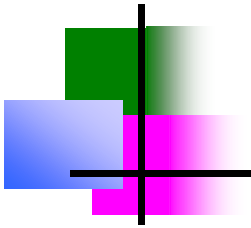


Machine Learning and Computational Intelligence
Lecture 11

Sanjeeb Prasad Panday, PhD
Associate Professor
Dept. of Electronics and Computer Engineering
Director (ICTC)
IOE, TU

Ensemble Learning in Machine Learning

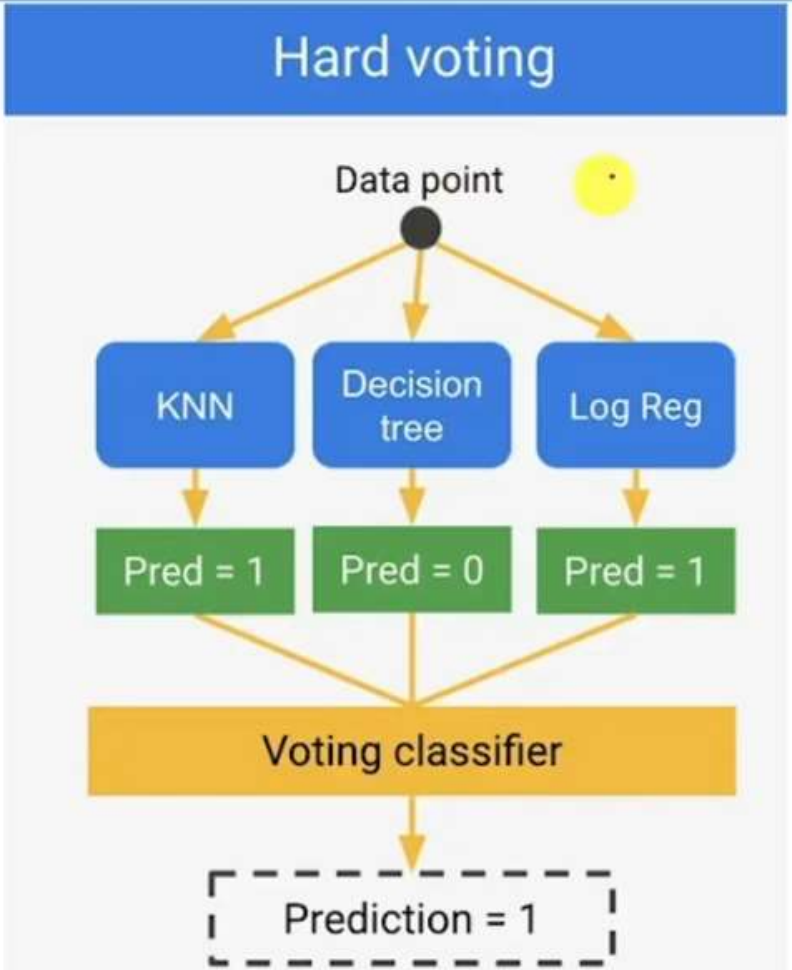
- Ensemble learning is a supervised learning technique used in machine learning to improve overall performance by combining the predictions from multiple models.

Ensemble Learning - Types of Ensemble Methods

- Voting (Averaging)
- Bootstrap aggregation (bagging)
- Random Forests
- Boosting
- Stacked Generalization (Blending)

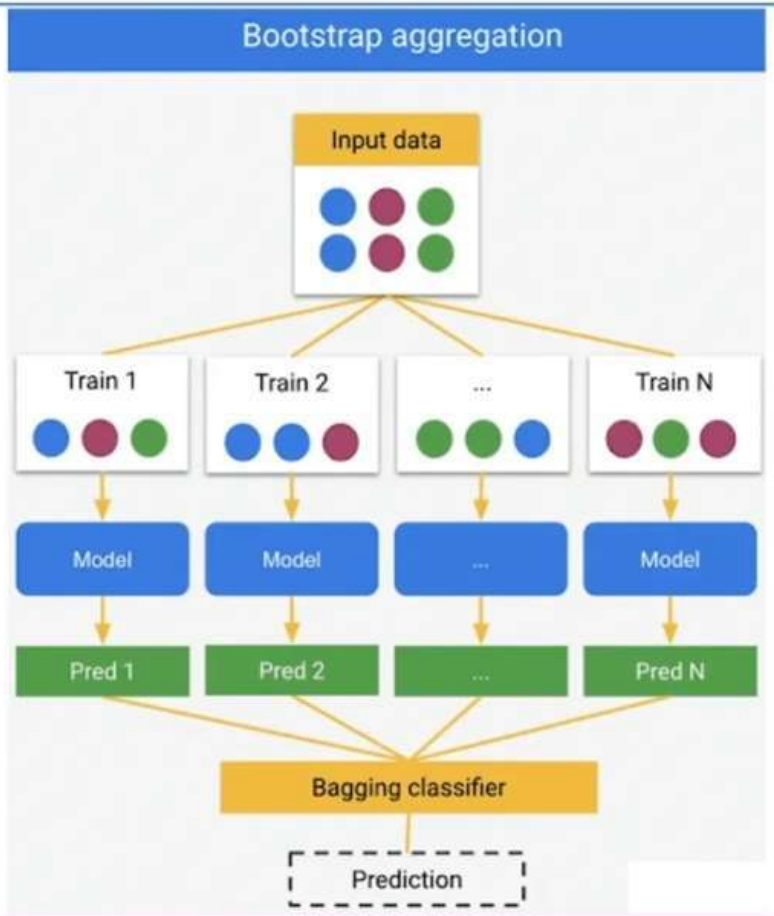
Ensemble Learning – Voting (Averaging)

- Voting is an ensemble machine learning algorithm that involves making a prediction that is the average (regression) or the sum (classification) of multiple machine learning models.



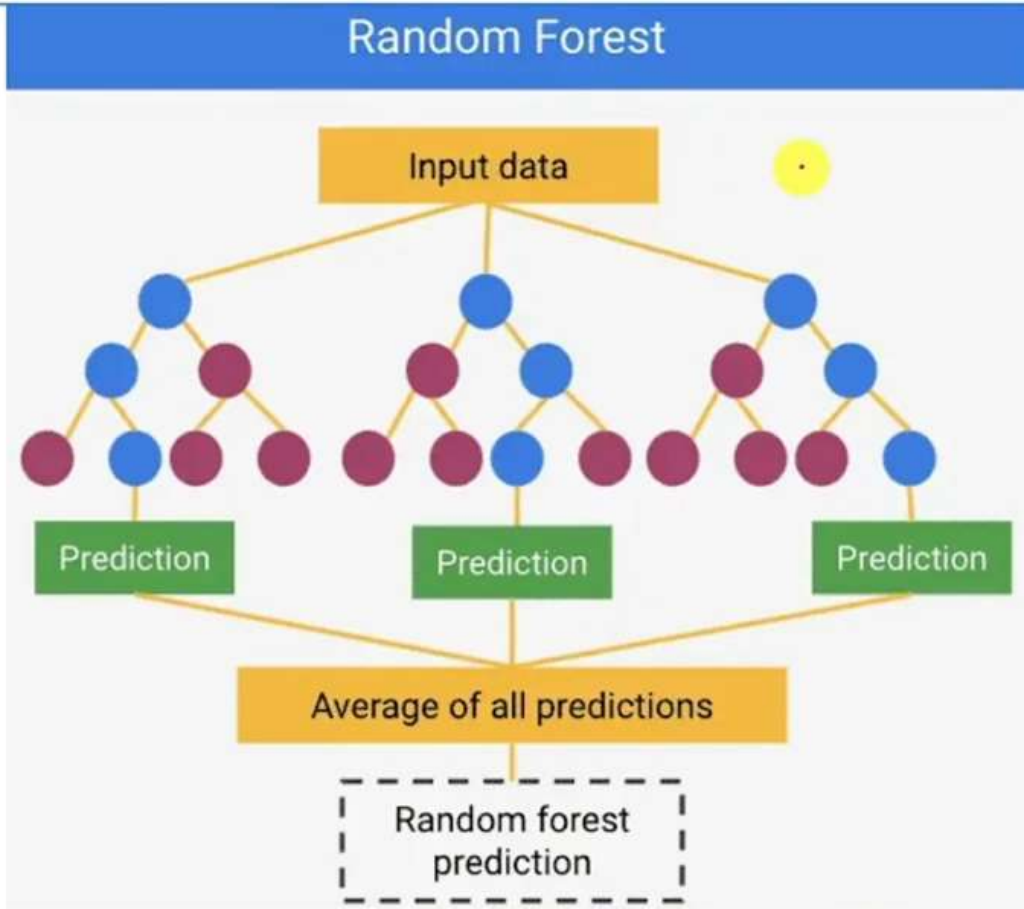
Ensemble Learning – Bootstrap aggregation (bagging)

- Bootstrap Aggregating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms like classification and regression.
- It decreases the variance and helps to avoid overfitting.
- It is usually applied to decision tree methods.
- Bagging is a special case of the model averaging approach.



Ensemble Learning – Random Forest

- Random forest is a commonly-used machine learning algorithm.
- A random forest is an ensemble learning method where multiple decision trees are constructed and then they are merged to get a more accurate prediction.



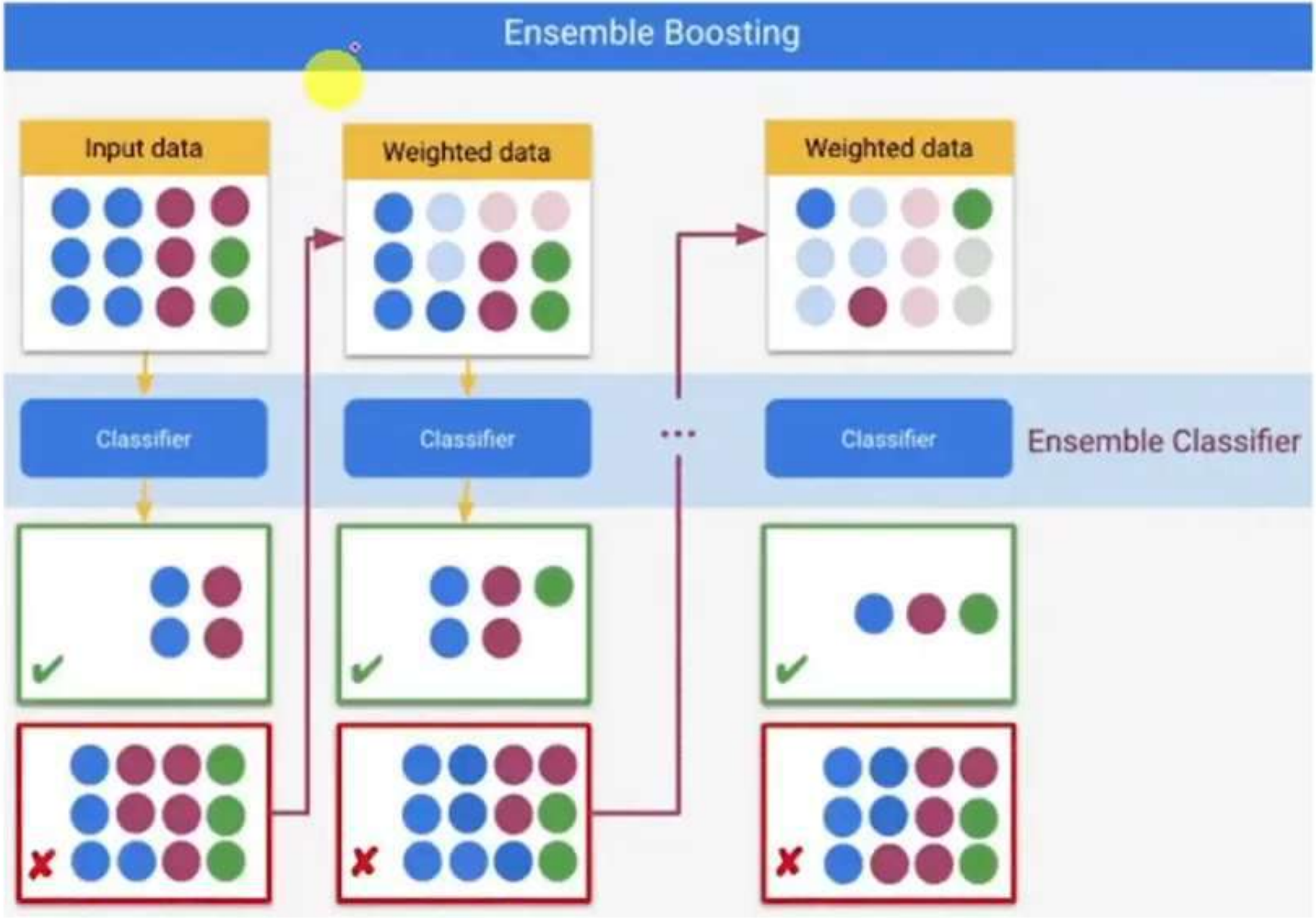
Ensemble Learning – Boosting

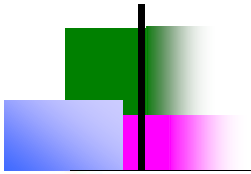
- Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers.
- It is done by building a model by using weak models in series.
- Firstly, a model is built from the training data.

Ensemble Learning – Boosting

- Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers.
- It is done by building a model by using weak models in series.
- Firstly, a model is built from the training data.
- Then the second model is built which tries to correct the errors present in the first model.
- This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.

Ensemble Learning – Boosting



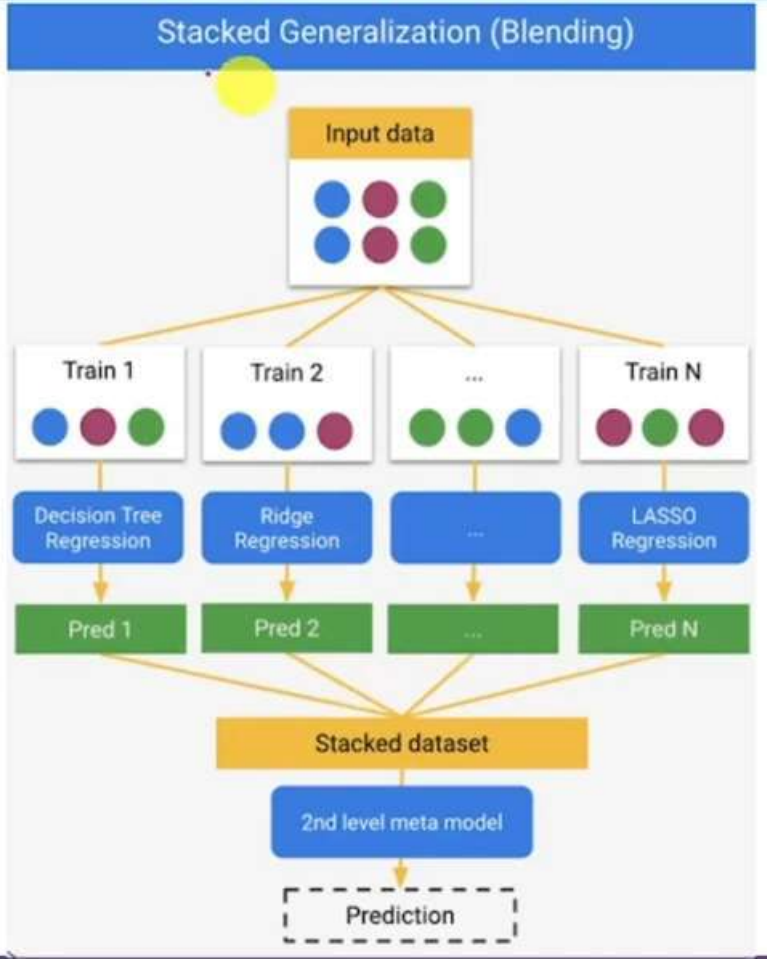


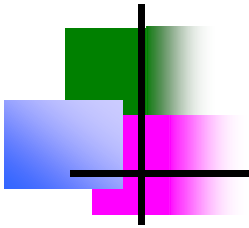
Ensemble Learning – Stacked Generalization (Blending)

- Stacking, Blending and and Stacked Generalization are all the same thing with different names. It is a kind of ensemble learning.
- In traditional ensemble learning, we have multiple classifiers trying to fit to a training set to approximate the target function.
- Since each classifier will have its own output, we will need to find a combining mechanism to combine the results.
- This can be through voting (majority wins), weighted voting (some classifier has more authority than the others), averaging the results, etc.
- This is the traditional way of ensemble learning.

Ensemble Learning – Stacked Generalization (Blending)

- In stacking, the combining mechanism is that the output of the classifiers (Level 0 classifiers) will be used as training data for another classifier (Level 1 classifier) to approximate the same target function.
- Basically, you let the Level 1 classifier to figure out the combining mechanism.





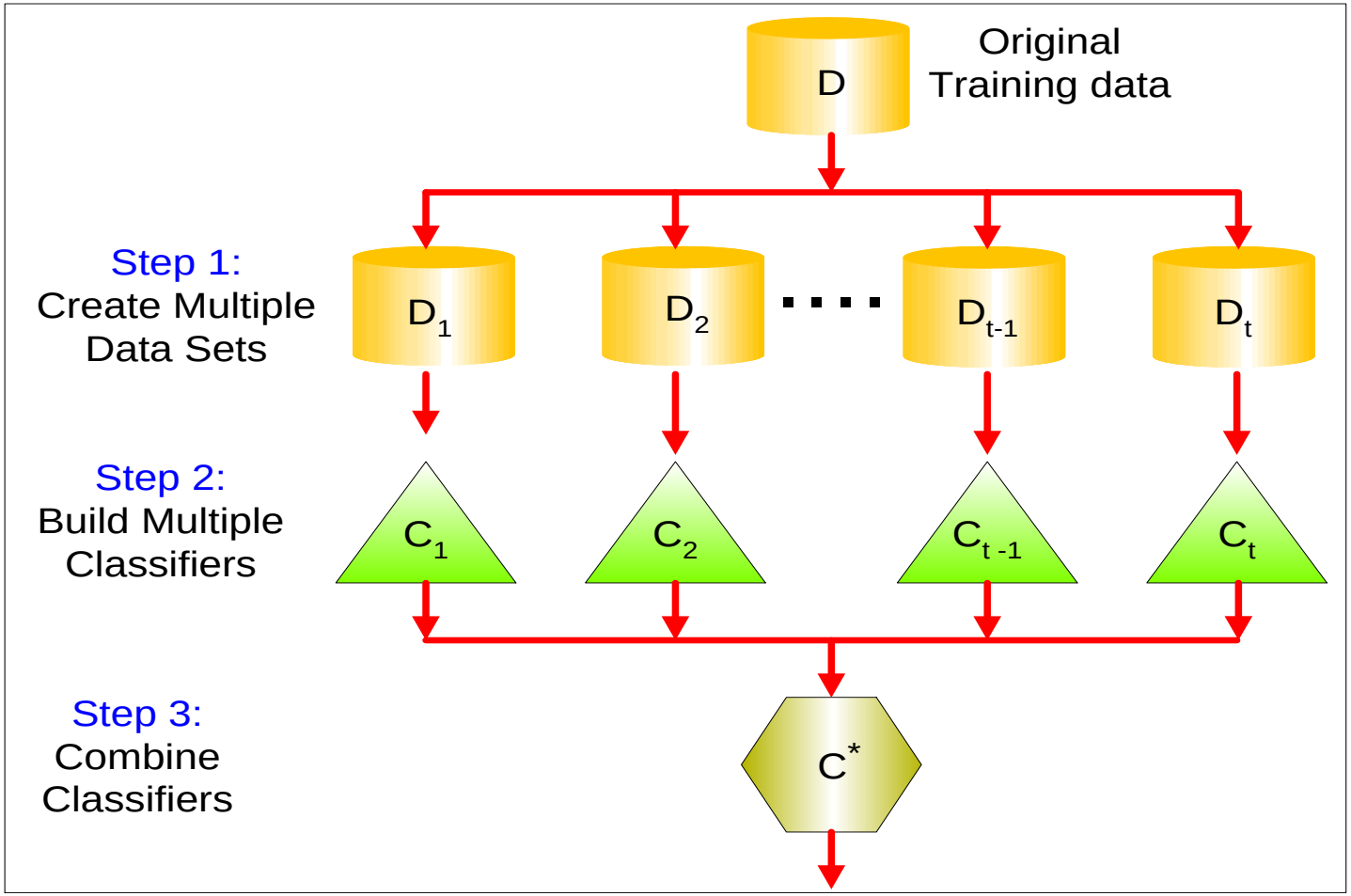
Outline

- **Bias/Variance Tradeoff**

- Ensemble methods that minimize variance
  - Bagging
  - Random Forests

- Ensemble methods that minimize bias
  - Functional Gradient Descent
  - Boosting
  - Ensemble Selection

General Idea



Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
  - Bagging
  - Boosting

Bagging

- Sampling with replacement

Training Data

Data ID

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- Each sample has probability $(1 – 1/n)^n$ of being selected as test data
- Training data = $1- (1 – 1/n)^n$ of the original data

The 0.632 bootstrap

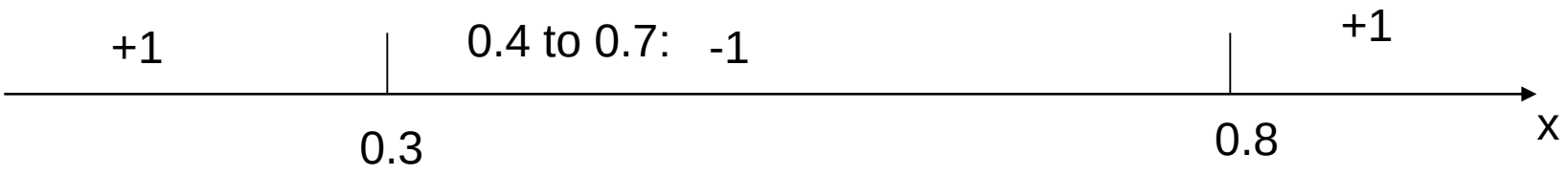
- This method is also called the *0.632 bootstrap*
  - A particular training data has a probability of 1-1/*n* of *not* being picked
  - Thus its probability of ending up in the test data (not selected) is:

$$\left(1-\frac{1}{n}\right)^{n} \approx e^{-1} = 0.368$$

  - This means the training data will contain approximately 63.2% of the instances

Example of Bagging

Assume that the training data is:



Goal: find a collection of 10 simple thresholding classifiers that collectively can classify correctly.

-Each simple (or weak) classifier is:

(x<=K → class = +1 or -1 depending on which value yields the lowest error; where K is determined by entropy minimization)

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = -1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ==> y = -1
x > 0.05 ==> y = 1

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ==> y = -1
x > 0.05 ==> y = 1

Figure 5.35. Example of bagging.

Bagging (applied to training data)

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

Figure 5.36. Example of combining classifiers constructed using the bagging approach.

Accuracy of ensemble classifier: 100% ☺

Bagging- Summary

- Works well if the base classifiers are unstable (complement each other)
- Increased accuracy because it ***reduces the variance*** of the individual classifier
- Does not focus on any particular instance of the training data
  - Therefore, less susceptible to model over-fitting when applied to noisy data
- What if we want to focus on a particular instances of training data?

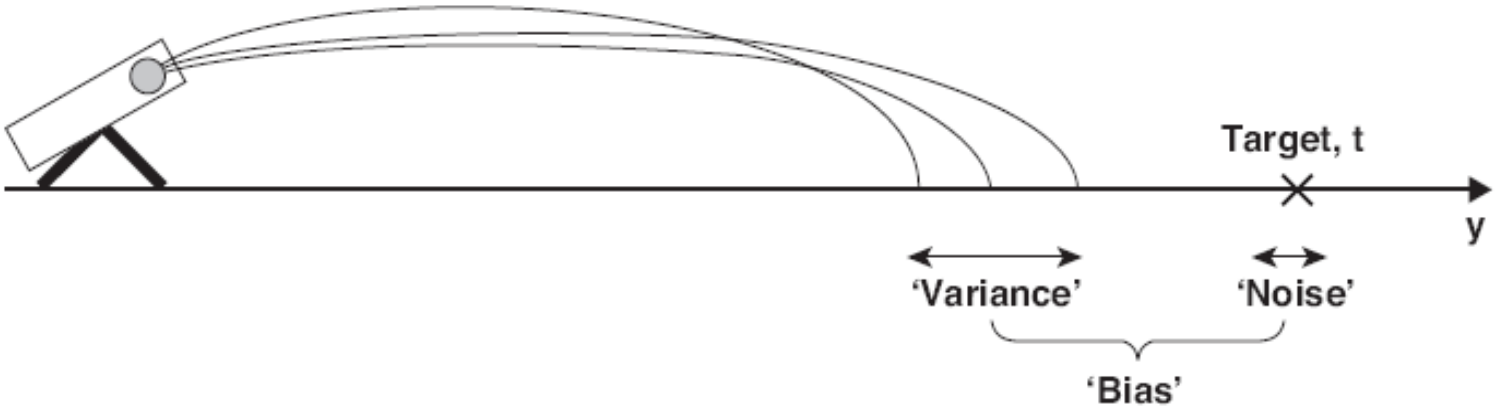


Figure 5.32. Bias-variance decomposition.

In general,
- Bias is contributed to by the training error; a complex model has low bias.
-Variance is caused by future error; a complex model has High variance.
- Bagging reduces the variance in the base classifiers.

AdaBoost (Adaptive Boosting)

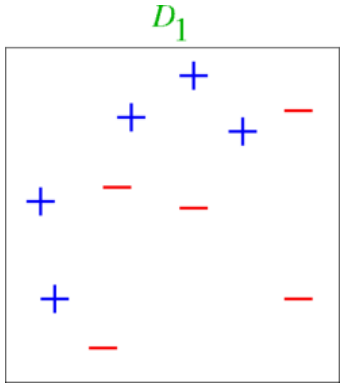
- We can now describe the AdaBoost algorithm.
- Given a base classifier, the key steps of AdaBoost are:
  1. At each iteration, re-weight the training samples by assigning larger weights to samples (i.e., data points) that were classified incorrectly.
  2. Train a new base classifier based on the re-weighted samples.
  3. Add it to the ensemble of classifiers with an appropriate weight.
  4. Repeat the process many times.
- Requirements for base classifier:
  - Needs to minimize weighted error.
  - Ensemble may get very large, so base classifier must be fast. It turns out that any so-called weak learner/classifier suffices.
- Individually, weak learners may have high bias (underfit). By making each classifier focus on previous mistakes, AdaBoost reduces bias.

Weak Learner/Classifier

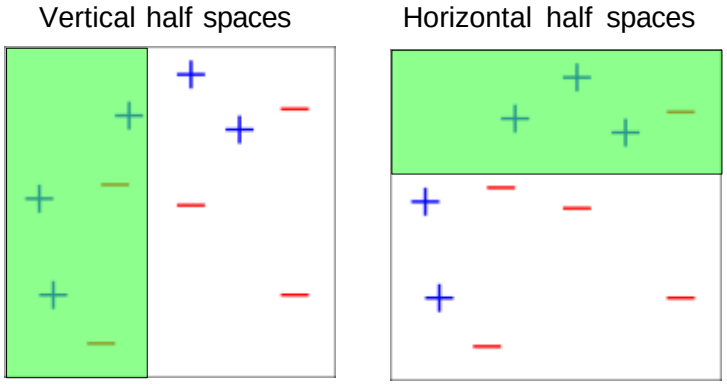
- (Informal) Weak learner is a learning algorithm that outputs a hypothesis (e.g., a classifier) that performs slightly better than chance, e.g., it predicts the correct label with probability 0.51 in binary label case.
- We are interested in weak learners that are *computationally* efficient.
  - Decision trees
  - Even simpler: Decision Stump: A decision tree with a single split

[Formal definition of weak learnability has quantifies such as "for any distribution over data" and the requirement that its guarantee holds only probabilistically.]

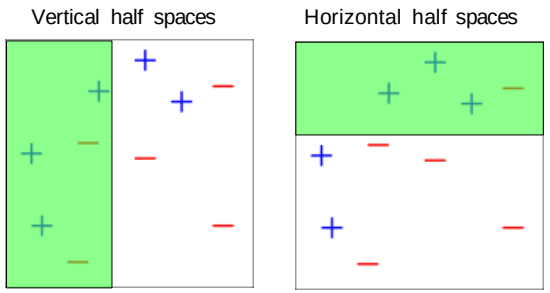
Weak Classifiers



These weak classifiers, which are decision stumps, consist of the set of horizontal and vertical half spaces.



Weak Classifiers



- A *single* weak classifier is not capable of making the training error small
- But if can guarantee that it performs slightly better than chance, i.e., the weighted error of classifier h according to the given weights $\mathbf{w} = (w_1, \ldots, w_N)$ is at most $\frac{1}{2} - \gamma$ for some $\gamma > 0$, using it with AdaBoost gives us a universal function approximator!
- Last lecture we used information gain as the splitting criterion. When using decision stumps with AdaBoost we often use a "GINI Impurity", which (roughly speaking) picks the split that directly minimizes error.
- Now let's see how AdaBoost combines a set of weak classifiers in order to make a better ensemble of classifiers...

Notation in this lecture

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^{N}$ where $t^{(n)} \in \{-1, +1\}$
  - This is different from previous lectures where we had $t^{(n)} \in \{0, +1\}$
  - It is for notational convenience, otw equivalent.
- A classifier or hypothesis $h : \mathbf{x} \to \{-1, +1\}$
- 0-1 loss: $\mathbb{I}[h(x^{(n)}) \neq t^{(n)}] = \frac{1}{2}(1 - h(x^{(n)}) \cdot t^{(n)})$

AdaBoost Algorithm

- Input: Data $\mathcal{D}_N$, weak classifier WeakLearn (a classification procedure that returns a classifier h, e.g. best decision stump, from a set of classifiers $\mathcal{H}$, e.g. all possible decision stumps), number of iterations T
- Output: Classifier $H(x)$
- Initialize sample weights: $w^{(n)} = \frac{1}{N}$ for $n = 1, \ldots, N$
- For $t = 1, \ldots, T$
  - Fit a classifier to weighted data ($h_t \leftarrow \text{WeakLearn}(\mathcal{D}_N, \mathbf{w})$), e.g.,

$$h_t \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{n=1}^{N} w^{(n)} \mathbb{I}\{h(\mathbf{x}^{(n)}) \neq t^{(n)}\}$$

  - Compute weighted error $\text{err}_t = \frac{\sum_{n=1}^{N} w^{(n)} \mathbb{I}\{h_t(\mathbf{x}^{(n)}) \neq t^{(n)}\}}{\sum_{n=1}^{N} w^{(n)}}$
  - Compute classifier coefficient $\alpha_t = \frac{1}{2} \log \frac{1-\text{err}_t}{\text{err}_t}$ $\quad (\in (0, \infty))$
  - Update data weights

$$w^{(n)} \leftarrow w^{(n)} \exp\left(-\alpha_t t^{(n)} h_t(\mathbf{x}^{(n)})\right) \left[\equiv w^{(n)} \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(n)}) \neq t^{(n)}\}\right)\right]$$

    Homework 3: prove the above equivalence.

- Return $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

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 hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].$$

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

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

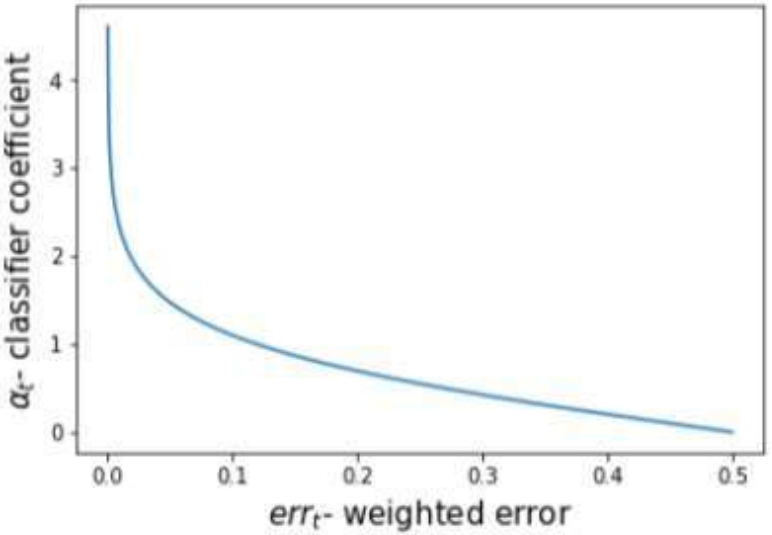
where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

Weighting Intuition

- Recall: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$ where $\alpha_t = \frac{1}{2}\log\frac{1-\text{err}_t}{\text{err}_t}$



- Weak classifiers which get lower weighted error get more weight in the final classifier
- Also: $w^{(n)} \leftarrow w^{(n)} \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(n)}) \neq t^{(n)}\}\right)$
  - If $\text{err}_t \approx 0$, α_t high so misclassified examples get more attention
  - If $\text{err}_t \approx 0.5$, α_t low so misclassified examples are not emphasized

AdaBoost Ensemble Learning – Solved Example

- Consider a training dataset of six data instances as shown in Table.

CGPA	Interactiveness	Practical Knowledge	Communication Skill	Job Profile
>=9	Yes	Good	Good	Yes
<9	No	Good	Moderate	Yes
>=9	No	Average	Moderate	No
<9	No	Average	Good	No
>=9	Yes	Good	Moderate	Yes
>=9	Yes	Good	Moderate	Yes

- Consider a training dataset of six data instances as shown in Table.
- Use 4 Decision Stumps for each of the 4 attributes.
- Apply AdaBoost algorithm & classify the dataset with Job Offer as target attribute.

CGPA	Interactiveness	Practical Knowledge	Communication Skill	Job Profile
>=9	Yes	Good	Good	Yes
<9	No	Good	Moderate	Yes
>=9	No	Average	Moderate	No
<9	No	Average	Good	No
>=9	Yes	Good	Moderate	Yes
>=9	Yes	Good	Moderate	Yes

- **Step 1:** Initial **weight** assigned to each item= **1/6**.
- **Step 2:** Iterate for each Weak classifier.
- **I. Decision Stump for CGPA**
- **Step 2 (a):** Train the Decision Stump H_{CGPA} with a random bootstrap sample from the training dataset T.
- Since there are only 6 data instances, use the full training dataset.

CGPA	Interactiveness	Practical Knowledge	Communication Skill	Job Profile
>=9	Yes	Good	Good	Yes
<9	No	Good	Moderate	Yes
>=9	No	Average	Moderate	No
<9	No	Average	Good	No
>=9	Yes	Good	Moderate	Yes
>=9	Yes	Good	Moderate	Yes

- The **first Decision stump** classifies the instances based on the CGPA attribute as shown in Table.
- If **CGPA >=9**, the data instance is predicted to have **'Job Offer'** as **'Yes'** else **'No'**.

CGPA	Predicted Job Offer	Actual Job Offer	Weight
>=9		Yes	1/6
<9		Yes	1/6
>=9		No	1/6
<9		No	1/6
>=9		Yes	1/6
>=9		Yes	1/6

- The **first Decision stump** classifies the instances based on the CGPA attribute as shown in Table.
- If **CGPA >=9**, the data instance is predicted to have **'Job Offer'** as **'Yes'** else **'No'**.

CGPA	Predicted Job Offer	Actual Job Offer	Weight
>=9	Yes	Yes	1/6
<9	No	Yes	1/6
>=9	Yes	No	1/6
<9	No	No	1/6
>=9	Yes	Yes	1/6
>=9	Yes	Yes	1/6

- **Step 2 (b):** Compute the **weighted error**

  ε_{CGPA} of H_{CGPA} on current training dataset T:

- $\varepsilon_i = \sum_{j=1}^{N} H_i(d_j) wt(d_j)$
- *where*
- $H_i(d_j) = 0$ *for Correct prediction*
- $H_i(d_j) = 1$ *for Wrong prediction*
- $\boldsymbol{\varepsilon_{CGPA} = 2 * \frac{1}{6} = 0.333}$

CGPA	Predicted Job Offer	Actual Job Offer	Weight
>=9	Yes	Yes	1/6
<9	No	Yes	1/6
>=9	Yes	No	1/6
<9	No	No	1/6
>=9	Yes	Yes	1/6
>=9	Yes	Yes	1/6

- **Step 2 (c):** Compute the weight of each weak classifier:

$$\alpha_{CGPA} = \frac{1}{2} \frac{ln(1 - \varepsilon_{CGPA})}{\varepsilon_{CGPA}}$$

$$\alpha_{CGPA} = \frac{1}{2} \frac{ln(1 - 0.333)}{0.333}$$

$$\boldsymbol{\alpha_{CGPA} = 0.347}$$

CGPA	Predicted Job Offer	Actual Job Offer	Weight
>=9	Yes	Yes	1/6
<9	No	Yes	1/6
>=9	Yes	No	1/6
<9	No	No	1/6
>=9	Yes	Yes	1/6
>=9	Yes	Yes	1/6

- **Step 2 (d):** Calculate the normalizing factor Z_{CGPA}
- $Z_{CGPA} = wt(Correct\ Classifed\ Instance) * No\ of\ Correct\ Classification * e^{-\alpha_{CGPA}} + wt(Wrong\ Classifed\ Instance) * No\ of\ Wrong\ Classification * e^{+\alpha_{CGPA}}$
- $Z_{CGPA} = \frac{1}{6} * 4 * e^{-0.347} + \frac{1}{6} * 2 * e^{0.347}$
- $Z_{CGPA} = \mathbf{0.9428}$

CGPA	Predicted Job Offer	Actual Job Offer	Weight
>=9	Yes	Yes	1/6
<9	No	Yes	1/6
>=9	Yes	No	1/6
<9	No	No	1/6
>=9	Yes	Yes	1/6
>=9	Yes	Yes	1/6

- **Step 2 (e):** Update the weight of all data instances:

- $wt(d_j)_{i+1} = \frac{wt(d_j)_{CGPA}\ of\ correct\ Instance * e^{-\propto CGPA}}{Z_{CGPA}}$
- $wt(d_j)_{i+1} = \frac{\frac{1}{6} * e^{-0.347}}{0.9428} = 0.1249$
- $wt(d_j)_{i+1} = \frac{wt(d_j)_{CGPA}\ of\ Incorrect\ Instance * e^{\propto CGPA}}{Z_{CGPA}}$
- $wt(d_j)_{i+1} = \frac{\frac{1}{6} * e^{0.347}}{0.9428} = 0.2501$

CGPA	Predicted Job Offer	Actual Job Offer	Weight
>=9	Yes	Yes	1/6
<9	No	Yes	1/6
>=9	Yes	No	1/6
<9	No	No	1/6
>=9	Yes	Yes	1/6
>=9	Yes	Yes	1/6

- **Step 2 (e):** Update the weight of all data instances:

- $wt(d_j)_{i+1} = \frac{wt(d_j)_{CGPA}\ of\ correct\ Instance * e^{-\propto CGPA}}{Z_{CGPA}}$
- $wt(d_j)_{i+1} = \frac{\frac{1}{6} * e^{-0.347}}{0.9428} = 0.1249$
- $wt(d_j)_{i+1} = \frac{wt(d_j)_{CGPA}\ of\ Incorrect\ Instance * e^{\propto CGPA}}{Z_{CGPA}}$
- $wt(d_j)_{i+1} = \frac{\frac{1}{6} * e^{0.347}}{0.9428} = 0.2501$

CGPA	Predicted Job Offer	Actual Job Offer	Weight
>=9	Yes	Yes	0.1249
<9	No	Yes	0.2501
>=9	Yes	No	0.2501
<9	No	No	0.1249
>=9	Yes	Yes	0.1249
>=9	Yes	Yes	0.1249

- **II. Decision Stump for Interactiveness**
- **Step 2 (a):** Train the Decision Stump $H_{Interact}$ with the sample obtained in the previous weak classifier $H_{Interact}$
- If **Interactiveness** is **Yes**, the data instance is predicted to have **'Job Offer'** as **'Yes'** else **'No'**.

Interactiveness	Predicted Job Offer	Actual Job Offer	Weight
Yes	Yes	Yes	0.1249
No	No	Yes	0.2501
No	No	No	0.2501
No	No	No	0.1249
Yes	Yes	Yes	0.1249
Yes	Yes	Yes	0.1249

- **Step 2 (b):** Compute the **weighted error** $\varepsilon_{Interact}$ of $H_{Interact}$ on current training dataset T:
- $\varepsilon_i = \sum_{j=1}^{N} H_i(d_j) wt(d_j)$
- *where*
- $H_i(d_j) = 0 \; for \; Correct \; prediction$
- $H_i(d_j) = 1 \;\; for \; Wrong \; prediction$

Interactiveness	Predicted Job Offer	Actual Job Offer	Weight
Yes	Yes	Yes	0.1249
No	No	Yes	0.2501
No	No	No	0.2501
No	No	No	0.1249
Yes	Yes	Yes	0.1249
Yes	Yes	Yes	0.1249

- **Step 2 (b):** Compute the **weighted error** $\varepsilon_{Interact}$ of $H_{Interact}$ on current training dataset T:
- $\varepsilon_i = \sum_{j=1}^{N} H_i(d_j) wt(d_j)$
- *where*
- $H_i(d_j) = 0 \; for \; Correct \; prediction$
- $H_i(d_j) = 1 \; for \; Wrong \; prediction$
- $\boldsymbol{\varepsilon_{Interact} = 1 * 0.2501 = 0.2501}$

Interactiveness	Predicted Job Offer	Actual Job Offer	Weight
Yes	Yes	Yes	0.1249
No	No	Yes	0.2501
No	No	No	0.2501
No	No	No	0.1249
Yes	Yes	Yes	0.1249
Yes	Yes	Yes	0.1249

- **Step 2 (c):** Compute the weight of each weak classifier:

$$\alpha_{Interact} = \frac{1}{2} \frac{ln(1 - \varepsilon_{Interact})}{\varepsilon_{Interact}}$$

$$\alpha_{Interact} = \frac{1}{2} \frac{ln(1 - 0.2501)}{0.2501}$$

$$\boldsymbol{\alpha_{Interact} = 0.5490}$$

Interactiveness	Predicted Job Offer	Actual Job Offer	Weight
Yes	Yes	Yes	0.1249
No	No	Yes	0.2501
No	No	No	0.2501
No	No	No	0.1249
Yes	Yes	Yes	0.1249
Yes	Yes	Yes	0.1249

- **Step 2 (d):** Calculate the normalizing factor

  $Z_{Interact}$

- $Z_{Interact} = wt(Correct\ Classifed\ Instance) * No\ of\ Correct\ Classification * e^{-\alpha_{Interact}} + wt(Wrong\ Classifed\ Instance) * No\ of\ Wrong\ Classification * e^{+\alpha_{Interact}}$

- $Z_{Interact} = 0.1249 * 4 * e^{-0.549} + 0.2501 * 1 * e^{-0.549} + 0.2501 * 1 * e^{0.549}$

Interactiveness	Predicted Job Offer	Actual Job Offer	Weight
Yes	Yes	Yes	0.1249
No	No	Yes	0.2501
No	No	No	0.2501
No	No	No	0.1249
Yes	Yes	Yes	0.1249
Yes	Yes	Yes	0.1249

- **Step 2 (d):** Calculate the normalizing factor $Z_{Interact}$
- $Z_{Interact} = wt(Correct\ Classifed\ Instance) * No\ of\ Correct\ Classification * e^{-\alpha_{Interact}} + wt(Wrong\ Classifed\ Instance) * No\ of\ Wrong\ Classification * e^{+\alpha_{Interact}}$
- $Z_{Interact} = 0.1249 * 4 * e^{-0.549} + 0.2501 * 1 * e^{-0.549} + 0.2501 * 1 * e^{0.549}$

Interactiveness	Predicted Job Offer	Actual Job Offer	Weight
Yes	Yes	Yes	0.1249
No	No	Yes	0.2501
No	No	No	0.2501
No	No	No	0.1249
Yes	Yes	Yes	0.1249
Yes	Yes	Yes	0.1249

- **Step 2 (d):** Calculate the normalizing factor $Z_{Interact}$
- $\boldsymbol{Z_{Interact}} = wt(Correct\ Classifed\ Instance) * No\ of\ Correct\ Classification * e^{-\alpha_{Interact}} + wt(Wrong\ Classifed\ Instance) * No\ of\ Wrong\ Classification * e^{+\alpha_{Interact}}$
- $\boldsymbol{Z_{Interact}} = 0.1249 * 4 * e^{-0.549} + 0.2501 * 1 * e^{-0.549} + 0.2501 * 1 * e^{0.549}$
- $\boldsymbol{Z_{Interact}} = \mathbf{0.866}$

Interactiveness	Predicted Job Offer	Actual Job Offer	Weight
Yes	Yes	Yes	0.1249
No	No	Yes	0.2501
No	No	No	0.2501
No	No	No	0.1249
Yes	Yes	Yes	0.1249
Yes	Yes	Yes	0.1249

- **Step 2 (e):** Update the weight of all data instances:

- $wt(d_j)_{i+1} = \frac{wt(d_j)_{Interact} \; of \; correct \; Instance * e^{-\propto Interact}}{Z_{Interact}}$

- $wt(d_j)_{i+1} = \frac{0.1249 * e^{-0.549}}{0.866} = 0.0832$

- $wt(d_j)_{i+1} = \frac{0.2501 * e^{-0.549}}{0.866} = 0.1667$

- $wt(d_j)_{i+1} = \frac{wt(d_j)_{Interact} \; of \; Incorrect \; Instance * e^{\propto Interact}}{Z_{Interact}}$

- $wt(d_j)_{i+1} = \frac{0.2501 * e^{0.549}}{0.866} = 0.5001$

Interactiveness	Predicted Job Offer	Actual Job Offer	Weight
Yes	Yes	Yes	0.1249
No	No	Yes	0.2501
No	No	No	0.2501
No	No	No	0.1249
Yes	Yes	Yes	0.1249
Yes	Yes	Yes	0.1249

- **Step 2 (e):** Update the weight of all data instances:

- $wt(d_j)_{i+1} = \frac{wt(d_j)_{Interact} \; of \; correct \; Instance * e^{-\propto Interact}}{Z_{Interact}}$

- $wt(d_j)_{i+1} = \frac{0.1249 * e^{-0.549}}{0.866} = 0.0832$

- $wt(d_j)_{i+1} = \frac{0.2501 * e^{-0.549}}{0.866} = 0.1667$

- $wt(d_j)_{i+1} = \frac{wt(d_j)_{Interact} \; of \; Incorrect \; Instance * e^{\propto Interact}}{Z_{Interact}}$

- $wt(d_j)_{i+1} = \frac{0.2501 * e^{0.549}}{0.866} = 0.5001$

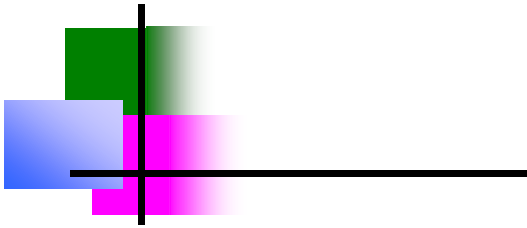
Interactiveness	Predicted Job Offer	Actual Job Offer	Weight
Yes	Yes	Yes	0.0832
No	No	Yes	0.5001
No	No	No	0.1667
No	No	No	0.0832
Yes	Yes	Yes	0.0832
Yes	Yes	Yes	0.0832

- **III. Decision Stump for Practical Knowledge**
- **Step 2 (a):** Train the Decision Stump H_{Pk} with the sample obtained in the previous weak classifier H_{Pk}
- If **Practical Knowledge** is **Good**, the data instance is predicted to have **'Job Offer'** as **'Yes'** else **'No'**.

Practical Knowledge	Predicted Job Offer	Actual Job Offer	Weight
Good	Y	Yes	0.0832
Good	Y	Yes	0.5001
Average	N	No	0.1667
Average		No	0.0832
Good		Yes	0.0832
Good		Yes	0.0832

- **III. Decision Stump for Practical Knowledge**
- **Step 2 (a):** Train the Decision Stump H_{Pk} with the sample obtained in the previous weak classifier H_{Pk}
- If **Practical Knowledge** is **Good**, the data instance is predicted to have **'Job Offer'** as **'Yes'** else **'No'**.

Practical Knowledge	Predicted Job Offer	Actual Job Offer	Weight
Good	Yes	Yes	0.0832
Good	Yes	Yes	0.5001
Average	No	No	0.1667
Average	No	No	0.0832
Good	Yes	Yes	0.0832
Good	Yes	Yes	0.0832



- **No instances** are **misclassified** by this Decision Stump.
- So, no need to change the weights of the data instances

Practical Knowledge	Predicted Job Offer	Actual Job Offer	Weight
Good	Yes	Yes	0.0832
Good	Yes	Yes	0.5001
Average	No	No	0.1667
Average	No	No	0.0832
Good	Yes	Yes	0.0832
Good	Yes	Yes	0.0832

- **IV. Decision Stump for Communication Skill**
- **Step 2 (a):** Train the Decision Stump H_{CS} with the sample obtained in the previous weak classifier H_{CS}
- If **Communication Skill** is **Good**, the data instance is predicted to have **'Job Offer'** as **'Yes'** else **'No'**.

Comm Skill	Predicted Job Offer	Actual Job Offer	Weight
Good	Y	Yes	0.0832
Moderate	N	Yes	0.5001
Moderate		No	0.1667
Good		No	0.0832
Moderate		Yes	0.0832
Moderate		Yes	0.0832

- **IV. Decision Stump for Communication Skill**
- **Step 2 (a):** Train the Decision Stump H_{CS} with the sample obtained in the previous weak classifier H_{CS}
- If **Communication Skill** is **Good**, the data instance is predicted to have **'Job Offer'** as **'Yes'** else **'No'**.

Comm Skill	Predicted Job Offer	Actual Job Offer	Weight
Good	Yes	Yes	0.0832
Moderate	No	Yes	0.5001
Moderate	No	No	0.1667
Good	Yes	No	0.0832
Moderate	No	Yes	0.0832
Moderate	No	Yes	0.0832

- **IV. Decision Stump for Communication Skill**
- **Step 2 (a):** Train the Decision Stump H_{CS} with the sample obtained in the previous weak classifier H_{CS}
- If **Communication Skill** is **Good**, the data instance is predicted to have **'Job Offer'** as **'Yes'** else **'No'**.

Comm Skill	Predicted Job Offer	Actual Job Offer	Weight
Good	Yes	Yes	0.0832
Moderate	No	Yes	0.5001
Moderate	No	No	0.1667
Good	Yes	No	0.0832
Moderate	No	Yes	0.0832
Moderate	No	Yes	0.0832

- **Step 2 (b):** Compute the **weighted error**

  ε_{CS} of H_{CS} on current training dataset T:

- $\varepsilon_i = \sum_{j=1}^{N} H_i(d_j) wt(d_j)$
- $where\ ,\ H_i(d_j) = 0\ for\ Correct\ prediction$
- $H_i(d_j) = 1\ \ for\ Wrong\ prediction$
- $\boldsymbol{\varepsilon_{CS} = 1 * 0.5001 + 3 * 0.0832}$
- $\boldsymbol{\varepsilon_{CS} = 0.7497}$

Comm Skill	Predicted Job Offer	Actual Job Offer	Weight
Good	Yes	Yes	0.0832
Moderate	No	Yes	0.5001
Moderate	No	No	0.1667
Good	Yes	No	0.0832
Moderate	No	Yes	0.0832
Moderate	No	Yes	0.0832

- **Step 2 (c):** Compute the weight of each weak classifier:

$$\alpha_{CS} = \frac{1}{2} \frac{ln(1 - \varepsilon_{CS})}{\varepsilon_{CS}}$$

$$\alpha_{CS} = \frac{1}{2} \frac{ln(1 - 0.7497)}{0.7497}$$

$$\boldsymbol{\alpha_{CS} = -0.5485}$$

Comm Skill	Predicted Job Offer	Actual Job Offer	Weight
Good	Yes	Yes	0.0832
Moderate	No	Yes	0.5001
Moderate	No	No	0.1667
Good	Yes	No	0.0832
Moderate	No	Yes	0.0832
Moderate	No	Yes	0.0832

- **Step 2 (d):** Calculate the normalizing factor Z_{CS}
- $Z_{CS} = wt(Correct\ Classifed\ Instance) * No\ of\ Correct\ Classification * e^{-\alpha cs} + wt(Wrong\ Classifed\ Instance) * No\ of\ Wrong\ Classification * e^{+\alpha cs}$
- $Z_{CS} = 0.0832 * 1 * e^{-(-0.5485)} + 0.1667 * 1 * e^{-(-0.5485)} + 0.5001 * 1 * e^{+(-0.5485)} + 0.0832 * 3 * e^{+(-0.5485)}$
- $Z_{CS} = \mathbf{0.866}$

Comm Skill	Predicted Job Offer	Actual Job Offer	Weight
Good	Yes	Yes	0.0832
Moderate	No	Yes	0.5001
Moderate	No	No	0.1667
Good	Yes	No	0.0832
Moderate	No	Yes	0.0832
Moderate	No	Yes	0.0832

- **Step 2 (e):** Update the weight of all data instances:

- $wt(d_j)_{i+1} = \frac{wt(d_j)_{CS}\ of\ correct\ Instance * e^{-\propto CS}}{Z_{CS}}$

- $wt(d_j)_{i+1} = \frac{0.0832 * e^{-(-0.5485)}}{0.866} = 0.1663$

- $wt(d_j)_{i+1} = \frac{0.1667 * e^{-(-0.5485)}}{0.866} = 0.3331$

- $wt(d_j)_{i+1} = \frac{wt(d_j)_{CS}\ of\ Incorrect\ Instance * e^{\propto CS}}{Z_{CS}}$

- $wt(d_j)_{i+1} = \frac{0.5001 * e^{+(-0.5485)}}{0.866} = 0.3337$

- $wt(d_j)_{i+1} = \frac{0.0832 * e^{+(-0.5485)}}{0.866} = 0.0555$

Comm Skill	Predicted Job Offer	Actual Job Offer	Weight
Good	Yes	Yes	0.0832
Moderate	No	Yes	0.5001
Moderate	No	No	0.1667
Good	Yes	No	0.0832
Moderate	No	Yes	0.0832
Moderate	No	Yes	0.0832

- **Step 2 (e):** Update the weight of all data instances:

- $wt(d_j)_{i+1} = \frac{wt(d_j)_{CS} \text{ of correct Instance} * e^{-\propto CS}}{Z_{CS}}$
- $wt(d_j)_{i+1} = \frac{0.0832 * e^{-(-0.5485)}}{0.866} = 0.1663$
- $wt(d_j)_{i+1} = \frac{0.1667 * e^{-(-0.5485)}}{0.866} = 0.3331$
- $wt(d_j)_{i+1} = \frac{wt(d_j)_{CS} \text{ of Incorrect Instance} * e^{\propto CS}}{Z_{CS}}$
- $wt(d_j)_{i+1} = \frac{0.5001 * e^{+(-0.5485)}}{0.866} = 0.3337$
- $wt(d_j)_{i+1} = \frac{0.0832 * e^{+(-0.5485)}}{0.866} = 0.0555$

Comm Skill	Predicted Job Offer	Actual Job Offer	Weight
Good	Yes	Yes	0.0832
Moderate	No	Yes	0.3337
Moderate	No	No	0.3331
Good	Yes	No	0.0555
Moderate	No	Yes	0.0555
Moderate	No	Yes	0.0555

- Step 3: Compute the final predicted value for each data instance:

Sl. No.	$\alpha_{CGPA} = 0.347$	$\propto_{Interact} = 0.5490$	$\propto_{CommSkill} = -0.5485$	Weighted Avg	Final Prediction
1	Yes	Yes	Yes		
2	No	No	No		
3	Yes	No	No		
4	No	No	Yes		
5	Yes	Yes	No		
6	Yes	Yes	No		

- $H_F(d_j) = \sum_{i=1}^{m} \propto_i * H_i(d_j)$
- $H_F(d_j) = \alpha_{CGPA} * Yes + \propto_{Interact} * Yes + \propto_{CSl} * Yes = 0.347 * 1 + 0.5490 * 1 - 0.5485 * \mathbf{1} = \mathbf{0.3475}$

- Step 3: Compute the final predicted value for each data instance:

Sl. No.	$\alpha_{CGPA} = 0.347$	$\propto_{Interact} = 0.5490$	$\propto_{CommSkill} = -0.5485$	Weighted Avg	Final Prediction
1	Yes	Yes	Yes	**0.3475**	
2	No	No	No	**0.0**	
3	Yes	No	No		
4	No	No	Yes		
5	Yes	Yes	No		
6	Yes	Yes	No		

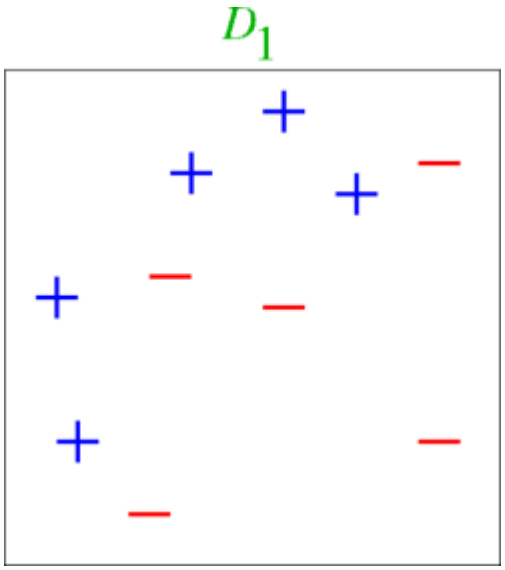
- $H_F(d_j) = \sum_{i=1}^{m} \propto_i * H_i(d_j)$
- $H_F(d_j) = \alpha_{CGPA} * No + \propto_{Interact} * No + \propto_{CSl} * No = 0.347 * 0 + 0.5490 * 0 - 0.5485 * \mathbf{0} = \mathbf{0.0}$

- Step 3: Compute the final predicted value for each data instance:

Sl. No.	$\alpha_{CGPA} = 0.347$	$\propto_{Interact} = 0.5490$	$\propto_{CommSkill} = -0.5485$	Weighted Avg	Final Prediction
1	Yes	Yes	Yes	0.3475	Y
2	No	No	No	0.0	N
3	Yes	No	No	0.347	Y
4	No	No	Yes	-0.5485	N
5	Yes	Yes	No	0.896	Y
6	Yes	Yes	No	0.896	Y

AdaBoost Example

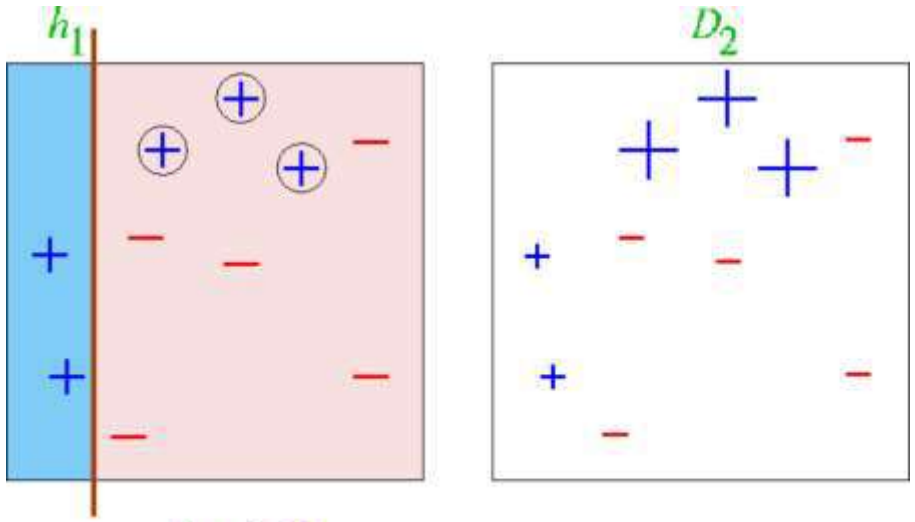
- Training data



[Slide credit: Verma & Thrun]

AdaBoost Example

- Round 1



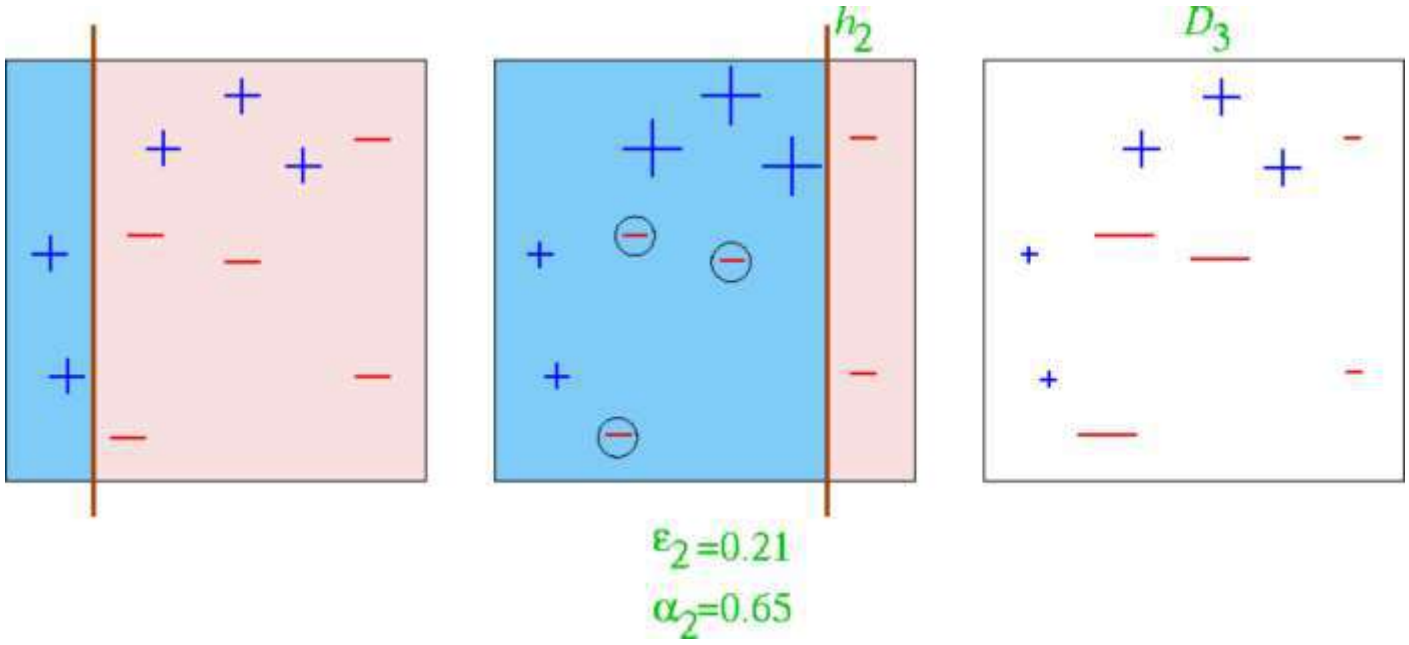
$$\mathbf{w} = \left(\frac{1}{10}, \ldots, \frac{1}{10}\right) \Rightarrow \text{Train a classifier (using } \mathbf{w}) \Rightarrow \text{err}_1 = \frac{\sum_{i=1}^{10} w_i \mathbb{I}\{h_1(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i} = \frac{3}{10}$$

$$\Rightarrow \alpha_1 = \frac{1}{2} \log \frac{1 - \text{err}_1}{\text{err}_1} = \frac{1}{2} \log\left(\frac{1}{0.3} - 1\right) \approx 0.42 \Rightarrow H(\mathbf{x}) = \text{sign}\left(\alpha_1 h_1(\mathbf{x})\right)$$

[Slide credit: Verma & Thrun]

AdaBoost Example

- Round 2



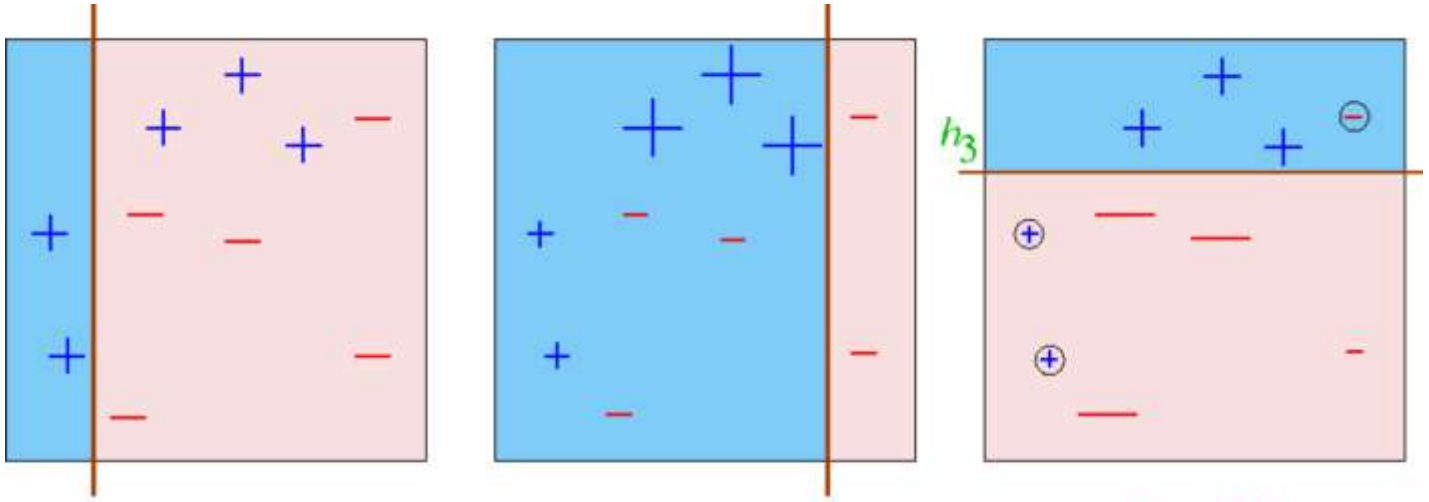
$$\mathbf{w} = \text{updated weights} \Rightarrow \text{Train a classifier (using } \mathbf{w}) \Rightarrow \text{err}_2 = \frac{\sum_{i=1}^{10} w_i \mathbb{I}\{h_2(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i} = 0.21$$

$$\Rightarrow \alpha_2 = \frac{1}{2}\log\frac{1-\text{err}_3}{\text{err}_3} = \frac{1}{2}\log(\frac{1}{0.21} - 1) \approx 0.66 \Rightarrow H(\mathbf{x}) = \text{sign}\left(\alpha_1 h_1(\mathbf{x}) + \alpha_2 h_2(\mathbf{x})\right)$$

[Slide credit: Verma & Thrun]

AdaBoost Example

- Round 3



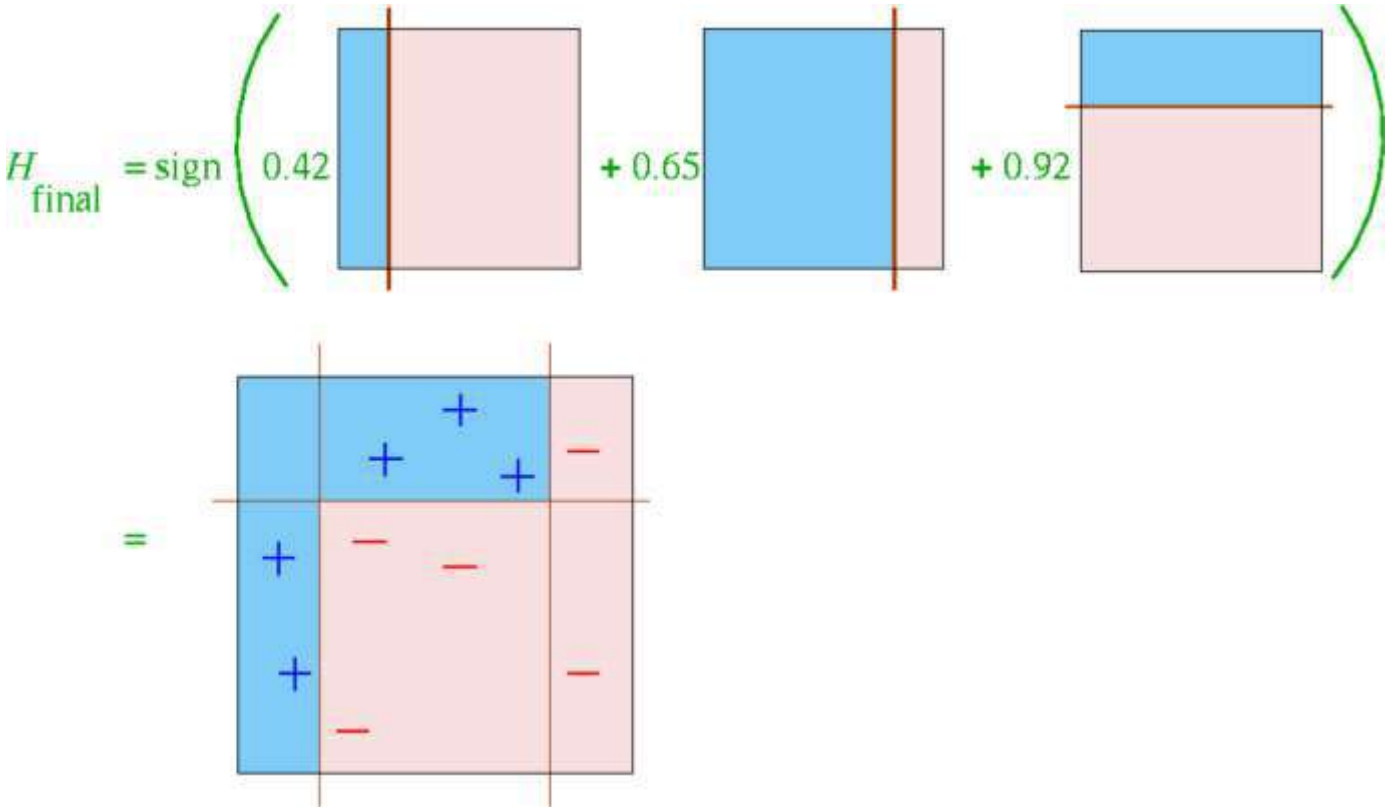
$$\mathbf{w} = \text{updated weights} \Rightarrow \text{Train a classifier (using } \mathbf{w}) \Rightarrow \text{err}_3 = \frac{\sum_{i=1}^{10} w_i \mathbb{I}\{h_3(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i} = 0.14$$

$$\Rightarrow \alpha_3 = \frac{1}{2}\log\frac{1-\text{err}_3}{\text{err}_3} = \frac{1}{2}\log(\frac{1}{0.14} - 1) \approx 0.91 \Rightarrow H(\mathbf{x}) = \text{sign}\left(\alpha_1 h_1(\mathbf{x}) + \alpha_2 h_2(\mathbf{x}) + \alpha_3 h_3(\mathbf{x})\right)$$

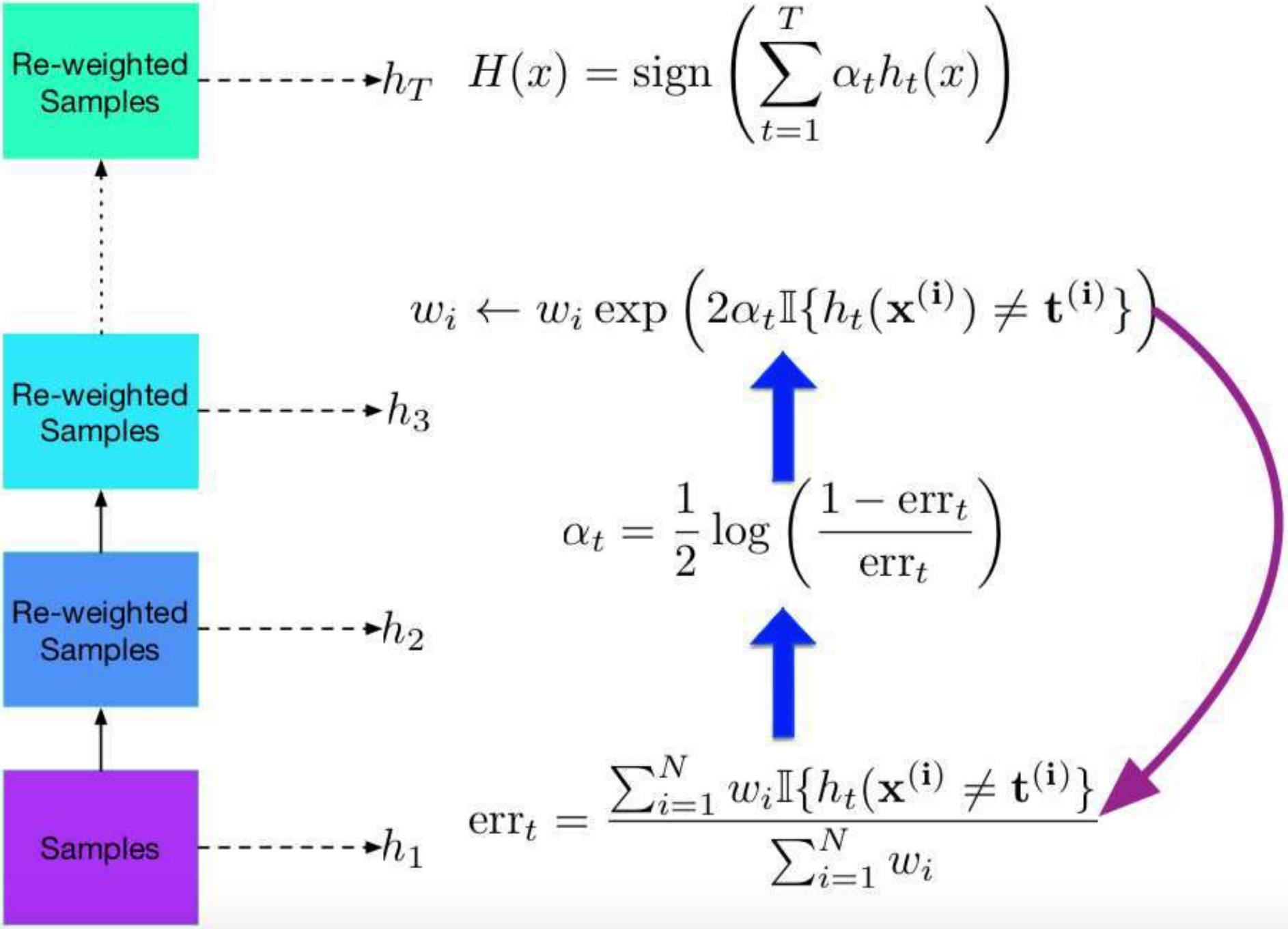
[Slide credit: Verma & Thrun]

AdaBoost Example

- Final classifier



[Slide credit: Verma & Thrun]


Re-weighted Samples
Re-weighted Samples
Re-weighted Samples
Samples
h_T
h_3
h_2
h_1
$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
$w_i \leftarrow w_i \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(\mathbf{i})}) \neq \mathbf{t}^{(\mathbf{i})}\}\right)$
$\alpha_t = \frac{1}{2}\log\left(\frac{1-\text{err}_t}{\text{err}_t}\right)$
$\text{err}_t = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\{h_t(\mathbf{x}^{(\mathbf{i})} \neq \mathbf{t}^{(\mathbf{i})}\}}{\sum_{i=1}^{N} w_i}$

Thank you for your attention