Boosting

- Simple / weak classifiers are good!
  - Logistic regression - w. simple features.
  - Shallow decision trees.
  - Decision stumps.
- They have low variance and hence "Learning is fast!" , but tend to have high bias.

A weaker learner has higher error and inorder to train the model well 'need stronger learner'. ¶

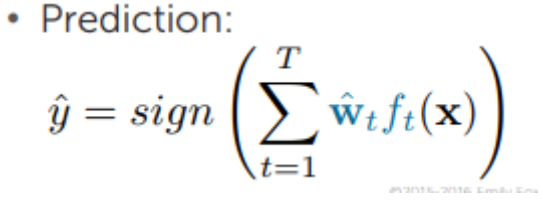
- This is can be achieved
  - Add more features (logistic regression, etc) or depth (decision trees, etc);
  - Boosting.

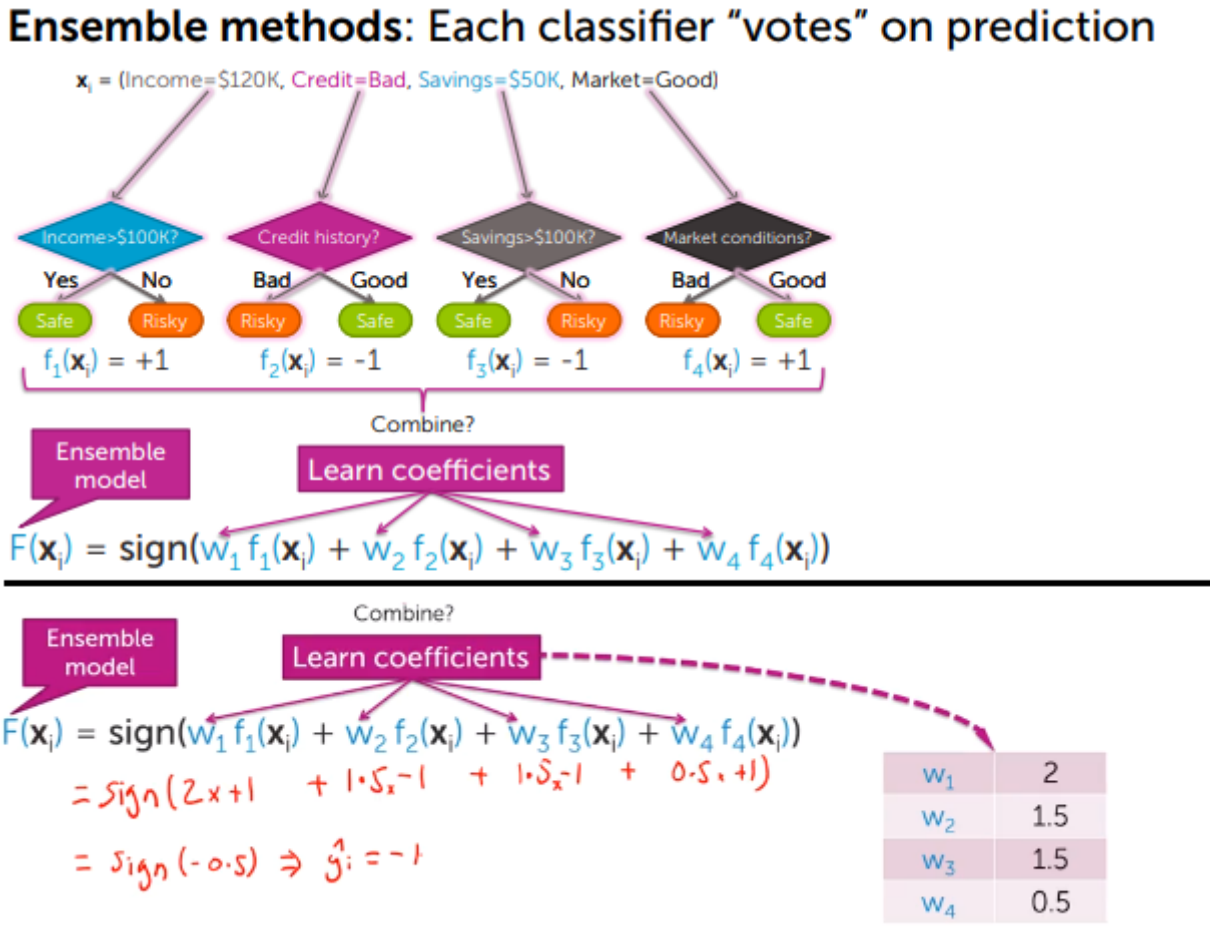
Boosting:

- Can a set of weak learners be combined to create a stronger learner?
  - **YES**.
- Amazing impact
  - Simple approach
  - widely used in industry
  - wins Kaggle competitions.

Ensemble classifiers

- It is a combination of multiple weak classifiers combined together.
- The output y-hat = f(x) -> Classifier.
- Goal:
  - Predict output y
    - Either +1 or -1
  - From input x
- Learn ensemble model:
  - Classifiers: f1(x), f2(x), ..., fT(x)
  - Coefficients : w-hat1, w-hat2, ..., w-hatT
- Prediction:





Boosting

Training a classifier

- **Training data ---> Learn classifier --f(x)-> Predict (y-hat=sign(f(x)))**
- In many situations the decision-trees will fail to deliever the appropriate results.
  - In case both the branches of a split features predict the same value as the result due to the majority vote of the data-points.
  - Although it is right from the decision-tree learning algorithm perspective.
  - It is not right from the data point of view. If the result of both the splits provide the same predicted value.

```
                        Income
                          |
                          |
--------- > $100K -------------- <= $100K ----------
       ( 3 safe, 1 risky)          (4 safe, 3 risky)
```

Predicted Value SAFE SAFE

- Boosting focus learning on hard points.
- It evaluates the model and learns from the mistake of f(x). **Focus next classifier on places where f(x) does less well**.
- More weight is places on "hard" or more important points.
  - Weighted dataset:

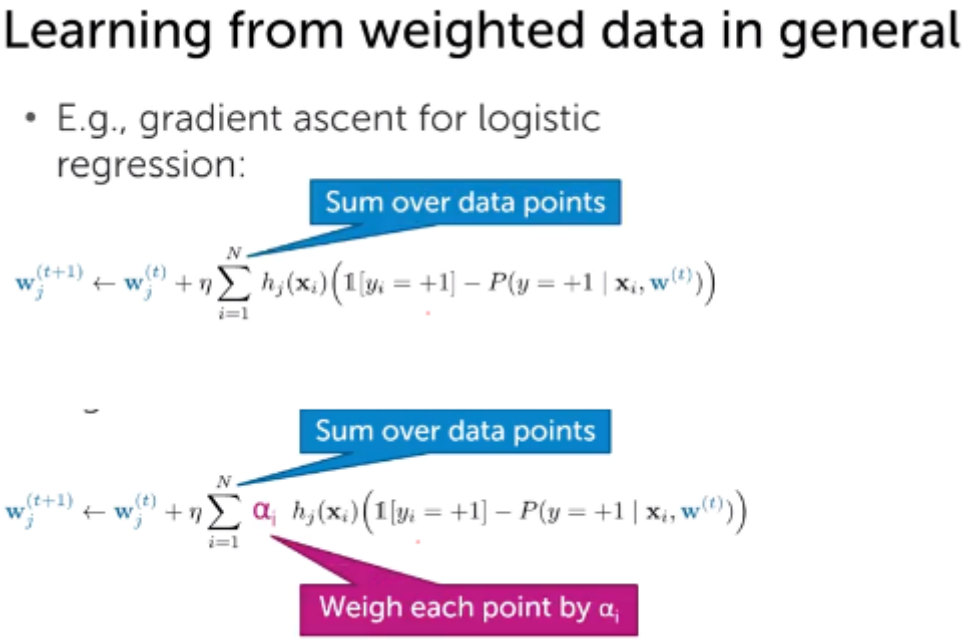
    - Each xi, yi weighted by αi
        - More important point has higher weight αi.
- Learning:
    - Data point j counts as αi data points.
        - E.g . αi = 2 -> count point twice.

Learning a decision stump on weighted data.

- Rather than predicting the label of a split branch based on the popularity or the number of occurence of the value.
- Each row has a **weight α** associated with it.(Increase weight α of harder/misclassified points)
    - Based on the weight the predicted label is obtained.

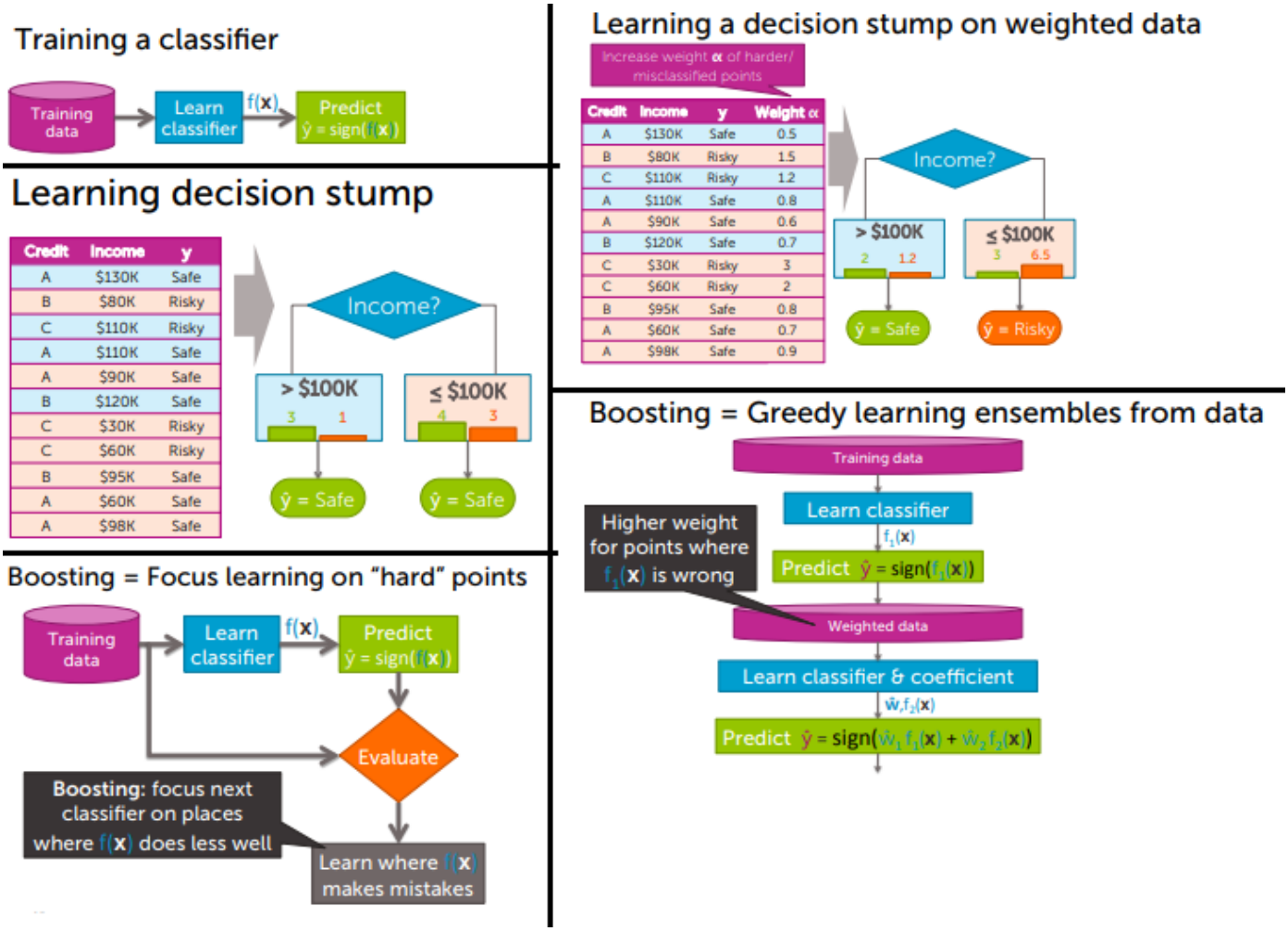
Learning from weighted data in general.

- Adding weighted coefficients is not just applicable to decision trees. It is a more general concept.
- It can be associated with logistic regression as well.



Boosting -> Greedy learning ensembles from data

- **Training data -> Learning classifier -f(x) -> Predicy y-hat = sign(f(x))**
- Create a weighted data (Higher weight for points where f1(x) is wrong).
- **Weighted data -> Learn classifier & coefficients -> w-hat * f2(x) -> Predict y-hat = sign(w-hat1 f1(x) + w-hat2 f2(x))**.



AdaBoost : Learning Ensemble

Inception 1999

Algorithm

- Step 1 - Start same weight for all points: **αi = 1/N**.
- For each iteration of the classier fi(x)
  - Step 2 - Learn ft(x) with data weights αi.
  - Step 3 - Compute coefficients w-hat(t).
  - Step 4 - Recompute weights αi.
- Step N - Final model predicts:
  - y-hat = sign (Σ(t = 1...T) w-hat . ft(x))

Problems:

- Problem 1 - Computing coefficients w-hat(t).
  - How must to trust the classifier ft(x)?
- Problem 2 - Recomputing weight αi.
  - How to weigh mistakes more?

Problem 1 : Computing coefficient w-hat of classifier ft(x)

- If the classifier ft(x) is good -> then w-hat large , else small.
- ft(x) is good based on the training error.
- **Training error** of the classifier is **low**

    - ft(x) classifier is good
        - **w-hat is large (coefficients)**
- **Training error** of the classifier is **high**
    - ft(x) classifier is poor
        - **w-hat is small (coefficients)**

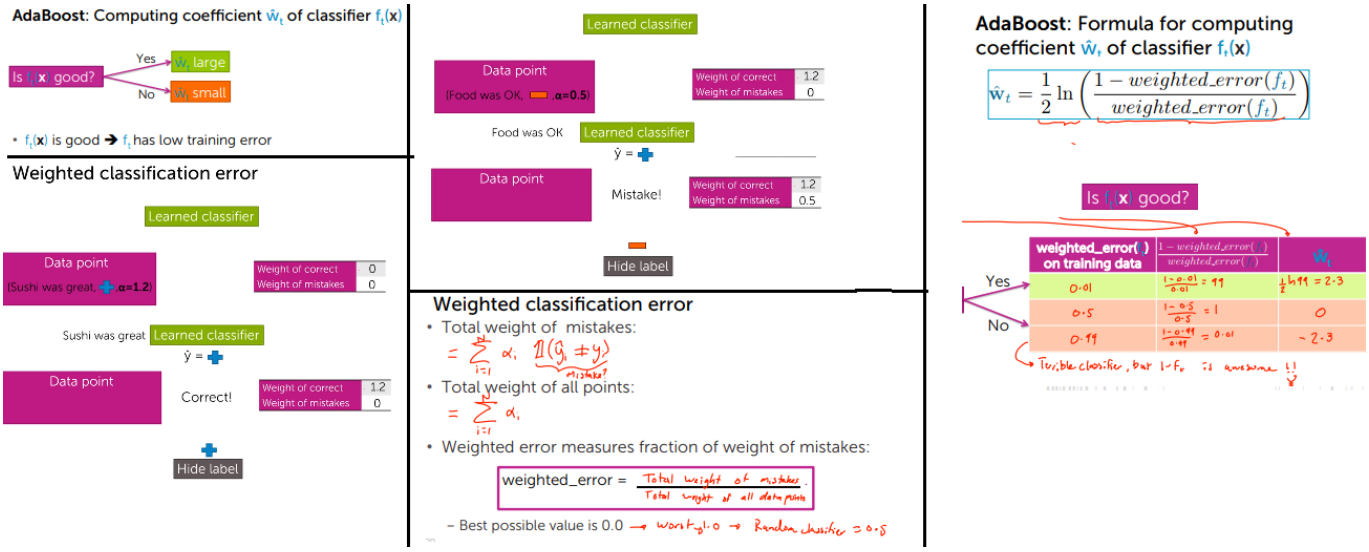
Measuring the weighted classification error.

- It is similar to measuring normal data, without weight.
- Example 1 : Datapoint (Sushi was awesome, + , 1.2)
    - It has the review, sentiment, weight α.
    - The learned classifier is feed with the review while the sentiment is hidden. Then the predicted sentiment is compared to the true label.
        - In case it is a match then it is correct.
        - Else it is a mistake.
            - In this manner the weights of the mistakes are calculated.
- Weighted error meaures fraction of weight of mistakes.
    - Best = 0.0, Worst = 1.0; Random = 0.5;

Formula for computing coefficient w-hat(t) of classifier ft(x).

- W-hat(t) = 1/2 ln ((1 - weighted_error(ft)) / weighted_error(ft))

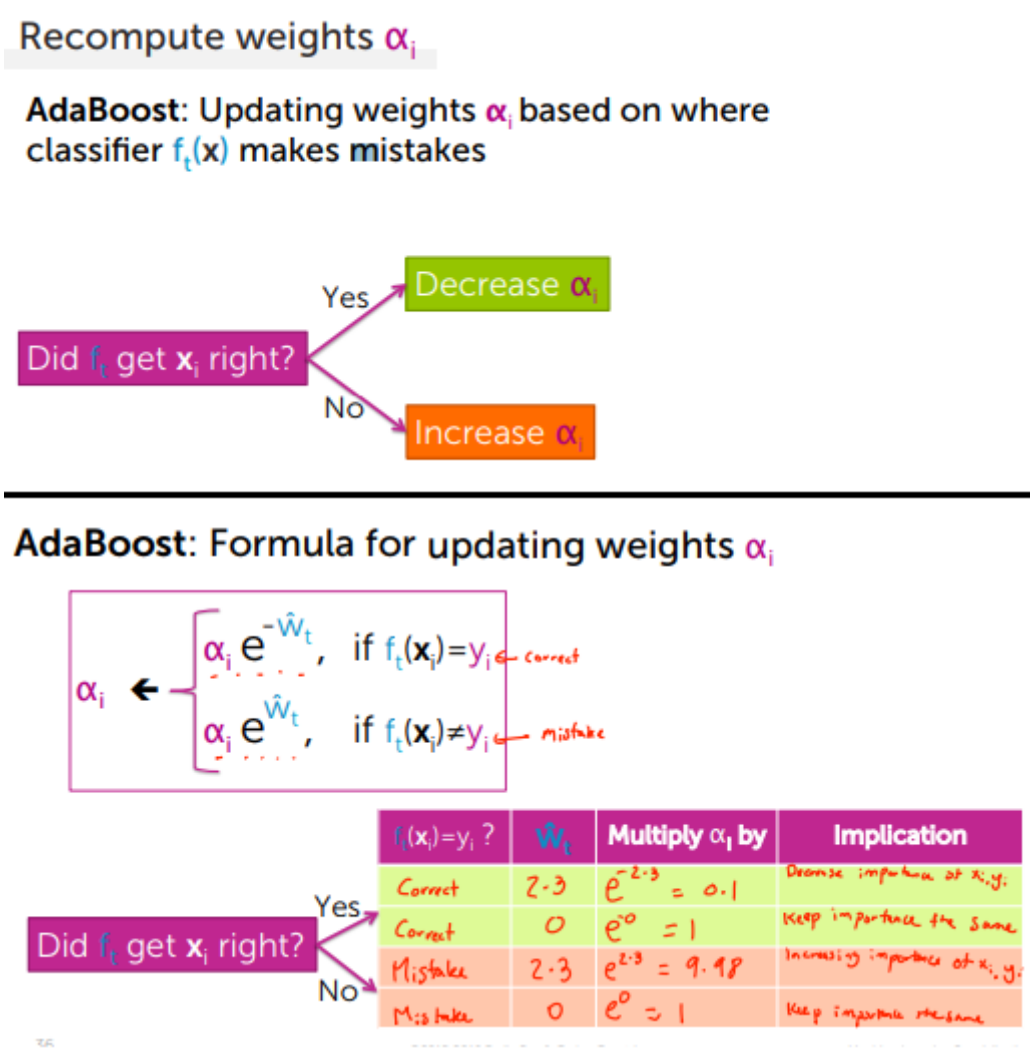
- If the classifier ft(x) is good - Then training error is low , and coefficients weight is high.
- If the classifier ft(x) is random - Then training error is random (0.5) then the coefficients will bear no significance, weight of coefficient = 0;
- If the classifier ft(x) is poor - Then training error is high and coefficient value is low.



Problem 2 : Recompute weights αi

- Need to update weights **αi** based on where classifier ft(x) makes mistakes.
- In case the classifier ft(x) predicts xi correctly decrease αi.
- In case the classifier ft(x) makes a mistake increase αi.

Increase the weights of the mistakes.



AdaBoost : Normalizing weights αi

- If xi is often a mistake, then weight αi get very large. (Since it is multiplied by the coeffient each time). (mistakes are emphasised)
- If xi often is correct, weight αi gets very small. (Since it is multplied by a smaller coefficient.)
- This can cause numberical instability after multiple iterations.

Therefore Normalize weights to add up to 1 after every iteration.

- αi = αi / (Σ(j = 1 ... N))αj

AdaBoost: learning ensemble

- Start same weight for all points: $\alpha_i = 1/N$
- For t = 1,...,T
  - Learn $f_t(\mathbf{x})$ with data weights α_i
  - Compute coefficient $\hat{w}_t$
  - Recompute weights α_i
  - Normalize weights α_i
- Final model predicts by:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{w}_t f_t(\mathbf{x})\right)$$

$$\hat{w}_t = \frac{1}{2}\ln\left(\frac{1 - weighted\_error(f_t)}{weighted\_error(f_t)}\right)$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

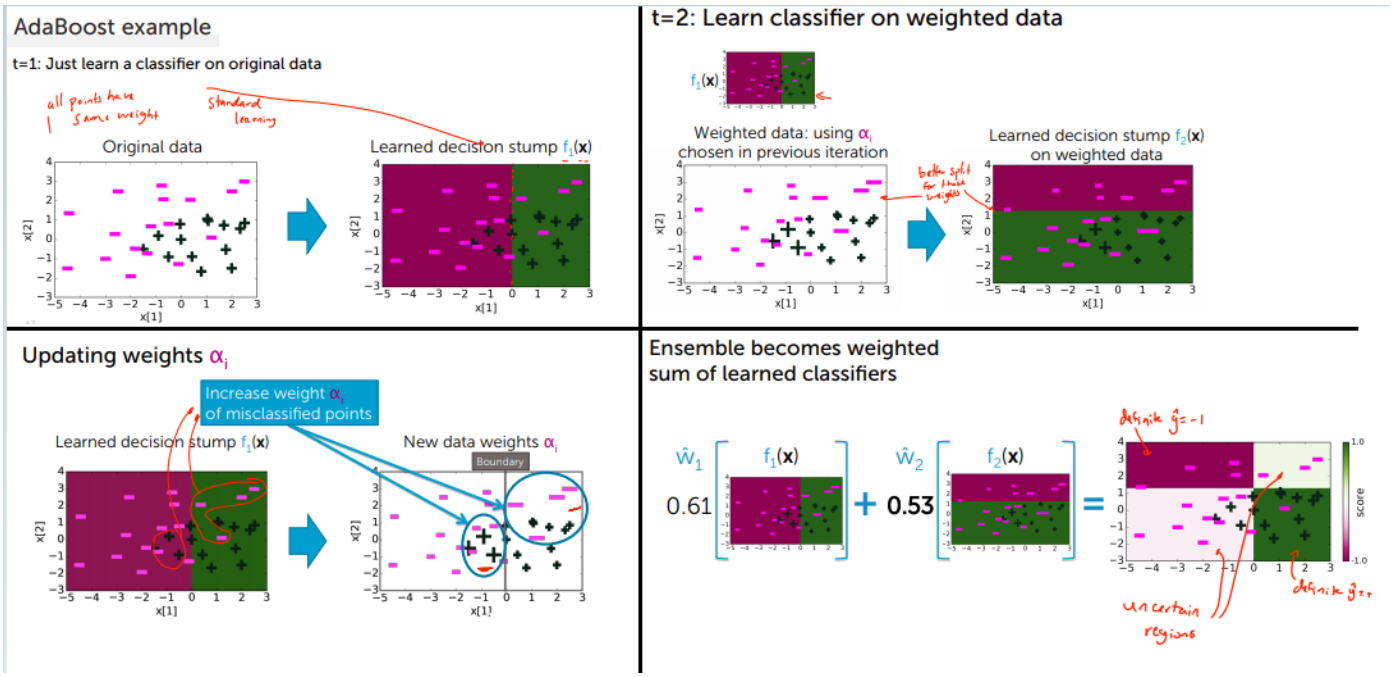
$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j}$$

AdaBoost Example

- For the 1st iteration all the datapoints are assigned the same weight = 1/N.
  - The classifier is a standard decision tree learning algorithm.
- After that, update the weights αi. It will be high for the mistakes.
- In the second iteration, the classifier learns on the weighted data.
  - It is better split then in the first case.

Ensemble becomes weigted sum of learned classifiers.

- For each iteration compute the coefficiens from the weighted error.
- The sum of the learned classifier results in a better decision boundary of the data.
- For the given datatset, after 30 iteration, the training_error = 0. It is an indication of overfitting.

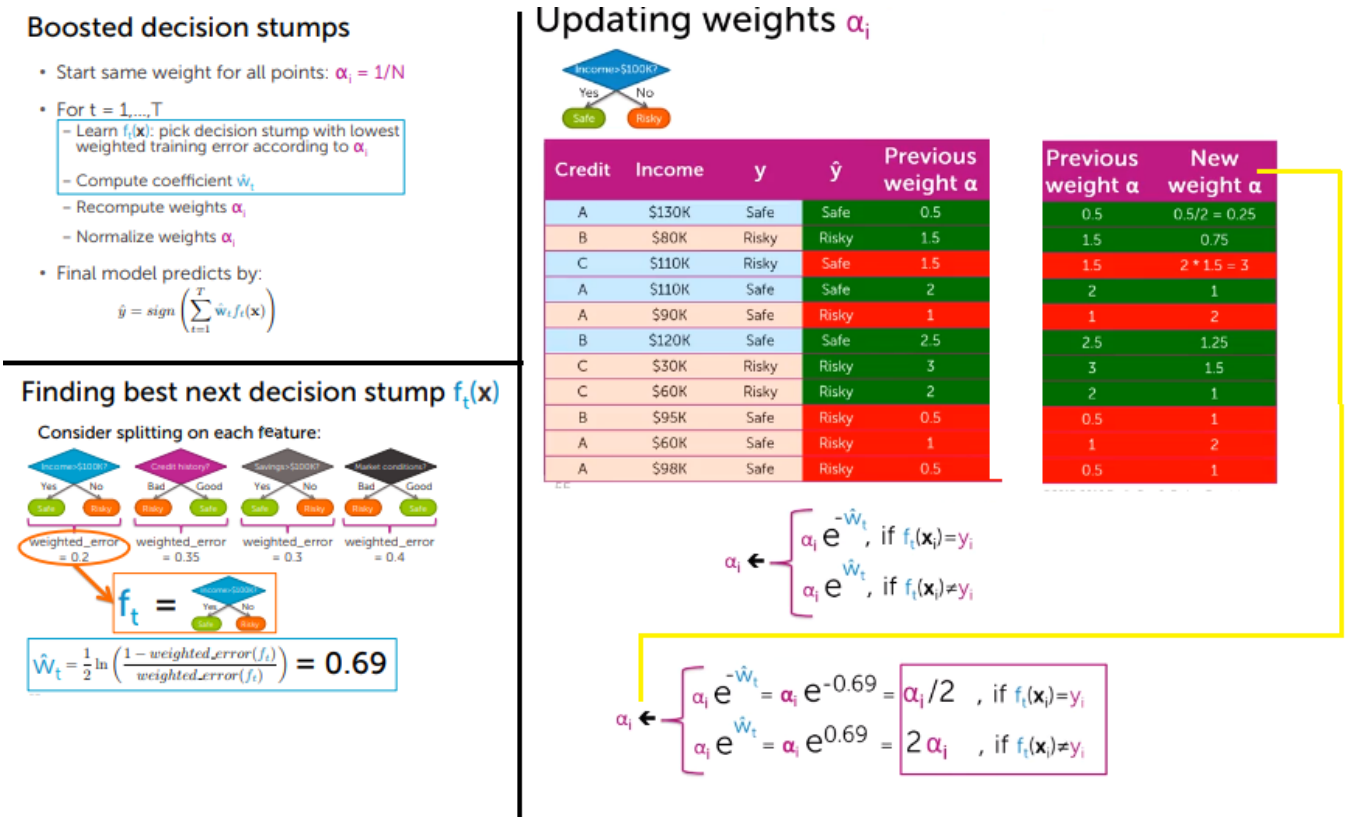


Adaboost Summary:

- For each iteration, need to select a decision stump with the lowest weighted training error according to αi.

Finding the besy next decision stump ft(x):

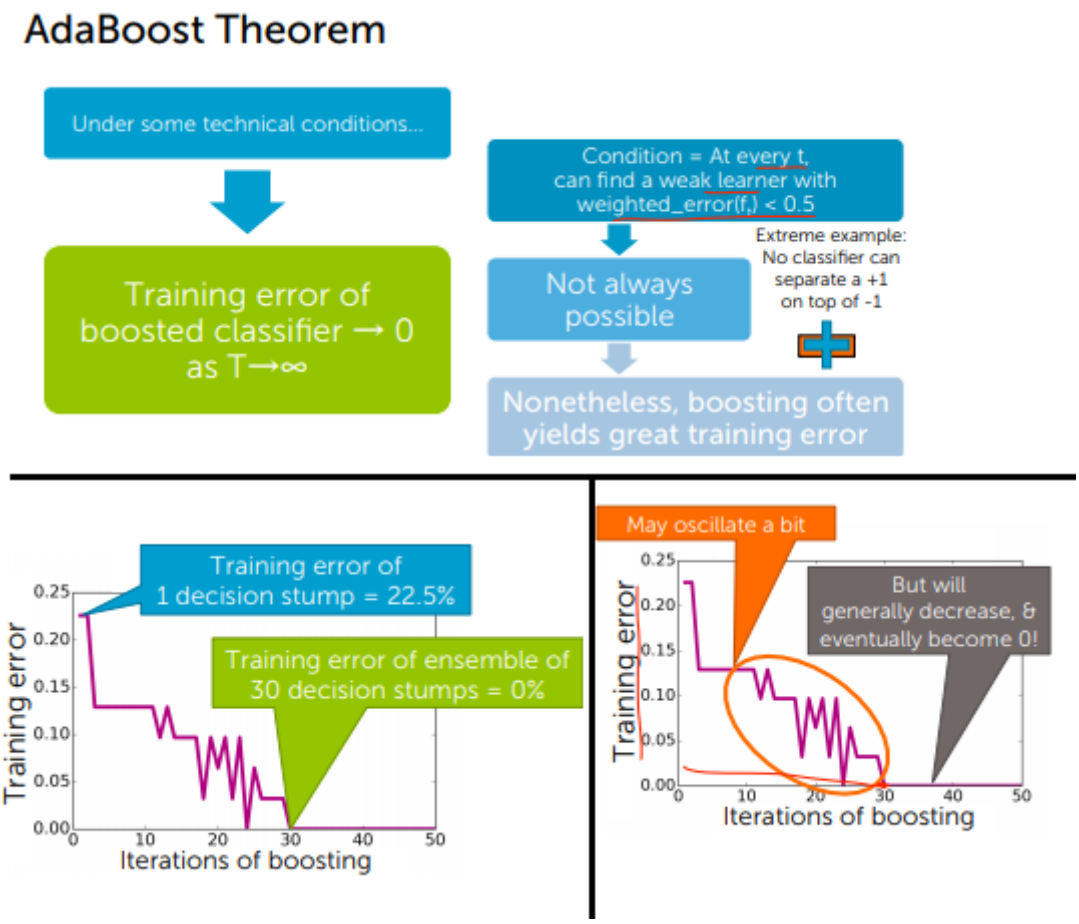
- Consider splitting on each feature.
- For each feature that it split -> consider the training error associated with the mistakes made by the split.
- Choose the split on the feature with least weighted_error.
- Then compute the weighted error.
- Next update the coefficient weights αi.



The Boosting Theorem

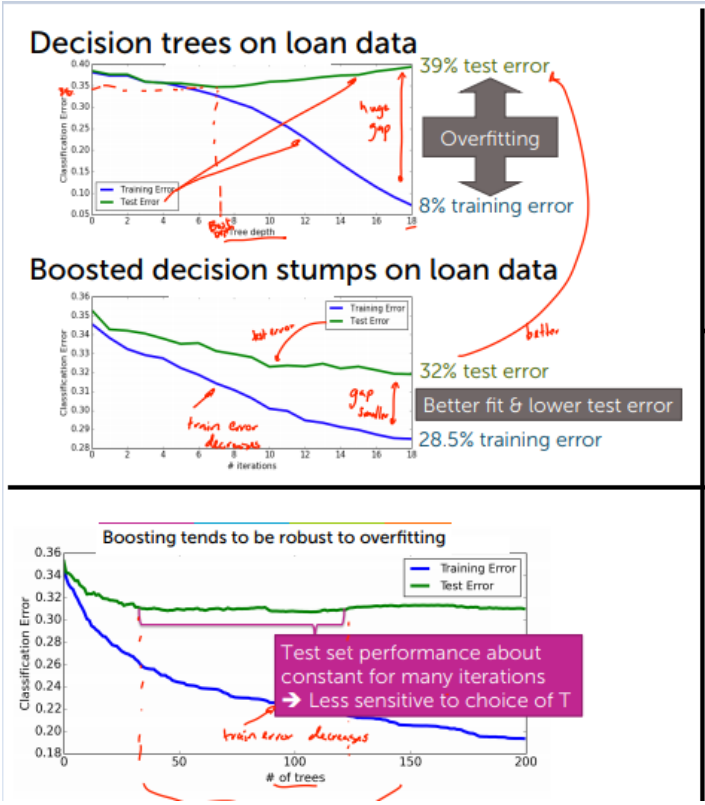
Under certain technical conditions -> Training error of the boosted classifier tends to 0 as the interations T tend to infinity.

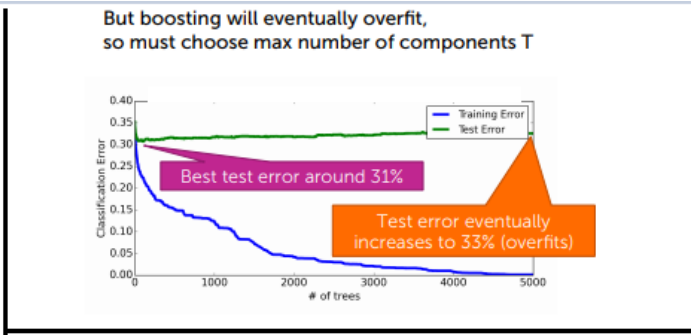
- Technical Conditions:
  - Condition = At every t can find a weak learner with weighted_error(ft) < 0.5
  - Not always possible. **(Extreme example : No classifier can separate a +1 on top of a -1)**
  - Nonetheless boosting often yields great training error.

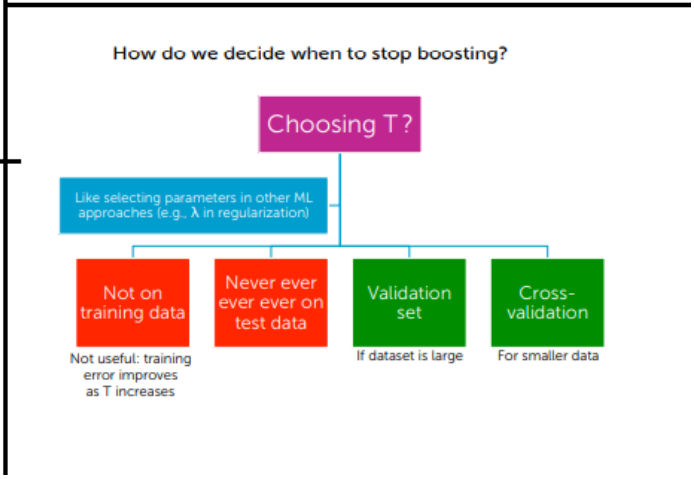


Decision Tree V/s Boosted decision stumps on loan data

- The gap between the test and training error is lower in boosted than in decision tree.
- Boosting tends to be robust to overfitting. The test data is less sensitive to the number of iterations.
- Boosting will eventually overfit, so must choose max number of components T.
- **Choosing when to stop Boosting**
  - Just like other algorithm, need to be caution while selecting appropriate T.
  - Need to test it on Validation Set / Cross-Validation set.
  - Do not choose it on training data / test data.







Variants of boosting and related algorithms

- Many variant, Important once are listed.
- **Gradient Boosting** - Like AdaBoost but useful beyond basic classification.
- Many ensemble approaches, most important.
- **Random Forests**
  - **Bagging** - Pick a random subset of the data.
    - Learn a tree in each subset.
    - Average predictions.
  - Simpler than boosting & easier to parallelize.
  - Typically higher error than boosting for same number of trees (# of iterations T).

Boosting has huge IMPACT!

Quiz

1. Which of the following is NOT an ensemble method?

- ○ Gradient boosted trees
- ○ AdaBoost
- ○ Random forests
- ● Single decision trees

- Gradient boosted trees [is an ensemble method.]
- AdaBoost [is an ensemble method.]
- Random forests [is an ensemble method.]
- Single decision trees [not an ensemble method.]

2. Each binary classifier in an ensemble makes predictions on an input x as listed in the table below. Based on the ensemble coefficients also listed in the table, what is the final ensemble model's prediction for x?

	Classifier coefficient w_t	Prediction for x
Classifier 1	0.61	+1
Classifier 2	0.53	-1
Classifier 3	0.88	-1
Classifier 4	0.34	+1

- ( ) +1
- (•) -1

- sum(Wt x pred(x)) = 0.61 x 1 + 0.53 x -1 + 0.88 x -1 + 0.34 x +1 = -0.46
- sign(-0.46) = -1
- answer : -1

3. (True/False) Boosted trees tend to be more robust to overfitting than decision trees.

- (•) True
- ( ) False

4. (True/False) AdaBoost focuses on data points it incorrectly predicted by increasing those weights in the data set.

- (•) True
- ( ) False

5. In an iteration in AdaBoost, recall that learning the coefficient w_t for learned weak learner f_t is calculated by

$$\frac{1}{2}\ln\left(\frac{1-\texttt{weighted\_error}(f_t)}{\texttt{weighted\_error}(f_t)}\right)$$

If the weighted error of f_t is equal to .25, what is the value of w_t? Round your answer to 2 decimal places.

0.55

6. If you were using AdaBoost and in an iteration of the algorithm were faced with the following classifiers, which one would you be more inclined to include in the ensemble? A classifier with:

- (●) weighted error = 0.1
- ( ) weighted error = 0.3
- ( ) weighted error = 0.5
- ( ) weighted error = 0.7
- ( ) weighted error = 0.99

- weighted error = 0.1 yields highest value of w^t.

```
0.1          1.09861228866811
0.3          0.423648930193602
0.5          0
0.7          -0.423648930193602
0.99        -2.29755992506729
```

- answer : weighted error = 0.1 [CORRECT]

---

7. Imagine we are training a decision stump in an iteration of AdaBoost, and we are at a node. Each data point is (x1, x2, y), where x1,x2 are features, and y is the label. Also included are the weights of the data. The data at this node is:

Weight	x1	x2	y
0.3	0	1	+1
0.35	1	0	-1
0.1	0	1	+1
0.25	1	1	+1

Suppose we assign the same class label to all data in this node. (Pick the class label with the greater total weight.) What is the weighted error at the node? Round your answer to 2 decimal places.

0.35

- Needs a calculation of the weighted error. From the lectures weighted error is given by weight of mistakes / total weight.
- Since three of the four data points have the same label I assume that there is just one mistake and since all the weights sum to 1 that weight is then equal to the weighted error, which is 0.35

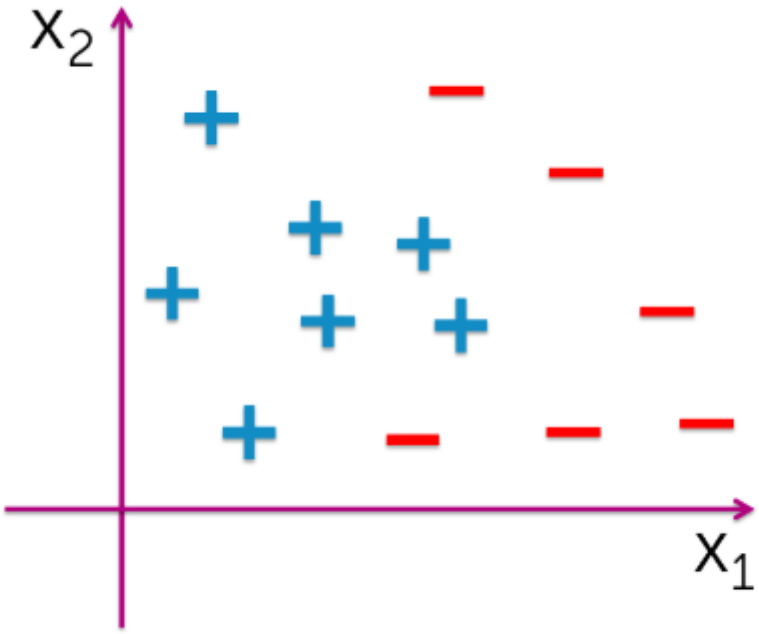
--

8. After each iteration of AdaBoost, the weights on the data points are typically normalized to sum to 1. This is used because

- (●) of issues with numerical instability (underflow/overflow)
- ( ) the weak learners can only learn with normalized weights
- ( ) none of the above

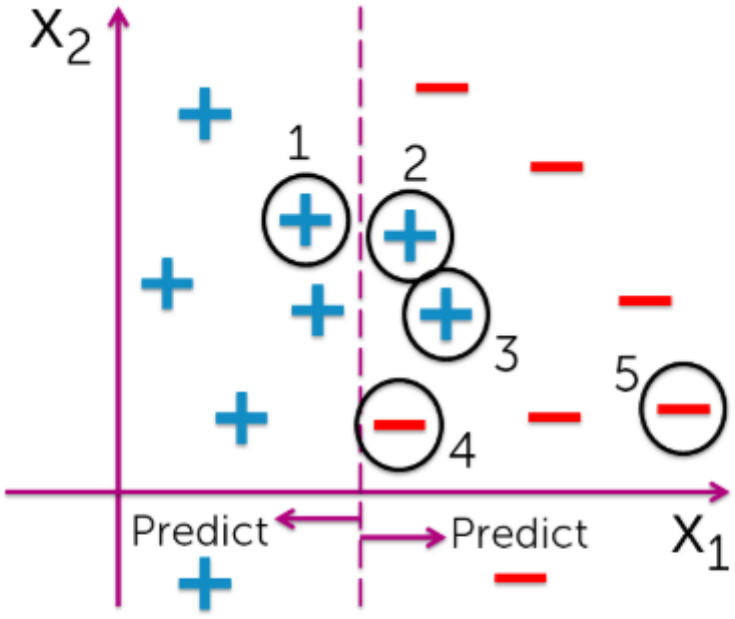
---

9. Consider the following 2D dataset with binary labels.



We train a series of weak binary classifiers using AdaBoost. In one iteration, the weak binary classifier produces the decision boundary as follows:

We train a series of weak binary classifiers using AdaBoost. In one iteration, the weak binary classifier produces the decision boundary as follows:



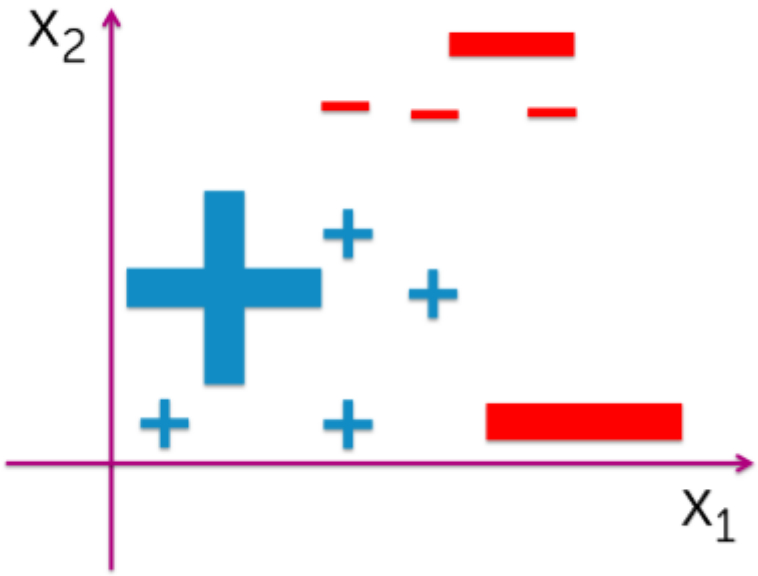
Which of the five points (indicated in the second figure) will receive higher weight in the following iteration? Choose all that apply.

- [ ] (1)
- [x] (2)
- [x] (3)
- [ ] (4)
- [ ] (5)

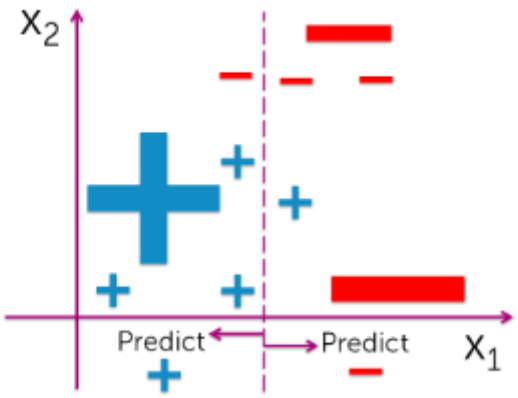
item 1 is positive and in predicted positive area, not weighted.
items 4 & 5 are negative and in predicted negative area, not weighted.
items 2&3 are positive in predicted negative area, these will be weighted.

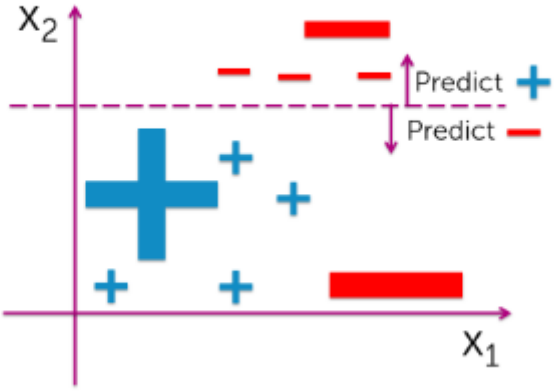
---

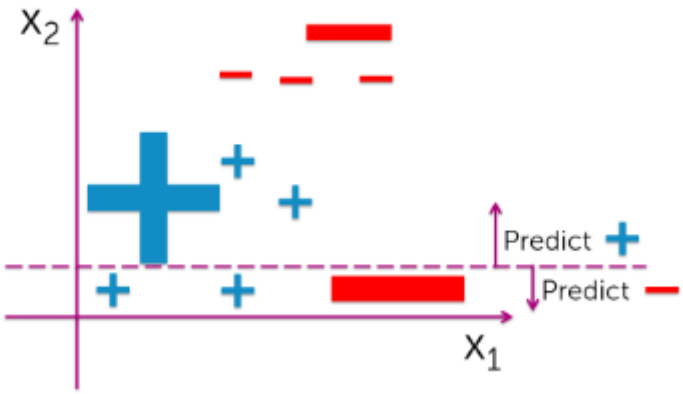
10. Suppose we are running AdaBoost using decision tree stumps. At a particular iteration, the data points have weights according the figure. (Large points indicate heavy weights.)



Which of the following decision tree stumps is most likely to be fit in the next iteration?







11. (True/False) AdaBoost can boost any kind of classifier, not just a decision tree stump.

- True
- False

- "AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem."

In [ ]: