An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants

Eric Bauer
Computer Science Department
Stanford University

Ron Kohavi
Data Mining and Visualization
Silicon Graphics Inc.

Presented by:
Nick Hamatake
Karan Singh

Motivation

- **Methods for voting classification algorithms show much promise**
- **Review algorithms and present a large empirical study comparing several variants**
- **Improve understanding of why and when these algorithms work**
- **Explore practical problems**

Outline

- Introduction
- The base inducers
- The voting algorithms
- Bias/Variance decomposition
- Experimental design
- Bagging and variants
- Boosting: AdaBoost and Arc-x4
- Future work
- Conclusions

Outline

- **Introduction**
- The base inducers
- The voting algorithms
- Bias/Variance decomposition
- Experimental design
- Bagging and variants
- Boosting: AdaBoost and Arc-x4
- Future work
- Conclusions

Introduction

- **Two types of voting algorithms**
  - **Perturb and Combine**
  - **Adaptively Resample and Combine**
- **Authors describe a large empirical study to understand why and when algorithms affect classification error**

Outline

- Introduction
- **The base inducers**
- The voting algorithms
- Bias/Variance decomposition
- Experimental design
- Bagging and variants
- Boosting: AdaBoost and Arc-x4
- Future work
- Conclusions

The base inducers

- **4 base inducers, implemented in *MLC++***
- **Decision trees**
  - **MC4 (MLC++ C4.5)**
  - **MC4(1) (decision stump)**
  - **MC4(1)-disc (entropy discretization)**
- **Naïve-Bayes**

Outline

- Introduction
- The base inducers
- **The voting algorithms**
- Bias/Variance decomposition
- Experimental design
- Bagging and variants
- Boosting: AdaBoost and Arc-x4
- Future work
- Conclusions

Bagging (Brieman 1996)

Input: training set S, Inducer $\mathcal{I}$, integer T (number of bootstrap samples).

1. for $i = 1$ to T {
2. $S' =$ bootstrap sample from S (i.i.d. sample with replacement).
3. $C_i = \mathcal{I}(S')$
4. }
5. $C^*(x) = \arg\max_{y \in Y} \sum_{i:C_i(x)=y} 1$ (the most often predicted label y)

Output: classifier C^*.

AdaBoost.M1 (Freund & Schapire 1995)

Input: training set S of size m, Inducer $\mathcal{I}$, integer T (number of trials).

1. $S' = S$ with instance weights assigned to be 1.
2. For $i = 1$ to T {
3. $\quad C_i = \mathcal{I}(S')$
4. $\quad \epsilon_i = \frac{1}{m} \sum_{x_j \in S' : C_i(x_j) \neq y_j} \text{weight}(x)$ (weighted error on training set).
5. $\quad$ If $\epsilon_i > 1/2$, set S' to a bootstrap sample from S with weight 1 for every instance and goto step 3 (this step is limited to 25 times after which we exit the loop).
6. $\quad \beta_i = \epsilon_i / (1 - \epsilon_i)$
7. $\quad$ For-each $x_j \in S'$, if $C_i(x_j) = y_j$ then $\text{weight}(x_j) = \text{weight}(x_j) \cdot \beta_i$.
8. $\quad$ Normalize the weights of instances so the total weight of S' is m.
9. }
10. $C^*(x) = \arg\max_{y \in Y} \sum_{i : C_i(x) = y} \log \frac{1}{\beta_i}$

Output: classifier C^*.

Arc-x4 (Breiman 1996)

Input: training set S of size m, Inducer $\mathcal{I}$, integer T (number of trials).

1. For $i = 1$ to T {
2. $S' = S$ with instance weight for x set to $(1 + e(x)^4)$ (normalized so total weight is m), where $e(x)$ is the number of misclassifications made on x by classifiers C_1 to C_{i-1}.
3. $C_i = \mathcal{I}(S')$
4. }
5. $C^*(x) = \arg\max_{y \in Y} \sum_{i:C_i(x)=y} 1$ (the most often predicted label y)

Output: classifier C^*.

Outline

- Introduction
- The base inducers
- The voting algorithms
- **Bias/Variance decomposition**
- Experimental design
- Bagging and variants
- Boosting: AdaBoost and Arc-x4
- Future work
- Conclusions

Bias/Variance decomposition

Intrinsic "target noise" (σ^2) This quantity is a lower bound on the expected error of any learning algorithm. It is the expected error of the Bayes-optimal classifier.

Squared "bias" (bias^2) This quantity measures how closely the learning algorithm's average guess (over all possible training sets of the given training set size) matches the target.

"Variance" (variance) This quantity measures how much the learning algorithm's guess fluctuates for the different training sets of the given size.

Bias/Variance decomposition

$$\text{Error} = \sum_x P(x)\left(\sigma_x^2 + \text{bias}_x^2 + \text{variance}_x\right)$$

where

$$\sigma_x^2 \equiv \frac{1}{2}\left(1 - \sum_{y \in Y} P(Y_F = y|x)^2\right)$$

$$\text{bias}_x^2 \equiv \frac{1}{2}\sum_{y \in Y}\left[P(Y_F = y|x) - P(Y_H = y|x)\right]^2$$

$$\text{variance}_x \equiv \frac{1}{2}\left(1 - \sum_{y \in Y} P(Y_H = y|x)^2\right) .$$

Bias/Variance decomposition

- **Kohavi and Wolpert (1996)**
  - **two-stage sampling procedure**
    1) **test set is split from from training set**
    2) **remaining data, D, is sampled repeatedly to estimate bias/variance on the test set**
  - **whole process can be repeated multiple times to improve estimates**

Outline

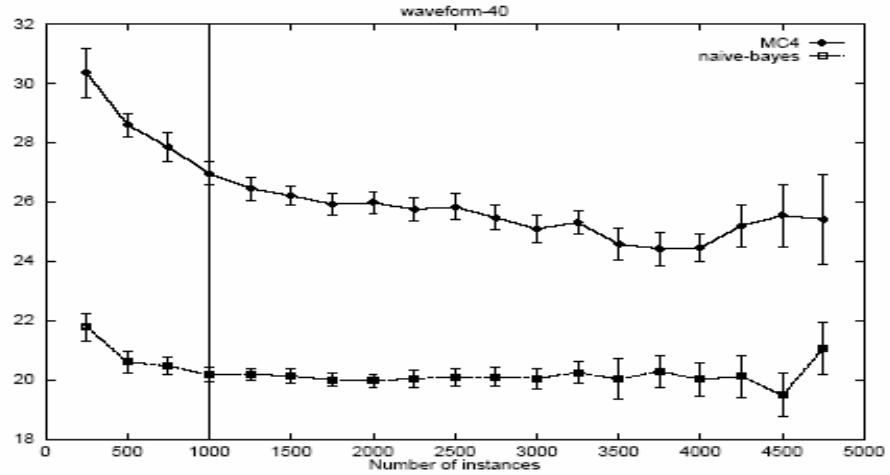
- Introduction
- The base inducers
- The voting algorithms
- Bias/Variance decomposition
- **Experimental design**
- Bagging and variants
- Boosting: AdaBoost and Arc-x4
- Future work
- Conclusions

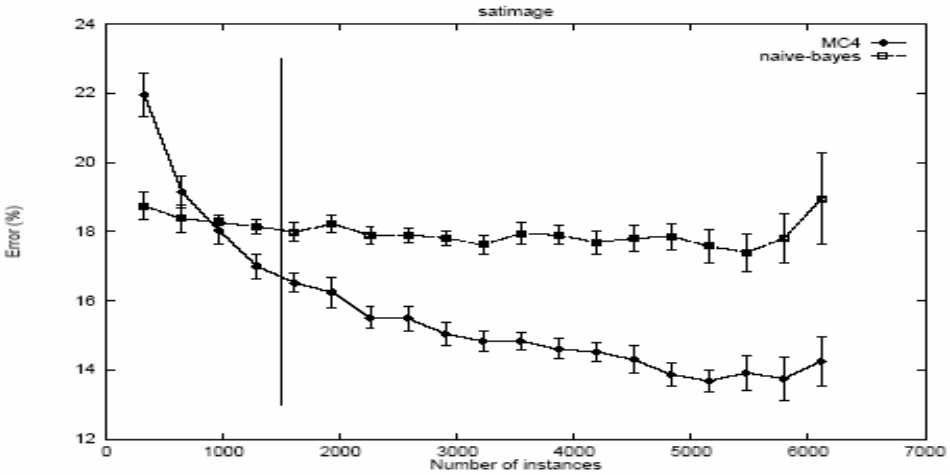
Experimental design

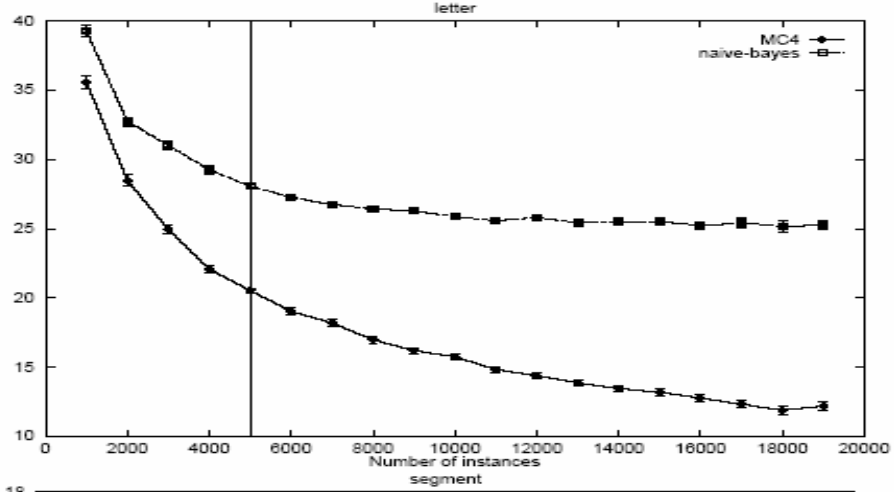
- all data sets have at least 1000 instances
- training set size selected at a point where error was still decreasing
- at least half of the data used for test set
- only 25 sub-classifiers voted for each algorithm
- authors verified their implementations of voting algorithms against original papers

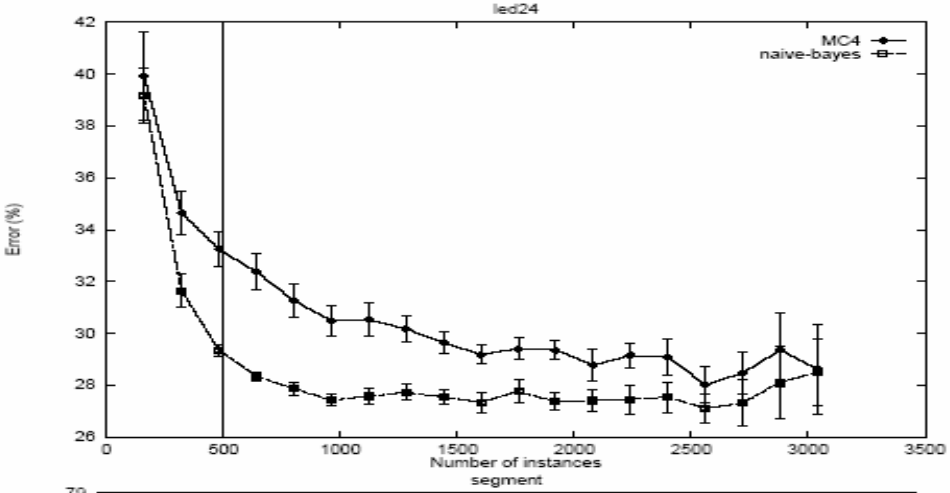
Experimental design: datasets and training set sizes used

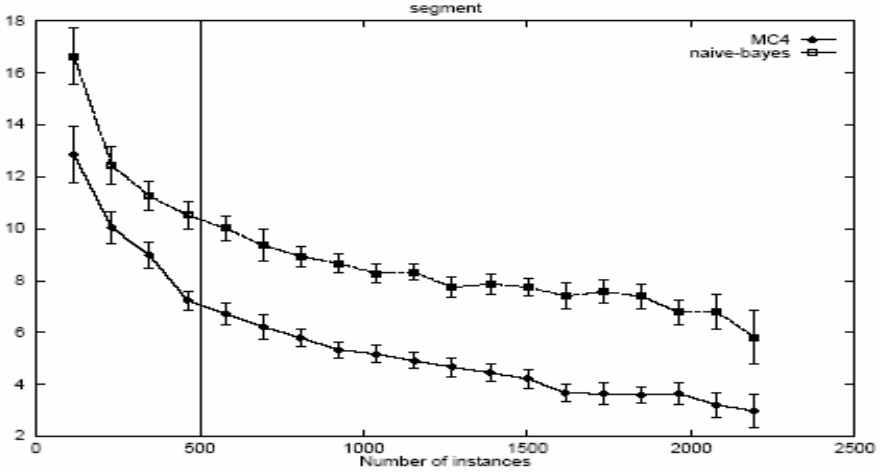
Data set	Dataset size	Training set size	Attributes Cont/nominal	Classes
Credit (German)	1,000	300	7/13	2
Image Segmentation (segment)	2,310	500	19/0	7
Hypothyroid	3,163	1,000	7/18	2
Sick-euthyroid	3,163	800	7/18	2
DNA	3,186	500	0/60	3
Chess	3,196	500	0/36	2
LED-24	3,200	500	0/24	10
Waveform-40	5,000	1,000	40/0	3
Satellite image (satimage)	6,435	1,500	36/0	7
Mushroom	8,124	1,000	0/22	2
Nursery	12,960	3,000	0/8	5
Letter	20,000	5,000	16/0	26
Adult	48,842	11,000	6/8	2
Shuttle	58,000	5,000	9/0	7

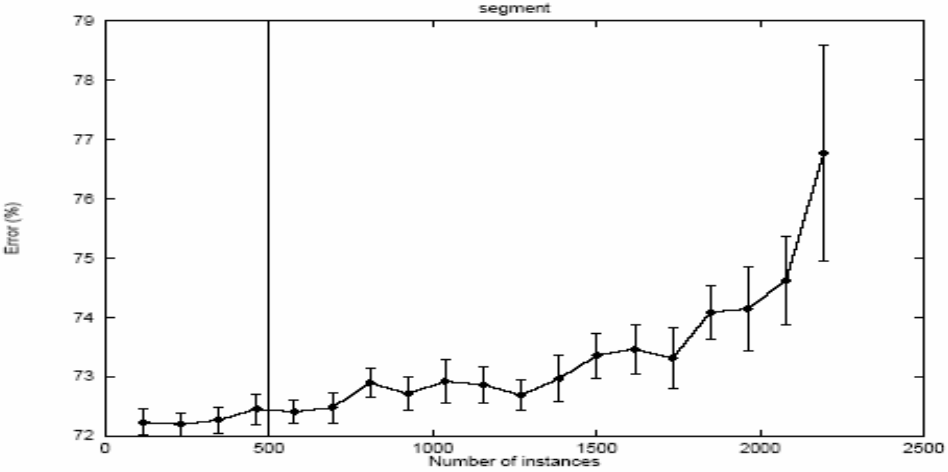

waveform-40
MC4
naive-bayes
Number of instances


satimage
MC4
naive-bayes
Error (%)
Number of instances


letter
MC4
naive-bayes
Number of instances

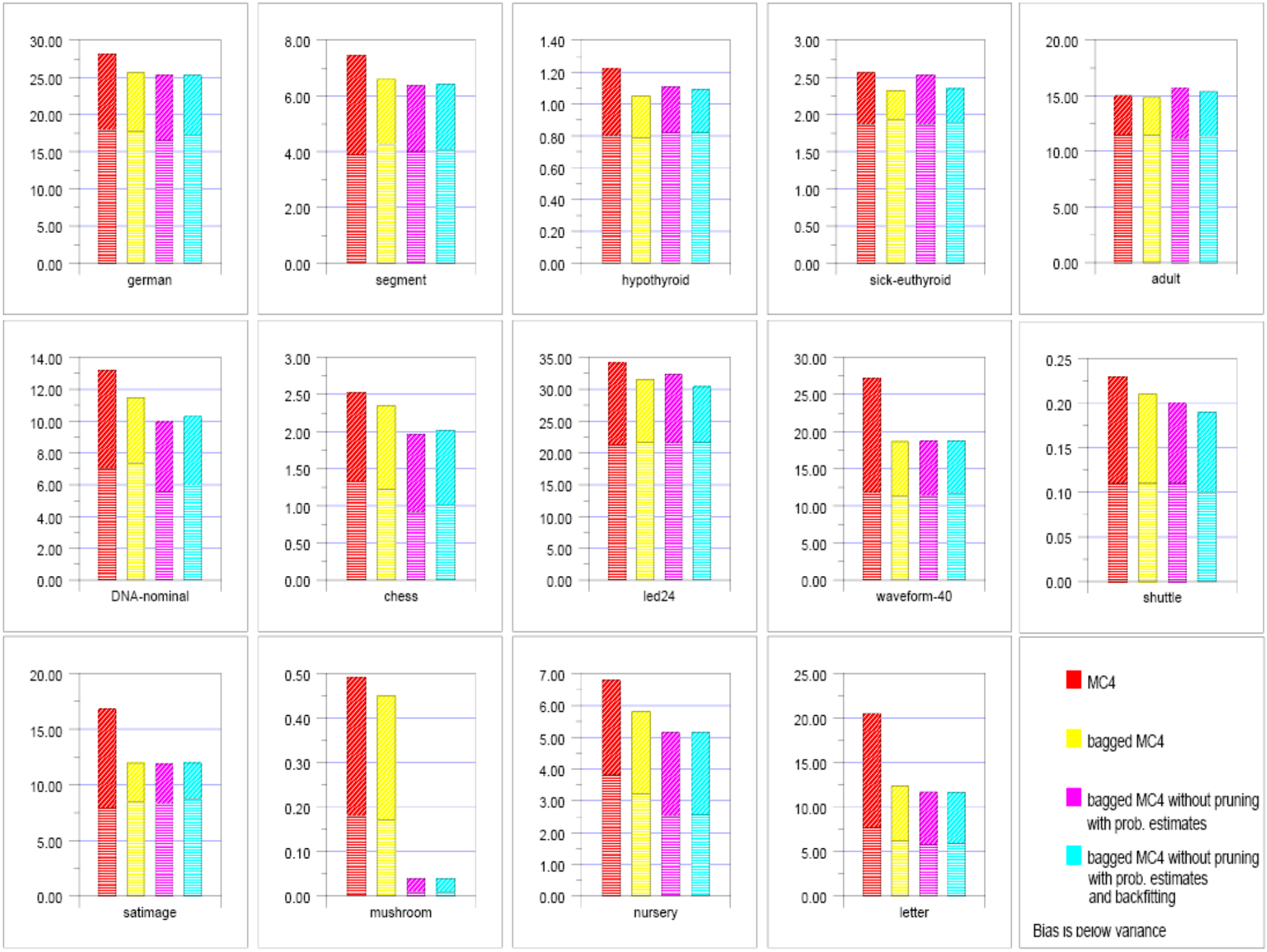

led24
MC4
naive-bayes
Error (%)
Number of instances


segment
MC4
naive-bayes
Number of instances


segment
Error (%)
Number of instances

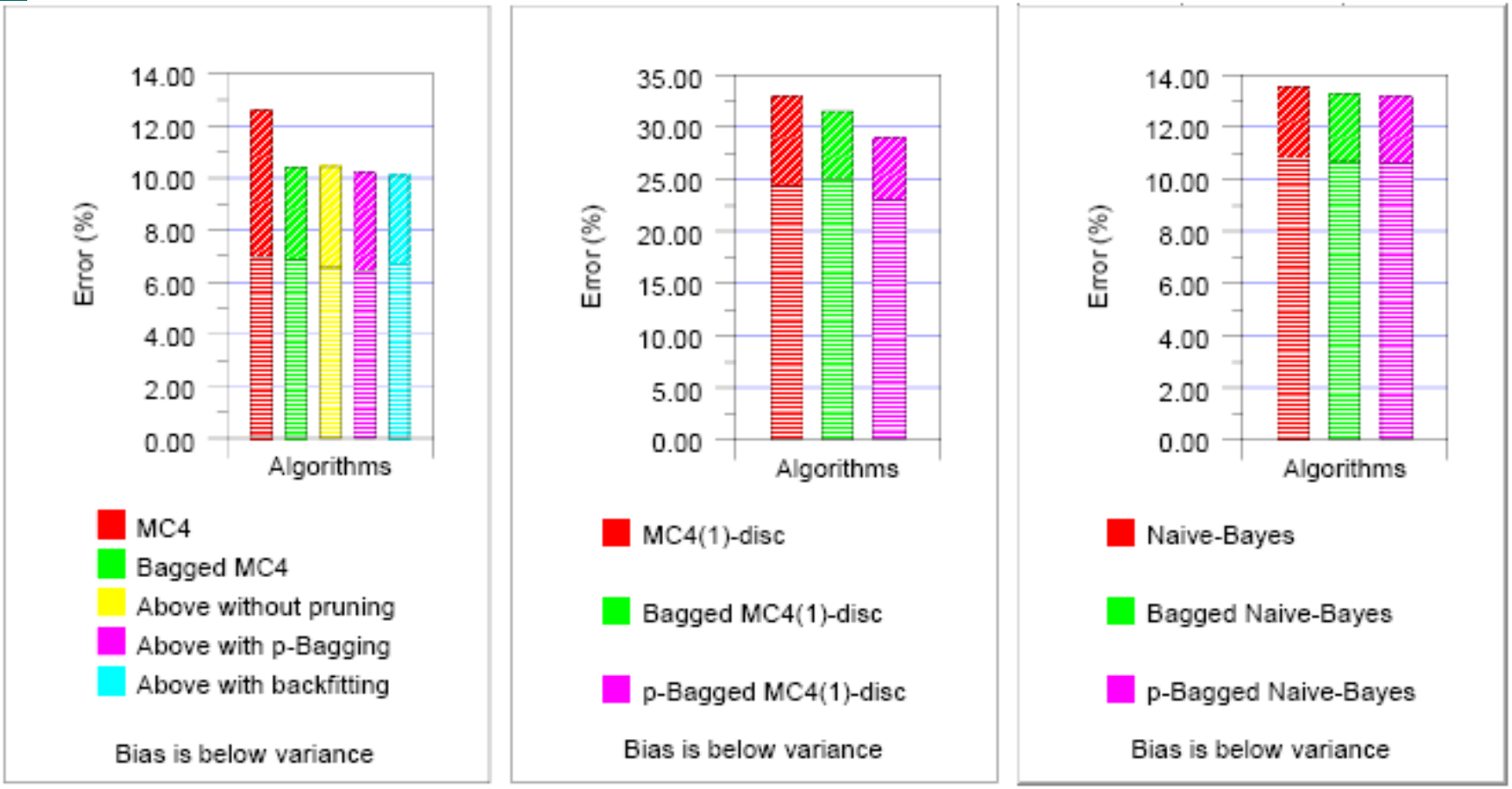
Outline

- Introduction
- The base inducers
- The voting algorithms
- Bias/Variance decomposition
- Experimental design
- **Bagging and variants**
- Boosting: AdaBoost and Arc-x4
- Future work
- Conclusions


30.00
25.00
20.00
15.00
10.00
5.00
0.00
german
8.00
6.00
4.00
2.00
0.00
segment
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
hypothyroid
3.00
2.50
2.00
1.50
1.00
0.50
0.00
sick-euthyroid
20.00
15.00
10.00
5.00
0.00
adult
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
DNA-nominal
3.00
2.50
2.00
1.50
1.00
0.50
0.00
chess
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
led24
30.00
25.00
20.00
15.00
10.00
5.00
0.00
waveform-40
0.25
0.20
0.15
0.10
0.05
0.00
shuttle
20.00
15.00
10.00
5.00
0.00
satimage
0.50
0.40
0.30
0.20
0.10
0.00
mushroom
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
nursery
25.00
20.00
15.00
10.00
5.00
0.00
letter
MC4
bagged MC4
bagged MC4 without pruning with prob. estimates
bagged MC4 without pruning with prob. estimates and backfitting
Bias is below variance

Bagging Results

- Performance boost across the board
- Most gain due to reduced variance (but some bias)
- Biggest effect on many-attr sets
- Little effect on large training sets
- Increased tree size (198 to 240)
- Naïve Bayes - little to no change
  - 3% average relative decrease in error
  - Already highly-stable technique


14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Error (%)
Algorithms
MC4
Bagged MC4
Above without pruning
Above with p-Bagging
Above with backfitting
Bias is below variance
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
Error (%)
Algorithms
MC4(1)-disc
Bagged MC4(1)-disc
p-Bagged MC4(1)-disc
Bias is below variance
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Error (%)
Algorithms
Naive-Bayes
Bagged Naive-Bayes
p-Bagged Naive-Bayes
Bias is below variance

Why the increased tree size?

- **Theory: Less pruning due to duplicate data**
- **Test: Bagged trees actually *smaller (25%) before pruning***
- ***Idea: Skip pruning***
  - ***Pruning dampens variance while increasing bias***
  - ***Bagging does well with high variance***
  - ***Results: 11% increase in rel. error for un-bagged trees; slight decrease in error for bagged***

Bagging Probabilities (p-Bagging)

- **Motivation: MC4 can generative probabilities; why not use them?**
- **Skipping pruning process should help predict prob.'s even better (why?)**
- **Worked well**
  - **2%, 4%, 8% rel. reduction in rel. error for MC4, MC4(1), MC4(1)-disc.**
  - **No significant change for Naive-Bayes (when alg. tuned to predict prob.'s)**

Wagging and Backfitting

- **Wagging - Weight Aggregation**
  - **Different method of perturbing train set -- noise vs. re-sampling**
  - **Tunable: higher variance in noise leads to reduced variance (higher bias)**
  - **Performs on-par with bagging when tuned optimally**
- **Backfitting - Use whole training set to calculate leaf probabilities**
  - **Avg. rel. var. decrease of 11% (3% total error); improvement (or no change) in *all models***

Bagging: Summary

- **Three new variants: disabling pruning, p-Bagging, backfitting**
- **Each variant improved performance; 4% rel. decrease in error at the end**
- **Surprising reduce in bias for stumps**
- **Naïve Bayes not helped much**
- **MSE improved dramatically with p-Bagging**

Outline

- Introduction
- The base inducers
- The voting algorithms
- Bias/Variance decomposition
- Experimental design
- Bagging and variants
- **Boosting: AdaBoost and Arc-x4**
- Future work
- Conclusions

Boosting: AdaBoost and Arc-x4

- **but first, an example:**
  - **to get a better understanding of the process of boosting**
  - **highlight the issue of numerical instabilities and underflows**

Example

- **"A training set of size 5000 was used with the shuttle dataset. The MC4 algorithm already has relatively small test error of 0.38%."**

- **Update rule:**
  For each x_j, divide weight(x_j) by $2\epsilon_i$ if $C_j(x_j) \neq y_j$ and $2(1-\epsilon_i)$ otherwise

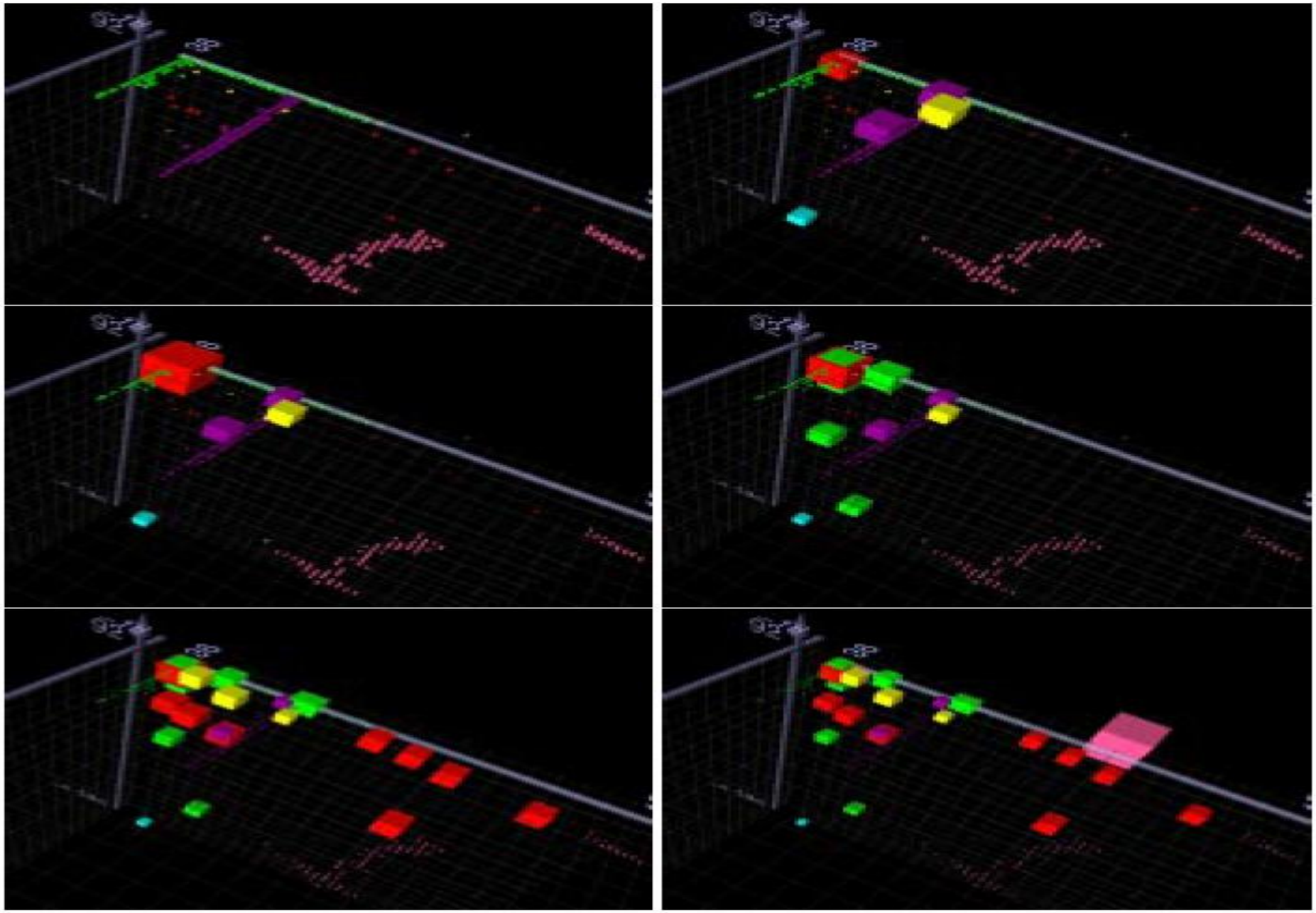


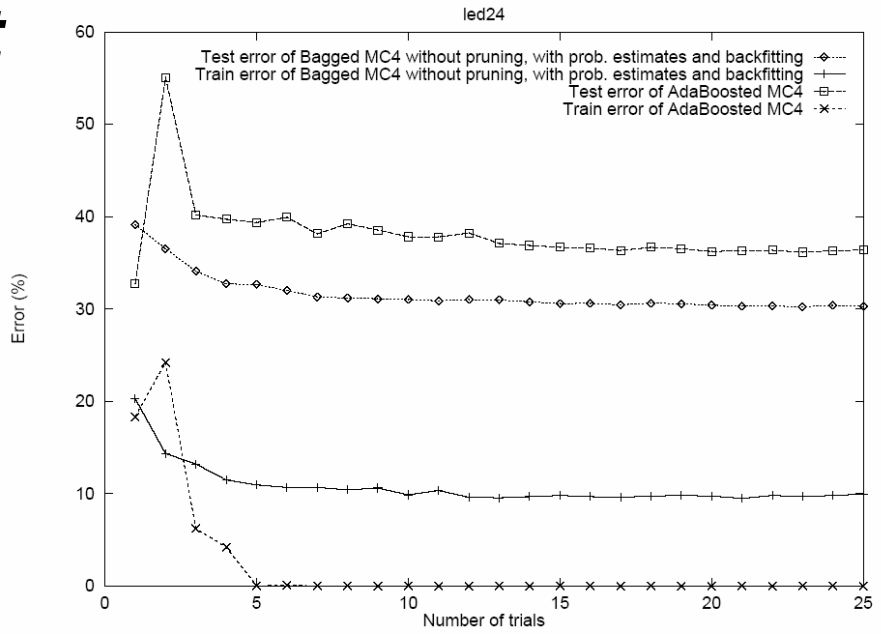
Figure 8. The shuttle dataset projected on the three most discriminatory axes. Color/shading denotes the class of instances and the cube sizes correspond to the instance weights. Each picture shows one AdaBoost trial, where progression is from the top left to top right, then middle left, etc.

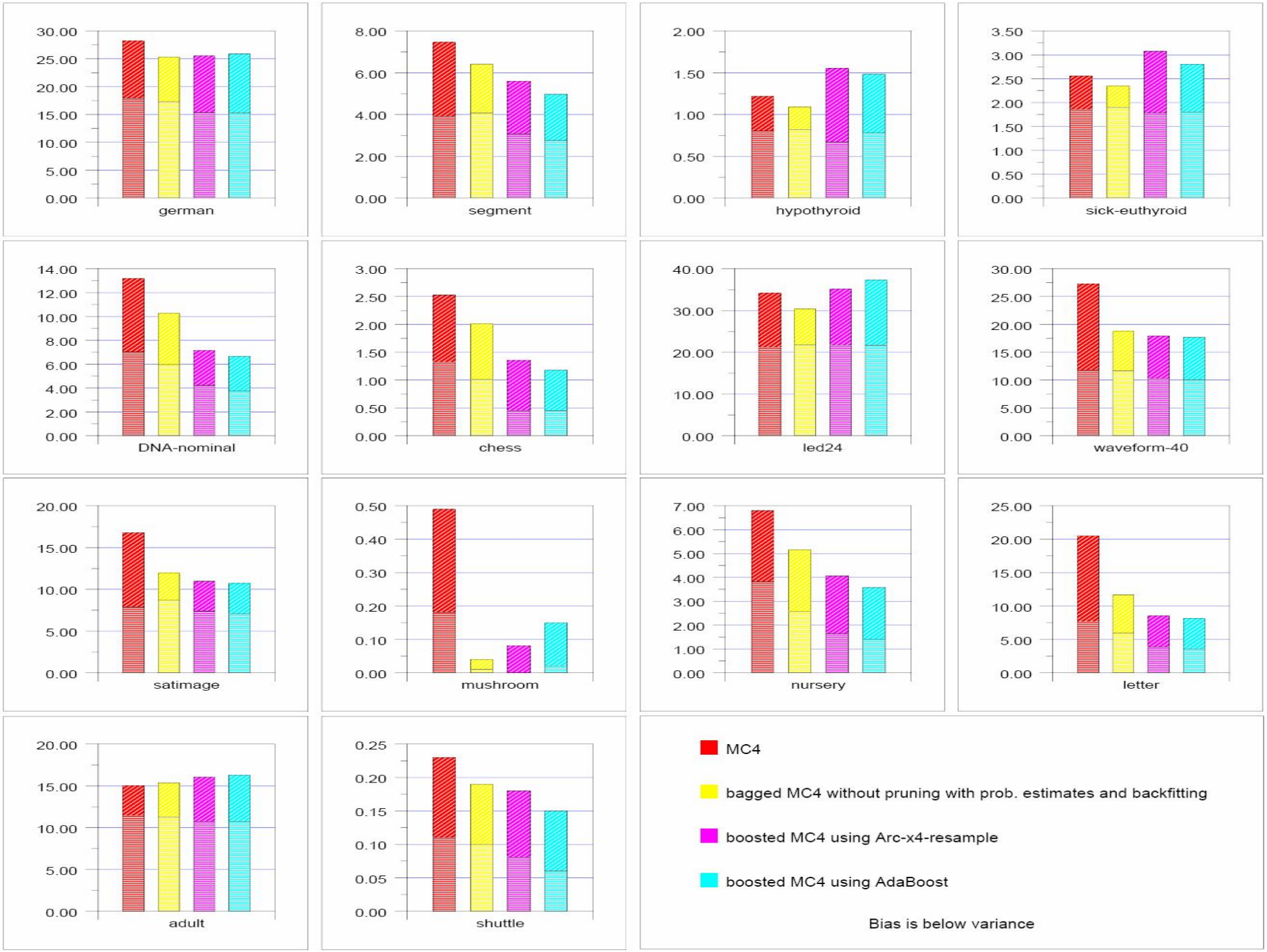
Example: lesson learned …

- **using the update rule that avoids the normalization step circumvents underflows early in the process, but they can still happen**
- **authors set instances with weights falling below the minimum weight to have the minimum weight**

Boosting: AdaBoost results

- **AdaBoosted MC4 gave average relative error reduction of 27%**
- **AdaBoost beat backfit-p-Bagging (average absolute error: 9.8% vs 10.1%)**
- **AdaBoost was *not* uniformly better for all datasets**




30.00
25.00
20.00
15.00
10.00
5.00
0.00
german
8.00
6.00
4.00
2.00
0.00
segment
2.00
1.50
1.00
0.50
0.00
hypothyroid
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
sick-euthyroid
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
DNA-nominal
3.00
2.50
2.00
1.50
1.00
0.50
0.00
chess
40.00
30.00
20.00
10.00
0.00
led24
30.00
25.00
20.00
15.00
10.00
5.00
0.00
waveform-40
20.00
15.00
10.00
5.00
0.00
satimage
0.50
0.40
0.30
0.20
0.10
0.00
mushroom
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
nursery
25.00
20.00
15.00
10.00
5.00
0.00
letter
20.00
15.00
10.00
5.00
0.00
adult
0.25
0.20
0.15
0.10
0.05
0.00
shuttle
MC4
bagged MC4 without pruning with prob. estimates and backfitting
boosted MC4 using Arc-x4-resample
boosted MC4 using AdaBoost
Bias is below variance

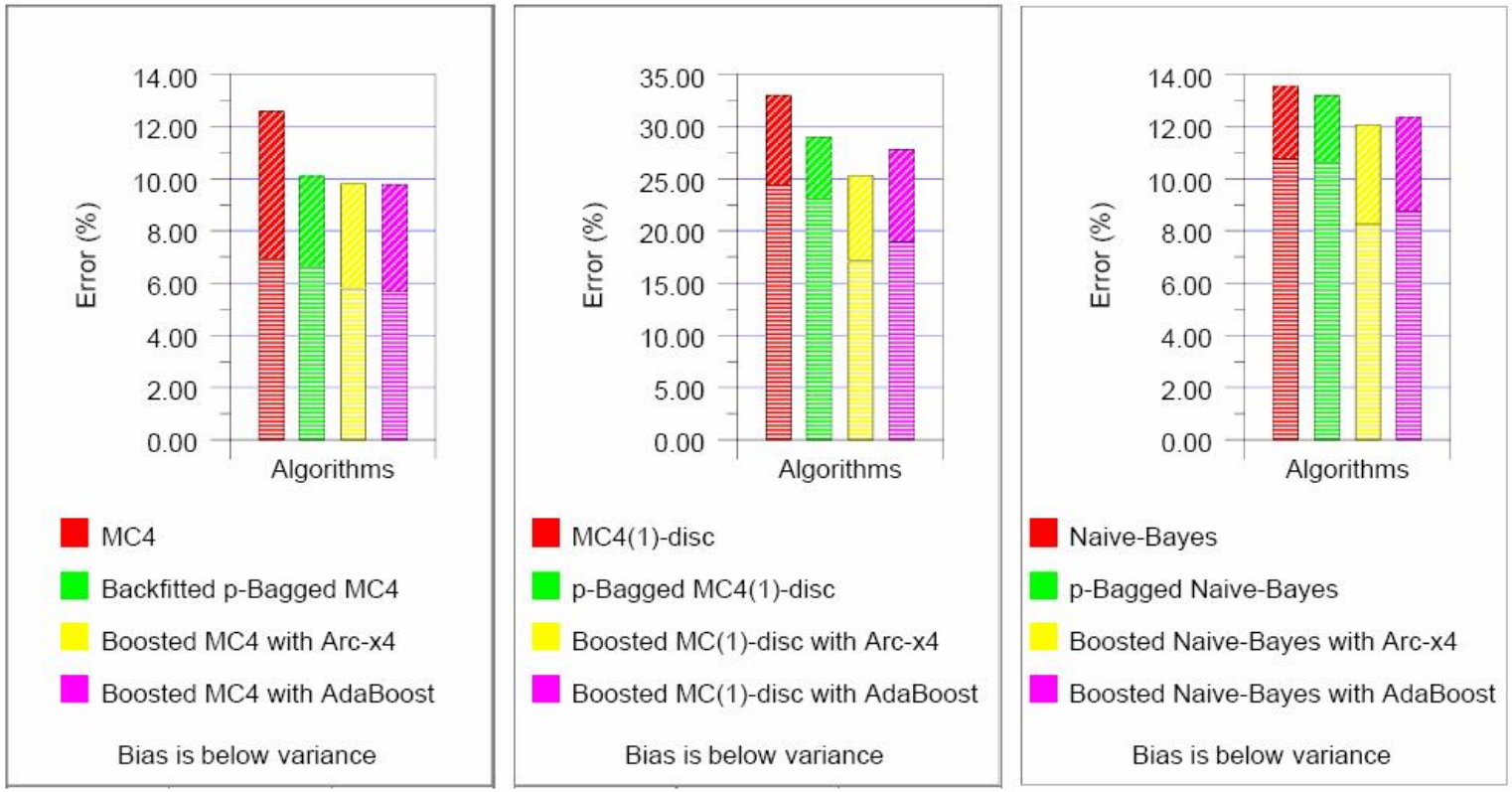
Boosting: AdaBoost results

- **MC4(1) with boosting fails**
- **AdaBoosted MC4(1)-disc gave average relative error reduction of 31%**
- **AdaBoosted Naïve-Bayes gave average relative error reduction of 24%**

Boosting: Arc-x4 results

- **two versions:**
  - **Arc-x4-reweight**
  - **Arc-x4-resample**
- **Arc-x4-resample MC4 9.81% avg error**
- **Arc-x4-reweight MC4 10.86% avg error**
- **Arc-x4-reweight has higher variance**
- **does worse on same datasets as AdaBoost, likely due to noise**
- **Arc-x4 outperforms AdaBoost for MC4(1)-disc and Naïve-Bayes**

Boosting: summary



Boosting: summary

- ***on average*, boosting is better than bagging for the given datasets**
- **boosting is *not uniformly* better than bagging**
- **AdaBoost does not deal well with noise**
- **both algorithms reduced bias and variance for decision trees; increased variance for naïve-bayes**

Outline

- Introduction
- The base inducers
- The voting algorithms
- Bias/Variance decomposition
- Experimental design
- Bagging and variants
- Boosting: AdaBoost and Arc-x4
- **Future work**
- Conclusions

Future work

- Improving robustness of boosting w.r.t. noise
- Are there better ways to handle a “perfect” classifier when boosting?
- Bagging & boosting seem to drift from Occam's razor; can we quantify this bias toward complex models?
- Are there cases when pruning helps bagging?
- Can stacking improve bagging & boosting even further?
- Applying boosting to other methods: kNN
- Other techniques of combining probabilities
- Can boosting be parallelized?

Outline

- Introduction
- The base inducers
- The voting algorithms
- Bias/Variance decomposition
- Experimental design
- Bagging and variants
- Boosting: AdaBoost and Arc-x4
- Future work
- **Conclusions**

Conclusions

- **Performance of bagging and boosting depends highly on the data set**
- **Bagging very helpful w.r.t. MSE**
- **Larger trees in AdaBoost trials correlated with greater success in reducing error**
- **Arc-x4 works far better with re-sampling, unlike other boosting methods**