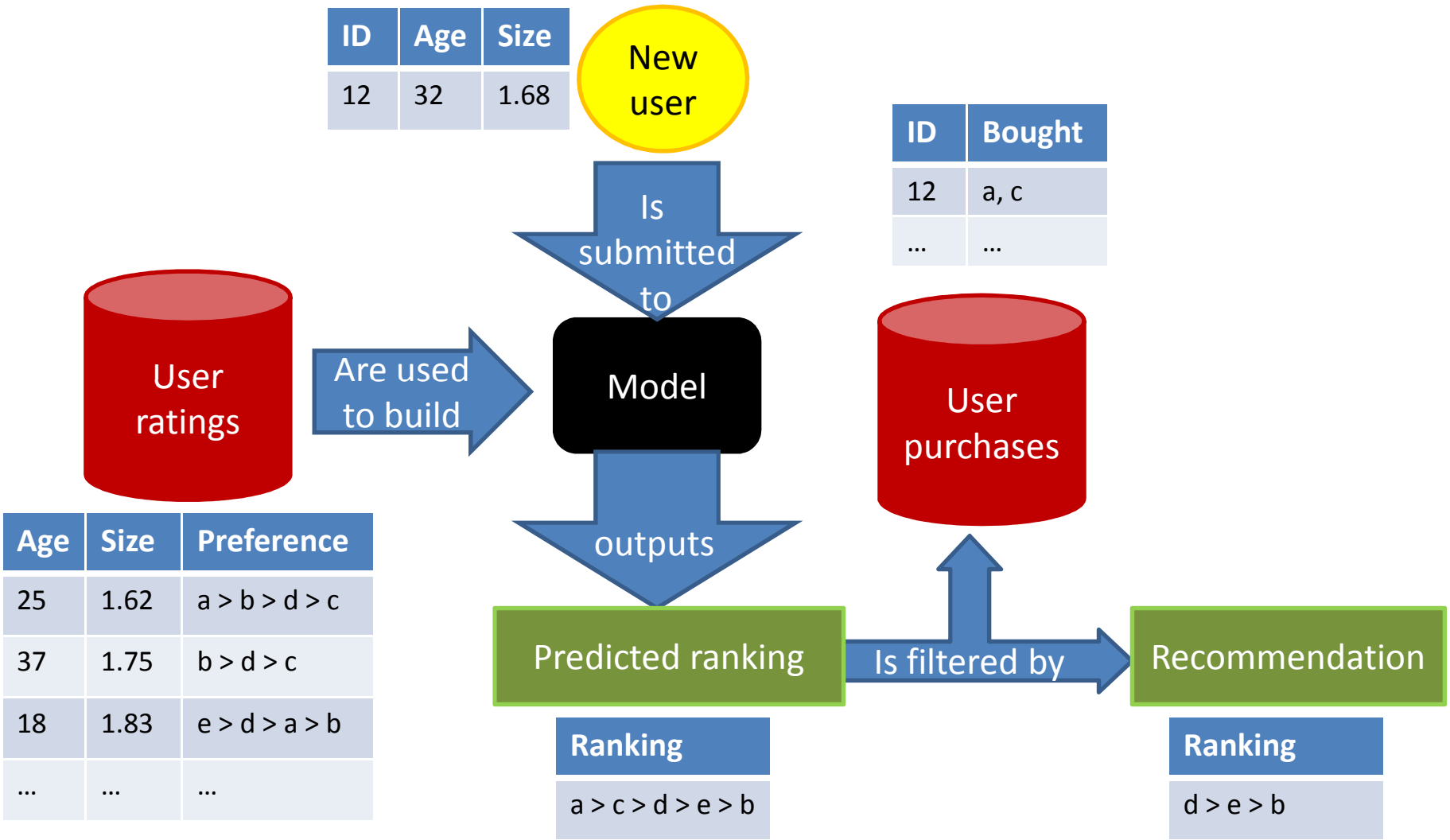
outputs

Predicted ranking

Age	Size	Preference
25	1.62	a > b > d > c
37	1.75	b > d > c
18	1.83	e > d > a > b
…	…	…

Ranking
a > c > d > e > b

Recommender systems



ID	Age	Size
12	32	1.68

ID	Bought
12	a, c
…	…

Age	Size	Preference
25	1.62	a > b > d > c
37	1.75	b > d > c
18	1.83	e > d > a > b
…	…	…

Ranking
a > c > d > e > b

Ranking
d > e > b

Label Ranking

- Could be solved by standard SL algorithms, by attributing a number to each possible permutation (e.g. lexicographic order) , but:
  - Would be inefficient with partial rankings
  - Would not take advantage of the structure, e.g.:

Single label displacement

Very different permutations

A > B > C > D > E → 1

E > A > B > C > D → 97

D > E > C > B > A → 96

Very distant numbers

Close numbers

→ Algorithms specific to label ranking

Label Ranking

- Decompose the problem
  - Learn a utility function per label,

    « How much would you rate this item? »

    e.g.:

    - U() = 2, U() = 1, U() = 3,

    →  >  > 

Label Ranking

- Decompose the problem
  - Learn a model for each pair of labels,

    « Would you rate item A higher than item B? »

    e.g.:

    -  >  and  >  and  > 

    →  >  > 

Strategy used in this work (RPC)

Ranking by Pairwise Comparison (RPC)

Eyke Hüllermeier and Johannes Fürnkranz, 2004

- Step 1: Decomposition into pairwise comparisons

x^i = (age, size, …) for i^{th} object

Original data

Input	Preference
x^1	a > b > c
x^2	b > c > a
x^3	c > b > a
x^4	a > c

a > b

Input	Preference
x^1	true
x^2	false
x^3	false
x^4	-

a > c

Input	Preference
x^1	true
x^2	false
x^3	false
x^4	true

b > c

Input	Preference
x^1	true
x^2	true
x^3	false
x^4	-

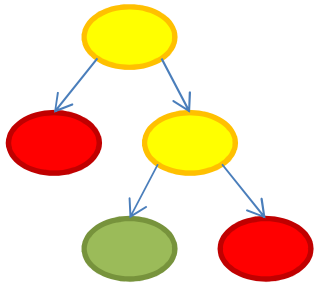
Ranking by Pairwise Comparison (RPC)

Eyke Hüllermeier and Johannes Fürnkranz, 2004

- Step 2: Training of all pairwise comparators

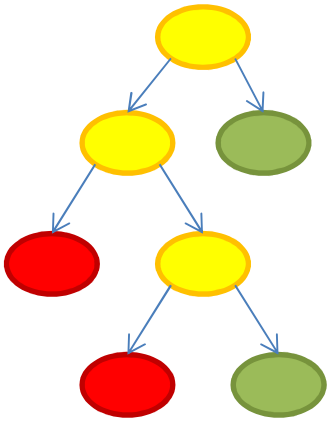
a > b

Input	Preference
x^1	true
x^2	false
x^3	false



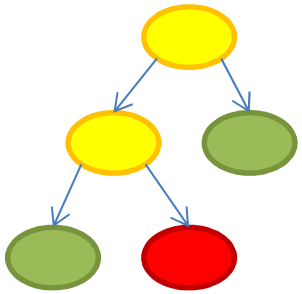
a > c

Input	Preference
x^1	true
x^2	false
x^3	false
x^4	true



b > c

Input	Preference
x^1	true
x^2	true
x^3	false



Ranking by Pairwise Comparison (RPC)

Eyke Hüllermeier and Johannes Fürnkranz, 2004

- Prediction: Aggregating comparator outputs

New Input x^5

Comparator	Prediction		
a > b	a → + 0	b → + 1	
a > c	a → + 1		c → + 0
b > c		b → + 1	c → + 0

Class	Score
a	1
b	2
c	0

b > a > c

Complexity issue of RPC

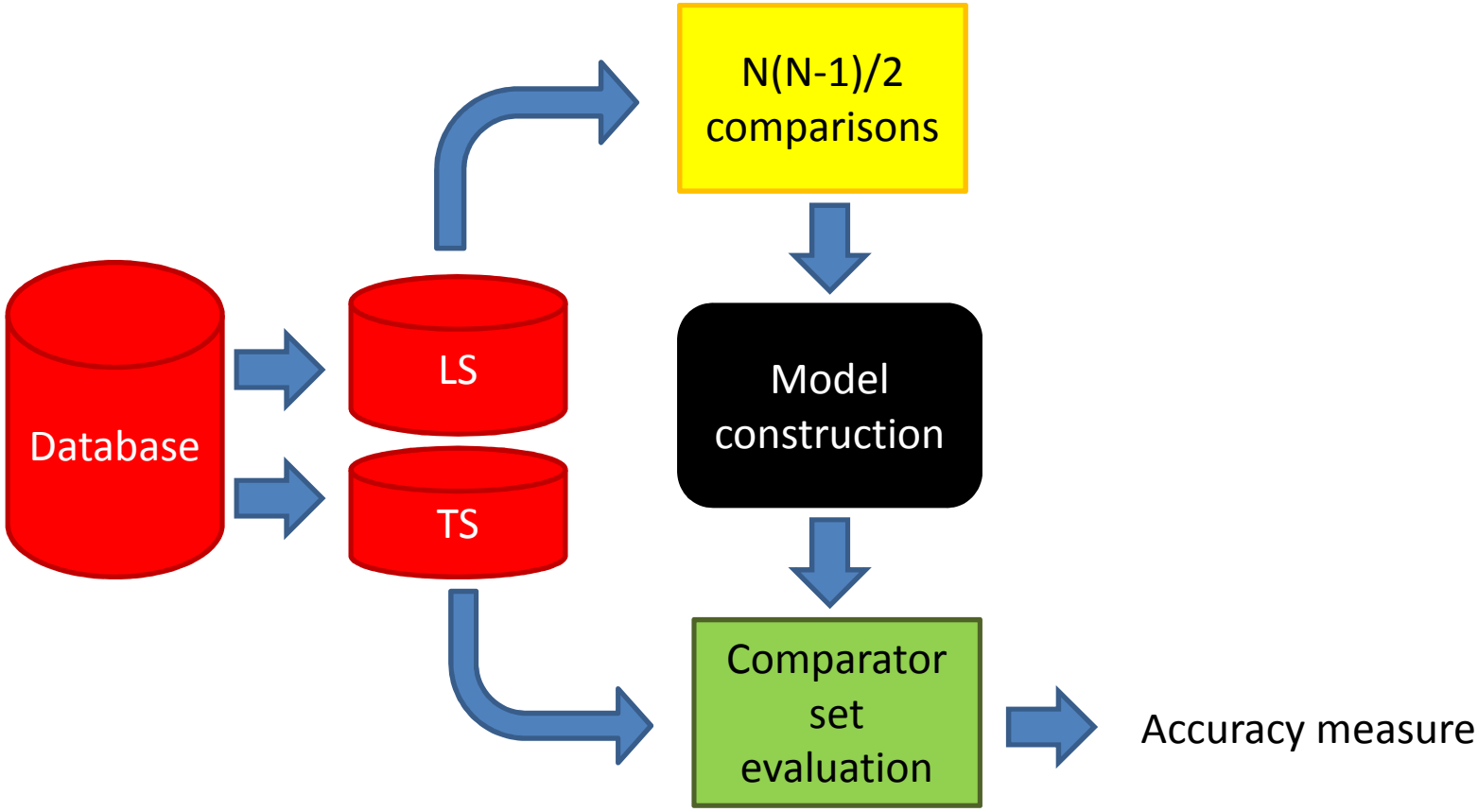
N = 10	→	#Comp = 45
N = 100	→	#Comp = 4,950
N = 1000	→	#Comp = 499,500

- Needs $O(N^2)$ in terms of number N of labels
- Computational and Space complexity
- Non applicable in practice to huge datasets

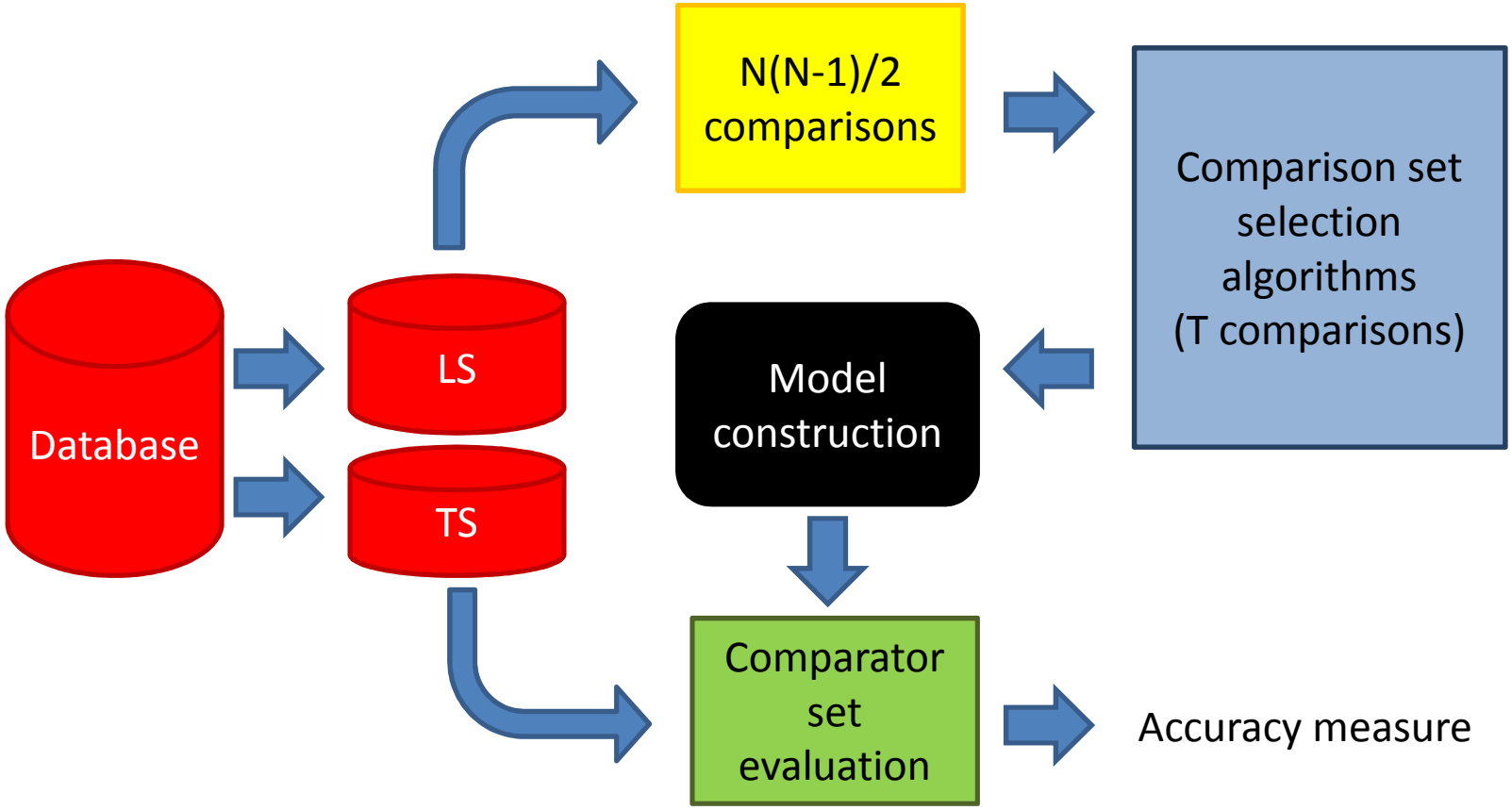
Contribution

- Intuition:
  - There might be some redundant and/or noisy comparisons in this set of $O(N^2)$ comparisons (no theoretical evidence)
  - Reducing this set would improve the speed performance of RPC and could be (nearly) as accurate
- Contribution:
  - The design of algorithms aiming at selecting the optimal subset of *T* comparisons and the study of the behaviour of a RPC scheme in terms of accuracy vs. number *T* of comparisons.

Contribution



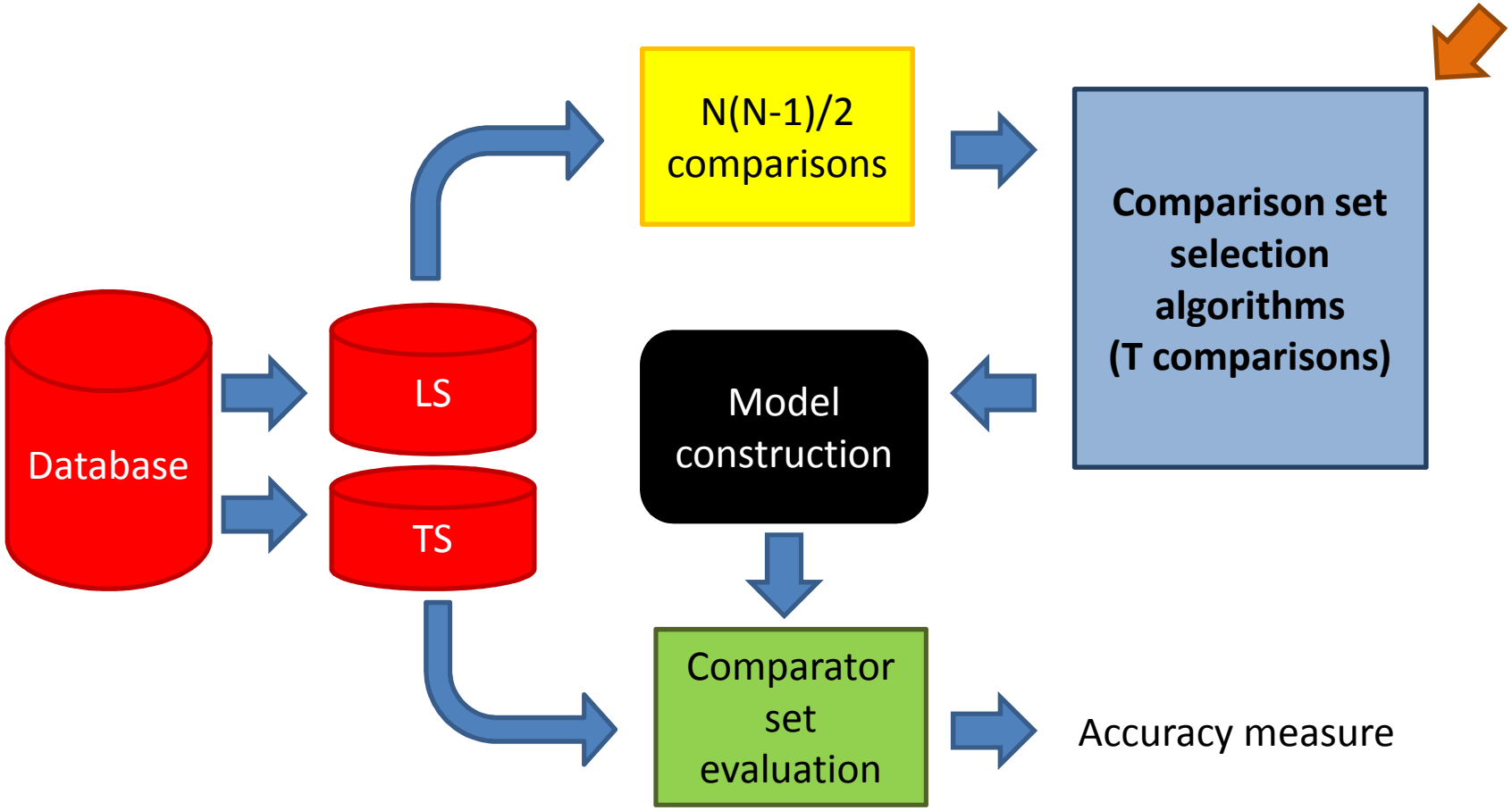
Contribution



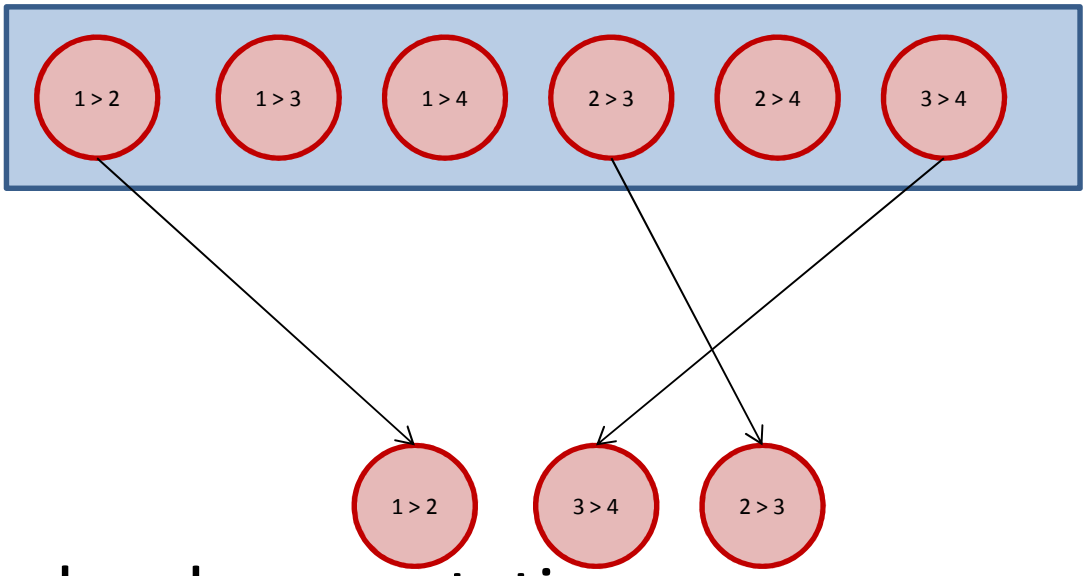
Outline

- 1st Part
  - Context and definitions
  - Problem setting
  - Contribution
- 2nd Part
  - Methods and algorithms
    - Subset selection methods (PR, EDA, EGS, RGS)
    - Evaluating a prediction and a subset of comparisons
    - Base learner : Extra-trees
  - Empirical validation
    - Databases
    - Results and parameters discussion
    - Complexity reduction
  - Conclusion

Guideline


N(N-1)/2 comparisons
Comparison set selection algorithms (T comparisons)
Database
LS
TS
Model construction
Comparator set evaluation
Accuracy measure

Pure random selection (PR)



All available comparisons

Random uniform selection

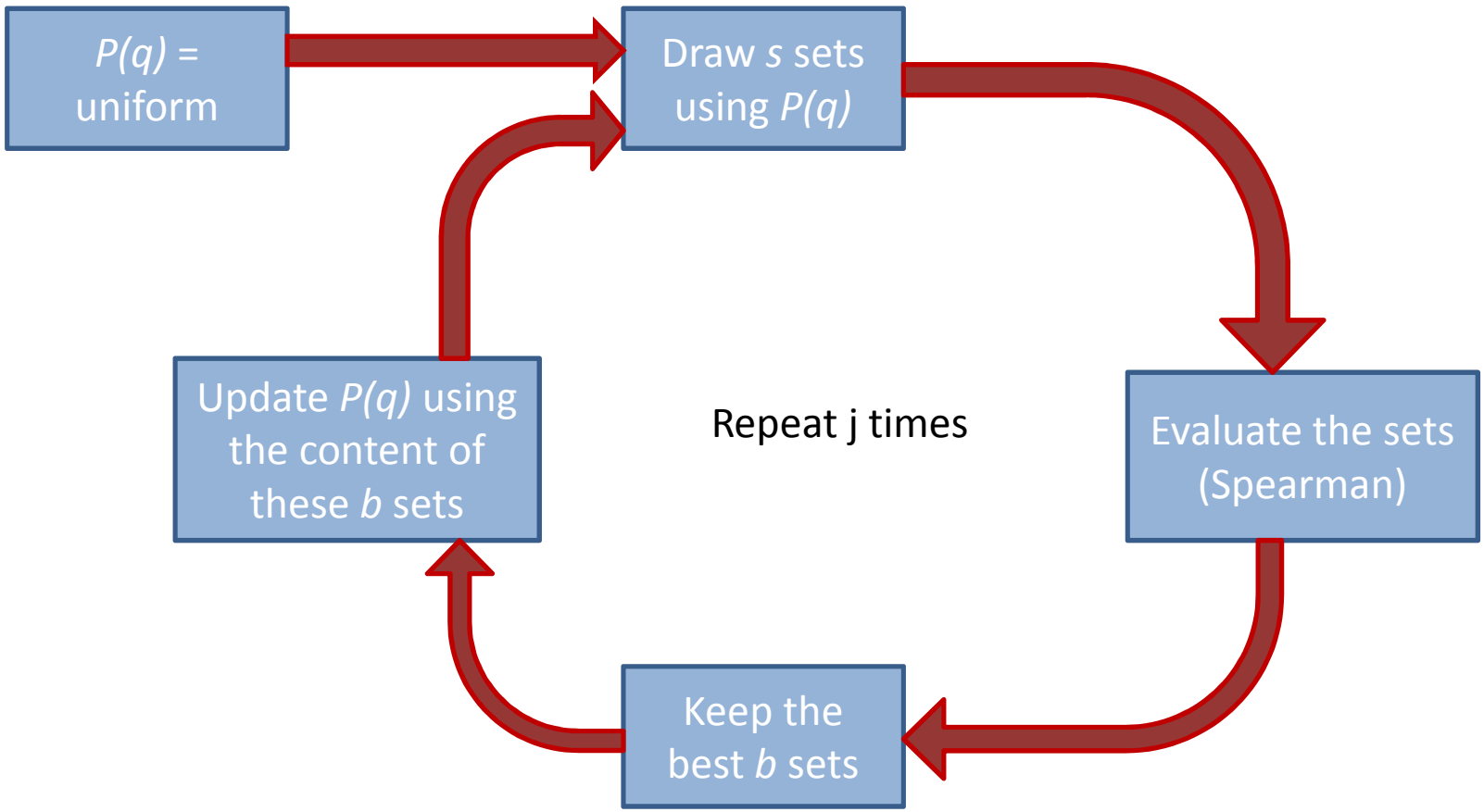
Subset of size T

- No hard computations
- No need to store the LS outputs

  → Baseline

Complexity (# subset evaluations) : 0

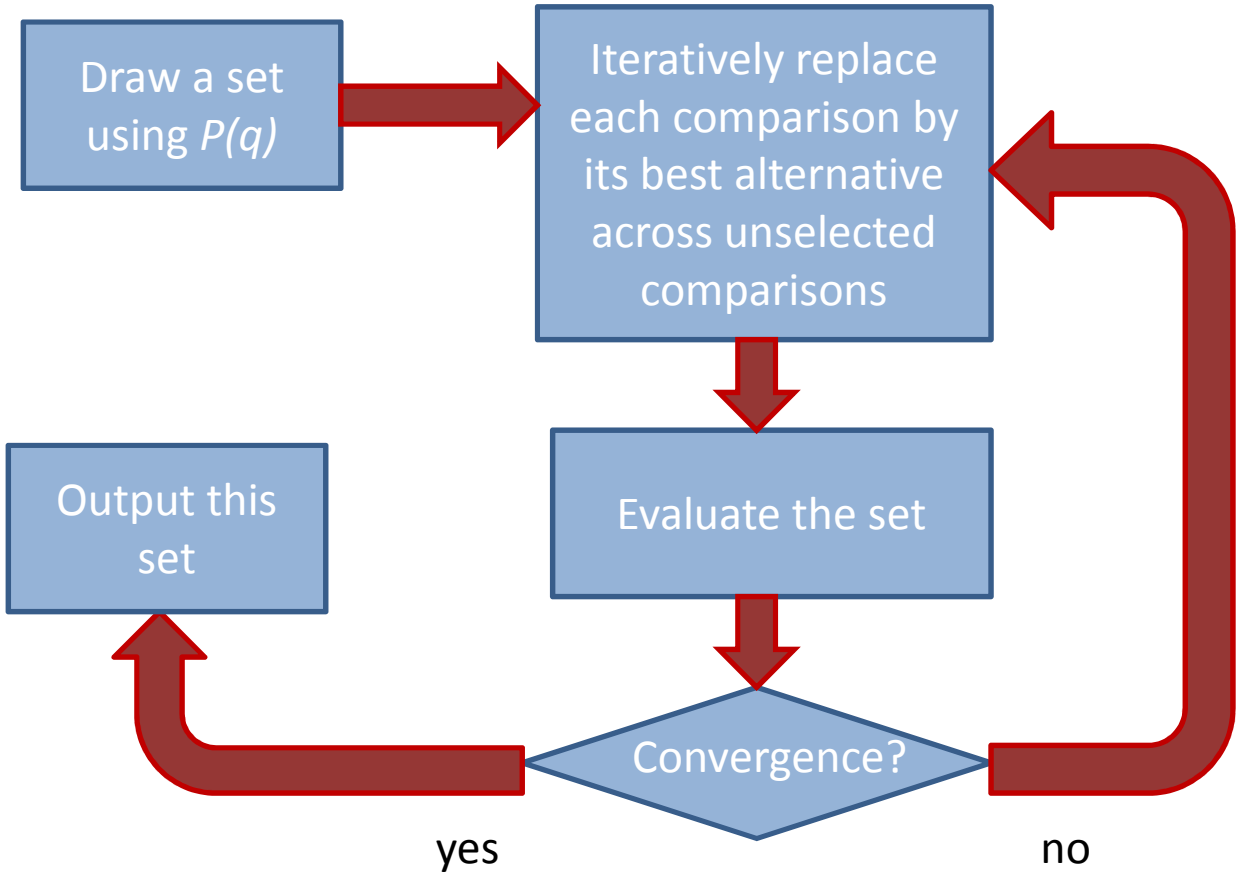
Estimation of distribution algorithm (EDA)



Output: Best set across all iterations

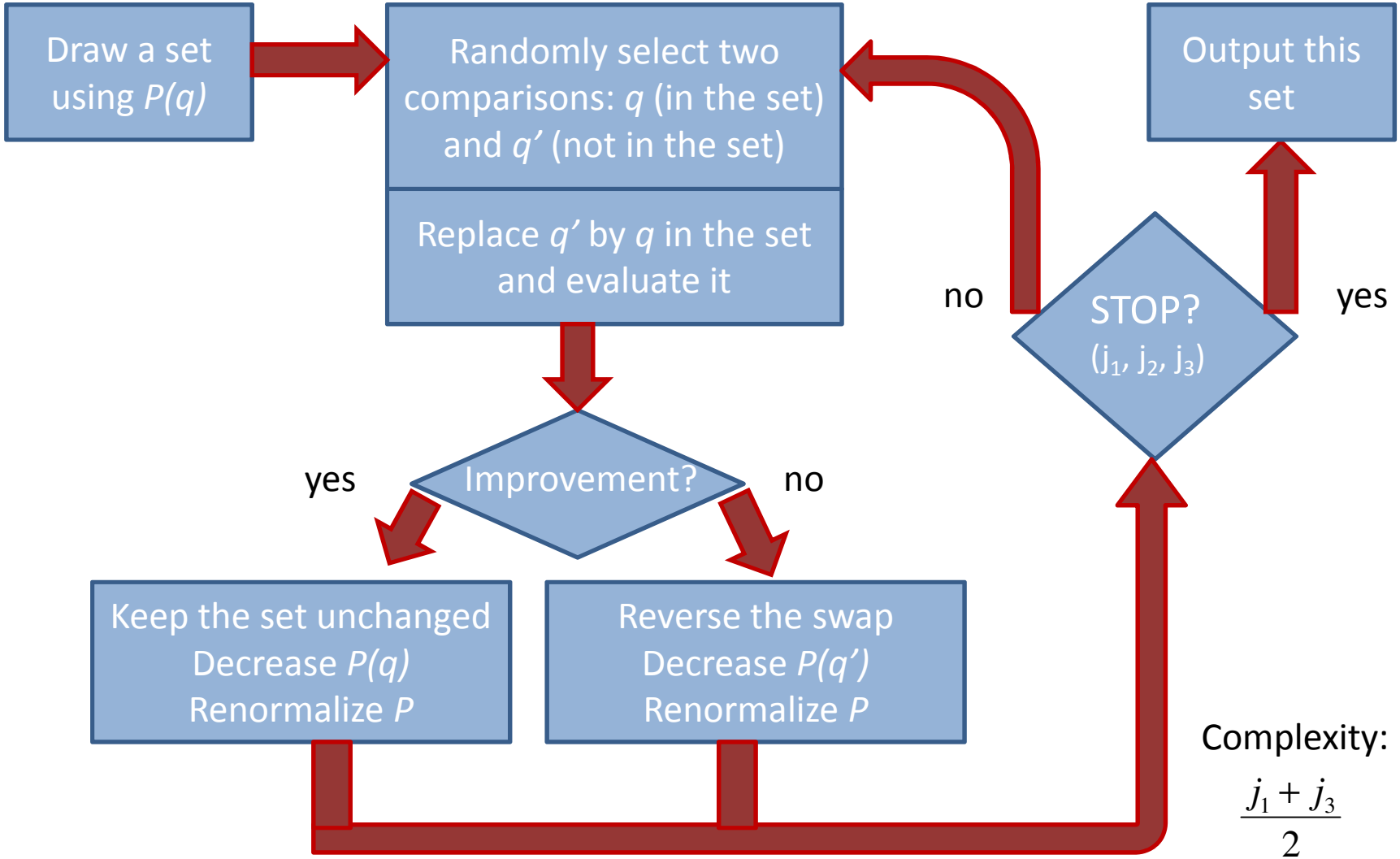
Complexity: *sj*

« Exhaustive » Greedy Search (EGS)



Complexity: $jT\,\mathcal{O}(N^2)$

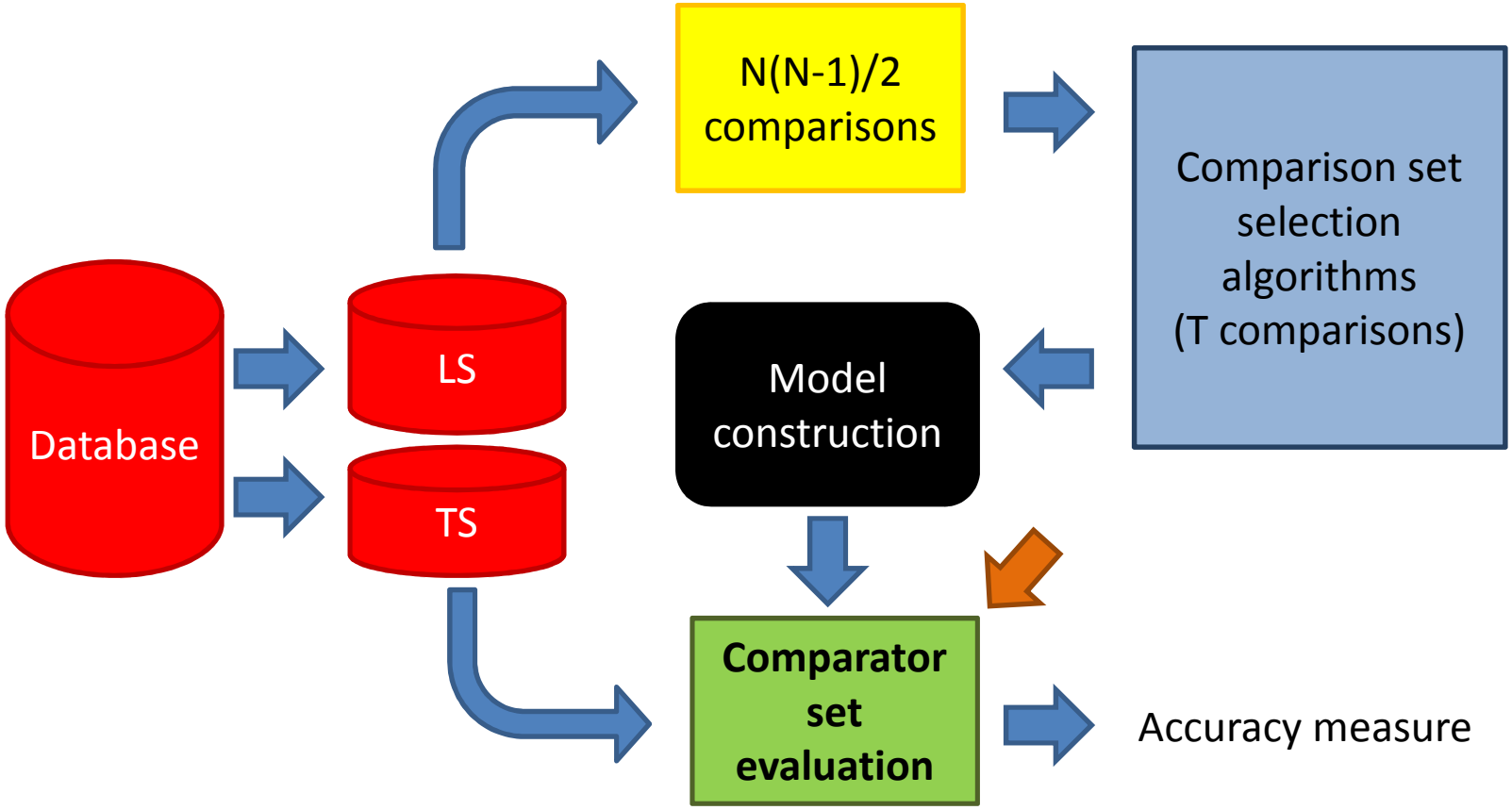
Randomized Greedy Search (RGS)



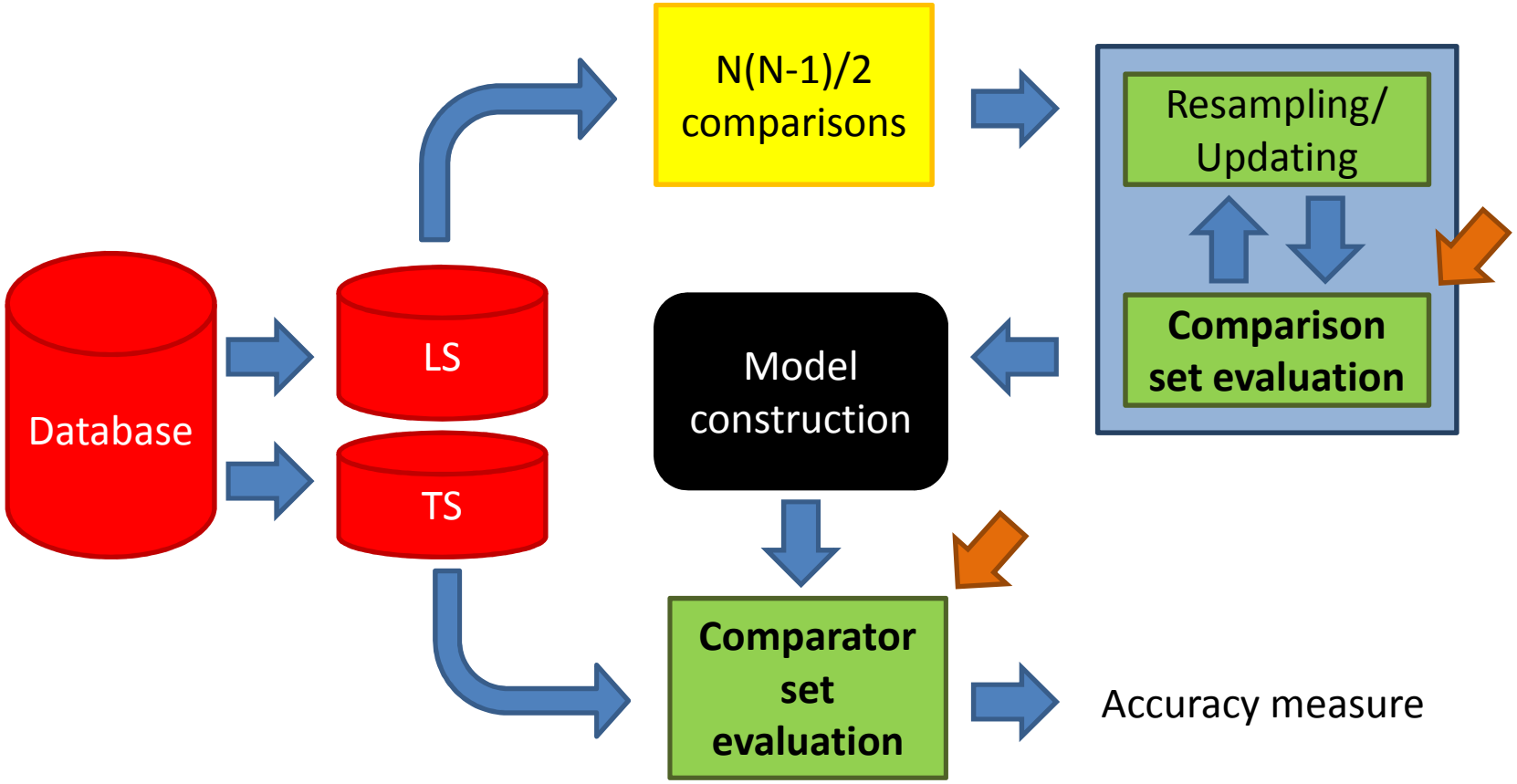
Complexity:

$$\frac{j_1 + j_3}{2}$$

Guideline


N(N-1)/2 comparisons
Comparison set selection algorithms (T comparisons)
Database
LS
TS
Model construction
Comparator set evaluation
Accuracy measure

Guideline


N(N-1)/2 comparisons
Resampling/ Updating
Comparison set evaluation
Database
LS
TS
Model construction
Comparator set evaluation
Accuracy measure

Evaluating a model on a sample

- 0/1 Loss

$$L = \sum_{i=1}^{\#TS} \begin{cases} 1 & \hat{y}^i \neq y^i \\ 0 & \hat{y}^i = y^i \end{cases}$$

y^i = true output of object i
$\hat{y}^i$ = predicted output for object i

- Least squares error

$$L = \sum_{i=1}^{\#TS} (\hat{y}^i - y^i)^2$$

Computing the similarity between two rankings

- Spearman's rank correlation coefficient

$$(y, \hat{y}) \rightarrow \rho_S = 1 - \frac{6\sum_{k=0}^{\# y}(y_k - \hat{y}_k)^2}{\# y((\# y)^2 - 1)}$$

$\# y =$ Number of labels in ranking y

$y_k =$ Position of label k in ranking y

- Kendall Tau rank correlation coefficient

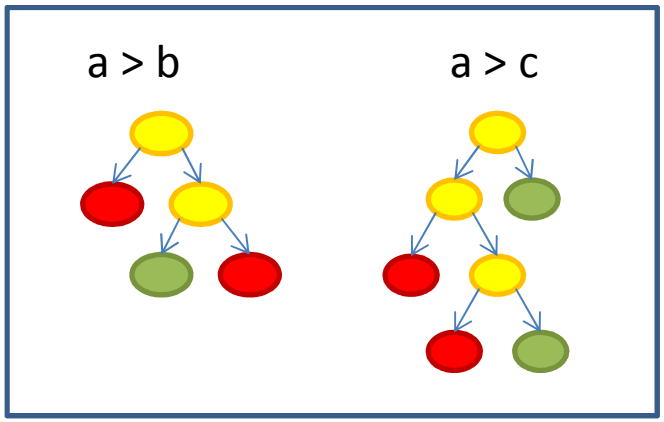
$$(y, \hat{y}) \rightarrow \rho_K = 1 - \frac{4\#\{(k,l) \mid (k < l) \wedge (y_k > y_l) \wedge (\hat{y}_k < \hat{y}_l)\}}{\# y(\# y - 1)}$$

Evaluating the subset of comparisons

- Overall ranking score S_{TS} on the sample *TS*

$$S_{TS} = \frac{\sum_{i=1}^{\#TS} \rho(\hat{y}^i, y^i)}{\#TS}$$

Evaluating a set of comparisons on a sample



Default order = a > b > c

Input	Prediction
x^1	a > c > b
x^2	b > c > a
x^3	a > b > c
x^4	b > a > c

Ordered Subset
a > c > b
b > c > a
a > b > c
a > c

Spearman
0.5
1
-1
1

Input	Supervision
x^1	a > b > c
x^2	b > c > a
x^3	c > b > a
x^4	a > c

Overall ranking score S_{TS} = (0.5+1-1+1)/4 = 0.375

Optimization phase: another computational issue

Evaluating a set of comparisons during the optimization phase

- N = 4, #Comp = 6, T = 3, RGS

Models

No	No	No	No	No	No
0	1	2	3	4	5

Evaluating a set of comparisons during the optimization phase

- N = 4, #Comp = 6, T = 3, RGS

Models

No	Yes	No	Yes	Yes	No
0	1	2	3	4	5

- Iteration 1: Comparisons = {1, 3, 4}

Evaluating a set of comparisons during the optimization phase

- N = 4, #Comp = 6, T = 3, RGS

Models

No	Yes	Yes	Yes	Yes	No
0	1	2	3	4	5

- Iteration 1: Comparisons = {1, 3, 4}
- Iteration 2: Comparisons = {2, 3, 4}

Evaluating a set of comparisons during the optimization phase

- N = 4, #Comp = 6, T = 3, RGS

Models

Yes	Yes	Yes	Yes	Yes	Yes
0	1	2	3	4	5

- Iteration 1: Comparisons = {1, 3, 4}
- Iteration 2: Comparisons = {2, 3, 4}
- …
- Iteration j_3: Comparisons = {0, 3, 5}

Evaluating a set of comparisons during the optimization phase

- N = 4, #Comp = 6, T = 3, RGS

Models

Yes	Yes	Yes	Yes	Yes	Yes
0	1	2	3	4	5

- Iteration 1: Comparisons = {1, 3, 4}
- Iteration 2: Comparisons = {2, 3, 4}
- …
- Iteration j_3: Comparisons = {0, 3, 5}

- → Each model was built and stored.

  Same complexity issue as RPC.

Evaluating a set of comparisons on the LS

a > b

Input	Preference
x_1	true
x_2	false
x_3	false

a > c

Input	Preference
x_1	true
x_2	false
x_3	false
x_4	true

Default order = a > b > c

Input	Prediction
x_1	a > b > c
x_2	b > c > a
x_3	b > c > a
x_4	a > b > c

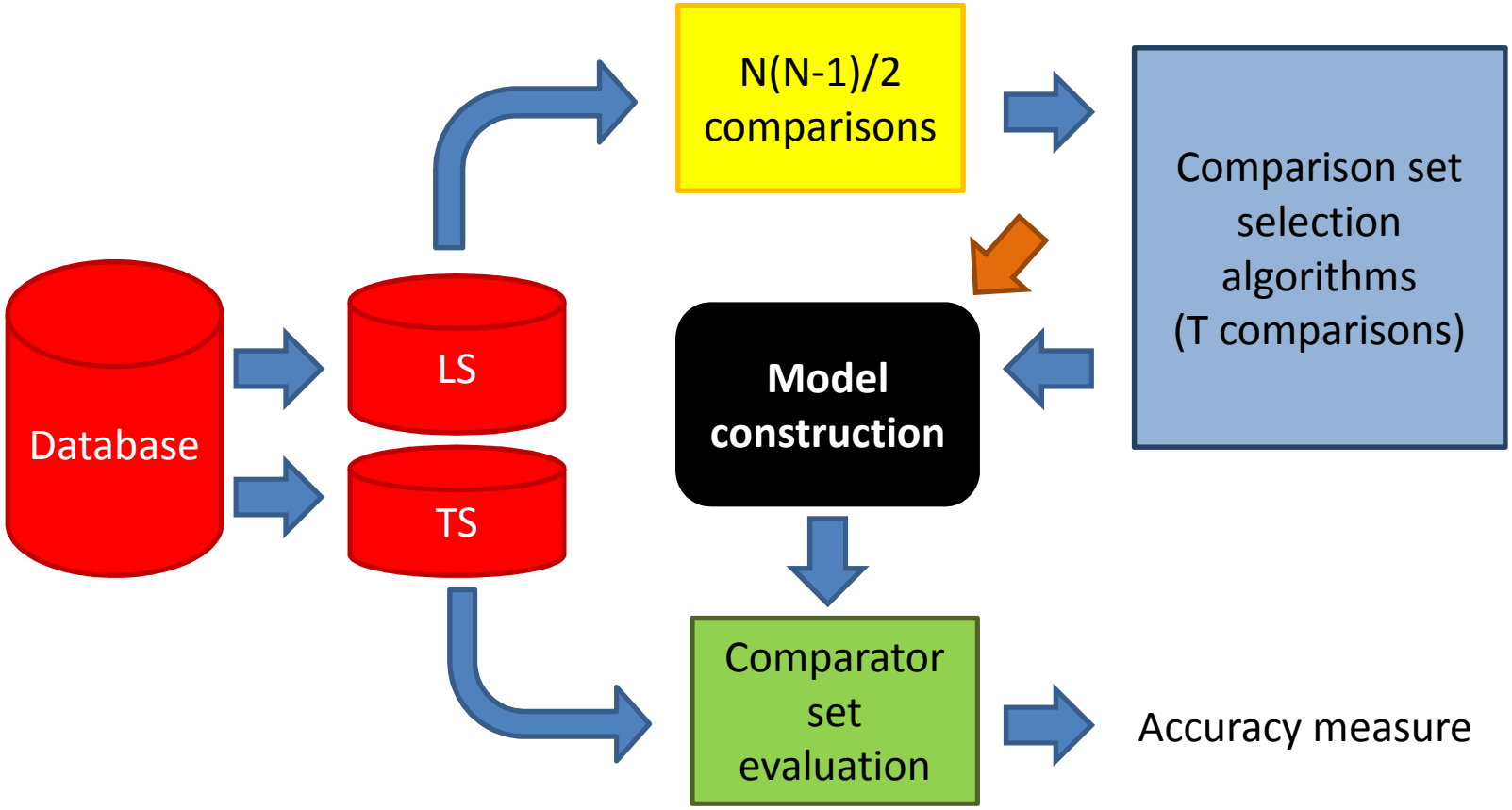
Ordered Subset
a > b > c
b > c > a
b > c > a
a > c

Spearman
1
1
0.5
1

Input	Supervision
x_1	a > b > c
x_2	b > c > a
x_3	c > b > a
x_4	a > c

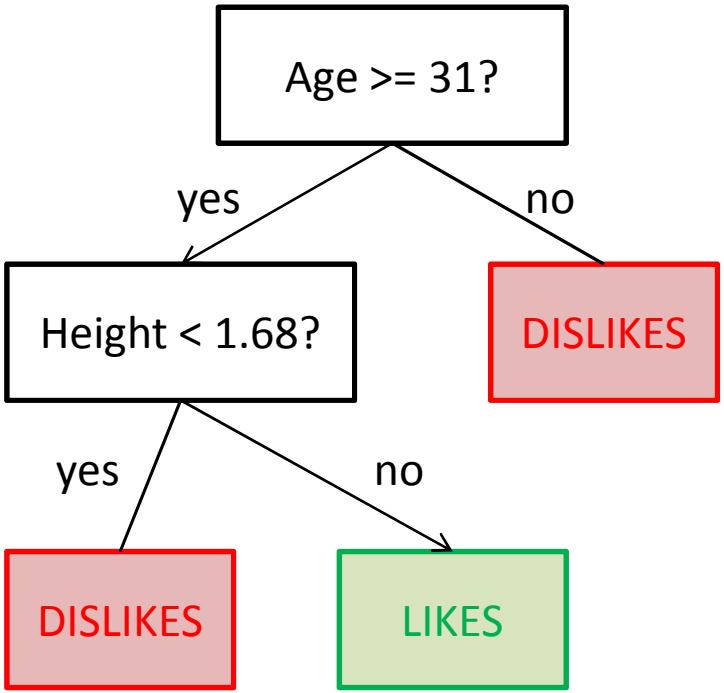
Overall ranking score = (1+1+0.5+1)/4 = 0.875

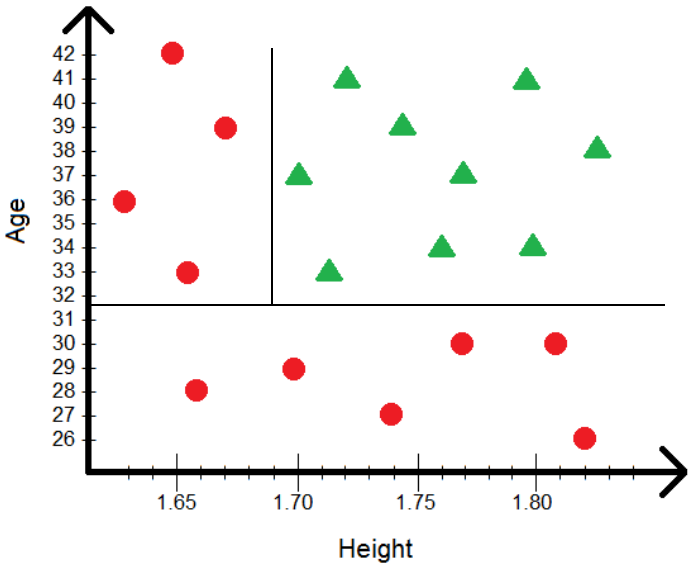
Guideline


N(N-1)/2 comparisons
Comparison set selection algorithms (T comparisons)
Database
LS
TS
Model construction
Comparator set evaluation
Accuracy measure

Extremely Randomized Trees

Decision tree:

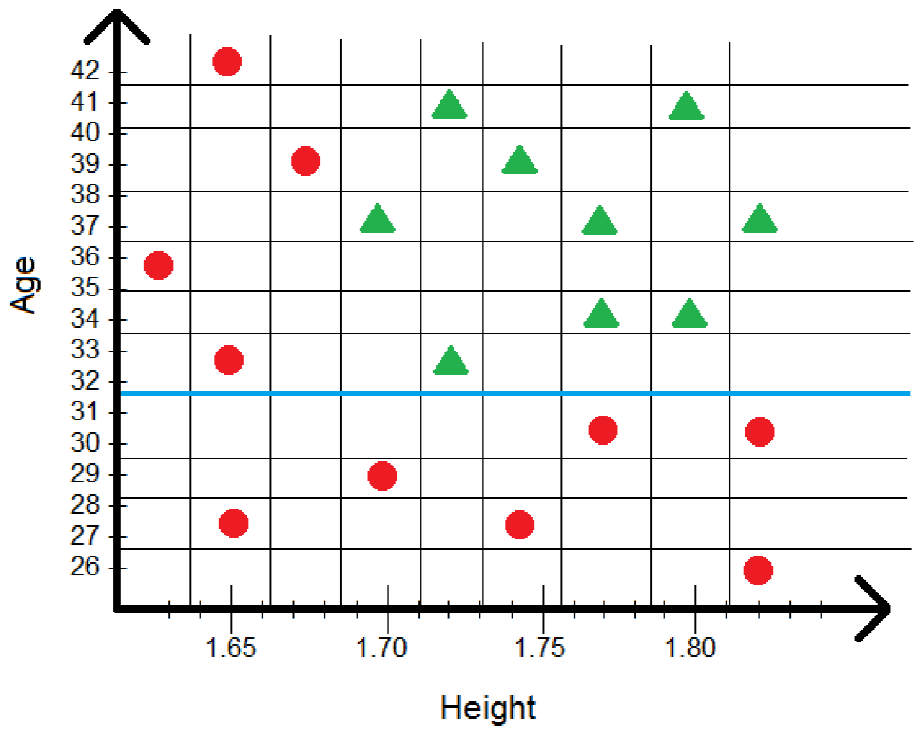




Age	Size	Liking
39	1.75	LIKES
42	1.65	DISLIKES
27	1.82	DISLIKES
…	…	…

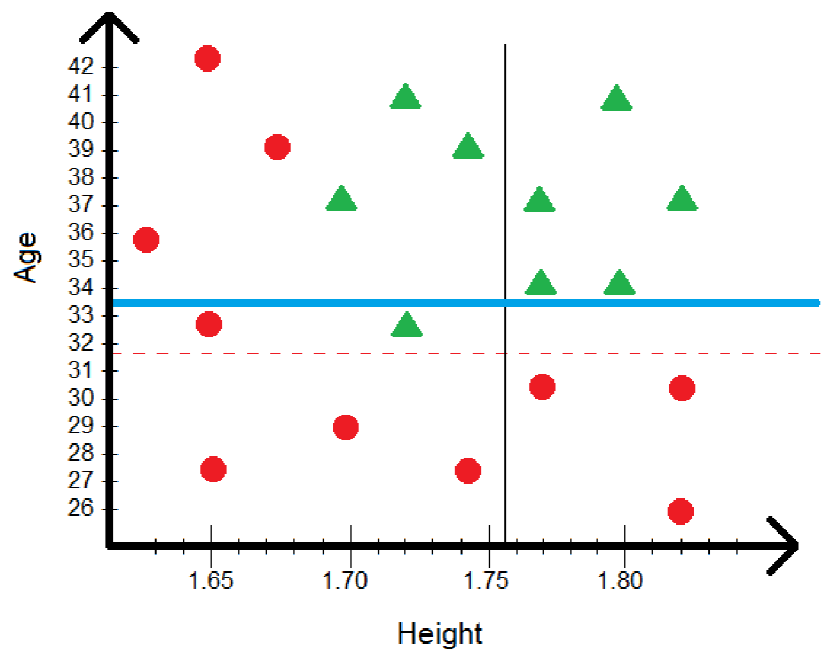
Extremely Randomized Trees

- Decision trees : The node split which minimizes the entropy is chosen (deterministic)



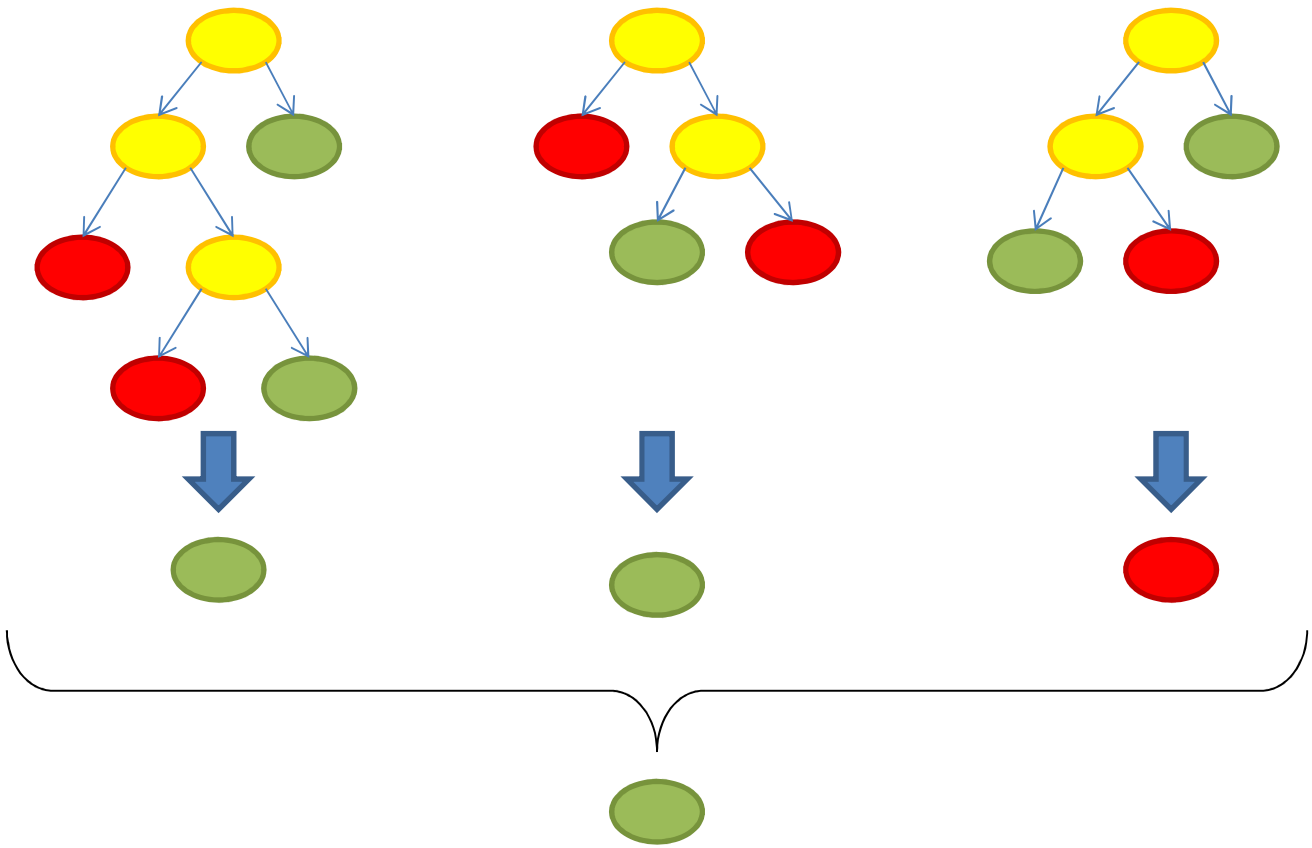
Extremely Randomized Trees

- Extra-trees : *K* couples feature/threshold are randomly selected as node split candidates. The best node split among these *K* candidates is chosen (added randomization)



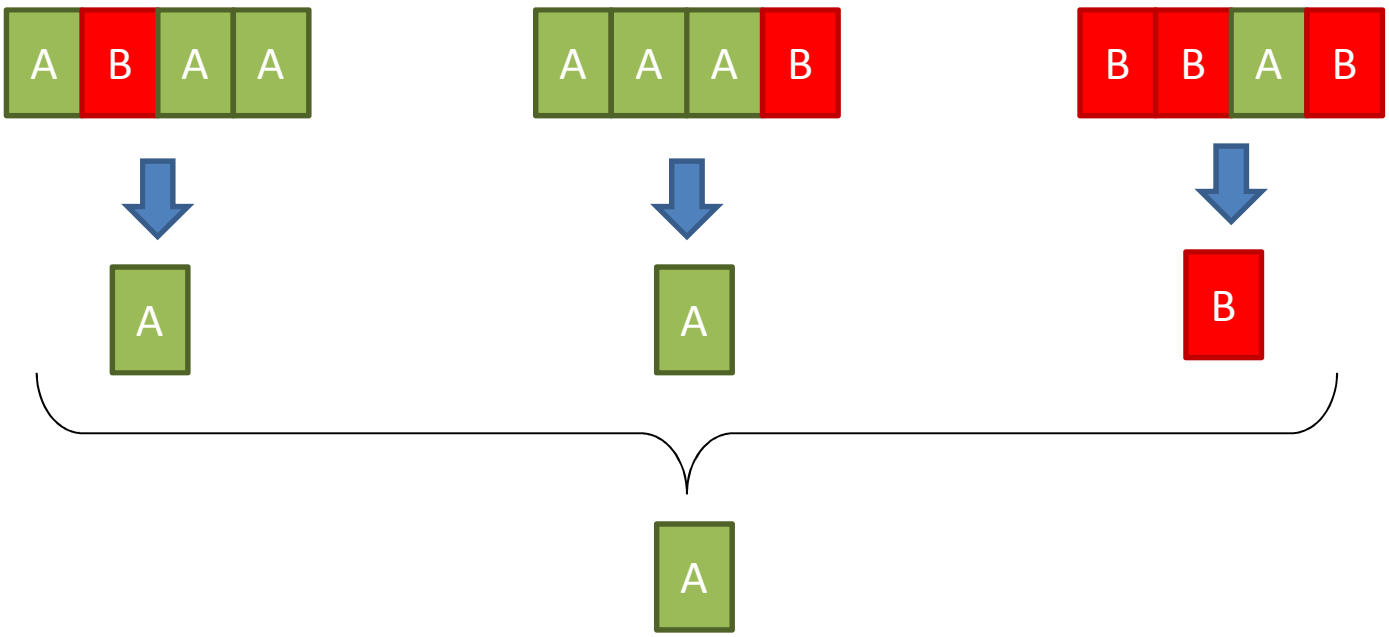
Extremely Randomized Trees

- *M* randomized trees are built, and their predictions are aggregated



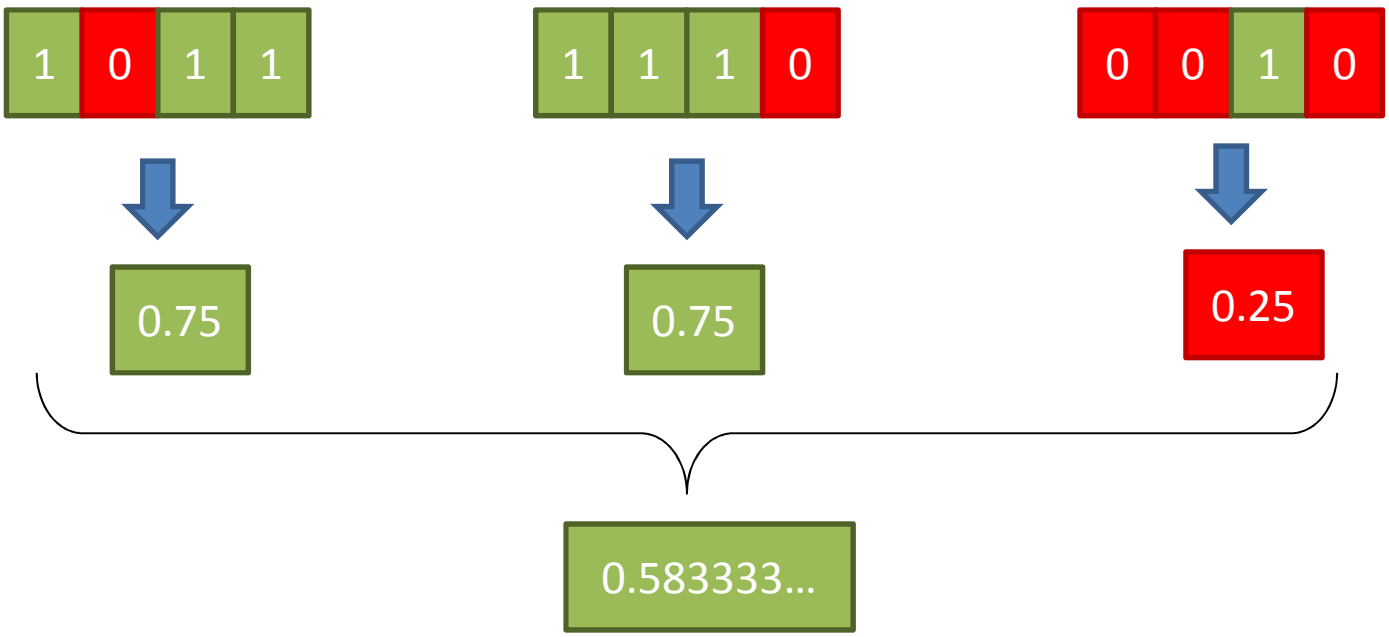
Classification vs. Regression

- **<u>Classification:</u>** Each tree outputs the majority class of the reached leaf. The majority class among all *M* predictions is the final result.



Classification vs. Regression

- **<u>Regression:</u>** Each tree outputs the mean value of the labels in the reached leaf. The mean value of all *M* predictions is the final result.



Extremely Randomized Trees

Parameters:

- $n_{\min}$: Number of objects for node split, pruning control
  - $n_{\min} = 2$ for classification (fully grown)
  - $n_{\min} = 5$ for regression
- K: Number of couples feature/threshold, output dependence
  - $K = \sqrt{\#\,Attributes}$ for classification
  - $K = \#Attributes$ for regression
- M: Number of trees, computational effort control
  - No default values. Accuracy vs Computational time

Extremely Randomized Trees

Advantages:

- Strongly reduces the variance with no significant increase in bias
- Very fast

Inconvenients:

- May require fine-tuning the parameters

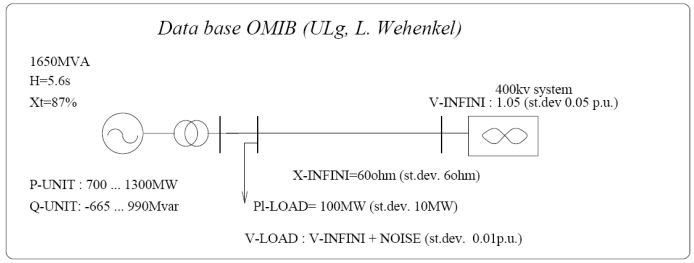
(although standard values are often optimal)

Outline

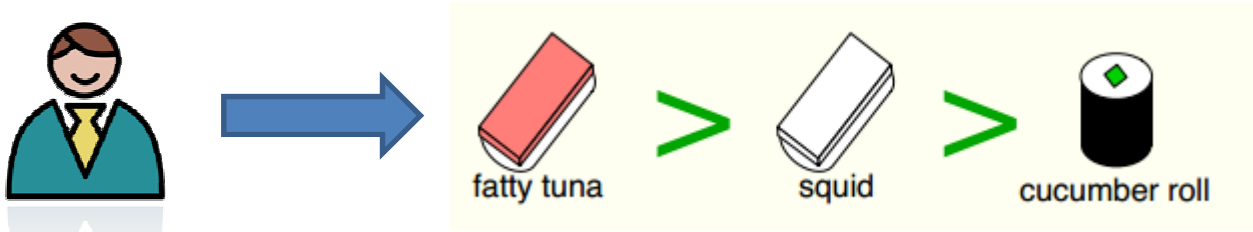
- 1st Part
  - Context and definitions
  - Problem setting
  - Contribution
- 2nd Part
  - Methods and algorithms
    - Subset selection methods (PR, EDA, EGS, RGS)
    - Evaluating a prediction and a subset of comparisons
    - Base learner : Extra-trees
  - Empirical validation
    - Databases
    - Results and parameters discussion
    - Complexity reduction
  - Conclusion

Used databases

- OMIB, synthetic, stability of power system



- Sushi, real life, user preference over sushi's



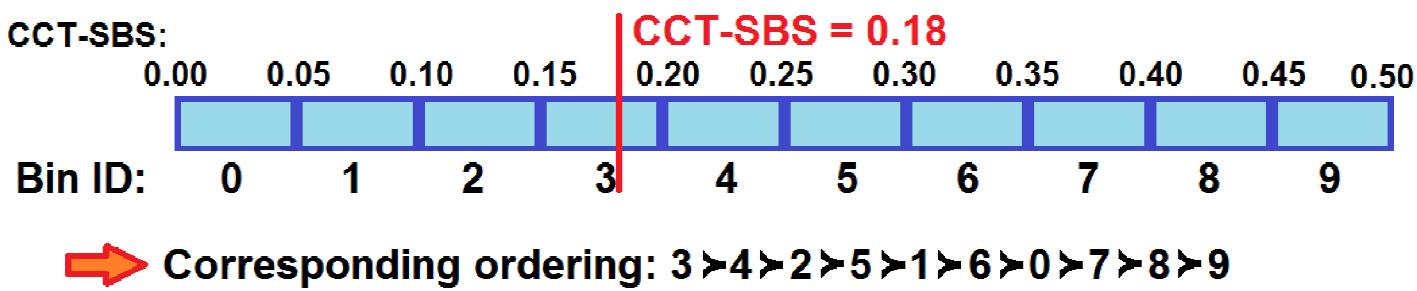
- MovieLens, real life, user preference over movies

OMIB

7 numerical attributes | 1 numerical output

20,000 objects

Object ID	PL	PU	QU	VINF	VL	XINF	CCT-SBS
1	-97.479	950.889	-6.4902	1.00208	0.998238	81.9069	0.1265
2	-90.052	704.46	888.322	1.04706	1.038	77.6292	0.3687
3	-68.203	1038.83	-550.47	1.10339	1.10283	51.0616	0.1792
…	…	…	…	…	…	…	…
20,000	-101.76	1073.68	700.649	1.07136	1.08313	46.2877	0.2632



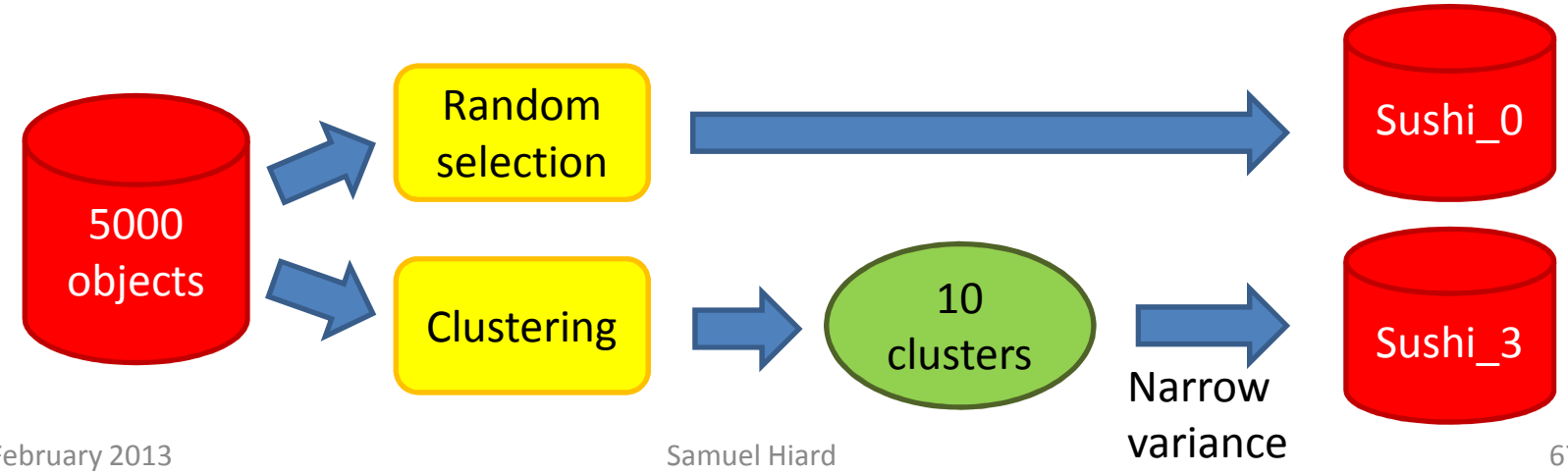
Corresponding ordering: 3 ≻ 4 ≻ 2 ≻ 5 ≻ 1 ≻ 6 ≻ 0 ≻ 7 ≻ 8 ≻ 9

Sushi

11 attributes ↔ 1 ranking (10 out of 100)

500 objects ↕

User ID	Gender	Age	Time ToFill	MP	MR	MEW	…	Ranking
8,270	M	1	219	13	3	E	…	10,20,13,9,16,23,30,17,18,91
8,053	M	3	259	24	5	E	…	43,47,9,27,10,66,5,18,21,42
6,823	F	3	249	30	7	W	…	15,2,19,16,55,46,95,32,3,18
…	…	…	…		…	…	…	…
8,756	M	3	316	13	3	E	…	8,1,76,0,2,5,32,4,35,80



MovieLens

5 attributes

1 numerical

100,000 ratings
943 objects

User ID	Age	Gender	Occupation	Zip code	Movie ID	Rating
1	24	M	0	85711	46	4
1	24	M	0	85711	132	2
2	53	F	1	94043	35	5
…	…	…	…	…	…	…
943	22	M	5	77841	1534	1

Input:
(User1,Movie16,3)
(User1,Movie235,5)
(User1,Movie527,2)
(User1,Movie988,3)

Corresponding ordering:
235 ≻ 16 ≻ 988 ≻ 527

(20 → 736) out of 1682

Sparsity issue

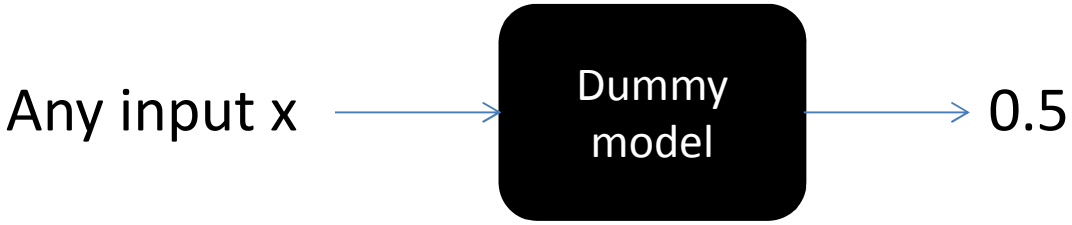
- The data is so sparse that some comparisons cannot be modeled, e.g.

Objects in the learning sample (x,y)
(x^1, a > c)
(x^2, b > c > d)
(x^3, a > d > c)

How do we infer the preference relation between « a » and « b »?

Sparsity issue

- Using a dummy model (UDM)



This comparator will thus provide half a point to both the labels that it compares

Sparsity issue

- Dropping empty comparison (DEC)

All objects in LS

a > b	a > c	a > d	a > e	b > c	b > d	b > e	c > d	c > e	d > e
true	true	false	-	true	false	-	false	-	-
false	false	-	-	false	-	-	-	-	-
-	-	-	-	-	true	true	-	-	true
true	-	-	-	-	-	-	-	-	-
true	true	true	-	true	false	-	false	-	-
false	true	true	-	true	true	-	true	-	-
…	…	…	…	…	…	…	…	…	…
15	**6**	**3**	**0**	**4**	**9**	**1**	**3**	**0**	**1**

Sparsity issue

- Dropping empty comparison (DEC)

All objects in LS

a > b	a > c	a > d	a > e	b > c	b > d	b > e	c > d	c > e	d > e
true	true	false	-	true	false	-	false	-	-
false	false	-	-	false	-	-	-	-	-
-	-	-	-	-	true	true	-	-	true
true	-	-	-	-	-	-	-	-	-
true	true	true	-	true	false	-	false	-	-
false	true	true	-	true	true	-	true	-	-
…	…	…	…	…	…	…	…	…	…
15	**6**	**3**	**0**	**4**	**9**	**1**	**3**	**0**	**1**

Sparsity issue

- Dropping empty comparison (DEC)

All objects in LS

a > b	a > c	a > d	a > e	b > c	b > d	b > e	c > d	c > e	d > e
true	true	false	-	true	false	-	false	-	-
false	false	-	-	false	-	-	-	-	-
-	-	-	-	-	true	true	-	-	true
true	-	-	-	-	-	-	-	-	-
true	true	true	-	true	false	-	false	-	-
false	true	true	-	true	true	-	true	-	-
…	…	…	…	…	…	…	…	…	…
15	**6**	**3**	**0**	**4**	**9**	**1**	**3**	**0**	**1**

Sparsity issue

- Dropping empty comparison (DEC)

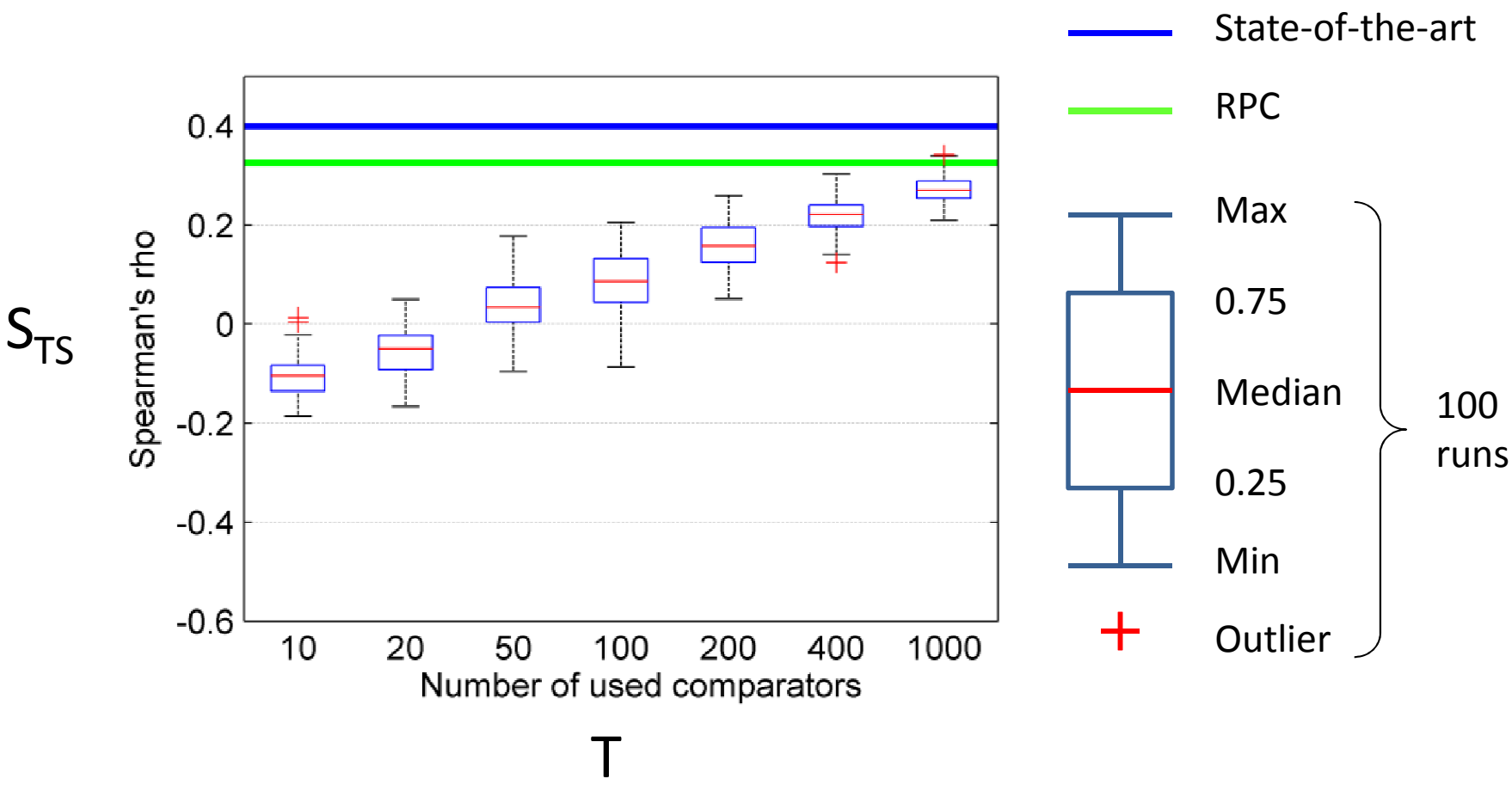
All objects in LS

a > b	a > c	a > d	a > e	b > c	b > d	b > e	c > d	c > e	d > e
true	true	false	-	true	false	-	false	-	-
false	false	-	-	false	-	-	-	-	-
-	-	-	-	-	true	true	-	-	true
true	-	-	-	-	-	-	-	-	-
true	true	true	-	true	false	-	false	-	-
false	true	true	-	true	true	-	true	-	-
…	…	…	…	…	…	…	…	…	…
15	**6**	**3**	**0**	**4**	**9**	**1**	**3**	**0**	**1**

Outline

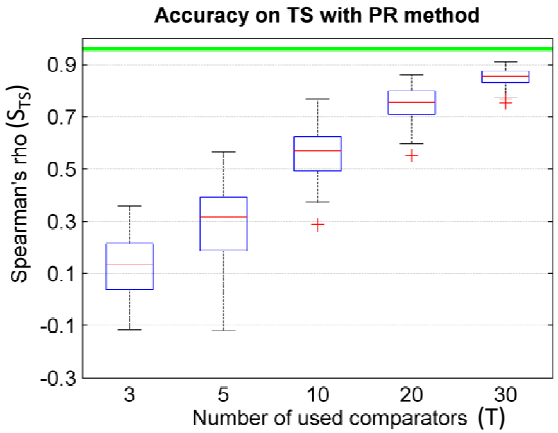
- 1st Part
  - Context and definitions
  - Problem setting
  - Contribution
- 2nd Part
  - Methods and algorithms
    - Subset selection methods (PR, EDA, EGS, RGS)
    - Evaluating a prediction and a subset of comparisons
    - Base learner : Extra-trees
  - Empirical validation
    - Databases
    - Results and parameters discussion
    - Complexity reduction
  - Conclusion

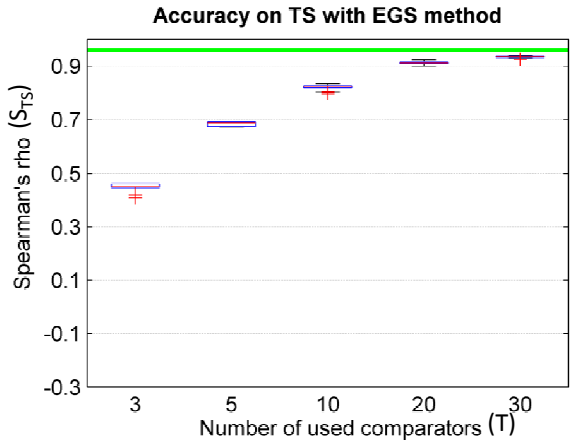
Accuracy figures



Accuracy on Omib (Classification)

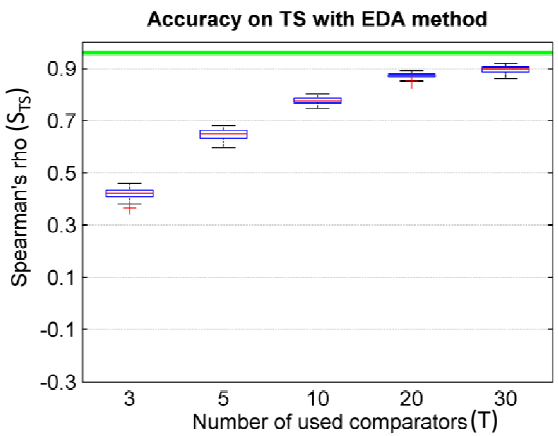
PR

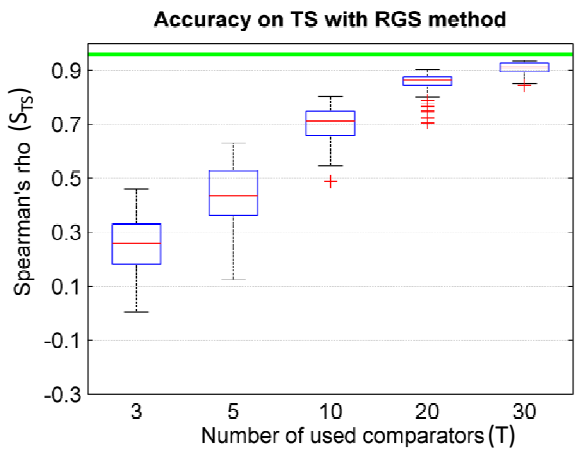




EGS

EDA



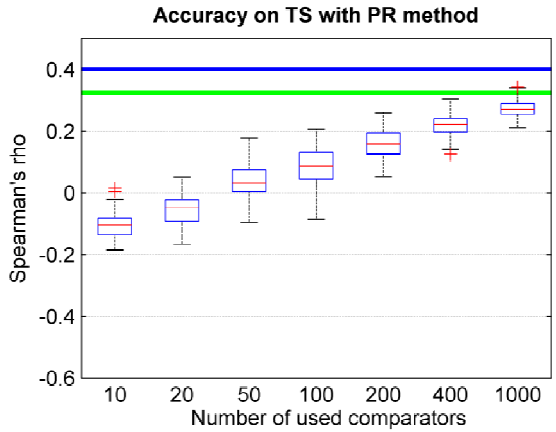


RGS

EGS and EDA are the most efficient

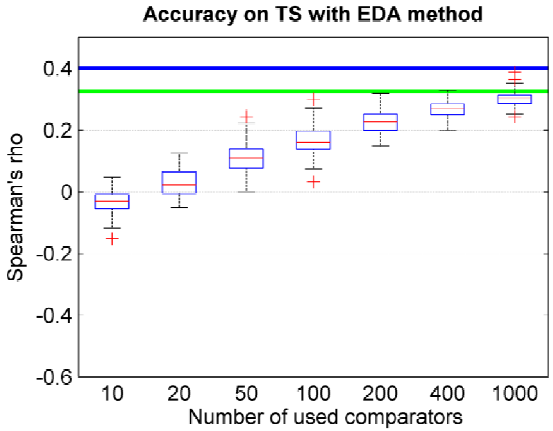
Accuracy on Sushi_0 (Classification, DEC)

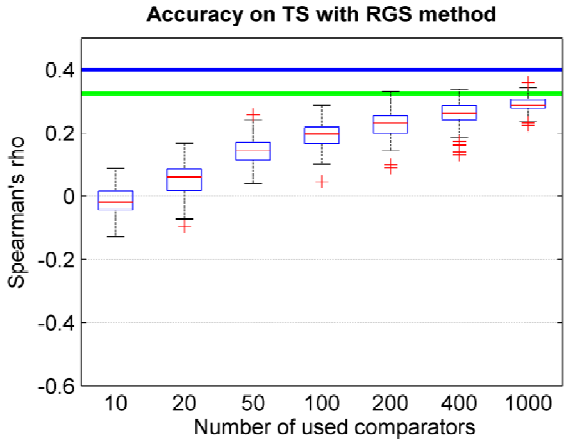
PR



EGS ?

EDA



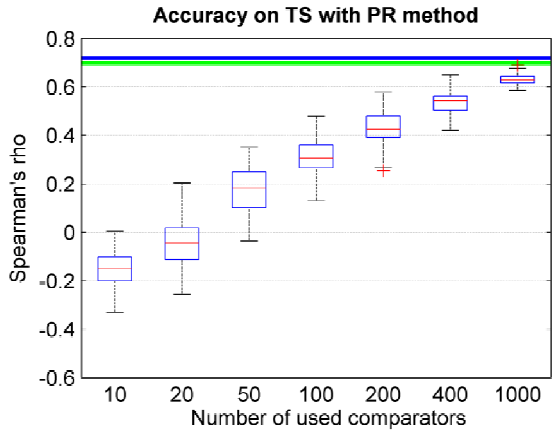


RGS

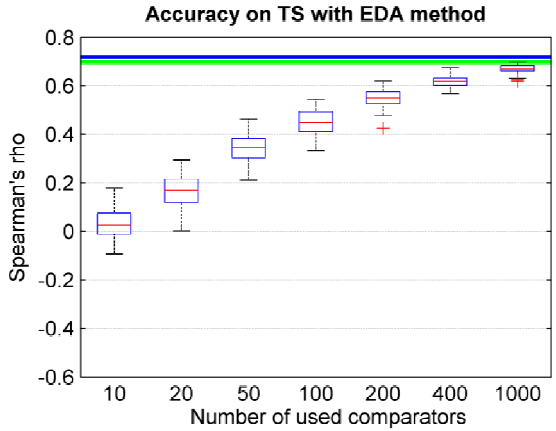
RGS is the most efficient

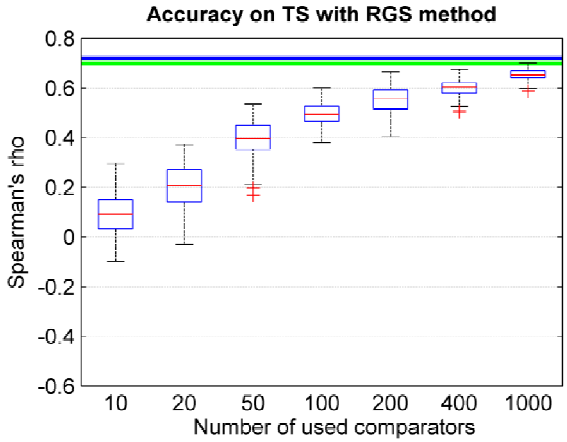
Accuracy on Sushi_3 (Classification, DEC)

PR



EDA



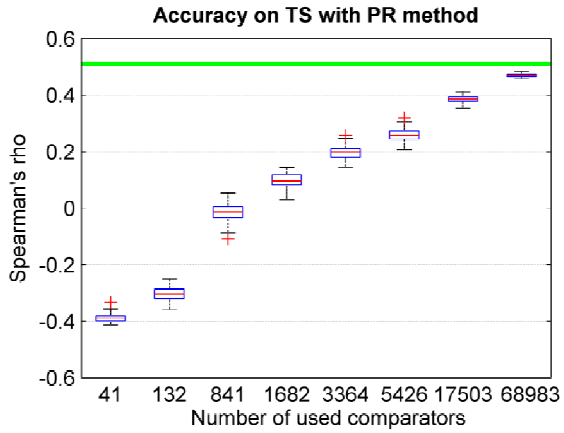


RGS

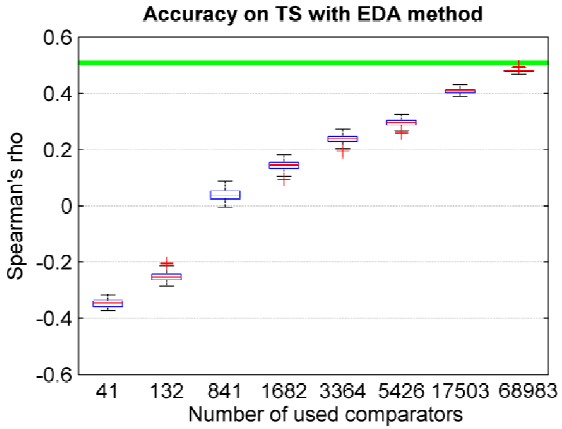
RGS is the most efficient

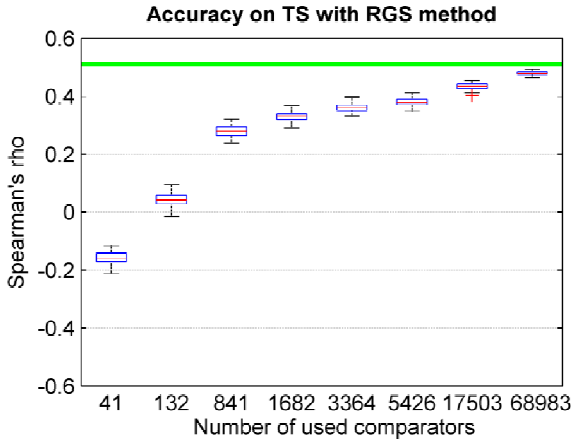
Accuracy on MovieLens (Classification, DEC)

PR



EDA

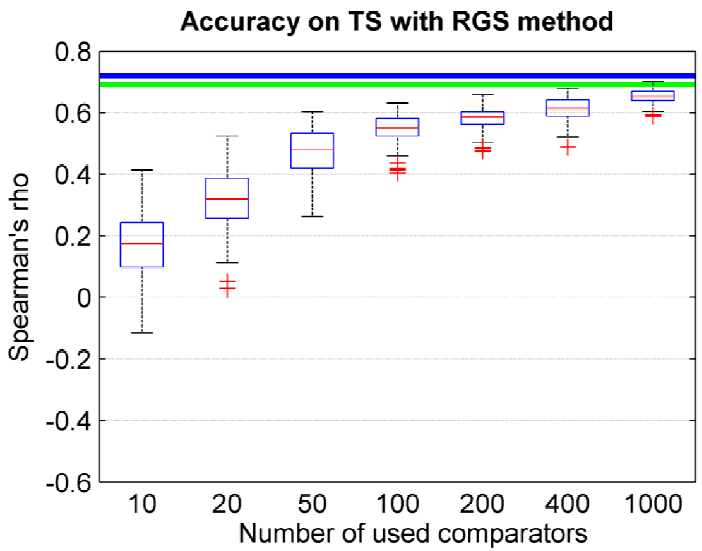




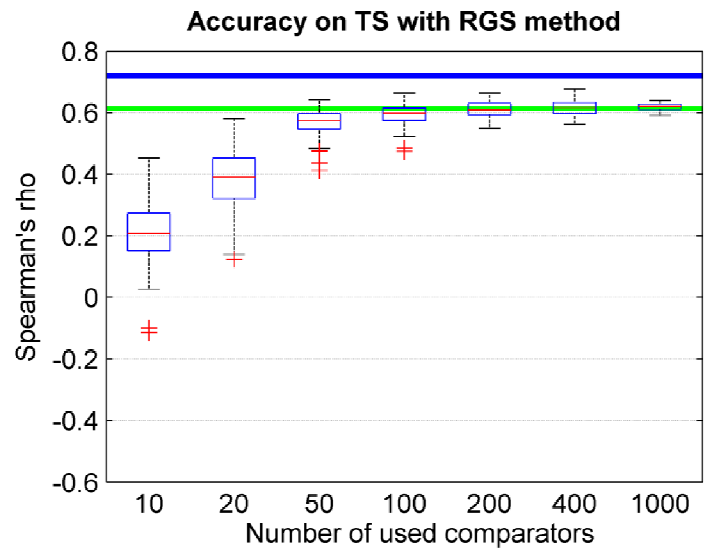
RGS

RGS is the most efficient

Pre-trimming the comparisons which have few examples



#examples = 1

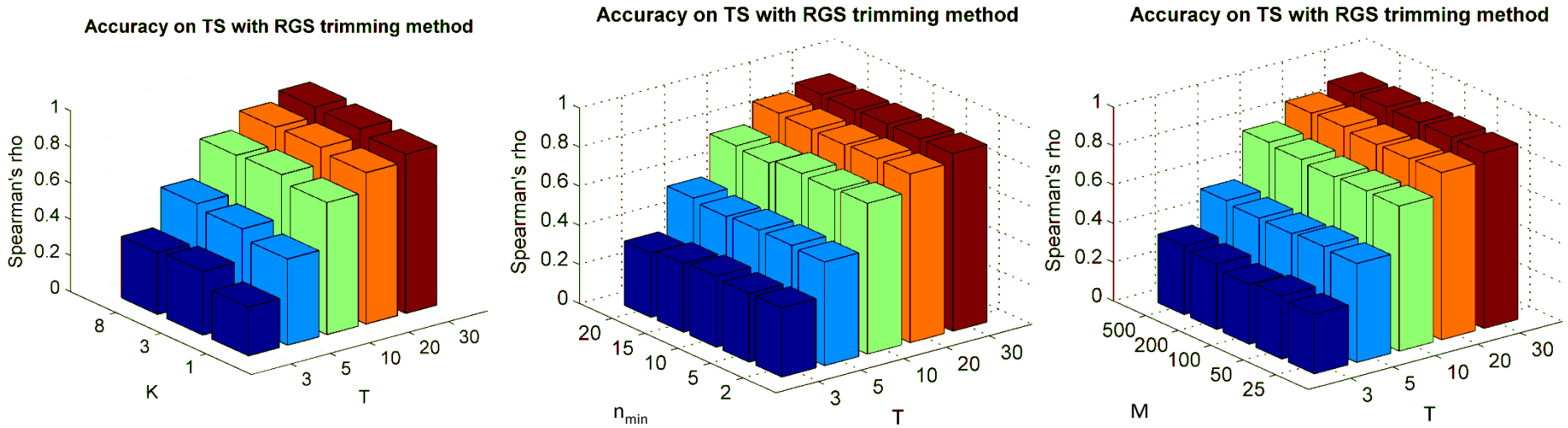


#examples = 5

(Sushi_3, RGS, Regression mode)

- RPC's performance is decreased
- RGS converges faster

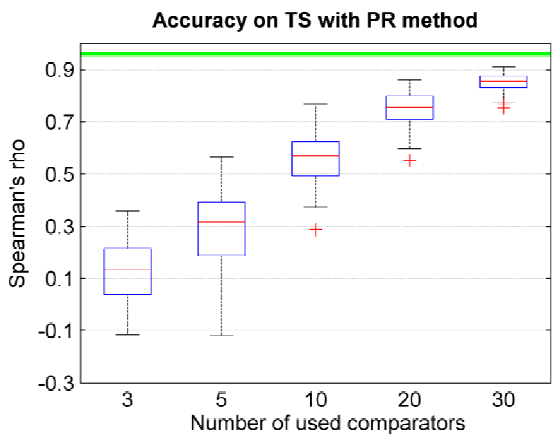
Robustness w.r.t. tree parameters

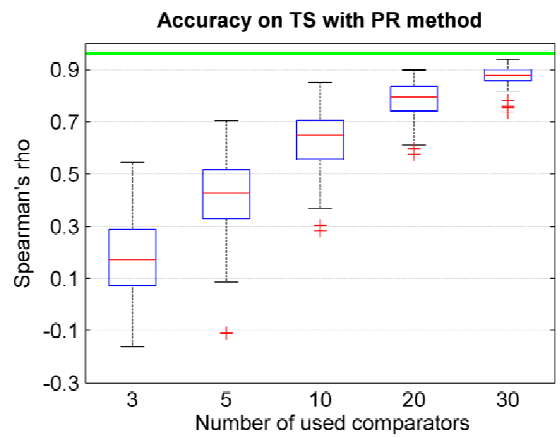


- Varying one parameter at a time and keeping the standard selection for the remaining ones (Omib, RGS, Classification mode)
- The tree parameters have slim to none influence on the outcome

Accuracy on Omib (Clas vs. Regr)

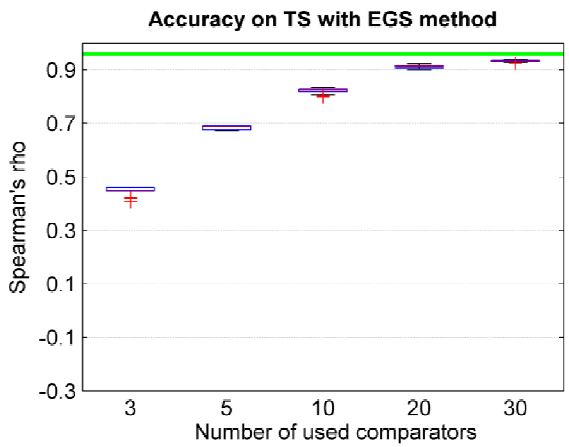
PR
Clas

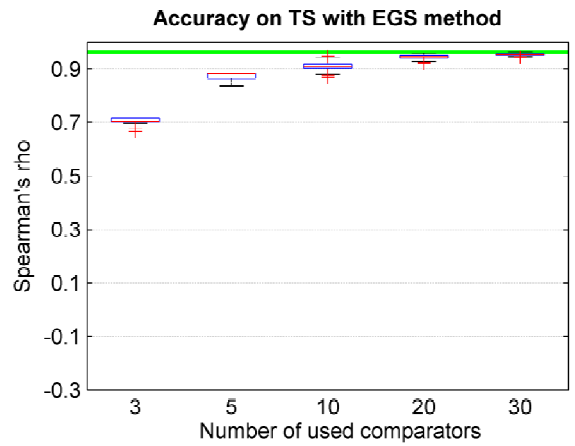




PR
Regr

EGS
Clas





EGS
Regr

Regression improves the predictions

Computing the mean order

For each given ranking, we attribute

$$\frac{(p-i+1)(N+1)}{(p+1)}$$ votes to each label on rank $i \in \{1 \ldots p\}$, and

$$\frac{(N+1)}{2}$$ votes to each missing label

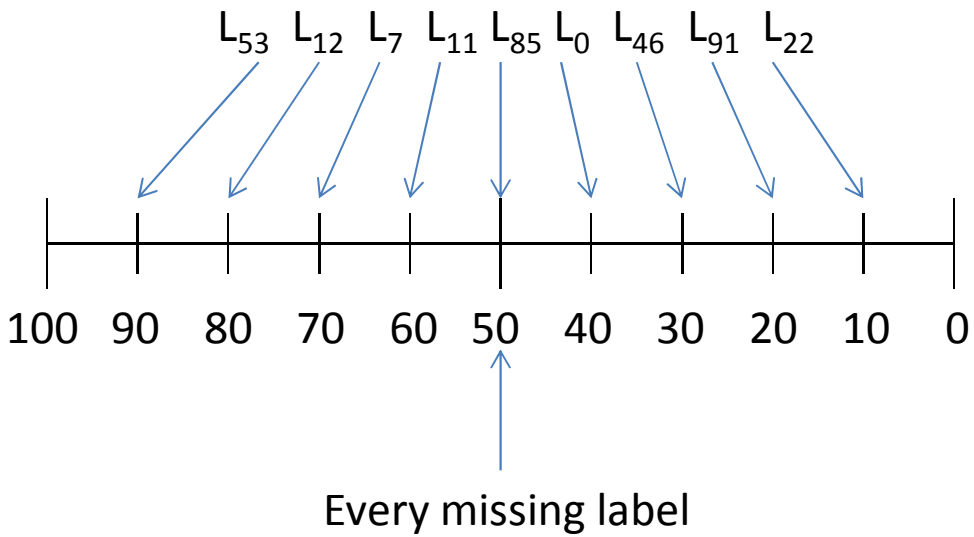
(W. Cheng and E. Hüllermeier, *A new instance-based label ranking approach using the mallows model*, 2009)

Computing the mean order

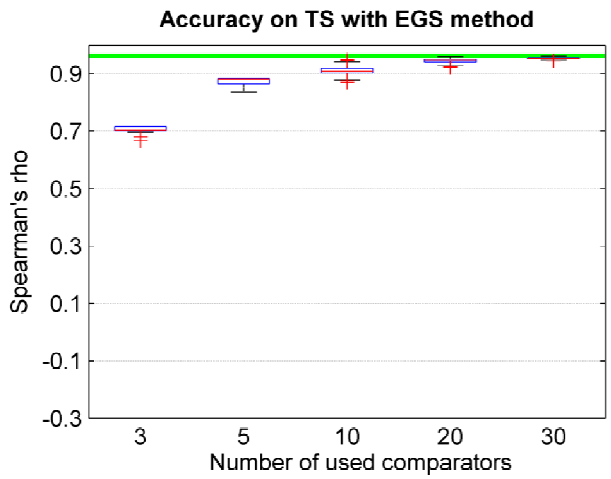
Example: N = 99

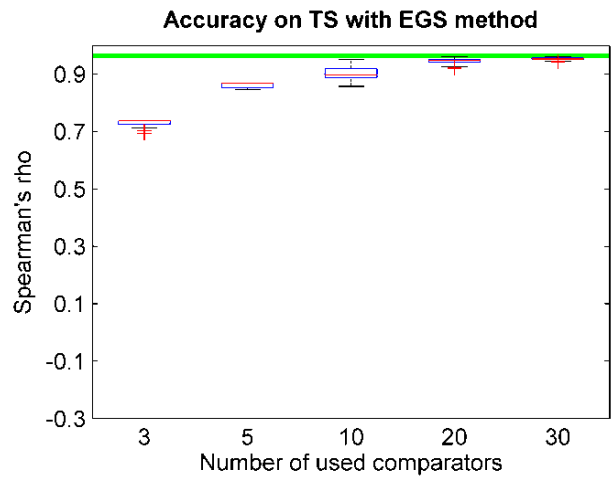
Ranking = $L_{53} > L_{12} > L_7 > L_{11} > L_{85} > L_0 > L_{46} > L_{91} > L_{22}$ (9 labels)

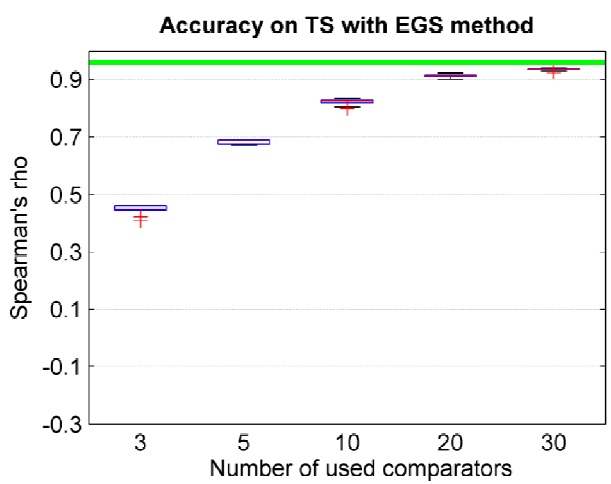
Label	Votes
L_{53}	90
L_{12}	80
L_7	70
L_{11}	60
L_{85}	50
L_0	40
L_{46}	30
L_{91}	20
L_{22}	10
All other	50

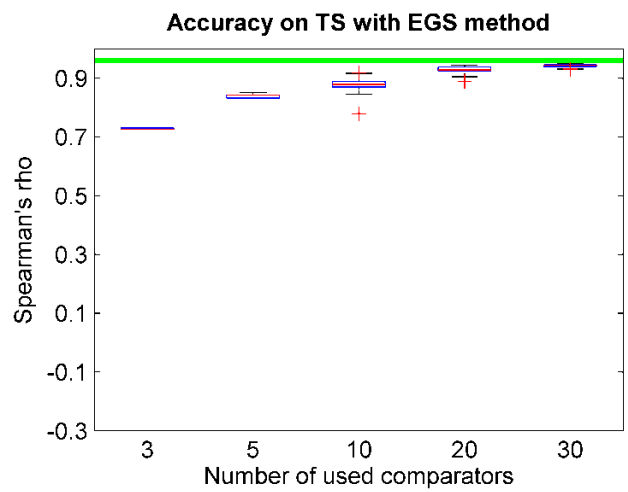


- Repeat this process for each object in LS
- Sort labels according to total score









Using the mean order to break ties (Omib)

Regr

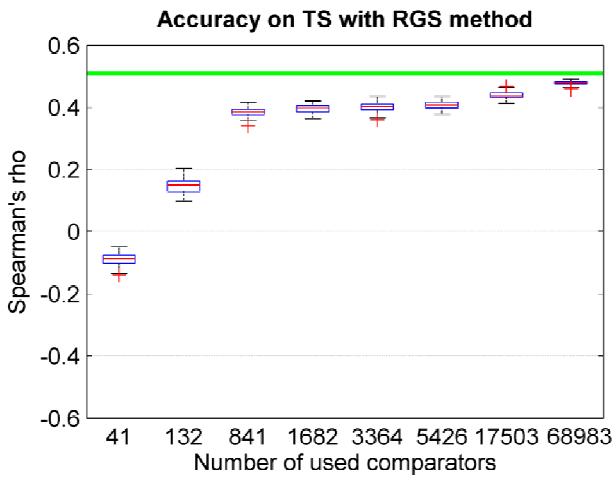
Clas

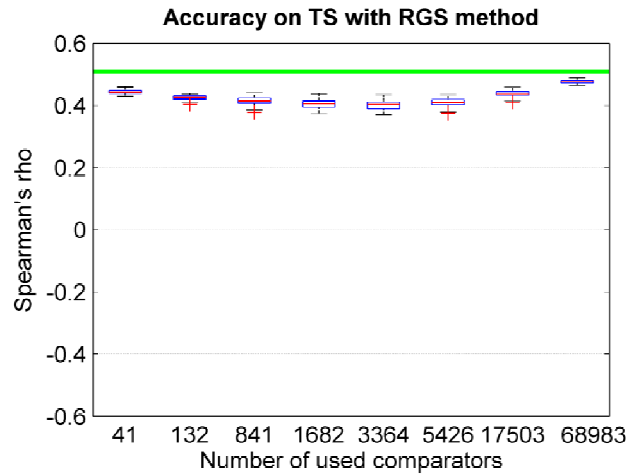
Reverse alphanum

Mean order

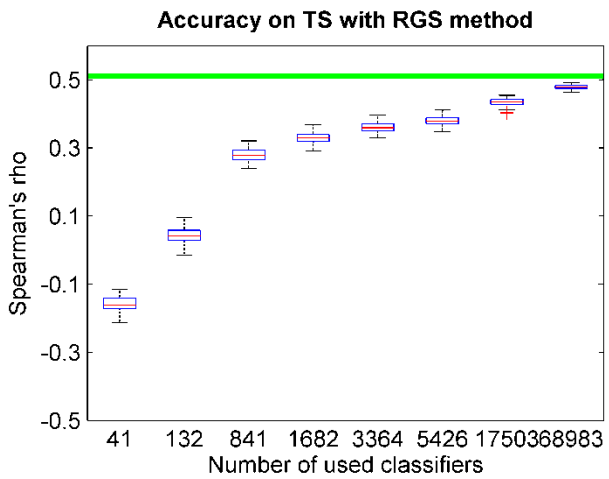
Using the mean order to break ties (MovieLens)

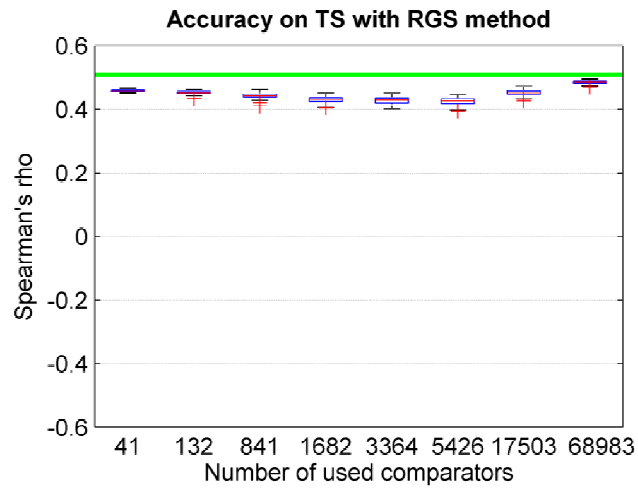
Regr





Clas





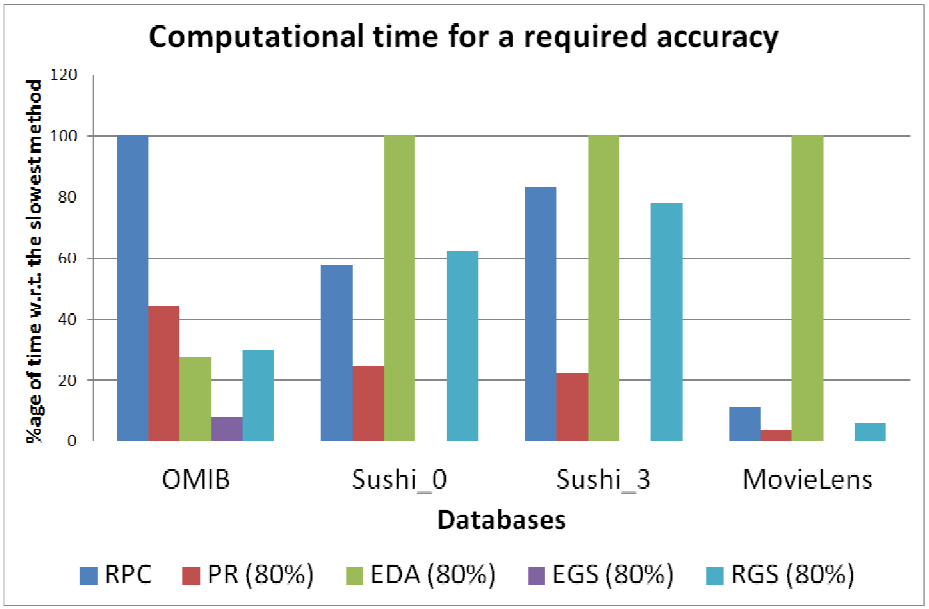
Reverse alphanum

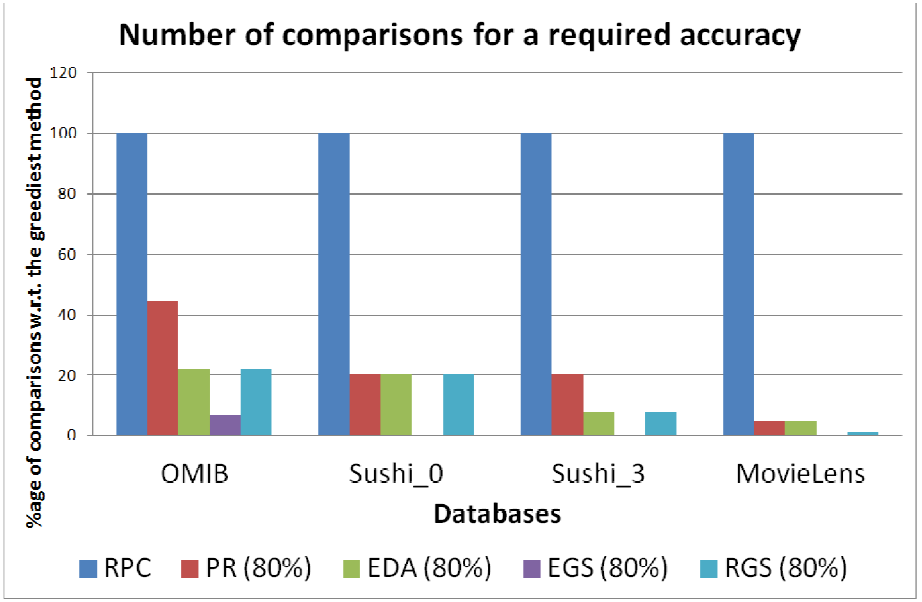
Mean order

Outline

- 1st Part
  - Context and definitions
  - Problem setting
  - Contribution
- 2nd Part
  - Methods and algorithms
    - Subset selection methods (PR, EDA, EGS, RGS)
    - Evaluating a prediction and a subset of comparisons
    - Base learner : Extra-trees
  - Empirical validation
    - Databases
    - Results and parameters discussion
    - Complexity reduction
  - Conclusion

Complexity gain





- RGS: Subset selection time << model building time
  → Computational time gain

Summary

- Definitions:
  - Artificial Intelligence
  - Supervised Learning
  - Label Ranking
- Ranking by Pairwise Comparison:
  - Algorithm
  - Evaluation
  - Complexity issue
- Subset selection algorithms:
  - Description
  - Evaluation during optimization proxy
  - Model for base learner
- Validation
  - Databases and sparsity issue
  - Accuracy
  - Complexity

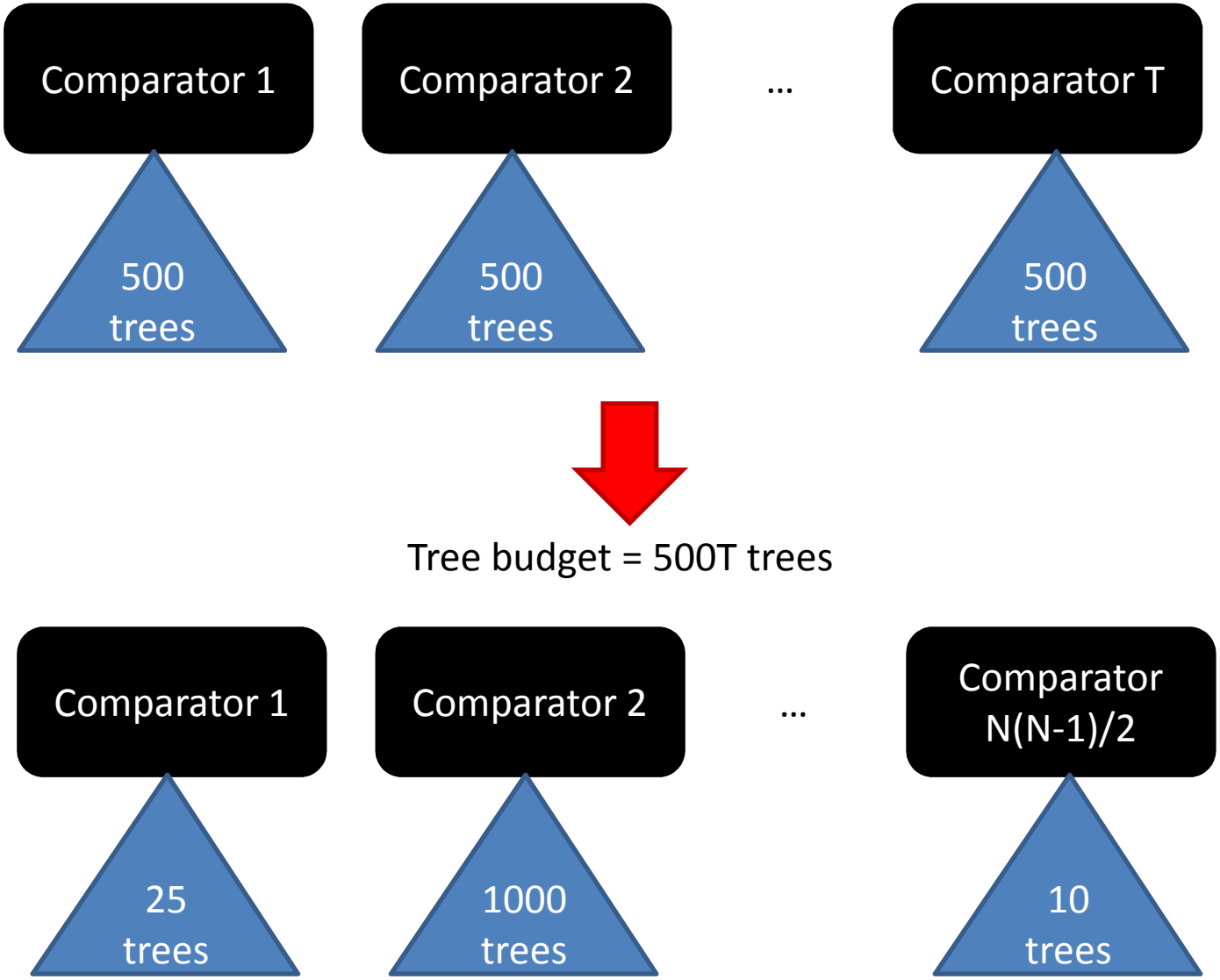
Conclusion

- The RPC algorithm has a complexity burden
- We proposed to reduce the set of $N(N-1)/2$ comparisons to a subset of size $T \approx O(N)$
- We designed several algorithms which aim at finding the optimal subset for a given T, prior to building a SL model, RGS being so far the best alternative
- We also proposed an *a priori* reduction based on the number of objects in the comparator's LS which improves the performance of RGS
- We thus reduced the complexity of RPC without significantly dropping the accuracy, making this approach applicable to problems with a large number N of labels

Future work

- Prior knowledge (ex: Sushi)
- Controlling partial ranking and noise (OMIB)
  - Feature noise
  - Output noise (partial ranking, label displacement)
- Algorithmic improvements
  - Optimizing with a given budget
  - Using the mean order to select *Q* (subset of *T* comparisons)
  - Weighted distribution based on LS

Optimizing with a given budget



Using the mean order to select *Q*

- Compute mean order
  - e.g. 5 > 2 > 1 > 6 > 4 > 3
- Compare labels at position i and $i+1$
  - 5 > 2
  - 2 > 1
  - 1 > 6
  - 6 > 4
  - 4 > 3
- Better than PR?
  - Preliminary results (Sushi_3):
    - Classification mode : Always worse than PR (especially when *T* is large)
    - Regression mode : Always better than PR. Better than RGS if T ≤ 20

  → The top ranked items seems to contain usefull information

  → Room for improvement

Weighted distribution based on LS

All objects in LS

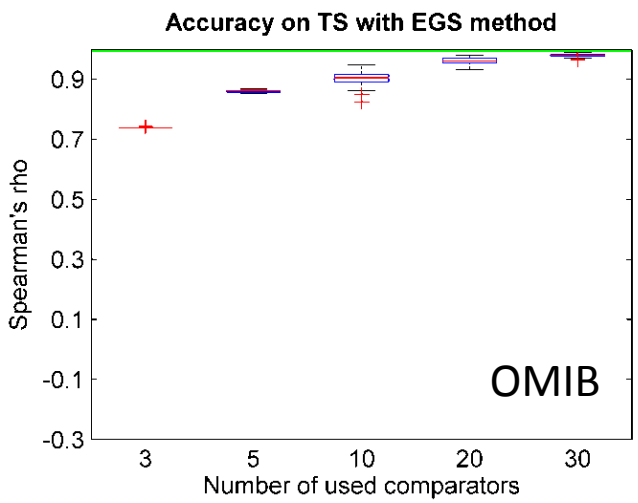
a > b	a > c	a > d	a > e	b > c	b > d	b > e	c > d	c > e	d > e
true	true	false	-	true	false	-	false	-	-
false	false	-	-	false	-	-	-	-	-
-	-	-	-	-	true	true	-	-	true
true	-	-	-	-	-	-	-	-	-
true	true	true	-	true	false	-	false	-	-
false	true	true	-	true	true	-	true	-	-
…	…	…	…	…	…	…	…	…	…

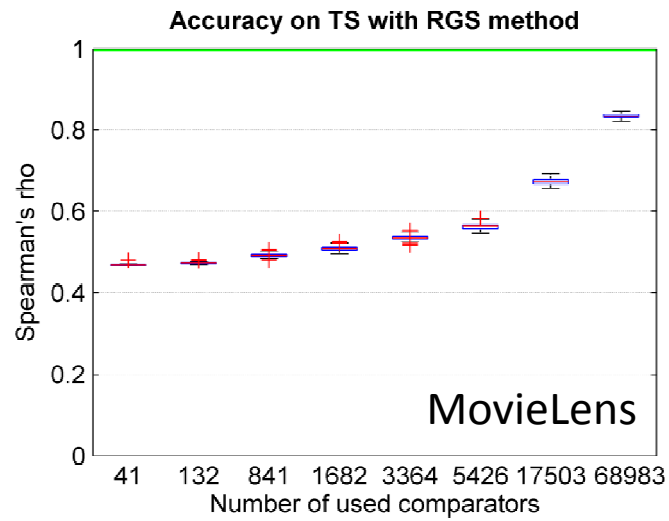
#obj	**15**	**6**	**3**	**0**	**4**	**9**	**1**	**3**	**0**	**1**
P(q)	**0.37**	**0.14**	**0.07**	**0**	**0.10**	**0.21**	**0.02**	**0.07**	**0**	**0.02**

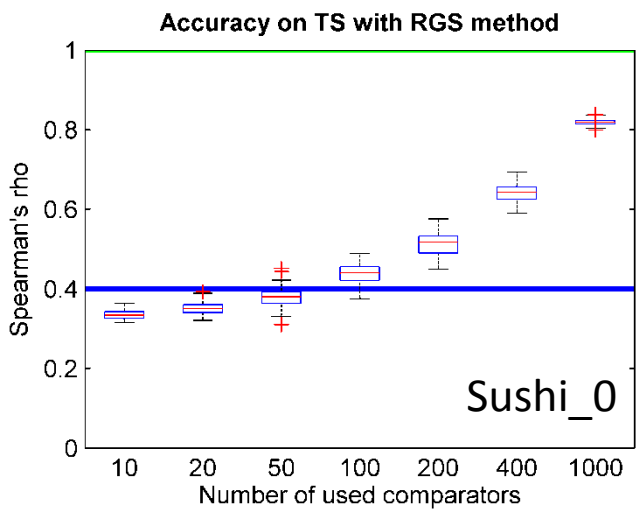
Thank you for your attention

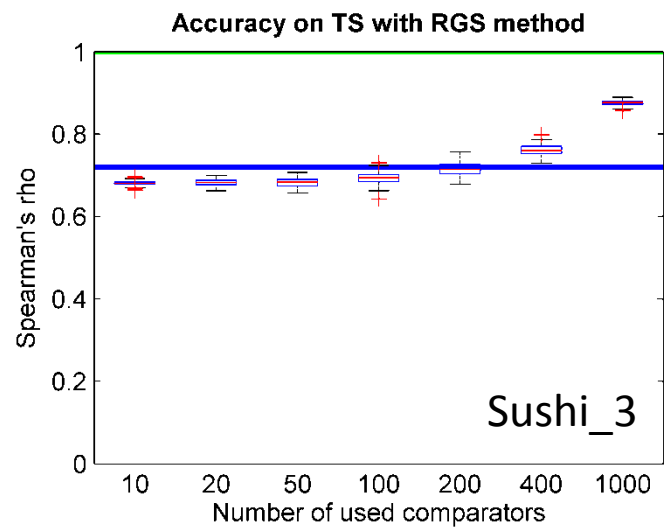
Thank you for your attention

Using perfect models









MovieLens : imposing alphanum order

- Votes → Ranking
- Partial ranking (e.g. a>b and c>d)
  - Training the pairwise comparators? OK
  - Spearman evaluation? KO (ground truth unknown)
- Solutions:
  - Evaluating with Kendall Tau (state-of-the-art?)
  - Output = ordered subset of labels

  → Arbitrarily breaking ties