Università degli Studi di Milano
Master Degree in Computer Science

Information Management course

Teacher: Alberto Ceselli

Lectures 11 and 12: 11/11/2014

L. C. Molina, L. Belanche, A. Nebot
“Feature Selection Algorithms: A Survey and Experimental Evaluation”, IEEE ICDM (2002)

and

L. Belanche, F. Gonzales “Review and Evaluation of Feature Selection Algorithms in Synthetic Problems”, arXiv – available online (2011)

Feature Selection Algorithms

- Introduction
- Relevance of a feature
- Algorithms
- Description of fundamental FSAs
- Empirical evaluation
- Experimental evaluation

Introduction

- The Feature selection problem:
  - Given a set of candidate features, select a subset defined by one of the following approaches:
    - Having a fixed size and maximizing an evaluation measure;
    - Of smaller size that satisfies a constraint on an evaluation measure
    - Best tradeoff between size and evaluation measure

- FSA are motivated by a definition of *relevance* (not obvious)

- FSAs can be classified according to their output
  1) Giving a weighted linear order of features
  2) Giving a subset of original features (the one we focus on)
  - N.B. (2) is (1) with binary weighting

Feature Selection Algorithms

- Introduction
- Relevance of a feature
- Algorithms
- Description of fundamental FSAs
- Empirical evaluation
- Experimental evaluation

Relevance with respect to an objective

- Relevance must be defined with respect to an objective: assuming the objective is classification and the set of features is X:
  - A feature $x \in X$ is relevant to an objective c() if there exist two examples A and B that
    - differ only in the value of x
    - $c(A) \neq c(B)$
  - i.e. there are two elements that can be classified correctly only by looking at x
- However, our datasets are samples in the feature space:
  - A feature $x \in X$ is strongly relevant to the sample S to an objective c() if there exist two elements A and B of S that
    - differ only in the value of x
    - $c(A) \neq c(B)$
  - A feature x is weakly relevant if there exists a $X' \subset X$ with $x \in X'$, where x is strongly relevant with respect to S

Relevance as a complexity measure

- Idea: given a data sample S and an objective c(), define r(S,c) as the smallest number of relevant features to c() such that the error in S is the least possible for the inducer
- i.e. the smallest number of features required by a specific inducer to reach optimum performance in modeling c() using S
- Examples of such complexity measures:
  - Incremental usefulness:
    after choosing X', x is useful if the accuracy of c() computation is higher on x U X' than on X'
  - Entropic relevance:
    compute the amount of (Shannon) entropy in the dataset before and after the removal of a feature

Feature Selection Algorithms

- Introduction
- Relevance of a feature
- Algorithms
- Description of fundamental FSAs
- Generating weighted feature orders
- Empirical and experimental evaluation

Algorithms for Feature Selection

- A FSA can be seen as a “computational approach to a definition of relevance”
  - Let X be the original set of features, $|X| = n$
  - Let $J(X')$ be an evaluation measure to be optimized:

    $J: X' \subseteq X \to \mathbb{R}$

  (1) Set $|X'| = m < n$; find $X' \subset X$ such that $J(X')$ is maximum

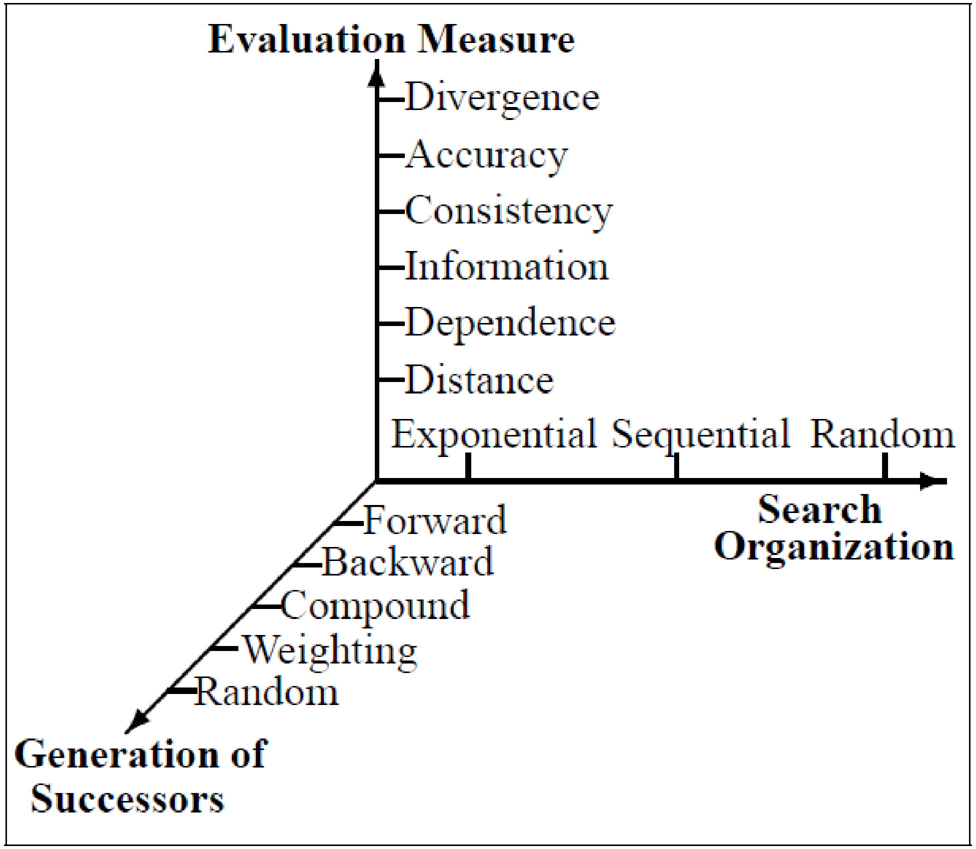
  (2) Set a value J_0; find $X' \subset X$ such that $|X'|$ is minimum, and $J(X') \geq J_0$

  - Find a compromise between (1) and (2)
- Remark: an optimal subset of features in not necessarily unique
- Characterization of FSAs
  - Search organization
  - Generation of successors
  - Evaluation measure

Characterization of a FSA

Each algo can be represented as a triple <Org, GS, J>

- Org: search organization
- GS: Generation of Successors
- J: Evaluation measure



Characterization of FSAs search organization

- General strategy with which the space of hypothesis is explored
- Search space: all possible subsets of features
- A partial order in the search space can be defined, as $S1 \prec S2$ if $S1 \subset S2$
- Aim of search: explore only a part of all subsets of features → for each subset relevance should be upper and lower bounded (estimates or heuristics)
  - Let L be a (labeled) list of (weighted) subsets of features → *states*
  - L maintains the current list of (partial) solutions, and the labels indicate the corresponding evaluation measure

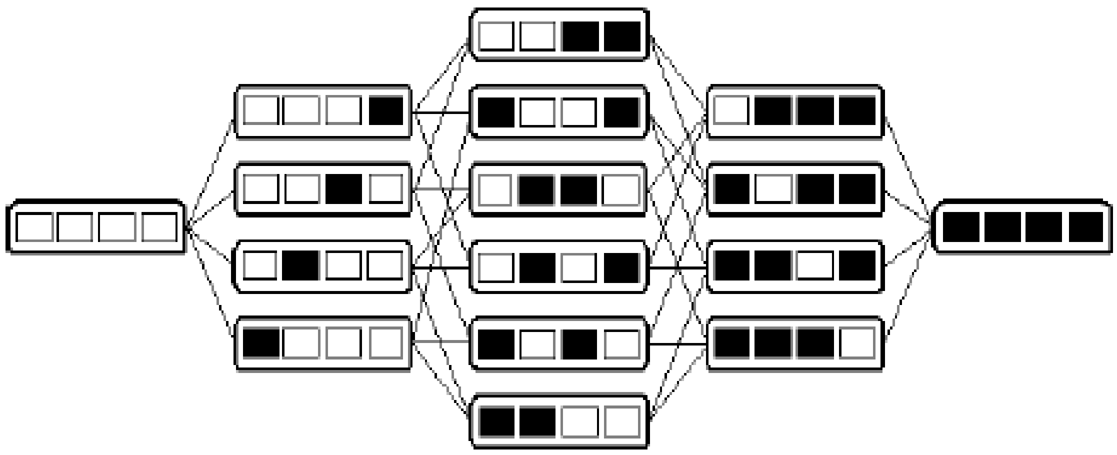


Figure 1. States in the binary search space involving 4 features. A black square represents the inclusion of a feature in the state and a white square represents its exclusion.

Characterization of FSAs search organization

We consider three types of search:

- Exponential search (|L| > 1):
  - Search cost $O(2^n)$
  - Extreme case: exhaustive search
  - If given S1 and S2 with S1 ⊆ S2 then J(S1) ≤ J(S2)
    → then J() is monotonic and branch-and-bound is optimal!
  - A* with heuristics is another option
- Sequential search (|L| = 1):
  - Start with a certain state and select a certain successor
  - Never backtrack
  - Search cost is polynomial, but no optimality guarantee
- Random search (|L| > 1):
  - Pick a state and change it somehow (local search)
  - Escape from local minima with random (worsening) moves

Characterization of FSAs generation of successors

Five operators can be used to move from a state to the next

- Forward: start with X' = empty set
  - Given a state X', pick a feature x ∉ X' such that J(X' U {x}) is largest
  - Stop when J(X' U {x}) = J(X'), or |X'| = certain card., or …
- Backward: start with X' = X
  - Given a state X', pick a feature x ∊ X such that J(X' \ {x}) is largest
  - Stop when J(X' \ {x}) = J(X'), or |X'| = certain card., or …
- Generalized Forward and Backward: consider <u>sets</u> of features for addition / removal at each step
- Compound: perform f consecutive forward moves and b consecutive backward moves
- Random

Characterization of FSAs evaluation measures

- Several problem dependent approaches
- What counts is the relative values assigned to different subsets: e.g. classification
  - Probability of error: what's the behavior of a classifier using the subset of features?
  - Divergence: probabilistic distance among the class-conditional probability densities
  - Dependence: covariance or correlation coefficients
  - Interclass distance: e.g. dissimilarity
  - Information or Uncertainty: exploit entropy measurements on single features
  - Consistency: an inconsistency in X' and S is defined as two instances in S that are equal when considering only the features in X', but actually belong to different classes (aim: find the minimum subset of features leading to zero inconsistencies)

Characterization of FSAs evaluation measures

- Example: Consistency
  - an inconsistency in X' and S is defined as two instances in S that are equal when considering only the features in X', but actually belong to different classes (aim: find the minimum subset of features leading to zero inconsistencies)

    $IC_{X'}(A) = X'(A) – \max_k X'_k(A)$

    X'(A) = number of instances of S equal to A when only the features in X' are considered

    $X'_k(A)$ = number of instances of S <u>of class k</u> equal to A when only the features in X' are considered
  - Inconsistency rate:

    $IR(X') = \sum_{A \in S} IC_{X'}(A) / |S|$
  - J(X') = 1 / (IR(X') + 1)
- N.B. IR is a monotonic measure

General schemes for feature selection

- Main forms of relation between FSA and “inducer”
  - Embedded scheme: the external method has its own FSA (e.g. decision trees or ANN)
  - Filter scheme: the feature selection takes place before the induction step
  - Wrapper scheme: FSA uses subalgorithms (e.g. learning algorithms) as internal routines

General algorithm for feature selection

```
Input:
    S  – data sample with features X, |X| = n
    J  – evaluation measure to be maximized
    GS – successor generation operator
Output:
    Solution – (weighed) feature subset

L := Start_Point(X);
Solution := {best of L according to J};
repeat
    L := Search_Strategy(L, GS(J), X);
    X' := {best of L according to J};
    if J(X') ≥ J(Solution) or (J(X') = J(Solution)
          and |X'| < |Solution|)
    then Solution := X';
until Stop(J, L)
```

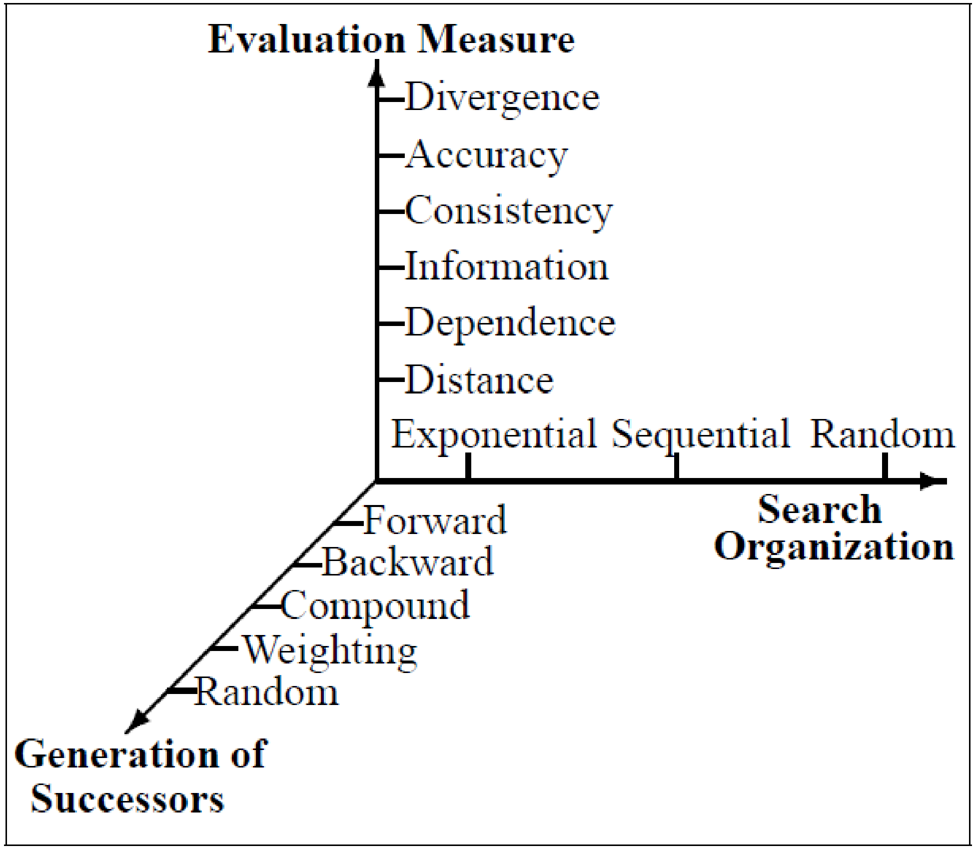
Feature Selection Algorithms

- Introduction
- Relevance of a feature
- Algorithms
- Description of fundamental FSAs
- Generating weighted feature orders
- Empirical and experimental evaluation

Characterization of a FSA

Each algo can be represented as a triple <Org, GS, J>

- Org: search organization
- GS: Generation of Successors
- J: Evaluation measure



Las Vegas Filter (LVF) <random, random, any>

```
Input:
   max – the maximum number of iterations
   J – evaluation measure
   S(X) – a sample S described by X, |X| = n
Output:
   L – all equivalent solutions found

L := []            // L stores equally good sets
Best := X          // Initialize best solution
J0 := J(S(X))      // minimum allowed value of J
repeat max times
    X' := Random_SubSet(Best)   // |X'| ≤ |Best|
    if J(S(X')) ≥ J0 then
        if |X'| < |Best| then
            Best := X'
            L := [X']             // L is reinitialized
        else if |X'| = |Best| then
                L := append(L, X')
             end
        end
    end
end
```

Las Vegas Incremental (LVI) <random, random, consist.>

```
Input:
  max – the maximum number of iterations
  J – evaluation measure
  S(X) – a sample S described by X, |X| = n
  p – initial percentage
Output:
  X' – solution found

S0 := portion(S, p)   // Initial portion
S1 := S \ S0          // Test set
J0 := J(S(X))         // Minimum allowed value of J
repeat forever
    X' := LVF (max, J, S0(X))
    if J(S(X')) ≥ J0 then stop
    else
        C := {elements in S1 with low
              contribution to J using X'}
        S0 := S0 ∪ C
        S1 := S1 \ C
    end
end
```

Rule of thumb: $p = 10\%$

SBG/SFG <sequential, F/B, any>

Input:
$S(X)$ – a sample S described by $X, |X| = n$
J – evaluation measure
Output:
X' – solution found

$X' := \emptyset$ *// forward*
$X' := X$ *// backward*
repeat
$x' := argmax\{J(S(X' \cup \{x\})) \,|\, x \in X \setminus X'\}$ *// forward*
$x' := argmax\{J(S(X' \setminus \{x\})) \,|\, x \in X'\}$ *// backward*
$X' := X' \cup \{x'\}$ *// forward*
$X' := X' \setminus \{x'\}$ *// backward*
until no improvement in J in last j steps
or $X' = X$ *// forward*
or $X' = \emptyset$ *// backward*

Focus <exponential, forward, consist.>

```
Input:
  S(X) – a sample S described by X, |X| = n
  J – evaluation measure (consistency)
  J0 – minimum allowed value of J
Output:
  X' – solution found

for i ∈ [1..n] do
    for each X' ⊂ X , with |X'| = i do
        if J(S(X')) ≥ J0 then stop
    end
end
```

Sequential Floating FS <exponential, F+B, consist.>

Input:

$S(X)$ – a sample S described by $X, |X| = n$

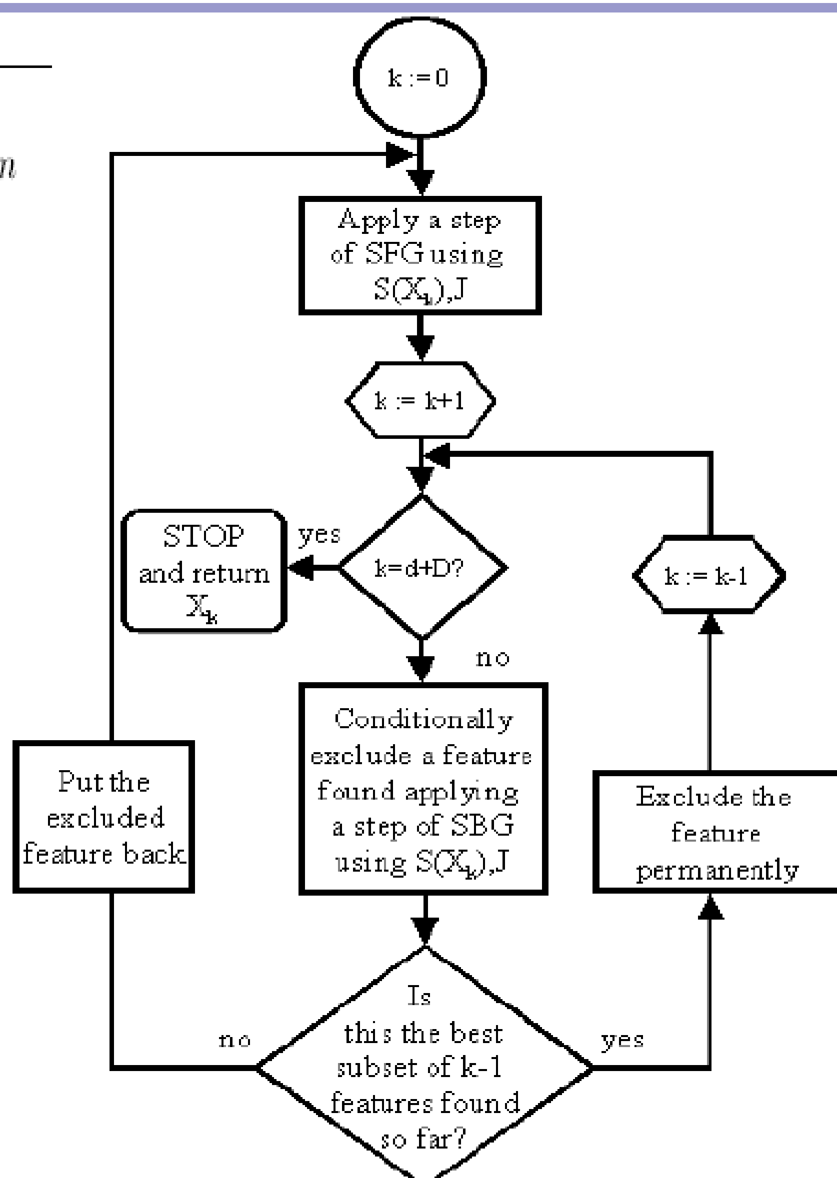
J – evaluation measure

d – desired size of the solution

D – maximum deviation allowed with respect to d

Output:

solution of size $d \pm D$



(Auto) branch&bound <exponential,backward,monotonic>

```
Input:
  S(X) — a sample S described by X, |X| = n
  J — evaluation measure (monotonic)
Output:
  L — all equivalent solutions found

procedure ABB (S(X): sample; var L': list
    of set)
  for each x in X do
    enqueue(Q, X \ {x})  // remove a feature at a time
  end
  while not empty(Q) do
    X' := dequeue(Q)
    // X' is legitimate if it is not a subset of a pruned state
    if legitimate(X') and J(S(X')) ≥ J_0 then
      L' := append(L', X')
      ABB(S(X'), L')
    end
  end
end

begin
   Q := ∅           // Queue of pending states
   L' := [X]        // List of solutions
   J_0 := J(S(X))   // Minimum allowed value of J
   ABB (S(X), L')   // Initial call to ABB
   k := smallest size of a subset in L'
   L := set of elements of L' of size k
end
```

Quick branch&bound <rndm/exp,rndm/back,monotonic>

- Use LVF to find a good solution
- Use ABB to explore efficiently the remaining search space

Feature Selection Algorithms

- Introduction
- Relevance of a feature
- Algorithms
- Description of fundamental FSAs
- Generating weighted feature orders
- Empirical and experimental evaluation

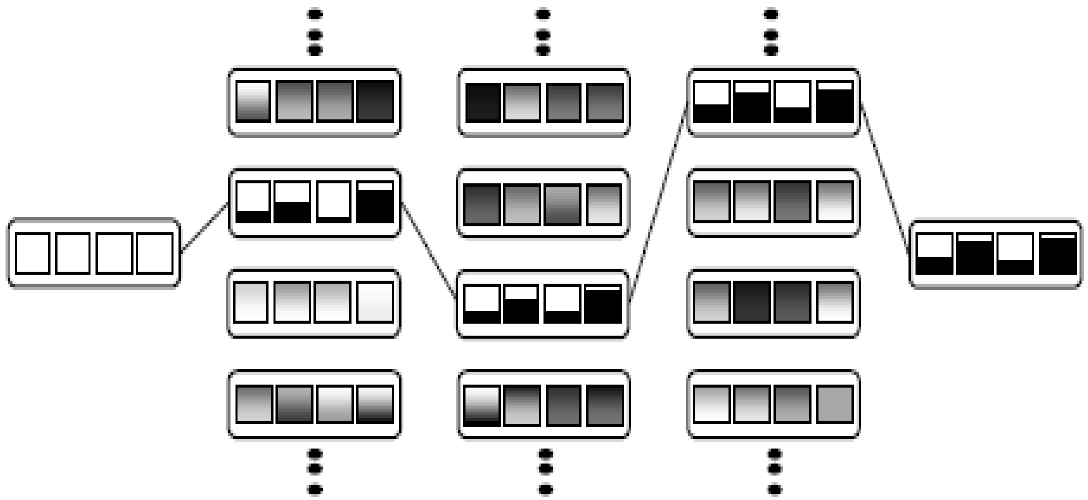


Figure 2. A path of states in the continuous search space involving 4 features. Relevances are represented as a degree of filling.

Feature Selection Algorithms

```
Input:
  p – sampling percentage
  d – distance measure
  S(X) – a sample S described by X, |X| = n
Output:
  W – array of feature weights

initialize W[] to zero
do p|S| times
   A := Random_Element (S)
   A_nh := Near–Hit (A, S)
   A_nm := Near–Miss (A, S)
   for each i ∈ [1..n] do
      W[i] := W[i] + d_i(A, A_nm) − d_i(A, A_nh)
   end
end
```

Closest element to A in S in the same (hit) or a different (miss) class

Feature Selection Algorithms

- Introduction
- Relevance of a feature
- Algorithms
- Description of fundamental FSAs
- Generating weighted feature orders
- Empirical and experimental evaluation

Empirical evaluation of FSAs

- First question: how do we evaluate the effectiveness of a FSA on a given dataset?
  - Relevance: features having an influence on the output
  - Irrelevance: features having no influence on the output (e.g. random values / IDs)
  - Redundance: a feature can play the role of another (e.g. strong correlation)
  - Sample size: number of tuples included in each sample by the algorithm

Scoring solutions

- Notation: $X = X_R \cup X_I \cup X_E$
  - X_R = set of Relevant features ($|X_R| = N_R$)
  - X_I = set of Irrelevant features ($|X_I| = N_I$)
  - X_E = set of rEdundant features ($|X_E| = N_E$)
  - $X^* \subseteq X$ = optimal solution
  - $A^k \subseteq X$ = solution found by the algorithm k
  - $s_X(A)$ = score: how much A and X* have in common
    - $s_X(A) = 0$ if $A = X_I$; $s_X(A) = 1$ if $A = X^*$
- Bad properties (lowering s()):
  - Relevant features lacking in A
  - Redundant features in A
  - Irrelevant features in A
- Weights α_R, α_I, α_E, can be given to these properties

Scoring solutions

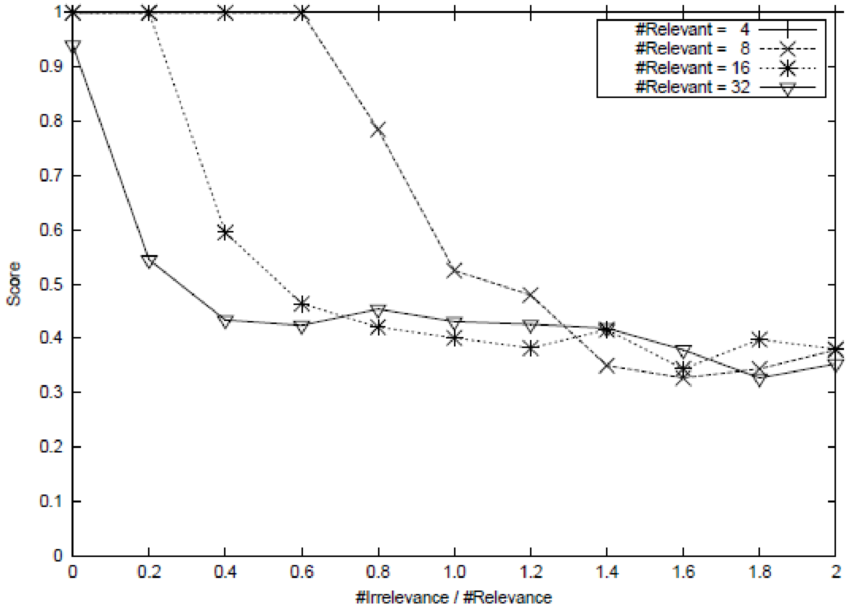
- Rough idea of the score:
  - $R = |A^k_R| / |X_R|$
  - $I = 1 - |A^k_I| / |X_I|$
  - E = ratio between the number of equivalence classes in which the original dataset is split (F) when A or X is considered (roughly speaking $E \simeq 1/|X_E| * (F(A) / F(X))$)
  - $\alpha_R + \alpha_I + \alpha_E = 1$
  - $s_X(A) = \alpha_R R + \alpha_I I + \alpha_E E$

  (for formal definition see Molina et al. 2001)
- Remark: FSAs are not optimizing the *score*!
  - FSA optimize a (local) measure of quality (e.g. consistency)
  - Results are then scored a posteriori with respect to the overall result (weighted score)

Experimental setup

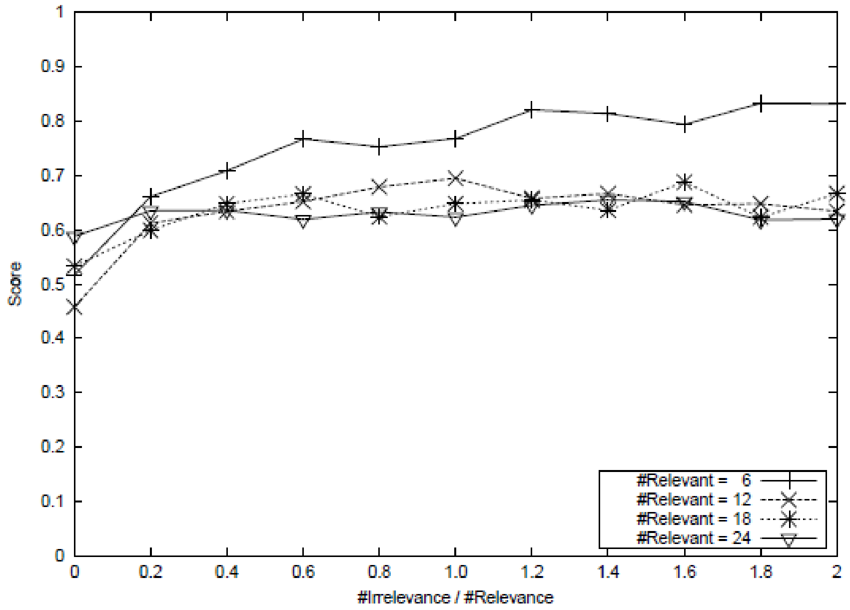
- Consider three problems:
  - Parity
  - Gmonks
  - Disjunction
- Generate synthetic instances by controlling the number of relevant, irrelevant and redundant features
- Run experiments and take average values for different settings of the parameters (e.g. sample size)

Performance of FSAs



(a) Irrelevance vs. Relevance - Parity - C-SBG

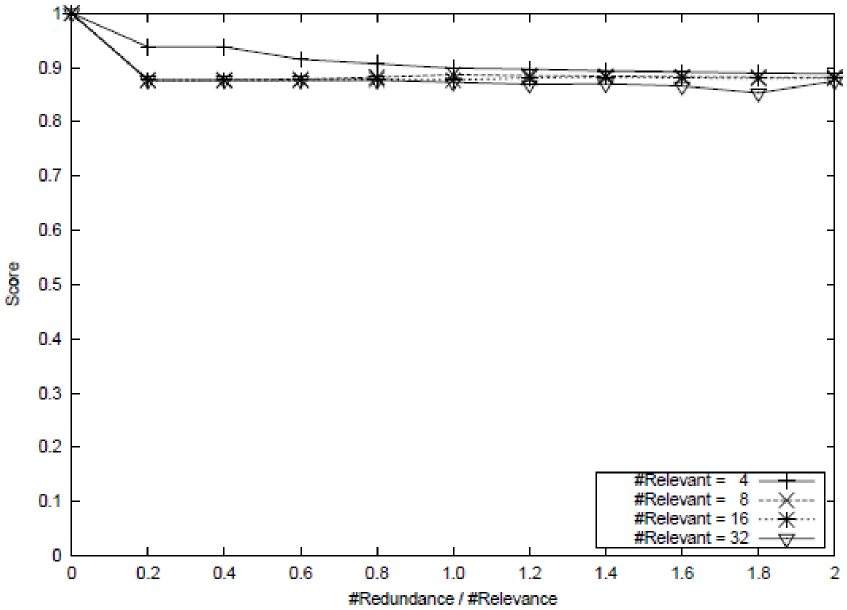
Good in the beginning, but worsens as irrelevance ratio increases



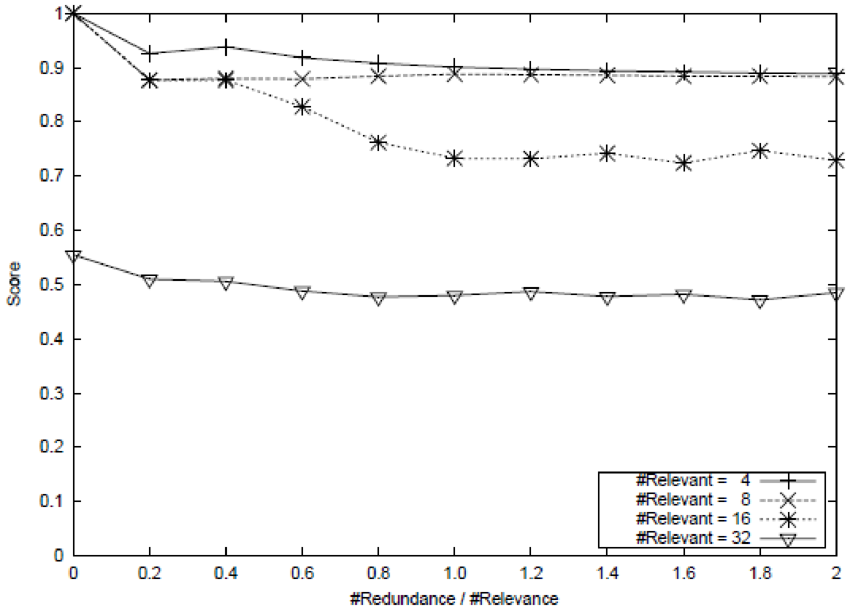
(b) Irrelevance vs. Relevance - GMonks - RELIEF

Improves as irrelevance ratio increases

Performance of FSAs



(c) Redundance vs. Relevance - Parity - LVF

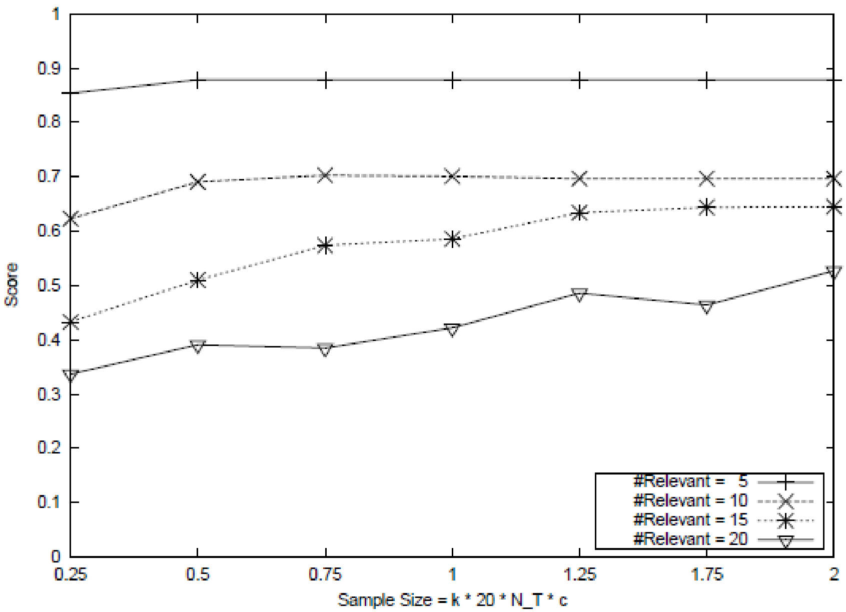


(d) Redundance vs. Relevance - Disjunction - QBB

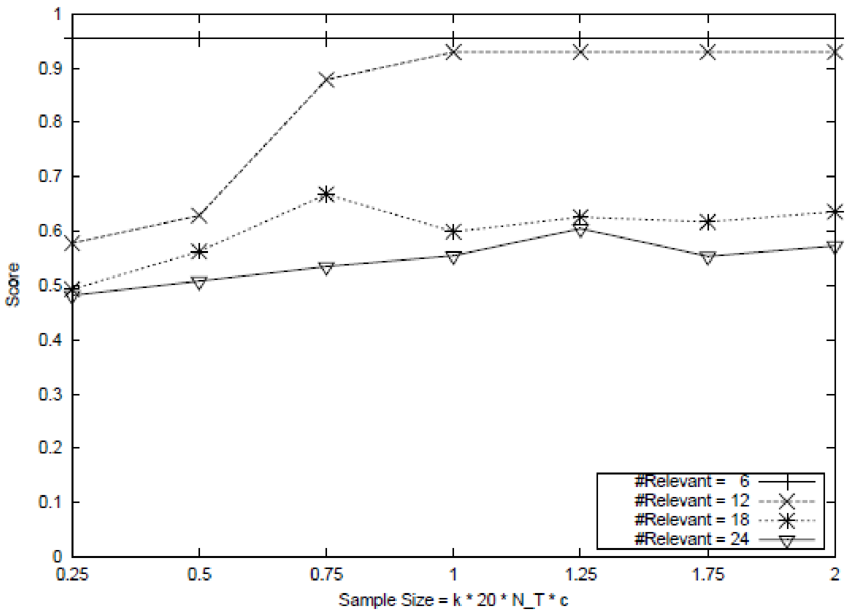
Good and stable

Very stable, but worsens as number of relevant features increases

Performance of FSAs



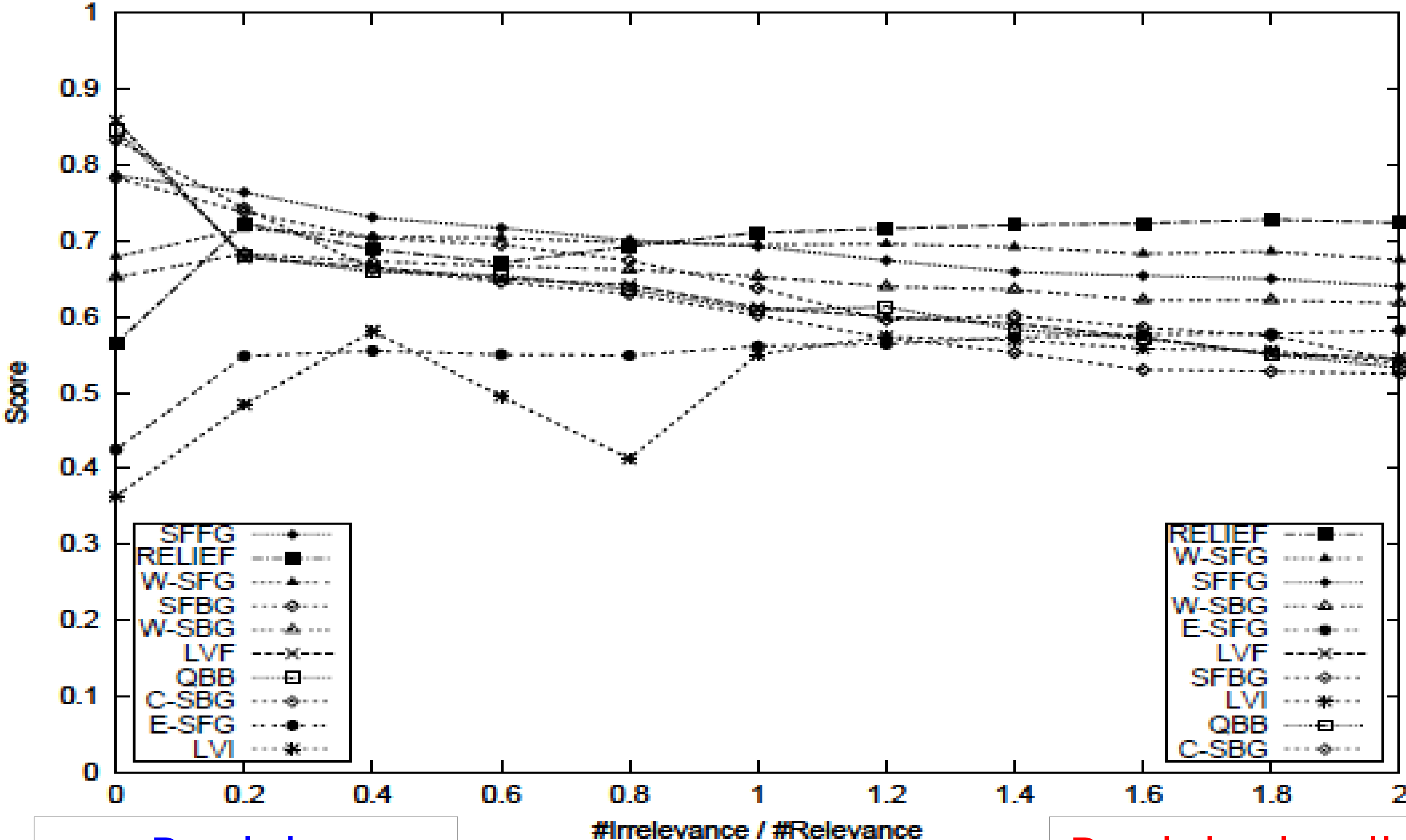
(e) Sample Size - Disjunction - LVI



(f) Sample Size - Parity - W-SBG

Curse of dimensionality effect: performance increase with sample size (more evident for higher number of relevant features)

Comparison of FSAs

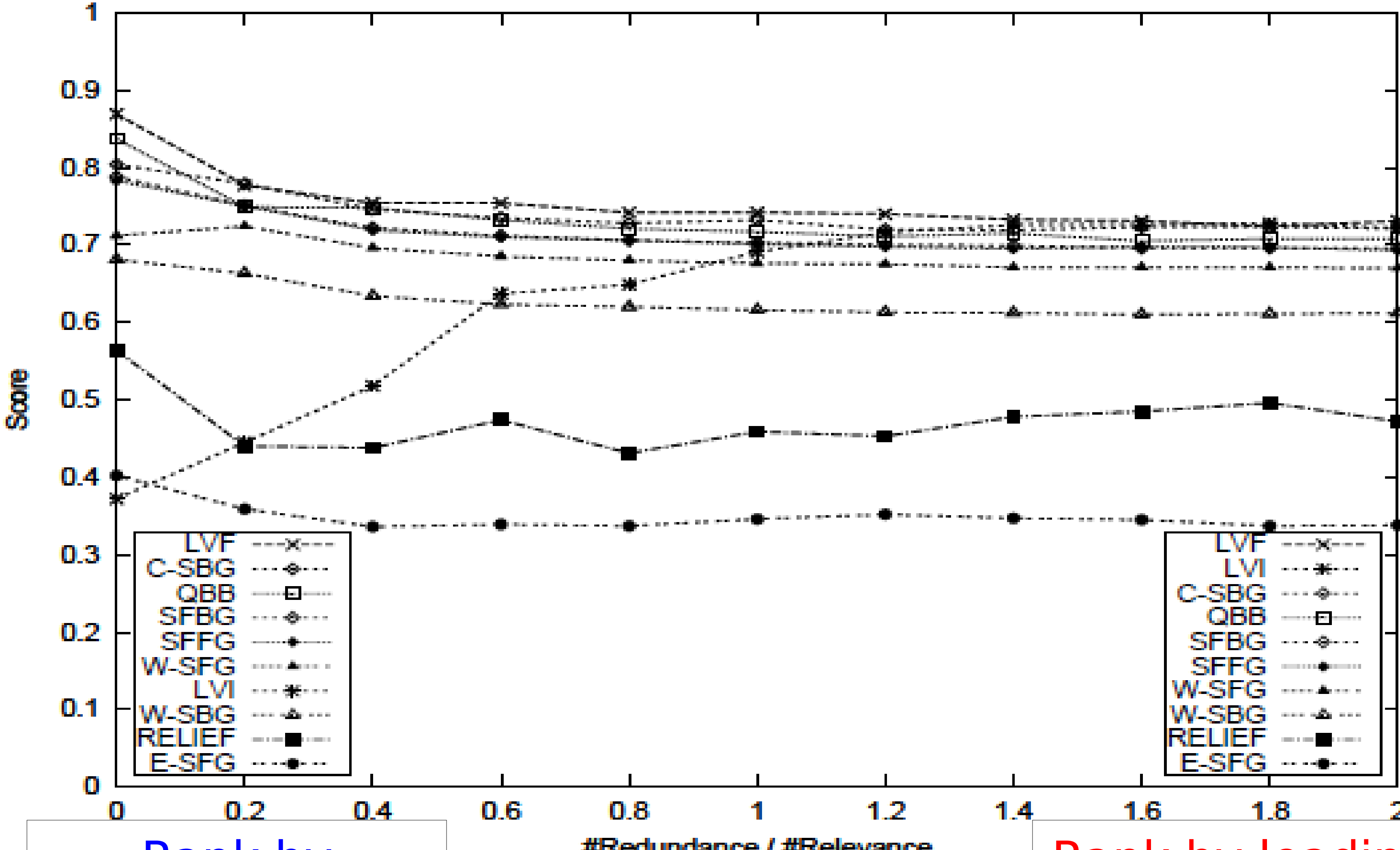


(a) Irrelevance

Rank by average results

Rank by leading in the end

Comparison of FSAs

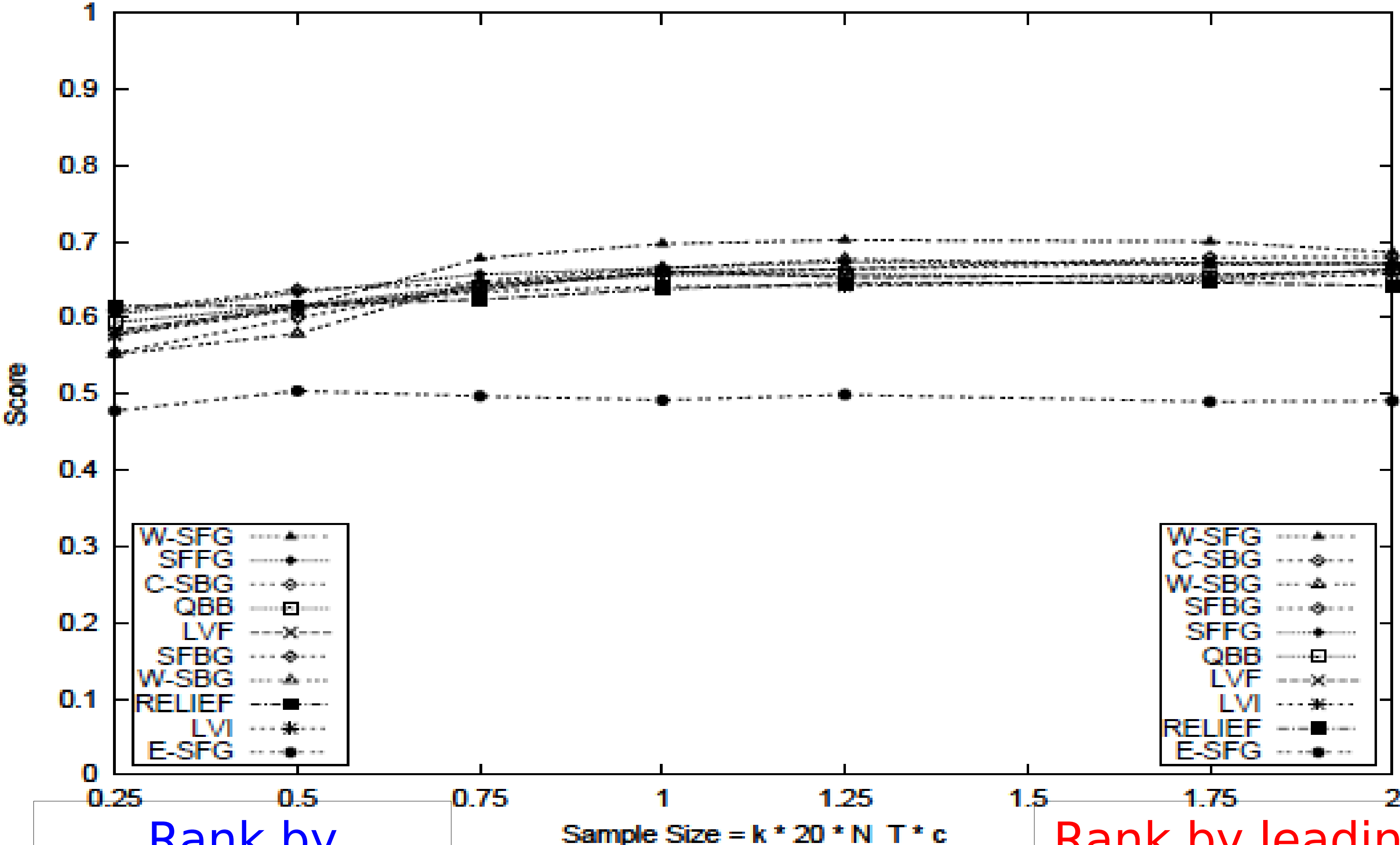


Rank by average results

(c) Redundance

Rank by leading in the end

Comparison of FSAs



(e) Sample Size

Rank by average results

Rank by leading in the end