Boostrap aggregating (Bagging)

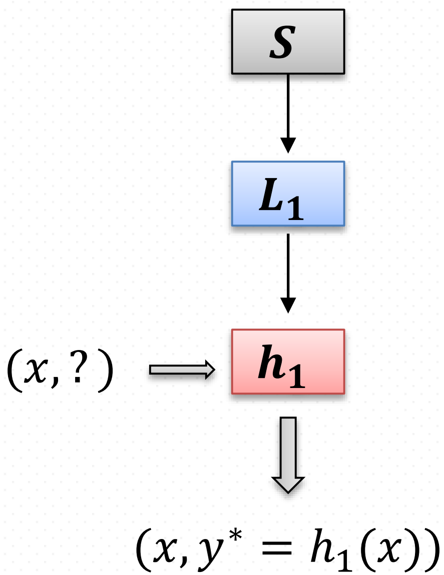
- An ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms
- Can be used in both regression and classification
- Reduces variance and helps to avoid overfitting
- Usually applied to decision trees, though it can be used with any type of method

An Aside: Ensemble Methods

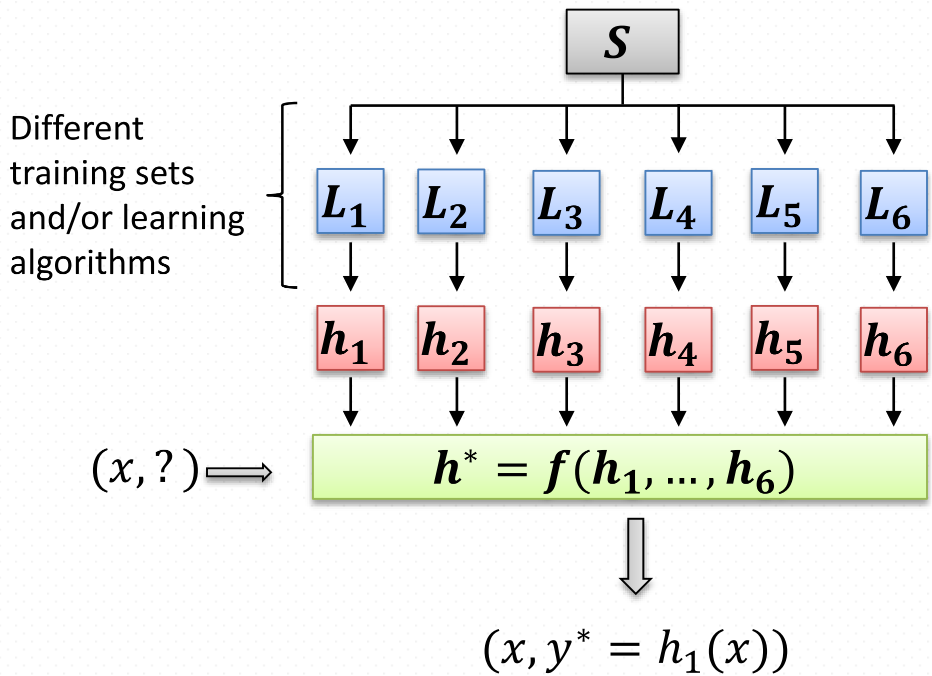
- In a nutshell:
  - A combination of multiple learning algorithms with the goal of achieving better predictive performance than could be obtained from any of these classifiers alone
  - A meta-algorithm that can be considered to be, in itself, a supervised learning algorithm since it produces a single hypothesis
  - Tend to work better when there is diversity among the models
  - Examples:
    - Bagging
    - Boosting
    - Bucket
    - Stacking

An Aside: Ensemble Methods

Traditional:



Ensemble Method:



Back to Bagging

- The idea:
  1. Create N boostrap samples $\{S_1, \dots, S_N\}$ of S as follows:
     - For each S_i, randomly draw $|S|$ examples from S with replacement
  2. For each $i = 1, \dots, N$

     $h_i = \mathrm{Learn}(\mathrm{S_i})$
  1. Output $H =< \{h_1, \dots, h_N\}, majorityVote >$

Most notable benefits

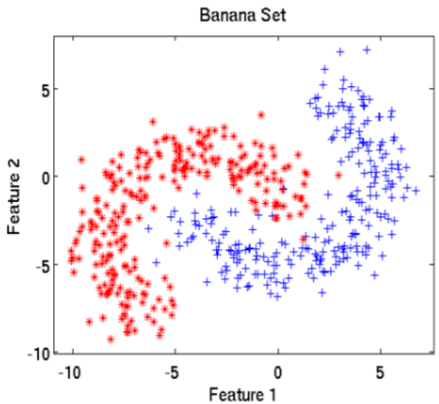
1. Surprisingly competitive performance & rarely overfits
2. Is capable of reducing variance of constituent models
3. Improves ability to ignore irrelevant features

Remember:

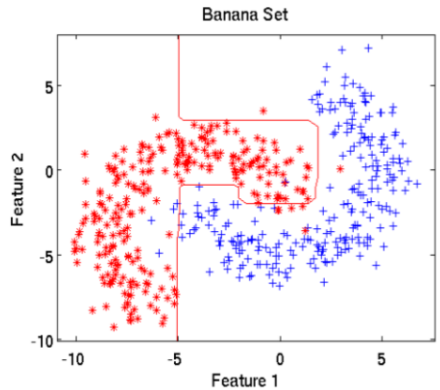
$$\boldsymbol{error(x)} = \mathbf{noise(x)} + \mathbf{bias(x)} + \mathbf{variance(x)}$$

Variance: how much does prediction change if we change the training set?

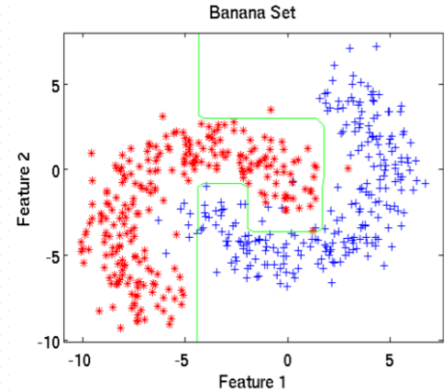
Bagging Example 1



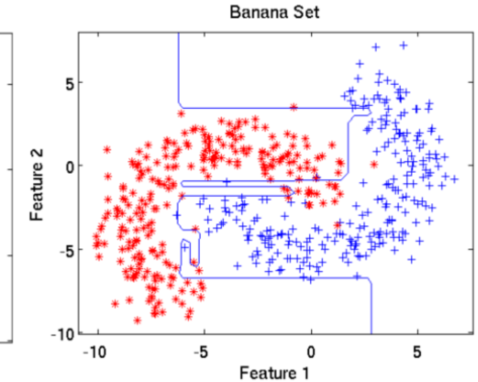
Training data



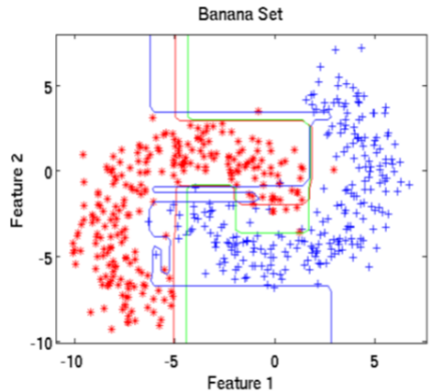
Decision boundary produced by one tree



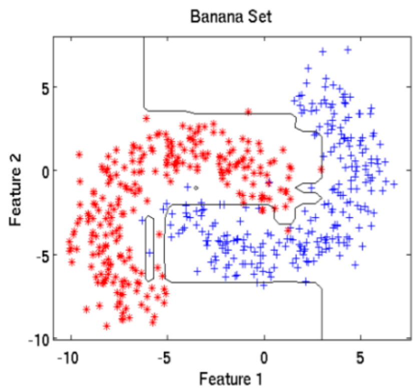
Decision boundary produced by a second tree



Decision boundary produced by a third tree

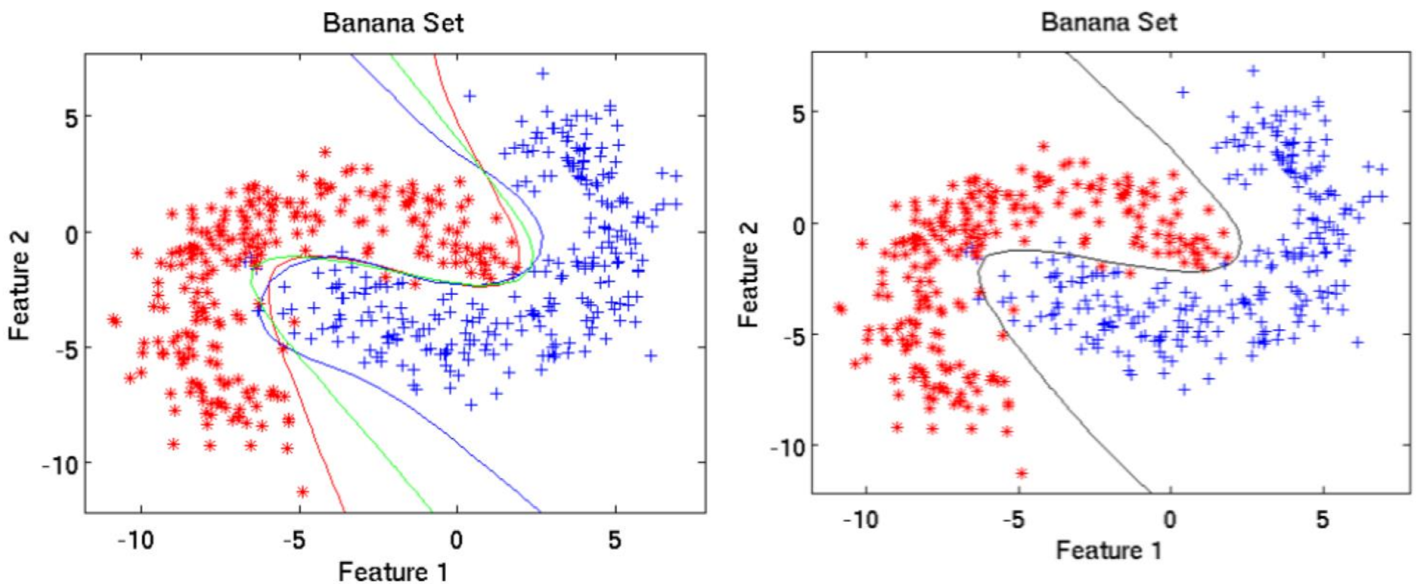


Three trees and final boundary overlaid



Final result from bagging all trees.

Bagging Example 2



Three neural nets generated with default settings [bpxnc]

Final output from bagging 10 neural nets

Bagging: Neural Net

Bagging Example 3 (1)

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	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> 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 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

Figure 5.35. Example of bagging.

Bagging Example 3 (2)

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
True Class	1	1	1	-1	-1	-1	-1	1	1	1

Figure 5.36. Example of combining classifiers constructed using the bagging approach.

Accuracy: 100%

How does bagging minimize error?

- Ensemble reduces the overall variance
- Let $f(x)$ be the target value of x, h_1 to h_n be the set of base hypothesis, and h_{avg} be the prediction of the base hypotheses
- $Error(h,x) = \left(f(x) - h(x)\right)^2$ → Squared error
- Is there any relation between h_{avg} and variance?
  – Yes

How does bagging minimize error?

- $Error(h, x) = \left(f(x) - h(x)\right)^2$
- $Error\left(h_{avg}, x\right) = \frac{\sum_1^n Error(h_i, x)}{n} = \frac{\sum_1^n \left(h_i(x) - h_{avg}(x)\right)^2}{n}$
- By the above, we see that the squared error of the average hypothesis equals the average squared error of the base hypotheses minus the variance of the base hypotheses

Stability of Learn

- A learning algorithm is **unstable** if small changes in the training data can produce large changes in the output hypothesis (otherwise stable)
- Clearly bagging will have little benefit when used with stable base learning algorithms (i.e., most ensemble members will be very similar)
- Bagging generally works best when used with unstable yet relatively accurate base learners

Bagging Summary

- Works well if the base classifiers are unstable (complement each other)
- Increased accuracy because it ***reduces the variance*** of the individual classifier
- Does not focus on any particular instance of the training data
  - Therefore, less susceptible to model over-fitting when applied to noisy data

Boosting

- Key differences with respect to bagging:
  - It is iterative:
    - **Bagging:** Each individual classifier is independent
    - **Boosting:**
      - Looks at the errors from previous classifiers to decide what to focus on for the next iteration
      - Successive classifiers depend on their predecessors
      - Key idea: place more weight on “hard” examples (i.e., instances that were misclassified on previous iterations)

Historical Notes

- The idea of boosting began with a learning theory question first asked in the late 80’s
- The question was answered in 1989 by Robert Shapire resulting in the first theoretical boosting algorithm
- Shapire and Freund later developed a practical boosting algorithm called Adaboost
- Many empirical studies show that Adaboost is highly effective(very often they outperform ensembles produced by bagging)

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of a boosting round
  - Different implementations vary in terms of (1) how the weights of the training examples are updated and (2) how the predictions are combined

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

Boosting

- Equal weights are assigned to each training instance (1/N for round 1) at first round
- After a classifier $\boldsymbol{C_i}$ is learned, the weights are adjusted to allow the subsequent classifier $\boldsymbol{C_{i+1}}$ to "pay more attention" to data that were misclassified by $\boldsymbol{C_i}$.
- Final boosted classifier $\boldsymbol{C}^*$ combines the votes of each individual classifier
  - Weight of each classifier's vote is a function of its accuracy
- **Adaboost** – popular boosting algorithm

Adaboost (Adaptive Boost)

- Input:
  - Training set D containing N instances
  - T rounds
  - A classification learning scheme
- Output:
  - A composite model

Adaboost: Training Phase

- Training data D contain N labeled data :
  - $(X_1, y_1), (X_2, y_2\ ), (X_3, y_3), \ldots .\ (X_N, y_N)$
- Initially assign equal weight $1/N$ to each data
- To generate T base classifiers, we need T rounds or iterations
- Round i:
  - data from D are sampled with replacement , to form D_i (size N)
- Each data's chance of being selected in the next rounds depends on its weight
  - Each time the new sample is generated directly from the training data D with different sampling probability according to the weights; these weights are not zero

Adaboost: Training Phase

- Base classifier C_i, is derived from training data of D_i
- Error of C_i is tested using D_i
- Weights of training data are adjusted depending on how they were classified
  - Correctly classified: Decrease weight
  - Incorrectly classified: Increase weight
- Weight of a data indicates how hard it is to classify it (directly proportional)

Adaboost: Testing Phase

- The lower a classifier error rate, the more accurate it is, and therefore, the higher its weight for voting should be
- Weight of a classifier C_i' s vote is

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

- Testing:
  - For each class c, sum the weights of each classifier that assigned class c to X (unseen data)
  - The class with the highest sum is the WINNER!

$$C^*(x_{test}) = \arg\max_y \sum_{i=1}^{T} \alpha_i \delta\left(C_i(x_{test}) = y\right)$$

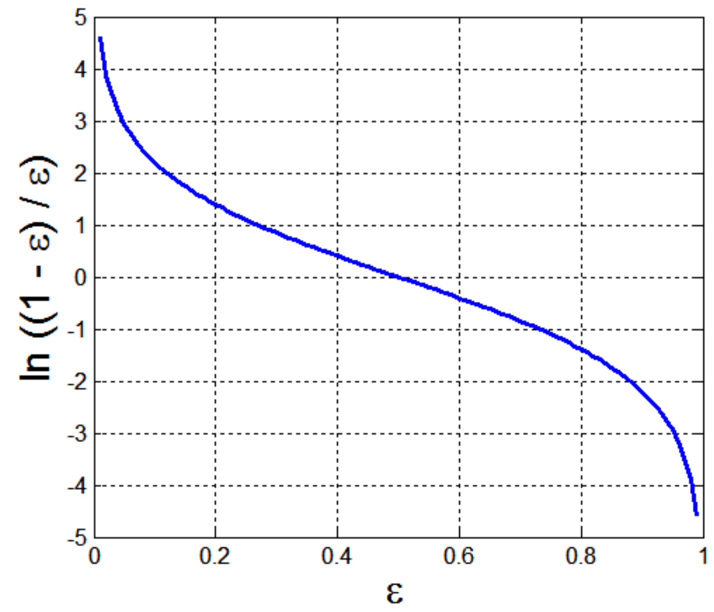
Example: Error and Classifier Weight in AdaBoost

- Base classifiers: $C_1, C_2, \ldots, C_T$
- Error rate:
  - i = index of classifier
  - j=index of instance

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

- Importance of a classifier:

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



Example: Data Instance Weight in AdaBoost

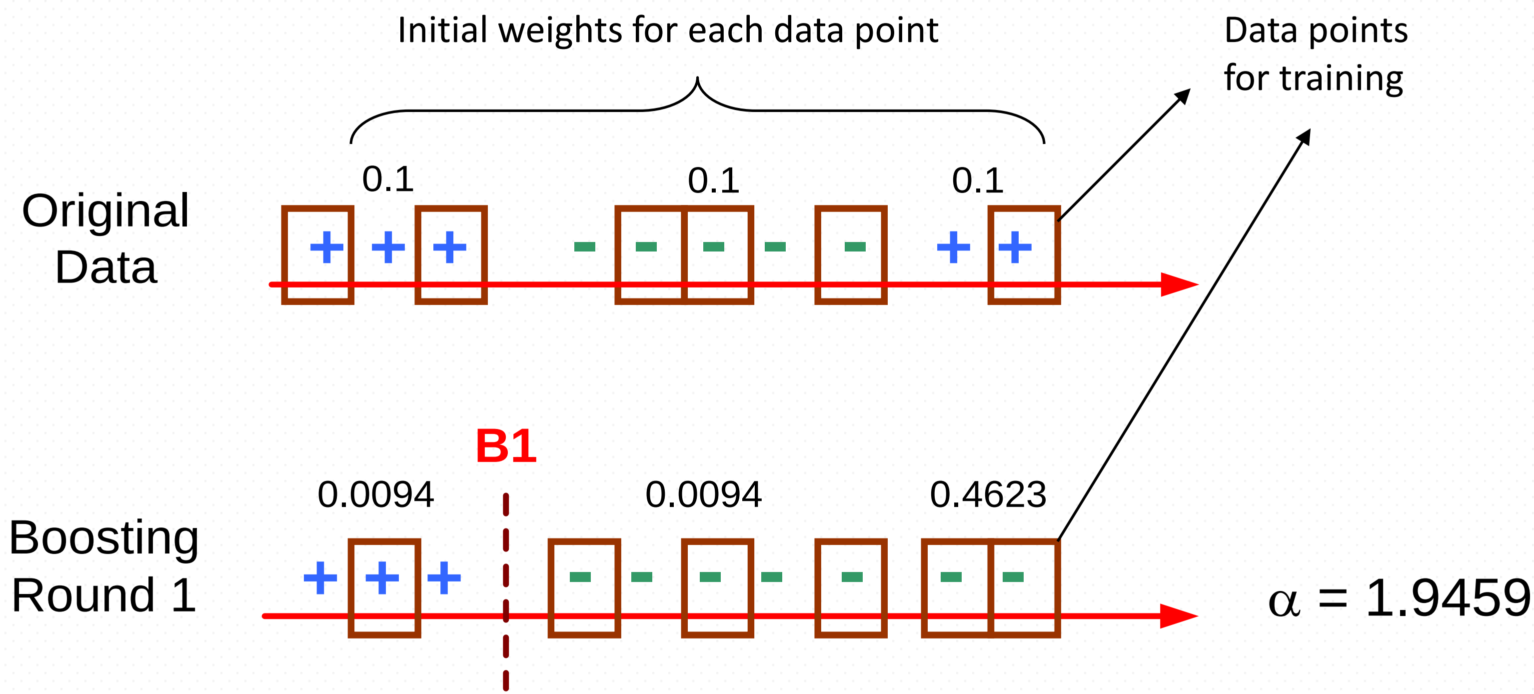
- Assume: N training data in D, T rounds, (x_j, y_j) are the training data, C_i, a_i are the classifier and weight of the i^{th} round, respectively
- Weight update on all training data in D:

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \begin{cases} \exp^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ \exp^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

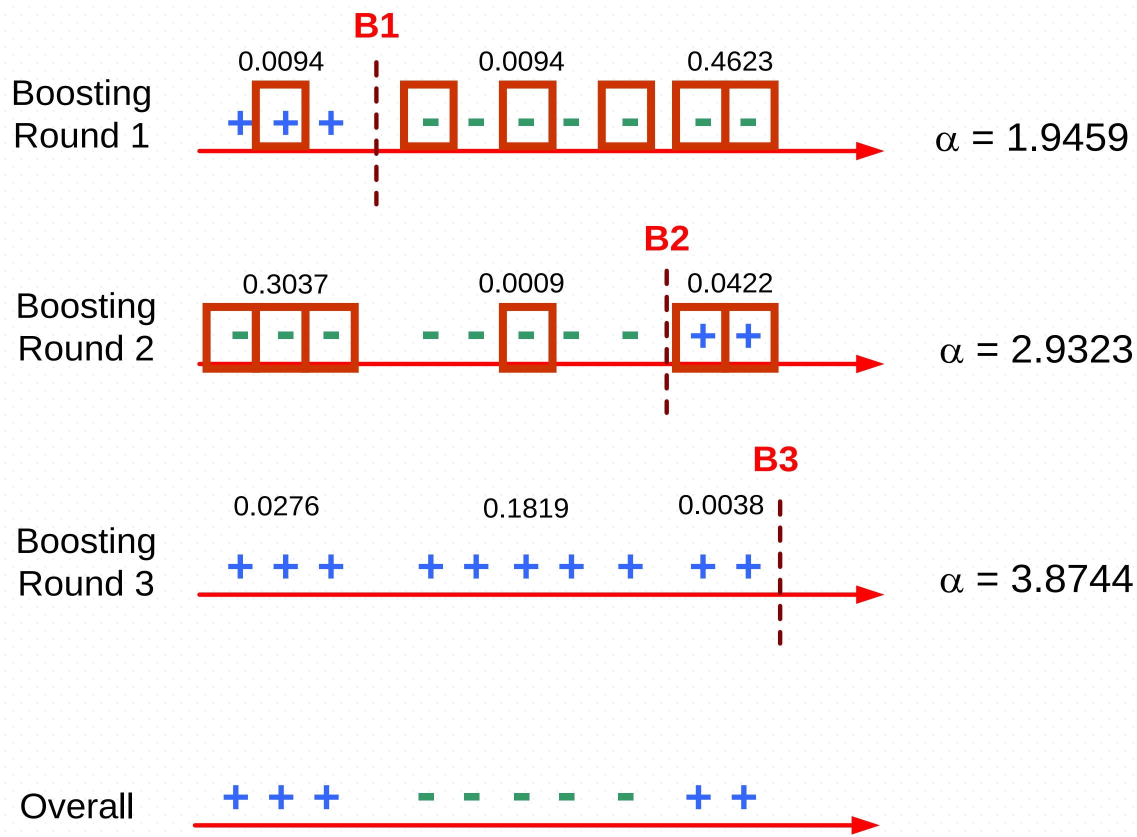
where Z_i is the normalization factor

$$C^*(x_{test}) = \arg\max_y \sum_{i=1}^{T} \alpha_i \delta\left(C_i(x_{test}) = y\right)$$

Illustrating AdaBoost



Illustrating AdaBoost



Bagging and Boosting Summary

- **Bagging:**
  - Resample data points
  - Weight of each classifier is the same
  - Only variance reduction
  - Robust to noise and outliers
- **Boosting:**
  - Reweight data points (modify data distribution)
  - Weight of classifier vary depending on accuracy
  - Reduces both bias and variance
  - Can hurt performance with noise and outliers