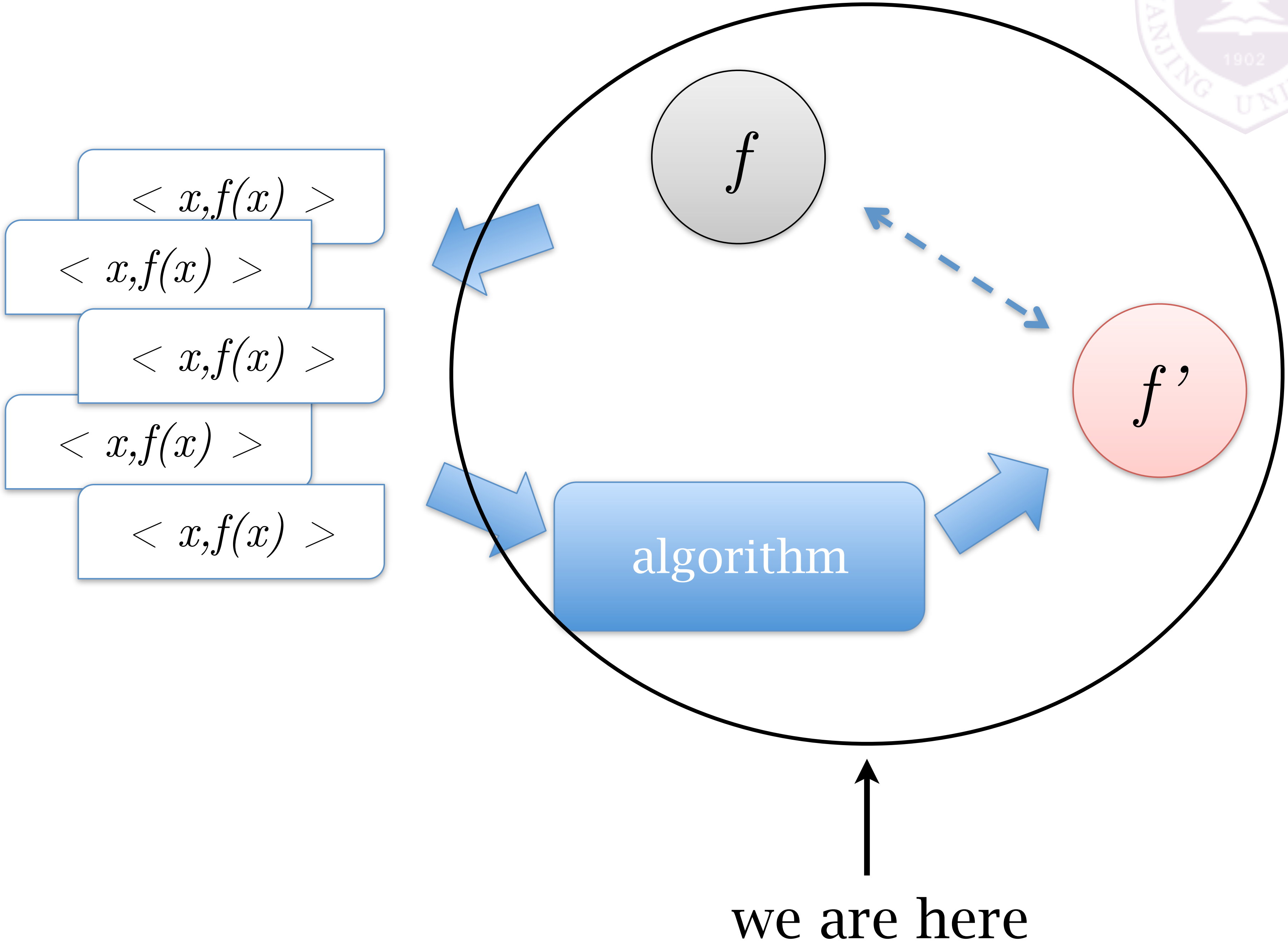
Lecture 3: Machine Learning I

Supervised Learning & Basic Algorithms

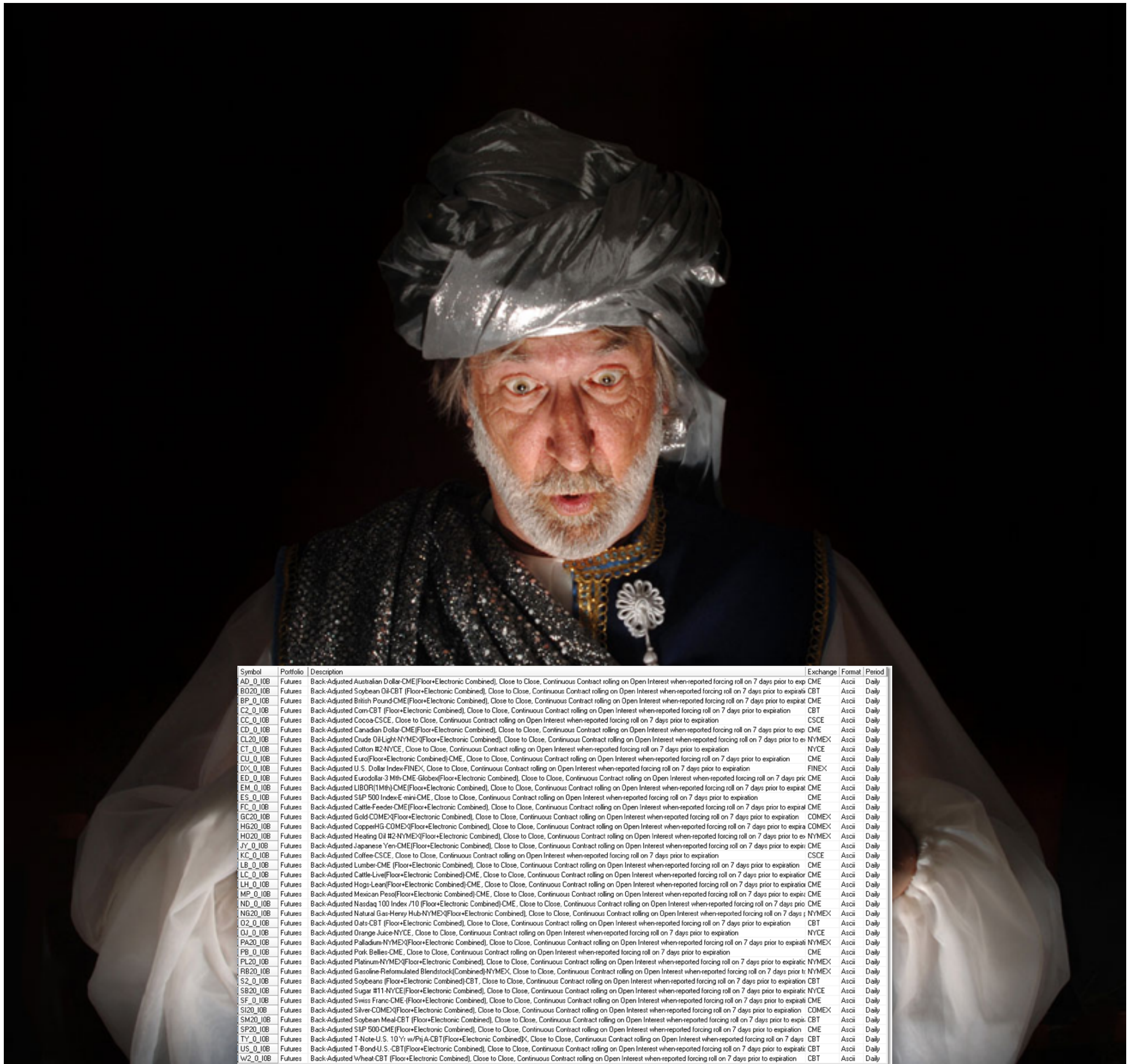
http://cs.nju.edu.cn/yuy/course_dm14ms.ashx

Position



The desire of prediction

The desire of prediction

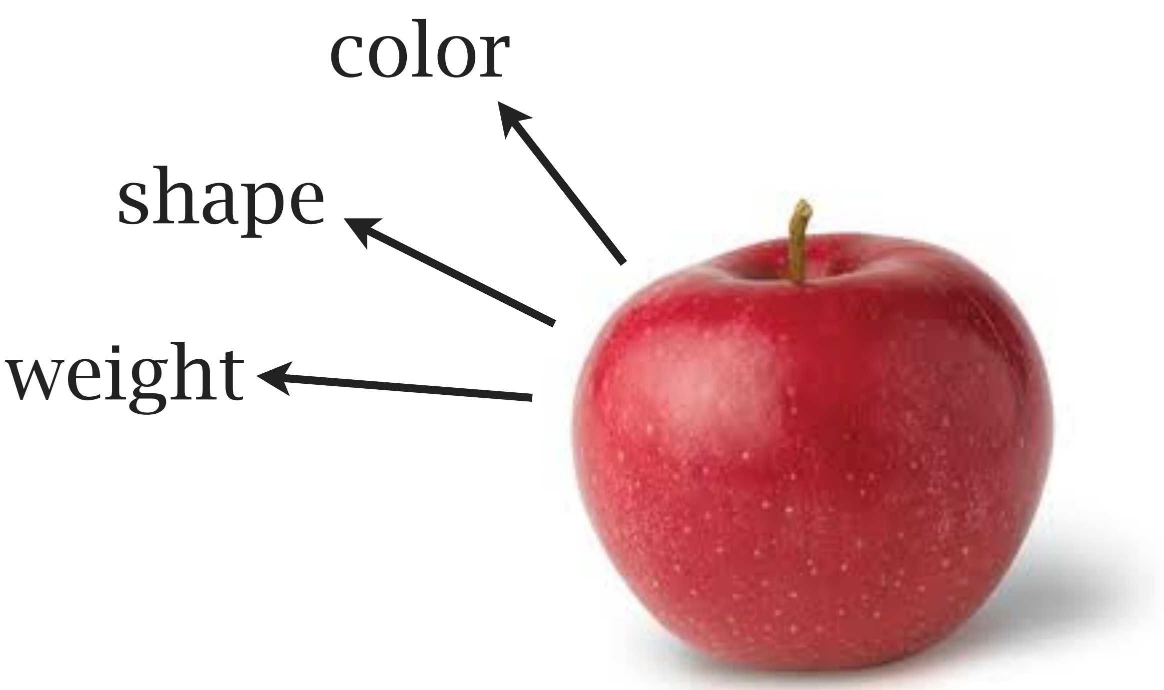


Predictive modeling

Find a relation between a set of variables (features) to target variables (labels).

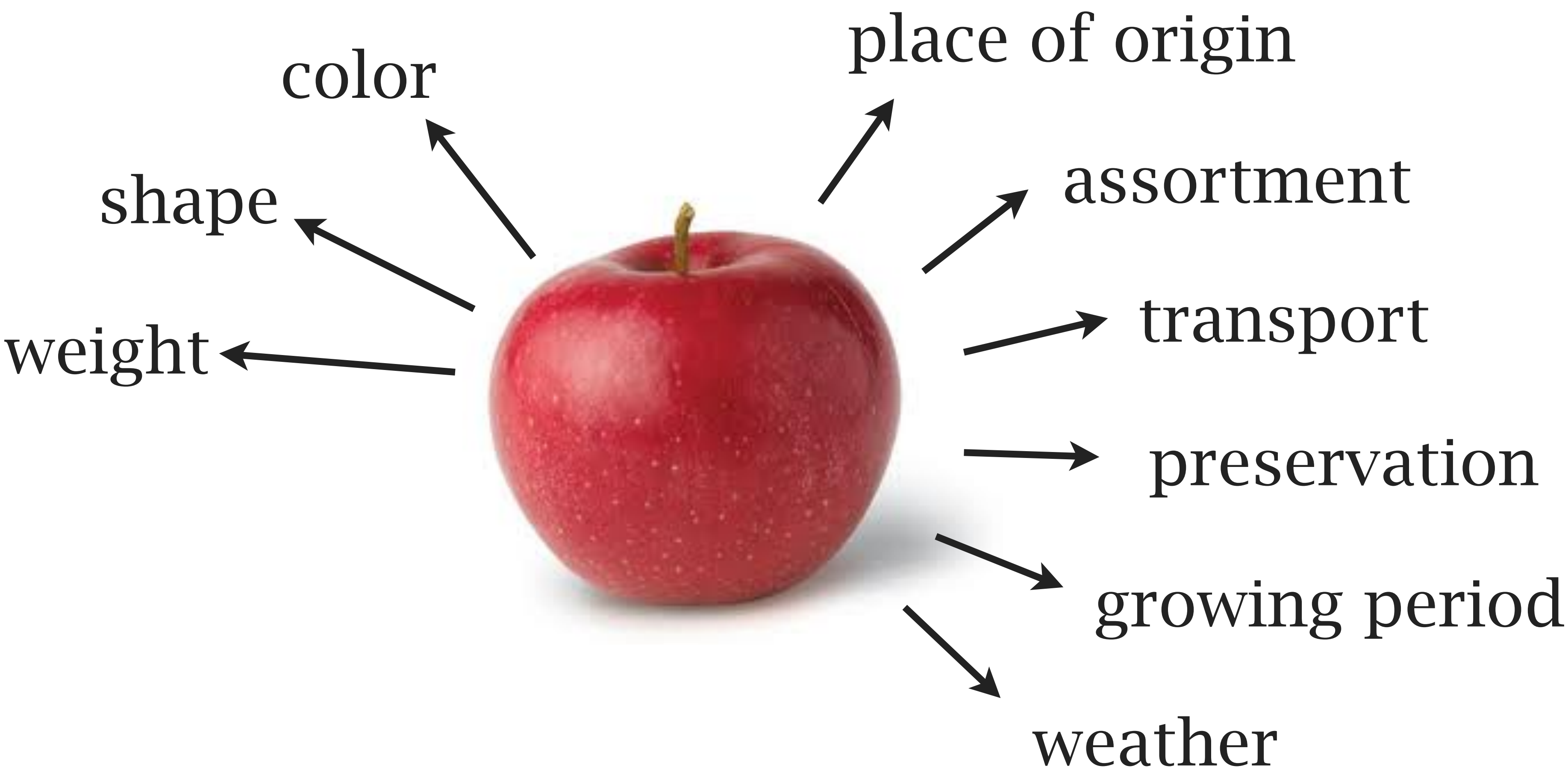
Predictive modeling

Find a relation between a set of variables (features) to target variables (labels).



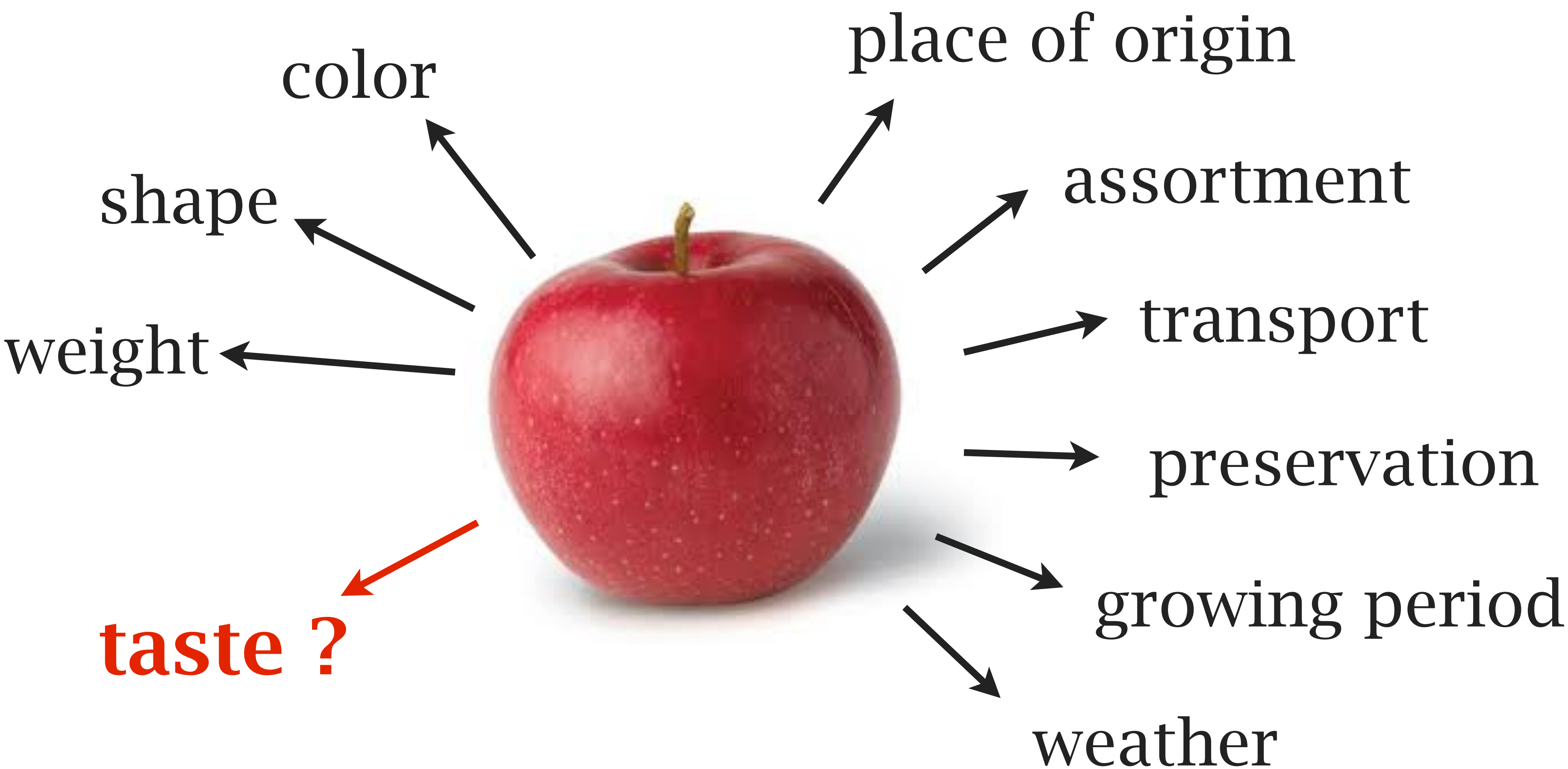
Predictive modeling

Find a relation between a set of variables (features) to target variables (labels).



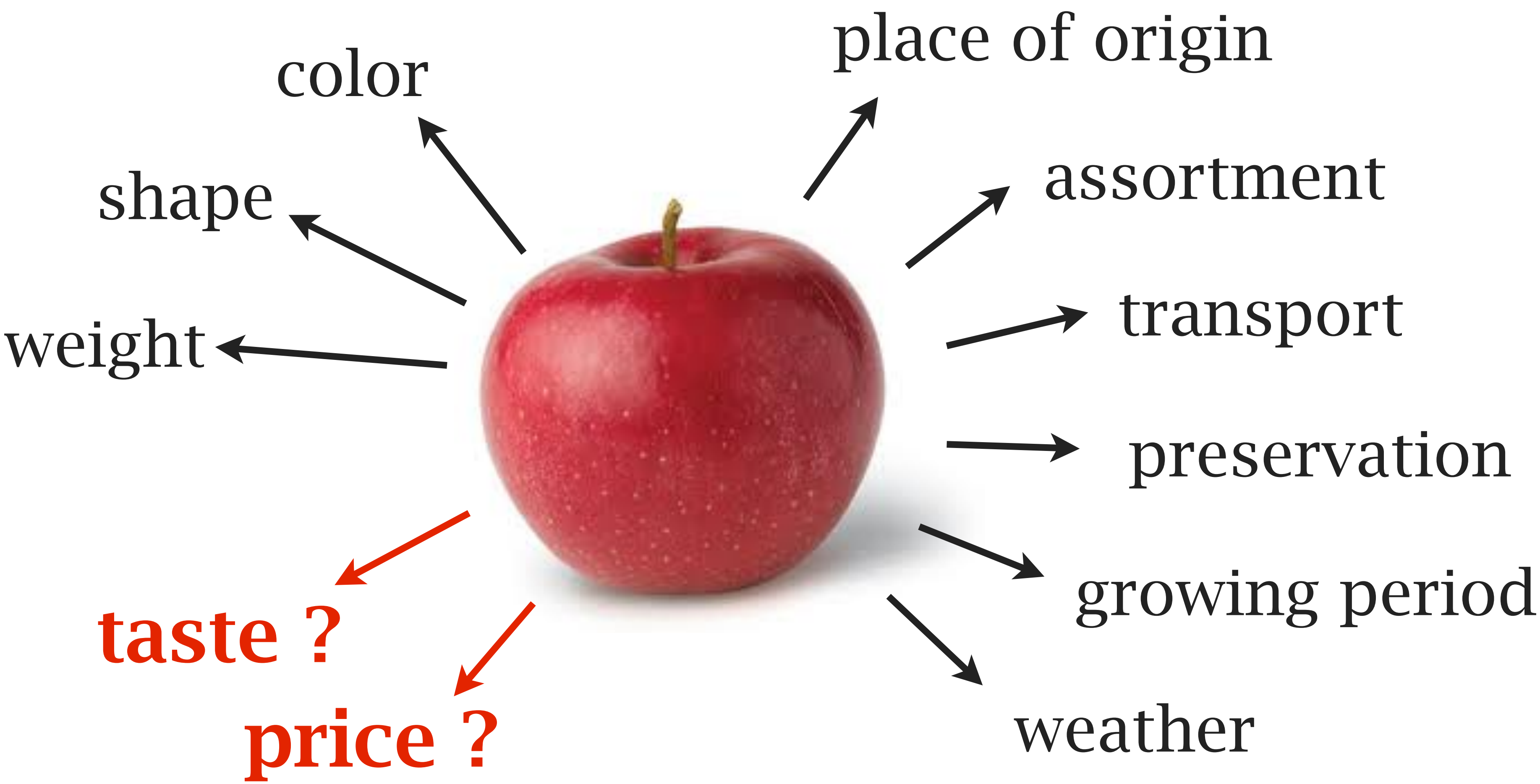
Predictive modeling

Find a relation between a set of variables (features) to target variables (labels).



Predictive modeling

Find a relation between a set of variables (features) to target variables (labels).



Supervised learning/inductive learning

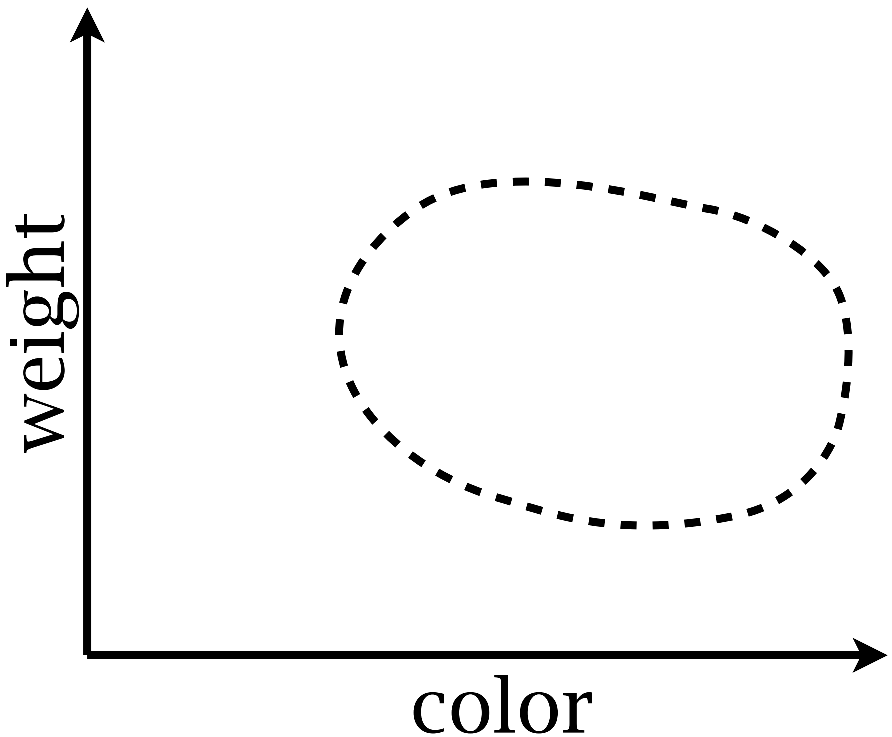
Find a relation between a set of variables (features) to target variables (labels) *from finite examples*.

tasks

- Classification: label is a nominal feature
- Regression: label is a numerical feature
- Ranking: label is a ordinal feature
- ...

Classification

Features: color, weight
Label: taste is sweet (positive/+) or not (negative/-)



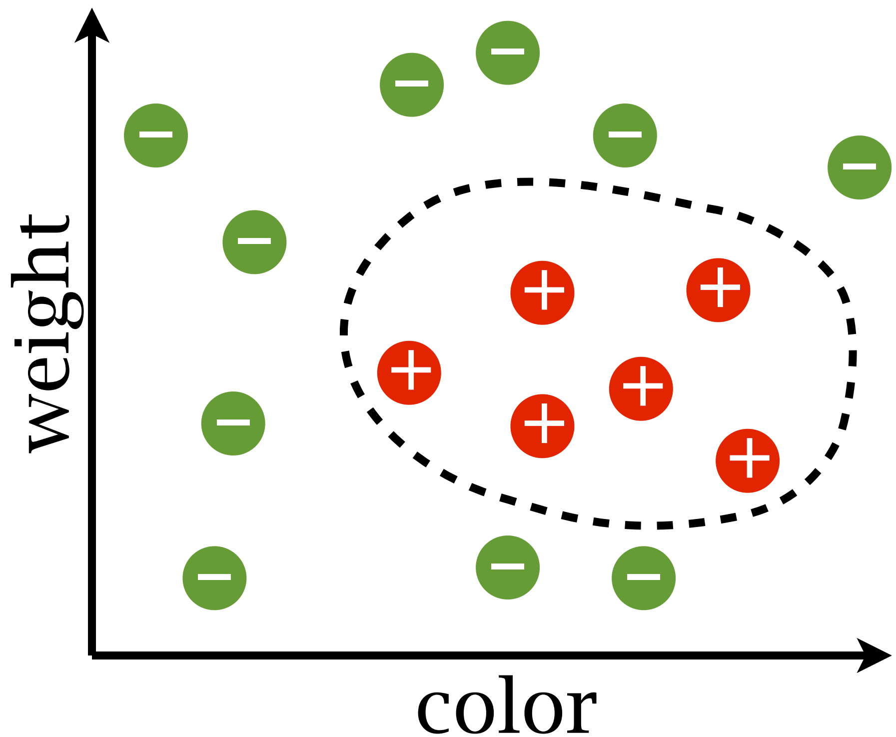
(color, weight) → sweet ?

$\mathcal{X} \quad\quad \to \{-1, +1\}$

ground-truth function f

Classification

Features: color, weight
Label: taste is sweet (positive/+) or not (negative/-)



(color, weight) → sweet ?

$\mathcal{X} \quad \to \{-1, +1\}$

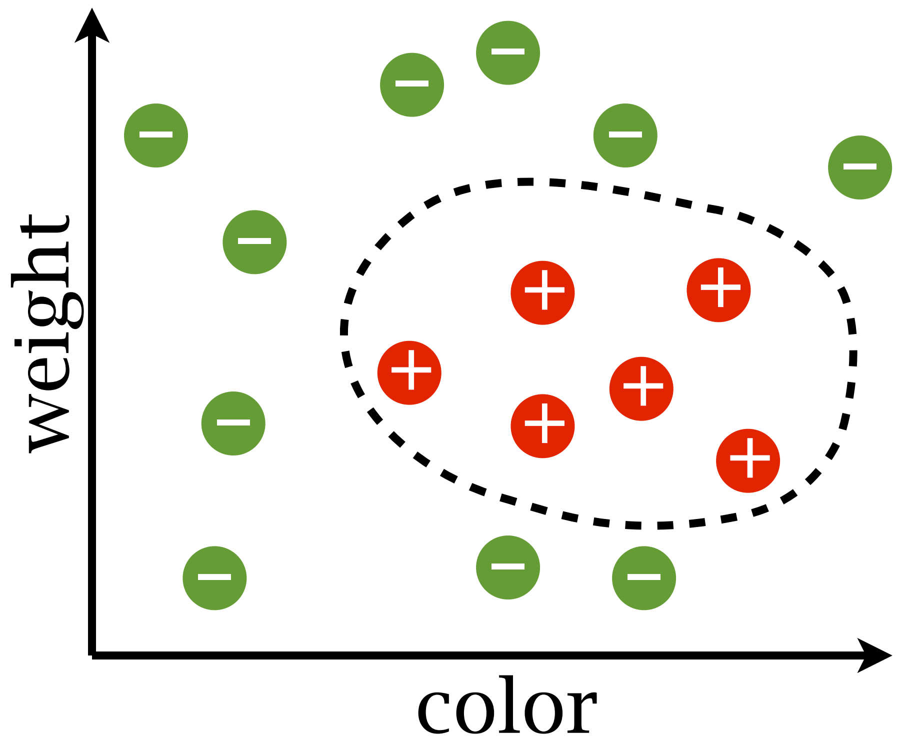
ground-truth function f

examples/training data:
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

$y_i = f(\boldsymbol{x}_i)$

Classification

Features: color, weight
Label: taste is sweet (positive/+) or not (negative/-)



(color, weight) → sweet ?

$\mathcal{X} \quad \rightarrow \{-1, +1\}$

ground-truth function f

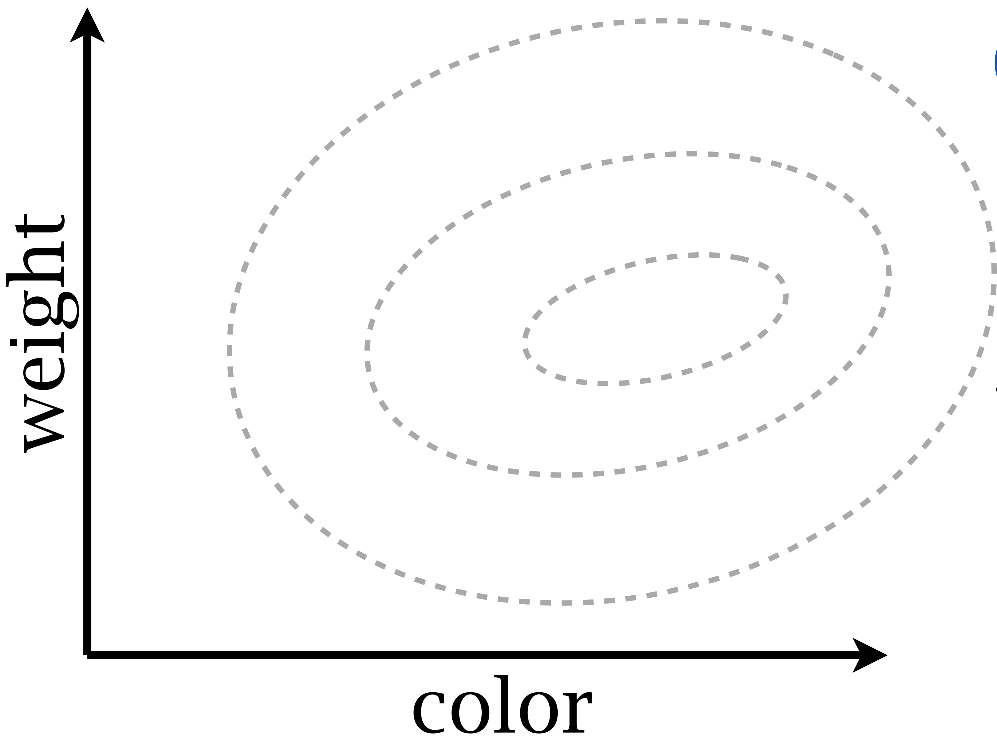
examples/training data:
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

$y_i = f(\boldsymbol{x}_i)$

learning: find an f' that is close to f

Regression

Features: color, weight
Label: price [0,1]



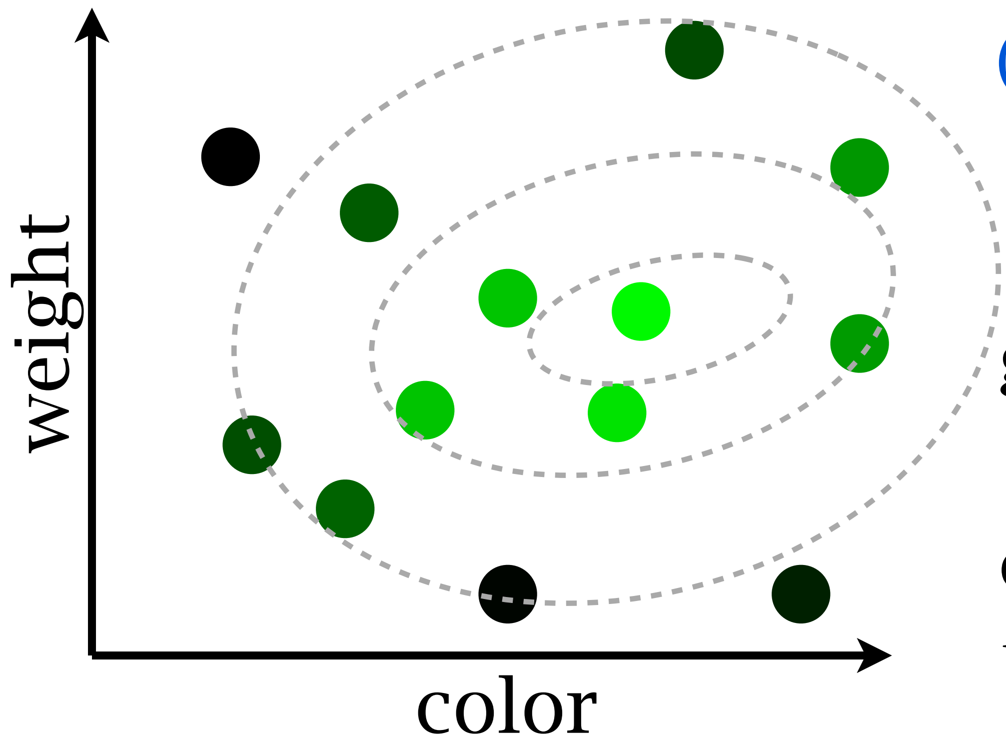
(color, weight) → price

$\mathcal{X} \rightarrow [0, +1]$

ground-truth function f

Regression

Features: color, weight
Label: price [0,1]



(color, weight) → price

$\mathcal{X} \rightarrow [0, +1]$

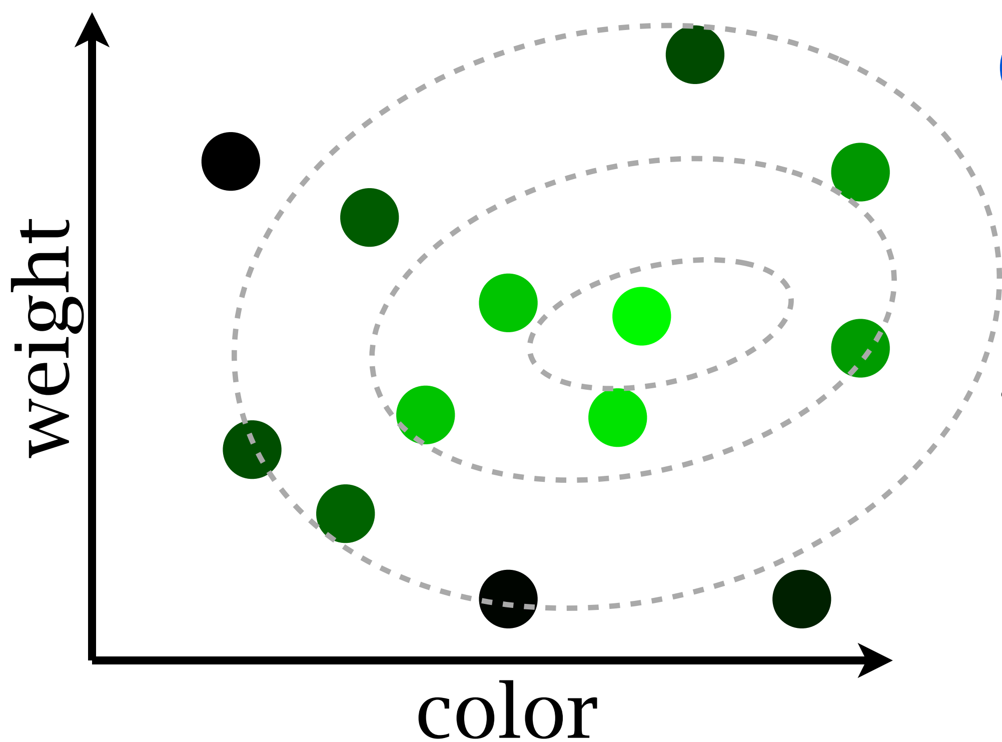
ground-truth function f

examples/training data:
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

$y_i = f(\boldsymbol{x}_i)$

Regression

Features: color, weight
Label: price [0,1]



(color, weight) → price

$\mathcal{X} \rightarrow [0, +1]$

ground-truth function f

examples/training data:
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

$y_i = f(\boldsymbol{x}_i)$

learning: find an f' that is close to f

Learning algorithms

Decision tree

Neural networks

Linear classifiers

Bayesian classifiers

Lazy classifiers

...

Why different classifiers?

- heuristics
- viewpoint
- performance

Three basic algorithms

Probabilistic Model: Naive Bayes

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

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

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

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

...

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

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

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

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

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

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

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.

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

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

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

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

Three basic algorithms

Nonparametric Model: Decision Tree

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

what the f' would be?

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

what the f' would be?

$$f' = \begin{cases} \text{sweet}, & \text{color} = \text{red} \\ \text{not-sweet}, & \text{color} \neq \text{red} \end{cases}$$

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

what the f' would be?

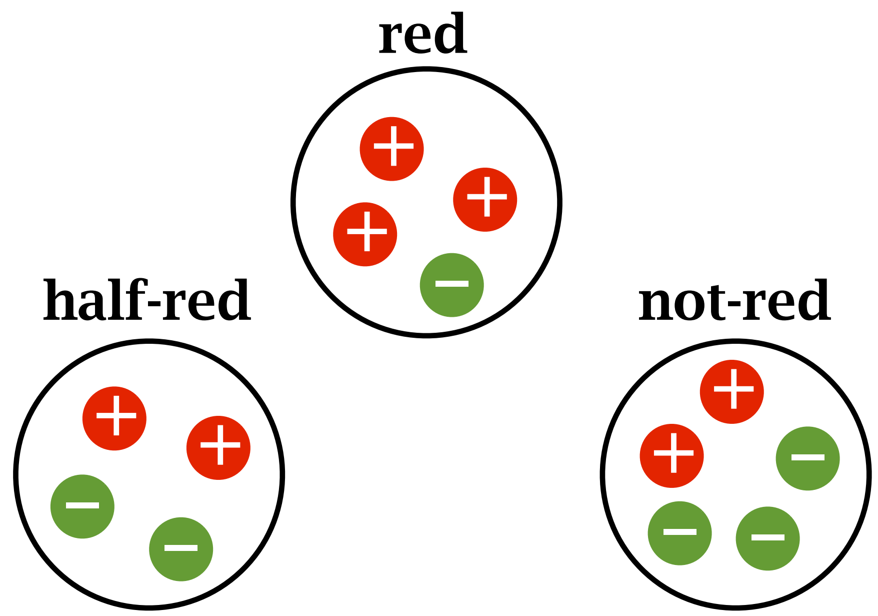
$$f' = \begin{cases} \text{sweet}, & \text{color} = \text{red} \\ \text{not-sweet}, & \text{color} \neq \text{red} \end{cases}$$

perfect
but not realistic

Consider a very simple case

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

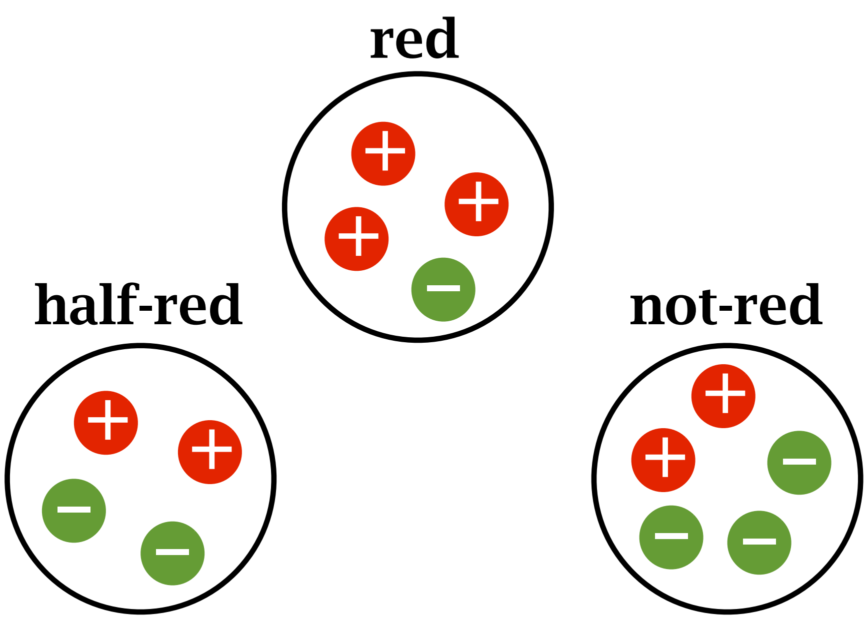
what the f' would be?



Consider a very simple case

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

what the f' would be?

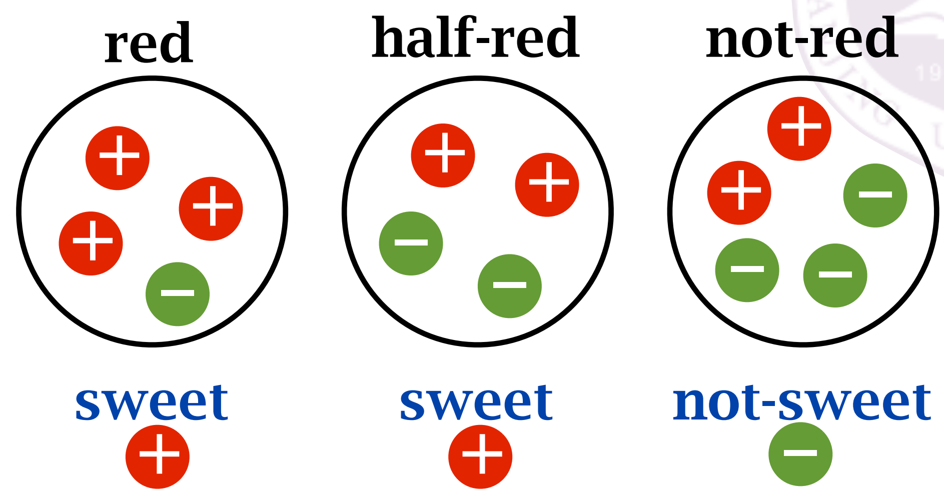


$$f' = \begin{cases} \text{sweet}, & \text{color} = \text{red} \\ \text{sweet}, & \text{color} = \text{half-red} \\ \text{not-sweet}, & \text{color} = \text{not-red} \end{cases}$$

not perfect
but how good?

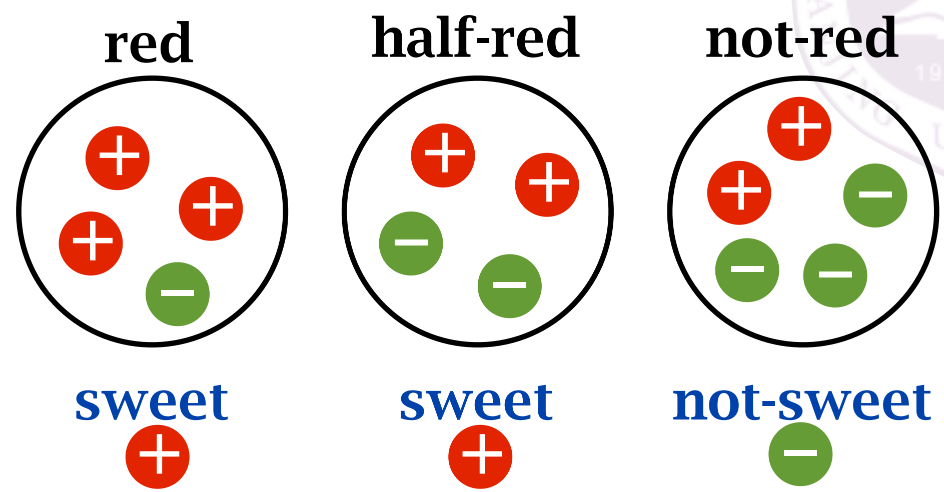
Consider a very simple case

$$f' = \begin{cases} \text{sweet}, & \text{color} = \text{red} \\ \text{sweet}, & \text{color} = \text{half-red} \\ \text{not-sweet}, & \text{color} = \text{not-red} \end{cases}$$



Consider a very simple case

$$f' = \begin{cases} \text{sweet}, & \text{color} = \text{red} \\ \text{sweet}, & \text{color} = \text{half-red} \\ \text{not-sweet}, & \text{color} = \text{not-red} \end{cases}$$



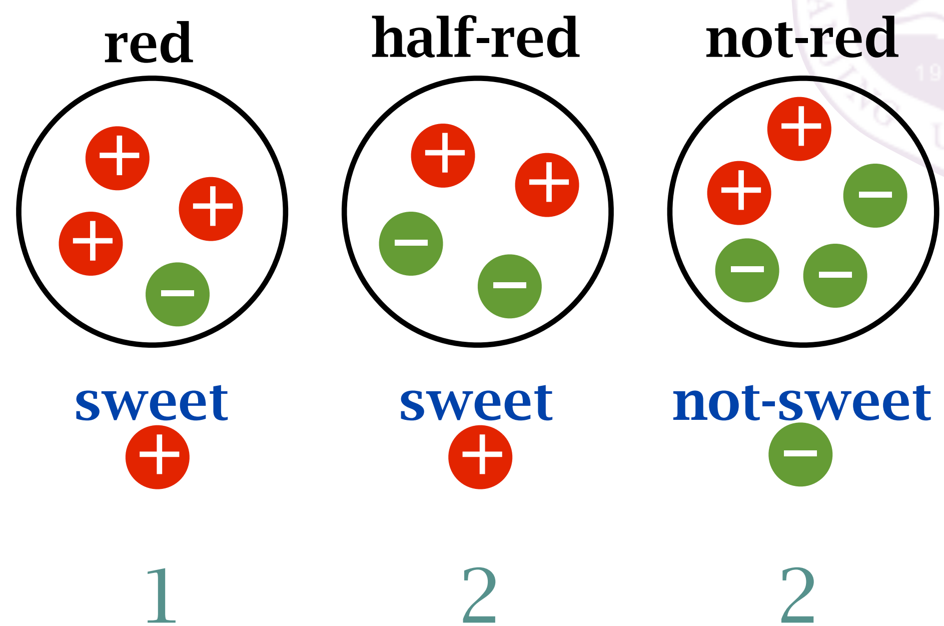
1 2 2

training error:

(1+2+2)/13=0.3846

Consider a very simple case

$$f' = \begin{cases} \text{sweet}, & \text{color} = \text{red} \\ \text{sweet}, & \text{color} = \text{half-red} \\ \text{not-sweet}, & \text{color} = \text{not-red} \end{cases}$$



training error:

(1+2+2)/13=0.3846

information gain:

entropy before split: $H(X) = -\sum_i ratio(class_i) \ln ratio(class_i) = 0.6902$

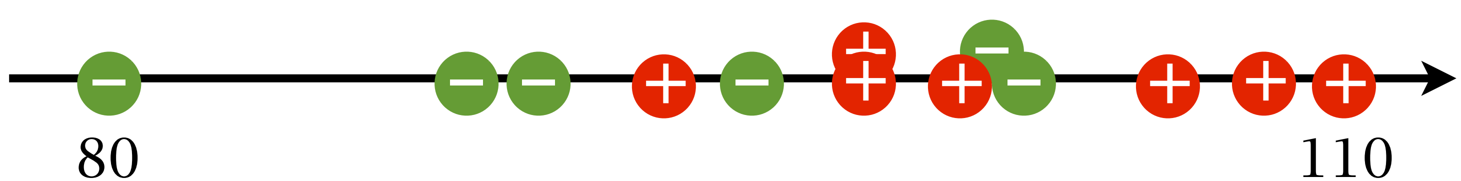
entropy after split: $I(X;\text{split}) = \sum_i ratio(split_i) H(split_i)$

information gain: $= \frac{4}{13} 0.5623 + \frac{4}{13} 0.6931 + \frac{5}{13} 0.6730 = 0.6452$

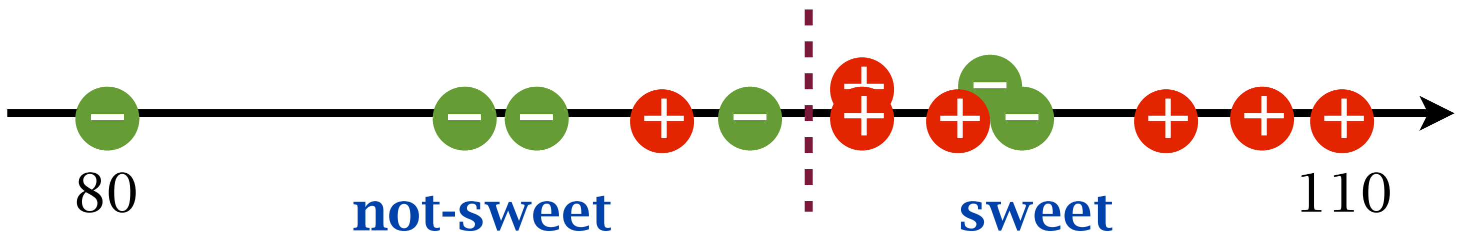
$$Gain(X; split) = H(X) - I(X;\text{split}) = 0.045$$

A little more complex case

id	color	weight	taste
1	[redacted]	110	sweet
2	[redacted]	105	sweet
3	[redacted]	100	sweet
4	[redacted]	93	sweet
5	[redacted]	80	not-sweet
6	[redacted]	98	sweet
7	[redacted]	95	not-sweet
8	[redacted]	102	not-sweet
9	[redacted]	98	sweet
10	[redacted]	90	not-sweet
11	[redacted]	108	sweet
12	[redacted]	101	not-sweet
13	[redacted]	89	not-sweet



A little more complex case



for every split point

training error:

(1+2)/13=0.2307

information gain:

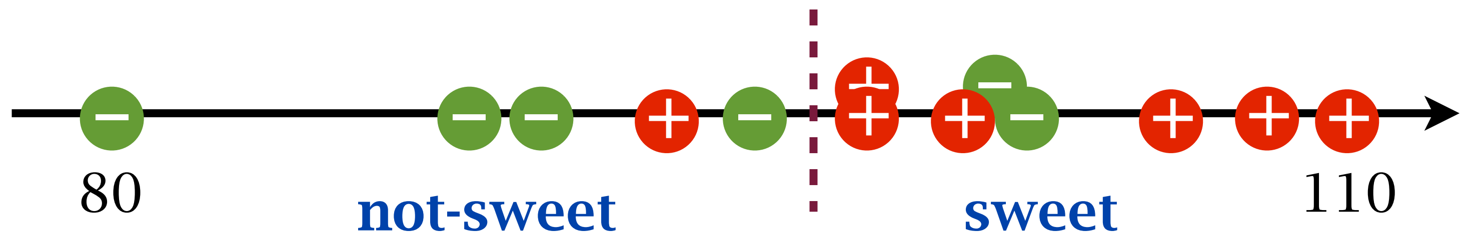
$$H(X) = -\sum_i ratio(class_i) \ln ratio(class_i) = 0.6902$$

$$I(X;\text{split}) = \sum\nolimits_i ratio(split_i) H(split_i)$$

$$= \frac{5}{13} 0.5004 + \frac{8}{13} 0.5623 = 0.5385$$

$$Gain(X;split) = H(X) - I(X;\text{split}) = 0.1517$$

A little more complex case



for every split point

training error:

(1+2)/13=0.2307

information gain:

entropy before split: $H(X) = -\sum_i ratio(class_i) \ln ratio(class_i) = 0.6902$

entropy after split: $I(X;\text{split}) = \sum_i ratio(split_i) H(split_i)$

$$= \frac{5}{13} 0.5004 + \frac{8}{13} 0.5623 = 0.5385$$

information gain:

$$Gain(X; split) = H(X) - I(X; \text{split}) = 0.1517$$

A little more complex case

id	color	weight	taste
1	red	110	sweet
2	red	105	sweet
3	half-red	100	sweet
4	not-red	93	sweet
5	not-red	80	not-sweet
6	half-red	98	sweet
7	red	95	not-sweet
8	not-red	102	not-sweet
9	not-red	98	sweet
10	half-red	90	not-sweet
11	red	108	sweet
12	half-red	101	not-sweet
13	not-red	89	not-sweet

color v.s. best split of weight

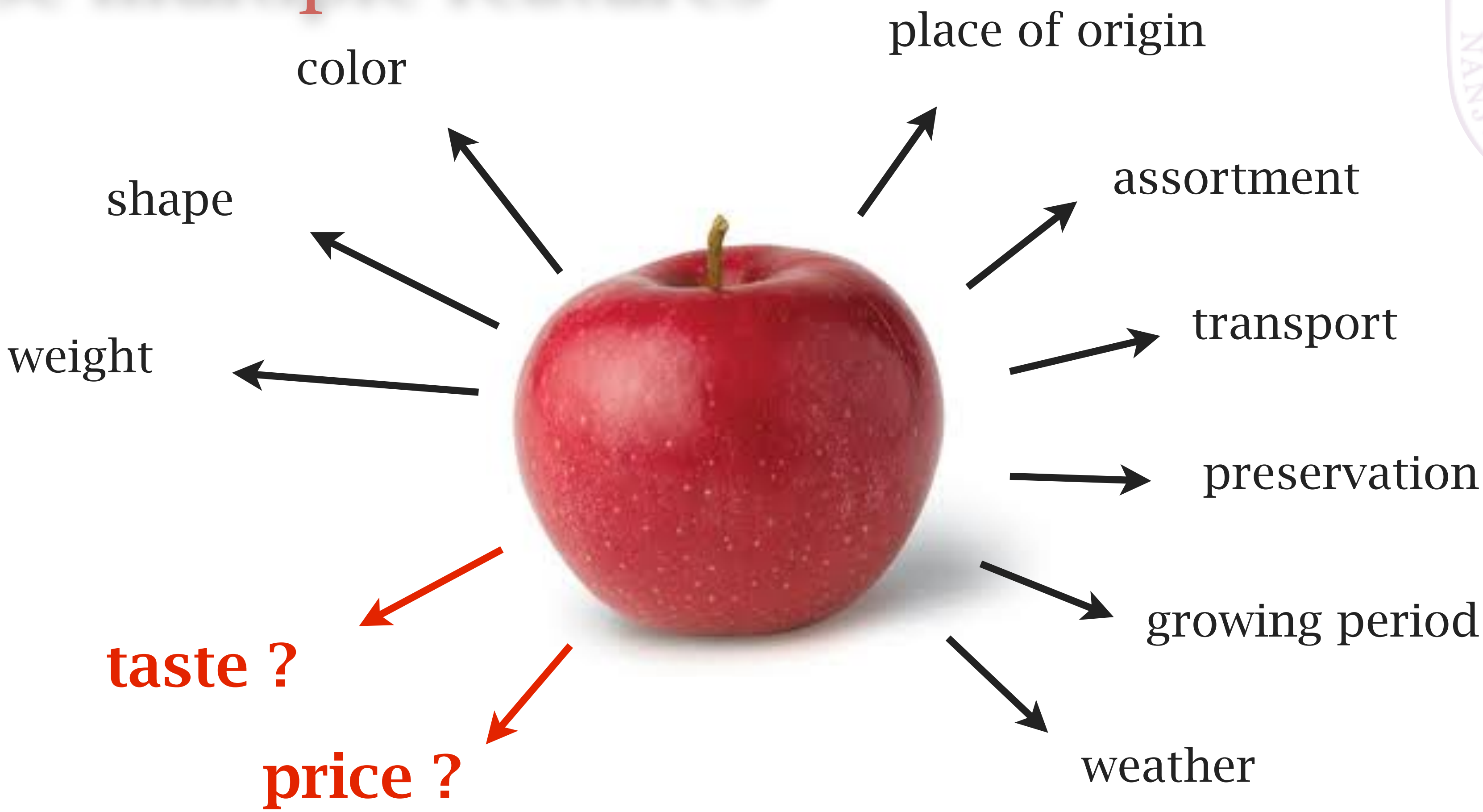
$$f' = \begin{cases} \text{sweet}, & \text{color} = \text{red} \\ \text{sweet}, & \text{color} = \text{half-red} \\ \text{not-sweet}, & \text{color} = \text{not-red} \end{cases}$$

$$f' = \begin{cases} \text{sweet}, & \text{weight} > 95 \\ \text{not-sweet}, & \text{weight} \leq 95 \end{cases}$$

what the f' would be?

the best split among all features

Use multiple features

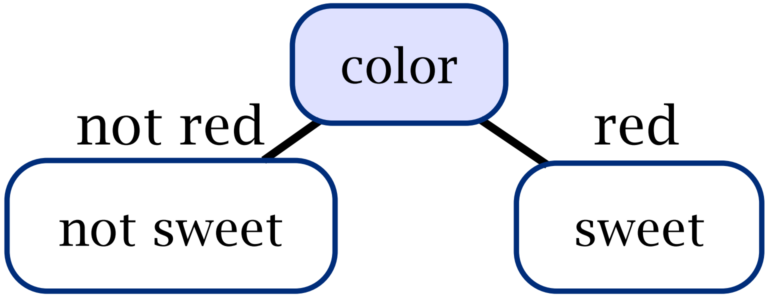


find a model by find the best feature/best split

but only one feature/split is used

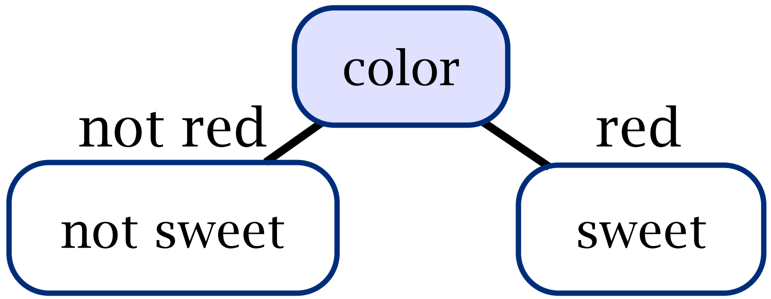
Use multiple features

one feature model: decision stump

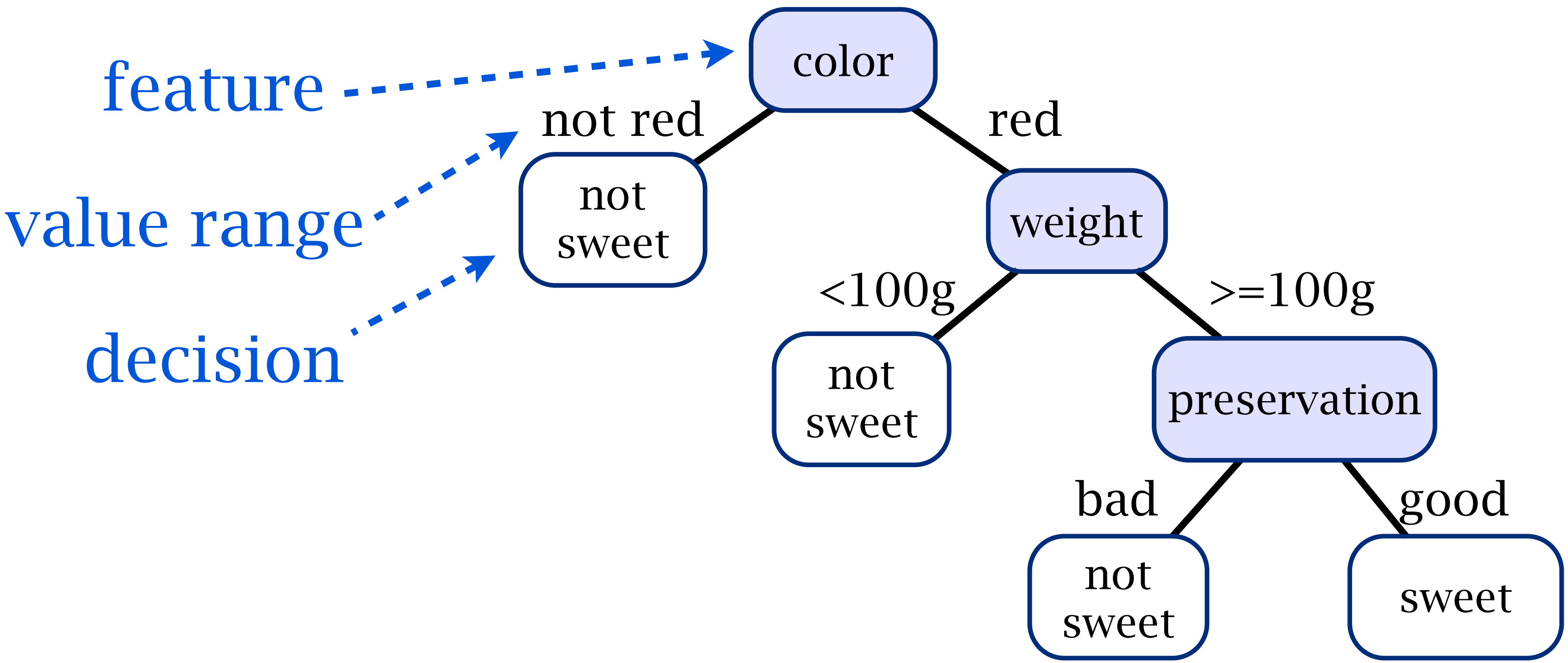


Use multiple features

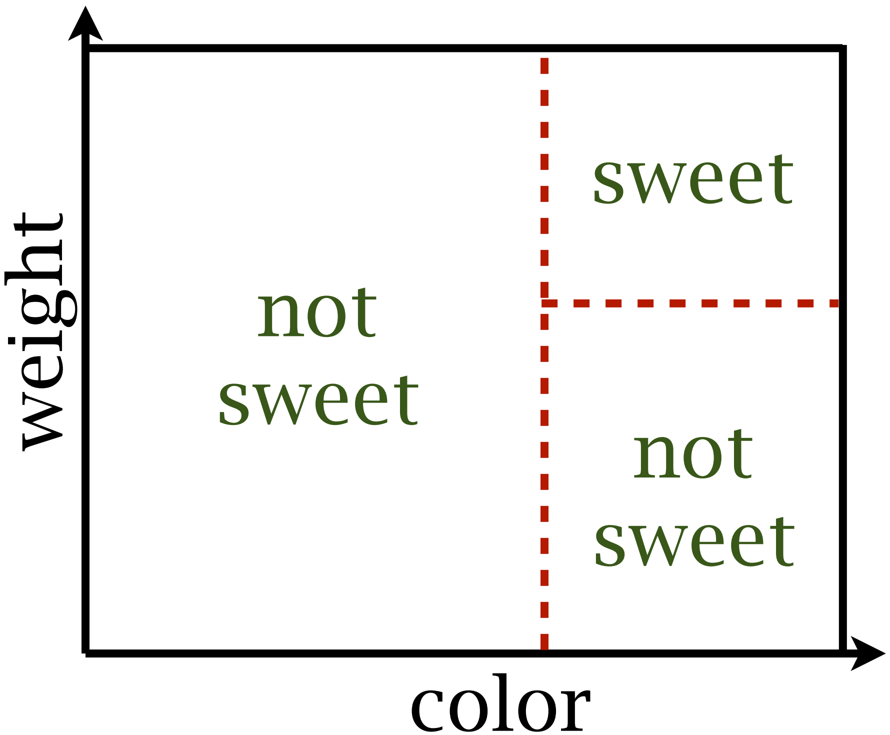
one feature model: decision stump

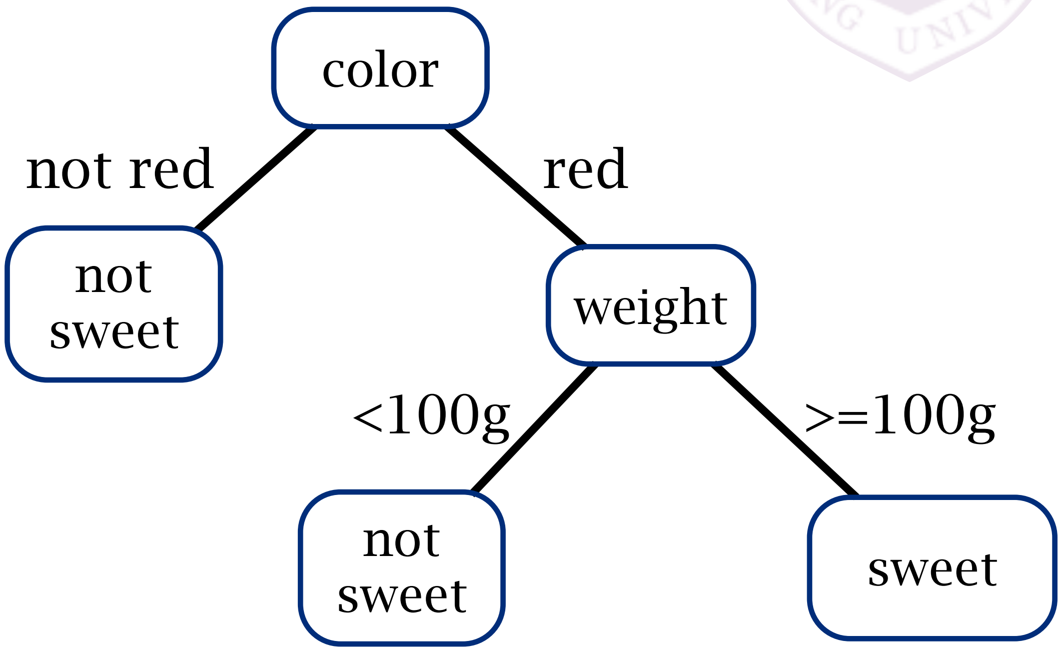


hierarchical model uses many features: decision tree

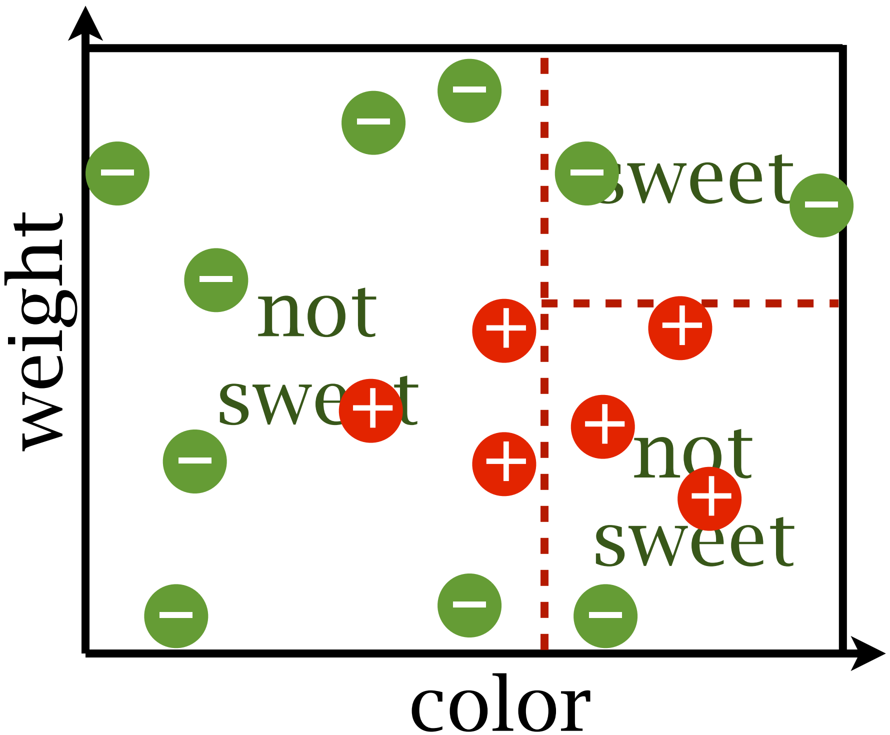


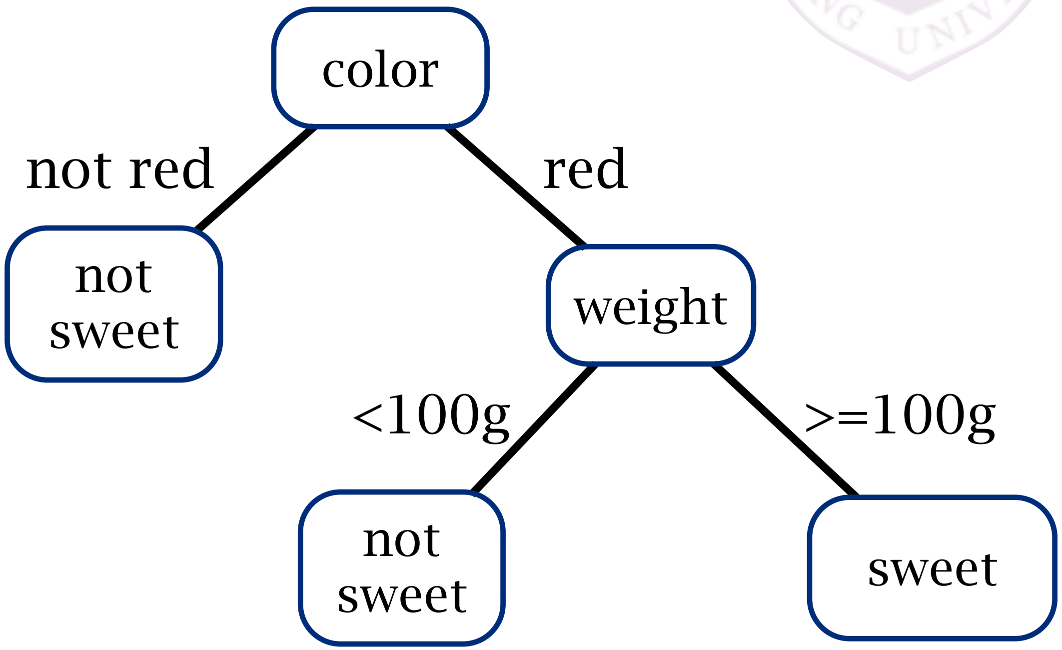
Decision tree model





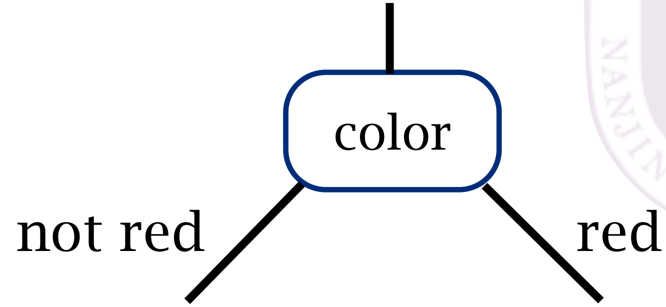
Decision tree model





find a decision tree that matches the data

Top-down induction



function construct-node(*data*) :

1. *feature*, *value* ←**split-criterion** (*data*)
2. if feature is valid
3. *subdata*[] ← split(*data*, *feature*, *value*)
4. for each branch *i*
5. **construct-node** (*subdata*[*i*])
6. else
7. **make a leaf**
8. return

divide and conquer

Decision tree learning algorithms

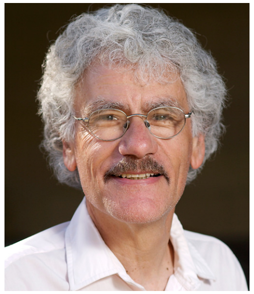
ID3: information gain

C4.5: gain ratio, handling missing values

Ross Quinlan

CART: gini index

Leo Breiman 1928-2005



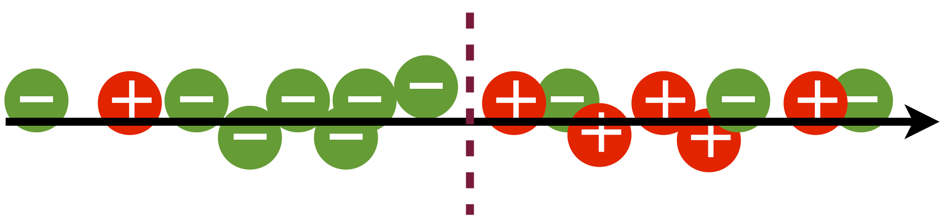
Jerome H. Friedman

Gini index

Gini index (CART):

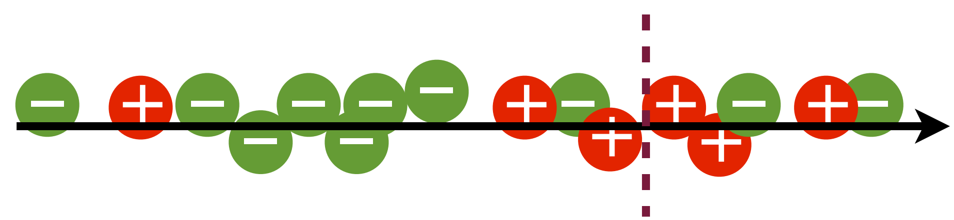
Gini: $Gini(X) = 1 - \sum_i p_i^2$

Gini after split: $\frac{\#\text{left}}{\#\text{all}} Gini(\text{left}) + \frac{\#\text{right}}{\#\text{all}} Gini(\text{right})$



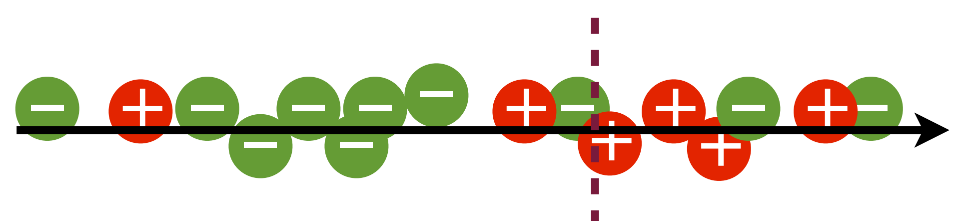
$\text{IG} = H(X) - 0.5192$

$Gini = 0.3438$



$\text{IG} = H(X) - 0.6132$

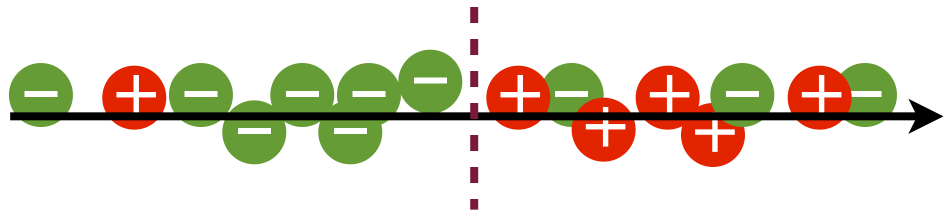
$Gini = 0.4427$

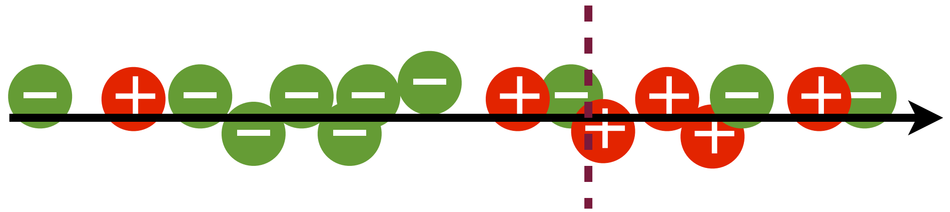


$\text{IG} = H(X) - 0.5514$

$Gini = 0.3667$

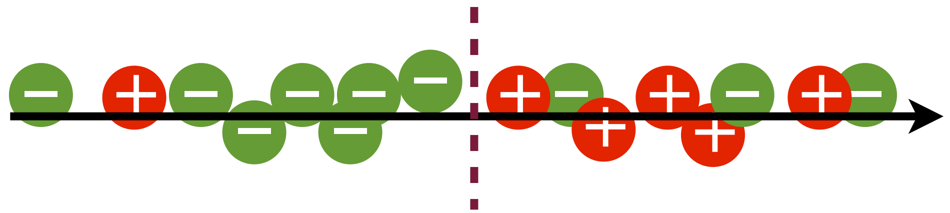
Training error v.s. Information gain



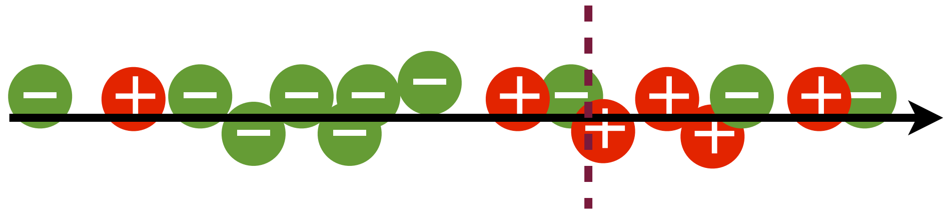


training error is less smooth

Training error v.s. Information gain



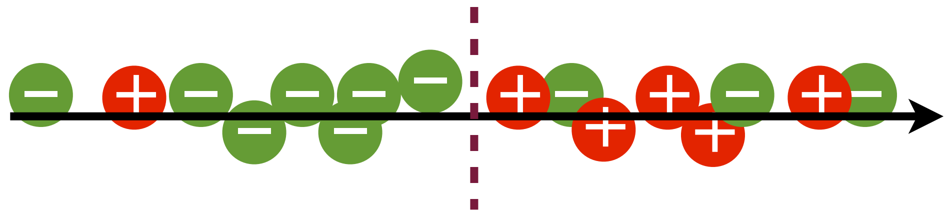
training error: 4



training error: 4

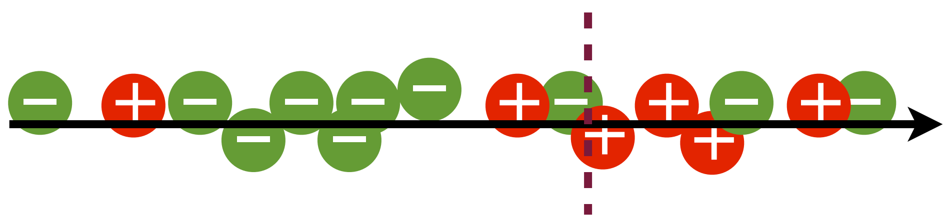
training error is less smooth

Training error v.s. Information gain



training error: 4

information gain: $\mathrm{IG} = H(X) - 0.5192$



training error: 4

information gain: $\mathrm{IG} = H(X) - 0.5514$

training error is less smooth

Non-generalizable feature

id	color	weight	taste
1	red	110	sweet
2	red	105	sweet
3	half-red	100	sweet
4	not-red	93	sweet
5	not-red	80	not-sweet
6	half-red	98	sweet
7	red	95	not-sweet
8	not-red	102	not-sweet
9	not-red	98	sweet
10	half-red	90	not-sweet
11	red	108	sweet
12	half-red	101	not-sweet
13	not-red	89	not-sweet

the system may not know non-generalizable features

$$\mathrm{IG} = H(X) - 0$$

Non-generalizable feature

id	color	weight	taste
1	red	110	sweet
2	red	105	sweet
3	half-red	100	sweet
4	not-red	93	sweet
5	not-red	80	not-sweet
6	half-red	98	sweet
7	red	95	not-sweet
8	not-red	102	not-sweet
9	not-red	98	sweet
10	half-red	90	not-sweet
11	red	108	sweet
12	half-red	101	not-sweet
13	not-red	89	not-sweet

the system may not know non-generalizable features

$$\mathrm{IG} = H(X) - 0$$

Gain ratio as a correction:

$$\text{Gain ratio}(X) = \frac{H(X) - I(X;\text{split})}{IV(\text{split})}$$

$$IV(\text{split}) = H(\text{split})$$

A regression case

color

weight

→ price ?

id	color	weight	price
1	red	110	12
2	red	105	10
3	half-red	100	10
4	not-red	93	15
5	not-red	80	5
6	half-red	98	8
7	red	95	8
8	not-red	102	9
9	not-red	98	6
10	half-red	90	7
11	red	108	11
12	half-red	101	12
13	not-red	89	6

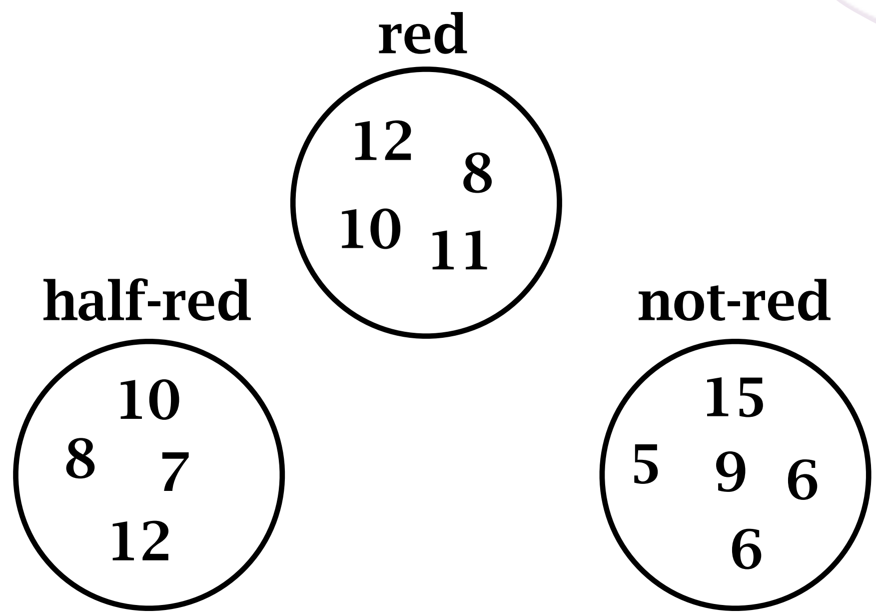
what the f' would be to minimize:

$$MSE = \frac{1}{n}\sum_i (f(x_i) - f'(x_i))^2$$

A regression case

id	color	weight	price
1	red	110	12
2	red	105	10
3	half-red	100	10
4	not-red	93	15
5	not-red	80	5
6	half-red	98	8
7	red	95	8
8	not-red	102	9
9	not-red	98	6
10	half-red	90	7
11	red	108	11
12	half-red	101	12
13	not-red	89	6

for *color* feature:



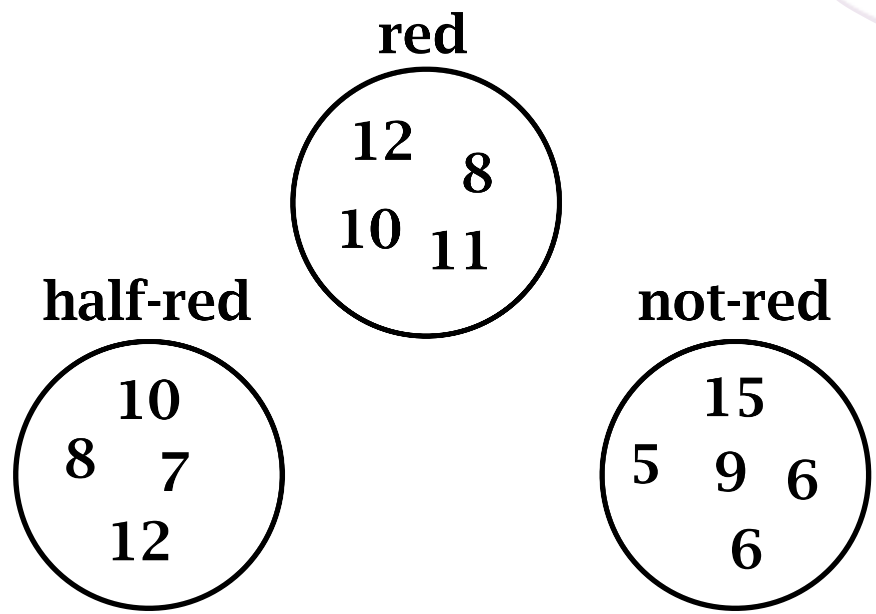
what is the prediction value of each color to minimize the mean square error?

$$MSE = \frac{1}{n}\sum_i (f(x_i) - f'(x_i))^2$$

A regression case

id	color	weight	price
1	red	110	12
2	red	105	10
3	half-red	100	10
4	not-red	93	15
5	not-red	80	5
6	half-red	98	8
7	red	95	8
8	not-red	102	9
9	not-red	98	6
10	half-red	90	7
11	red	108	11
12	half-red	101	12
13	not-red	89	6

for *color* feature:



what is the prediction value of each color to minimize the mean square error?

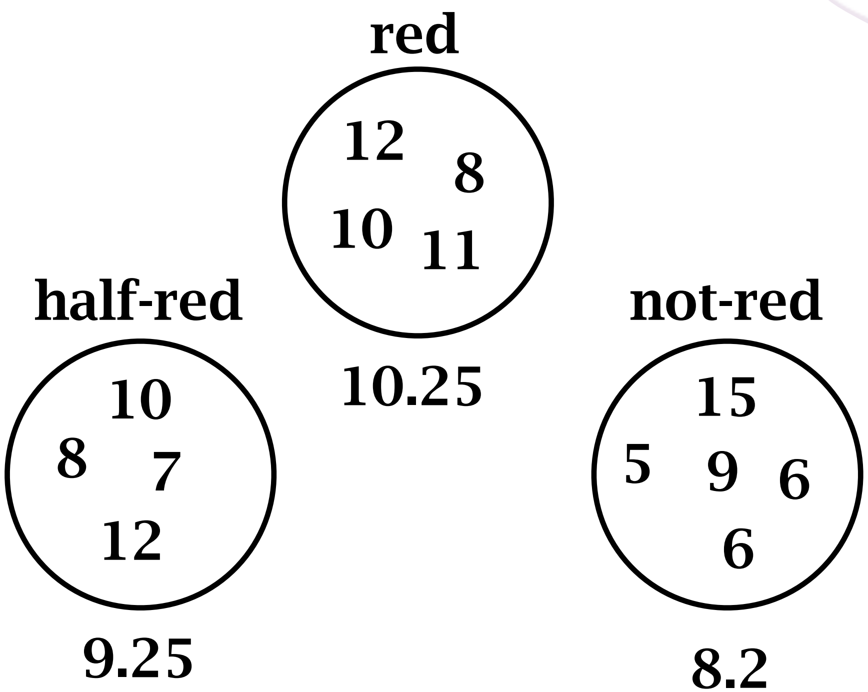
$$MSE = \frac{1}{n}\sum_i (f(x_i) - f'(x_i))^2$$

mean value

A regression case

id	color	weight	price
1	red	110	12
2	red	105	10
3	half-red	100	10
4	not-red	93	15
5	not-red	80	5
6	half-red	98	8
7	red	95	8
8	not-red	102	9
9	not-red	98	6
10	half-red	90	7
11	red	108	11
12	half-red	101	12
13	not-red	89	6

for *color* feature:

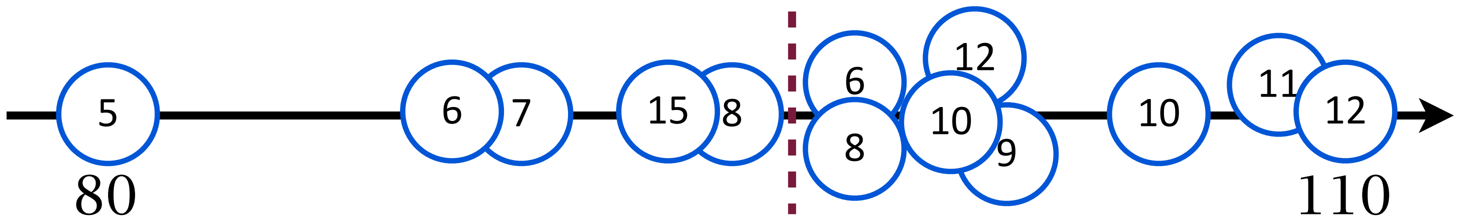


$$f' = \begin{cases} 10.25, & \text{color} = \text{red} \\ 9.25, & \text{color} = \text{half-red} \\ 8.2, & \text{color} = \text{not-red} \end{cases}$$

A regression case

for *weight* feature:

for any split:



mean: 8.2 mean: 9.75

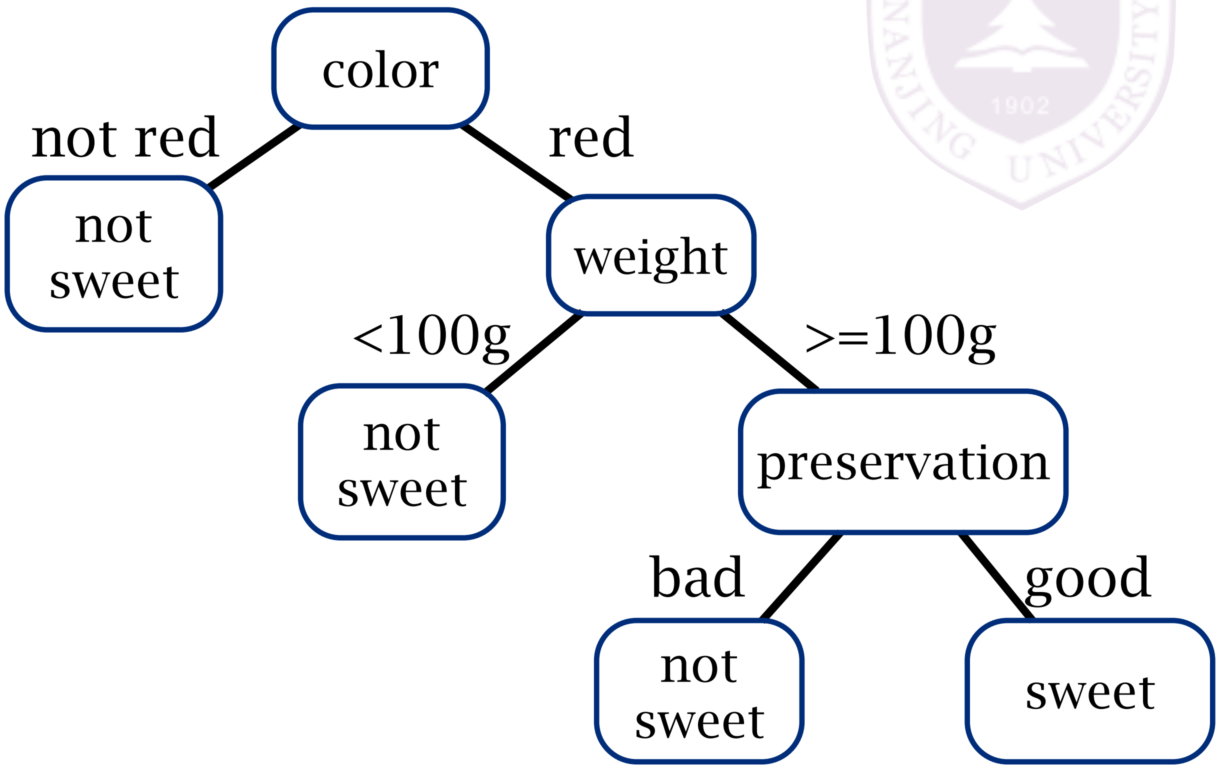
$$f' = \begin{cases} 9.75, & \text{weight} > 95 \\ 8.2, & \text{weight} \leq 95 \end{cases}$$

MSE: 12.56 MSE: 3.6875

overall MSE: 7.1

choose the split with minimal MSE

Split-criterion: stop



Stop criterion:
no feature to use

Classification: examples are pure of class

Regression: MSE small enough

Three basic algorithms

Linear Model: Logistic Regression

Linear model

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

Linear model

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

$$w_1, w_2, \ldots, w_n \quad b$$

$$\Downarrow$$

$$w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n + b$$

Linear model

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

$$\boldsymbol{w} = \; w_1, w_2, \ldots, w_n \qquad b$$

$$w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n + b$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$$

Linear model

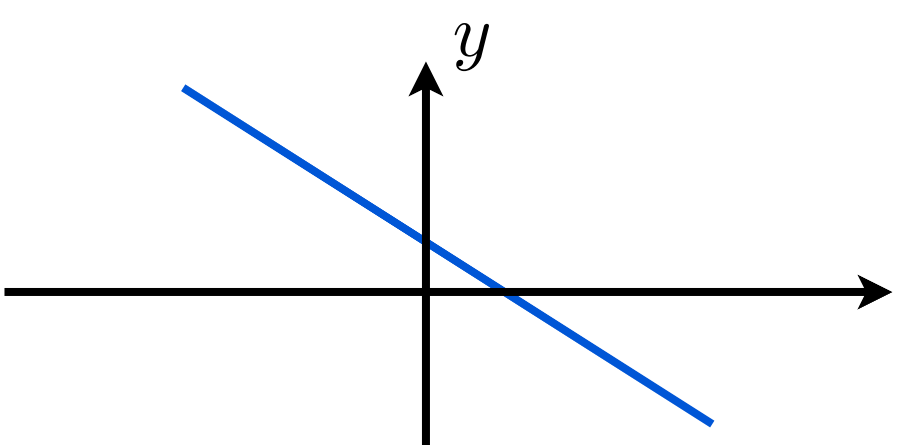
$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

$$\boldsymbol{w} = \ w_1, w_2, \ldots, w_n \quad b$$

$$w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n + b$$

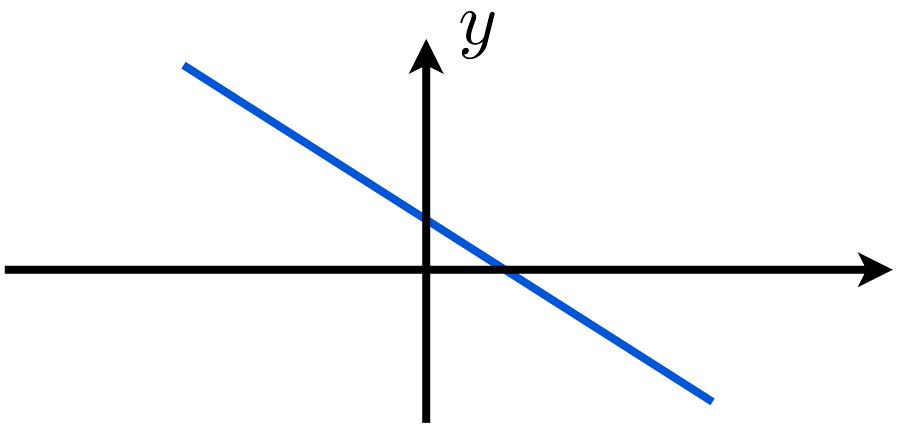
$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$$

$y = ax + b$

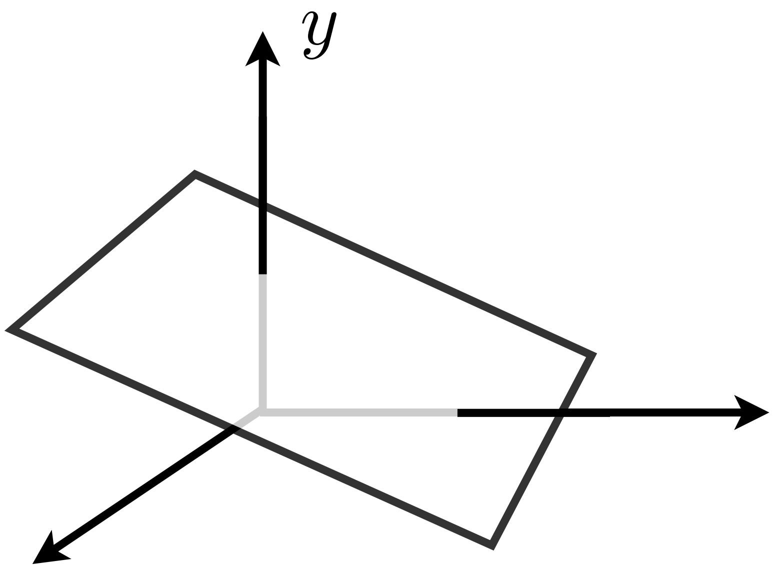


Linear model

$$y = ax + b$$

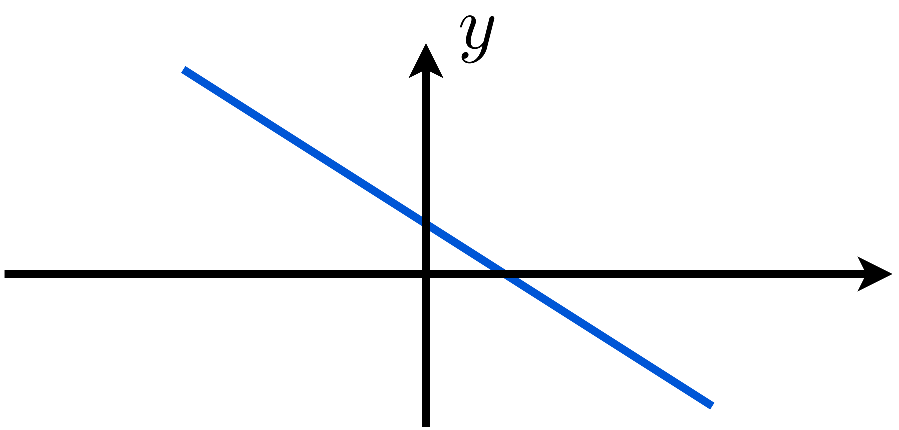


$$y = w_1 \cdot x_1 + w_2 \cdot x_2 + b$$

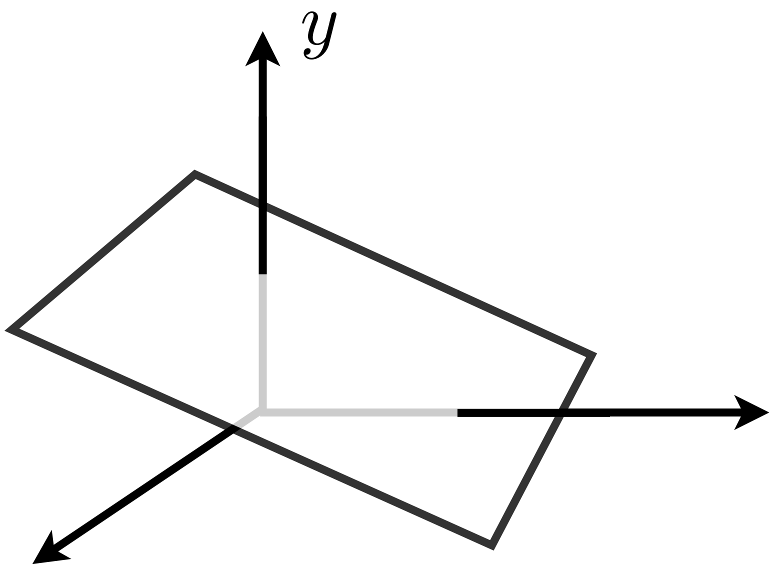


Linear model

$$y = ax + b$$



$$y = w_1 \cdot x_1 + w_2 \cdot x_2 + b$$



is the following a linear model?

$$y = w_1 \cdot x + w_2 \cdot x^2 + b$$

Linear model

y ⟷ x_1, x_2, $\dots$, x_n

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$$

output/response variable

linear relationship
independent parameters

basis

model space: $\mathbb{R}^{n+1}$

we sometimes omit the bias

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$$

1. $\boldsymbol{x}$ is with a constant element
2. practically as good as with bias (centered data)

Linear classifier

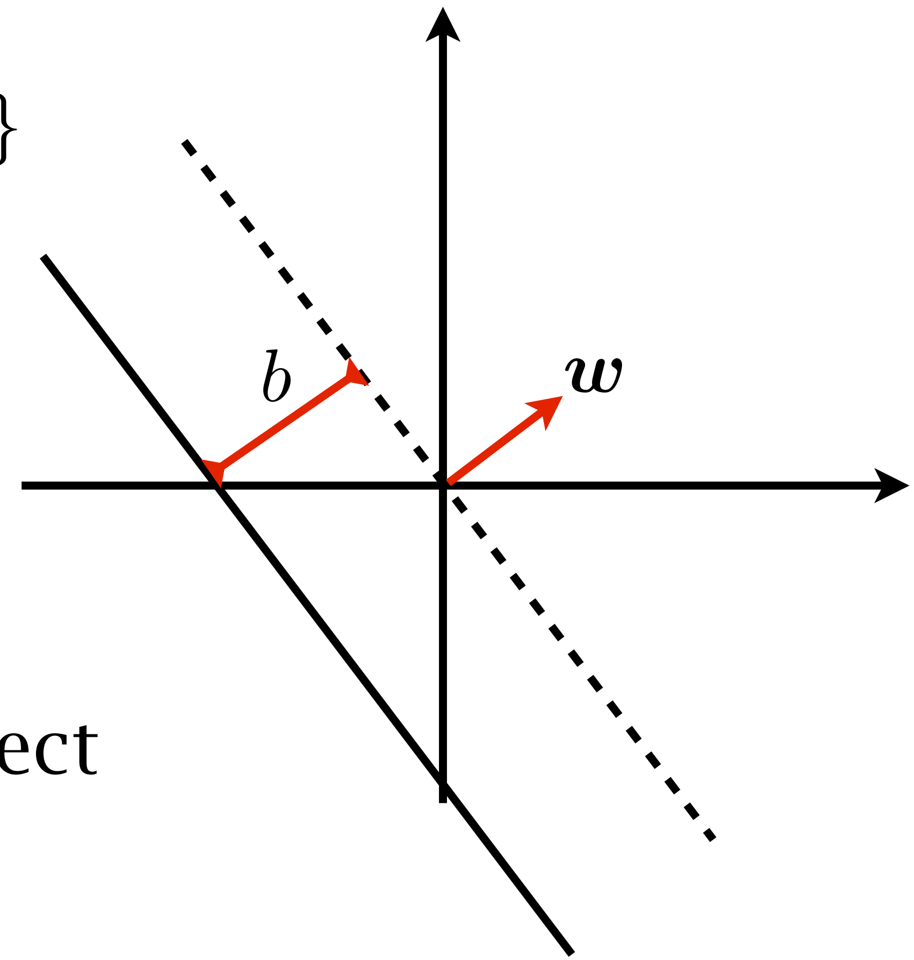
model space: $\mathbb{R}^{n+1}$

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$$

for classification $y \in \{-1, +1\}$

we predict an instance by

$$\text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b) = \begin{cases} +1, & \boldsymbol{w}^\top \boldsymbol{x} + b > 0 \\ -1, & \boldsymbol{w}^\top \boldsymbol{x} + b < 0 \\ \text{random}, & otherwise \end{cases}$$



for an example $(\boldsymbol{x}, y)$, a correct prediction means

$$y(\boldsymbol{w}^\top \boldsymbol{x} + b) > 0$$

Prototype

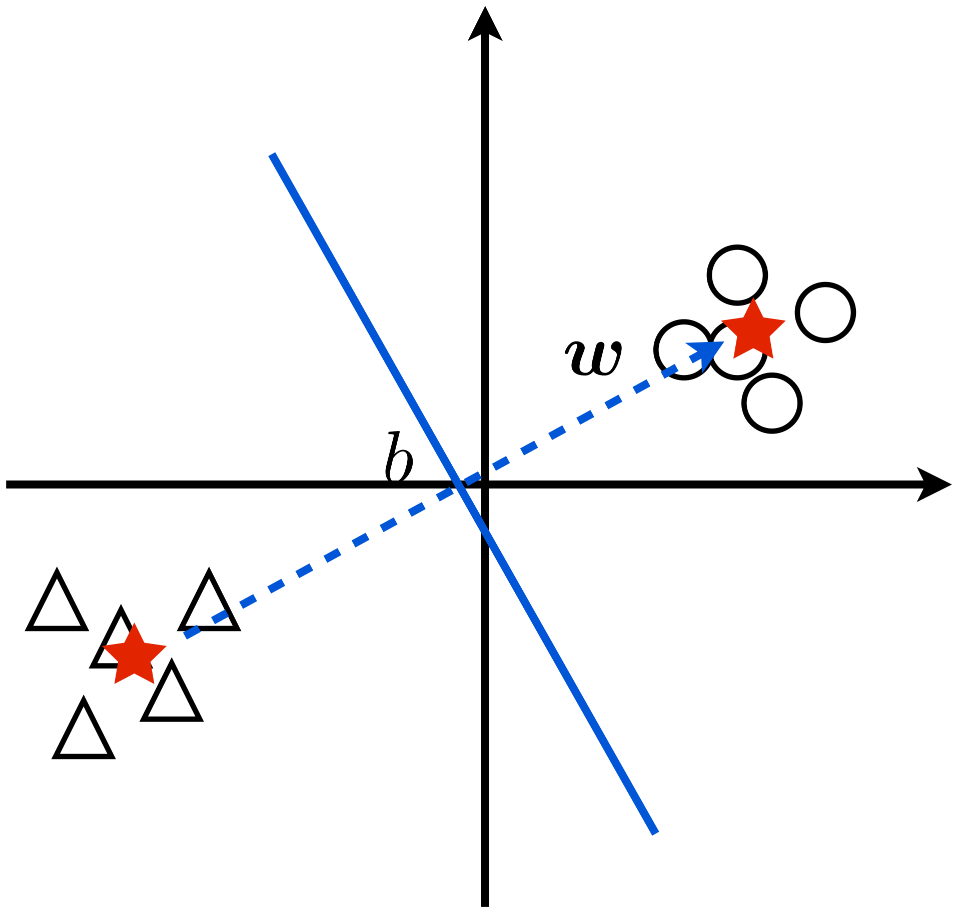
simple, but too restricted

$$\bar{\boldsymbol{x}}^+ = \frac{1}{\sum_{i:y_i=+1} 1} \sum_{i:y_i=+1} \boldsymbol{x}_i$$

$$\bar{\boldsymbol{x}}^- = \frac{1}{\sum_{i:y_i=-1} 1} \sum_{i:y_i=-1} \boldsymbol{x}_i$$

$$\boldsymbol{w} = \bar{\boldsymbol{x}}^+ - \bar{\boldsymbol{x}}^-$$

$$b = -\boldsymbol{w}^\top \cdot \frac{\bar{\boldsymbol{x}}^+ + \bar{\boldsymbol{x}}^-}{2}$$

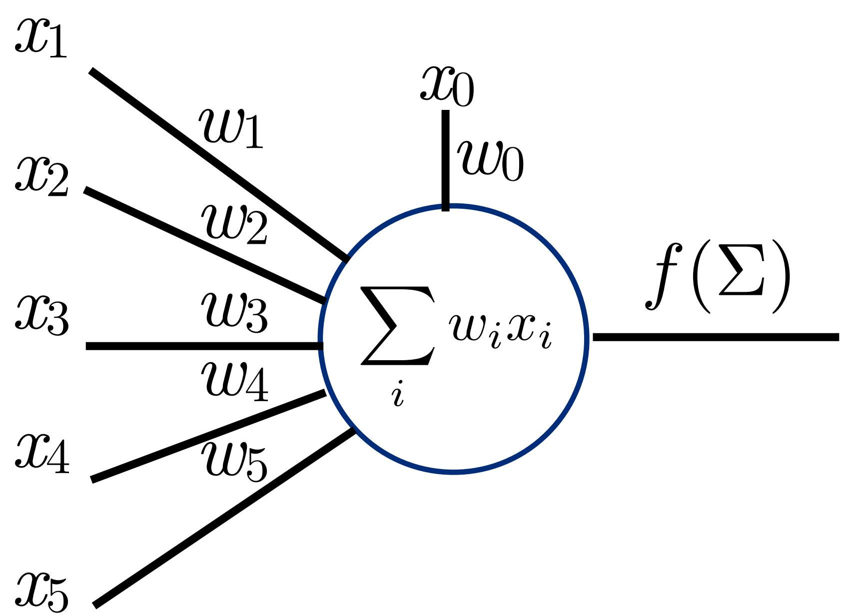


Perceptron

feed training examples one by one

1. $\boldsymbol{w} = 0$
2. for each example $(\boldsymbol{x}, y)$
   if $\text{sign}(y\boldsymbol{w}^\top \boldsymbol{x}) < 0$

$$\boldsymbol{w} = \boldsymbol{w} + y\boldsymbol{x}$$



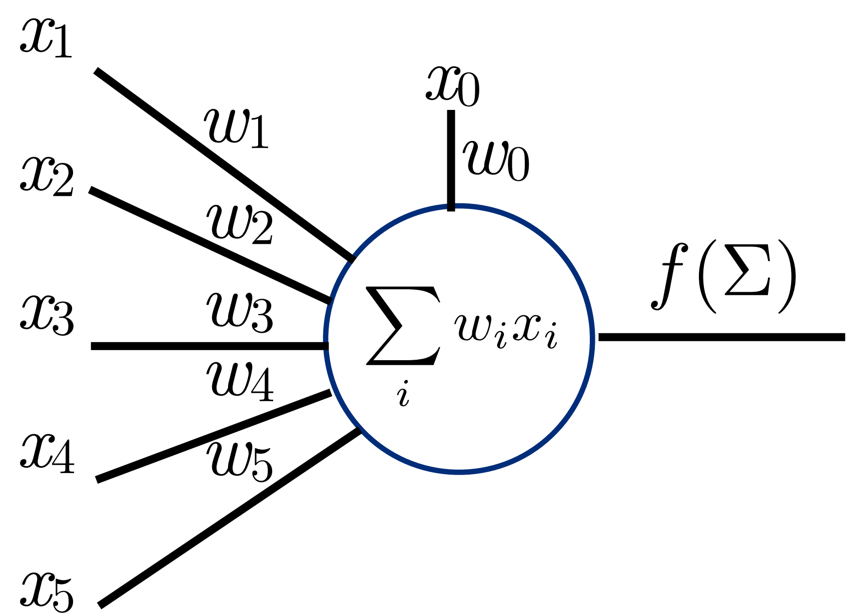
$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$$

Perceptron

feed training examples one by one

1. $\boldsymbol{w} = 0$
2. for each example $(\boldsymbol{x}, y)$
   if $\text{sign}(y\boldsymbol{w}^\top \boldsymbol{x}) < 0$

$$\boldsymbol{w} = \boldsymbol{w} + y\boldsymbol{x}$$



$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$$

gradient ascent

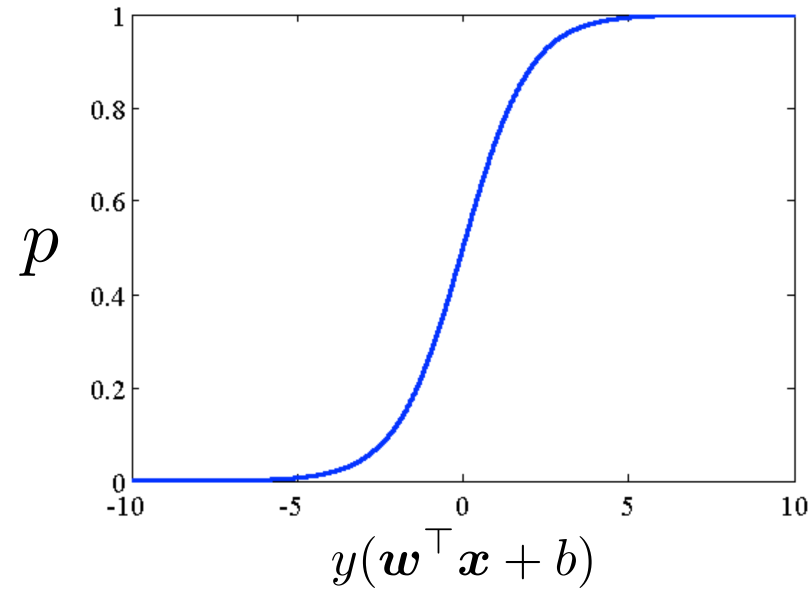
$$\frac{\partial y\boldsymbol{w}^\top \boldsymbol{x}}{\partial \boldsymbol{w}} = y\boldsymbol{x}$$

Logistic regression

assume logit model: for a positive example

$$\boldsymbol{w}^\top \boldsymbol{x} = \log \frac{p(+1 \mid \boldsymbol{x})}{1 - p(+1 \mid \boldsymbol{x})}$$

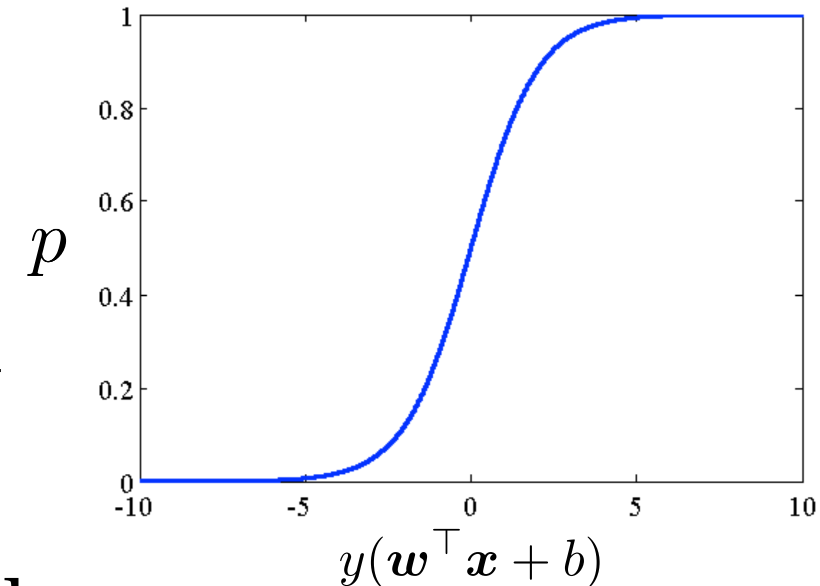
so that $p(y \mid \boldsymbol{x}, \boldsymbol{w}) = \dfrac{1}{1 + e^{-y(\boldsymbol{w}^\top \boldsymbol{x})}}$



Logistic regression

assume logit model: for a positive example

$$\boldsymbol{w}^\top \boldsymbol{x} = \log \frac{p(+1 \mid \boldsymbol{x})}{1 - p(+1 \mid \boldsymbol{x})}$$

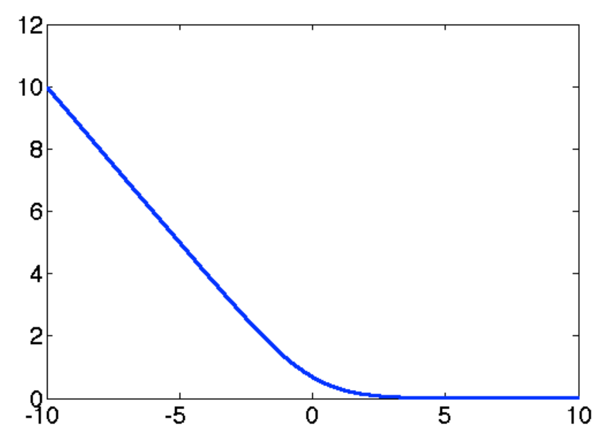


so that $p(y \mid \boldsymbol{x}, \boldsymbol{w}) = \dfrac{1}{1 + e^{-y(\boldsymbol{w}^\top \boldsymbol{x})}}$

minimize negative log-likelihood:

$$\arg\min_{\boldsymbol{w}} - \log \prod_{i=1}^{m} p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) = -\sum_i \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w})$$

$$= \sum_i \log \left(1 + e^{-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i)}\right)$$



convex

Optimization

objective function:

$$\arg\min_{\boldsymbol{w}} \sum_i \log\left(1 + e^{-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i)}\right)$$

general optimization: gradient descent

$$\boldsymbol{w} = \boldsymbol{w} - \eta \frac{\partial \sum_i \log\left(1 + e^{-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i)}\right)}{\partial \boldsymbol{w}}$$

Optimization

objective function:

$$\arg\min_{\boldsymbol{w}} \sum_i \log\left(1 + e^{-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i)}\right)$$

general optimization: gradient descent

$$\boldsymbol{w} = \boldsymbol{w} - \eta \frac{\partial \sum_i \log\left(1 + e^{-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i)}\right)}{\partial \boldsymbol{w}}$$

cheaper optimization: stochastic gradient descent

$$\boldsymbol{w} = \boldsymbol{w} - \eta \frac{\partial \log\left(1 + e^{-y(\boldsymbol{w}^\top \boldsymbol{x})}\right)}{\partial \boldsymbol{w}}$$

习题

监督学习的目标是否是最小化训练误差?

朴素贝叶斯假设是指数据的属性之间相互独立?

对于分类问题，当训练数据没有冲突时，决策树学习算法是否一定能取得0训练错误率? （冲突样本：两个完全相同的样本却被标记为不同类别）

决策树学习算法是否需要训练样本规范化(normalization)?

Logistic regression是用于回归还是分类?

视频

Chapter 5