CS60020: Foundations of Algorithm Design and Machine Learning

Sourangshu Bhattacharya

BAGGING

Ensemble methods

- A single decision tree does not perform well
- But, it is super fast
- What if we learn multiple trees?

We need to make sure they do not all just learn the same

Bagging

If we split the data in random different ways, decision trees give different results, **high variance.**

Bagging: Bootstrap **agg**regat**ing** is a method that result in low variance.

If we had multiple realizations of the data (or multiple samples) we could calculate the predictions multiple times and take the average of the fact that averaging multiple onerous estimations produce less uncertain results

Bagging

Say for each sample *b*, we calculate $f^b(x)$, then:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

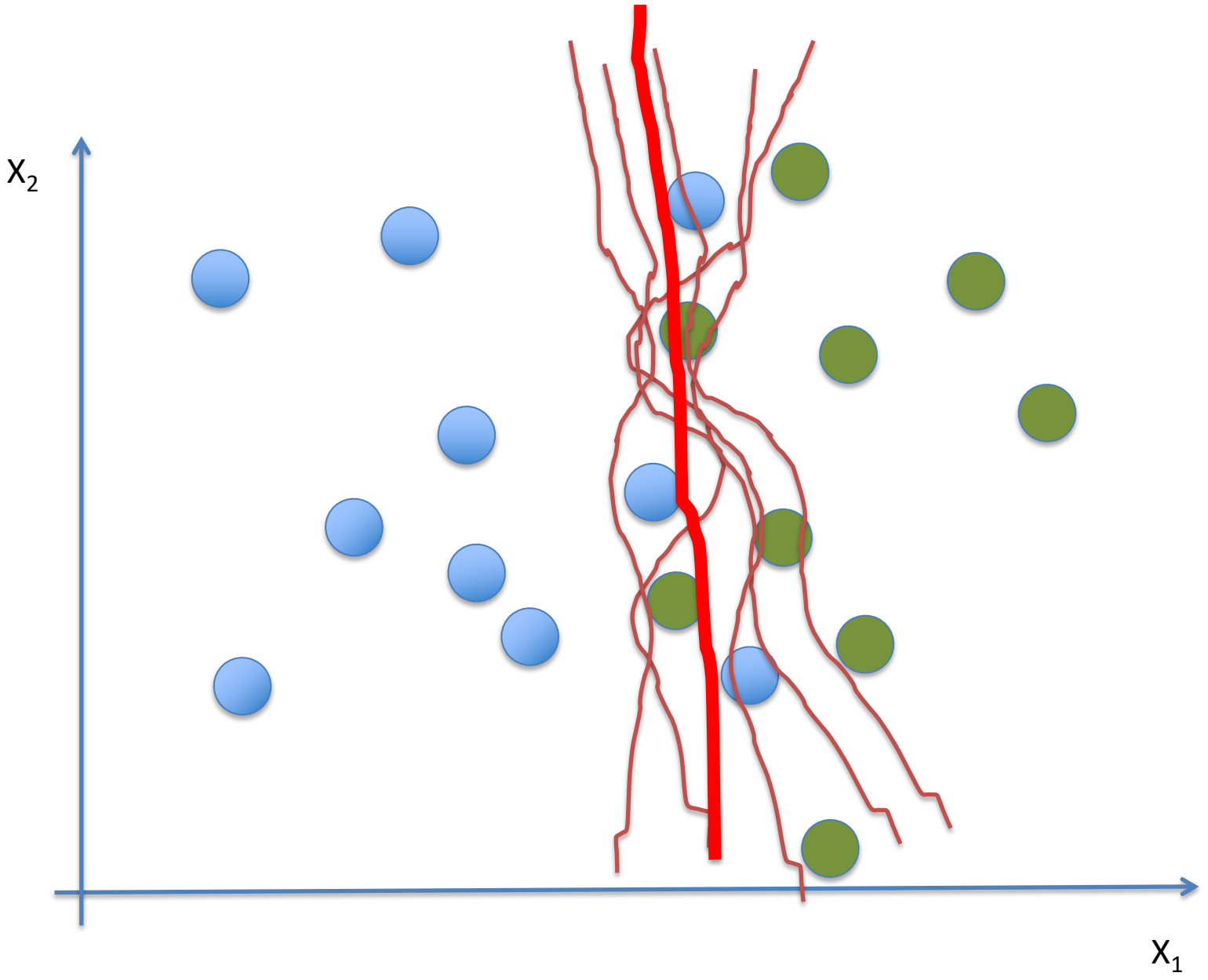
How?

Bootstrap

Construct B (hundreds) of trees (no pruning)

Learn a classifier for each bootstrap sample and average them

Very effective


X_2
X_1

Out-of-Bag Error Estimation

- No cross validation?
- Remember, in bootstrapping we sample with replacement, and therefore **not all observations are used for each bootstrap sample**. On average 1/3 of them are not used!
- We call them out-of-bag samples (OOB)
- We can predict the response for the *i-th* observation using each of the trees in which that observation was OOB and do this for *n* observations
- Calculate overall OOB MSE or classification error

Bagging

- Reduces overfitting (variance)
- Normally uses one type of classifier
- Decision trees are popular
- Easy to parallelize

Bagging - issues

Each tree is identically distributed (i.d.)

➔ the expectation of the average of *B* such trees is the same as the expectation of any one of them

➔the bias of bagged trees is the same as that of the individual trees

i.d. and not i.i.d

Bagging - issues

An average of *B* i.i.d. random variables, each with variance σ^2, has variance*: σ^2/B*

If i.d. (identical but not independent) and pair correlation ρ is present, then the variance is:

$$\rho\,\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

As *B* increases the second term disappears but the first term remains

Why does bagging generate correlated trees?

Bagging - issues

Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.

Then all bagged trees will select the strong predictor at the top of the tree and therefore all trees will look similar.

How do we avoid this?

RANDOM FORESTS

Random Forests

As in bagging, we build a number of decision trees on bootstrapped training samples each time a split in a tree is considered, a random sample of *m* predictors is chosen as split candidates from the full set of p predictors.

Note that if $m = p$, then this is bagging.

Random Forests

Random forests are popular. Leo Breiman's and Adele Cutler maintains a random forest website where the software is freely available, and of course it is included in every ML/STAT package

http://www.stat.berkeley.edu/~breiman/RandomForests/

Random Forests Algorithm

For b = 1 to B:

(a) Draw a bootstrap sample Z^* of size *N* from the training data.

(b) Grow a random-forest tree to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $\boldsymbol{n_{min}}$ is reached.

i. Select ***m*** variables at random from the *p* variables.

ii. Pick the best variable/split-point among the *m*.

iii. Split the node into two daughter nodes.

Output the ensemble of trees.

To make a prediction at a new point *x* we do:

For regression: average the results

For classification: majority vote

Random Forests Tuning

The inventors make the following recommendations:

- For classification, the default value for m is $\sqrt{p}$ and the minimum node size is one.
- For regression, the default value for m is $p/3$ and the minimum node size is five.

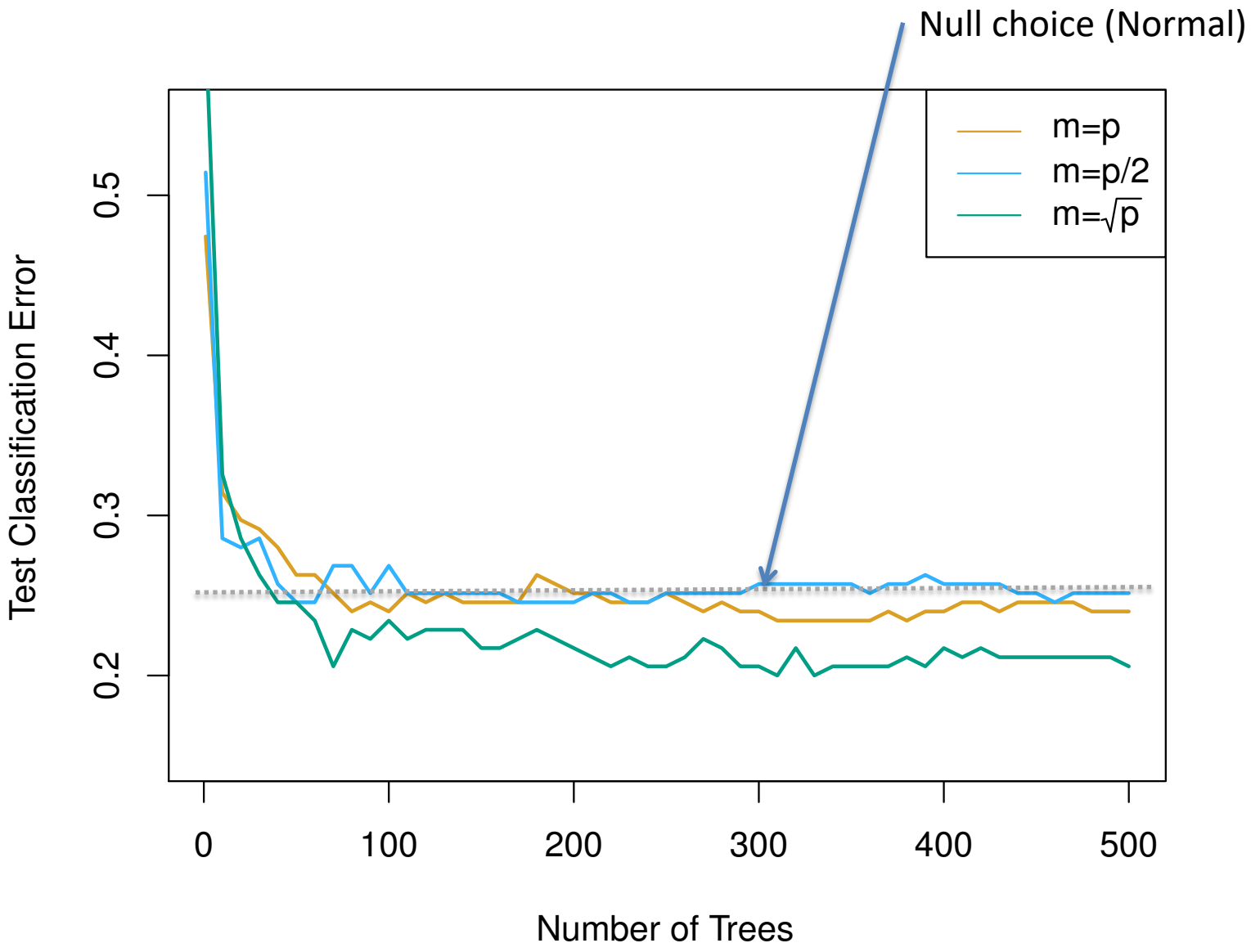
In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters.

Like with Bagging, we can use OOB and therefore RF can be fit in one sequence, with cross-validation being performed along the way. Once the OOB error stabilizes, the training can be terminated.

Example

- 4,718 genes measured on tissue samples from 349 patients.
- Each gene has different expression
- Each of the patient samples has a qualitative label with 15 different levels: either normal or 1 of 14 different types of cancer.

Use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.


Null choice (Normal)
m=p
m=p/2
m=√p
Test Classification Error
0.2
0.3
0.4
0.5
0
100
200
300
400
500
Number of Trees

Random Forests Issues

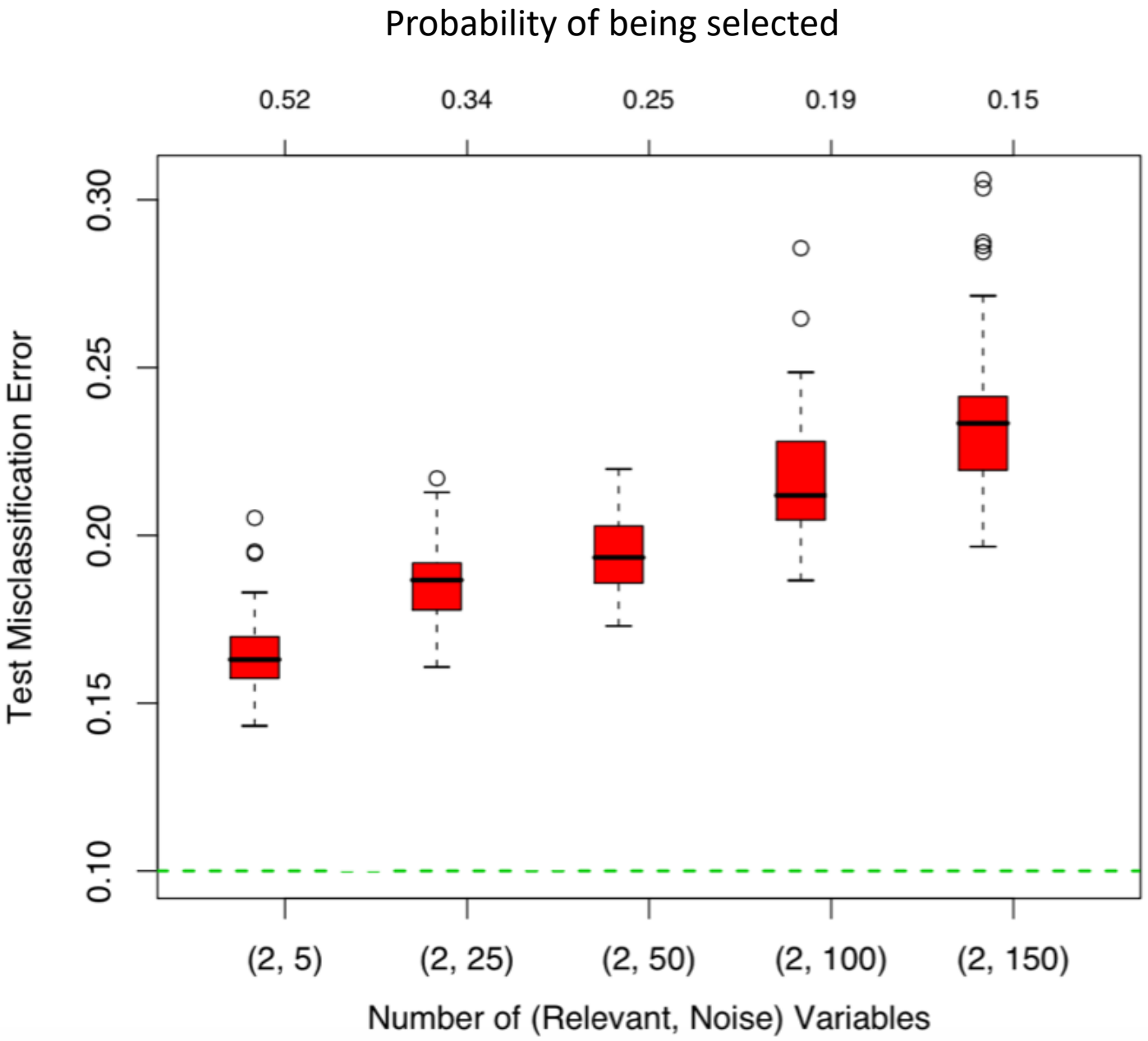
When the number of variables is large, but the fraction of relevant variables is small, random forests are likely to perform poorly when *m* is small

Why?

Because:

At each split the chance can be small that the relevant variables will be selected

For example, with 3 relevant and 100 not so relevant variables the probability of any of the relevant variables being selected at any split is ~0.25


Probability of being selected
0.52
0.34
0.25
0.19
0.15
Test Misclassification Error
0.10
0.15
0.20
0.25
0.30
(2, 5)
(2, 25)
(2, 50)
(2, 100)
(2, 150)
Number of (Relevant, Noise) Variables

Can RF overfit?

Random forests “cannot overfit” the data wrt to number of trees.

Why?

The number of trees, *B* does not mean increase in the flexibility of the model

Constructing Confusion Matrix for multiple classes

For each pair of classes $< c_1, c_2 >$ how many documents from c_1 were incorrectly assigned to c_2? (when $c_2 \neq c_1$)

Docs in test set	Assigned UK	Assigned poultry	Assigned wheat	Assigned coffee	Assigned interest	Assigned trade
True UK	95	1	13	0	1	0
True poultry	0	1	0	0	0	0
True wheat	10	90	0	1	0	0
True coffee	0	0	0	34	3	7
True interest	-	1	2	13	26	5
True trade	0	0	2	14	5	10

Averaging: Micro vs. Macro

- We now have an evaluation measure (F_1) for one class.
- But we also want a single number that measures the aggregate performance over all classes in the collection.
- Macroaveraging
  - Compute F_1 for each of the C classes
  - Average these C numbers
- Microaveraging
  - Compute TP, FP, FN for each of the C classes
  - Sum these C numbers (e.g., all TP to get aggregate TP)
  - Compute F_1 for aggregate TP, FP, FN

23

Micro- vs. Macro-average: Example

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macro-averaged precision: (0.5 + 0.9)/2 = 0.7
- Micro-averaged precision: 100/120 = 0.83

Micro-averaged score is dominated by score on common classes

BOOSTING

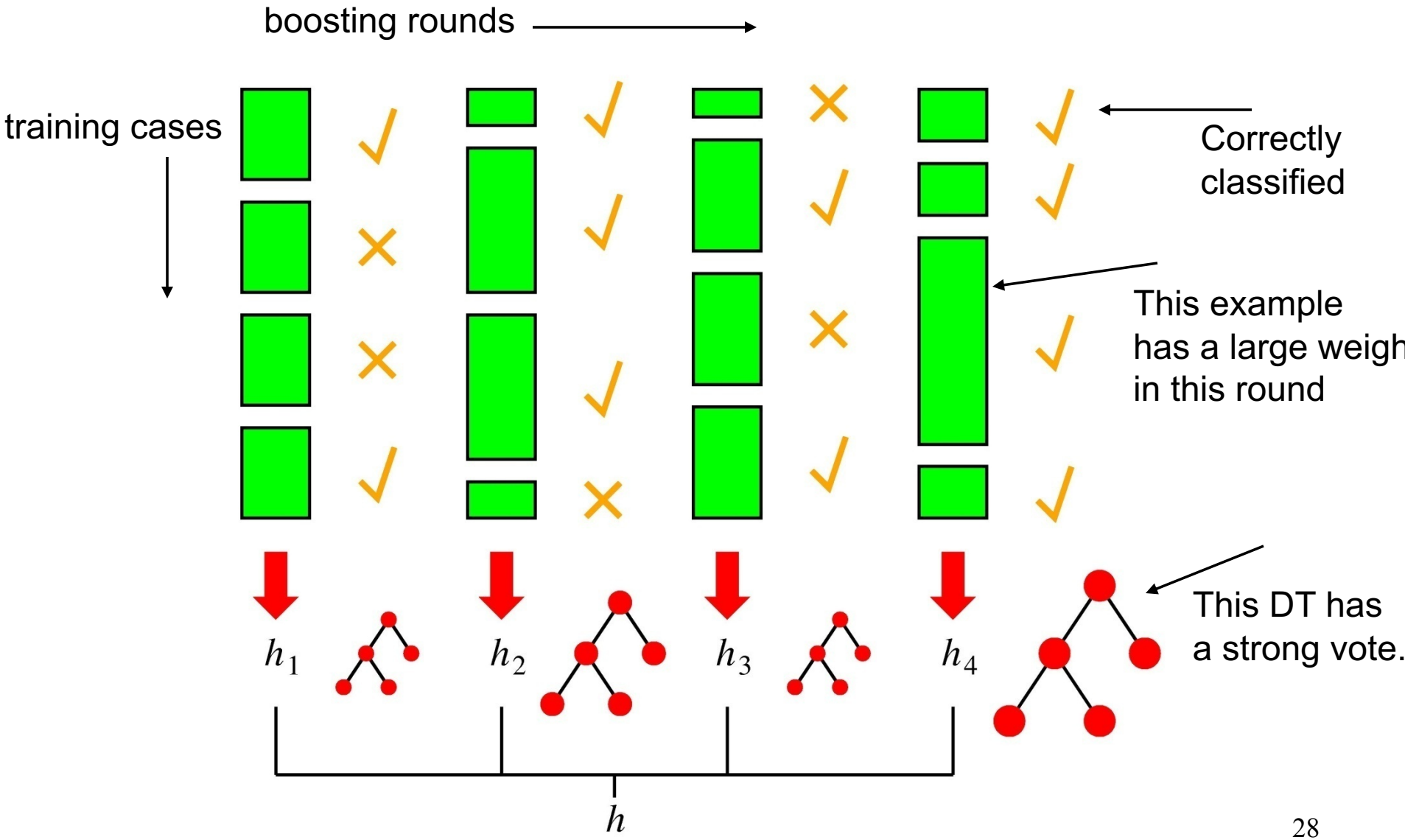
Boosting

- Train classifiers (e.g. decision trees) in a sequence.
- A new classifier should focus on those cases which were incorrectly classified in the last round.
- Combine the classifiers by letting them vote on the final prediction (like bagging).
- Each classifier is “weak” but the ensemble is “strong.”
- AdaBoost is a specific boosting method.

Boosting Intuition

- We adaptively weigh each data case.
- Data cases which are wrongly classified get high weight (the algorithm will focus on them)
- Each boosting round learns a new (simple) classifier on the weighed dataset.
- These classifiers are weighed to combine them into a single powerful classifier.
- Classifiers that that obtain low training error rate have high weight.
- We stop by using monitoring a hold out set (cross-validation).

Boosting in a Picture



Boosting

- Combining multiple “base” classifiers to come up with a “good” classifier.
- Base classifiers have to be “weak learners”, accuracy > 50%
- Base classifiers are trained on a weighted training dataset.
- Boosting involves sequentially learning α_m and $y_m(x)$.

Adaboost

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$

   where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

   (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

   and then use these to evaluate

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}.$$

Adaboost (contd..)

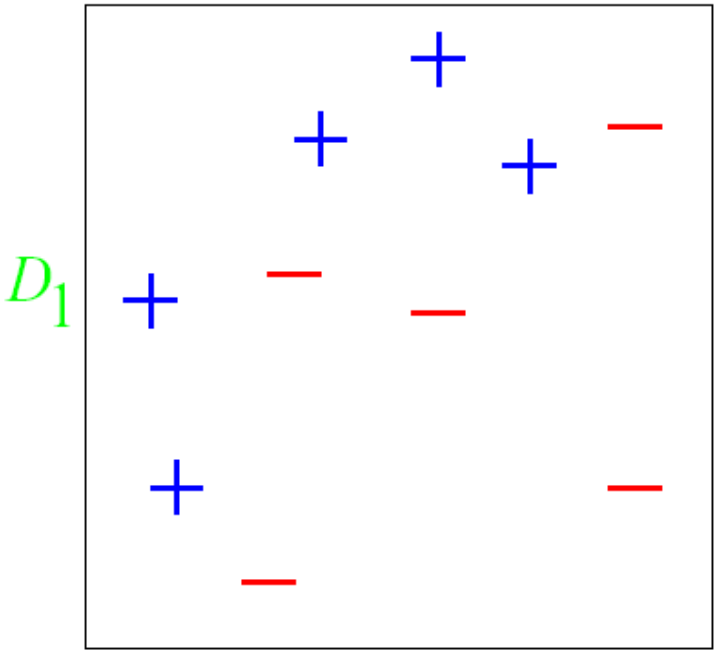
(c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\right\}$$

3. Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right).$$

And in animation

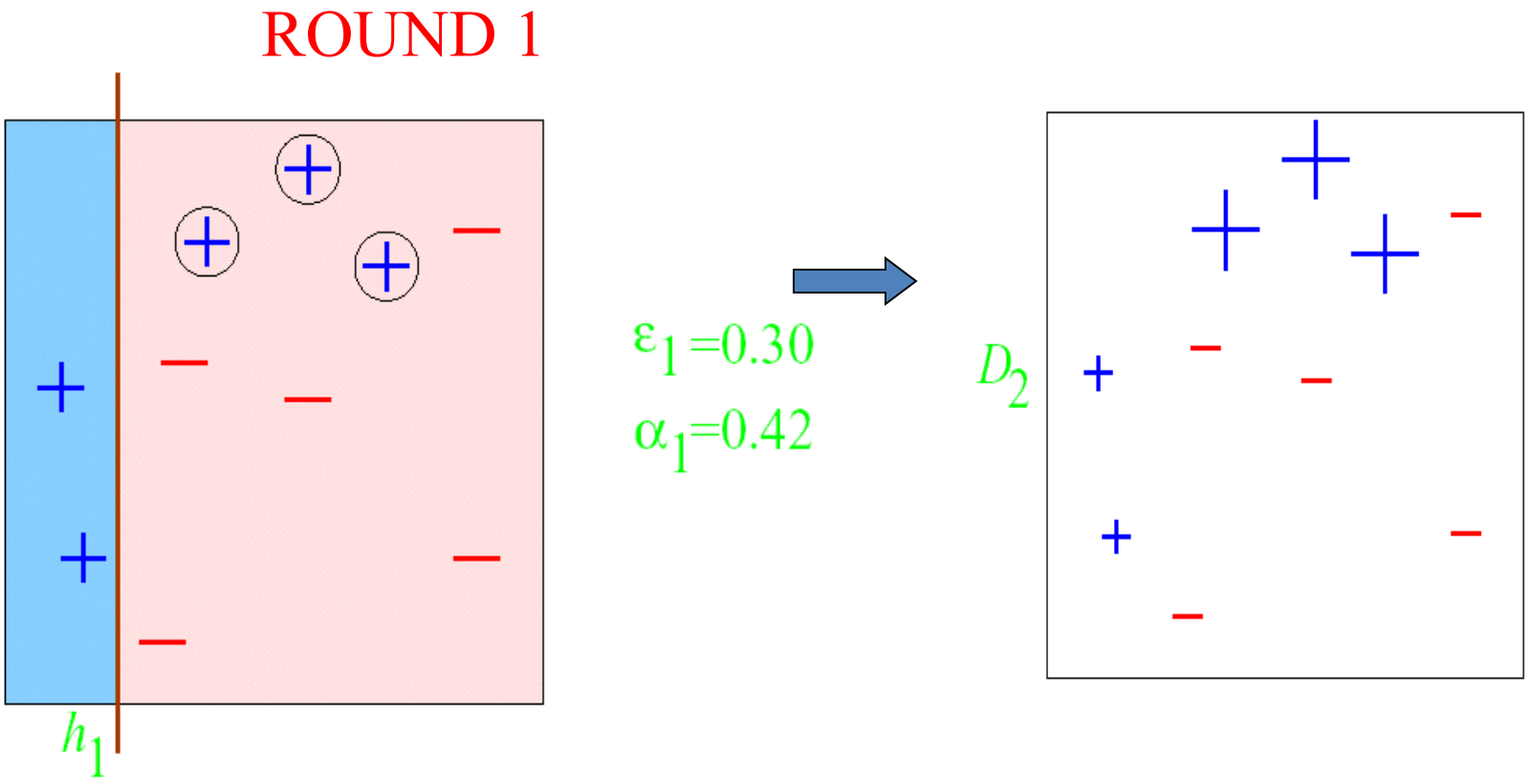


Original training set: equal weights to all training samples

Taken from "**A Tutorial on Boosting**" **by** Yoav Freund and Rob Schapire

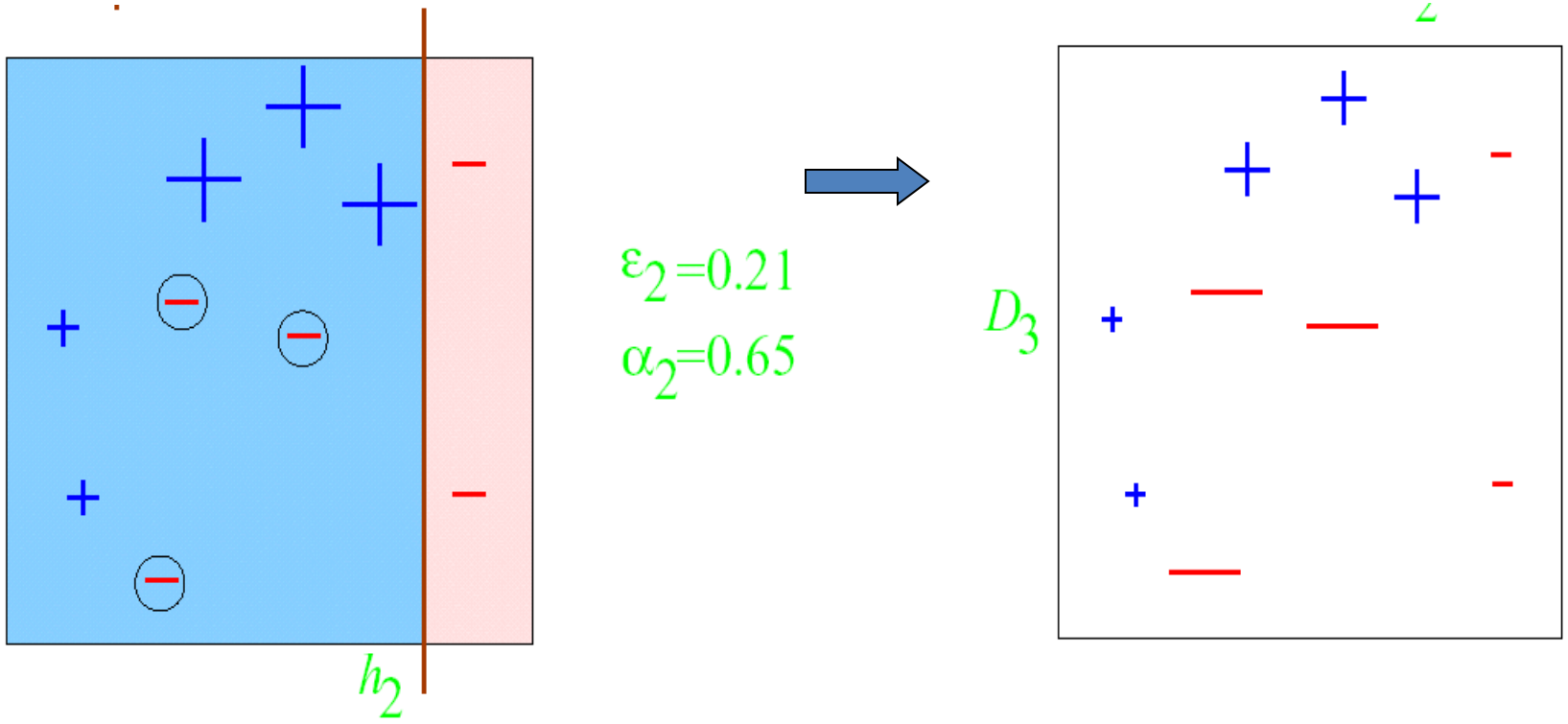
AdaBoost example

ε = error rate of classifier
α = weight of classifier



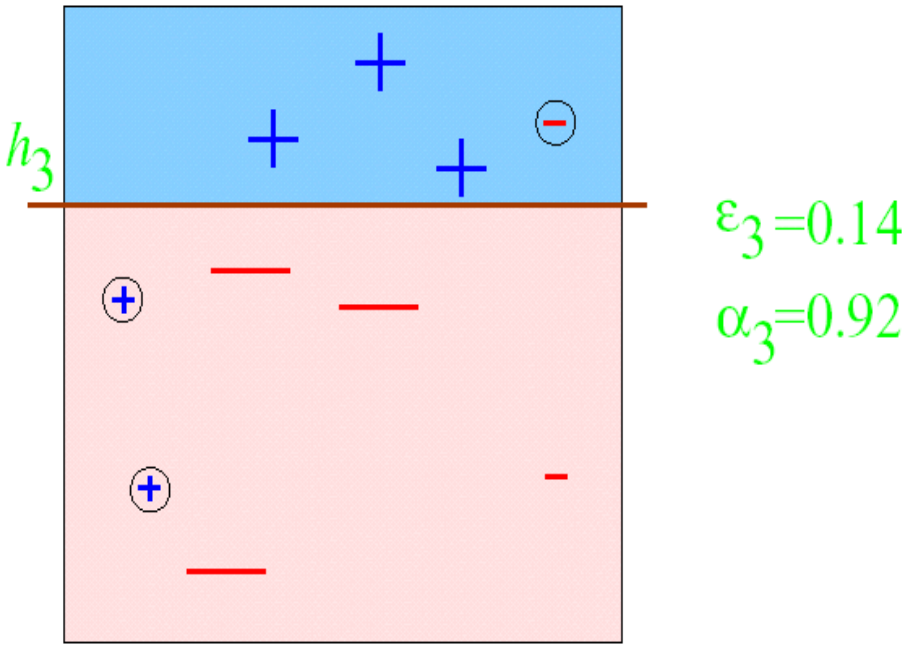
AdaBoost example

ROUND 2



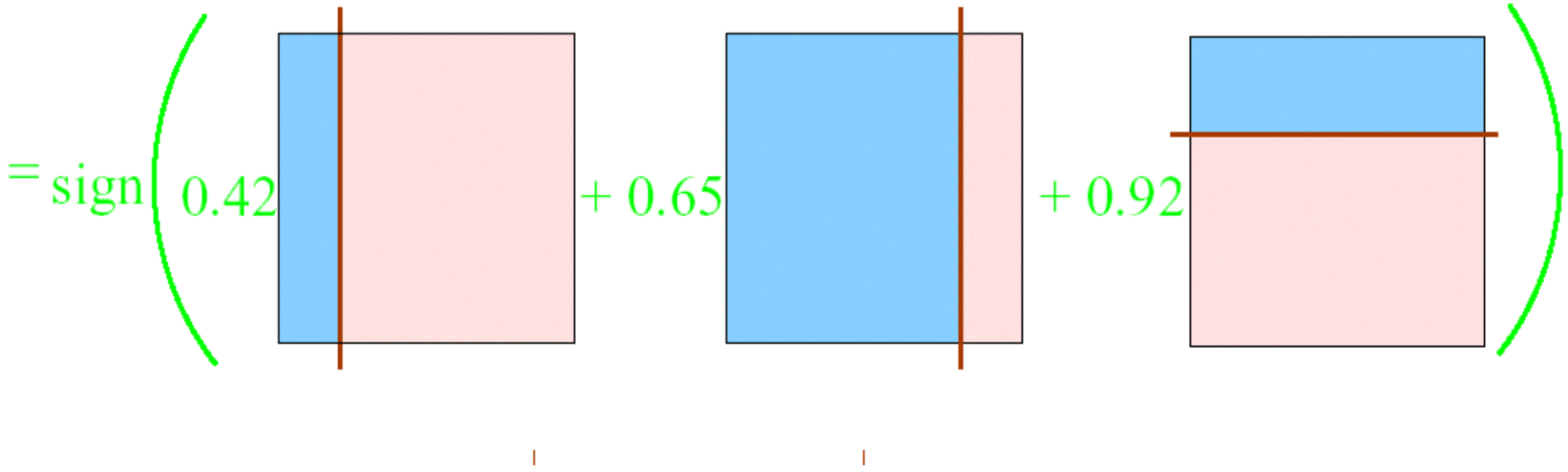
AdaBoost example

ROUND 3

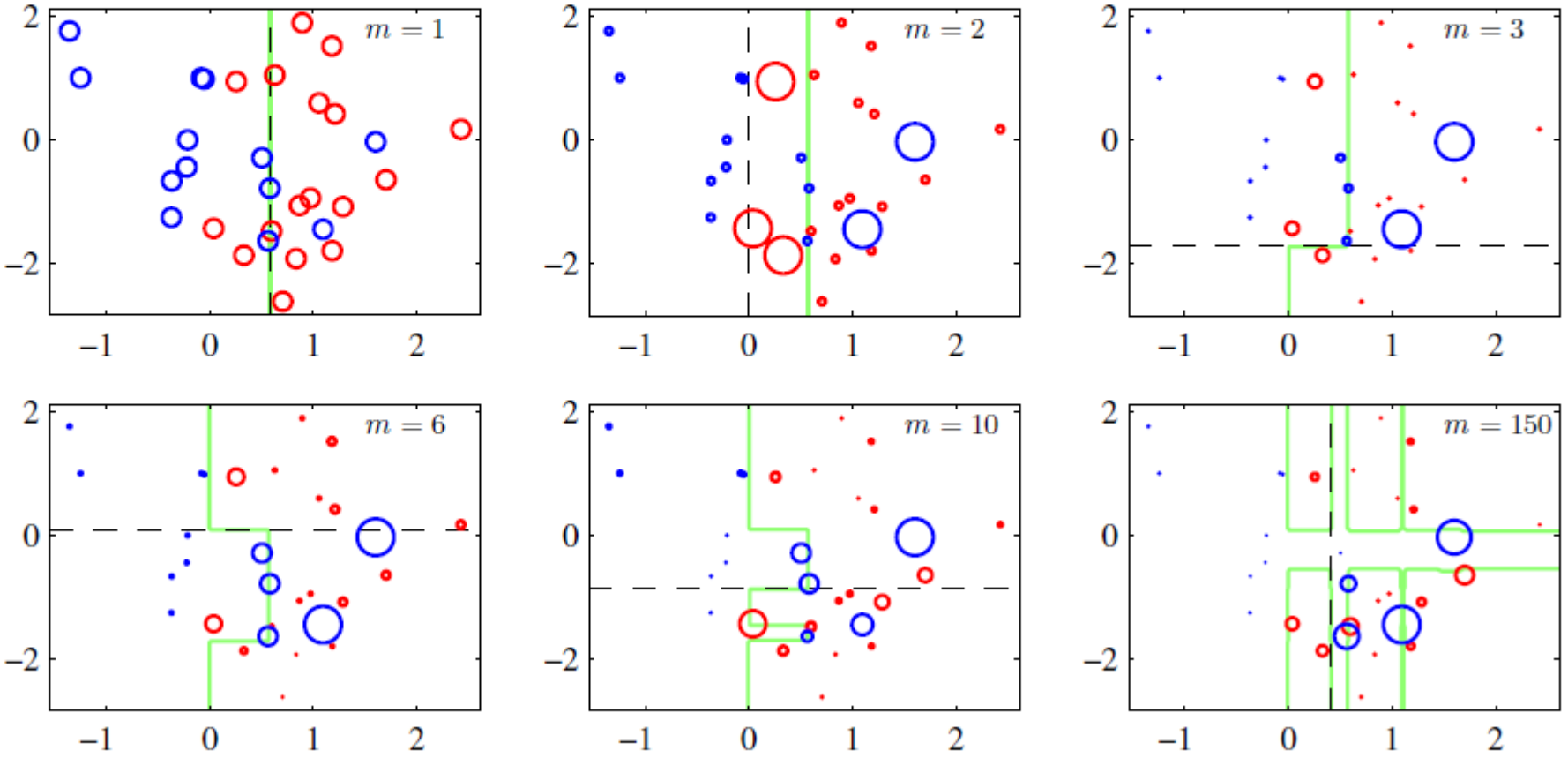


AdaBoost example

H_{final}



Adaboost illustration



Adaboost - Observations

- ϵ_m: weighted error $\in [0,0.5)$
- $\alpha_m \geq 0$
- w_i^{m+1} is higher than w_i^m by a factor $(1 - \epsilon_m)/\epsilon_m$, when *i* is misclassified.

Adaboost - derivation

- Consider the error function:

$$E = \sum_{n=1}^{N} \exp\{-t_n f_m(\mathbf{x}_n)\}$$

- Where

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^{m} \alpha_l y_l(\mathbf{x})$$

- Goal: Minimize E w.r.t. α_l and $y_l(x)$, sequentially.

Adaboost - derivation

- Minimize w.r.t. α_m

$$
\begin{aligned}
E &= \sum_{n=1}^{N} \exp\left\{-t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\} \\
&= \sum_{n=1}^{N} w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\}
\end{aligned}
$$

- Let τ_m be the set of datapoints correctly classified by y_m.

$$
\begin{aligned}
E &= e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \\
&= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}.
\end{aligned}
$$

Adaboost - derivation

- Minimizing w.r.t. y_m and α_m, we get the updates 2(a) and 2(b).
- We can see that:

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\}.$$

- Using:

$$t_n y_m(\mathbf{x}_n) = 1 - 2I(y_m(\mathbf{x}_n) \neq t_n)$$

- We get:

$$w_n^{(m+1)} = w_n^{(m)} \exp(-\alpha_m/2) \exp\left\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\right\}.$$

Adaboost

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$

   where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

   (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

   and then use these to evaluate

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}.$$

Adaboost (contd..)

(c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\right\}$$

3. Make predictions using the final model, which is given by

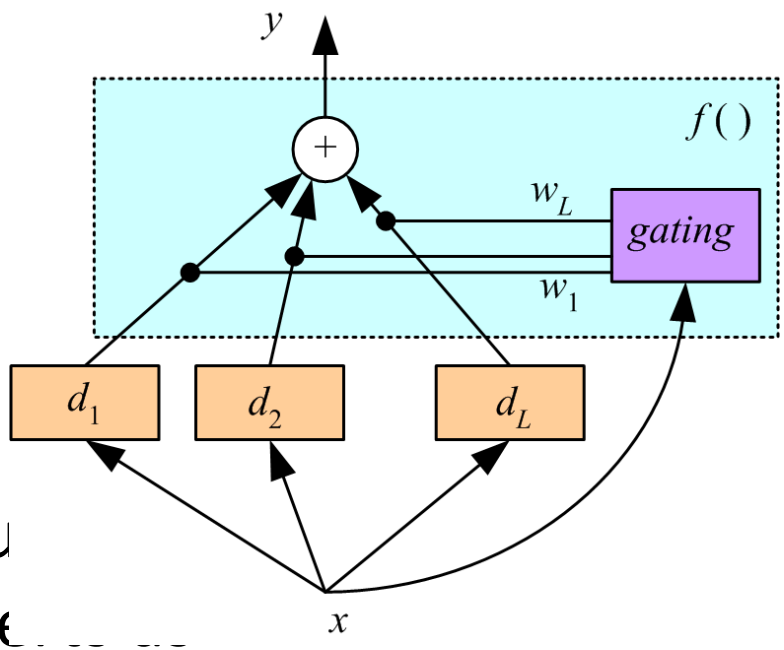
$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right).$$

CONCLUSION

Mixture of experts

- Voting where weights are input-dependent (gating)
- Different input regions convered by different learners (Jacobs et al., 1991)

$$y = \sum_{j=1}^{L} w_j d_j$$



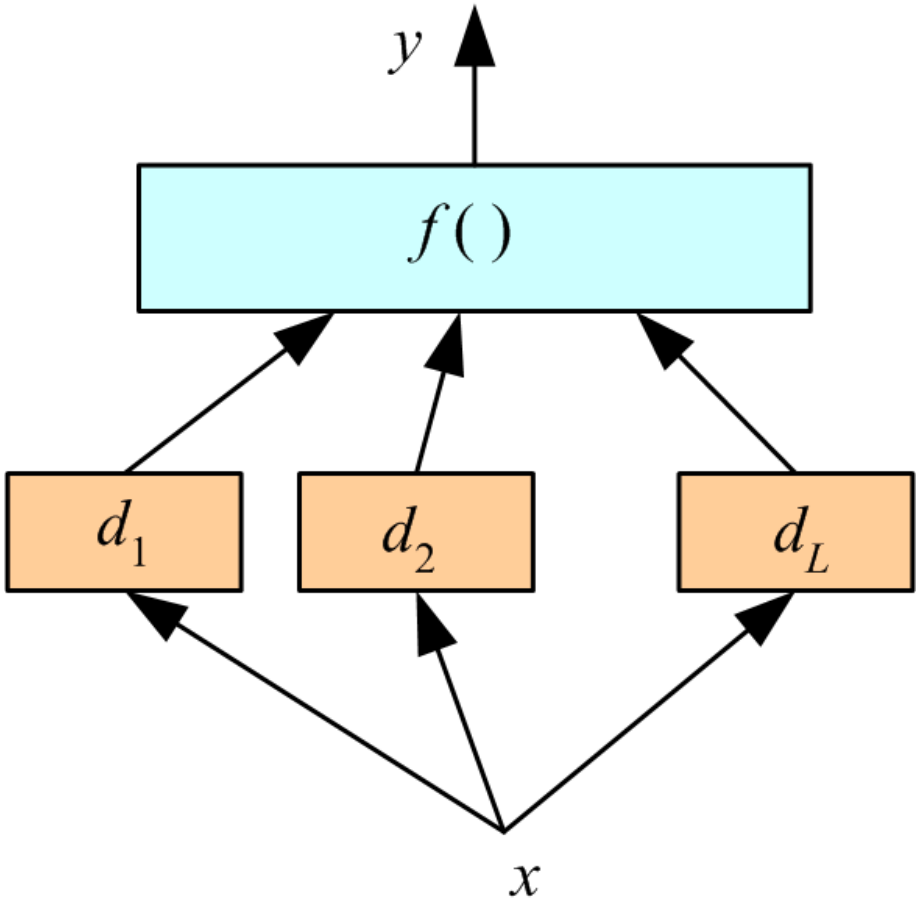
- Gating decides which expert to u
- Need to learn the individual expe

well as the gating functions $w_i(x)$:

$\Sigma w_j(x) = 1$, for all x

Stacking

- Combiner f () is another learner (Wolpert, 1992)



THANKS
QUESTIONS ?