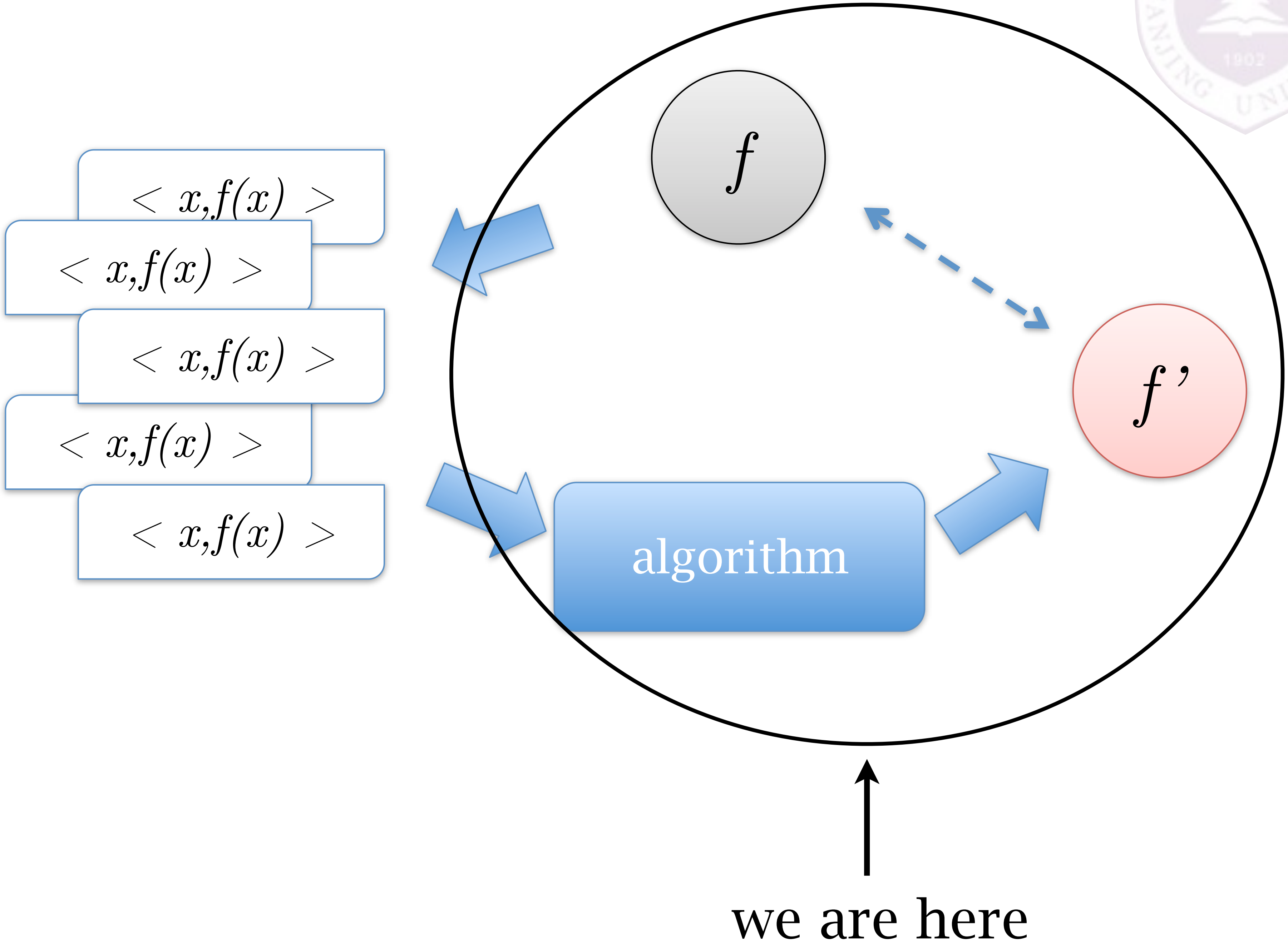
Lecture 3: Machine Learning I

Supervised Learning & Decision Tree

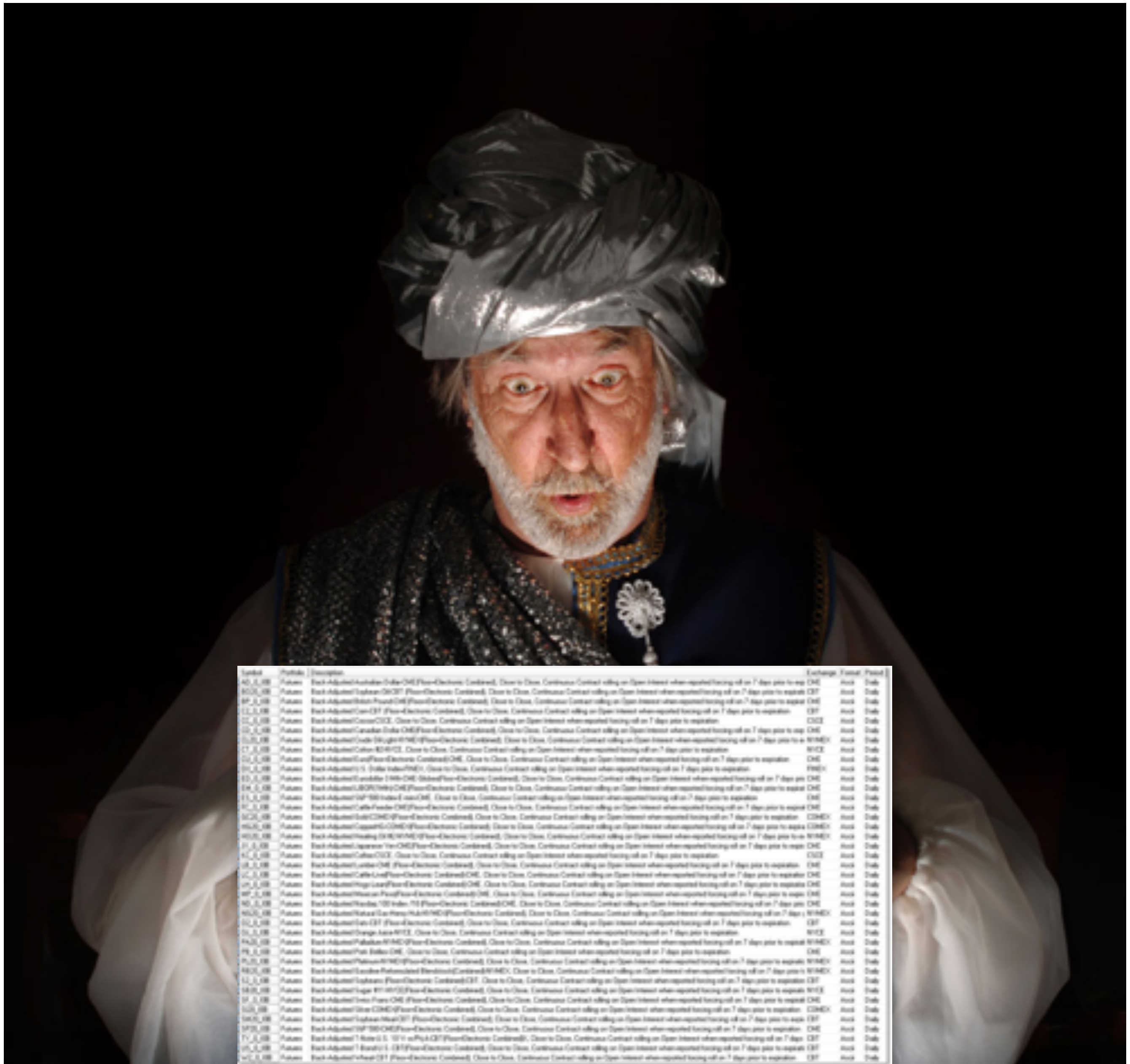
http://cs.nju.edu.cn/yuy/course_dm13ms.ashx

Position



The desire of prediction

The desire of prediction

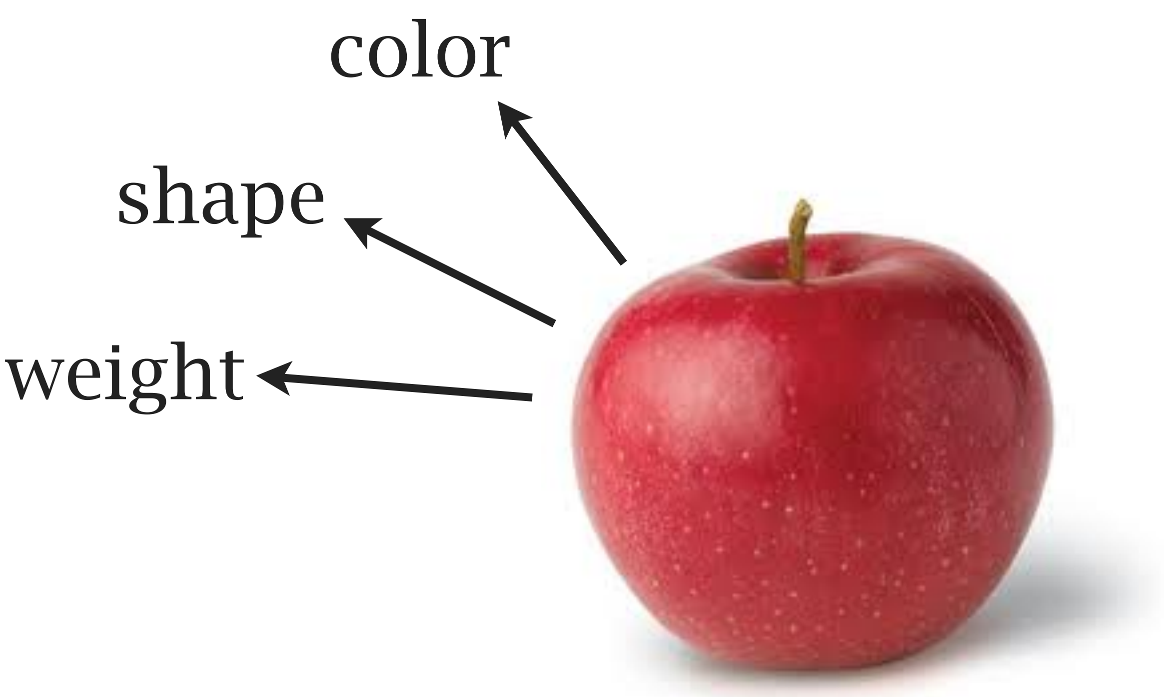


Predictive modeling

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

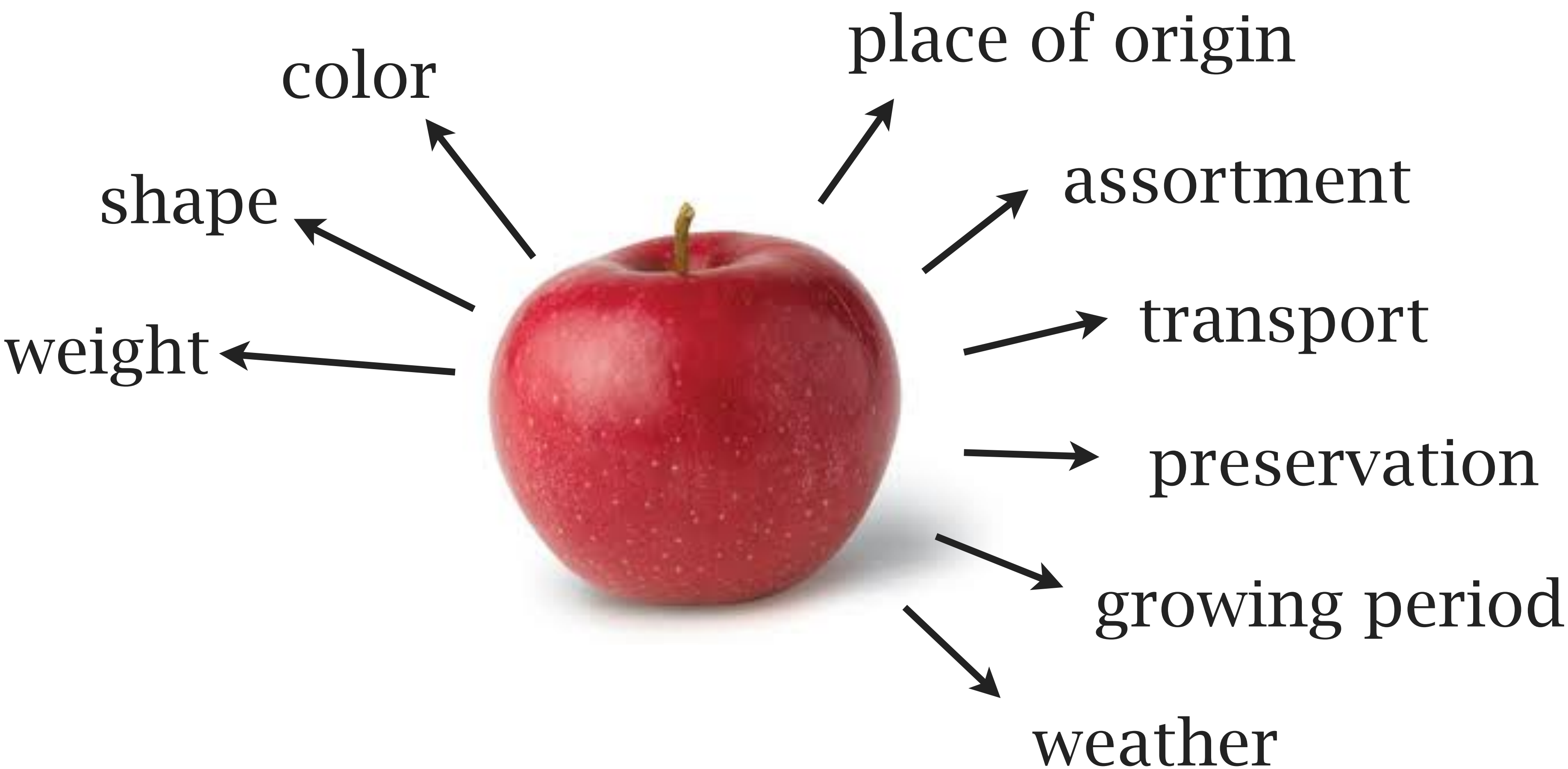
Predictive modeling

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



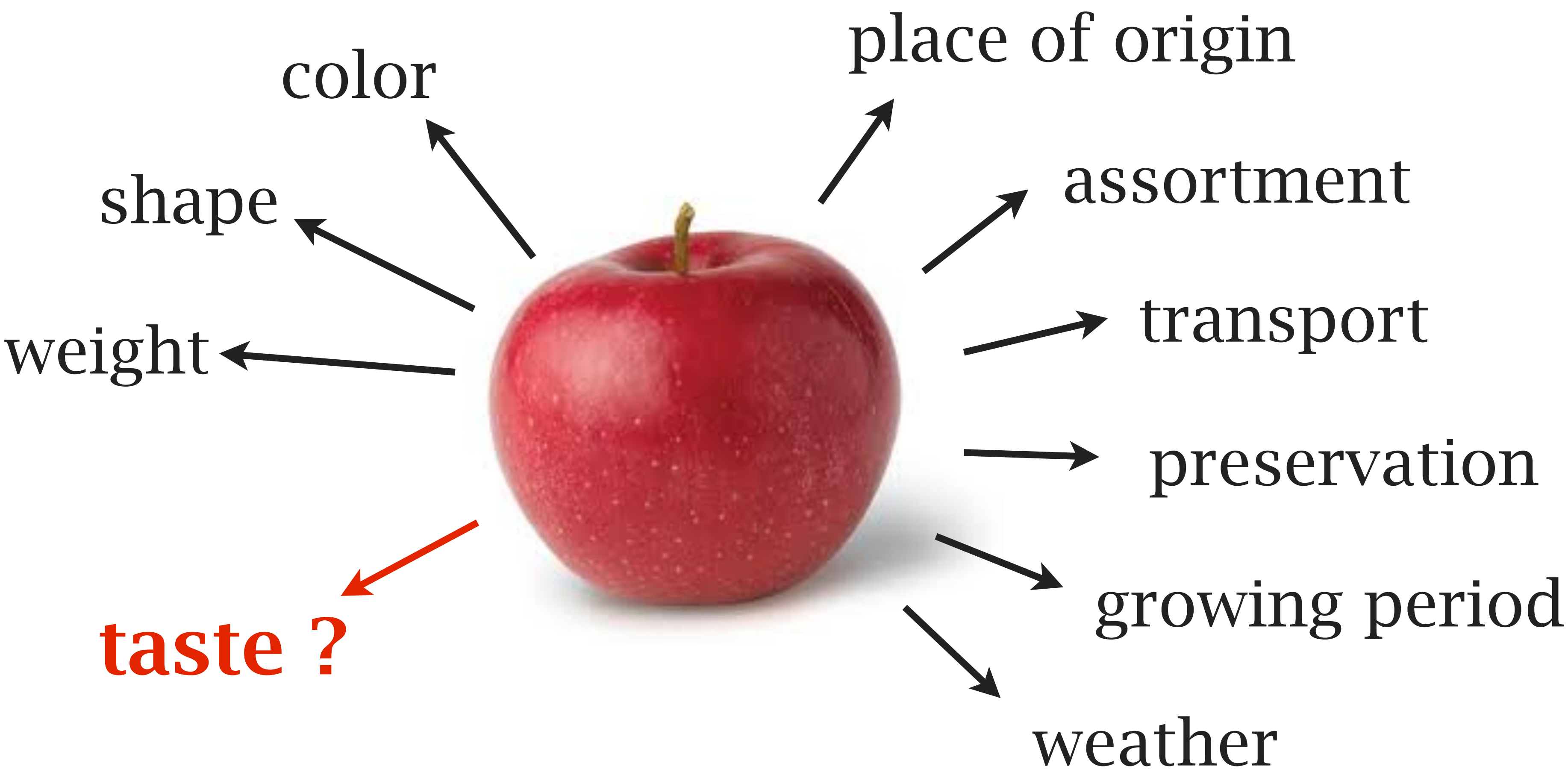
Predictive modeling

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



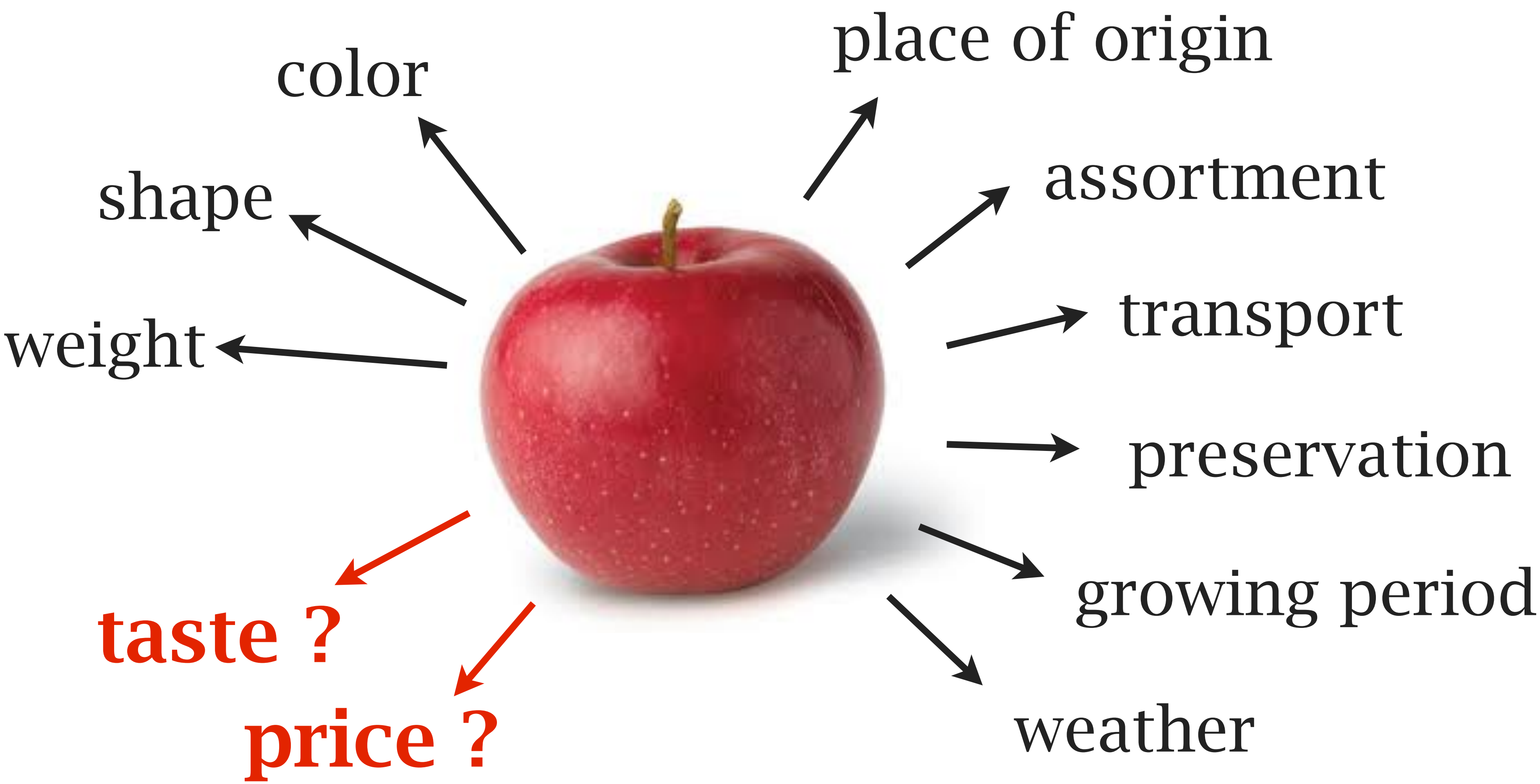
Predictive modeling

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).



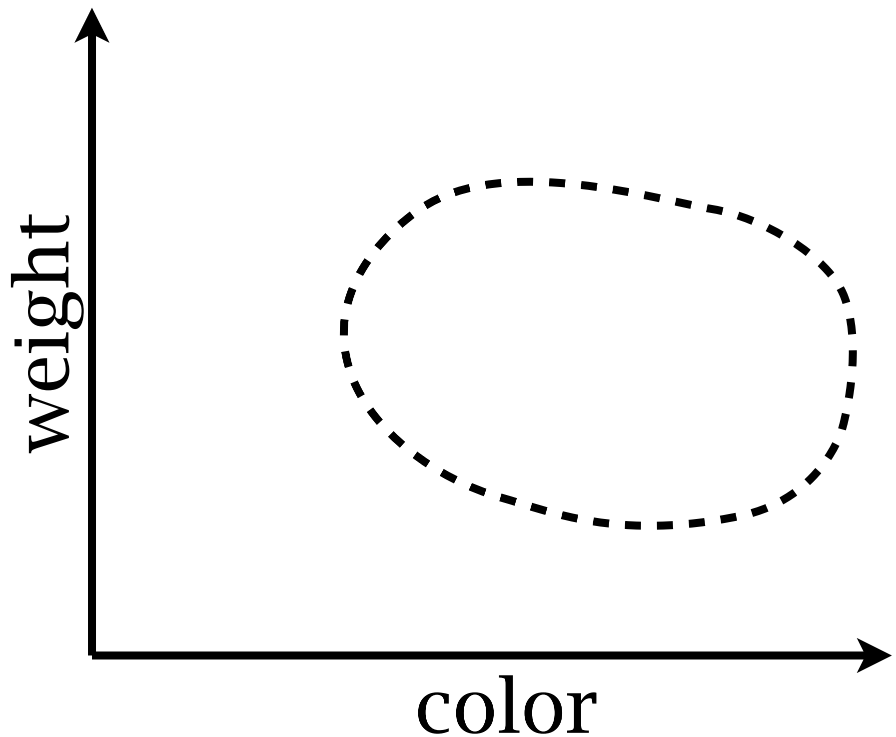
Supervised learning/inductive learning

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/-)



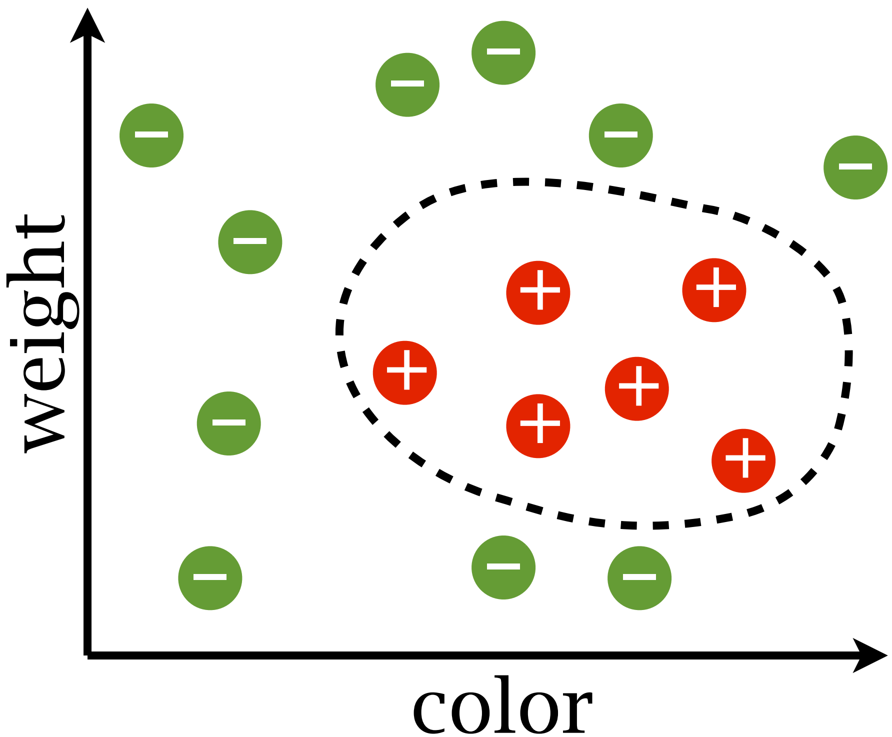
(color, weight) $\rightarrow$ sweet ?

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

ground-truth function f

Classification

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



(color, weight) $\rightarrow$ sweet ?

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

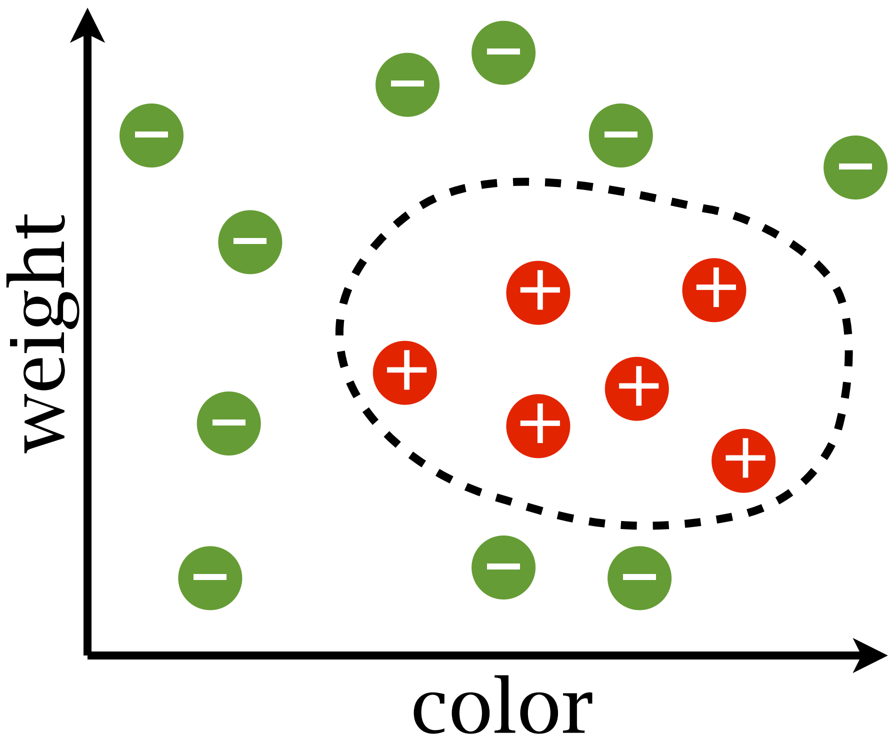
ground-truth function f

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

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

Classification

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



(color, weight) $\rightarrow$ sweet ?

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

ground-truth function f

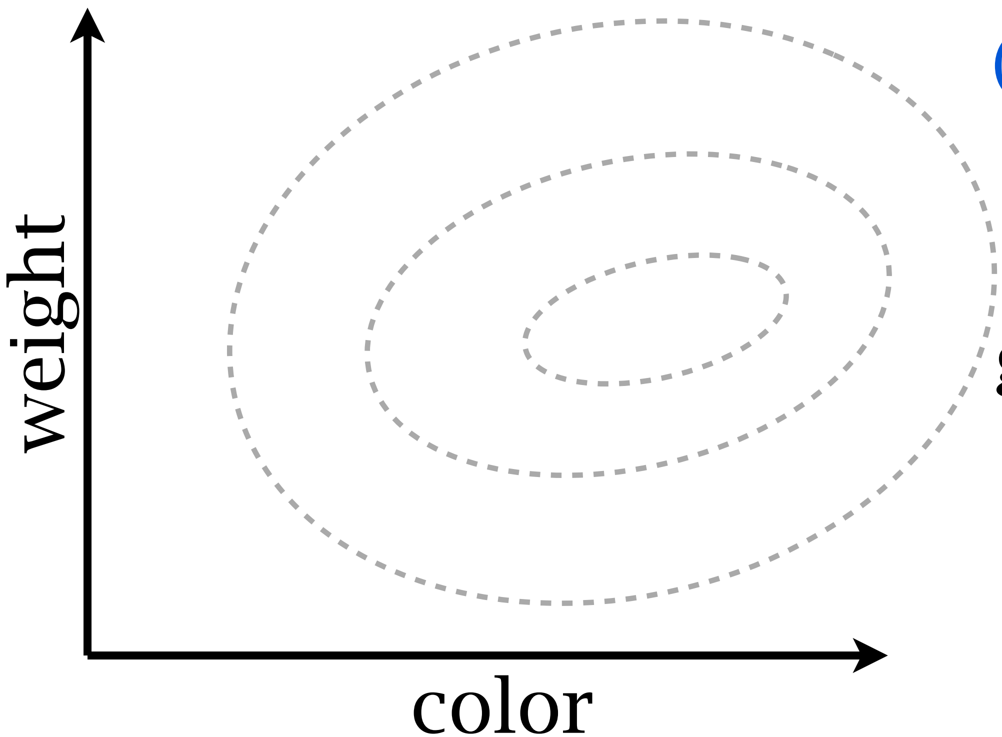
examples/training data:
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

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

learning: find an f' that is close to f

Regression

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



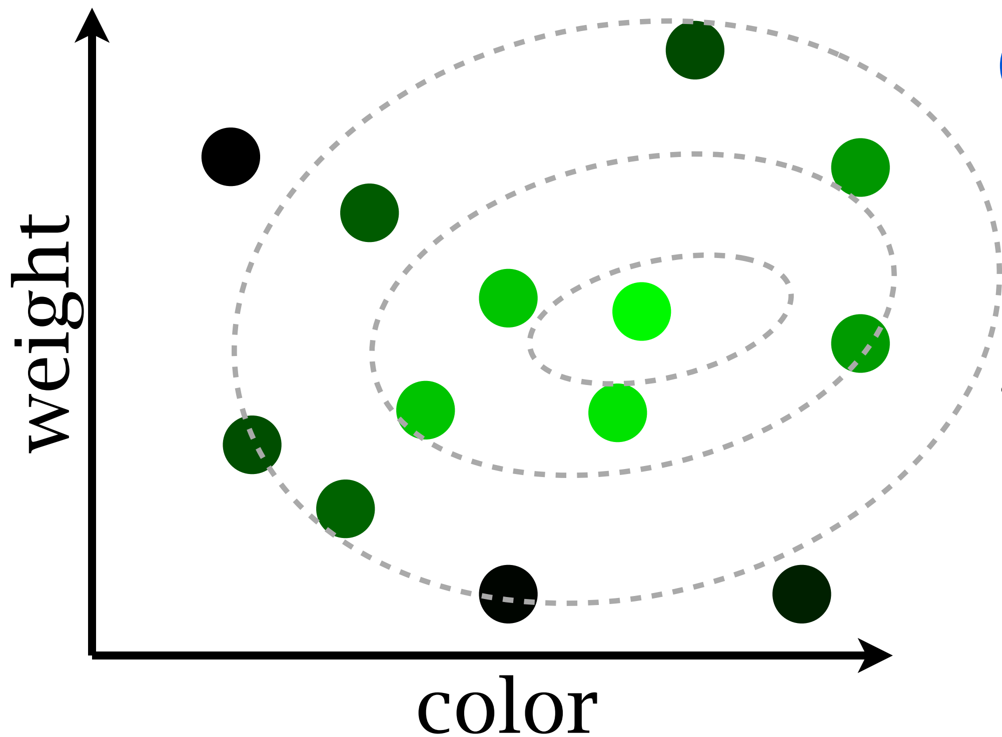
(color, weight) $\rightarrow$ price

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

ground-truth function f

Regression

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



(color, weight) → price

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

ground-truth function f

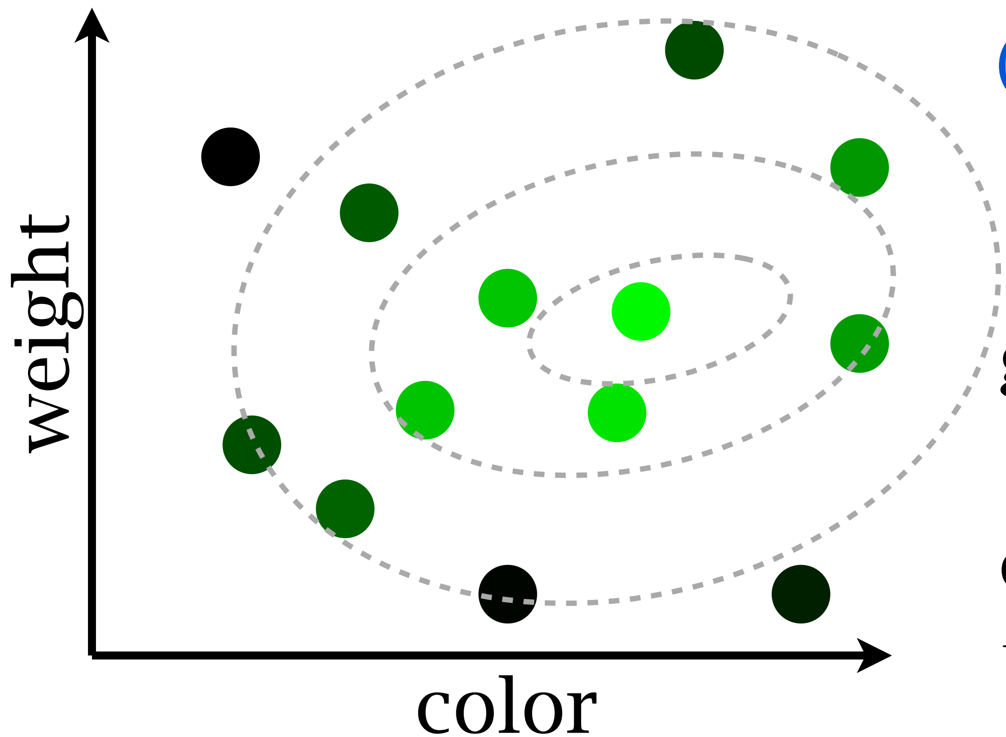
examples/training data:

$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

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

Regression

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



(color, weight) $\rightarrow$ 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

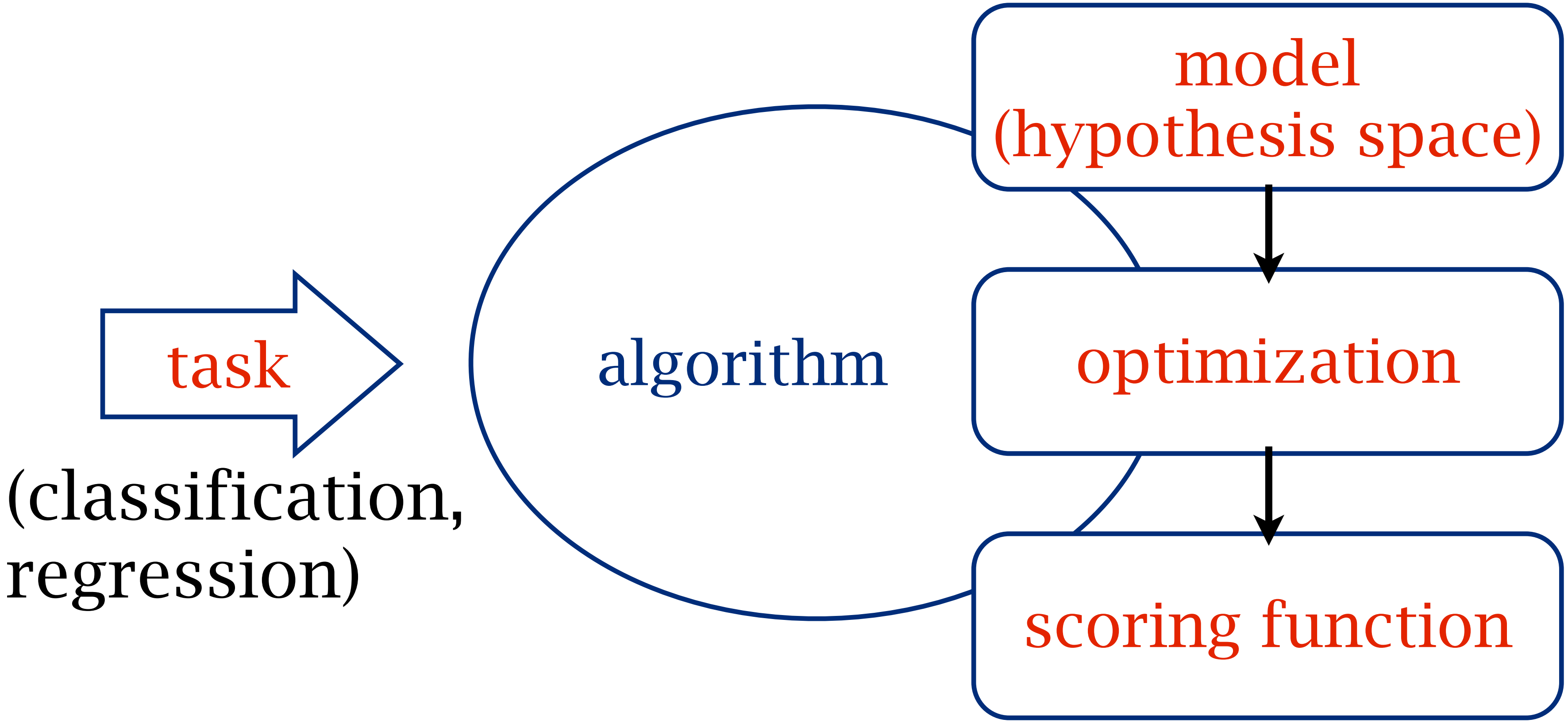
Ensemble methods

Handling big data

Why different classifiers?

- heuristics
- viewpoint
- performance

Learning algorithm components



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?

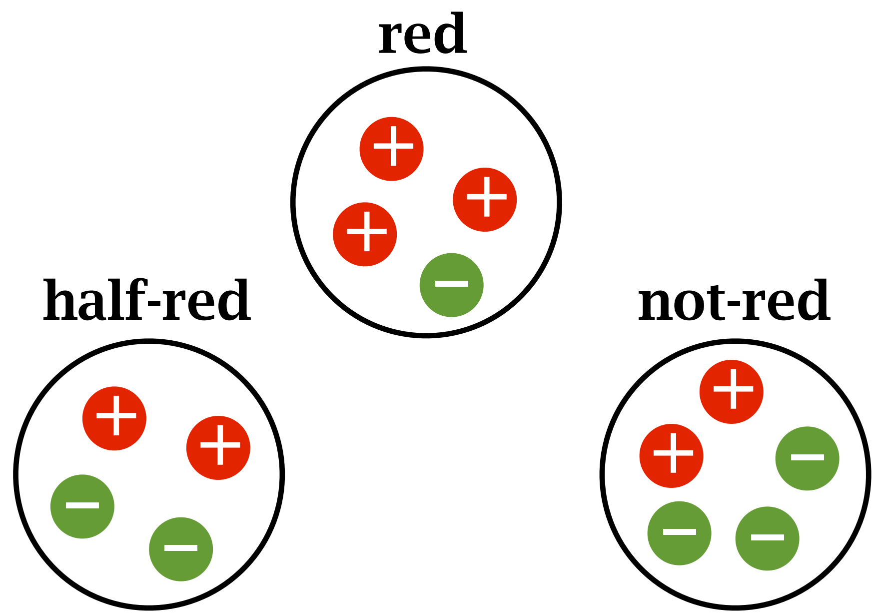
$$f' = \begin{cases} \text{sweet}, & \text{color} = \text{red} \\ \text{not-sweet}, & \text{color} \neq \text{red} \end{cases}$$

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?

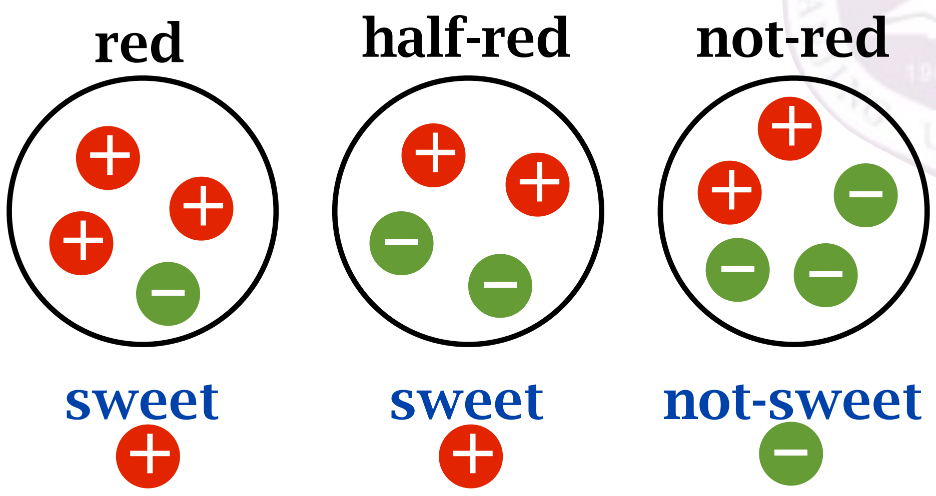


$$f' = \begin{cases} \text{sweet}, & \text{color} = \text{red} \\ \text{sweet}, & \text{color} = \text{half-red} \\ \text{not-sweet}, & \text{color} = \text{not-red} \end{cases}$$

not perfect
but how good?

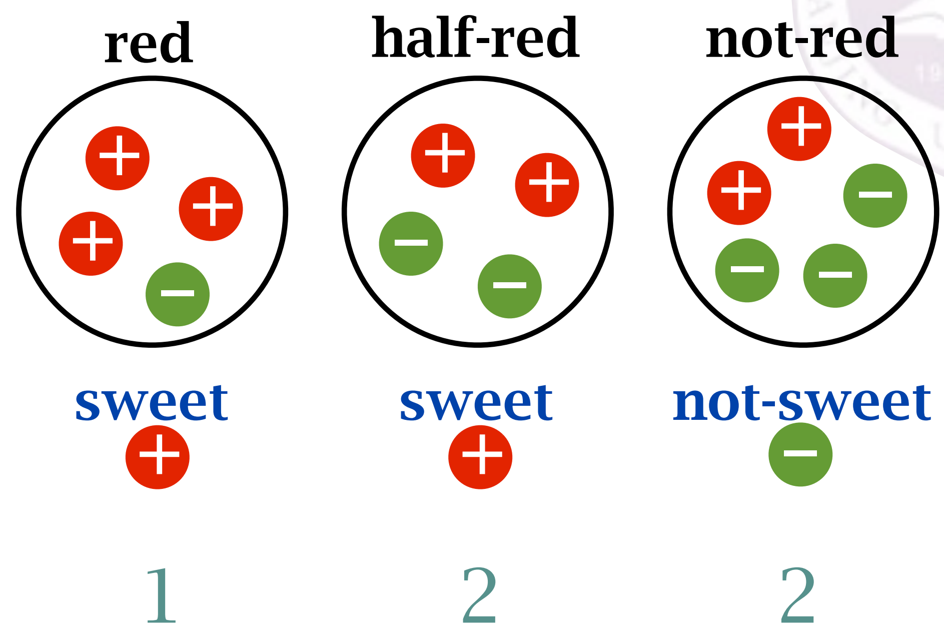
Consider a very simple case

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



Consider a very simple case

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

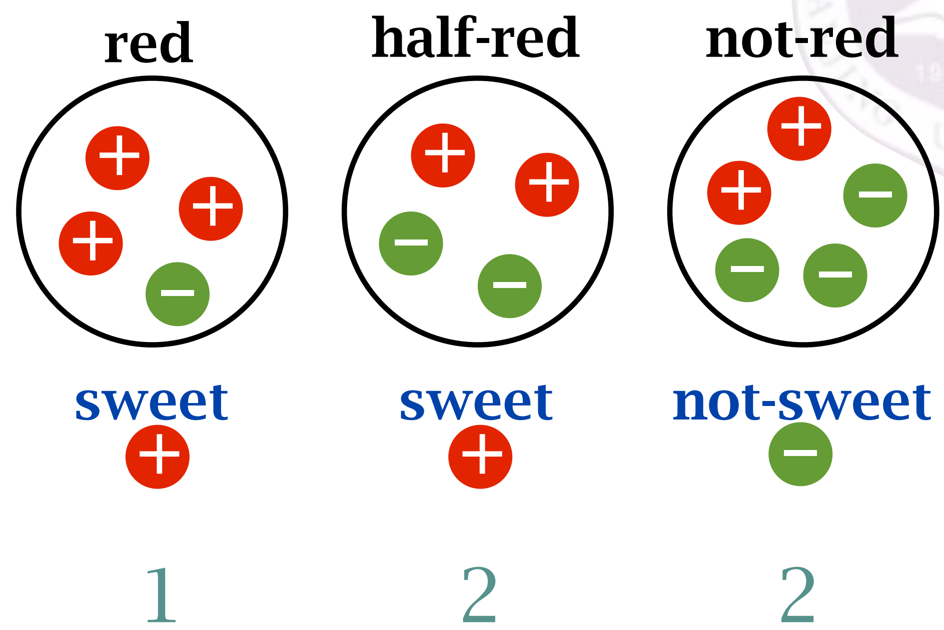


training error:

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

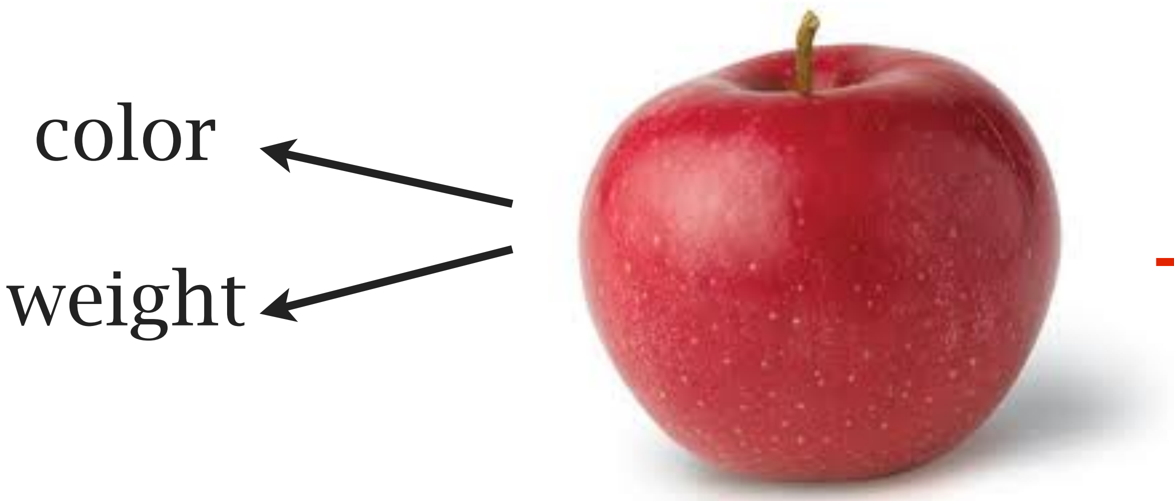
entropy after split: $I(X; \text{split}) = \sum_i ratio(split_i) H(split_i)$

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

information gain:

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

A little more complex case



→ **taste ?**

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

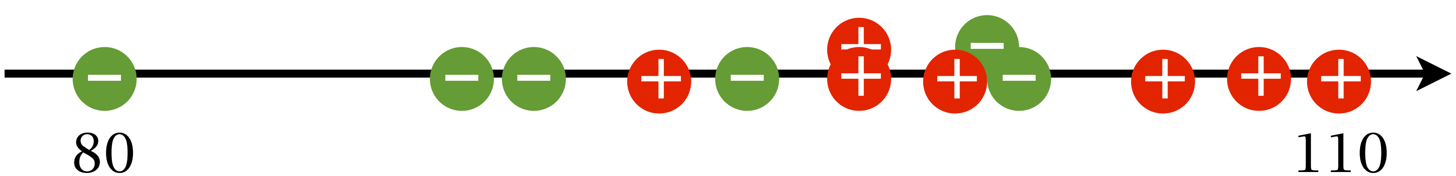
what the f' would be?

compare features and use the better one

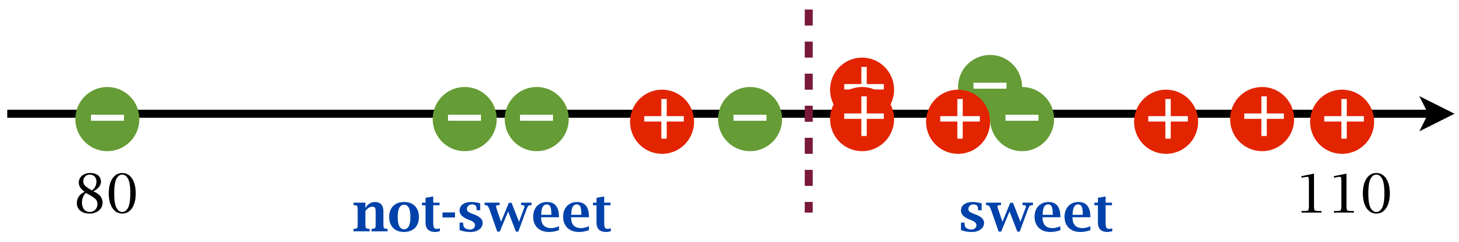
use color only -> known
use weight only -> ?

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



A little more complex case



for every split point

training error:

(1+2)/13=0.2307

information gain:

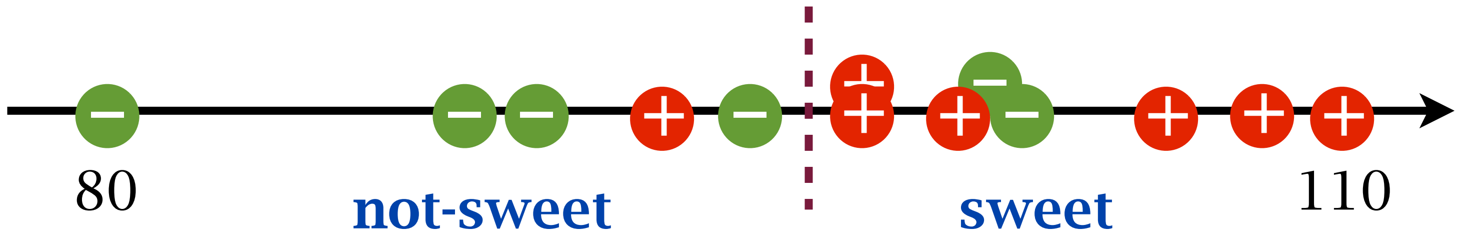
$$H(X) = -\sum_i ratio(class_i) \ln ratio(class_i) = 0.6902$$

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

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

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

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

color

weight

taste ?

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

what the f' would be?

color v.s. best split of weight

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

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

what the f' would be?

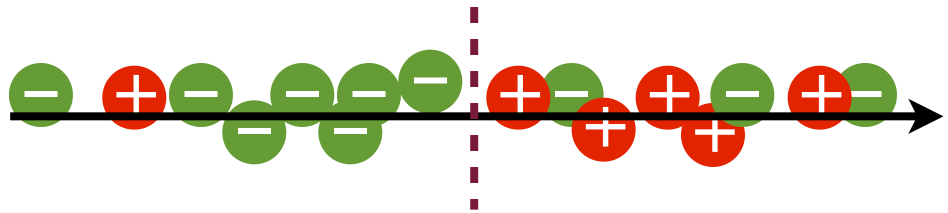
color v.s. best split of weight

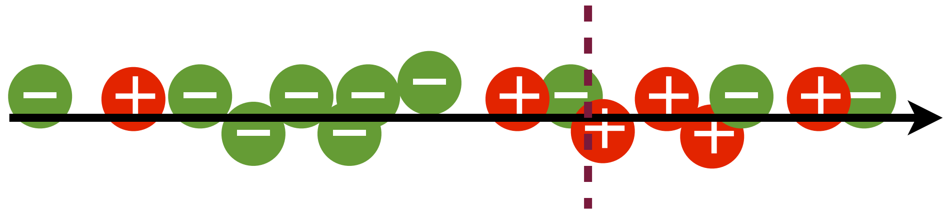
$$f' = \begin{cases} \text{sweet}, & \text{weight} > 95 \\ \text{not-sweet}, & \text{weight} \le 95 \end{cases}$$

training error v.s. info-gain

non-generalizable feature

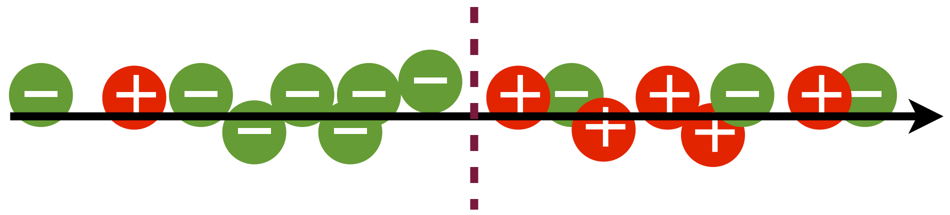
Training error v.s. Information gain



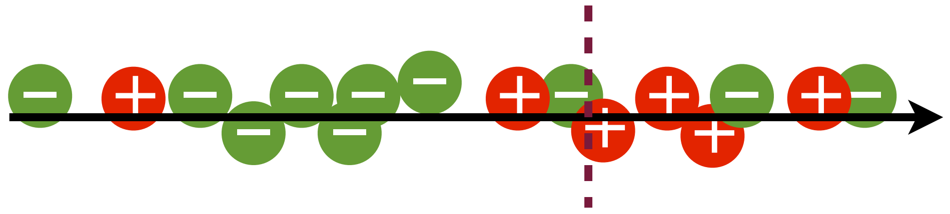


training error is less smooth

Training error v.s. Information gain



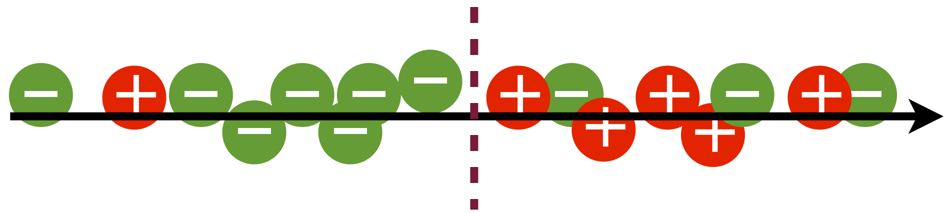
training error: 4



training error: 4

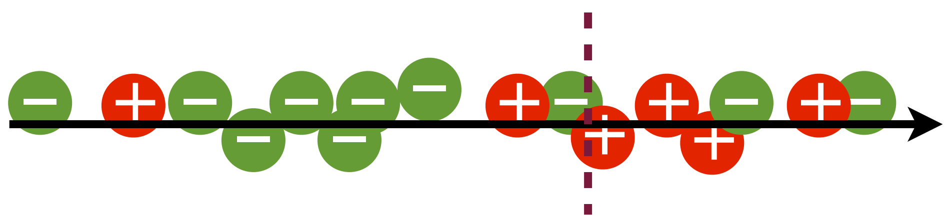
training error is less smooth

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

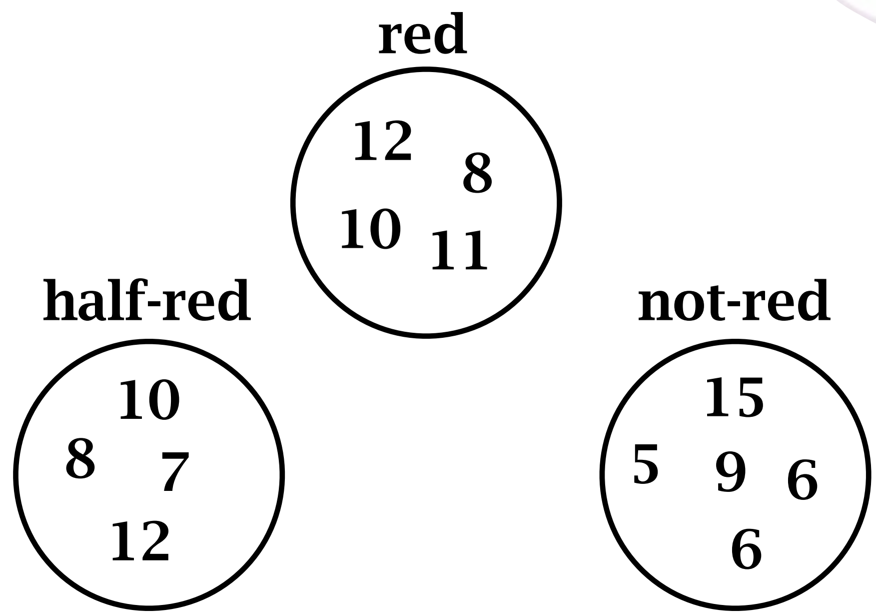
what the f' would be to minimize:

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

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:



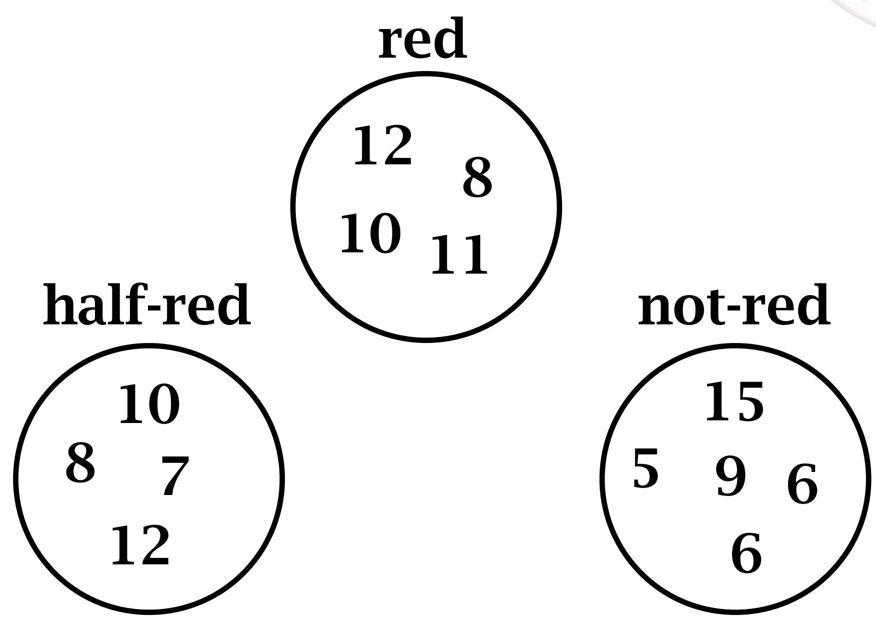
what is the prediction value of each color to minimize the mean square error?

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

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?

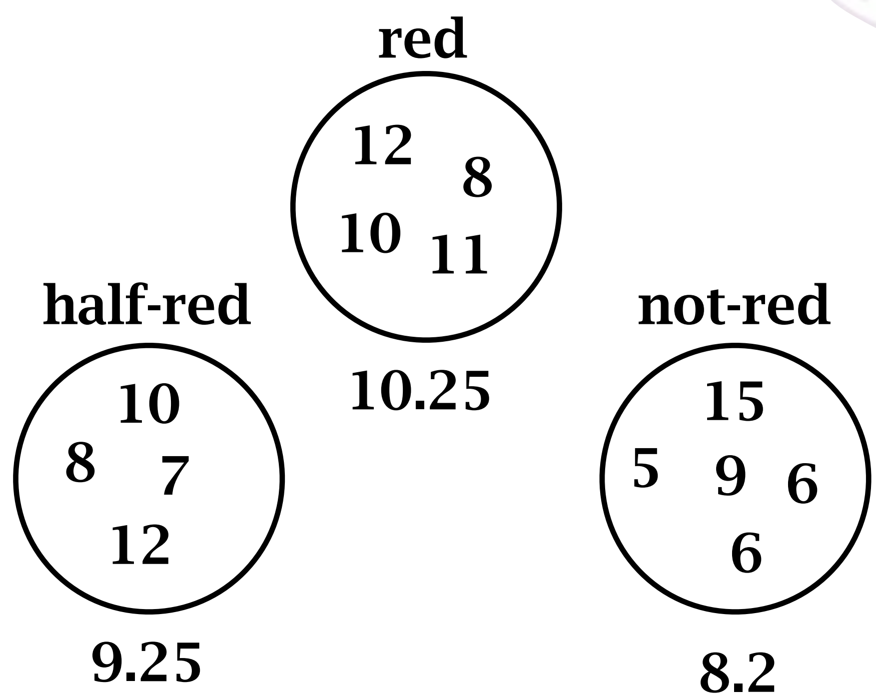
$$MSE = \frac{1}{n}\sum_i (f(x_i) - f'(x_i))^2$$

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:

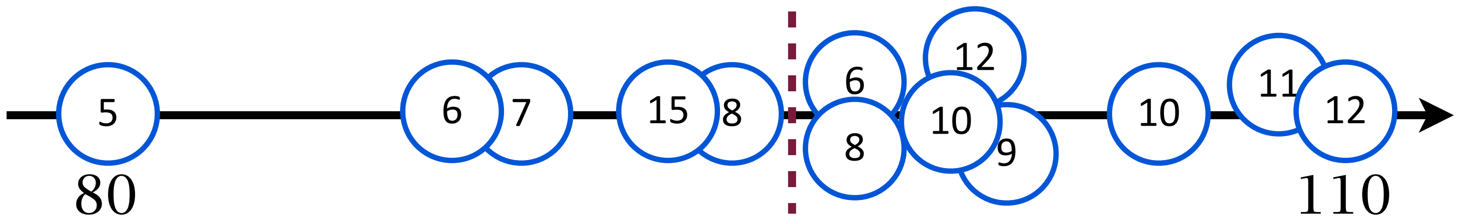


$$f' = \begin{cases} 10.25, & \text{color} = \text{red} \\ 9.25, & \text{color} = \text{half-red} \\ 8.2, & \text{color} = \text{not-red} \end{cases}$$

A regression case

for *weight* feature:

for any split:



mean: 8.2 mean: 9.75

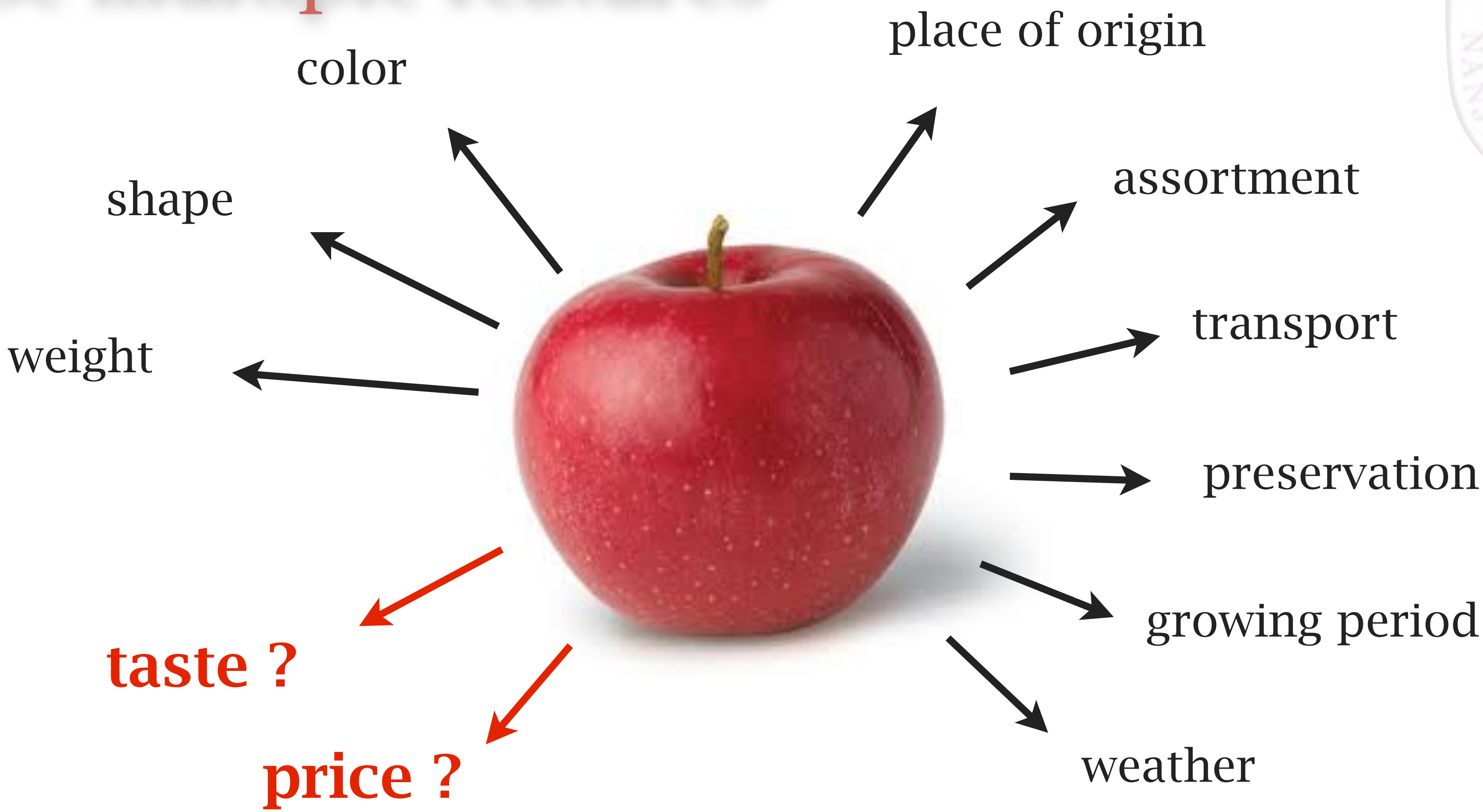
$$f' = \begin{cases} 9.75, & \text{weight} > 95 \\ 8.2, & \text{weight} \leq 95 \end{cases}$$

MSE: 12.56 MSE: 3.6875

overall MSE: 7.1

choose the split with minimal MSE

Use multiple features

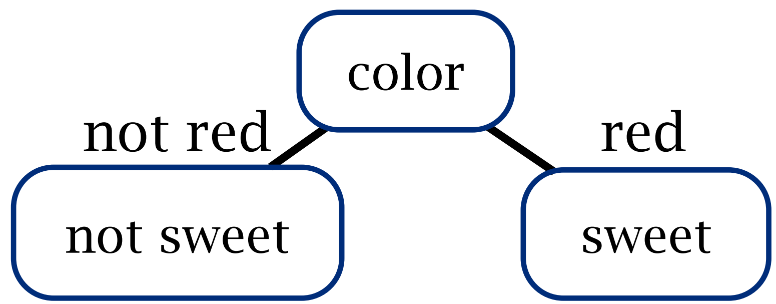


find a model by find the best feature/best split

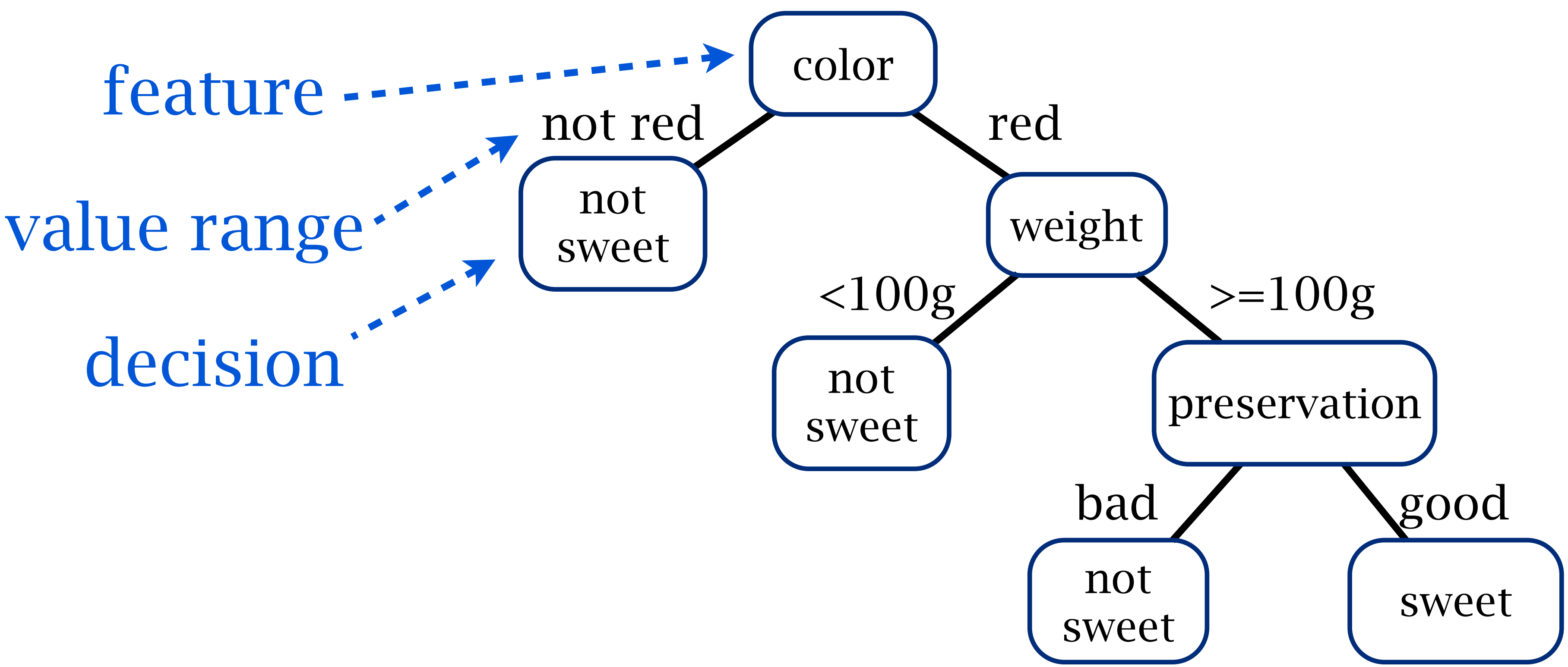
but only one feature/split is used

Use multiple features

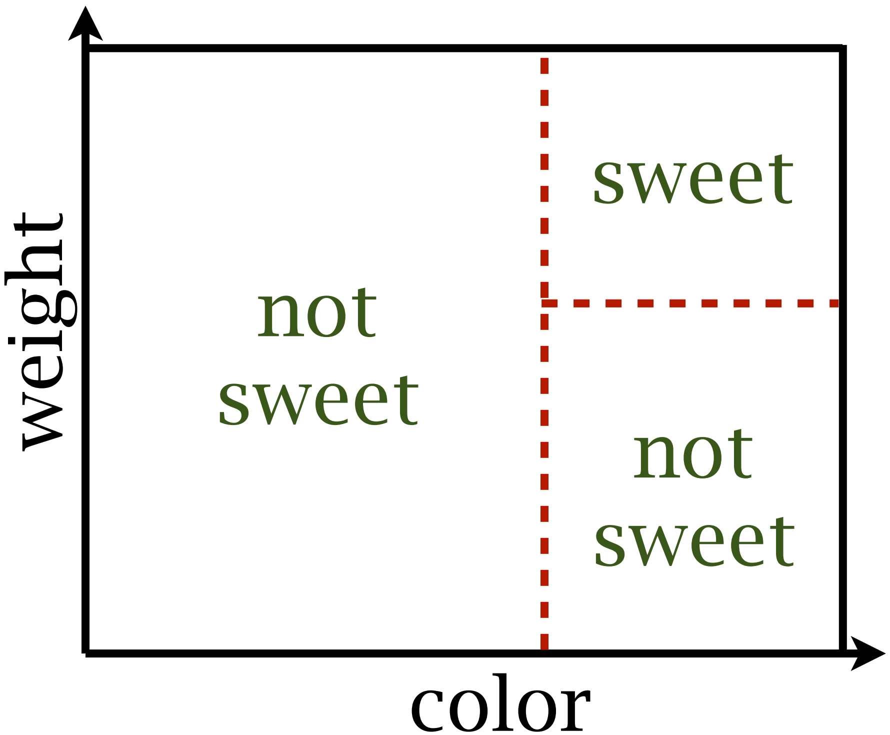
one feature model: decision stump

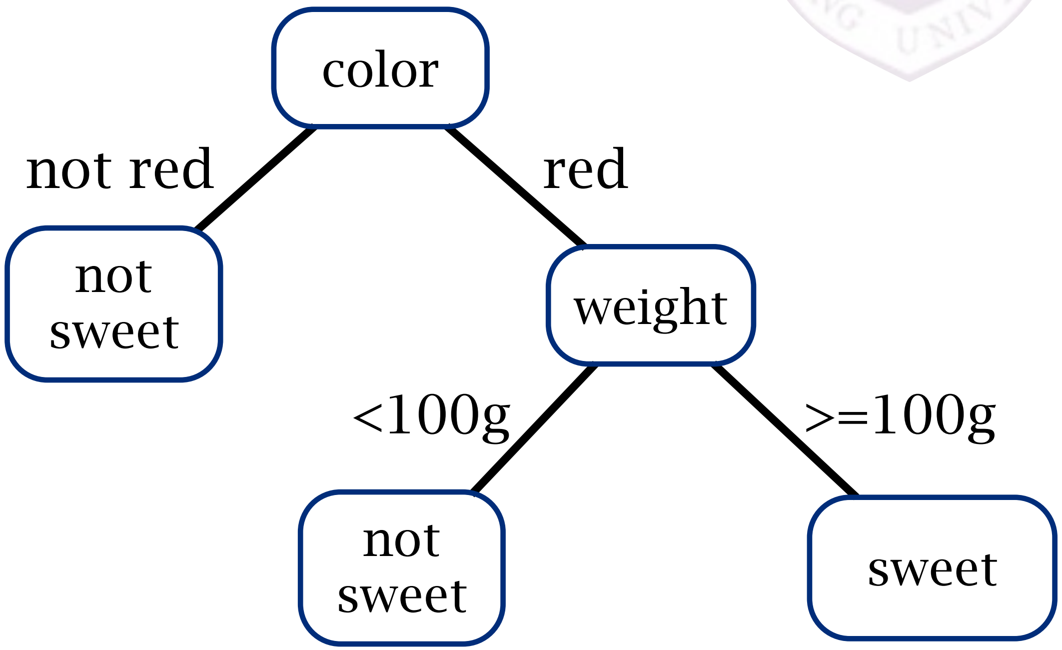


hierarchical model uses many features: decision tree

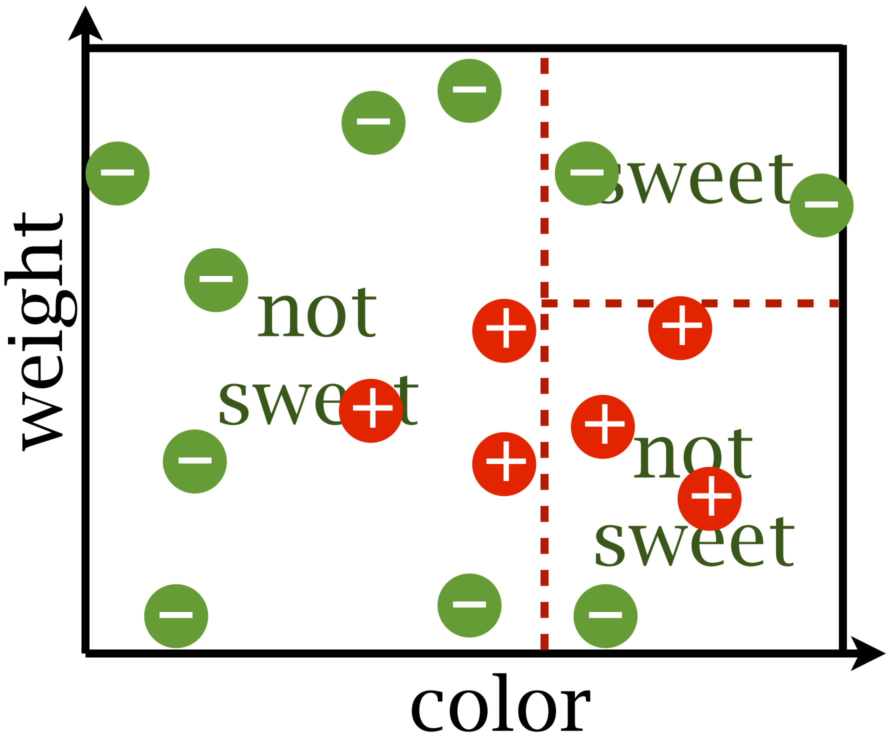


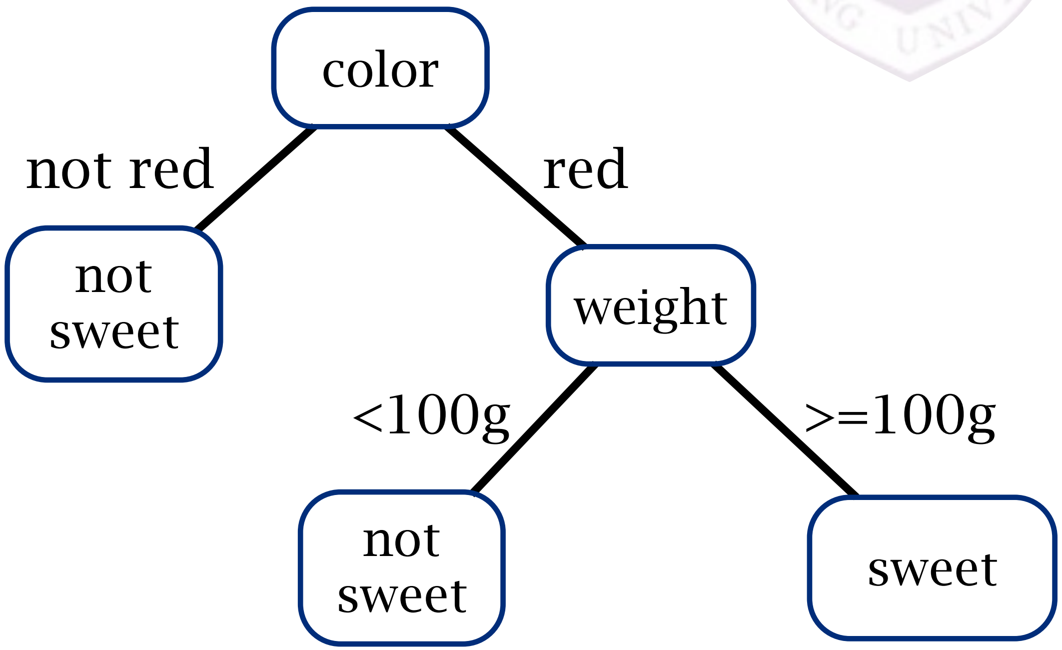
Decision tree model





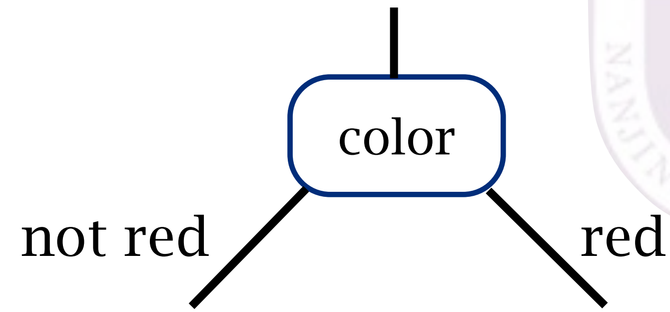
Decision tree model





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

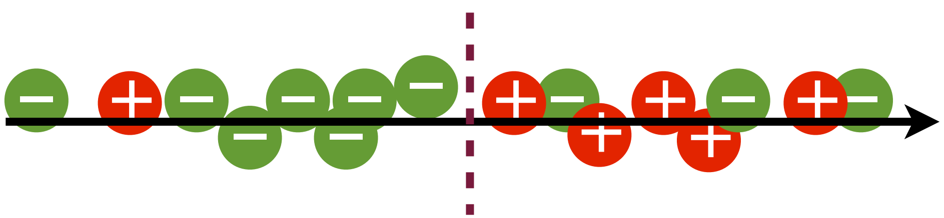
Jerome H. Friedman

Gini index

Gini index (CART):

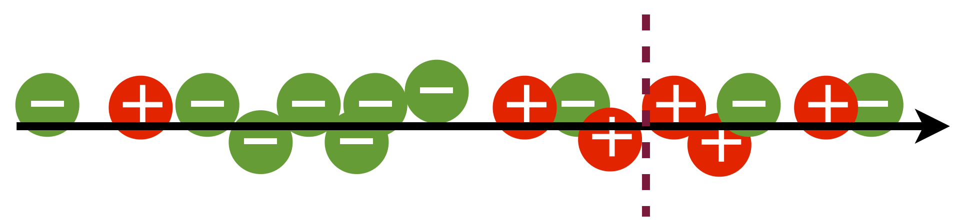
Gini: $Gini(X) = 1 - \sum_i p_i^2$

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



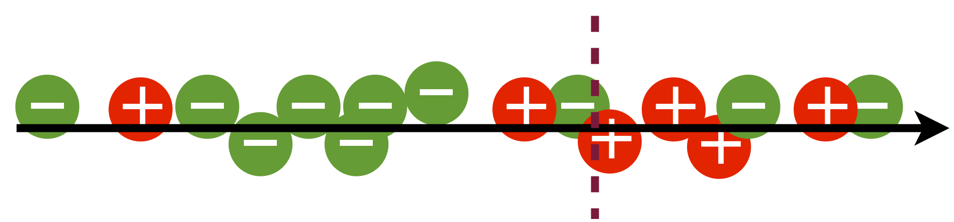
$\text{IG} = H(X) - 0.5192$

$Gini = 0.3438$



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

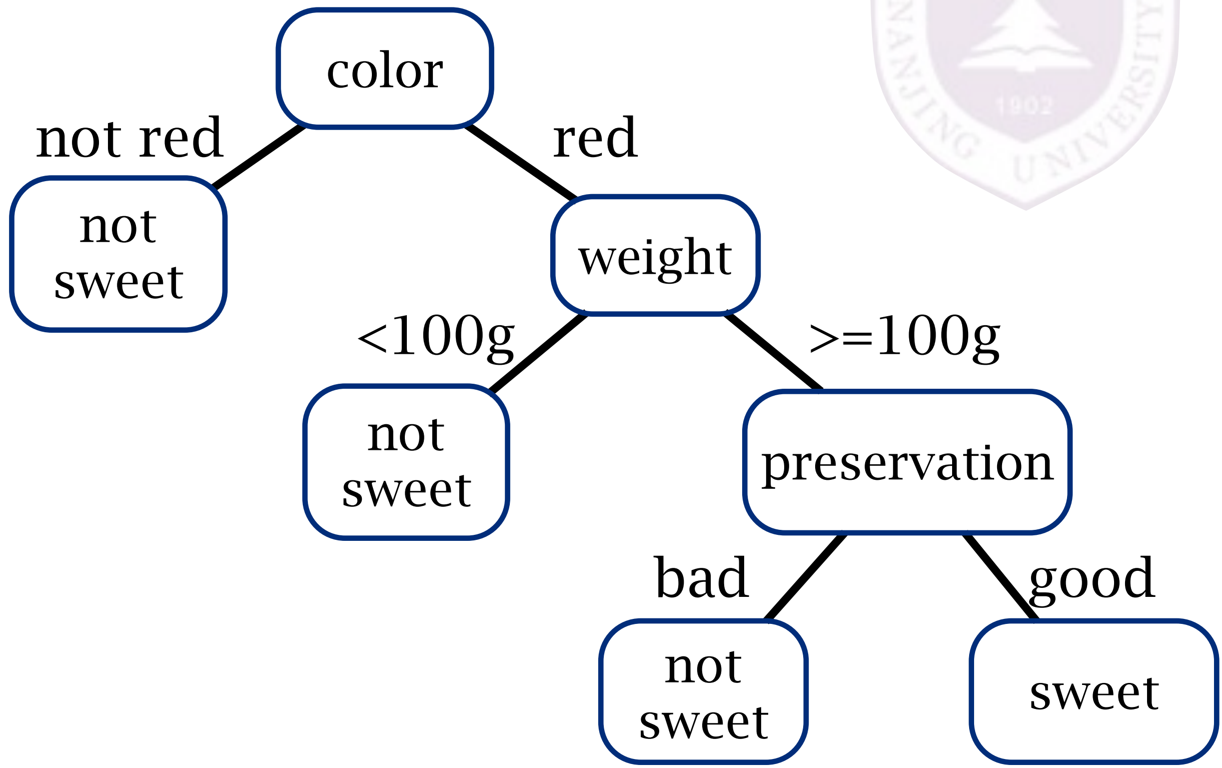
$Gini = 0.4427$



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

$Gini = 0.3667$

Split-criterion: stop

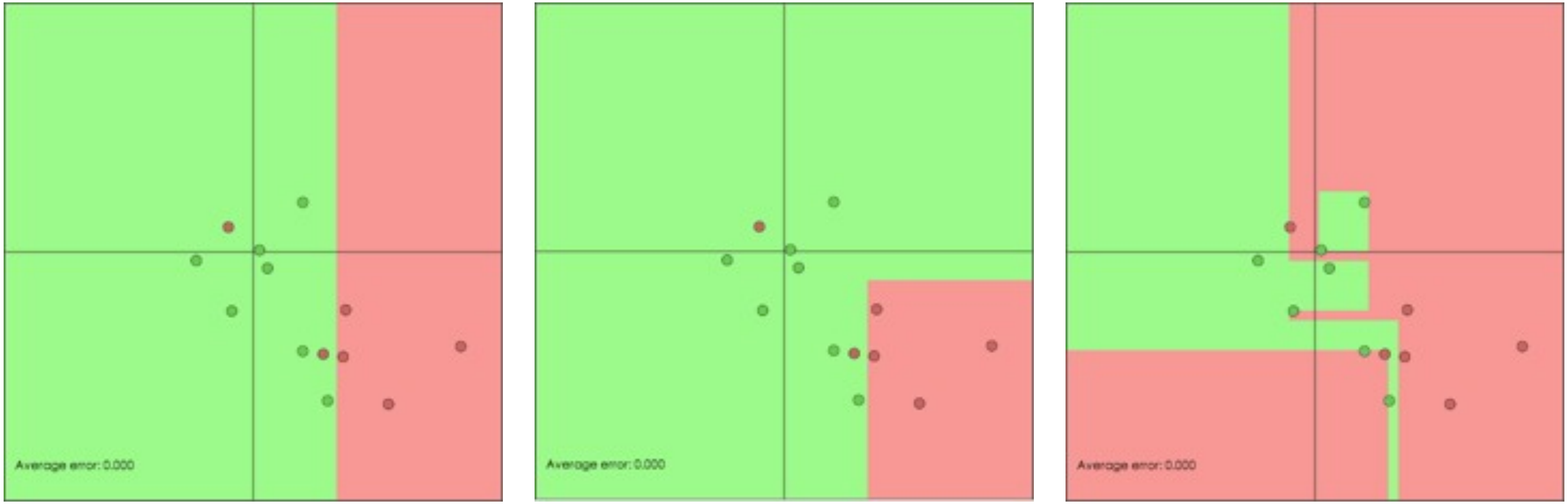


Stop criterion:
    no feature to use

Classification: examples are pure of class

Regression: MSE small enough

DT boundary visualization



decision stump　　max depth=2　　max depth=12

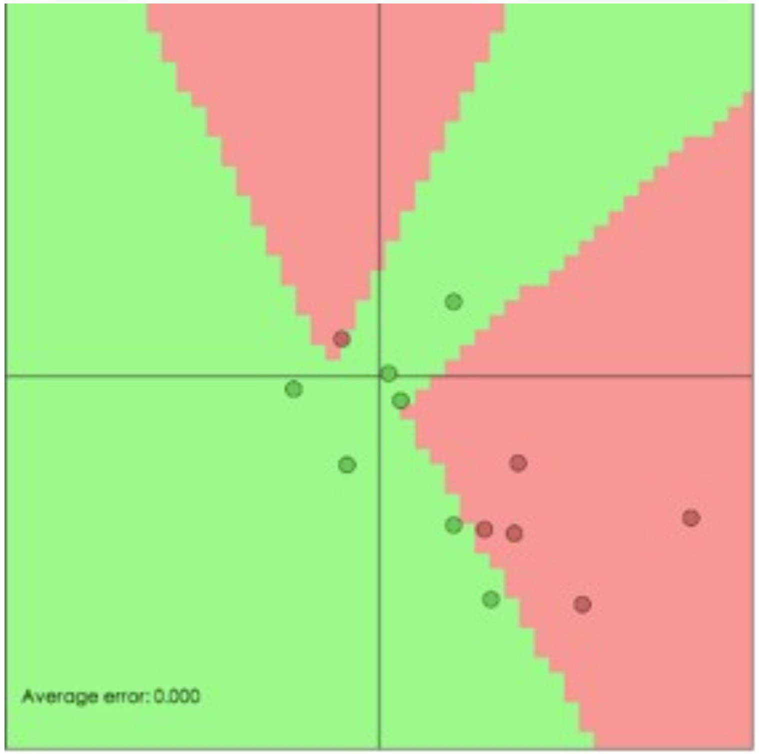
Oblique decision tree

choose a linear combination in each node:

axis parallel:
$X_1 > 0.5$

oblique:
$0.2\ X_1 + 0.7\ X_2 + 0.1\ X_3 > 0.5$

hard to train



Advantages

Fast to train

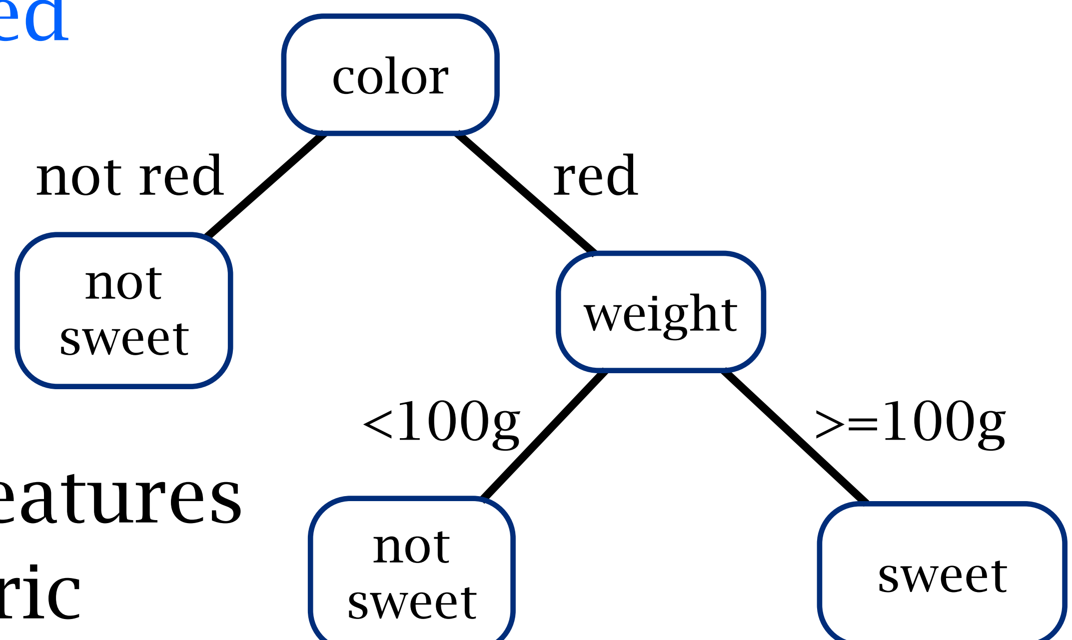
- samples: *m*
- features: *n*
- feature splits: *k*
- depth: *d*<*n*

training time:

- one node: $O(mkn)$
- *d* depth tree: $O(2^d mkn)$
- full tree: $O(m^2 kn)$

Fast to test

- not all features are tested



Regression/Classification

- Multi-class

Comprehensibility

Nominal and numerical features

Non-parametric, non-metric

Pruning

To make decision tree less complex

Pre-pruning: early stop

- minimum data in leaf
- maximum depth
- maximum accuracy

Post-pruning: prune full grown DT

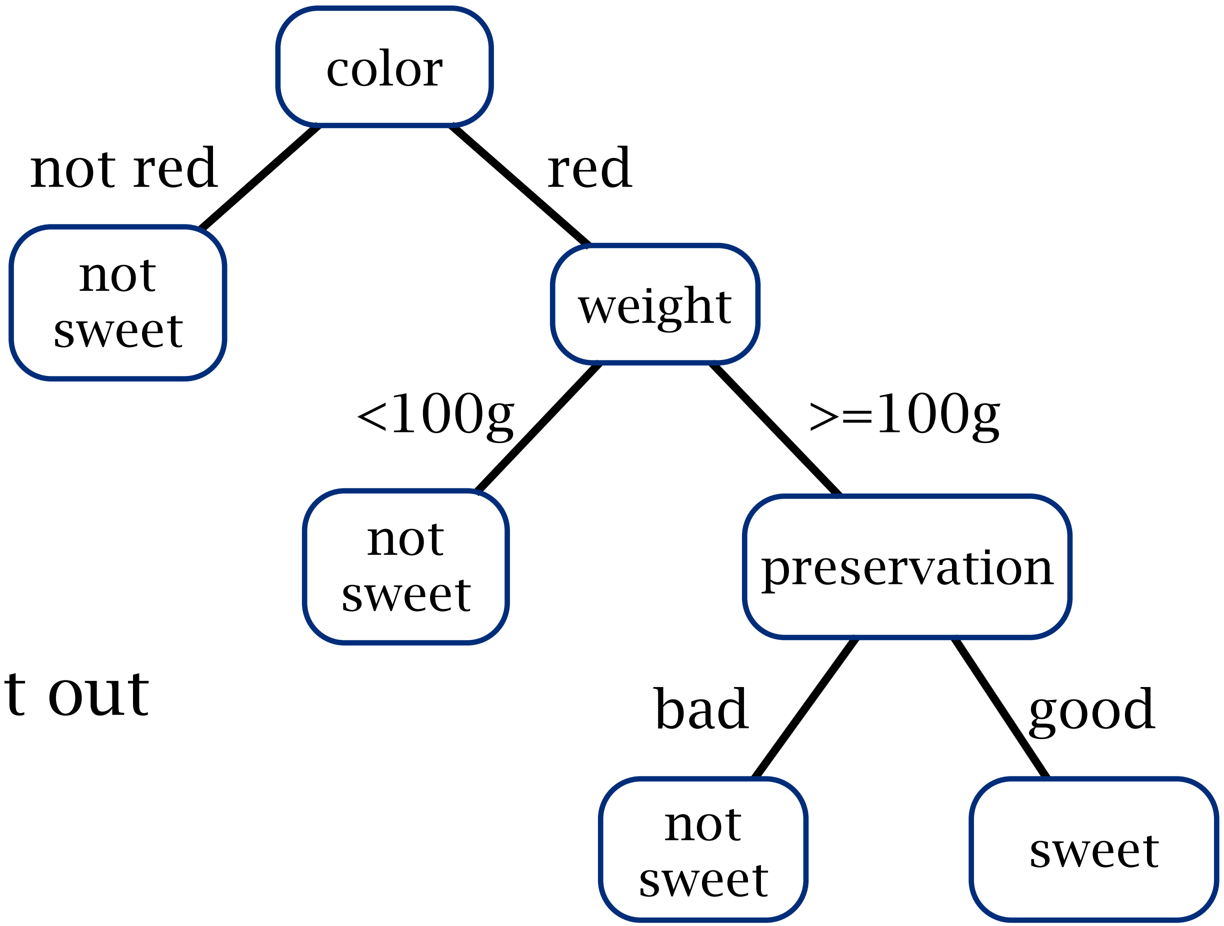
reduced error pruning

Reduced error pruning

1. Grow a decision tree

2. For every node starting from the leaves

3. Try to make the node leaf, if does not increase the error, keep as the leaf



could split a validation set out from the training set to evaluate the error

习题

监督学习的目标是否是最小化训练误差?

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)?