Ensemble learning

Wisdom of the Crowd

- Guess the weight of an ox
- Average of people's votes close to true weight
- Better than most individual members' votes and cattle experts' votes
- Intuitively, the law of large numbers…

Main Idea behind Ensemble Learning

- In ensemble learning, we look at multiple classifiers and combining the output of the multiple classifiers in order to get better prediction or classification accuracy.
- And under certain conditions, where the classifier outputs are independent of each other and make errors in an independent manner, it is possible that by combining the outputs of several classifiers.
- We get a resulting classifier, which is better than any of the constituent classifiers.

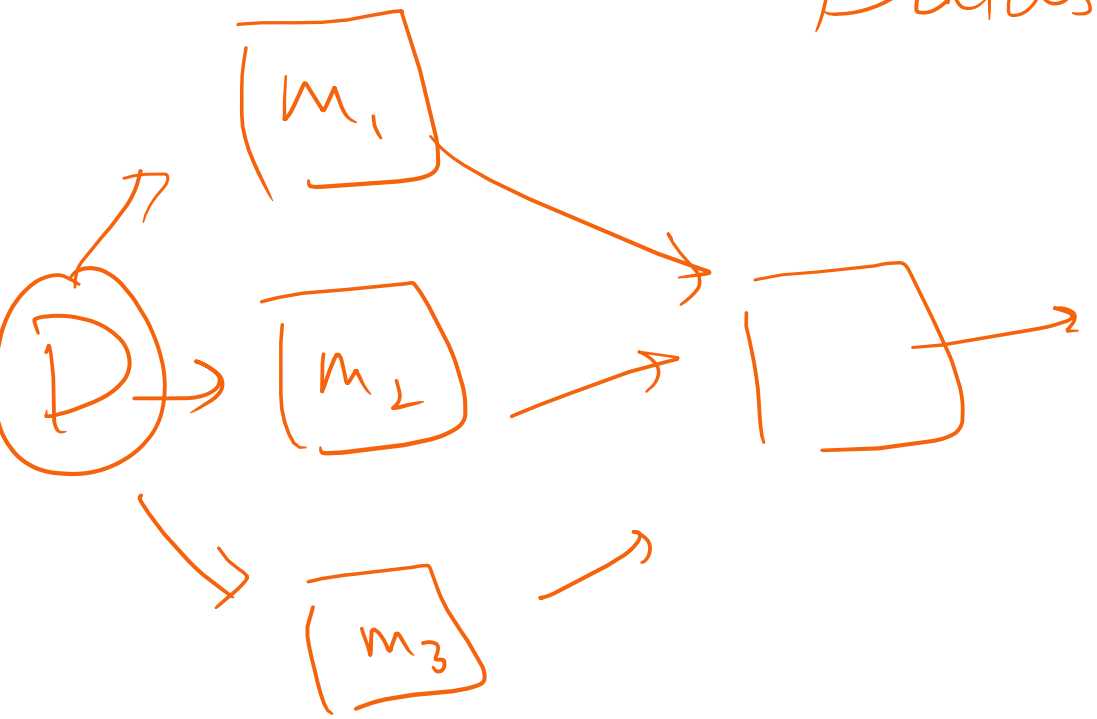
- An ensemble of classifiers must be more accurate than any of its individual members.

  - Base learner
    - Arbitrary learning algorithm which could be used on its own
  - Ensemble
    - A learning algorithm composed of a set of base learners. The base learners may be organized in some structure

Approaches

- Voting
- Bagging
- Boosting
- Stacking

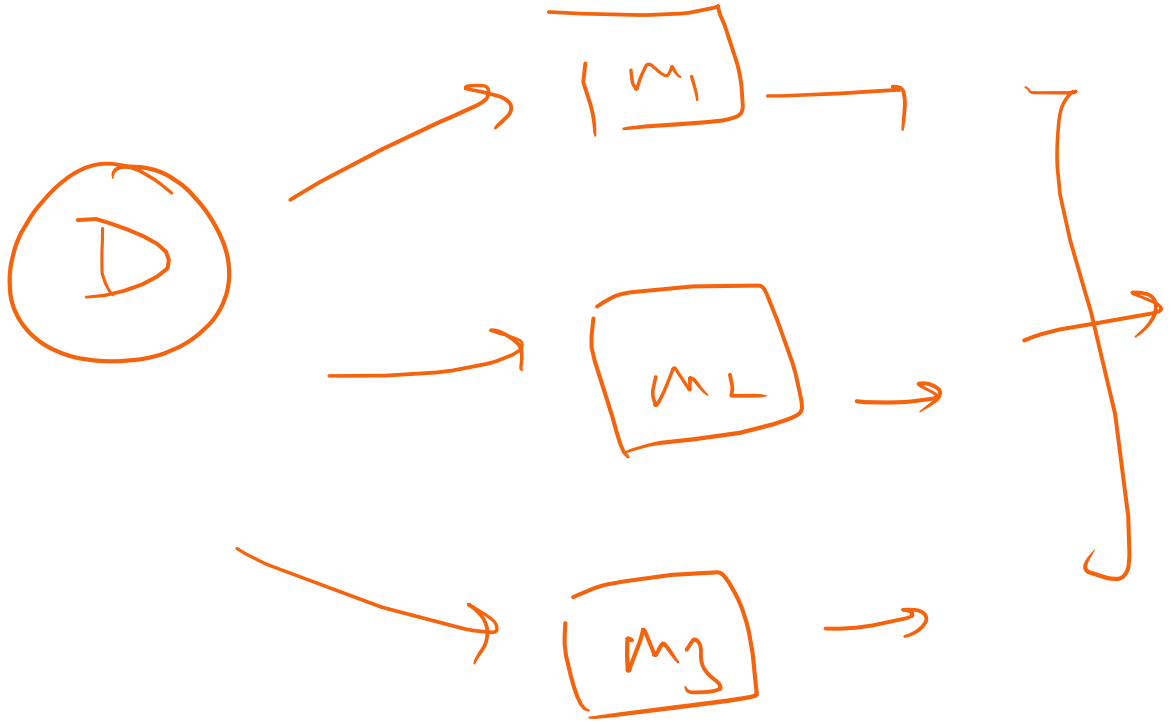
Voting



Dataset (D) same dataset is passed to all the models.

Output Based on maximum votes

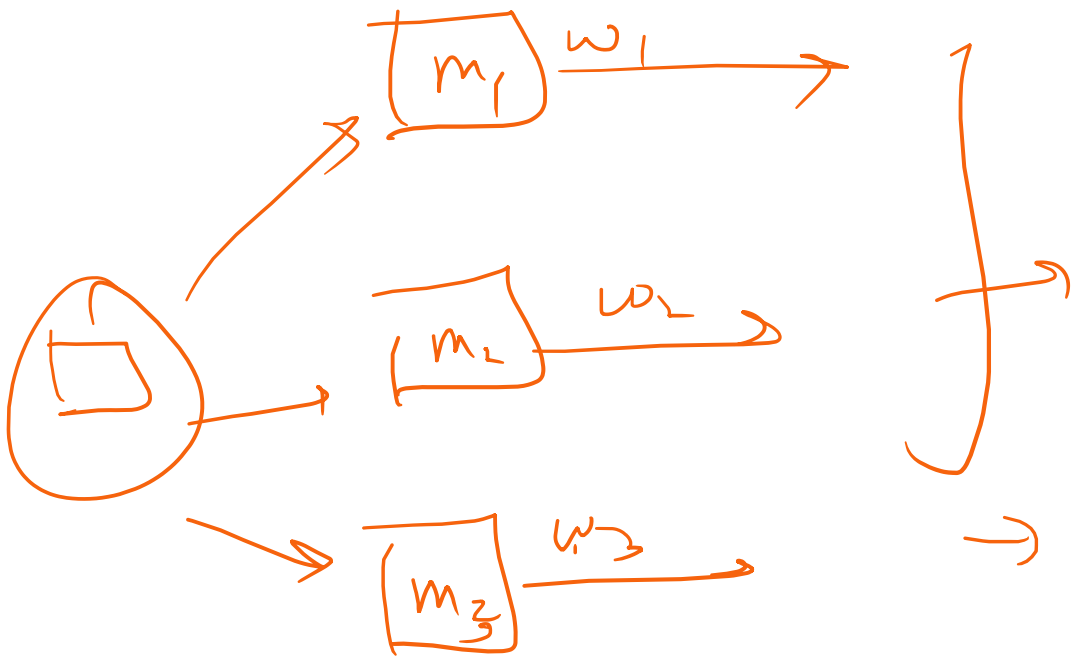
→ For Regression Based voting method



mean of the values will be the final output.

averaging type of voting.

The third type is weighted average where all the models are assigned some weight Based on their importance



→ mean as output

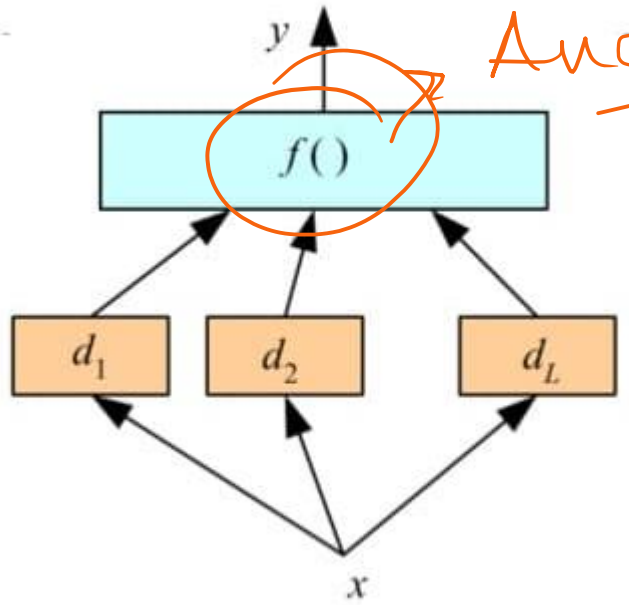
→ similarly we can do, it give different weight to different classifiers.

⇒ Stacking

→ So the input to f() will be the outputs of the models.

Stacking (i)

- In stacked generalization, the combiner **f( ) is another learner** and is not restricted to being a linear combination as in voting.



Another →

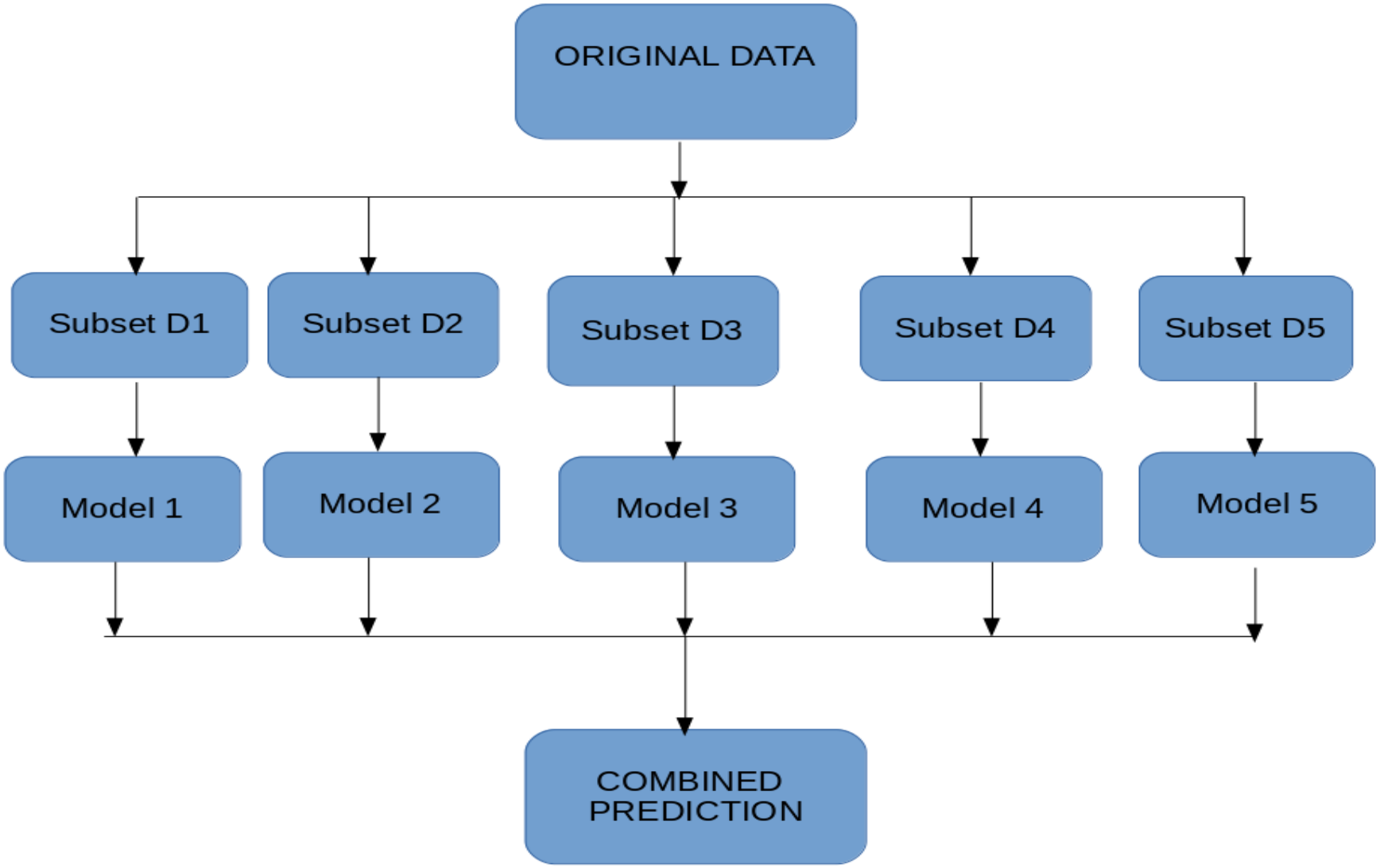
Voting → max

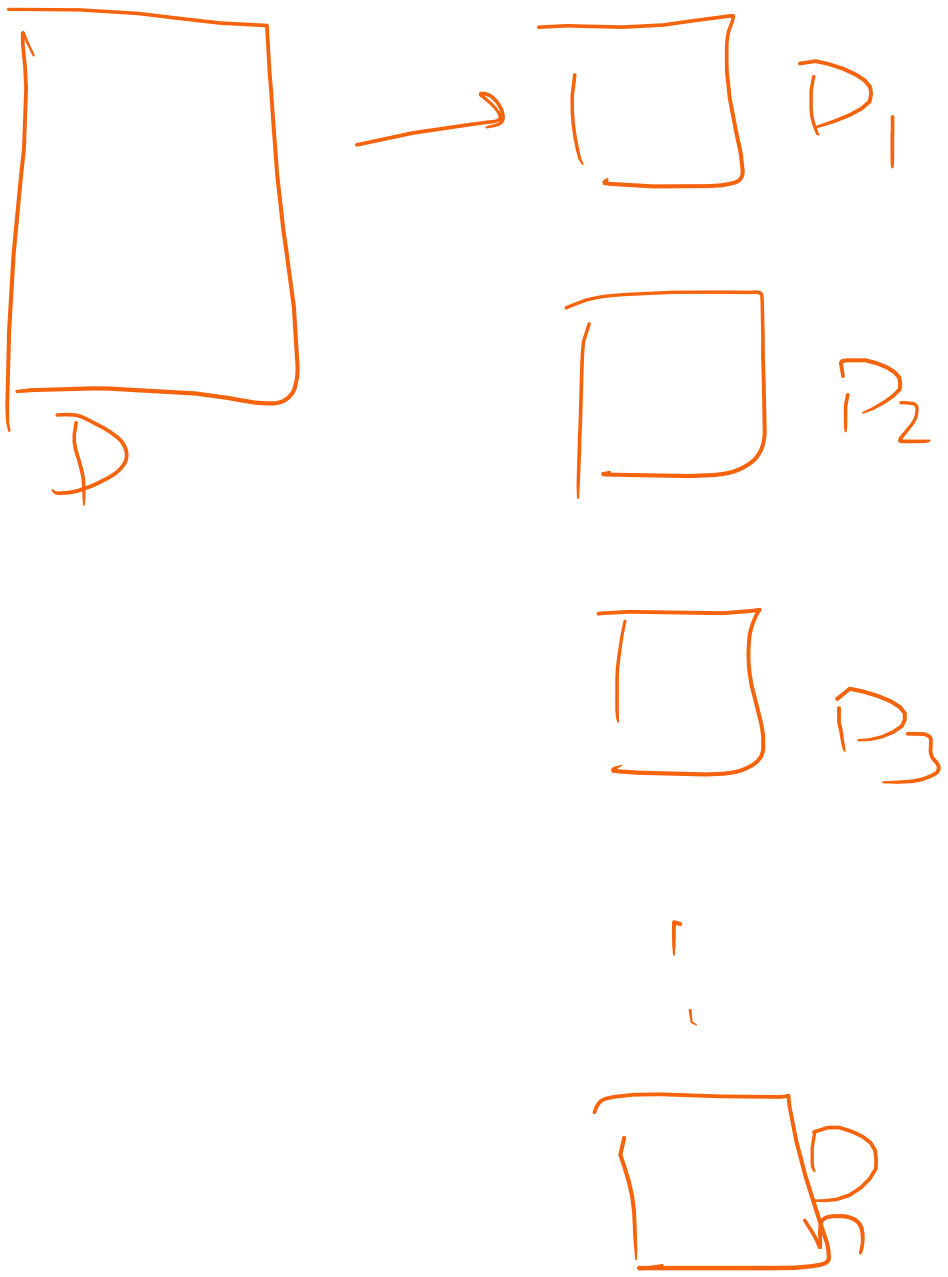
votes

→ output

Here x → data, P_1 P_2 P_3 → models & f() is another learner.

→ Bagging :-


ORIGINAL DATA
Subset D1
Subset D2
Subset D3
Subset D4
Subset D5
Model 1
Model 2
Model 3
Model 4
Model 5
COMBINED PREDICTION

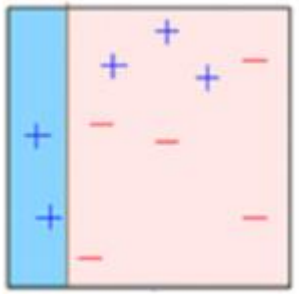


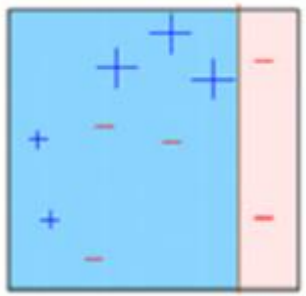
→ Drawing random sample with replacement.

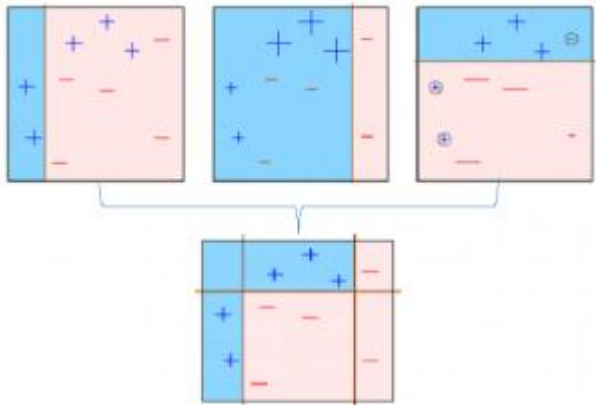
→ then pass through different models → samples

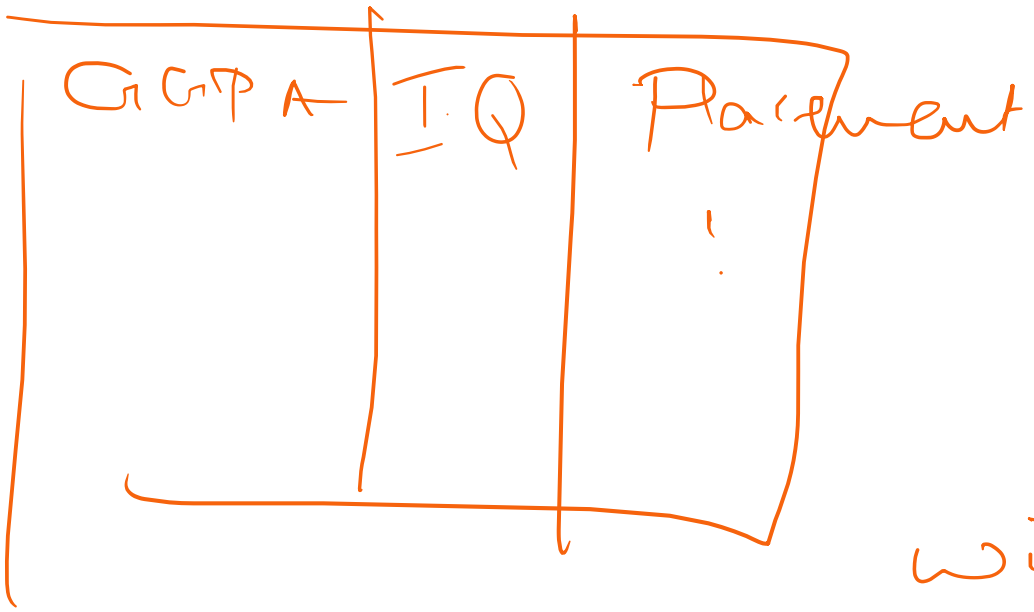
combine the output through voting.

Boosting









→ then ~~there~~ Here some of the data will assign wrong output.

→ So, the weights of those will be increased in the next model (next learner) where the data will passed.

Why does Ensemble work?

What are the disadvantages of using Ensemble methods?

→ ?

Algorithm: Bagging. The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.

Input:

- D, a set of d training tuples;
- k, the number of models in the ensemble;
- a classification learning scheme (decision tree algorithm, naïve Bayesian, etc.).

Output: The ensemble—a composite model, $M*$.

Method:

(1) **for** $i = 1$ to k **do** // create k models:
(2) create bootstrap sample, D_i, by sampling D with replacement;
(3) use D_i and the learning scheme to derive a model, M_i;
(4) **endfor**

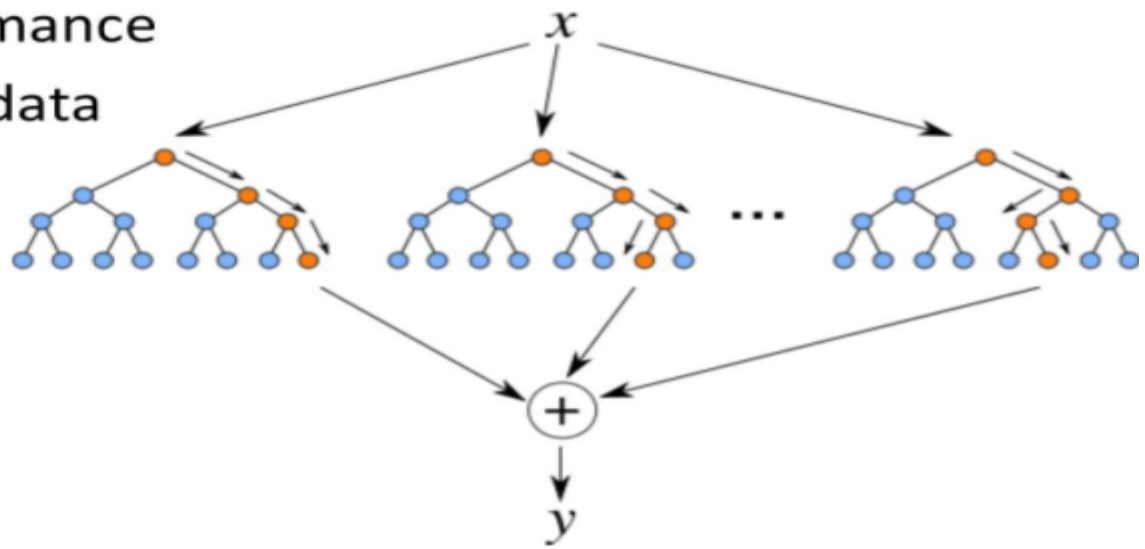
To use the ensemble to classify a tuple, X:

let each of the k models classify X and return the majority vote;

Random Forest

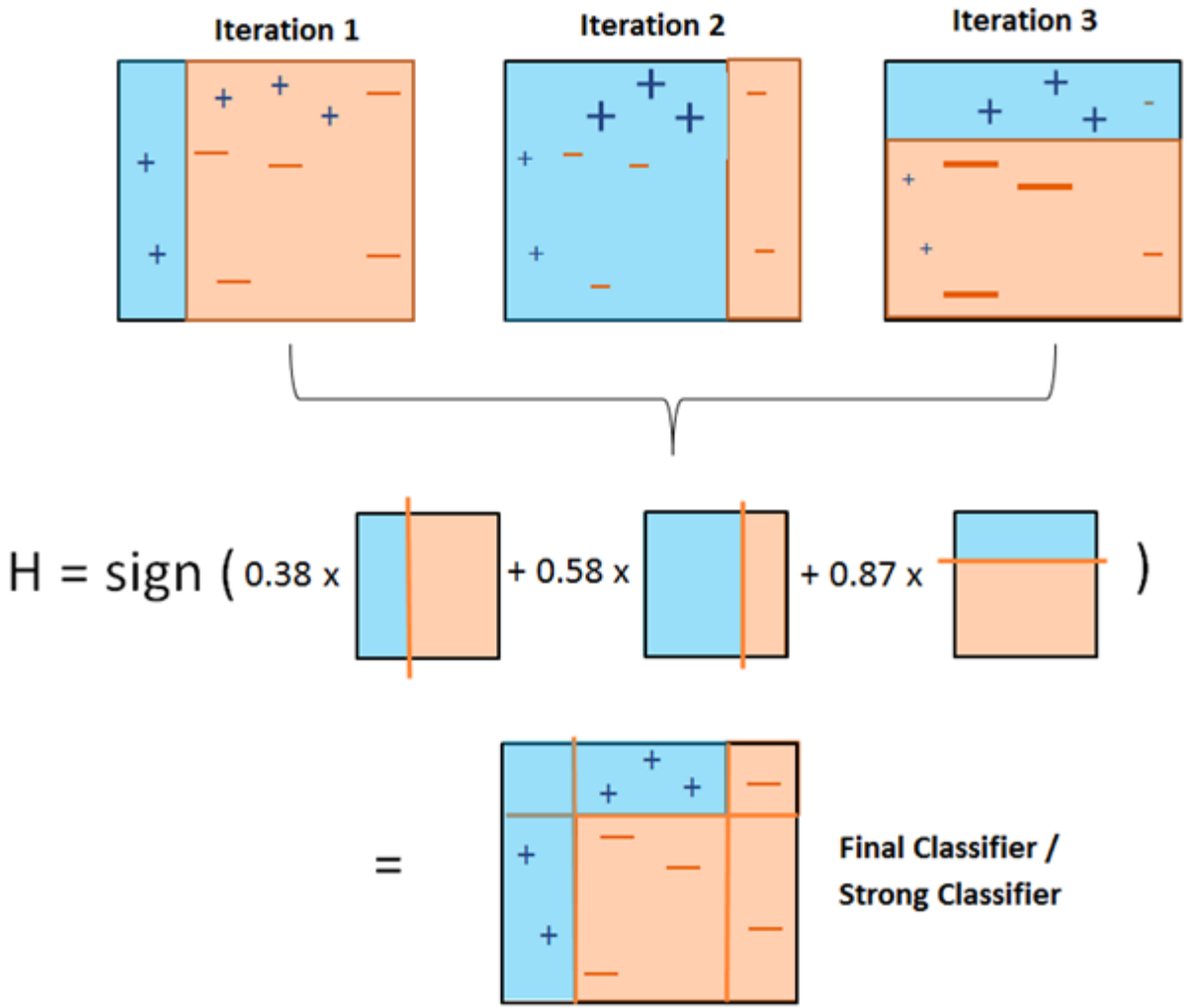
Random Forests

- Construct decision trees on bootstrap replicas
  - Restrict the node decisions to a small subset of features picked randomly for each node
- Do not prune the trees
  - Estimate tree performance on out-of-bootstrap data
- Average the output of all trees



[Based on slide by Léon Bottou]

AdaBoost Classifier Working Principle with Decision Stump as a Base Classifier



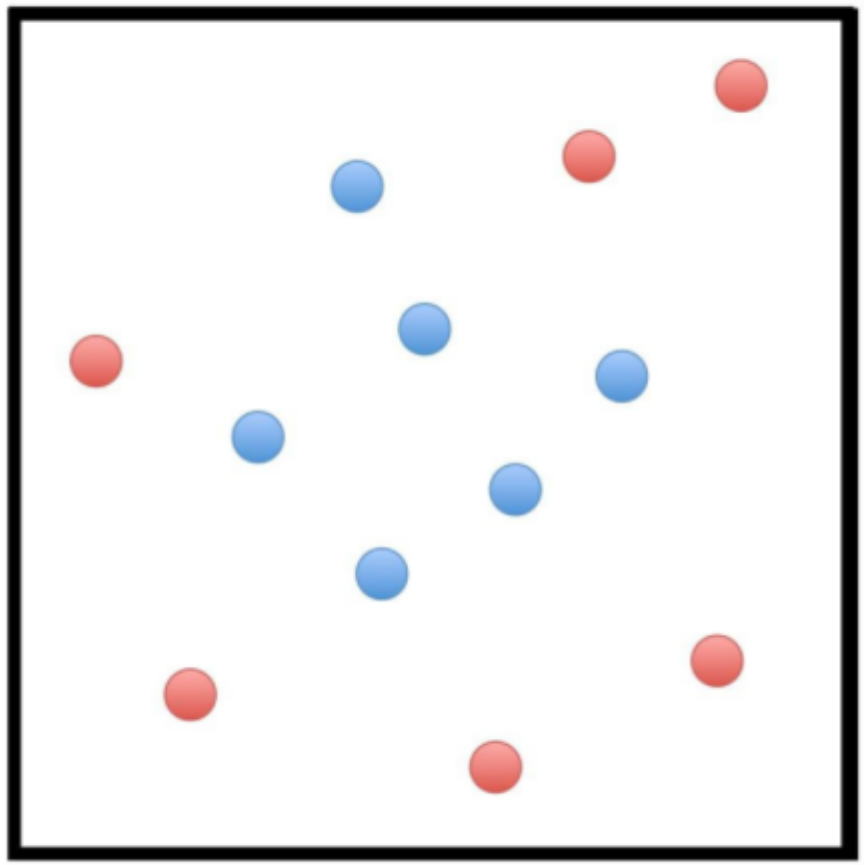
AdaBoost

1: Initialize a vector of n uniform weights $\mathbf{w}_1$
2: **for** $t = 1, \ldots, T$
3: Train model h_t on X, y with weights $\mathbf{w}_t$
4: Compute the weighted training error of h_t
5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution
8: **end for**
9: **Return** the hypothesis

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



- Size of point represents the instance's weight

AdaBoost

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = \left[\frac{1}{n}, \ldots, \frac{1}{n}\right]$

2: **for** $t = 1, \ldots, T$

3: Train model h_t on X, y with instance weights $\mathbf{w}_t$

4: Compute the weighted training error rate of h_t:

$$\epsilon_t = \sum_{i:y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

Error is the sum the weights of all misclassified instances

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

6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right) \quad \forall i = 1, \ldots, n$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^{n} w_{t+1,j}} \quad \forall i = 1, \ldots, n$$

8: **end for**

9: **Return** the hypothesis

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

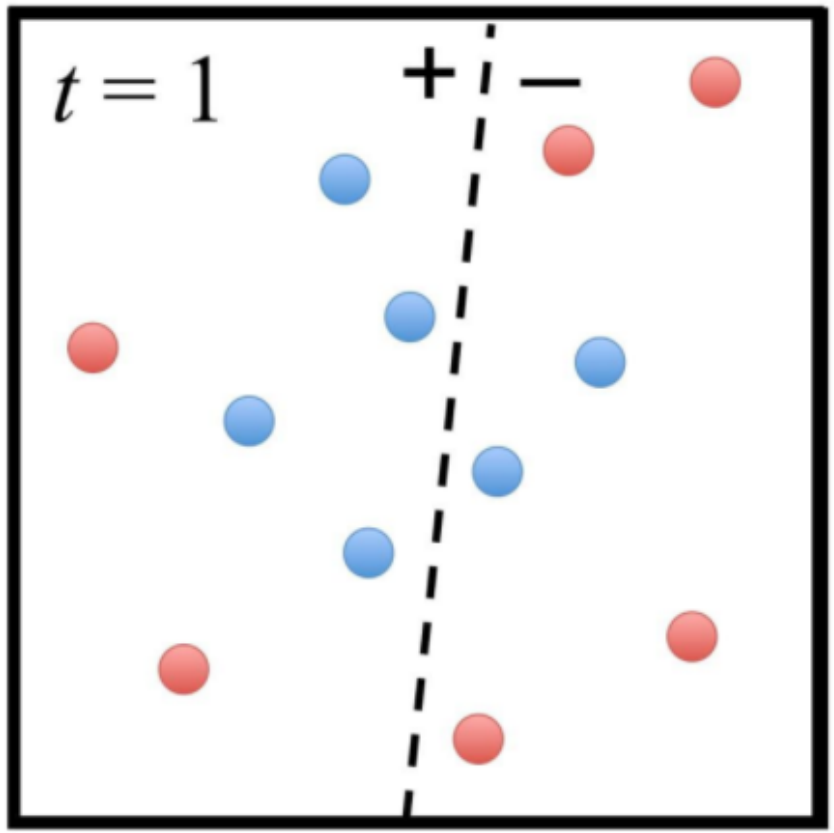
AdaBoost

1: Initialize a vector of n uniform weights $\mathbf{w}_1$
2: **for** $t = 1, \ldots, T$
3: Train model h_t on X, y with weights $\mathbf{w}_t$
4: Compute the weighted training error of h_t
5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution
8: **end for**
9: **Return** the hypothesis

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



- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

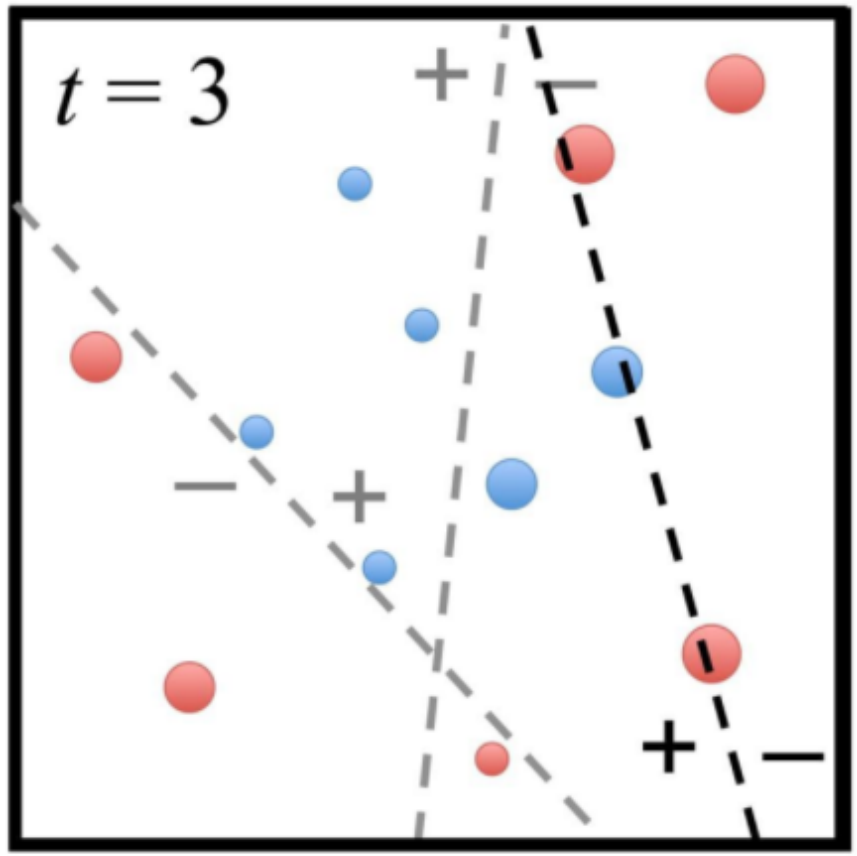
AdaBoost

1: Initialize a vector of n uniform weights $\mathbf{w}_1$
2: **for** $t = 1, \ldots, T$
3: Train model h_t on X, y with weights $\mathbf{w}_t$
4: Compute the weighted training error of h_t
5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution
8: **end for**
9: **Return** the hypothesis

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



- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (β_t grows as ϵ_t gets smaller)

AdaBoost

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = \left[\frac{1}{n}, \ldots, \frac{1}{n}\right]$

2: **for** $t = 1, \ldots, T$

3: Train model h_t on X, y with instance weights $\mathbf{w}_t$

4: Compute the weighted training error rate of h_t:

$$\epsilon_t = \sum_{i:y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

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

6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right) \quad \forall i = 1, \ldots$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^{n} w_{t+1,j}} \quad \forall i = 1, \ldots, n$$

8: **end for**

9: **Return** the hypothesis

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

Member classifiers with less error are given more weight in the final ensemble hypothesis

Final prediction is a weighted combination of each member's prediction

x1	x2	Decision
2	3	true
2.1	2	true
4.5	6	true
4	3.5	false
3.5	1	false
5	7	true
5	3	false
6	5.5	true
8	6	false
8	2	false

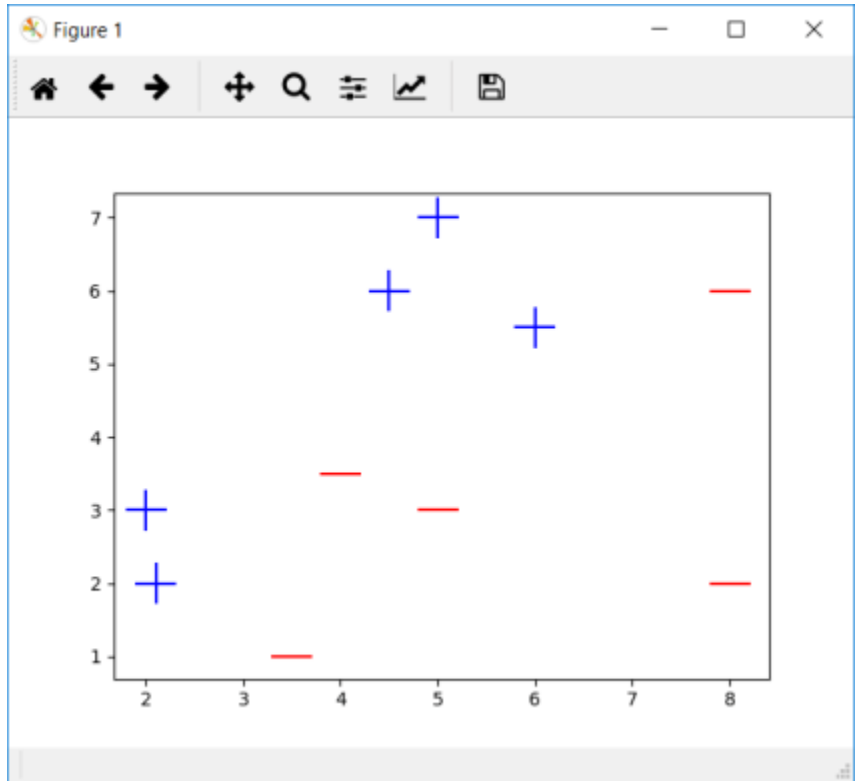
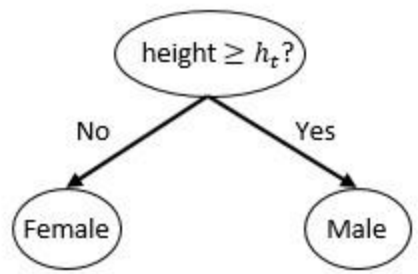


Figure 1
7
6
5
4
3
2
1
2
3
4
5
6
7
8


height $\geq h_t$?
No
Yes
Female
Male

x1	x2	actual	weight	weighted_actual
2	3	1	0.1	0.1
2	2	1	0.1	0.1
4	6	1	0.1	0.1
4	3	-1	0.1	-0.1
4	1	-1	0.1	-0.1
5	7	1	0.1	0.1
5	3	-1	0.1	-0.1
6	5	1	0.1	0.1
8	6	-1	0.1	-0.1
8	2	-1	0.1	-0.1

x1	x2	actual	weight	weighted_actual	prediction	loss	weight * loss
2	3	1	0.1	0.1	1	0	0
2	2	1	0.1	0.1	1	0	0
4	6	1	0.1	0.1	-1	1	0.1
4	3	-1	0.1	-0.1	-1	0	0
4	1	-1	0.1	-0.1	-1	0	0
5	7	1	0.1	0.1	-1	1	0.1
5	3	-1	0.1	-0.1	-1	0	0
6	5	1	0.1	0.1	-1	1	0.1
8	6	-1	0.1	-0.1	-1	0	0
8	2	-1	0.1	-0.1	-1	0	0

- Sum of weight times loss column stores the total error. It is 0.3 in this case. Here, we'll define a new variable alpha. It stores logarithm (1 – epsilon)/epsilon to the base e over 2.
- alpha = ln[(1-epsilon)/epsilon] / 2 = ln[(1 – 0.3)/0.3] / 2
- alpha = 0.42
- $w_{i+1} = w_i$ * math.exp(-alpha * actual * prediction) where i refers to instance number.

x1	x2	actual	weight	prediction	w_(i+1)	norm(w_(i+1))
2	3	1	0.1	1	0.065	0.071
2	2	1	0.1	1	0.065	0.071
4	6	1	0.1	-1	0.153	0.167
4	3	-1	0.1	-1	0.065	0.071
4	1	-1	0.1	-1	0.065	0.071
5	7	1	0.1	-1	0.153	0.167
5	3	-1	0.1	-1	0.065	0.071
6	5	1	0.1	-1	0.153	0.167
8	6	-1	0.1	-1	0.065	0.071
8	2	-1	0.1	-1	0.065	0.071

x1	x2	actual	weight	weighted_actual
2	3	1	0.071	0.071
2	2	1	0.071	0.071
4	6	1	0.167	0.167
4	3	-1	0.071	-0.071
4	1	-1	0.071	-0.071
5	7	1	0.167	0.167
5	3	-1	0.071	-0.071
6	5	1	0.167	0.167
8	6	-1	0.071	-0.071
8	2	-1	0.071	-0.071

x1	x2	actual	weight	prediction	loss	weight * loss
2	3	1	0.071	-1	1	0.071
2	2	1	0.071	-1	1	0.071
4	6	1	0.167	1	0	0.000
4	3	-1	0.071	-1	0	0.000
4	1	-1	0.071	-1	0	0.000
5	7	1	0.167	1	0	0.000
5	3	-1	0.071	-1	0	0.000
6	5	1	0.167	1	0	0.000
8	6	-1	0.071	1	1	0.071
8	2	-1	0.071	-1	0	0.000

- epsilon = 0.21, alpha = 0.65

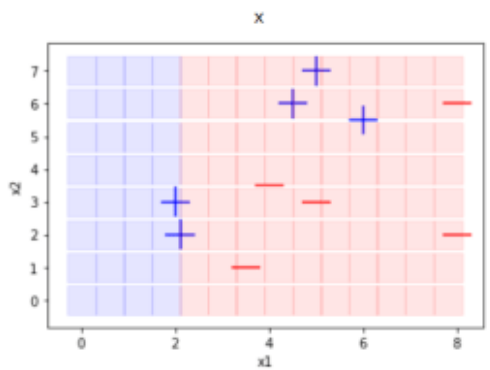
x1	x2	actual	weight	prediction	w_(i+1)	norm(w_(i+1))
2	3	1	0.071	-1	0.137	0.167
2	2	1	0.071	-1	0.137	0.167
4	6	1	0.167	1	0.087	0.106
4	3	-1	0.071	-1	0.037	0.045
4	1	-1	0.071	-1	0.037	0.045
5	7	1	0.167	1	0.087	0.106
5	3	-1	0.071	-1	0.037	0.045
6	5	1	0.167	1	0.087	0.106
8	6	-1	0.071	1	0.137	0.167
8	2	-1	0.071	-1	0.037	0.045

x1	x2	actual	weight	prediction	loss	w * loss	w_(i+1)	norm(w_(i+1))
2	3	1	0.167	1	0	0.000	0.114	0.122
2	2	1	0.167	1	0	0.000	0.114	0.122
4	6	1	0.106	-1	1	0.106	0.155	0.167
4	3	-1	0.045	-1	0	0.000	0.031	0.033
4	1	-1	0.045	-1	0	0.000	0.031	0.033
5	7	1	0.106	-1	1	0.106	0.155	0.167
5	3	-1	0.045	-1	0	0.000	0.031	0.033
6	5	1	0.106	-1	1	0.106	0.155	0.167
8	6	-1	0.167	-1	0	0.000	0.114	0.122
8	2	-1	0.045	-1	0	0.000	0.031	0.033

- Epsilon = 0.31 alpha = 0.38

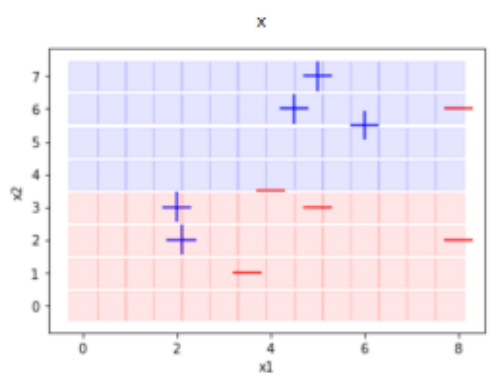
- Round 4: epsilon = 0.10, alpha = 1.10
- For example, prediction of the 1st instance will be
- 0.42 x 1 + 0.65 x (-1) + 0.38 x 1 + 1.1 x 1 = 1.25
- And we will apply sign function
- Sign(0.25) = +1 … true which is correctly classified

0.4236489301936017



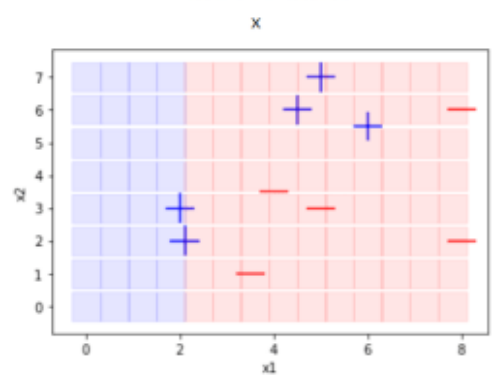
+

0.6496414920651304



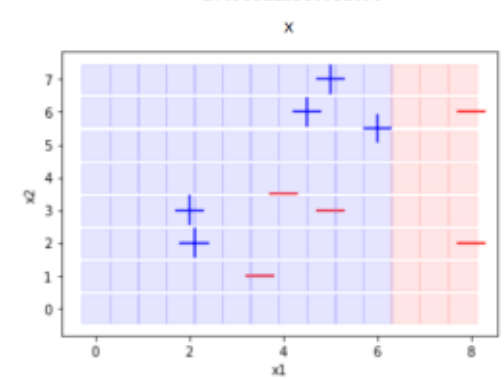
+

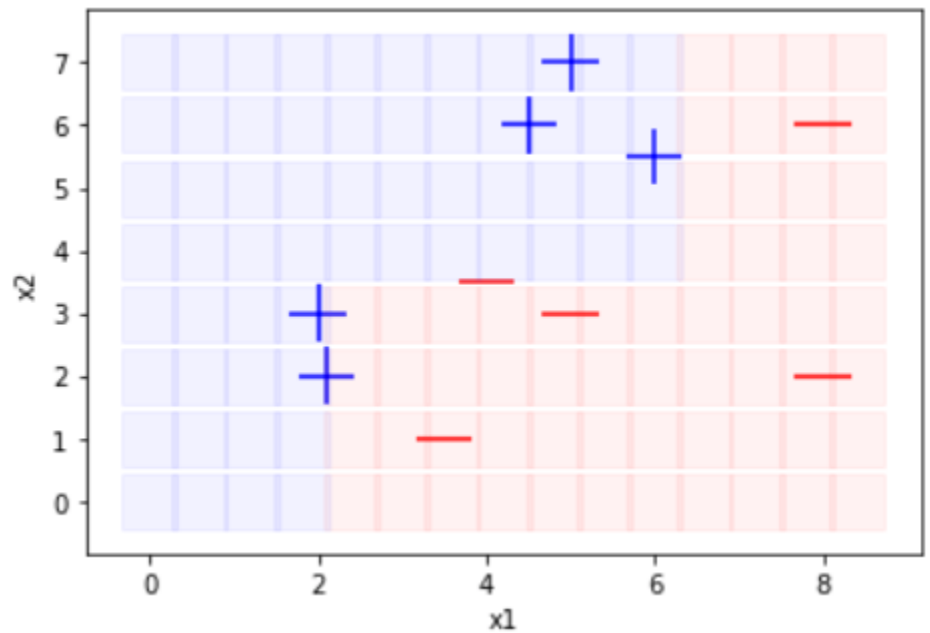
0.38107002602344847



+

1.0986122886681096




7
6
5
4
3
2
1
0
x2
0
2
4
6
8
x1

References

- https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/
- https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/
- https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f
- https://www.youtube.com/watch?v=nelJ3svz0_o&t=2008s
- https://www.youtube.com/watch?v=bHK1fE_BUms
- https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/