CSE 4553
Machine Learning
Lecture 12: Ensembles

Winter 2022

Hasan Mahmud | hasan@iut-dhaka.edu

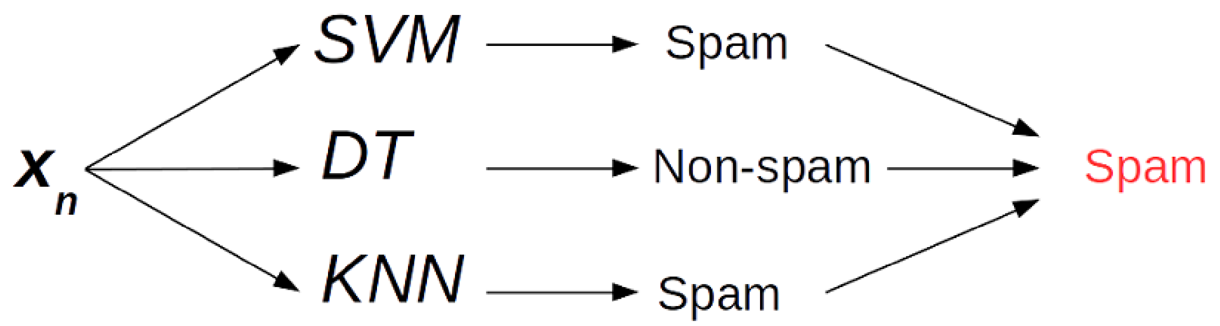
Contents

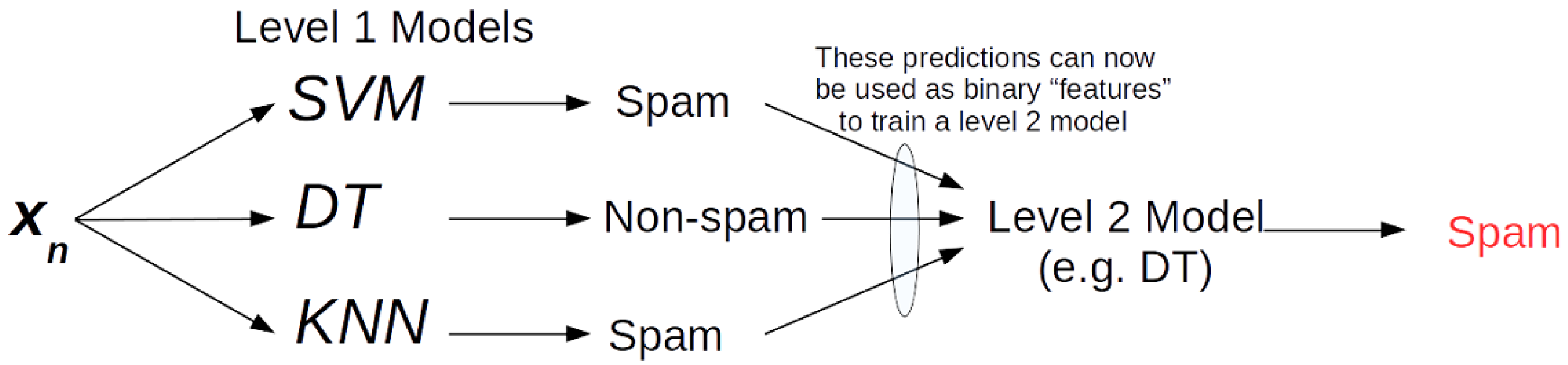
- Introduction
- Simple ensembles
  - Max voting
  - Average
  - Weighted average
- Stacking
- Bagging
  - Random forest
- Boosting
  - Adaboost

Introduction

- An ensemble is a set of classifiers that learn a target function, and their individual predictions are combined to classify new examples.

Simple ensembles





Simple ensemble techniques

- Max voting

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4

- Averaging

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4.4

- Weighted averaging

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating	
weight	0.23	0.23	0.18	0.18	0.18	
rating	5	4	5	4	4	4.41

Ensemble: Stacking

Given a set of observations $\chi = \left\{ x_i \in R^M \right\}$ and a set of labels $Y = \{y_i \in N\}$ and a Training Set $D = \{(x_i, y_i)\}$ as an Input, we want to solve the problem of Supervised Classification where we learn the model M based on the D .

Algorithm 1 - Stacking

Input : $D = \{(x_i, y_i) | x_i \in \chi,\ y_i \in Y\}$

Output : An ensemble classifier H

1. **Step 1 :** Learn first-level classifiers
2. For $t \leftarrow 1$ to T do
3. Learn a base classifier h_t based on D
4. **Step 2 :** Construct new data set from D
5. For $i \leftarrow 1$ *to* m do
6. Construct a new data set that contains $\{x_i^{new}, y_i\}$, where $x_i^{new} = \{h_j(x_i)\ for\ j = 1\ to\ T\}$
7. **Step 3 :** Learn a second-level classifier
8. Learn a new classifier h^{new} based on the newly constructed data set
9. **Return** $H(x) = h^{new}(h_1(x), h_2(x), \ldots, h_T(x))$

Advanced Ensemble techniques

- Stacking

Table 1 : Training Data Set

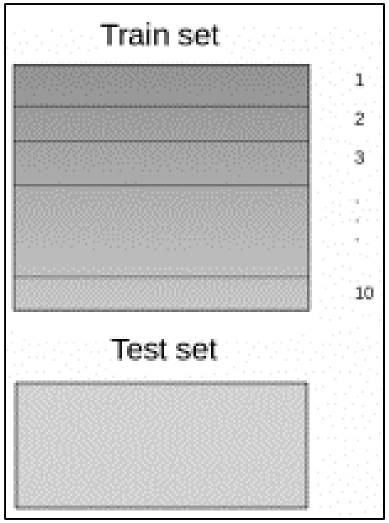
x	1	2	3	4	5	6	7	8	9	10
y	+1	+1	+1	-1	-1	-1	-1	+1	+1	+1

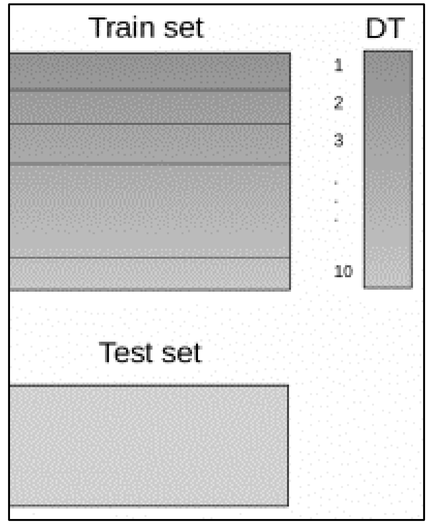
Table 2 : Stacking New Features

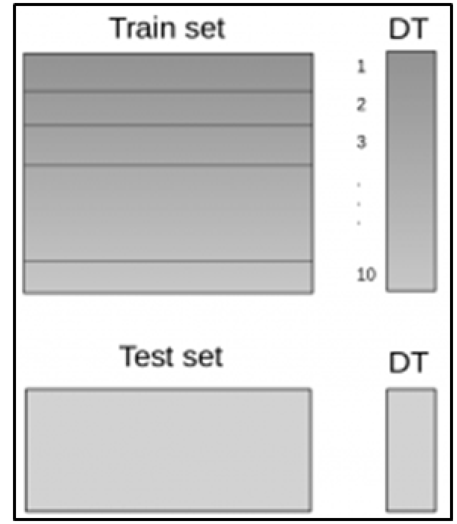
x	(1,-1)	(1,-1)	(1,-1)	(-1,-1)	(-1,-1)	(-1,-1)	(-1,-1)	(-1,1)	(-1,1)	(-1,1)
y	+1	+1	+1	-1	-1	-1	-1	+1	+1	+1

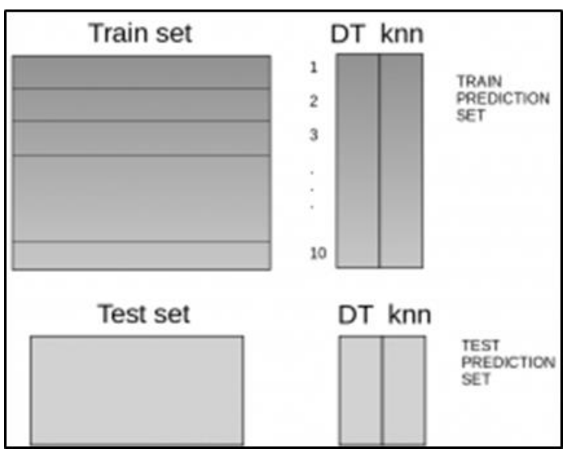
Steps of stacking

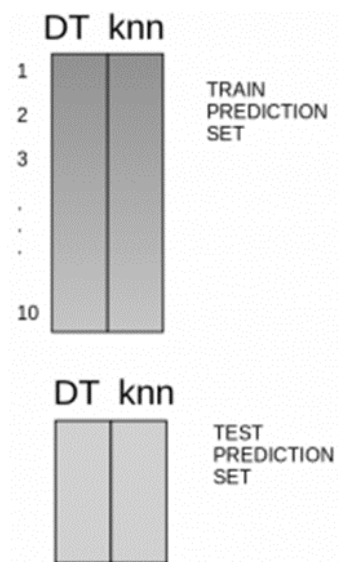
1. The train set is split into 10 parts.
2. A base model (suppose a decision tree) is fitted on 9 parts and predictions are made for the 10th part. This is done for each part of the train set.
3. The base model (in this case, decision tree) is then fitted on the whole train dataset.
4. Using this model, predictions are made on the test set.
5. Steps 2 to 4 are repeated for another base model (say KNN) resulting in another set of predictions for the train set and test set.
6. The predictions from the train set are used as features to build a new model.
7. This model is used to make final predictions on the test prediction set.


Train set
1
2
3
10
Test set


Train set
DT
1
2
3
10
Test set

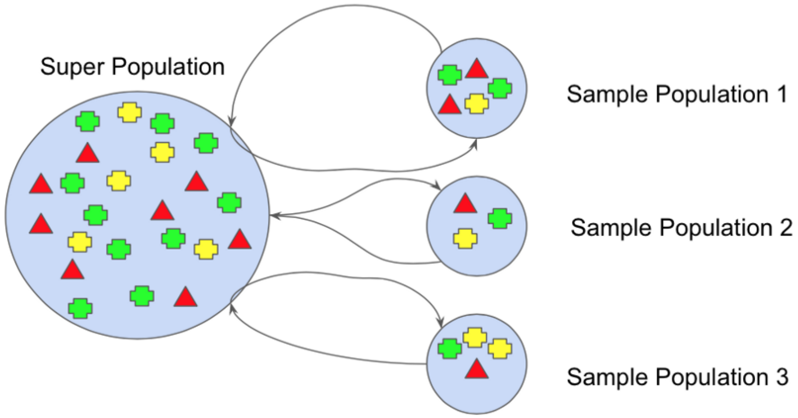

Train set
DT
1
2
3
10
Test set
DT


Train set
DT knn
1
2
3
10
TRAIN PREDICTION SET
Test set
DT knn
TEST PREDICTION SET


DT knn
1
2
3
10
TRAIN PREDICTION SET
DT knn
TEST PREDICTION SET

Bootstrapping

- Bootstrap method refers to random sampling with replacement.
- Bootstrapping is a sampling technique in which we create subsets of observations from the original dataset, **with replacement**.
- This sample is referred to as a resample.

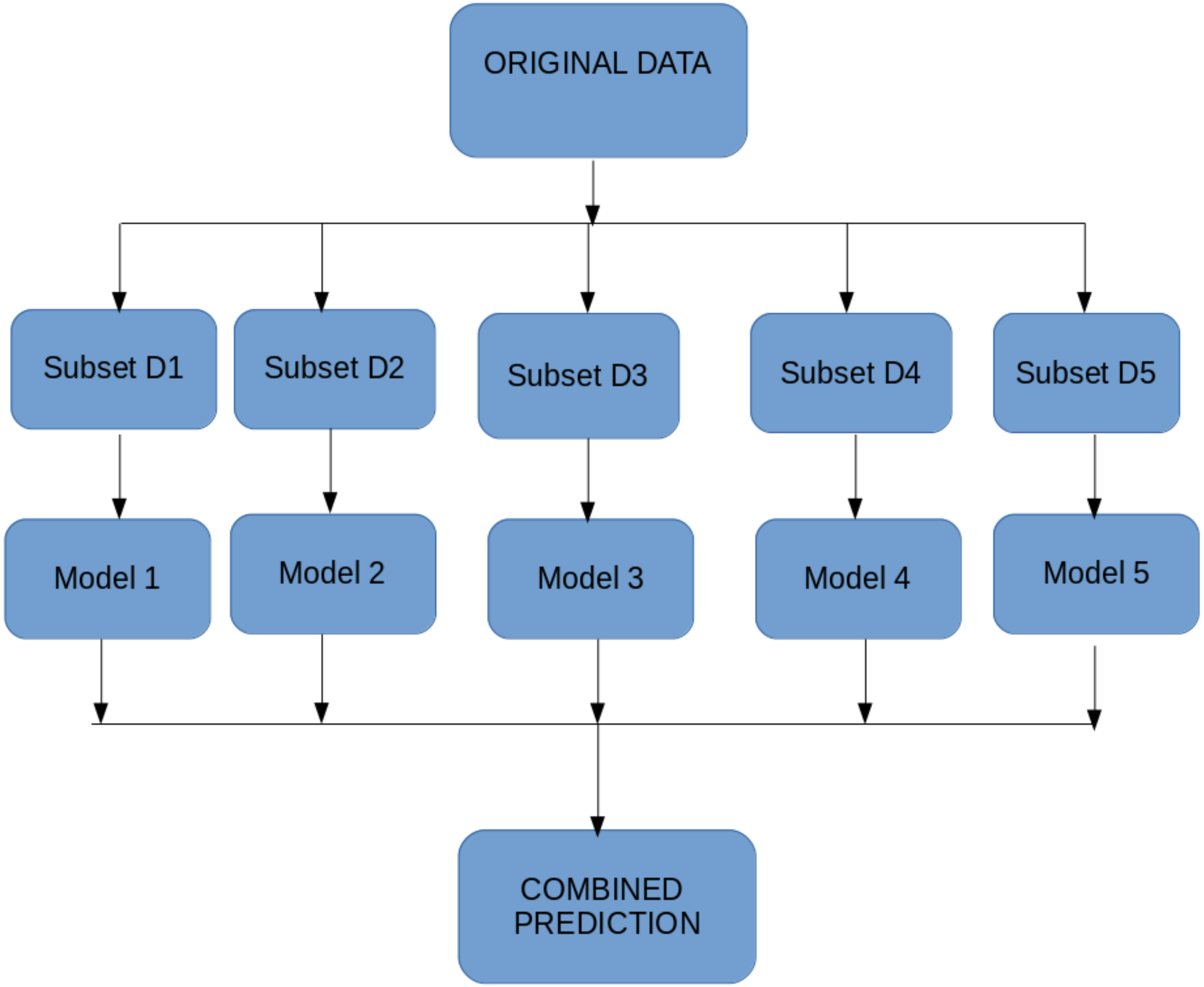


- The size of the subsets may be the same as the size of the original set.
- This allows the model or algorithm to get a better understanding of the various biases, variances and features that exist in the resample.
- Bootstrapping is also great for small size data sets that can have a tendency to overfit.

Bagging

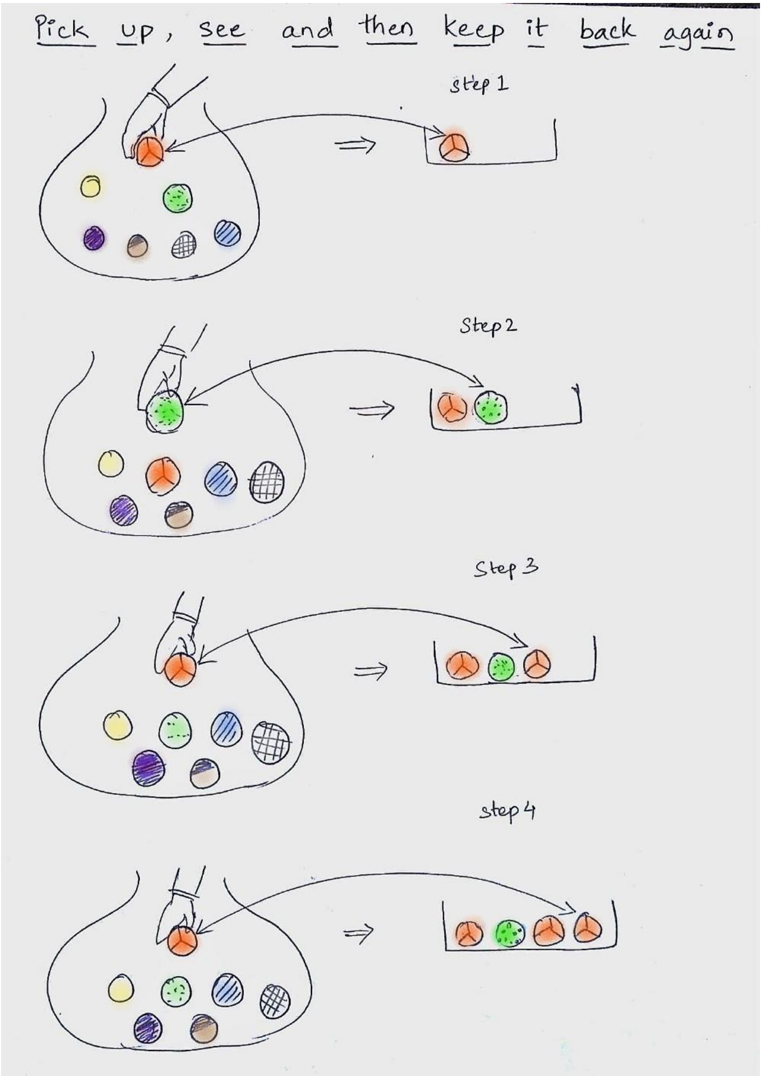
- Bagging (or Bootstrap Aggregating) technique uses these subsets (bags) to get a fair idea of the distribution (complete set).
- The size of subsets created for bagging may be less than the original set.
- Multiple subsets are created from the original dataset, selecting observations with replacement (Bootstrapping).
- A base model (weak model) is created on each of these subsets.
- The models run in parallel and are independent of each other.
- The final predictions are determined by combining the predictions from all the models.

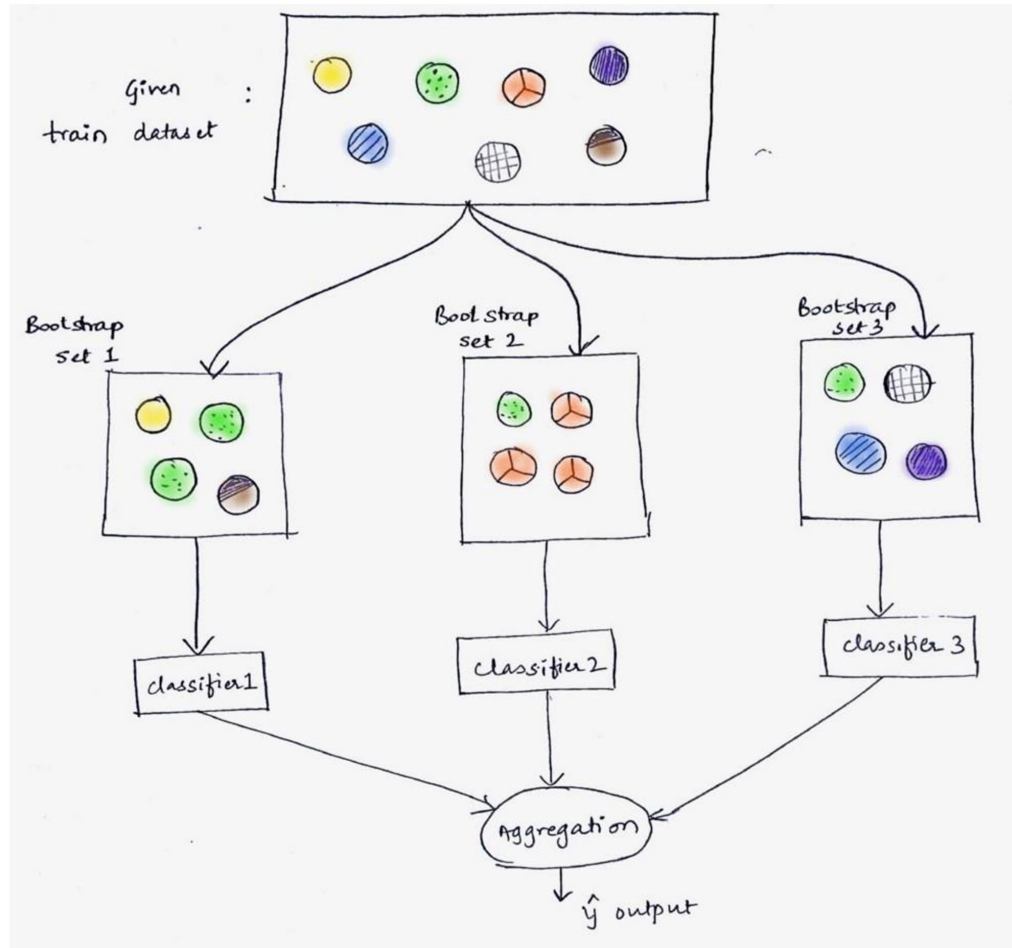
Bagging



ORIGINAL DATA
Subset D1
Subset D2
Subset D3
Subset D4
Subset D5
Model 1
Model 2
Model 3
Model 4
Model 5
COMBINED PREDICTION

Bagging

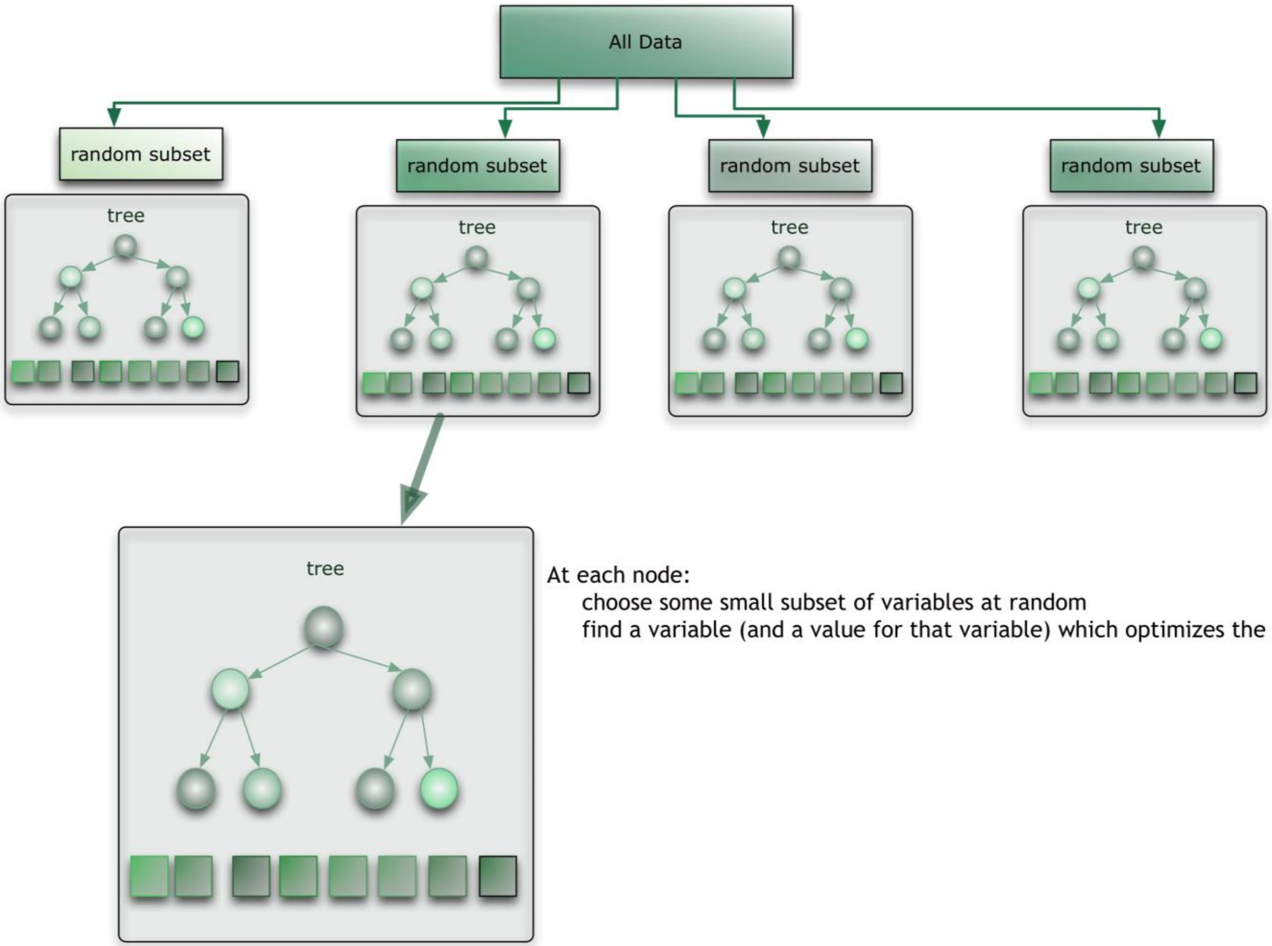

Pick up, see and then keep it back again
step 1
Step 2
Step 3
step 4


Given train dataset :
Bootstrap set 1
Bootstrap set 2
Bootstrap set 3
classifier 1
classifier 2
classifier 3
Aggregation
ŷ output

Random forest

1. Random subsets are created from the original dataset (bootstrapping).
2. At each node in the decision tree, only a random set of features are considered to decide the best split.
3. A decision tree model is fitted on each of the subsets.
4. The final prediction is calculated by averaging the predictions from all decision trees.

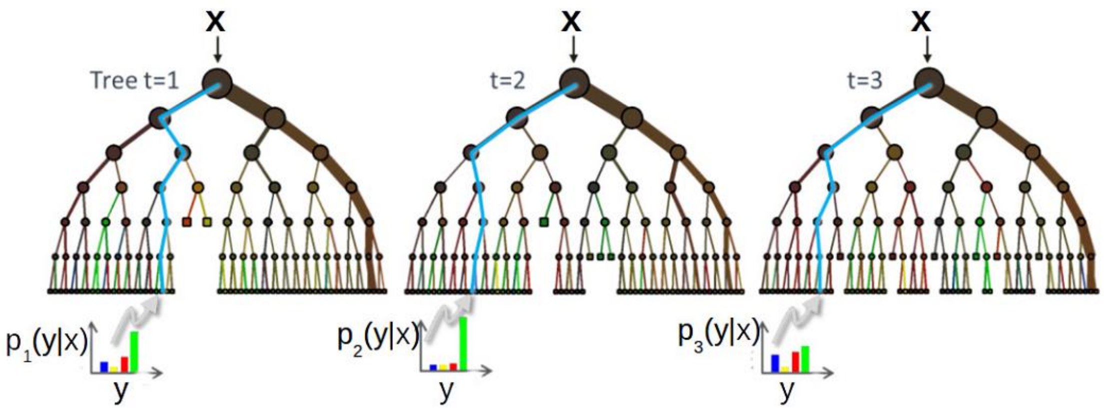
Random Forest



At each node:

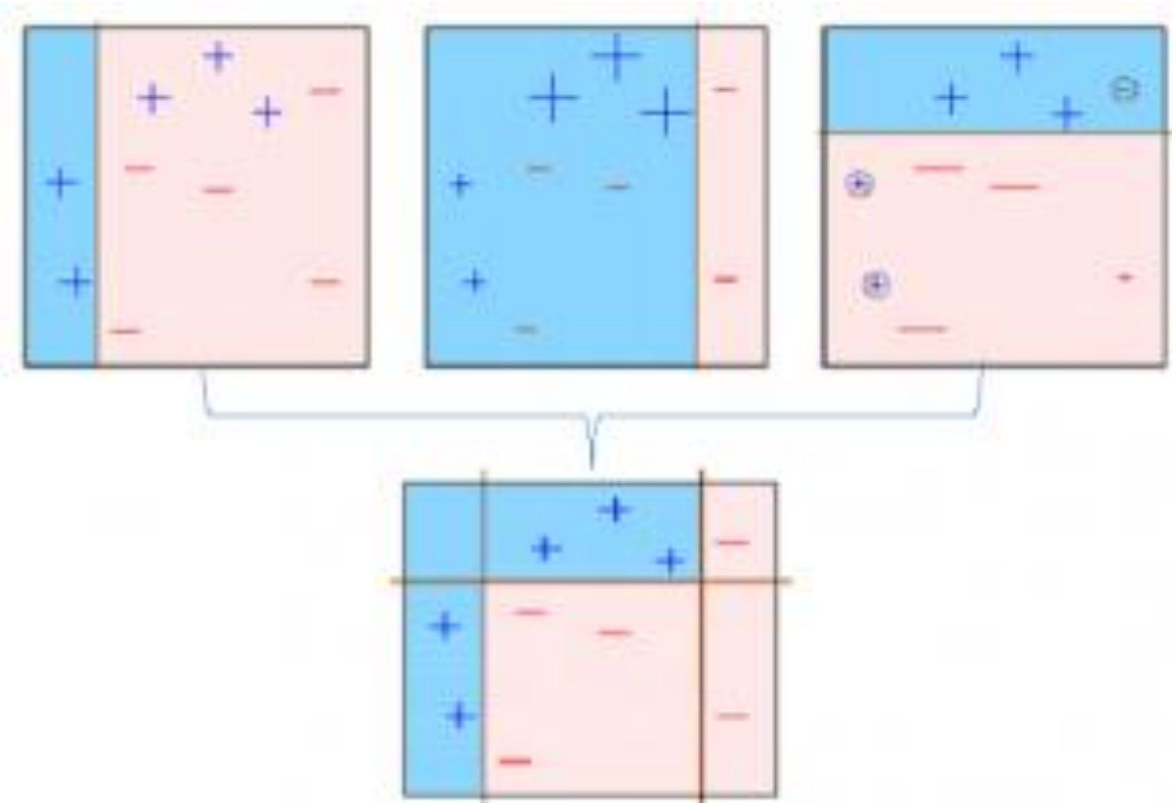
- choose some small subset of variables at random
- find a variable (and a value for that variable) which optimizes the split

Random forest



- An ensemble of decision tree (DT) classifiers
- Uses bagging on features (each DT will use a random set of features)
  - Given a total of D features, each DT uses $\sqrt{D}$ randomly chosen features
  - Randomly chosen features make the different trees uncorrelated
- All DTs usually have the same depth
- Each DT will split the training data differently at the leaves
- Prediction for a test example votes on/averages predictions from all the DTs

Boosting



Adaboost algorithm

- Given: Training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ with $y_n \in \{-1, +1\}$, $\forall n$
- Initialize weight of each example $(\boldsymbol{x}_n, y_n)$: $D_1(n) = 1/N$, $\forall n$
- For round $t = 1 : T$
  - Learn a weak $h_t(\boldsymbol{x}) \rightarrow \{-1, +1\}$ using training data weighted as per D_t
  - Compute the weighted fraction of errors of h_t on this training data

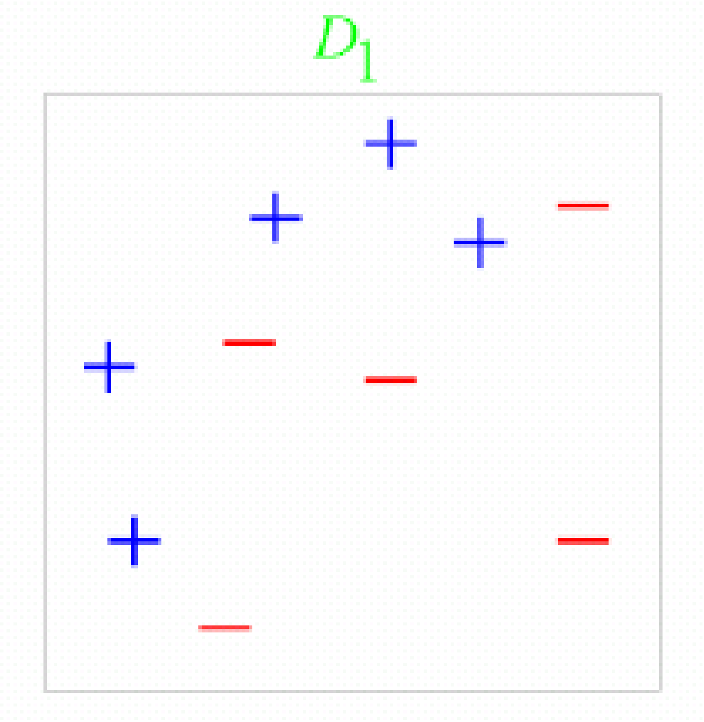
$$\epsilon_t = \sum_{n=1}^{N} D_t(n) \mathbb{1}[h_t(\boldsymbol{x}_n) \neq y_n]$$

  - Set "importance" of h_t: $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$(gets larger as ϵ_t gets smaller)
  - Update the weight of each example

$$\begin{aligned} D_{t+1}(n) &\propto \begin{cases} D_t(n) \times \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_n) = y_n \quad \text{(correct prediction: decrease weight)} \\ D_t(n) \times \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_n) \neq y_n \quad \text{(incorrect prediction: increase weight)} \end{cases} \\ &= D_t(n) \exp(-\alpha_t y_n h_t(\boldsymbol{x}_n)) \end{aligned}$$

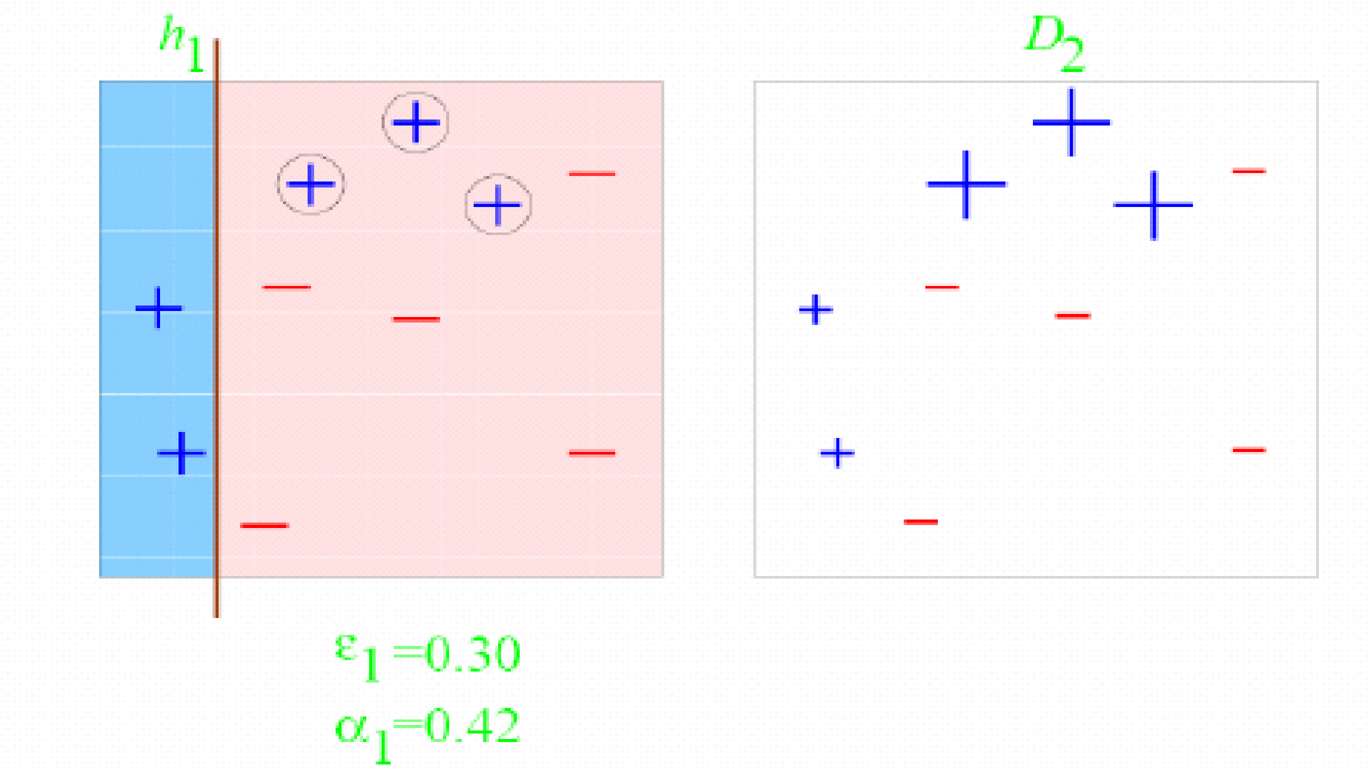
  - Normalize D_{t+1} so that it sums to 1: $D_{t+1}(n) = \frac{D_{t+1}(n)}{\sum_{m=1}^{N} D_{t+1}(m)}$
- Output the "boosted" final hypothesis $H(\boldsymbol{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}))$

Example



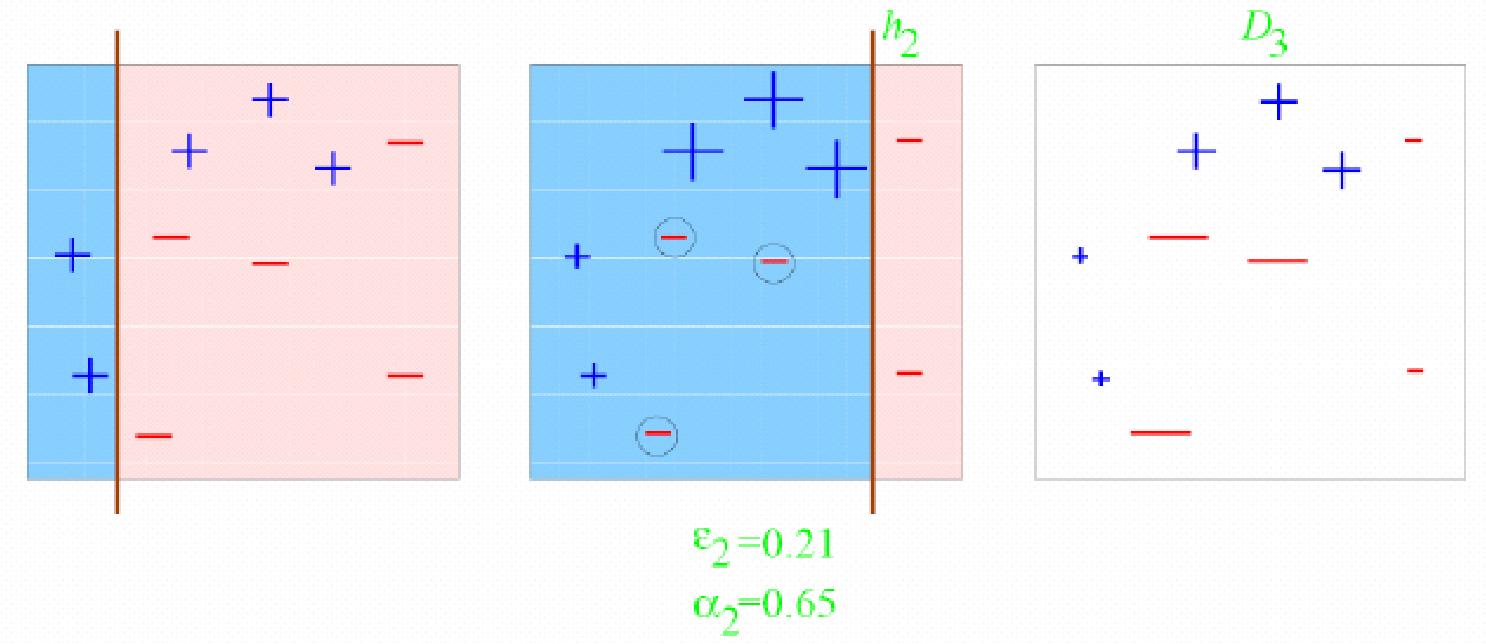
Training set: 10 points
(represented by plus or minus)
Original Status: Equal Weights for all training samples

Example(cont'd)



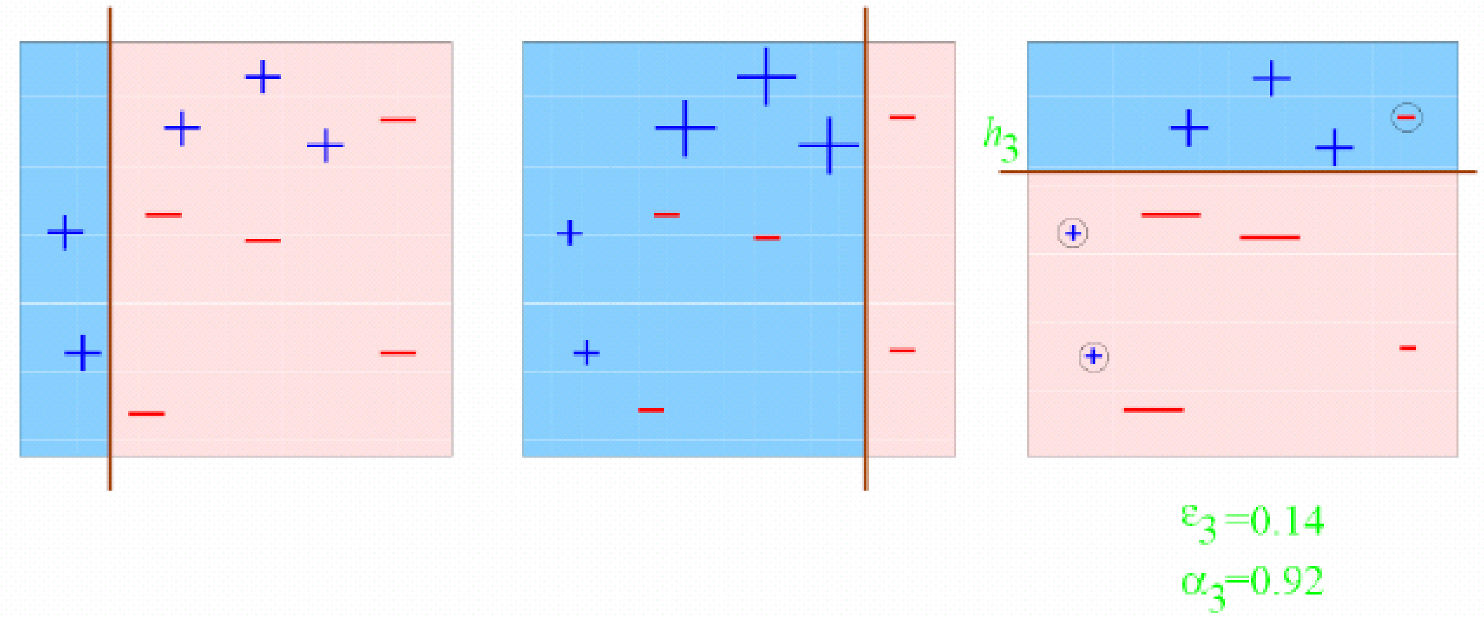
Round 1: Three "plus" points are not correctly classified;
They are given higher weights.

Example(cont'd)



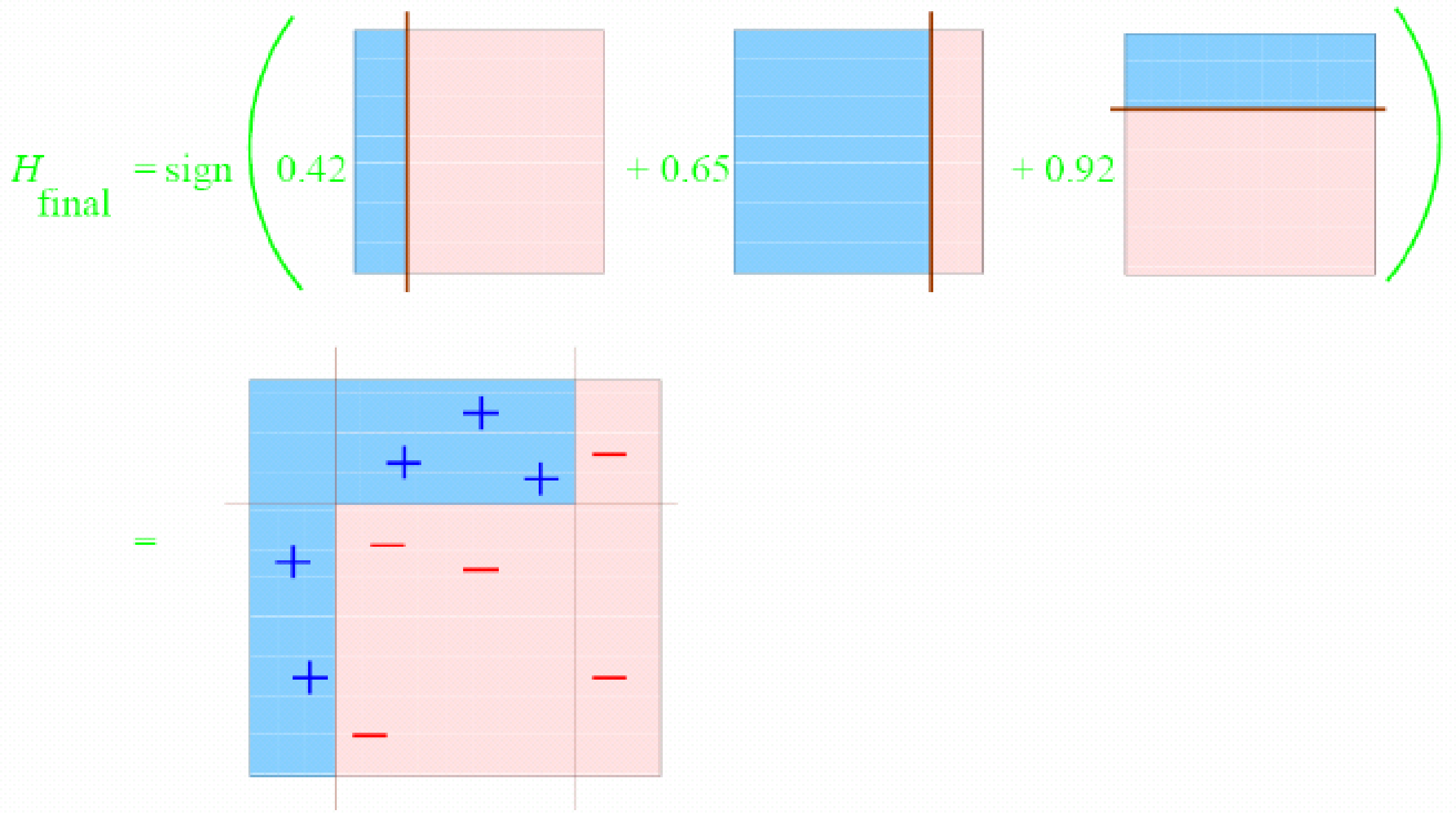
Round 2: Three "minuse" points are not correctly classified;
They are given higher weights.

Example(cont'd)



Round 3: One “minuse” and two “plus” points are not correctly classified;
They are given higher weights.

Example(cont'd)



Final Classifier: integrate the three "weak" classifiers and obtain a final strong classifier.