Ensemble

Introduction

- We are almost at the end of the semester/final competition.
  - https://inclass.kaggle.com/c/ml2016-cyber-security-attack-defender/leaderboard
  - https://www.kaggle.com/c/outbrain-click-prediction/leaderboard
  - https://www.kaggle.com/c/transfer-learning-on-stack-exchange-tags/leaderboard
- You already developed some algorithms and codes. Lazy to modify them.
- Ensemble: improving your machine with little modification

Framework of Ensemble

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

坦　　補　　DD　　They should be diverse.

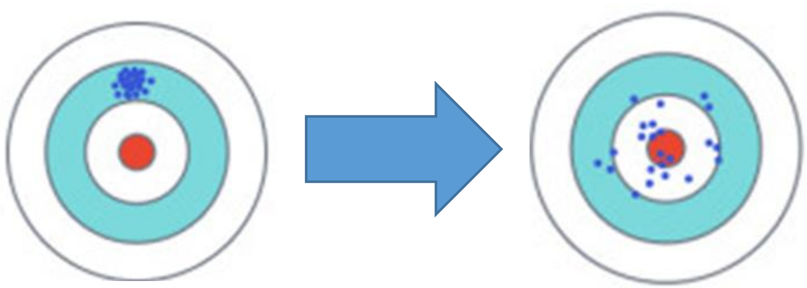
- Aggregate the classifiers (*properly*)
  - 在打王時每個人都有該站的位置

Ensemble: Bagging

Review: Bias v.s. Variance

Large Bias

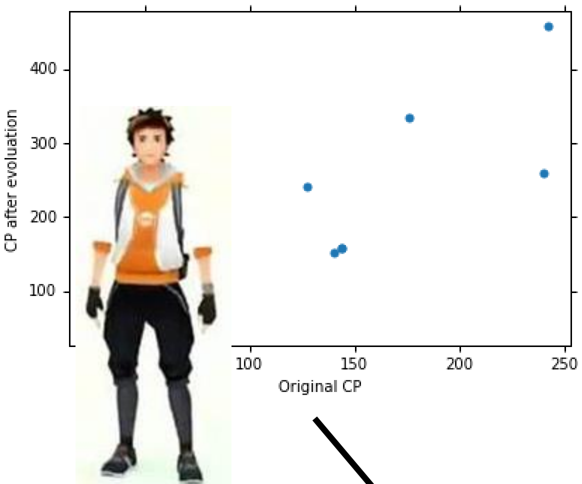
Small Variance



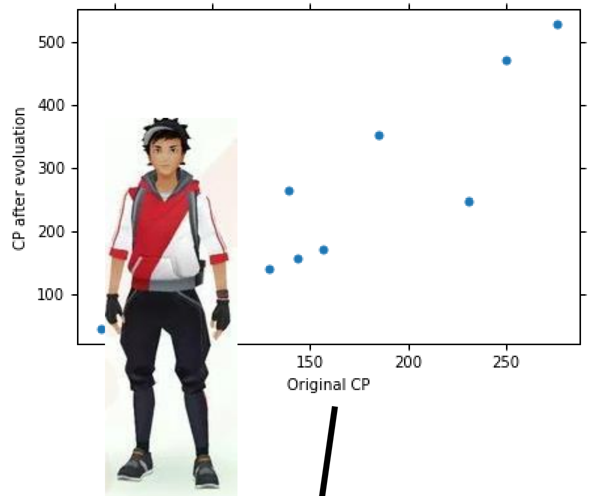
Small Bias

Large Variance

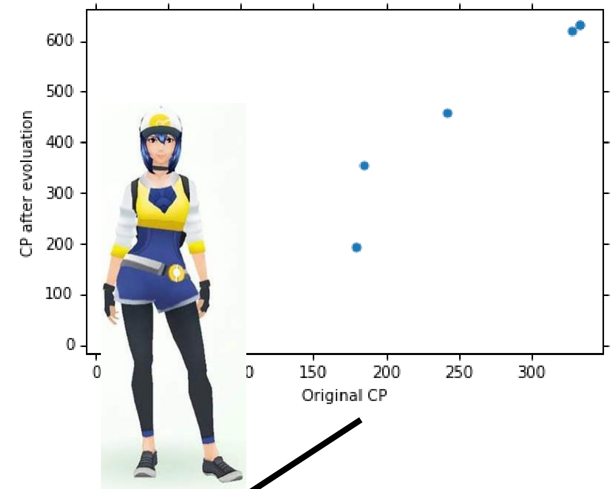
Universe 1



Universe 2

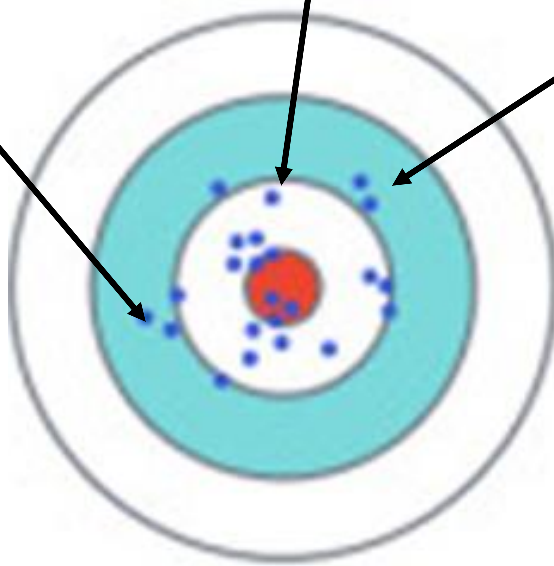


Universe 3



A complex model will have large variance.

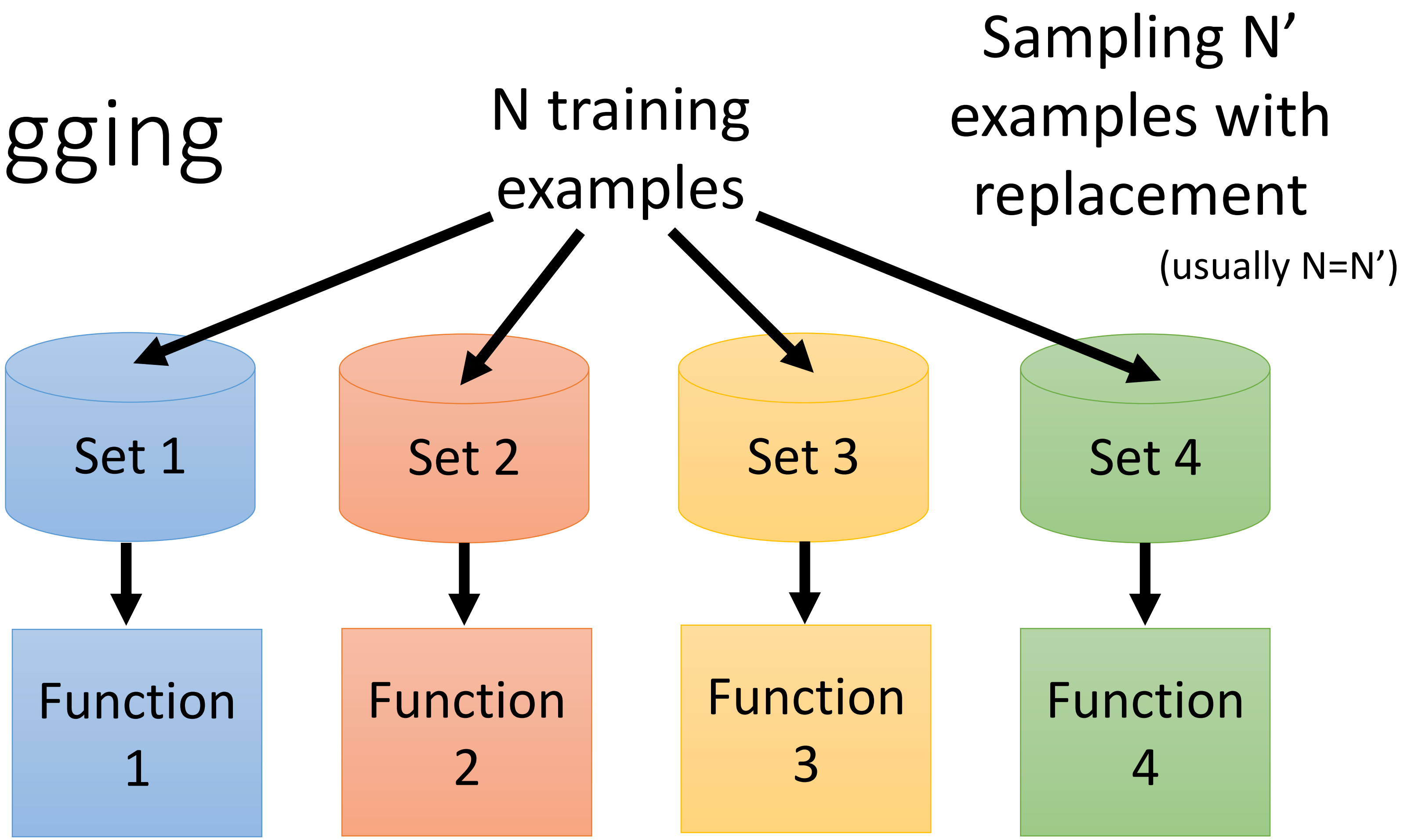
We can average complex models to reduce variance.



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

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

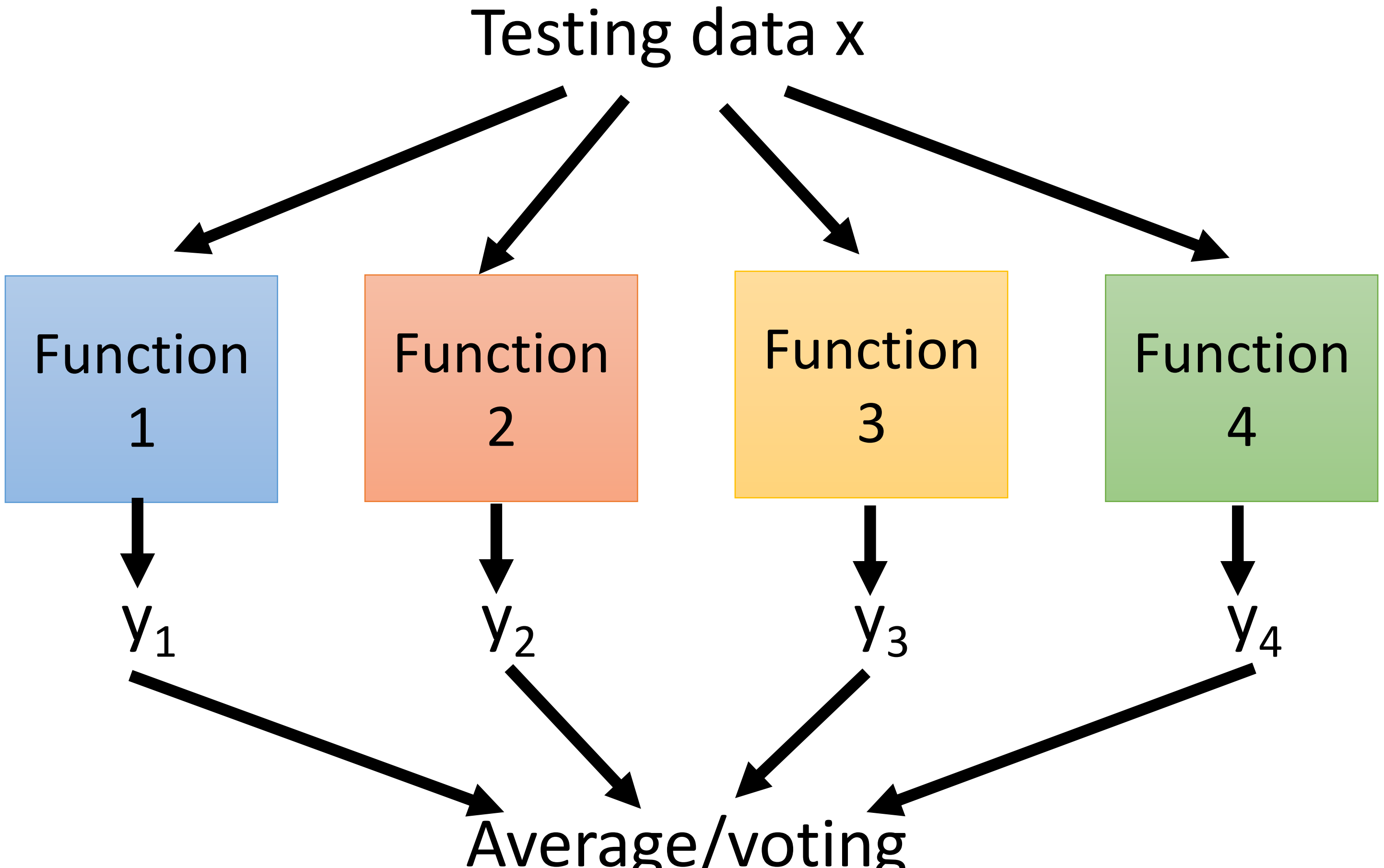
Bagging


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

Bagging

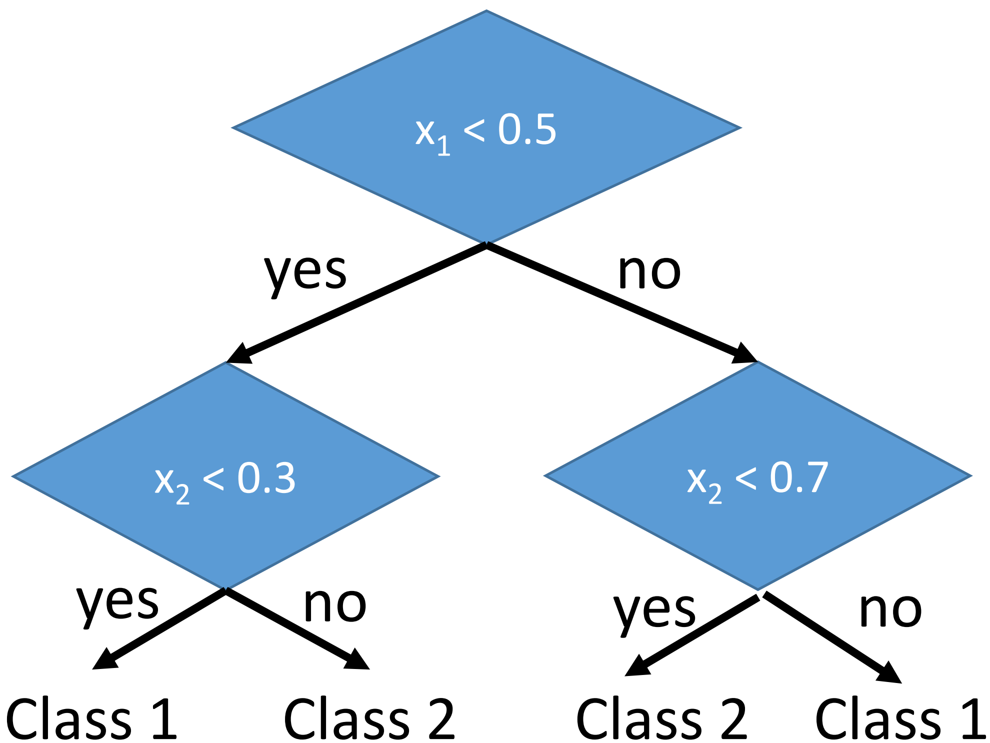
This approach would be helpful when your model is complex, easy to overfit.

e.g. decision tree

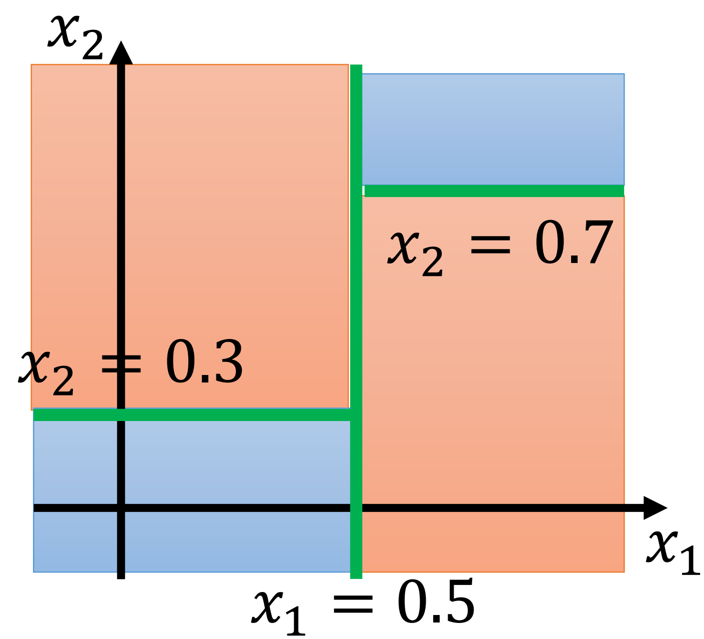


Decision Tree

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



Can have more complex questions



The questions in training .....

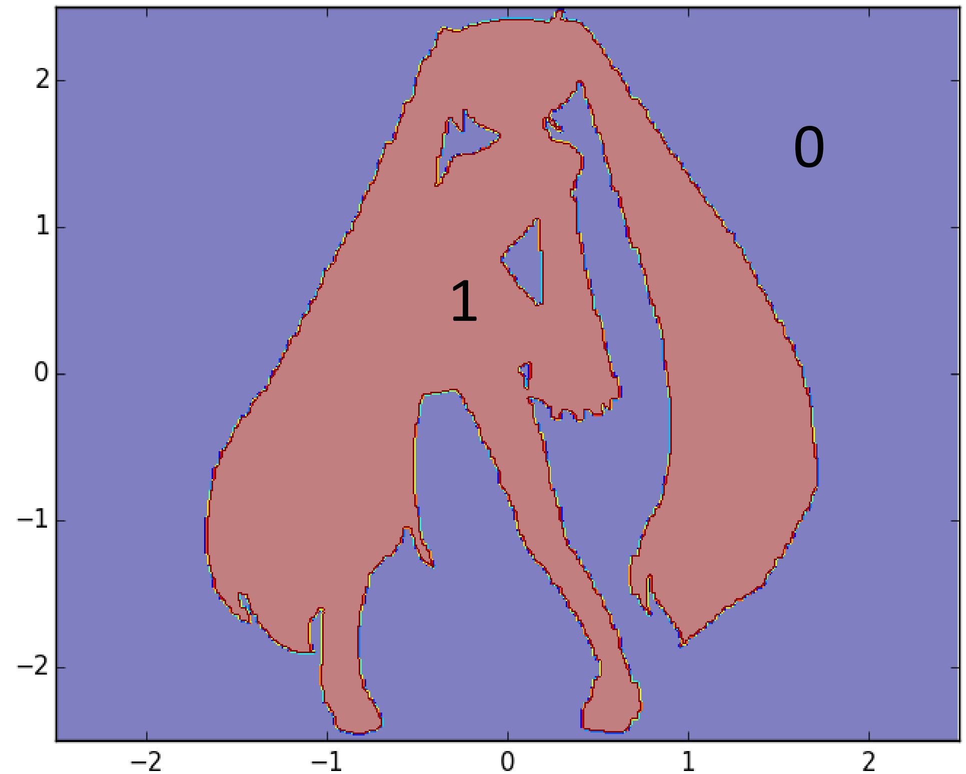
number of branches,

Branching criteria,

termination criteria,

base hypothesis

Experiment: Function of Miku

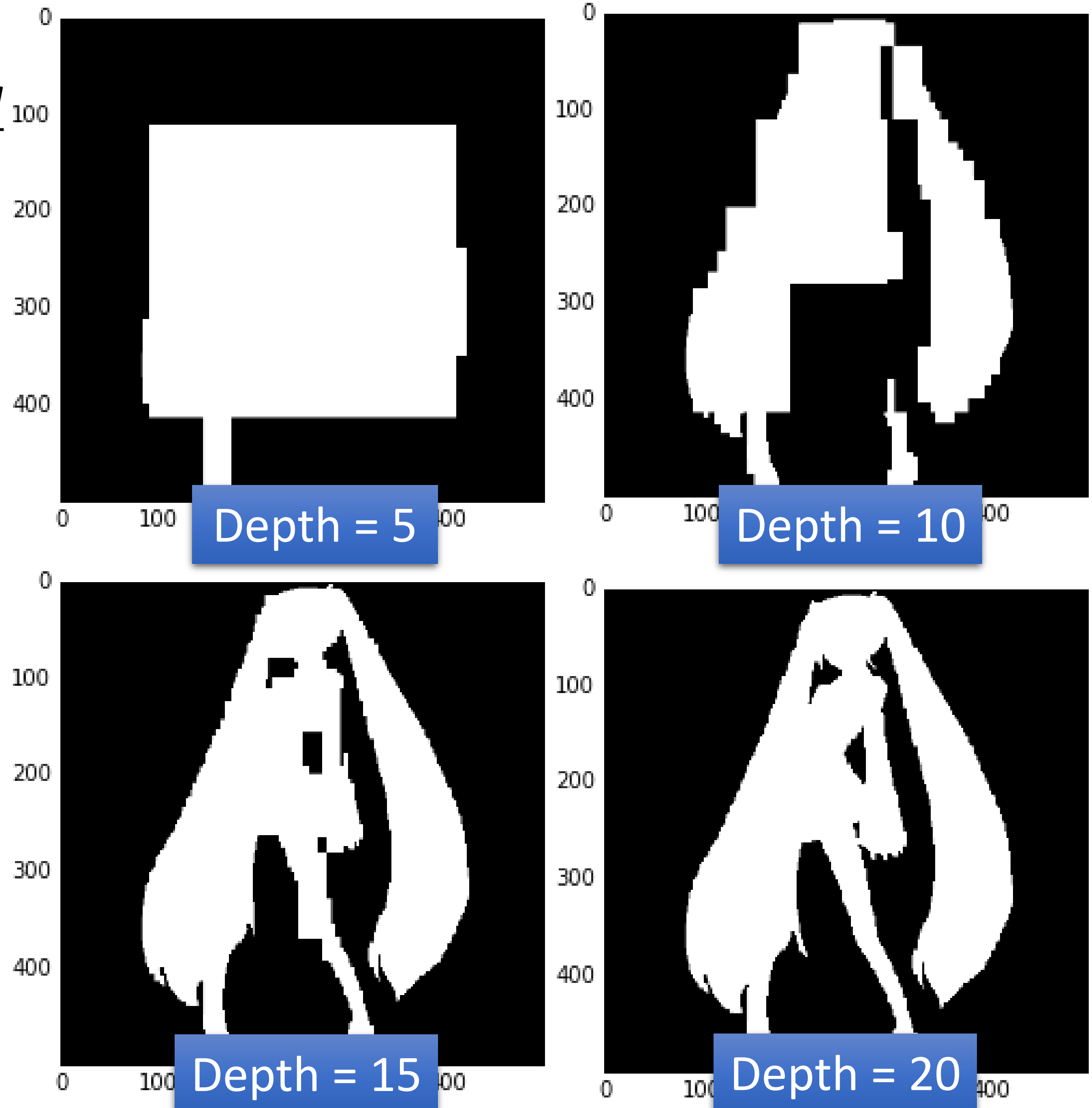


http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/theano/miku

(1st column: x, 2nd column: y, 3rd column: output (1 or 0))

Experiment: Function of Miku

Single Decision Tree



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

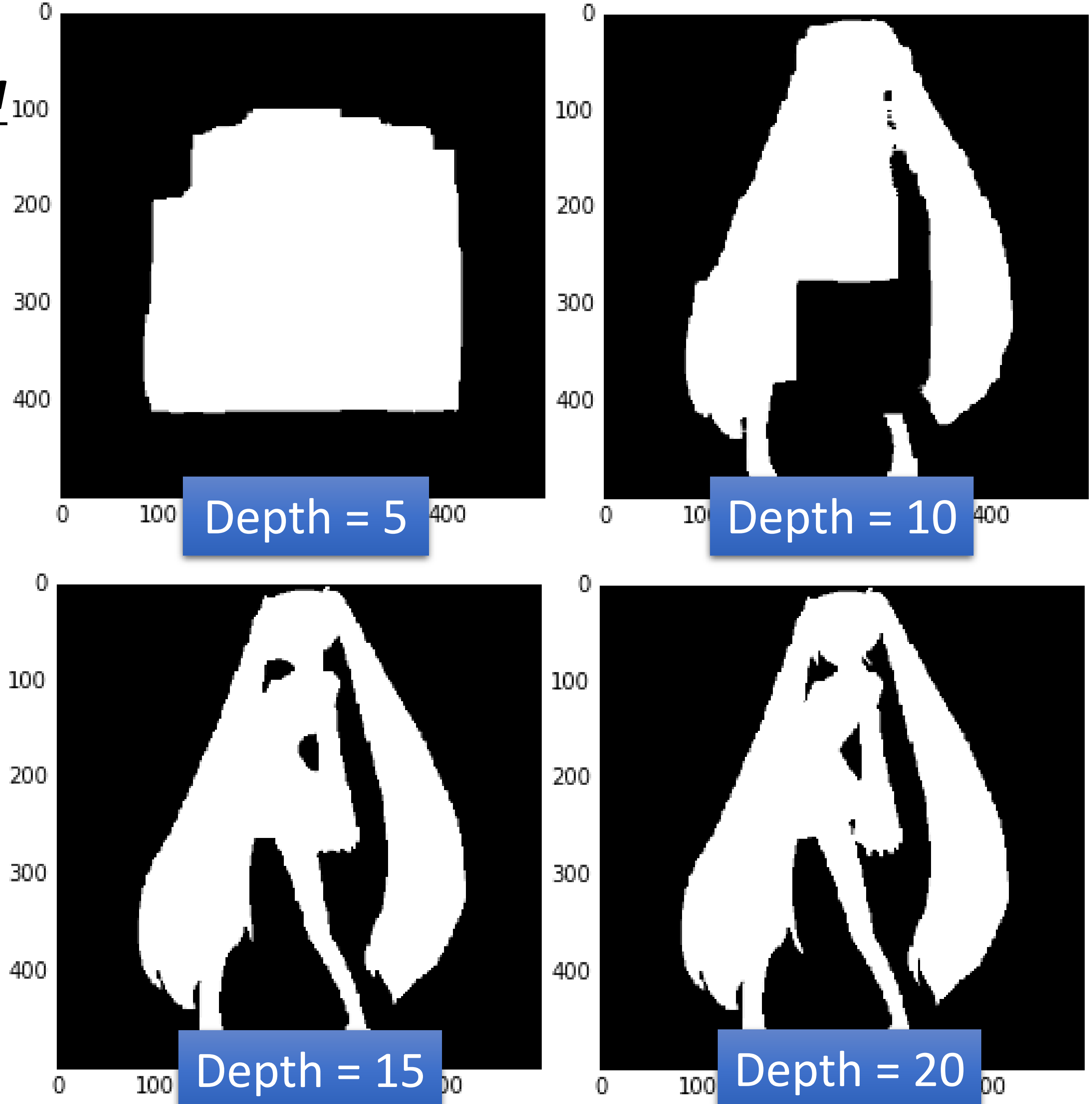
Out-of-bag (OOB) error

Good error estimation of testing set

Experiment: Function of Miku

Random Forest

(100 trees)



Ensemble: Boosting

Improving Weak Classifiers

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
  - 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 $\boldsymbol{f_2(x)}$ 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.

Training $f_2(x)$ based on the new 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 $f_1(x)$?

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

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

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

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

ε_1 =0.25 $\rightarrow$ $f_1(x)$ $\rightarrow$ 0.5 $\rightarrow$ ε_2 <0.5 $\rightarrow$ $f_2(x)$

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 $f_1(x)$?

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

$u_2^n \leftarrow u_1^n$ multiplying d_1 increase

If x^n correctly classified by f_1 ($f_1(x^n) = \hat{y}^n$)

$u_2^n \leftarrow u_1^n$ devided by d_1 decrease

f_2 will be learned based on example weights u_2^n

What is the value of d_1?

Re-weighting Training Data

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

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

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

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

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

$$= \sum_n u_2^n$$

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

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

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

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

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

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

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

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

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

$$d_1 = \sqrt{(1-\varepsilon_1)/\varepsilon_1} > 1$$

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)
- 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 classified by $f_t(x)$:
    - $u_{t+1}^n = u_t^n \times d_t = u_t^n \times \exp(\alpha_t)$
    - Else:
    - $u_{t+1}^n = u_t^n / d_t = u_t^n \times \exp(-\alpha_t)$

$\hat{y}^n \neq f_t(x^n)$

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

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

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

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

Smaller error ε_t, larger weight for final voting

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

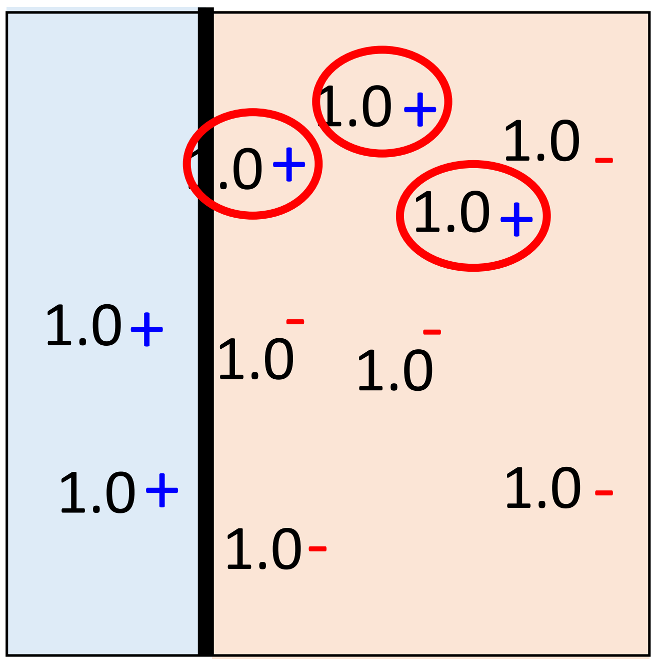
$u_{t+1}^{n} = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$

$\epsilon^t = 0.1$	$\epsilon^t = 0.4$
$\alpha^t = 1.10$	$\alpha^t = 0.20$

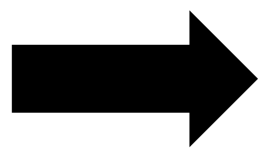
Toy Example

T=3, weak classifier = decision stump

- t=1



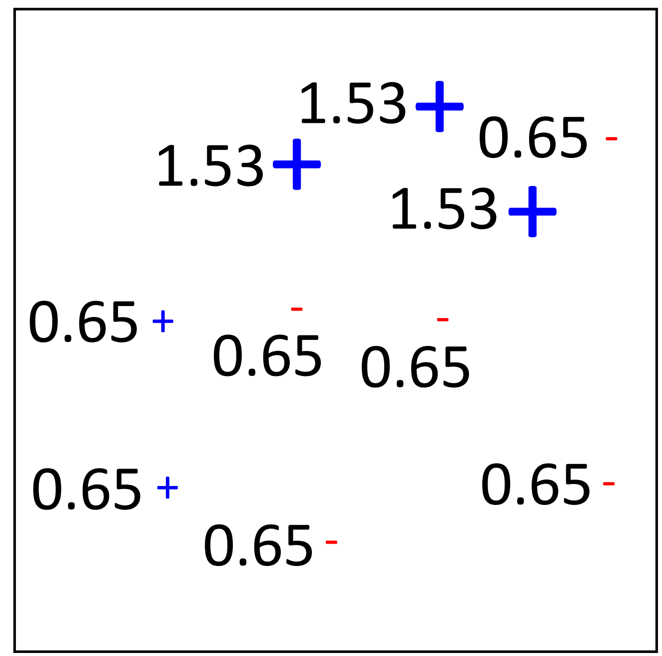
$f_1(x)$



$\varepsilon_1 = 0.30$

$d_1 = 1.53$

$\alpha_1 = 0.42$



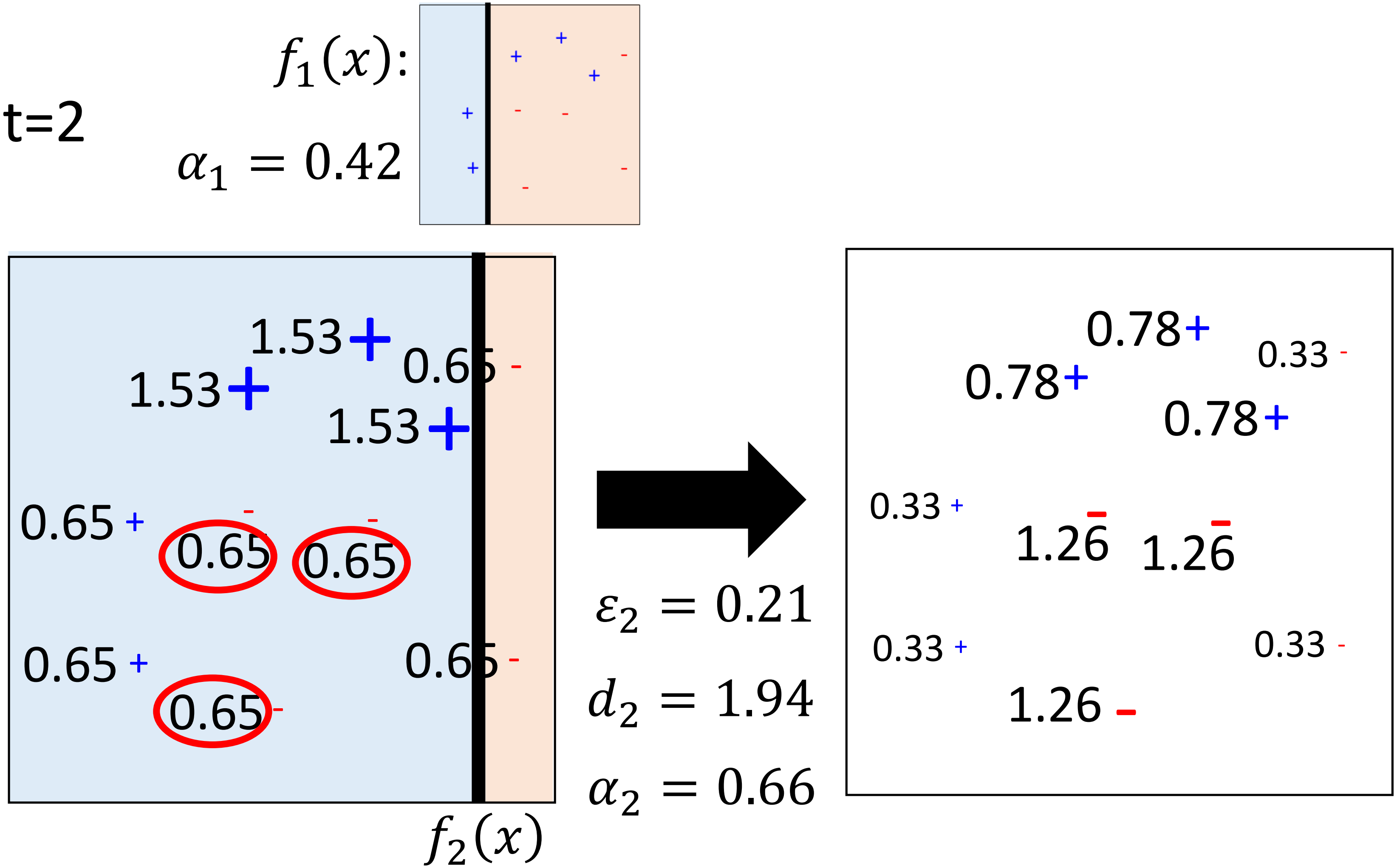
Toy Example

T=3, weak classifier = decision stump

- t=2

$f_1(x)$:

$\alpha_1 = 0.42$



$\varepsilon_2 = 0.21$

$d_2 = 1.94$

$\alpha_2 = 0.66$

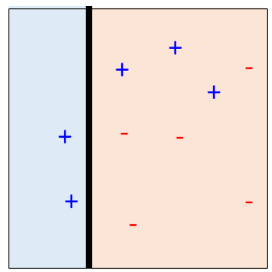
Toy Example

T=3, weak classifier = decision stump

- t=3

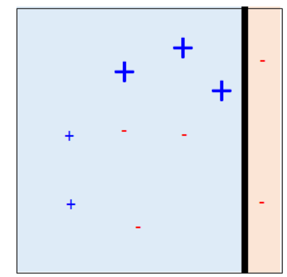
$f_1(x)$:

$\alpha_1 = 0.42$

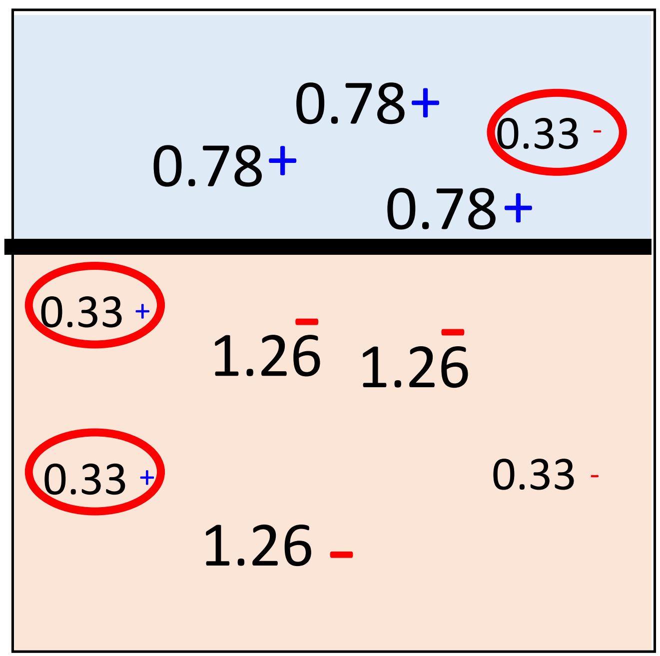


$f_2(x)$:

$\alpha_2 = 0.66$



$f_3(x)$



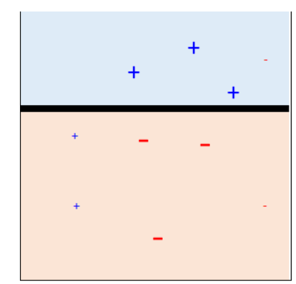
$\varepsilon_3 = 0.13$

$d_3 = 2.59$

$\alpha_3 = 0.95$

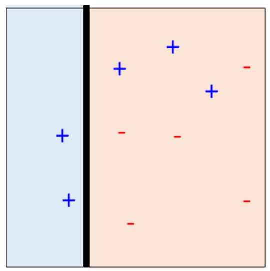
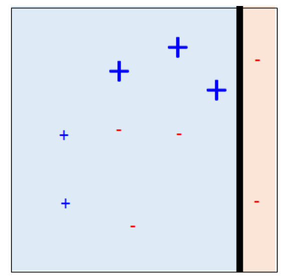
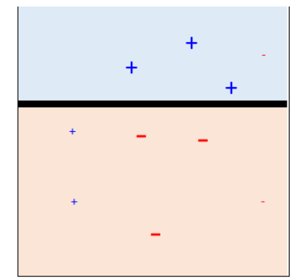
$f_3(x)$:

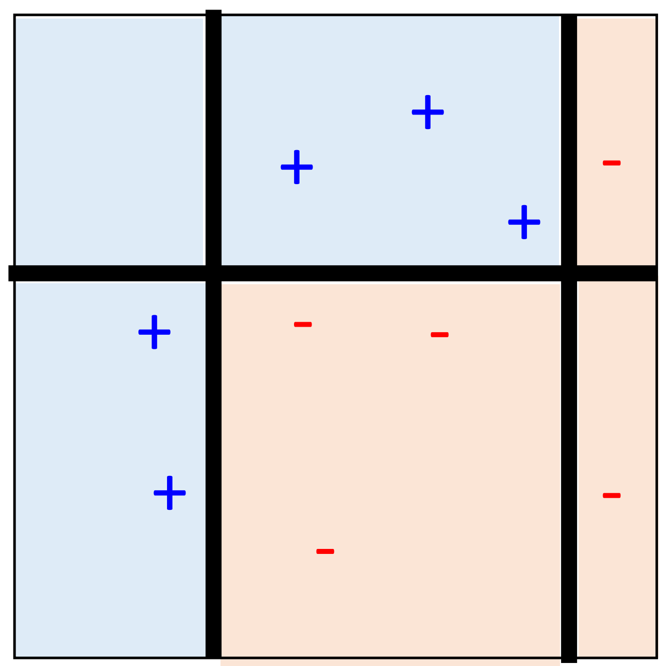
$\alpha_3 = 0.95$



Toy Example

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

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



Warning of Math

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right) \quad \alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

As we have more and more f_t (T increases), $H(x)$ achieves smaller and smaller error rate on training data.

Error Rate of Final Classifier

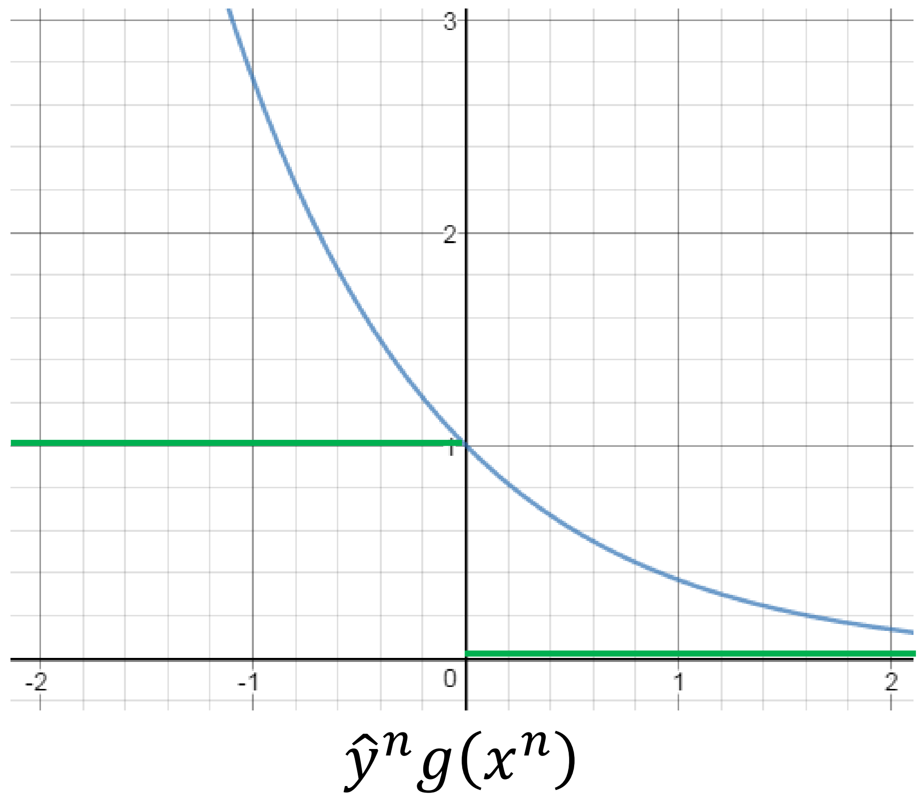
- Final classifier: $H(x) = sign(\underbrace{\sum_{t=1}^{T} \alpha_t f_t(x)}_{g(x)})$
  - $\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$

Training Data Error Rate

$$= \frac{1}{N}\sum_n \delta(H(x^n) \neq \hat{y}^n)$$

$$= \frac{1}{N}\sum_n \delta(\hat{y}^n g(x^n) < 0)$$

$$\leq \frac{1}{N}\sum_n exp(-\hat{y}^n g(x^n))$$



Training Data Error Rate

$$\le \frac{1}{N}\sum_{n} exp(-\hat{y}^n g(x^n)\ ) \boxed{= \frac{1}{N} Z_{T+1}}$$

$$g(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

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

Z_t: the summation of the weights of training data for training f_t

What is $Z_{T+1} =?$ $\quad Z_{T+1} = \sum_{n} u_{T+1}^n$

$$\left.\begin{aligned} &u_1^n = 1 \\ &u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t) \end{aligned}\right\} \quad u_{T+1}^n = \prod_{t=1}^{T} exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$$Z_{T+1} = \sum_{n}\prod_{t=1}^{T} exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$$= \sum_{n} exp\left(-\hat{y}^n \underbrace{\sum_{t=1}^{T} f_t(x^n)\alpha_t}_{g(x)}\right)$$

Training Data Error Rate

$$\leq \frac{1}{N}\sum_n exp(-\hat{y}^n g(x^n)) = \frac{1}{N} Z_{T+1}$$

$$g(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

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

$Z_1 = N$ (equal weights)

$$Z_t = \underbrace{Z_{t-1}\varepsilon_t}_{\text{Misclassified portion in } Z_{t-1}} exp(\alpha_t) + \underbrace{Z_{t-1}(1-\varepsilon_t)}_{\text{Correctly classified portion in } Z_{t-1}} exp(-\alpha_t)$$

$$= Z_{t-1}\varepsilon_t\sqrt{(1-\varepsilon_t)/\varepsilon_t} + Z_{t-1}(1-\varepsilon_t)\sqrt{\varepsilon_t/(1-\varepsilon_t)}$$

$$= Z_{t-1} \times 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

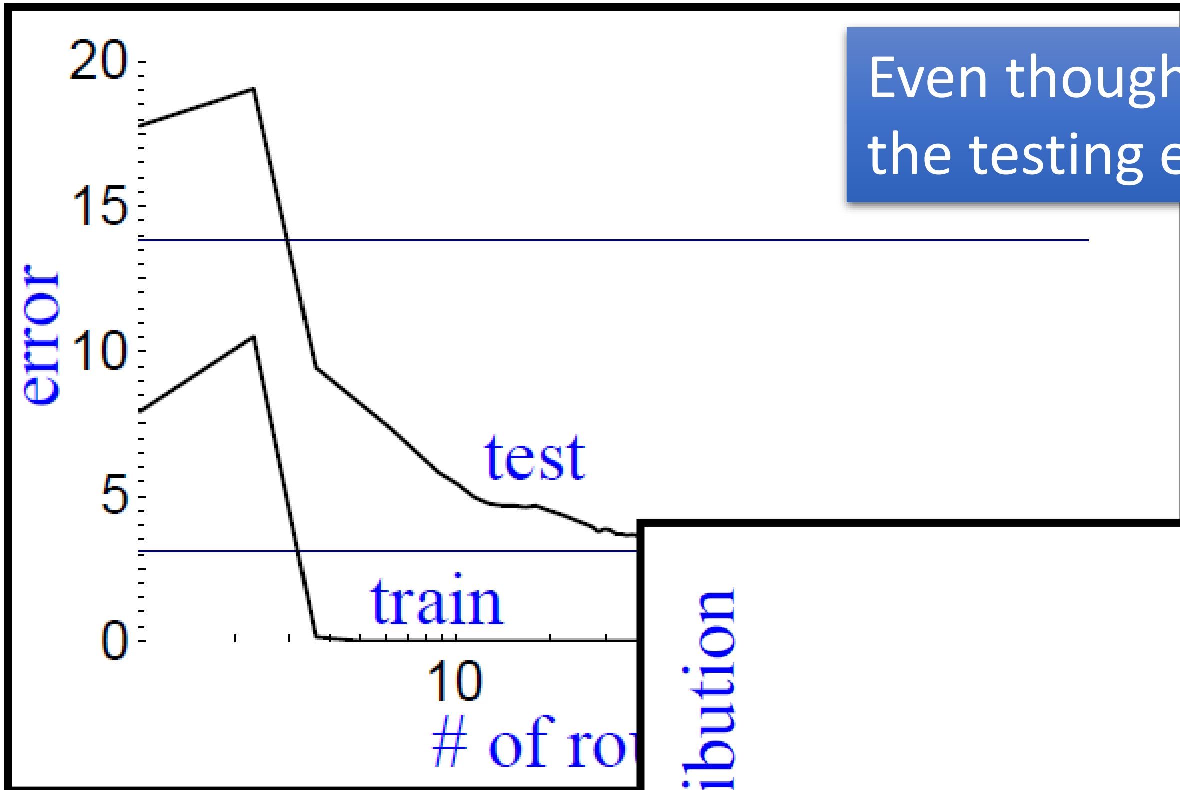
$$Z_{T+1} = N\prod_{t=1}^{T} 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

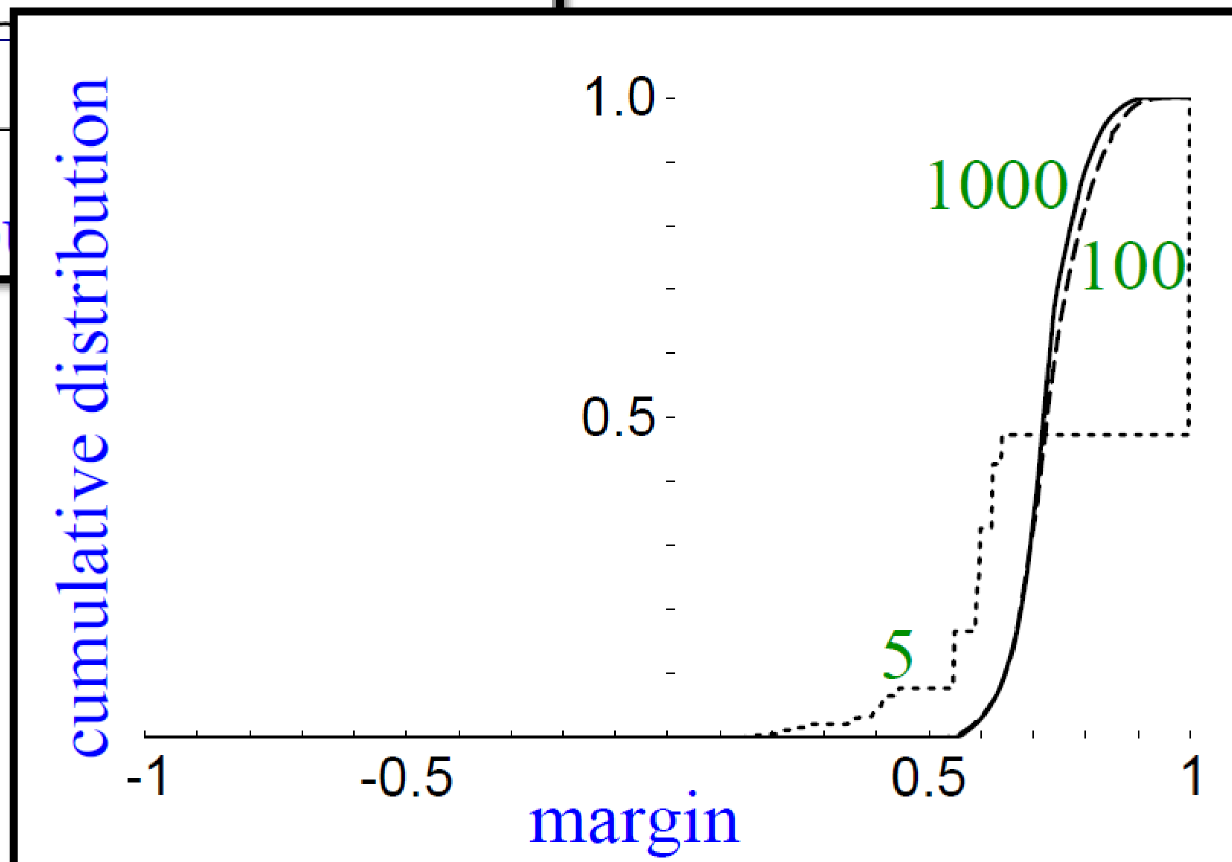
$$\text{Training Data Error Rate} \leq \prod_{t=1}^{T} \underbrace{2\sqrt{\epsilon_t(1-\epsilon_t)}}_{<1}$$

Smaller and smaller

End of Warning

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





$$H(x) = sign\underbrace{\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right)}_{g(x)}$$

Margin = $\hat{y} g(x)$

Large Margin?

$$H(x) = sign\Bigg(\underbrace{\sum_{t=1}^{T} \alpha_t f_t(x)}_{g(x)}\Bigg)$$

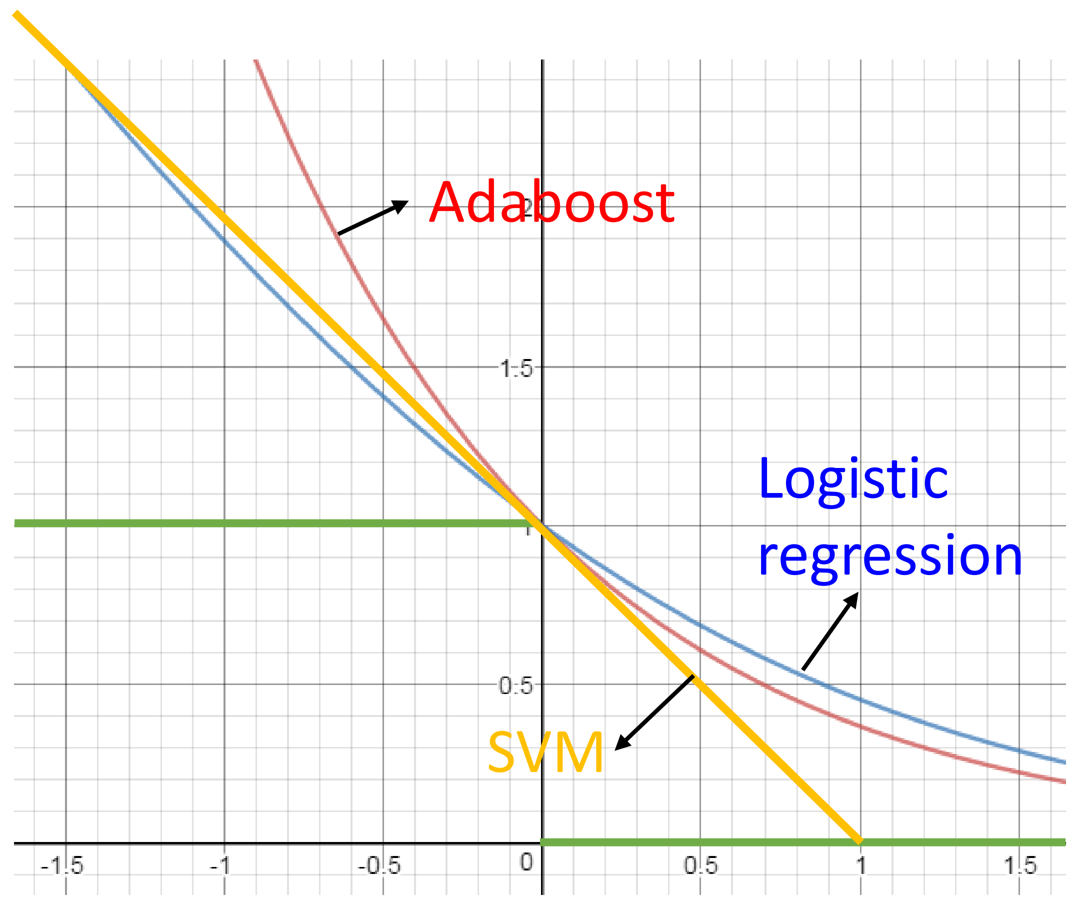
Training Data Error Rate =

$$= \frac{1}{N}\sum_{n} \delta\left(H(x^n) \neq \hat{y}^n\right)$$

$$\leq \frac{1}{N}\sum_{n} exp\left(-\hat{y}^n g(x^n)\right)$$

$$= \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Getting smaller and smaller as T increase

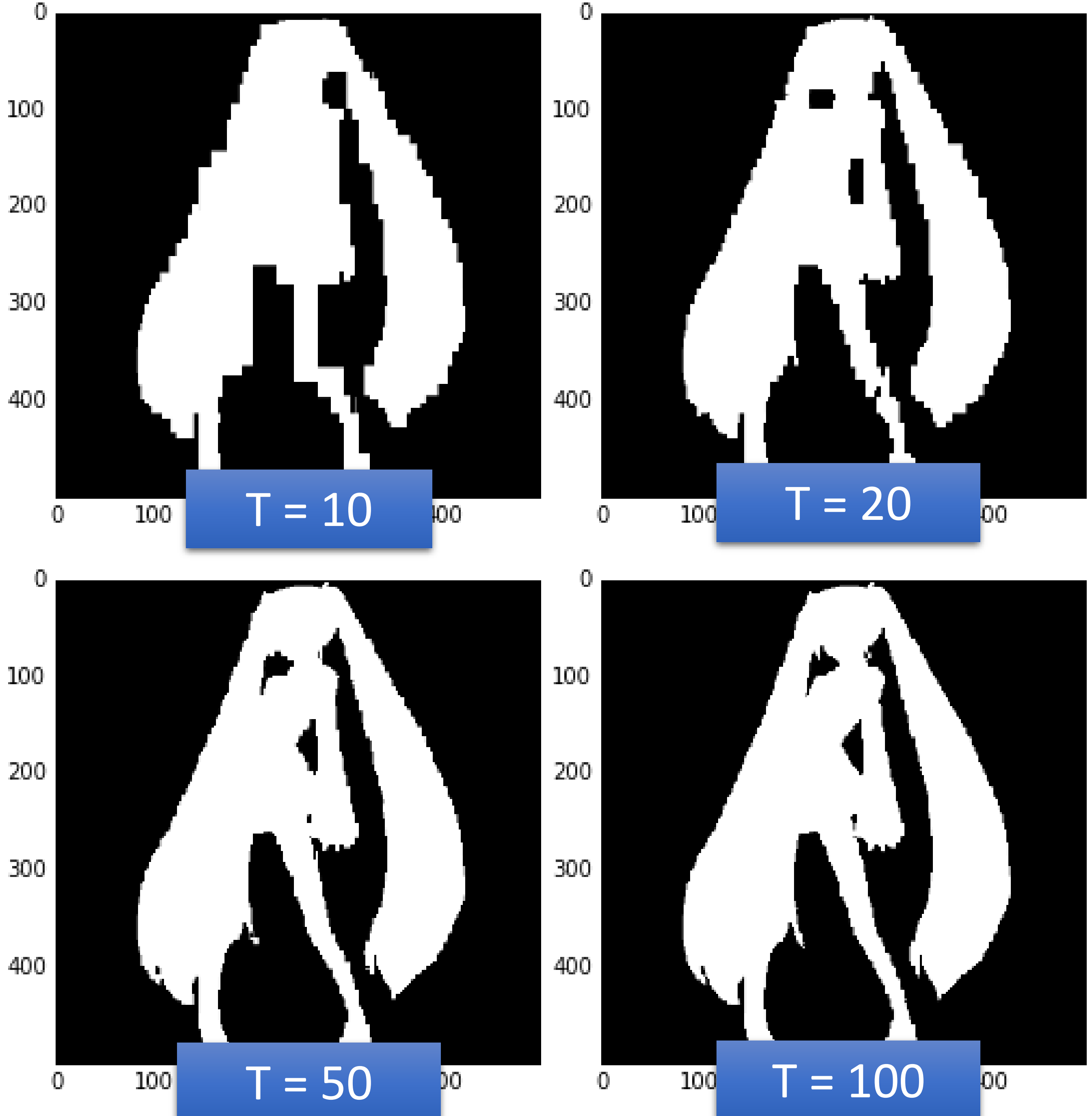


$$\hat{y}^n g(x^n)$$

Experiment: Function of Miku

Adaboost
+Decision Tree

(depth = 5)



To learn more …

- Introduction of Adaboost:
  - Freund; Schapire (1999). "A Short Introduction to Boosting“
- Multiclass/Regression
  - Y. Freund, R. Schapire, “A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting”, 1995.
  - Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In Proceedings of the Eleventh Annual Conference on Computational Learning Theory, pages 80–91, 1998.
- Gentle Boost
  - Schapire, Robert; Singer, Yoram (1999). "Improved Boosting Algorithms Using Confidence-rated Predictions".

General Formulation of Boosting

- Initial function $g_0(x) = 0$
- For t = 1 to T:
  - Find a function $f_t(x)$ and α_t to $\boxed{\text{improve}}$ $g_{t-1}(x)$
    - $g_{t-1}(x) = \sum_{i=1}^{t-1} \alpha_i f_i(x)$
  - $g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$
- Output: $H(x) = sign\big(g_T(x)\big)$

What is the learning target of $g(x)$?

Minimize $L(g) = \sum_n l\big(\hat{y}^n, g(x^n)\big) = \sum_n exp(-\hat{y}^n g(x^n))$

Gradient Boosting

- Find $\mathrm{g}(x)$, minimize $L(g) = \sum_n exp(-\hat{y}^n g(x^n))$
  - If we already have $\mathrm{g}(x) = g_{t-1}(x)$, how to update $\mathrm{g}(x)$?

Gradient Descent:

$$g_t(x) = g_{t-1}(x) \boxed{- \eta \left. \frac{\partial L(g)}{\partial g(x)} \right|_{\mathrm{g}(x) = g_{t-1}(x)}}$$

$$\sum_n exp(-\hat{y}^n g_{t-1}(x^n))(\quad \hat{y}^n)$$

Same direction

$$g_t(x) = g_{t-1}(x) \boxed{+ \alpha_t f_t(x)}$$

Gradient Boosting

$$f_t(x) \xleftrightarrow[\text{Same direction}]{\qquad\qquad} \sum_n exp(-\hat{y}^n g_t(x^n))(\hat{y}^n)$$

We want to find $f_t(x)$ maximizing

$$\sum_n \underbrace{exp(-\hat{y}^n g_{t-1}(x^n))}_{\text{example weight } u_t^n} \overset{\text{Minimize Error}}{\underset{\text{Same sign}}{\boxed{(\hat{y}^n) f_t(x^n)}}}$$

$$u_t^n = exp(-\hat{y}^n g_{t-1}(x^n)) = exp\left(-\hat{y}^n \sum_{i=1}^{t-1} \alpha_i f_i(x^n)\right)$$

$$= \prod_{i=1}^{t-1} exp(-\hat{y}^n \alpha_i f_i(x^n))$$

Exactly the weights we obtain in Adaboost

Gradient Boosting

- Find g(x), minimize $L(g) = \sum_n exp(-\hat{y}^n g(x^n)\ )$

$$g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$$

α_t is something like learning rate

Find α_t minimzing $L(g_{t+1})$

$$L(g) = \sum_n exp\left(-\hat{y}^n\left(g_{t-1}(x) + \alpha_t f_t(x)\right)\ \right)$$

$$= \sum_n exp\left(-\hat{y}^n g_{t-1}(x)\right) exp\left(-\hat{y}^n \alpha_t f_t(x)\right)$$

$$= \sum_{\hat{y}^n \neq f_t(x)} exp\left(-\hat{y}^n g_{t-1}(x^n)\right) exp(\alpha_t)$$

$$+ \sum_{\hat{y}^n = f_t(x)} exp\left(-\hat{y}^n g_{t-1}(x^n)\right) exp(-\alpha_t)$$

Find α_t such that

$$\frac{\partial L(g)}{\partial \alpha_t} = 0$$

$\alpha_t =$

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

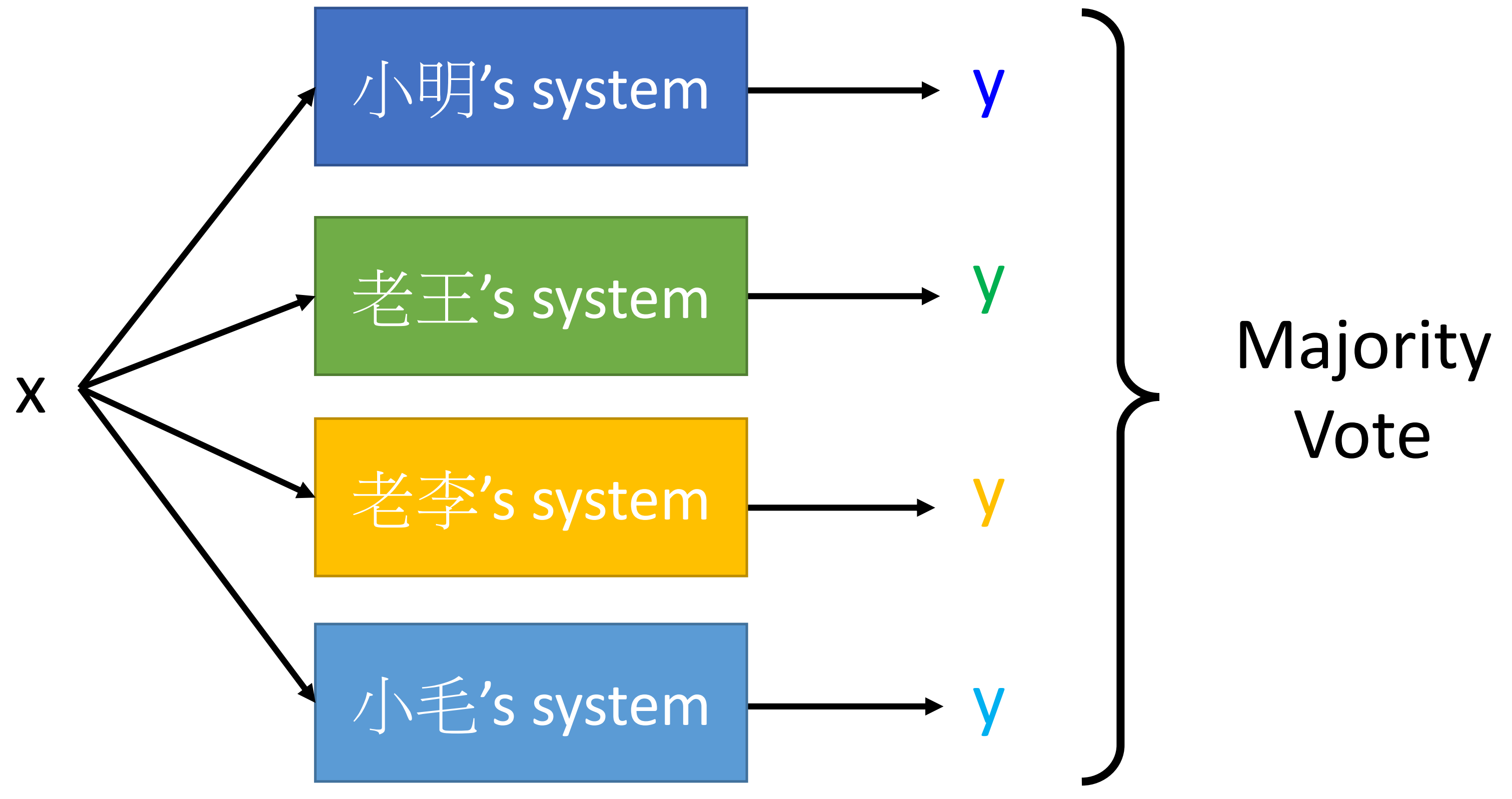
Adaboost!

Cool Demo

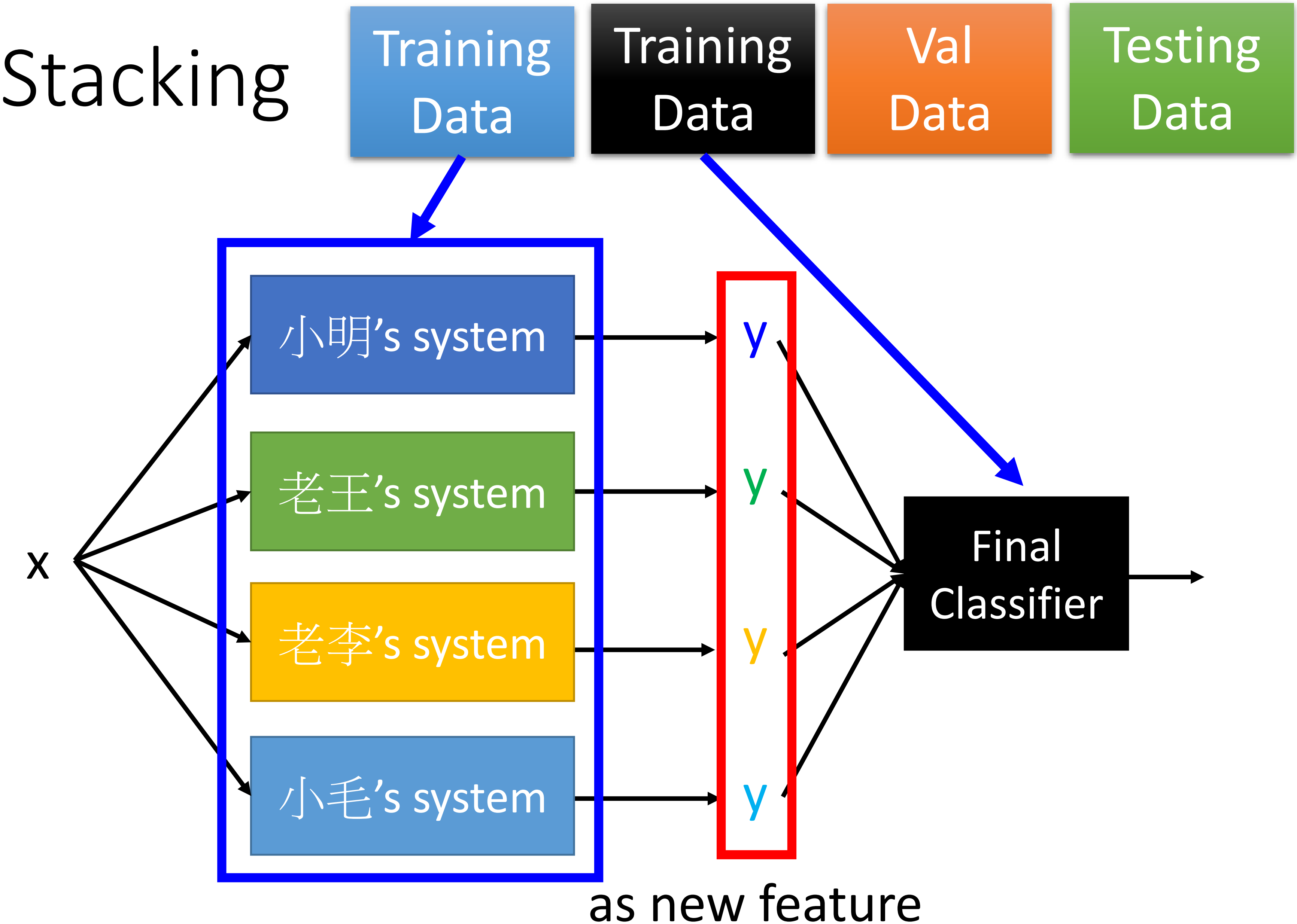
- http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html

Ensemble: Stacking

Voting


x
小明's system
老王's system
老李's system
小毛's system
y
y
y
y
Majority Vote

Stacking


Training Data
Training Data
Val Data
Testing Data
x
小明's system
老王's system
老李's system
小毛's system
y
y
y
y
Final Classifier
as new feature

2017

新年快樂

Happy New Year