Boosting

Boosting

- We’ve have seeked to improve upon single Decision Trees with Random Forest models.
- Let’s now explore how to improve on the single decision tree, known as **boosting**.

- Boosting and Meta-Learning
- AdaBoost (Adaptive Boosting) Theory
- Example of AdaBoost
- Gradient Boosting Theory
- Example of Gradient Boosting

Boosting is not actually a machine learning algorithm, it is methodology *applied* to an existing machine learning algorithm, most commonly applied to the decision tree.

Let’s explore this idea of a meta-learning algorithm by reviewing a simple application and formula.

Main formula for boosting:

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

$$F_T(x) = \sum_{t=1}^{T} \boxed{f_t(x)} \qquad \boxed{f_t(x) = \alpha_t h(x)}$$

$$F_T(x) = \sum_{t=1}^{T} \boxed{f_t(x)} \qquad \boxed{f_t(x) = \alpha_t h(x)}$$

A combination of **estimators** with an applied **coefficient** could act as an effective **ensemble estimator**.

$$F_T(x) = \sum_{t=1}^{T} \boxed{f_t(x)} \qquad \boxed{f_t(x) = \alpha_t h(x)}$$

Note **h(x)** can in theory be **any** machine learning algorithm (estimator/learner).

$$F_T(x) = \sum_{t=1}^{T} \boxed{f_t(x)} \qquad \boxed{f_t(x) = \alpha_t h(x)}$$

Can an ensemble of **weak learners** (very simple models) be a **strong learner** when combined?

$$F_T(x) = \sum_{t=1}^{T} \boxed{f_t(x)} \qquad \boxed{f_t(x) = \alpha_t h(x)}$$

For decision tree models, we can use simple trees in place of h(x) and combine them with the coefficients on each model.

Let’s focus on AdaBoost and understand how to combine weak learners to create a strong estimator

We will also explore why Decision Trees are so well suited for boosting.

AdaBoost

Intuition and Theory

AdaBoost (Adaptive Boosting) works by using an ensemble of **weak learners** and then combining them through the use of a weighted sum.

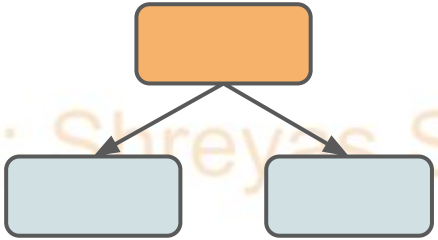
It uses previously created **weak learners** in order to adjust misclassified instances for the next created **weak learner.**

weak learner

A model that is too simple to perform well on its own.

Weak learner

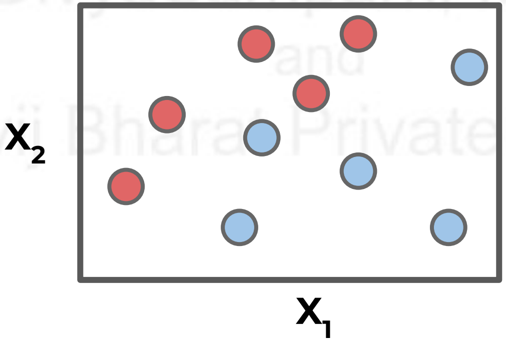
- A model that is too simple to perform well on its own.
- The weakest decision tree possible would be a **stump**, one node and two leaves!



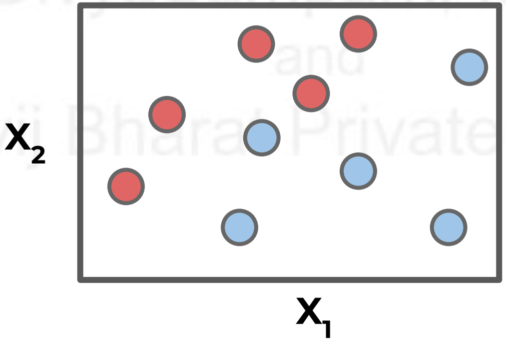
Unlike a single decision tree which fits to all the data at once (*fitting the data hard*), AdaBoost aggregates multiple weak learners, allowing the overall **ensemble** model to *learn slowly* from the features.

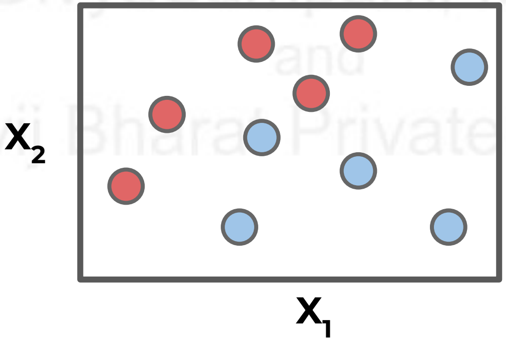
Let's first understand how this works from a data perspective!

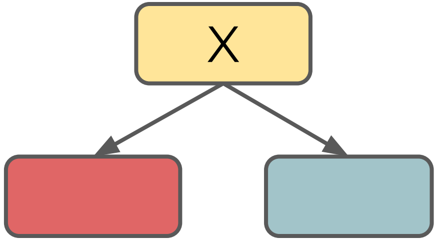
Imagine a classification task:

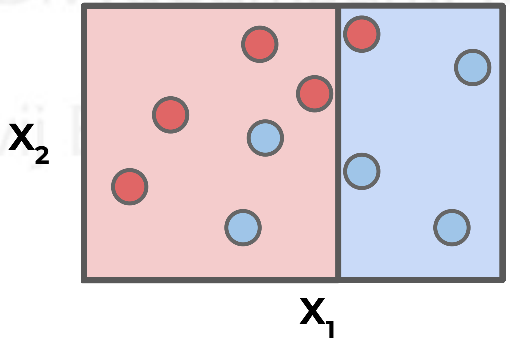


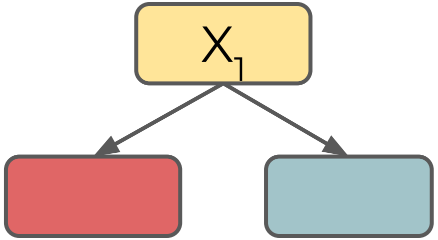
What would a stump classification look like?

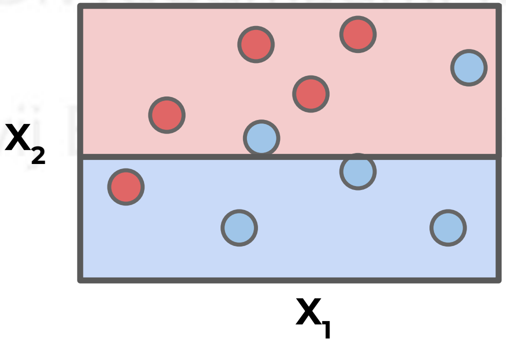


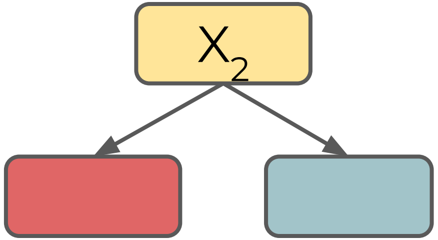

X_2
X_1


X

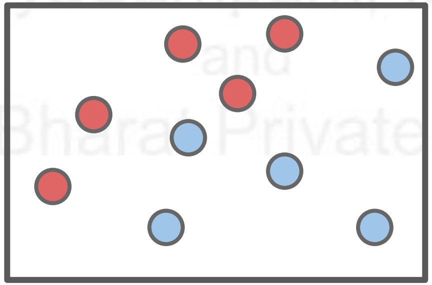

X_2
X_1


X_1

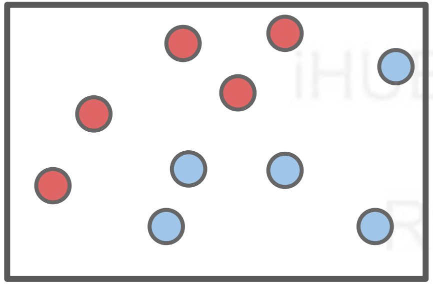

X2
X1


X2

How can we combine stumps? How to improve performance with an ensemble?

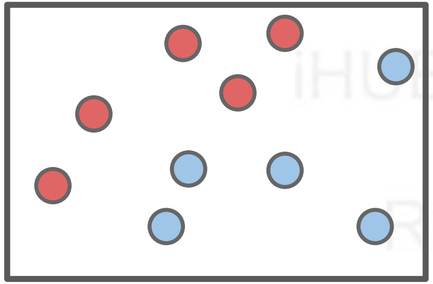


AdaBoost Process:



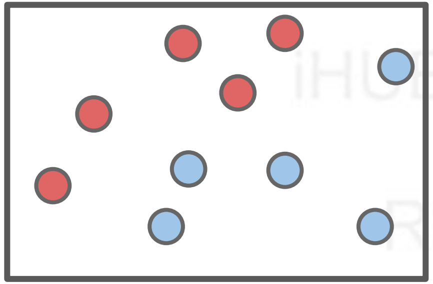
- Main Formulas
- Algorithmic Steps
- Visual Walkthrough of Algorithm

Main Formulas



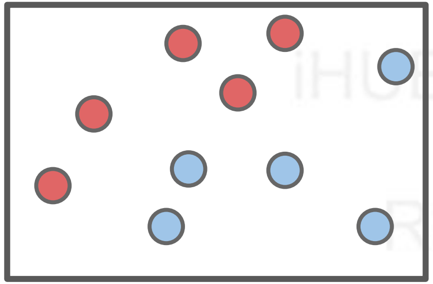
$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

AdaBoost Process:



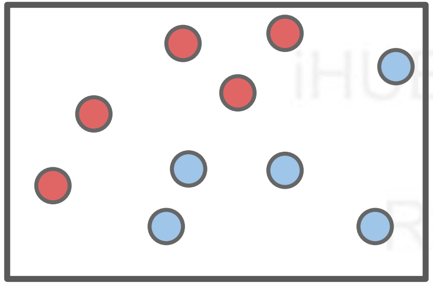
$$F_T(x) = \sum_{t=1}^{T} \boxed{f_t(x)}$$

$$\boxed{f_t(x) = \alpha_t h(x)}$$



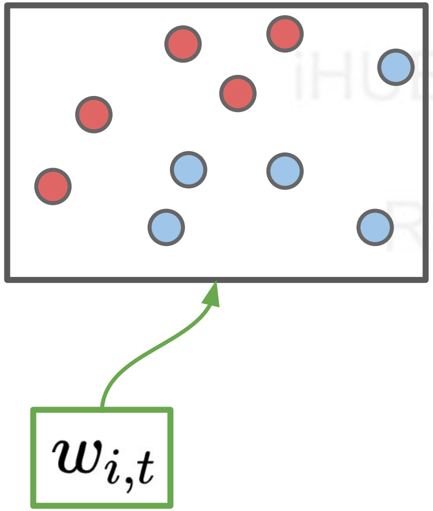
$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

$$f_t(x) = \boxed{\alpha_t} h(x)$$



$$F_T(x) = \sum_{t=1}^{T} f_t(x) \qquad f_t(x) = \boxed{\alpha_t} h(x)$$

$$E_t = \sum_i E[F_{t-1}(x_i) + \boxed{\alpha_t} h(x_i)]$$

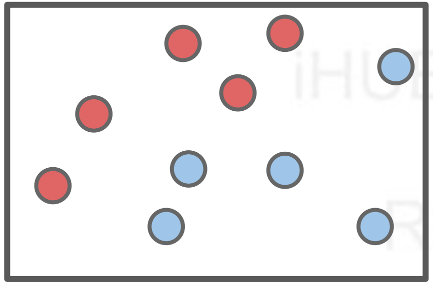


$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

$$f_t(x) = \alpha_t h(x)$$

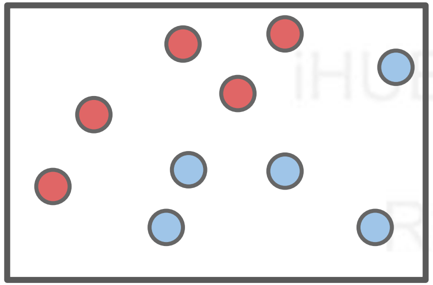
$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_t h(x_i)]$$

Algorithm Steps



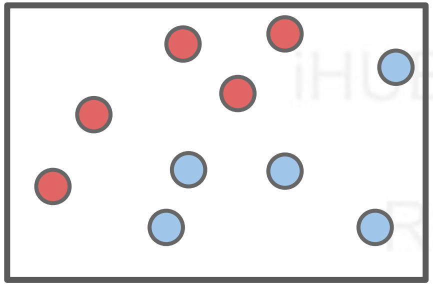
With:

- Samples $x_1 \dots x_n$
- Desired outputs $y_1 \dots y_n, y \in \{-1, 1\}$
- Initial weights $w_{1,1} \dots w_{n,1}$ set to $\frac{1}{n}$
- Error function $E(f(x), y, i) = e^{-y_i f(x_i)}$
- Weak learners $h \colon x \rightarrow \{-1, 1\}$



For t in $1 \dots T$:

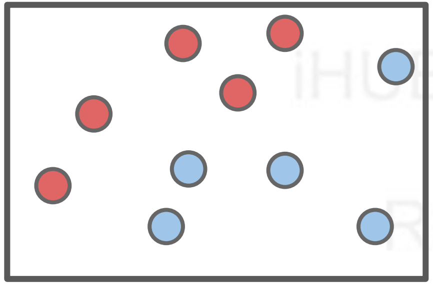
- Choose $h_t(x)$:
  - Find weak learner $h_t(x)$ that minimizes ϵ_t, the weighted sum error for misclassified points $\epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^{n} w_{i,t}$
  - Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$



For t in $1 \dots T$:

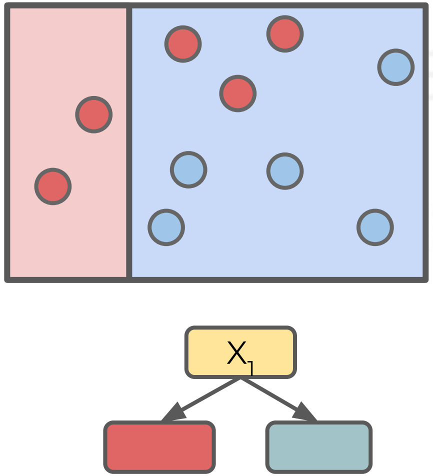
- Add to ensemble:
  - $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$
- Update weights:
  - $w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)}$ for i in $1 \dots n$
  - Renormalize $w_{i,t+1}$ such that $\sum_i w_{i,t+1} = 1$

Visual Walkthrough

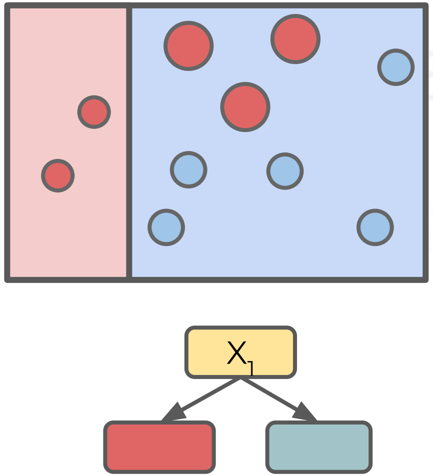


With:

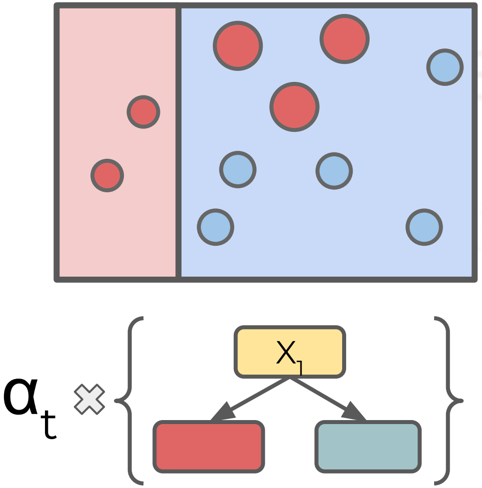
- Samples $x_1 \ldots x_n$
- Desired outputs $y_1 \ldots y_n, y \in \{-1, 1\}$
- Initial weights $w_{1,1} \ldots w_{n,1}$ set to $\frac{1}{n}$
- Error function $E(f(x), y, i) = e^{-y_i f(x_i)}$
- Weak learners $h \colon x \to \{-1, 1\}$



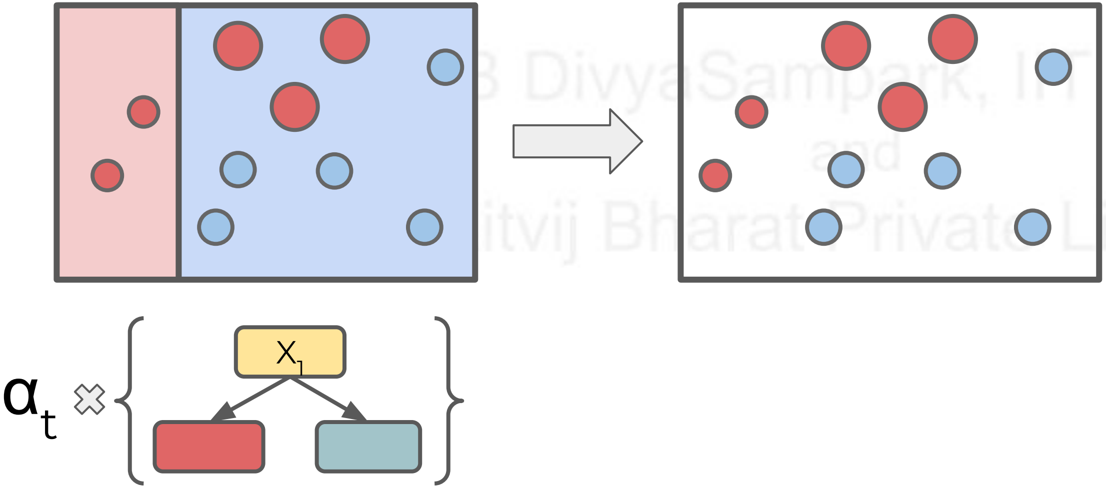
- Choose $h_t(x)$:
  - Find weak learner $h_t(x)$ that minimizes ϵ_t, the weighted sum error for misclassified points $\epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^{n} w_{i,t}$

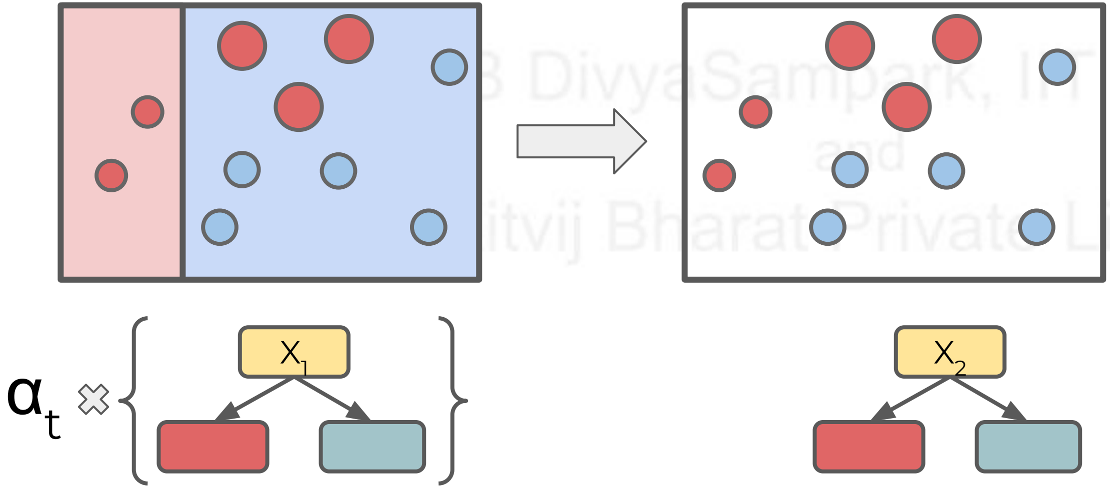


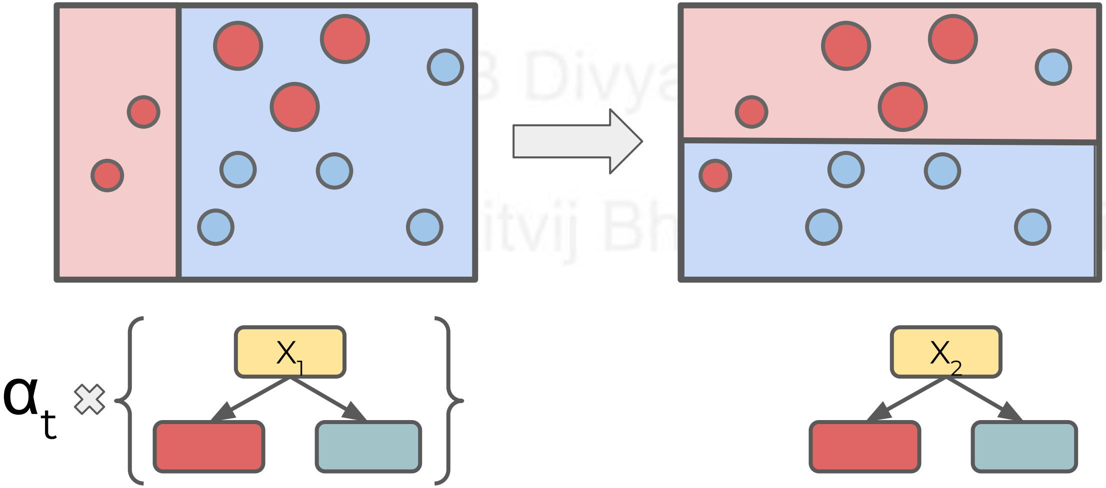
- Choose $h_t(x)$:
  - Find weak learner $h_t(x)$ that minimizes ϵ_t, the weighted sum error for misclassified points $\epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^{n} w_{i,t}$

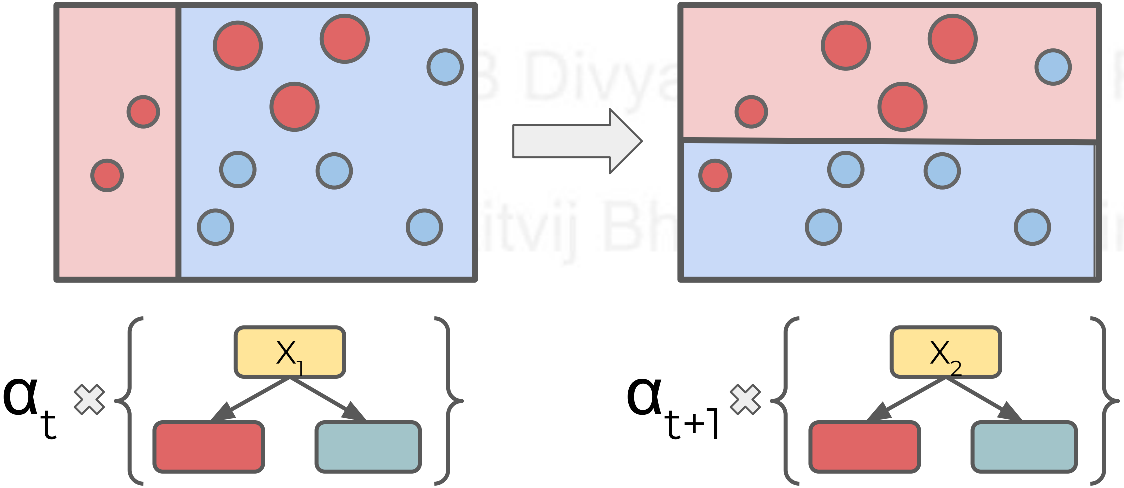


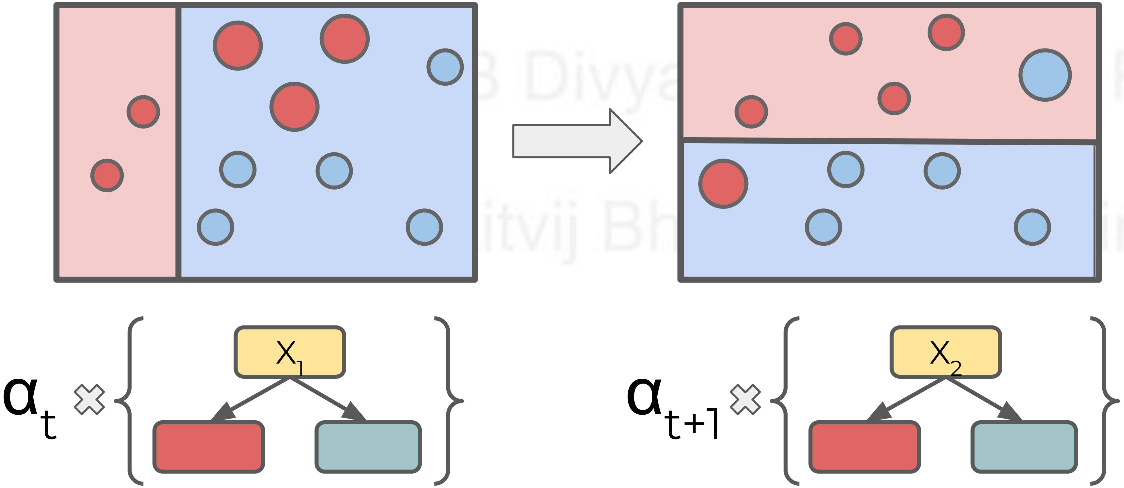
- Choose $h_t(x)$:
  - Find weak learner $h_t(x)$ that minimizes ϵ_t, the weighted sum error for misclassified points $\epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^{n} w_{i,t}$
- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

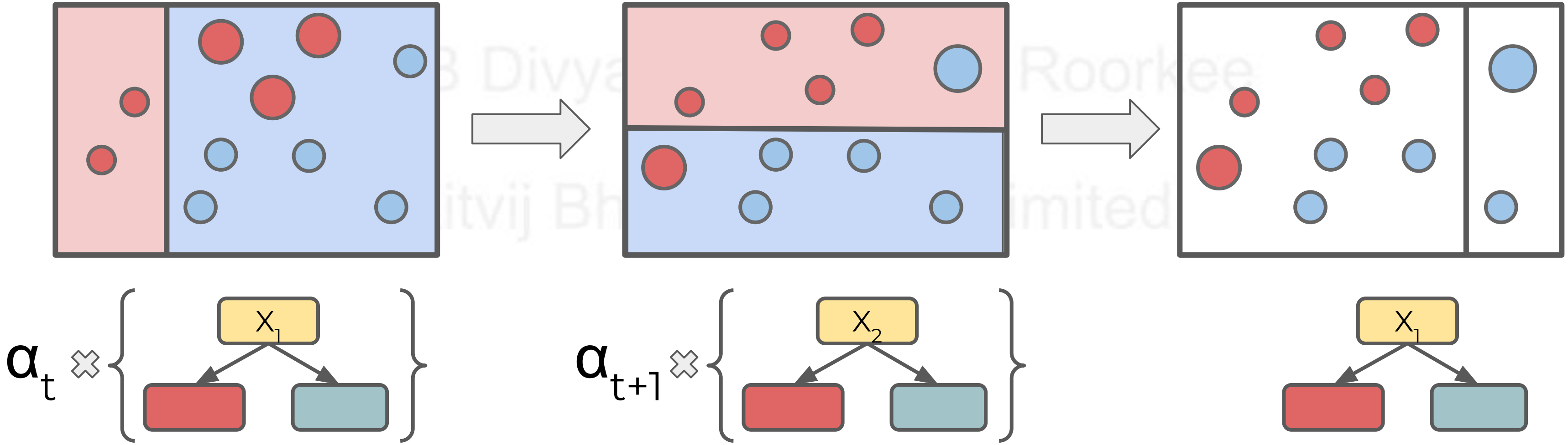

X_1
α_t

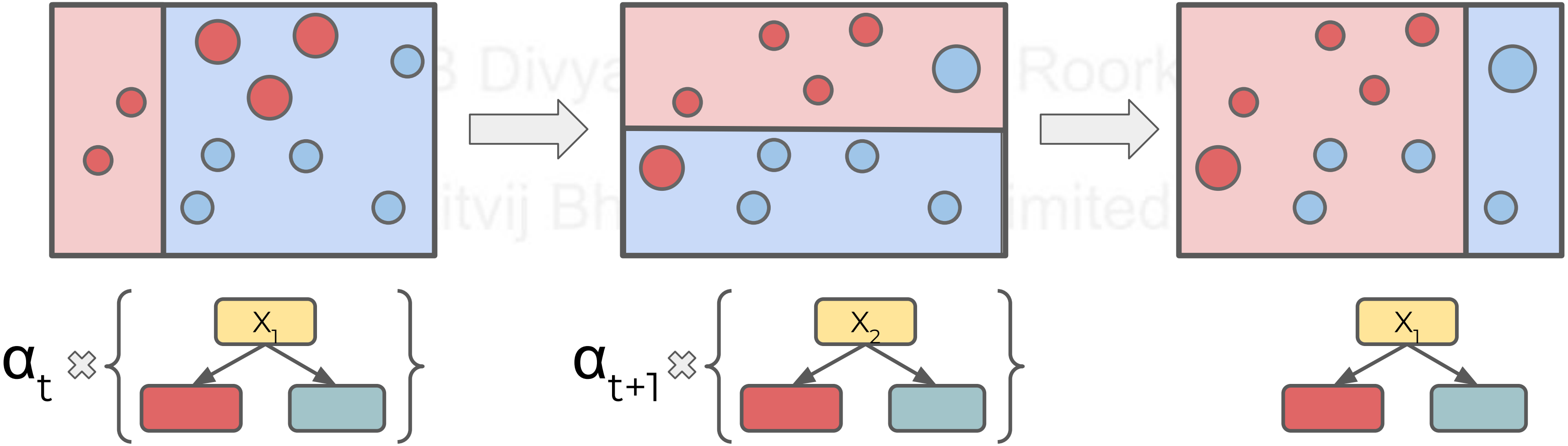

α_t
X_1
X_2

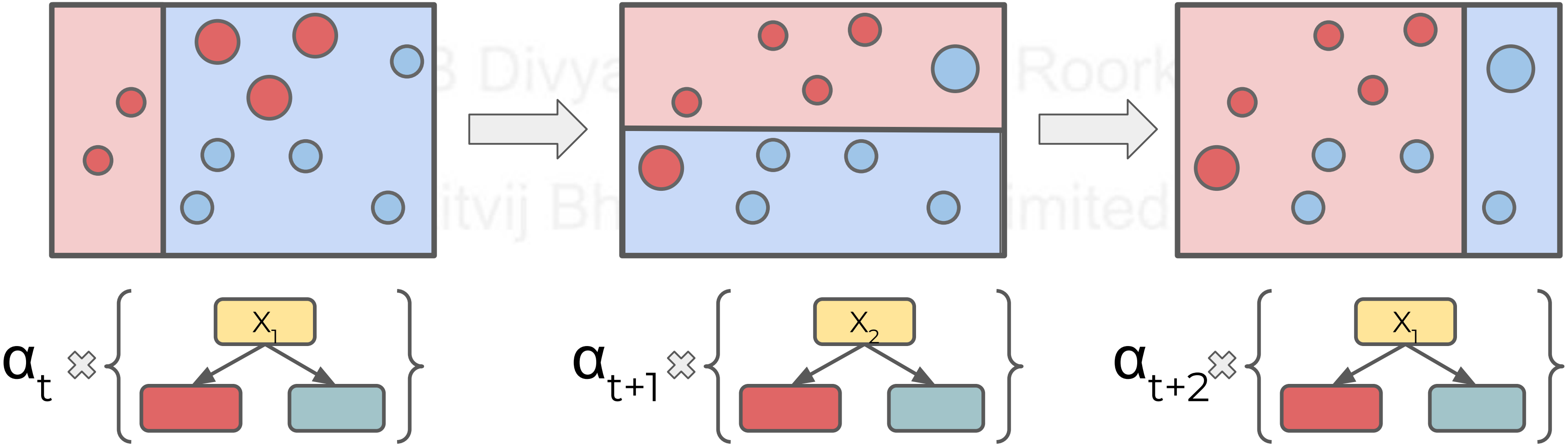

α_t
X_1
X_2

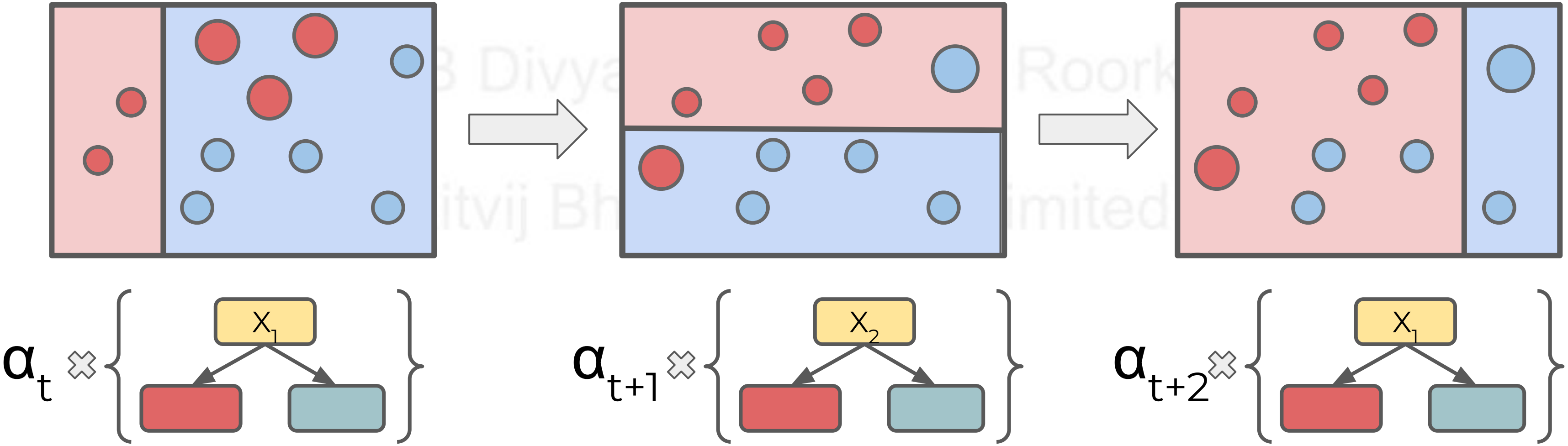

αt
X1
αt+1
X2


X1
X2
αt
αt+1

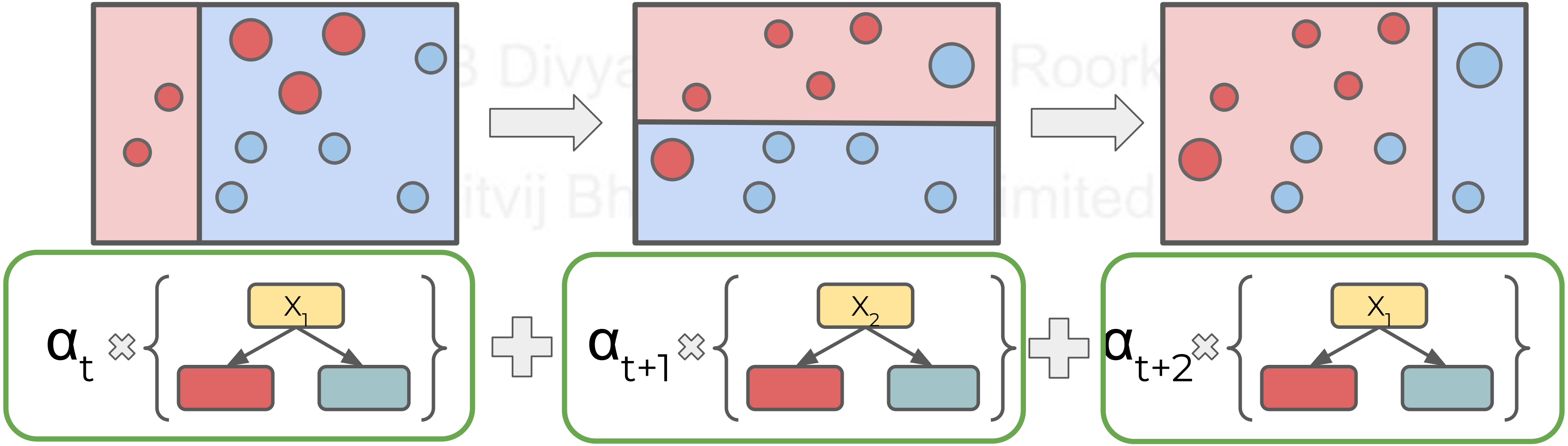

α_t
X_1
α_{t+1}
X_2
X_1

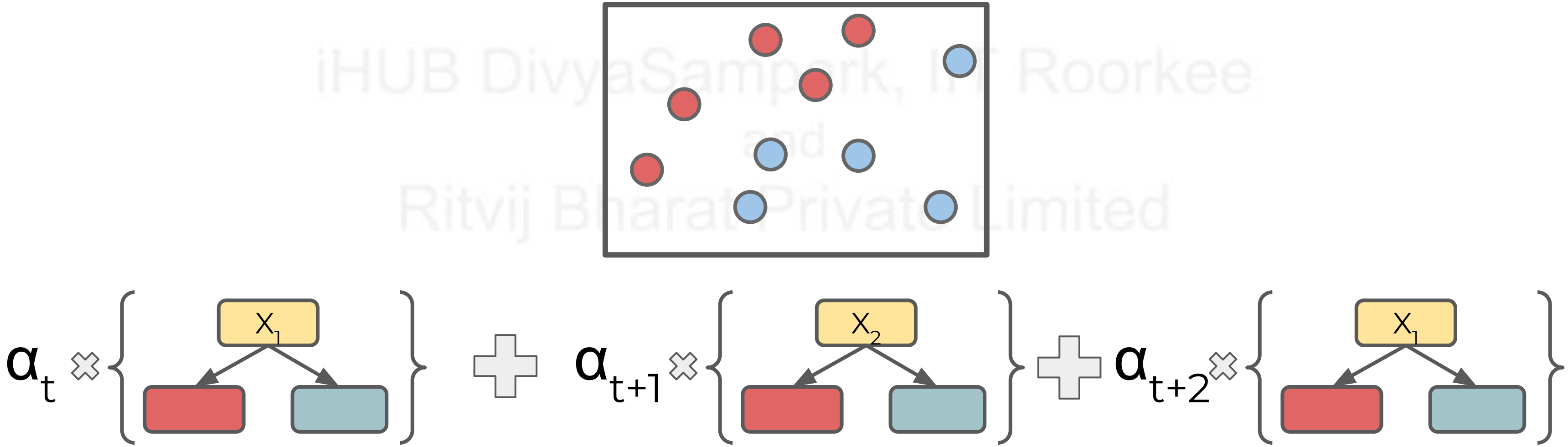

α_t
X_1
α_{t+1}
X_2
X_1

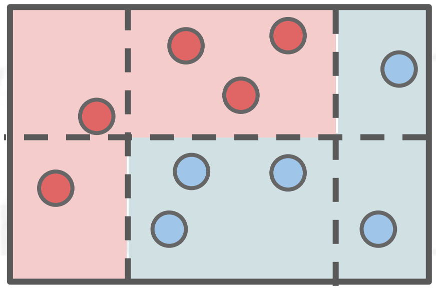

α_t
X_1
α_{t+1}
X_2
α_{t+2}
X_1

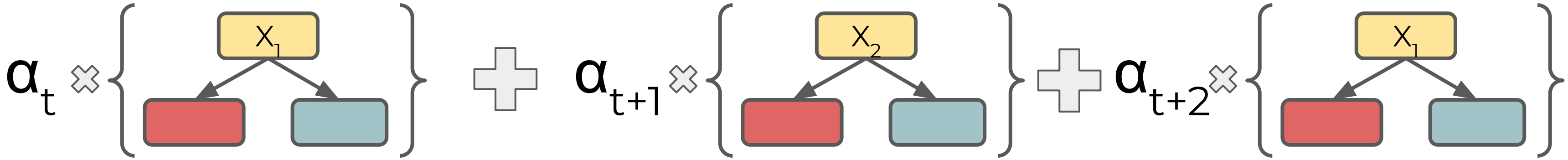


$$F_T(x) = \sum_{t=1}^{T} f_t(x) \qquad f_t(x) = \alpha_t h(x)$$


α_t
X_1
α_{t+1}
X_2
α_{t+2}
X_1


α_t
X_1
α_{t+1}
X_2
α_{t+2}
X_1




α_t
X_1
α_{t+1}
X_2
α_{t+2}
X_1

AdaBoost uses an ensemble of **weak learners** that learn slowly in series.

Certain weak learners have more weightage in in the final output than others due to the multiplied alpha parameter.

Each subsequent **t** weak learner is built using a reweighted data set from the **t-1** weak learner.

Intuition of Adaptive Boosting:

- Each stump essentially represents the strength of a feature to predict.
- Building these stumps in series and adding in the alpha parameter allows us to intelligently combine the importance of each feature together.

Unlike Random Forest, it is possible to overfit with AdaBoost, however it takes many trees to do this.

Usually error has already stabilized way before enough trees are added to cause overfitting.

Let’s put to practice!