Bagging Decision Multi-Trees

Vicent Estruch, Cèsar Ferri,
José Hernández-Orallo, M.José Ramírez-Quintana
{vestruch, cferri, jorallo, mramirez}@dsic.upv.es
Dep. de Sistemes Informàtics i Computació,
Universitat Politècnica de València,
Valencia, Spain

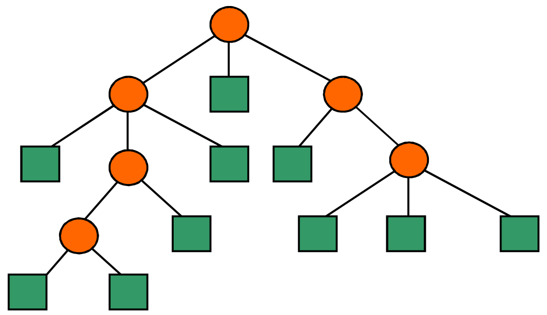
5th Workshop on Multiple Classifier Systems, MCS'04,
Cagliari, Italy*, 9-11 June 2004*

Bagging

- Bagging is derived from the technique known as *bootstrap aggregation*.
- Constructs subsets by generating a sample of *m* training examples, selected randomly (and with replacement) from the original training set of *m* instances.
- On datasets with noise, Bagging usually outperforms Boosting.

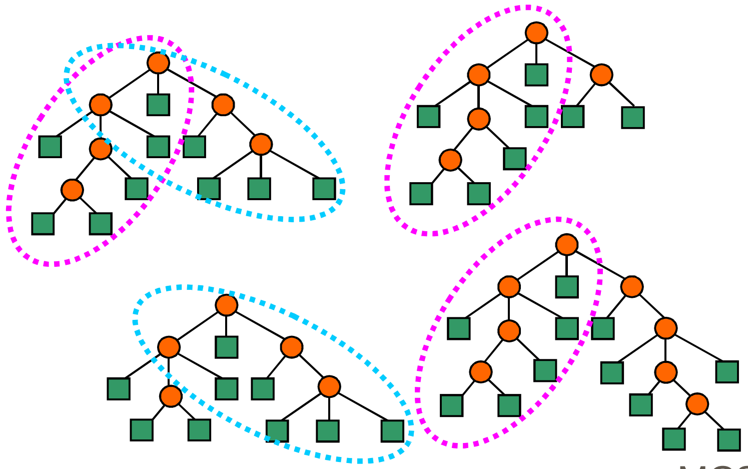
Ensembles of Decision Trees

- Decision Tree:



  - Each internal node represents a condition.
  - Each leaf assigns a class to the examples that fall under that leaf.

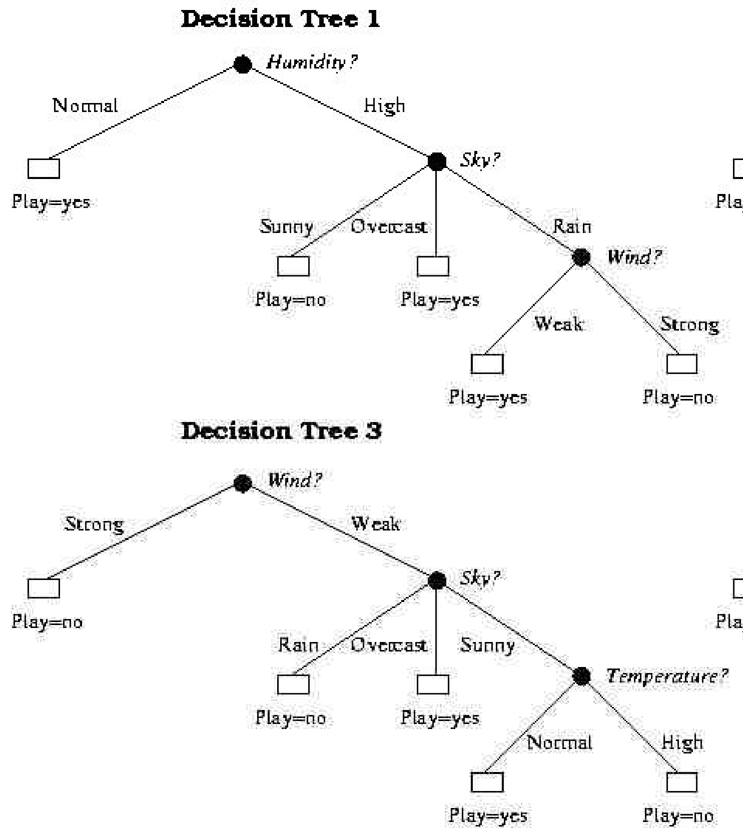
- Forest: several decision trees can be constructed.

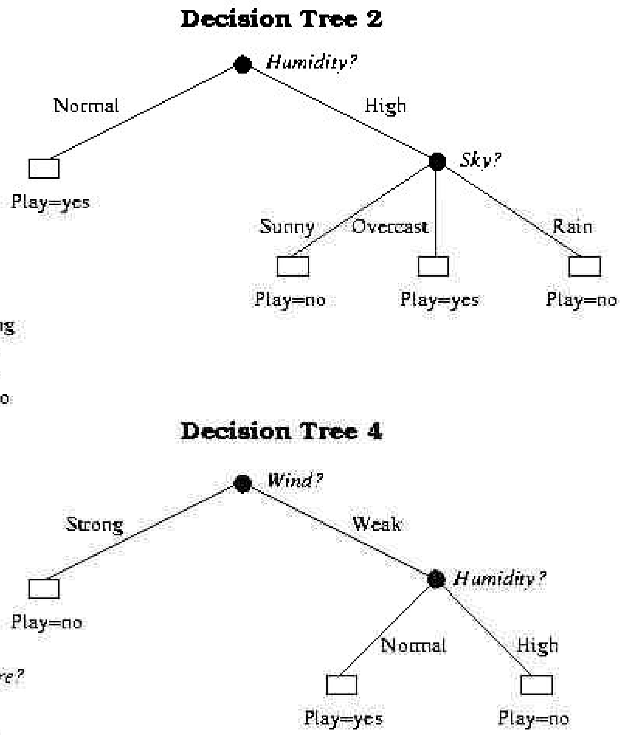


  - Many trees have common parts.
  - Traditional ensemble methods repeat those parts:
    - memory and time ↑↑↑.
    - comprehensibility is lost.

Bagging Decision Trees

- Bagging tends to construct DT's with many similarities (even redundant)

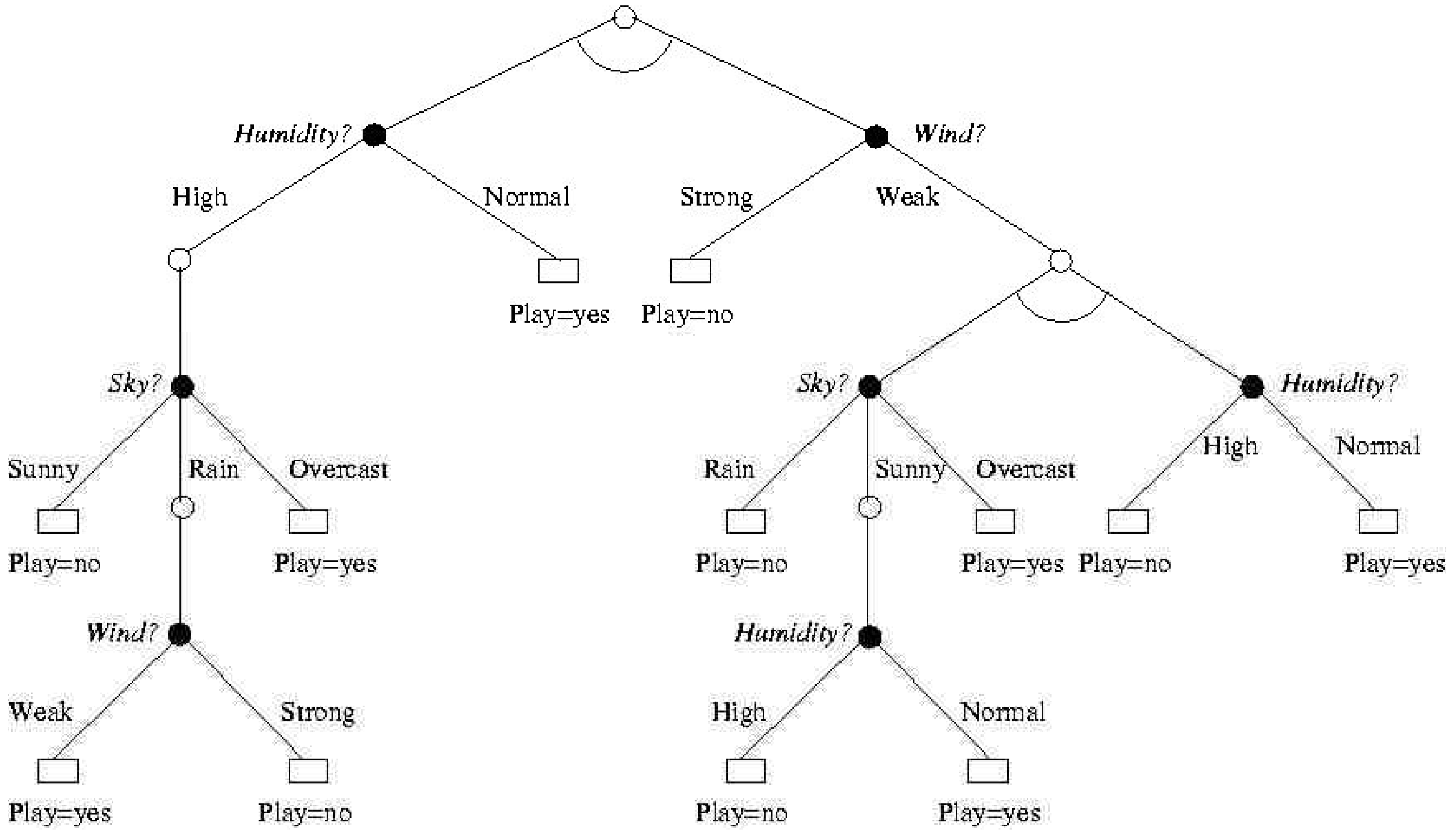




Decision Tree Multi-trees

- Decision Multi-tree:
  - Common parts are shared in an AND/OR tree structure.
  - Construction space and time resources are highly reduced.
  - Throughput is also improved by this technique.

Decision Tree *Shared* Ensembles



Algorithm

Algorithm Bagging-Multi-tree (INPUT E:dataset, n:integer; OUTPUT M:multi-tree) {n is the number of iterations}

```
    M=Initialize multi tree(); {M only contains an empty AND-node}
    for i=1 to n do
            D=Bootstrap replicate(E); {a bootstrap replicate is generated}
    if i=1 then LearnM(M:root; D)
    else LearnMBagg(M:root; D)
    end for
end
```

Algorithm

```
Procedure LearnM (INPUT X:AND-node, D:training dataset; OUTPUT M:multi-tree)
  if X=leaf then exit;
  List of OR nodes=Create OR nodes(X;D); {generate a list with one OR-node for each possible
  split and their descendants (AND-nodes)}
  B=Select Best OR node(L; D); {the best node according to the split optimality criterion is
  selected}
  Activate(B); {the selected OR-node is activated}
  for Y ∈ children of(B) do
      D'=filter(D; Y ); {the examples of D that fall in node Y are selected}
      LearnM(Y;D')
  end for
end procedure
```

Algorithm

```
Procedure LearnMBagg (INPUT X:AND-node, D:training dataset; OUT-
  PUT M:multi-tree)
if X=leaf then exit;
List of OR nodes=Update OR nodes(X;D);{update the split optimality of OR-nodes according to
  the training set D}
B=Select Best OR node(L; D);
if B is active then
  for Y ∈ children of (B) do
      D'=filter(D; Y );
      LearnMBagg(Y;D')
  end for
else
  Activate(B);
  for Y ∈ children of(B) do
      D'=filter(D; Y );
      LearnM(Y;D') {the multi-tree is expanded from node Y }
  end for
end procedure
```

Bagging DT versus Bagging DMT

- In Bagging DT, there is a significant probability of learning similar trees. In the prediction phase, the repeated decision trees will be more determinant in the final decision. In bagging DMT, since we avoid duplicated trees, all the leaves have identical weight in the final decision.

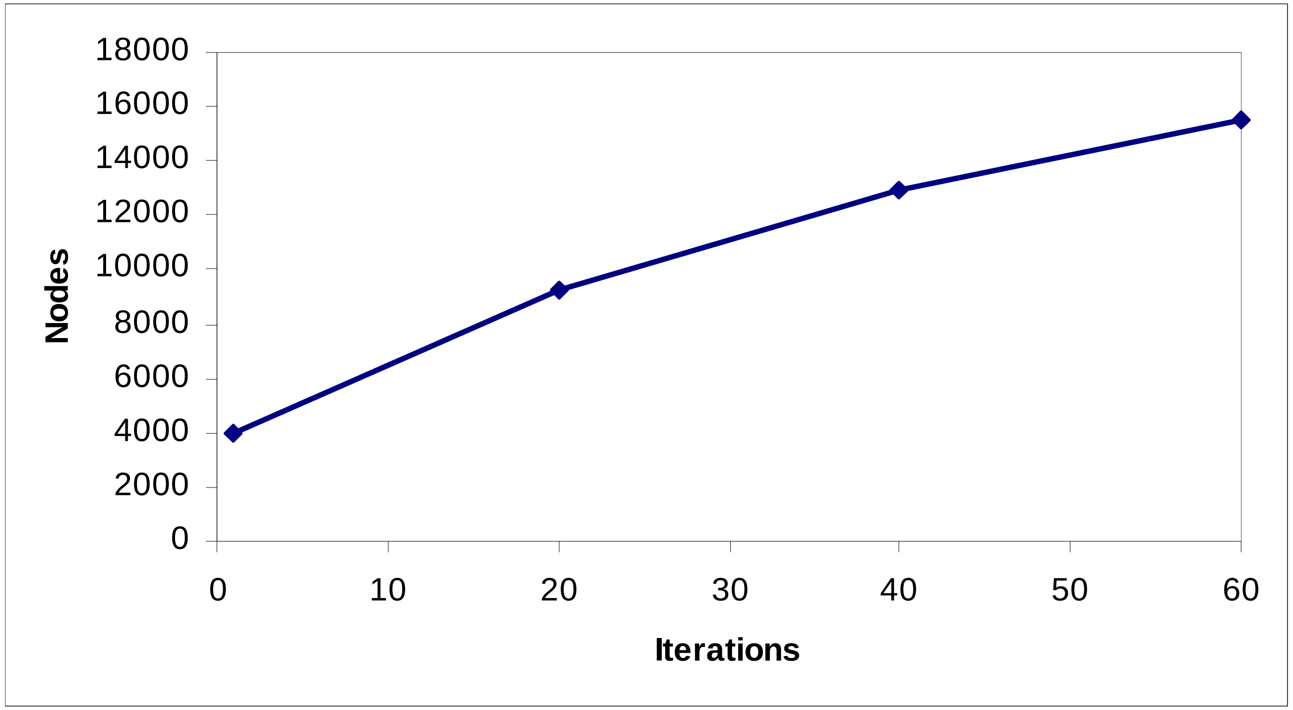
- In a DMT, the fusion of the predictions is performed internally at the OR-nodes, while in an ensemble of decision trees the voting is performed using each independent decision tree.

Experiments (1/4)

- Experimental setting:
  - 22 datasets from the UCI repository.
  - 10 X 10 Cross Validation.
  - Multi-tree implemented in the **SMILES** system.
  - Splitting criterion: GainRatio (C4.5) without pruning.
  - Bagging from WEKA.

Experiments (2/4)

- Mean size of decision multi-trees in number of nodes



- Bagging tends to repeat the same DTs !!!

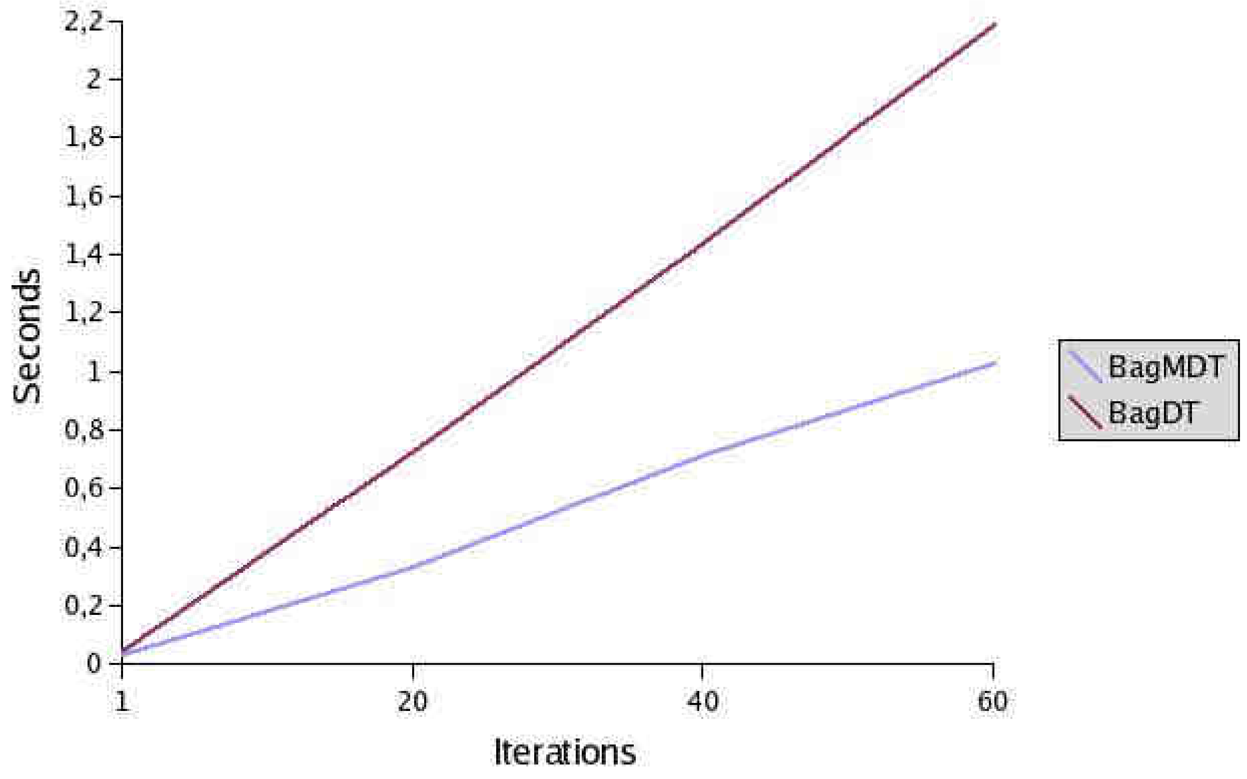
Experiments (3/4)

- Mean Accuracy compared to other Ensemble Methods:

	1		20		40		60	
Dataset	MDT	DT	MDT	DT	MDT	DT	MDT	DT
1	77,94	79,4	79,26	81,92	80,13	82,85	80,63	82.92
2	93,81	94,12	94,28	96,07	94,64	96,11	94,81	96.28
3	92,43	94,22	93,48	95,61	93,13	95,91	93,29	95.91
4	99,61	99,4	99,37	99,43	99,42	99,4	99,28	99.40
5	91,58	93,8	94	96,43	94,31	96,89	94,75	97.05
6	72,13	72,5	73,44	47,31	73,44	54,25	73,19	52.56
7	51,64	47,26	54,58	46,21	54,72	46,45	55,98	46.52
8	76,2	78,98	77,27	83,23	77,47	82,38	77,8	82.76
9	62,72	65,82	64,14	65,66	64,81	65,77	65,33	65.79
10	78,53	83,15	81,92	82,86	82,19	83,12	82,5	83.06
11	94,74	95,47	94,95	96,45	94,77	96,55	94,81	96.58
12	94,13	94,6	95,13	94,57	94,13	94,33	94,47	94.20
13	96,73	95,11	95,93	99,84	96,42	100	95,73	100.00
14	70,97	62,66	67,62	66,38	67,45	66,91	66,17	67.19
15	97,47	98,67	98	98,74	97,69	98,9	97,93	98.88
16	92,81	92,95	92,76	94,24	93,29	94,56	93,1	94.76
17	66	59,89	66,25	63,23	65,88	64,44	65,5	65.00
18	95,91	96,9	95,29	97,67	95,23	97,69	95,23	97.62
19	61,93	56,46	56,93	59,35	57,33	61,5	55,6	60.51
20	99,23	99,72	99,15	99,65	99,06	99,65	98,98	99.65
21	77,16	79,16	79,54	83,17	80,32	83,82	80,34	83.80
22	93	93,49	93,12	95,26	93,06	95,79	92,94	95.90
Geomean	82,18	81,66	82,6	81,67	82,73	82,55	82,69	82.46
Average	83,49	83,35	83,93	83,79	84,04	84,42	84,02	84.38

Experiments (4/4)

- Combination Resources compared to other Ensemble Methods:



Conclusions

- We present an algorithm that reduces the high computational cost (time and memory) characteristic of Bagging method.
- When Bagging with DT, the set of decision trees obtained presents many structural similarities (low diversity). MDT reduces redundancy, however, all the leaves have the same weight in the final decision.
- The multi-tree structure can be viewed as a feasible and elegant way to overcome the main inherent drawbacks (huge amount of the computational resources and redundancy) of Bagging.

Future Work

- Investigate how we can improve accuracy by an adequate adjustment of the diversity and redundancy parameters in Bagging multi-trees.
- Study whether the multi-tree is able to enhance other well-known ensemble methods, such as boosting.