Ensembles of learners

- not a learning technique on its own, but a method in which a family of [“weakly”] learning agents (simple learners) is used for learning
- based on the fact that multiple classifiers that disagree with one another can be *together* more accurate than its component classifiers
- if there are L classifiers, each with an error rate < 1/2, and the errors are independent, then the prob. that the majority vote is wrong is the area under binomial distribution for more than L/2 hypotheses

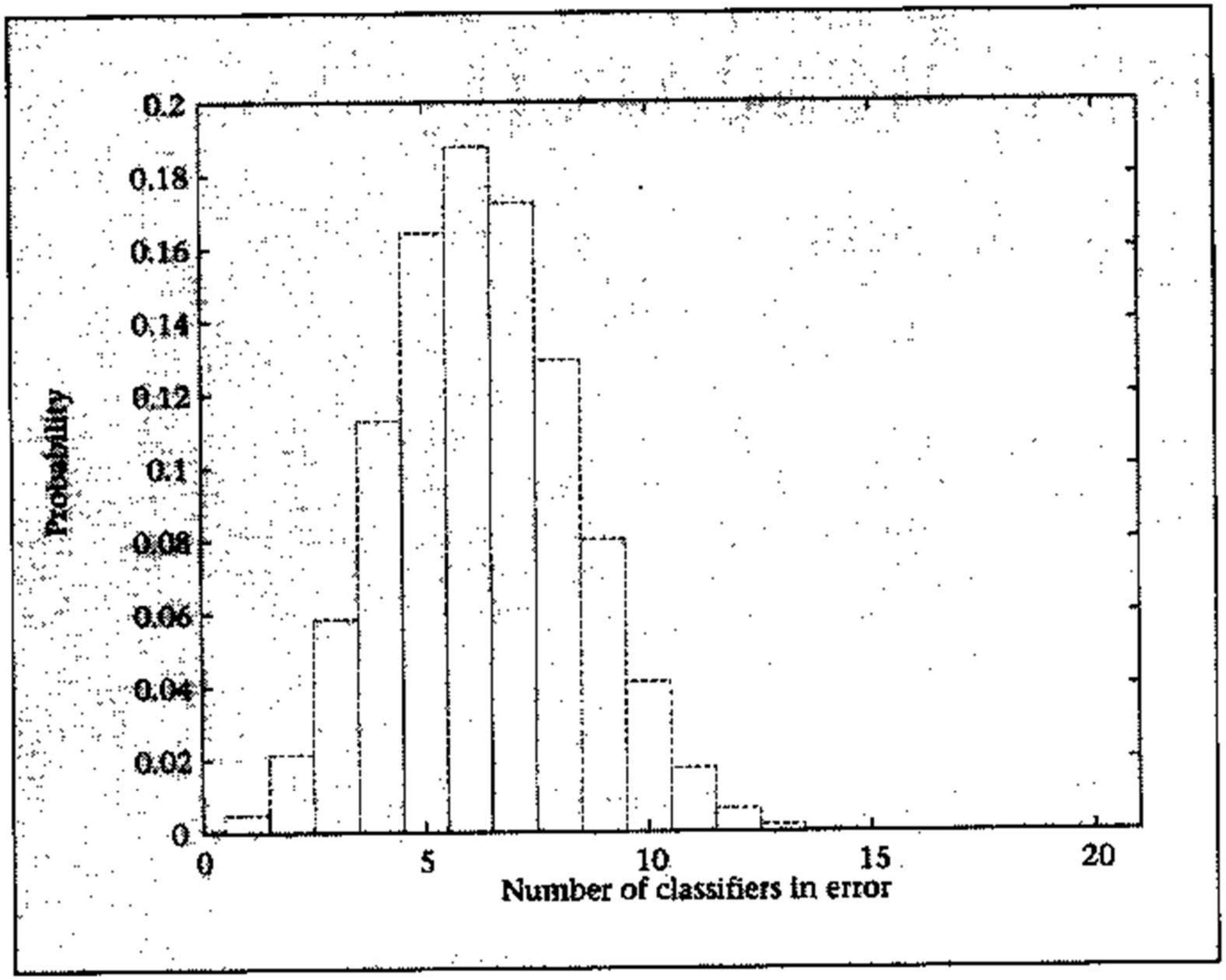


Figure 1. The Probability That Exactly ℓ (of 21) Hypotheses Will Make an Error, Assuming Each Hypothesis Has an Error Rate of 0.3 and Makes Its Errors Independently of the Other Hypotheses.

Boosting - idea

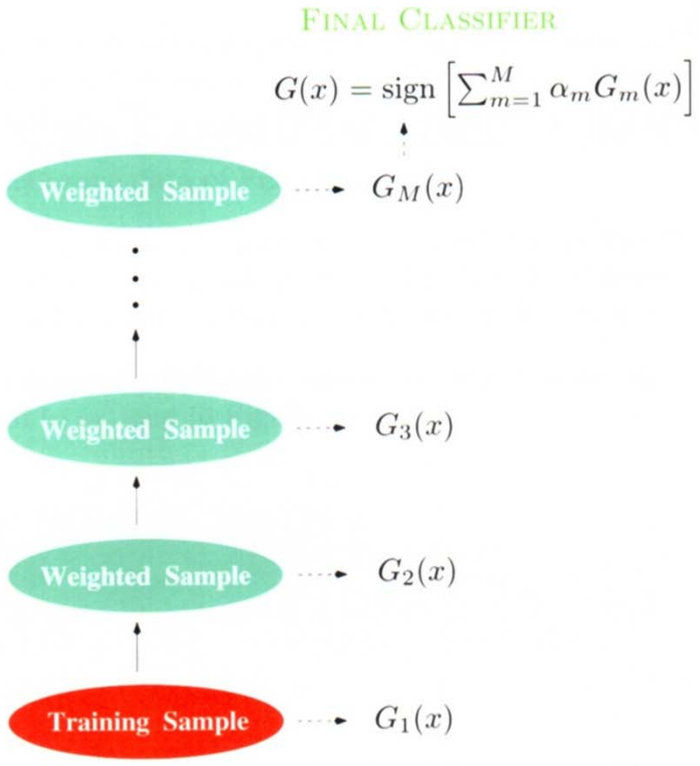


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

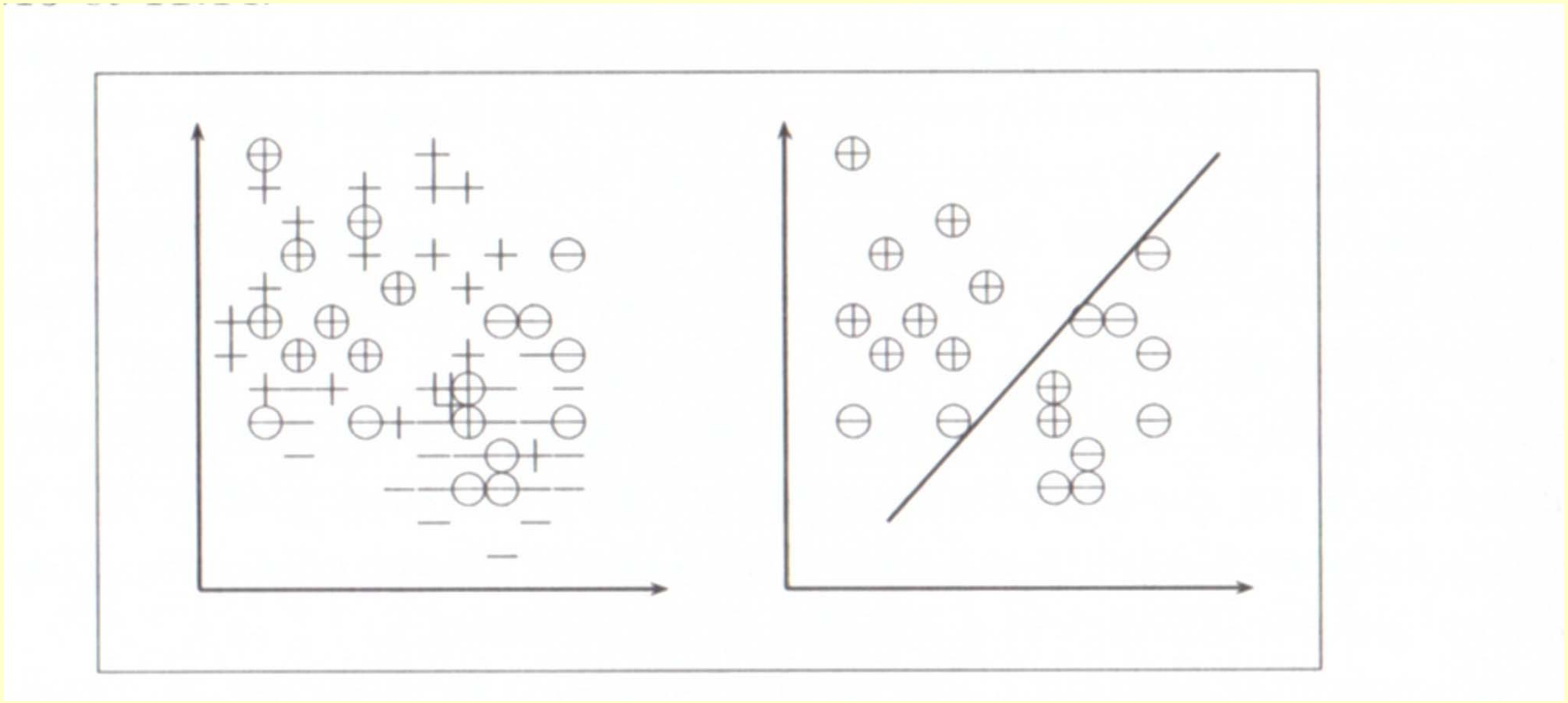
From “Elements of Statistical learning”, Hastie, Tibshirani, Friedman [HTF]

Boosting as ensemble of learners

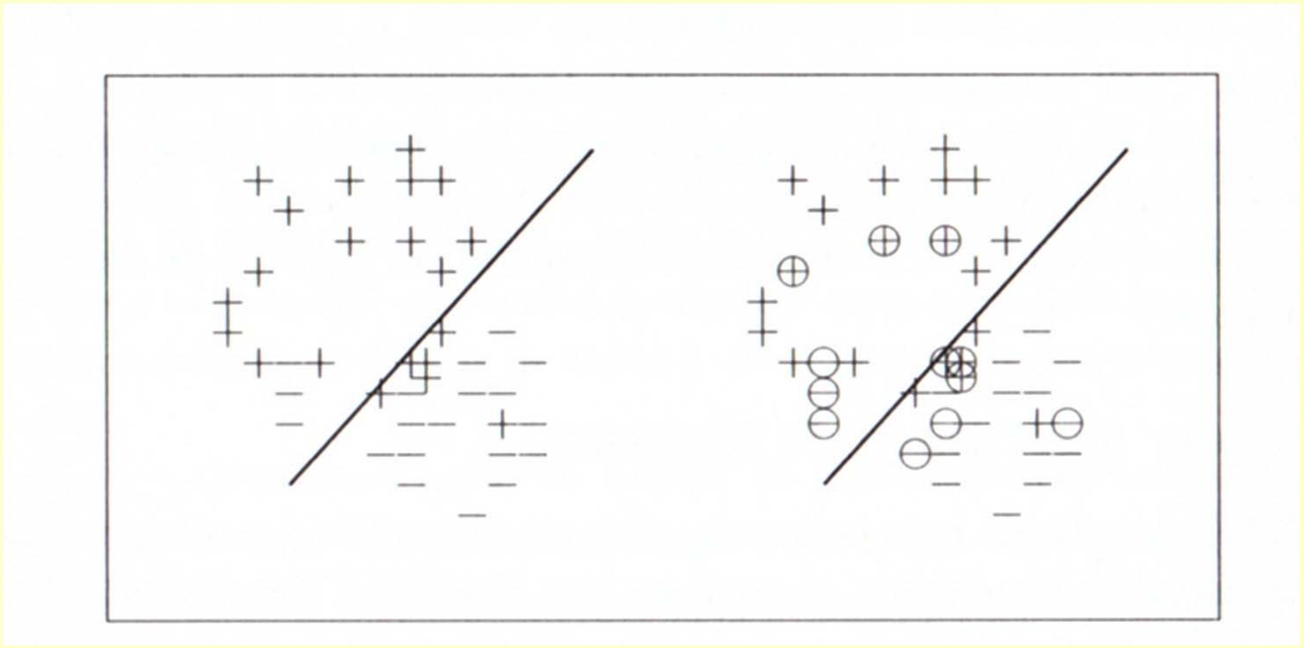
- The very idea: focus on ‘difficult’ parts of the example space
- Train a number of classifiers
- Combine their decision in a weighed manner

Boosting - illustration

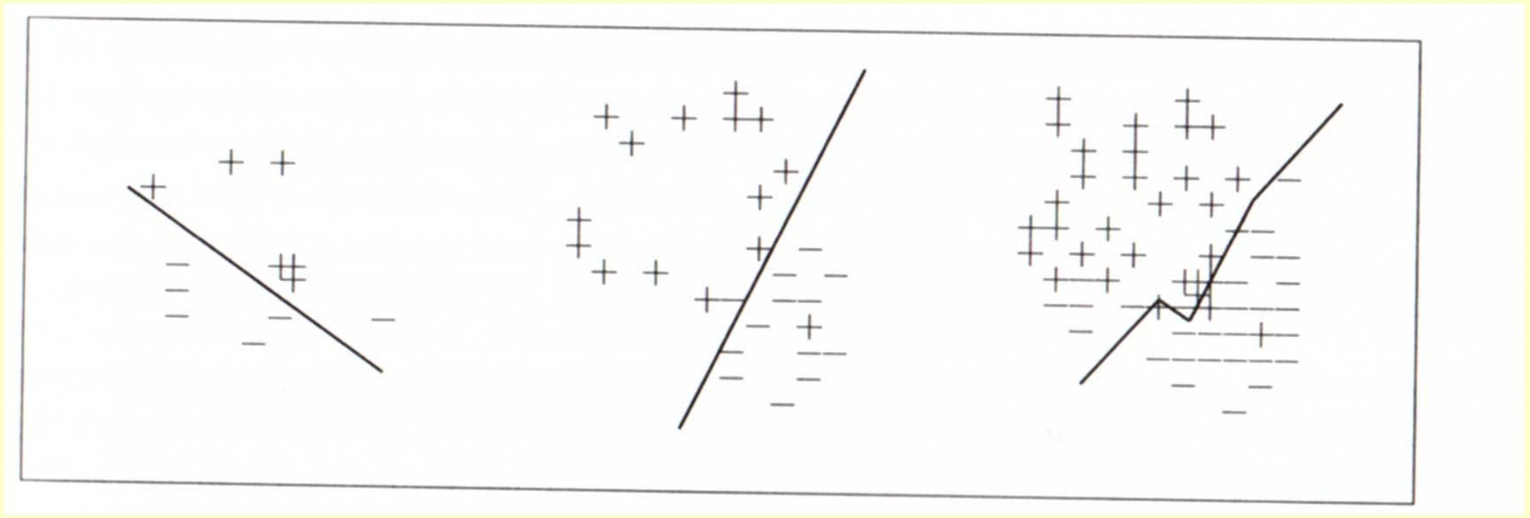
- *S* – training set of size *m*; h_i are **weak** hypotheses, i.e. only ε better than chance (50%) on the training set
- Learn *h1* on a subset *S1* of size *m1* < *m*
- Learn *h2* on a subset *S2* of size *m2* chosen from *S-S1;* at least half of the examples of *S2* are misclassified by *h1*
- Learn *h* on *m3* examples from *S-S1-S2* for which *h1* and *h2* disagree
- Final hypothesis *h*=majority-vote(*h1,h2,h3*)



- On the left, *S* and the subset *S1* (circled instances)
- On the right, *S1* and *h1*



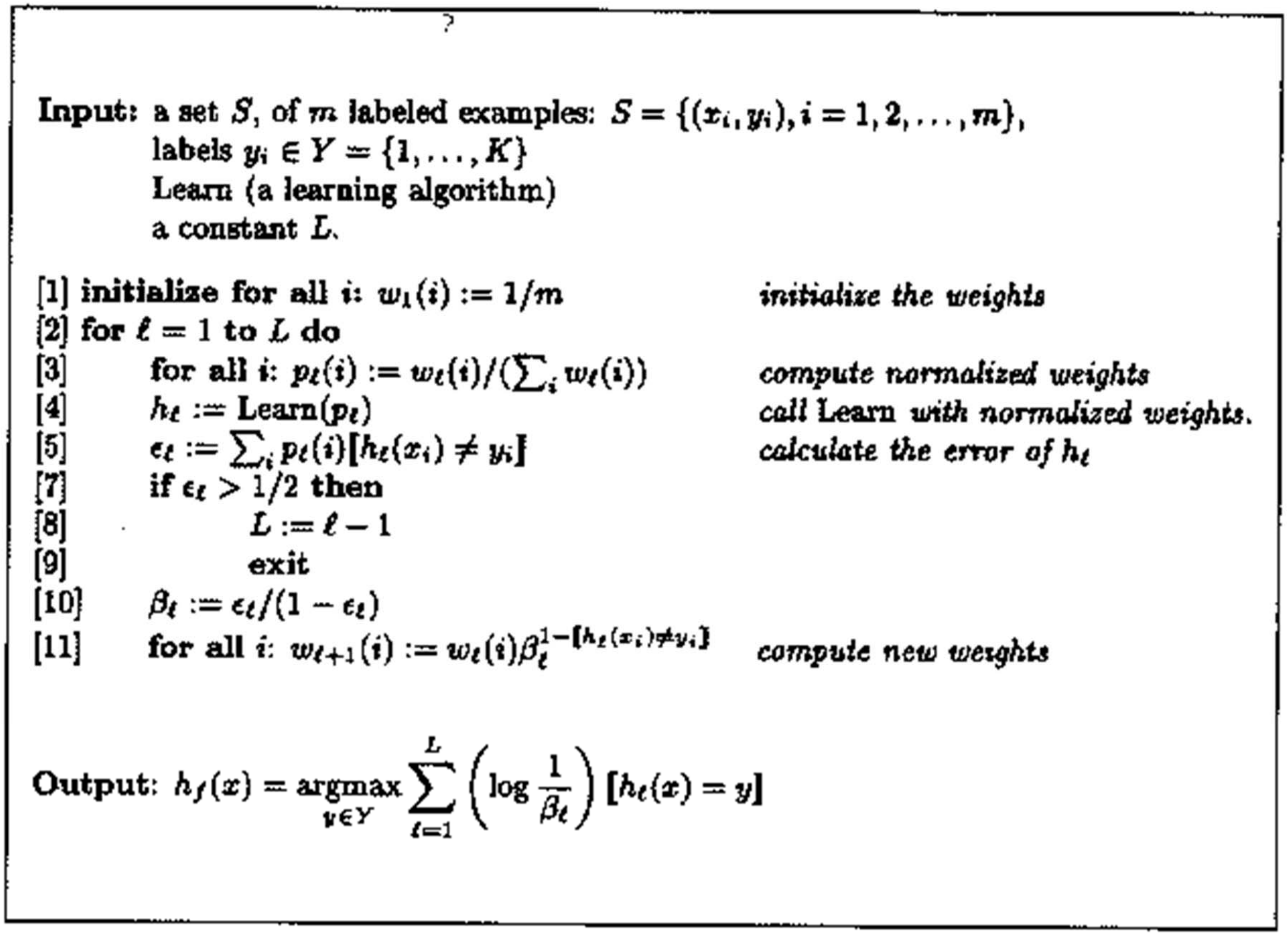
- On the left, *S-S1* and *h1*
- On the right, subset *S2* (circled instances) of *S-S1* most informative for *h1*;



- On the left, *S2* and a separating hyperplane *h2*
- In the middle, set *S3 = S-S1-S2* and hypothesis *h3* learned on *S3;*
- On the right, *S* and *h* = the combination of the 3 hypotheses *h1, h2, h3*

- We could recursively iterate with 9, 27, … subsets
- instead we will smooth that recursion by replacing a discrete “characteristic” function of a subset (values 0 or 1) with a distribution on the whole training set.

- Boosting: the sample is drawn according to a distribution, and that distribution emphasizes the examples misclassified in the previous iteration. Then a vote is taken.
- All that’s needed is a **weak** learner “*Learn*” (error = 0.5 – ε)
- *Learn* has to take into account the weights of examples, given to it in the form of a distribution



```
Input: a set $S$, of $m$ labeled examples: $S = \{(x_i, y_i), i = 1, 2, \ldots, m\}$,
       labels $y_i \in Y = \{1, \ldots, K\}$
       Learn (a learning algorithm)
       a constant $L$.

[1] initialize for all $i$: $w_1(i) := 1/m$                    initialize the weights
[2] for $\ell = 1$ to $L$ do
[3]     for all $i$: $p_\ell(i) := w_\ell(i)/(\sum_i w_\ell(i))$     compute normalized weights
[4]     $h_\ell :=$ Learn$(p_\ell)$                              call Learn with normalized weights.
[5]     $\epsilon_\ell := \sum_i p_\ell(i)[h_\ell(x_i) \neq y_i]$          calculate the error of $h_\ell$
[7]     if $\epsilon_\ell > 1/2$ then
[8]         $L := \ell - 1$
[9]         exit
[10]    $\beta_\ell := \epsilon_\ell/(1 - \epsilon_\ell)$
[11]    for all $i$: $w_{\ell+1}(i) := w_\ell(i)\beta_\ell^{1-[h_\ell(x_i) \neq y_i]}$   compute new weights

Output: $h_f(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{\ell=1}^{L} \left(\log \frac{1}{\beta_\ell}\right) [h_\ell(x) = y]$
```

Figure 2. The ADABOOST.M1 *Algorithm.*

The formula |[E|] is 1 if E is true and 0 otherwise.

- Let’s make sure we understand the makeup of the final classifier:
- We apply the subsequent hypotheses with the learned coefficients and get a class prediction for each hypothesis h_ℓ, count the vote for each class, choose the winner

- AdaBoost (Adaptive Boosting) uses the probability distribution. Either the learning algorithm uses it directly, or the distribution is used to produce the sample.
- See

  http://www.cs.technion.ac.il/~rani/LocBoost/

  for a demo.

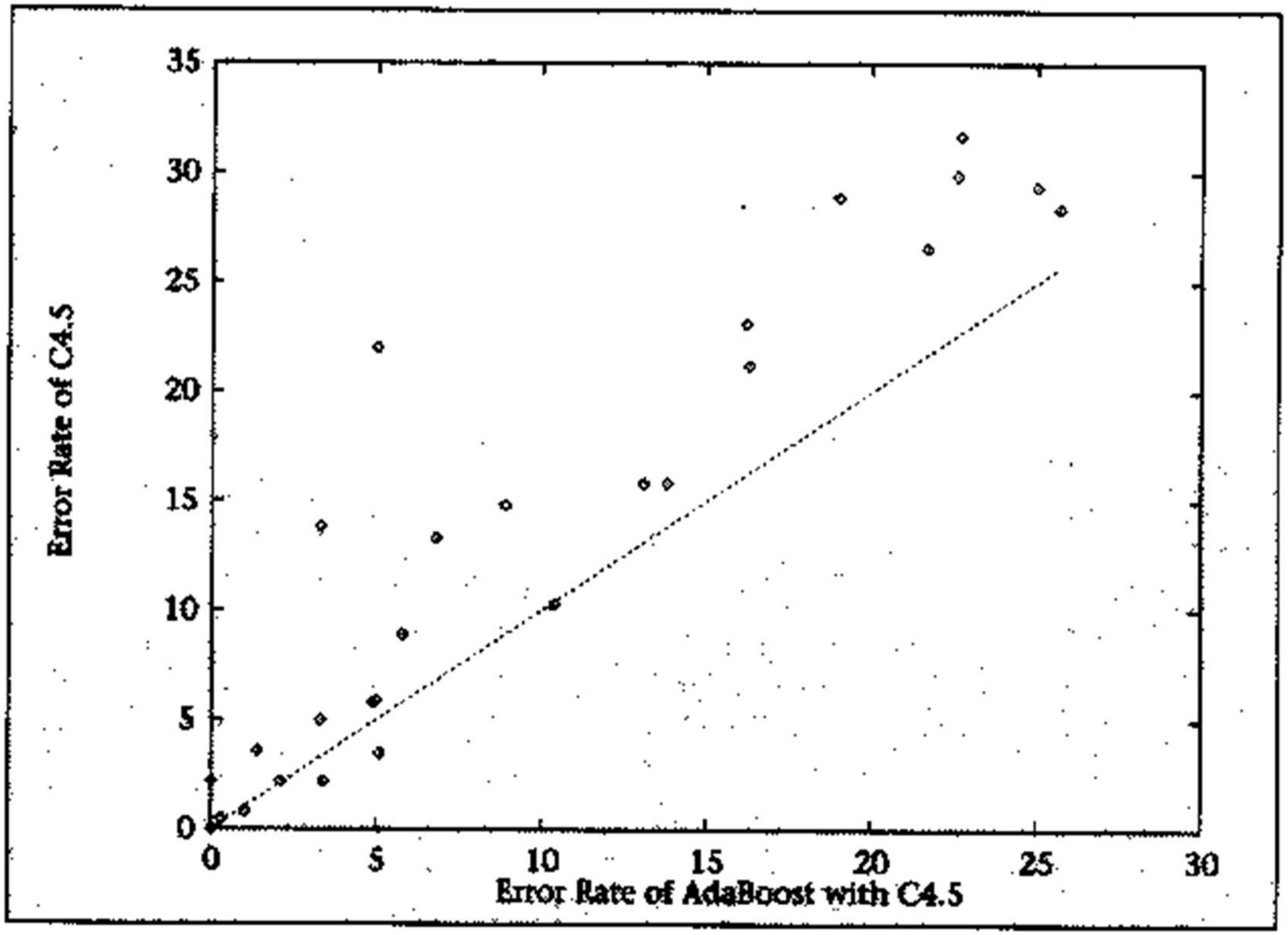


Figure 3. Comparison of ADABOOST.M1 *(applied to* C4.5*) with* C4.5 *by Itself.*

Each point represents 1 of 27 test domains. Points lying above the diagonal line exhibit lower error with ADABOOST.M1 than with C4.5 alone. Based on data from Freund and Schapire (1996). As many as 100 hypotheses were constructed.

Bagging

- [Breiman] is to learn multiple hypotheses from different subset of the training set, and then take majority vote.
- In fact, it is a general procedure to estimate a property of the training set
- The idea:
  - Sample N times (N~10**2) from the training set
  - Each sample is drawn randomly with replacement (a bootstrap). Each bootstrap contains, on avg., 63.2% of the training set (see [HTB] for explanation as to why 0.632
  - Learn a classifier on each sample
  - Make the classifiers vote

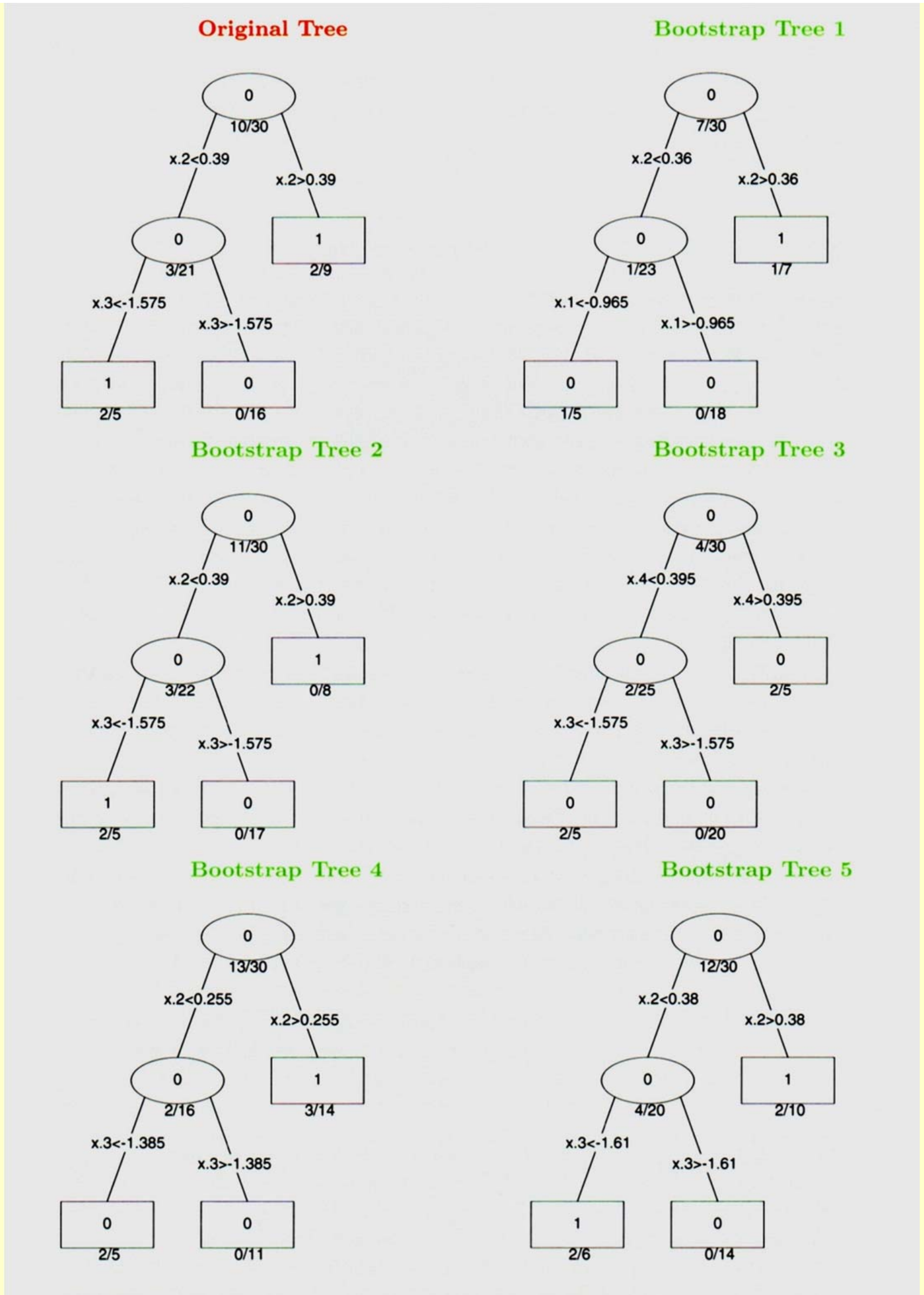
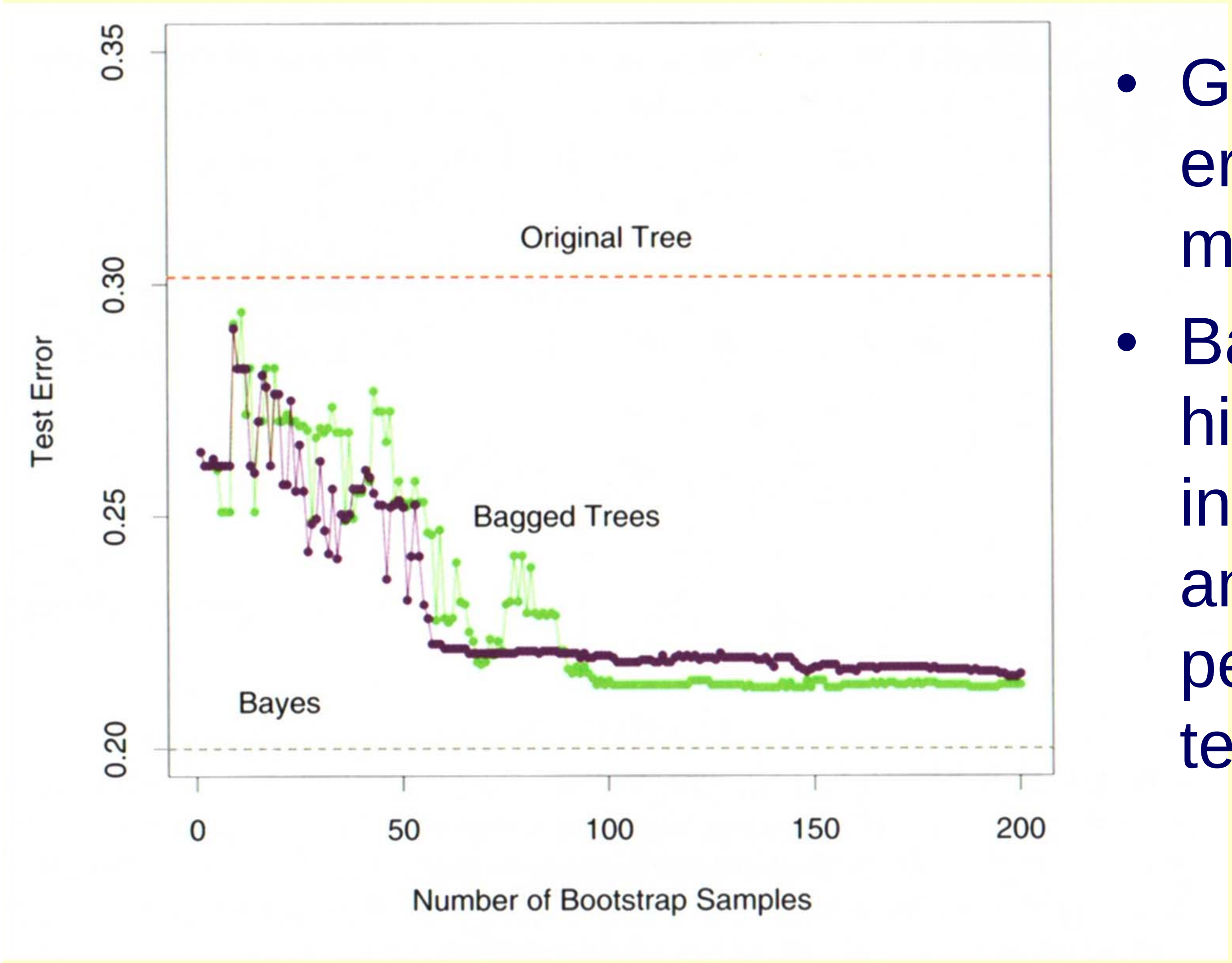


FIGURE 8.9. *Bagging trees on simulated dataset. Top left panel shows original tree. Five trees grown on bootstrap samples are shown.*

Bagging – example

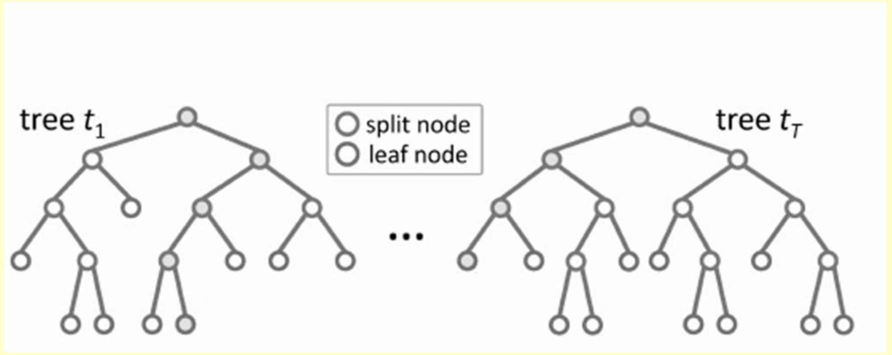
- Data: 30 instances, 2 classes, 5 features w/Gaussian distr. with pairwise correlation 0.95
- Class generated according to Pr(Y=1|x1<=0.5)=0.2, Pr(Y=1|x1>0.5)=0.8. NB error = 0.2
- test set of size 2000 from the same population
- Classifier built on the training and on each of 200 bootstraps; no pruning
- Predicted class indicated inside each node; misclassification and node size under each node
- Notice how the trees are different:
  - Different splitting features
  - Different cutpoints
- High variance (small change in data results in very different trees)

Results



- Green line = test error from majority vote
- Bagging reduces high variance of individual trees and so improves performance on test set

Random Forests (from Zhuowen Tu, UCLA)



- Random forests (RF) are a combination of tree predictors
- Each tree depends on the values of a random vector sampled independently
- The generalization error depends on the strength of the individual trees and the correlation between them
- Using a random selection of features yields results favorable to AdaBoost, and are more robust w.r.t. noise

The Random Forest Algorithm

```
Given a training set S
For  i = 1 to k do:
    Build subset Si by sampling with replacement from S
    Learn tree Ti from Si
        At each node:
            Choose best split from random subset of F features
        Each tree grows to the largest  depth, and no pruning
Make predictions according to majority vote of the set of k trees.
```

from Zhuowen Tu, UCLA

Features of Random Forests

- It is unexcelled in accuracy among current algorithms.
- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in unbalanced data sets.

from Zhuowen Tu, UCLA

Compared with Boosting

Pros:

- It is more robust.
- It is faster to train (no reweighting, each split is on a small subset of data and feature).
- Can handle missing/partial data.
- Is easier to extend to online version.

Cons:

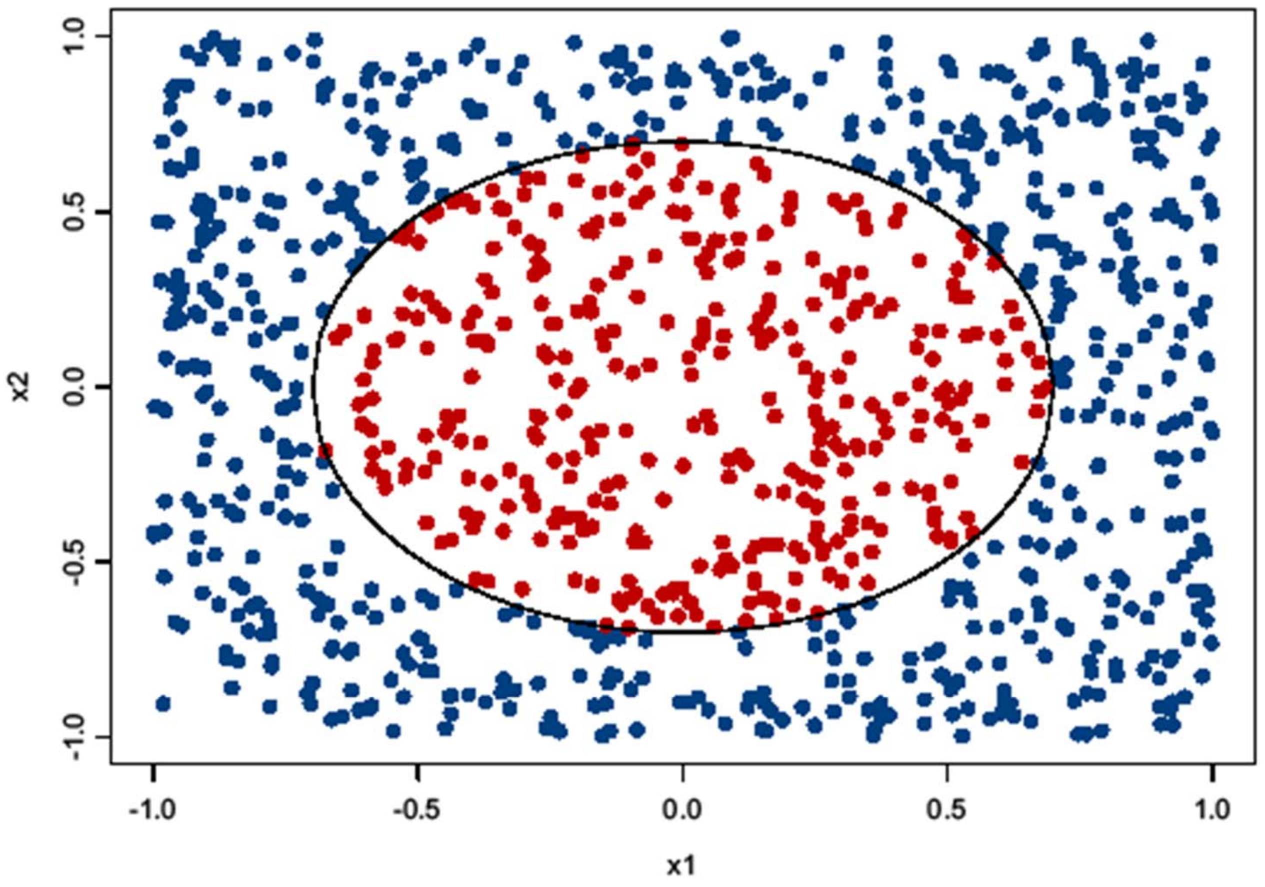
- The feature selection process is not explicit.
- Feature fusion is also less obvious.
- Has weaker performance on small size training data.

from Zhuowen Tu, UCLA

From **From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods**
Giovanni Seni, Santa Clara University
John Elder, Elder Research, Inc.

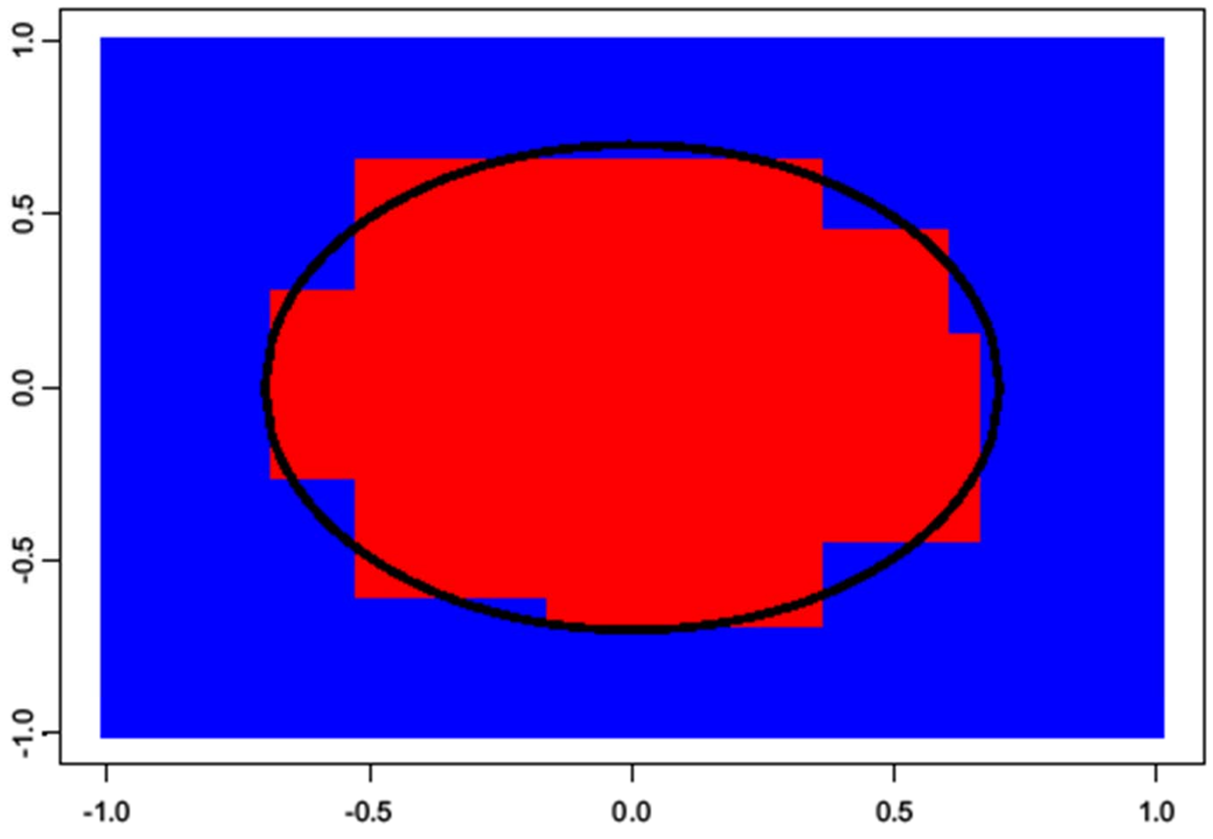
Appendix 1

Visualizing Bagging and AdaBoost (2-dimensional, 2-class example)



From **From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods**
Giovanni Seni, Santa Clara University
John Elder, Elder Research, Inc.

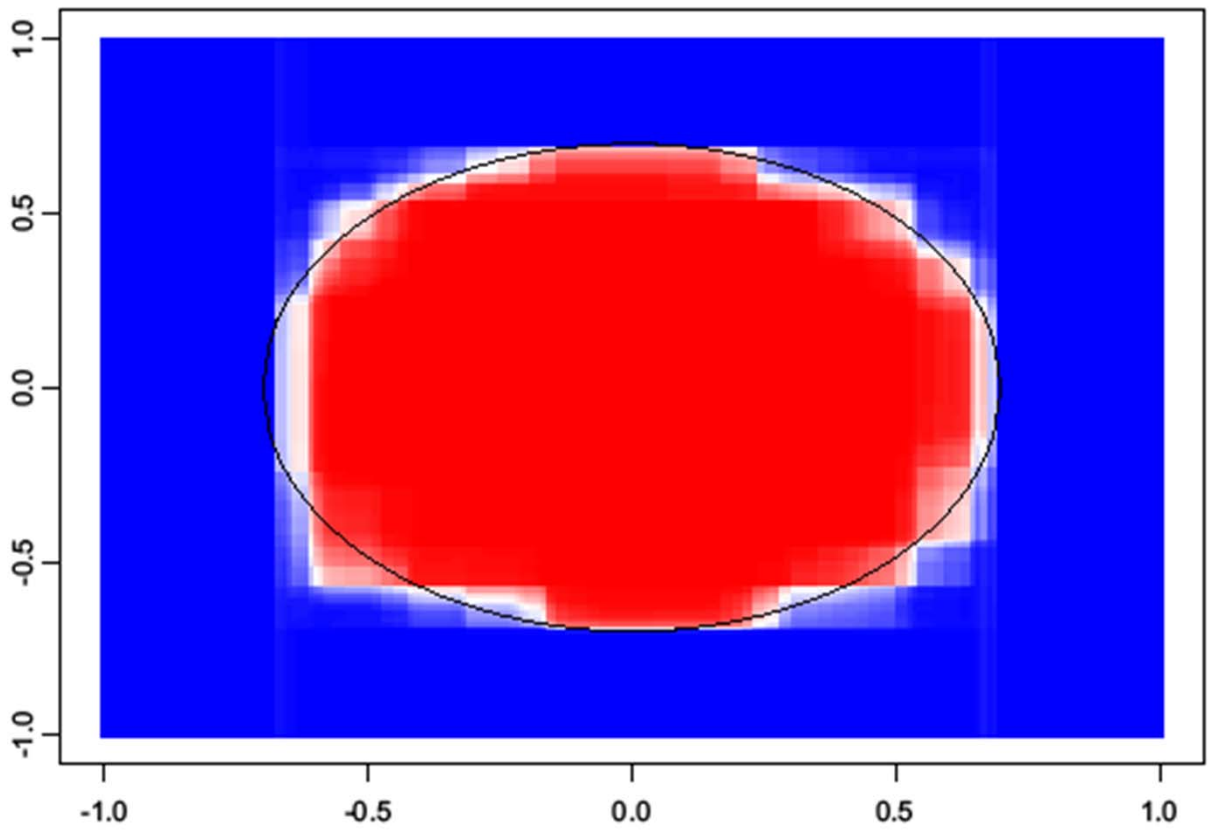
Appendix 1
Decision boundary of a single tree



From **From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods**
Giovanni Seni, Santa Clara University
John Elder, Elder Research, Inc.

Appendix 1

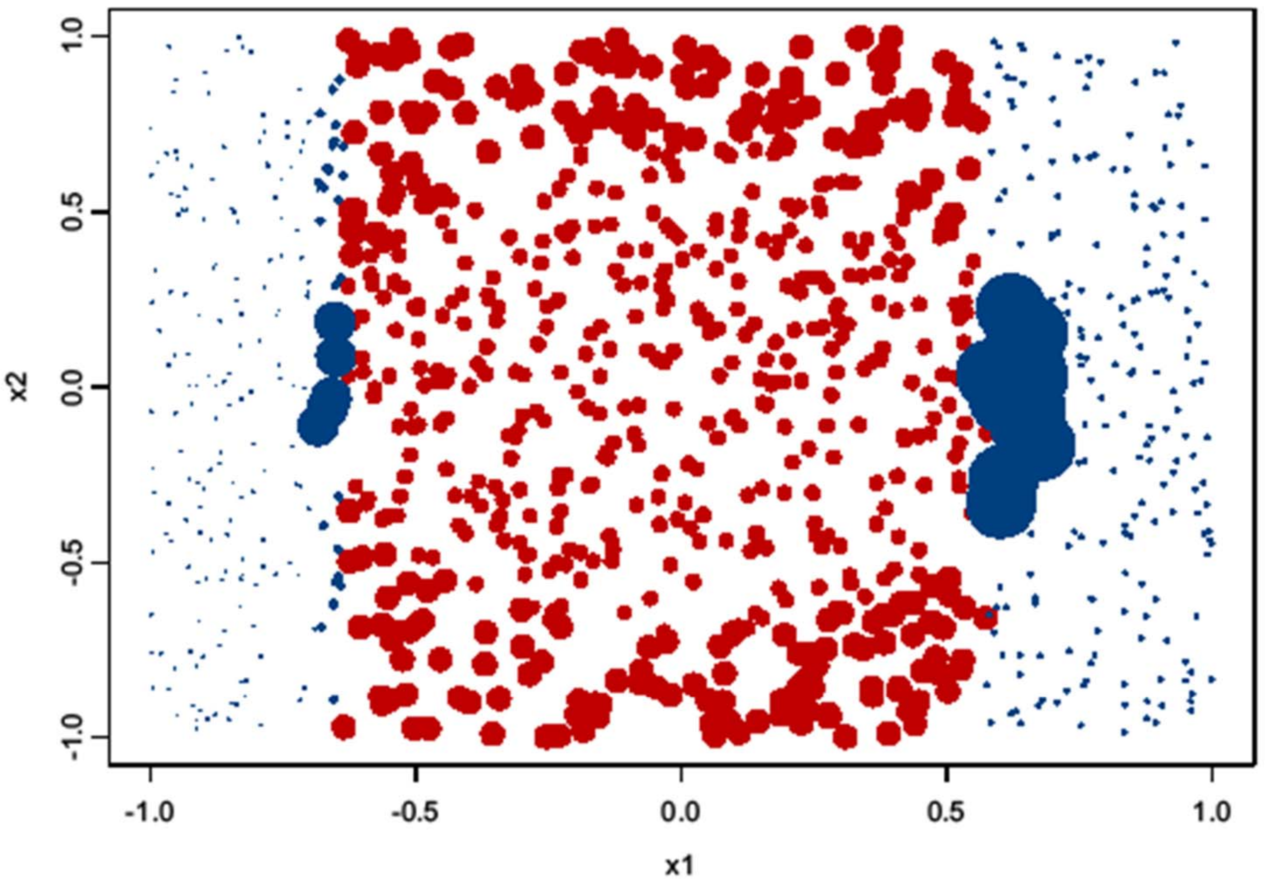
100 bagged trees leads to smoother boundary



From **From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods**
Giovanni Seni, Santa Clara University
John Elder, Elder Research, Inc.

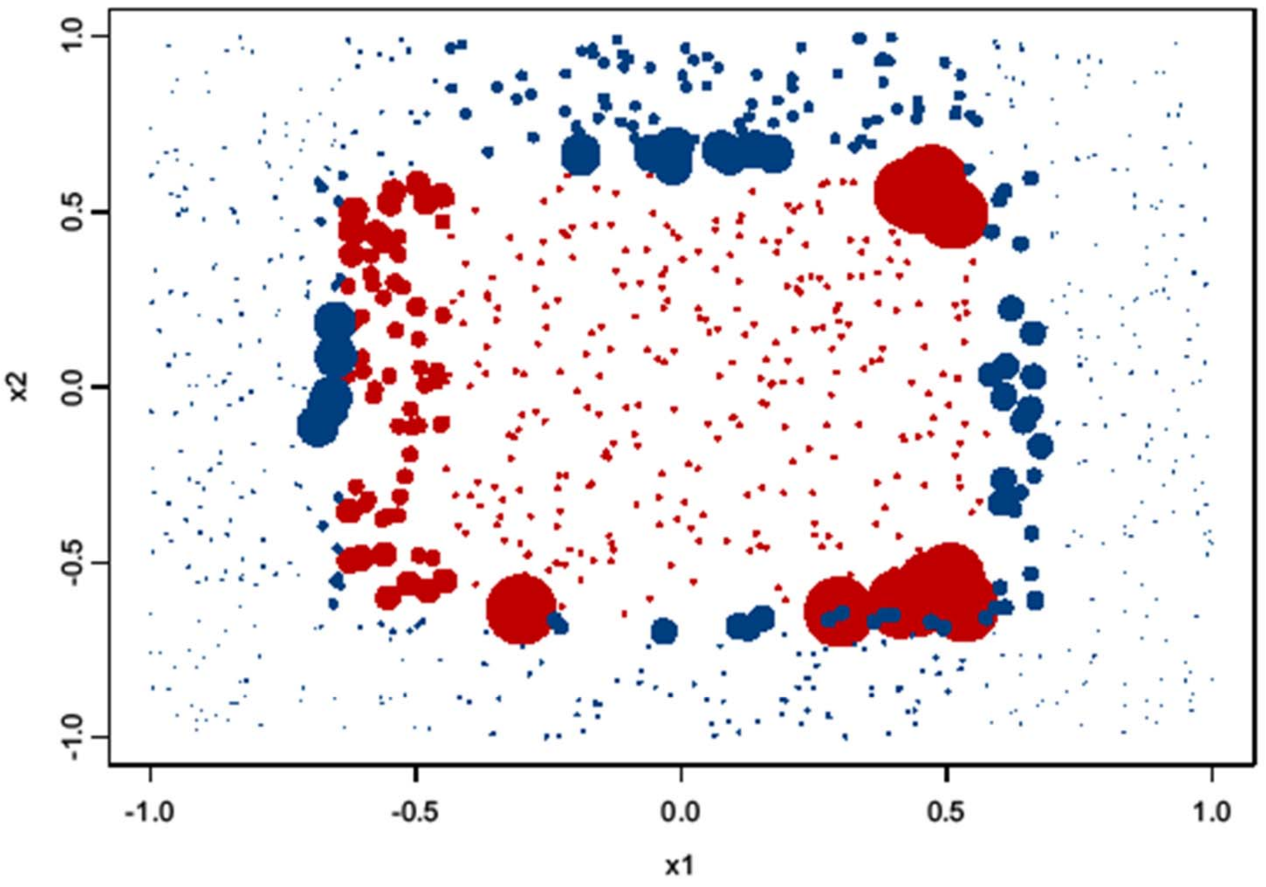
Appendix 1

AdaBoost, after one iteration (CART splits, larger points have great weight)



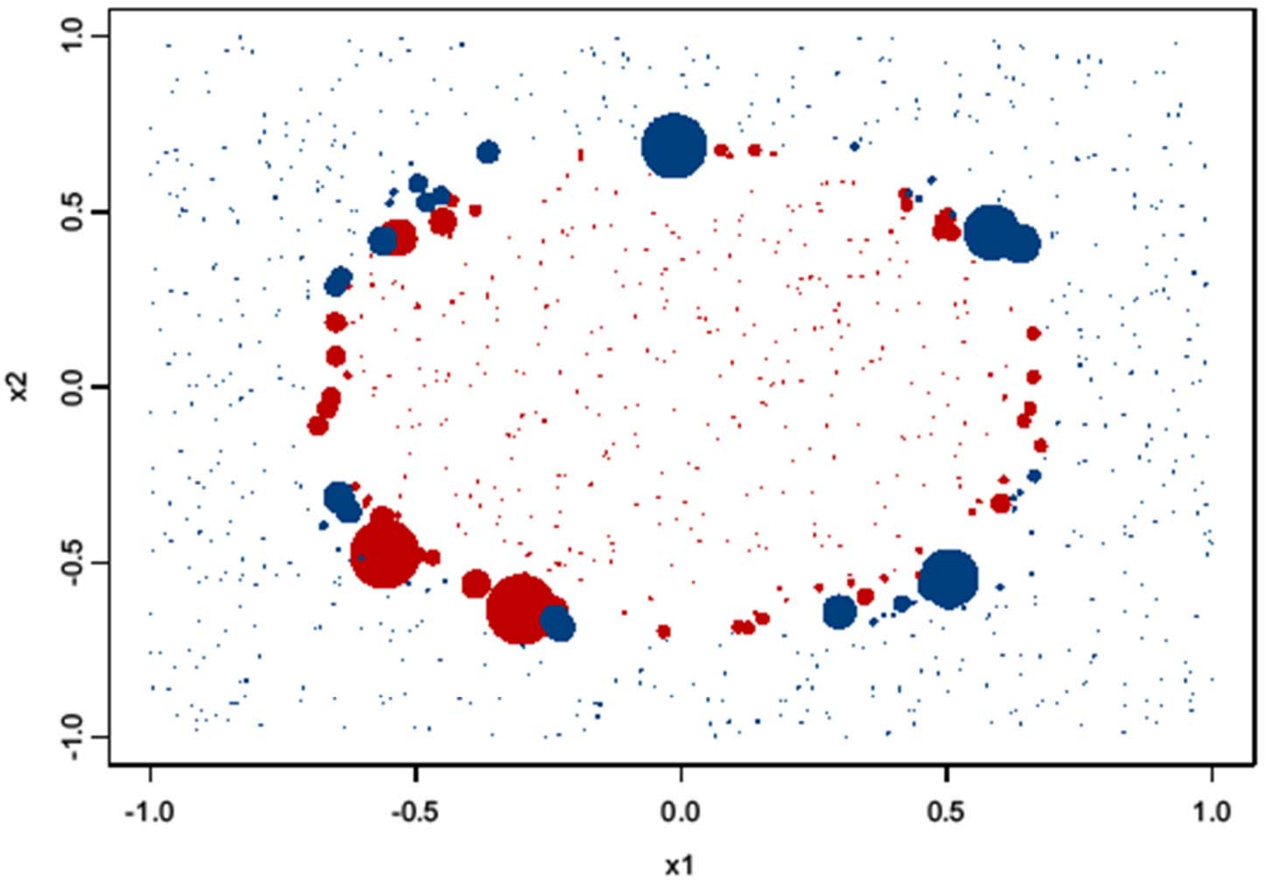
From **From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods**
Giovanni Seni, Santa Clara University
John Elder, Elder Research, Inc.

Appendix 1
After 3 iterations of AdaBoost



From **From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods**
Giovanni Seni, Santa Clara University
John Elder, Elder Research, Inc.

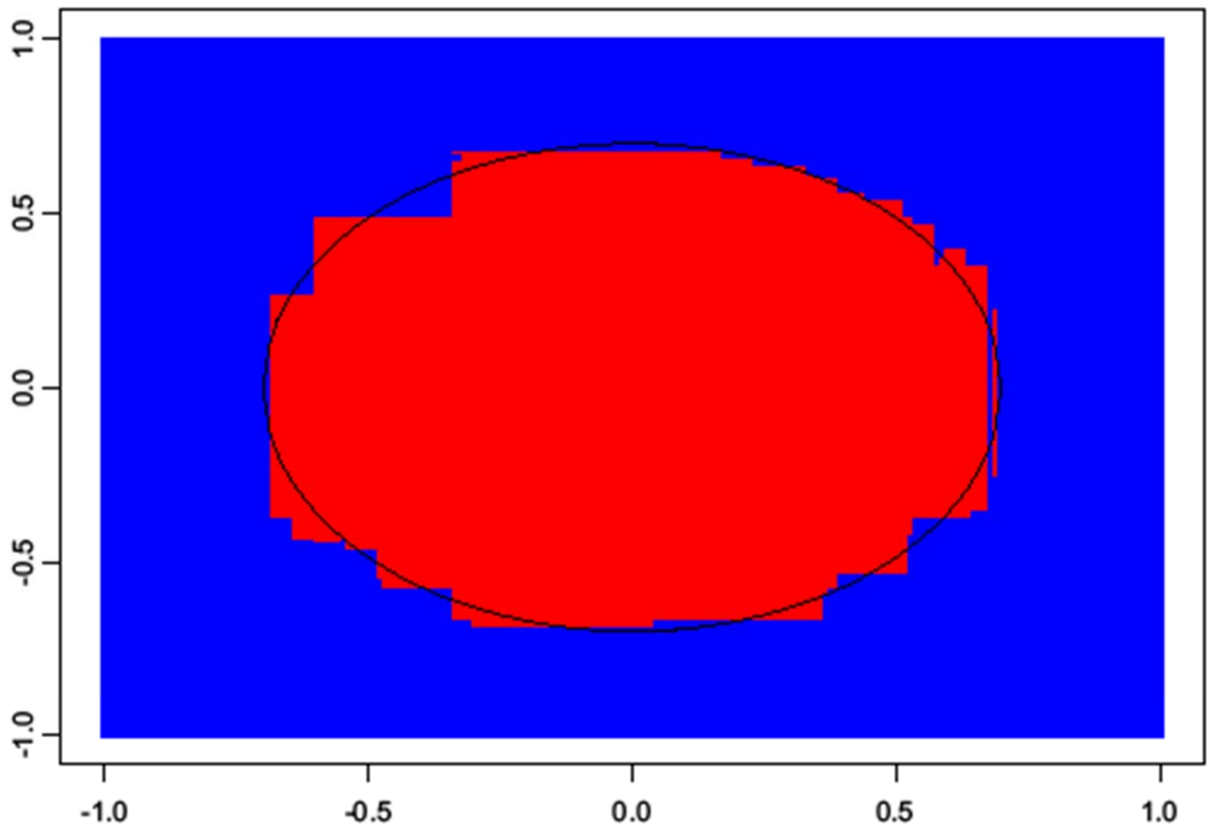
Appendix 1
After 20 iterations of AdaBoost



From **From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods**
Giovanni Seni, Santa Clara University
John Elder, Elder Research, Inc.

Appendix 1

Decision boundary after 100 iterations of AdaBoost



Other ensemble topics- Stacking

- Ensemble learns a set of models C1,...,Cn collect the output of each model on a new instance into a new set of data. For each instance x in the original training set, this data set represents every model's prediction of that instance's class (x-val’ed) with its true classification.
- This is a new learning problem: C1(x),...Cn(x), C(x).

Other ensemble topics- Random trees

- Random decision tree algorithm constructs multiple decision trees randomly.
- the algorithm picks a "remaining" feature randomly at each node expansion without any purity function check. Categorical feature used only once in a path. Each time the continuous feature is chosen, a random threshold is selected.

Wei Fan, IBM,
http://www.weifan.info/software.htm#Random%20DecisionTree

- Each node splits the data. A node becomes empty or there are no more examples to split in the current node, or the depth of tree exceeds some limits.
- Each node of the tree records class distributions. A node expansion is unnecessary, if none of its descendents have significantly different class distribution from this node.
- Classification is always done at the leaf node level. Each tree outputs a class probability distribution. The class distribution outputs from multiple trees are averaged as the final class distribution output

Error-correcting Output Codes (ECOC)

- Method of combining classifiers from a two-class problem to a k-class problem
- Often when working with a k-class problem k one-against-all classifiers are learned, and then combined using ECOC
- Consider a 4-class problem, and suppose that there are 7 classifiers, and classes are coded as follows:

- Suppose an instance ∈a is classified as 1011111 (mistake in the 2nd classifier).
- But the this classification is the closest to class a in terms of edit *distance* (Hamming dist. are 1, 3, 3, 5, for classes a, b, c, d). Also note that class encodings in col. 1 are not *error correcting* (any string other than 4 given has same distance to two classifiers)

class	class encoding
a 1000	**1111111**
b 0100	**0000111**
c 0010	**0011001**
d 0001	**0101010**

ECOC cont'd

- What makes an encoding error-correcting?
- Depends on the distance between encodings: an encoding with distance *d* between encodings may correct up to *(d-1)/2* errors (why?)
- In col. 1, *d=2*, so this encoding may correct 0 errors
- In col. 2, *d=4*, so single-bit errors will be corrected
- This example describes *row separation*; there must also be *column separation* (=1 in col. 2); otherwise two classifiers would make identical errors; this weakens error correction

ECOC cont’d

- For a small number of classes, exhaustive codes (all combinations of classifiers’ outputs without complements, all 0s and all 1s) as in col. 2 are used
- See the [Dietterich, Bakiri 95] paper on how to design good error-correcting codes
- Gives good results in practice, eg with SVM, decision trees, backprop NNs

Ensembles - conclusion

- Some of the best methods in terms of performance (boosting of decision tree stamps, random forest)
- Can deal with large datasets, both in terms of # of instances and # of attributes
- Do not require setting parameters
- Although invented for binary problems, generalize to multiclass thru one-vs-all + ECOC

Transduction (aka semi-supervised learning)

- We have limited set of labeled data and a very large set of unlabeled data. Can the unlabaled examples be used to train a better classifier than from the labeled data alone?
- Games example

Co-training

- Proposed by Blum and Mitchell (1998)
  - Powerful idea rooted in cognitive science/pedagogy:
  - Train two learning algorithms with a labeled sample using **two independent and sufficient views** of features (“ two students”).
  - Use each learning result to predict unlabeled examples and expand training set for the other (“students learn from each other”).

- Setting: X = $X_1 \times X_2$ – X_is are the *views*
- Views are sufficient: *for an instance x =* (x_1, x_2) with label $\boldsymbol{\ell}$, we have $f_1(x_1) = f_2\,(x_2) = f(x) = \boldsymbol{\ell}$
- Views are *conditionally independent* given the class: $p(x_1 | c) = p(x_2 | c)$

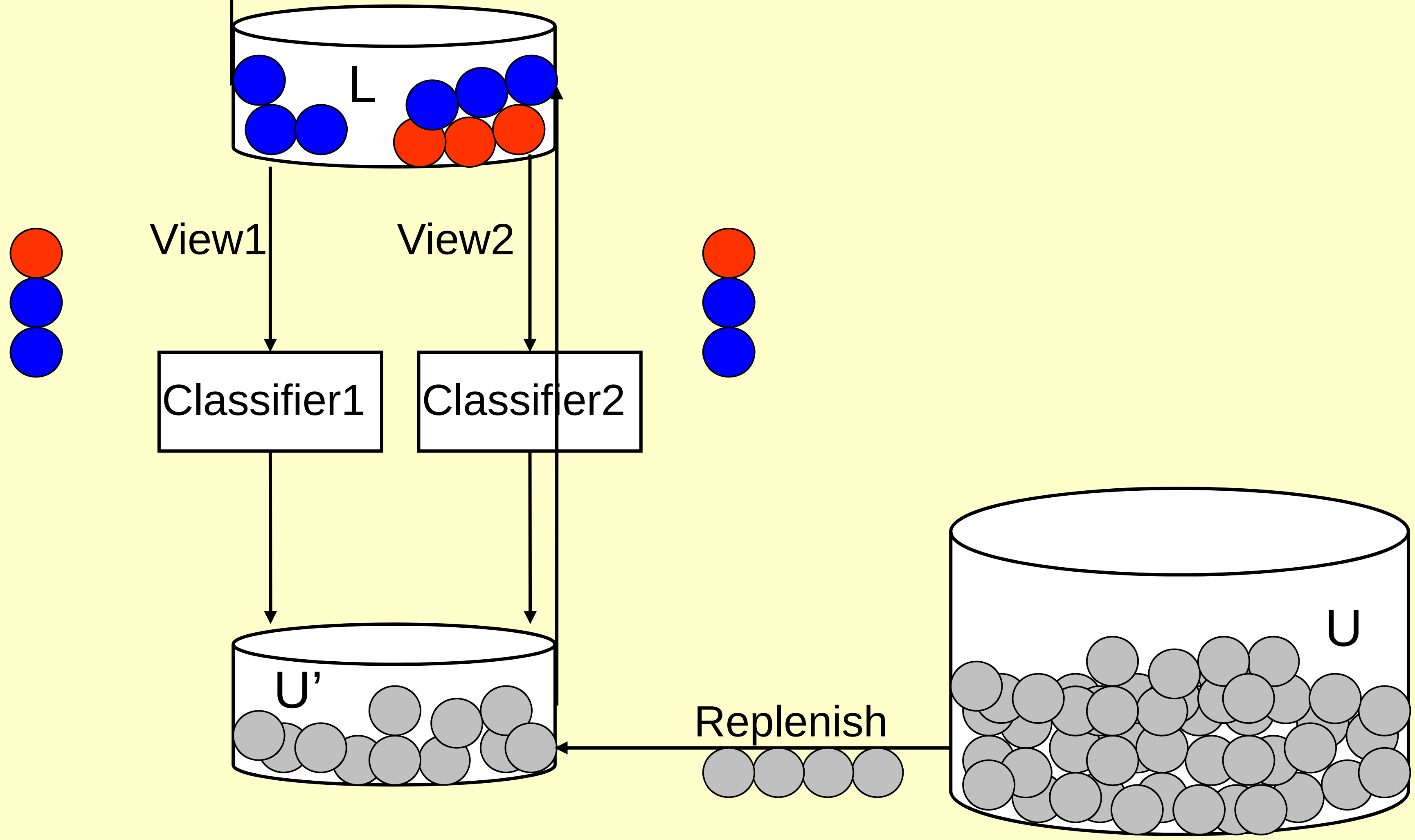
Two views in co-training

- Co-training assumption: two independent and sufficient views
- Example: the task of classification for CS faculty web pages
  - View 1: Page text. Words in home pages, such as research interests, teaching courses, etc.
  - View2: Hyperlink text. Words in hyperlinks that point to that page.

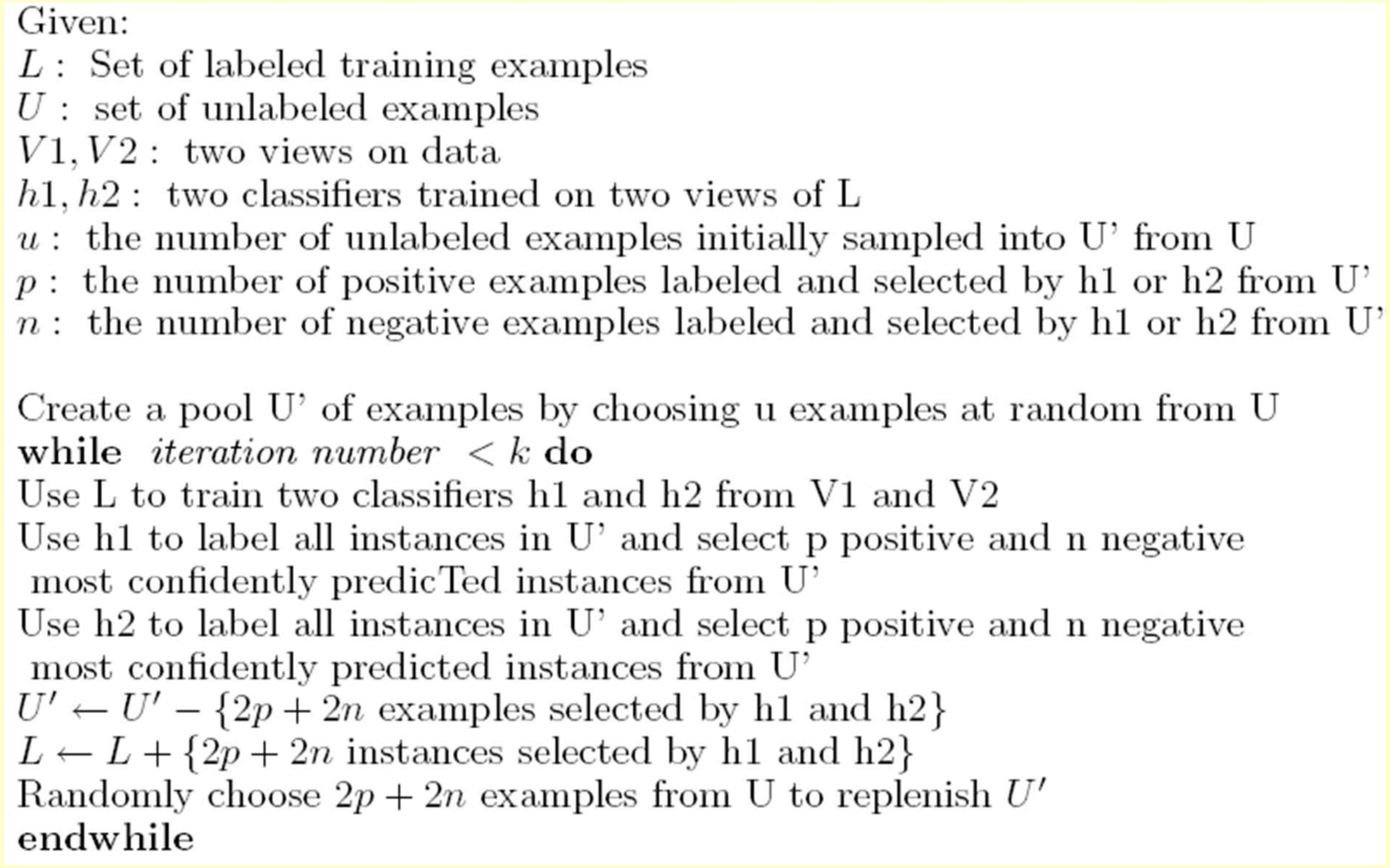
Related work

- (Balcan et. al. 2005) relaxed the assumption of two independent and sufficient views to an expansion of underlying data distribution.
- (Nigam & Ghani, 2000) showed dependent views may lead to successful co-training, but inferior to independent views.
- (Wang & Zhou, 2007) showed two arbitrary classifiers with large difference can be successful in co-training.

Illustration of co-training process



Co-training algorithm



```
Given:
$L$ :  Set of labeled training examples
$U$ :  set of unlabeled examples
$V1, V2$ :  two views on data
$h1, h2$ :  two classifiers trained on two views of L
$u$ :  the number of unlabeled examples initially sampled into U' from U
$p$ :  the number of positive examples labeled and selected by h1 or h2 from U'
$n$ :  the number of negative examples labeled and selected by h1 or h2 from U'

Create a pool U' of examples by choosing u examples at random from U
while  iteration number $< k$ do
Use L to train two classifiers h1 and h2 from V1 and V2
Use h1 to label all instances in U' and select p positive and n negative
 most confidently predicTed instances from U'
Use h2 to label all instances in U' and select p positive and n negative
 most confidently predicted instances from U'
$U' \leftarrow U' - \{2p + 2n$ examples selected by h1 and h2$\}$
$L \leftarrow L + \{2p + 2n$ instances selected by h1 and h2$\}$
Randomly choose $2p + 2n$ examples from U to replenish $U'$
endwhile
```

Co-training – discussion

- The assumption of independence of views
- Two scenarios for co-training:
  - As a classifier
  - As a labeler

Active learning

- The transductive setting
- The learner **chooses** the best unlabeled instances to be labeled
- The best performance: Query By Committee:
  - A committee of classifiers
  - Classify unlabeled instances
  - Choose the instances on which there is most disagreement among the committee

Sampling techniques in active learning

- Some sampling approaches in active learning to improve over random sampling
  - (Tong & Koller, 2000) used a sampling strategy by minimizing the version space.
  - (Freund et al., 1997) sampled unlabeled instances that member classifiers of a committee disagree most: QBC
  - (Saar-Tsechansky & Provost, 2001) sampled unlabeled instances according to variances of probability estimated by multiple models.

Can active learning sampling techniques be applied to co-training?

- The answer is “No”.
  - Co-training is a passive learning process.
  - Active learning usually selects most unconfidently predicted instances, which are very likely to be misclassified in co-training.
- We have to design new sampling method for co-training.