Ensemble Learning

Problem with Decision Tree

- The decision trees suffer from *high variance*.
- This means that if we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite different.
- In contrast, a procedure with *low variance* will yield similar results.
- One solution to deal with this problem is ensembled learning.

Ensembled Learning

- An ensemble is a composite model, combines a series of low performing classifiers with the aim of creating an improved classifier.
- Here, individual classifier vote and final prediction label returned that performs majority voting.
- Ensembles offer more accuracy than individual or base classifier.
- Ensemble methods can parallelize by allocating each base learner to different-different machines.
- Finally, you can say Ensemble learning methods are meta-algorithms that combine several machine learning methods into a single predictive model to increase performance.

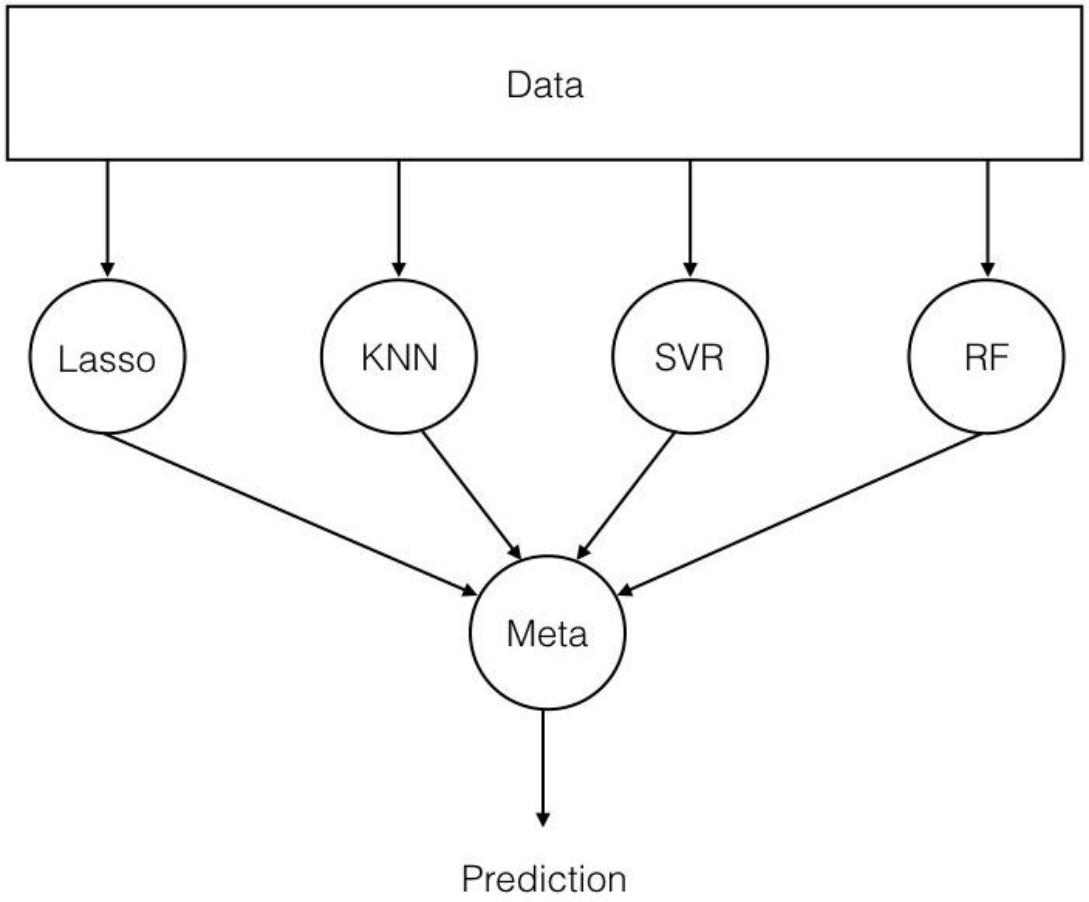
- Let's take a real example to build the intuition.
- Suppose, you want to invest in a company XYZ. You are not sure about its performance though.
- So, you look for advice on whether the stock price will increase by more than 8% per annum or not?
- You decide to approach various experts having diverse domain experience.

- Employee of Company XYZ:
  - In the past, he has been right 70% times.
- Financial Advisor of Company XYZ:
  - In the past, he has been right 75% times.
- Stock Market Trader:
  - In the past, he has been right 70% times.
- Employee of a competitor:
  - In the past, he has been right 60% times.
- Market Research team in the same segment:
  - In the past, he has been right 75% times.
- Social Media Expert:
  - In the past, he has been right 65% times.

- Given the broad spectrum of access you have, you can probably combine all the information and make an informed decision.
- In a scenario when all the 6 experts/teams verify that it’s a good decision(assuming all the predictions are independent of each other), you will get a combined accuracy rate of 1 - (30% . 25% . 30% . 40% . 25% . 35%) = 1 - 0.07875 = 99.92125%
- The assumption used here that all the predictions are completely independent is slightly extreme as they are expected to be correlated. However, you can see how we can be so sure by combining various forecasts together.

- An ensemble is the art of combining a diverse set of learners (individual models) together to improvise on the stability and predictive power of the model.
- In our example, the way we combine all the predictions collectively will be termed as Ensemble learning.

Basic Ensemble Structure



- Use multiple learning algorithms (classifiers)
- Combine the decisions
- Can be more accurate than the individual classifiers
- Generate a group of base-learners
- Different learners use different
    - Algorithms
    - Hyperparameters
    - Training sets

Why ensembles ?

- There are two main reasons to use an ensemble over a single model, and they are related; they are:
  - – Performance: An ensemble can make better predictions and achieve better performance than any single contributing model.
  - – Robustness: An ensemble reduces the spread or dispersion of the predictions and model performance.

Ensemble Creation Approaches

- Unweighted Voting (e.g. Bagging)
- Weighted voting – based on accuracy (e.g. Boosting), Expertise, etc.

Bootstrap

- The bootstrap is a powerful statistical method for estimating a quantity from a data sample. This is easiest to understand if the quantity is a descriptive statistic such as a mean or a standard deviation.
- Let’s assume we have a sample of 100 values (x) and we’d like to get an estimate of the mean of the sample.
- We can calculate the mean directly from the sample as:

$$\text{mean}(x) = 1/100 * \text{sum}(x)$$

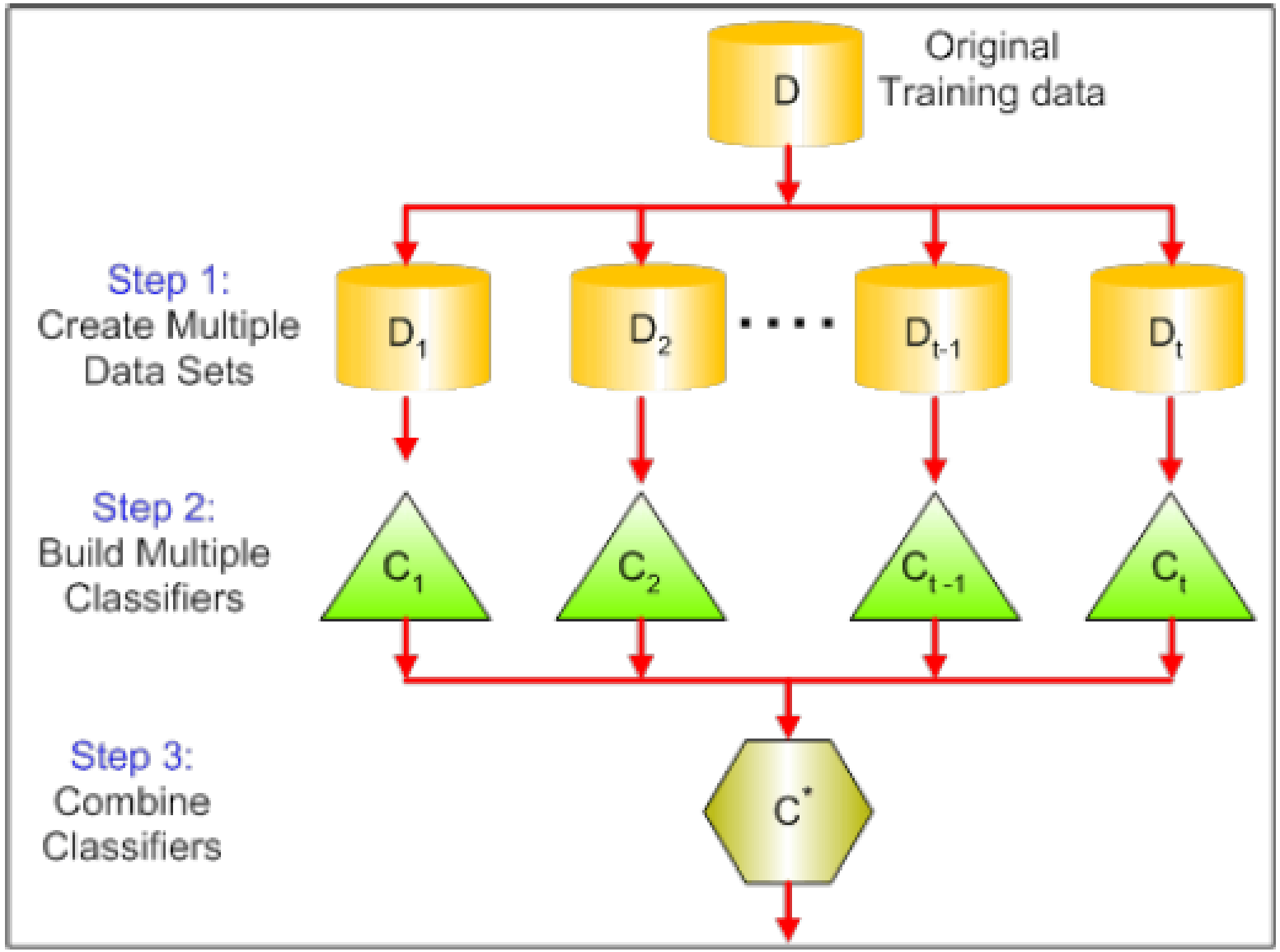
- We know that our sample is small and that our mean has error in it. We can improve the estimate of our mean using the bootstrap procedure:
- Create many (e.g. 1000) random sub-samples of our dataset with replacement (meaning we can select the same value multiple times).
- Calculate the mean of each sub-sample.
- Calculate the average of all of our collected means and use that as our estimated mean for the data.

- For example, let’s say we used 3 resamples and got the mean values 2.3, 4.5 and 3.3.
- Taking the average of these we could take the estimated mean of the data to be 3.367.
- This process can be used to estimate other quantities like the standard deviation and even quantities used in machine learning algorithms, like learned coefficients.

Bagging

- Bagging stands for bootstrap aggregation.
- It combines multiple learners in a way to reduce the variance of estimates.
- For example, random forest trains M Decision Tree, you can train M different trees on different random subsets of the data and perform voting for final prediction.
- Example:
  - Random Forest
  - Extra Trees.

Bagging


D
Original
Training data
Step 1:
Create Multiple
Data Sets
D1
D2
Dt-1
Dt
Step 2:
Build Multiple
Classifiers
C1
C2
Ct-1
Ct
Step 3:
Combine
Classifiers
C*

- Bootstrap sample of N instances is obtained by drawing N examples at random, with replacement.
- On average each bootstrap sample has 63% of unique instances

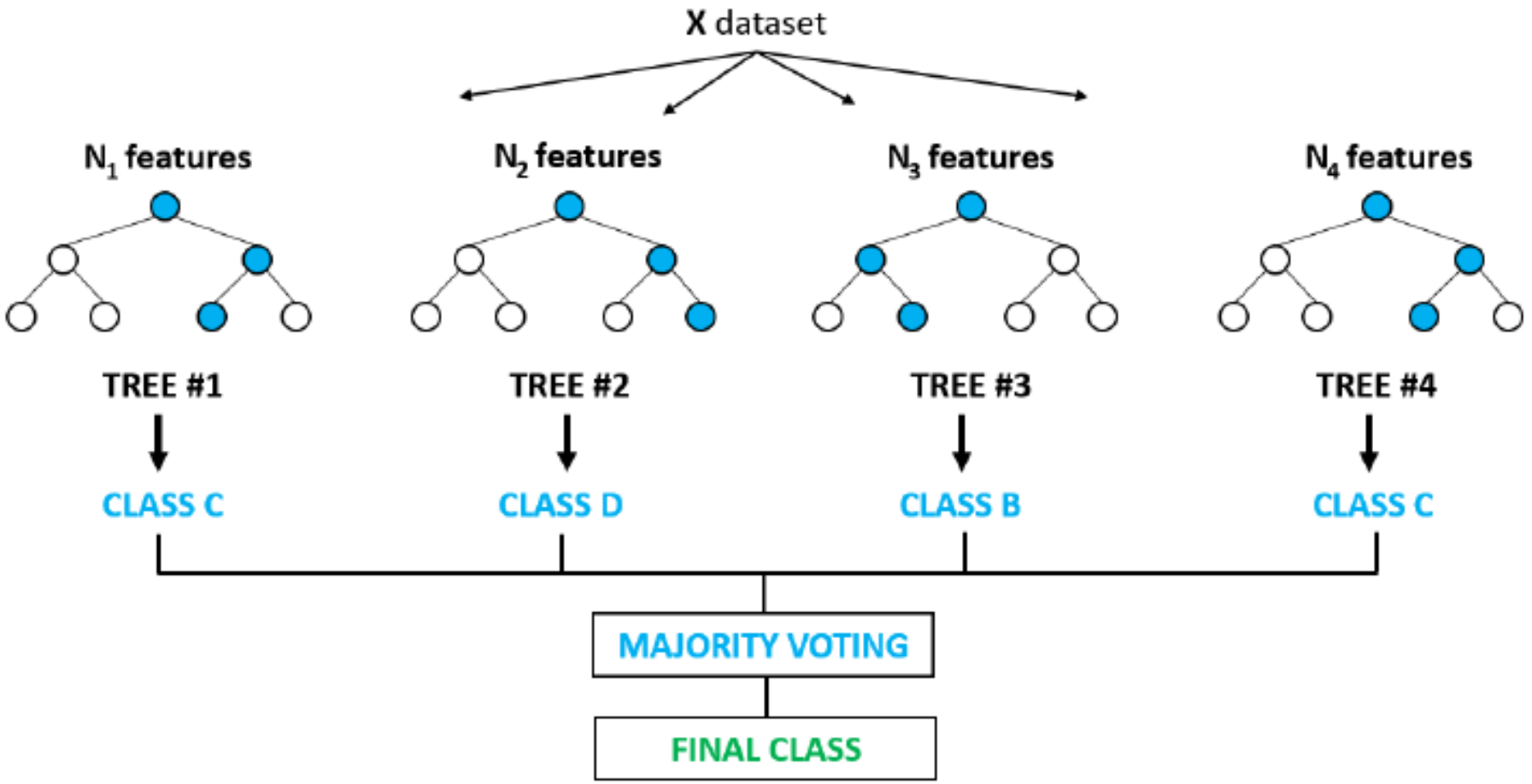
Random Forest

- Random forest is a type of supervised machine learning algorithm based on Ensemble learning .
- Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model.
- The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest".
- The random forest algorithm can be used for both regression and classification tasks.

How it works ?

- Step 1: Pick N random records from the dataset.
- Step 2:Build a decision tree based on these N records.
  - *Random forests* provide an improvement over bagged trees by way of a random small tweak that *decorrelates* the trees.
  - when building these decision trees, 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.
  - The split is allowed to use only one of those *m* predictors.
  - typically we choose $m \approx \sqrt{p}$
- Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
- For a new record, each tree in the forest predicts the category to which the new record belongs.
- Finally, the new record is assigned to the category that wins the majority vote.

Majority Voting



Boosting

- Boosting algorithms depend on a set of the low accurate classifier to create a highly accurate classifier.
- Low accuracy classifier (or weak classifier) offers the accuracy better than the flipping of a coin.
- This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.
- Highly accurate classifier(or strong classifier) offer error rate close to 0. Boosting algorithm can track the model who failed the accurate prediction.
- Boosting algorithms are less affected by the overfitting problem.

Boosting

- An iterative procedure. Adaptively change distribution of training data.
  - Initially, all N records are assigned equal weights
  - Weights change at the end of boosting round
- On each iteration t:
  - Weight each training example by how incorrectly it was classified
  - Learn a hypothesis: h_t
  - A strength for this hypothesis: α_t
- Final classifier:
  - A linear combination of the votes of the different classifiers weighted by their strength
- "weak" learners
  - P(correct) > 50%, but not necessarily much better

Adaboost

- Boosting can turn a weak algorithm into a strong learner.
- Input: S=$\{(x_1, y_1), \ldots, (x_m, y_m)\}$
- $D_t(i)$: weight of i th training example
- Weak learner A
- For $t = 1,2, \ldots, T$
  - Construct D_t on $\{x_1, x_2 \ldots\}$
  - Run A on D_t producing $h_t: X \rightarrow \{-1,1\}$

    ϵ_t =error of h_t over D_t

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak classifier $h_t: X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_t + 1(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Where Z_t is a normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) exp\,(-\alpha_t y_i h_t(x_i))$$

Output the final classifier:

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

Choose α_t to minimize training error

$$\alpha_t = \frac{1}{2} In \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$$

where

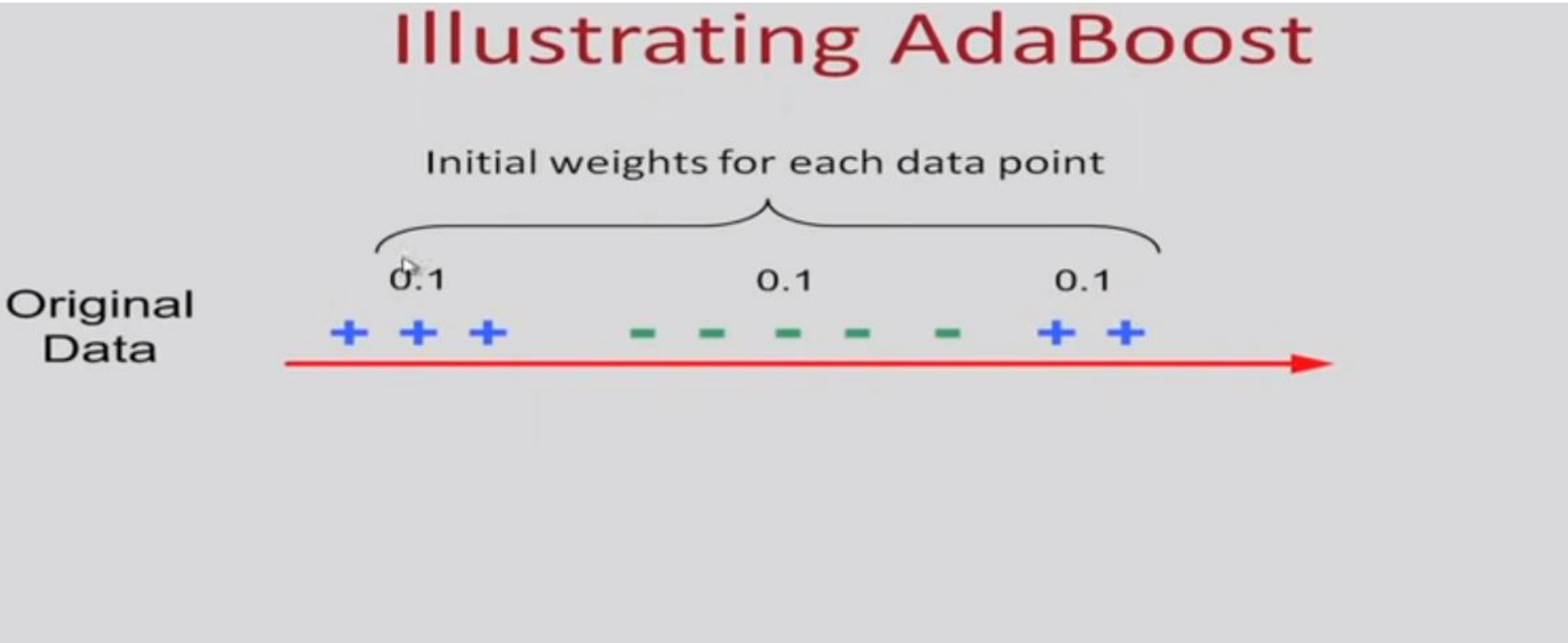
$$\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(h_t(x_i) \neq y_i)$$

Strong weak classifiers

- If each classifiers is (at least slightly) better than random
$$\epsilon_t < 0.5$$
- Ican be shown that AdaBoost will achieve zero training error (expotentially fast):

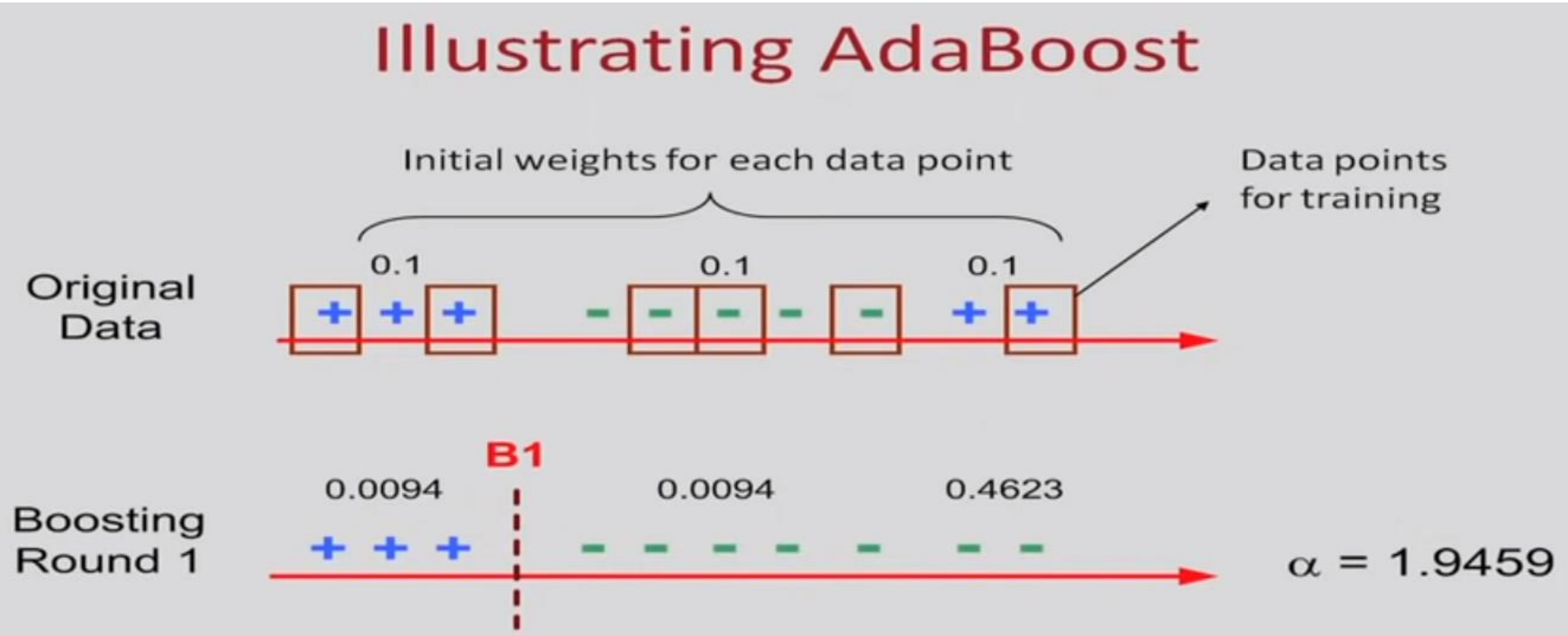
$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i) \neq y_i) \leq \prod_t Z_t \leq exp\left(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\right)$$

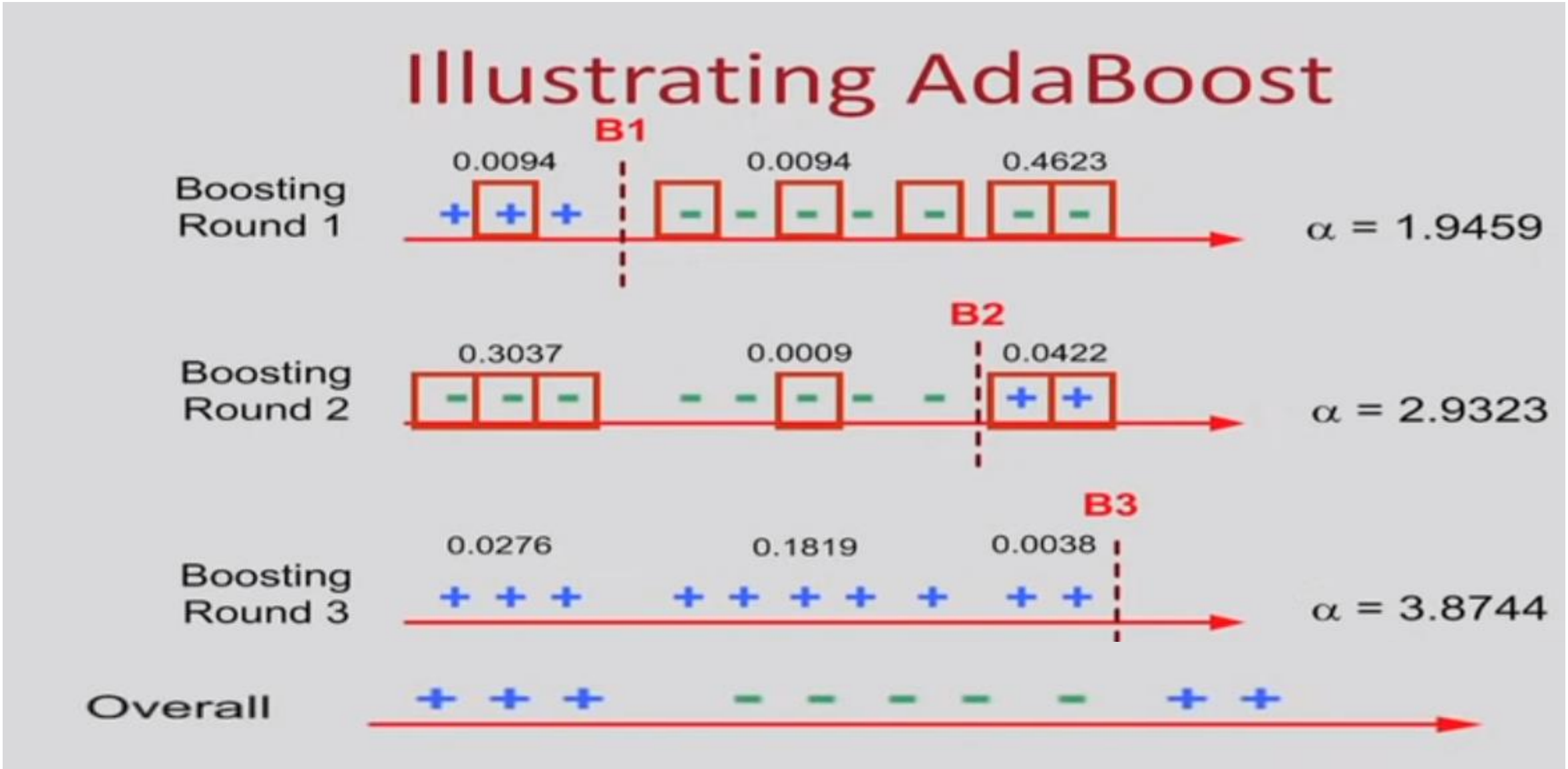
Illustrating AdaBoost



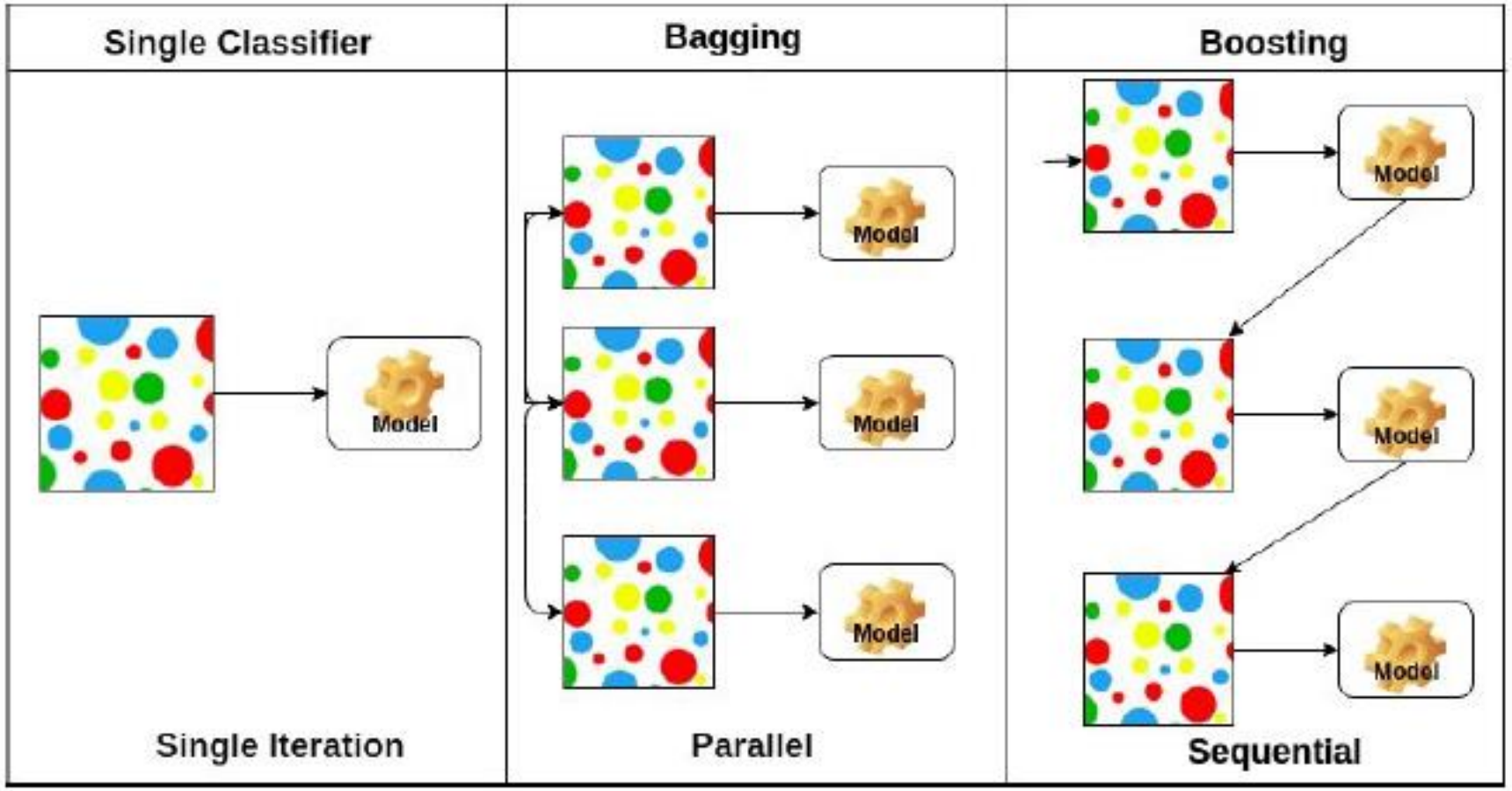
Illustrating AdaBoost

Illustrating AdaBoost




Illustrating AdaBoost
B1
0.0094
0.0094
0.4623
Boosting Round 1
α = 1.9459
B2
0.3037
0.0009
0.0422
Boosting Round 2
α = 2.9323
B3
0.0276
0.1819
0.0038
Boosting Round 3
α = 3.8744
Overall

Comparison


Single Classifier
Bagging
Boosting
Model
Model
Model
Model
Model
Model
Model
Model
Single Iteration
Parallel
Sequential