Ensembles

Key Idea: “Wisdom of the crowd“

groups of people can often make better decisions than individuals

Apply this to ML

Learn multiple classifiers and combine their predictions

Combining Multiple Classifiers by Voting

Train several classifiers and take majority of predictions

For regression use mean or median of the predictions

For ranking and collective classification use some form of averaging

A common family of approaches is called **bagging**

Bagging: Split the Data

Q: What can go wrong with option 1?

Option 1: Split the data into K pieces and train a classifier on each

Bagging: Split the Data

Q: What can go wrong with option 1?

A: Small sample → poor performance

Option 1: Split the data into K pieces and train a classifier on each

Bagging: Split the Data

Q: What can go wrong with option 1?

A: Small sample → poor performance

Option 1: Split the data into K pieces and train a classifier on each

Option 2: Bootstrap aggregation (bagging) resampling

Bagging: Split the Data

Option 1: Split the data into K pieces and train a classifier on each

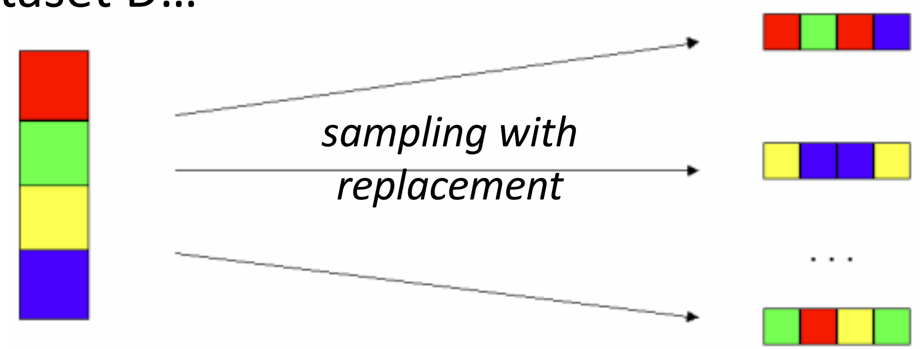
Q: What can go wrong with option 1?

A: Small sample → poor performance

Option 2: Bootstrap aggregation (bagging) resampling

Obtain datasets $D_1, D_2, \ldots, D_N$ using bootstrap resampling from D

Given a dataset D…



get new datasets D̂ by random sampling with replacement from D

Courtesy Hamed Pirsiavash

Bagging: Split the Data

Q: What can go wrong with option 1?

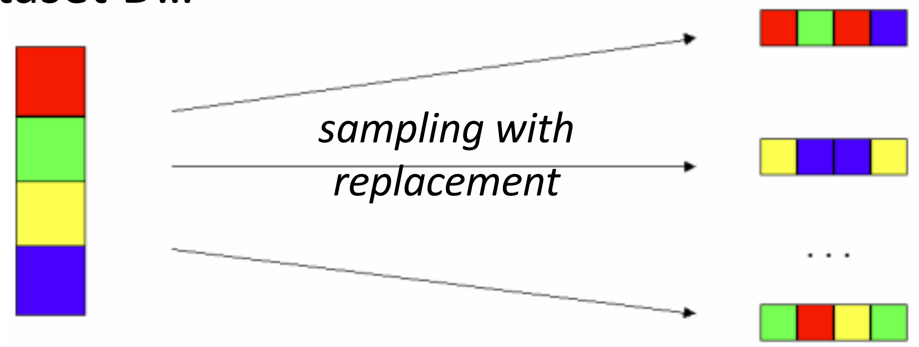
A: Small sample → poor performance

Option 1: Split the data into K pieces and train a classifier on each

Option 2: Bootstrap aggregation (bagging) resampling

- Obtain datasets $D_1, D_2, \ldots, D_N$ using bootstrap resampling from D
- Train classifiers on each dataset and average their predictions

Given a dataset D…



get new datasets D̂ by random sampling with replacement from D

Courtesy Hamed Pirsiavash

Bagging: Bootstrap Aggregating

- For $b = 1, \ldots, B$ do
  - S_b = bootstrap replicate of S
  - Apply learning algorithm to S_b to learn h_b
- Classify new points by unweighted vote:
  - $[\sum_b h_b(x)]/B > 0$

Bagging Decision Trees

How would it work?

Bagging Decision Trees

How would it work?

Bootstrap S samples $\{(X_1, Y_1), \ldots, (X_S, Y_S)\}$

Train a tree t_s on (X_s, Y_s)

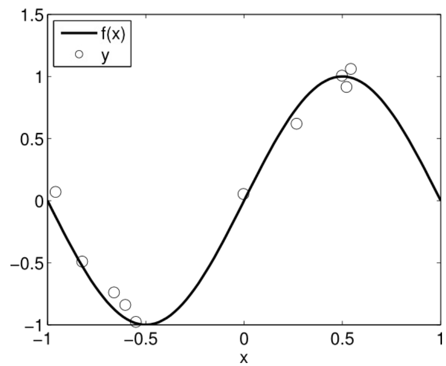
At test time: $\hat{y} = \text{avg}(t_1(x) \ , \ldots t_S(x))$

Bagging

- Bagging makes predictions according to
  $y = \Sigma_b h_b(x) / B$
- Hence, bagging's predictions are $\underline{h}(x)$

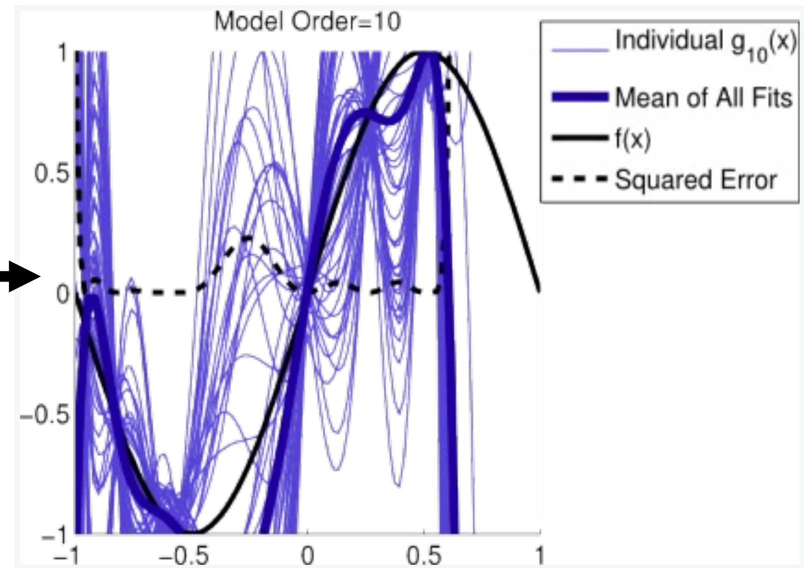
Why does averaging work?

Averaging reduces the variance of estimators

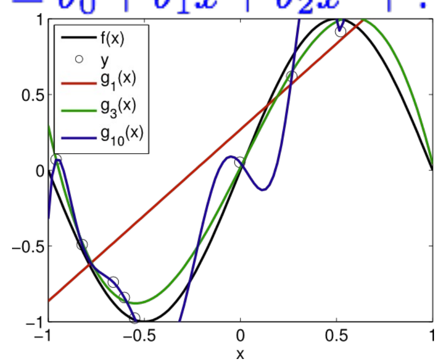


$$y = f(x) + \epsilon$$
$$f(x) = \sin(\pi x)$$
$$\epsilon = N(0, \sigma^2)$$
$$\sigma = 0.1$$

50 samples



$$g_n(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_n x^n$$



Averaging is a form of regularization: each model can individually overfit but the average is able to overcome the overfitting

Courtesy Hamed Pirsiavash

Random Forests

Bagging trees with one modification

At each split point, choose a **random subset of features** of size **k** and pick the best among these

Train decision trees of depth **d**

Average results from multiple randomly trained trees

Q: What's the difference between bagging decision trees and random forests?

Random Forests

Bagging trees with one modification

At each split point, choose a random subset of features of size **k** and pick the best among these

Train decision trees of depth **d**

Average results from multiple randomly trained trees

Q: What's the difference between bagging decision trees and random forests?

A: Bagging → highly correlated trees (reuse good features)

Courtesy Hamed Pirsiavash

Bias, Variance, and Noise

- Variance: $E[\,(h(x^*) - \underline{h(x^*)})^2\,]$

  Describes how much $h(x^*)$ varies from one training set S to another
- Bias: $[\underline{h(x^*)} - f(x^*)]$

  Describes the <u>average</u> error of $h(x^*)$.
- Noise: $E[\,(y^* - f(x^*))^2\,] = E[\varepsilon^2] = \sigma^2$

  Describes how much y^* varies from $f(x^*)$

Estimated Bias and Variance of Bagging

- If we estimate bias and variance using the same B bootstrap samples, we will have:
  - Bias = ($\underline{h}$ – y)   [same as before]
  - Variance = Σ_k ($\underline{h}$ – $\underline{h}$)^2/(K – 1) = 0
- Hence, according to this approximate way of estimating variance, bagging removes the variance while leaving bias unchanged.
- In reality, bagging only *reduces* variance and tends to slightly increase bias

Bias/Variance Heuristics

- Models that fit the data poorly have high bias: "inflexible models" such as linear regression, regression stumps
- Models that can fit the data very well have low bias but high variance: "flexible" models such as nearest neighbor regression, regression trees
- This suggests that bagging of a flexible model can reduce the variance while benefiting from the low bias

Decomposition over an entire data set

- Given a set of test points

  $T = \{(x^*_1, y^*_1), \ldots, (x^*_n, y^*_n)\}$,

  we want to decompose the average loss:

  $\underline{L} = \Sigma_i \, E[\, L(h(x^*_i), y^*_i) \,] / n$
- We will write it as

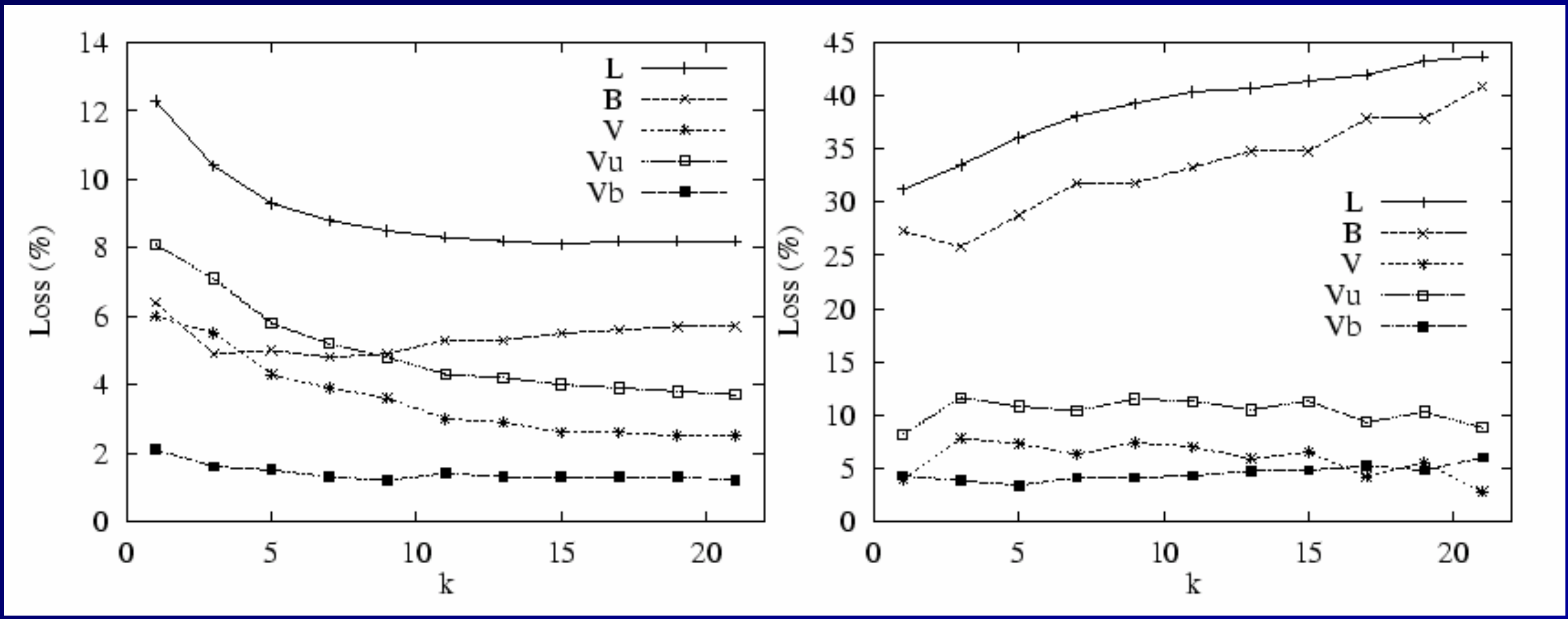
  $\underline{L} = \underline{B} + \underline{Vu} - \underline{Vb}$

  where $\underline{B}$ is the average bias, $\underline{Vu}$ is the average unbiased variance, and $\underline{Vb}$ is the average biased variance (We ignore the noise.)
- $\underline{Vu} - \underline{Vb}$ will be called "net variance"

Algorithms to Study

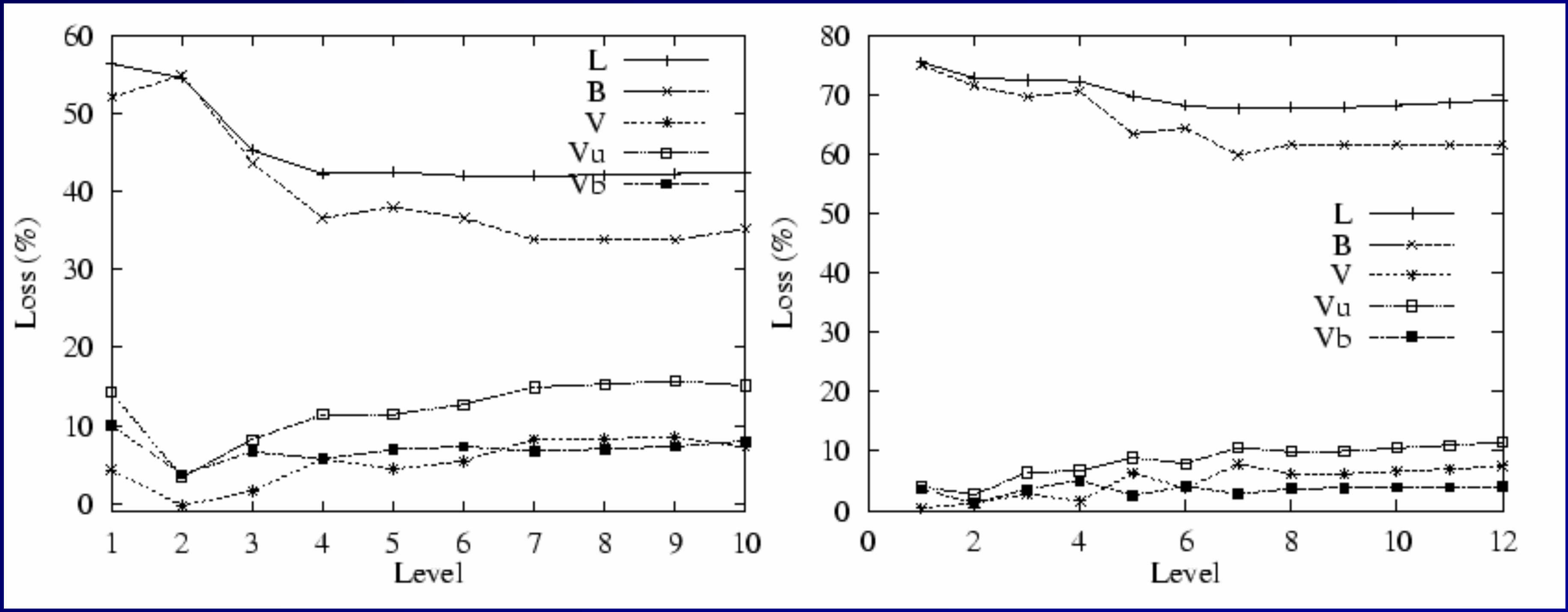
- K-nearest neighbors: What is the effect of K?
- Decision trees: What is the effect of pruning?
- Support Vector Machines: What is the effect of kernel width σ?

K-nearest neighbor (Domingos, 2000)



- Chess (left): Increasing K primarily reduces Vu
- Audiology (right): Increasing K primarily increases B.

Size of Decision Trees



- Glass (left), Primary tumor (right): deeper trees have lower B, higher Vu

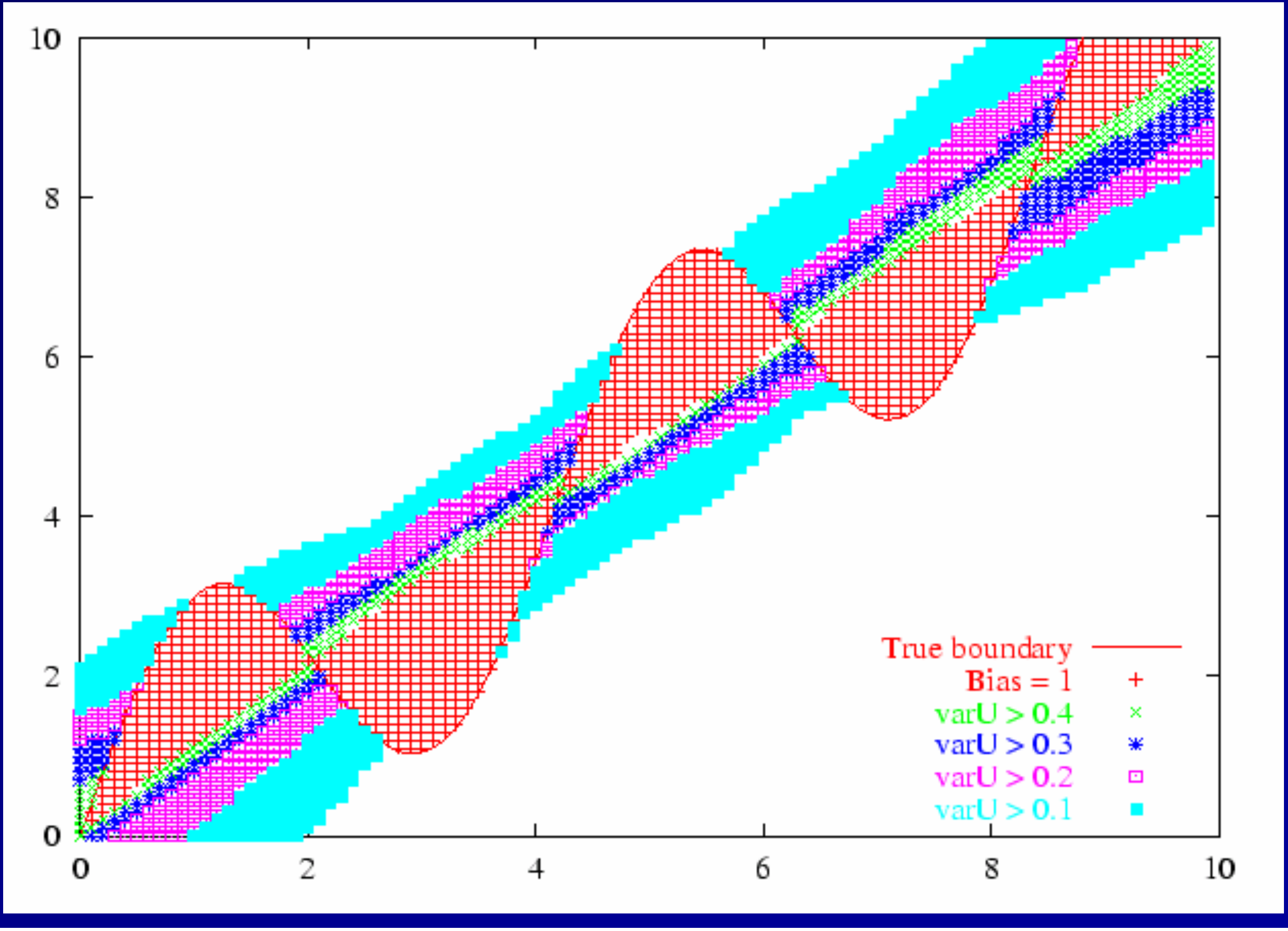
Example: 200 linear SVMs (training sets of size 20)

Error: 13.7%

Bias: 11.7%

Vu: 5.2%

Vb: 3.2%



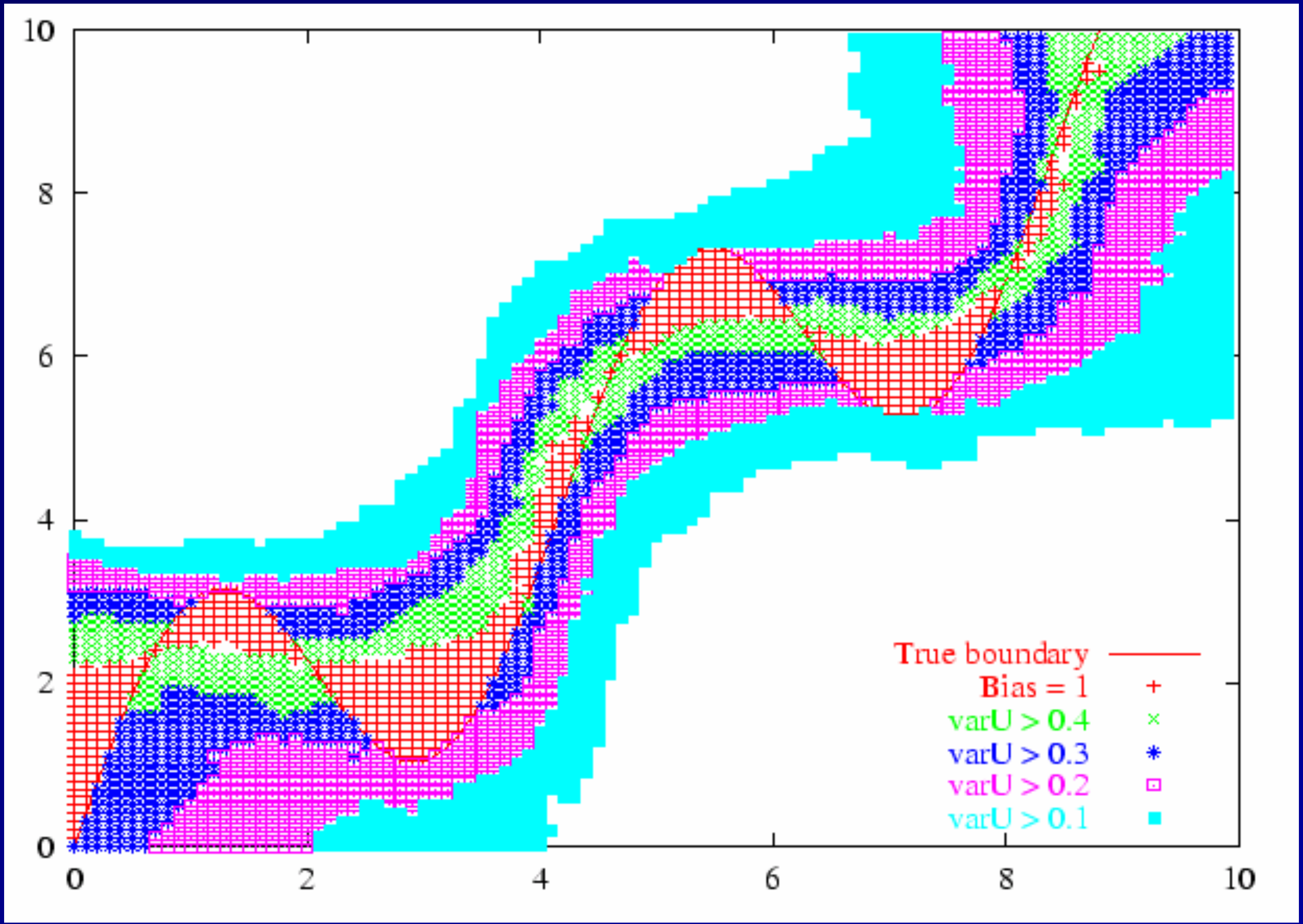
Example: 200 RBF SVMs
$\sigma = 5$

Error: 15.0%

Bias: 5.8%

Vu: 11.5%

Vb: 2.3%



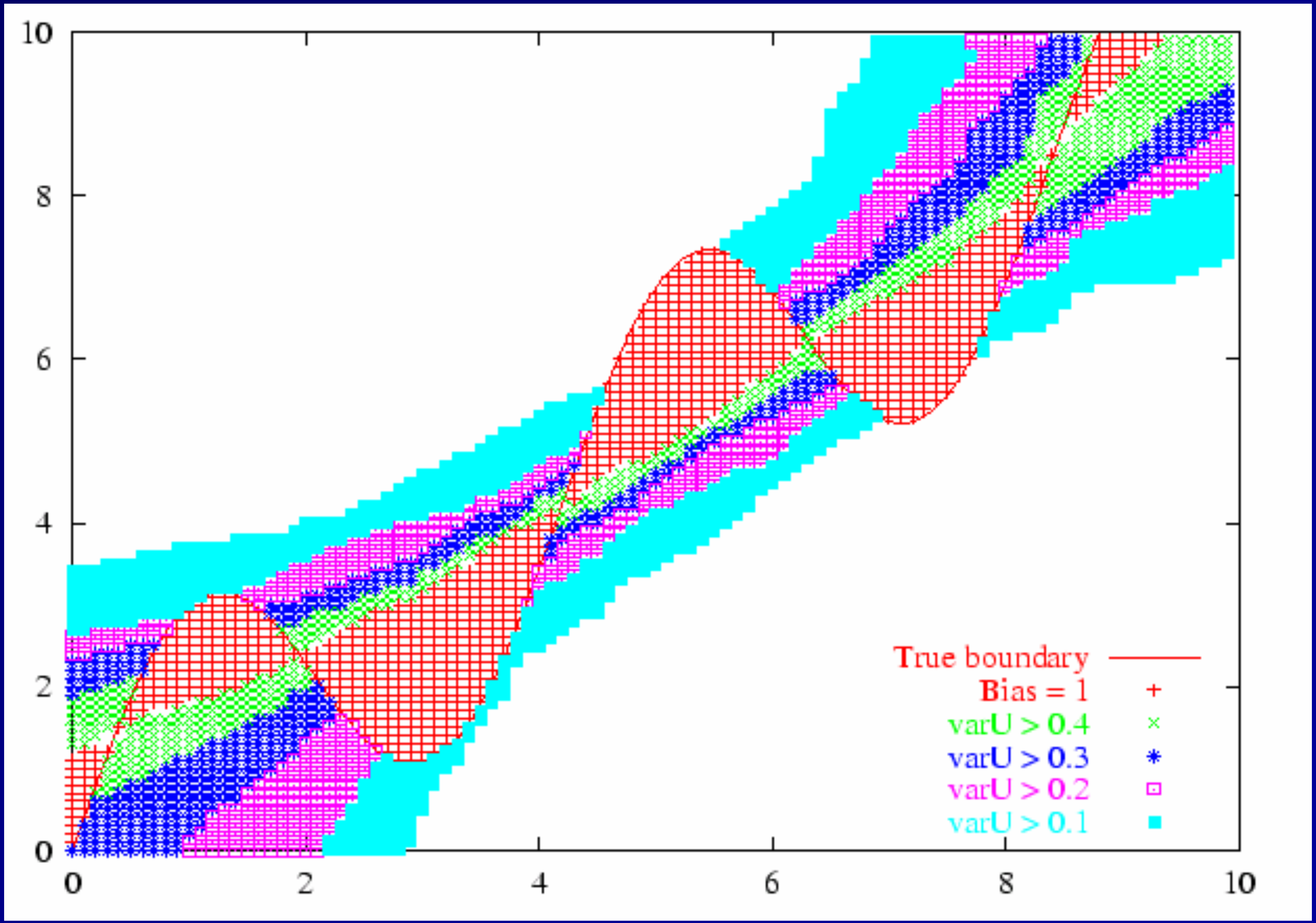
Example: 200 RBF SVMs
σ = 50

Error: 14.9%

Bias: 10.1%

Vu: 7.8%

Vb: 3.0%



SVM Bias and Variance

	Error	Bias	Var_U	Var_B	Net var	Tot var
linear	0.137	0.117	0.052	0.032	0.020	0.084
rbf $\sigma = 5$	0.150	0.058	0.115	0.023	0.092	0.137
rbf $\sigma = 50$	0.149	0.101	0.078	0.030	0.048	0.109

- Bias-Variance tradeoff controlled by σ
- Biased classifier (linear SVM) gives better results than a classifier that can represent the true decision boundary!

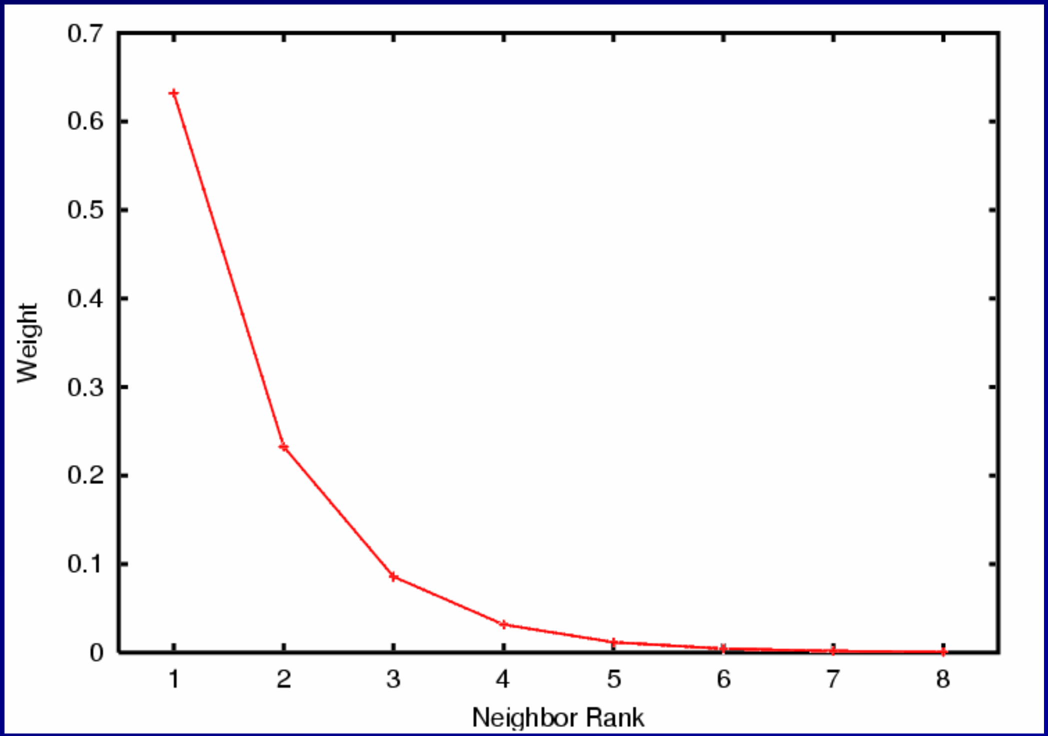
B/V Analysis of Bagging

- Under the bootstrap assumption, bagging reduces only variance
  - Removing Vu reduces the error rate
  - Removing Vb increases the error rate
- Therefore, bagging should be applied to low-bias classifiers, because then Vb will be small
- Reality is more complex!

Bagging Nearest Neighbor

Bagging first-nearest neighbor is equivalent (in the limit) to a weighted majority vote in which the k-th neighbor receives a weight of

exp(-(k-1)) – exp(-k)

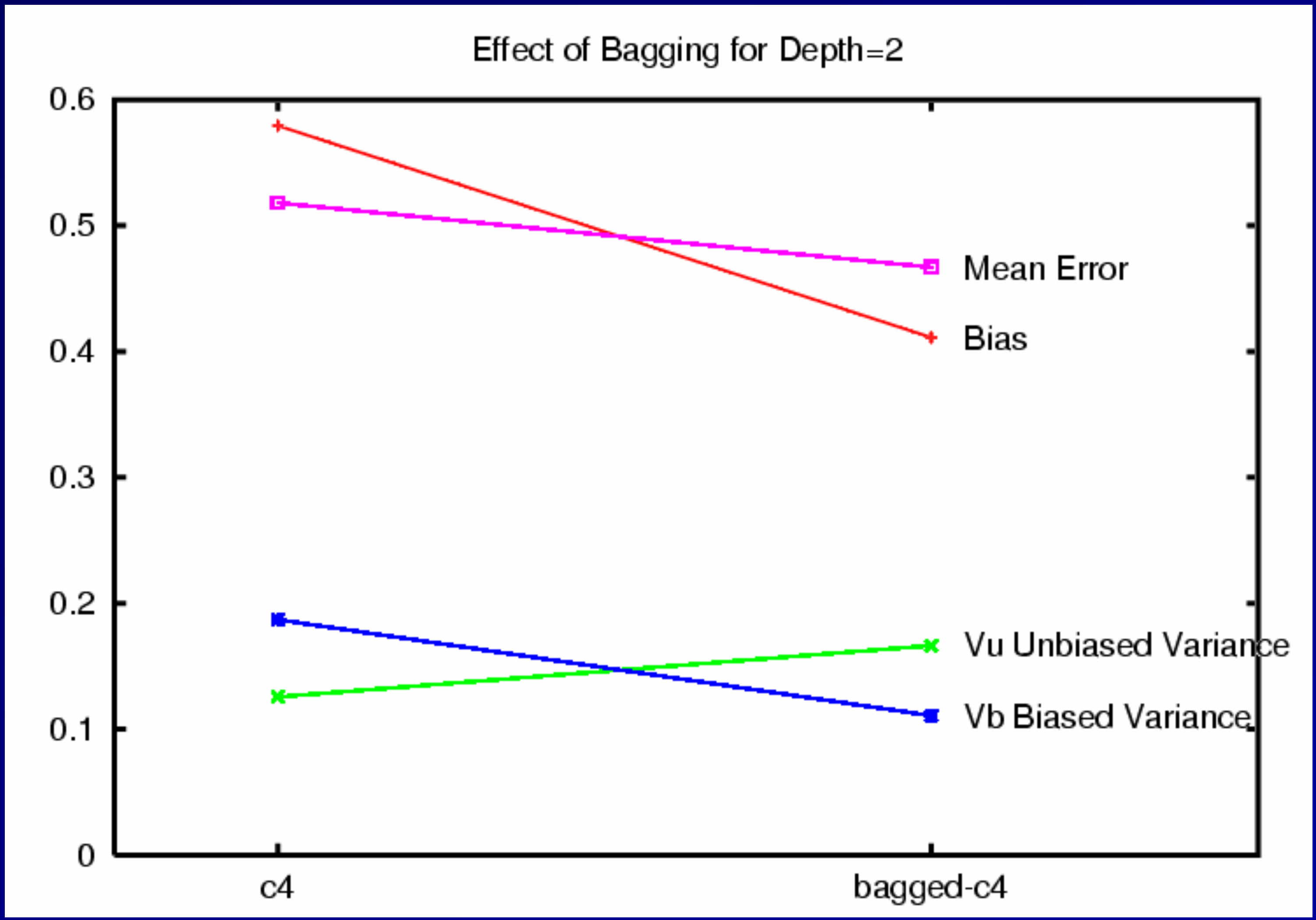


Since the first nearest neighbor gets more than half of the vote, it will always win this vote. Therefore, Bagging 1-NN is equivalent to 1-NN.

Bagging Decision Trees

- Consider unpruned trees of depth 2 on the Glass data set. In this case, the error is almost entirely due to bias
- Perform 30-fold bagging (replicated 50 times; 10-fold cross-validation)
- What will happen?

Bagging Primarily Reduces Bias!



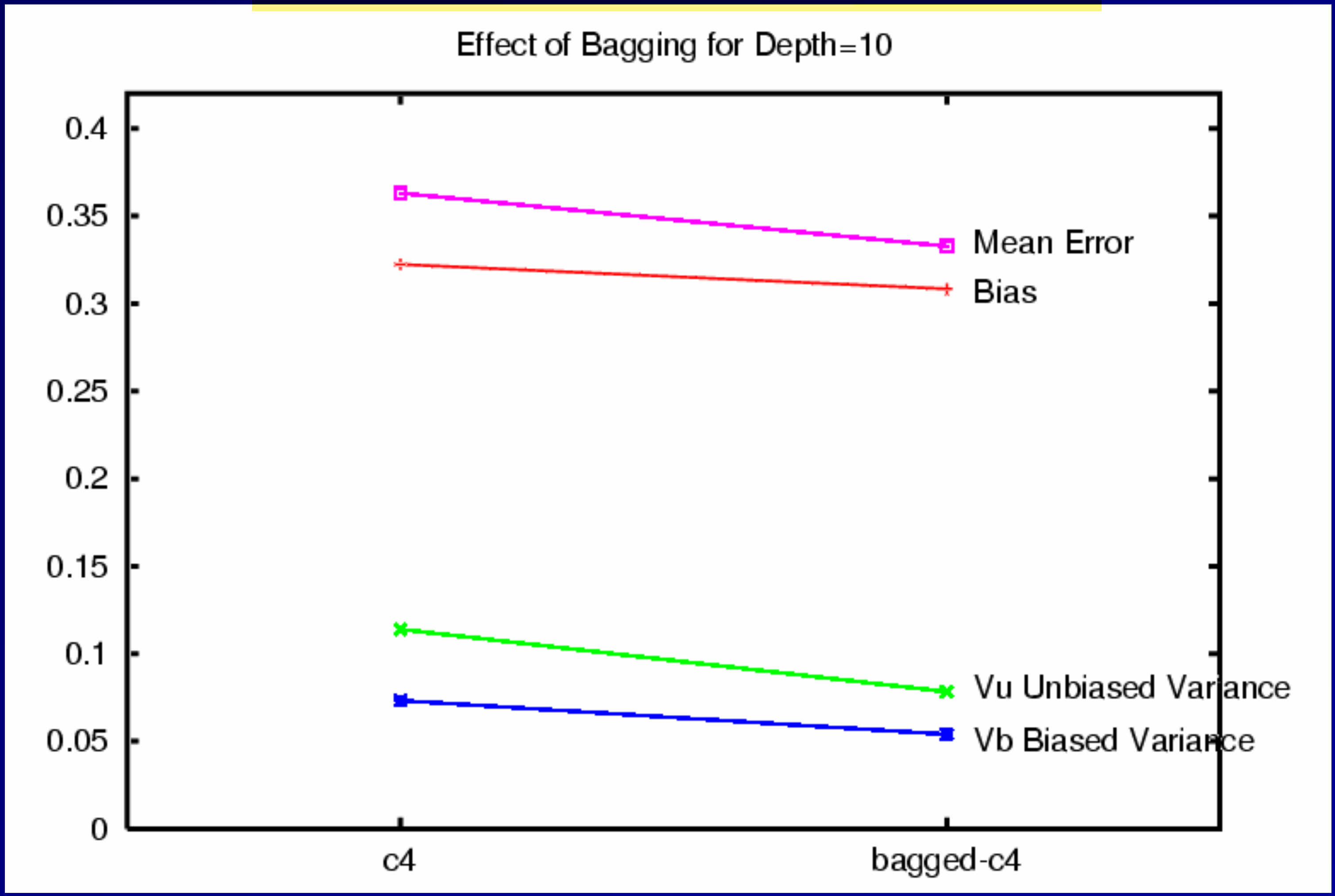
Questions

- Is this due to the failure of the bootstrap assumption in bagging?
- Is this due to the failure of the bootstrap assumption in estimating bias and variance?
- Should we also think of Bagging as a simple additive model that expands the range of representable classifiers?

Bagging Large Trees?

- Now consider unpruned trees of depth 10 on the Glass dataset. In this case, the trees have much lower bias.
- What will happen?

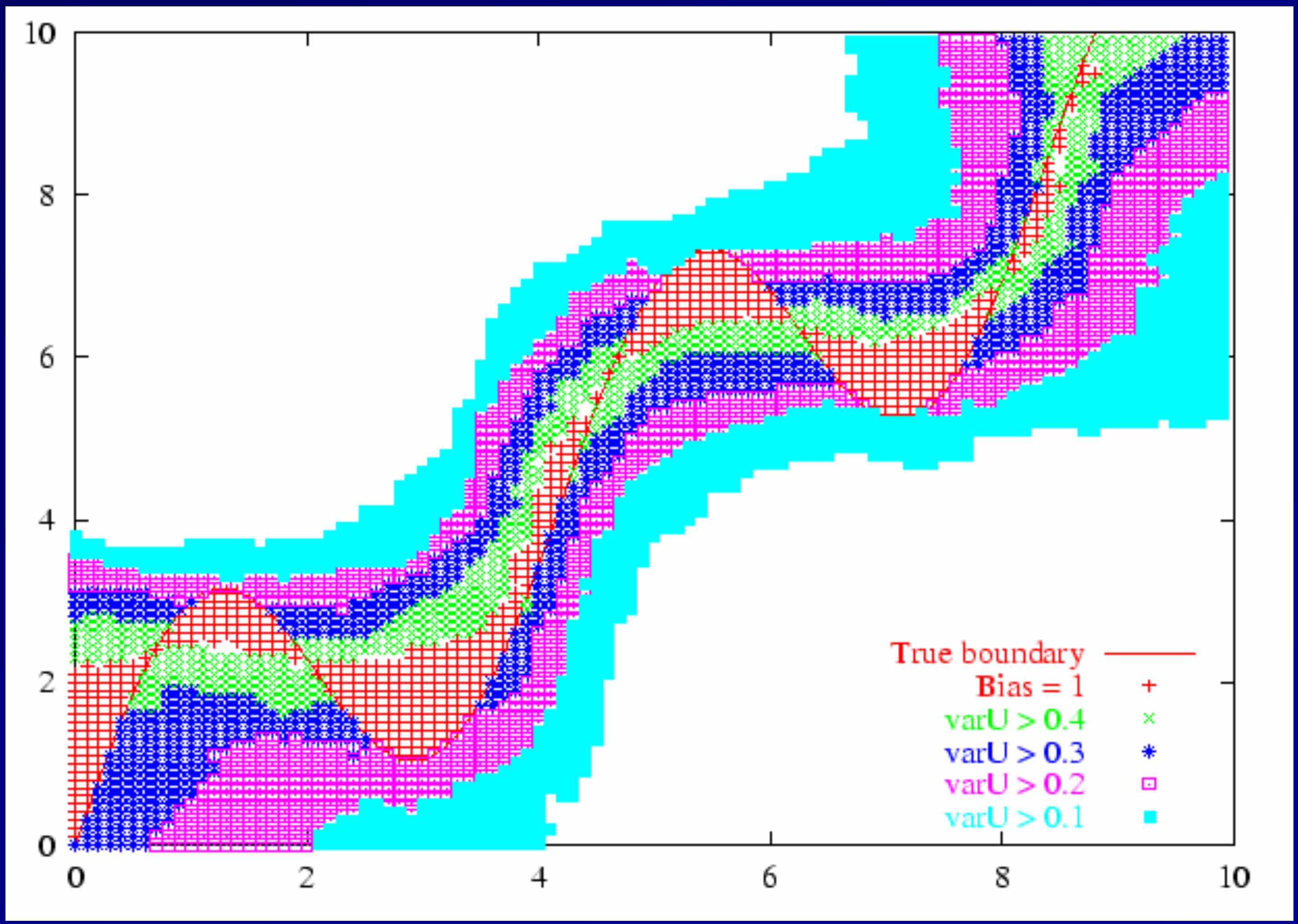
Answer: Bagging Primarily Reduces Variance



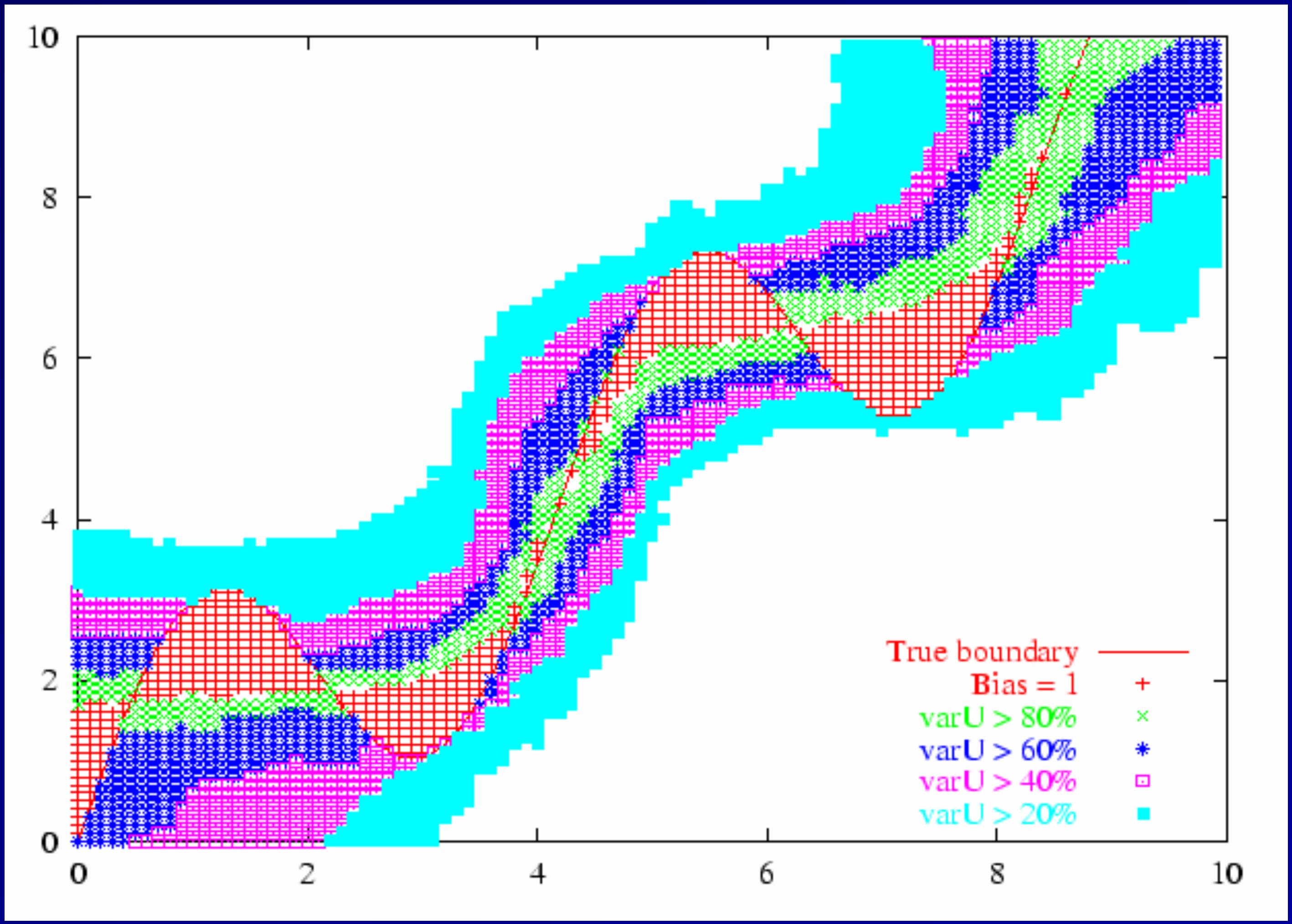
Bagging of SVMs

- We will choose a low-bias, high-variance SVM to bag: RBF SVM with σ=5

RBF SVMs again: $\sigma = 5$



Effect of 30-fold Bagging: Variance is Reduced



Effects of 30-fold Bagging

	Error	Bias	Var_U	Var_B	Net var	Tot var
rbf $\sigma = 5$	0.150	0.058	0.115	0.023	0.092	0.137
bagged rbf $\sigma = 5$	0.145	0.063	0.105	0.023	0.082	0.128

- Vu is decreased by 0.010; Vb is unchanged
- Bias is increased by 0.005
- Error is reduced by 0.005

A Formal View of Boosting

- given training set $(x_1, y_1), \ldots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \ldots, T$:
  - construct distribution D_t on $\{1, \ldots, m\}$
  - find weak hypothesis ("rule of thumb")

    $$h_t : X \to \{-1, +1\}$$

    with small error ϵ_t on D_t:

    $$\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$$
- output final hypothesis H_{final}

AdaBoost

[Freund & Schapire]

- constructing D_t:
  - $D_1(i) = 1/m$
  - given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

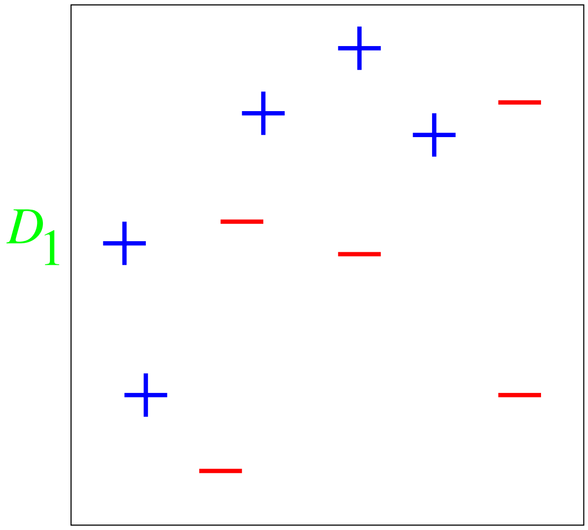
$$= \frac{D_t(i)}{Z_t} \cdot \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where $Z_t =$ normalization constant

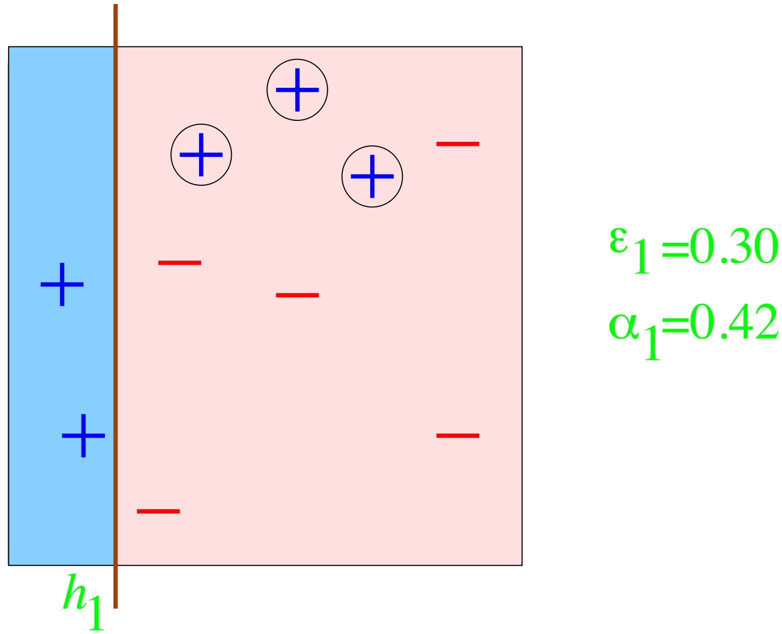
$$\alpha_t = \tfrac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) > 0$$

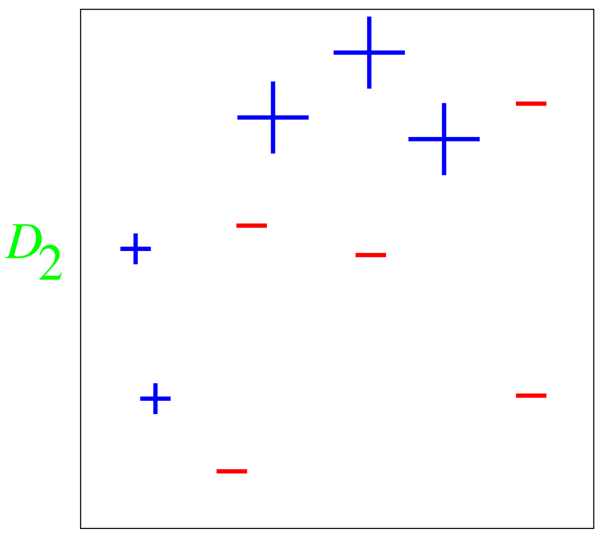
- final hypothesis:
  - $H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x) \right)$

Toy Example

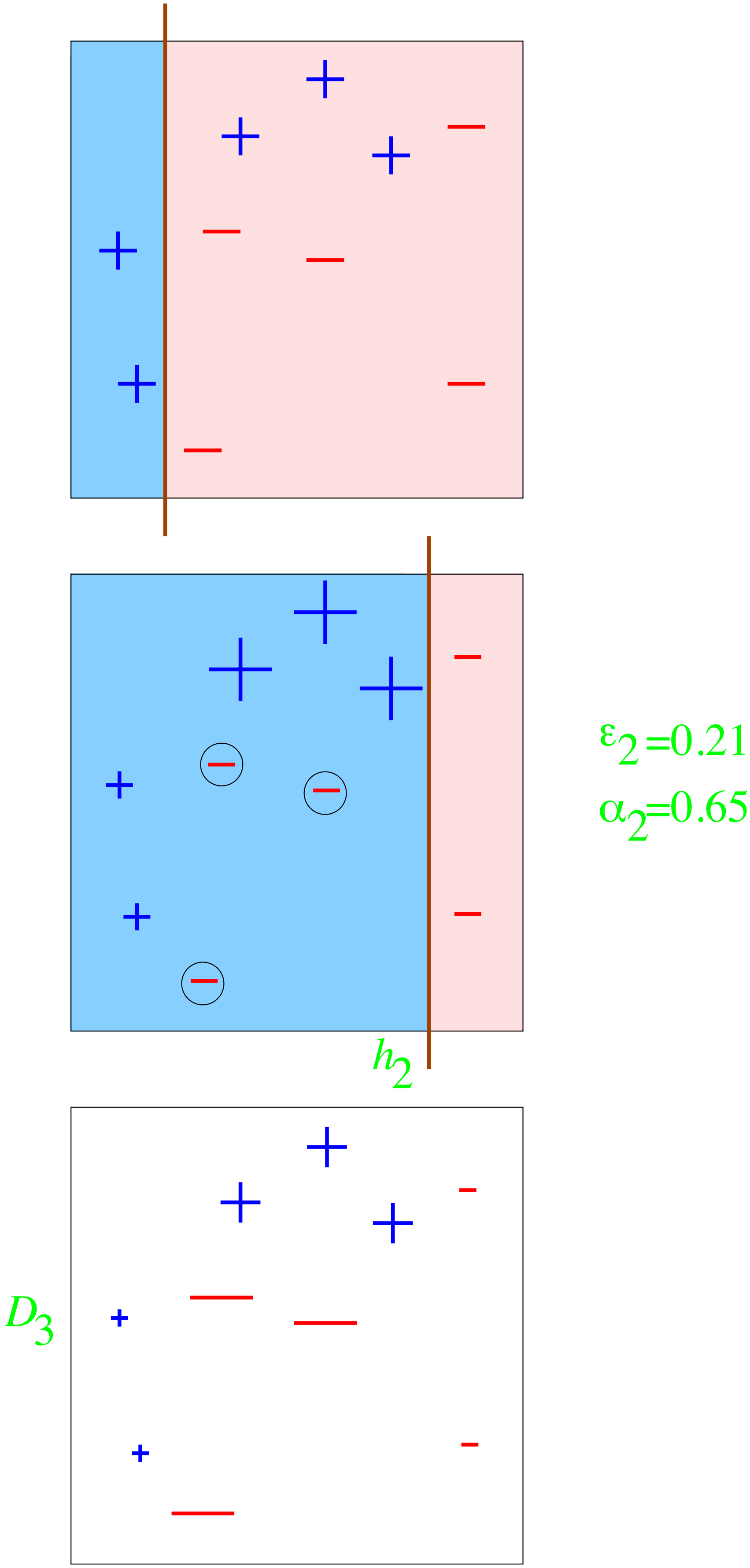


Round 1

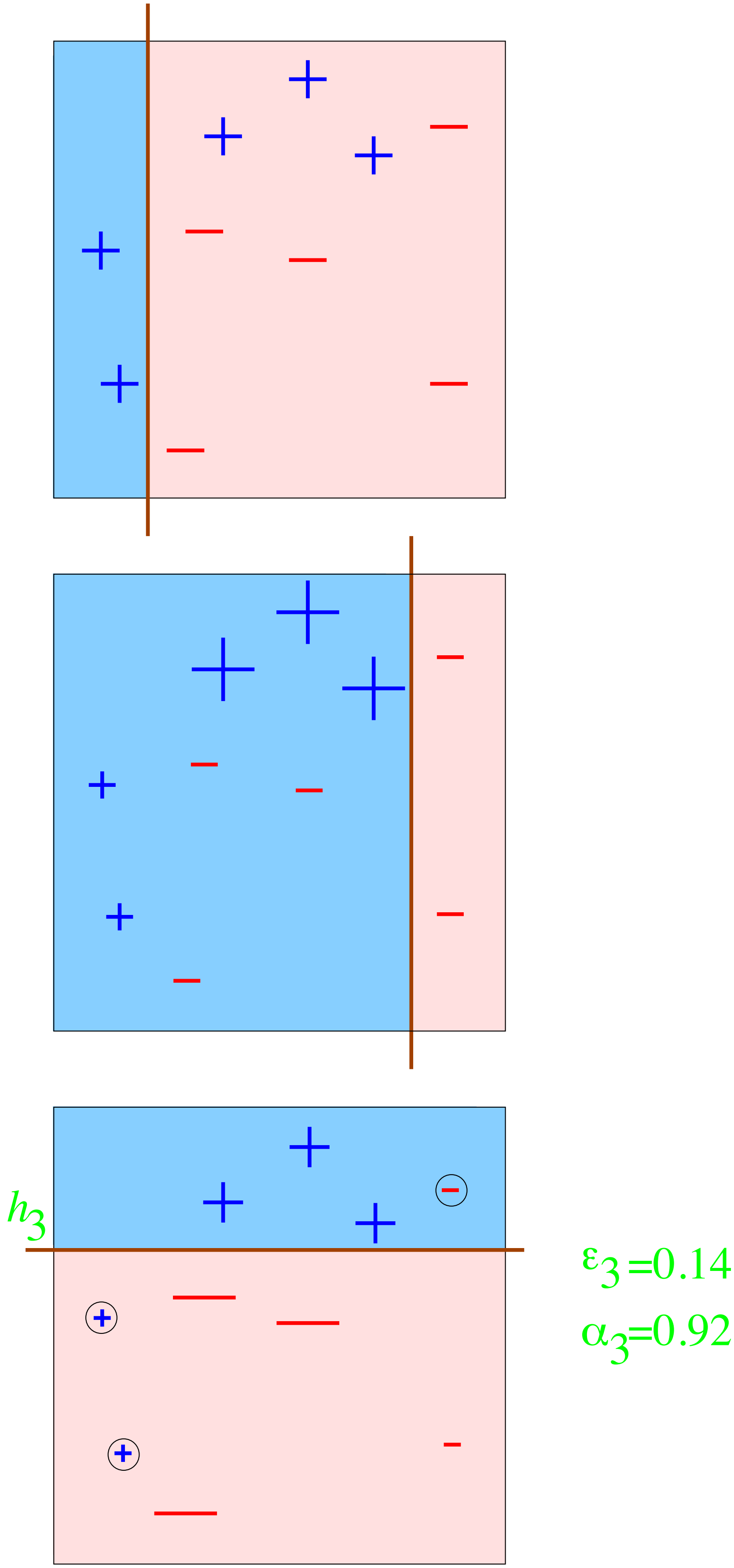




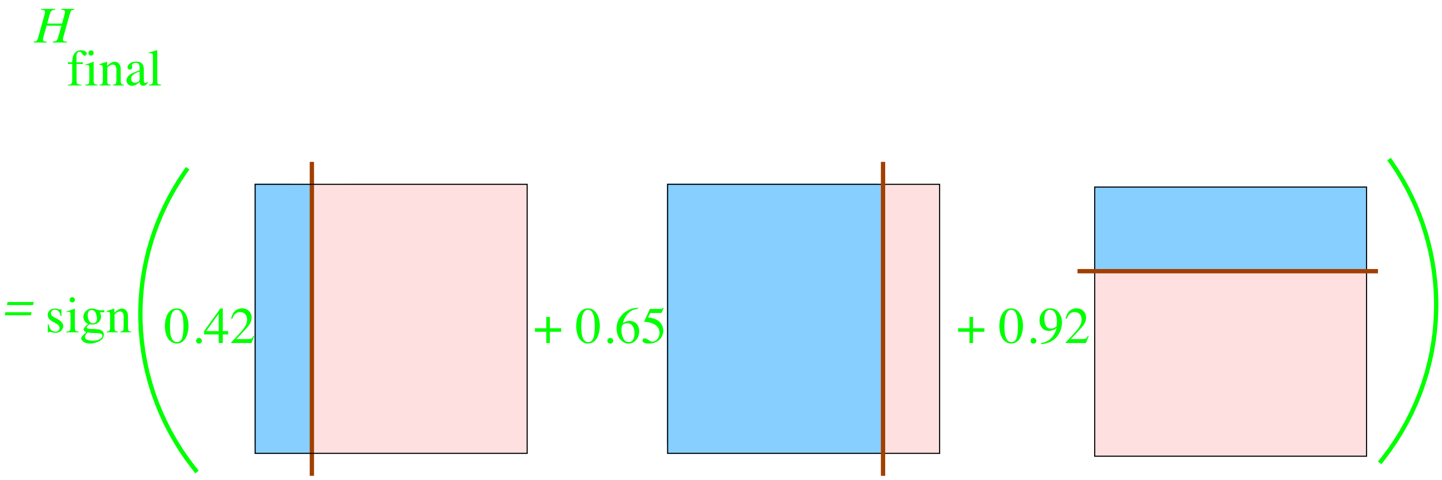
Round 2

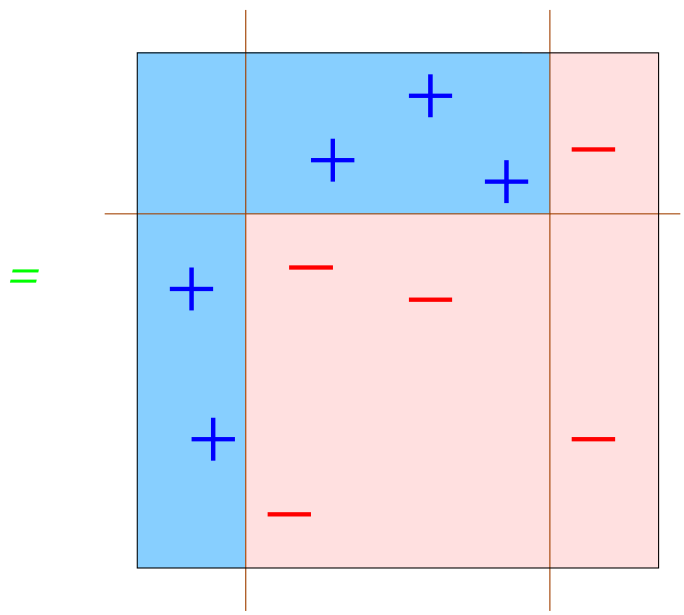


Round 3



Final Hypothesis



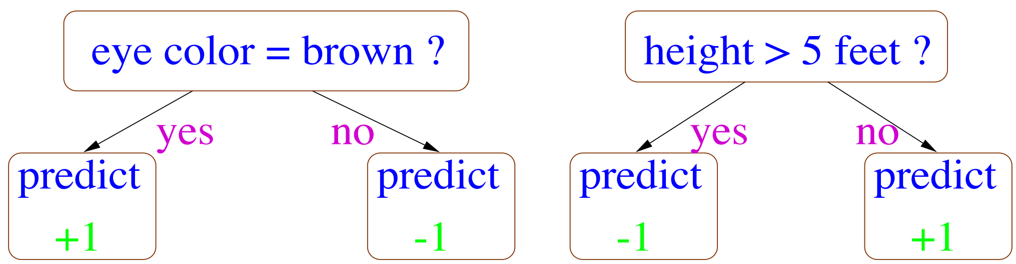


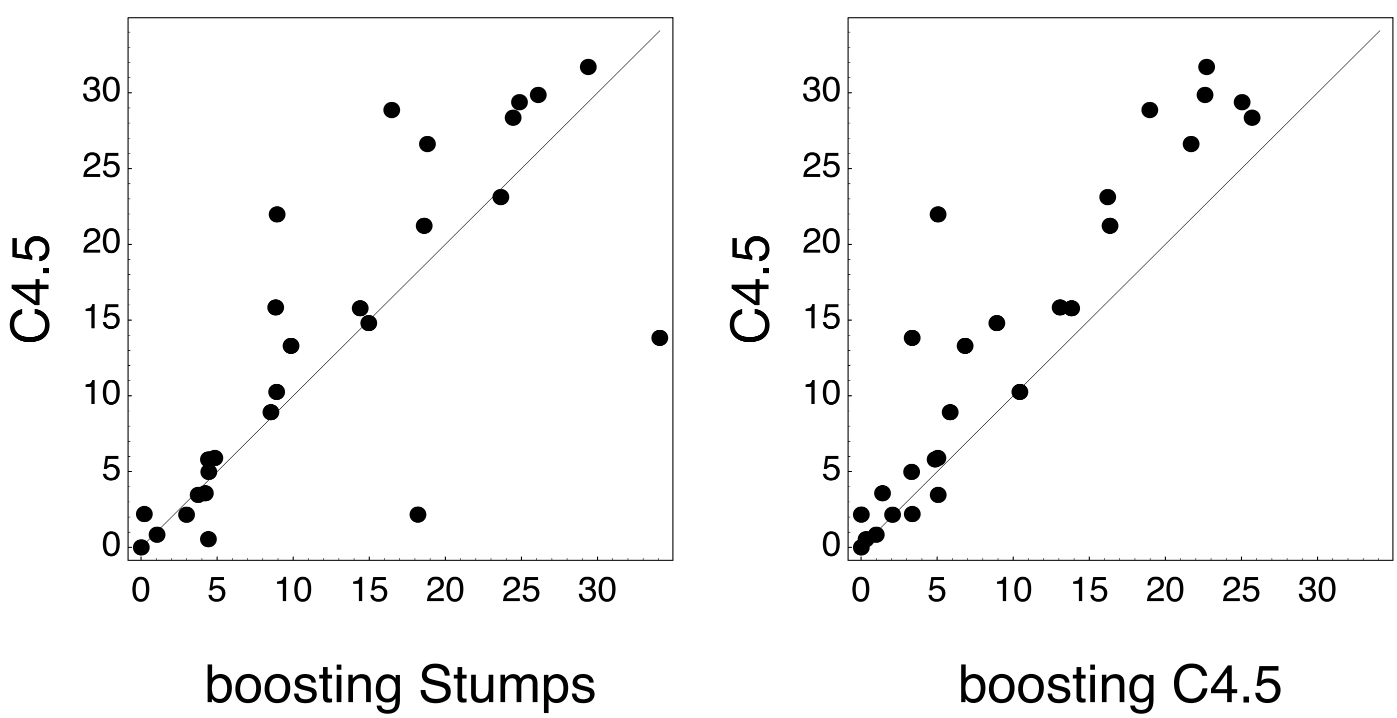
* See demo at
`www.research.att.com/~yoav/adaboost`

UCI Experiments

[Freund & Schapire]

- tested AdaBoost on UCI benchmarks
- used:
  - C4.5 (Quinlan's decision tree algorithm)
  - "decision stumps": very simple rules of thumb that test on single attributes





Multiclass Problems

- say $y \in Y = \{1, \ldots, k\}$
- direct approach (AdaBoost.M1):

$$h_t : X \to Y$$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$H_{\text{final}}(x) = \arg\max_{y \in Y} \sum_{t : h_t(x) = y} \alpha_t$$

- can prove same bound on error <u>if</u> $\forall t : \epsilon_t \leq 1/2$
  - in practice, not usually a problem for "strong" weak learners (e.g., C4.5)
  - significant problem for "weak" weak learners (e.g., decision stumps)

Reducing to Binary Problems

[Schapire & Singer]

- e.g.:
  - say possible labels are $\{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}\}$
  - each training example replaced by five $\{-1, +1\}$-labeled examples:

$$x \;,\; \mathrm{c} \rightarrow \begin{cases} (x, \mathrm{a}) \;,\; -1 \\ (x, \mathrm{b}) \;,\; -1 \\ (x, \mathrm{c}) \;,\; +1 \\ (x, \mathrm{d}) \;,\; -1 \\ (x, \mathrm{e}) \;,\; -1 \end{cases}$$

AdaBoost.MH

- formally:

$$h_t : X \times Y \to \{-1, +1\} (\text{or } \mathbb{R})$$

$$D_{t+1}(i, y) = \frac{D_t(i, y)}{Z_t} \cdot \exp(-\alpha_t \, v_i(y) \, h_t(x_i, y))$$

where $v_i(y) = \begin{cases} +1 & \text{if } y_i = y \\ -1 & \text{if } y_i \neq y \end{cases}$

$$H_{\text{final}}(x) = \arg\max_{y \in Y} \sum_t \alpha_t h_t(x, y)$$

- can prove:

$$\text{training error}(H_{\text{final}}) \leq \frac{k}{2} \cdot \prod Z_t$$

Using Output Codes

[Schapire & Singer]

- alternative: reduce to "random" binary problems
- choose "code word" for each label

	π_1	π_2	π_3	π_4
a	$-$	$+$	$-$	$+$
b	$-$	$+$	$+$	$-$
c	$+$	$-$	$-$	$+$
d	$+$	$-$	$+$	$+$
e	$-$	$+$	$-$	$-$

- each training example mapped to one example per column

$$x \ , \ \mathrm{c} \rightarrow \begin{cases} (x, \pi_1) \ , \ +1 \\ (x, \pi_2) \ , \ -1 \\ (x, \pi_3) \ , \ -1 \\ (x, \pi_4) \ , \ +1 \end{cases}$$

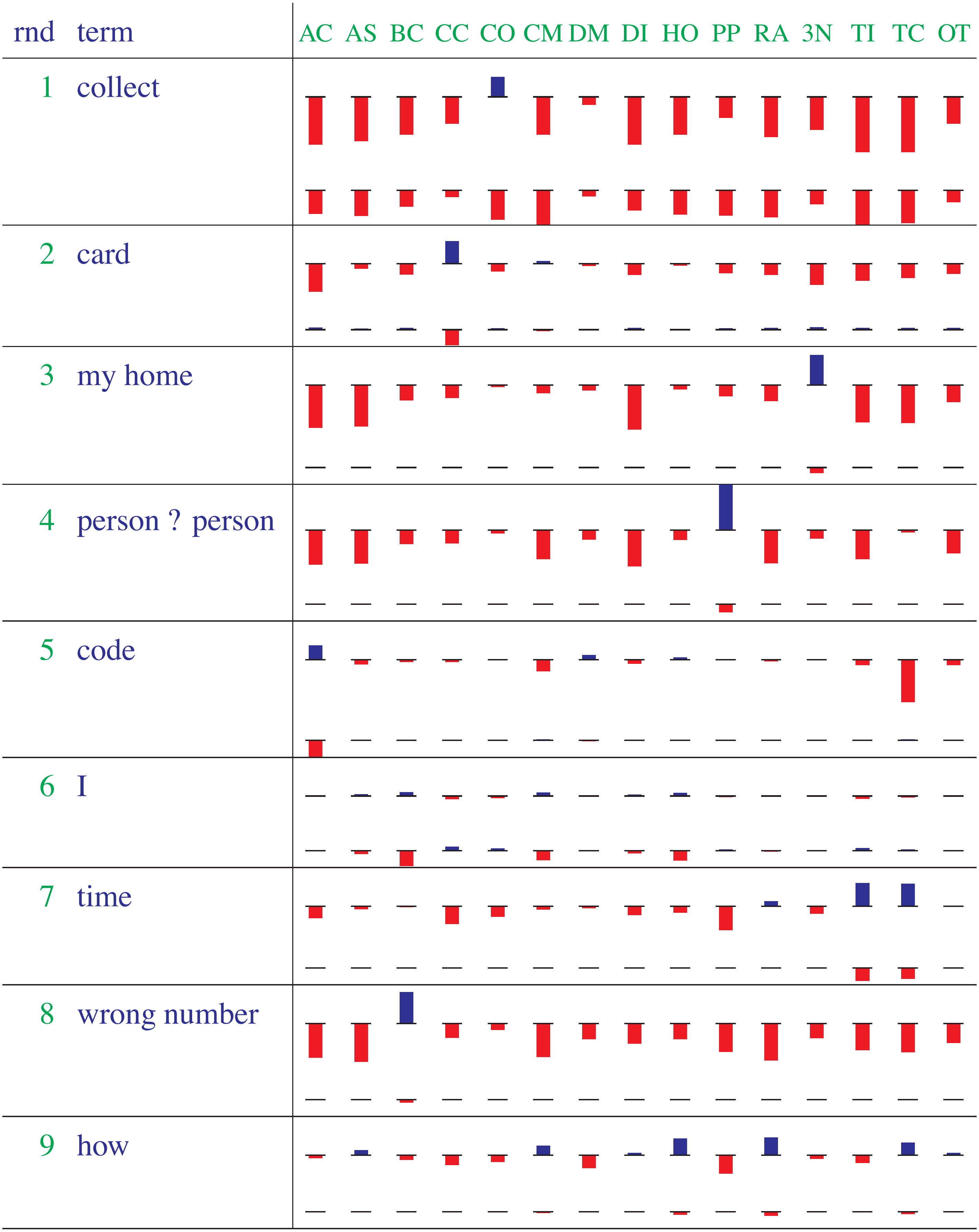
- to classify new example x:
  - evaluate hypothesis on $(x, \pi_1), \ldots, (x, \pi_4)$
  - choose label "most consistent" with results
- training error bounds independent of # of classes
- may be more efficient for very large # of classes

Example: Boosting for Text Categorization

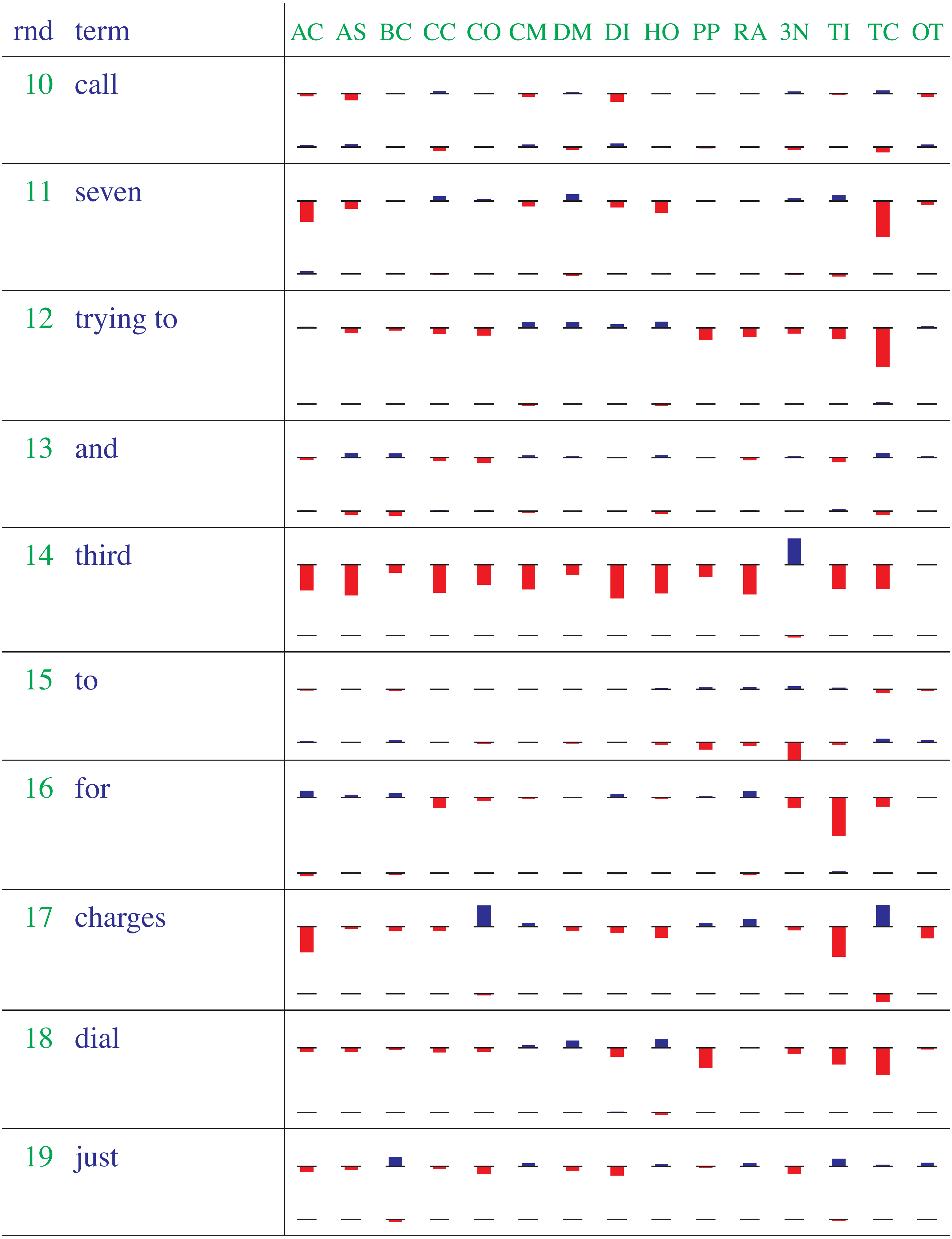
[Schapire & Singer]

- weak hypotheses: very simple weak hypotheses that test on simple patterns, namely, (sparse) n-grams
  - find parameter α_t and rule h_t of given form which minimize Z_t
  - use efficiently implemented exhaustive search
- "How may I help you" data:
  - 7844 training examples (hand-transcribed)
  - 1000 test examples (both hand-transcribed and from speech recognizer)
  - categories: AreaCode, AttService, BillingCredit, CallingCard, Collect, Competitor, DialForMe, Directory, HowToDial, PersonToPerson, Rate, ThirdNumber, Time, TimeCharge, Other.

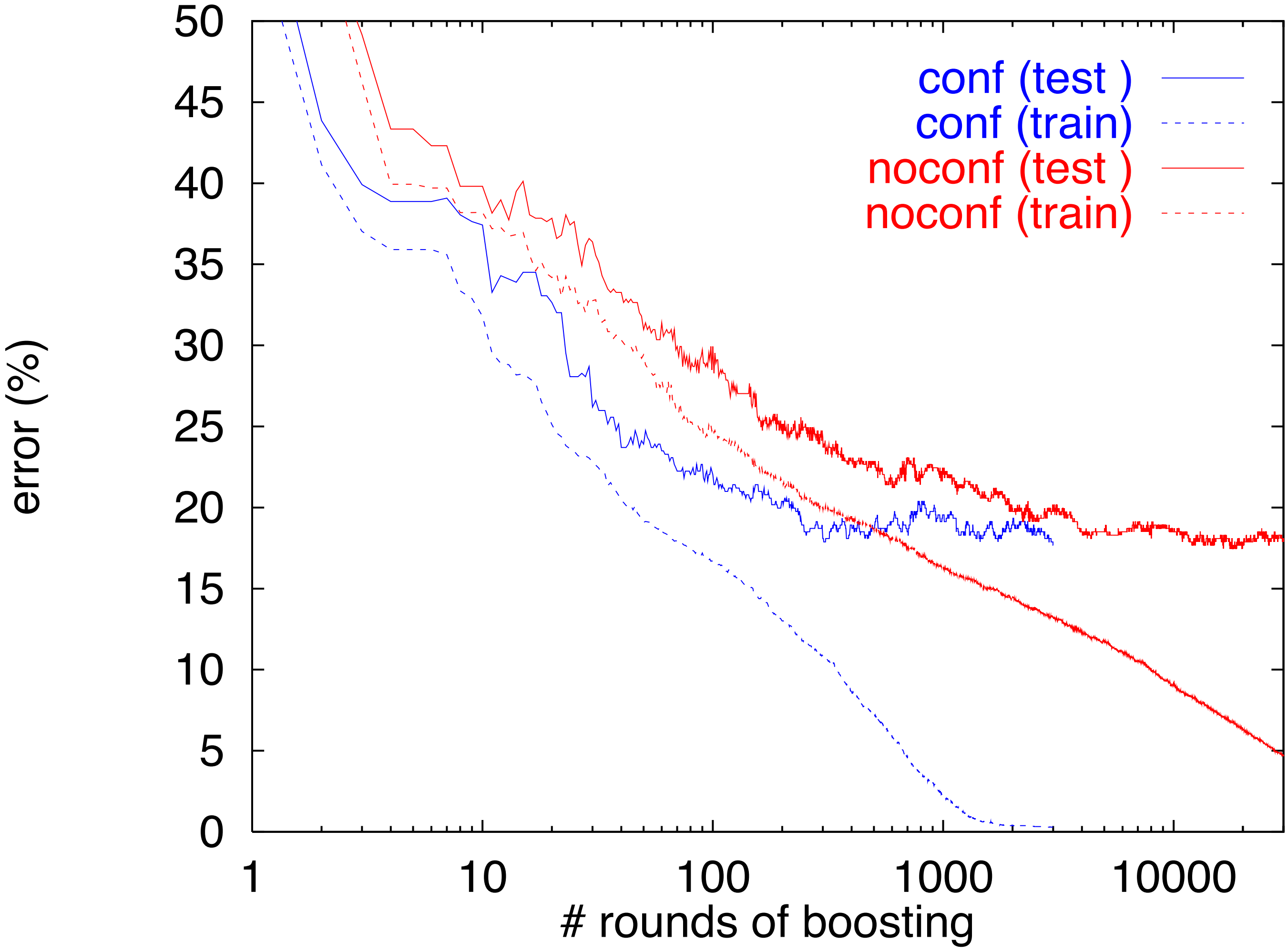
Weak Hypotheses



More Weak Hypotheses



Learning Curves

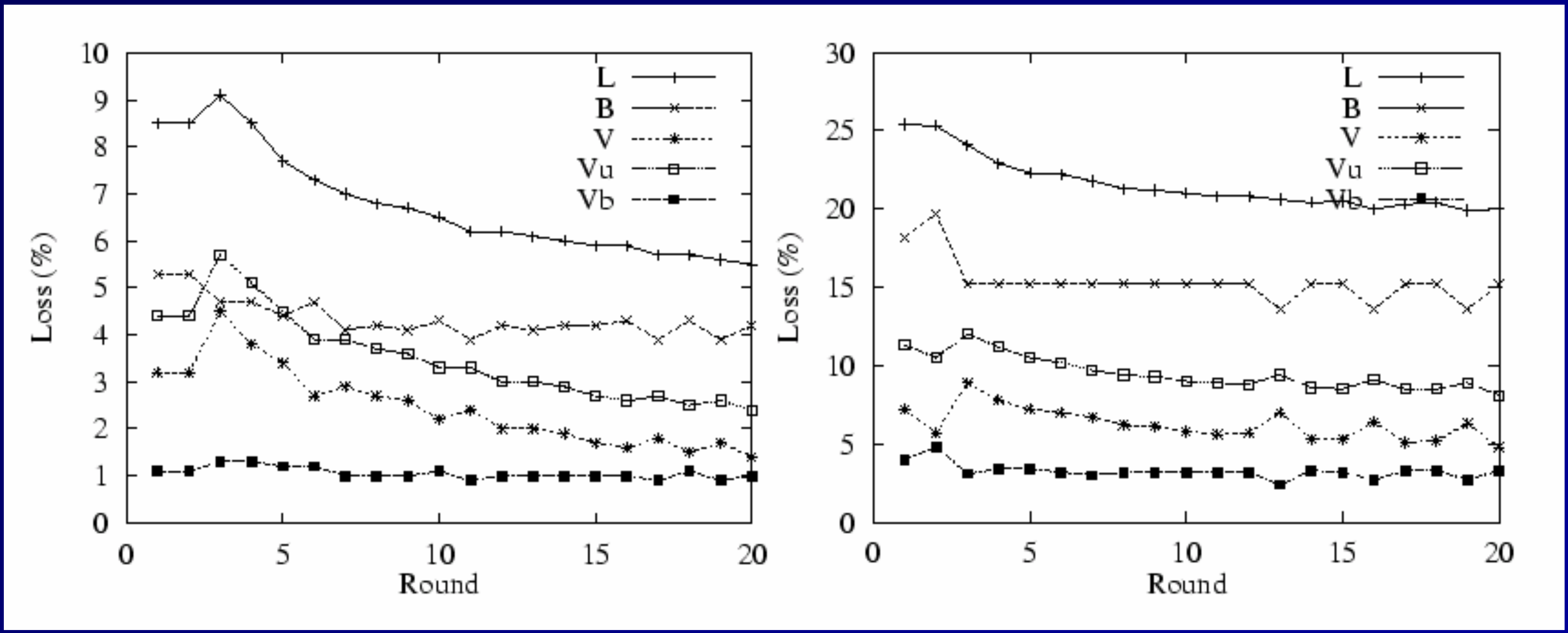


- test error reaches 20% for the first time on round...
  - 1,932 without confidence ratings
  - 191 with confidence ratings
- test error reaches 18% for the first time on round...
  - 10,909 without confidence ratings
  - 303 with confidence ratings

Bias-Variance Analysis of Boosting

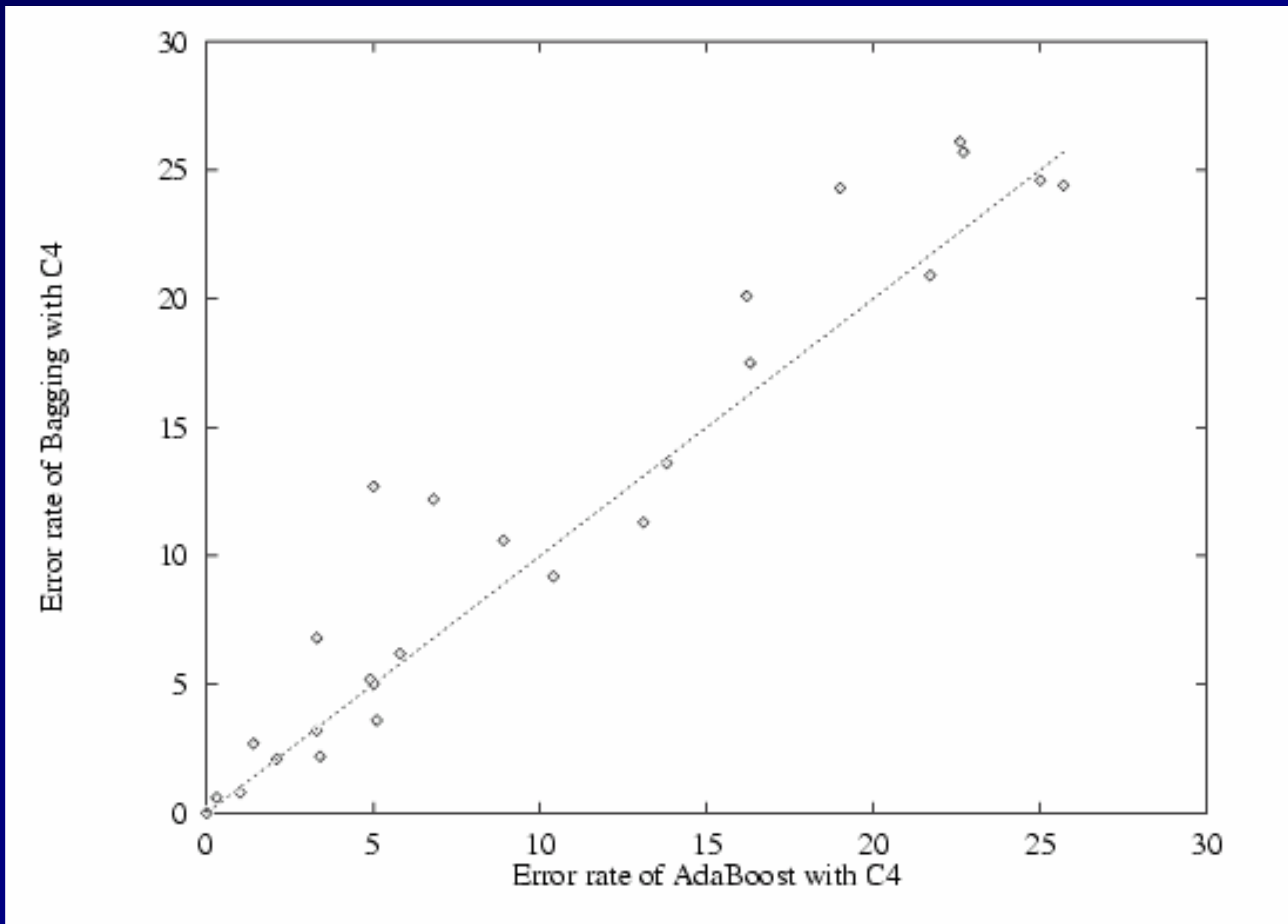
- Boosting seeks to find a weighted combination of classifiers that fits the data well
- Prediction: Boosting will primarily act to reduce bias

Boosting DNA splice (left) and Audiology (right)



Early iterations reduce bias. Later iterations also reduce variance

Boosting vs Bagging (Freund & Schapire)



Review and Conclusions

- For regression problems (squared error loss), the expected error rate can be decomposed into
  - $\text{Bias}(x^*)^2 + \text{Variance}(x^*) + \text{Noise}(x^*)$
- For classification problems (0/1 loss), the expected error rate depends on whether bias is present:
  - if $B(x^*) = 1$: $B(x^*) - [V(x^*) + N(x^*) - 2\,V(x^*)\,N(x^*)]$
  - if $B(x^*) = 0$: $B(x^*) + [V(x^*) + N(x^*) - 2\,V(x^*)\,N(x^*)]$
  - or $B(x^*) + Vu(x^*) - Vb(x^*)$ [ignoring noise]

Review and Conclusions (2)

- For classification problems with log loss, the expected loss can be decomposed into noise + bias + variance

$$E[\ KL(y, h)\ ] = H(p) + KL(p, \underline{h}) + E_S[\ KL(\underline{h}, h)\ ]$$

Sources of Bias and Variance

- Bias arises when the classifier cannot represent the true function – that is, the classifier underfits the data
- Variance arises when the classifier overfits the data
- There is often a tradeoff between bias and variance

Effect of Algorithm Parameters on Bias and Variance

- k-nearest neighbor: increasing k typically increases bias and reduces variance
- decision trees of depth D: increasing D typically increases variance and reduces bias
- RBF SVM with parameter σ: increasing σ increases bias and reduces variance

Effect of Bagging

- If the bootstrap replicate approximation were correct, then bagging would reduce variance without changing bias
- In practice, bagging can reduce both bias and variance
  - For high-bias classifiers, it can reduce bias (but may increase Vu)
  - For high-variance classifiers, it can reduce variance

Effect of Boosting

- In the early iterations, boosting is primary a bias-reducing method
- In later iterations, it appears to be primarily a variance-reducing method