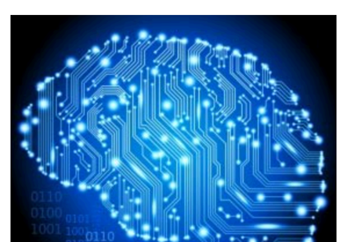
# Lecture 17: Learning 5

http://cs.nju.edu.cn/yuy/course_ai15.ashx



# Previously...

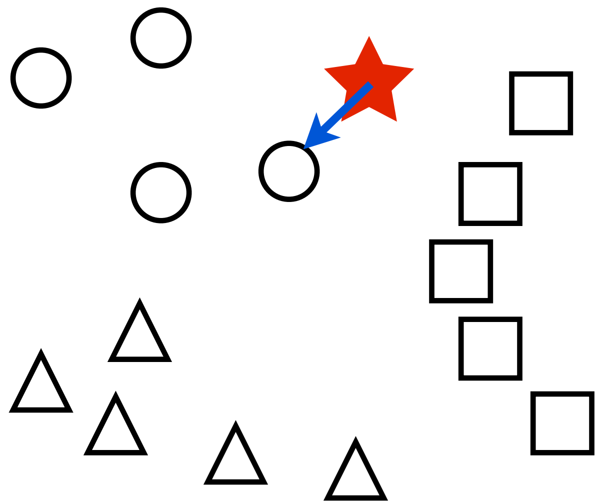
Learning

- Decision tree learning
- Neural networks
- Why we can learn
- Linear models

# Nearest Neighbor Classifier

# Nearest neighbor

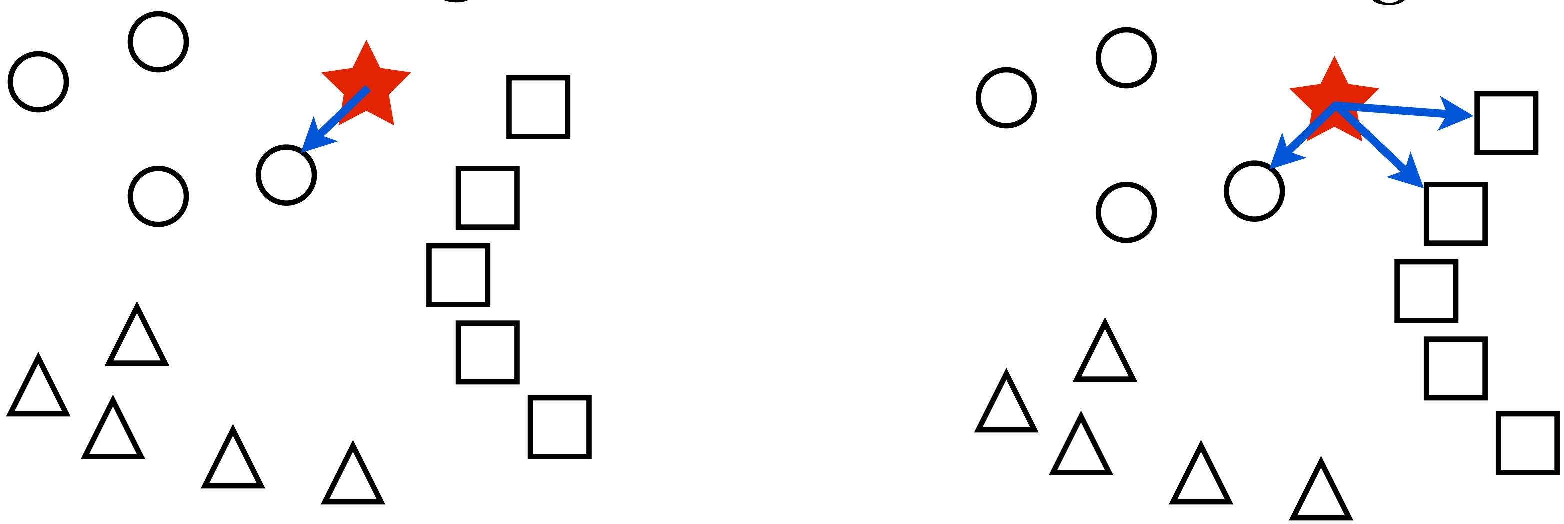
what looks similar are similar



# Nearest neighbor

for classification:

1-nearest neighbor: *k*-nearest neighbor:

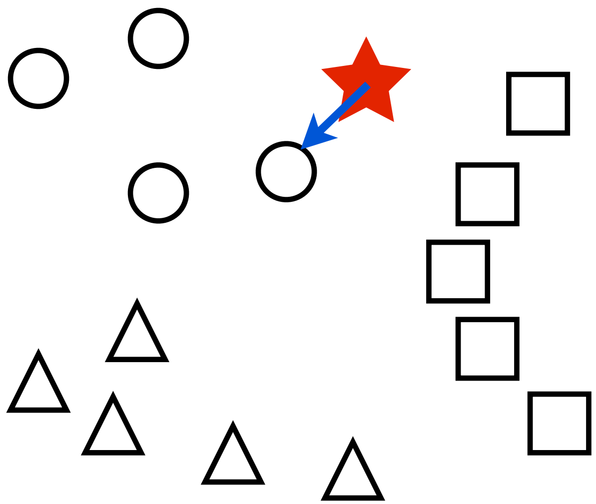


Predict the label as that of the NN
or the (weighted) majority of the k-NN

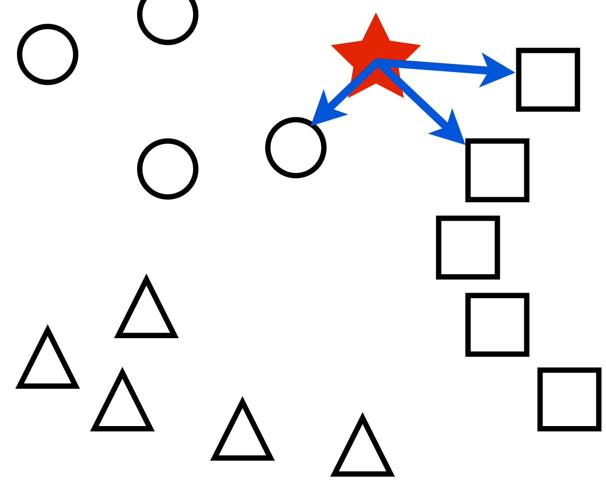
# Nearest neighbor

for regression:

1-nearest neighbor:



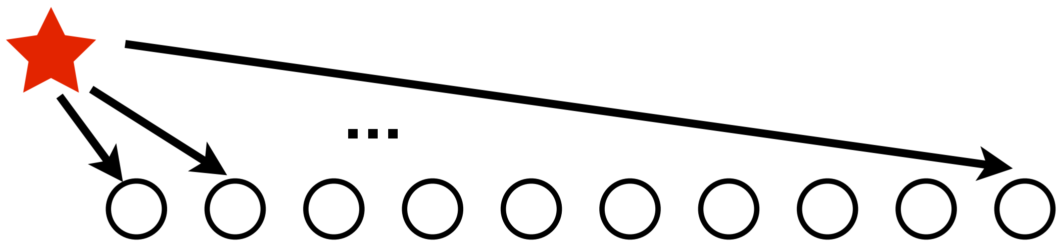
*k*-nearest neighbor:



Predict the label as that of the NN
or the (weighted) *average* of the k-NN

# Search for the nearest neighbor

Linear search




*n* times of distance calculations

$O(dn \ln k)$

*d* is the dimension, *n* is the number of samples

# Nearest neighbor classifier

- as classifier, asymptotically less than 2 times of the optimal Bayes error
- naturally handle multi-class
- no training time
- nonlinear decision boundary

- slow testing speed for a large training data set
- have to store the training data
- sensitive to similarity function

nonparametric method

# Naive Bayes Classifier

# Bayes rule

classification using posterior probability

for binary classification

$$f(x) = \begin{cases} +1, & P(y = +1 \mid \boldsymbol{x}) > P(y = -1 \mid \boldsymbol{x}) \\ -1, & P(y = +1 \mid \boldsymbol{x}) < P(y = -1 \mid \boldsymbol{x}) \\ \text{random}, & otherwise \end{cases}$$

in general

$$f(x) = \arg\max_y P(y \mid \boldsymbol{x})$$

# Bayes rule

classification using posterior probability

for binary classification

$$f(x) = \begin{cases} +1, & P(y=+1 \mid \boldsymbol{x}) > P(y=-1 \mid \boldsymbol{x}) \\ -1, & P(y=+1 \mid \boldsymbol{x}) < P(y=-1 \mid \boldsymbol{x}) \\ \text{random}, & otherwise \end{cases}$$

in general

$$\begin{aligned} f(x) &= \arg\max_y P(y \mid \boldsymbol{x}) \\ &= \arg\max_y P(\boldsymbol{x} \mid y) P(y) / P(\boldsymbol{x}) \\ &= \arg\max_y P(\boldsymbol{x} \mid y) P(y) \end{aligned}$$

how the probabilities be estimated

# Naive Bayes

$$f(x) = \arg\max_{y} P(\boldsymbol{x} \mid y) P(y)$$

estimation the a priori by frequency:

$$P(y) \leftarrow \tilde{P}(y) = \frac{1}{m} \sum_{i} I(y_i = y)$$

# Consider a very simple case

color ←

→ taste ?

| id | color | taste |
|---|---|---|
| 1 | red | sweet |
| 2 | red | sweet |
| 3 | half-red | not-sweet |
| 4 | not-red | not-sweet |
| 5 | not-red | not-sweet |
| 6 | half-red | not-sweet |
| 7 | red | sweet |
| 8 | not-red | not-sweet |
| 9 | not-red | not-sweet |
| 10 | half-red | not-sweet |
| 11 | red | sweet |
| 12 | half-red | not-sweet |
| 13 | not-red | not-sweet |

$$P(\text{red} \mid \text{sweet}) = 1$$

$$P(\text{half-red} \mid \text{sweet}) = 0$$

$$P(\text{not-red} \mid \text{sweet}) = 0$$

$$P(\text{sweet}) = 4/13$$

$$P(\text{red} \mid \text{not-sweet}) = 0$$

$$P(\text{half-red} \mid \text{not-sweet}) = 4/9$$

$$P(\text{not-red} \mid \text{not-sweet}) = 5/9$$

$$P(\text{not-sweet}) = 9/13$$

# Consider a very simple case

| id | color | taste |
|---|---|---|
| 1 | red | sweet |
| 2 | red | sweet |
| 3 | half-red | not-sweet |
| 4 | not-red | not-sweet |
| 5 | not-red | not-sweet |
| 6 | half-red | not-sweet |
| 7 | red | sweet |
| 8 | not-red | not-sweet |
| 9 | not-red | not-sweet |
| 10 | half-red | not-sweet |
| 11 | red | sweet |
| 12 | half-red | not-sweet |
| 13 | not-red | not-sweet |

what the $f'$ would be?

$$f(x) = \arg\max_y P(\boldsymbol{x} \mid y) P(y)$$

# Consider a very simple case

| id | color | taste |
|---|---|---|
| 1 | red | sweet |
| 2 | red | sweet |
| 3 | half-red | not-sweet |
| 4 | not-red | not-sweet |
| 5 | not-red | not-sweet |
| 6 | half-red | not-sweet |
| 7 | red | sweet |
| 8 | not-red | not-sweet |
| 9 | not-red | not-sweet |
| 10 | half-red | not-sweet |
| 11 | red | sweet |
| 12 | half-red | not-sweet |
| 13 | not-red | not-sweet |

what the $f'$ would be?

$$f(x) = \arg\max_y P(\boldsymbol{x} \mid y) P(y)$$

$$P(\text{red} \mid \text{sweet}) P(\text{sweet}) = 4/13$$

$$P(\text{red} \mid \text{not-sweet}) P(\text{not-sweet}) = 0$$

# Consider a very simple case

| id | color | taste |
|---|---|---|
| 1 | red | sweet |
| 2 | red | sweet |
| 3 | half-red | not-sweet |
| 4 | not-red | not-sweet |
| 5 | not-red | not-sweet |
| 6 | half-red | not-sweet |
| 7 | red | sweet |
| 8 | not-red | not-sweet |
| 9 | not-red | not-sweet |
| 10 | half-red | not-sweet |
| 11 | red | sweet |
| 12 | half-red | not-sweet |
| 13 | not-red | not-sweet |

what the $f'$ would be?

$$f(x) = \arg\max_y P(\boldsymbol{x} \mid y)P(y)$$

$$P(\text{red} \mid \text{sweet})P(\text{sweet}) = 4/13$$

$$P(\text{red} \mid \text{not-sweet})P(\text{not-sweet}) = 0$$

$$P(\text{half-red} \mid \text{sweet})P(\text{sweet}) = 0$$

$$P(\text{half-red} \mid \text{not-sweet})P(\text{not-sweet}) = \frac{4}{9} \times \frac{9}{13} = \frac{4}{13}$$

# Consider a very simple case

| id | color | taste |
| --- | --- | --- |
| 1 | red | sweet |
| 2 | red | sweet |
| 3 | half-red | not-sweet |
| 4 | not-red | not-sweet |
| 5 | not-red | not-sweet |
| 6 | half-red | not-sweet |
| 7 | red | sweet |
| 8 | not-red | not-sweet |
| 9 | not-red | not-sweet |
| 10 | half-red | not-sweet |
| 11 | red | sweet |
| 12 | half-red | not-sweet |
| 13 | not-red | not-sweet |

what the $f'$ would be?

$$f(x) = \arg\max_y P(\boldsymbol{x} \mid y) P(y)$$

$$P(\text{red} \mid \text{sweet}) P(\text{sweet}) = 4/13$$

$$P(\text{red} \mid \text{not-sweet}) P(\text{not-sweet}) = 0$$

$$P(\text{half-red} \mid \text{sweet}) P(\text{sweet}) = 0$$

$$P(\text{half-red} \mid \text{not-sweet}) P(\text{not-sweet}) = \frac{4}{9} \times \frac{9}{13} = \frac{4}{13}$$

*perfect*
*but not realistic*

# Naive Bayes

$$f(x) = \arg\max_y P(\boldsymbol{x} \mid y) P(y)$$

estimation the a priori by frequency:

$$P(y) \leftarrow \tilde{P}(y) = \frac{1}{m} \sum_i I(y_i = y)$$

assume features are conditional independence given the class (naive assumption):

$$\begin{aligned} P(\boldsymbol{x} \mid y) &= P(x_1, x_2, \ldots, x_n \mid y) \\ &= P(x_1 \mid y) \cdot P(x_2 \mid y) \cdot \ldots P(x_n \mid y) \end{aligned}$$

decision function:

$$f(x) = \arg\max_y \tilde{P}(y) \prod_i \tilde{P}(x_i \mid y)$$

# Naive Bayes

color={0,1,2,3} weight={0,1,2,3,4}

| color | weight | sweet? |
|---|---|---|
| 3 | 4 | yes |
| 2 | 3 | yes |
| 0 | 3 | no |
| 3 | 2 | no |
| 1 | 4 | no |

$$P(y = yes) = 2/5$$

$$P(y = no) = 3/5$$

$$P(color = 3 \mid y = yes) = 1/2$$

...

# Naive Bayes

color={0,1,2,3} weight={0,1,2,3,4}

| color | weight | sweet? |
|---|---|---|
| 3 | 4 | yes |
| 2 | 3 | yes |
| 0 | 3 | no |
| 3 | 2 | no |
| 1 | 4 | no |

$$P(y = yes) = 2/5$$

$$P(y = no) = 3/5$$

$$P(color = 3 \mid y = yes) = 1/2$$

...

$$f(y \mid color = 3, weight = 3) \rightarrow$$

# Naive Bayes

color={0,1,2,3} weight={0,1,2,3,4}

| color | weight | sweet? |
|---|---|---|
| 3 | 4 | yes |
| 2 | 3 | yes |
| 0 | 3 | no |
| 3 | 2 | no |
| 1 | 4 | no |

$P(y = yes) = 2/5$

$P(y = no) = 3/5$

$P(color = 3 \mid y = yes) = 1/2$

...

$$f(y \mid color = 3, weight = 3) \rightarrow$$

$$P(color = 3 \mid y = yes)P(weight = 3 \mid y = yes)P(y = yes) = 0.5 \times 0.5 \times 0.4 = 0.1$$

$$P(color = 3 \mid y = no)P(weight = 3 \mid y = no)P(y = no) = 0.33 \times 0.33 \times 0.6 = 0.06$$

# Naive Bayes

color={0,1,2,3} weight={0,1,2,3,4}

| color | weight | sweet? |
|---|---|---|
| 3 | 4 | yes |
| 2 | 3 | yes |
| 0 | 3 | no |
| 3 | 2 | no |
| 1 | 4 | no |

$$P(y = yes) = 2/5$$

$$P(y = no) = 3/5$$

$$P(color = 3 \mid y = yes) = 1/2$$

...

$$f(y \mid color = 3, weight = 3) \rightarrow$$

$$P(color = 3 \mid y = yes)P(weight = 3 \mid y = yes)P(y = yes) = 0.5 \times 0.5 \times 0.4 = 0.1$$

$$P(color = 3 \mid y = no)P(weight = 3 \mid y = no)P(y = no) = 0.33 \times 0.33 \times 0.6 = 0.06$$

$$f(y \mid color = 0, weight = 1) \rightarrow$$

# Naive Bayes

color={0,1,2,3} weight={0,1,2,3,4}

| color | weight | sweet? |
|---|---|---|
| 3 | 4 | yes |
| 2 | 3 | yes |
| 0 | 3 | no |
| 3 | 2 | no |
| 1 | 4 | no |

$$P(y = yes) = 2/5$$

$$P(y = no) = 3/5$$

$$P(color = 3 \mid y = yes) = 1/2$$

...

$$f(y \mid color = 3, weight = 3) \to$$

$$P(color = 3 \mid y = yes)P(weight = 3 \mid y = yes)P(y = yes) = 0.5 \times 0.5 \times 0.4 = 0.1$$

$$P(color = 3 \mid y = no)P(weight = 3 \mid y = no)P(y = no) = 0.33 \times 0.33 \times 0.6 = 0.06$$

$$f(y \mid color = 0, weight = 1) \to$$

$$P(color = 0 \mid y = yes)P(weight = 1 \mid y = yes)P(y = yes) = 0$$

$$P(color = 0 \mid y = no)P(weight = 1 \mid y = no)P(y = no) = 0$$

# Naive Bayes

color={0,1,2,3} weight={0,1,2,3,4}

| color | weight | sweet? |
|---|---|---|
| 3 | 4 | yes |
| 2 | 3 | yes |
| 0 | 3 | no |
| 3 | 2 | no |
| 1 | 4 | no |

+

| color | sweet? |
|---|---|
| 0 | yes |
| 1 | yes |
| 2 | yes |
| 3 | yes |

**smoothed (Laplacian correction) probabilities:**

$$P(color = 0 \mid y = yes) = (0 + 1)/(2 + 4)$$

$$P(y = yes) = (2 + 1)/(5 + 2)$$

for counting frequency, assume every event has happened once.

$$f(y \mid color = 0, weight = 1) \rightarrow$$

$$P(color = 0 \mid y = yes)P(weight = 1 \mid y = yes)P(y = yes) = \frac{1}{6} \times \frac{1}{7} \times \frac{3}{7} = 0.01$$

$$P(color = 0 \mid y = no)P(weight = 1 \mid y = no)P(y = no) = \frac{2}{7} \times \frac{1}{8} \times \frac{4}{7} = 0.02$$

# Naive Bayes

advantages:

- very fast:
  - scan the data once, just count: $O(mn)$
  - store class-conditional probabilities: $O(n)$
  - test an instance: $O(cn)$ ($c$ the number of classes)
- good accuracy in many cases
- parameter free
- output a probability
- naturally handle multi-class

disadvantages:

# Naive Bayes

advantages:

- very fast:
  - scan the data once, just count: $O(mn)$
  - store class-conditional probabilities: $O(n)$
  - test an instance: $O(cn)$ ($c$ the number of classes)
- good accuracy in many cases
- parameter free
- output a probability
- naturally handle multi-class

disadvantages:

- the strong assumption may harm the accuracy
- does not handle numerical features naturally

# Ensemble Learning

# How can we improve an algorithm

*for free*

one classifier with error 0.49

# 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

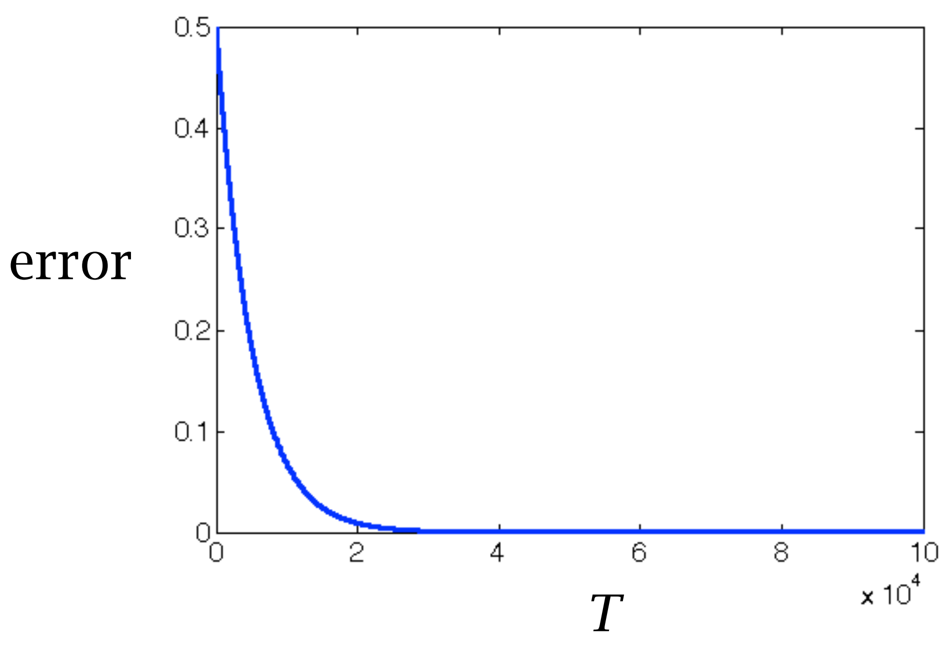
# Motivation theories

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:

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

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



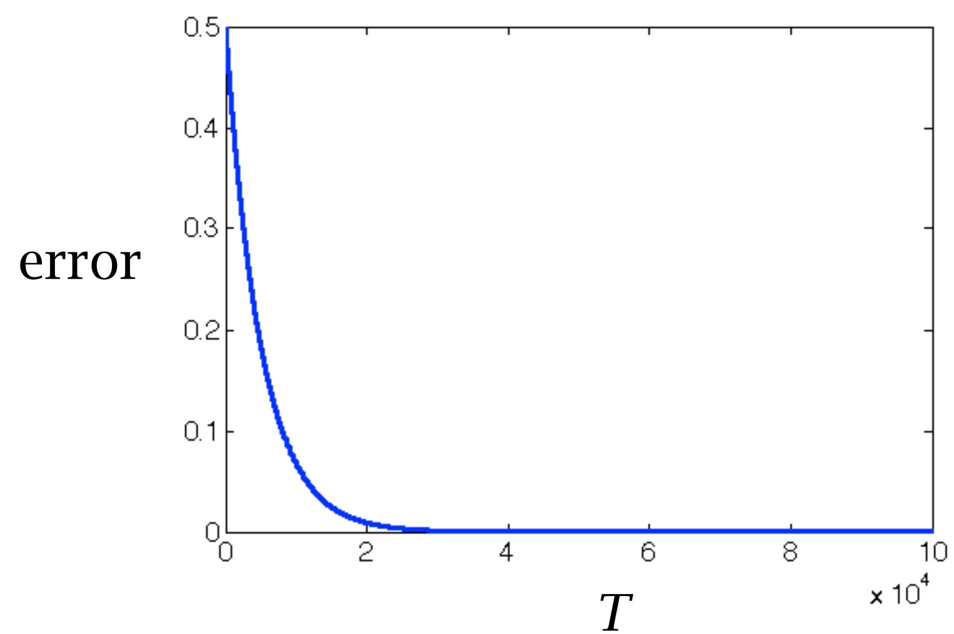
# Motivation theories

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:

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

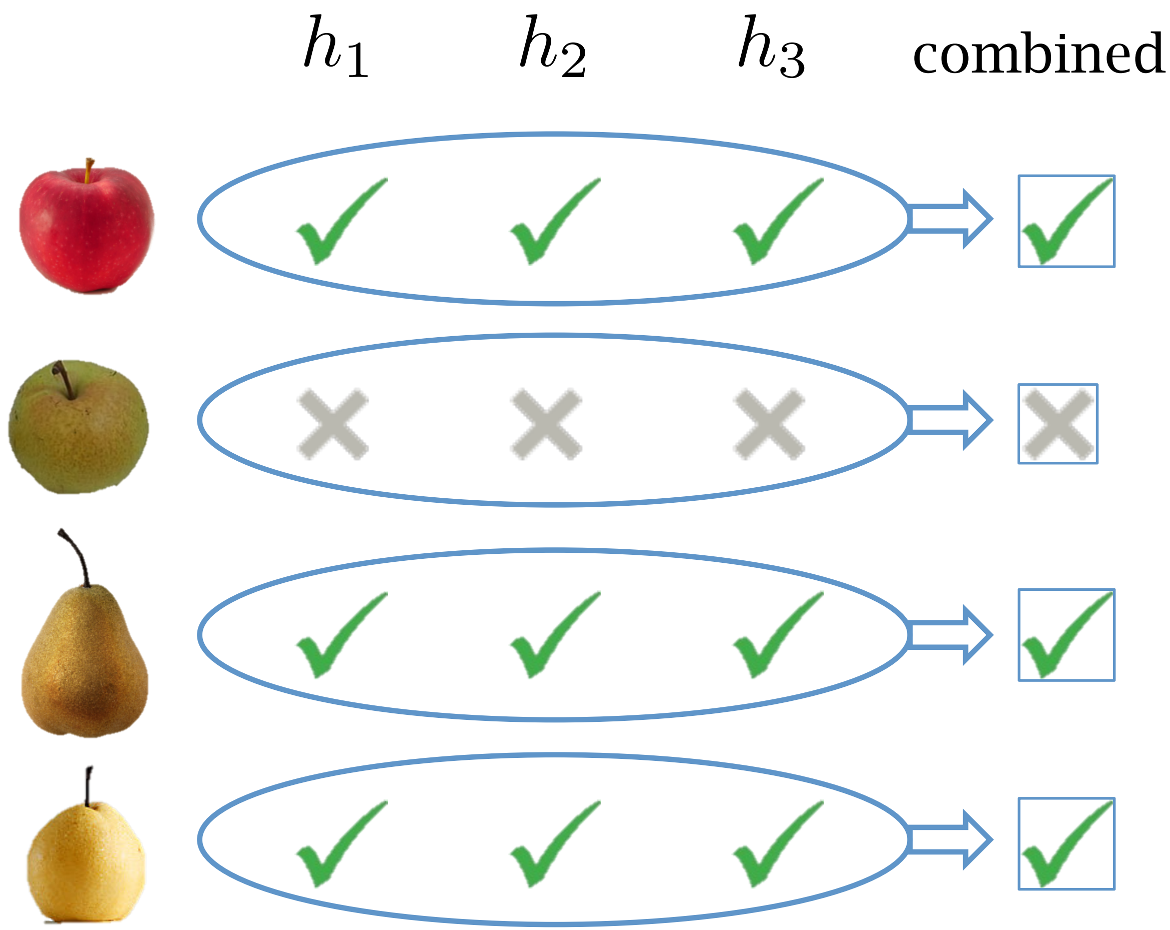
$$\leq \frac{1}{2} e^{-2T(0.5-0.49)^2}$$




*but independent classifiers are not achievable*

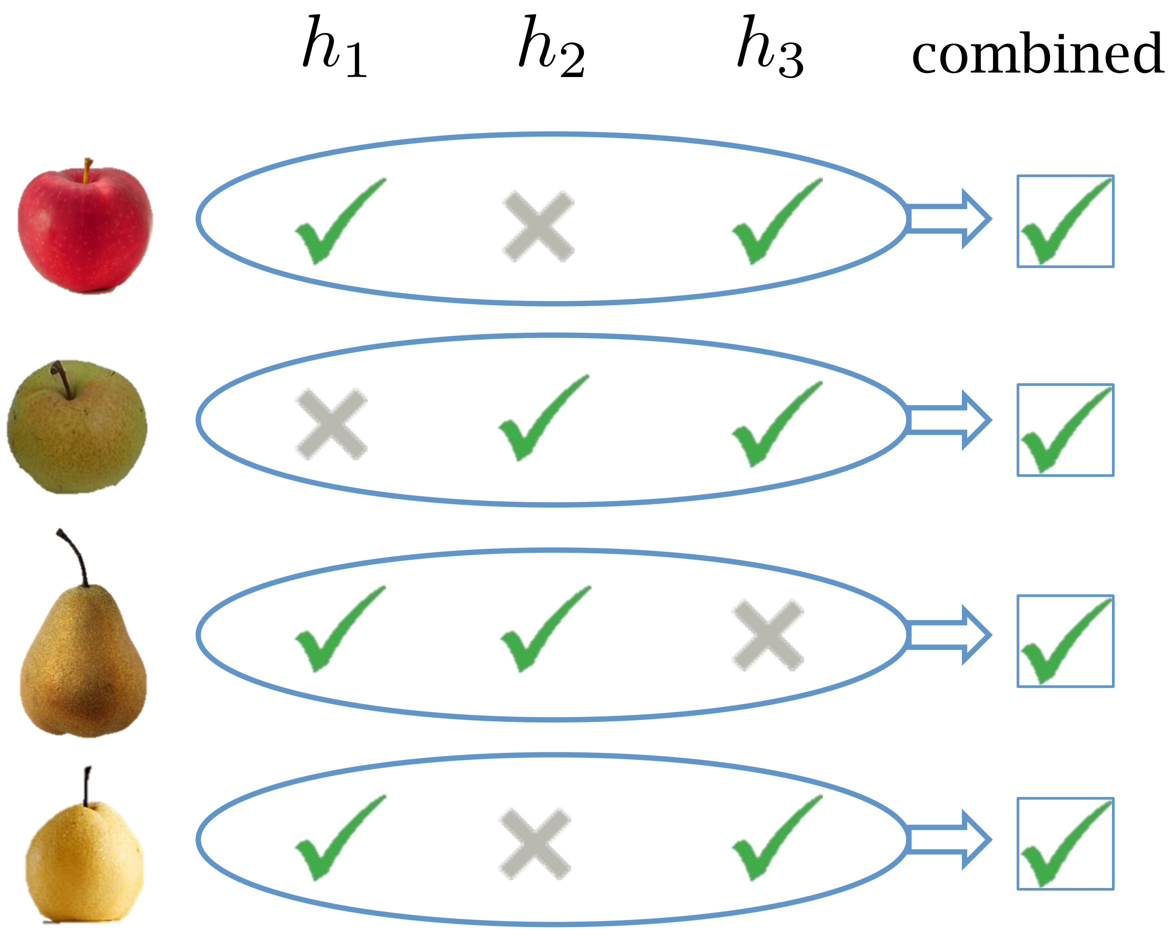
# The importance of diversity

not useful to combine identical base learners



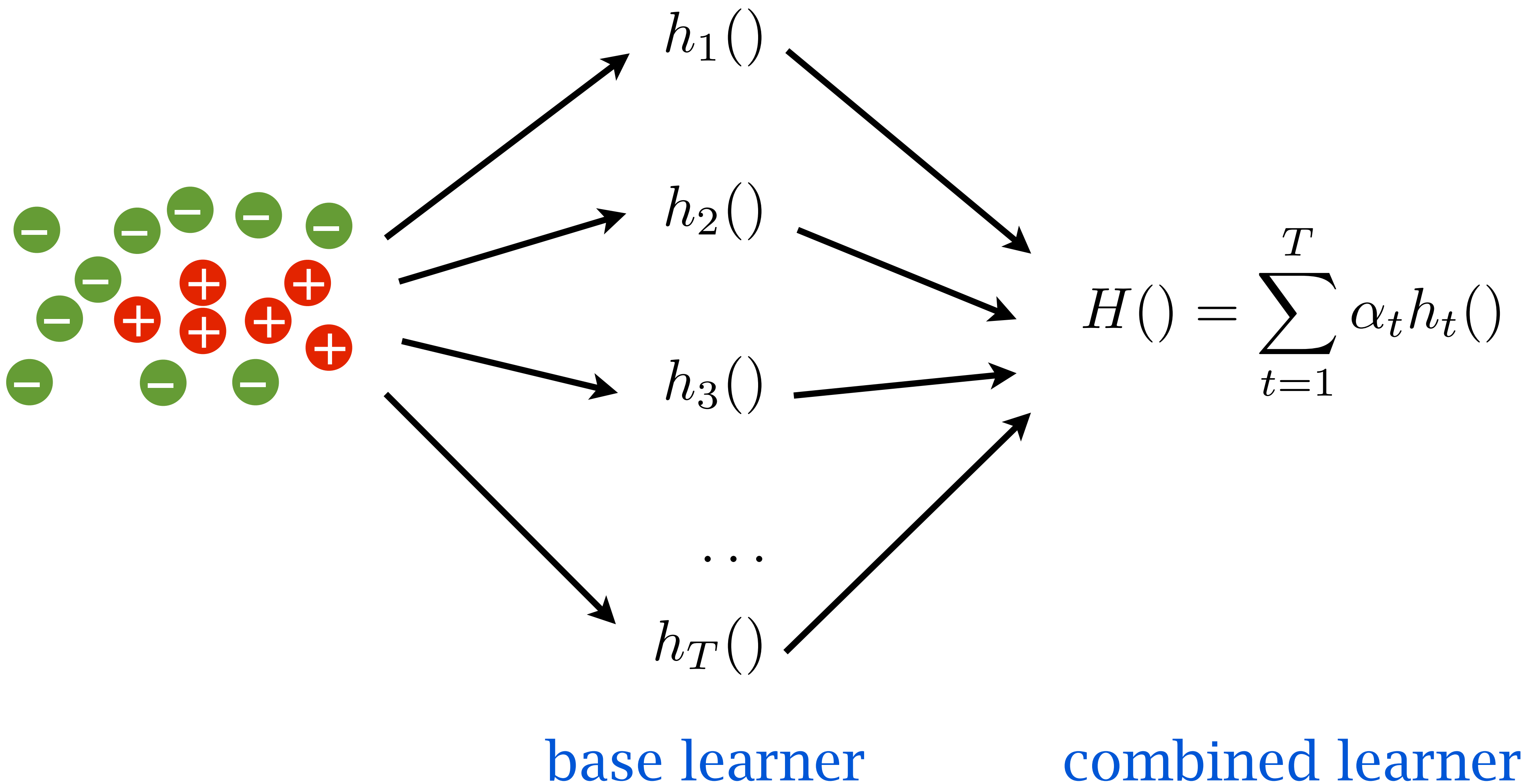
# The importance of diversity

good to combine different learners



# Ensemble learning

combination of multiple classifiers/regressors



# Ensemble methods

Parallel ensemble

create diverse base learners by introducing randomness

Sequential ensemble

create base learners by complementarity

# Parallel ensemble methods

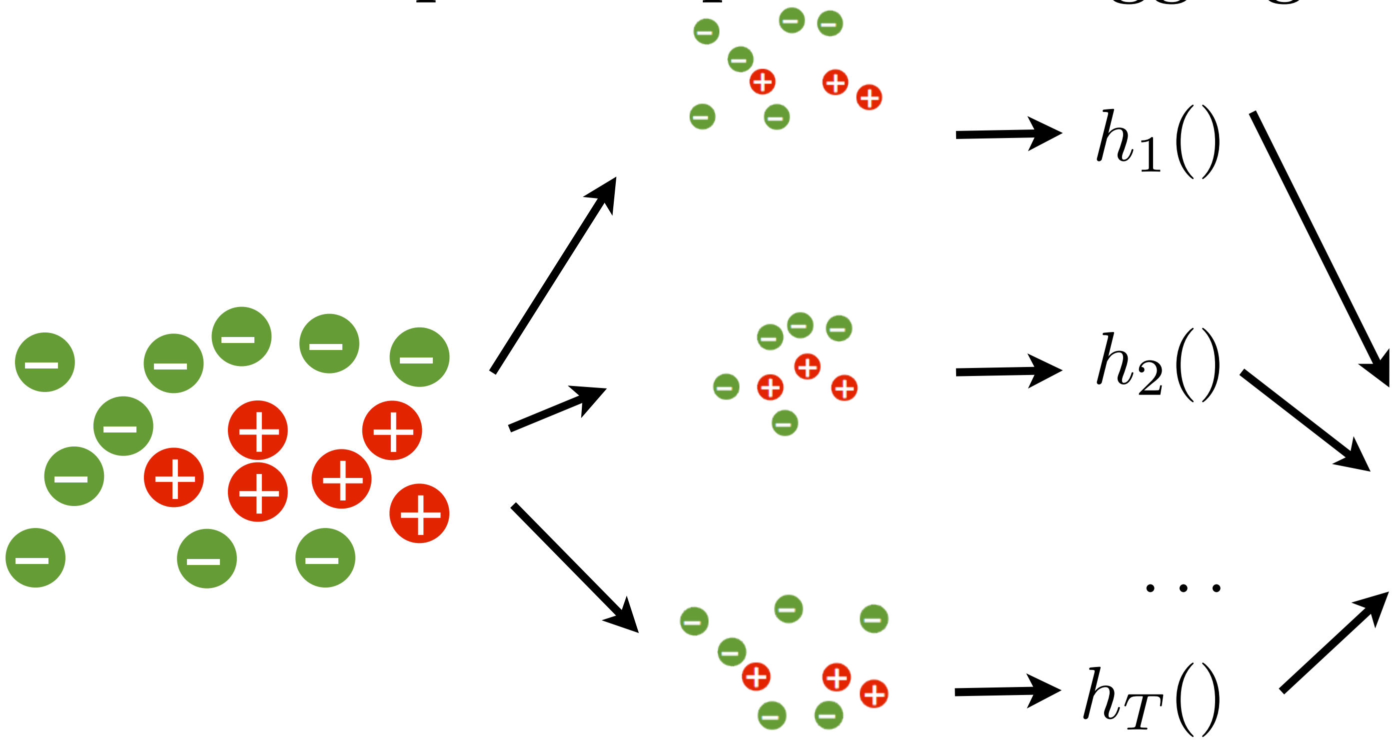
Diversity generating categories:

- 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




randomly sample data

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

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

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

(classification)

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:**

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

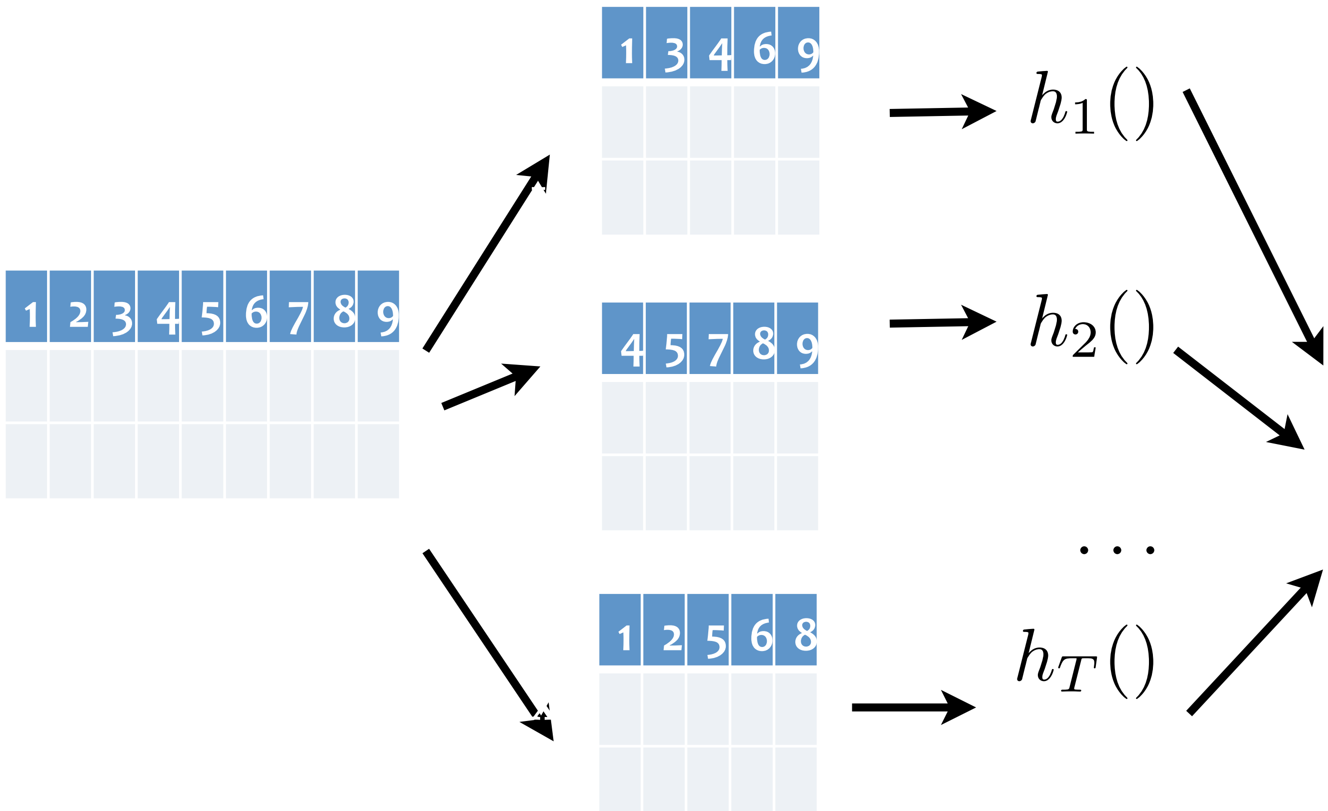
» NB, linear classifier are not

Good for handling large data set

Leo Breiman
1928-2005

# Parallel ensemble methods

## Input Feature Manipulation: Random subspace




randomly sample subspace

$H(\boldsymbol{x}) =$

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

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

(classification)

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.

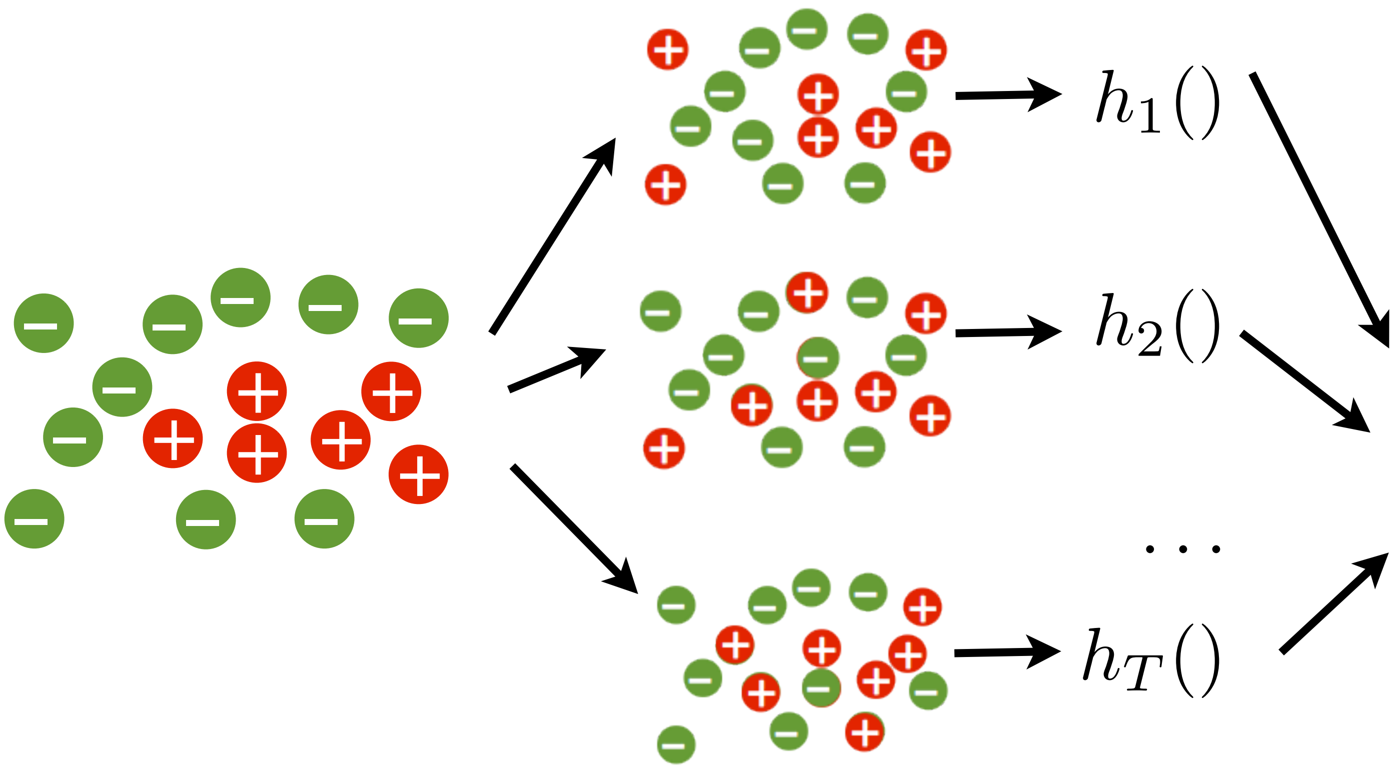
**Process:**
1. **for** $t = 1, \ldots, T$:
2. $\mathcal{F}_t = RS(D, d)$ % $\mathcal{F}_t$ is a set of $d$ randomly selected features;
3. $D_t = Map_{\mathcal{F}_t}(D)$ % $D_t$ keeps only the features in $\mathcal{F}_t$
4. $h_t = \mathfrak{L}(D_t)$ % Train a learner
5. **end**

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

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

# Parallel ensemble methods

## Output Representation Manipulation: Output flipping




randomly modify the label

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

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

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

(classification)

May drastically reduce the accuracy of base learners

# Parallel ensemble methods

Learning Parameter Manipulation: Random forest

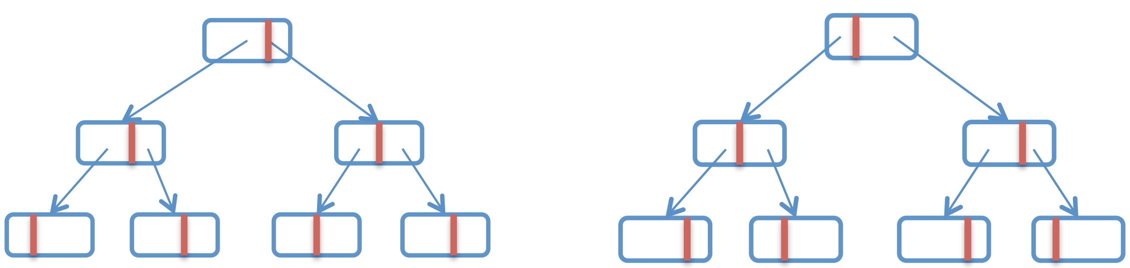
Randomized decision tree

**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

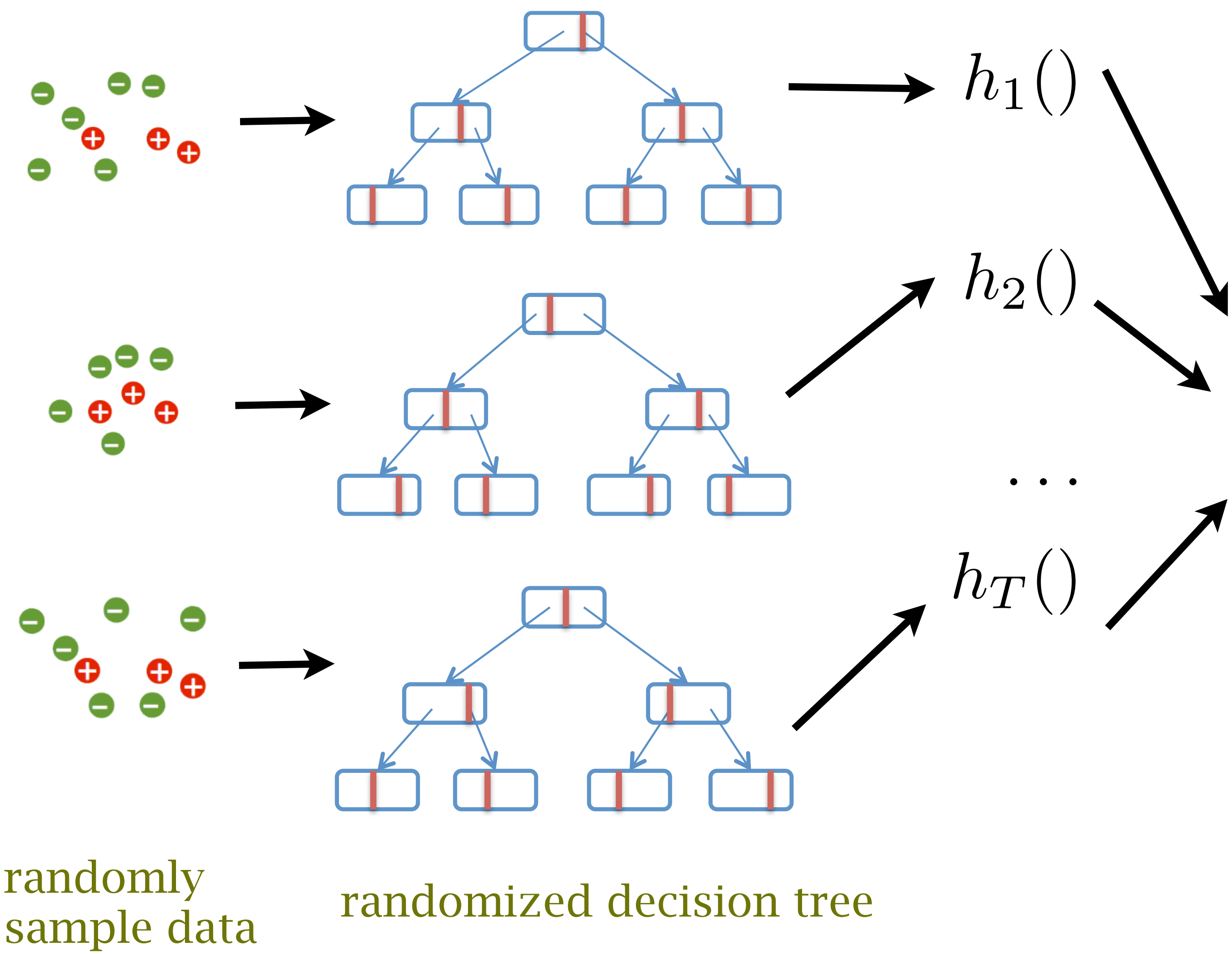
(other variants are available)



every run produce a different tree

# Parallel ensemble methods

## Learning Parameter Manipulation: Random forest




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

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

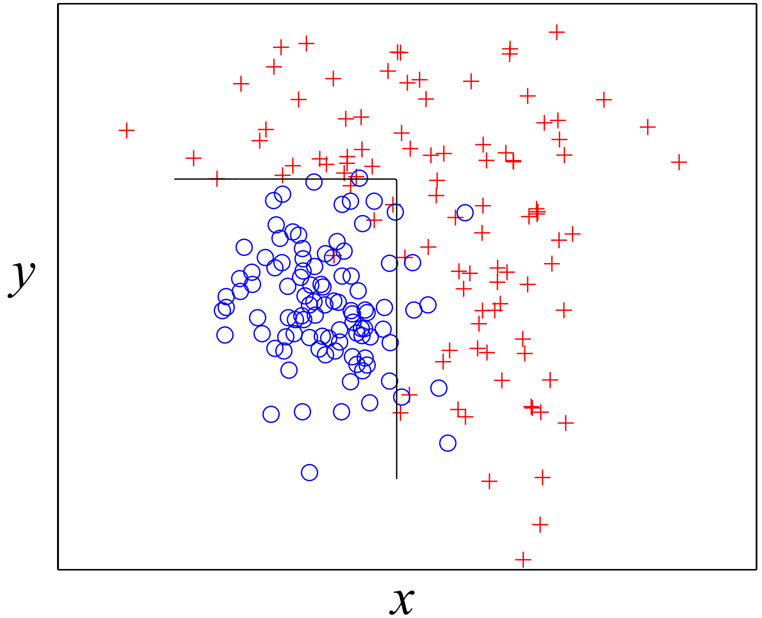
$$\arg\max_{y} \sum_{t=1}^{T} I(h_t(\boldsymbol{x}) = y)$$

(classification)

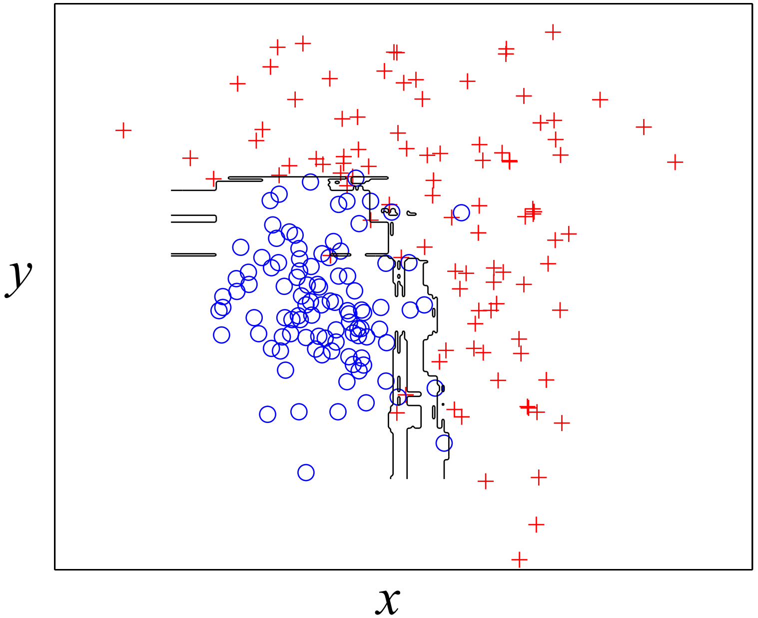
Bagging of randomized decision tree

# 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:

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

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

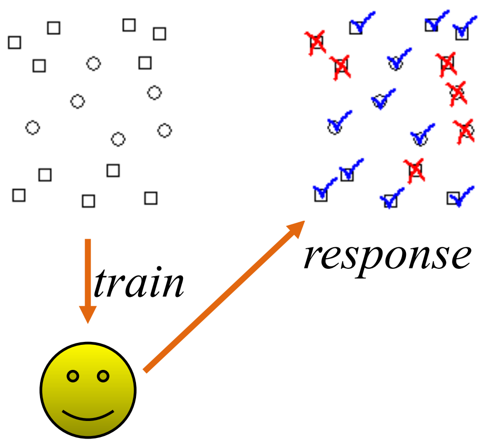
decision tree: the purity of the leave node

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

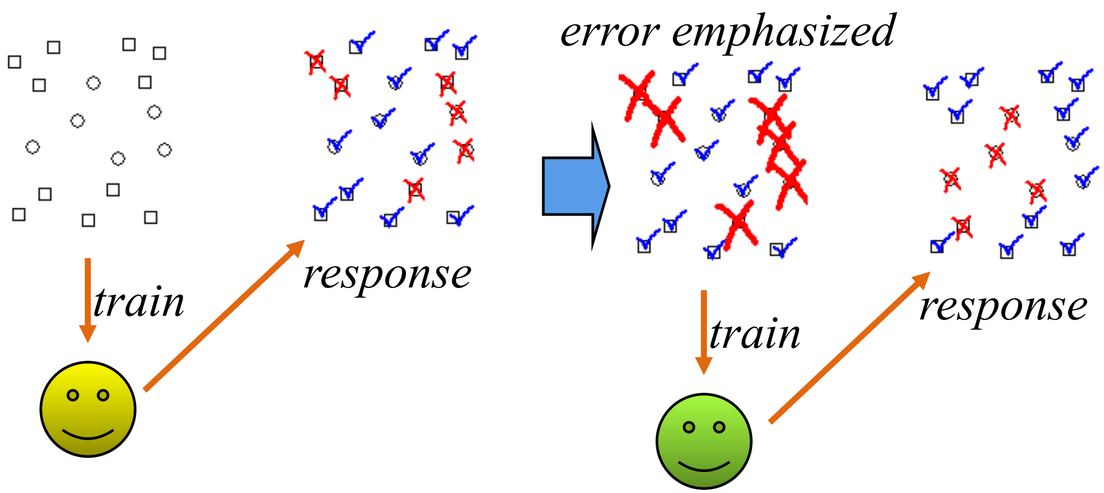
Generate learners sequentially,
focus on previous errors




so that the combination of learners will have a high accuracy

# Sequential ensemble methods

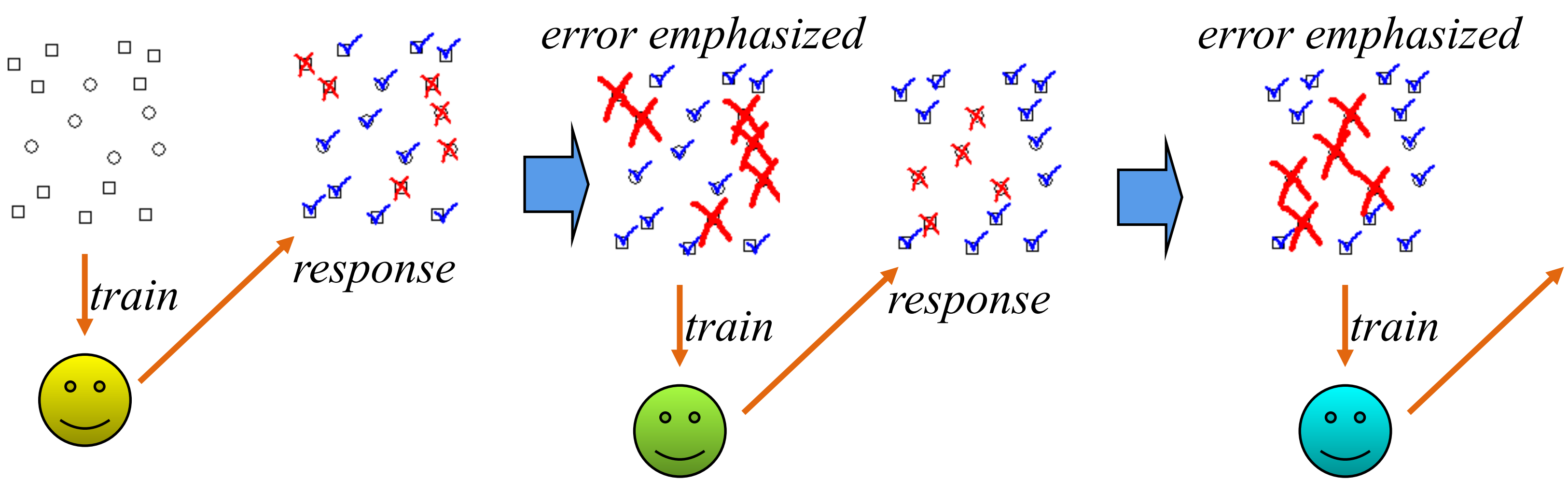
Generate learners sequentially,
focus on previous errors




so that the combination of learners will have a high accuracy

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

# 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

# AdaBoost

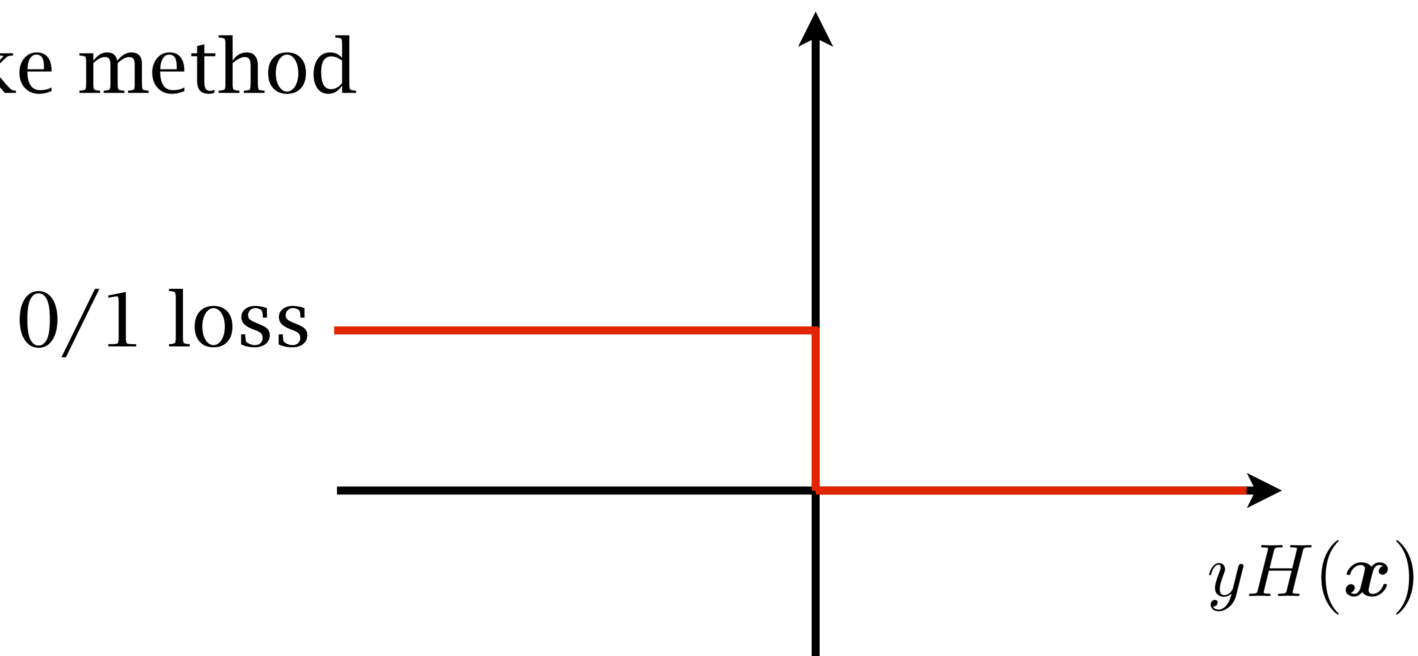
fit an additive model, sequentially

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

to minimize exponential loss

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

by Newton-like method



# AdaBoost

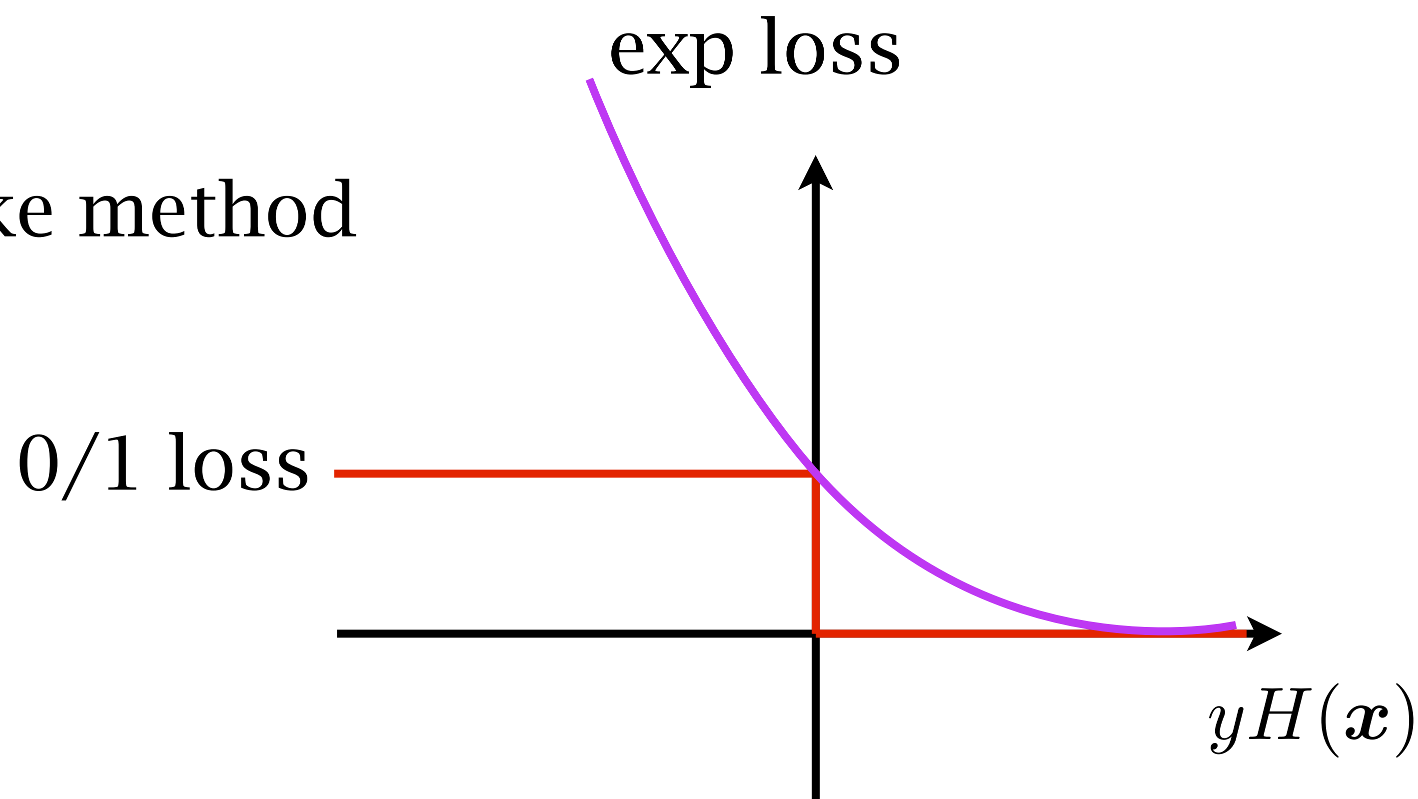
fit an additive model, sequentially

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

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

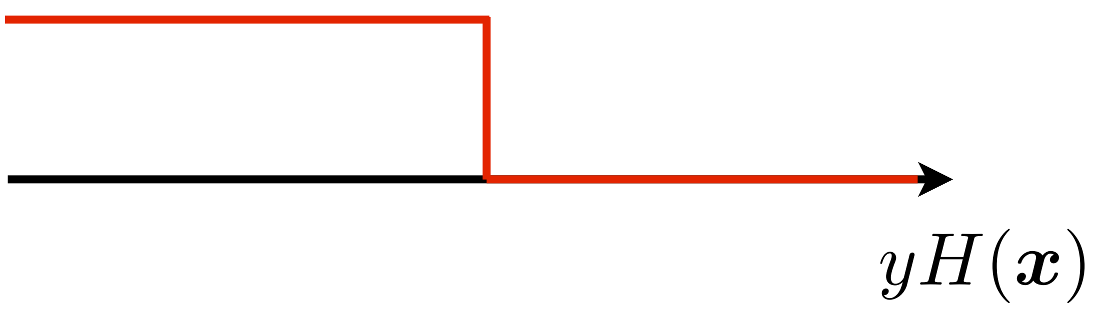
Gradient boosting (for classification)

# Gradient boosting

Gradient boosting (for classification)

0-1 loss

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



# Gradient boosting

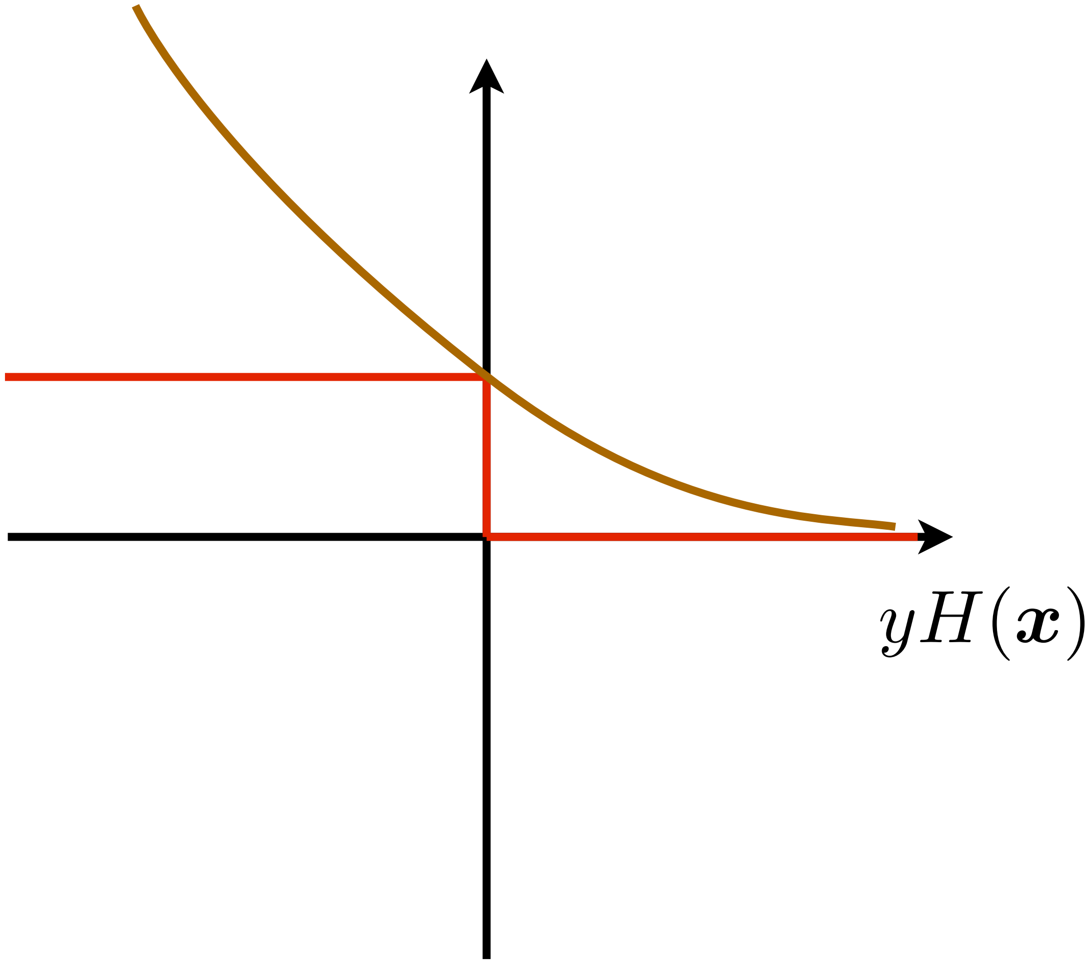
Gradient boosting (for classification)

0-1 loss

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

logistic regression

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



# Gradient boosting

Gradient boosting (for classification)

0-1 loss

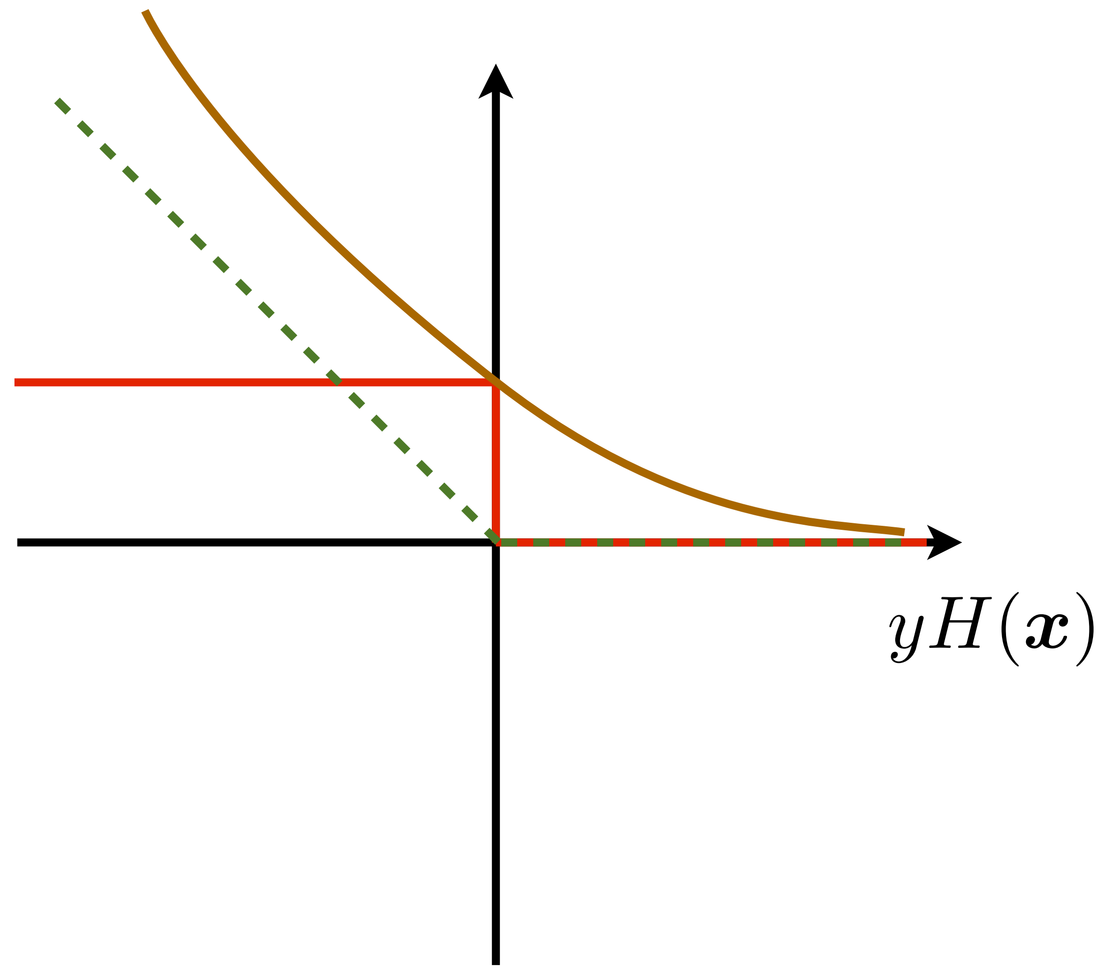
$$\min I(yH(\boldsymbol{x}) \leq 0)$$

logistic regression

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

perceptron

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



# Gradient boosting

Gradient boosting (for classification)

0-1 loss

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

logistic regression

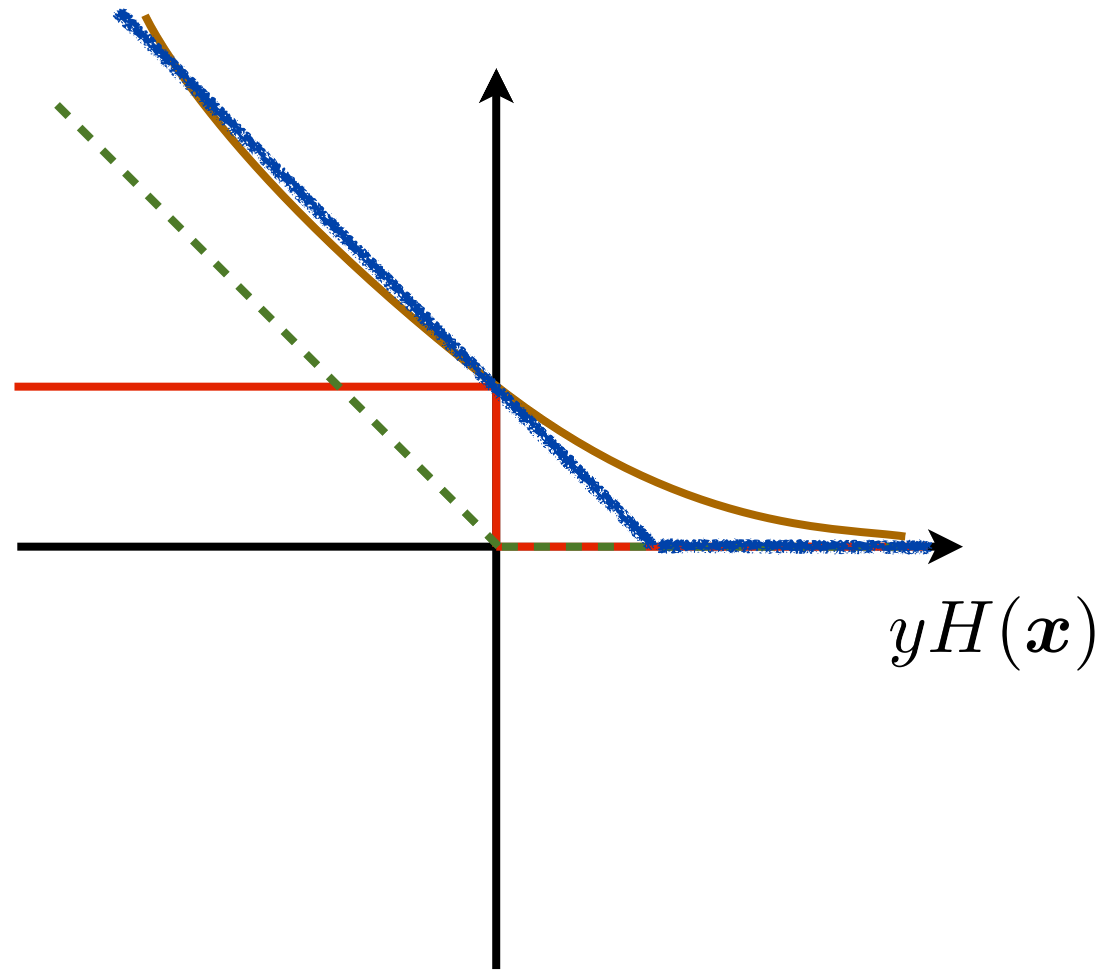
$$\min \log(1 + e^{-yH(\boldsymbol{x})})$$

perceptron

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

hinge loss

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



# 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

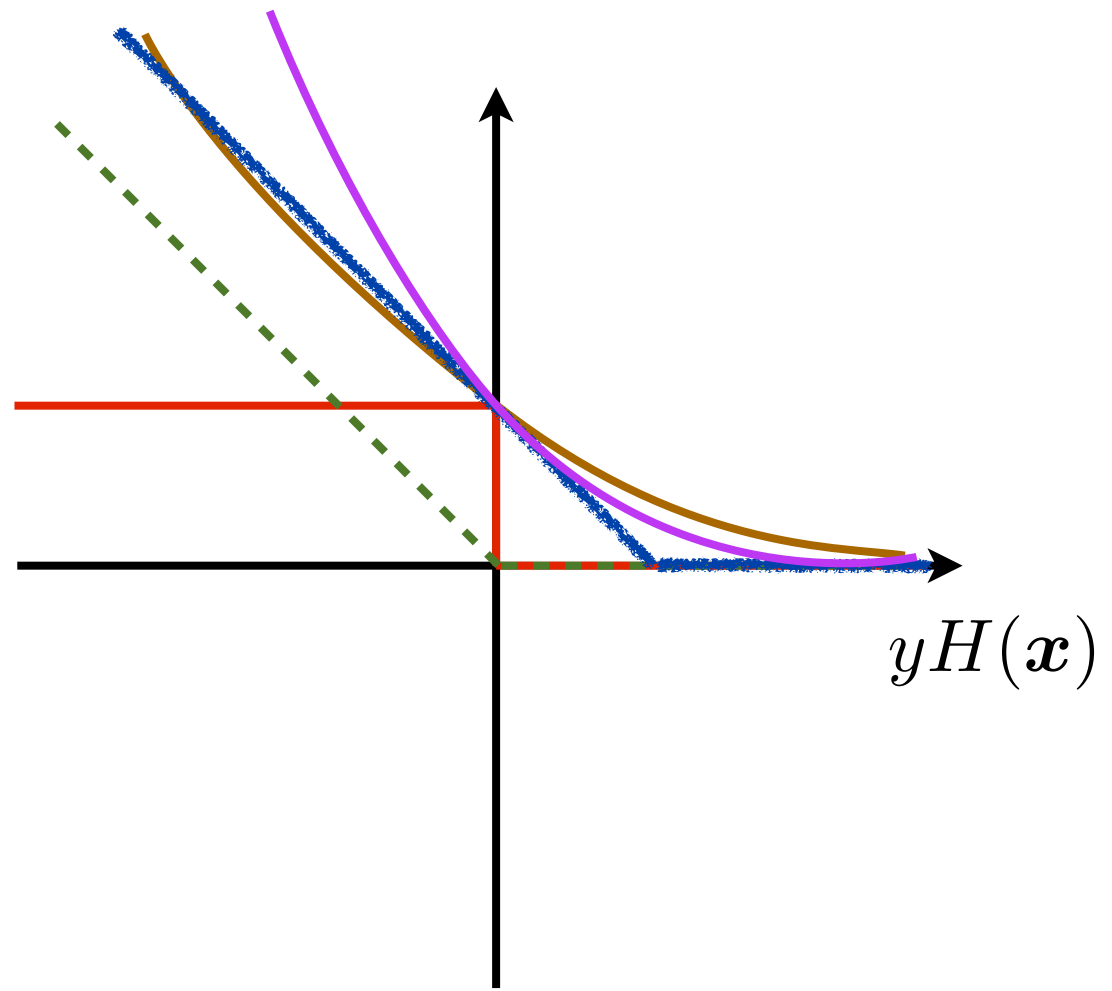
$$\min \max\{-yH(\boldsymbol{x}), 0\}$$

hinge loss

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

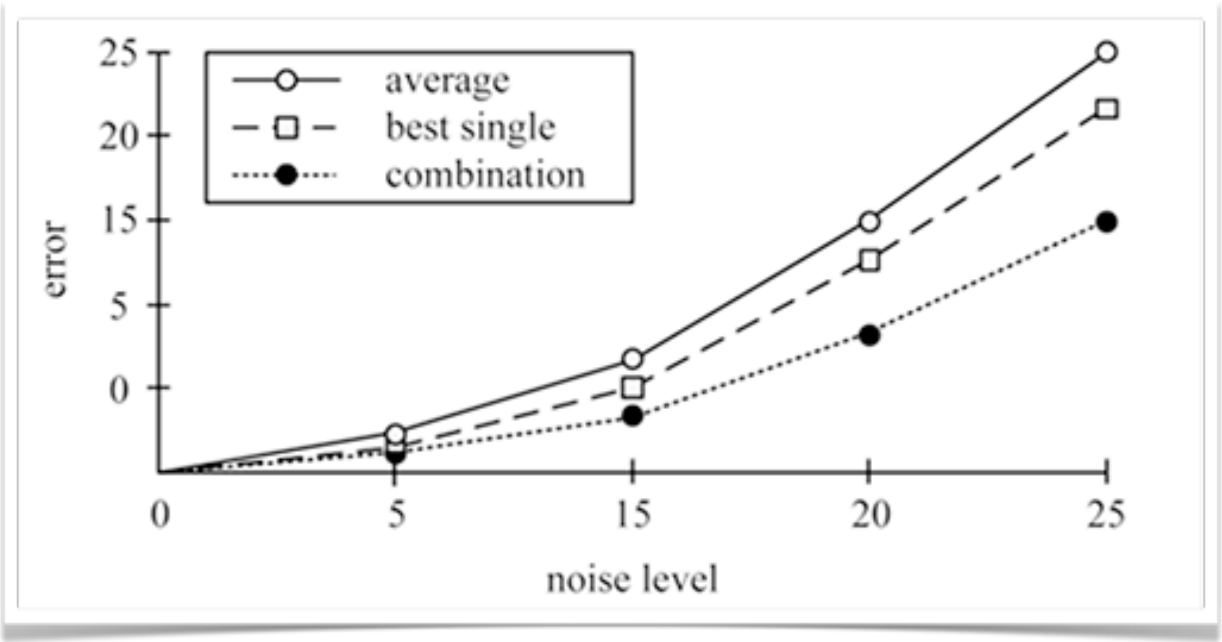
exponential loss

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



# 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*

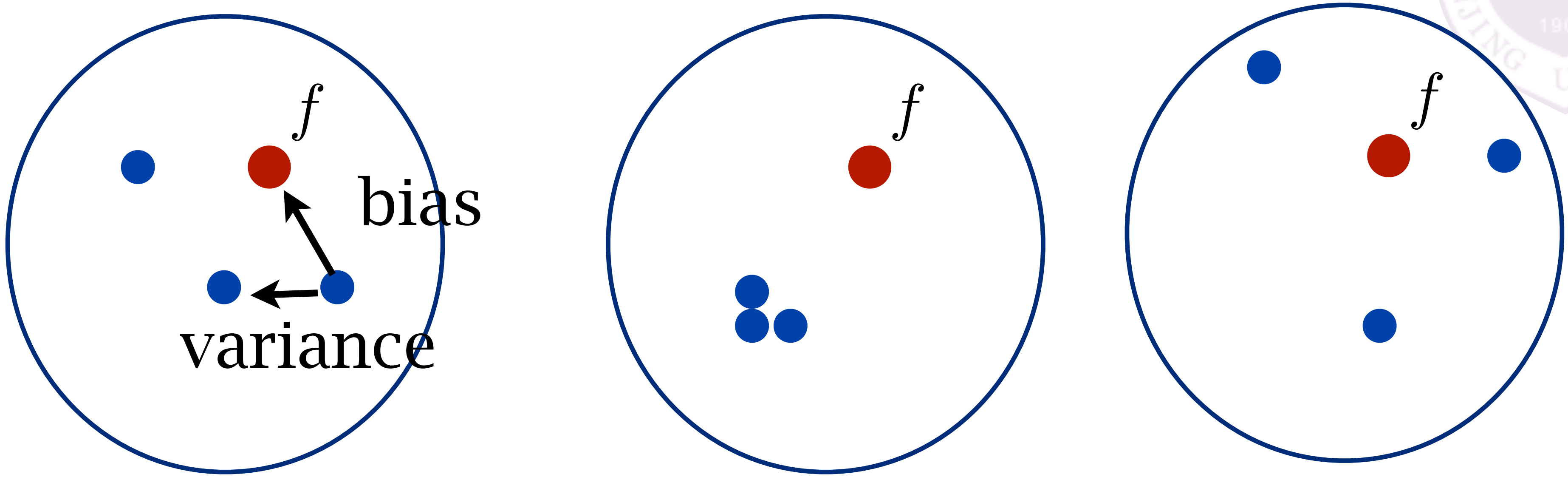
# More about ensemble

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:

AdaBoost

# 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

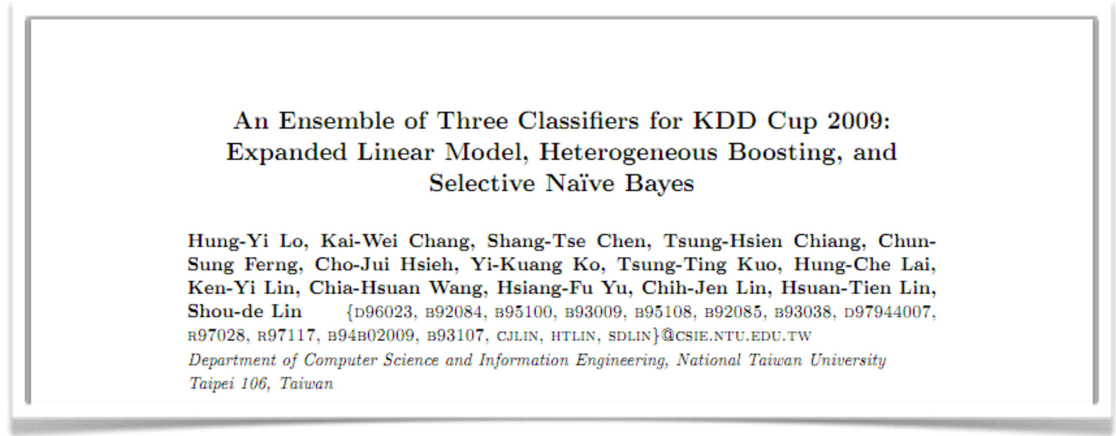
(Gödel Prize 2003)

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.