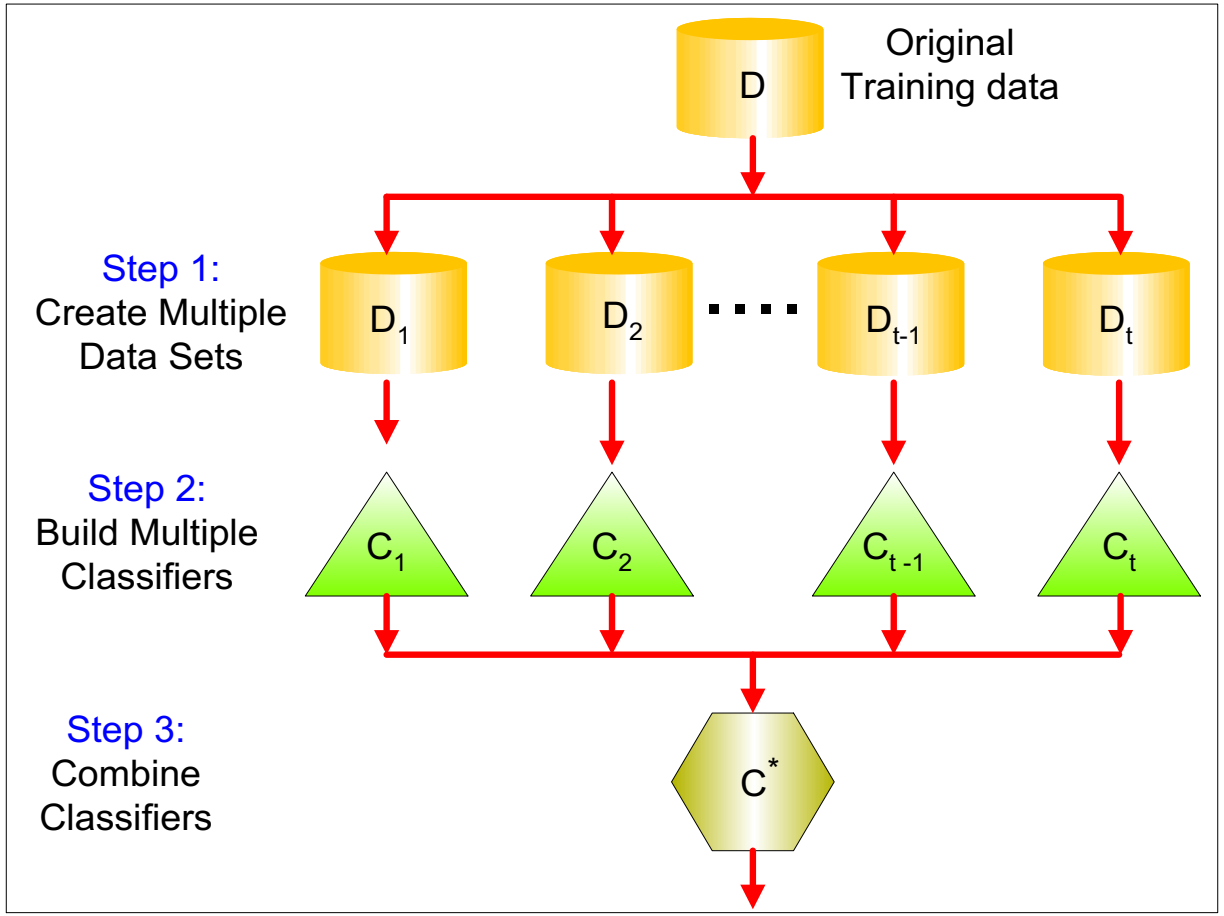
DATA MINING 2
Ensemble Methods

Riccardo Guidotti

a.a. 2019/2020

Ensemble Methods

- Improves the accuracy by **aggregating the predictions** of multiple classifiers.
- Construct a set of **base classifiers** from the training data.
- Predict class label of test records by combining the predictions made by multiple classifiers.

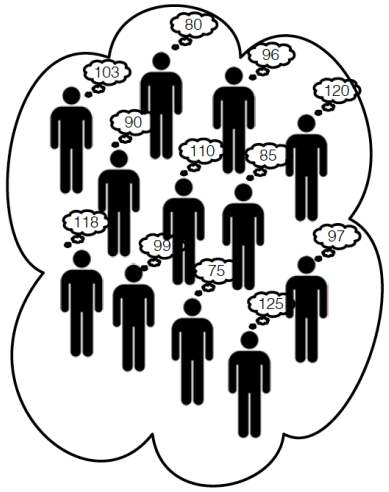


The Wisdom of the Crowds

The Wisdom of the Crowds

- The **collective knowledge** of a diverse and independent body of people typically **exceeds** the knowledge of any **single individual** and can be harnessed by voting.

- How much does the cow weigh?
- Crowd's prediction is the average of all predictions.
- This crowd predicts 99.8333 Kg
- The cow weighs 99.657 Kg.

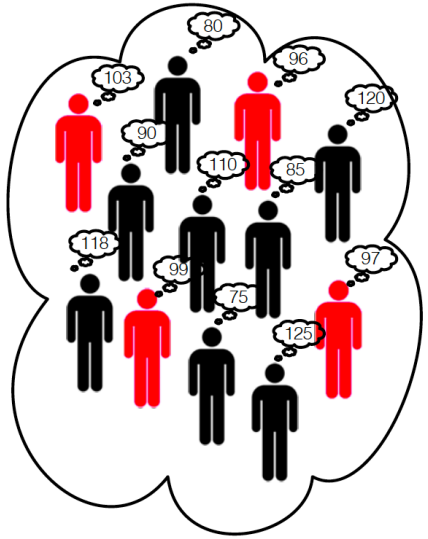


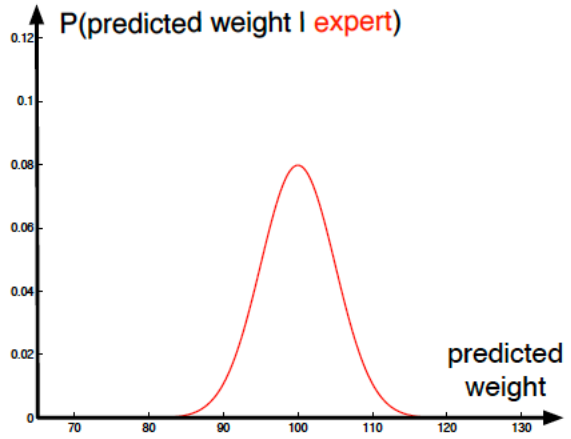
Has crowd made a good estimate?

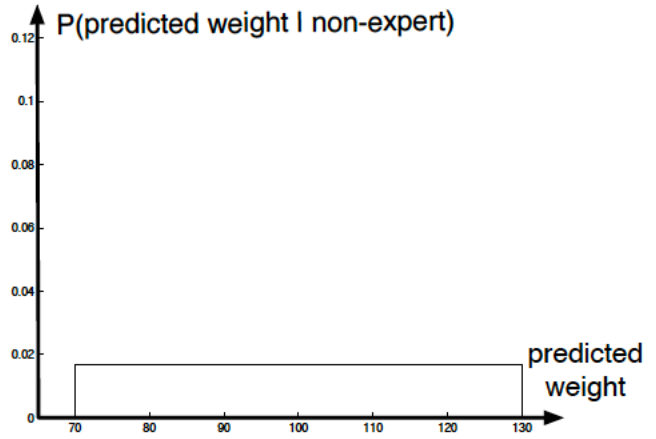
If composition of crowd:

- 30% experts.
- 70% non-experts.

and their level of expertise:







$P(\text{pred. weight} \mid \text{expert}) :$

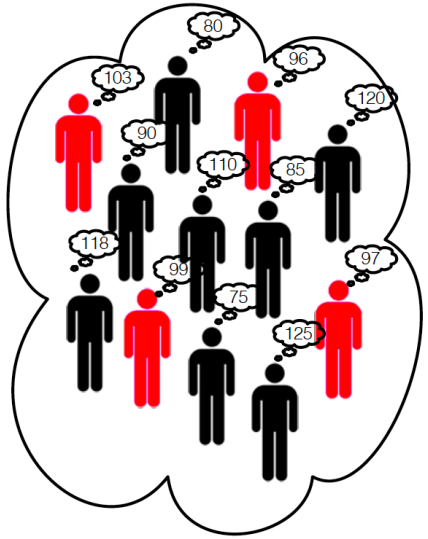
$\mathcal{N}(100, 5^2)$

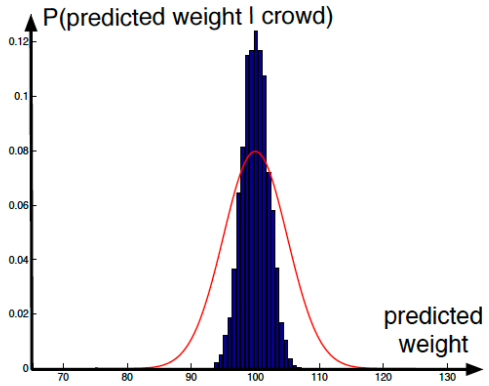
$P(\text{pred. weight} \mid \text{non-expert}) :$

$\mathcal{U}(70, 130)$

Has crowd made a good estimate?

- If the crowd contains 50 independent people:
- On average this crowd will make better estimates than the experts.
- Thus the crowd wiser than each of the experts!





$\Uparrow$

$P(\text{pred. weight}|\text{crowd})$ **and**
$P(\text{pred. weight}|\text{expert})$

Why didn’t I just asked a bunch of experts?

- Large enough crowd → high probability a sufficient number of experts will be in crowd (for any question).
- Random selection → don’t make a biased choice in experts.
- For some questions it may be hard to identify a diverse set of experts.

What makes a crowd wise?

- According to James Surowiecki there are four elements required to form a wise crowd
- **Diversity of opinion**. People in crowd should have a range of experiences, education and opinions (this encourages independent predictions).
- **Independence**. Prediction by person in crowd is not influenced by other people in the crowd.
- **Decentralization**. People have specializations and local knowledge.
- **Aggregation**. There is a mechanism for aggregating all predictions into one single prediction.

Back to Machine Learning

Back to Machine Learning

Will exploit *Wisdom of crowd* ideas for specific tasks by

- combining classifier predictions and
- aim to combine independent and diverse classifiers.

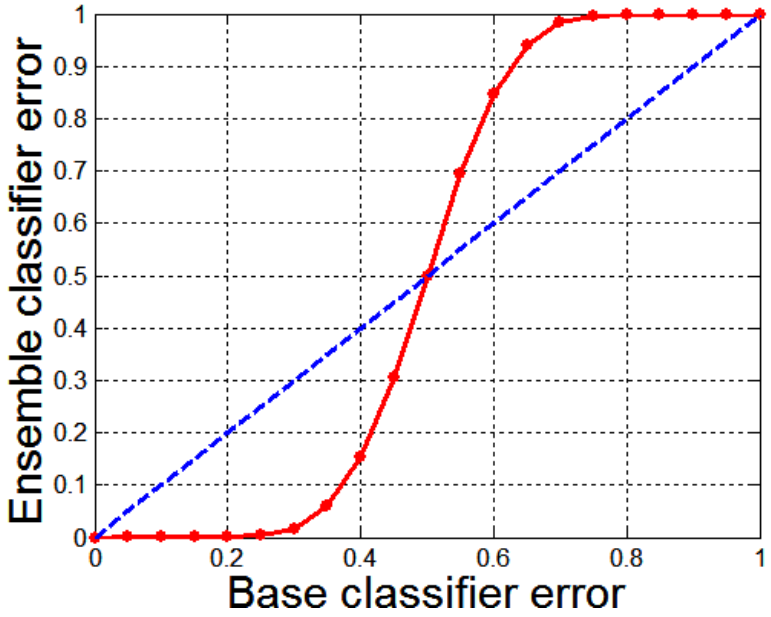
But will use labelled training data

- to identify the **expert** classifiers in the pool;
- to identify **complementary** classifiers;
- to indicate how to best **combine** them.

Why Ensemble Methods work?

Suppose there are 25 base classifiers

- Each classifier has error rate, $\varepsilon = 0.35$
- Assume errors made by classifiers are uncorrelated
- Probability that the ensemble classifier makes a wrong prediction:



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

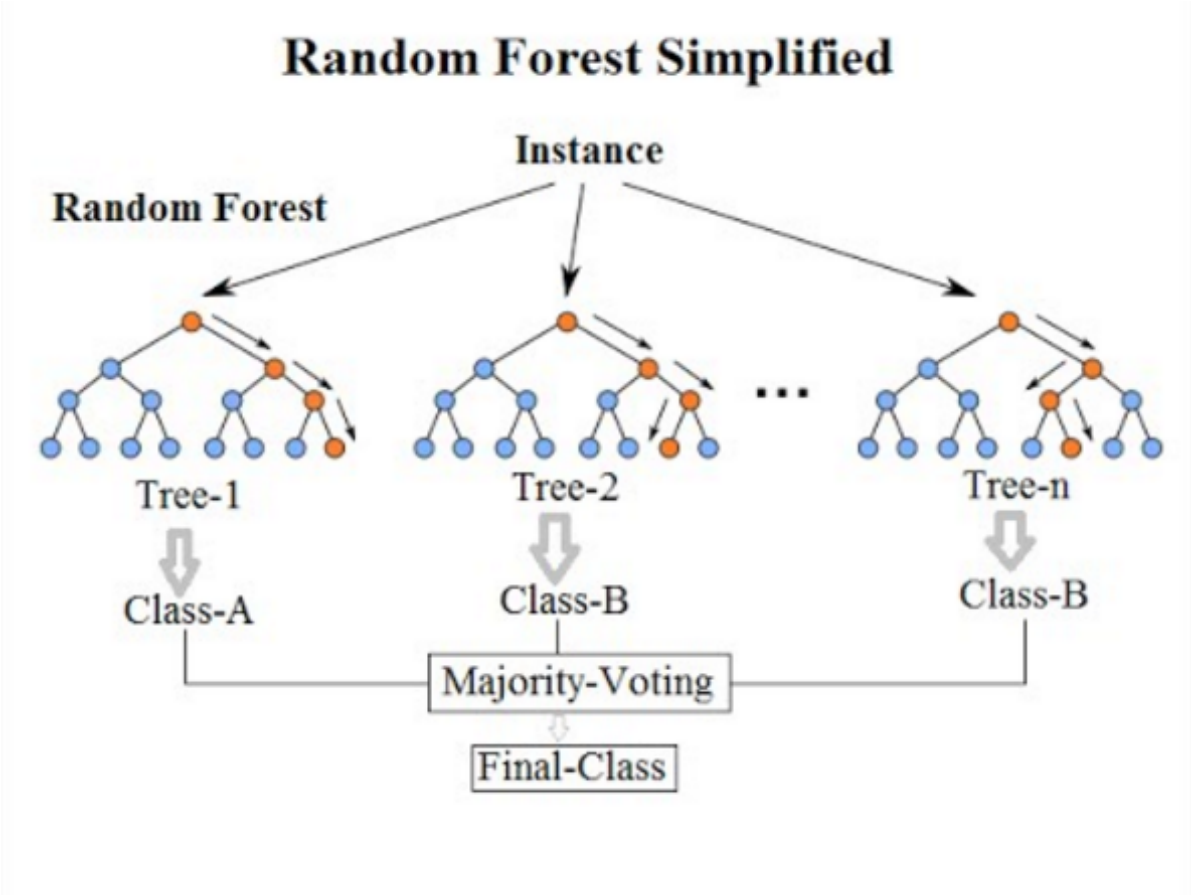
Types of Ensemble Methods

- Manipulate data distribution
  - Example: bagging, boosting
- Manipulate input features
  - Example: random forests
- Manipulate class labels
  - Example: error-correcting output coding

Random Forests

Random Forests

- Is a class of ensemble methods specifically designed for **decision trees**.
- It combines the predictions made by multiple decision trees and outputs the class that is the mode of the class's output by individual trees.



Random Forest

- Each decision tree is built on a **bootstrap sample** based on the values of an **independent** set of random vectors.
  - Unlike AdaBost (see next slides), the random vector are generated from a fixed probability distribution.
  - Bagging using decision trees is a special case of random forests where randomness is injected into the model-building process.
- Each decision tree is evaluated among ***m* randomly chosen attributes** from the M available attributes
  - m ~ $\sqrt{M}$ or m ~ $\log(M)$

Random Forest - Advantages

- It is one of the most accurate learning algorithms available. For many data sets, it produces a high accurate classifier.
- It runs efficiently on large databases.
- 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.

Bagging

Bagging (a.k.a. Bootstrap AGGregatING)

- Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- Each sample has probability (1 – 1/n)n of being selected

Algorithm 5.6 Bagging Algorithm

1: Let k be the number of bootstrap samples.
2: **for** $i = 1$ to k **do**
3: Create a bootstrap sample of size n, D_i.
4: Train a base classifier C_i on the bootstrap sample D_i.
5: **end for**
6: $C^*(x) = \arg\max_y \sum_i \delta\big(C_i(x) = y\big)$, $\{\delta(\cdot) = 1$ if its argument is true, and 0 otherwise.$\}$

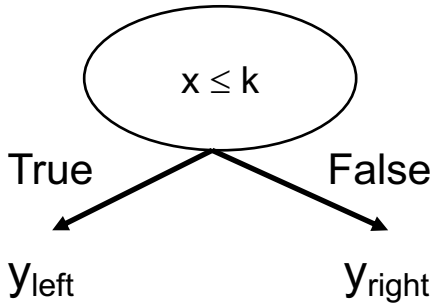
Bagging Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump
  - Decision rule: x <= k versus x > k
  - Split point k is chosen based on entropy



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

x <= 0.7 ➔ y = 1
x > 0.7 ➔ y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ➔ y = 1
x > 0.3 ➔ y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ➔ y = 1
x > 0.05 ➔ y = 1

Bagging Example

- Summary of Training sets:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

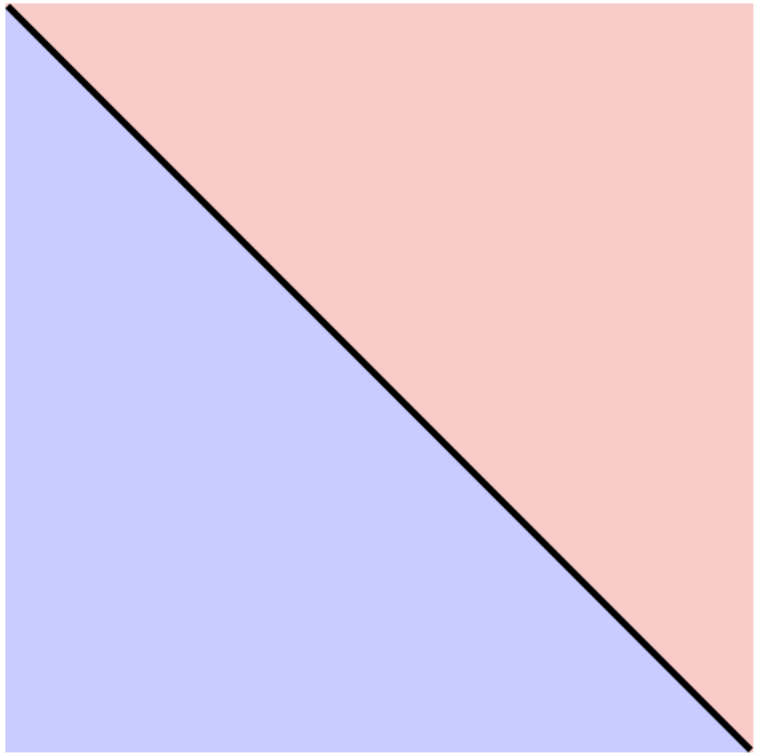
Bagging Example

- Assume test set is the same as the original data
- Use majority vote to determine class of ensemble classifier

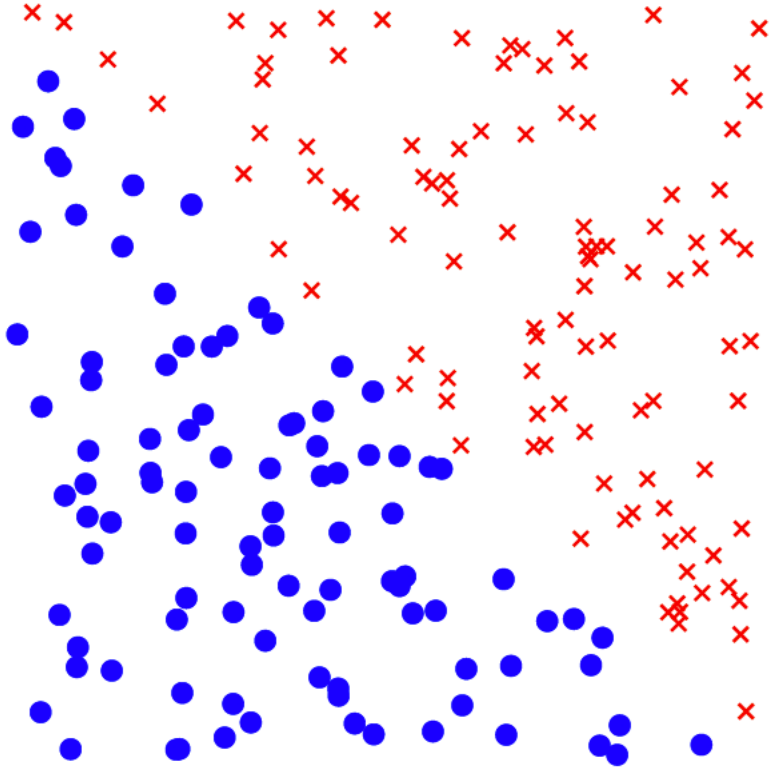
Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

Bagging Visual Example

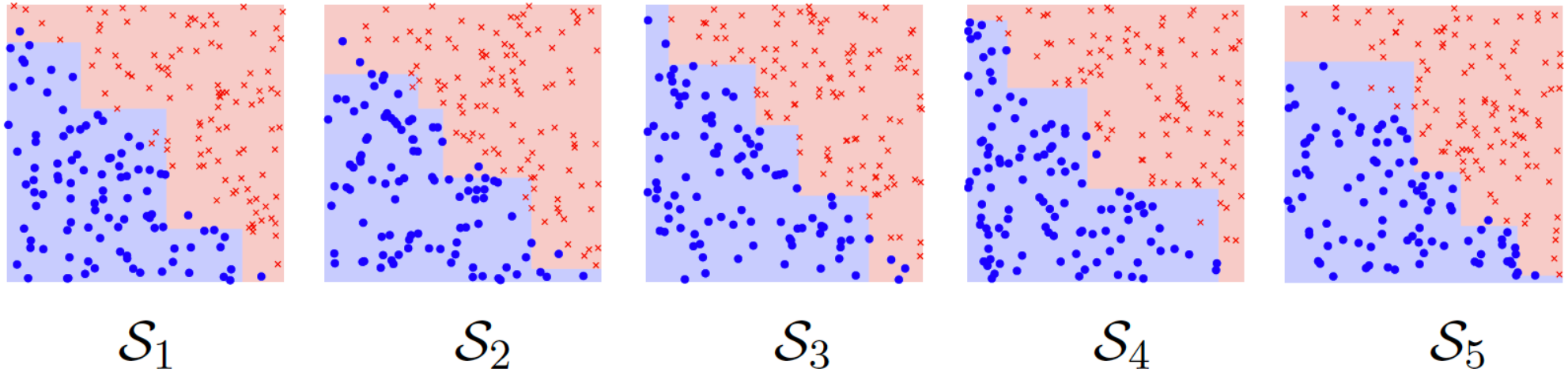


True decision boundary



Training data

Bagging Visual Example



Boosting

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records.

- Initially, all the records are assigned equal weights.
- Unlike bagging, weights may change at the end of each boosting round.

Boosting

- Records that are wrongly classified will have their weights increased.
- Records that are classified correctly will have their weights decreased.

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

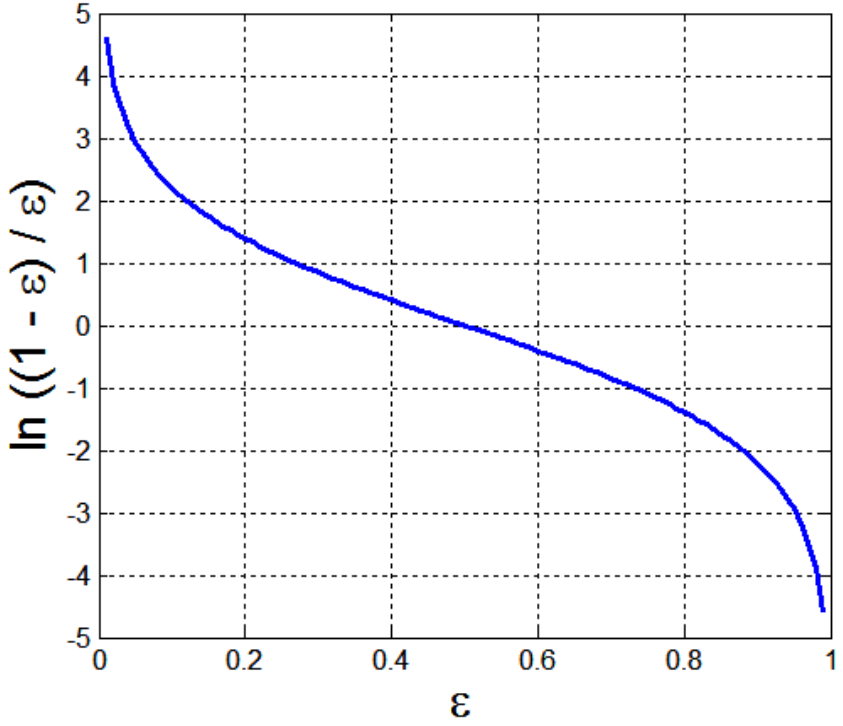
AdaBoost

- Base classifiers: C_1, C_2, …, C_T
- Error rate:

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



High positive importance when error is close to 0,
High negative importance when error is close to 1

AdaBoost Algorithm

- Weight update:

Weight associated to x_i during the j boosting round

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

$$\text{where } Z_j \text{ is the normalization factor}$$

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated

- Classification:

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

AdaBoost Algorithm

Algorithm 5.7 AdaBoost Algorithm

1: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \cdots, n\}$. {Initialize the weights for all n instances.}
2: Let k be the number of boosting rounds.
3: **for** $i = 1$ to k **do**
4: Create training set D_i by sampling (with replacement) from D according to $\mathbf{w}$.
5: Train a base classifier C_i on D_i.
6: Apply C_i to all instances in the original training set, D.
7: $\epsilon_i = \frac{1}{n}\left[\sum_j w_j \, \delta\left(C_i(x_j) \neq y_j\right)\right]$ {Calculate the weighted error}
8: **if** $\epsilon_i > 0.5$ **then**
9: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \cdots, n\}$. {Reset the weights for all n instances.}
10: Go back to Step 4.
11: **end if**
12: $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13: Update the weight of each instance according to equation (5.88).
14: **end for**
15: $C^*(\mathbf{x}) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y))$.

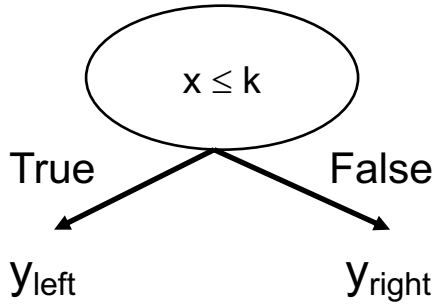
AdaBoost Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump
  - Decision rule: $x \leq k$ versus $x > k$
  - Split point k is chosen based on entropy



AdaBoost Example

- Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

- Weights:

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

AdaBoost Example

- Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

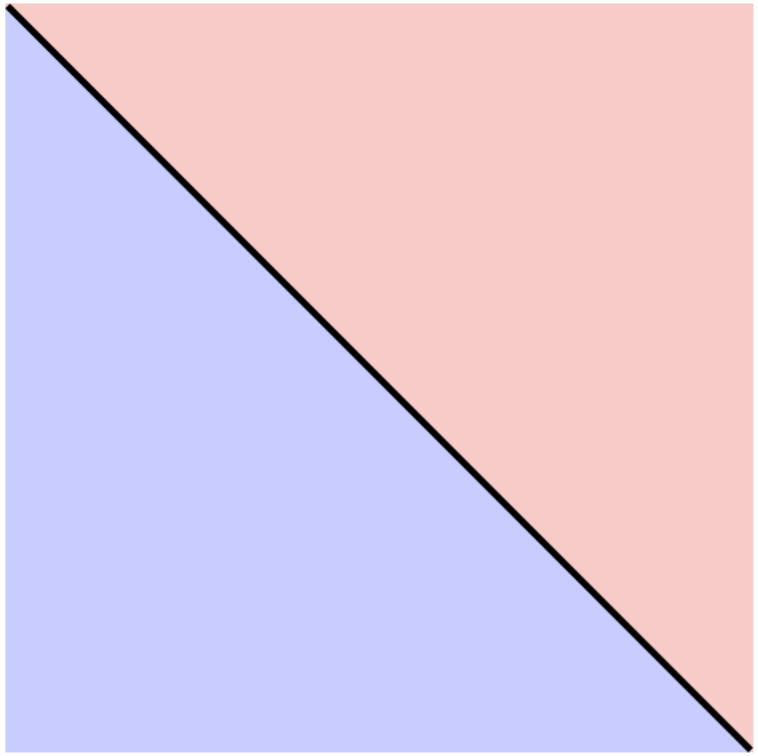
- Classification

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

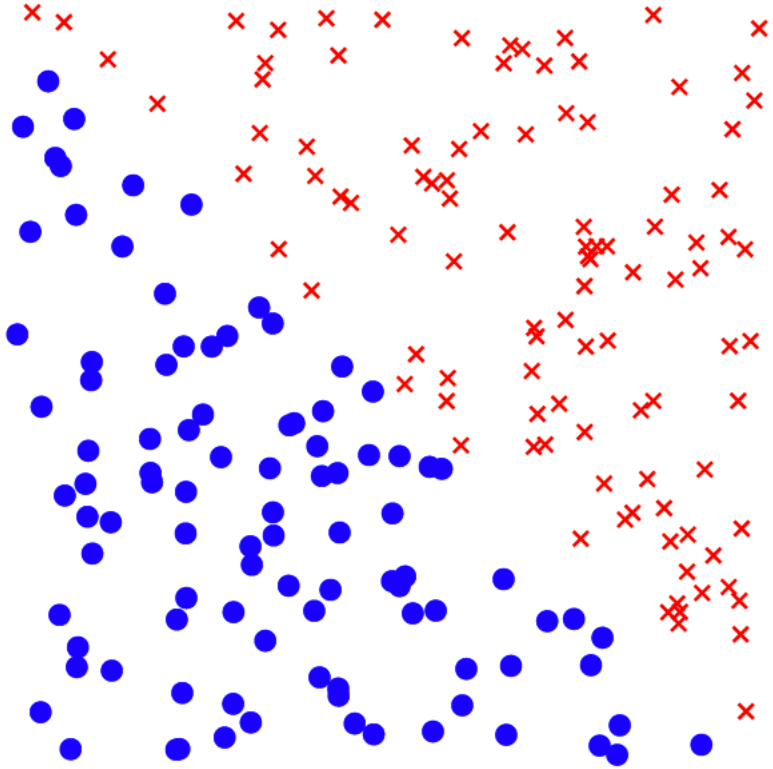
Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

AdaBoost Visual Example



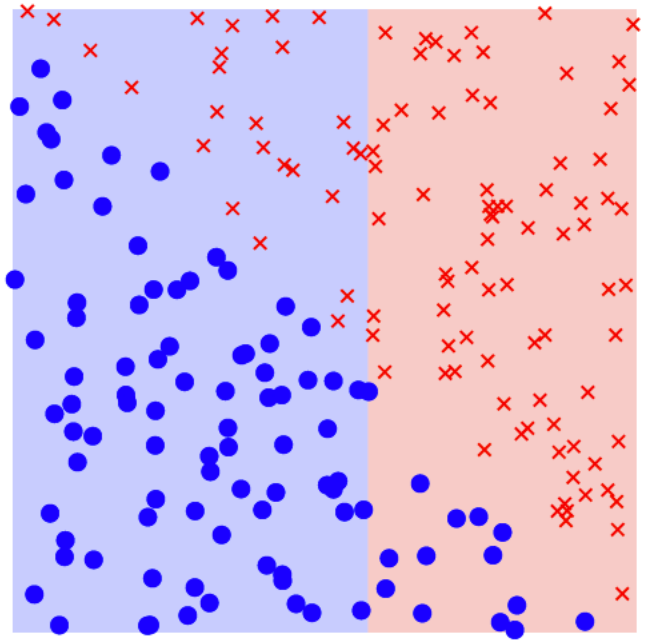
True decision boundary



Training data

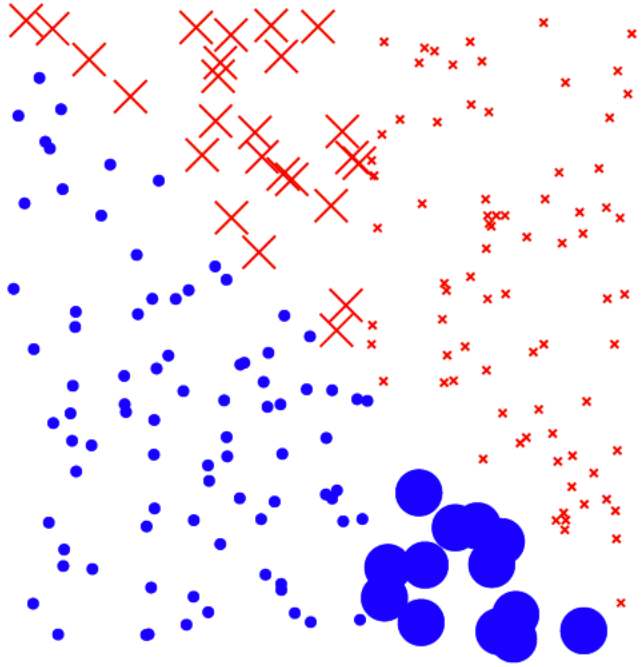
AdaBoost Visual Example

Round 1



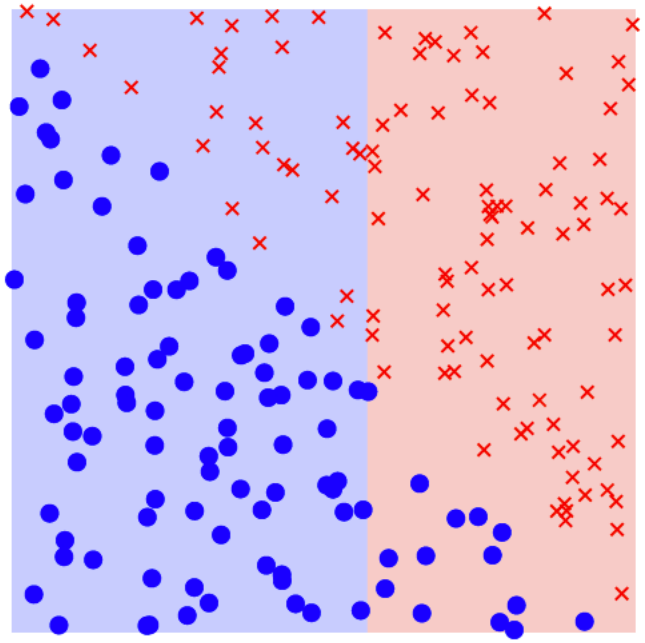
Chosen weak classifier

$\epsilon_1 = 0.19,\ \alpha_1 = 1.45$



Re-weight training points

$w_i^{(2)}$**'s**

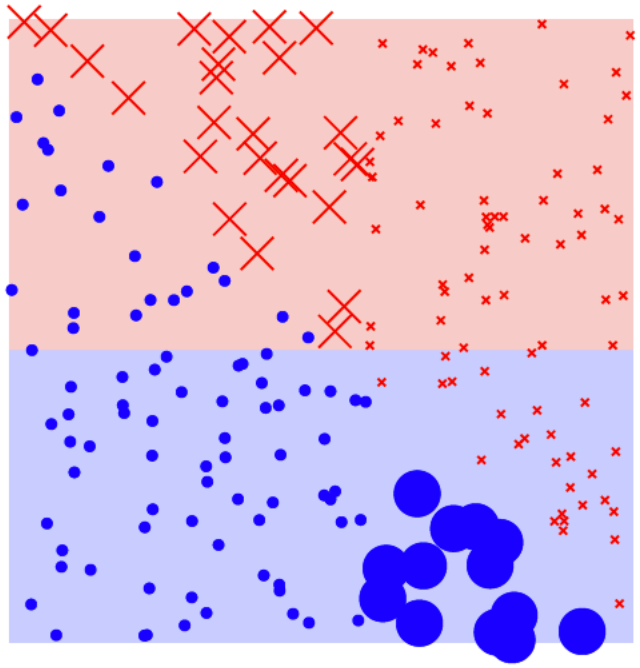


Current strong classifier

$f_2(\mathbf{x})$

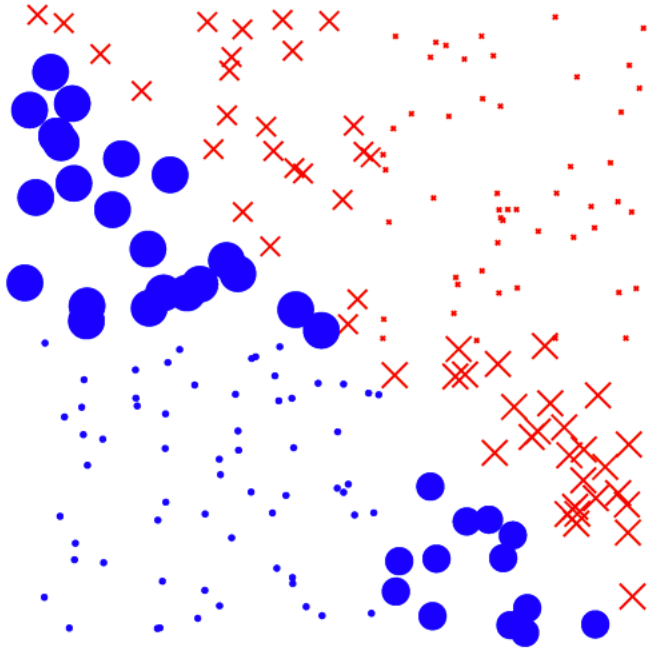
AdaBoost Visual Example

Round 2



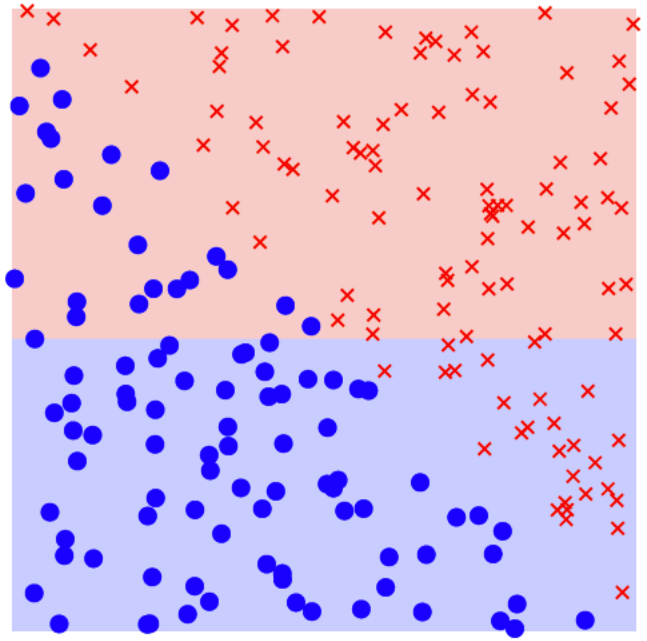
Chosen weak classifier

$\epsilon_2 = 0.1512,\ \alpha_2 = 1.725$



Re-weight training points

$w_i^{(3)}$**'s**

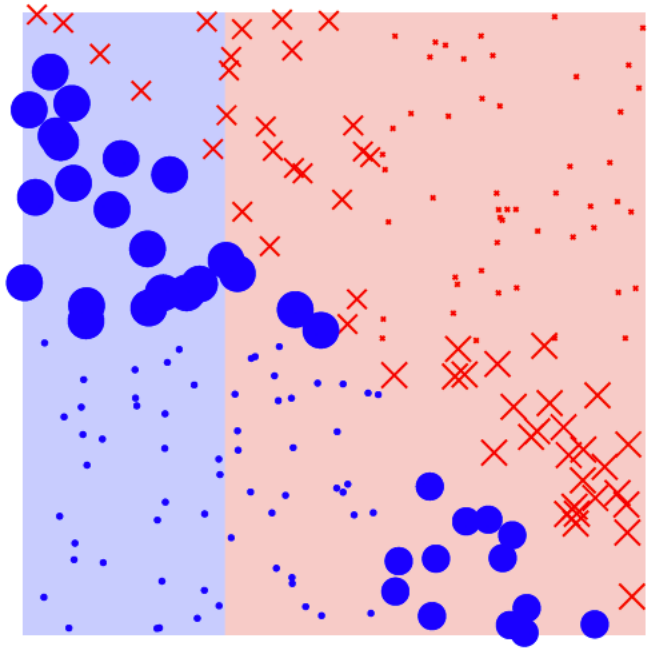


Current strong classifier

$f_2(\mathbf{x})$

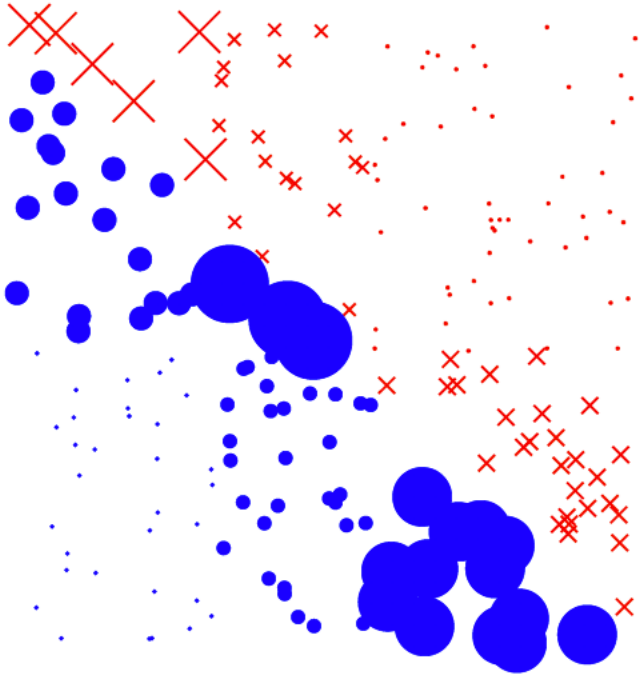
AdaBoost Visual Example

Round 3



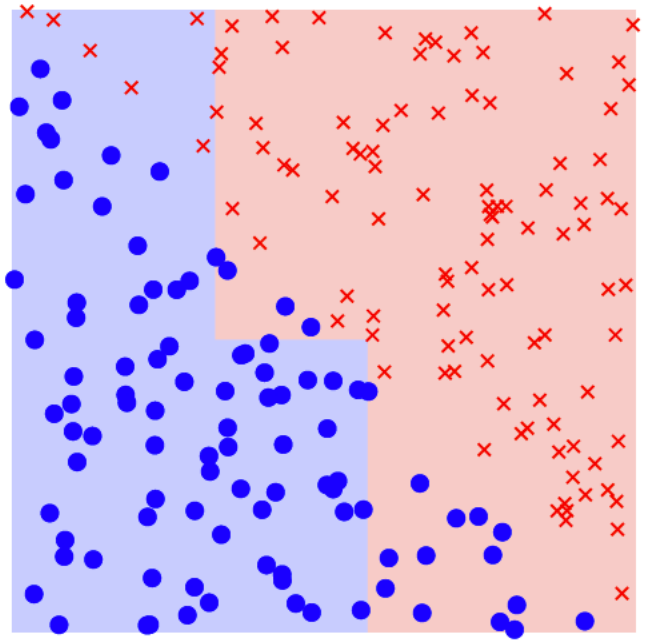
Chosen weak classifier

$\epsilon_3 = 0.2324,\ \alpha_3 = 1.1946$



Re-weight training points

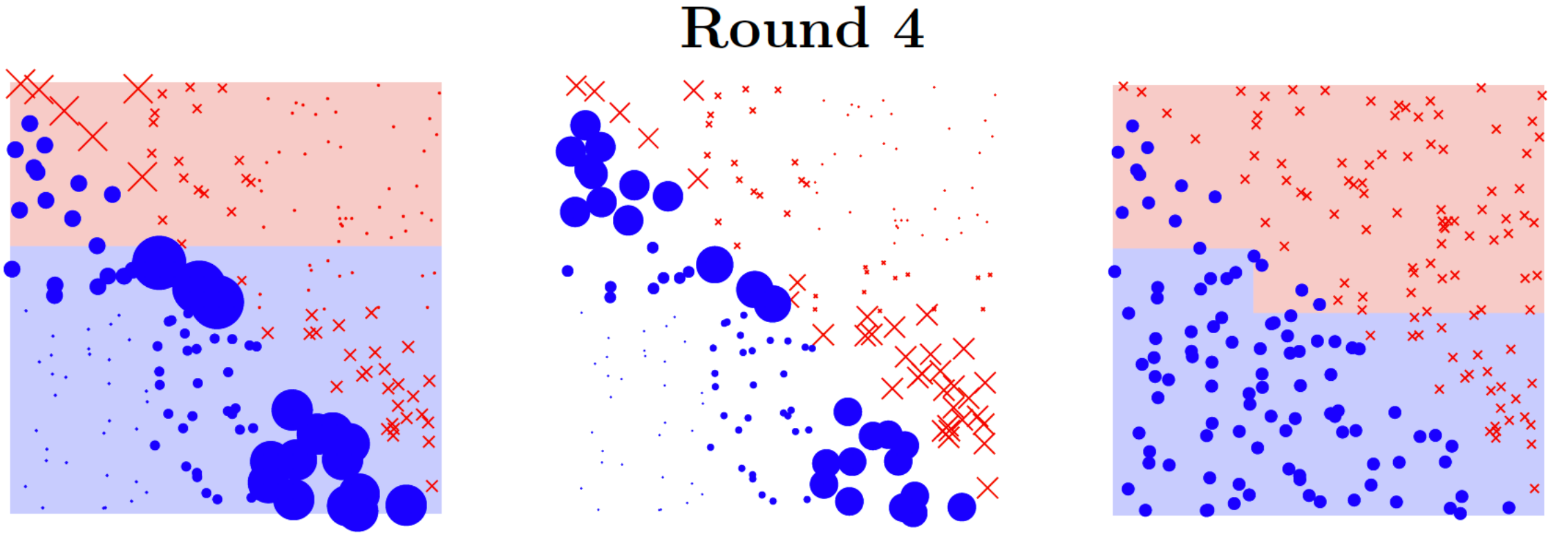
$w_i^{(4)}$**'s**



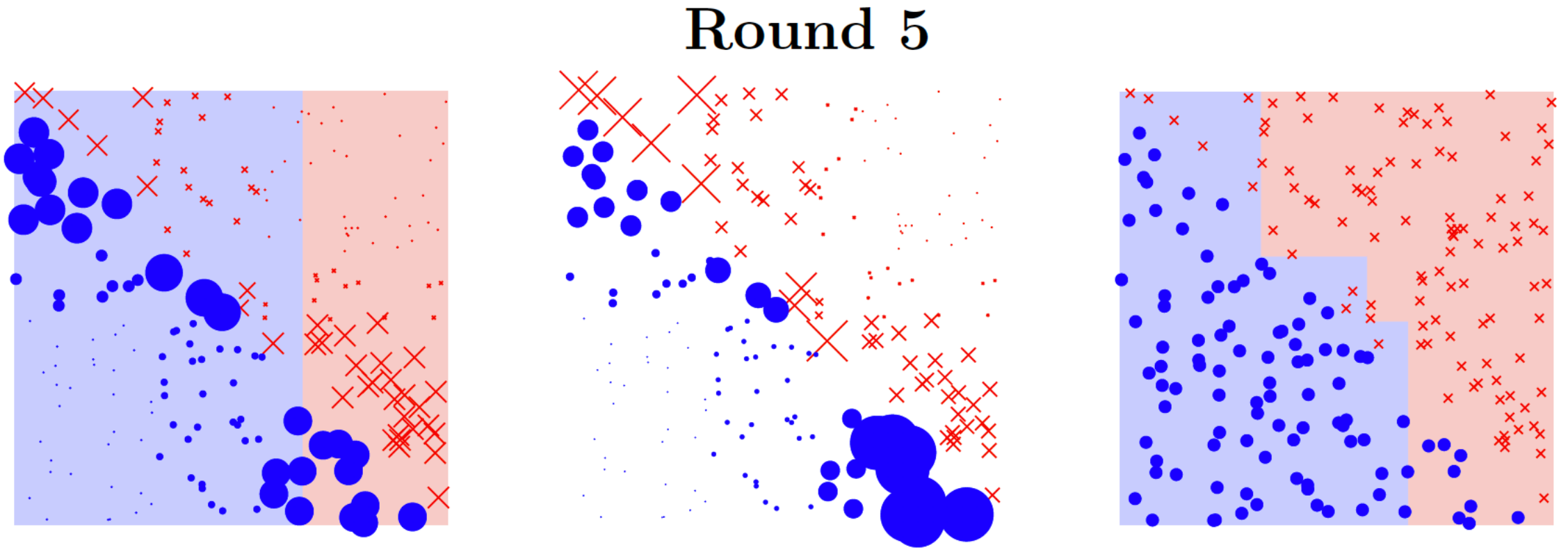
Current strong classifier

$f_3(\mathbf{x})$

AdaBoost Visual Example



AdaBoost Visual Example



Chosen weak classifier

$\epsilon_5 = 0.2616,\ \alpha_5 = 1.0375$

Re-weight training points

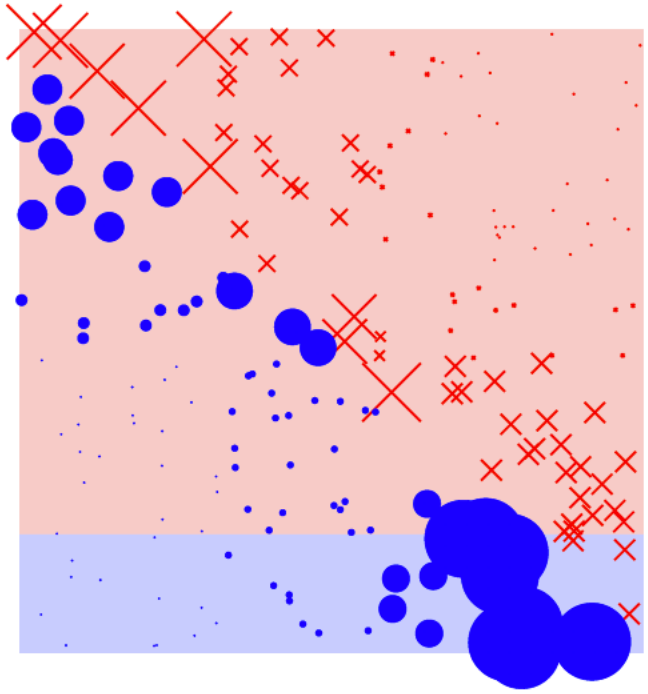
$w_i^{(6)}$**'s**

Current strong classifier

$f_5(\mathbf{x})$

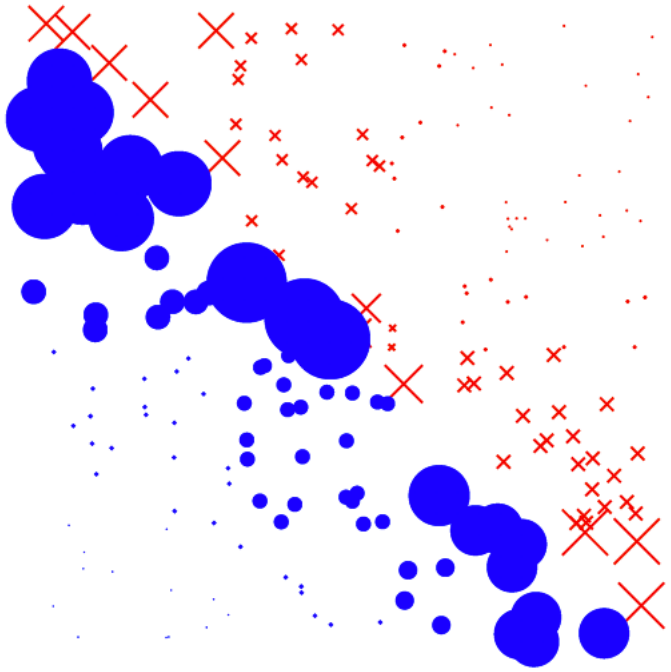
AdaBoost Visual Example

Round 6



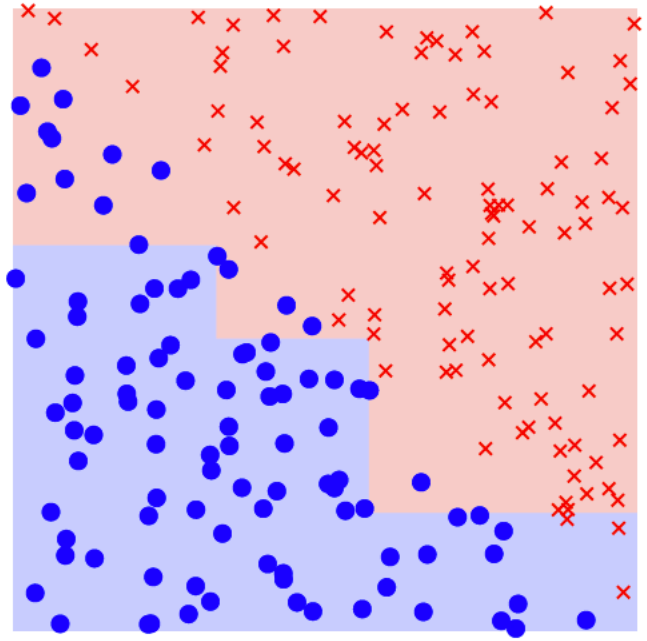
Chosen weak classifier

$\epsilon_6 = 0.2262,\ \alpha_6 = 1.2298$



Re-weight training points

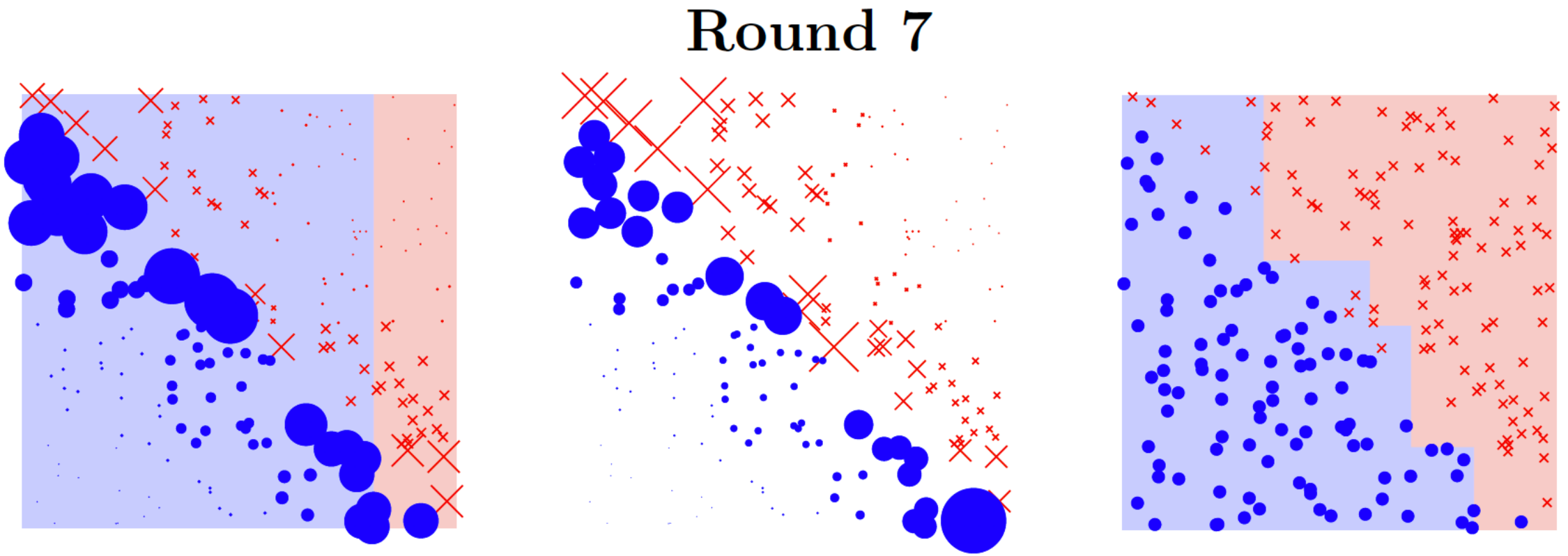
$w_i^{(7)}$**'s**



Current strong classifier

$f_6(\mathbf{x})$

AdaBoost Visual Example



Chosen weak classifier

$\epsilon_7 = 0.2680,\ \alpha_7 = 1.0049$

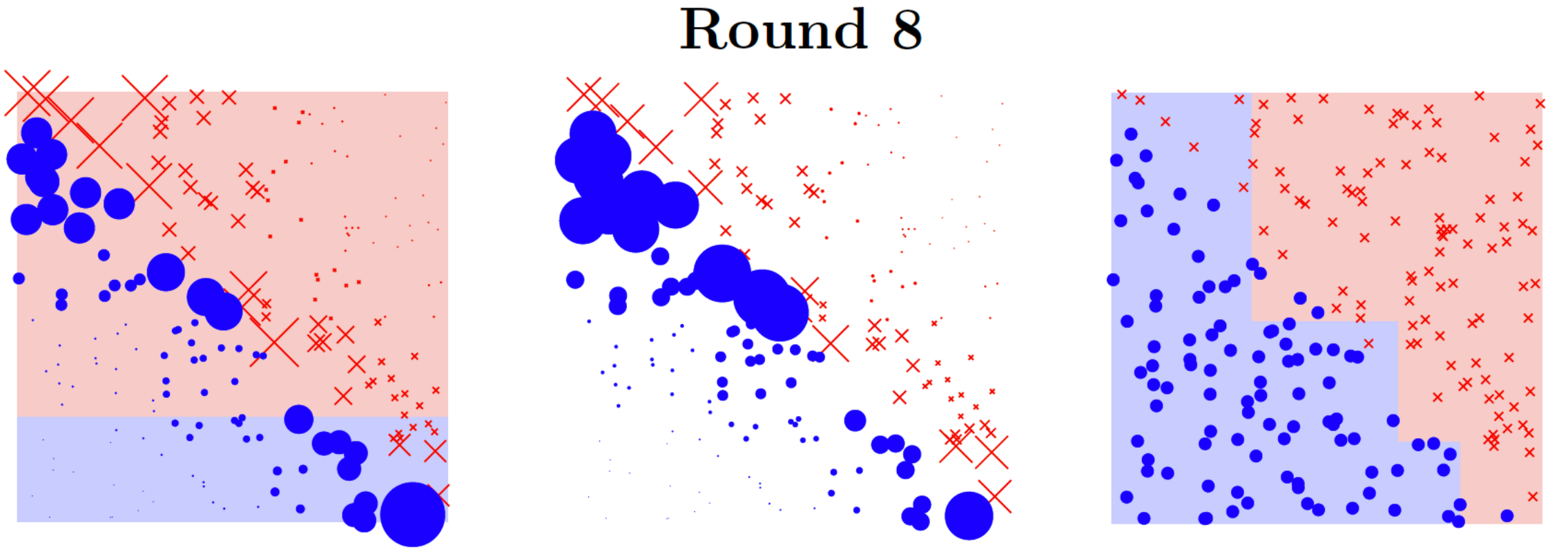
Re-weight training points

$w_i^{(8)}$**'s**

Current strong classifier

$f_7(\mathbf{x})$

AdaBoost Visual Example



Chosen weak classifier

$\epsilon_8 = 0.3282,\ \alpha_8 = 0.7165$

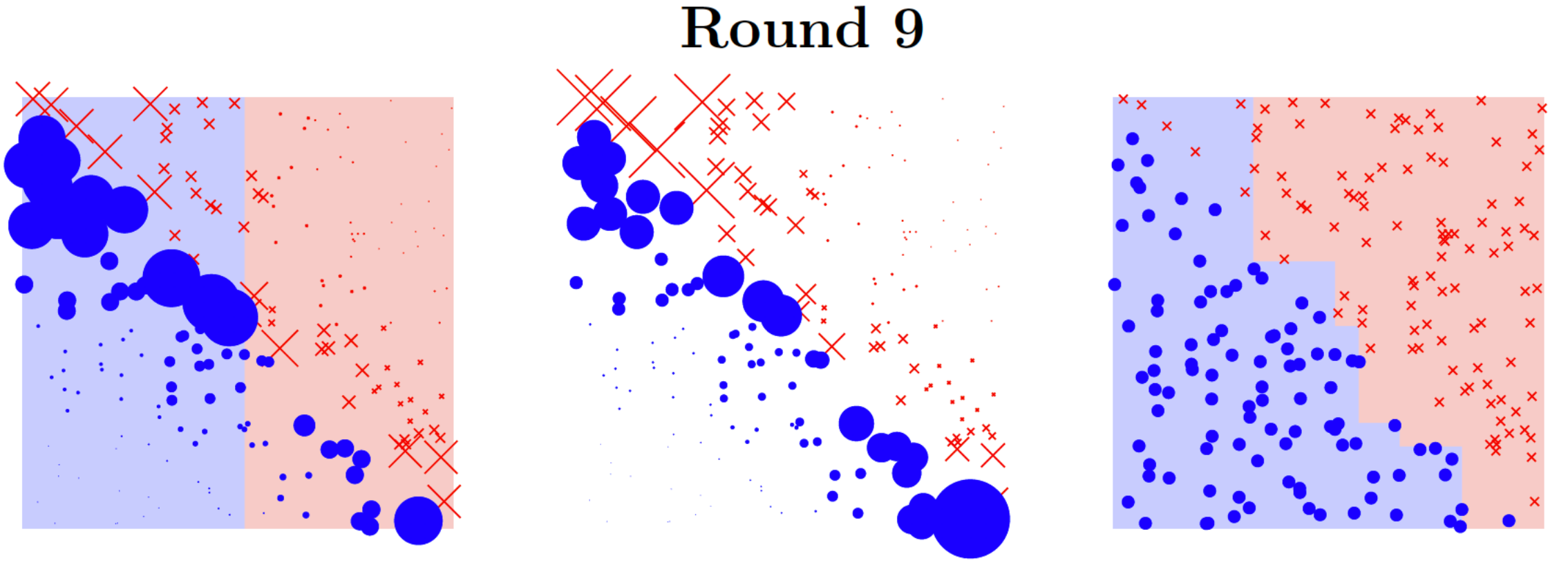
Re-weight training points

$w_i^{(9)}$**'s**

Current strong classifier

$f_8(\mathbf{x})$

AdaBoost Visual Example



Chosen weak classifier

$\epsilon_9 = 0.3048,\ \alpha_9 = 0.8246$

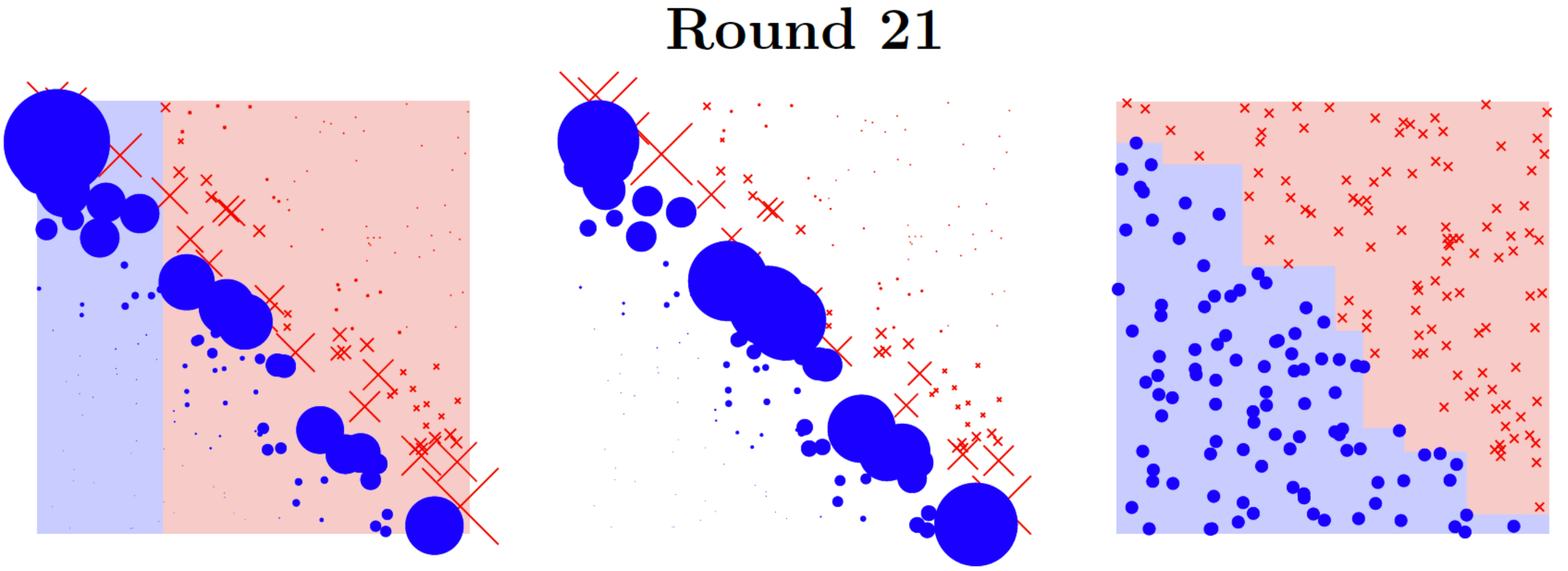
Re-weight training points

$w_i^{(10)}$**'s**

Current strong classifier

$f_9(\mathbf{x})$

AdaBoost Visual Example



Chosen weak classifier

$\epsilon_{21} = 0.3491,\ \alpha_{21} = 0.6232$

Re-weight training points

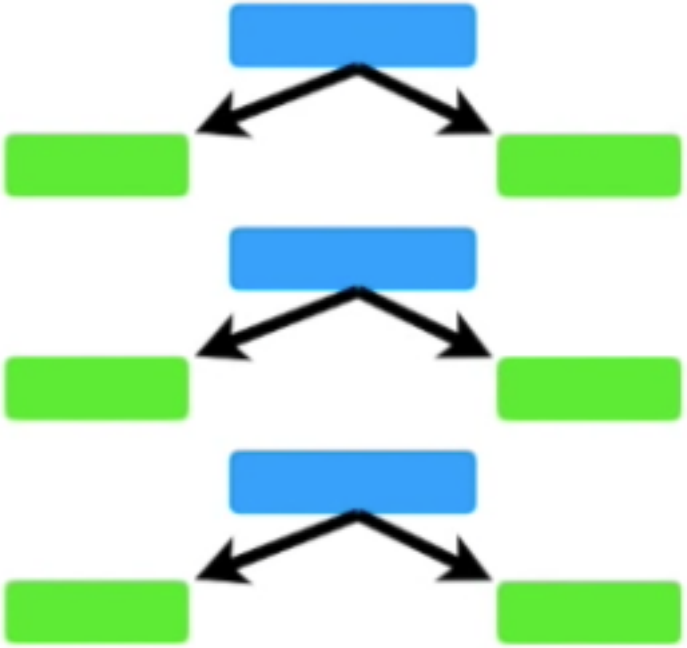
$w_i^{(22)}$**'s**

Current strong classifier

$f_{21}(\mathbf{x})$

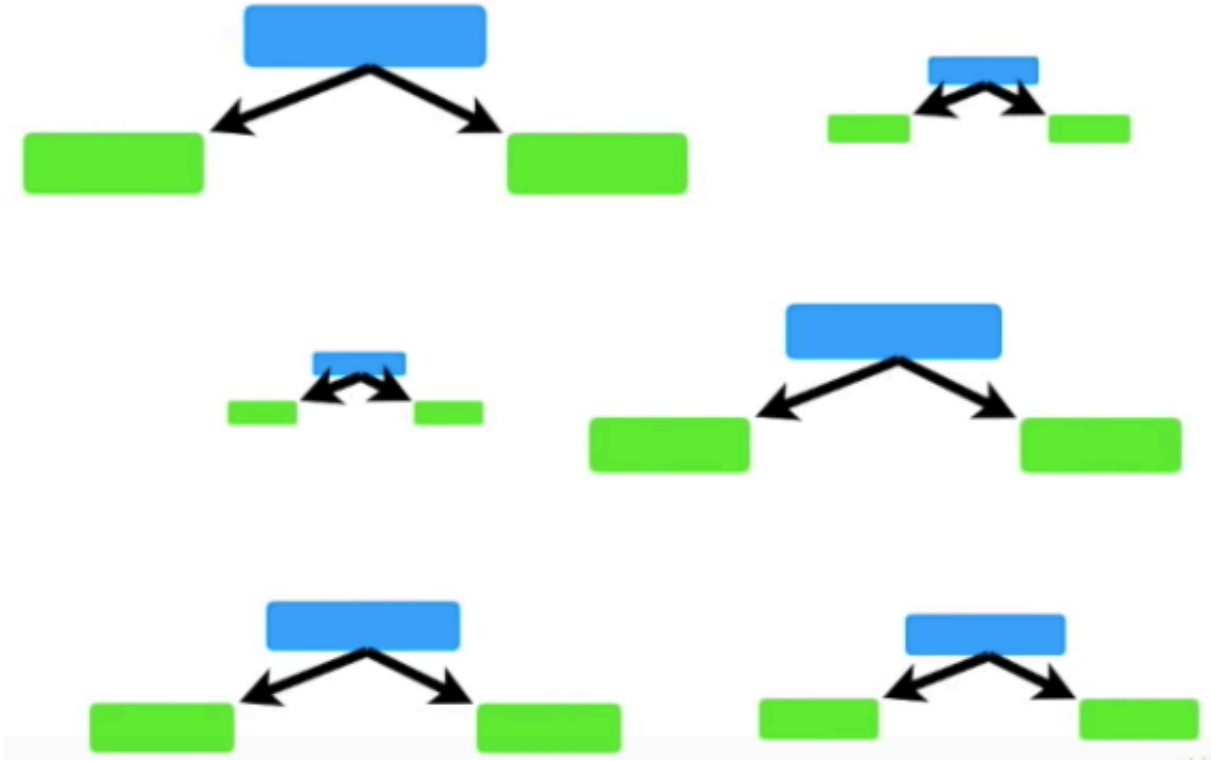
AdaBoost - Reloaded

- AdaBoost uses **stumps**, i.e., decision tree with only one node and two leaves.
- Thus, it uses a forest of stumps.
- Stumps are not good at making accurate classifications.
- Stumps are *weak* learners.



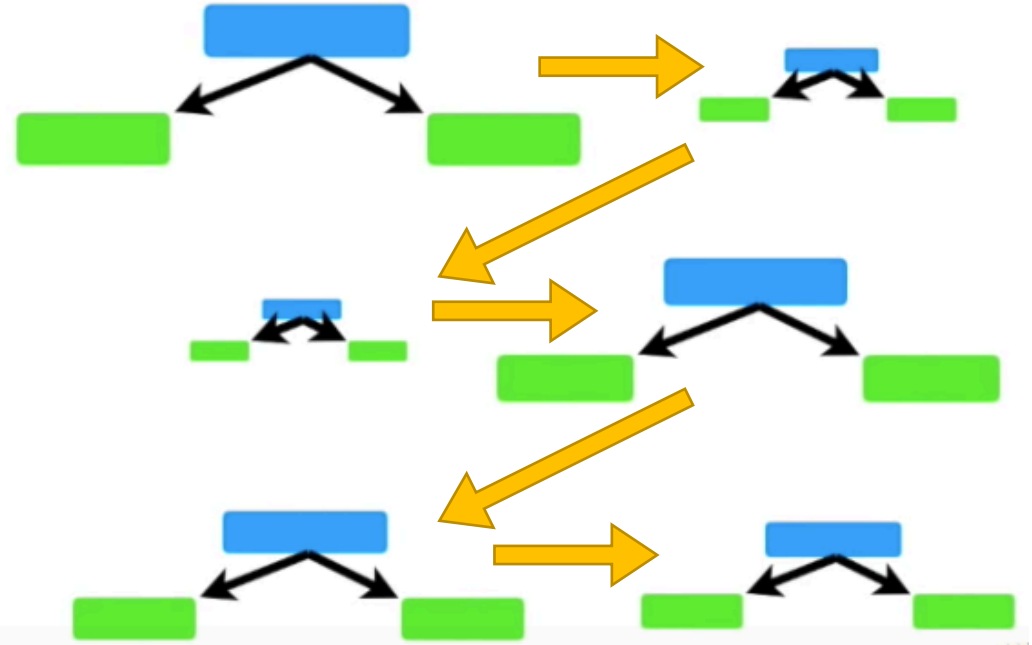
AdaBoost - Reloaded

- In forest of stumps some stumps have more importance in the final classification.



AdaBoost - Reloaded

- Each stump **is not** made independently from the others.
- The error that the first stump makes influences the second stump and so on and so forth.



AdaBoost – Training Step by Step

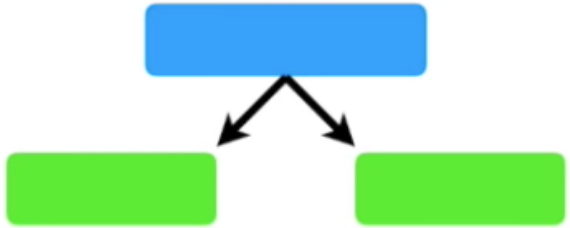
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

- Initially all the records get the same weight: 1/number of samples.

AdaBoost – Training Step by Step

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

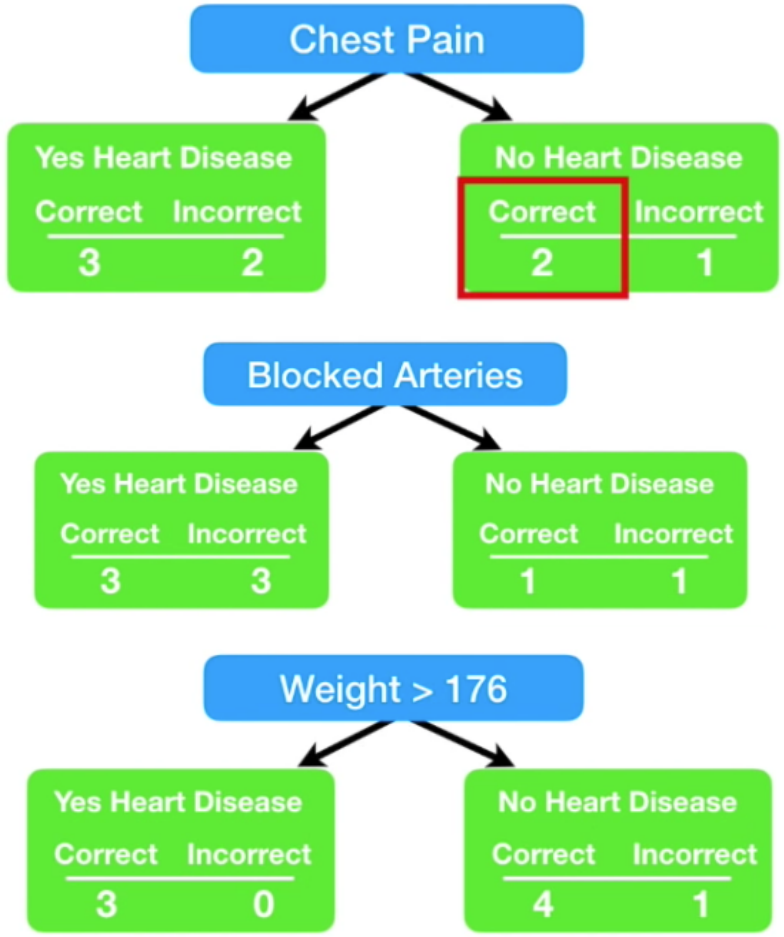
- We search for the first stump of the forest.



- All the weights are the same thus we can ignore them now.

AdaBoost – Training Step by Step

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

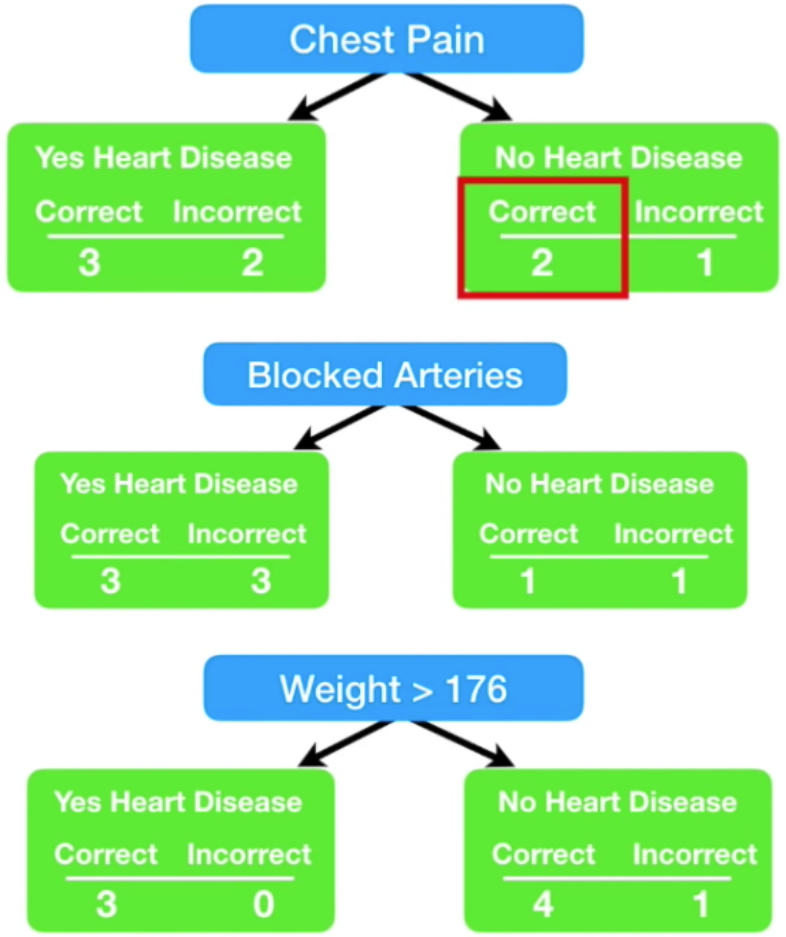


AdaBoost – Training Step by Step

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

Gini Index



First Stump in the Forest

AdaBoost – Training Step by Step

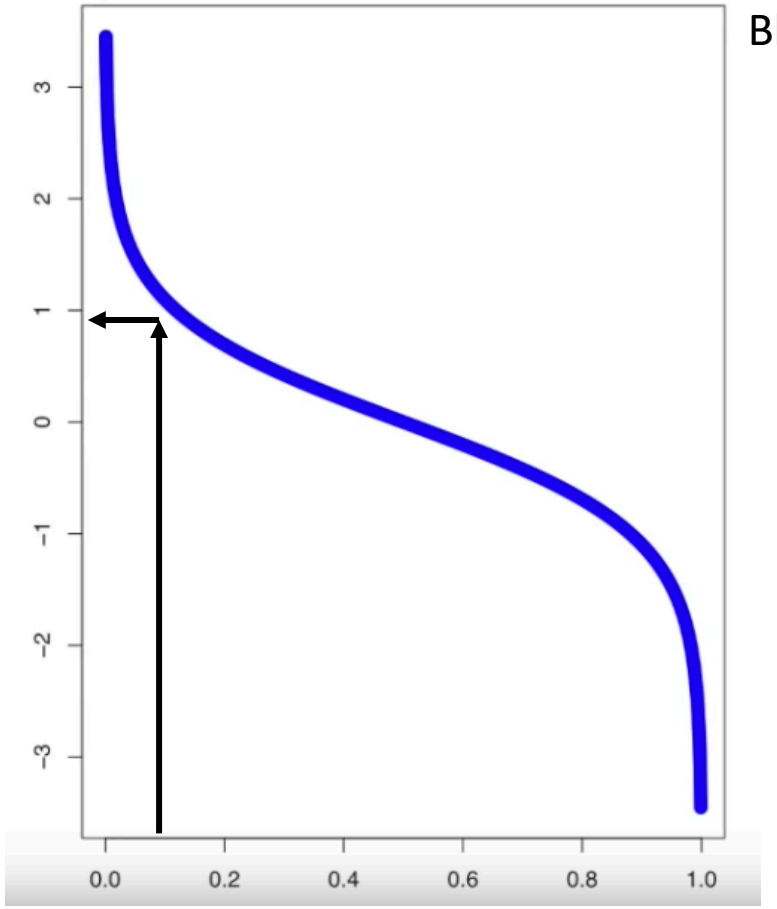
- Now we need to determine the importance of this stump in the final classification.
- We determine the importance based on how well it classified the samples in terms of *total error* in the following formula:
- The total error is always between 0 and 1.

Total Error = 1/8

$$\frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

AdaBoost – Training Step by Step

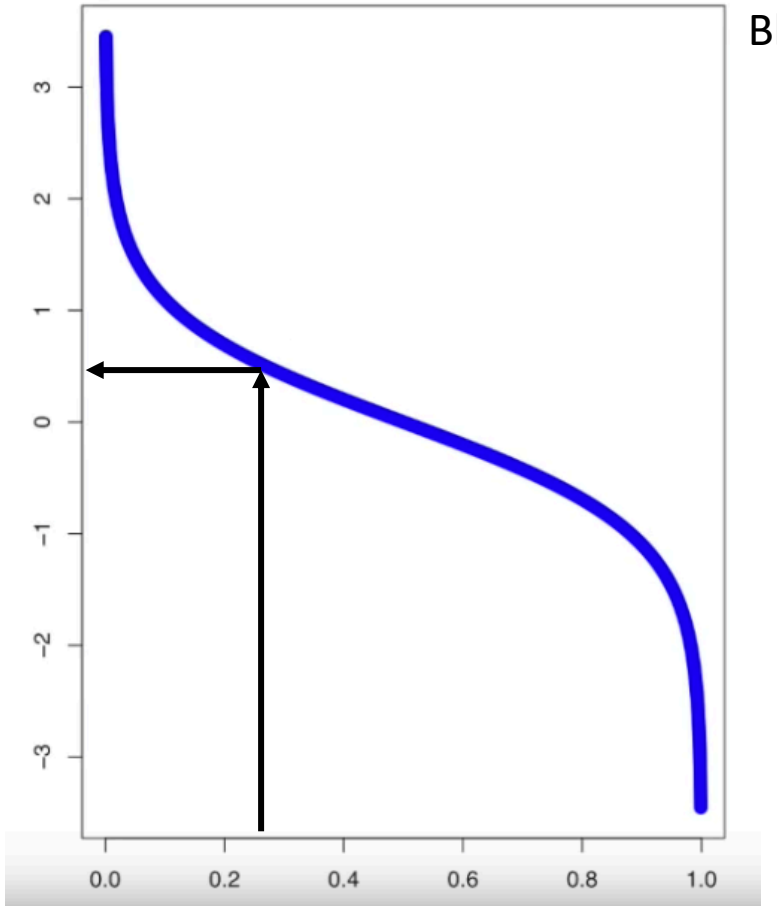


Total Error

Blue line is the Importance

- In our case the importance is 0.97 = 1/2 log(7)

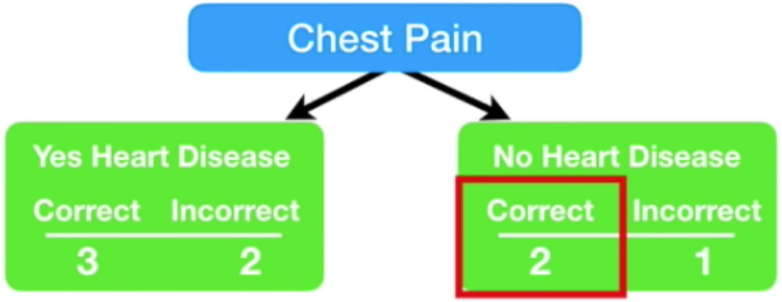
AdaBoost – Training Step by Step



Total Error

Blue line is the Importance

- If it had been Chest Pain
- Total error = 3/8
- Importance 1/2 log(7/3) = 0.42



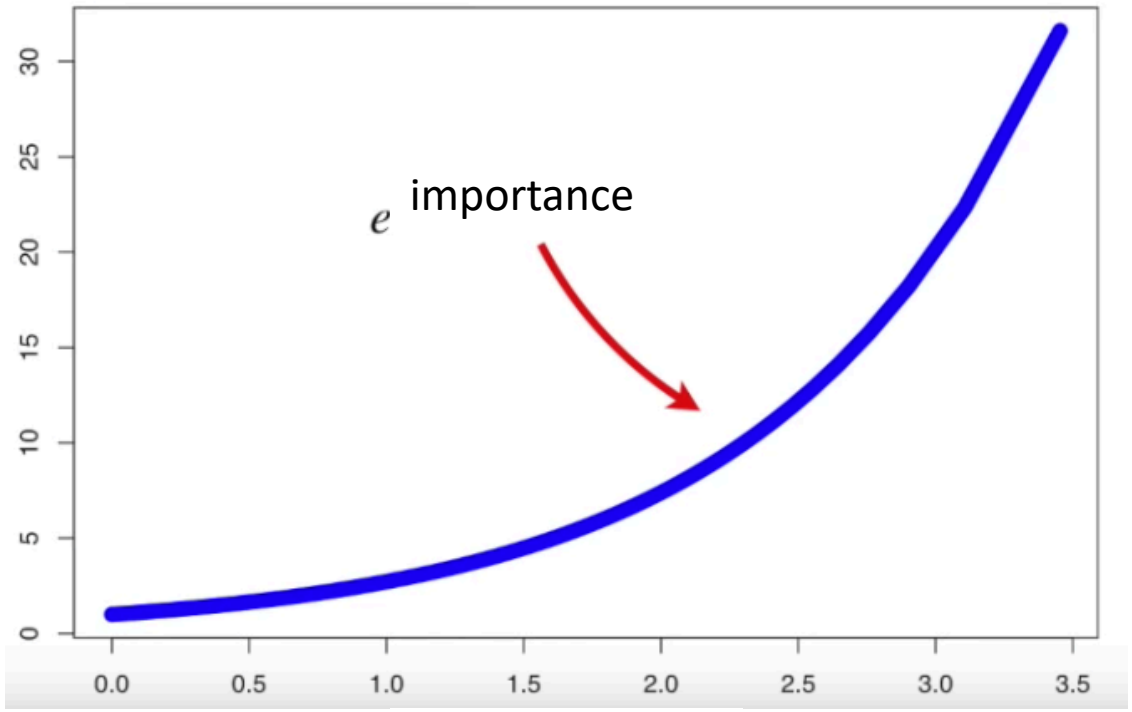
AdaBoost – Training Step by Step

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

- Update weights.
- We need to emphasize the usage of this sample in the next iteration by increasing its weight and decreasing the others.

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{importance}}$$

AdaBoost – Training Step by Step



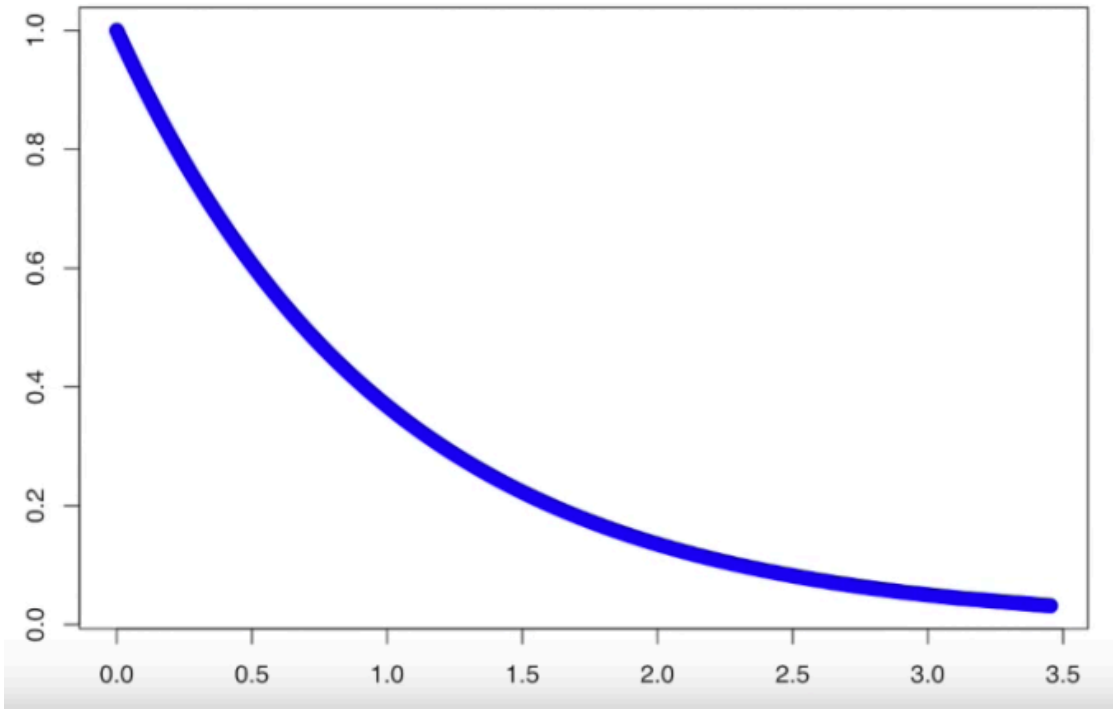
$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{importance}}$$

$= 1/8\ e^{0.97} = 1/8 * 2.64 = 0.33 > 0.125 = 1/8$

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

AdaBoost – Training Step by Step



importance

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{-importance}}$$

$= 1/8\ e^{-0.97} = 1/8 * 0.38 = 0.05 < 0.125 = 1/8$

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

AdaBoost – Training Step by Step

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	New Weight	Norm. Weight
Yes	Yes	205	Yes	1/8	0.05	0.07
No	Yes	180	Yes	1/8	0.05	0.07
Yes	No	210	Yes	1/8	0.05	0.07
Yes	Yes	167	Yes	1/8	0.33	0.49
No	Yes	156	No	1/8	0.05	0.07
No	Yes	125	No	1/8	0.05	0.07
Yes	No	168	No	1/8	0.05	0.07
Yes	Yes	172	No	1/8	0.05	0.07

- Report the new weights.
- Normalize the new weights so that they sum up to 1.
- Repalce the old ones.

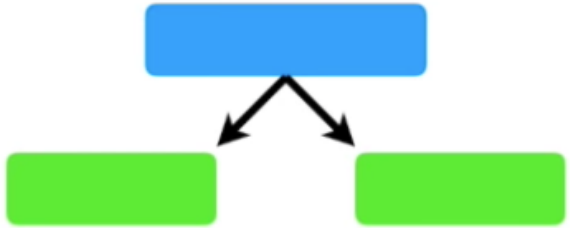
$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

AdaBoost – Training Step by Step

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

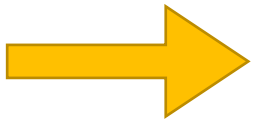
- Now we can use the modified sample weight to make the second stump in the forest.



- In theory we could use weighted Gini Index to determine which variable should split the next stump.
- Alternatively, we can make a new training set that contains duplicate copies of the samples with the largest sample weights.

AdaBoost – Training Step by Step

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07



Then we use the new collection of samples to repeat the process.

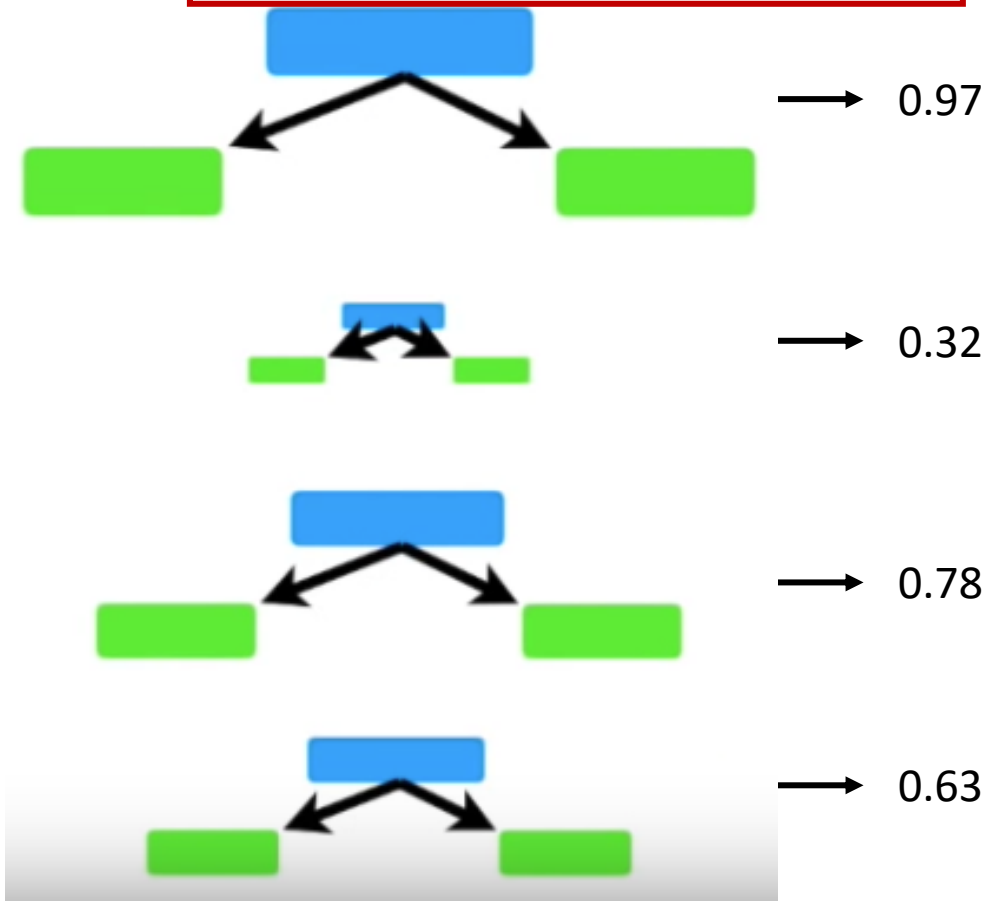
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
No	Yes	156	No
Yes	Yes	167	Yes
No	Yes	125	No
Yes	Yes	167	Yes
Yes	Yes	167	Yes
Yes	Yes	172	No
Yes	Yes	205	Yes
Yes	Yes	167	Yes

AdaBoost – Classification

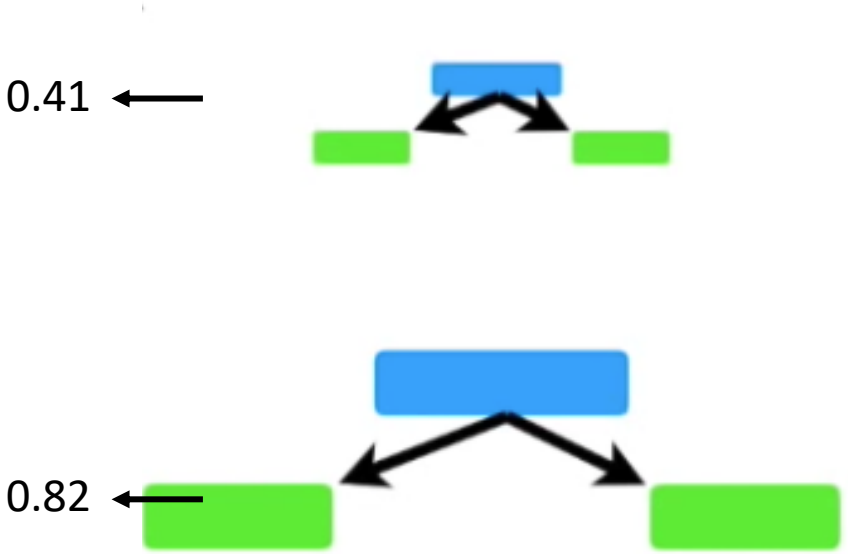
Given a patient

Chest Pain	Blocked Arteries	Patient Weight
Yes	Yes	205

Heart Disease = Yes Total = 2.7



Total = 1.23 Heart Disease = No



References

- Ensemble Methods. Chapter 5.6. Introduction to Data Mining.