



Boosting



Boosting

- We've learned about single Decision Trees and have sought to improve upon them with Random Forest models.
- Let's now explore another methodology of seeking to improve on the single decision tree, known as **boosting**.



Boosting

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



Boosting

- Related Reading:
 - ISLR: Section 8.2.3
- Relevant Wikipedia Articles:
 - [wikipedia.org/wiki/Boosting_\(machine_learning\)](https://wikipedia.org/wiki/Boosting_(machine_learning))
 - wikipedia.org/wiki/AdaBoost



Let's get started!



Boosting

Motivation and History



Boosting

- The concept of **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.



Boosting

- Main formula for boosting:

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



Boosting

- Main formula for boosting:

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



Boosting

- Main formula for boosting:

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

- Implies that a combination of **estimators** with an applied **coefficient** could act as an effective **ensemble estimator**.



Boosting

- Main formula for boosting:

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

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



Boosting

- Main formula for boosting:

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

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



Boosting

- Main formula for boosting:

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

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



Boosting

- The idea of **gradient boosting** originated from Leo Breiman when he observed that boosting can be interpreted as an optimization algorithm on a cost function in publications in the late 1990s.
- Later on Jerome H. Friedman and many others developed more explicit formulations of gradient boosting.



Boosting

- Also in the late 1990s Yoav Freund and Robert Schapire developed the AdaBoost (Adaptive Boosting) algorithm, which also combines weak learners in an ensemble to create a stronger model.



Boosting

- Let's continue by focusing first on AdaBoost and building an understanding of 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



Boosting

- AdaBoost (Adaptive Boosting) works by using an ensemble of **weak learners** and then combining them through the use of a weighted sum.
- Adaboost adapts by using previously created **weak learners** in order to adjust misclassified instances for the next created **weak learner**.



Boosting

- What is a **weak learner**?



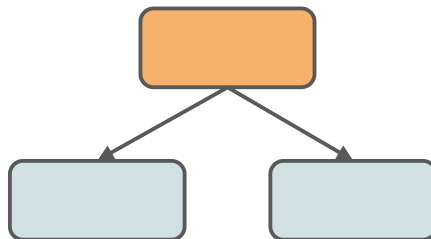
Boosting

- What is a **weak learner**?
 - A weak model is a model that is too simple to perform well on its own.



Boosting

- What is a **weak learner**?
 - A weak model is 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!





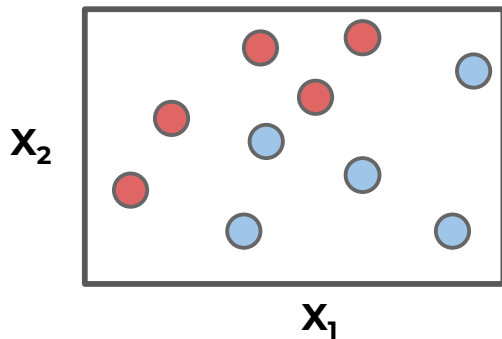
Boosting

- 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!



Boosting

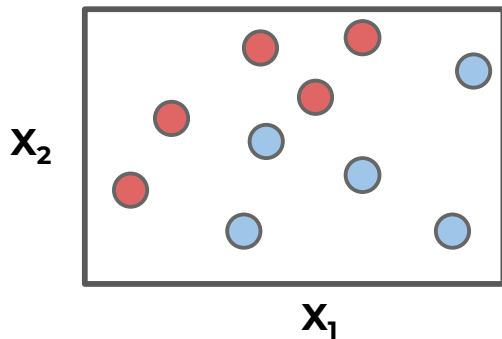
- Imagine a classification task:





Boosting

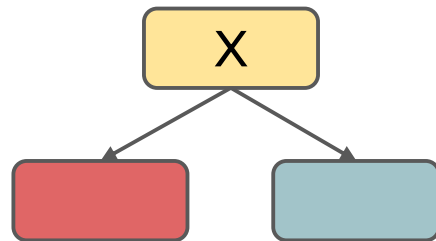
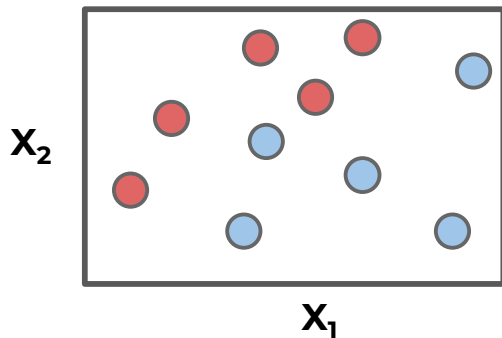
- What would a stump classification look like?





Boosting

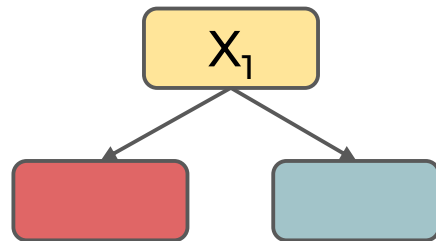
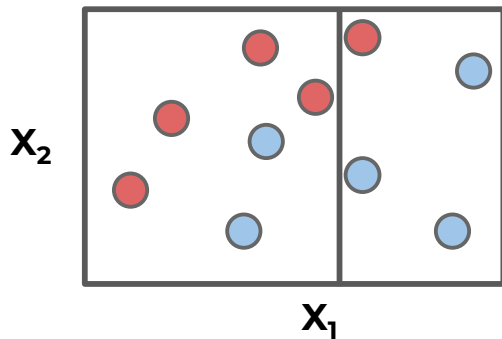
- What would a stump classification look like?





Boosting

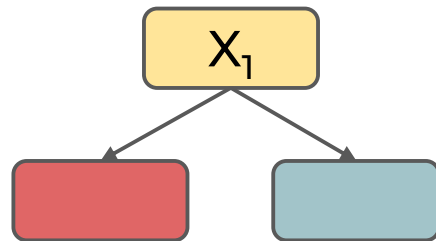
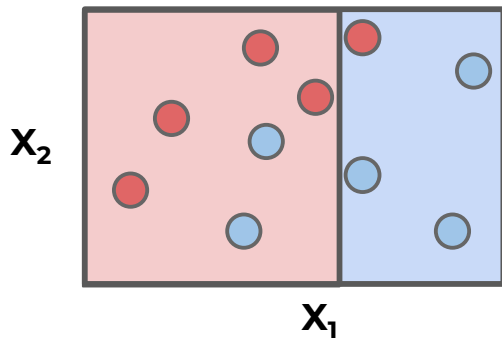
- What would a stump classification look like?





Boosting

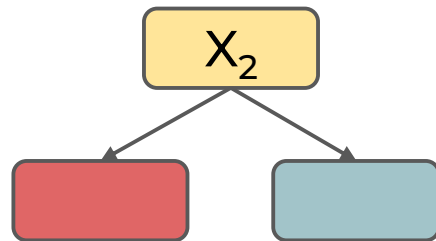
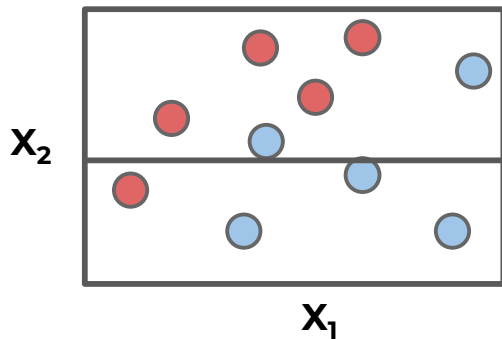
- What would a stump classification look like?





Boosting

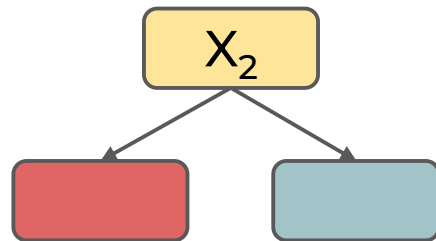
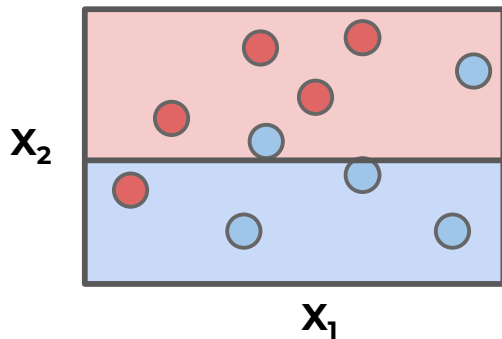
- What would a stump classification look like?





Boosting

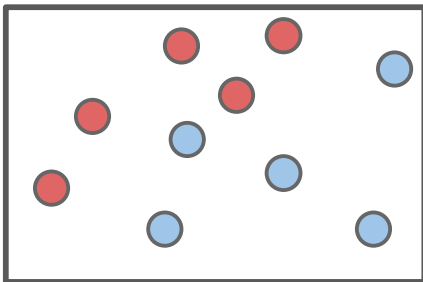
- What would a stump classification look like?





Boosting

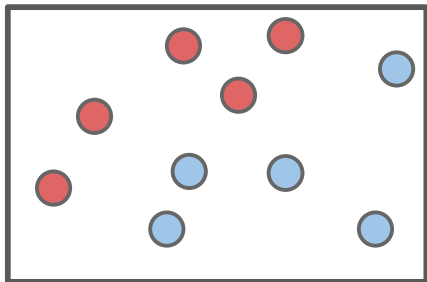
- How can we combine stumps? How to improve performance with an ensemble?





Boosting

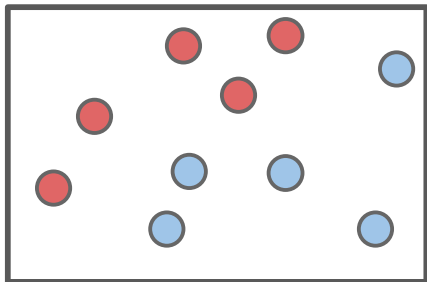
- AdaBoost Process:





Boosting

- AdaBoost Process:

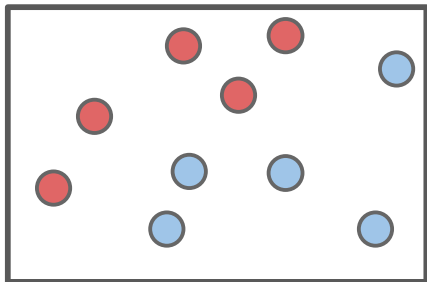


- Main Formulas
- Algorithmic Steps
- Visual Walkthrough of Algorithm



Boosting

- AdaBoost Process: Main Formulas

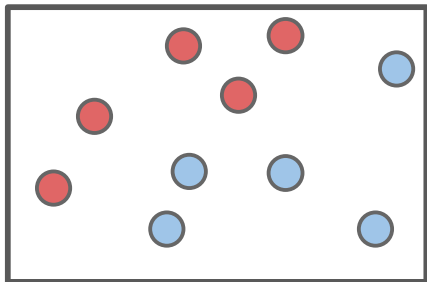


$$F_T(x) = \sum_{t=1}^T f_t(x)$$



Boosting

- AdaBoost Process:



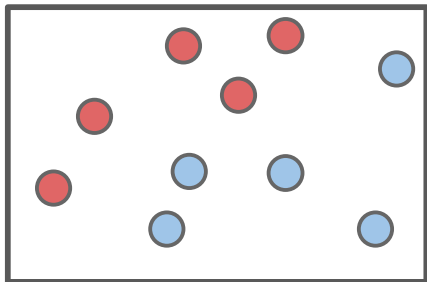
$$F_T(x) = \sum_{t=1}^T f_t(x)$$

A curved arrow points from the boxed $f_t(x)$ term in the summation to the boxed equation $f_t(x) = \alpha_t h(x)$.



Boosting

- AdaBoost Process:



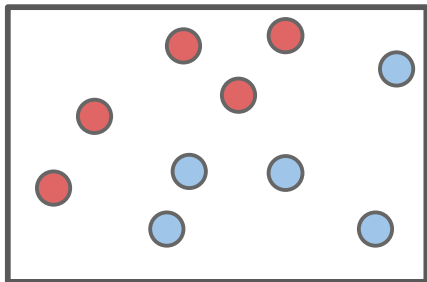
$$F_T(x) = \sum_{t=1}^T f_t(x)$$

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



Boosting

- AdaBoost Process:



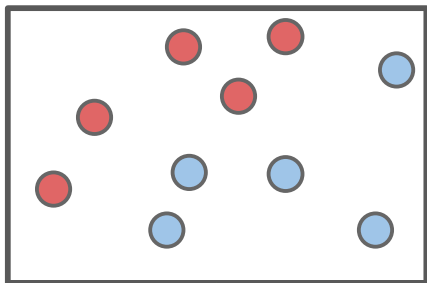
$$F_T(x) = \sum_{t=1}^T f_t(x) \quad f_t(x) = \boxed{\alpha_t} h(x)$$

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



Boosting

- AdaBoost Process:



$w_{i,t}$

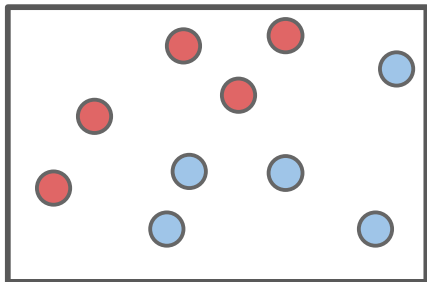
$$F_T(x) = \sum_{t=1}^T f_t(x) \quad f_t(x) = \alpha_t h(x)$$

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



Boosting

- AdaBoost Process: 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 $\frac{1}{n}$

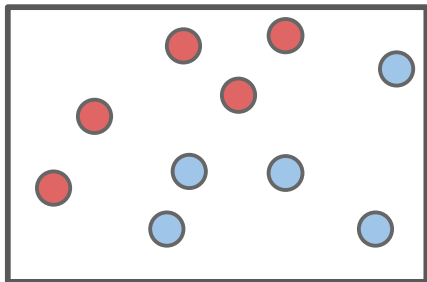
- Error function $E(f(x), y, i) = e^{-y_i f(x_i)}$

- Weak learners $h: x \rightarrow \{-1, 1\}$



Boosting

- AdaBoost Process:



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

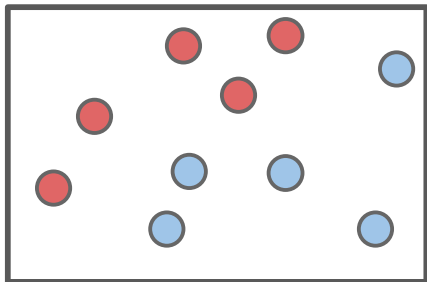
points
$$\epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^n w_{i,t}$$

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



Boosting

- AdaBoost Process:



For t in $1 \dots T$:

- Add to ensemble:

$$F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$$

- Update weights:

$$w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)} \text{ for } i \text{ in } 1 \dots n$$

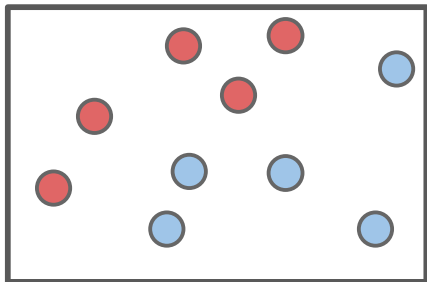
- Renormalize $w_{i,t+1}$ such that

$$\sum_i w_{i,t+1} = 1$$



Boosting

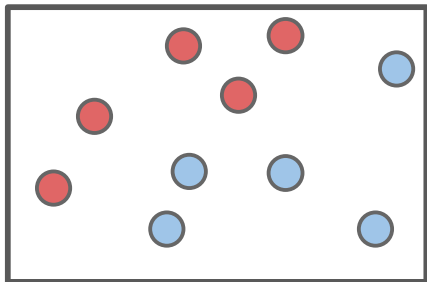
- AdaBoost Process: Visual Walkthrough





Boosting

- AdaBoost Process: Visual Walkthrough



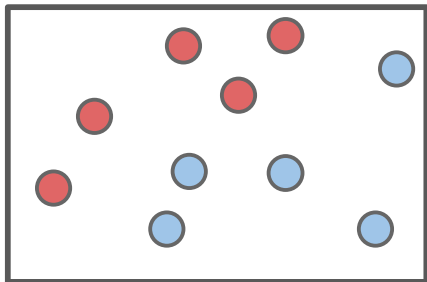
With:

- 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 $h: x \rightarrow \{-1, 1\}$



Boosting

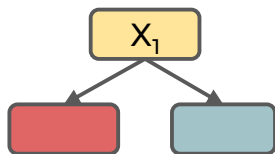
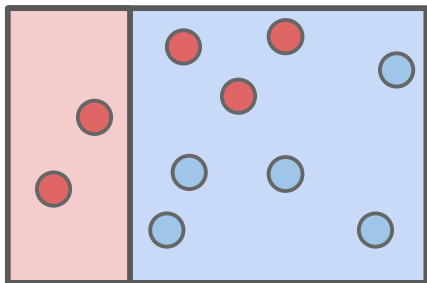
- AdaBoost Process: Visual Walkthrough





Boosting

- AdaBoost Process:



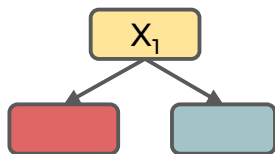
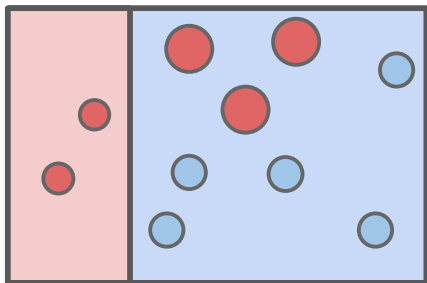
- Choose $h_t(x)$:
- Find weak learner $h_t(x)$ that minimizes ϵ_t , the weighted sum error for misclassified

$$\epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^n w_{i,t}$$



Boosting

- AdaBoost Process:



- Choose $h_t(x)$:

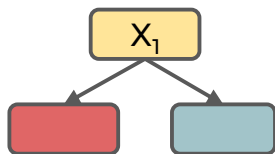
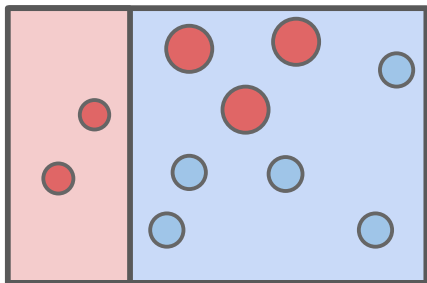
- Find weak learner $h_t(x)$ that minimizes ϵ_t , the weighted sum error for misclassified

$$\text{points } \epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^n w_{i,t}$$



Boosting

- AdaBoost Process:



- Choose $h_t(x)$:

- Find weak learner $h_t(x)$ that minimizes ϵ_t , the weighted sum error for misclassified

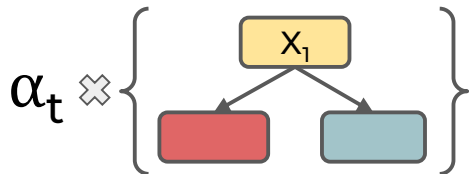
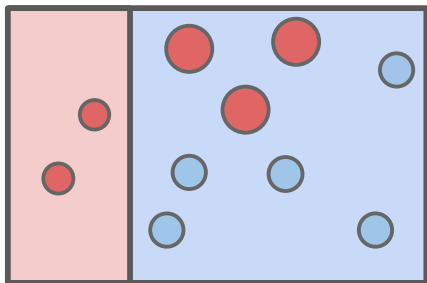
$$\text{points } \epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^n w_{i,t}$$

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



Boosting

- AdaBoost Process:



- Choose $h_t(x)$:

- Find weak learner $h_t(x)$ that minimizes ϵ_t , the weighted sum error for misclassified

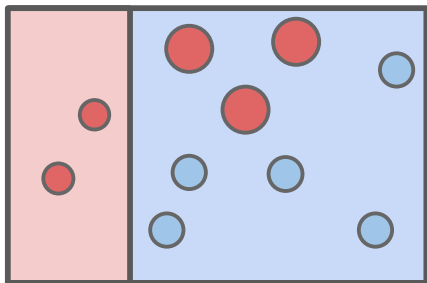
$$\text{points } \epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^n w_{i,t}$$

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



Boosting

- AdaBoost Process:

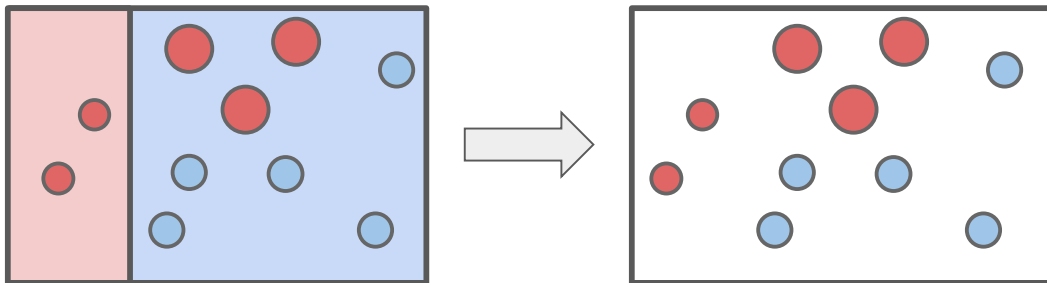


$$\alpha_t \otimes \left\{ \begin{array}{c} x_1 \\ \swarrow \quad \searrow \\ \text{red box} \quad \text{blue box} \end{array} \right\}$$



Boosting

- AdaBoost Process:

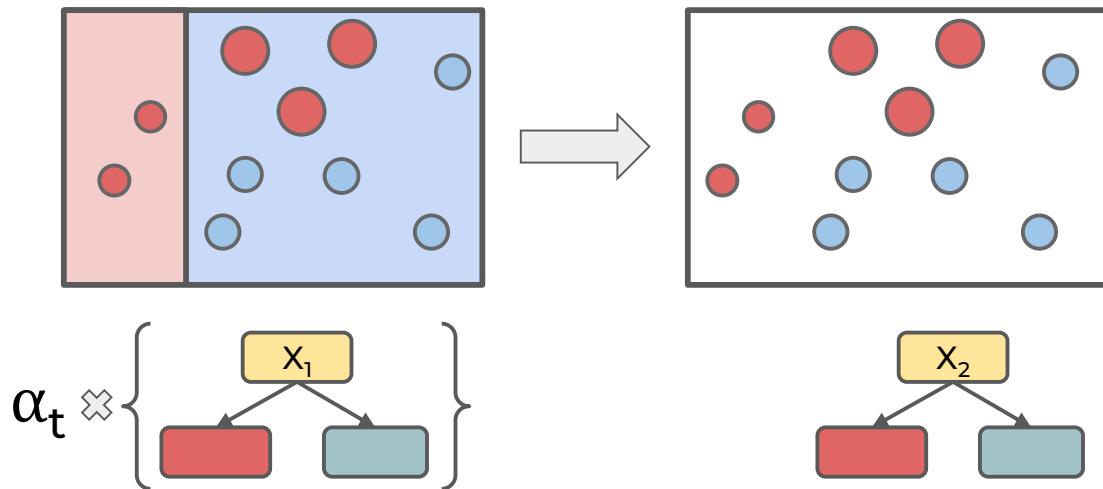


$$\alpha_t \otimes \left\{ \begin{array}{c} \boxed{x_1} \\ \swarrow \quad \searrow \\ \boxed{\text{red}} \quad \boxed{\text{blue}} \end{array} \right\}$$



Boosting

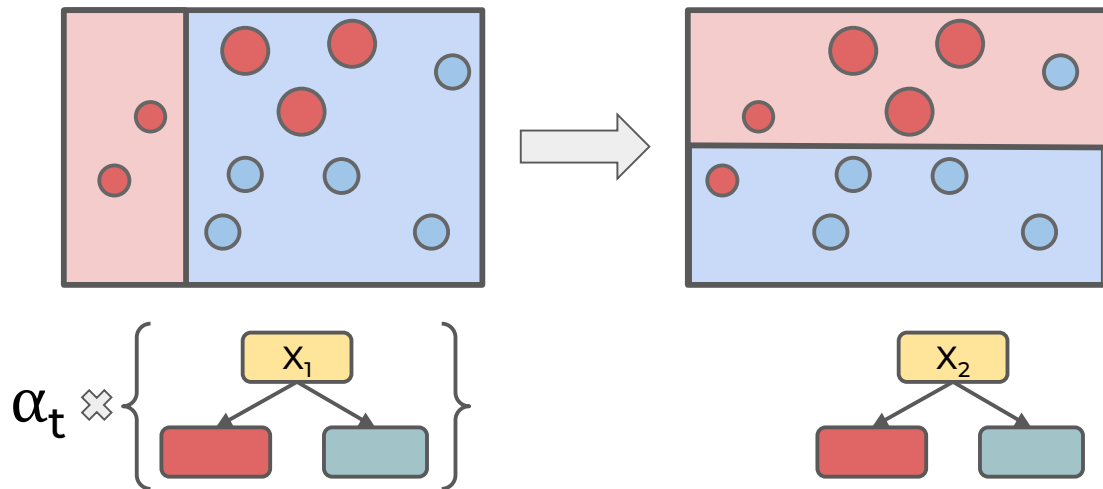
- AdaBoost Process:





Boosting

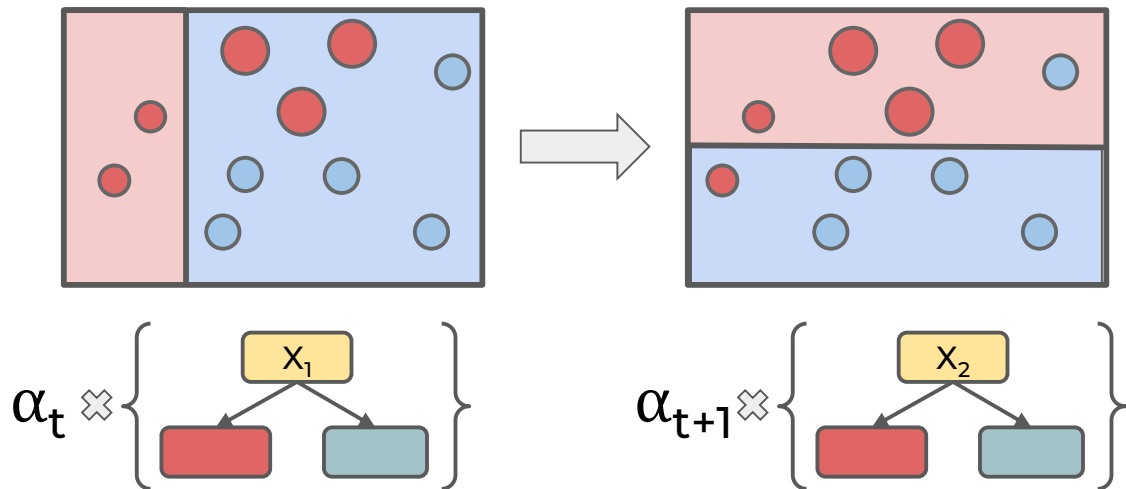
- AdaBoost Process:





Boosting

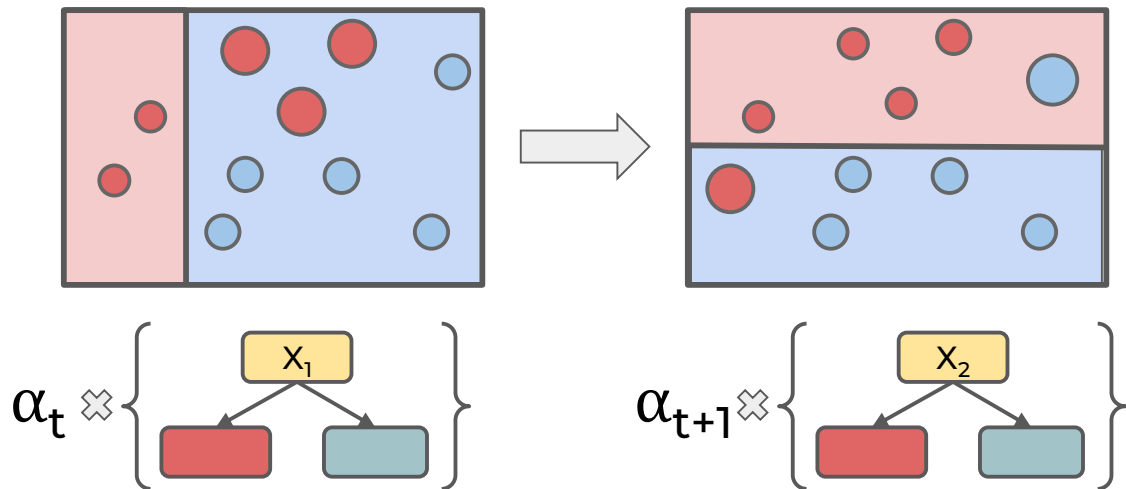
- AdaBoost Process:





Boosting

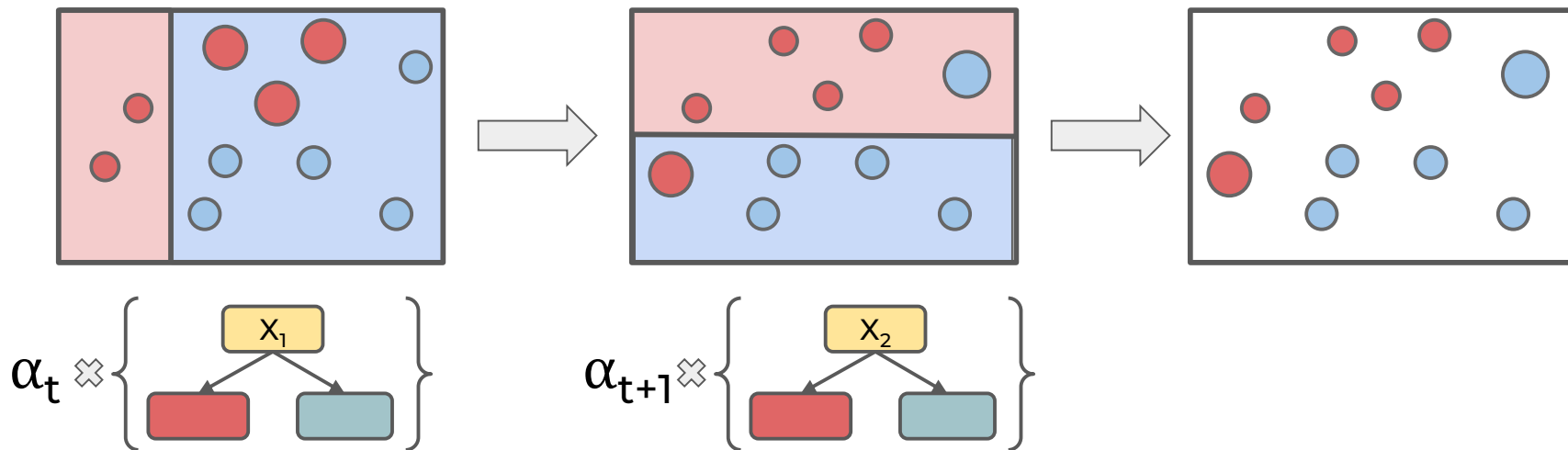
- AdaBoost Process:





Boosting

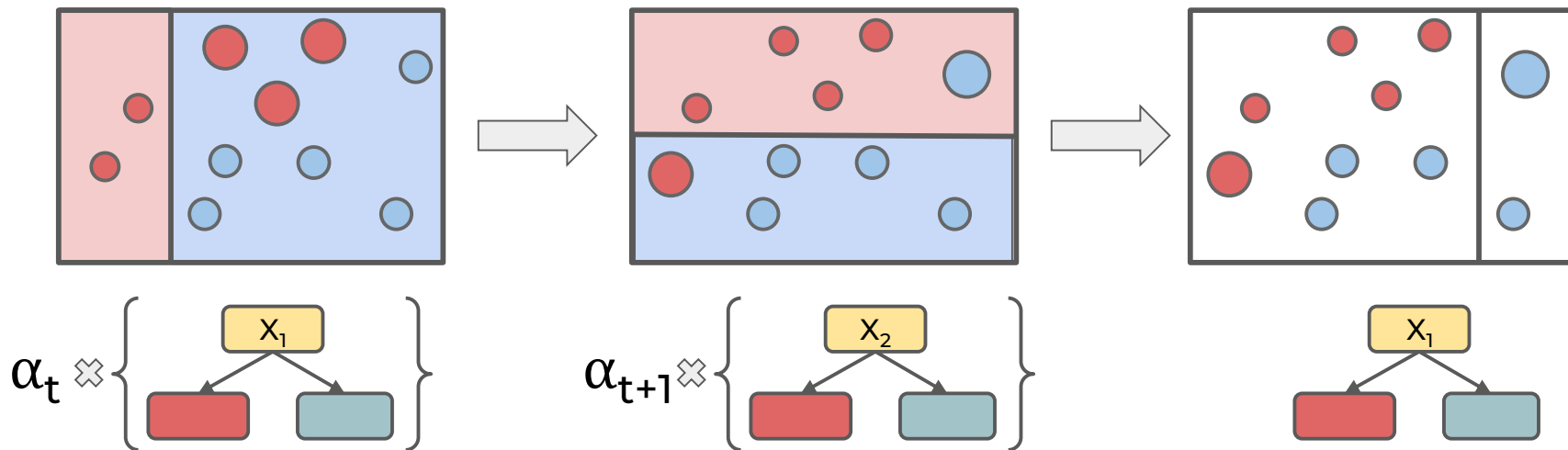
- AdaBoost Process:





Boosting

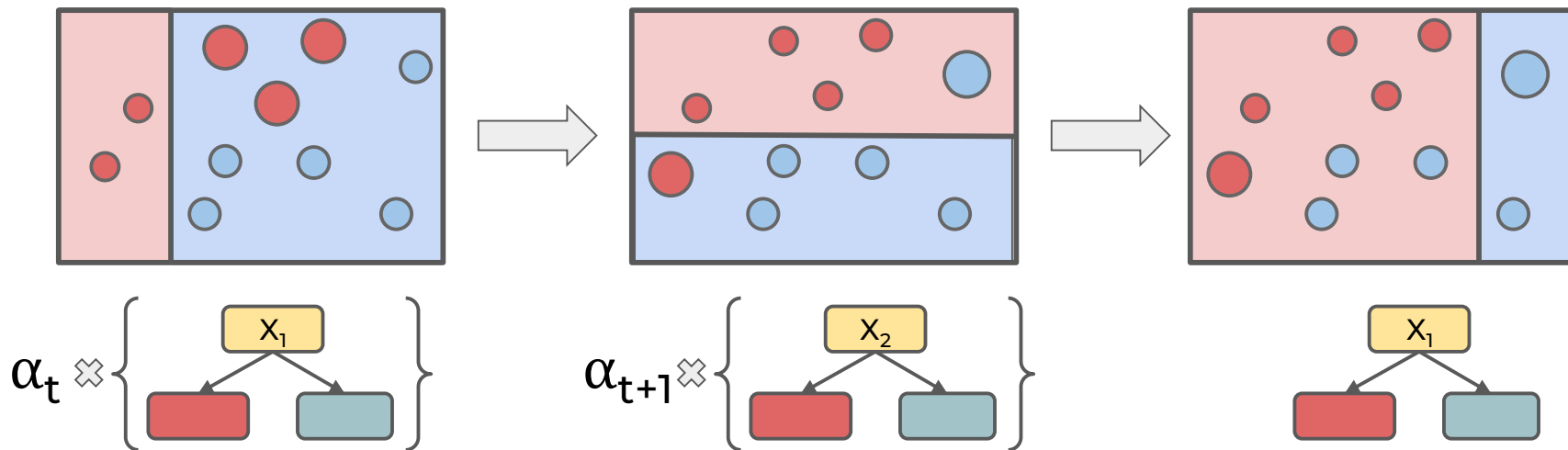
- AdaBoost Process:





Boosting

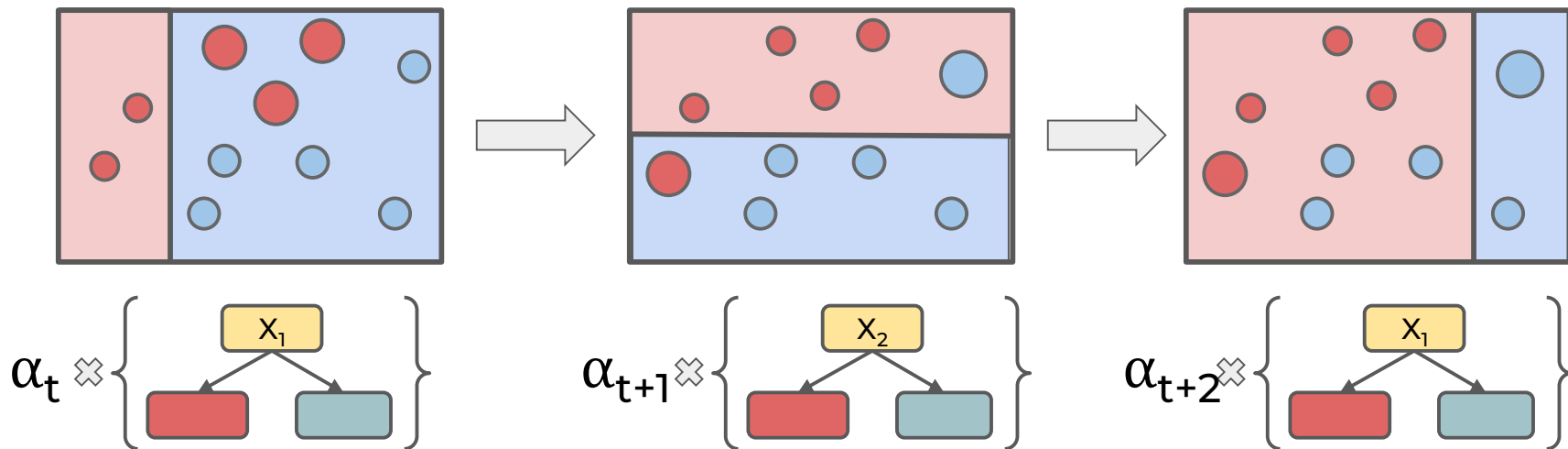
- AdaBoost Process:





Boosting

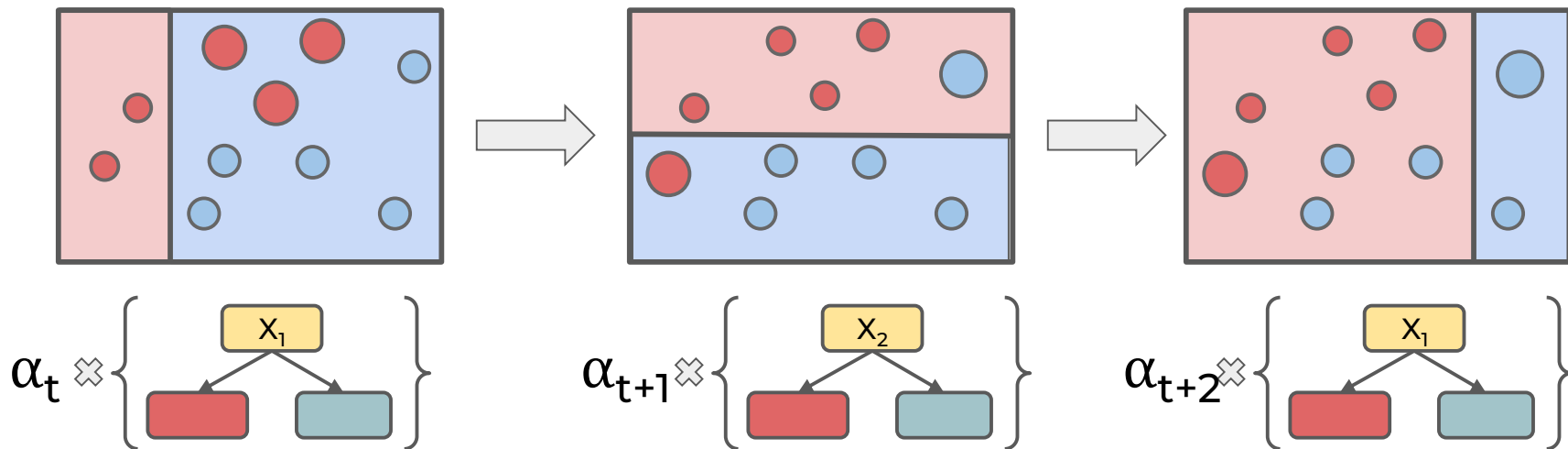
- AdaBoost Process:





Boosting

- AdaBoost Process:

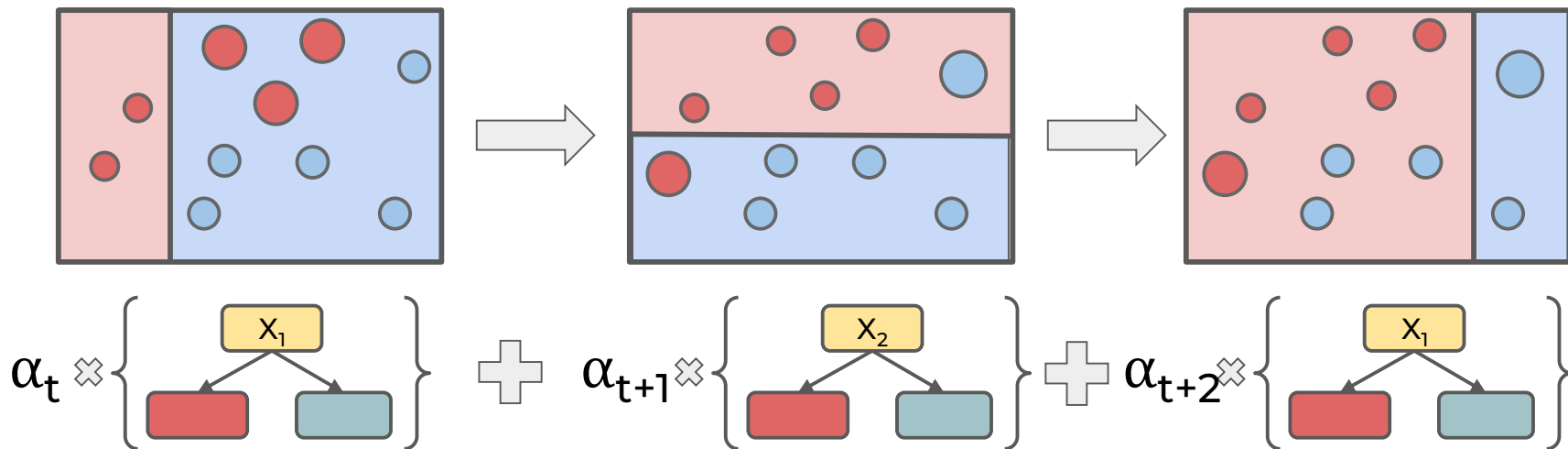


$$F_T(x) = \sum_{t=1}^T f_t(x) \quad f_t(x) = \alpha_t h(x)$$



Boosting

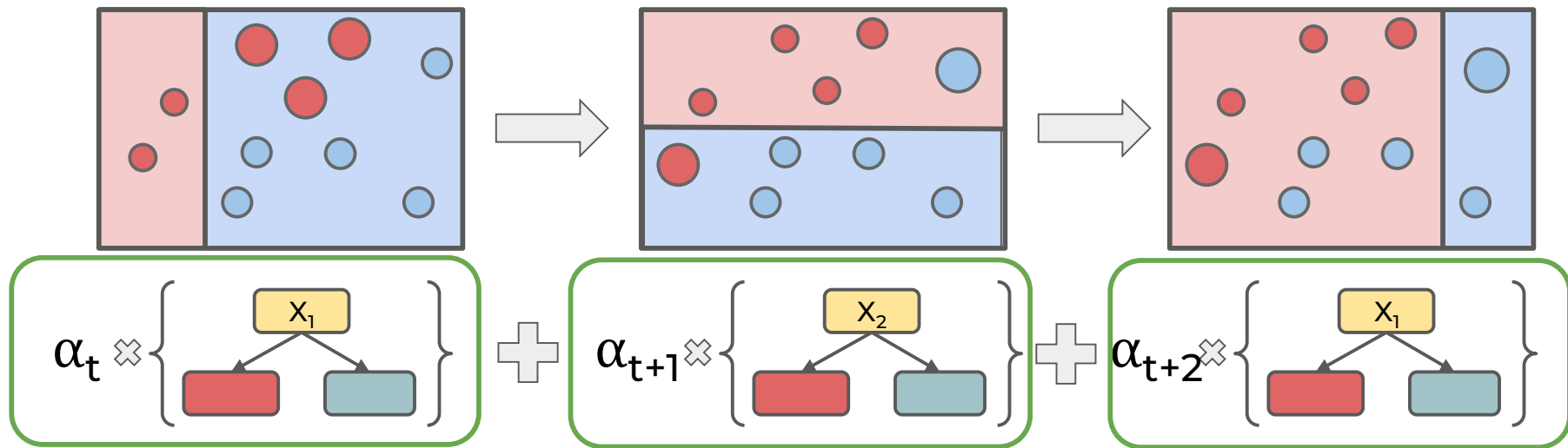
- AdaBoost Process:





Boosting

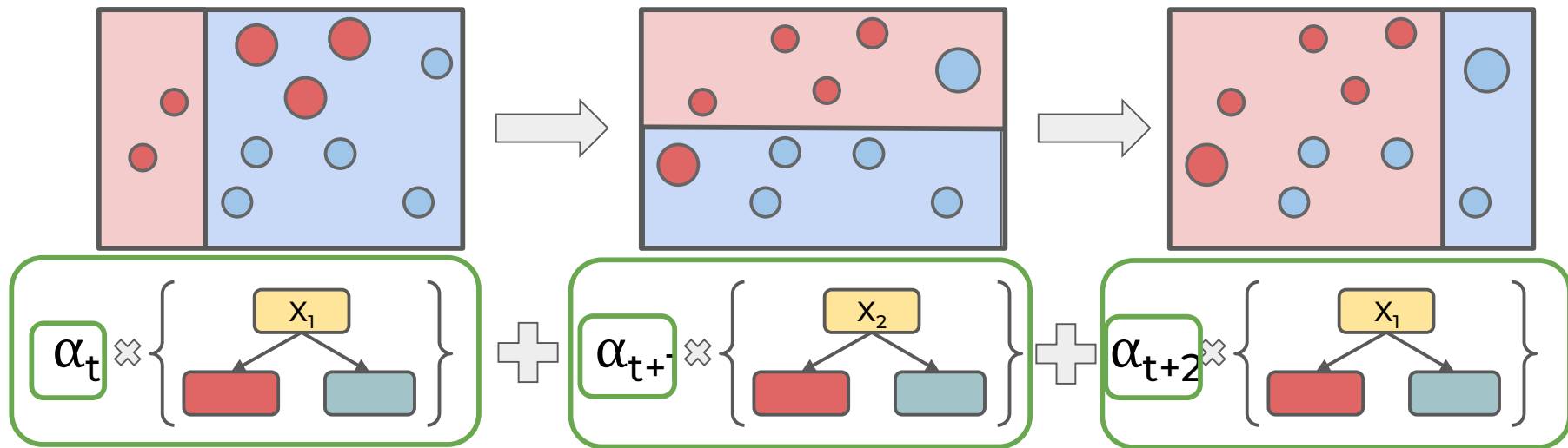
- AdaBoost Process:





Boosting

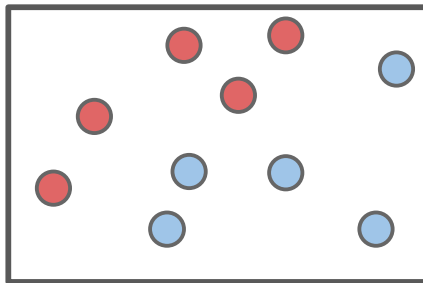
- AdaBoost Process:





Boosting

- AdaBoost Process:

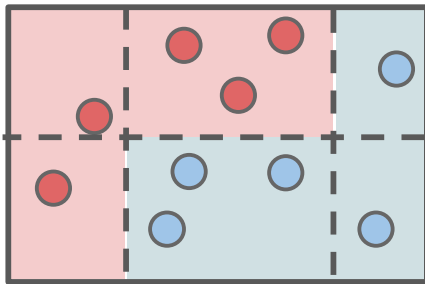


$$\alpha_t \times \left\{ \begin{array}{c} \text{X}_1 \\ \swarrow \quad \searrow \\ \text{Red Box} \quad \text{Blue Box} \end{array} \right\} + \alpha_{t+1} \times \left\{ \begin{array}{c} \text{X}_2 \\ \swarrow \quad \searrow \\ \text{Red Box} \quad \text{Blue Box} \end{array} \right\} + \alpha_{t+2} \times \left\{ \begin{array}{c} \text{X}_1 \\ \swarrow \quad \searrow \\ \text{Red Box} \quad \text{Blue Box} \end{array} \right\}$$



Boosting

- AdaBoost Process:



$$\alpha_t \times \left\{ \begin{array}{c} x_1 \\ \swarrow \quad \searrow \\ \text{red box} \quad \text{blue box} \end{array} \right\} + \alpha_{t+1} \times \left\{ \begin{array}{c} x_2 \\ \swarrow \quad \searrow \\ \text{red box} \quad \text{blue box} \end{array} \right\} + \alpha_{t+2} \times \left\{ \begin{array}{c} x_1 \\ \swarrow \quad \searrow \\ \text{red box} \quad \text{blue box} \end{array} \right\}$$



Boosting

- AdaBoost uses an ensemble of **weak learners** that learn slowly in series.
- Certain weak learners have more “say” 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.



Boosting

- 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.



Boosting

- Notes on Adaptive Boosting:
 - 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.



Boosting

- Let's move on to seeing AdaBoost in practice!



AdaBoost

Coding Walkthrough Part One: Data



Boosting

- For this particular ML project, we will be walking through an example where a predictive model is not the main goal!
- Let's give you a brief overview of the data to fully understand the scope of the project.



Boosting

- Mushroom Data
 - Variety of categorical features on mushrooms.
 - Can we use ML to accomplish two tasks:
 - Predict Poisonous vs. Edible.
 - Create cautionary guidelines for people picking mushrooms.



Boosting

- Tree based methods have great capabilities to report feature importance.
- Adaboost specifically has stumps focusing on one feature at a time, which could be useful in creating mushroom picking guidelines.
- Let's check out the data in a notebook!



AdaBoost

Coding Walkthrough Part Two: Model



Gradient Boosting

Theory and Intuition



Boosting

- Gradient Boosting is a very similar idea to AdaBoost, where weak learners are created in series in order to produce a strong ensemble model.
- Gradient Boosting makes use of the residual error for learning.



Boosting

- Gradient Boosting vs. Adaboost:
 - Larger Trees allowed in Gradient Boosting.
 - Learning Rate coefficient same for all weak learners.
 - Gradual series learning is based on training on the **residuals** of the previous model.



Boosting

- Gradient Boosting Regression Example

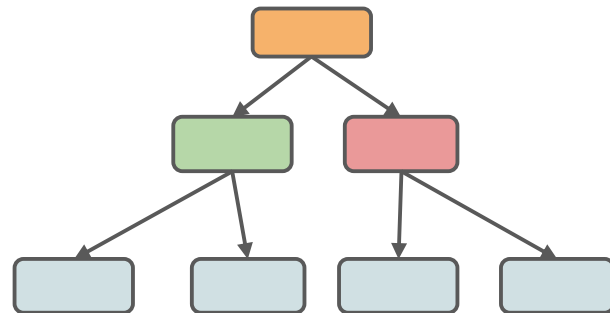
Area m ²	Bedrooms	Bathrooms	Price
200	3	2	\$500,000
190	2	1	\$462,000
230	3	3	\$565,000



Boosting

- Train a decision tree on data

Area m ²	Bedrooms	Bathrooms	Price
200	3	2	\$500,000
190	2	1	\$462,000
230	3	3	\$565,000

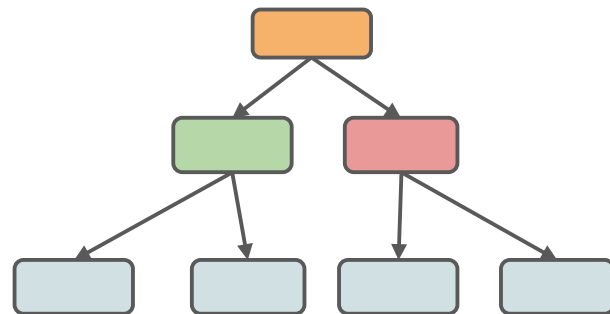




Boosting

- Note - not just a stump!

Area m ²	Bedrooms	Bathrooms	Price
200	3	2	\$500,000
190	2	1	\$462,000
230	3	3	\$565,000

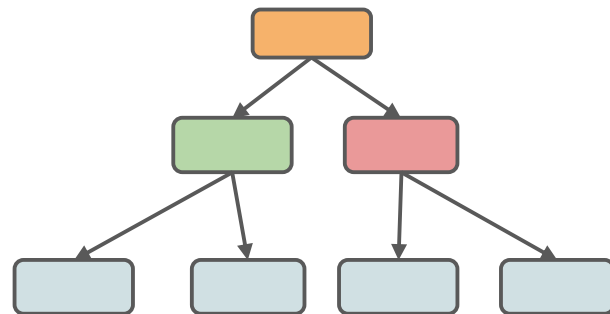




Boosting

- Get predicted \hat{y} value

Area m ²	Bedrooms	Bathrooms	Price
200	3	2	\$500,000
190	2	1	\$462,000
230	3	3	\$565,000

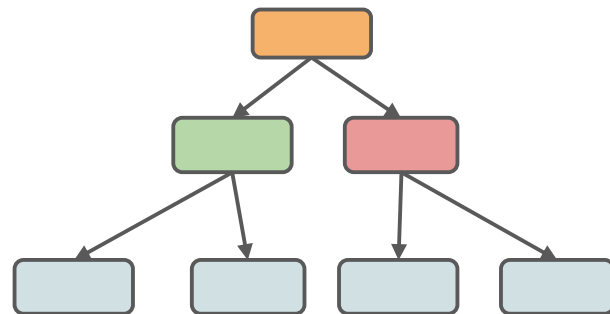




Boosting

- Get predicted \hat{y} value

Area m ²	Bedrooms	Bathrooms	y
200	3	2	\$500,000
190	2	1	\$462,000
230	3	3	\$565,000

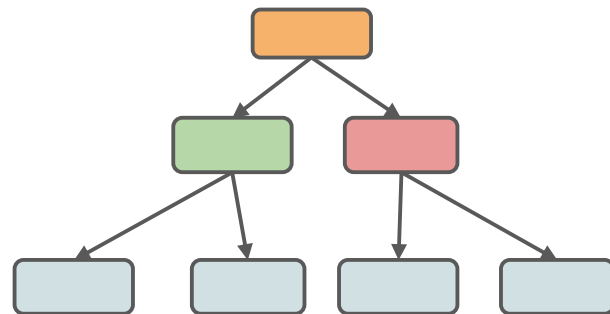




Boosting

- Get predicted \hat{y} value

y	\hat{y}
\$500,000	\$509,000
\$462,000	\$509,000
\$565,000	\$509,000

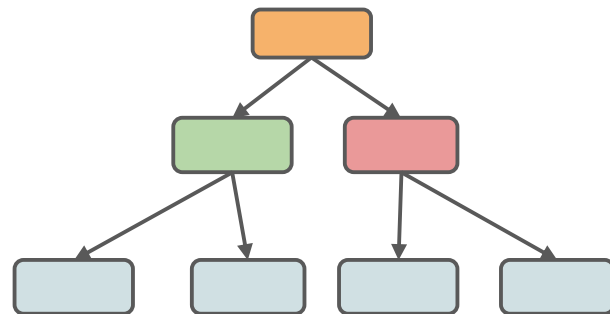




Boosting

- Calculate residual: $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$

y	\hat{y}	e
\$500,000	\$509,000	-\$9,000
\$462,000	\$509,000	-\$47,000
\$565,000	\$509,000	\$56,000





Boosting

- Create new model to predict the **error**

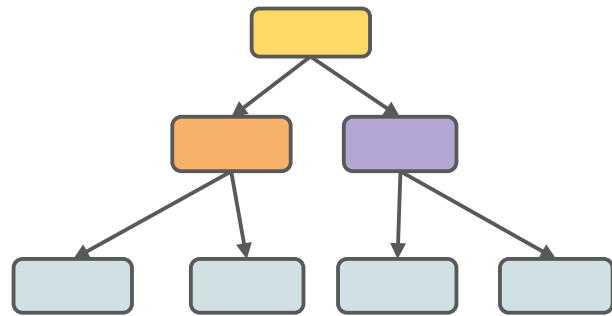
y	\hat{y}	e
\$500,000	\$509,000	-\$9,000
\$462,000	\$509,000	-\$47,000
\$565,000	\$509,000	\$56,000



Boosting

- Create new model to predict the **error**

y	\hat{y}	e
\$500,000	\$509,000	-\$9,000
\$462,000	\$509,000	-\$47,000
\$565,000	\$509,000	\$56,000

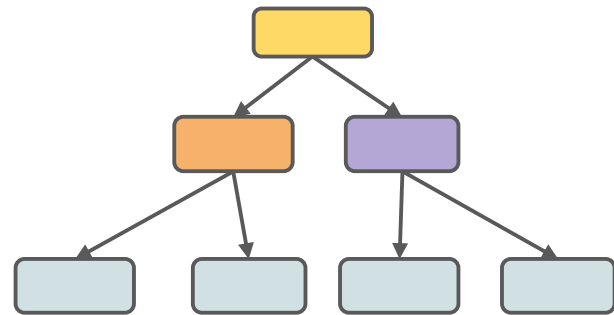




Boosting

- Create new model to predict the **error**

y	\hat{y}	e	$f1$
\$500,000	\$509,000	-\$9,000	-\$8,000
\$462,000	\$509,000	-\$47,000	-\$50,000
\$565,000	\$509,000	\$56,000	\$50,000



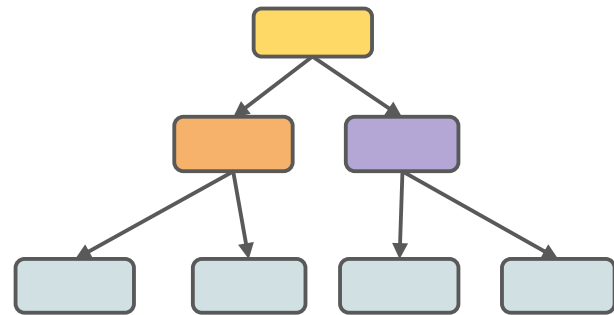


Boosting

- Create new model to predict the **error**

y	\hat{y}	e	$f1$
\$500,000	\$509,000	-\$9,000	-\$8,000
\$462,000	\$509,000	-\$47,000	-\$50,000
\$565,000	\$509,000	\$56,000	\$50,000

Area m ²	Bedrooms	Bathrooms
200	3	2
190	2	1
230	3	3



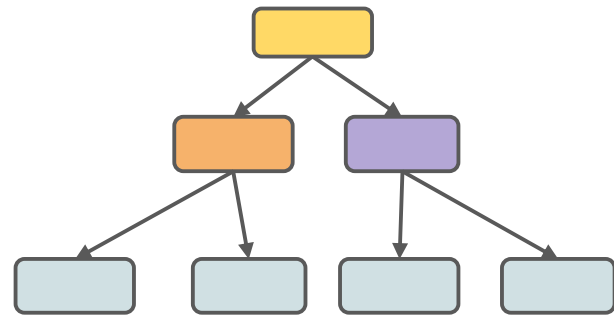


Boosting

- Create new model to predict the **error**

y	\hat{y}	e	$f1$
\$500,000	\$509,000	-\$9,000	-\$8,000
\$462,000	\$509,000	-\$47,000	-\$50,000
\$565,000	\$509,000	\$56,000	\$50,000

Area m ²	Bedrooms	Bathrooms
200	3	2
190	2	1
230	3	3

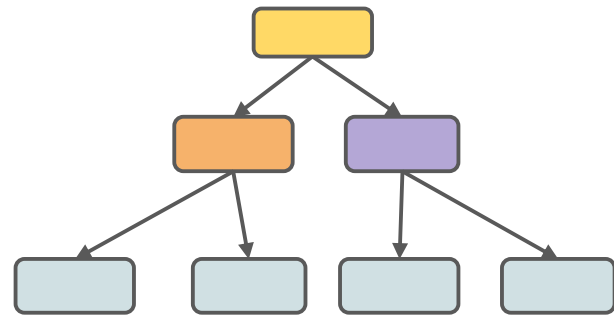




Boosting

- Update prediction using **error prediction**

y	\hat{y}	e	$f1$
\$500,000	\$509,000	-\$9,000	-\$8,000
\$462,000	\$509,000	-\$47,000	-\$50,000
\$565,000	\$509,000	\$56,000	\$50,000

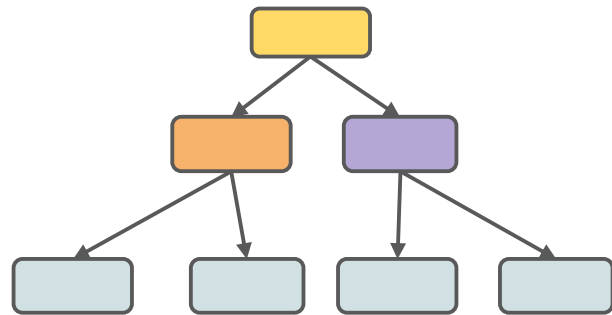




Boosting

- Update prediction using **error prediction**

y	\hat{y}	e	$f1$	$F1 = \hat{y} + f1$
\$500,000	\$509,000	-\$9,000	-\$8,000	
\$462,000	\$509,000	-\$47,000	-\$50,000	
\$565,000	\$509,000	\$56,000	\$50,000	

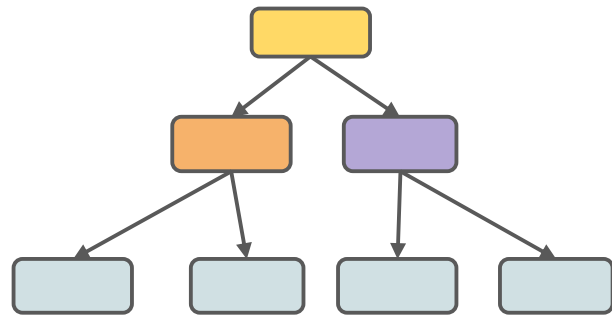




Boosting

- Update prediction using **error prediction**

y	\hat{y}	e	$f1$	$F1 = \hat{y} + f1$
\$500,000	\$509,000	-\$9,000	-\$8,000	\$501,000
\$462,000	\$509,000	-\$47,000	-\$50,000	\$459,000
\$565,000	\$509,000	\$56,000	\$50,000	\$559,000

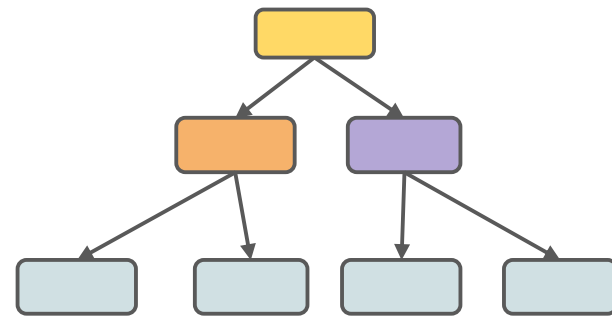




Boosting

- We can continue this process in series

y	\hat{y}	e	$f1$	$F1 = \hat{y} + f1$
\$500,000	\$509,000	-\$9,000	-\$8,000	\$501,000
\$462,000	\$509,000	-\$47,000	-\$50,000	\$459,000
\$565,000	\$509,000	\$56,000	\$50,000	\$559,000





Boosting

- Gradient Boosting Process

$$F_m = F_{m-1} + f_m$$



Boosting

- Gradient Boosting Process

$$F_m = F_{m-1} + f_m$$

$$F_m = F_{m-1} + (\text{learning rate} * f_m)$$



Boosting

- Gradient Boosting Process
 - Create initial model: f_0
 - Train another model on error
 - $e = y - f_0$
 - Create new prediction
 - $F_1 = f_0 + \eta f_1$
 - Repeat as needed
 - $F_m = f_{m-1} + \eta f_m$



Boosting

- Note, for classification we can use the logit as an error metric:

$$\hat{y} = \log \left(\frac{\hat{p}}{1 - \hat{p}} \right) \qquad \hat{p} = \frac{1}{1 + e^{-\hat{y}}}$$



Boosting

- Note, the learning rate is the same for each new model in the series, it is **not** unique to each subsequent model (unlike AdaBoost's alpha coefficient).
- Gradient Boosting is fairly robust to overfitting, allowing for the number of estimators to be set high by default (~100).



Boosting

- Gradient Boosting Intuition
 - We optimize the series of trees by learning on the residuals, forcing subsequent trees to attempt to correct for the error in the previous trees.



Boosting

- Gradient Boosting Intuition
 - The trade-off is training time.
 - A learning rate is between 0-1, which means a very low value would mean each subsequent tree has little “say”, meaning more trees need to be created, causing a longer computational training time.



Boosting

- Let's explore Gradient Boosting in practice in the next lecture!



Gradient Boosting

Coding Walkthrough Example



Supervised Learning Project Exercise

Overview Lecture



Supervised Learning Project

- We've learned a lot of Supervised Learning algorithms!
- Let's wrap up all these sections and the overview of tree methods with a supervised learning project.



Supervised Learning Project

- For this project we will explore the churn of an internet and telephone provider, including:
 - Exploratory data analysis
 - Churn cohort analysis
 - Predictive machine learning



Supervised Learning Project

- Three approaches to this project:



Supervised Learning Project

- Three approaches to this project:
 - Load up the dataset and create a predictive model
 - Create any visualizations you find useful.
 - Use any supervised learning model you prefer.



Supervised Learning Project

- Three approaches to this project:
 - Use the notebook as a guide and complete the tasks in the notebook.
 - Exploratory Data Analysis
 - Churn and Cohort Analysis
 - Predictive Tree-Based Models



Supervised Learning Project

- Three approaches to this project:
 - Relax and treat this as a code along, go straight to the solutions notebook and video, following along to see where you would approach the data differently as we guide you.



Supervised Learning Project

- Let's jump to the exercise project notebook and check it out!