Data Mining: Classification Ensemble Methods

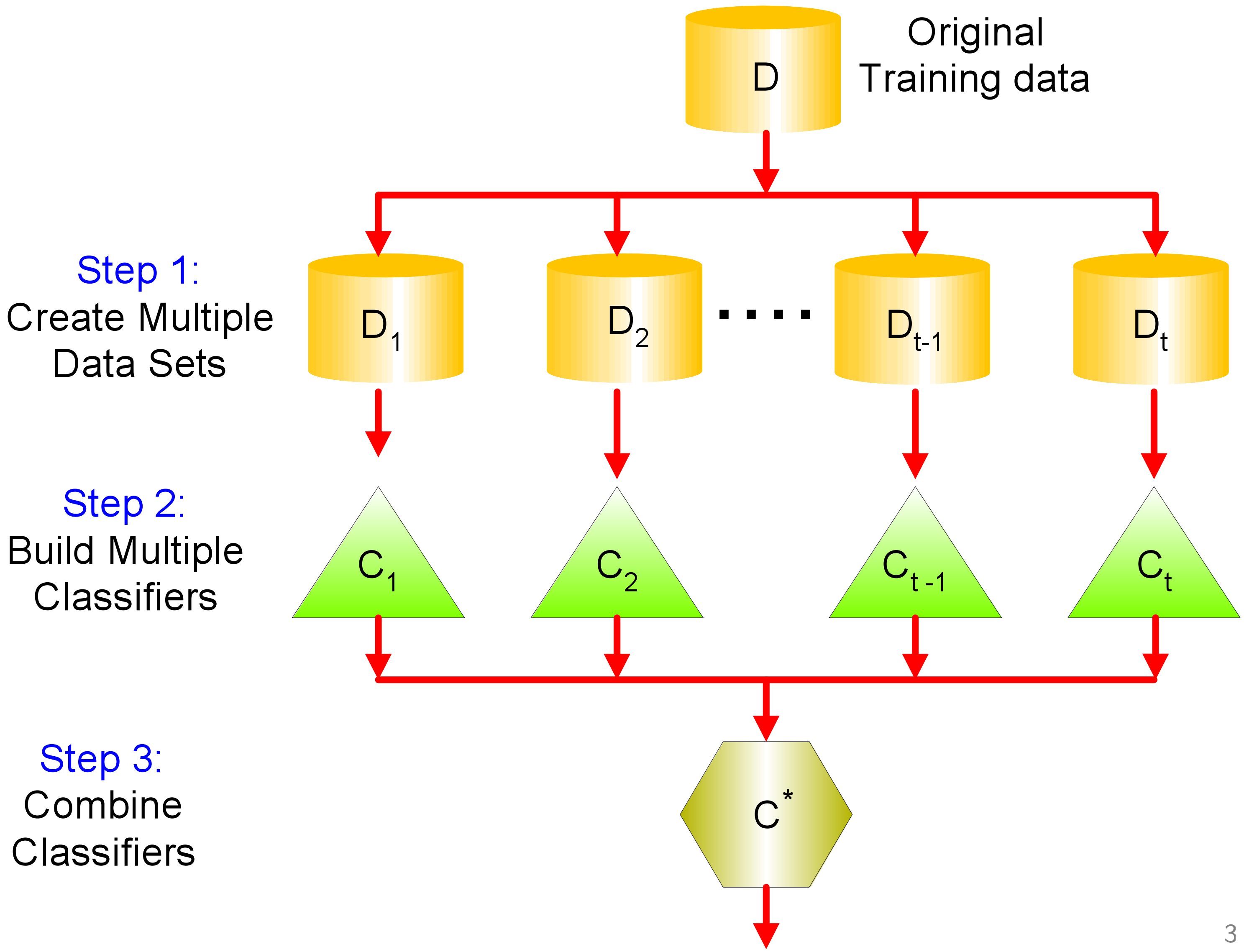
Laura Brown

Some slides adapted from P. Smyth; A. Moore; D. Klein; R. Tibshirani
Han, Kamber, & Pei; Tan, Steinbach, & Kumar; L. Hannah;
J. Taylor and L. Kaebling

Basic Ensembles

- Build different models on data
  - E.g., build same model multiple times with different random seeds or different hyperparameters
- Average/Majority of model results
  - Soft voting: averaging the predicted probabilities and take the arg max
  - Hard voting: use each model’s prediction and select most commonly predicted

Ensemble Methods



Ensemble Learning

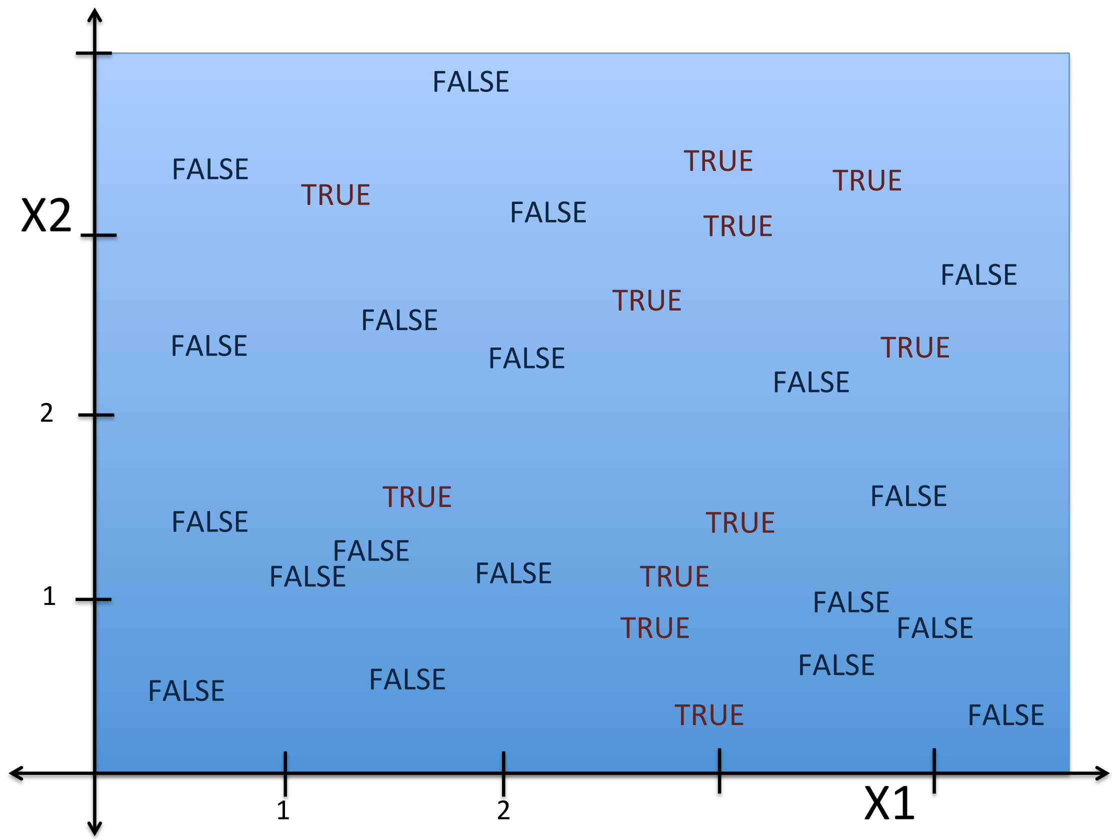
- Select a multitude of hypotheses to make predictions and combine their results
  - Use a number of different learners
  - Use the same learner with different hyperparameters
- If the errors are independent (or approaching independence), then the different hypotheses are complementary
- Combinations are most likely to be right than any individual hypothesis

Why Ensemble Methods Work?

- The simple models that go into the ensemble are easy to learn
  - But, they have a limited hypothesis space
- By combining (averaging) many different simple models, the models can fit the data well and have a large hypothesis space

Example: Ensemble Method

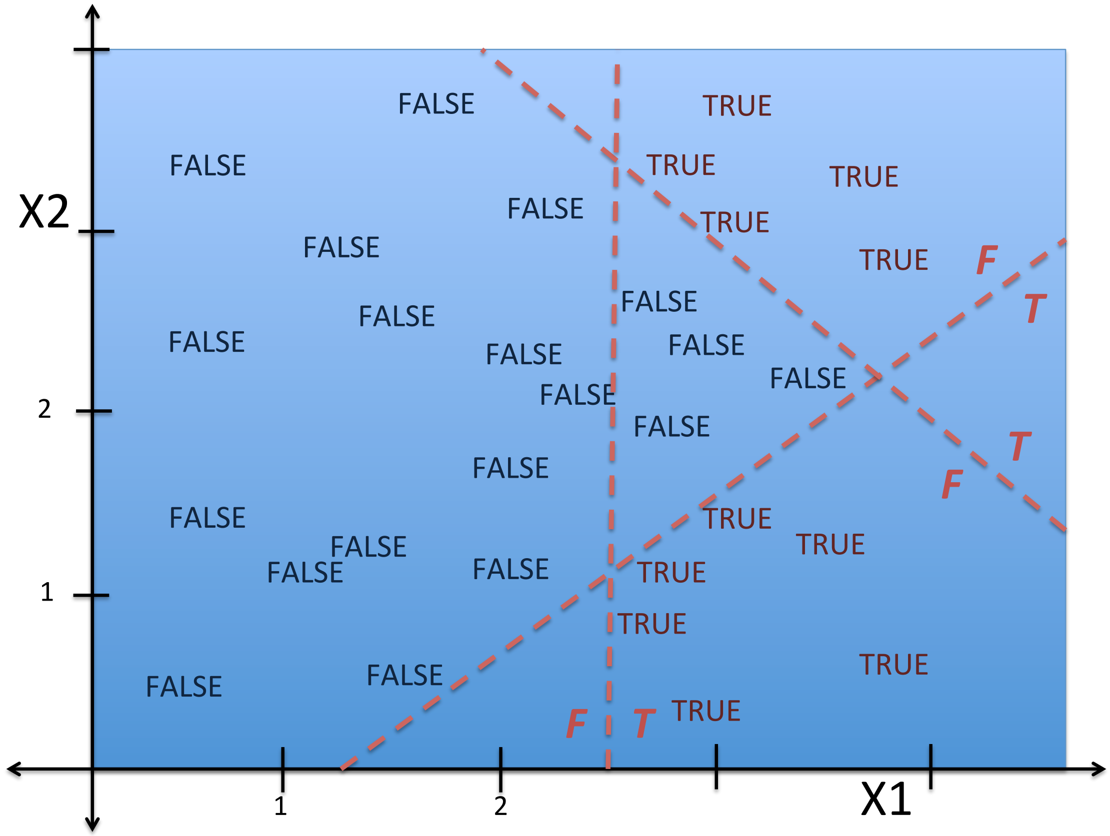
- Consider Linear Classifiers: divide space with a hyperplane



- No single hyperplane perfectly separate the two classes
- How can they be combined?

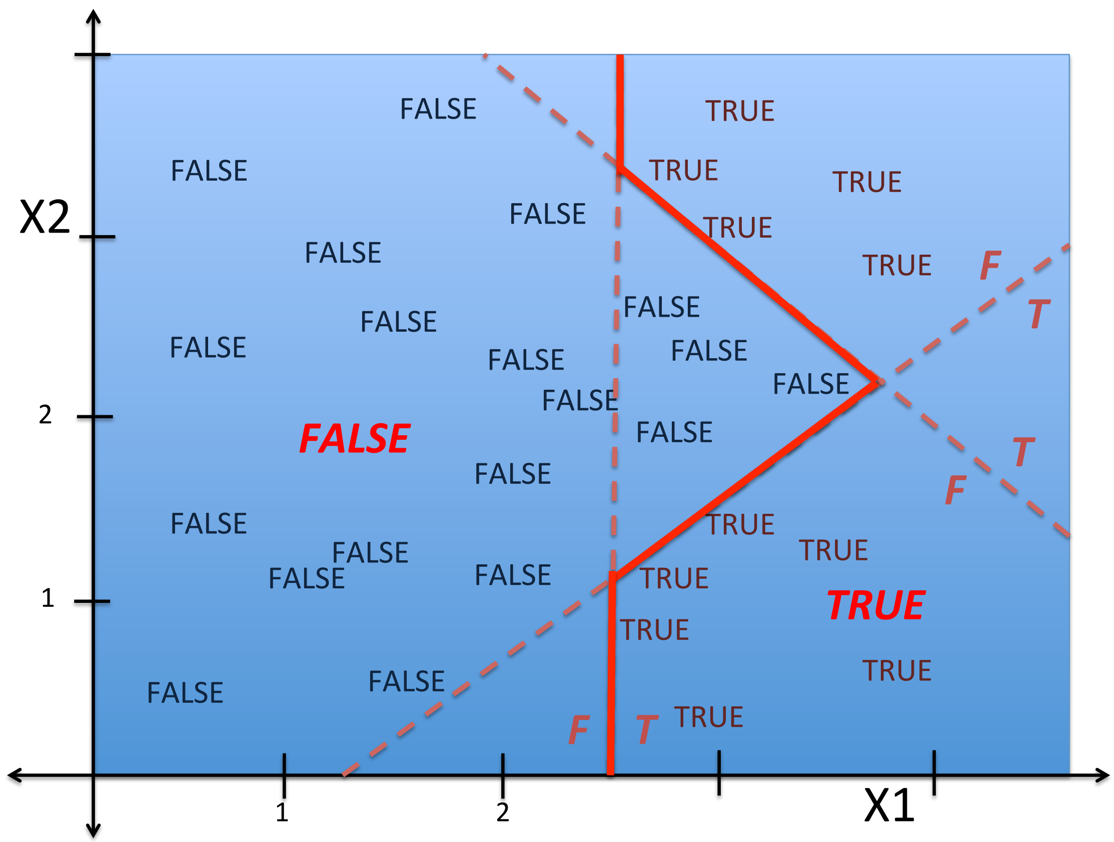
Example: Ensemble Method

- Consider Linear Classifiers: divide space with a hyperplane



- No single hyperplane perfectly separate the two classes
- How can they be combined?

Example: Ensemble Method



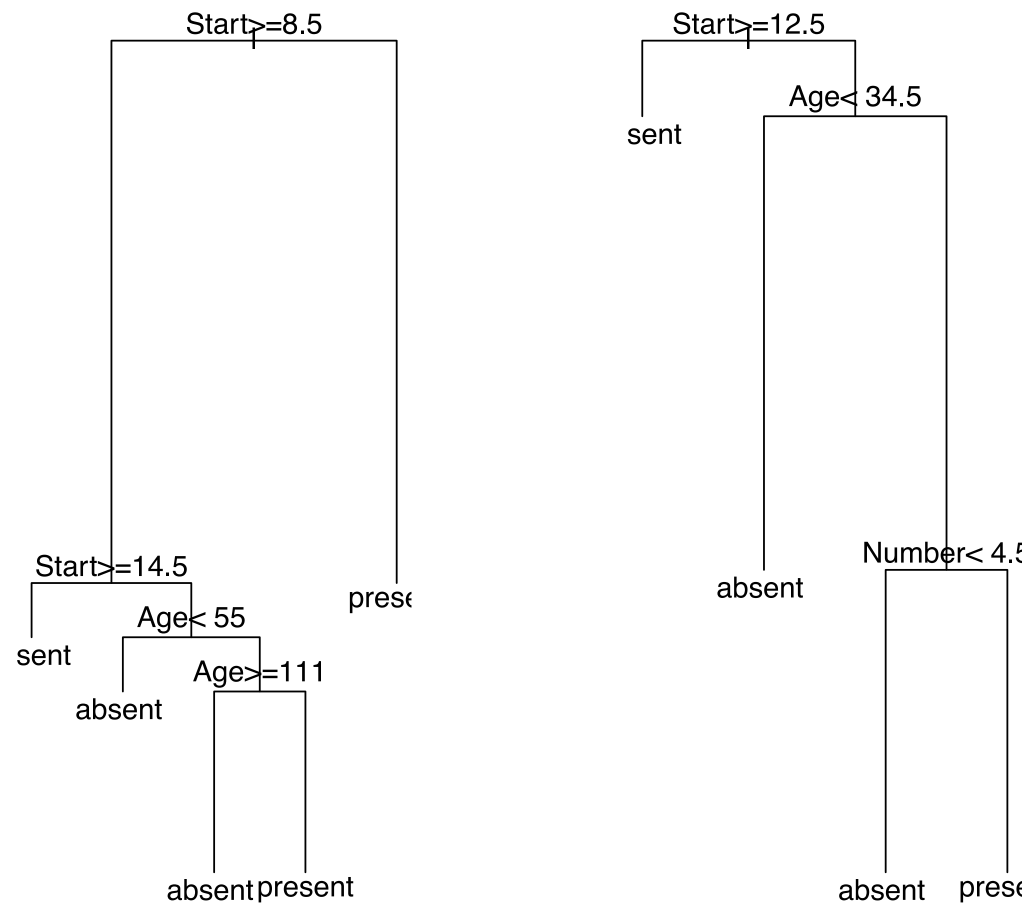
- Combine by assigning majority labels
- Results in non-linear surface

Ensemble Methods and Trees

Kyphosis data:

Trees

- Flexible models – work for both regression and classification
- Tend to fit pretty well, but do not always have best predictive error
- Trees are unstable



Ensemble Methods and Trees

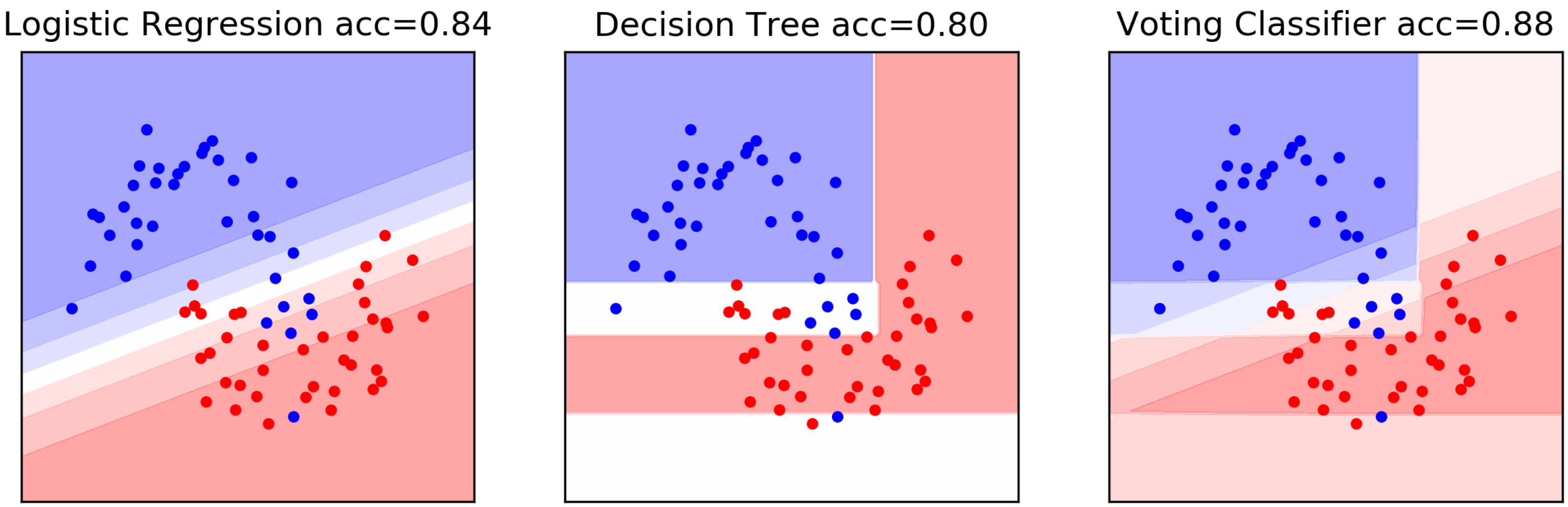
- Instability
  - Small changes in data (or fitting method) produce big changes in outcome
  - This is good for ensembles!
    Diverse results

Types of Ensemble Learning

- Model Averaging flavors
  - Fully Bayesian: average over uncertainty in parameters and models
  - “empirical Bayesian”: learn weights over multiple models
    - e.g., stacking and bagging
  - Build multiple models in a systematic way and combine them
    - VotingClassifier
    - Bagging
      - Random Forests
    - Stacking / Ensemble
    - Boosting

Voting Classifier

- Simplest Approach
- Learn multiple models or models with different hyperparameters on same data set
- Often better to use models that are different from each other



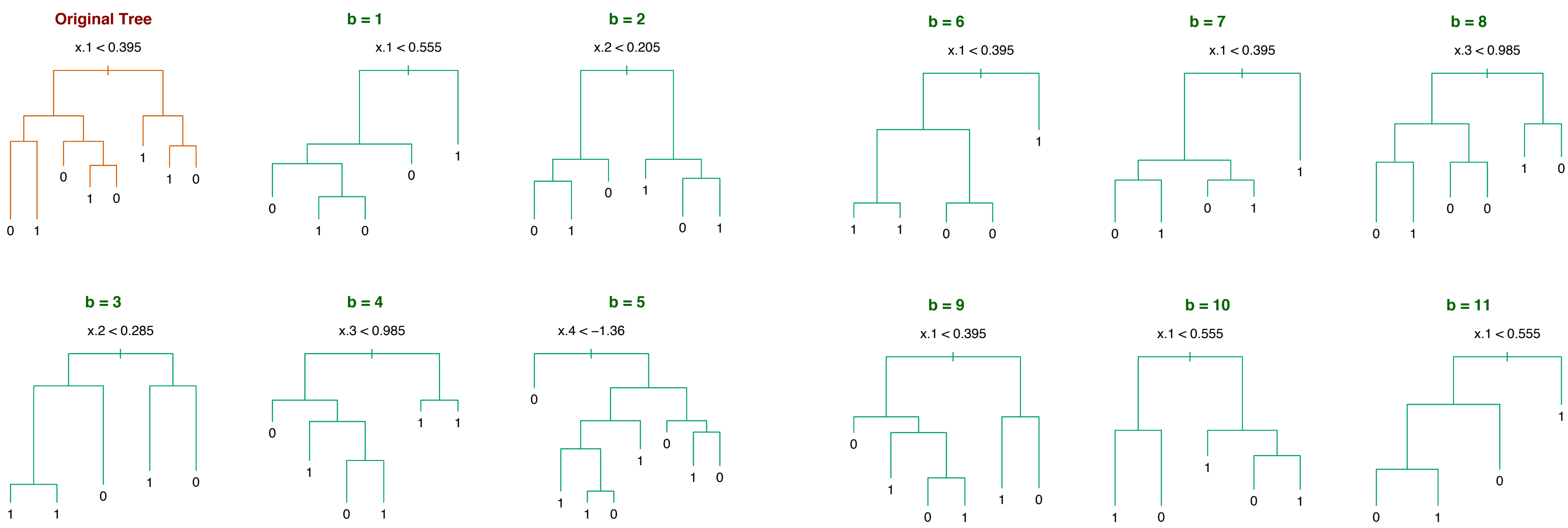
Bagging: Bootstrap Aggregation

- Training:
  - given a data set D with n tuples, at each iteration i, a training data set D_i of n tuples is sampled with replacement from D (bootstrap)
  - A classifier model M_i is learned for each D_i
- Classification: classify unknown sample x
  - Each classifier M_i returns its prediction
  - The bagged classifier M^* counts/averages the votes and assigns the class to $\boldsymbol{x}$
- Performance
  - Often better than single classifier on data D, but loses interpretability of model

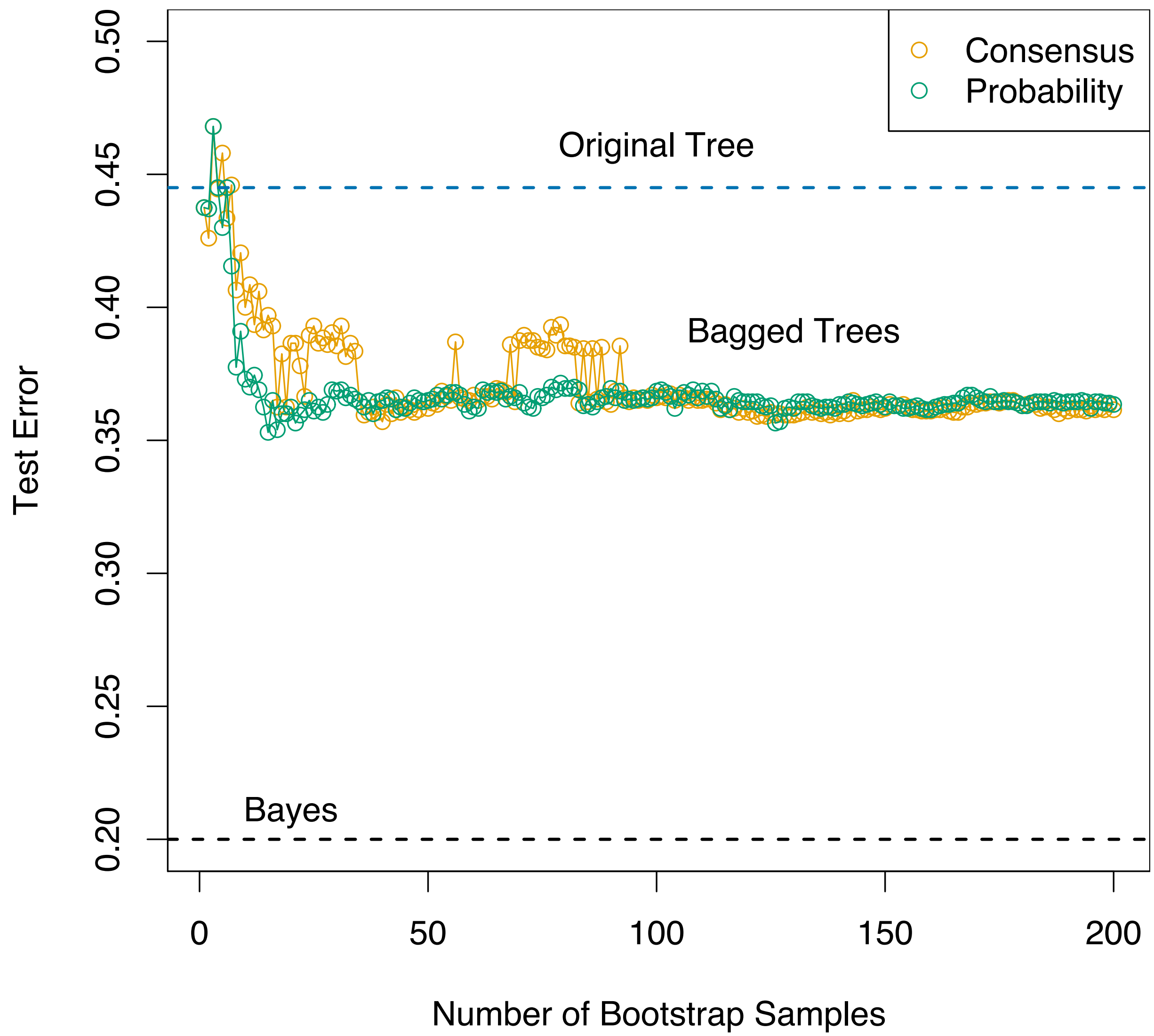
Breiman (1996): Bagging Predictors

Example: Simulated Data

- From ESL (8.7.1), n=30 training data points, *p*=5 features, and *2* classes.



Example: Simulated Data



Example: Breiman's Bagging

- From Breiman's paper: compare misclassification error of tree and that of bagging result

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
waveform	29.1	19.3	34%
heart	4.9	2.8	43%
breast cancer	5.9	3.7	37%
ionosphere	11.2	7.9	29%
diabetes	25.3	23.9	6%
glass	30.4	23.6	22%
soybean	8.6	6.8	21%

Bagging

- Pros:
  - Easy to implement
  - Works better than model on their own
  - Very fast and parallelizable

- Cons:
  - May not work as well as Boosting
  - Works best with high variance, low bias, low correlation estimators

Bagging

- Ensembles tend to work best with “not too complicated” hypotheses
- Why?
  - Simpler models are often less correlated
  - Cover a larger part of the hypothesis space

- Lets work again with trees, but try to de-correlate them

Random Forests

- Why de-correlate trees?
  - If inputs are the same, tree generation will produce the same branching path
  - But small changes in inputs, can lead to large changes in output
  - Force trees to split on different attributes

- **Randomly select a subset of attributes** that it can split on

Random Forests

- Grow each tree on independent bootstrap sample
- At each node:
  - Randomly select **m** variables out of all **p** (independently for each node)
  - Find the best split of selected **m** variables
- Grow each tree to maximum depth
- Vote / average the trees to get predictions

Hyperparameters for RFs

- Main hyperparameter: max_features
  - Number of features for each split
  - Sqrt(p) is common for classification
  - ~p for regression
- Number of estimators
  - More is better
- Pre-pruning can help save memory
  - Max_depth, max_leaf_nodes, min_samples_split, etc.

Stacked Generalization - Stacking

- Combine models in a different way (meta-learners)
  - combine learners of different types
- Idea
  - split training data into two sets
  - train several learners on first part
  - test these learners on second part
  - use the test predictions as inputs and target output to train a higher level of learner

Strong vs. Weak Learners

- So far, strategy has been:
  - gather a bunch of data
  - think hard, then make a single, large, complicated predictor
  - test the predictor on data

- What if we wanted to use a bunch of simple predictors instead, is there a principled way to do this?

Strong vs. Weak Learners

- A strong learner is a method that can learn a decision rule arbitrarily well
- A weak learner is a simple method that does better than guessing, but cannot learn a decision rule arbitrarily well.
- Example: trying to decide whether email is spam
  - Strong learner: method that uses words, syntax, etc. as features, and fits a high-accuracy decision rule
  - Weak learner: use simple rules, if phrase ”deal available” is in email, then predict is spam

Boosting

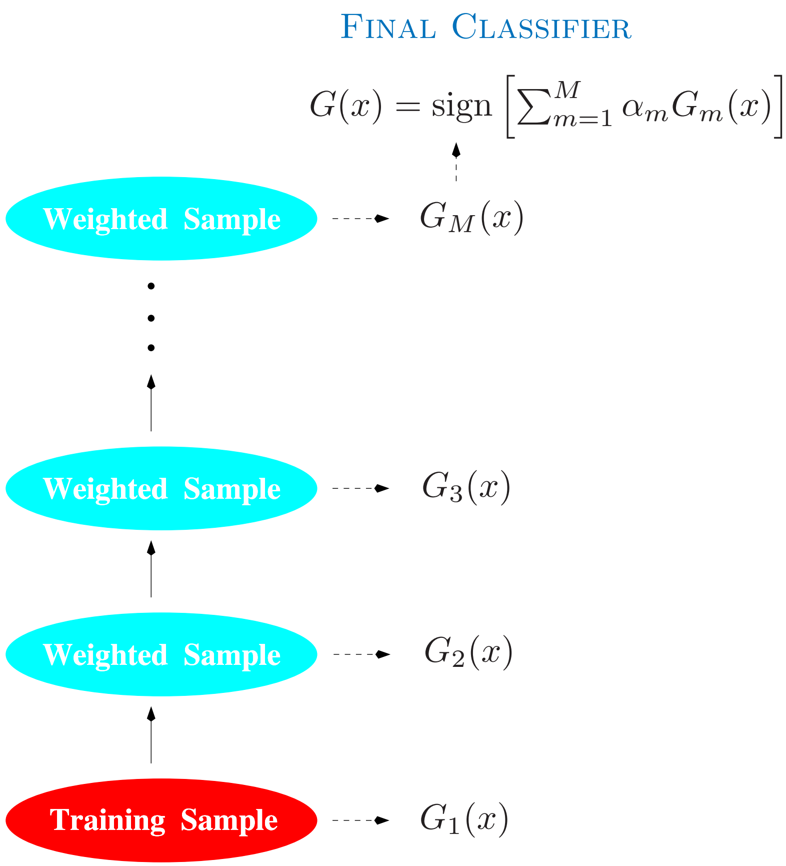
- Powerful technique originally designed for classification
  - extended to regression
- Basic Idea:
  - use a “weak” classifier (accuracy only slightly better than random)
  - create a series of such classifiers where the training data that was mis-classified on the previous iteration is given additional weight
  - combine successive models by voting to create a final model
- Example: Adaptive Boosting (AdaBoost)

Boosting

- Learning over weighted training set
  - **Weights** are assigned to each training tuple
  - A series of *k* classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to **pay more attention to the training tuples that were misclassified** by M_i
  - The final ***M**** **combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy

Boosting

- Difference between bagging and boosting
  - In boosting fit model to the entire training set, but adaptively weight the samples



ESL Fig 10.1

AdaBoost (Freund and Schapire, 1997)

- Given data set, $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$
- $W(x)$ is a distribution of weights over the n training examples
- Initially, set uniform weight distribution $w_i = 1/n$
- For each iteration, k
  - find model (hypothesis) $H_k(x)$ with min. error e_h using weights $W_k(x)$
  - compute α_k $$\alpha_k = \frac{1}{2}\ln\frac{1-e_k}{e_k}$$
  - Update weights
    - correctly labeled samples; decrease wt $W_{k+1} = W_k * \exp(-\alpha_k)$
    - incorrectly labeled samples; increase wt $W_{k+1} = W_k * \exp(\alpha_k)$
- Final Model

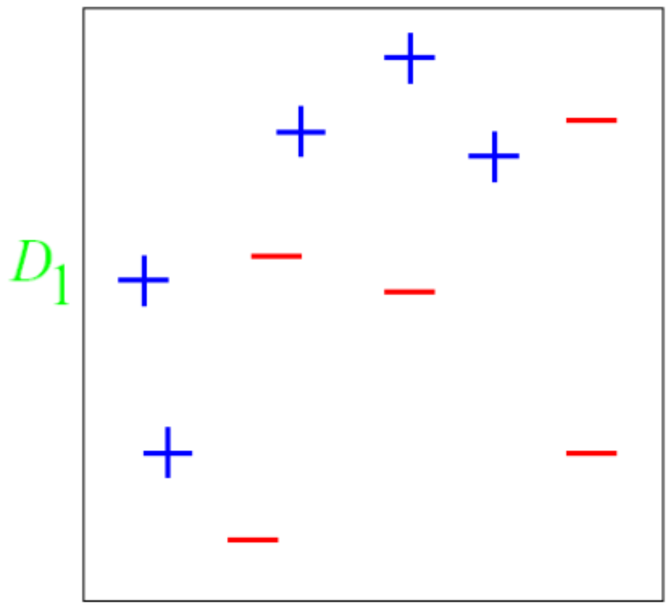
$$H_{final}(x) = sign(\sum \alpha_k H_k(x))$$

AdaBoost

- Why consider using AdaBoost?
  - No tunable parameters
  - Works with any weak learner
  - Computational feasible
  - Tends to avoid overfitting

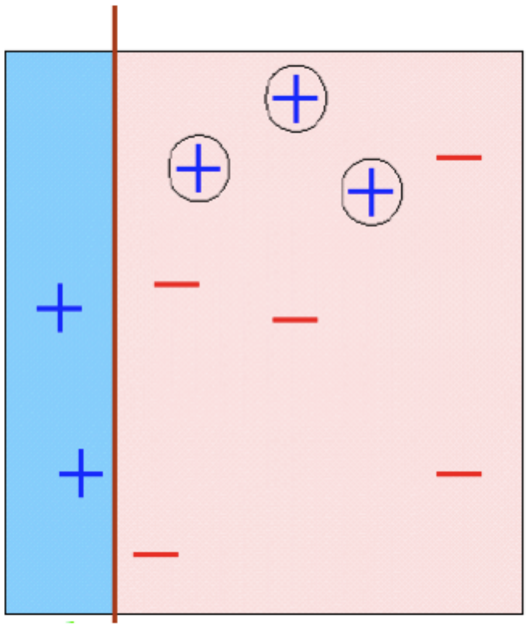
Example: AdaBoost

- Initial Training data
- All weights equal, for each sample



Example from "A Tutorial on Boosting" Freund and Schapire

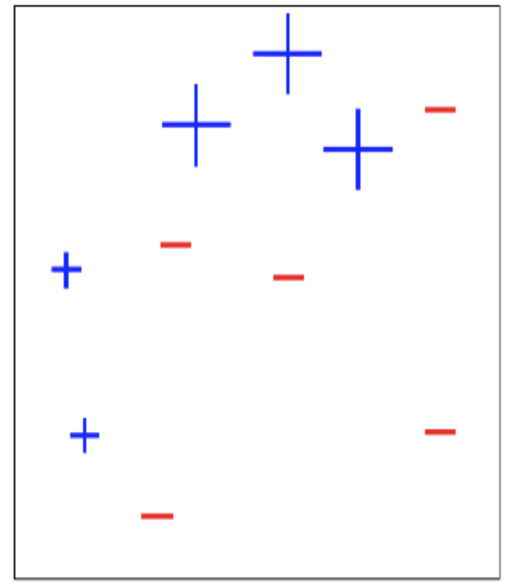
Example: AdaBoost



h_1

$$e_1 = 0.3$$
$$\alpha_1 = 0.42$$

D_2



$$e_1 = \sum_{incorrect} w(incorrect) = 0.3$$

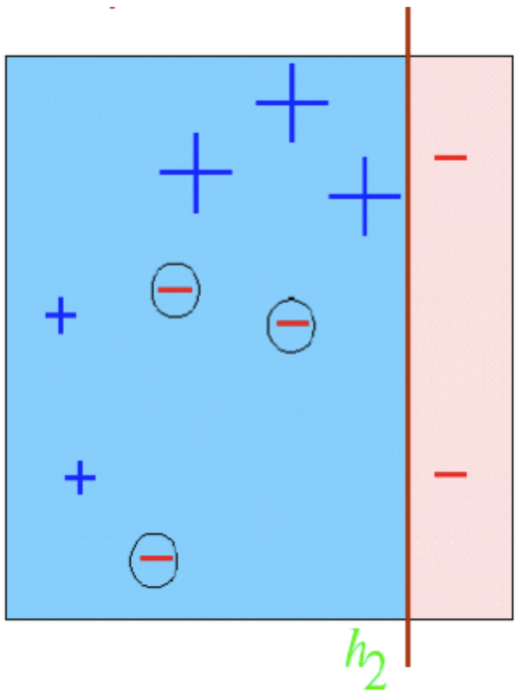
$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-e_1}{e_1}\right) = \frac{1}{2}\ln\left(\frac{0.7}{0.3}\right) = 0.42$$

$$w_{corr}' = C_N w_{corr} e^{-\alpha_1} = (1.091)0.1(0.6546) = 0.0714$$

$$w_{incorr}' = C_N w_{incorr} e^{\alpha_1} = (1.091)0.1(1.5275) = 0.1667$$

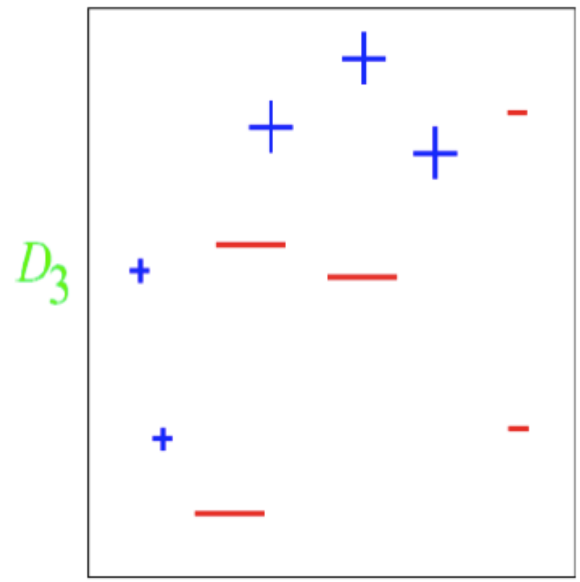
$$C_N = 1 / \sum_N w' = 1.091$$

Example: AdaBoost



$e_2 = 0.21$

$\alpha_2 = 0.65$



$$e_2 = \sum_{incorrect} w(incorrect) = 0.21$$

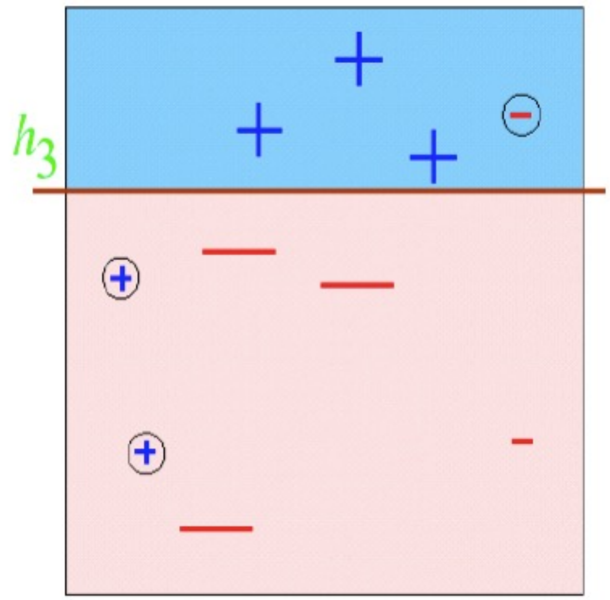
$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-e_1}{e_1}\right) = \frac{1}{2}\ln\left(\frac{0.79}{0.21}\right) = 0.65$$

$$w_{corr}' = C_N w_{corr} e^{-\alpha_2}$$

$$w_{incorr}' = C_N w_{incorr} e^{\alpha_2}$$

$$C_N = 1/\sum_N w'$$

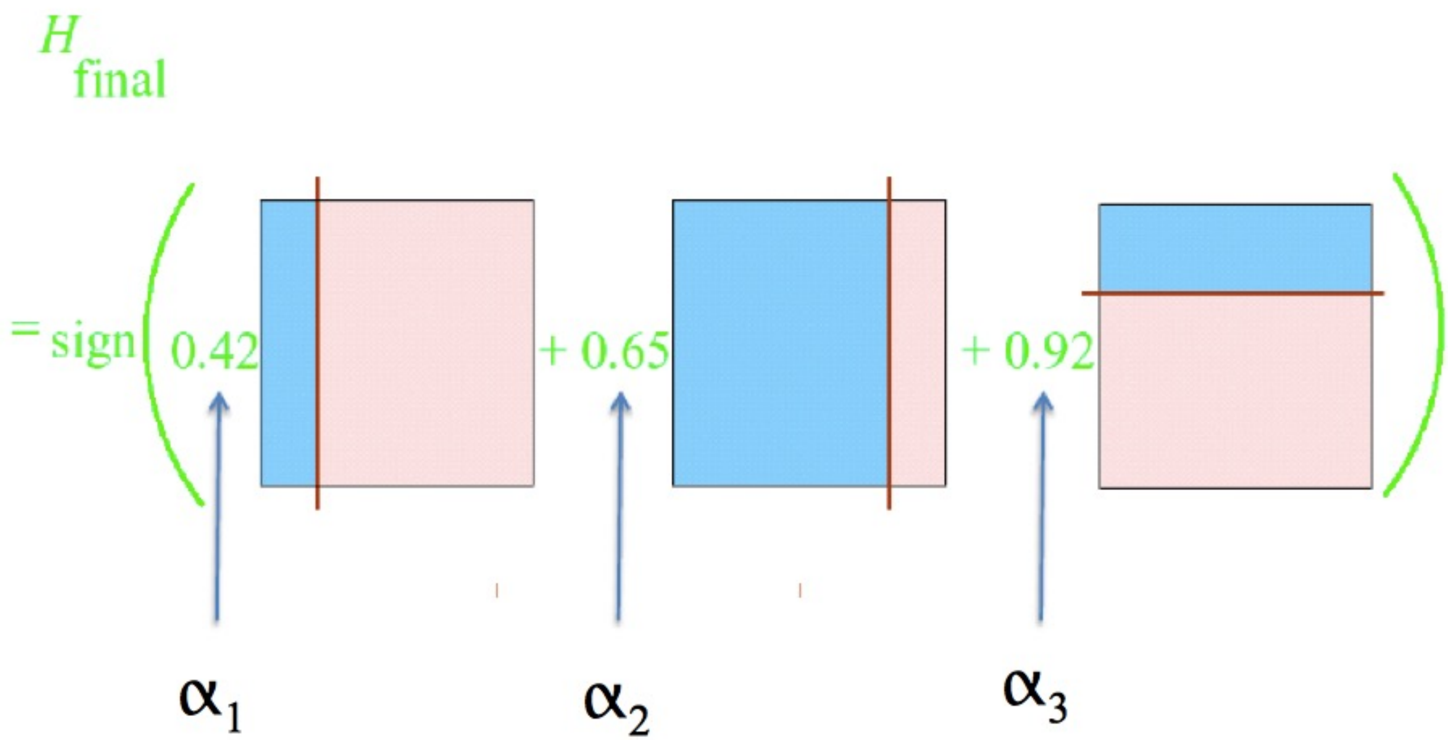
Example: AdaBoost



$$e_3 = 0.14$$
$$\alpha_3 = 0.92$$

Example: AdaBoost

Final Model (Hypothesis):

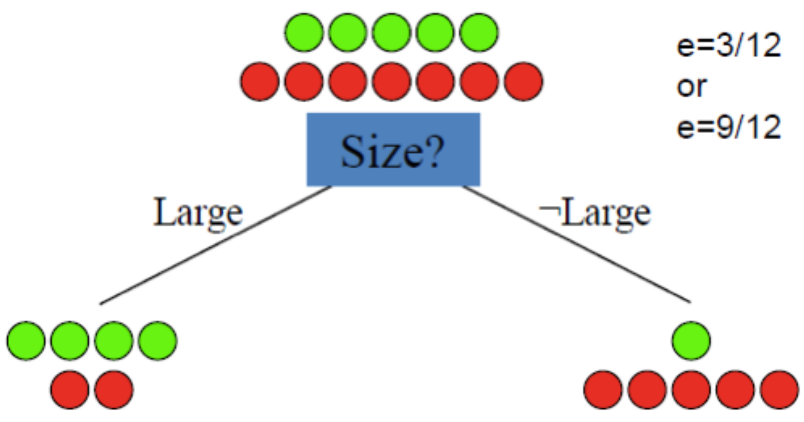


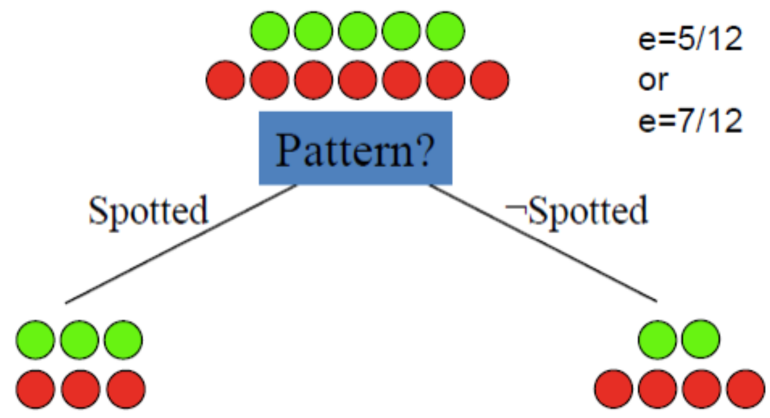
Example: AdaBoost Mushroom

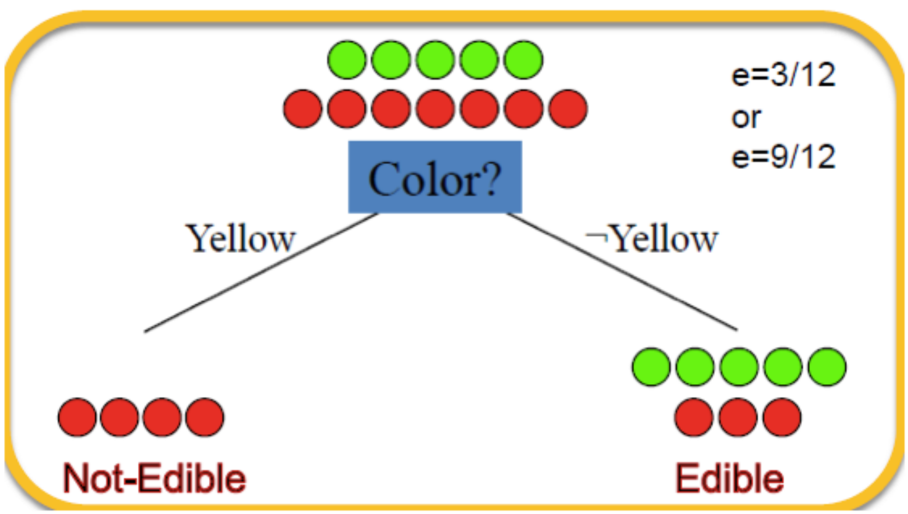
- Mushroom data set used
- Initial weights = 1/12 = 0.0833

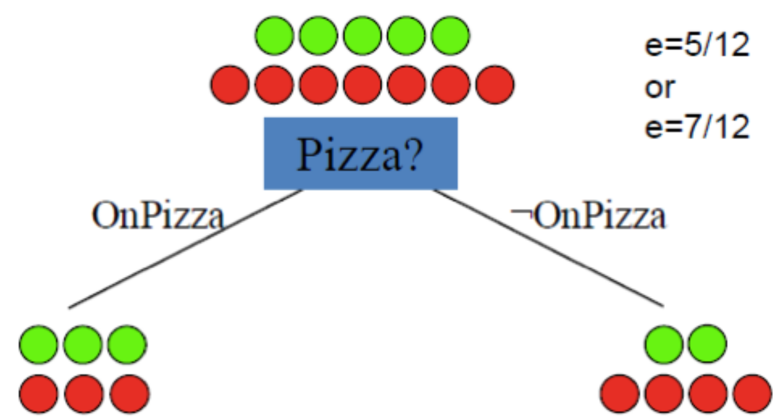
Pattern	Size	Color	OnPizza	Edible
S	L	Y	Y	No
S	L	N	Y	Yes
S	L	N	N	Yes
S	S	Y	N	No
S	S	N	Y	Yes
S	S	N	N	No
N	L	Y	N	No
N	L	N	Y	Yes
N	L	N	N	Yes
N	S	Y	Y	No
N	S	N	Y	No
N	S	N	N	No

Boosting on Features









Boosting on Features

- Compute Weights
- Decision stump is on Color:
  - Yellow = not edible, not yellow = edible

$$e_1 = \sum w(incorrect) = 1/12 + 1/12 + 1/12 = 1/4$$

$$\alpha_1 = \frac{1}{2}\ln\frac{3}{1} = 0.55$$

$$w(correct) = w(incorrect) = 1/12 = 0.0833$$

$$w'(correct) = 0.0555$$

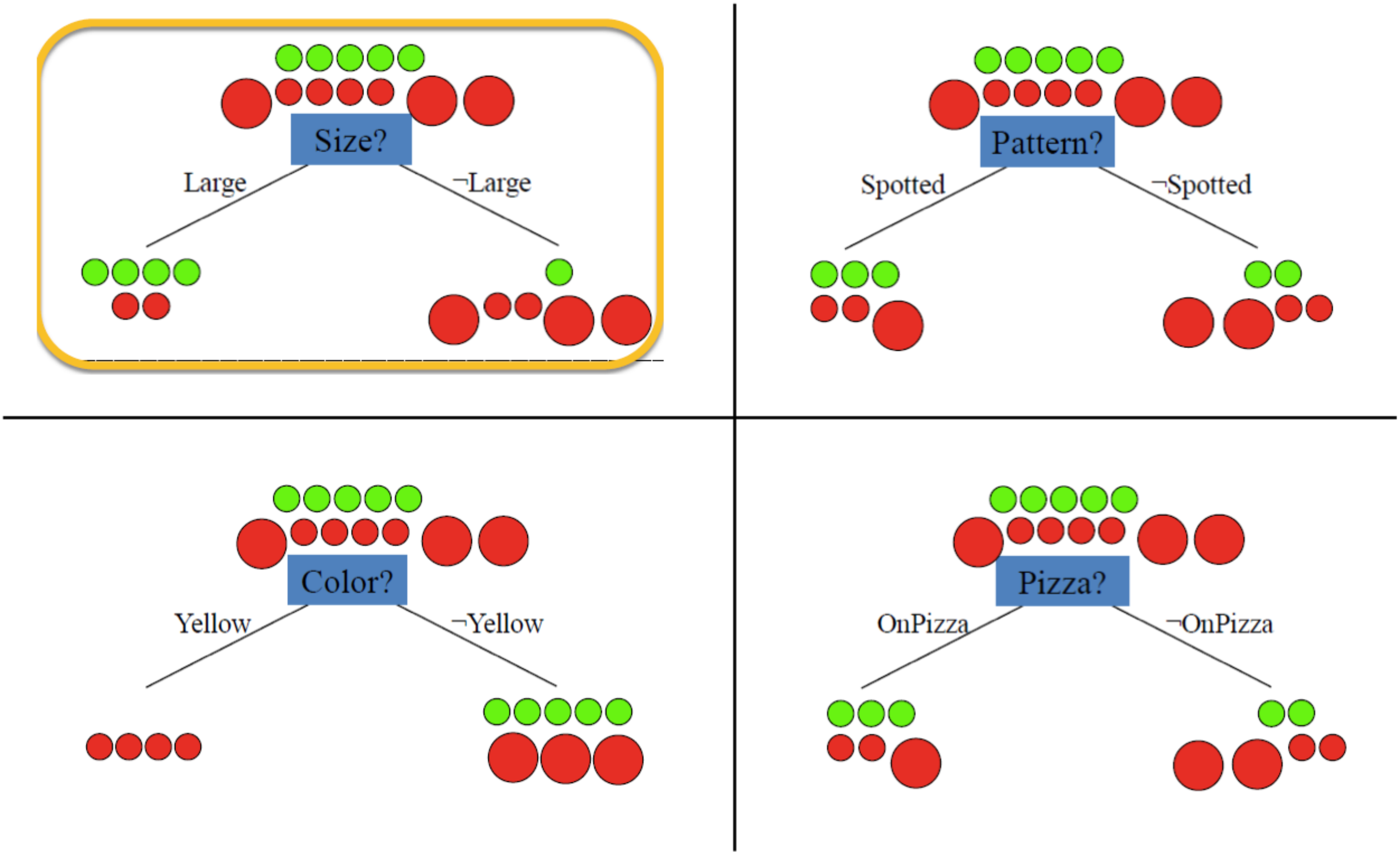
$$w'(incorrect) = 0.1666$$

Example: Next Iteration

- Weighted data

Pattern	Size	Color	OnPizza	Edible	
S	L	Y	Y	No	.0555
S	L	N	Y	Yes	.0555
S	L	N	N	Yes	.0555
S	S	Y	N	No	.0555
S	S	N	Y	Yes	.0555
S	S	N	N	No	.1666
N	L	Y	N	No	.0555
N	L	N	Y	Yes	.0555
N	L	N	N	Yes	.0555
N	S	Y	Y	No	.0555
N	S	N	Y	No	.1666
N	S	N	N	No	.1666

Boosting on Features (2)



Boosting on Features (2)

- Compute Weights
- Decision stump is for size
  - Large = Edible, not large = not edible

$$e_2 = \sum w(incorrect) = (2 * 0.0555) + (1 * 0.0555) = 0.1665$$

$$\alpha_2 = \frac{1}{2} \ln(\frac{0.8335}{0.1665}) = 0.80$$

Example: Boosting

- ESL p. 339, data with n=1000 points

- A single stump produces a misclassification rate of 45.8%

- With boosting with 400 iterations, the misclassification rate is 5.8%

- This also beats the misclassification rate of a single tree – 24.7%

Example: Boosting