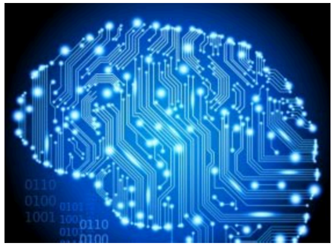
Lecture 16: Learning 5

http://cs.nju.edu.cn/yuy/course_ai17.ashx



How can we improve an algorithm

for free

one classifier with error 0.49

three independent classifiers each with error 0.49

two out of three are wrong: 0.367353
three of them are wrong: 0.117649
majority of the three are wrong: 0.485002

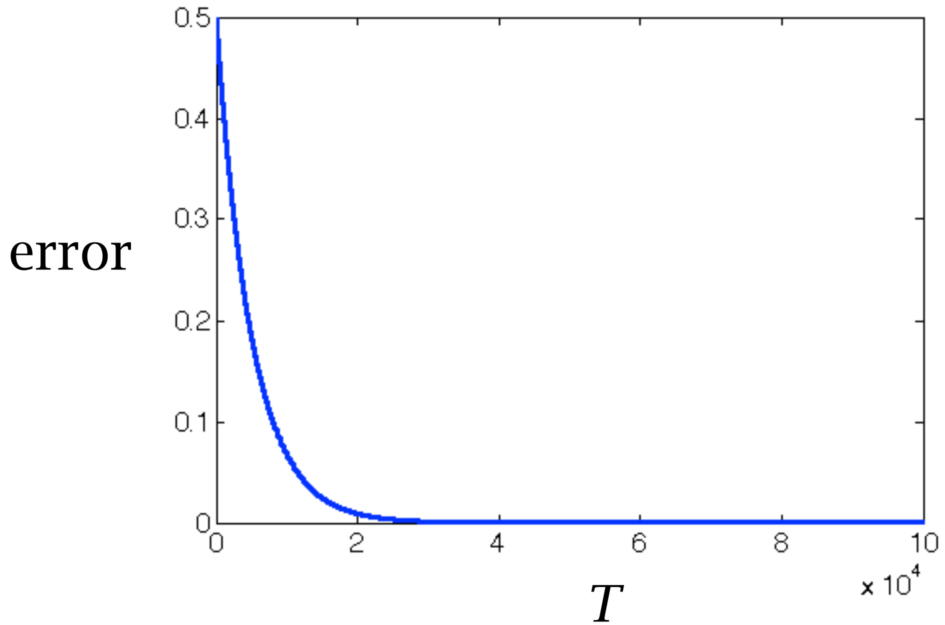
Motivation theories

for binary classification, what if the classifiers give *independent* output and are little bit better than random guess?

each classifier has error 0.49
error of combining T classifiers:

$$\sum_{t=\lceil T/2 \rceil}^{T} \binom{T}{t} \cdot 0.49^t \cdot 0.51^{T-t}$$

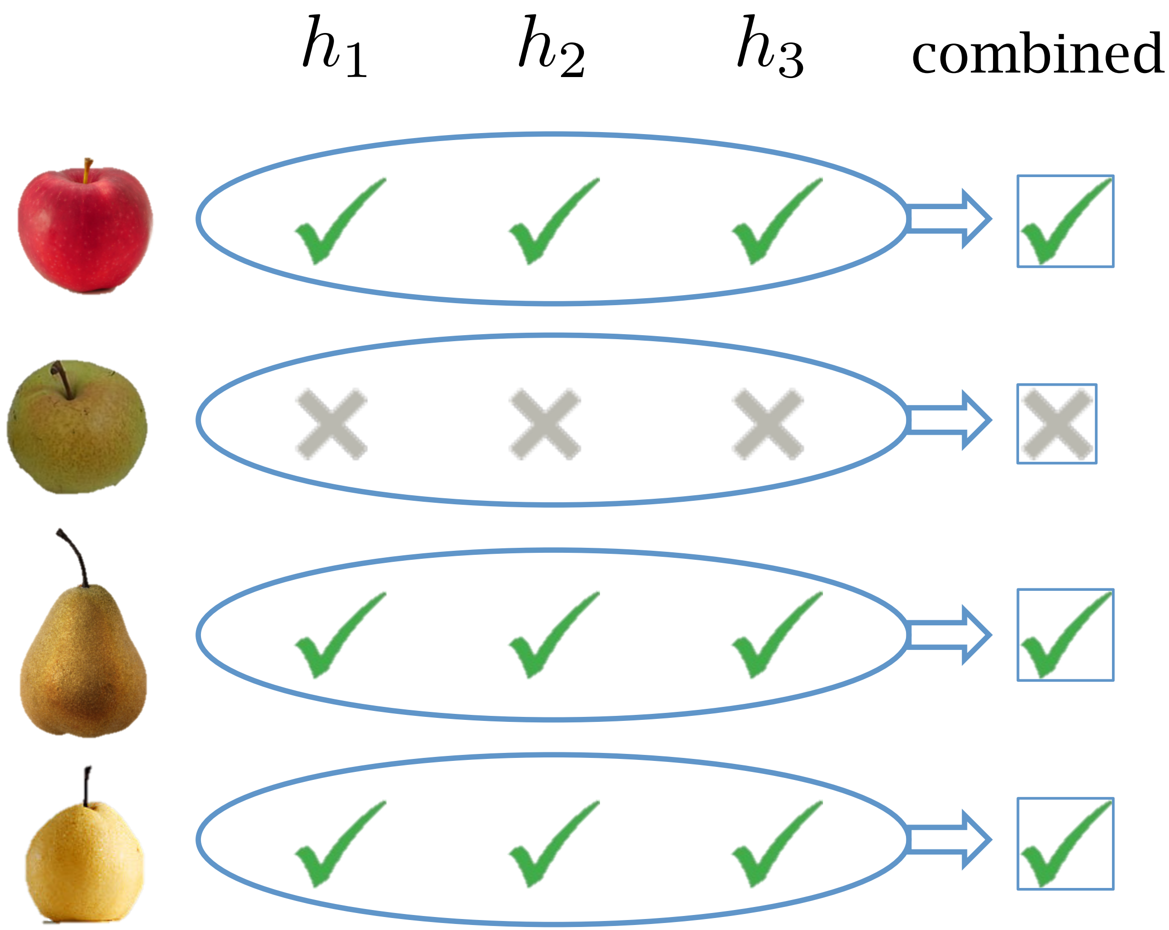
$$\leq \frac{1}{2} e^{-2T(0.5-0.49)^2}$$



but independent classifiers are not achievable

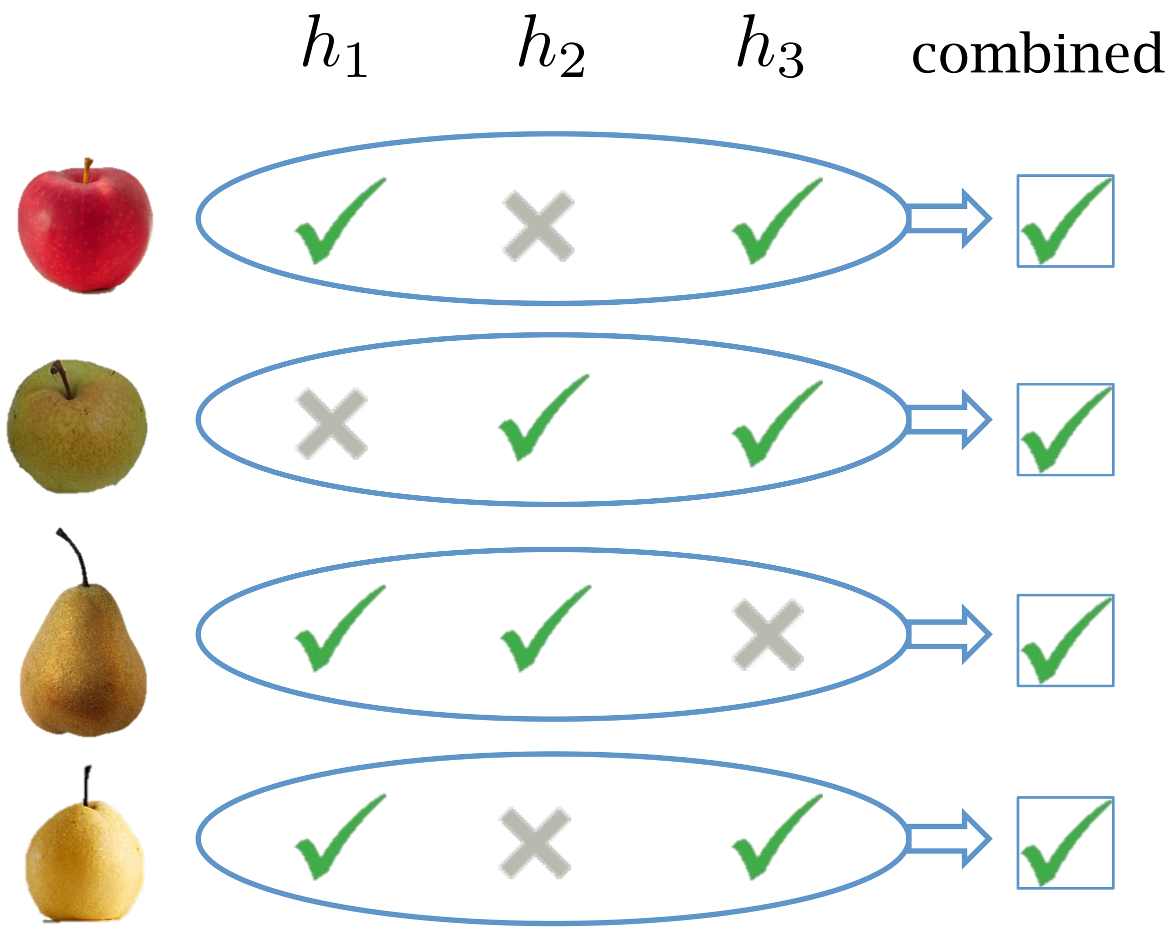
The importance of diversity

not useful to combine identical base learners



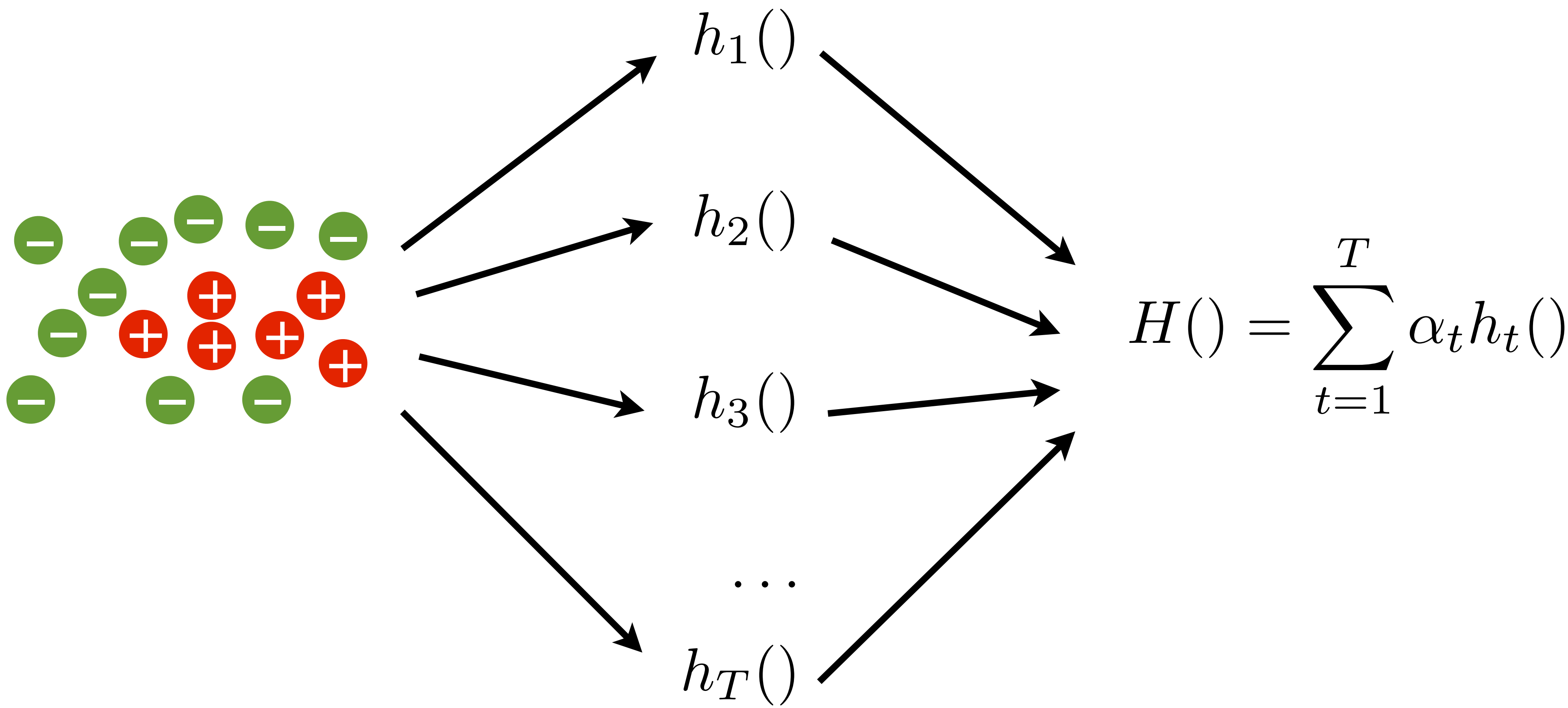
The importance of diversity

good to combine different learners



Ensemble learning

combination of multiple classifiers/regressors



base learner combined learner

Ensemble methods

Parallel ensemble

create diverse base learners by introducing randomness

Sequential ensemble

create base learners by complementarity

Parallel ensemble methods

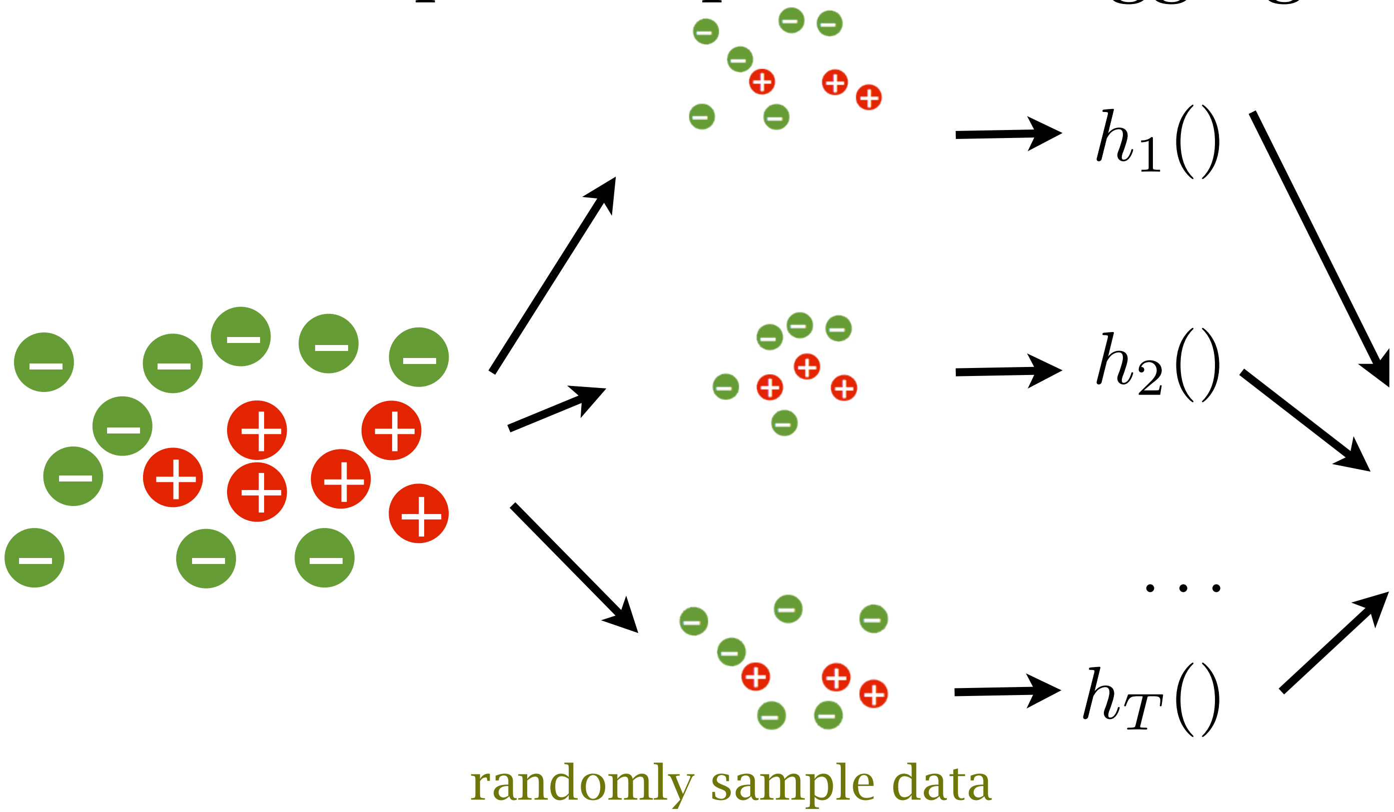
Diversity generating categories:

- Data Sample Manipulation
  - bootstrap sampling/Bagging
- Input Feature Manipulation
  - random subspace
- Output Representation Manipulation
  - flipping output/output smearing
- Learning Parameter Manipulation
  - random initialization
  - Random Forests

combine two or more categories

Parallel ensemble methods

Data Sample Manipulation: Bagging



$$H(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T} h_t(\boldsymbol{x}) \quad \text{(regression)}$$

$$\arg\max_{y} \sum_{t=1}^{T} I(h_t(\boldsymbol{x}) = y) \quad \text{(classification)}$$

Base classifiers should be sensitive to sampling

» decision tree, neural network are good

» NB, linear classifier are not

Good for handling large data set

Parallel ensemble methods

Data Sample Manipulation: Bagging

Input: D: Data set $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
$\mathfrak{L}$: Base learning algorithm;
T: Number of base learners.

Process:
1. **for** $t = 1, \ldots, T$:
2. $\quad h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$ % $\mathcal{D}_{bs}$ is the bootstrap distribution
3. **end**

Output: $H(\boldsymbol{x}) = \max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$

---

sample with replacement

Base classifiers should be sensitive to sampling

» decision tree, neural network are good

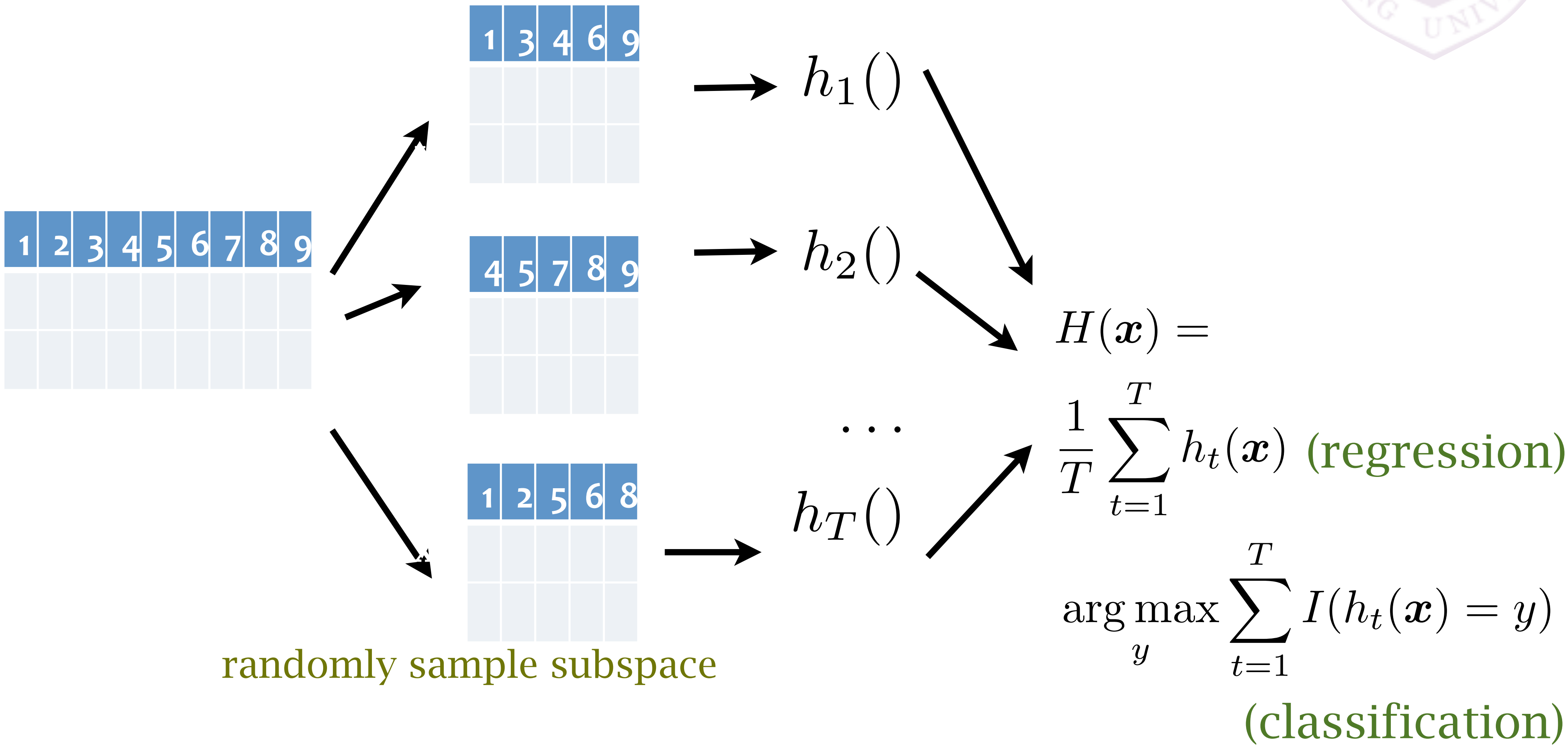
» NB, linear classifier are not

Good for handling large data set

Leo Breiman
1928-2005

Parallel ensemble methods

Input Feature Manipulation: Random subspace



Data should be rich in features
Good for handling high dimensional data

Parallel ensemble methods

Input Feature Manipulation: Random subspace

Input: D: Data set $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
$\mathfrak{L}$: Base learning algorithm;
T: Number of base learners;
d: Dimension of subspaces.

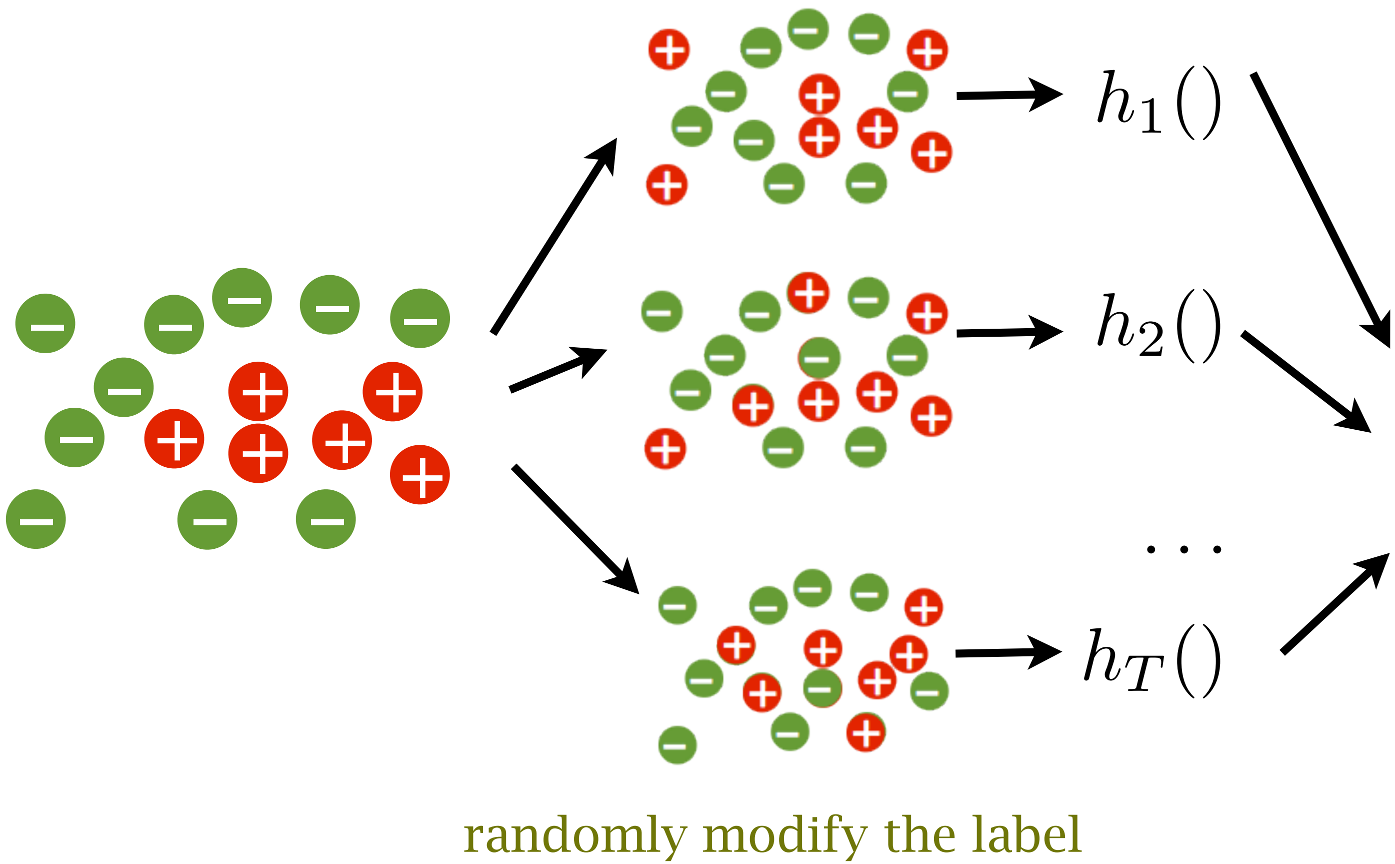
Process:
1. **for** $t = 1, \ldots, T$:
2. $\mathcal{F}_t = RS(D, d)$ % $\mathcal{F}_t$ is a set of d randomly selected features;
3. $D_t = Map_{\mathcal{F}_t}(D)$ % D_t keeps only the features in $\mathcal{F}_t$
4. $h_t = \mathfrak{L}(D_t)$ % Train a learner
5. **end**

Output: $H(\boldsymbol{x}) = \max\limits_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}\left(h_t\left(Map_{\mathcal{F}_t}\left(\boldsymbol{x}\right)\right) = y\right)$

Data should be rich in features
Good for handling high dimensional data

Parallel ensemble methods

Output Representation Manipulation: Output flipping



$$H(\boldsymbol{x}) =$$

$$\frac{1}{T}\sum_{t=1}^{T} h_t(\boldsymbol{x}) \quad \text{(regression)}$$

$$\arg\max_{y} \sum_{t=1}^{T} I(h_t(\boldsymbol{x}) = y)$$

(classification)

May drastically reduce the accuracy of base learners

Parallel ensemble methods

Learning Parameter Manipulation: Random forest

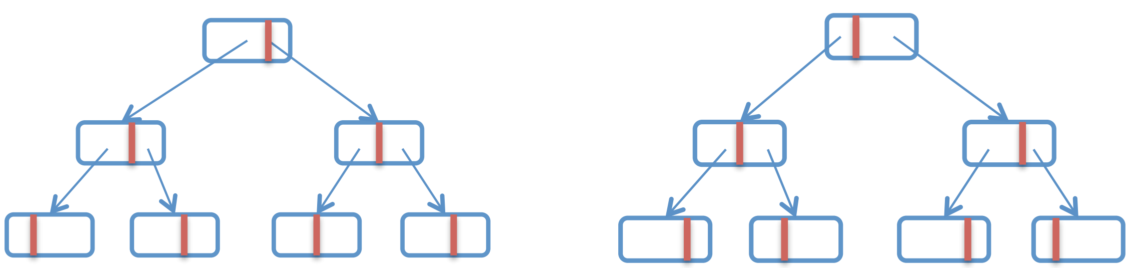
Randomized decision tree

at each node

1. randomly select a subset of features
2. use select a feature (and split point) from the subset to split the data

decision tree: select the best split from ALL features/splits

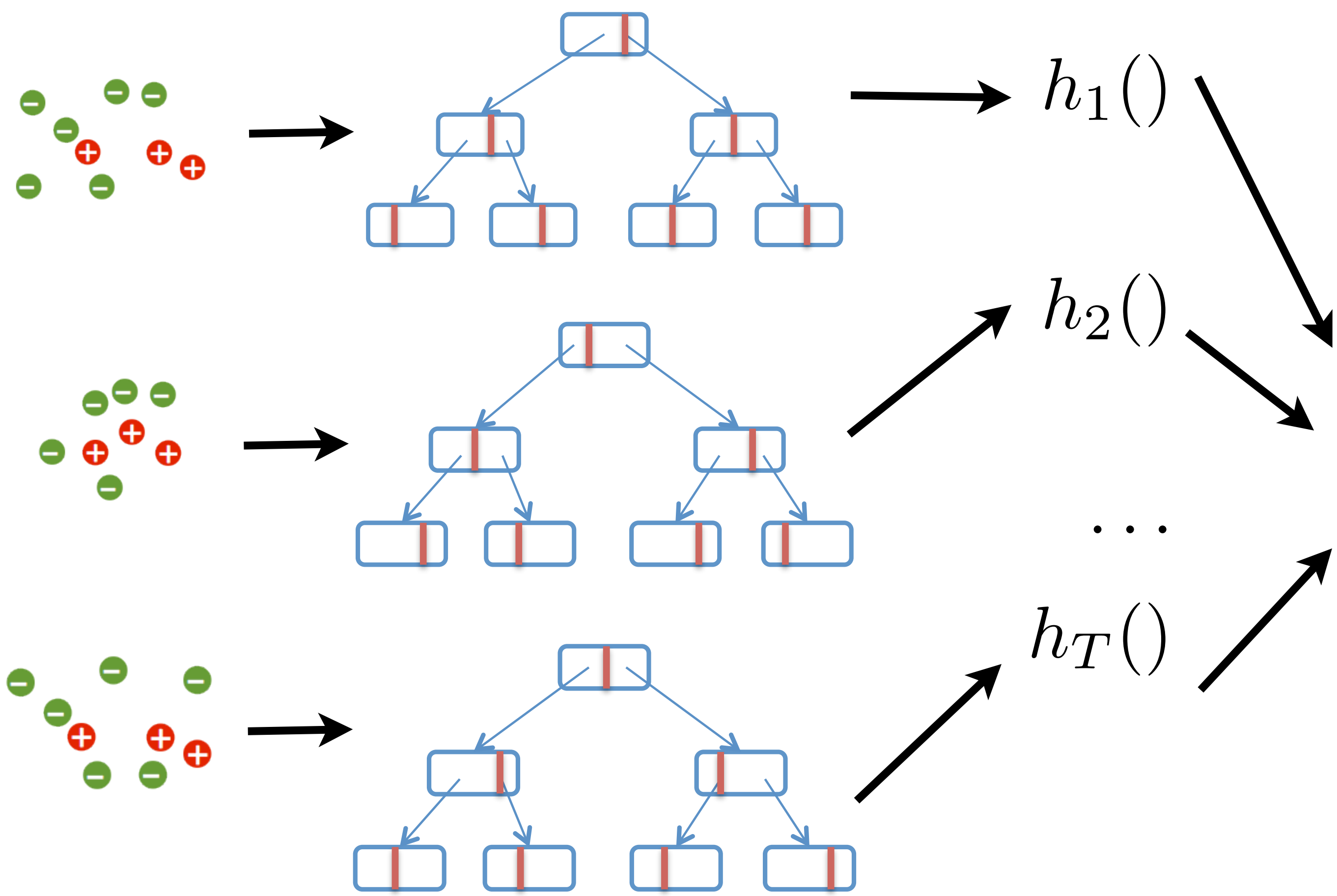
(other variants are available)



every run produce a different tree

Parallel ensemble methods

Learning Parameter Manipulation: Random forest



$$H(\boldsymbol{x}) =$$

$$\frac{1}{T}\sum_{t=1}^{T} h_t(\boldsymbol{x})$$ (regression)

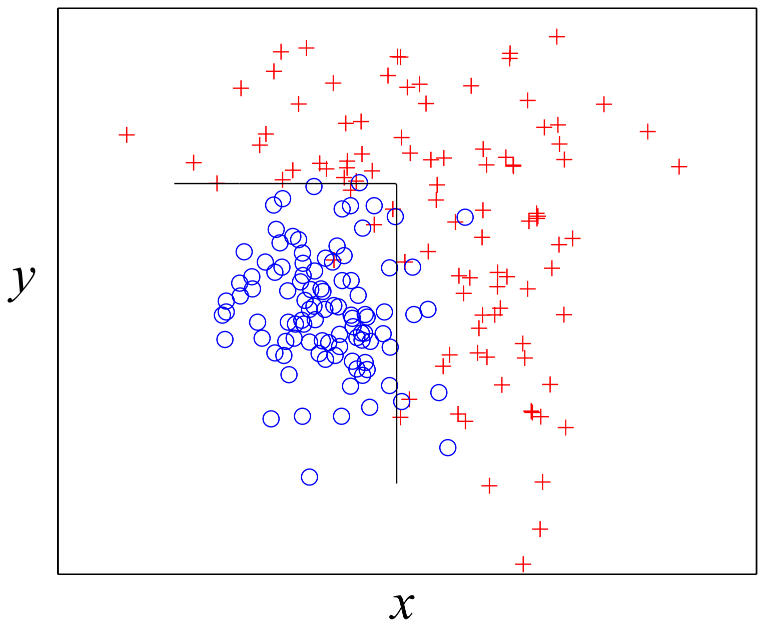
$$\arg\max_y \sum_{t=1}^{T} I(h_t(\boldsymbol{x}) = y)$$

(classification)

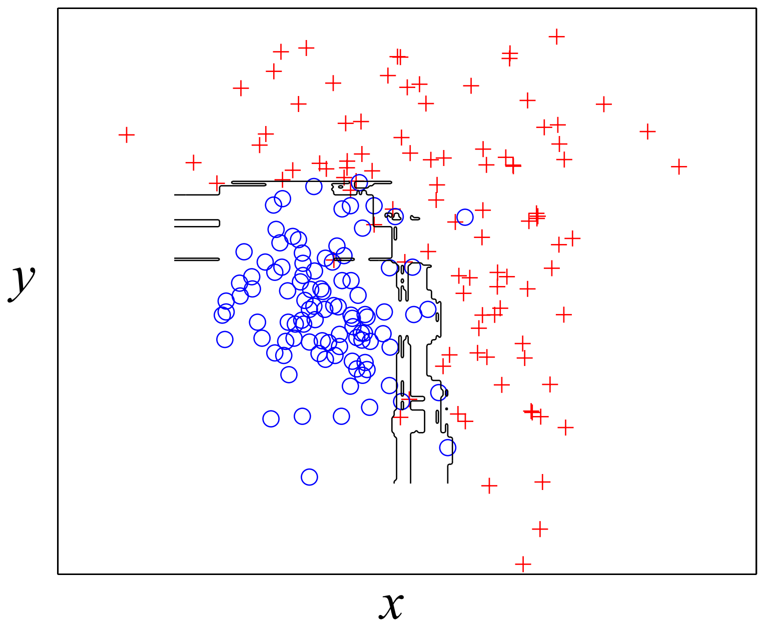
Bagging of randomized decision tree

Parallel ensemble methods

Random forest



decision boundary of
single decision tree



decision boundary of
random forest

Parallel ensemble methods

Diversity generating categories:

- Data Sample Manipulation
  - bootstrap sampling/Bagging
- Input Feature Manipulation
  - random subspace
- Output Representation Manipulation
  - flipping output/output smearing
- Learning Parameter Manipulation
  - random initialization
  - Random Forests

obtain diversity by randomization

Parallel ensemble methods

Simple combination:

$$\frac{1}{T}\sum_{t=1}^{T} h_t(\boldsymbol{x}) \qquad \text{(simple average for regression)}$$

$$\arg\max_{y} \sum_{t=1}^{T} I(h_t(\boldsymbol{x}) = y) \qquad \text{(majority vote for classification)}$$

Parallel ensemble methods

model-weighted combination:

better model has higher weight

$$\frac{1}{T}\sum_{t=1}^{T} w_t h_t(\boldsymbol{x})$$ (simple average for regression)

$$\arg\max_y \sum_{t=1}^{T} w_t I(h_t(\boldsymbol{x}) = y)$$ (majority vote for classification)

Parallel ensemble methods

instance-weighted combination:

weight by the confidence of the model

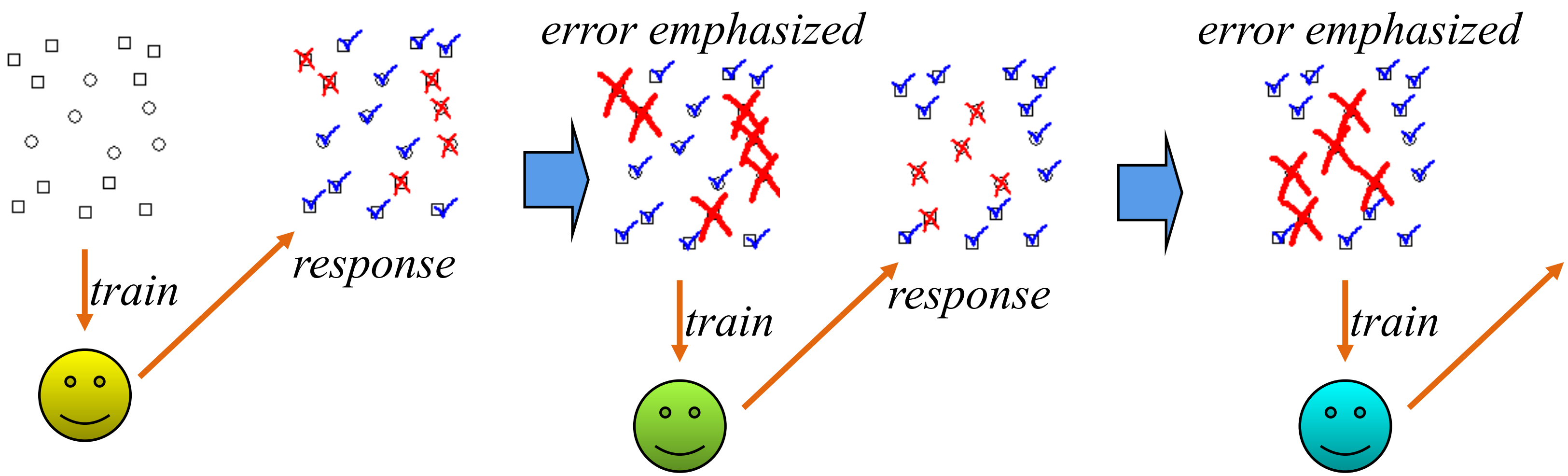
decision tree: the purity of the leave node

$$\frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{x})h_t(\boldsymbol{x})$$ (simple average for regression)

$$\arg\max_y \sum_{t=1}^{T} w_t(\boldsymbol{x})I(h_t(\boldsymbol{x}) = y)$$ (majority vote for classification)

Sequential ensemble methods

Generate learners sequentially,
focus on previous errors



so that the combination of learners will have
a high accuracy

AdaBoost

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:

1. $\mathcal{D}_1(\boldsymbol{x}) = 1/m$. % Initialize the weight distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(D, \mathcal{D}_t)$; % Train a classifier h_t from D under distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. **if** $\epsilon_t > 0.5$ **then break**
6. $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$; % Determine the weight of h_t
7. $$\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$$
   $$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$$
   % Update the distribution, where
   % Z_t is a normalization factor which
   % enables $\mathcal{D}_{t+1}$ to be a distribution
8. **end**

Output: $H(\boldsymbol{x}) = \texttt{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) \right)$

AdaBoost

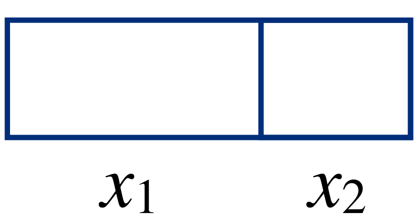
About the distribution:

$$\mathcal{D}_1(\boldsymbol{x}) = 1/m.$$

maintain a array to record the distribution

$h_t = \mathfrak{L}(D, \mathcal{D}_t)$; % Train a classifier h_t from D under distribution $\mathcal{D}_t$
$\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t

sample a training set according to the distribution



if random < 0.7, get an x1
else get an x2

AdaBoost

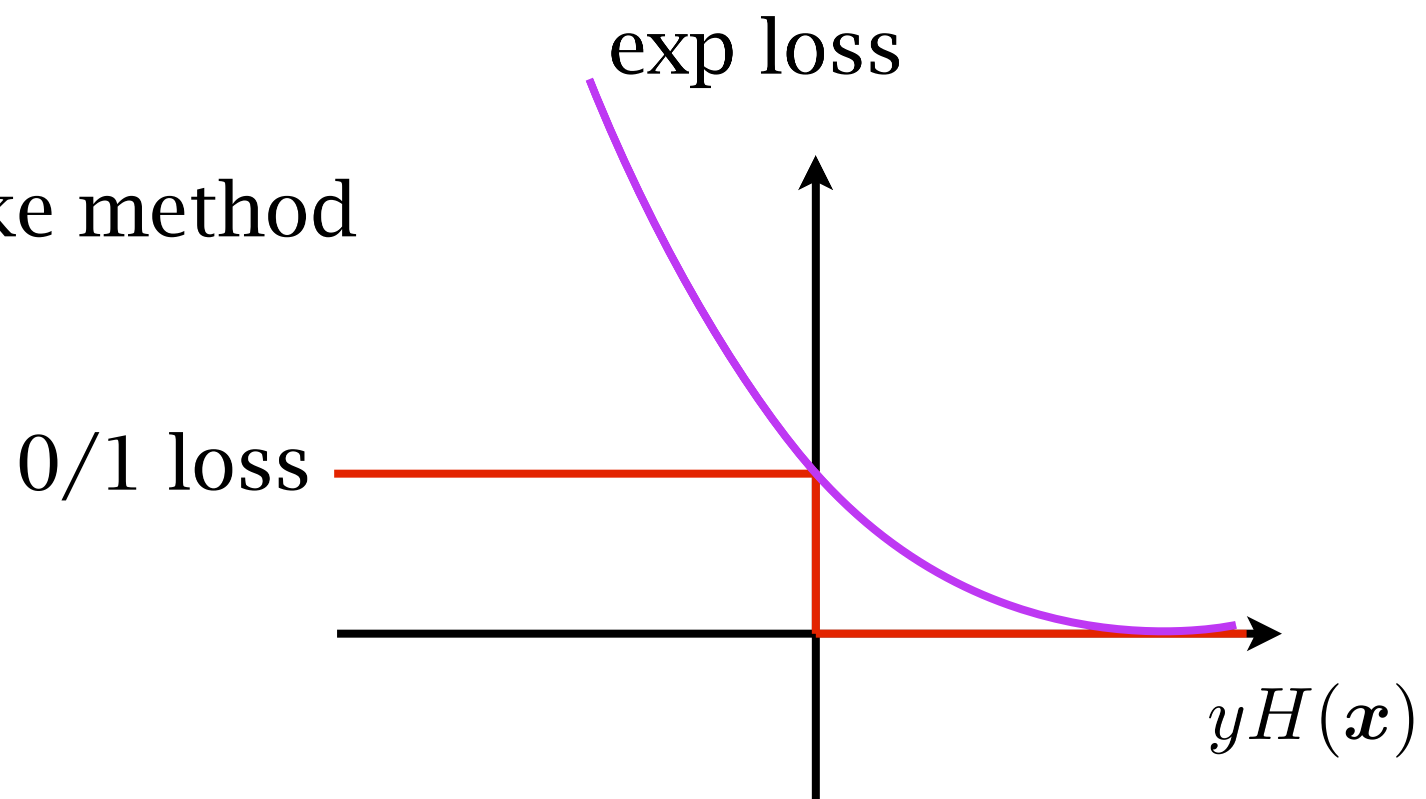
fit an additive model, sequentially

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$$

to minimize exponential loss

$$\min e^{-yH(\boldsymbol{x})}$$

by Newton-like method



Gradient boosting

fit an additive model, sequentially

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$$

to minimize any loss by gradient decent

Gradient boosting

example: least square regression

$$\min \frac{1}{m}\sum_{i=1}^{m}(H(\boldsymbol{x}_i) - y_i)^2$$

1. fit the first base regressor

$$\min \frac{1}{m}\sum_{i=1}^{m}(h_1(\boldsymbol{x}_i) - y_i)^2$$

then how to train the second base regressor ?

$$\min \frac{1}{m}\sum_{i=1}^{m}(h_1(\boldsymbol{x}_i)+h_2(\boldsymbol{x}_i) - y_i)^2$$

gradient descent *in function space*

Gradient boosting

$$\min \frac{1}{m}\sum_{i=1}^{m}(h_1(\boldsymbol{x}_i)+h_2(\boldsymbol{x}_i)-y_i)^2$$

gradient descent *in function space*

$$h_{\text{new}} \leftarrow -\frac{\partial (H-f)^2}{\partial H} = -2(H-f)$$

this function is not directly operable

operate through data

$$\forall \boldsymbol{x}_i : \hat{y}_i = -2(H(\boldsymbol{x}_i)-y_i)$$

fit h_2 point-wisely

$$h_{\text{new}} = \arg\min_h \frac{1}{m}\sum_{i=1}^{m}(h(\boldsymbol{x}_i)-\hat{y}_i)^2$$

Gradient boosting

Gradient boosting (for least square regression)

1. $h_0 = 0, H_0 = h_0$
2. For $t = 1$ to T
3. let $\forall \boldsymbol{x}_i : y_i = -2(H_{t-1}(\boldsymbol{x}_i) - y_i)$
4. solve $h_t = \arg\min_h \frac{1}{m}\sum_{i=1}^{m}(h(\boldsymbol{x}_i) - y_i)^2$

   (by some least square regression algorithm)
5. $H_t = H_{t-1} + \eta h_t$ (usually set $\eta = 0.01$)
6. next for

Output $H_T = \sum_{t=1}^{T} h_t$

Gradient boosting

Gradient boosting (for classification)

0-1 loss

$$\min I(yH(\boldsymbol{x}) \leq 0)$$

logistic regression

$$\min \log(1 + e^{-yH(\boldsymbol{x})})$$

perceptron

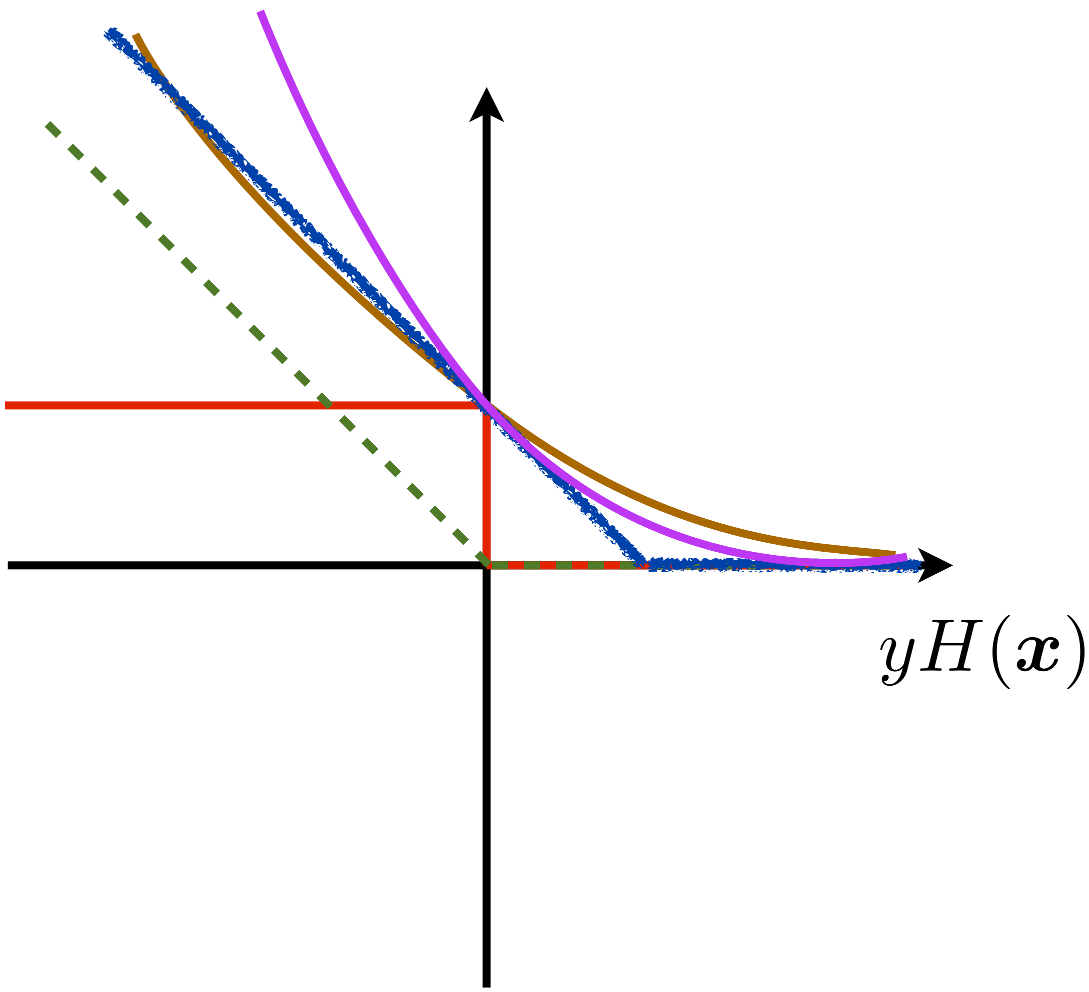
$$\min \max\{-yH(\boldsymbol{x}), 0\}$$

hinge loss

$$\min \max\{1 - yH(\boldsymbol{x}), 0\}$$

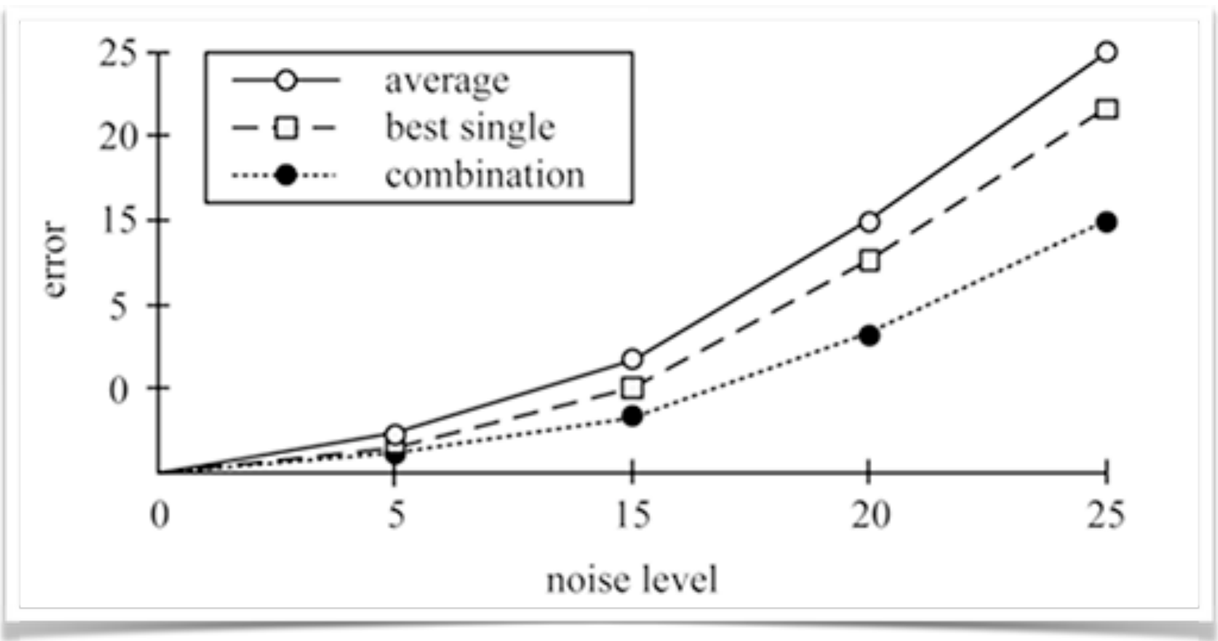
exponential loss

$$\min e^{-yH(\boldsymbol{x})}$$



More about ensemble

Hansen and Salamon [PAMI'90] reported an observation that combination of multiple BP-NN is better than the best single BP-NN



More about ensemble

for regression task:

mean error of base regressors

$$\frac{1}{T}\sum_t (h_t - f)^2$$

$$= \frac{1}{T}\sum_t (h_t - H + H - f)^2$$

$$= \frac{1}{T}\sum_t (h_t - H)^2 + \frac{1}{T}\sum_t (H - f)^2 - 2\frac{1}{T}\sum_t (h_t - H)(H - f)$$

$$= \frac{1}{T}\sum_t (h_t - H)^2 + (H - f)^2$$

error of combined regressor

mean difference to the combined regressor

error of ensemble =
mean error of base regressors
– mean difference base regressors to the ensemble

accurate and *diverse*

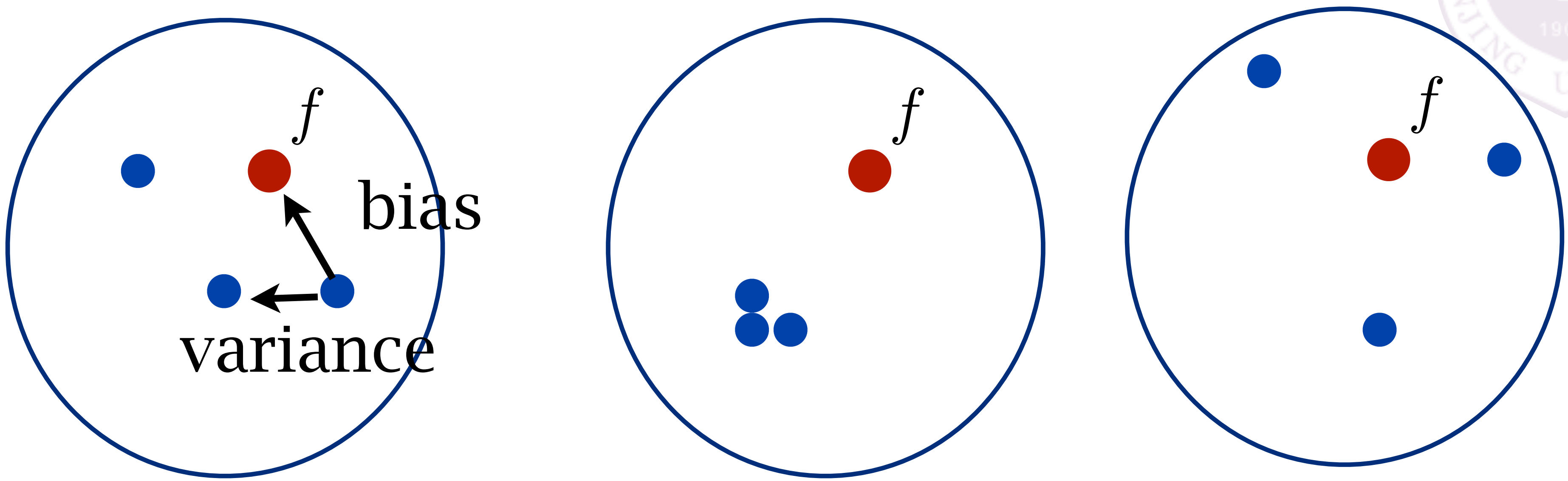
More about ensemble

for classification task:

$$err_g(f) \leq err_S^{\theta}(f) + \frac{C}{\sqrt{m}} \left(\frac{\ln n \ln \left(m \sqrt{1/n + (1 - 1/n)(1 - q)} \right)}{\theta^2} + \ln \frac{1}{\delta} \right)^{1/2}$$

pairwise diversity

Bias-variance analysis



low variance, high bias

low bias, high variance

parallel ensemble: reduce variance

use unpruned decision trees

sequential ensemble: reduce bias and variance

More about ensemble

Boosting:

is weak learnable class equals strong learnable class?

L. Valiant
Turing Award 2010

yes! The proof is the boosting algorithm

R. Schapire

AdaBoost

(Gödel Prize 2003)

AdaBoost is the first practical boosting algorithm

Applications

KDDCup: data mining competition organized by ACM SIGKDD

KDDCup 2009: to estimate the churn, appetency and up-selling probability of customers.

An Ensemble of Three Classifiers for KDD Cup 2009: Expanded Linear Model, Heterogeneous Boosting, and Selective Naïve Bayes

Hung-Yi Lo, Kai-Wei Chang, Shang-Tse Chen, Tsung-Hsien Chiang, Chun-Sung Ferng, Cho-Jui Hsieh, Yi-Kuang Ko, Tsung-Ting Kuo, Hung-Che Lai, Ken-Yi Lin, Chia-Hsuan Wang, Hsiang-Fu Yu, Chih-Jen Lin, Hsuan-Tien Lin, Shou-de Lin {D96023, B92084, B95100, B93009, B95108, B92085, B93038, D97944007, R97028, R97117, B94B02009, B93107, CJLIN, HTLIN, SDLIN}@CSIE.NTU.EDU.TW
Department of Computer Science and Information Engineering, National Taiwan University
Taipei 106, Taiwan

KDDCup 2010: to predict student performance on mathematical problems from logs of student interaction with Intelligent Tutoring Systems.

JMLR: Workshop and Conference Proceedings 1: 1-16 KDD Cup 2010

Feature Engineering and Classifier Ensemble for KDD Cup 2010

Hsiang-Fu Yu, Hung-Yi Lo, Hsun-Ping Hsieh, Jing-Kai Lou, Todd G. McKenzie, Jung-Wei Chou, Po-Han Chung, Chia-Hua Ho, Chun-Fu Chang, Yin-Hsuan Wei, Jui-Yu Weng, En-Syu Yan, Che-Wei Chang, Tsung-Ting Kuo, Yi-Chen Lo, Po Tzu Chang, Chieh Po, Chien-Yuan Wang, Yi-Hung Huang, Chen-Wei Hung, Yu-Xun Ruan, Yu-Shi Lin, Shou-de Lin, Hsuan-Tien Lin, Chih-Jen Lin
Department of Computer Science and Information Engineering, National Taiwan University
Taipei 106, Taiwan

KDDCup 2011, KDDCup 2012, and foreseeably, 2013, 2014 ...

Applications

Netflix Price: if one participating team improves Netflix's own movie recommendation algorithm by 10% accuracy, they would win the grand prize of $1,000,000.