CSCE 636 Neural Networks (Deep Learning)

Lecture 19: Ensemble Learning

Anxiao (Andrew) Jiang

Based on the interesting lecture of Prof. Hung-yi Lee "Ensemble"
https://www.youtube.com/watch?v=tH9FH1DH5n0&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=32

Ensemble Learning

Framework of Ensemble

Basic idea:
Build multiple models for the same application,
make them collaborate to achieve better performance.

In other words: use Team Work.

Framework of Ensemble

- Get a set of classifiers
    - $f_1(x), f_2(x), f_3(x)$, ……

They should be diverse.

Framework of Ensemble

- Get a set of classifiers
  - $f_1(x), f_2(x), f_3(x)$, ……

They should be diverse.

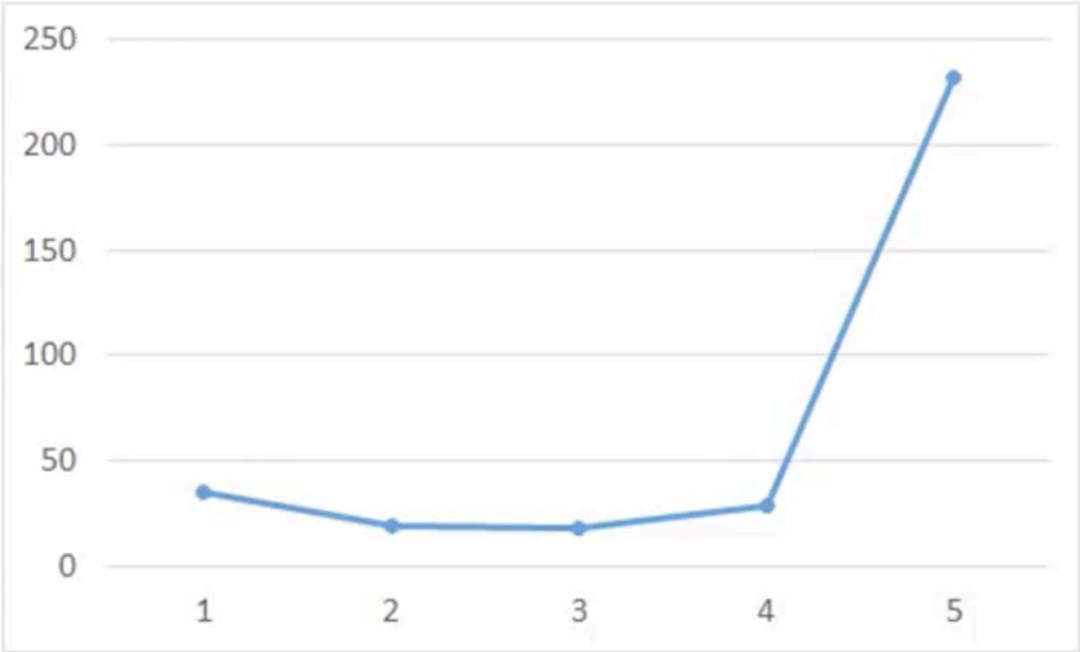
- Aggregate the classifiers (*properly*)

Ensemble is a popular technique for winning machine learning competitions.
It can improve the performance to the next level.
Requirement: need to train multiple models.

Ensemble: Bagging

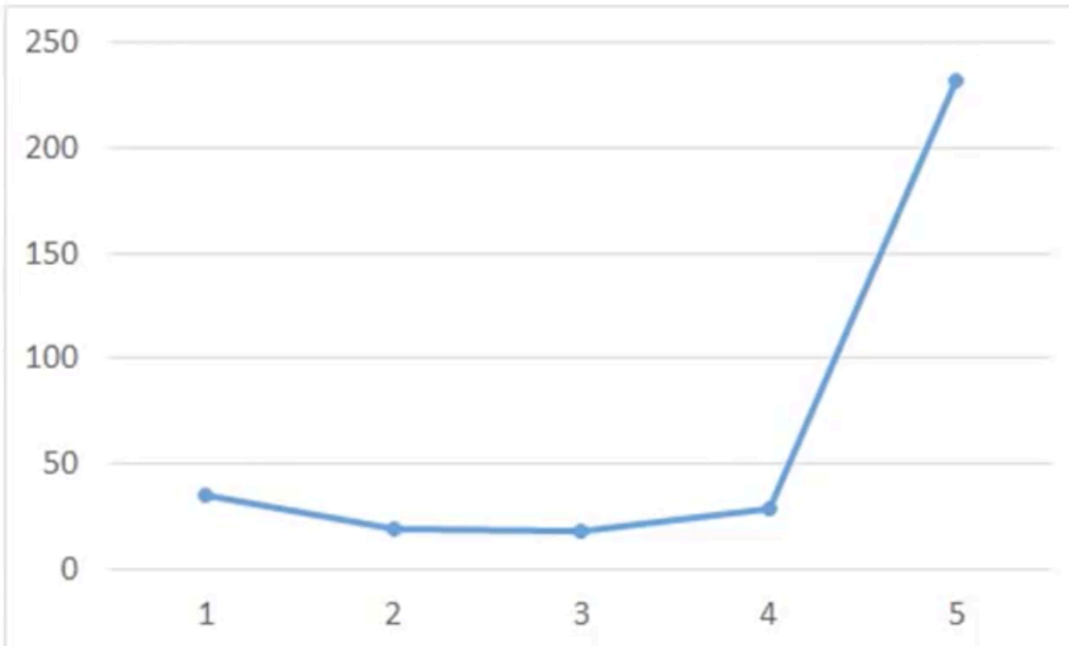
Combine multiple complex models

Review: Bias v.s. Variance



Simple models: large bias, small variance
Complex models: small bias, large variance

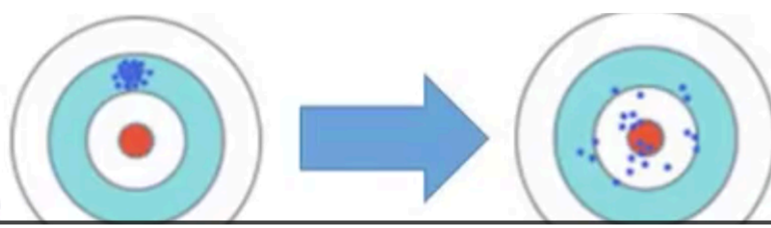
Review: Bias v.s. Variance



Simple Model:

Large Bias

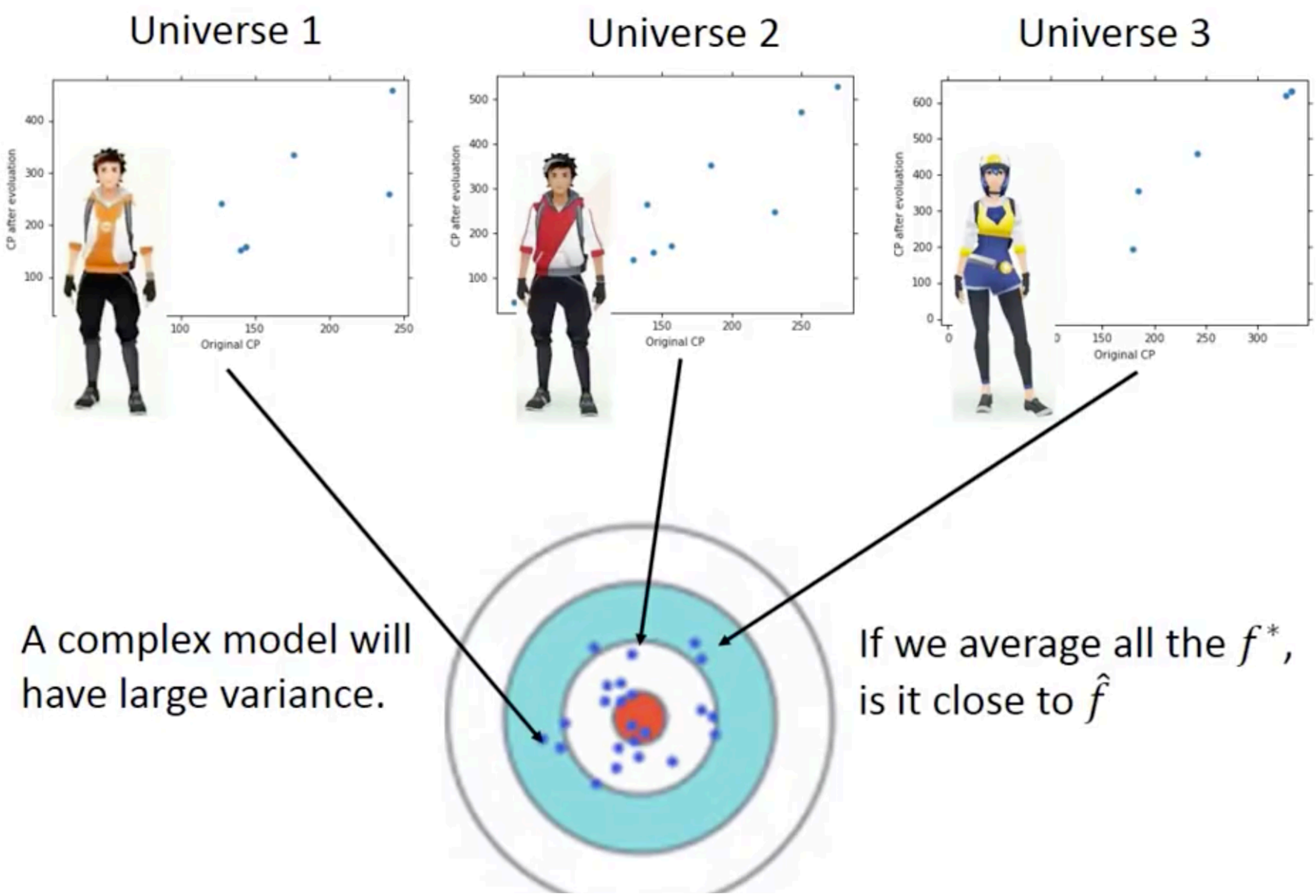
Small Variance



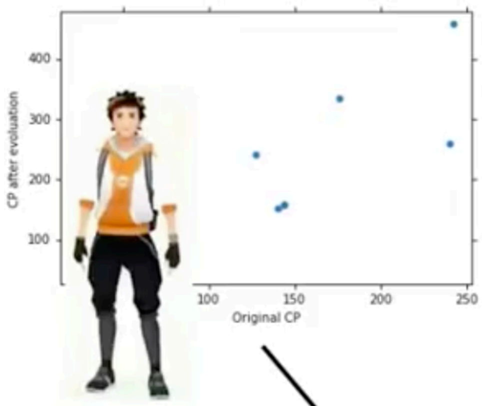
Small Bias

Large Variance

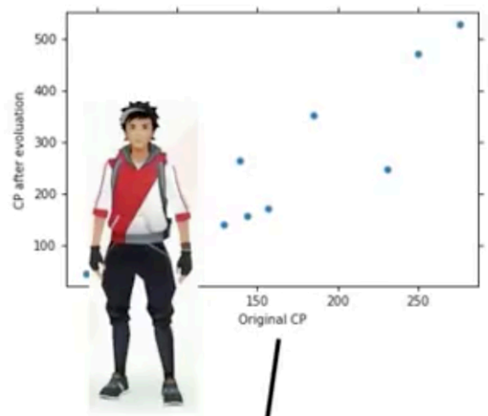
Complex Model


Universe 1
Universe 2
Universe 3
CP after evolution
Original CP
A complex model will have large variance.
If we average all the f^*, is it close to $\hat{f}$

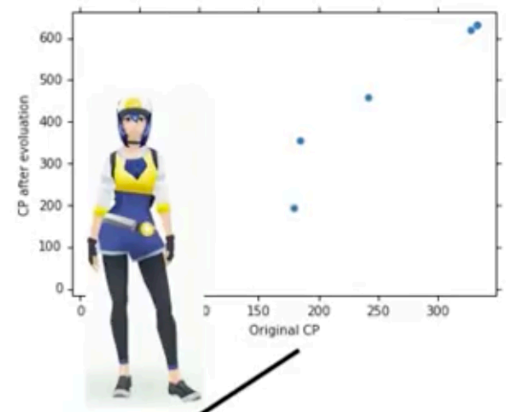
Universe 1



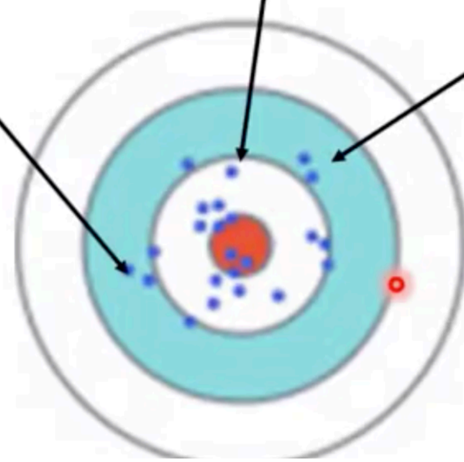
Universe 2



Universe 3



A complex model will have large variance.



If we average all the f^*, is it close to $\hat{f}$

$$E[f^*] = \hat{f}$$

Bagging

An idea:
Create multiple “different” datasets, and train one model for each dataset; then combine them.

But how to create different datasets?

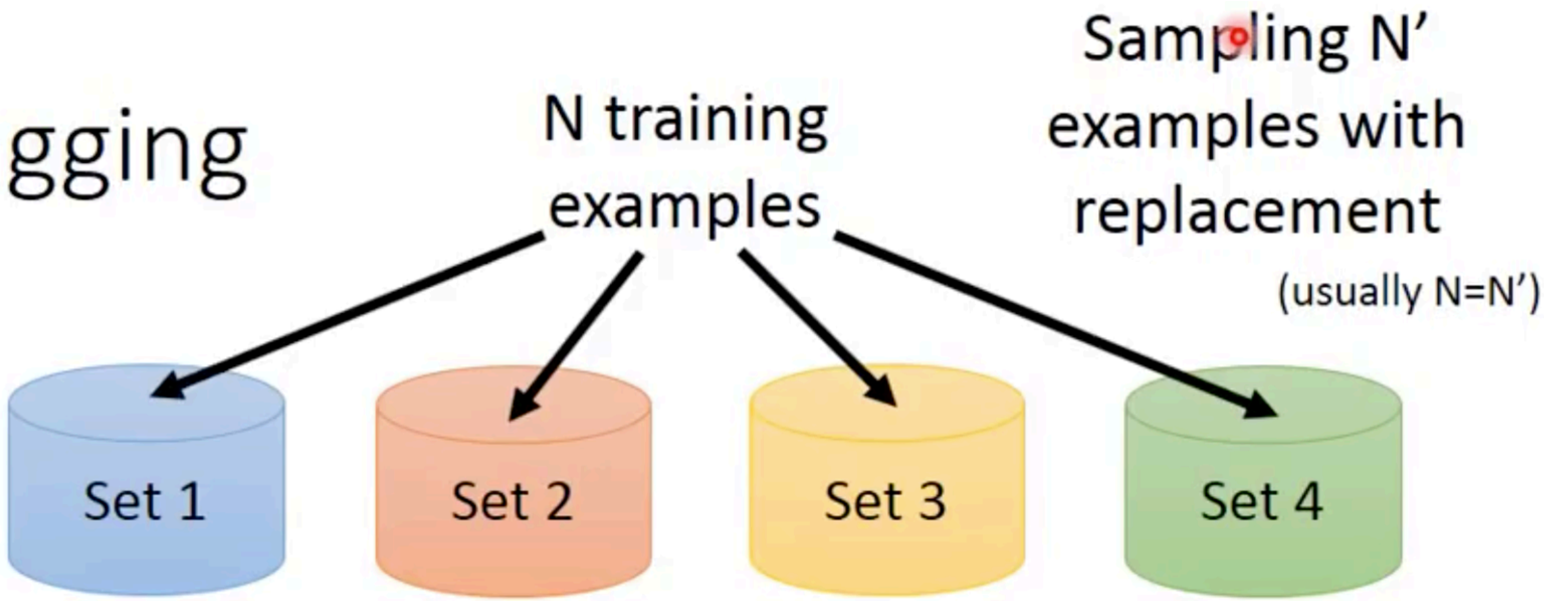
Bagging

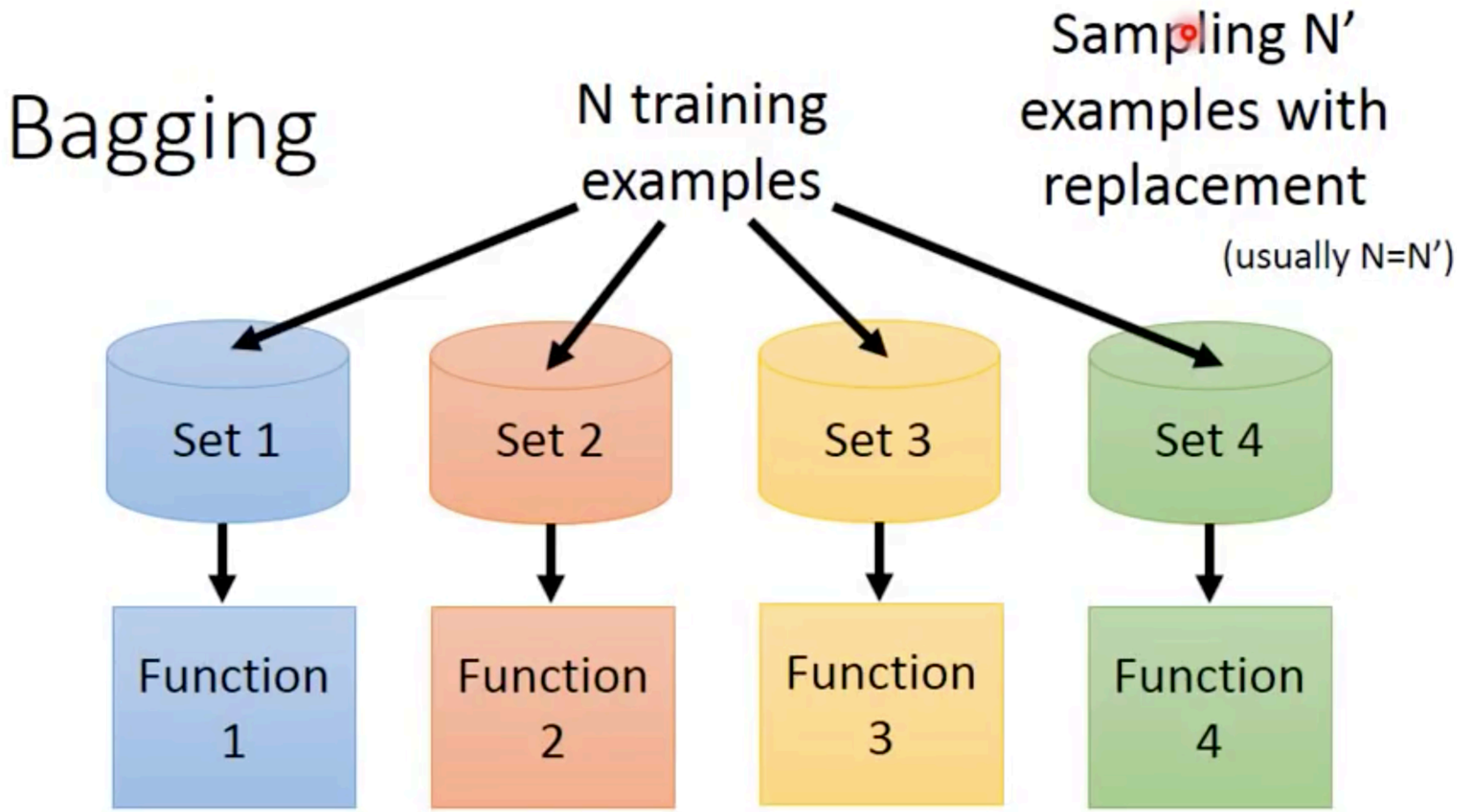
N training examples

Sampling N’ examples with replacement

(usually N=N’)

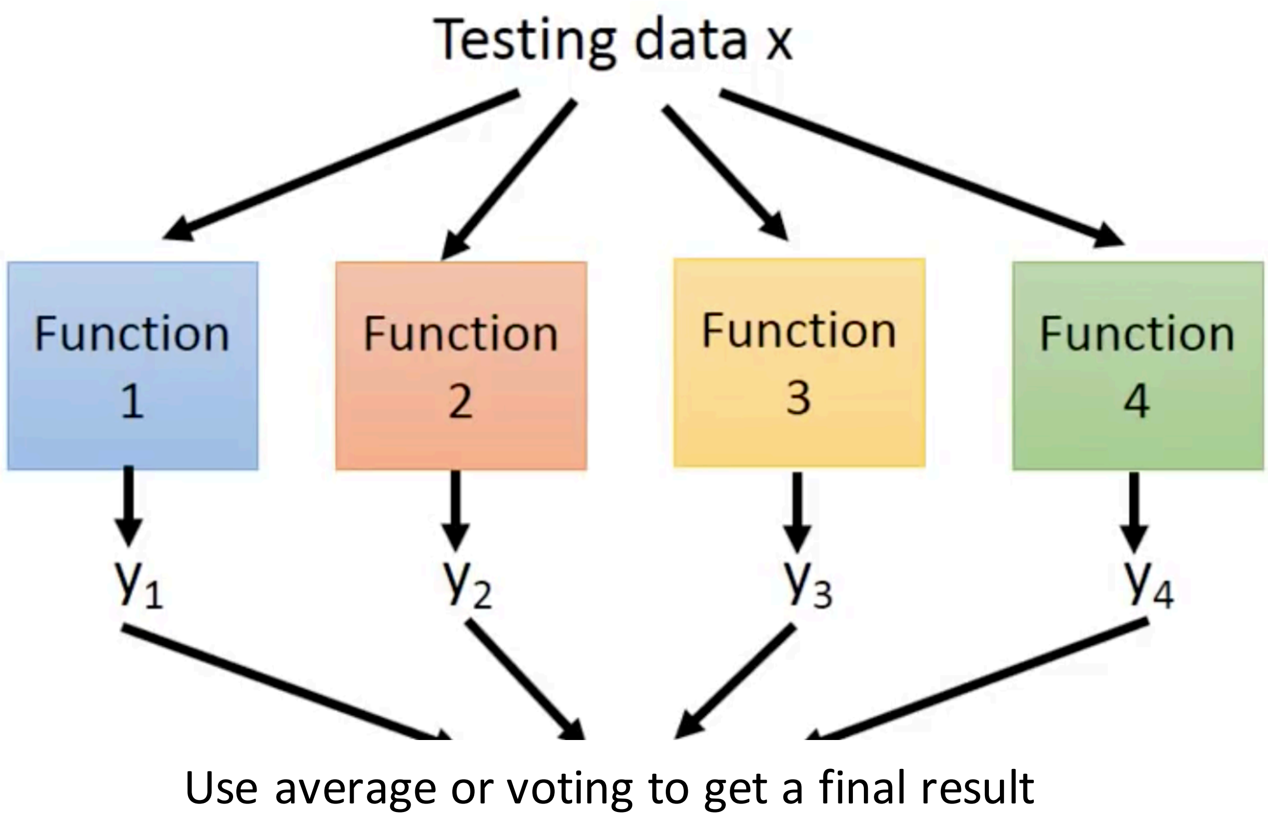
Bagging


N training examples
Sampling N' examples with replacement
(usually N=N')
Set 1
Set 2
Set 3
Set 4



Use a complex model to train 4 classification functions for the 4 datasets.

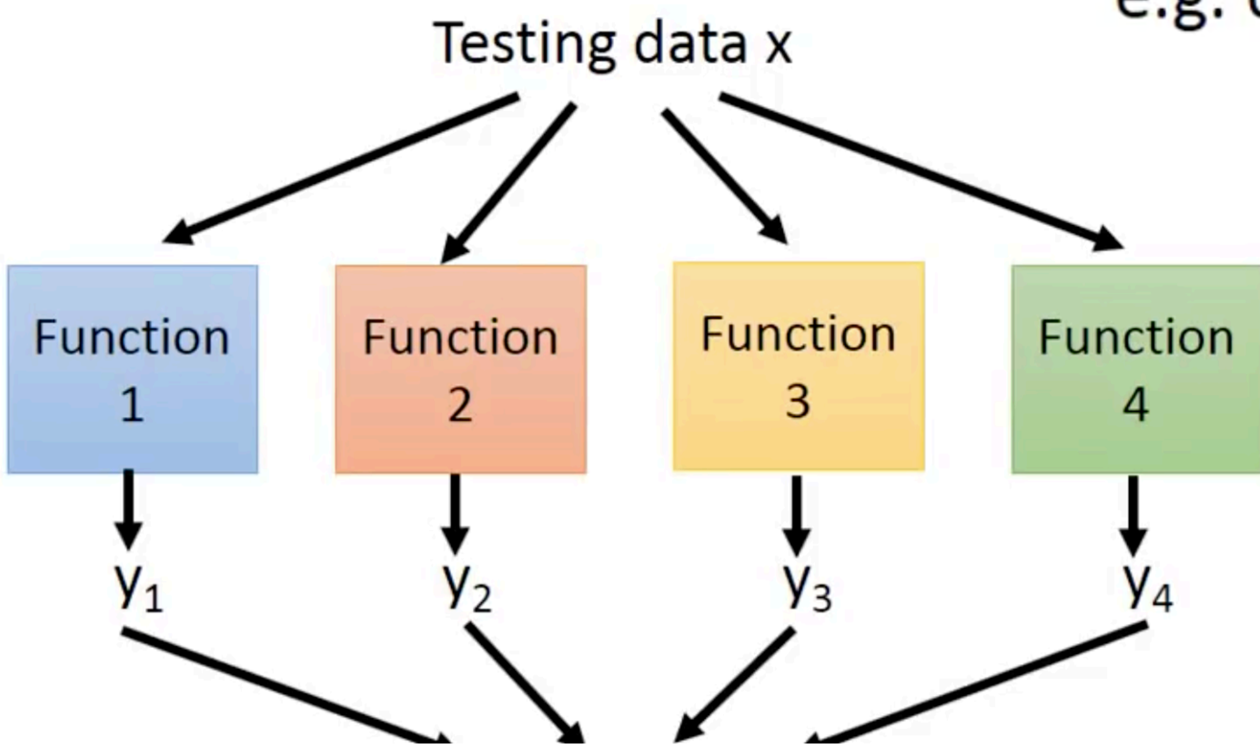
Bagging


Testing data x
Function 1
Function 2
Function 3
Function 4
y_1
y_2
y_3
y_4
Use average or voting to get a final result

Bagging

This approach would be helpful when your model is complex, easy to overfit.

e.g. decision tree



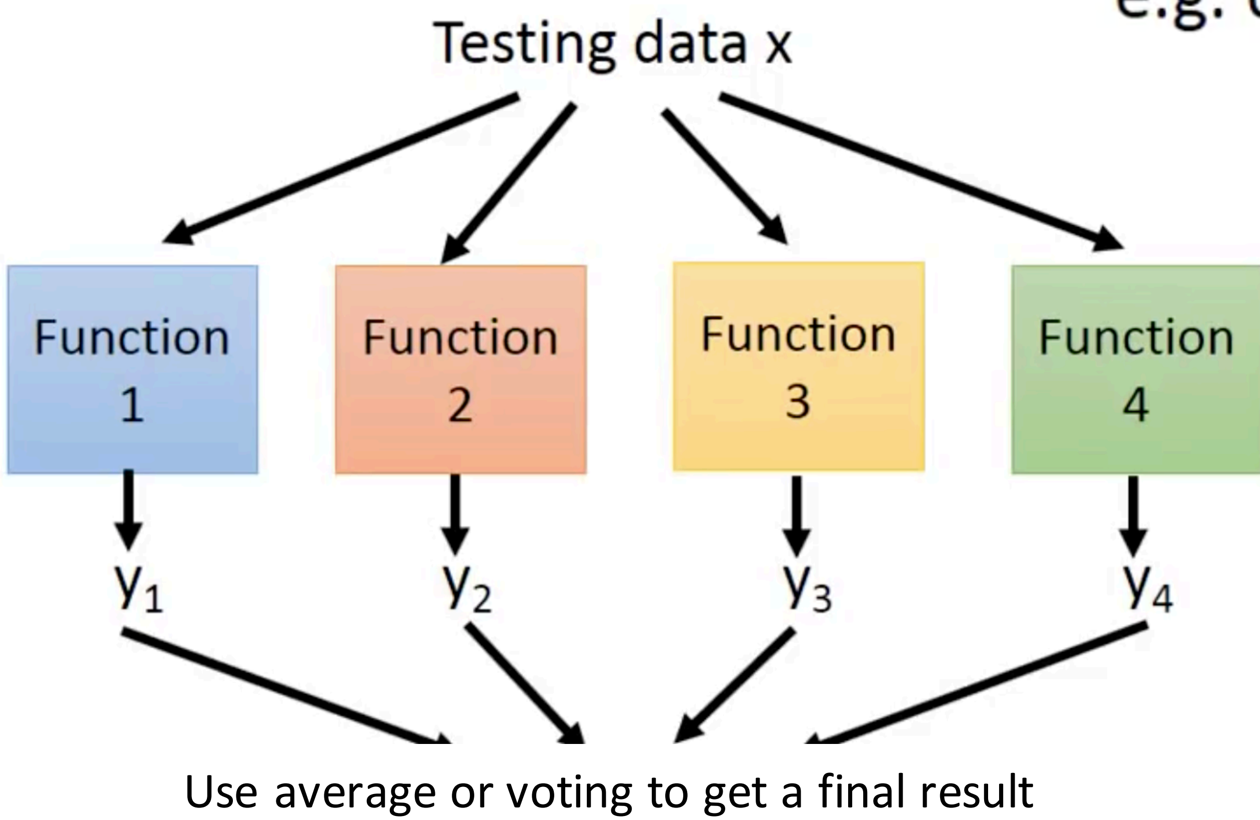
Use average or voting to get a final result

Bagging

This approach would be helpful when your model is complex, easy to overfit.

e.g. decision tree

A deep decision tree can easily get 100% accuracy on training data (but overfit)

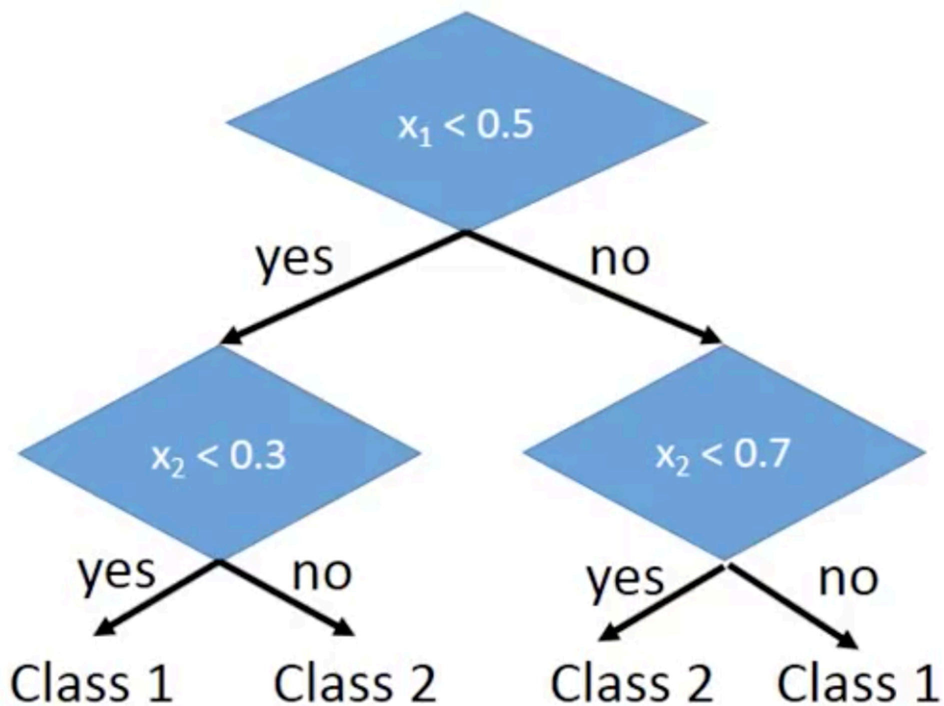


Decision Tree

The famous “Random Forest” method: Decision trees with bagging.

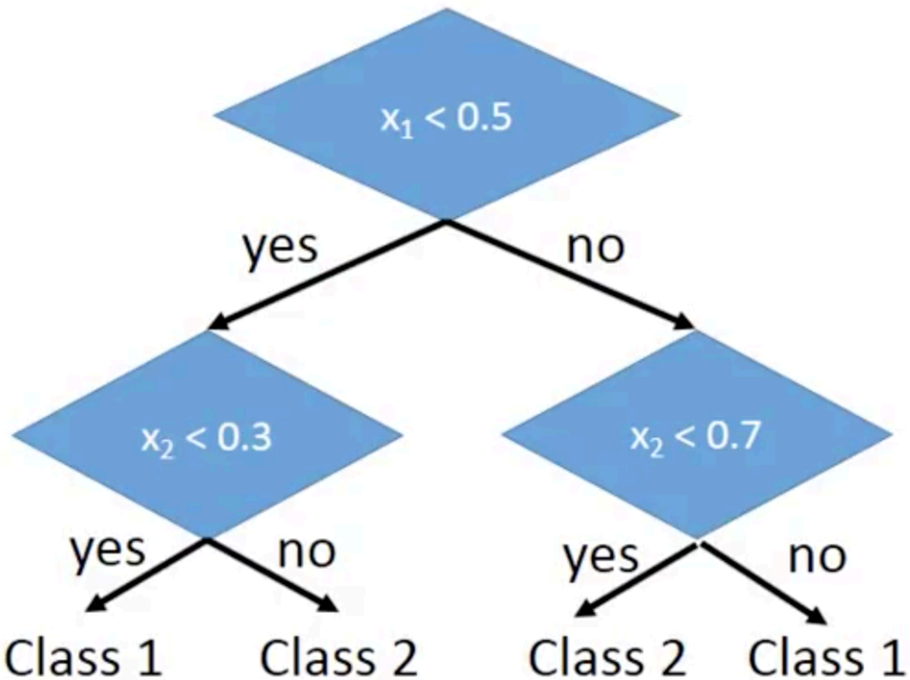
Decision Tree

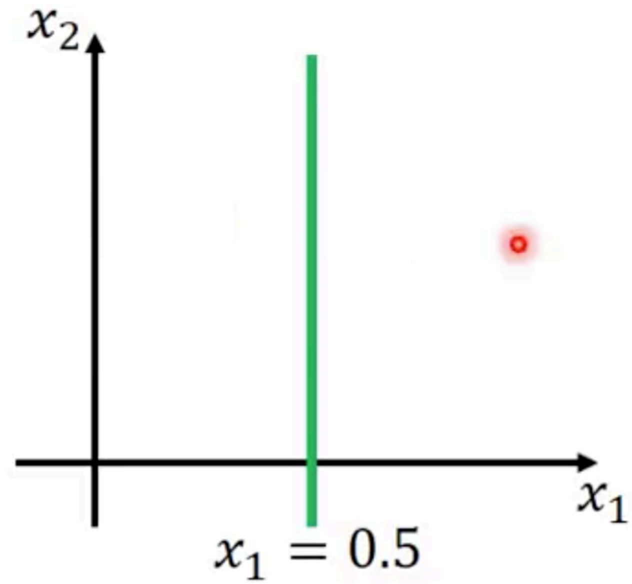
Assume each object x is represented by a 2-dim vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$



Decision Tree

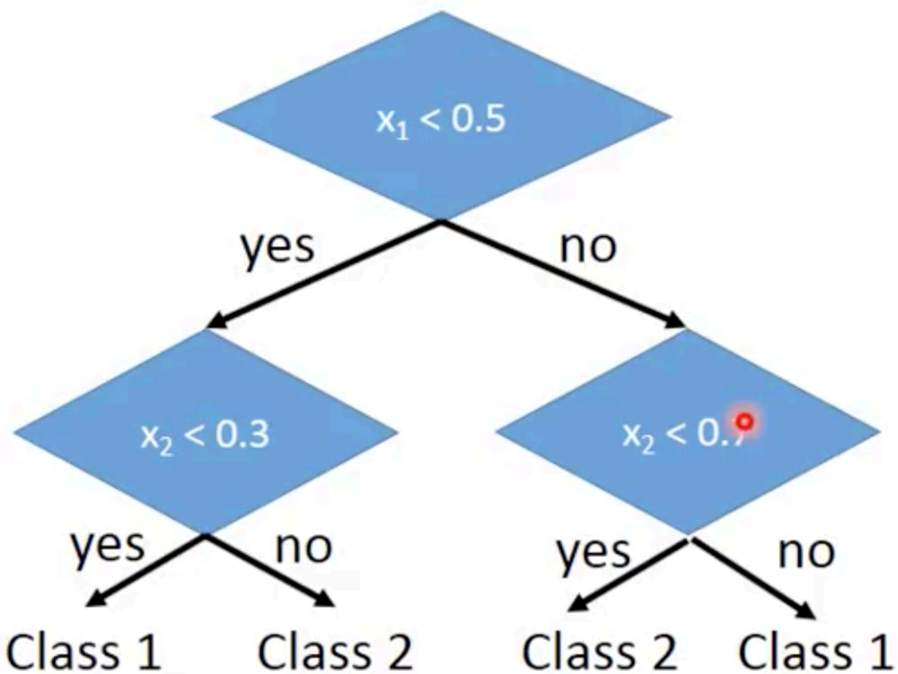
Assume each object x is represented by a 2-dim vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

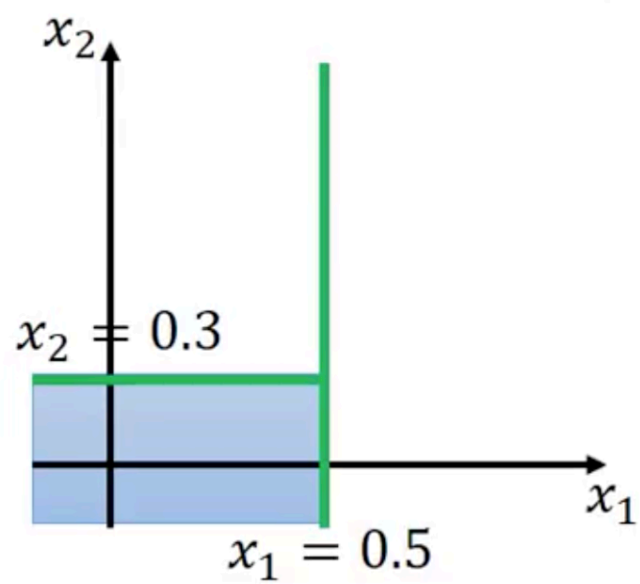




Decision Tree

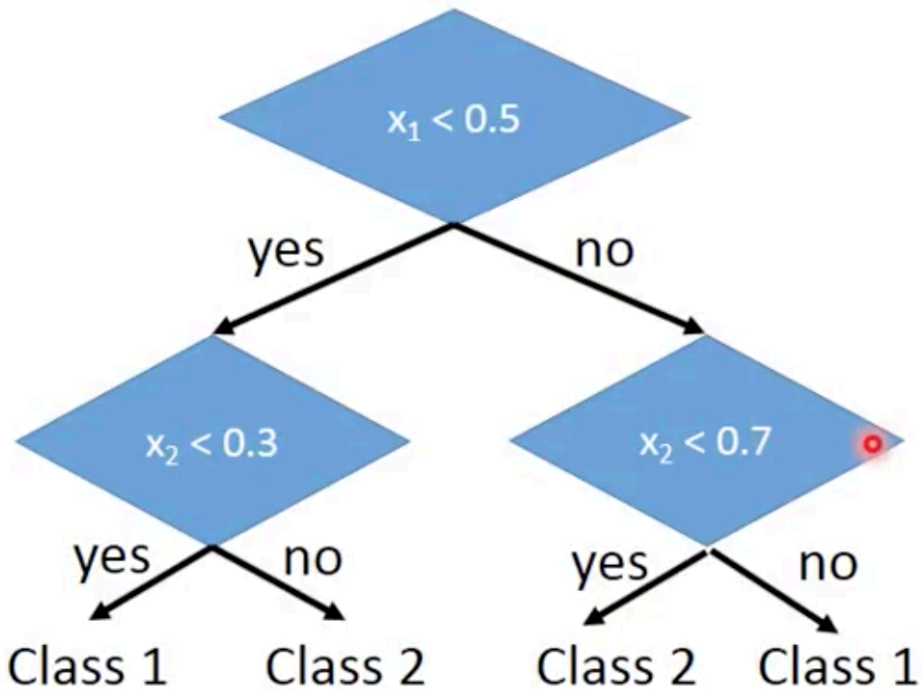
Assume each object x is represented by a 2-dim vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

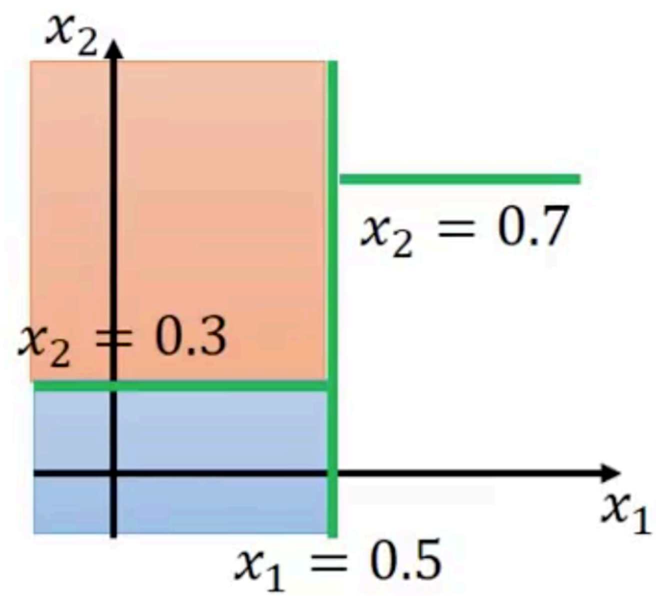




Decision Tree

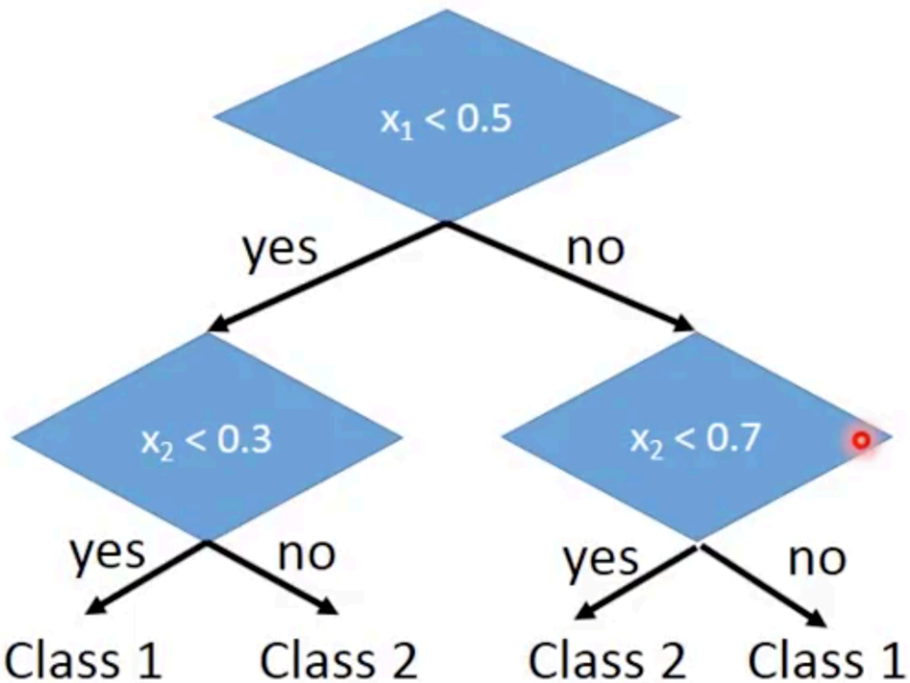
Assume each object x is represented by a 2-dim vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

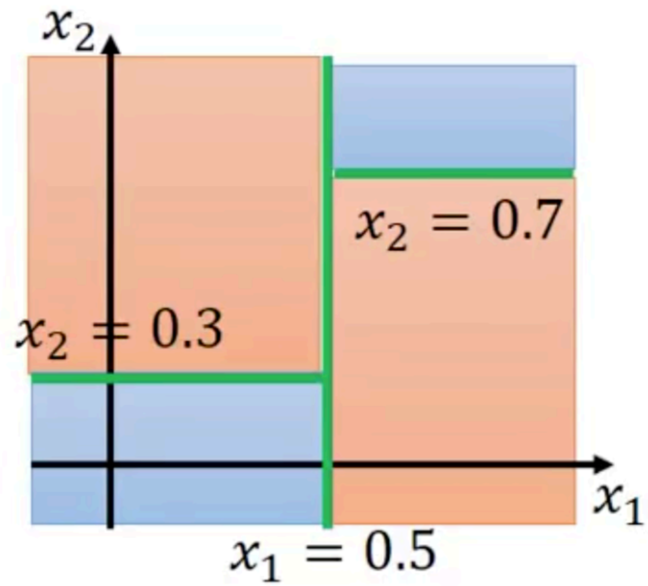




Decision Tree

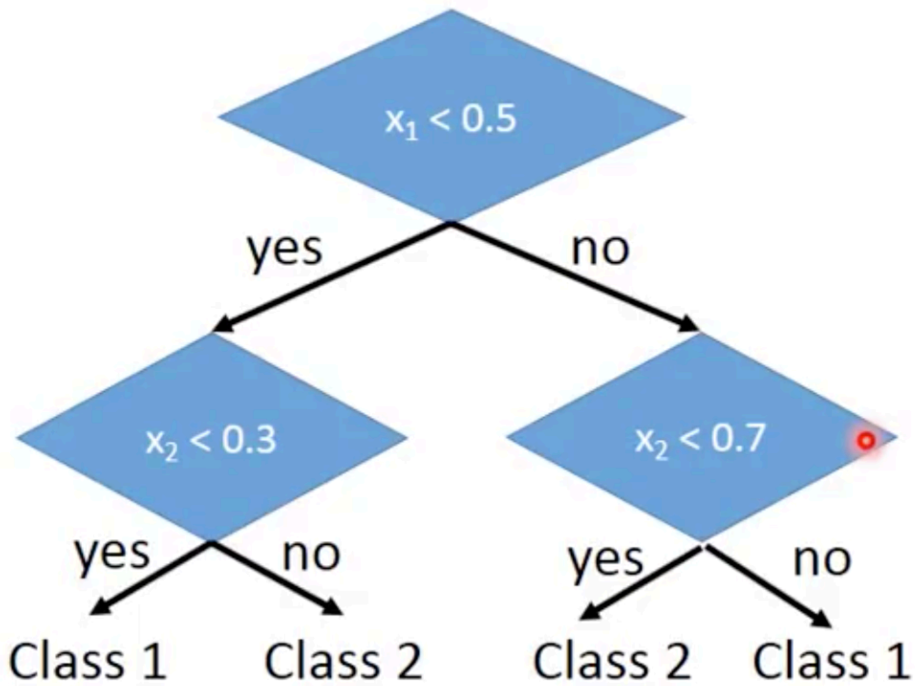
Assume each object x is represented by a 2-dim vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

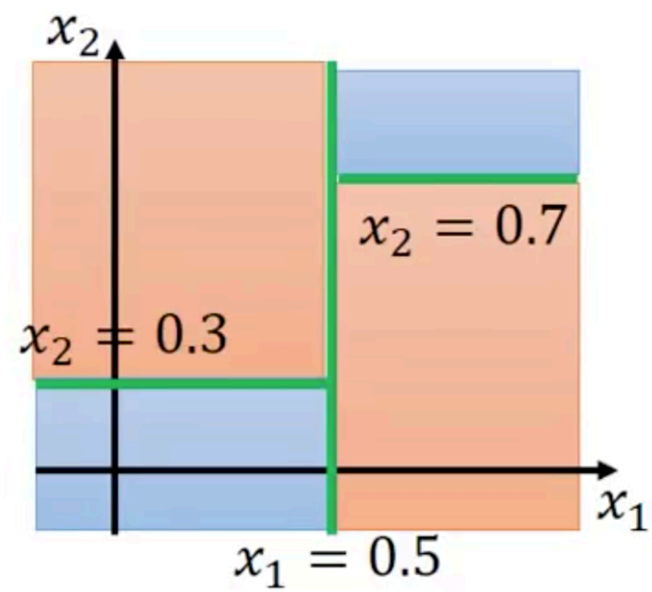




Decision Tree

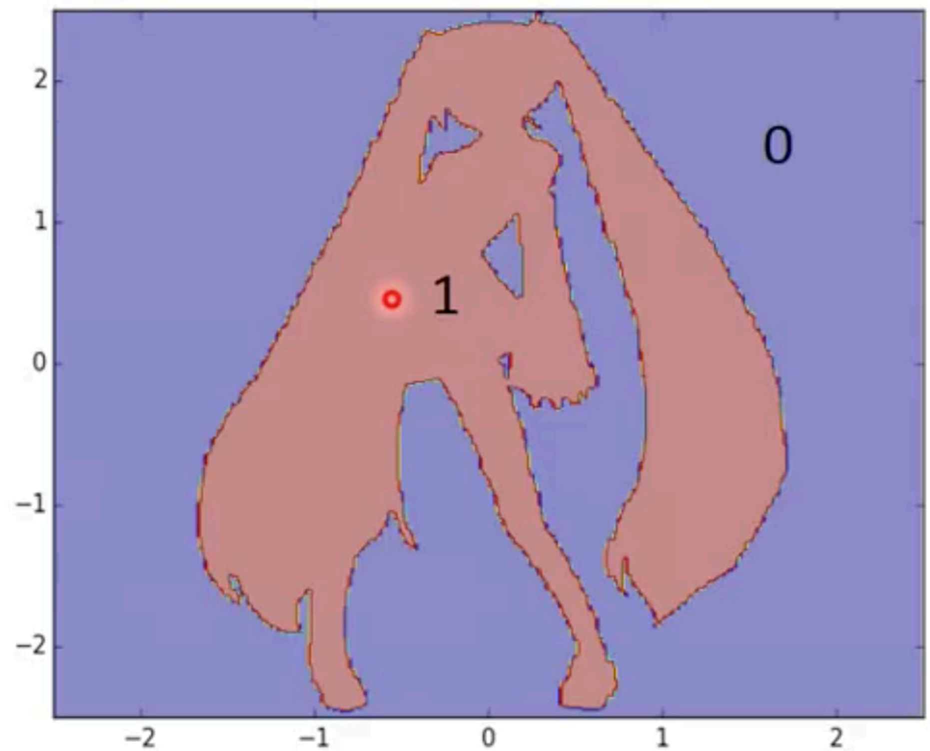
Assume each object x is represented by a 2-dim vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$





Can have more complex questions

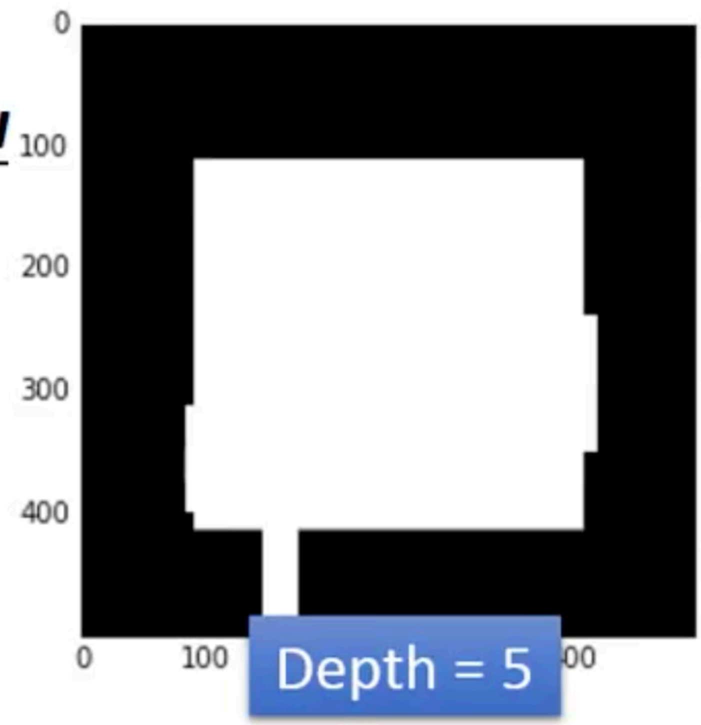
Experiment: Function of Miku



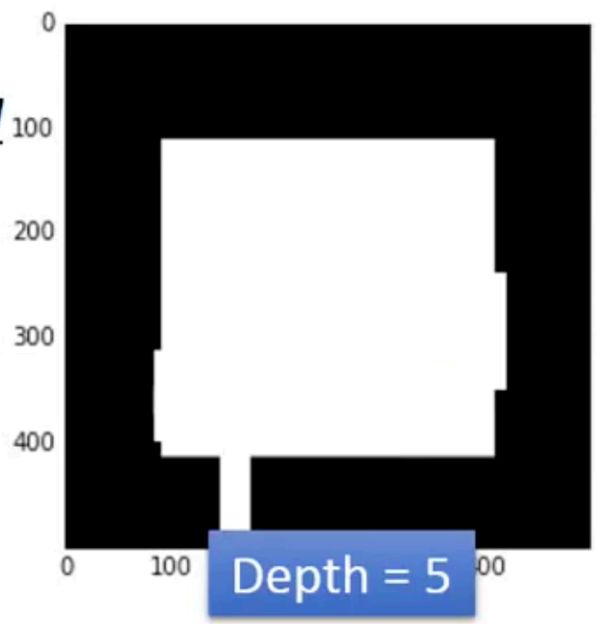
http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/theano/miku

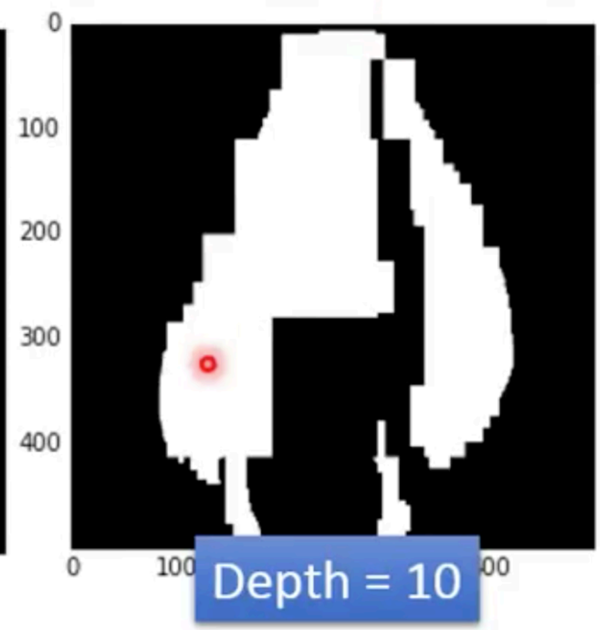
Experiment:
Function of Miku

Single
Decision
Tree

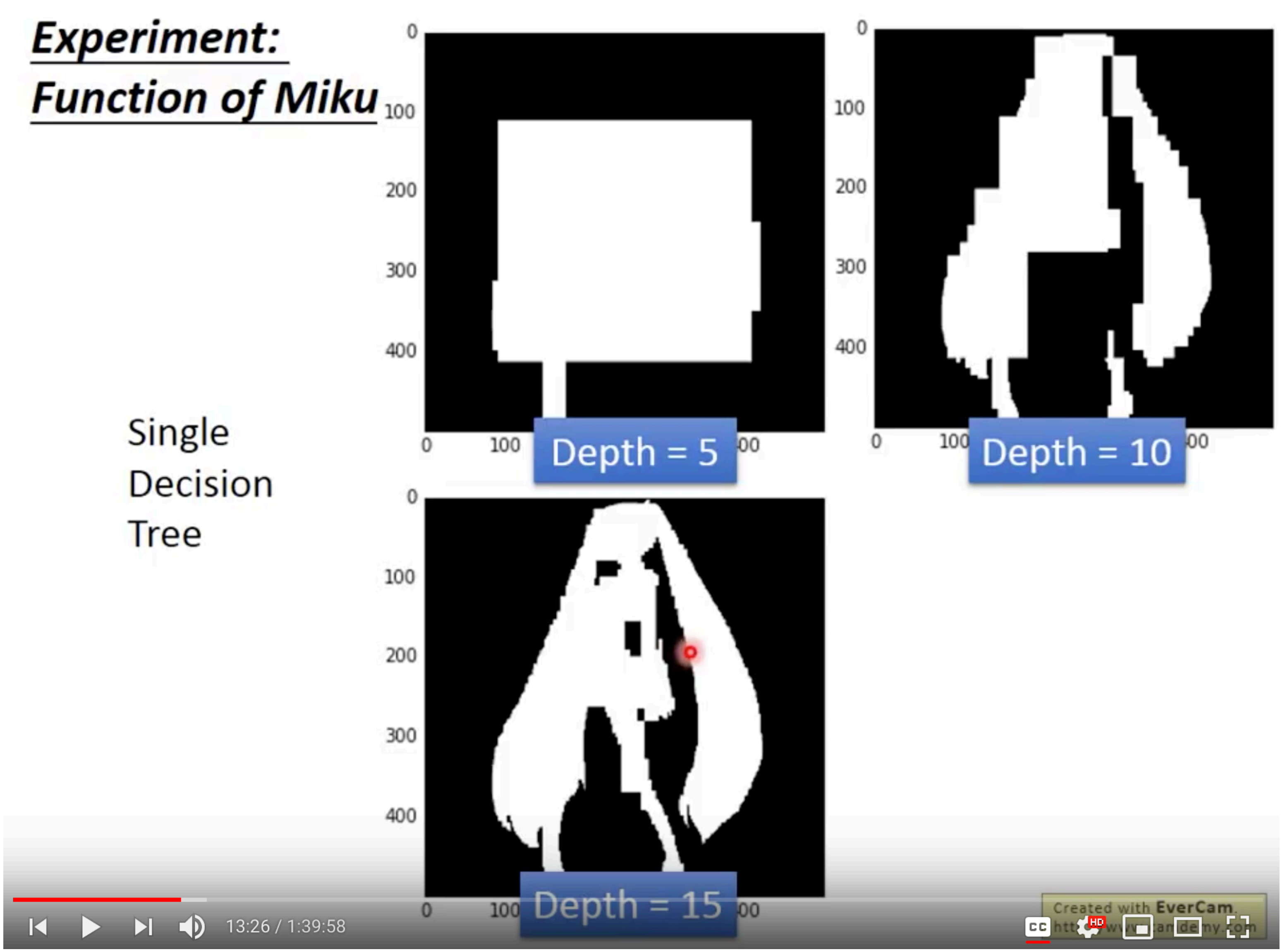


Experiment: Function of Miku



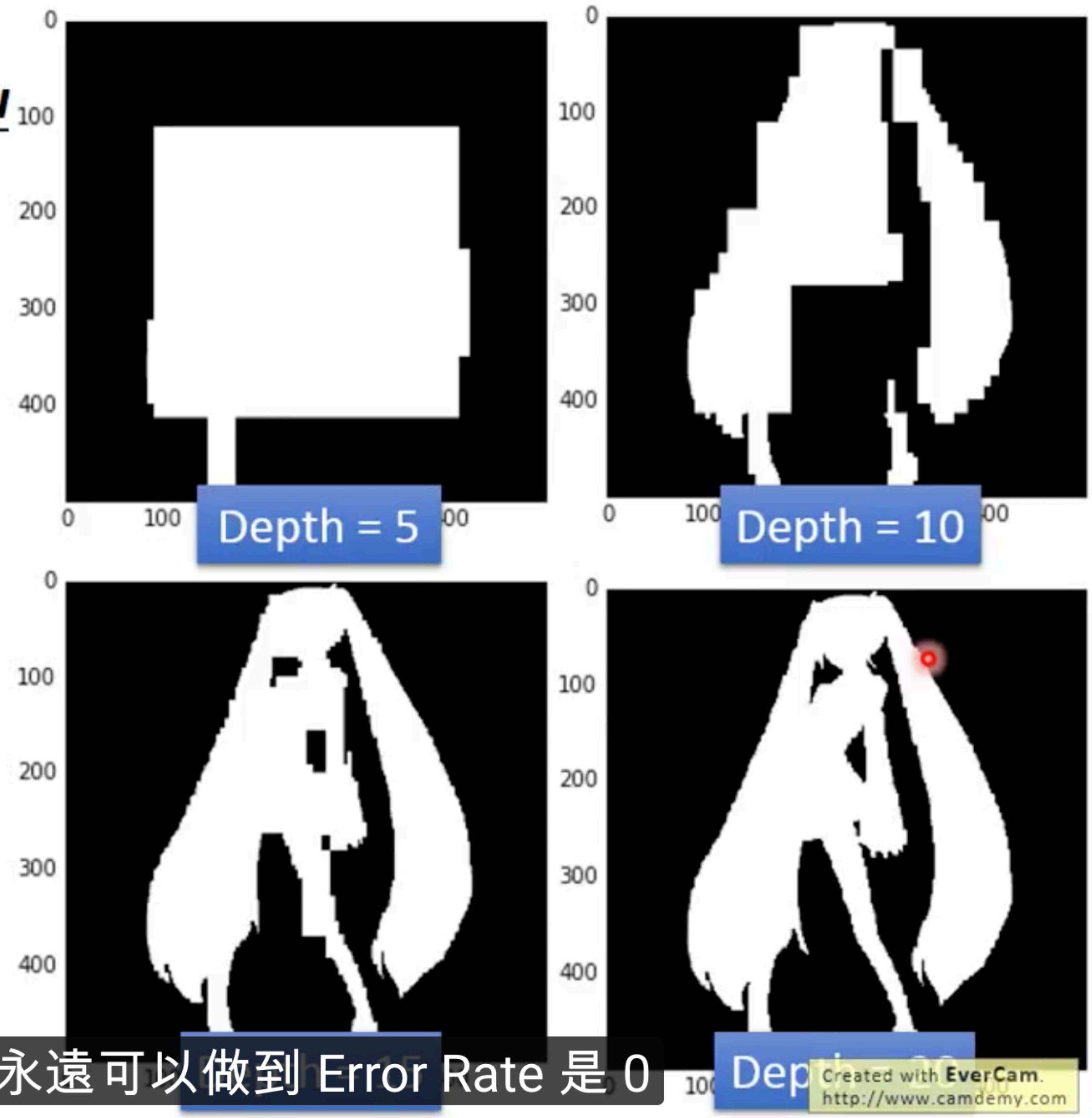


Single
Decision
Tree


Experiment:
Function of Miku
Single
Decision
Tree
Depth = 5
Depth = 10
Depth = 15
0
100
200
300
400
13:26 / 1:39:58
Created with EverCam

Experiment: Function of Miku

Single Decision Tree



永遠可以做到 Error Rate 是 0

Random Forest

- Decision tree:
  - Easy to achieve 0% error rate on training data

Random Forest

- Decision tree:
    - Easy to achieve 0% error rate on training data
        - If each training example has its own leaf ……

Complete overfitting, and nothing is learned.

- Random forest: Bagging of decision tree
    - Resampling training data is not sufficient

Random Forest

- Decision tree:
  - Easy to achieve 0% error rate on training data
    - If each training example has its own leaf ……
- Random forest: Bagging of decision tree
  - Resampling training data is not sufficient
  - Randomly restrict the features/questions used in each split

Random Forest

- Decision tree:
  - Easy to achieve 0% error rate on training data
    - If each training example has its own leaf ……
- Random forest: Bagging of decision tree
  - Resampling training data is not sufficient
  - Randomly restrict the features/questions used in each split
- Out-of-bag validation for bagging

Random Forest

train	f_1	f_2	f_3	f_4
x^1	O	X	O	X
x^2	O	X	X	O
x^3	X	O	O	X
x^4	X	O	X	O

- Decision tree:
  - Easy to achieve 0% error rate on training data
    - If each training example has its own leaf ……
- Random forest: Bagging of decision tree
  - Resampling training data is not sufficient
  - Randomly restrict the features/questions used in each split
- Out-of-bag validation for bagging

Random Forest

train	f_1	f_2	f_3	f_4
x^1	O	X	O	X
x^2	O	X	X	O
x^3	X	O	O	X
x^4	X	O	X	O

- Decision tree:
  - Easy to achieve 0% error rate on training data
    - If each training example has its own leaf ……
- Random forest: Bagging of decision tree
  - Resampling training data is not sufficient
  - Randomly restrict the features/questions used in each split
- Out-of-bag validation for bagging
  - Using RF = f_2+f_4 to test x^1
  - Using RF = f_2+f_3 to test x^2

Random Forest

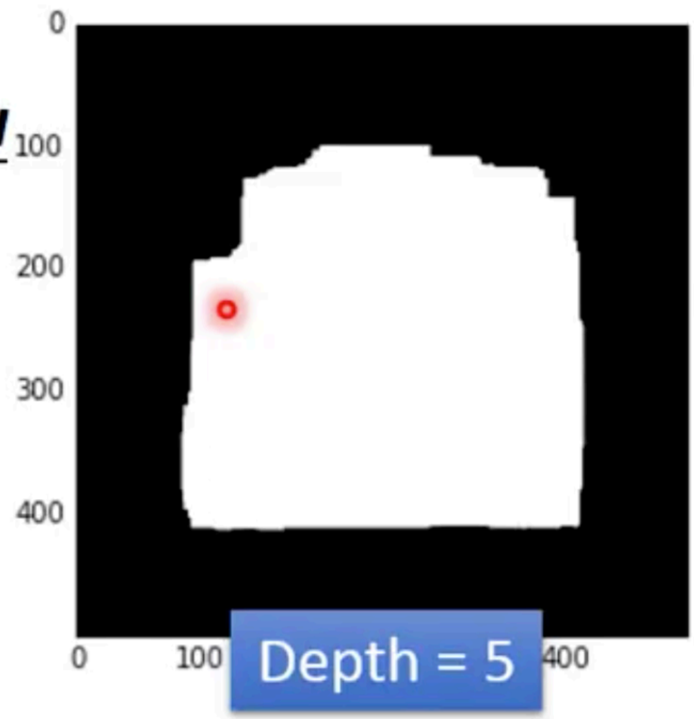
train	f_1	f_2	f_3	f_4
x^1	O	X	O	X
x^2	O	X	X	O
x^3	X	O	O	X
x^4	X	O	X	O

- Decision tree:
  - Easy to achieve 0% error rate on training data
    - If each training example has its own leaf ……
- Random forest: Bagging of decision tree
  - Resampling training data is not sufficient
  - Randomly restrict the features/questions used in each split
- Out-of-bag validation for bagging
  - Using RF = f_2+f_4 to test x^1
  - Using RF = f_2+f_3 to test x^2
  - Using RF = f_1+f_4 to test x^3
  - Using RF = f_1+f_3 to test x^4

Out-of-bag (OOB) error

Experiment:

Function of Miku



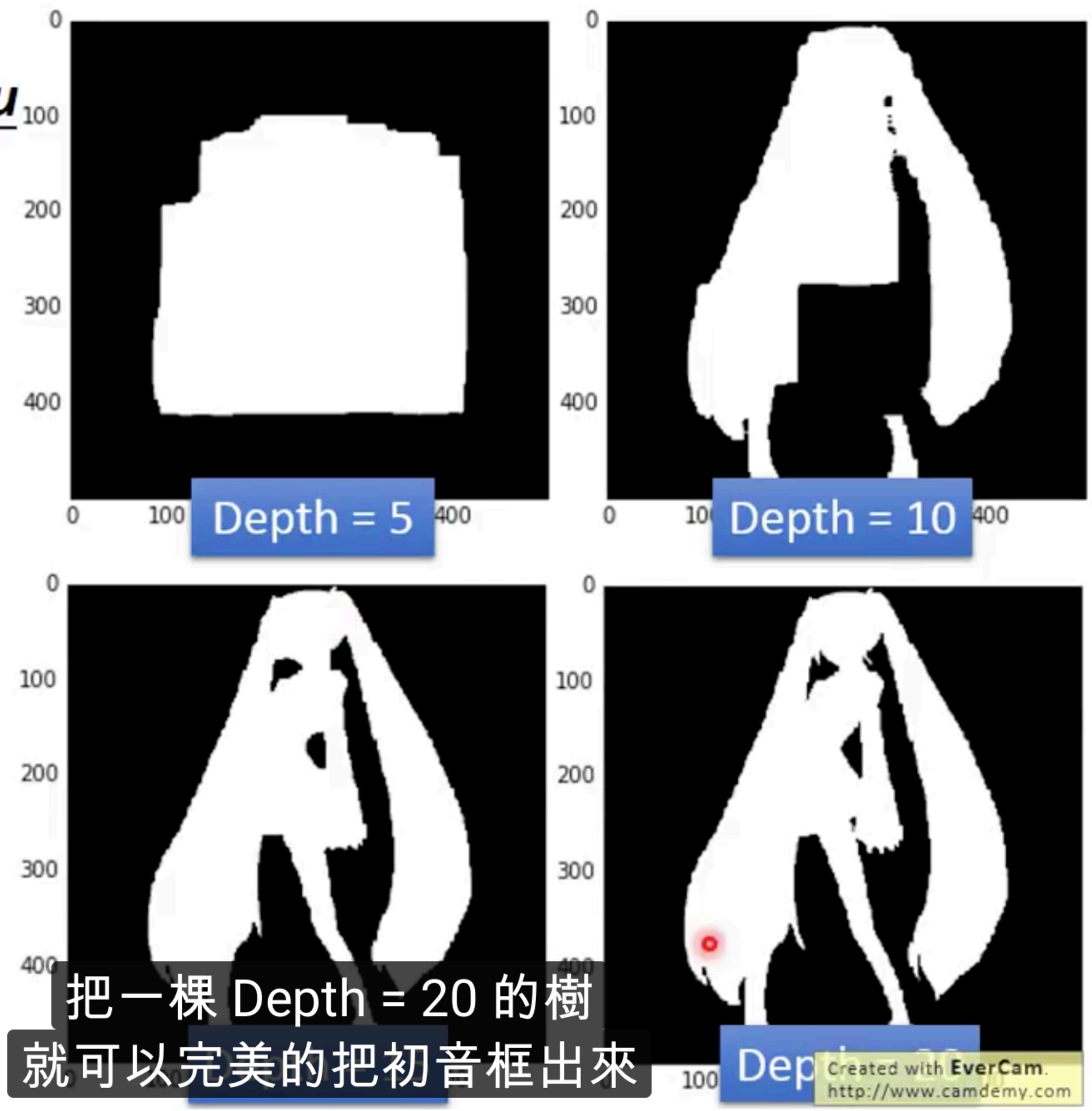
Random
Forest

(100 trees)

Experiment: Function of Miku

Random Forest

(100 trees)



把一棵 Depth = 20 的樹
就可以完美的把初音框出來

Ensemble: Boosting

Improving Weak Classifiers

Boosting

- Guarantee:
    - If your ML algorithm can produce classifier with error rate smaller than 50% on training data

Boosting

- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.

Boosting

- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.
- Framework of boosting
  - Obtain the first classifier $f_1(x)$
  - Find another function $f_2(x)$ to help $f_1(x)$

Boosting

- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.
- Framework of boosting
  - Obtain the first classifier $f_1(x)$
  - Find another function $f_2(x)$ to help $f_1(x)$
    - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot.

Boosting

- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.
- Framework of boosting
  - Obtain the first classifier $f_1(x)$
  - Find another function $f_2(x)$ to help $f_1(x)$
    - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot.
    - We want $f_2(x)$ to be complementary with $f_1(x)$ (How?)
  - Obtain the second classifier $f_2(x)$
  - …… Finally, combining all the classifiers

Boosting

- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.
- Framework of boosting
  - Obtain the first classifier $f_1(x)$
  - Find another function $f_2(x)$ to help $f_1(x)$
    - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot.
    - We want $f_2(x)$ to be complementary with $f_1(x)$ (How?)
  - Obtain the second classifier $f_2(x)$
  - …… Finally, combining all the classifiers
- The classifiers are learned sequentially.

Boosting

Training data:

$\{(x^1, \hat{y}^1), \cdots, (x^n, \hat{y}^n), \cdots, (x^N, \hat{y}^N)\}$

$\hat{y} = \pm 1$ (binary classification)

- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.
- Framework of boosting
  - Obtain the first classifier $f_1(x)$
  - Find another function $f_2(x)$ to help $f_1(x)$
    - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot.
    - We want $f_2(x)$ to be complementary with $f_1(x)$ (How?)
  - Obtain the second classifier $f_2(x)$
  - …… Finally, combining all the classifiers
- The classifiers are learned sequentially.

How to obtain different classifiers?

- Training on different training data sets
- How to have different training data sets
  - Re-sampling your training data to form a new set
  - Re-weighting your training data to form a new set

$(x^1, \hat{y}^1, u^1)$

$(x^2, \hat{y}^2, u^2)$

$(x^3, \hat{y}^3, u^3)$

How to obtain different classifiers?

- Training on different training data sets
- How to have different training data sets
  - Re-sampling your training data to form a new set
  - Re-weighting your training data to form a new set

$(x^1, \hat{y}^1, u^1) \quad u^1 = 1$

$(x^2, \hat{y}^2, u^2) \quad u^2 = 1$

$(x^3, \hat{y}^3, u^3) \quad u^3 = 1$

How to obtain different classifiers?

- Training on different training data sets
- How to have different training data sets
  - Re-sampling your training data to form a new set
  - Re-weighting your training data to form a new set
  - In real implementation, you only have to change the cost/objective function

$(x^1, \hat{y}^1, u^1)$ $\quad u^1 = \cancel{1}$ $\quad$ 0.4

$(x^2, \hat{y}^2, u^2)$ $\quad u^2 = \cancel{1}$ $\quad$ 2.1

$(x^3, \hat{y}^3, u^3)$ $\quad u^3 = \cancel{1}$ $\quad$ 0.7

How to obtain different classifiers?

- Training on different training data sets
- How to have different training data sets
  - Re-sampling your training data to form a new set
  - Re-weighting your training data to form a new set
  - In real implementation, you only have to change the cost/objective function

$(x^1, \hat{y}^1, u^1) \quad u^1 = \cancel{1} \quad 0.4$

$(x^2, \hat{y}^2, u^2) \quad u^2 = \cancel{1} \quad 2.1$

$(x^3, \hat{y}^3, u^3) \quad u^3 = \cancel{1} \quad 0.7$

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

How to obtain different classifiers?

- Training on different training data sets
- How to have different training data sets
  - Re-sampling your training data to form a new set
  - Re-weighting your training data to form a new set
  - In real implementation, you only have to change the cost/objective function

$(x^1, \hat{y}^1, u^1)$ $u^1 = \cancel{1}$ 0.4

$(x^2, \hat{y}^2, u^2)$ $u^2 = \cancel{1}$ 2.1

$(x^3, \hat{y}^3, u^3)$ $u^3 = \cancel{1}$ 0.7

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

$$\Downarrow$$

$$L(f) = \sum_n u^n l(f(x^n), \hat{y}^n)$$

Idea of Adaboost

- Idea: **training $f_2(x)$ on the new training set that fails $f_1(x)$**

Idea of Adaboost

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

Idea of Adaboost

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

ε_1: the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1}$$

Idea of Adaboost

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

ε_1: the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

Idea of Adaboost

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

ε_1: the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n \qquad \varepsilon_1 < 0.5$$

Idea of Adaboost

- Idea: **training $f_2(x)$ on the new training set that fails $f_1(x)$**
- How to find a new training set that fails $f_1(x)$?

ε_1: the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \quad Z_1 = \sum_n u_1^n \quad \varepsilon_1 < 0.5$$

Changing the example weights from u_1^n to u_2^n such that

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

Idea of Adaboost

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

ε_1: the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n \qquad \varepsilon_1 < 0.5$$

Changing the example weights from u_1^n to u_2^n such that

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

The performance of f_1 for new weights would be random.

Idea of Adaboost

- Idea: **training $f_2(x)$ on the new training set that fails $f_1(x)$**
- How to find a new training set that fails $f_1(x)$?

ε_1: the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \quad Z_1 = \sum_n u_1^n \quad \varepsilon_1 < 0.5$$

Changing the example weights from u_1^n to u_2^n such that

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

The performance of f_1 for new weights would be random.

Training $f_2(x)$ based on the new weights u_2^n

Re-weighting Training Data

- Idea: **training $f_2(x)$ on the new training set that fails $f_1(x)$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$

$(x^2, \hat{y}^2, u^2)$

$(x^3, \hat{y}^3, u^3)$

$(x^4, \hat{y}^4, u^4)$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$ $\quad u^1 = 1$

$(x^2, \hat{y}^2, u^2)$ $\quad u^2 = 1$

$(x^3, \hat{y}^3, u^3)$ $\quad u^3 = 1$

$(x^4, \hat{y}^4, u^4)$ $\quad u^4 = 1$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$ $\quad u^1 = 1$

$(x^2, \hat{y}^2, u^2)$ $\quad u^2 = 1$

$(x^3, \hat{y}^3, u^3)$ $\quad u^3 = 1$

$(x^4, \hat{y}^4, u^4)$ $\quad u^4 = 1$

$f_1(x)$

Re-weighting Training Data

- Idea: **training $f_2(x)$ on the new training set that fails $f_1(x)$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$ $u^1 = 1$ ✓

$(x^2, \hat{y}^2, u^2)$ $u^2 = 1$ ✗

$(x^3, \hat{y}^3, u^3)$ $u^3 = 1$ ✓

$(x^4, \hat{y}^4, u^4)$ $u^4 = 1$ ✓

$\varepsilon_1 = 0.25$

$f_1(x)$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$ $u^1 = 1$ ✔

$(x^2, \hat{y}^2, u^2)$ $u^2 = 1$ ✘

$(x^3, \hat{y}^3, u^3)$ $u^3 = 1$ ✔

$(x^4, \hat{y}^4, u^4)$ $u^4 = 1$ ✔

$\varepsilon_1 = 0.25$

$f_1(x)$

$u^1 = 1/\sqrt{3}$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$ $u^1 = 1$ $u^1 = 1/\sqrt{3}$

$(x^2, \hat{y}^2, u^2)$ $u^2 = 1$ $u^2 = \sqrt{3}$

$(x^3, \hat{y}^3, u^3)$ $u^3 = 1$ $u^3 = 1/\sqrt{3}$

$(x^4, \hat{y}^4, u^4)$ $u^4 = 1$ $u^4 = 1/\sqrt{3}$

$\varepsilon_1 = 0.25$

$f_1(x)$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$	$u^1 = 1$		$u^1 = 1/\sqrt{3}$
$(x^2, \hat{y}^2, u^2)$	$u^2 = 1$		$u^2 = \sqrt{3}$
$(x^3, \hat{y}^3, u^3)$	$u^3 = 1$		$u^3 = 1/\sqrt{3}$
$(x^4, \hat{y}^4, u^4)$	$u^4 = 1$		$u^4 = 1/\sqrt{3}$

$\varepsilon_1 = 0.25$

$f_1(x)$

0.5

Re-weighting Training Data

- Idea: **training $f_2(x)$ on the new training set that fails $f_1(x)$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$	$u^1 = 1$ ✓	$u^1 = 1/\sqrt{3}$ ✓
$(x^2, \hat{y}^2, u^2)$	$u^2 = 1$ ✗	$u^2 = \sqrt{3}$ ✗
$(x^3, \hat{y}^3, u^3)$	$u^3 = 1$ ✓	$u^3 = 1/\sqrt{3}$ ✓
$(x^4, \hat{y}^4, u^4)$	$u^4 = 1$ ✓	$u^4 = 1/\sqrt{3}$ ✓

$\varepsilon_1 = 0.25$

$f_1(x)$

0.5

$f_2(x)$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$	$u^1 = 1$	$u^1 = 1/\sqrt{3}$
$(x^2, \hat{y}^2, u^2)$	$u^2 = 1$	$u^2 = \sqrt{3}$
$(x^3, \hat{y}^3, u^3)$	$u^3 = 1$	$u^3 = 1/\sqrt{3}$
$(x^4, \hat{y}^4, u^4)$	$u^4 = 1$	$u^4 = 1/\sqrt{3}$

$\varepsilon_1 = 0.25$

$f_1(x)$

0.5

$\varepsilon_2 < 0.5$

$f_2(x)$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

If x^n misclassified by f_1 ($f_1(x^n) \neq \hat{y}^n$)

$$u_2^n \leftarrow u_1^n \text{ multiplying } d_1$$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$$
\begin{cases}
\text{If } x^n \text{ misclassified by } f_1 \ (f_1(x^n) \neq \hat{y}^n) \\
\qquad u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \\
\text{If } x^n \text{ correctly classified by } f_1 \ (f_1(x^n) = \hat{y}^n) \\
\qquad u_2^n \leftarrow u_1^n \text{ divided by } d_1
\end{cases}
$$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$$\begin{cases} \text{If } x^n \text{ misclassified by } f_1 \; (f_1(x^n) \neq \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \quad \boxed{\text{increase}} \\ \text{If } x^n \text{ correctly classified by } f_1 \; (f_1(x^n) = \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ divided by } d_1 \quad \boxed{\text{decrease}} \end{cases}$$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$$\begin{cases} \text{If } x^n \text{ misclassified by } f_1 \; (f_1(x^n) \neq \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \quad \boxed{\text{increase}} \\ \text{If } x^n \text{ correctly classified by } f_1 \; (f_1(x^n) = \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ divided by } d_1 \quad \boxed{\text{decrease}} \end{cases}$$

f_2 will be learned based on example weights u_2^n

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$$\begin{cases} \text{If } x^n \text{ misclassified by } f_1 \ (f_1(x^n) \neq \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \quad \boxed{\text{increase}} \\ \text{If } x^n \text{ correctly classified by } f_1 \ (f_1(x^n) = \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ divided by } d_1 \quad \boxed{\text{decrease}} \end{cases}$$

f_2 will be learned based on example weights u_2^n

What is the value of d_1?

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_1}} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_2}} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_1}} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_2}} = 0.5$$

$f_1(x^n) \neq \hat{y}^n \quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n \quad u_2^n \leftarrow u_1^n$ divided by d_1

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_1}} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_2}} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ divided by d_1

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1$$

$$= \sum_n u_2^n$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_1}} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_2}} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $u_2^n \leftarrow u_1^n$ divided by d_1

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1$$

$$= \sum_n u_2^n = \sum_{f_1(x^n) \neq \hat{y}^n} u_2^n + \sum_{f_1(x^n) = \hat{y}^n} u_2^n$$

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_1}} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_2}} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ divided by d_1

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 \qquad = \sum_{f_1(x^n) \neq \hat{y}^n} u_2^n + \sum_{f_1(x^n) = \hat{y}^n} u_2^n$$

$$= \sum_n u_2^n \qquad = \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1$$

$$\frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1} = 0.5$$

帶進去

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_1}} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_2}} = 0.5$$

$f_1(x^n) \neq \hat{y}^n \quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n \quad u_2^n \leftarrow u_1^n$ divided by d_1

$$= \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 \qquad = \sum_{f_1(x^n)\neq\hat{y}^n} u_2^n + \sum_{f_1(x^n)=\hat{y}^n} u_2^n$$

$$= \sum_n u_2^n \qquad = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1$$

$$\frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 2$$

然後把分子和分母倒過來

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $u_2^n \leftarrow u_1^n$ divided by d_1

$$\frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 2 \qquad \frac{\sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 1$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $u_2^n \leftarrow u_1^n$ divided by d_1

$$\frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 2 \qquad \frac{\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $u_2^n \leftarrow u_1^n$ divided by d_1

$$\frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 2 \qquad \frac{\sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 \quad \frac{1}{d_1} \sum_{f_1(x^n) = \hat{y}^n} u_1^n = d_1 \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ divided by d_1

$$\frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 2 \qquad \frac{\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 \qquad \frac{1}{d_1} \sum_{f_1(x^n)=\hat{y}^n} u_1^n = d_1 \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n$$

$$\varepsilon_1 = \frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}{Z_1}$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ divided by d_1

$$\frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 2 \qquad \frac{\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 \qquad \frac{1}{d_1} \underbrace{\sum_{f_1(x^n)=\hat{y}^n} u_1^n}_{Z_1(1-\varepsilon_1)} = d_1 \underbrace{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}_{Z_1\varepsilon_1}$$

$$\varepsilon_1 = \frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}{Z_1}$$

$$\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n = Z_1 \varepsilon_1$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ divided by d_1

$$\frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 2 \qquad \frac{\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 \qquad \frac{1}{d_1} \underbrace{\sum_{f_1(x^n)=\hat{y}^n} u_1^n}_{Z_1(1-\varepsilon_1)} = d_1 \underbrace{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}_{Z_1\varepsilon_1}$$

$$\varepsilon_1 = \frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}{Z_1}$$

$$\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n = Z_1 \varepsilon_1$$

$$Z_1(1-\varepsilon_1)/d_1 = Z_1 \varepsilon_1 d_1$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ divided by d_1

$$\frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 2 \qquad \frac{\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 \qquad \frac{1}{d_1} \underbrace{\sum_{f_1(x^n)=\hat{y}^n} u_1^n}_{Z_1(1-\varepsilon_1)} = d_1 \underbrace{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}_{Z_1\varepsilon_1}$$

$$\varepsilon_1 = \frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}{Z_1}$$

$$\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n = Z_1 \varepsilon_1$$

$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ divided by d_1

$$\frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 2 \qquad \frac{\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 \qquad \frac{1}{d_1} \underbrace{\sum_{f_1(x^n)=\hat{y}^n} u_1^n}_{Z_1(1-\varepsilon_1)} = d_1 \underbrace{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}_{Z_1 \varepsilon_1}$$

$$\varepsilon_1 = \frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}{Z_1}$$

$$\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n = Z_1 \varepsilon_1$$

$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t} \qquad > 1$$

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - For n = 1, …, N:
    - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
    - $u_{t+1}^n = u_t^n \times d_t$

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - For n = 1, …, N:
    - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
    - $u_{t+1}^n = u_t^n \times d_t$
    - Else:
    - $u_{t+1}^n = u_t^n / d_t$

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, ..., T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - For n = 1, ..., N:
    - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
    - $u_{t+1}^n = u_t^n \times d_t$
    - Else:
    - $u_{t+1}^n = u_t^n / d_t$

$$d_t = \sqrt{(1 - \varepsilon_t)/\varepsilon_t}$$

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - For n = 1, …, N:
    - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
    - $u_{t+1}^n = u_t^n \times d_t$
    - Else:
    - $u_{t+1}^n = u_t^n / d_t$

$$d_t = \sqrt{(1 - \varepsilon_t)/\varepsilon_t}$$

$$\alpha_t = ln\sqrt{(1 - \varepsilon_t)/\varepsilon_t}$$

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - For n = 1, …, N:
    - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
    - $u_{t+1}^n = u_t^n \times d_t = u_t^n \times \exp(\alpha_t)$
    - Else:
    - $u_{t+1}^n = u_t^n / d_t = u_t^n \times \exp(-\alpha_t)$

$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - For n = 1, …, N:
    - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
    - $u_{t+1}^n = u_t^n \times d_t = u_t^n \times \exp(\alpha_t)$
    - Else:
    - $u_{t+1}^n = u_t^n / d_t = u_t^n \times \exp(-\alpha_t)$

$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n \leftarrow u_t^n \times exp(\qquad \alpha_t)$$

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - For n = 1, …, N:
    - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
    - $u_{t+1}^n = u_t^n \times d_t = u_t^n \times \exp(\alpha_t)$
    - Else:
    - $u_{t+1}^n = u_t^n / d_t = u_t^n \times \exp(-\alpha_t)$

$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n \leftarrow u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

Algorithm for AdaBoost

- We obtain a set of functions: $f_1(x), \ldots, f_t(x), \ldots, f_T(x)$

Algorithm for AdaBoost

- We obtain a set of functions: $f_1(x), \dots, f_t(x), \dots, f_T(x)$
- How to aggregate them?

Algorithm for AdaBoost

- We obtain a set of functions: $f_1(x), \ldots, f_t(x), \ldots, f_T(x)$
- How to aggregate them?
  - Uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} f_t(x))$

Algorithm for AdaBoost

- We obtain a set of functions: $f_1(x), \ldots, f_t(x), \ldots, f_T(x)$
- How to aggregate them?
  - Uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} f_t(x))$
  - Non-uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

Algorithm for AdaBoost

- We obtain a set of functions: $f_1(x), \dots, f_t(x), \dots, f_T(x)$
- How to aggregate them?
  - Uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} f_t(x))$
  - Non-uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

Algorithm for AdaBoost

- We obtain a set of functions: $f_1(x), \ldots, f_t(x), \ldots, f_T(x)$
- How to aggregate them?
  - Uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} f_t(x))$
  - Non-uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$ $\qquad$ $\varepsilon_t = 0.1$

$u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$ $\qquad$ $\alpha_t = 1.10$

Algorithm for AdaBoost

- We obtain a set of functions: $f_1(x), \ldots, f_t(x), \ldots, f_T(x)$
- How to aggregate them?
  - Uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} f_t(x))$
  - Non-uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$ $\qquad$ $\varepsilon_t = 0.1$ $\qquad$ $\varepsilon_t = 0.4$

$u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$ $\qquad$ $\alpha_t = 1.10$ $\qquad$ $\alpha_t = 0.20$

Algorithm for AdaBoost

- We obtain a set of functions: $f_1(x), \dots, f_t(x), \dots, f_T(x)$
- How to aggregate them?
  - Uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} f_t(x))$
  - Non-uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

Smaller error ε_t, larger weight for final voting

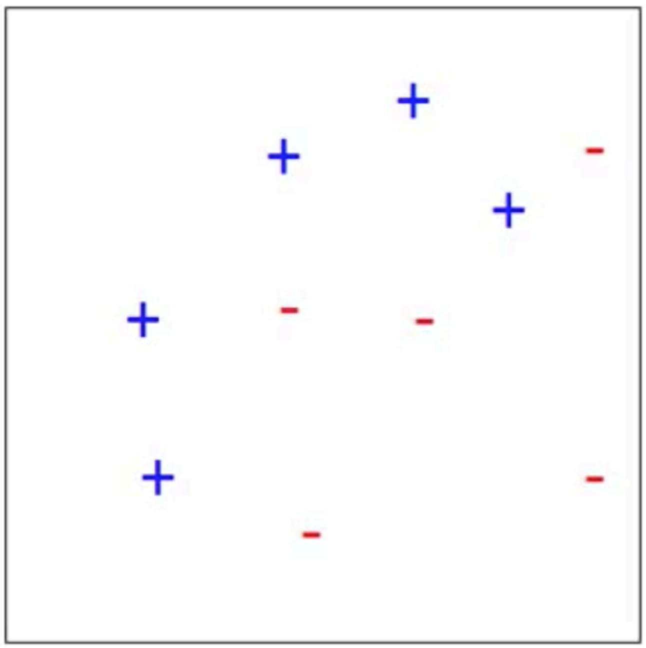
$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$\varepsilon_t = 0.1$	$\varepsilon_t = 0.4$
α_t =1.10	α_t =0.20

Toy Example

- t=1

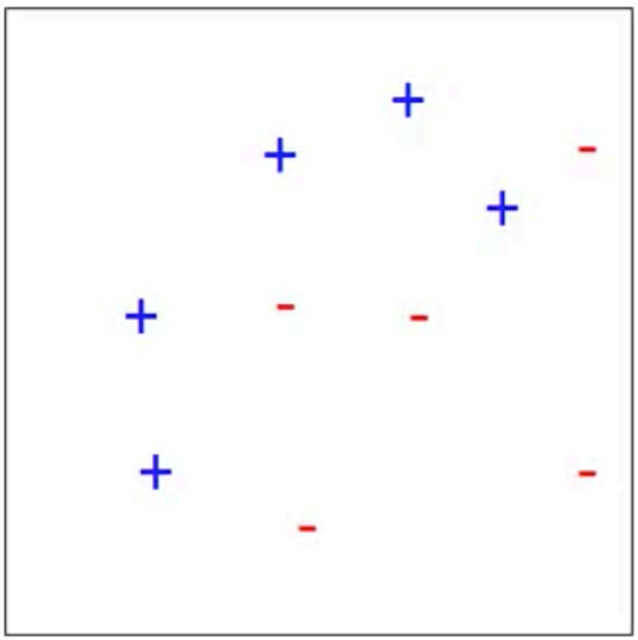


Toy Example

T=3, weak classifier = decision stump

Make a cut along a dimension

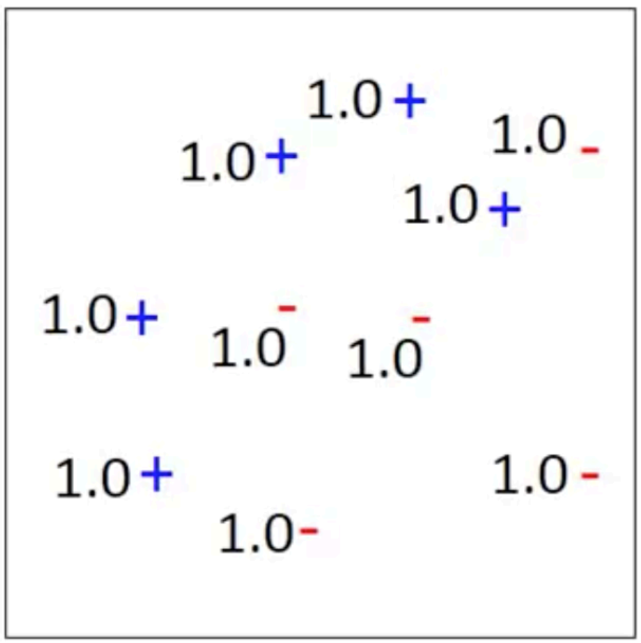
- t=1



Toy Example

T=3, weak classifier = decision stump

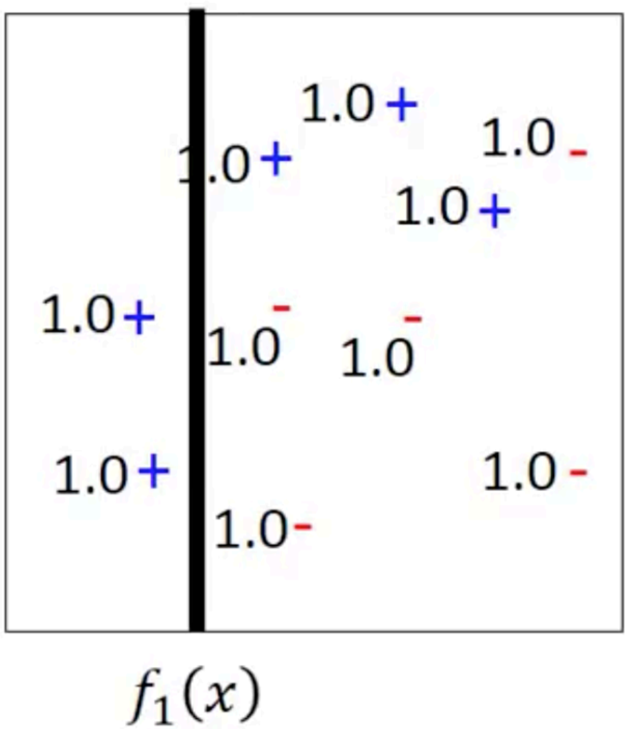
- t=1



Toy Example

T=3, weak classifier = decision stump

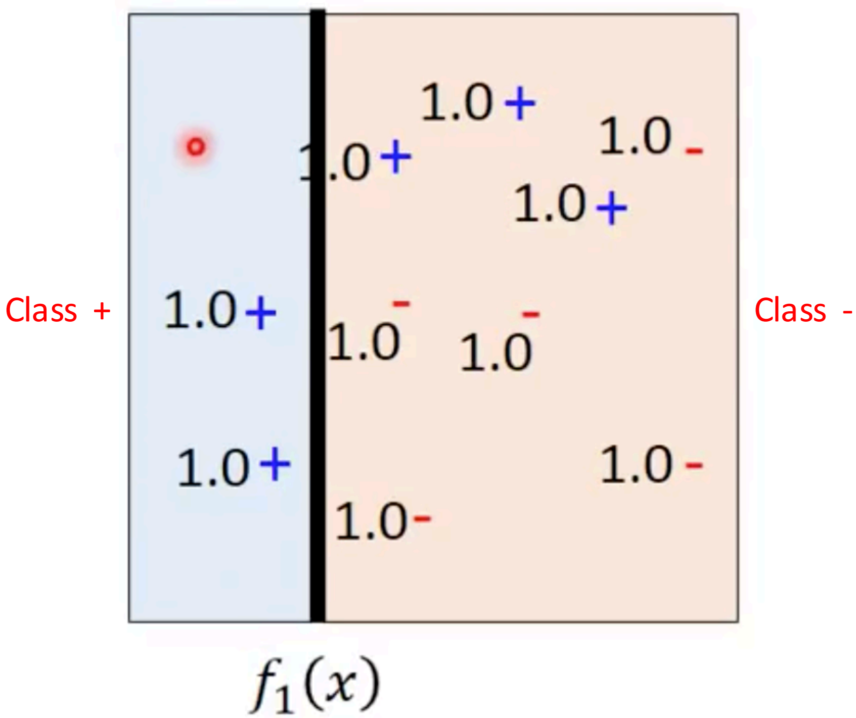
- t=1



Toy Example

T=3, weak classifier = decision stump

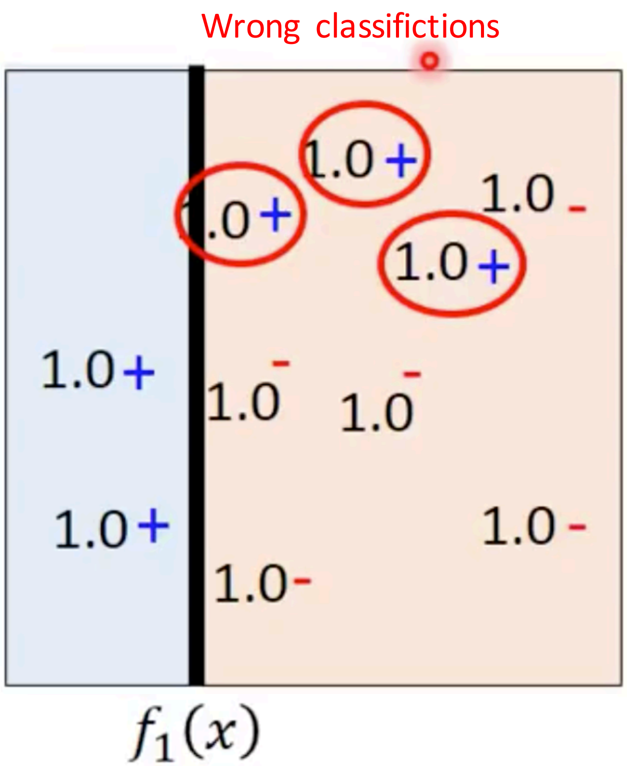
- t=1



Toy Example

T=3, weak classifier = decision stump

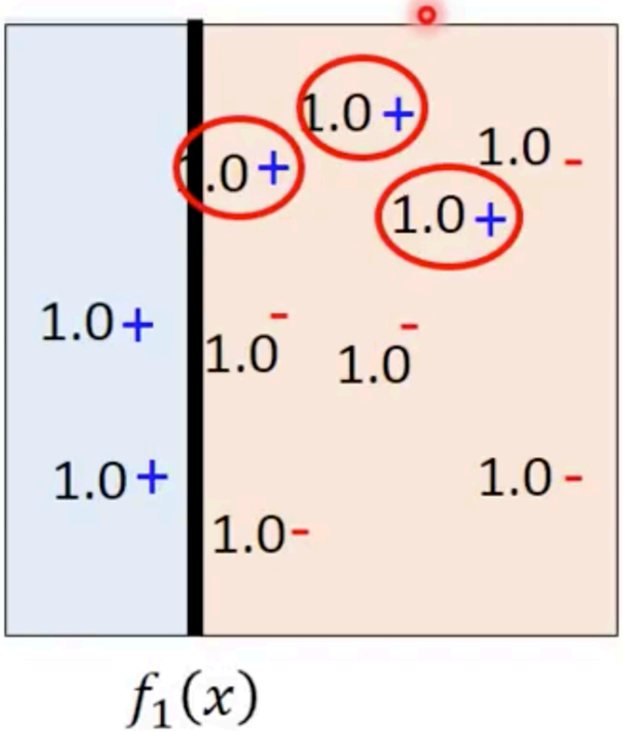
- t=1



Toy Example

T=3, weak classifier = decision stump

- t=1



$$\varepsilon_1 = 0.30$$

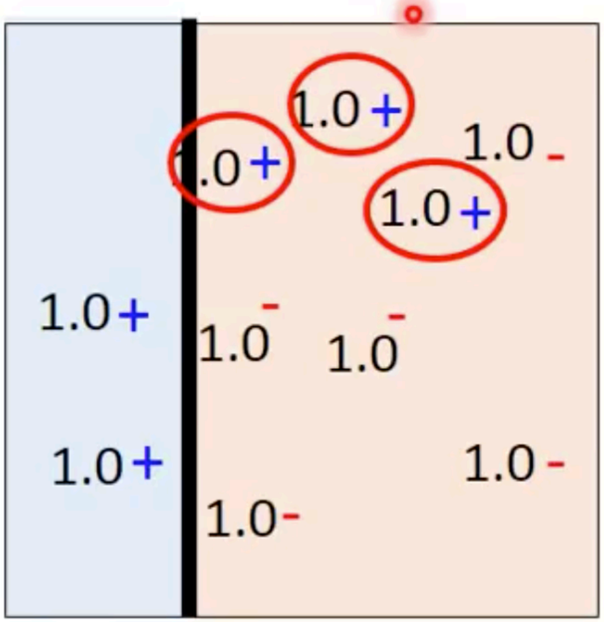
$$d_1 = 1.53$$

$$\alpha_1 = 0.42$$

Toy Example

T=3, weak classifier = decision stump

- t=1

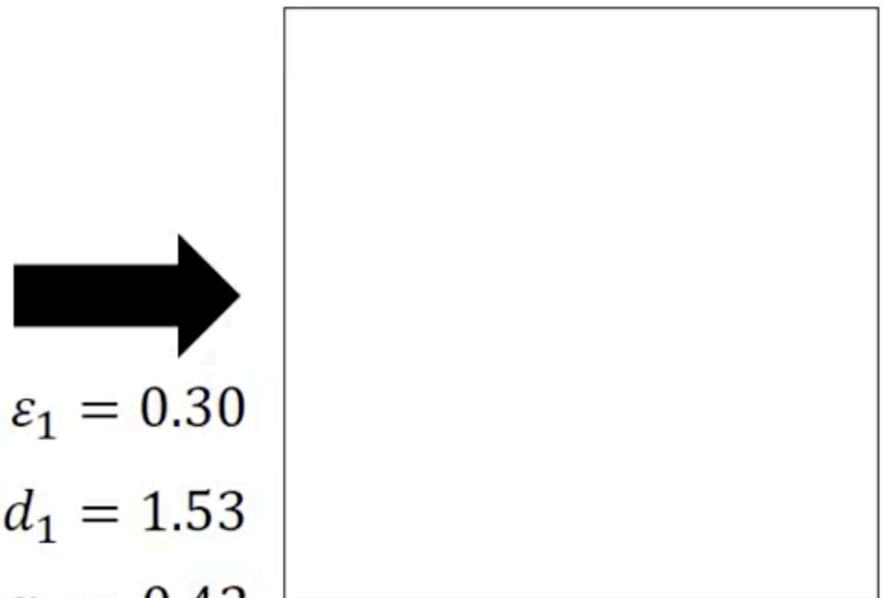


$f_1(x)$

$\varepsilon_1 = 0.30$

$d_1 = 1.53$

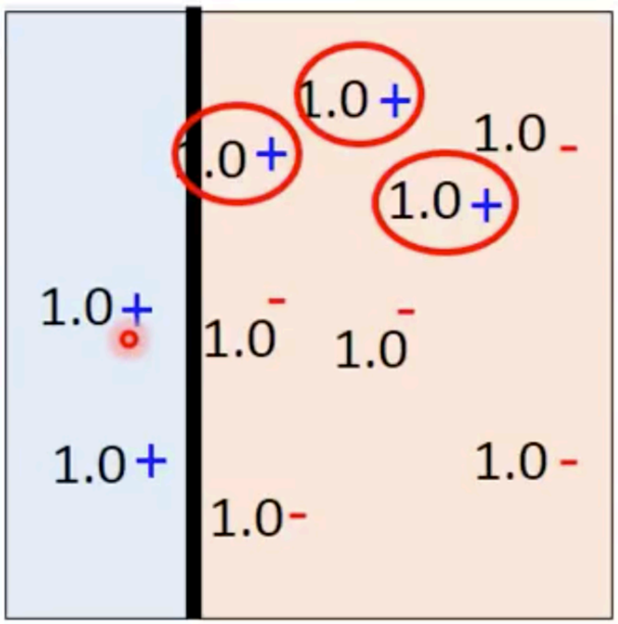
$\alpha_1 = 0.42$



Toy Example

T=3, weak classifier = decision stump

- t=1

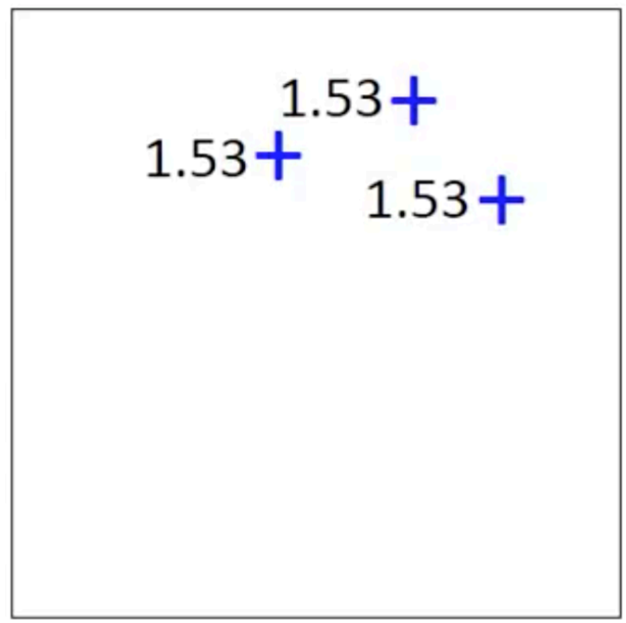


$f_1(x)$

$\varepsilon_1 = 0.30$

$d_1 = 1.53$

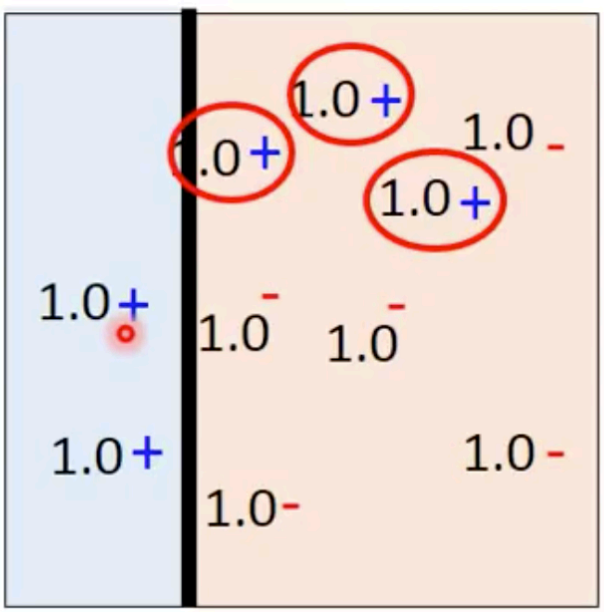
$\alpha_1 = 0.42$



Toy Example

T=3, weak classifier = decision stump

- t=1

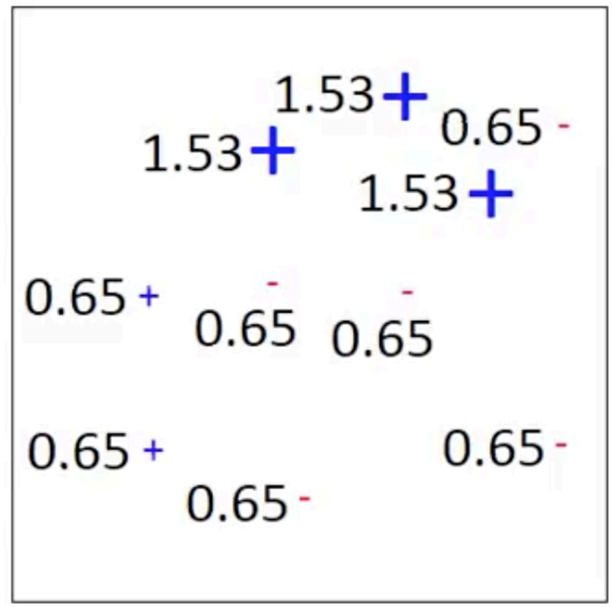


$f_1(x)$

$\varepsilon_1 = 0.30$

$d_1 = 1.53$

$\alpha_1 = 0.42$



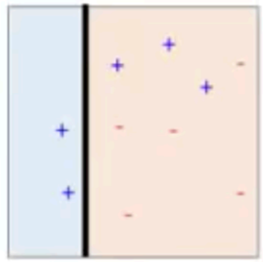
Toy Example

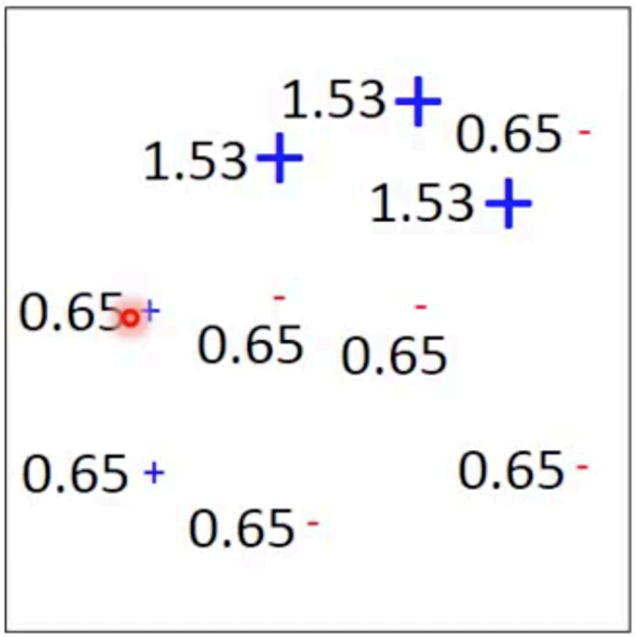
T=3, weak classifier = decision stump

- t=2

$f_1(x)$:

$\alpha_1 = 0.42$





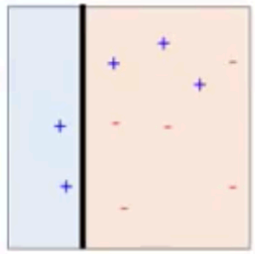
Toy Example

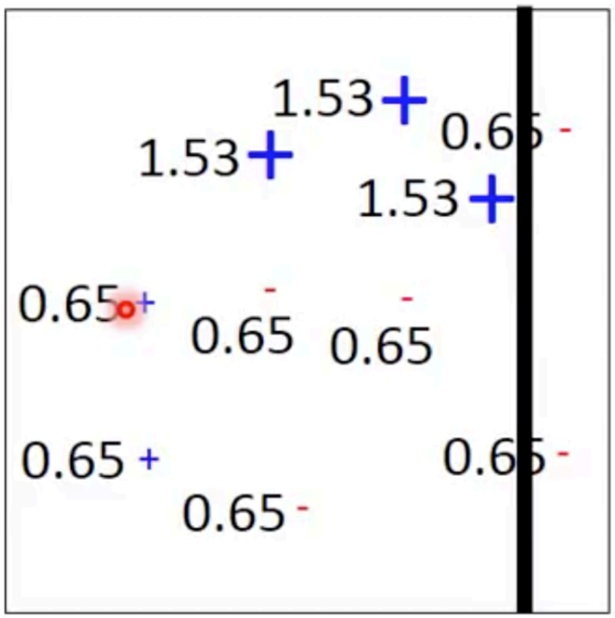
T=3, weak classifier = decision stump

- t=2

$f_1(x)$:

$\alpha_1 = 0.42$





$f_2(x)$

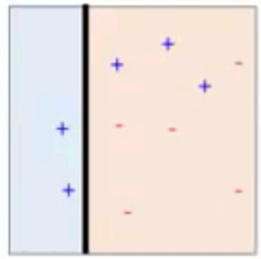
Toy Example

T=3, weak classifier = decision stump

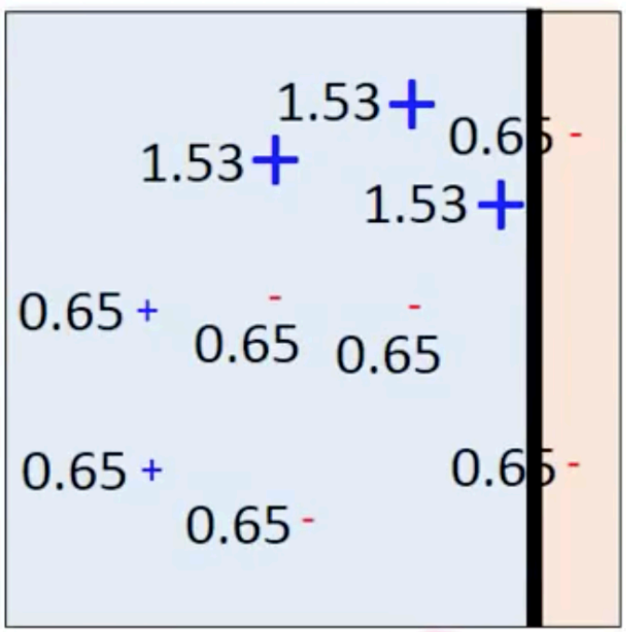
- t=2

$f_1(x)$:

$\alpha_1 = 0.42$



Class +



Class -

$f_2(x)$

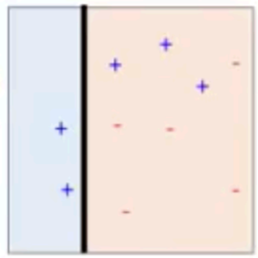
Toy Example

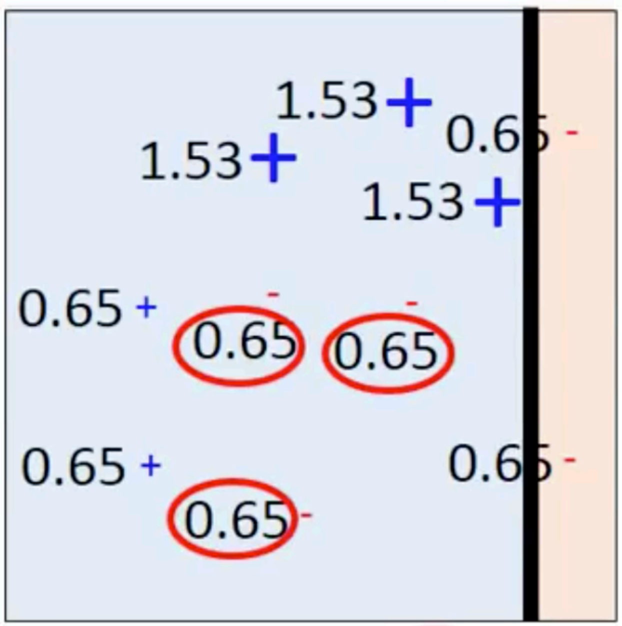
T=3, weak classifier = decision stump

- t=2

$f_1(x)$:

$\alpha_1 = 0.42$





$f_2(x)$

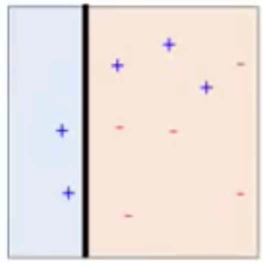
Toy Example

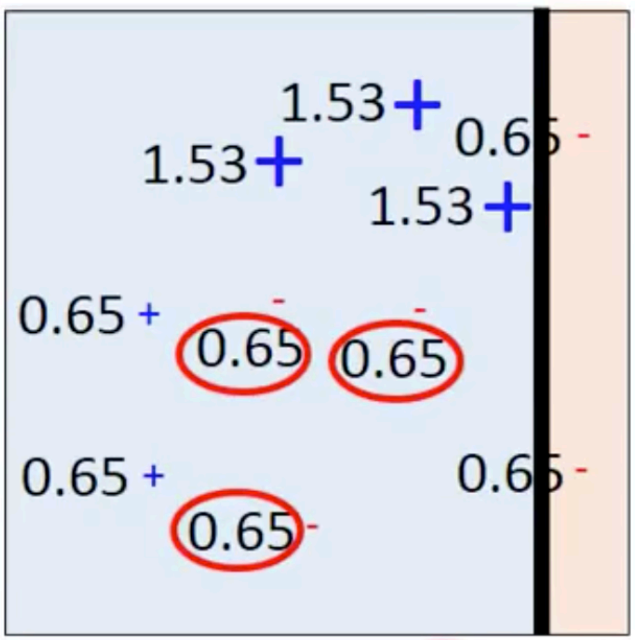
T=3, weak classifier = decision stump

- t=2

$f_1(x)$:

$\alpha_1 = 0.42$





$f_2(x)$

$\varepsilon_2 = 0.21$

$d_2 = 1.94$

$\alpha_2 = 0.66$

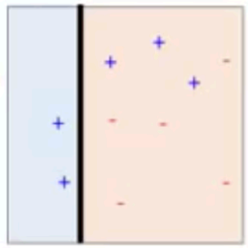
Toy Example

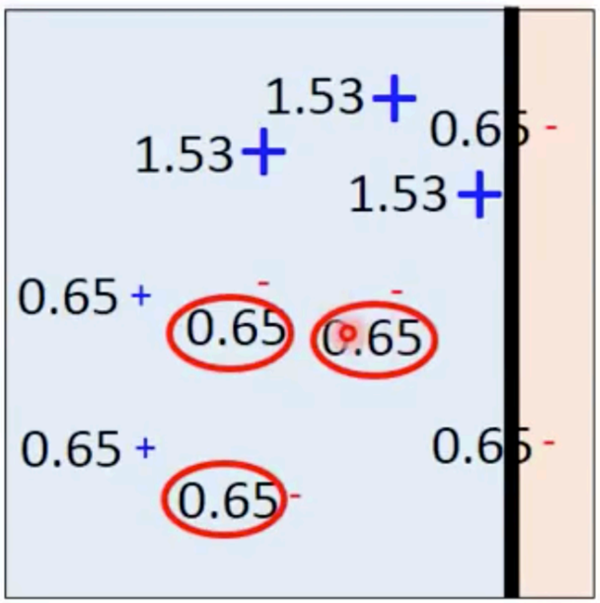
T=3, weak classifier = decision stump

- t=2

$f_1(x)$:

$\alpha_1 = 0.42$





$f_2(x)$

$\varepsilon_2 = 0.21$

$d_2 = 1.94$

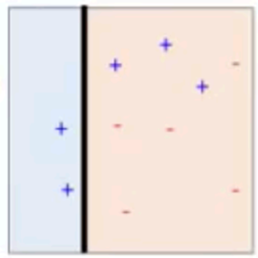
$\alpha_2 = 0.66$

Toy Example

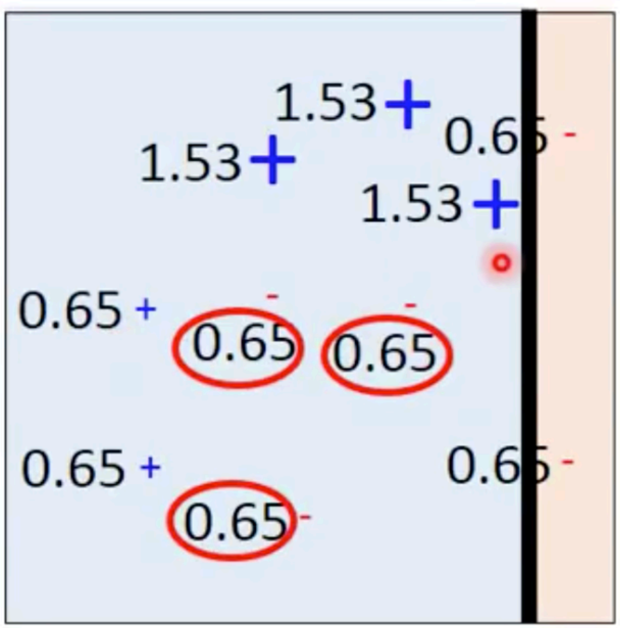
T=3, weak classifier = decision stump

- t=2

$f_1(x)$:



$\alpha_1 = 0.42$

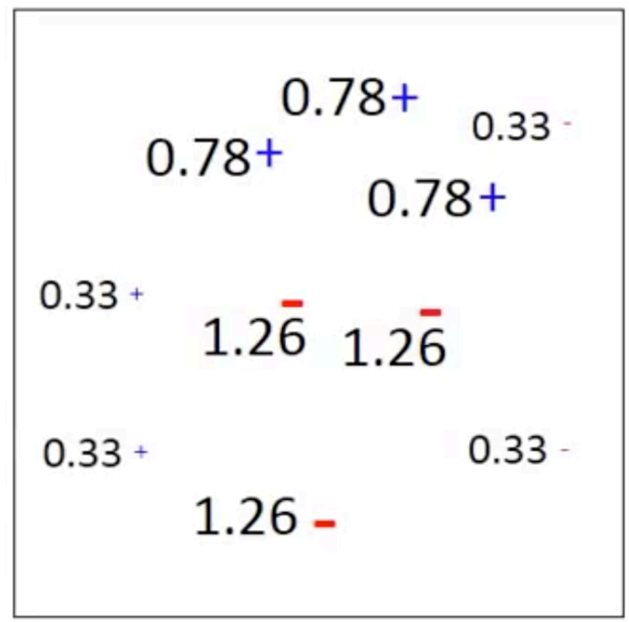


$f_2(x)$

$\varepsilon_2 = 0.21$

$d_2 = 1.94$

$\alpha_2 = 0.66$



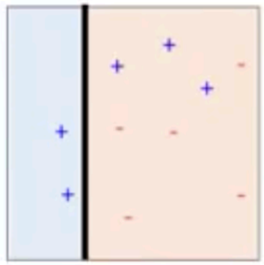
Toy Example

T=3, weak classifier = decision stump

- t=3

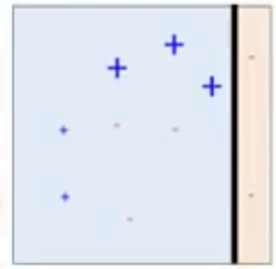
$f_1(x)$:

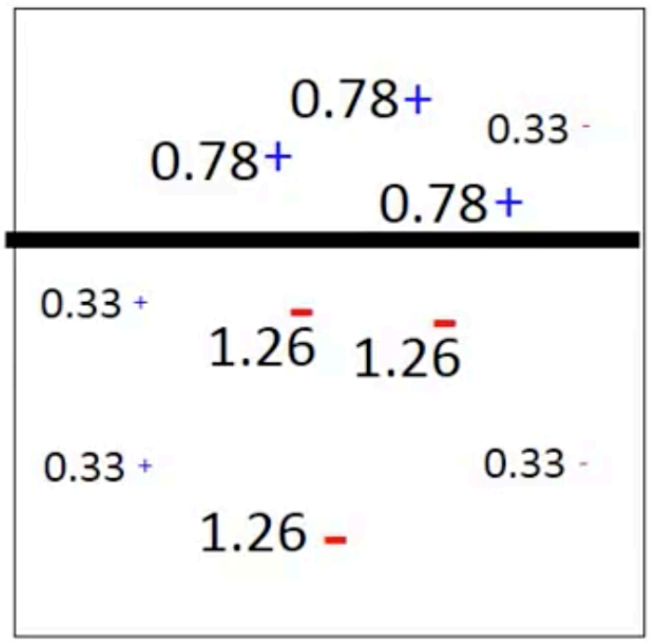
$\alpha_1 = 0.42$



$f_2(x)$:

$\alpha_2 = 0.66$





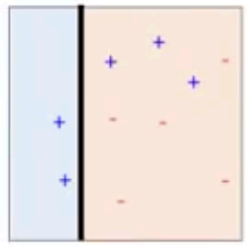
Toy Example

T=3, weak classifier = decision stump

- t=3

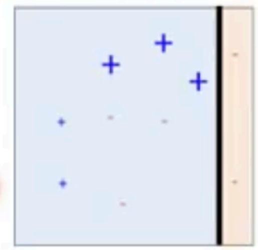
$f_1(x)$:

$\alpha_1 = 0.42$

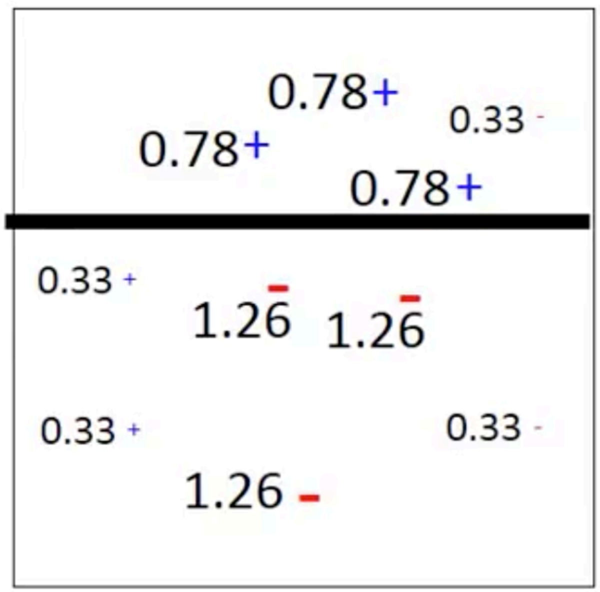


$f_2(x)$:

$\alpha_2 = 0.66$



$f_3(x)$



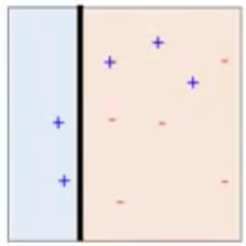
Toy Example

T=3, weak classifier = decision stump

- t=3

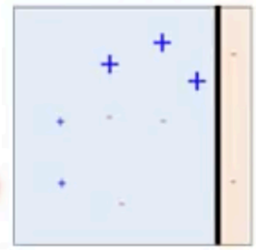
$f_1(x)$:

$\alpha_1 = 0.42$



$f_2(x)$:

$\alpha_2 = 0.66$



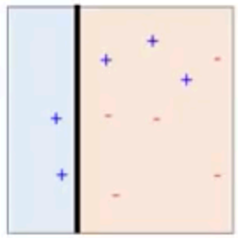
$f_3(x)$

Toy Example

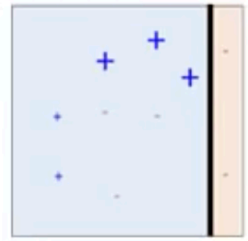
T=3, weak classifier = decision stump

- t=3

$f_1(x)$: $\alpha_1 = 0.42$



$f_2(x)$: $\alpha_2 = 0.66$



$f_3(x)$

$\varepsilon_3 = 0.13$

$d_3 = 2.59$

$\alpha_3 = 0.95$

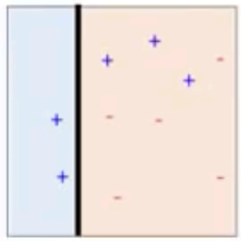
Toy Example

T=3, weak classifier = decision stump

- t=3

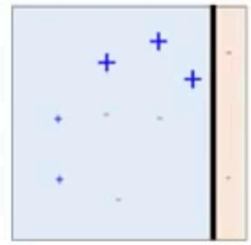
$f_1(x)$:

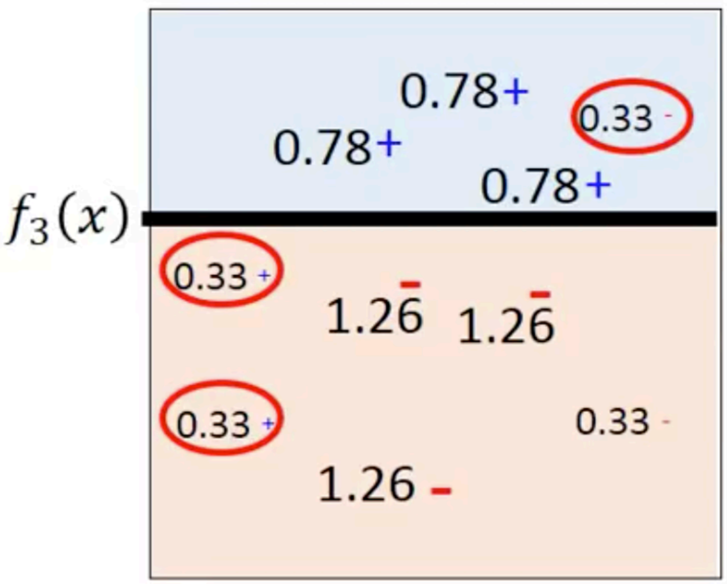
$\alpha_1 = 0.42$



$f_2(x)$:

$\alpha_2 = 0.66$





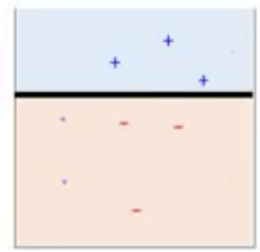
$\varepsilon_3 = 0.13$

$d_3 = 2.59$

$\alpha_3 = 0.95$

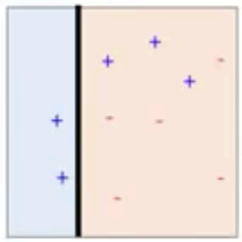
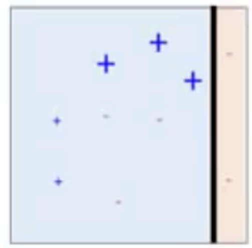
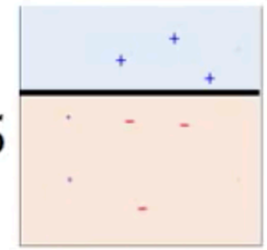
$f_3(x)$:

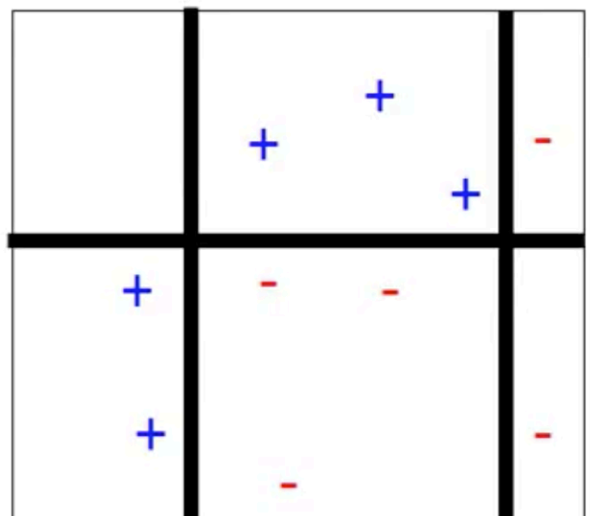
$\alpha_3 = 0.95$



Toy Example

- Final Classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

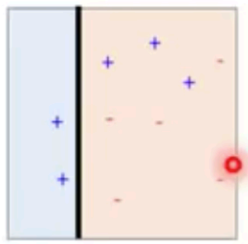
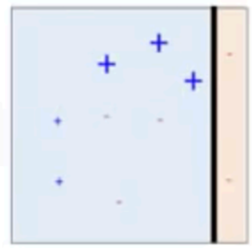
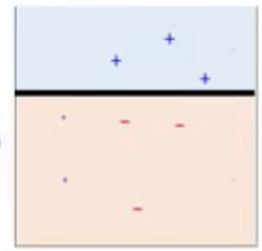
$sign(\ 0.42$  $+\ 0.66$  $+\ 0.95$  $)$

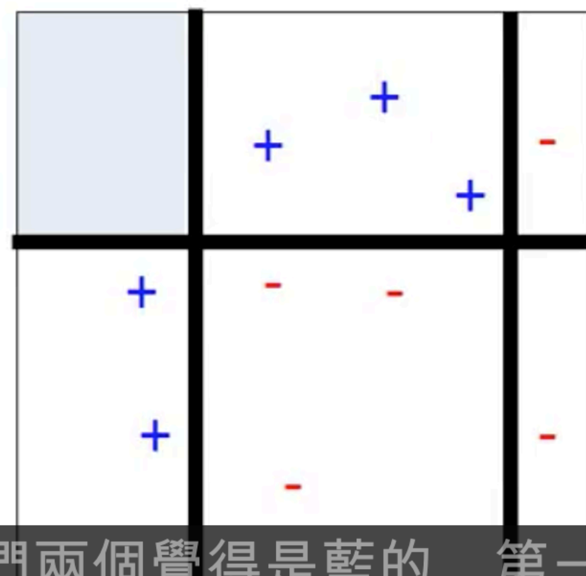


這個加起來的結果到底是怎麼回事

Toy Example

- Final Classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

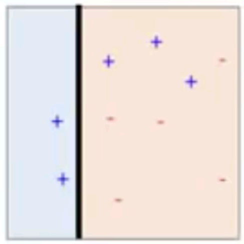
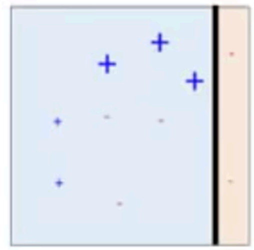
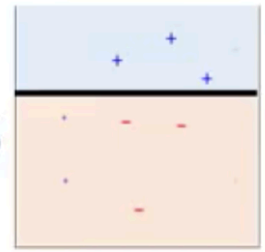
$sign($ 0.42  + 0.66  + 0.95  $)$

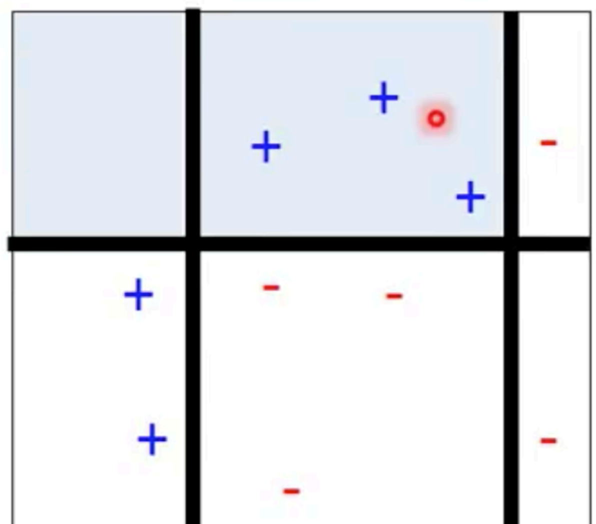


中間這一塊他們兩個覺得是藍的，第一個覺得是紅的

Toy Example

- Final Classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

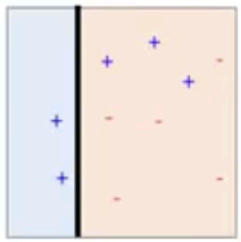
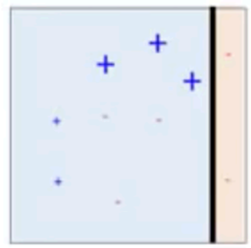
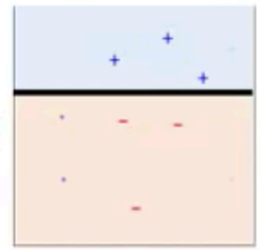
$sign($ 0.42  + 0.66  + 0.95  $)$

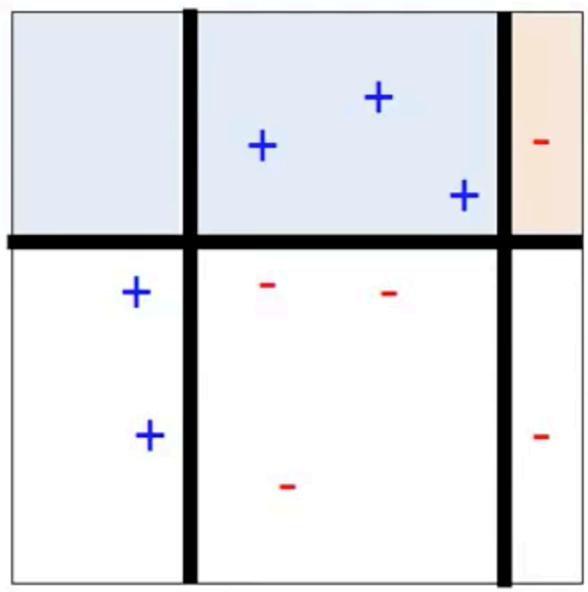


所以上面這組就是藍的

Toy Example

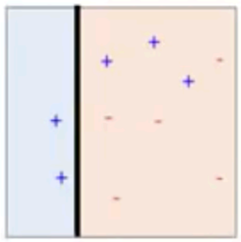
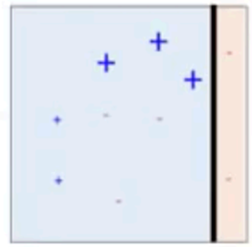
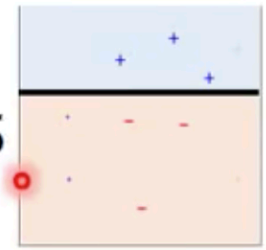
- Final Classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

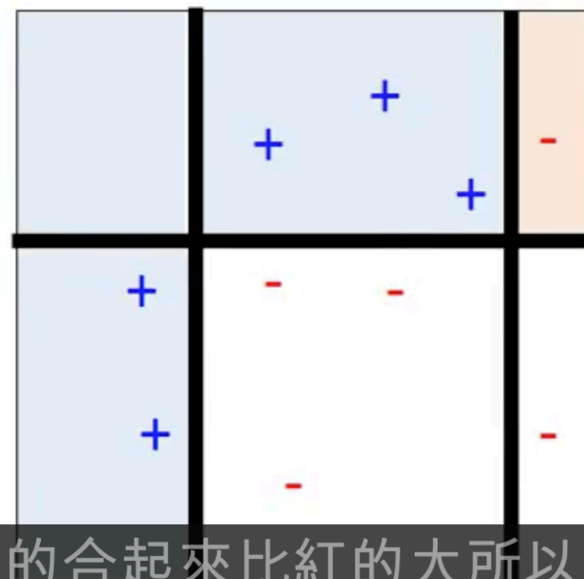
$sign($ 0.42  + 0.66  + 0.95 )



Toy Example

- Final Classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

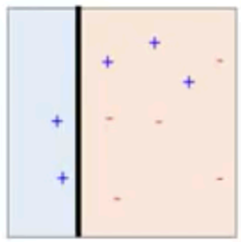
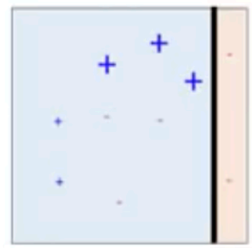
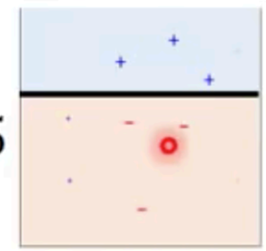
$sign($ 0.42  + 0.66  + 0.95  $)$

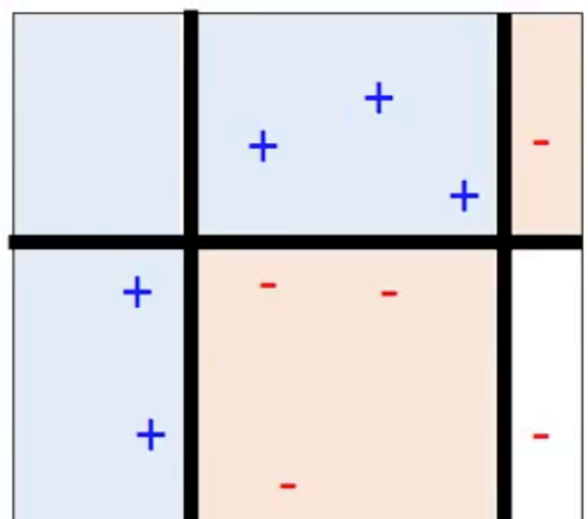


兩個藍的合起來比紅的大所以是藍的

Toy Example

- Final Classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

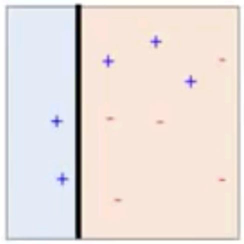
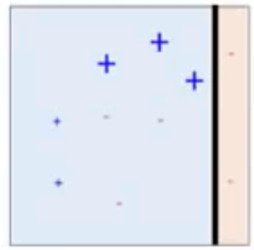
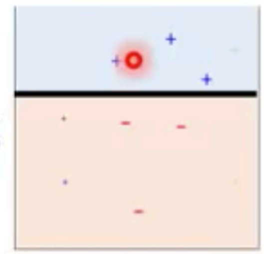
$sign($ 0.42  + 0.66  + 0.95  $)$

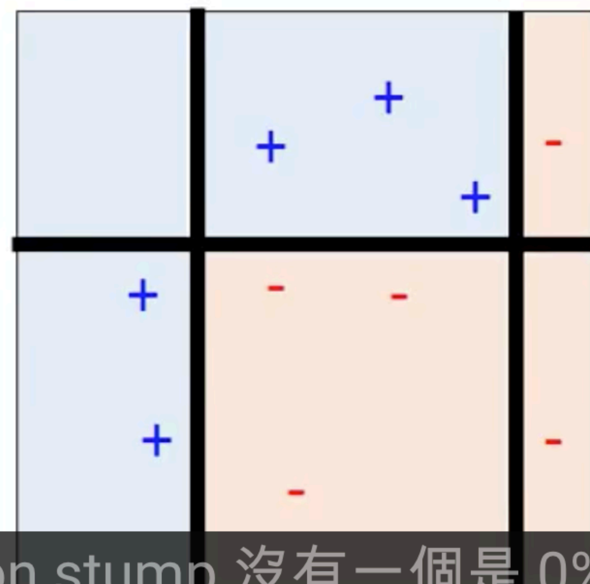


右下角三個 decision stump 都說是紅的所以是紅的

Toy Example

- Final Classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

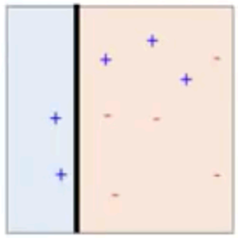
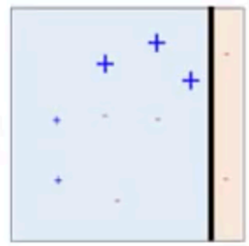
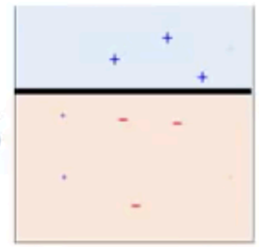
$sign($ 0.42  $+$ 0.66  $+$ 0.95  $)$

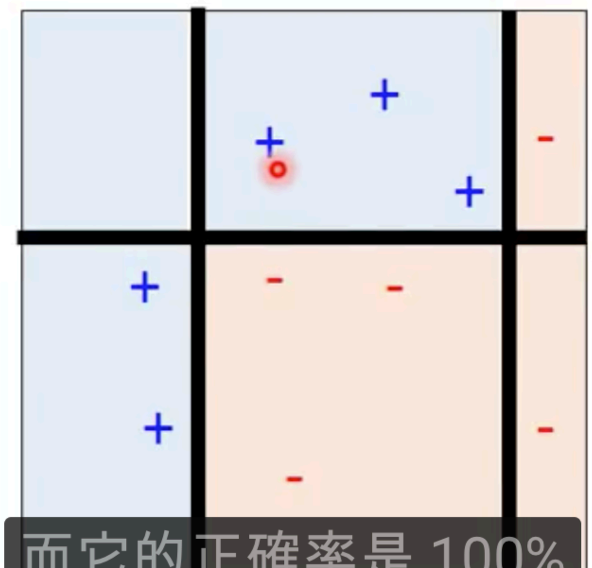


這三個 decision stump 沒有一個是 0% 的 Error Rate

Toy Example

- Final Classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

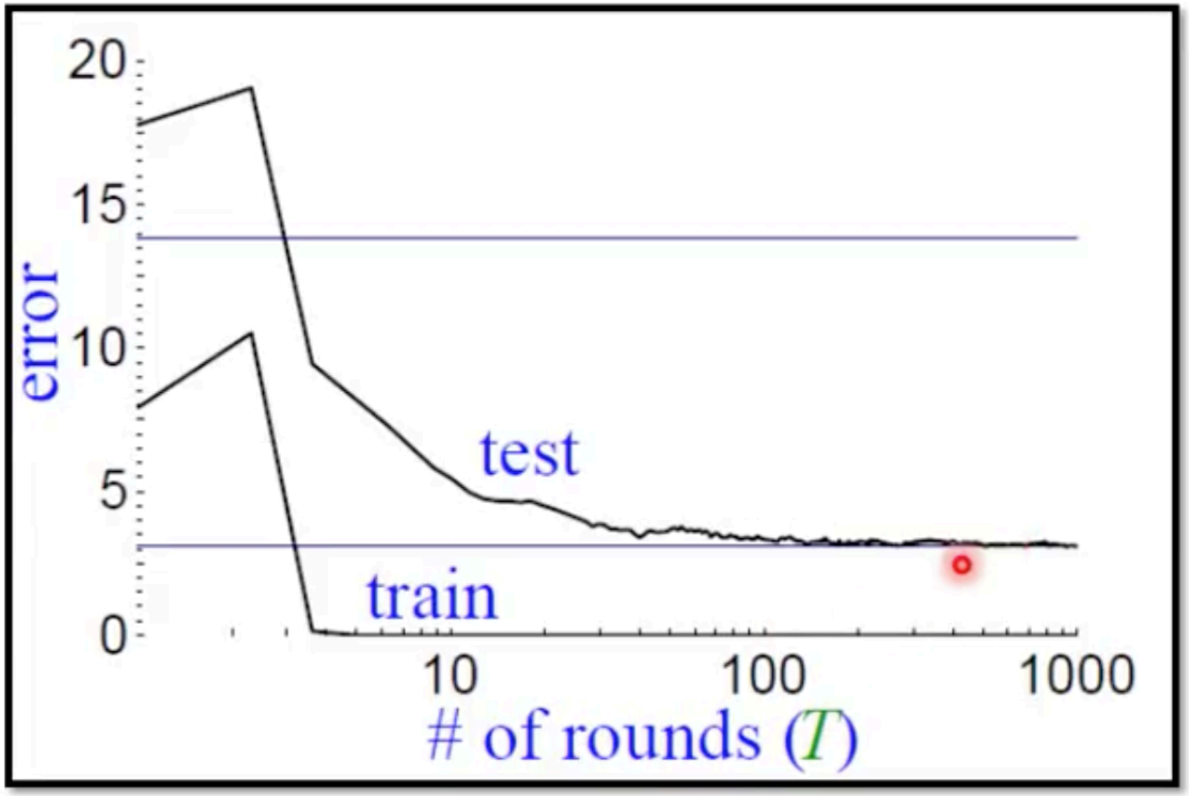
$sign($ 0.42  + 0.66  + 0.95  $)$

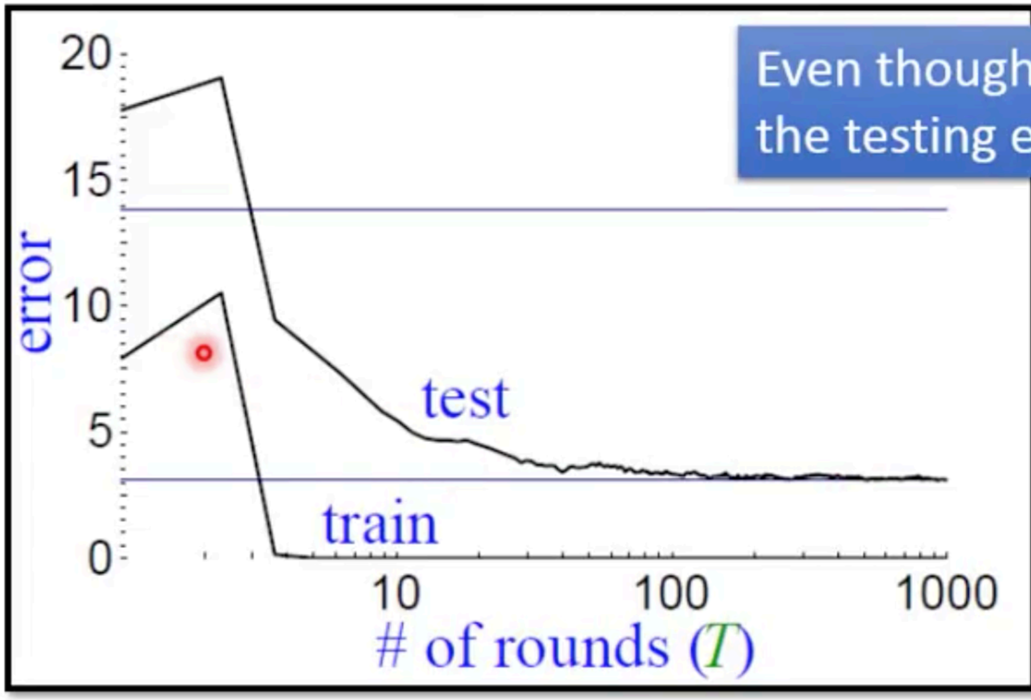


Final Error Rate = 0

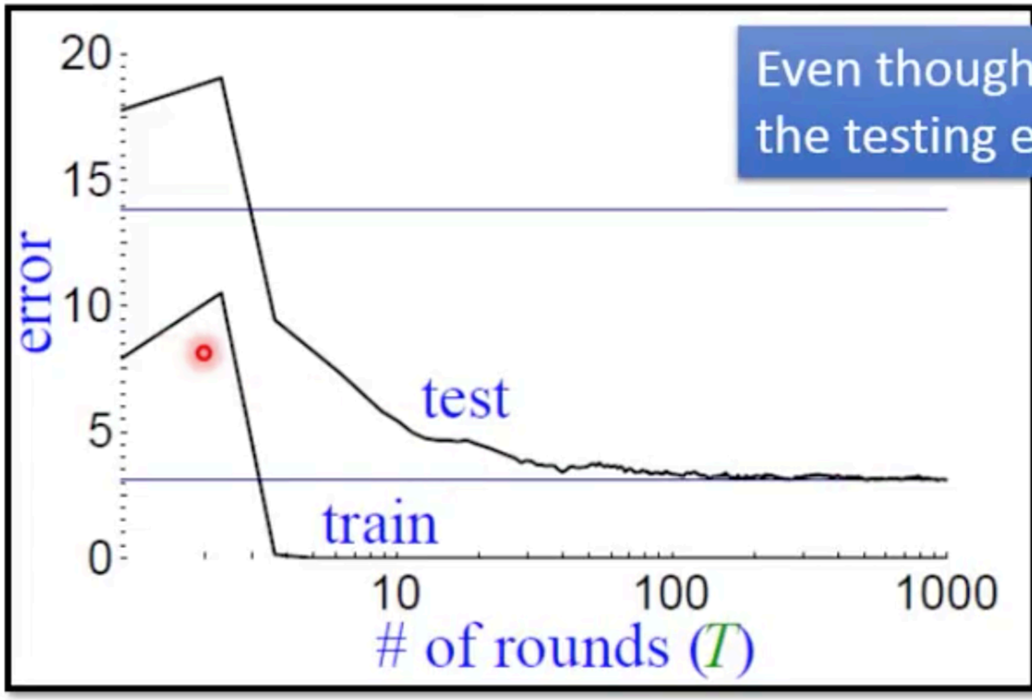
而它的正確率是 100%

Another example:





Even though the training error is 0, the testing error still decreases?



Even though the training error is 0, the testing error still decreases?

Yes, because even if the combined classifiers have error rate = 0 on training dataset, the last classifier for the training dataset is > 0. So we will continue to find a new classifier.