Gradient Boosting

Gradient Boosting (recap)

Learning from mistakes

- In gradient boosting, “weak” base estimators are iteratively added to an ensemble after being **fit on the residuals** of the current ensemble.
- The contribution of each new estimator is scaled by a hyperparameter, λ, called the **learning rate**.
- Additional hyperparameters are the number of estimators in the ensemble and the complexity of the base estimators (e.g., max depth).

Number of estimators in ensemble

ith weak estimator

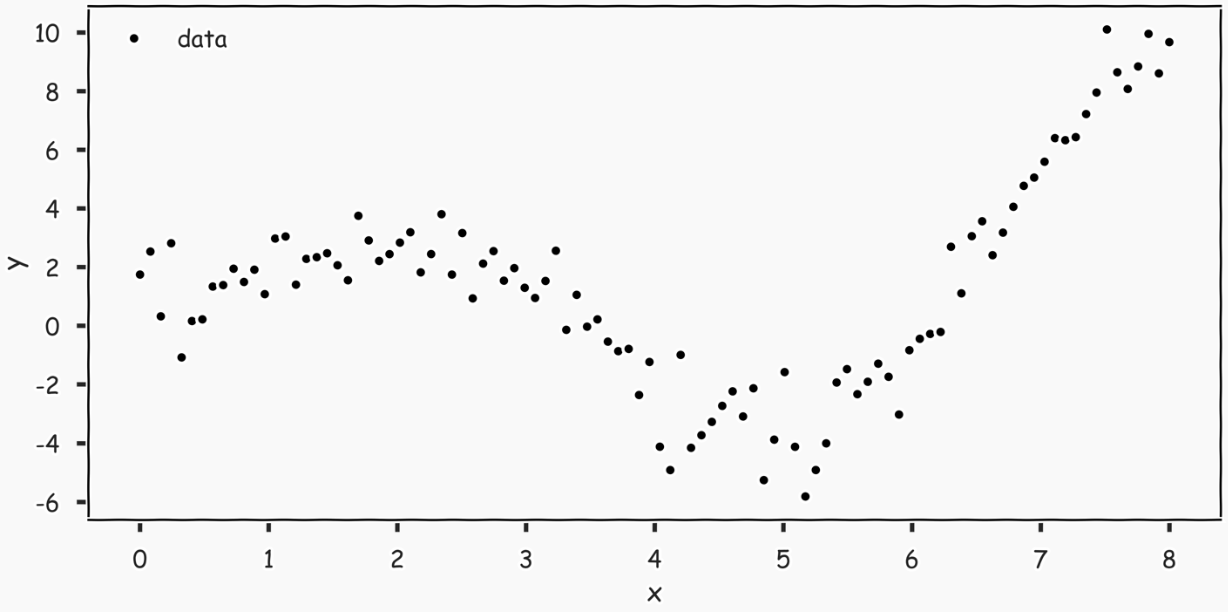
$$T(x) = \sum_{i=1}^{N} \lambda S^{(i)}(x)$$

Learning rate

Boosted ensemble model

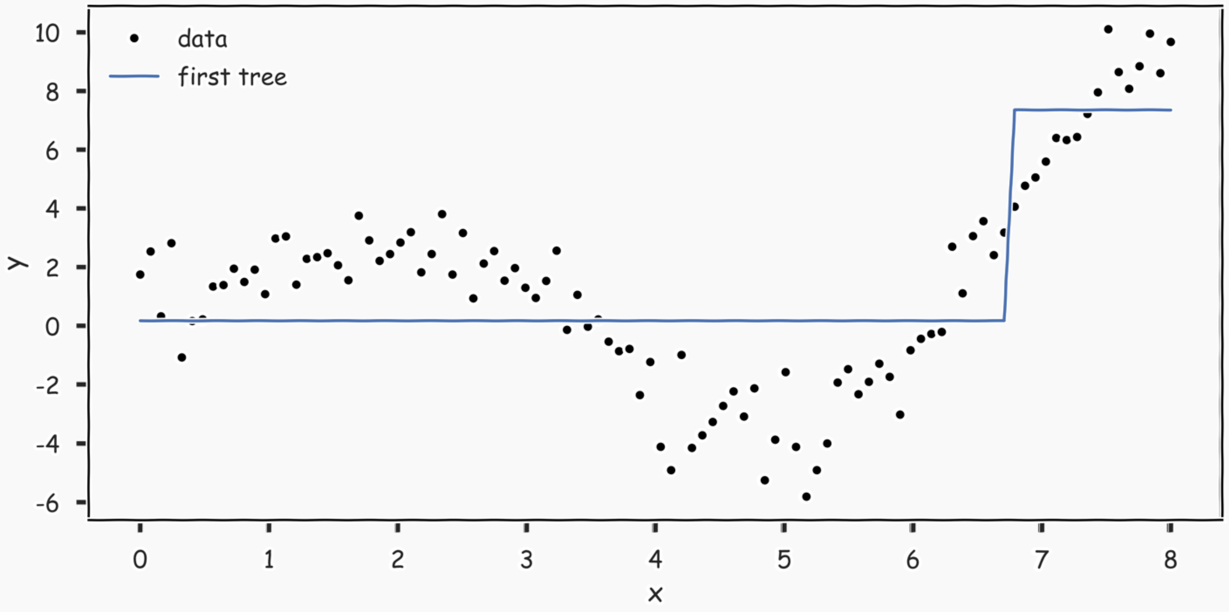
Gradient Boosting: Illustration

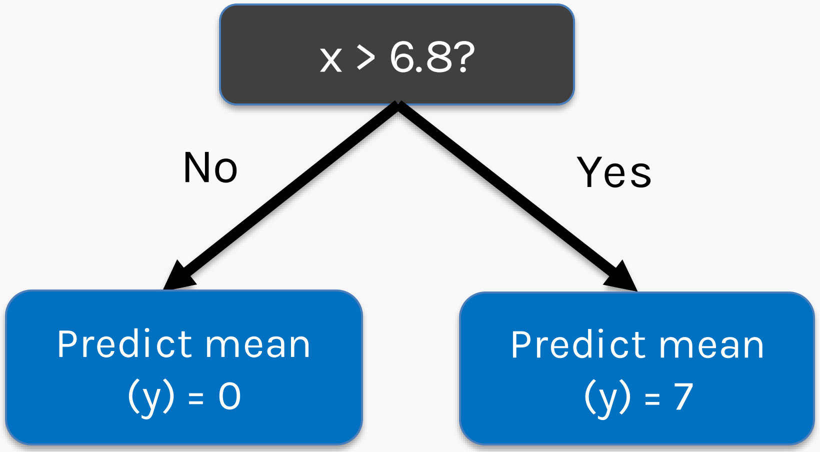
Consider the following dataset:



Gradient Boosting: Illustration

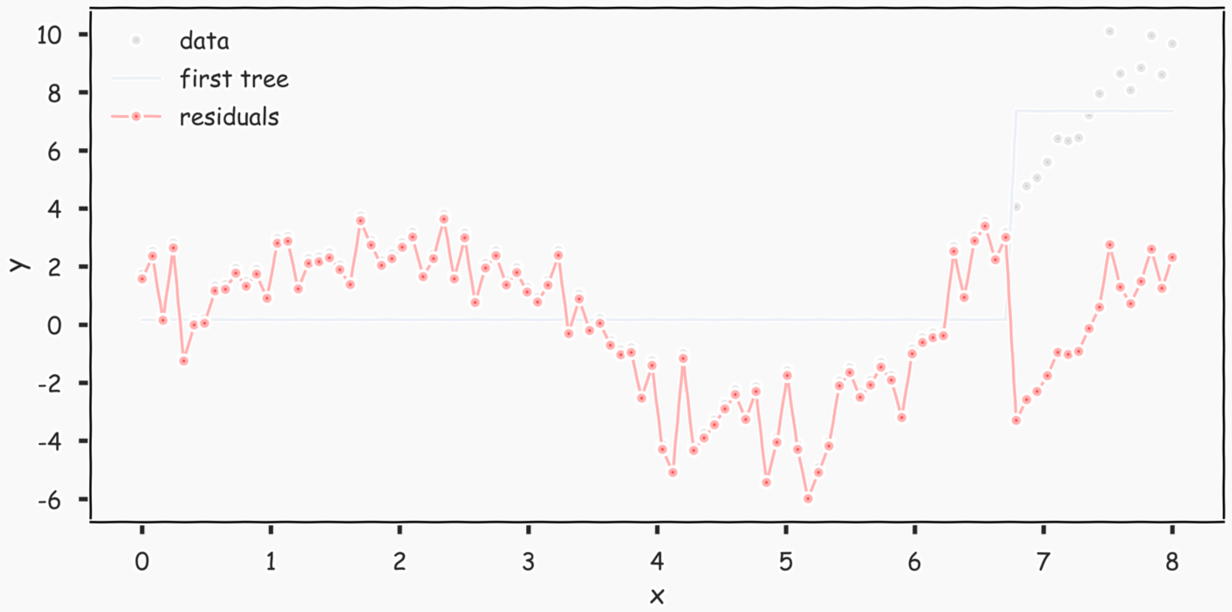
Step 1: Fit a simple model $S^{(0)}$ on the training data: $\{(x_1, y_1), \dots, (x_N, y_N)\}$.

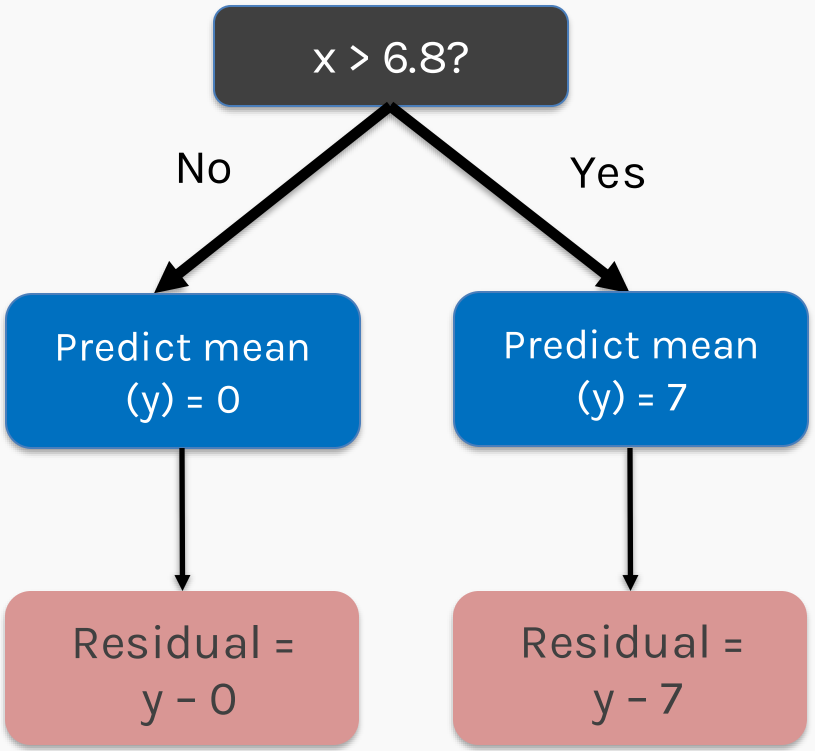




Gradient Boosting: Illustration

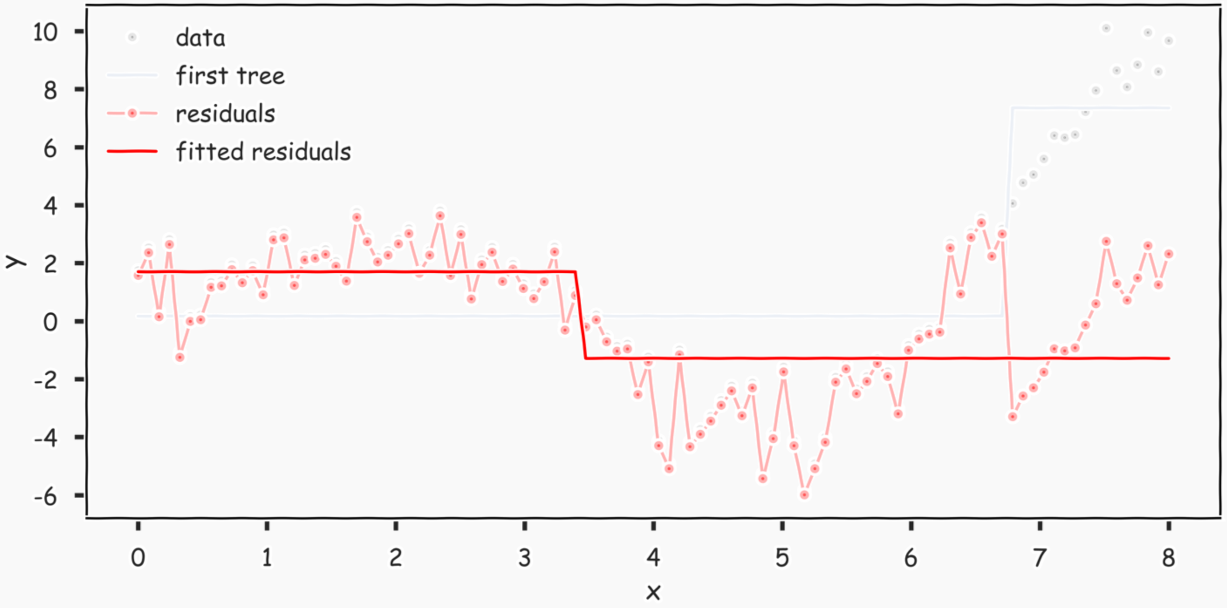
Step 2: Compute the residuals $\{r_1, \ldots, r_N\}$ for $S^{(0)}$. Set $T \leftarrow S^{(0)}$.

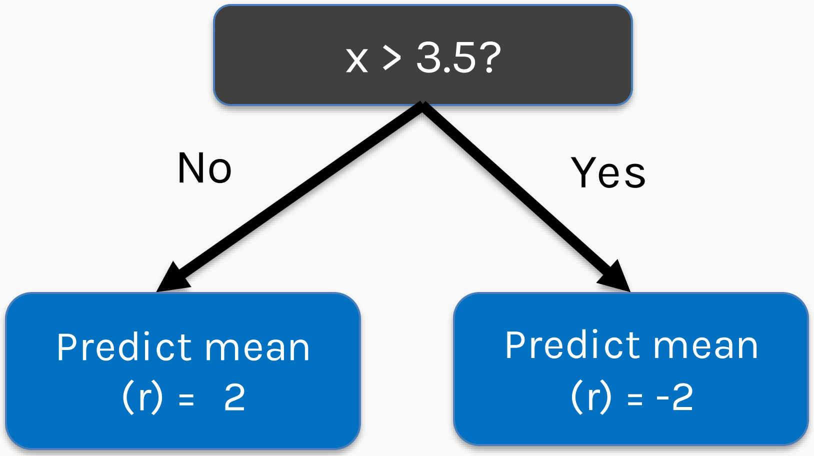




Gradient Boosting: Illustration

Step 3: Fit another model $S^{(1)}$ on: $\{(x_1, r_1), \ldots, (x_N, r_N)\}$.

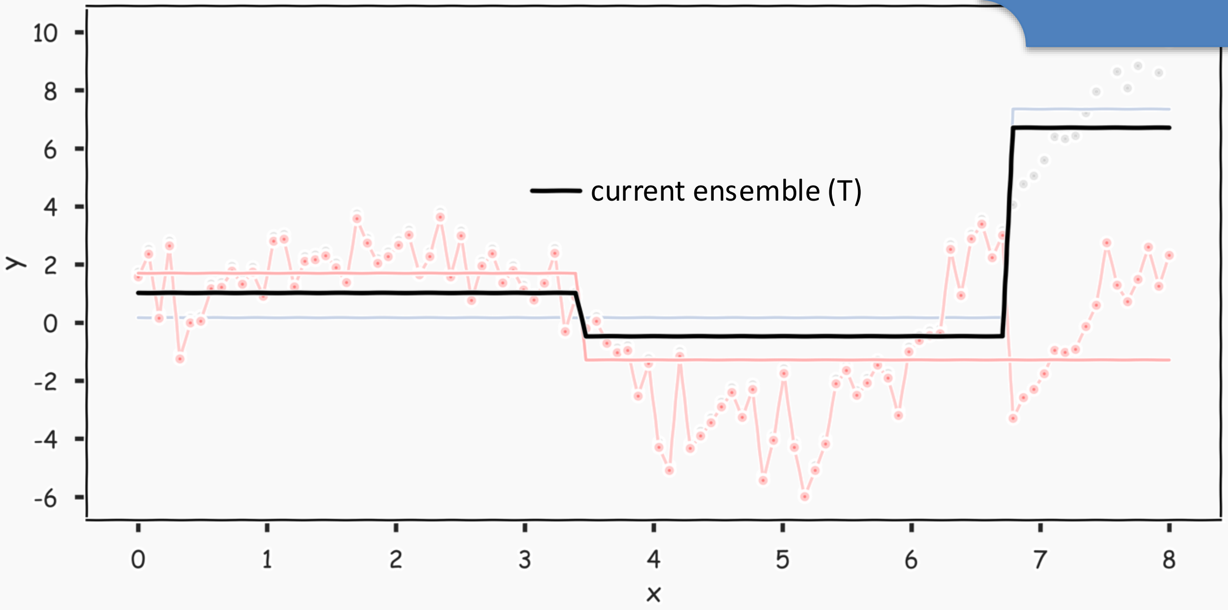


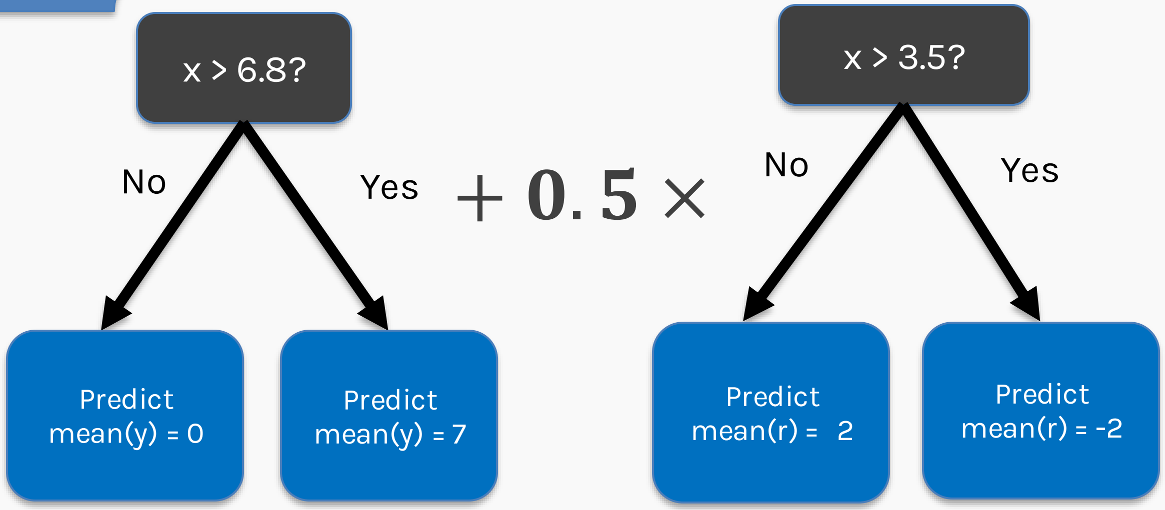


Gradient Boosting: Illustration

Step 4: Combine the two trees in step 1 and 3 by setting $T \leftarrow T + \lambda S^{(1)}$.
Assume $\lambda = 0.5$.

..and repeat!

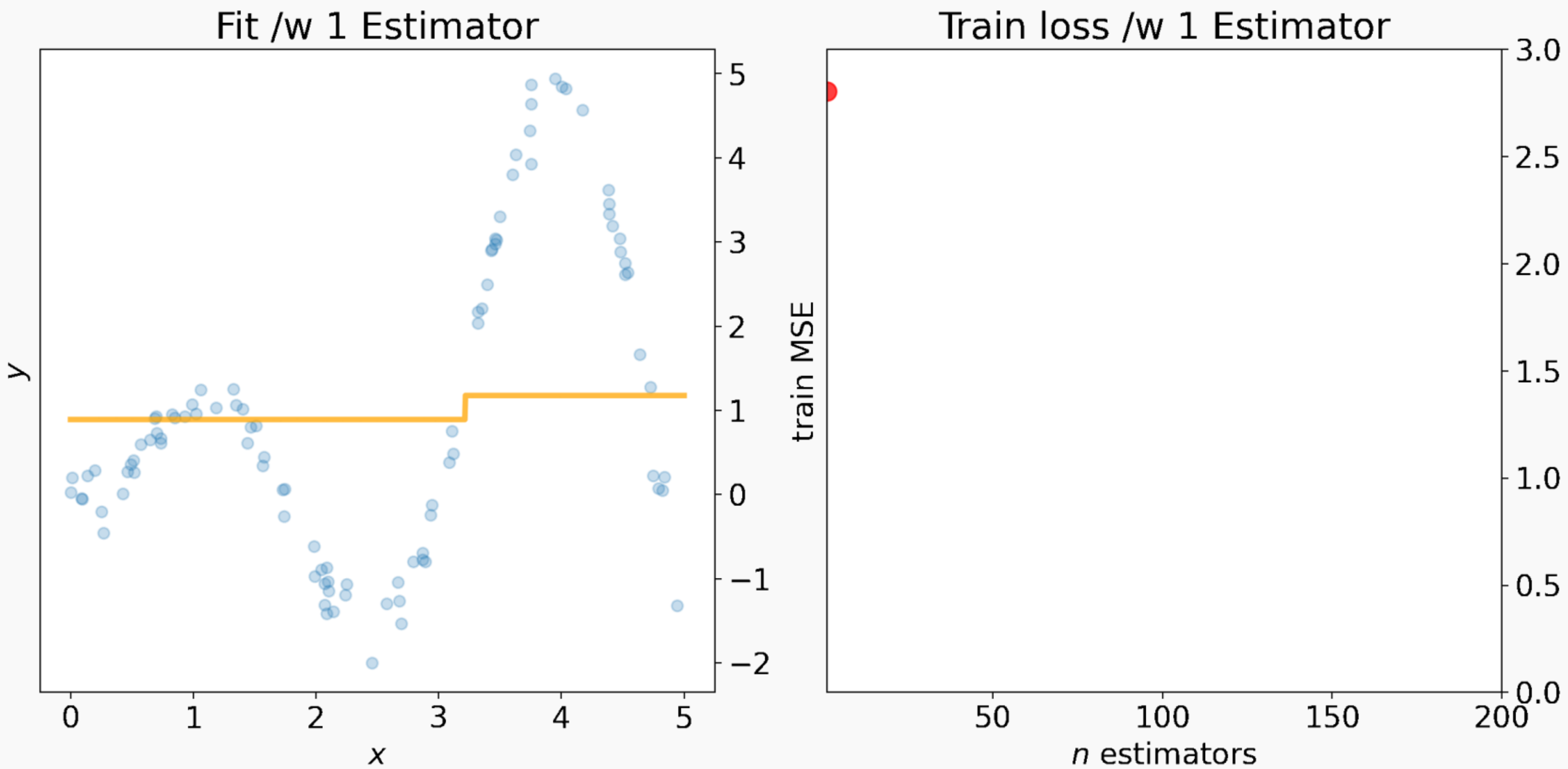




Gradient Boosting – fitting to residuals

Each new estimator is fit to approximate the residuals of the current ensemble.

$$r^{(i)} \leftarrow r^{(i-1)} - T^{(i)}(x)$$



Gradient Boosting: Algorithm

Step 1: Fit a simple model $S^{(0)}$ on the training data $\{(x_1, y_1), \ldots, (x_N, y_N)\}$. Set $\mathrm{T} \leftarrow S^{(0)}$.

Step 2: Compute the residuals $\{\mathrm{r}_1, \ldots, \mathrm{r}_\mathrm{N}\}$ for T.

For $i = 1 \ldots$ until stopping condition is met:

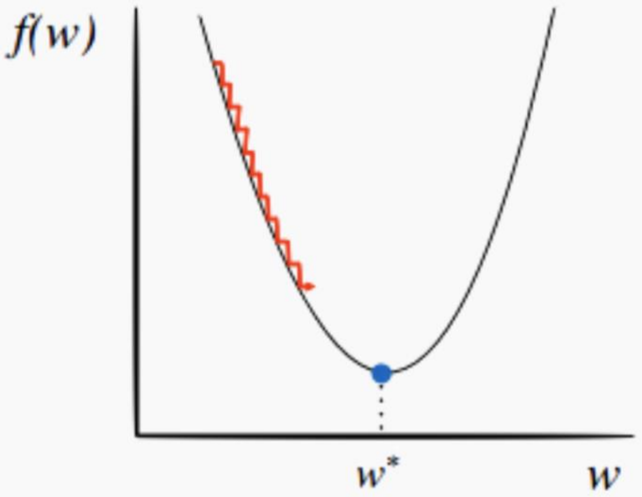
Step 3: Fit a simple model, $S^{(i)}$, to the current **residuals** i.e., train using $\{(x_1, r_1), \ldots, (x_N, r_N)\}$

Step 4: Set the current model T to $T \leftarrow T + \lambda S^{(i)}$

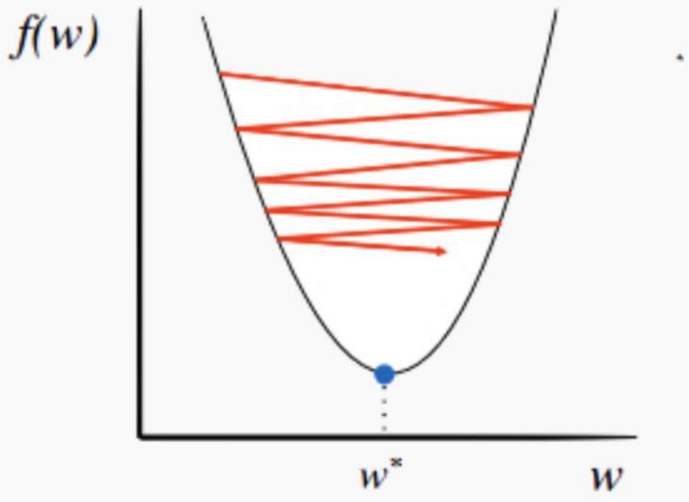
Step 5: Compute residuals, set $r_n \leftarrow r_n - \lambda S^{(i)}(x_n),\ n = 1, \ldots, N$ where λ is a constant called the **learning rate**.

Choosing a Learning Rate

For a constant learning rate λ:



If λ is too small, it takes too many iterations to reach the optimum.



If λ is too large, the algorithm may ‘bounce’ around the optimum and never get sufficiently close.

Choosing a Learning Rate

Choosing λ:

- If λ is a constant, then it should be tuned through cross validation.
- For better results, use a variable λ. That is, let the value of λ depend on the gradient

$$\lambda = h\left(\left|\left|\nabla f(x)\right|\right|\right)$$

where $\nabla f(x)$ is the magnitude of the gradient. So

  - around the optimum, when the gradient is small, λ should be small
  - far from the optimum, when the gradient is large, λ should be larger

Termination

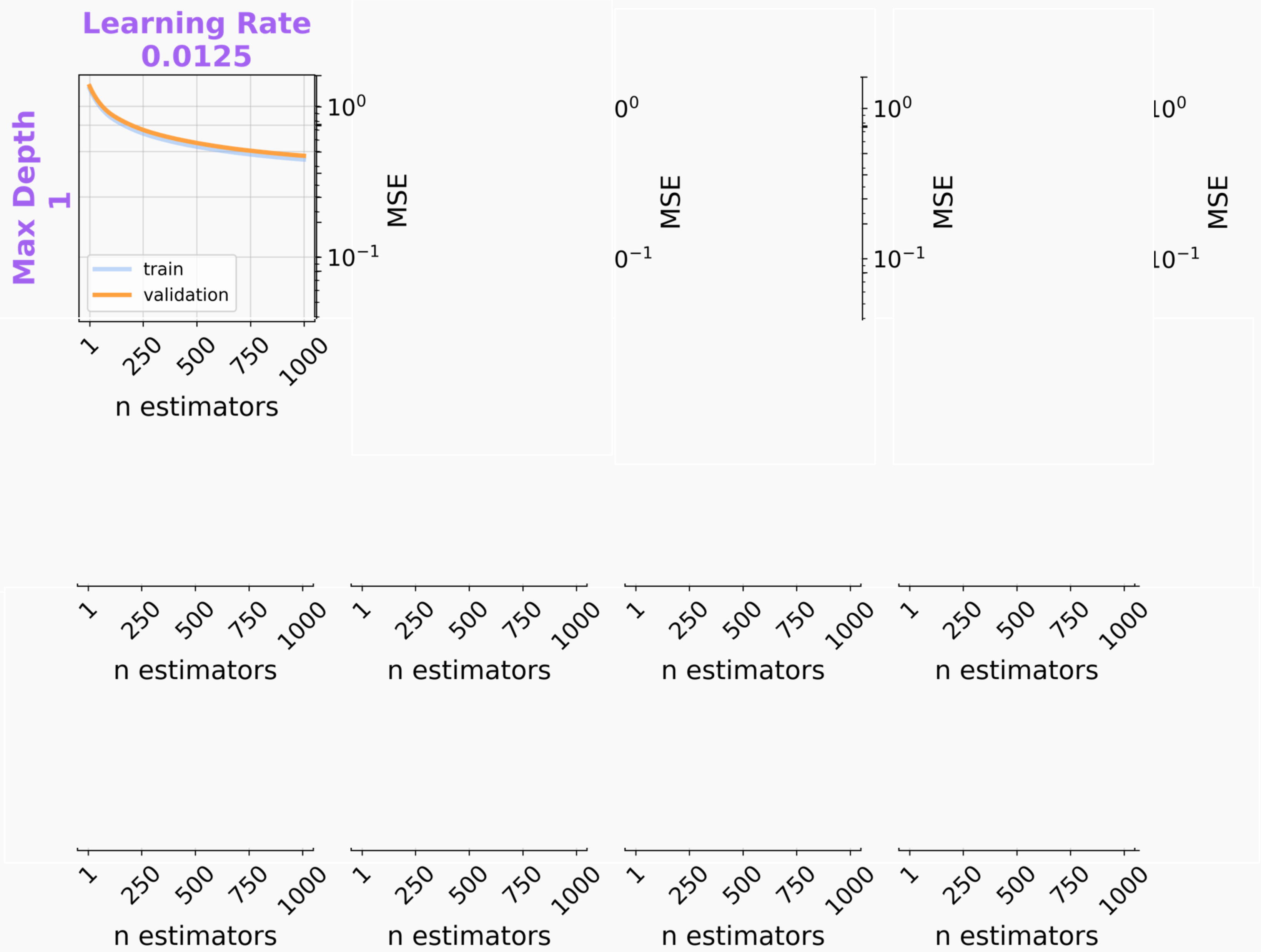
Under ideal conditions, gradient descent iteratively approximates and converges to the optimum.

When do we terminate gradient descent?

- We can limit the number of iterations in the descent. But for an arbitrary choice of maximum iterations, we cannot guarantee that we are sufficiently close to the optimum in the end.
- If the descent is stopped when the updates are sufficiently small (e.g. the residuals of T are small), we encounter a new problem: the algorithm may never terminate!

Both problems have to do with the magnitude of the learning rate, λ.

Gradient Boosting – hyperparameters

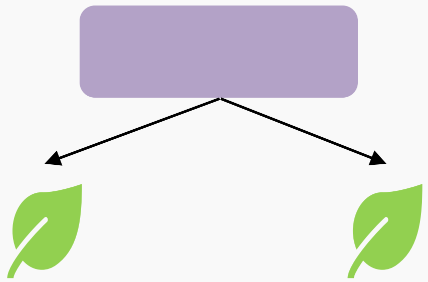


AdaBoost

AdaBoost

There are two main ideas in AdaBoost:

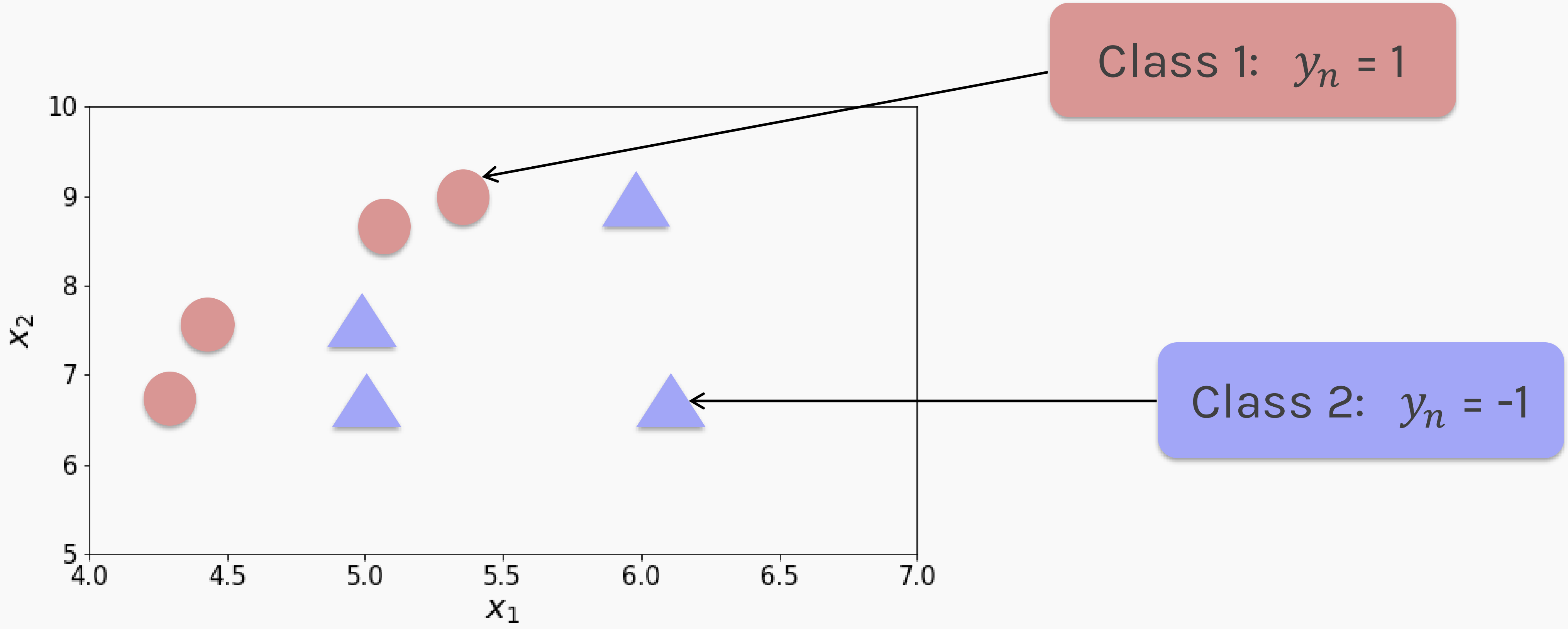
- AdaBoost is a method for **iteratively** building a complex model T by combining several **weak learners** to produce a strong model.
  In AdaBoost, the weak learner that is used is known as a **stump**, i.e., 1 node with 2 leaves.



- Each new stump added to the ensemble learns from the **mistakes** of the **previous** stumps. This is done by **reweighting** observations based on the current stump's predictions. Correctly classified observations are downweighted for future stumps while misclassified ones are upweighted.

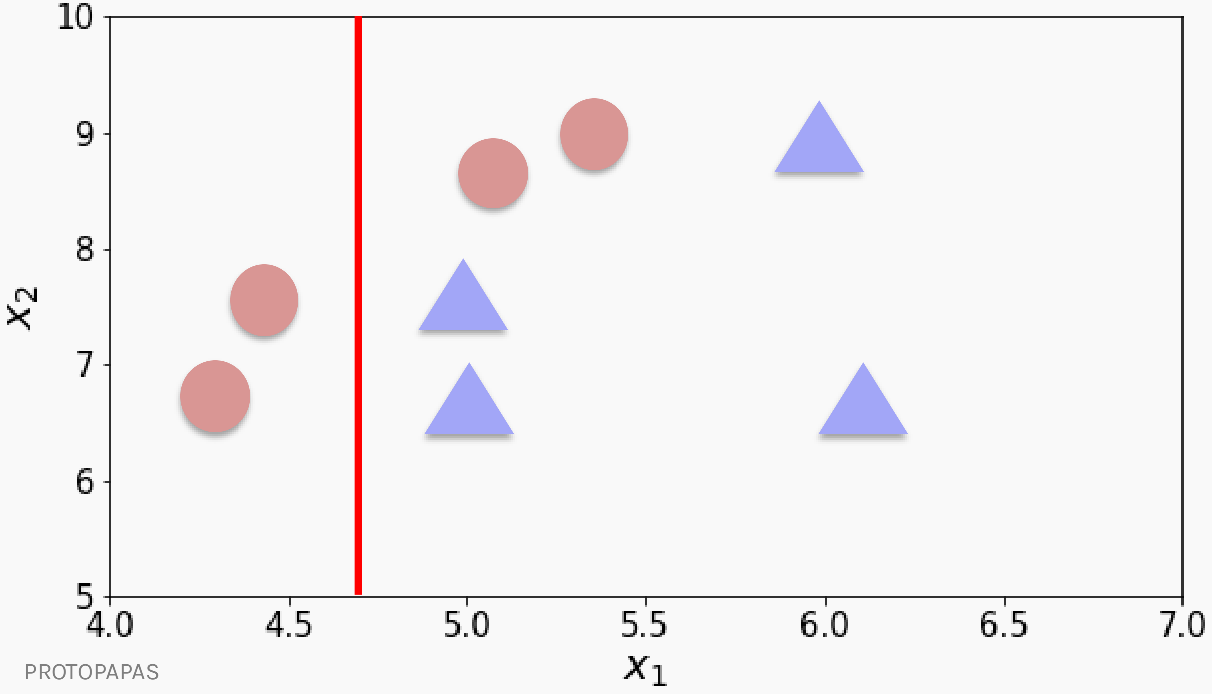
AdaBoost

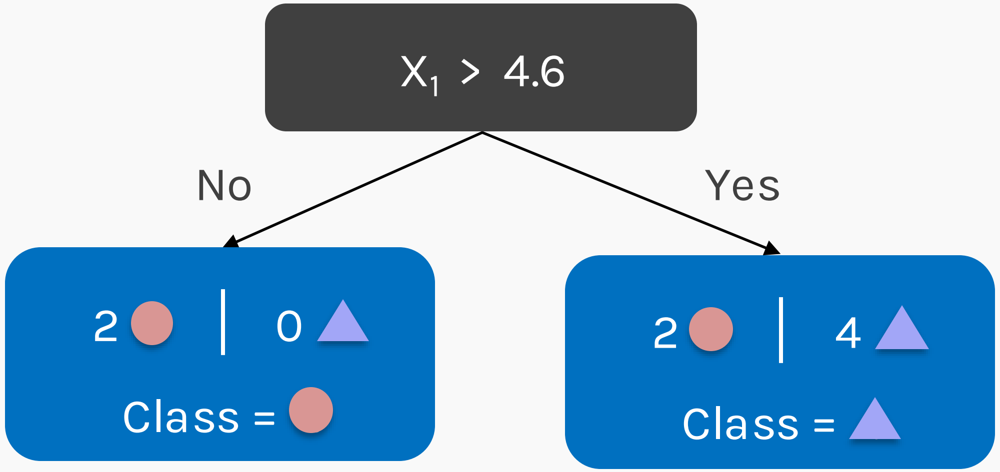
Consider the following dataset:



AdaBoost

Step 1: Fit a stump $S^{(0)}$ on the dataset.



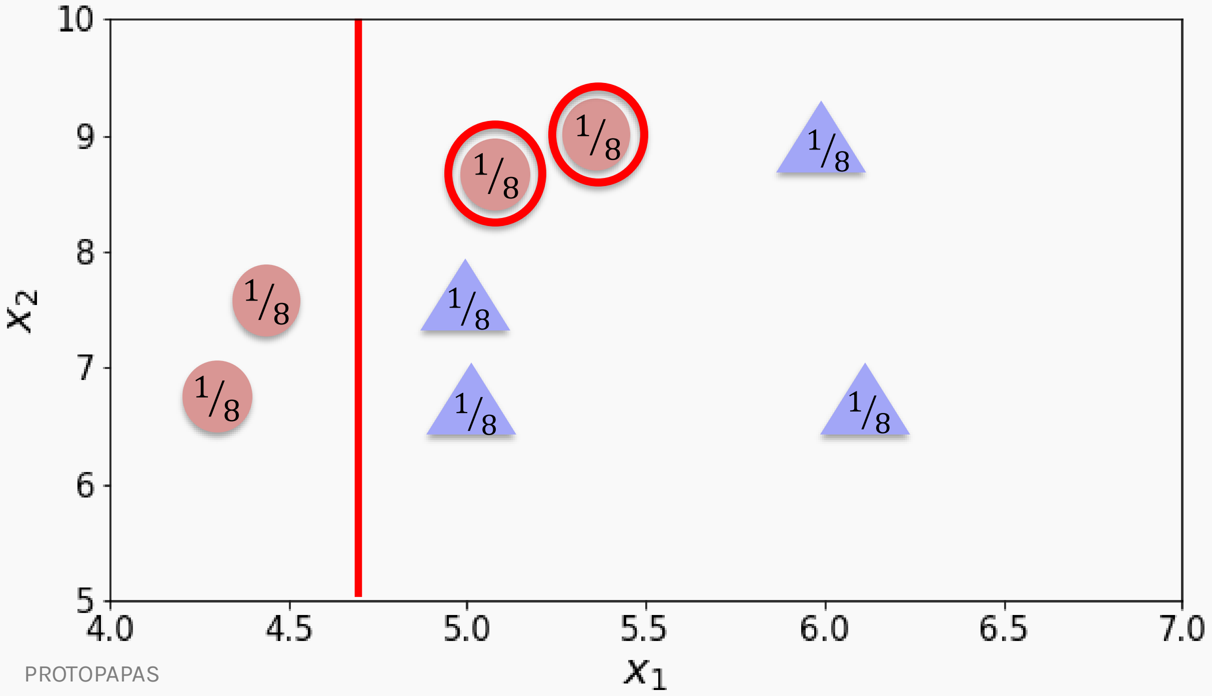


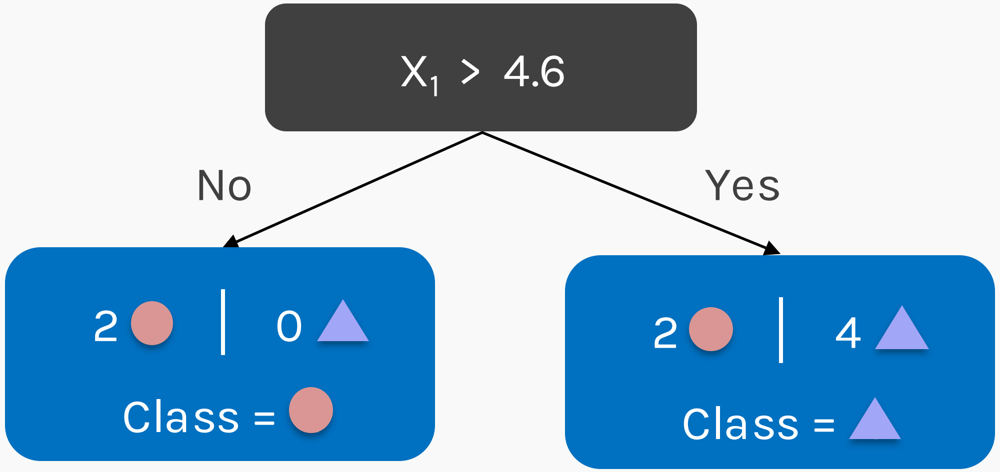
AdaBoost

Step 2: Assume we initialize each data point to have equal weight of $\frac{1}{N}$.

Calculate the total error in the stump using:

$$\epsilon^{(0)} = \sum_{n=1}^{N} w_n^{(0)} \mathbb{I}\,(y_n \neq \; S^{(0)}(x_n))$$





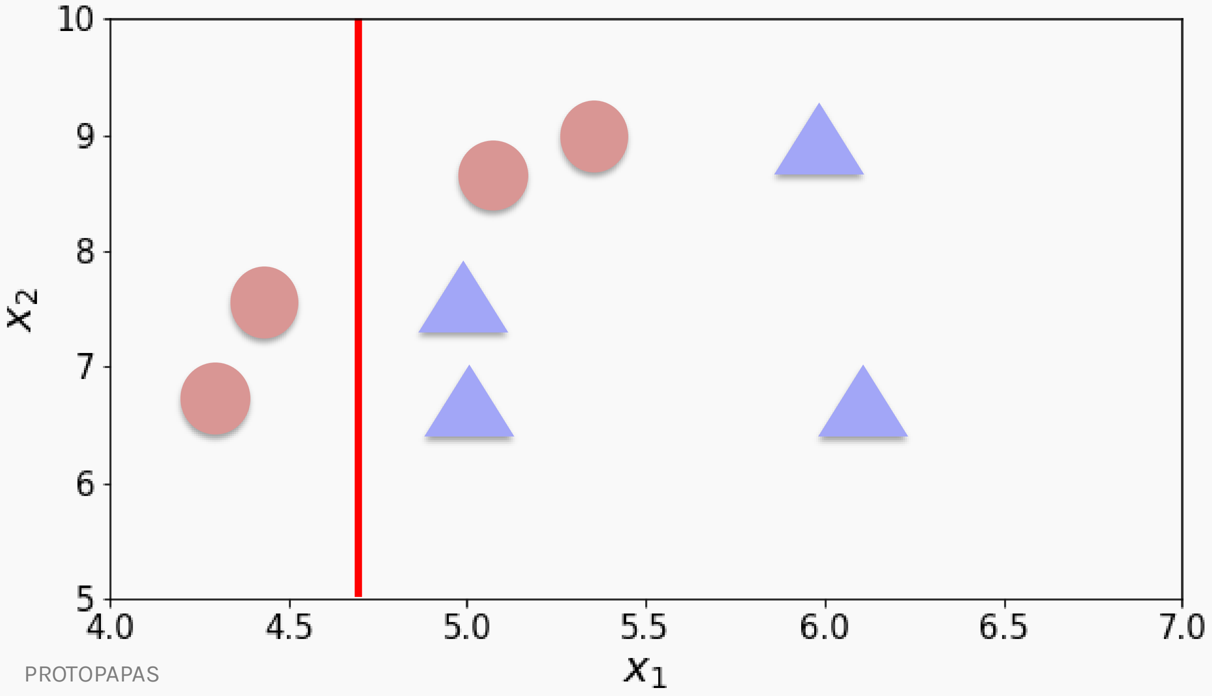
$$Error\ (\epsilon) = \frac{1}{8} + \frac{1}{8} = \frac{2}{8}$$

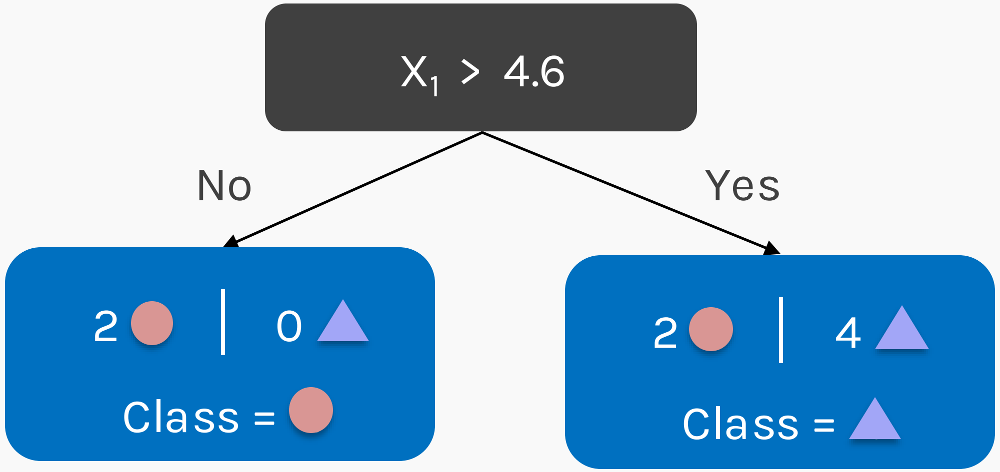
AdaBoost

Step 3: Now that the first weak learner has been built, we will assign the stump a scaling factor, $\lambda^{(0)}$, that indicates how much it **contributes to the entire ensemble**.

We assign λ to each successive stump as it offers some flexibility, and we can give more importance to stumps that perform better. Let this model's weight λ be 0.25.

$$\lambda^{(0)} = 0.25$$



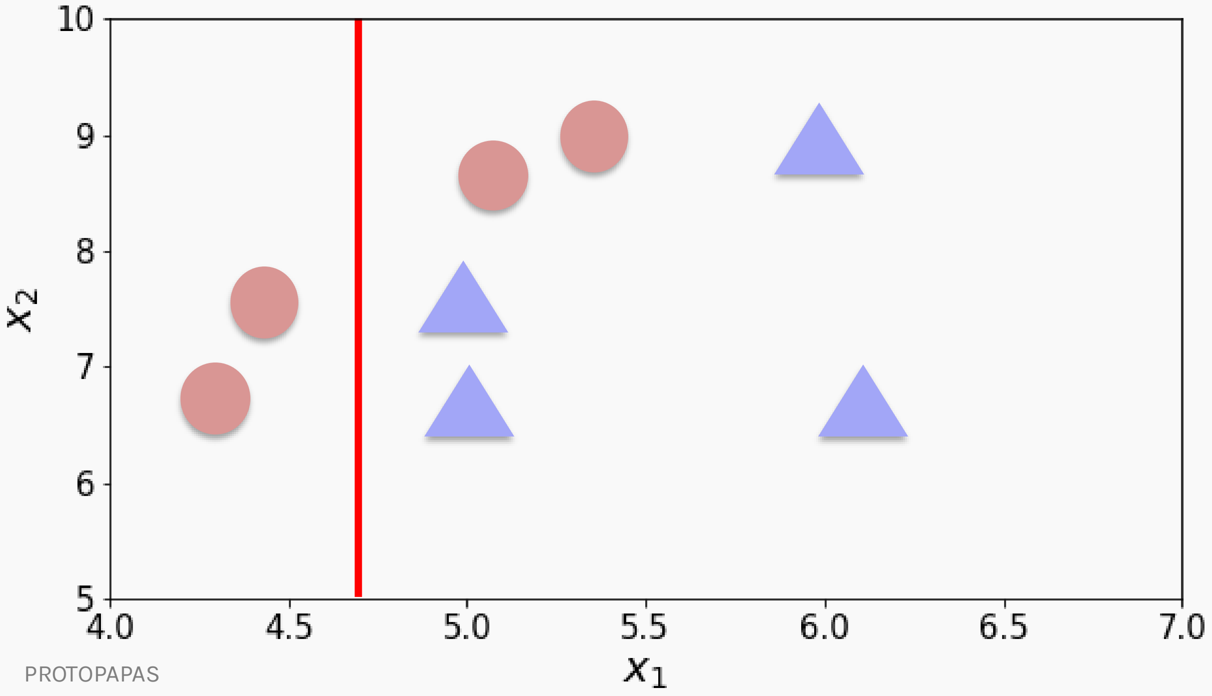


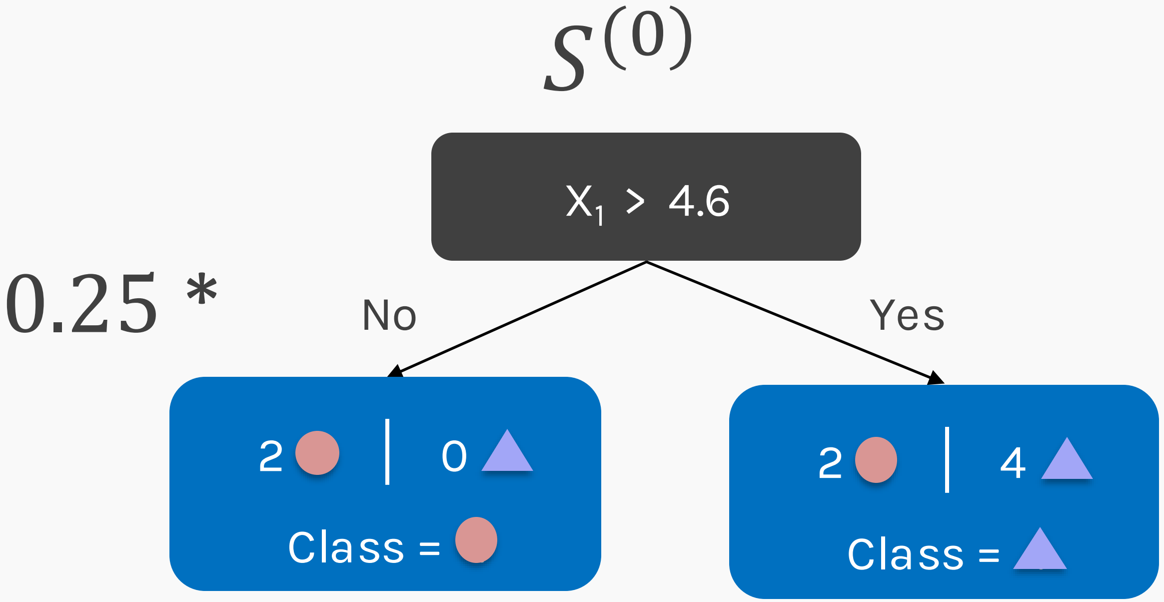
$$Error\ (\epsilon) = \frac{2}{8}$$

AdaBoost

Step 4: Construct the **ensemble model** $T^{(0)}$ using:

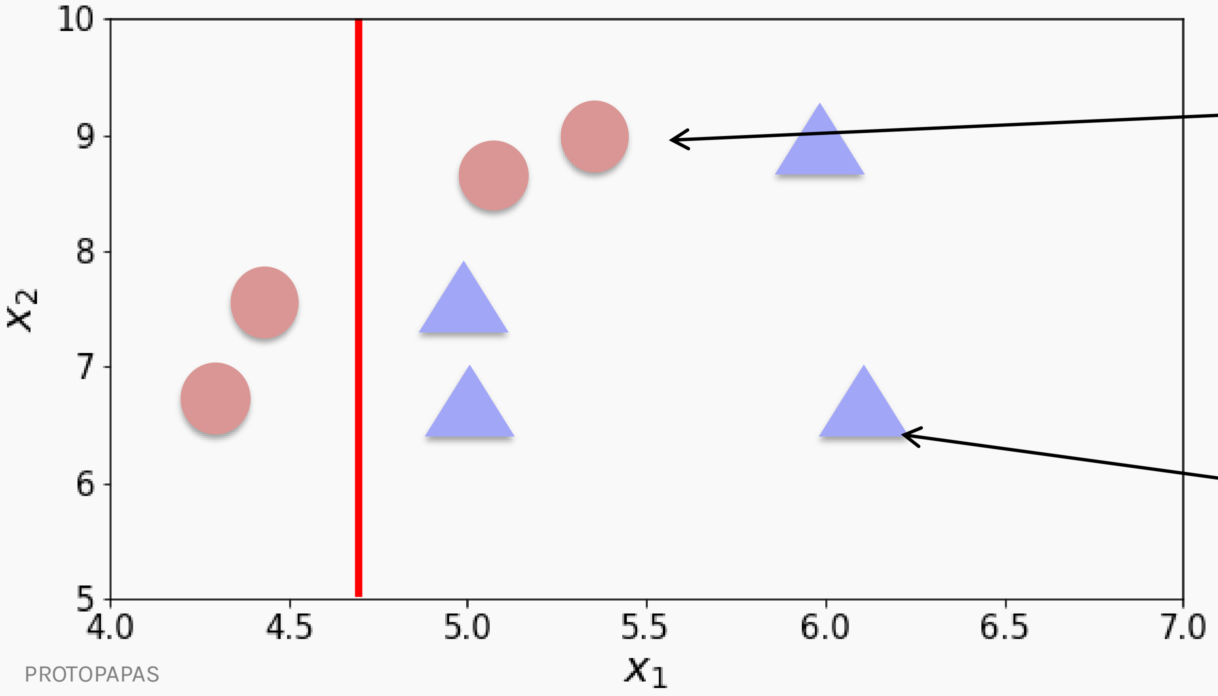
$$T^{(i)} \leftarrow sign\begin{cases} \lambda^{(i)}S^{(i)} & i = 0 \\ T^{(i-1)} + \lambda^{(i)}S^{(i)} & i = 1, 2, \ldots \end{cases}$$





AdaBoost

Step 5: Adjust the weights assigned to each data point to ensure the next stump focuses on the points **misclassified by the previous stump**.



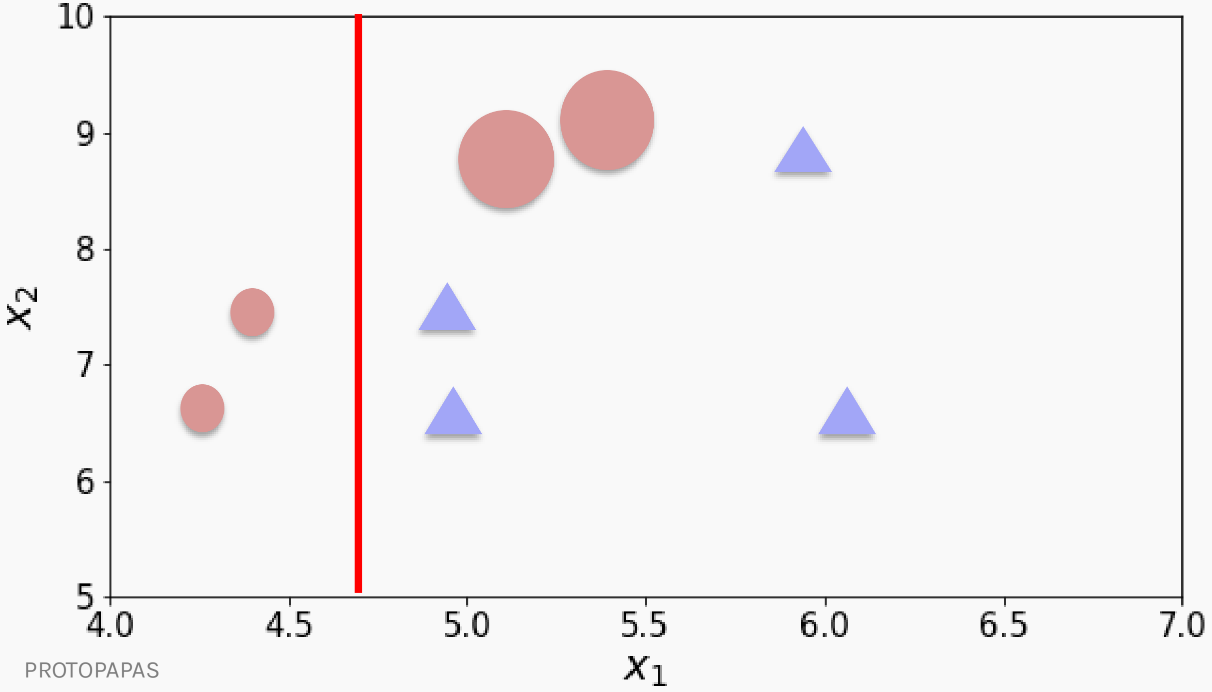
Increase weights of misclassified samples

Decrease weights of correctly classified samples

AdaBoost

Step 5: Adjust the weights assigned to each data point to ensure the next stump focuses on the points **misclassified by the previous stump**.

$$w_n^{(1)} \leftarrow \frac{w_n^{(0)} e^{-\lambda^{(0)} \, y_n \, s^{(0)}(x_n)}}{Z}$$



AdaBoost: weight update

ground truth -1 or 1

prediction from current stump

learning rate

$$w_n^{(1)} \leftarrow \frac{w_n^{(0)} e^{-\lambda^{(0)} y_n s^{(0)}(x_n)}}{Z}$$

new weights

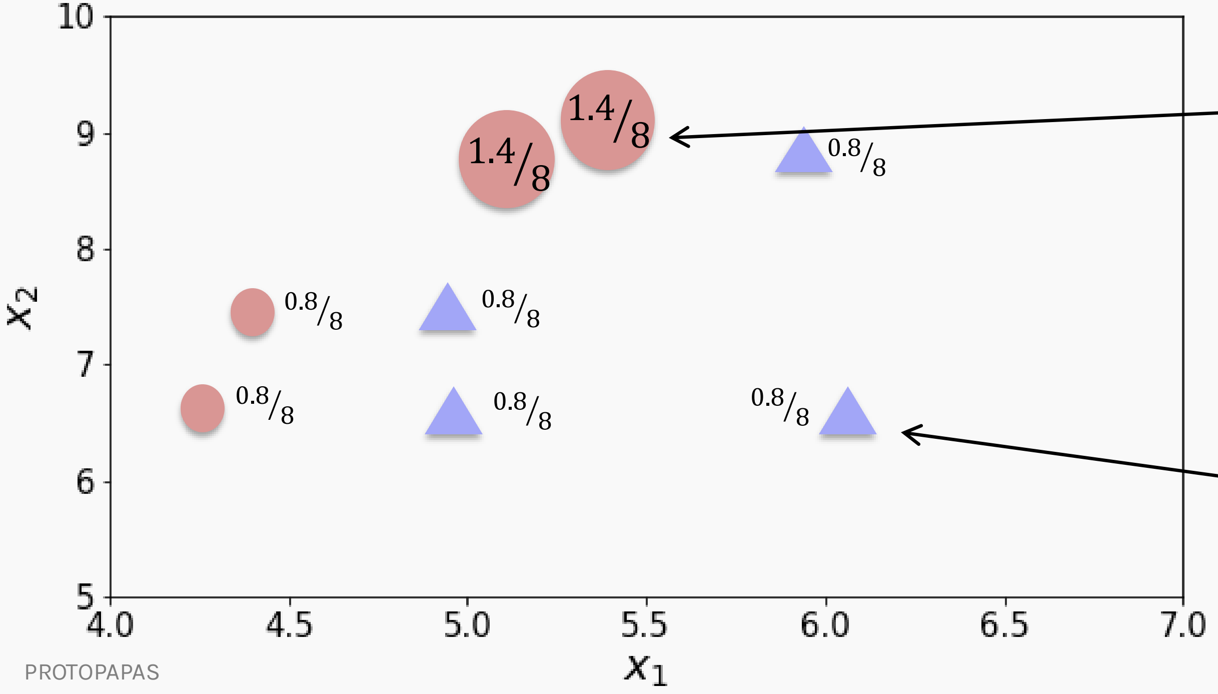
current weights

normalizes the new weights

AdaBoost

Step 5: Adjust the weights assigned to each data point to ensure the next stump focuses on the points **misclassified by the previous stump**.

$$w_n^{(1)} \leftarrow \frac{w_n^{(0)} e^{-\lambda^{(0)} y_n s^{(0)}(x_n)}}{Z} = \frac{w_n^{(1')}}{Z}$$



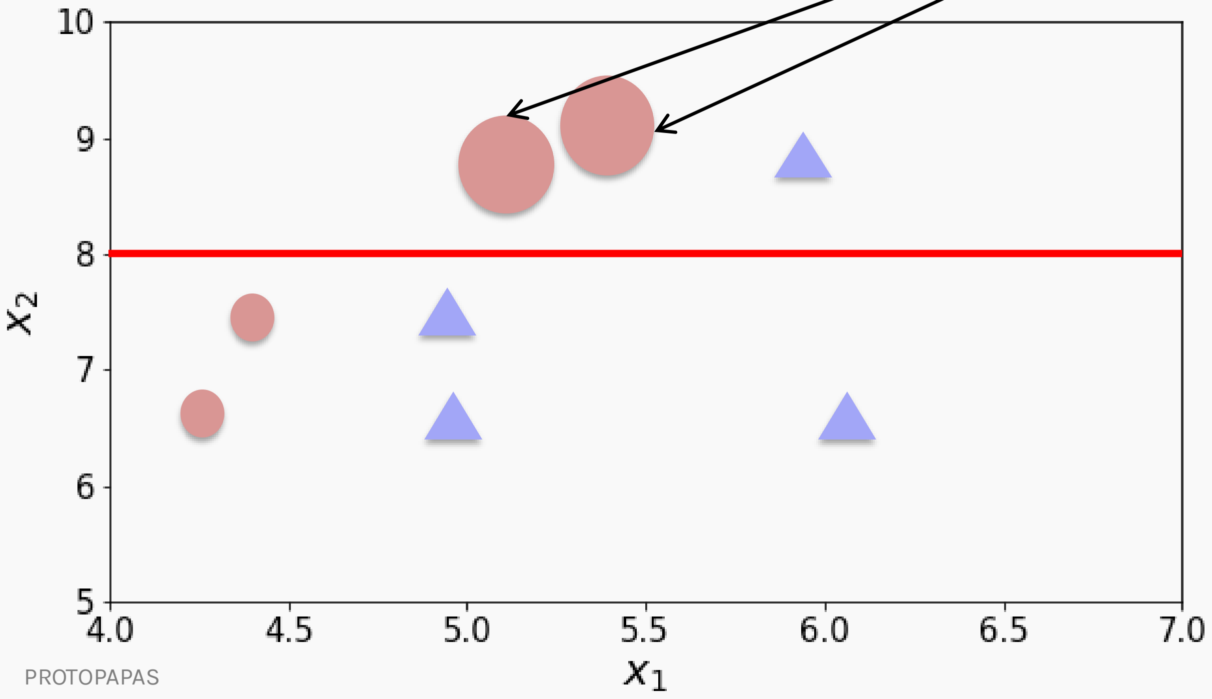
$$w_i^{(1)} \leftarrow \frac{w_i^{(1')}}{Z} \approx \frac{\frac{1.3}{8}}{\frac{1.3}{8} * 2 + \frac{0.8}{8} * 6} \approx \frac{1.4}{8}$$

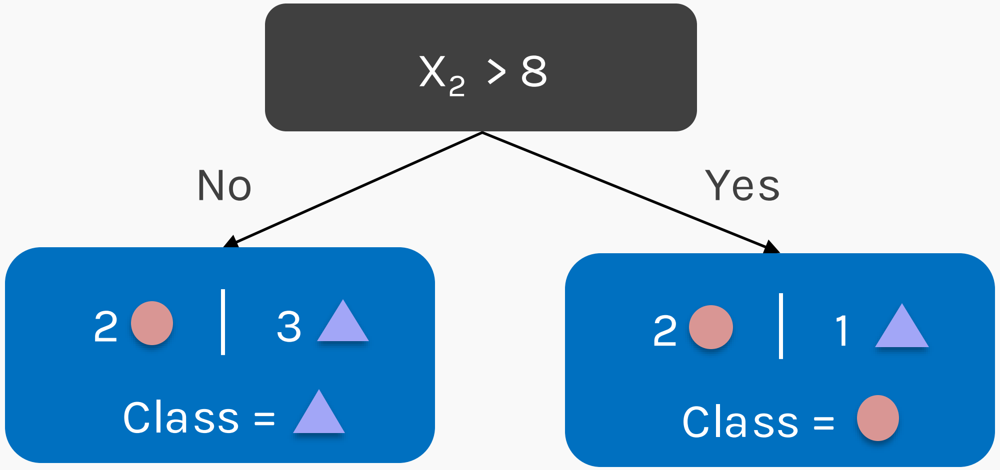
$$w_j^{(1)} \leftarrow \frac{w_j^{(1')}}{Z} \approx \frac{\frac{0.8}{8}}{\frac{1.3}{8} * 2 + \frac{0.8}{8} * 6} \approx \frac{0.8}{8}$$

AdaBoost

Step 6: Create another stump $S^{(1)}$ on the re-weighted data.

Notice that the $S^{(1)}$ has correctly classified the data points that $S^{(0)}$ misclassified.

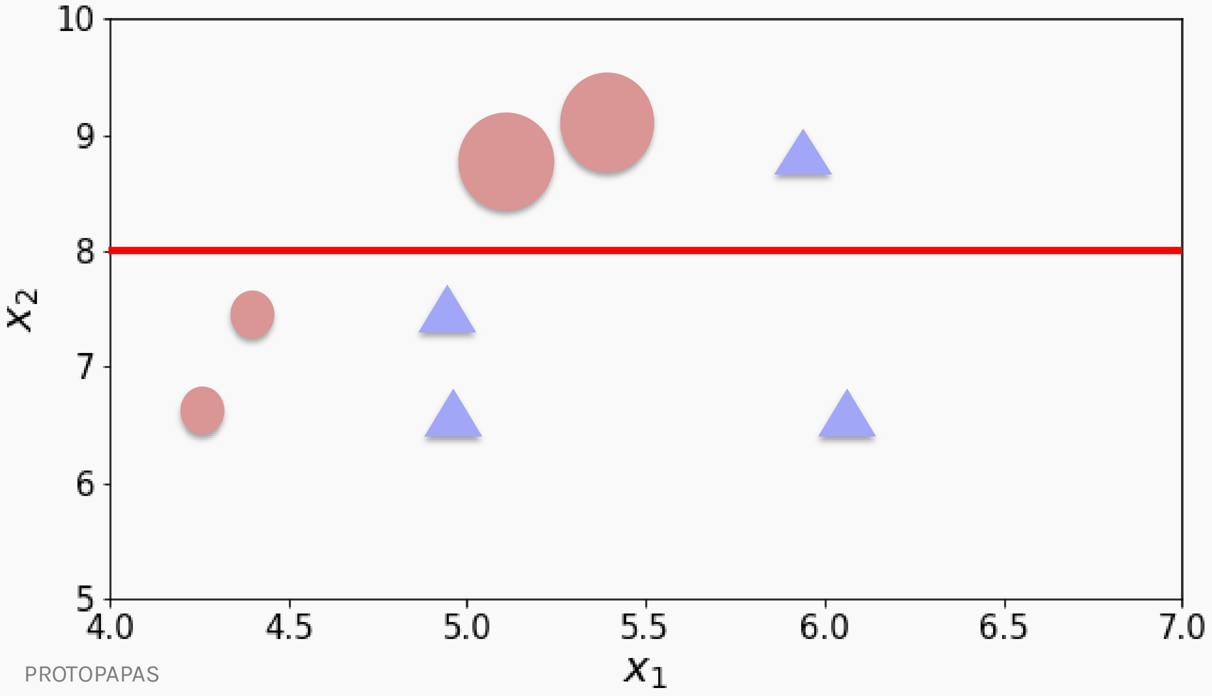


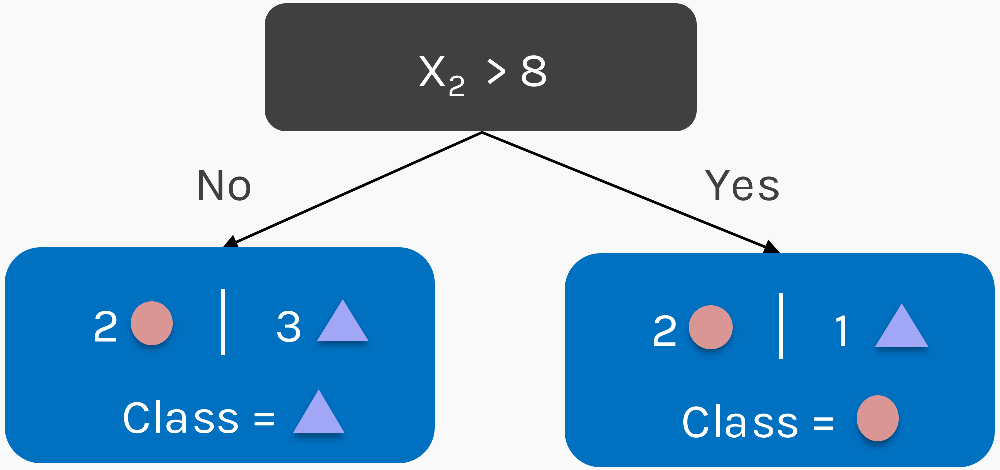


AdaBoost

Step 7: With the new weights, calculate the total error in the stump using:

$$\epsilon^{(1)} = \sum_{n=1}^{N} w_n^{(1)} \mathbb{I}\,(\,y_n \neq\; S^{(1)}(x_n))$$



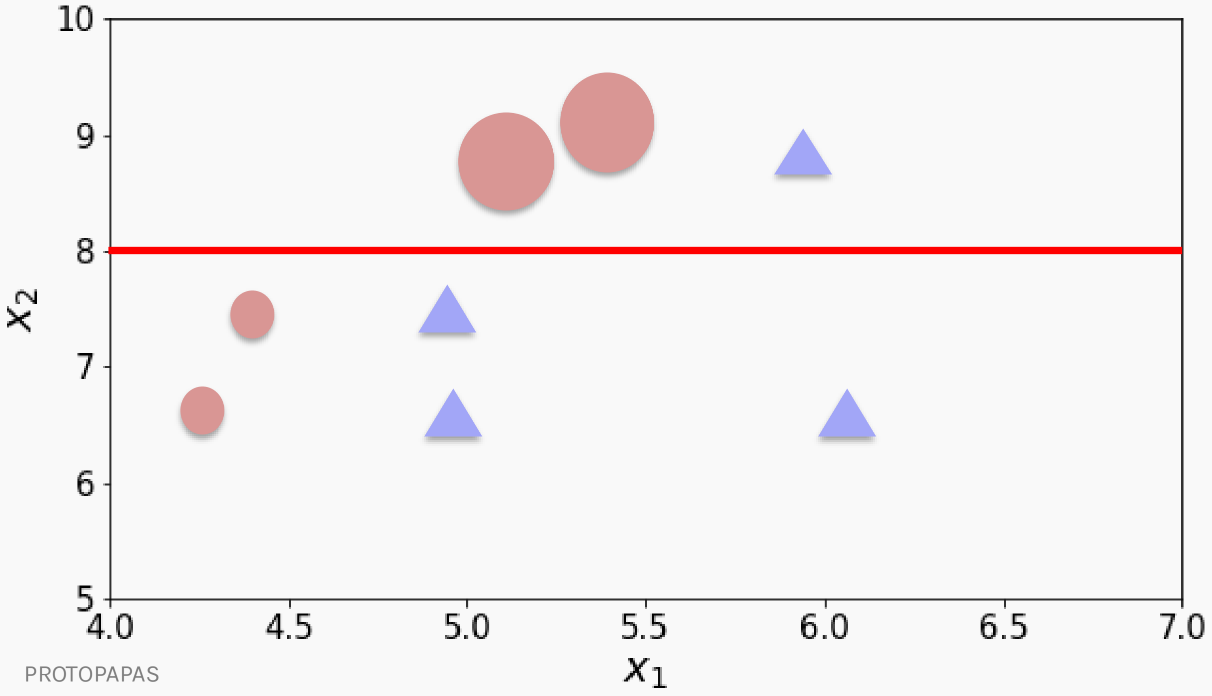


$$Error\;(\epsilon) = \frac{1.4}{8} + \frac{1.4}{8} + \frac{0.8}{8} = 0.45$$

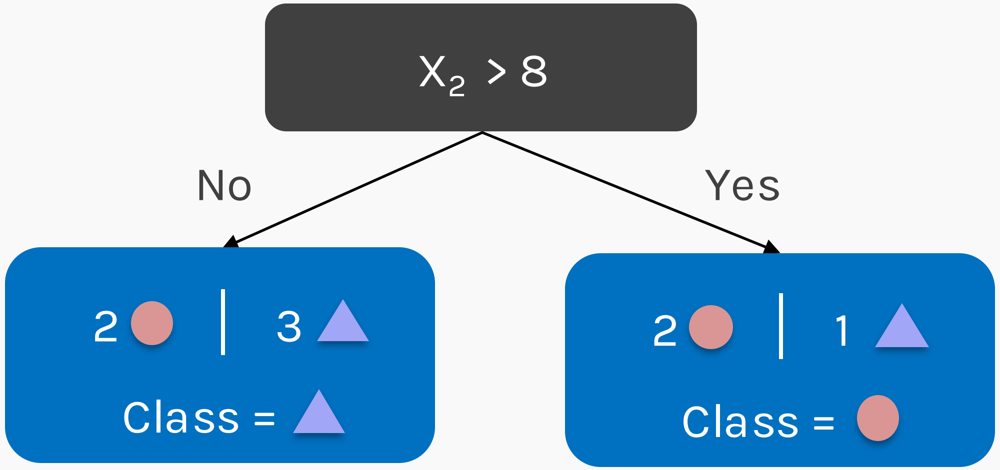
AdaBoost

Step 8: Assign the stump a scale, $\lambda^{(1)}$, that indicates how much it **contributes to the entire ensemble.**

Let this model weight $\lambda^{(1)}$ be 0.20.



$$\lambda^{(1)} = 0.20$$

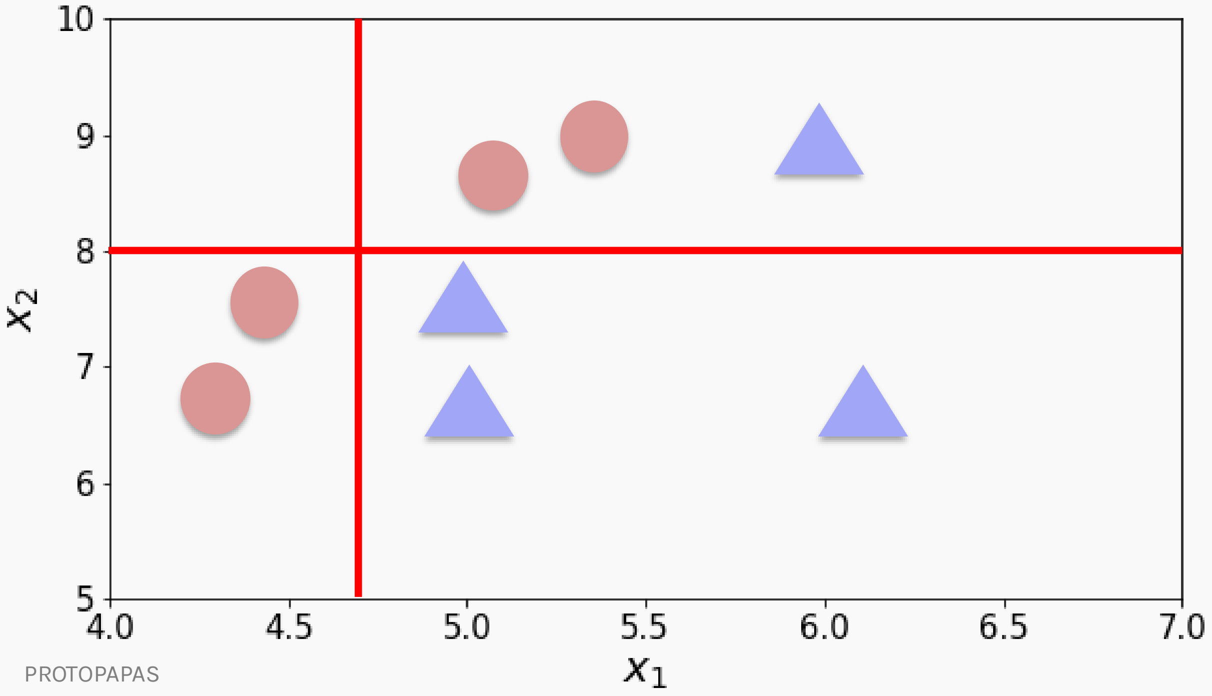


$$Error\ (\epsilon) = 0.45$$

AdaBoost

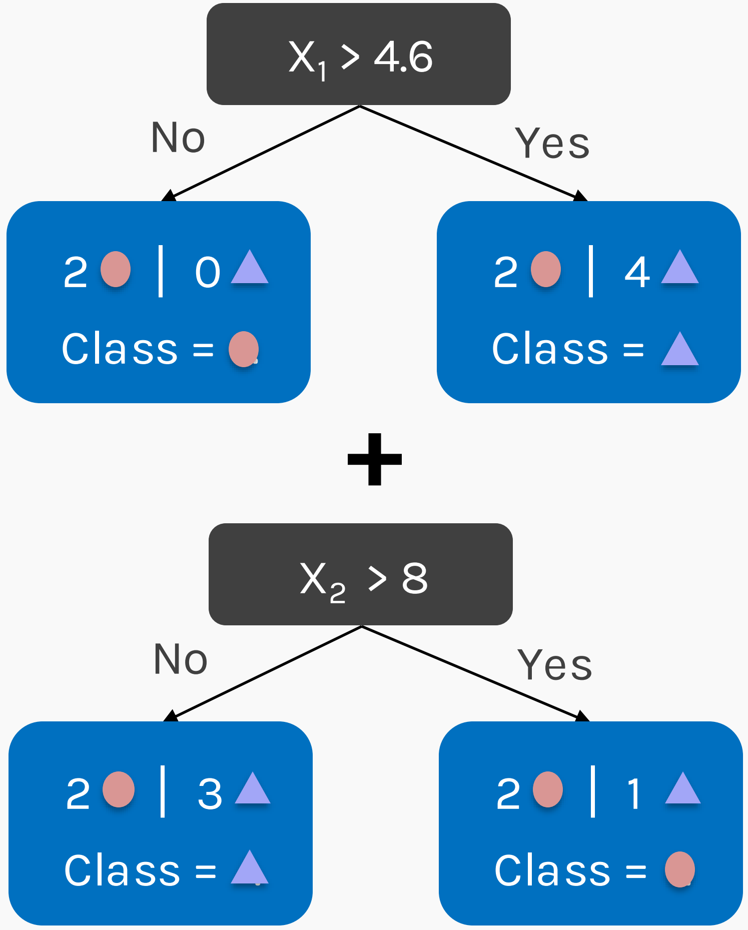
Step 9: Construct the **ensemble model** $T^{(1)}$ using:

$$T^{(i)} \leftarrow sign\begin{cases} \lambda^{(i)}S^{(i)} & i = 0 \\ T^{(i-1)} + \lambda^{(i)}S^{(i)} & i = 1, 2, \ldots \end{cases}$$



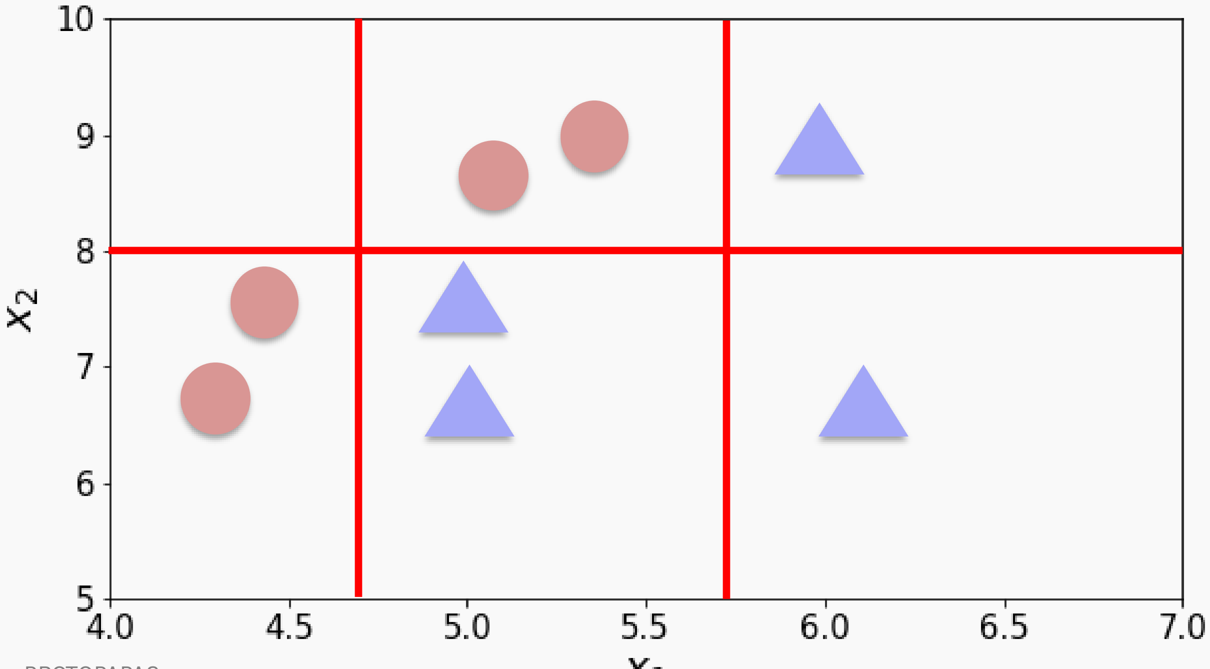
$0.25 *$

$0.20 *$



AdaBoost

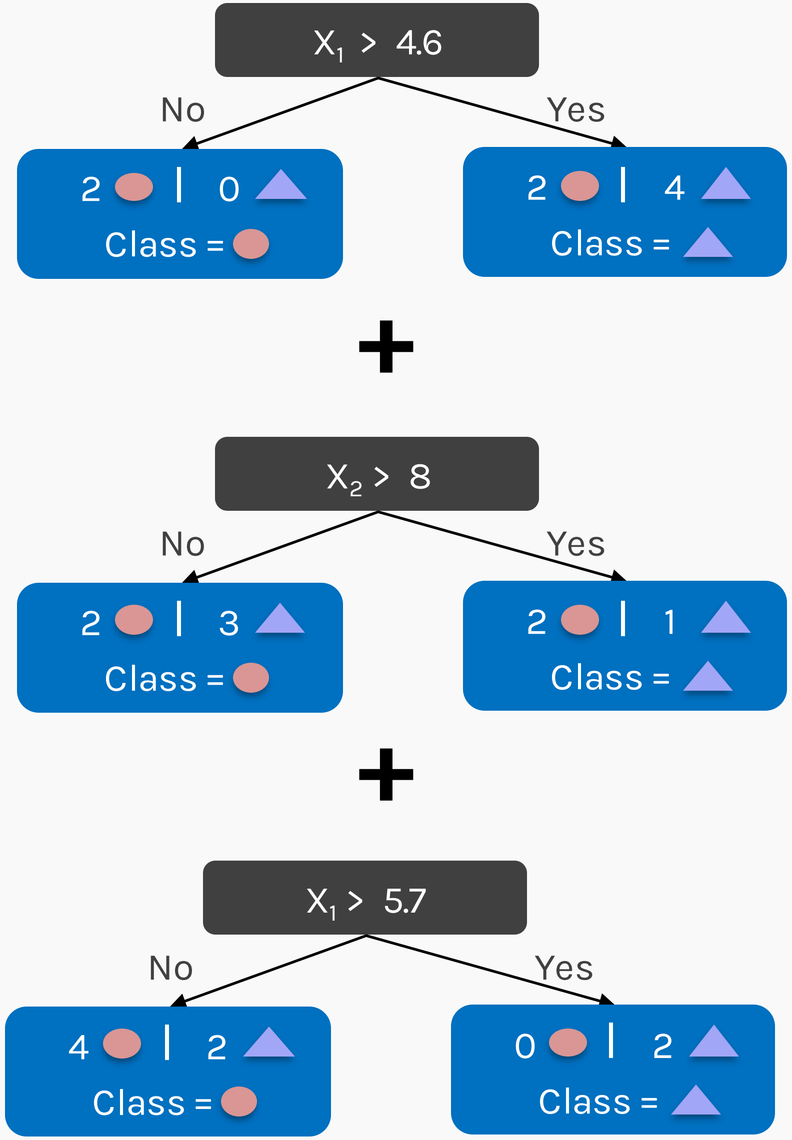
Repeating the same process again, we get:



0.25 *

0.20 *

0.42 *



AdaBoost

Step 1: Given training data $\{(x_1, y_1), \dots, (x_N, y_N)\}$, choose an initial distribution $w_n^{(0)} = 1/N$.

For $i = 0 \dots$ until stopping condition is met:

Step 2: Train a weak learner $S^{(i)}$ using weights $w_n^{(i)}$.

Step 3: Calculate the total error of the weak learner using:

$$\epsilon^{(i)} = \sum_{n=1}^{N} w_n^{(i)} \, \mathbb{I}\,(\, y_n \neq S^{(i)}(x_n))$$

Step 4: Calculate the importance of each stump, $\lambda^{(i)}$.

Step 5: Construct the ensemble model using:

$$T^{(i)} \leftarrow sign \begin{cases} \lambda^{(i)} S^{(i)} & i = 0 \\ T^{(i-1)} + \lambda^{(i)} S^{(i)} & i = 1, 2, \dots \end{cases}$$

Step 6: Adjust the weights assigned to each data point to ensure the next stump focuses on the points misclassified by the previous stump

$$w_n^{(i+1)} \leftarrow \frac{w_n^{(i)} \, e^{-\lambda^{(i)} \, y_n \, S^{(i)}(x_n)}}{Z}$$

Final model: $T^{(i)}(x) = sign\left[\sum_{i=1}^{M} \lambda^{(i)} \, S^{(i)}(x)\right]$

AdaBoost

Step 1: Given training data $\{(x_1, y_1), \dots, (x_N, y_N)\}$, choose an initial distribution $w_n^{(0)} = 1/N$.

For $i = 0 \dots$ until stopping condition is met:

Step 2: Train a weak learner $S^{(i)}$ using weights $w_n^{(i)}$.

Step 3: Calculate the total error of the weak learner using:

$$\epsilon^{(i)} = \sum_{n=1}^{N} w_n^{(i)} \, \mathbb{I}\,(\, y_n \neq S^{(i)}(x_n))$$

Step 4: Calculate the importance of each stump, $\lambda^{(i)}$.

Step 5: Construct the ensemble model using:

$$T^{(i)} \leftarrow sign \begin{cases} \lambda^{(i)} S^{(i)} & i = 0 \\ T^{(i-1)} + \lambda^{(i)} S^{(i)} & i = 1, 2, \dots \end{cases}$$

Step 6: Adjust the weights assigned to each data point to ensure the next stump focuses on the points misclassified by the previous stump

$$w_n^{(i+1)} \leftarrow \frac{w_n^{(i)} \, e^{-\lambda^{(i)} \, y_n \, S^{(i)}(x_n)}}{Z}$$

Final model: $T^{(i)}(x) = sign\left[\sum_{i=1}^{T} \lambda^{(i)} \, S^{(i)}(x)\right]$

AdaBoost

In gradient boosting for regression, we minimize the MSE loss.

In AdaBoost, the function we choose is called **exponential loss:**

$$Exp\ Loss = \frac{1}{N}\sum_{n=1}^{N} e^{(-y_n\hat{y}_n)} \quad \text{where} \quad y_n \in \{-1, 1\}$$

Exponential loss is differentiable with respect to $\hat{y}_n$ and it is an upper bound of Error.

Choosing the Learning Rate

Unlike in the case of gradient boosting for regression, we can analytically solve for the optimal learning rate for AdaBoost, by optimizing:

$$\underset{\lambda}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} e^{(-y_n(T+\lambda^{(i)} S^{(i)}(x_n)))}$$

Doing so, we get: $\lambda^{(i)} = \frac{1}{2} \ln\left(\frac{1 - \epsilon}{\epsilon}\right)$ where

$\epsilon = \sum_{n=1}^{N} w_n^{(i)} \, \mathbb{I}\,(\, y_n \neq T^{(i)}(x_n))$ and $w_n^{(i+1)} \leftarrow \frac{w_n^{(i)} \, e^{-\lambda^{(i)} \, y_n \, S^{(i)}(x_n)}}{Z}$

AdaBoost

Step 1: Given training data $\{(x_1, y_1), \dots, (x_N, y_N)\}$, choose an initial distribution $w_n^{(0)} = 1/N$.

For $i = 0 \dots$ until stopping condition is met:

Step 2: Train a weak learner $S^{(i)}$ using weights $w_n^{(i)}$.

Step 3: Calculate the total error of the weak learner using: $\epsilon^{(i)} = \sum_{n=1}^{N} w_n^{(i)} \, \mathbb{I}\,(\, y_n \neq S^{(i)}(x_n))$

Step 4: Calculate the scaling factor of each stump, $\lambda^{(i)}$:$\lambda^{(i)} = \frac{1}{2}\ln\left(\frac{1-\epsilon}{\epsilon}\right)$

Step 5: Construct the ensemble model using:

$$T^{(i)} \leftarrow sign \begin{cases} \lambda^{(i)} S^{(i)} & i = 0 \\ T^{(i-1)} + \lambda^{(i)} S^{(i)} & i = 1, 2, \dots \end{cases}$$

Step 6: Adjust the weights assigned to each data point to ensure the next stump focuses on the points misclassified by the previous stump

$$w_n^{(i+1)} \leftarrow \frac{w_n^{(i)} \, e^{-\lambda^{(i)} \, y_n \, S^{(i)}(x_n)}}{Z}$$

Final model: $T^{(i)}(x) = sign\left[\sum_{i=1}^{T} \lambda^{(i)} \, S^{(i)}(x)\right]$

AdaBoost – hyperparameters