Inspire…Educate…Transform.

Engineering Big Data

Classifier Ensembles

Lt. Suryaprakash Kompalli

ENSEMBLES

Ensembles

- Stacking
  - Combine multiple classifiers
- Bagging
  - Split data and create multiple classifiers on different training data
- Boosting
  - Boost the ability of a classifier to learn specific samples in the dataset

Build Ensemble Classifiers

- Basic idea:
  Build different “experts”, and let them vote

- Advantages:
  Improve predictive performance
  Other types of classifiers can be directly included
  Easy to implement
  Not much parameter tuning

- Disadvantages:
  The combined classifier is not so transparent (black box)
  Not a compact representation

Why do they work?

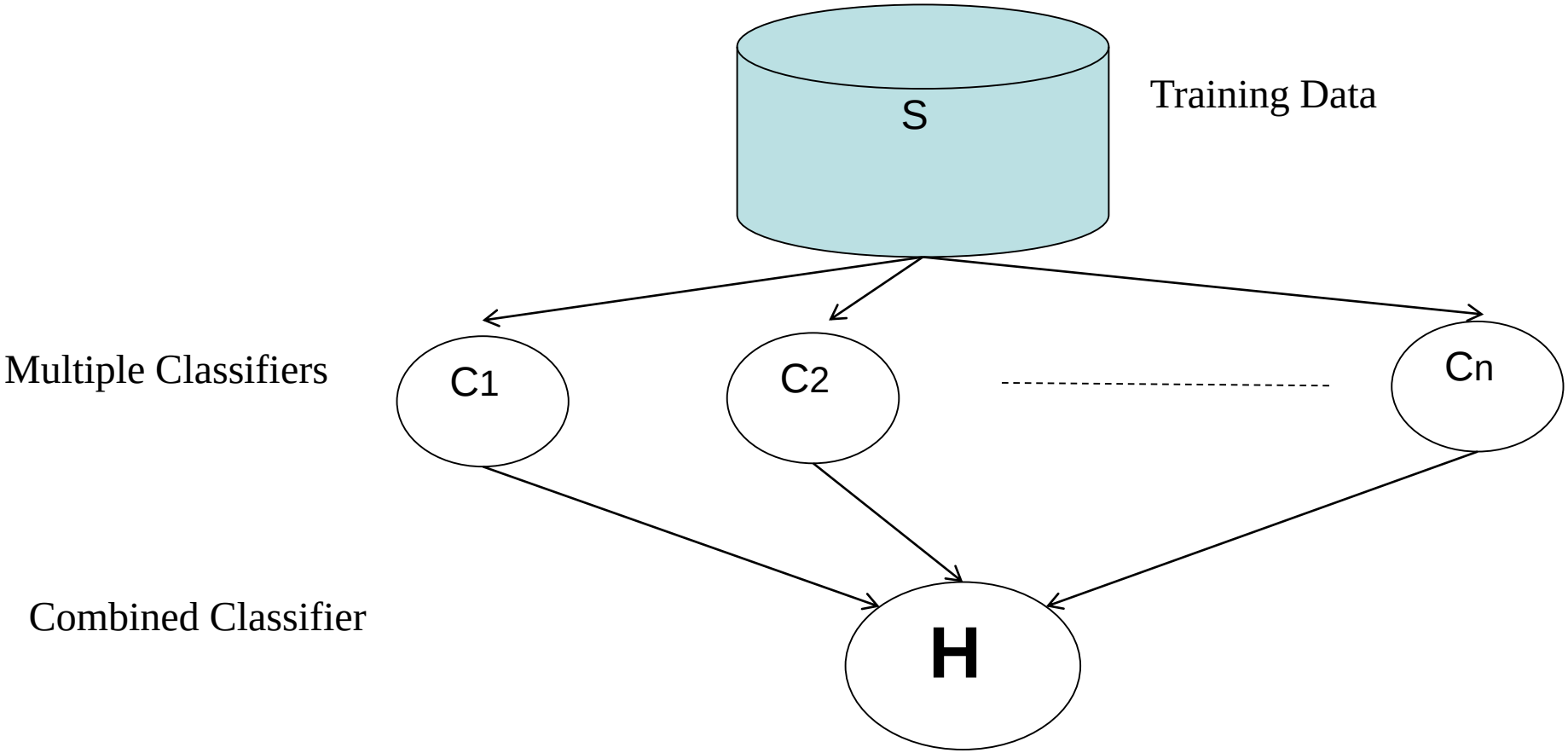
- Suppose there are 25 base classifiers
- Each classifier has error rate, $\varepsilon = 0.35$
- Assume independence among classifiers
- Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^{i} (1-\varepsilon)^{25-i} = 0.06$$

$$\sum_{i=13}^{25} \frac{25!}{i!(25-i)!} \varepsilon^{i} (1-\varepsilon)^{25-i} = 0.06$$

13 of the 25 classifiers have to make an error

Stacking



- Use simple classifiers at Combination stage. Example: Logistic regression, voting, mean

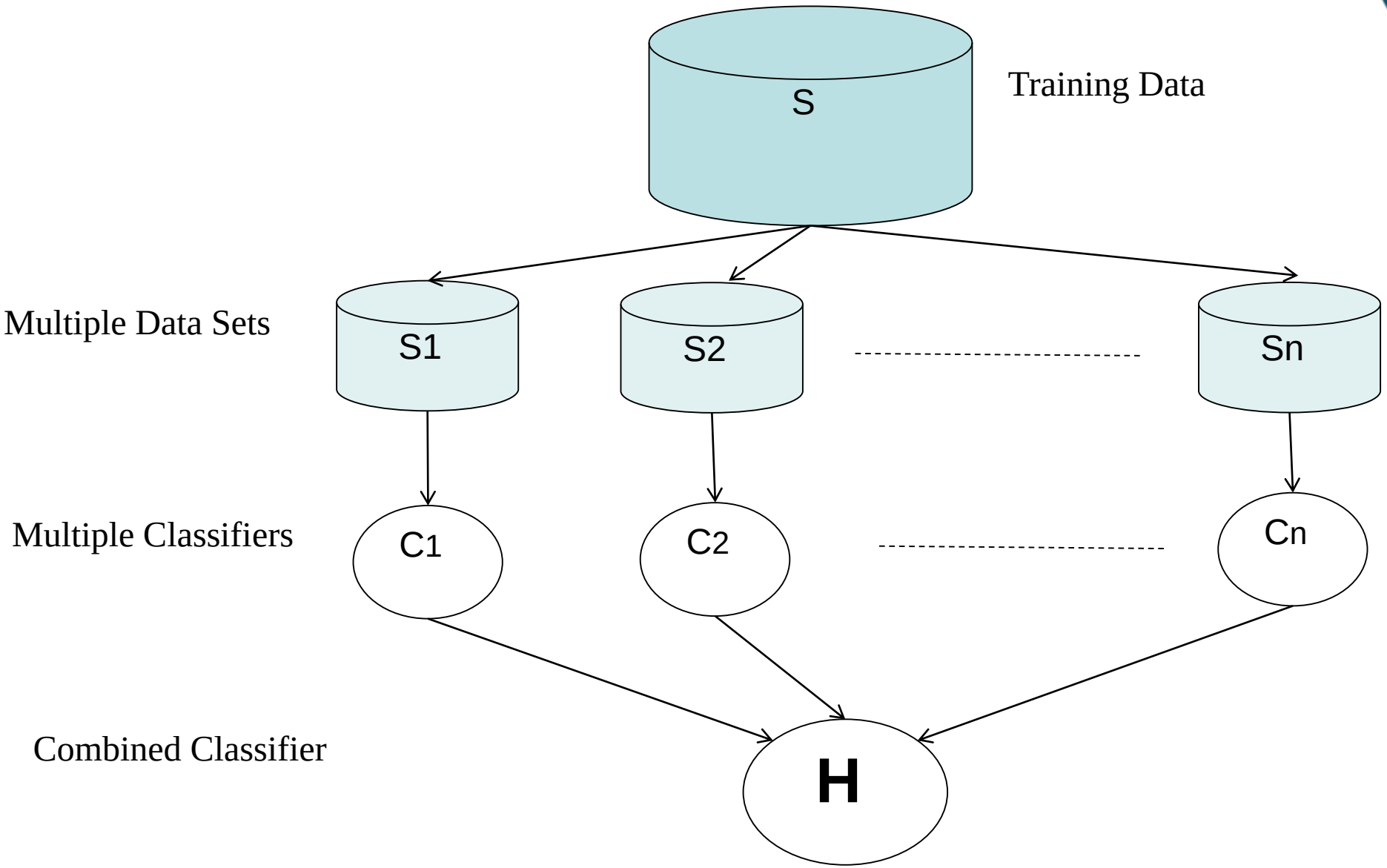
Bagging

- Training
  - Given a dataset S, at each iteration i, a training set S_i is sampled with replacement from S (i.e. bootstraping)
  - A classifier C_i is learned for each S_i

- Classification: given an unseen sample X,
  - Each classifier C_i returns its class prediction
  - The bagged classifier H counts the votes and assigns the class with the most votes to X

- Regression: can be applied to the prediction of continuous values by taking the average value of each prediction.

Bagging



Bagging

- Bagging is good for unstable learners as it reduces variance and overfitting
  - How do I generate a large number of unstable learners
    - Choose records randomly
    - Choose attributes randomly
- What is an unstable learner?
  - Logistic regression
  - Neural networks
  - Decision tree

Logisitic regression is biased towards data that shows linear behavior
DT is unbiased, but the tree can be very different even if a few samples change
Neural Network is not biased and is not sensitive to data variance. More stable

Bagging

- Logistic regression, Linear regression
  - fail with bootstrap of records
  - but ok for bootstrap of features
- Rules, decision trees
  - should work wonders

These apply a biased model to data

These are unbiased but can select best attribute

CSE 7405c

Random forests

- Select a large number of data sets through bagging (same number of samples in all sets)
- Use m random input variables at each node of each tree. m should be much less than M (total attributes)
- Each tree is fully grown and not pruned
- Mode (for classification) or average for regression of all the trees is used as prediction.

CSE 7405c

The Random Forests Algorithm

Given a training set S

For i = 1 to k do:

Build subset S_i by sampling with replacement from S

Learn tree T_i from Si

At each node:

Choose best split from random subset of F features

Each tree grows to the largest extent, and no pruning

Make predictions according to majority vote of the set of k trees.

Random Forests

- It is one of the most accurate learning algorithms available
- Accuracy would be best when the trees are least correlated and each one is strong
- Q: How to do attribute selection?
  - You built 200 trees in a random forest, and 180 trees selected CCAvg as feature at first node; 130 trees did not use Family in any of the nodes.
  - What if you counted the level at which a feature was used?

http://stats.stackexchange.com/questions/68692/feature-selection-with-random-forests
https://hal.archives-ouvertes.fr/hal-00755489/file/PRLv4.pdf

Ensemble Methods

- *Random forests (also true for many machine learning algorithms) is an example of a tool that is useful in doing analyses of scientific data.*
- *But the cleverest algorithms are no substitute for human intelligence and knowledge of the data in the problem.*
- *Take the output of random forests not as absolute truth, but as smart computer generated guesses that may be helpful in leading to a deeper understanding of the problem.*

Leo Brieman

Adaboost

BOOSTING

Boosting: How do classifiers learn?

Can we boost or enhance the learning of some samples?

Count some samples multiple times

- Decision Trees:
  - Entropy computed by number of times an attribute leads to target variable
- Neural network:
  - Backpropagation using $(output_i - true_i)$

Change to $Importance_i(output_i - true_i)$

- SVM:
  - Optimization problem for $\max\left(\alpha_i \alpha_j y_i y_j (x_i . x_j)\right)$

Add constraints to some of the α_i

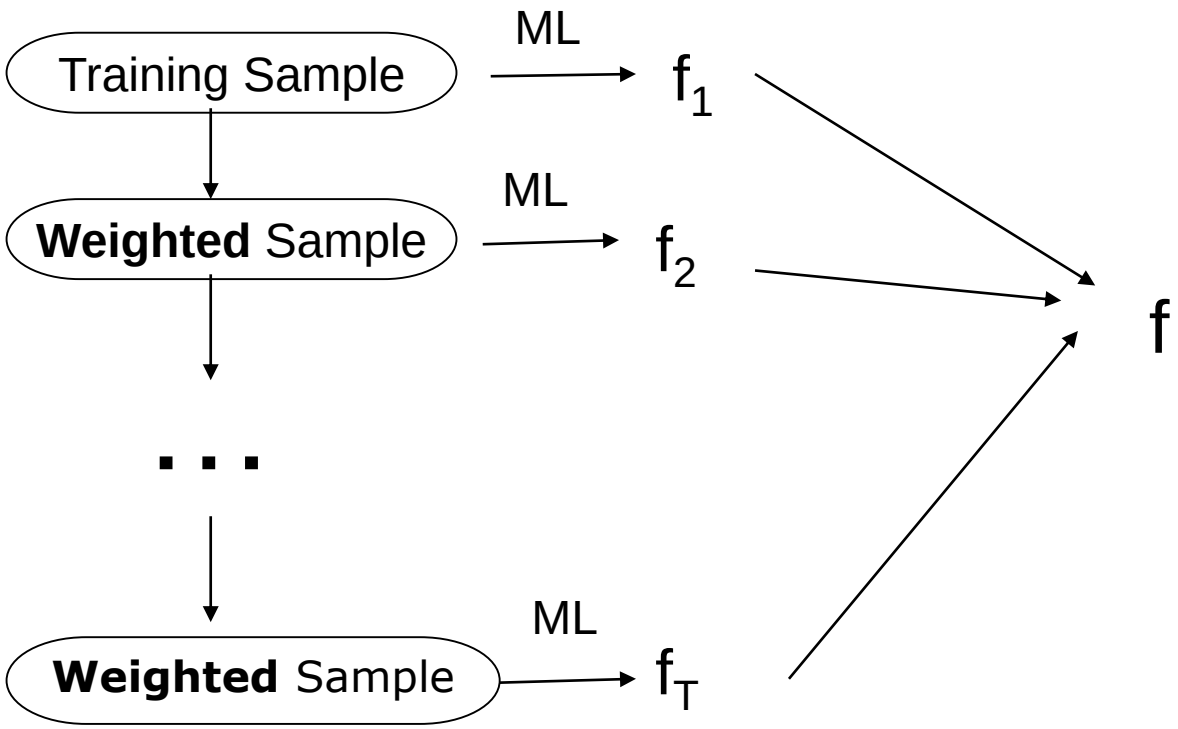
CSE 7405c

Boosting: Is weak learning related to strong learning?

- If the weak learner gives accuracy on all possible distributions, yes.

- Put together an algorithm
  - Do a classifier
  - Learn the second classifier that works better on instances where the first one failed
  - Third one where both failed

Boosting



Q: Can we use the final trained classifier (f_T)?

Adaboost: Intuition

- Train a set of weak hypotheses: h_1, …., h_T.
- During the training, focus on the examples that are misclassified.

➔At round 1, all samples have equal probability of being picked (1/m)

Algorithm recapitulation

Initial importance of each point $D_1(i)$ = 1/n, n: Number of samples

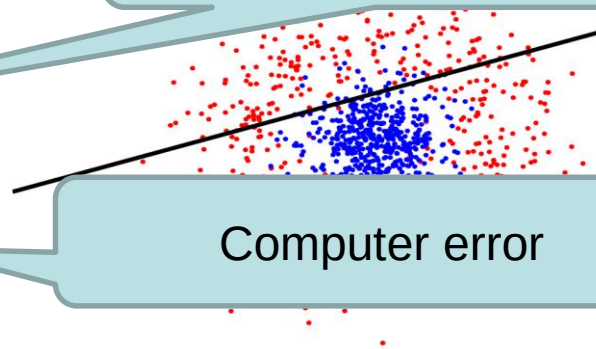
- In each iteration:
  - Find model $h_t = \arg min_{h_j \in H} e_j$
  - $e_j = \sum_{i=1}^{n} D_t(i)[y_i \neq h_j(x_i)]$
  - If $e_t > 0.5$ then stop
  - Set $\alpha_t = 0.5\log(\frac{1+e_t}{1-e_t})$
  - Update: $\widehat{D}_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

$$D_{t+1}(i) = \frac{\widehat{D}_{t+1}(i)}{\sum_{i=1\ldots m} \widehat{D}_{t+1}(i)}$$

- Final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t\, h_t(x)\right)$$

Get classification result



Computer error

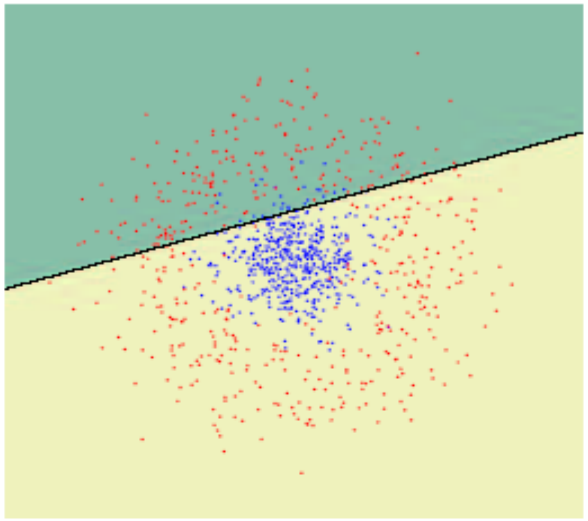
Increase importance of items that are mis-classified $y_i h_t(x_i) < 1$

Normalize the importance. Use these in next classifier

CSE 7405c

Algorithm recapitulation

$t = 1$



- In each iteration:
  - Find model $h_t = \arg min_{h_j \in H} e_j$

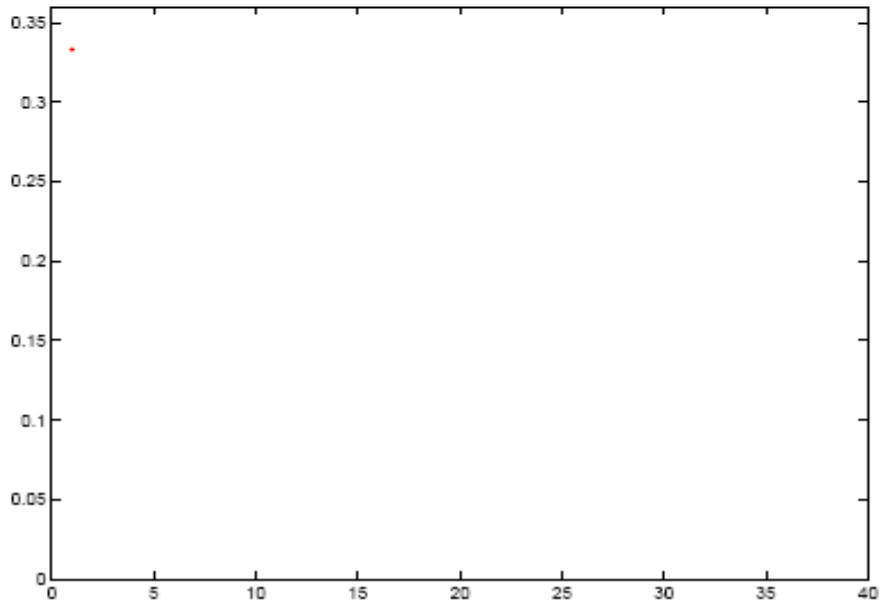
$$e_j = \sum_{i=1}^{n} D_t(i)[y_i \neq h_j(x_i)]$$

  - If $e_t > 0.5$ then stop
  - Set $\alpha_t = 0.5\log(\frac{1+e_t}{1-e_t})$
  - Update: $\widehat{D}_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

$$D_{t+1}(i) = \frac{\widehat{D}_{t+1}(i)}{\sum_{i=1\ldots m} \widehat{D}_{t+1}(i)}$$

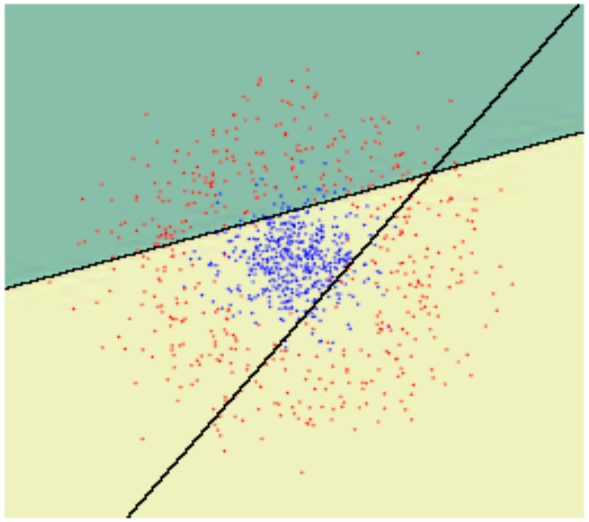
- Final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t\, h_t(x)\right)$$



Algorithm recapitulation

$t = 2$



- In each iteration:
  - Find model $h_t = \arg min_{h_j \in H} e_j$

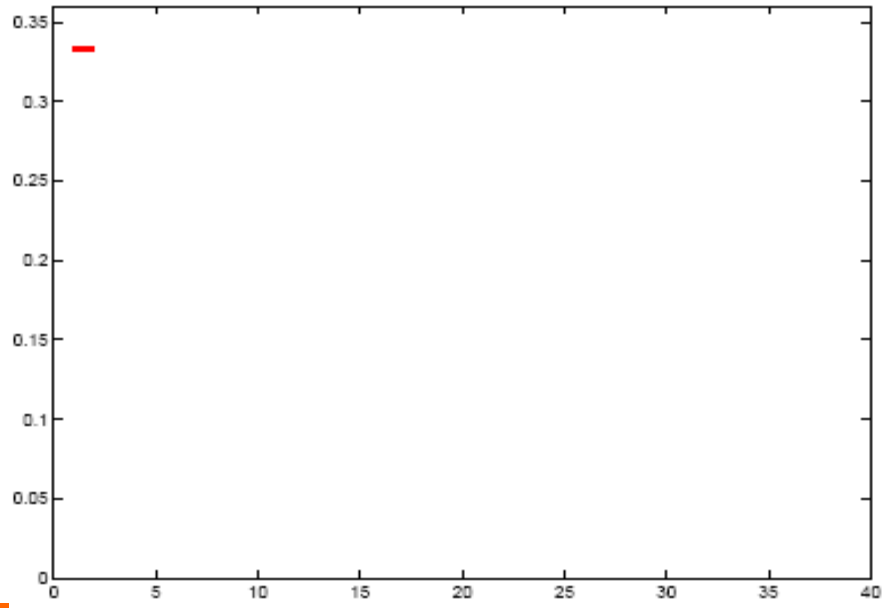
$$e_j = \sum_{i=1}^{n} D_t(i)[y_i \neq h_j(x_i)]$$

  - If $e_t > 0.5$ then stop
  - Set $\alpha_t = 0.5\log(\frac{1+e_t}{1-e_t})$
  - Update: $\widehat{D}_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

$$D_{t+1}(i) = \frac{\widehat{D}_{t+1}(i)}{\sum_{i=1\ldots m} \widehat{D}_{t+1}(i)}$$

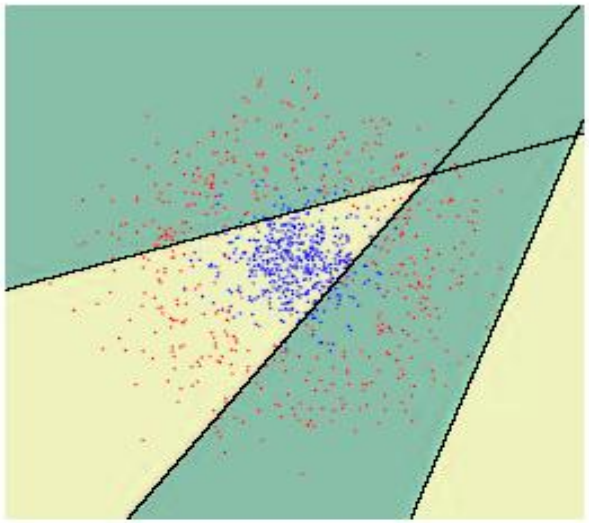
- Final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t\, h_t(x)\right)$$



Algorithm recapitulation

$t = 3$

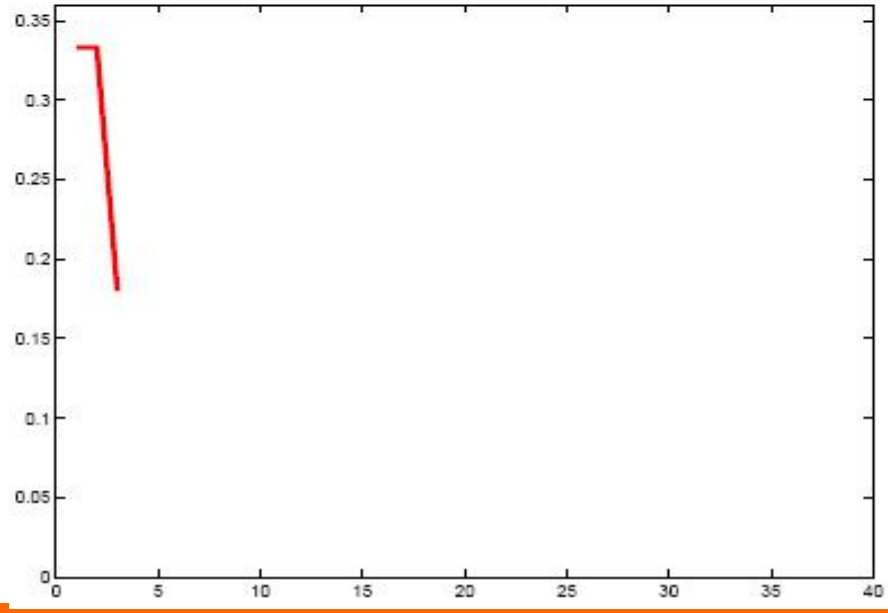


- In each iteration:
  - Find model $h_t = \arg min_{h_j \in H} e_j$

$$e_j = \sum_{i=1}^{n} D_t(i)[y_i \neq h_j(x_i)]$$

  - If $e_t > 0.5$ then stop
  - Set $\alpha_t = 0.5\log(\frac{1+e_t}{1-e_t})$
  - Update: $\widehat{D}_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

$$D_{t+1}(i) = \frac{\widehat{D}_{t+1}(i)}{\sum_{i=1\ldots m} \widehat{D}_{t+1}(i)}$$

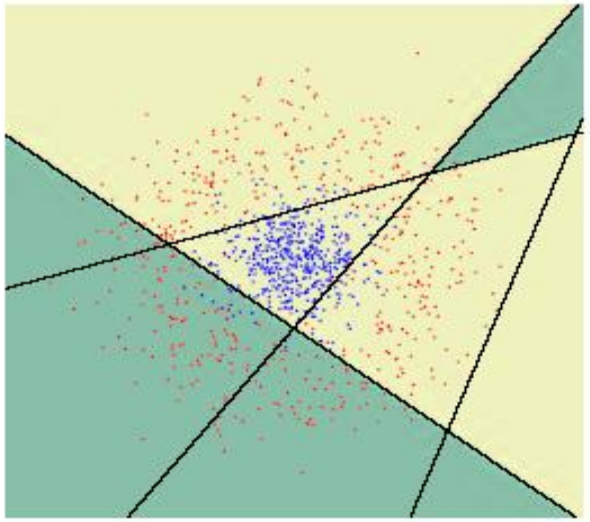


- Final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t\, h_t(x)\right)$$

Algorithm recapitulation

$t = 4$

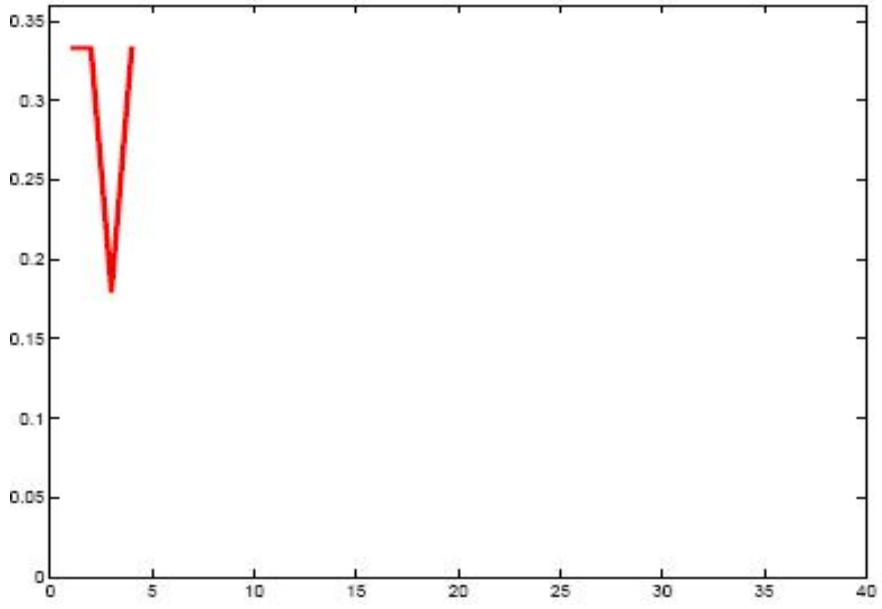


- In each iteration:
  - Find model $h_t = \arg min_{h_j \in H} e_j$

$$e_j = \sum_{i=1}^{n} D_t(i)[y_i \neq h_j(x_i)]$$

  - If $e_t > 0.5$ then stop
  - Set $\alpha_t = 0.5\log(\frac{1+e_t}{1-e_t})$
  - Update: $\widehat{D}_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

$$D_{t+1}(i) = \frac{\widehat{D}_{t+1}(i)}{\sum_{i=1\ldots m} \widehat{D}_{t+1}(i)}$$

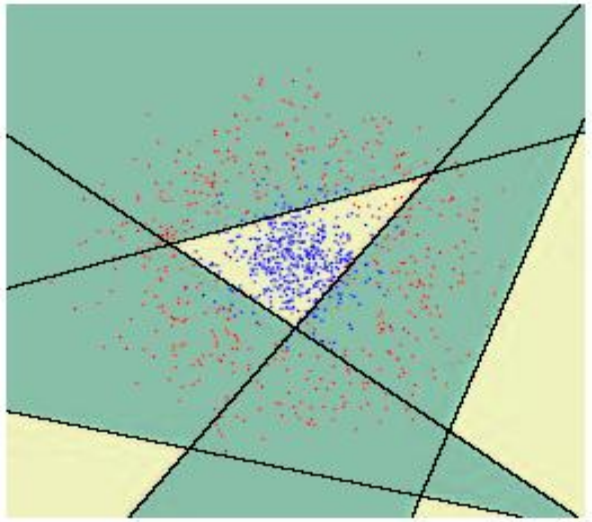


- Final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t\, h_t(x)\right)$$

Algorithm recapitulation

$t = 5$

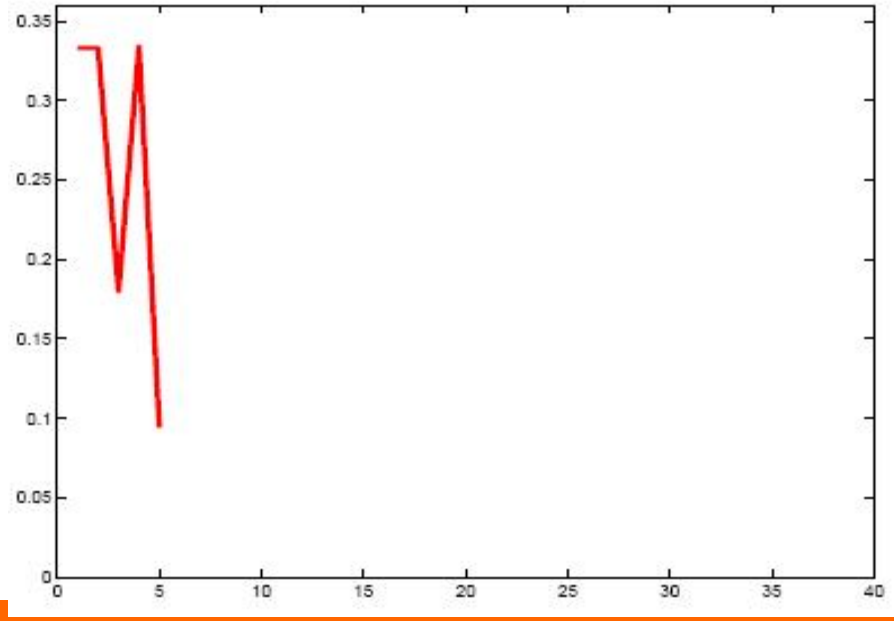


- In each iteration:
  - Find model $h_t = \arg min_{h_j \in H} e_j$

$$e_j = \sum_{i=1}^{n} D_t(i)[y_i \neq h_j(x_i)]$$

  - If $e_t > 0.5$ then stop
  - Set $\alpha_t = 0.5\log(\frac{1+e_t}{1-e_t})$
  - Update: $\widehat{D}_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

$$D_{t+1}(i) = \frac{\widehat{D}_{t+1}(i)}{\sum_{i=1\ldots m} \widehat{D}_{t+1}(i)}$$

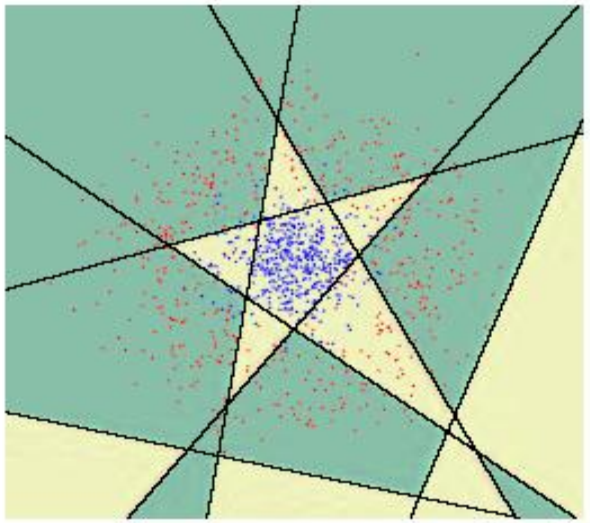


- Final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t\, h_t(x)\right)$$

Algorithm recapitulation

$t = 7$

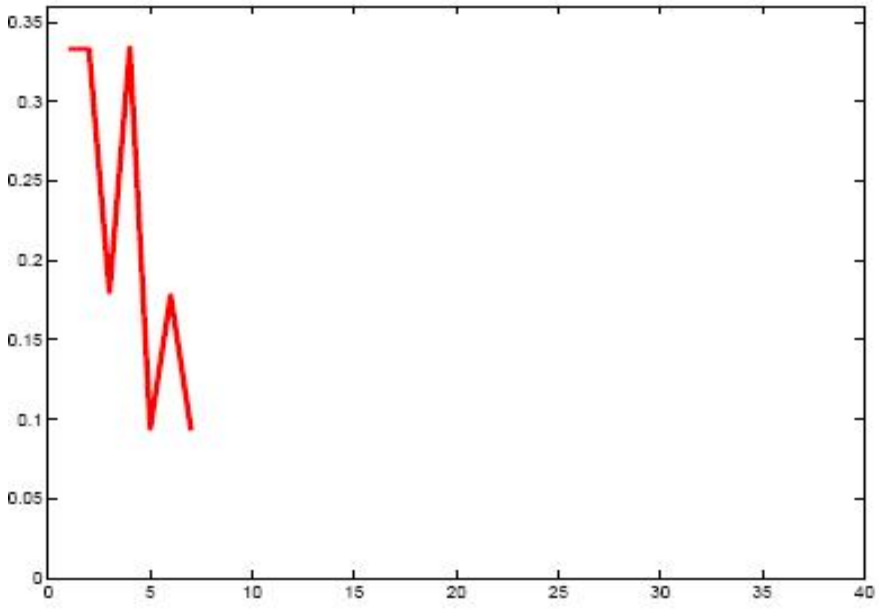


- In each iteration:
  - Find model $h_t = \arg min_{h_j \in H} e_j$

$$e_j = \sum_{i=1}^{n} D_t(i)[y_i \neq h_j(x_i)]$$

  - If $e_t > 0.5$ then stop
  - Set $\alpha_t = 0.5\log(\frac{1+e_t}{1-e_t})$
  - Update: $\widehat{D}_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

$$D_{t+1}(i) = \frac{\widehat{D}_{t+1}(i)}{\sum_{i=1\ldots m} \widehat{D}_{t+1}(i)}$$

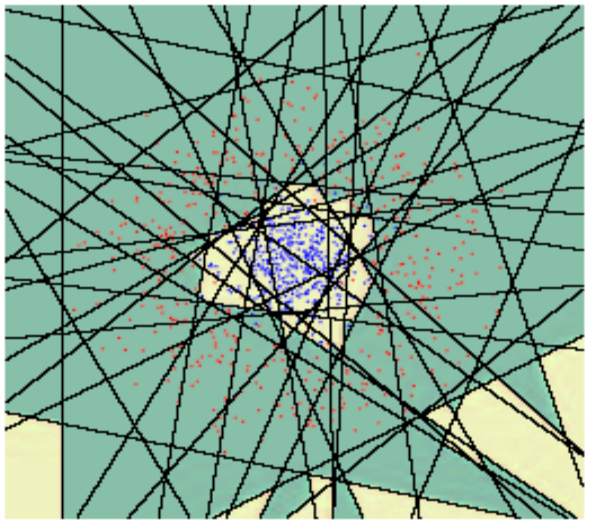


- Final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t\, h_t(x)\right)$$

Algorithm recapitulation

$t = 40$

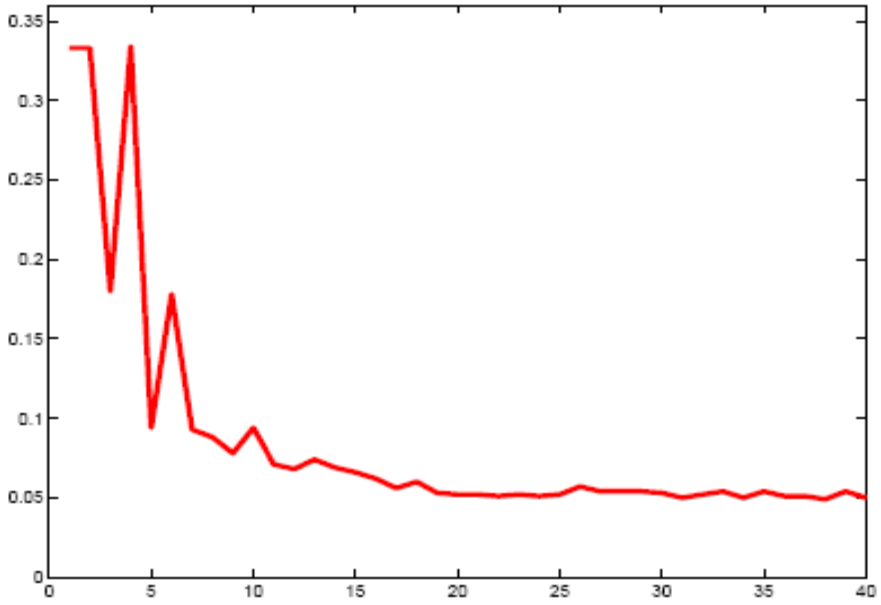


- In each iteration:
  - Find model $h_t = \arg min_{h_j \in H} e_j$

$$e_j = \sum_{i=1}^{n} D_t(i)[y_i \neq h_j(x_i)]$$

  - If $e_t > 0.5$ then stop
  - Set $\alpha_t = 0.5\log(\frac{1+e_t}{1-e_t})$
  - Update: $\widehat{D}_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

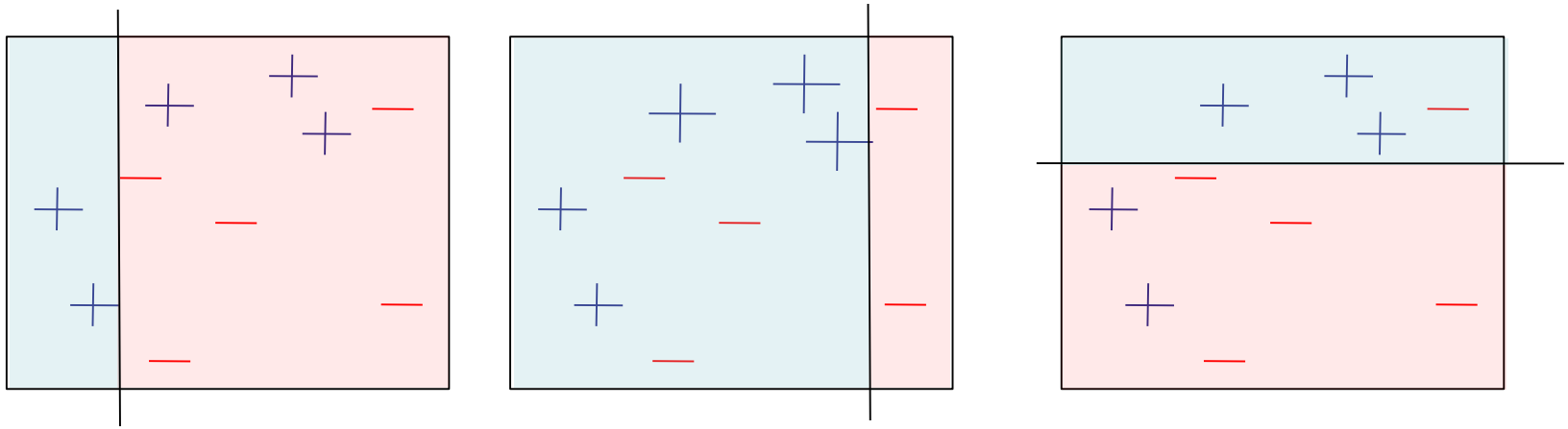
$$D_{t+1}(i) = \frac{\widehat{D}_{t+1}(i)}{\sum_{i=1\ldots m} \widehat{D}_{t+1}(i)}$$



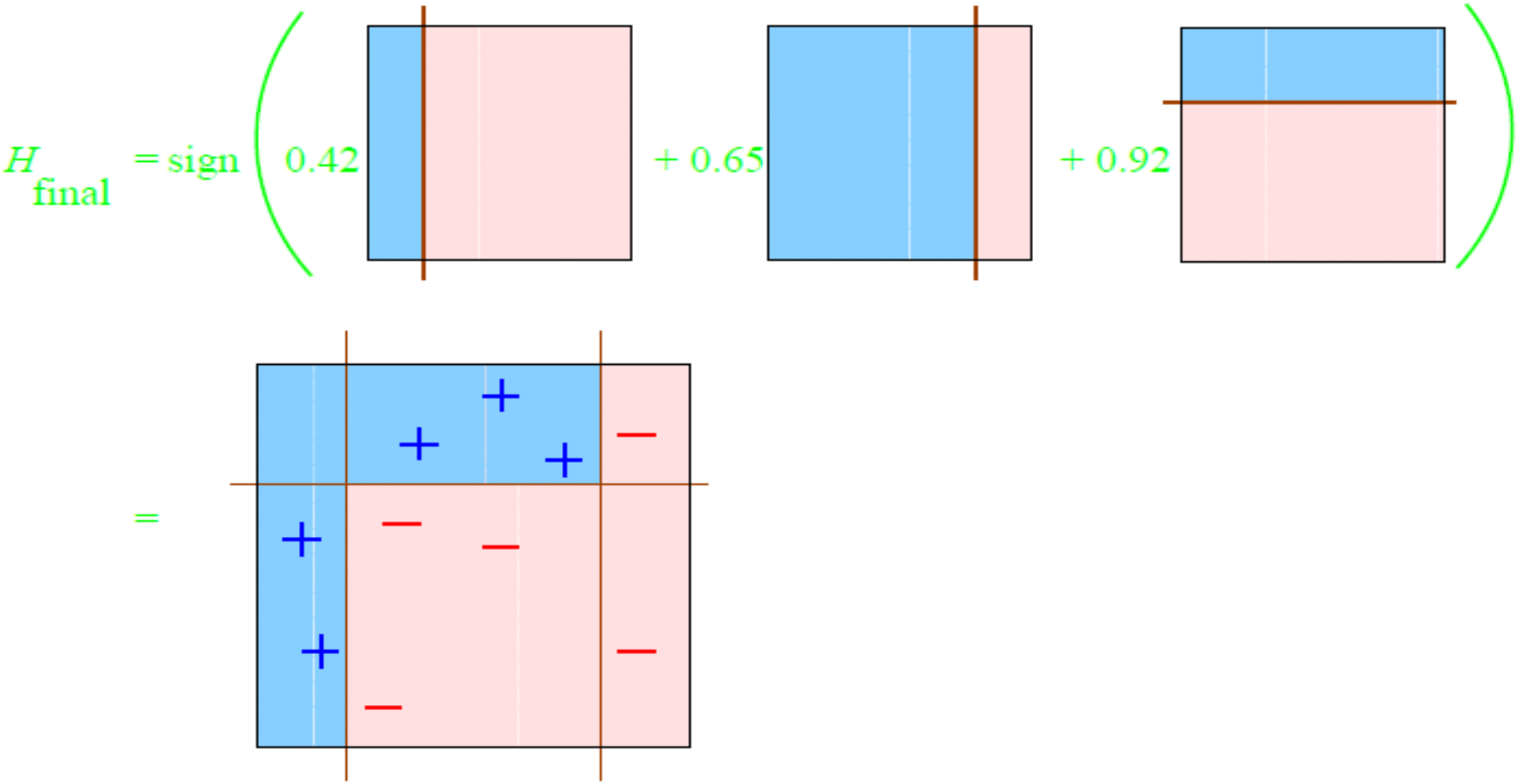
- Final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t\, h_t(x)\right)$$

Another Toy Example

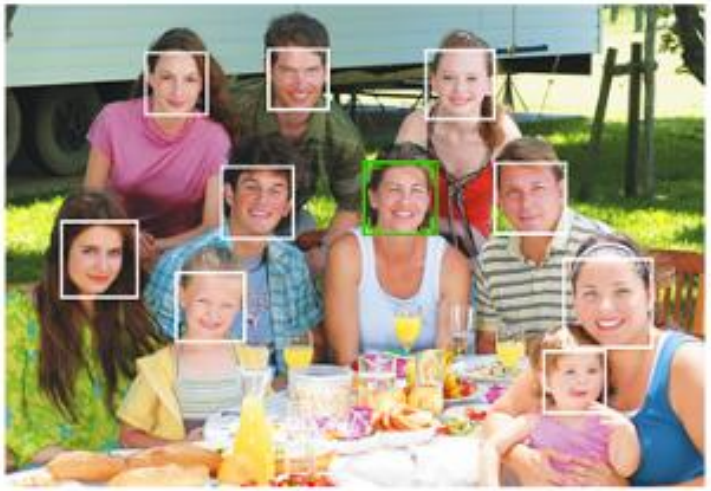


Final Classifier



AdaBoost Example: Face Detection

- Viola-Jones face detection algorithm
- Combine lots of very weak classifiers
  - Decision stumps = threshold on a single feature
- Define lots and lots of features
- Use AdaBoost to find good features
  - And weights for combining as well



Strengths of AdaBoost

- It has no parameters to tune (except for the number of rounds)
- It is fast, simple and easy to program.
- It comes with theoretical guarantee (e.g., training error, test error)
- Instead of trying to design a learning algorithm that is accurate over the entire space, we can focus on finding base learning algorithms that only need to be better than random.
- It can identify outliers: i.e. examples that are either mislabeled or that are inherently ambiguous and hard to categorize.

Weakness of AdaBoost

- The actual performance of boosting depends on the data and the base learner.
- When the number of outliers is very large, the emphasis placed on the hard examples can hurt the performance.

➔ "Gentle AdaBoost", "BrownBoost"

International School of Engineering

Plot 63/A, 1st Floor, Road # 13, Film Nagar, Jubilee Hills, Hyderabad - 500 033

For Individuals: +91-9502334561/63 or 040-65743991

For Corporates: +91-9618483483

Web: http://www.insofe.edu.in

Facebook: https://www.facebook.com/insofe

Twitter: https://twitter.com/Insofeedu

YouTube: http://www.youtube.com/InsofeVideos

SlideShare: http://www.slideshare.net/INSOFE

LinkedIn: http://www.linkedin.com/company/international-school-of-engineering

General Ensembles Technique: Randomization

- Can randomize learning algorithms instead of inputs
- Some algorithms already have random component: e.g. random initialization
- Most algorithms can be randomized
  - Pick from the N best options at random instead of always picking the best one
  - Split rule in decision tree
- Random projection in kNN (Freund and Dasgupta 08)

Simple Ensembles: Mixture of experts

- Use multiple learners
- Use a control switch which applies the suitable learner for each region of the data
- Control is done using expectation maximization

Other Caveats of Adaboost

- Performance of AdaBoost depends on data and weak learner
- Consistent with theory, AdaBoost can fail if
  - weak classifier too complex– overfitting
  - weak classifier too weak -- underfitting
- Empirically, AdaBoost seems especially susceptible to uniform noise