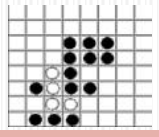
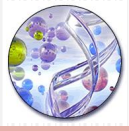



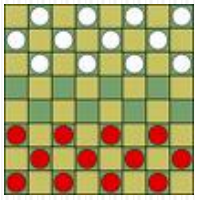

Welcome to

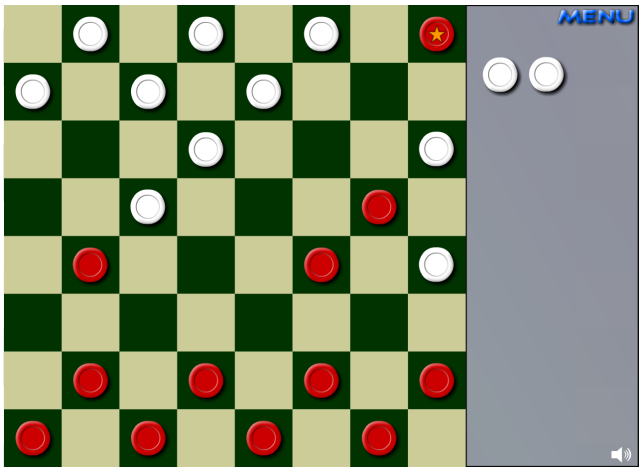
Introduction to Machine Learning!



Coffee Time

Playing Checker



- Feature Discussion:
  - What features do you think would be helpful to the automatic checker playing?

Topic 1: Introduction and General ML system design (cont.)

ZHANG Min
z-m@tsinghua.edu.cn

A general machine learning system design (cont.)

Designing a learning system e.g. Learning to play checkers

- What experience?
  - Be aware of training data bias: data, training procedure, features
- What exactly should be learned?
  - Correct vs. operational: Approximation, aka. Hypothesis
- How shall it be represented?

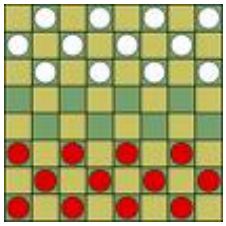
Representing (the Hypothesis Class $\hat{V}$)

- Possible representation
  - Tables of all states
  - Collection of rules
  - Polynomial function of board features
  - Neural network
  - ……
- Expressiveness of type of function must be chosen carefully
  - **Goodness** of approximation vs. **data** requirements

Representing the Hypothesis Class (Cont.)

- Learning examples $< b, \text{Vtrain}(b) >$

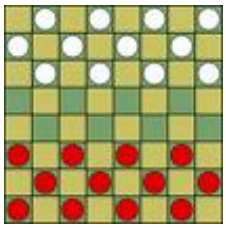
  where Vtrain(b) is the label of b
  - wp(b): # of white pieces on board b
  - rp(b) : # of red pieces on b
  - wk(b) : # of white kings on b
  - rk(b) : # of red kings on b
  - wt(b) : # of white pieces threatened by red (i.e. which can be taken on red's next turn)
  - rt(b) : # of red pieces threatened by white



  - e.g. <<wp=0, rp=3, wk=0, rk=1, wt=0, rt=0>,+100>
- An example function

$$\hat{V}(b) = w_0 + w_1 \cdot wp(b) + w_2 \cdot rp(b) + w_3 \cdot wk(b) + w_4 \cdot rk(b) + w_5 \cdot wt(b) + w_6 \cdot rt(b)$$

Designing a learning system e.g. Learning to play checkers



- What experience?
  - Be aware of training data bias: data, training procedure, features
- What exactly should be learned?
  - Correct vs. operational: Approximation, aka. Hypothesis
- How shall it be represented?
  - Expressiveness: **Goodness** of approximation vs. **data** requirements
- What specific algorithm to learn it?

Choose a weight training rule (Learning Algorithm) – An example

- Best fit of data
- Common goal: minimize squared error

$$\sum_{training\ set} (V_{train}(b) - \hat{V}(b))^2$$

- Popular alg.: **Least Mean Squares, LMS**
  - Initialize weights
  - Repeat:

1. Select a training example *b* at random
2. Compute error(b) = $V_{train}(b) - \hat{V}(b)$
3. For each board feature f_i, f_i belong to {wp, rp, …, rt}, update weight w_i

$w_i \leftarrow w_i + c \cdot f_i \cdot error(b)$

c is some small constant, say 0.1 for example, to moderate the rate of learning

(Intuition: larger |error| → greater change,
larger f_i → greater contribution to error)

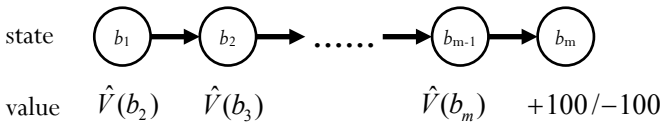
Designing a learning system e.g. Learning to play checkers

- What experience?
- What exactly should be learned?
- How shall it be represented?
- What specific algorithm to learn it?

Putting it Together – final design

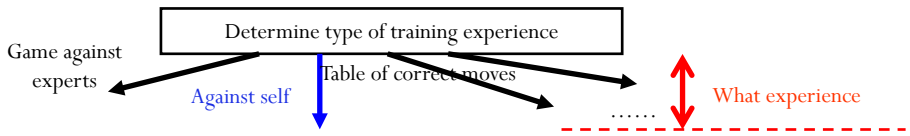
Putting it together

- Initialize weights of $\hat{V}$
- Use $\hat{V}$ to play games against self, output sequence of board states for each game
- Label each b with $\hat{V}(Successor(b))$

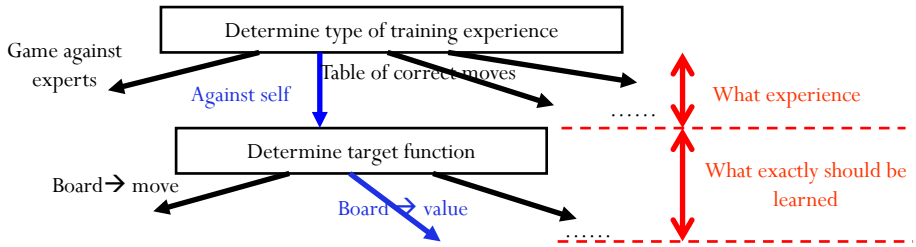


- Learn new weight, yielding new $\hat{V}$
- Start new set of games

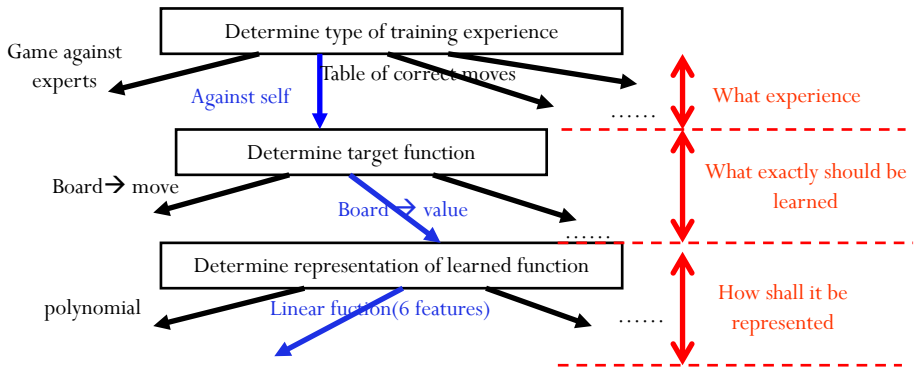
Overview – Design choices



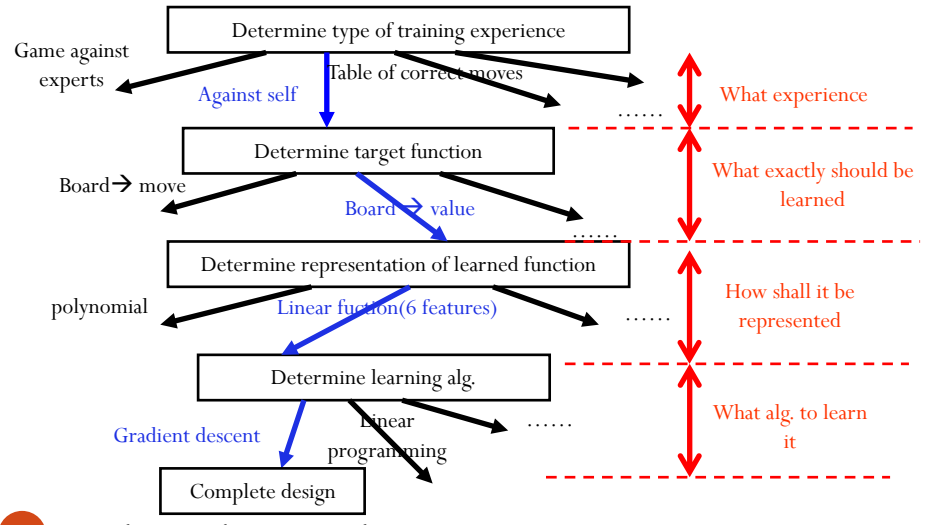
Overview – Design choices



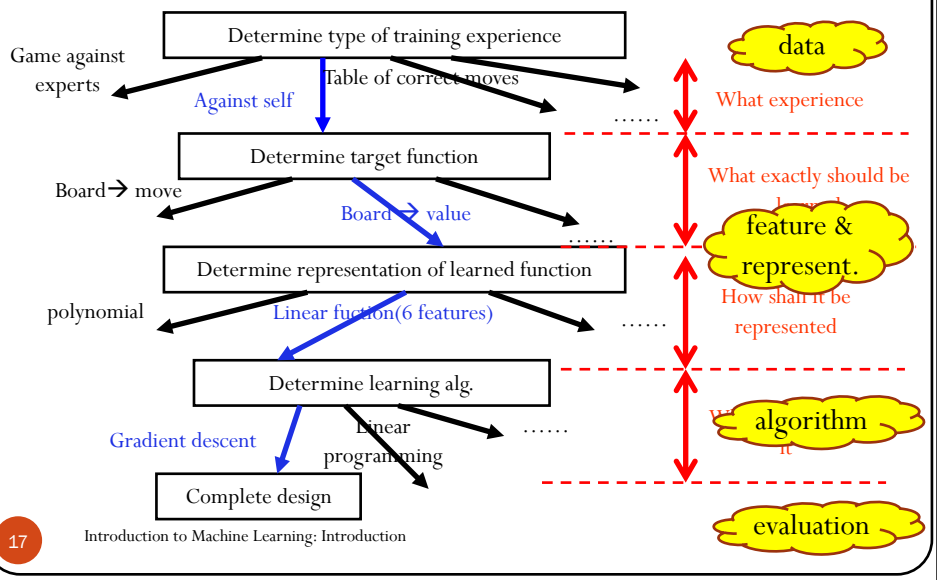
Overview – Design choices



Overview – Design choices



Overview – Design choices



Topic 1. Introduction (overview)

- Application background
- What's machine learning
  - T （Task）
  - E （Experience）
  - P （Performance）
- History
- General design of a machine learning system (An example)

Topic 2: Decision Trees

Part I. Basic Decision tree learning

An example: Enjoy Sport

- Known:

Sky	Temp	Humid	Wind	Water	Forecst	Enjoy
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

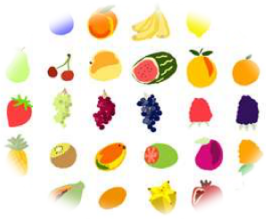
In a coming new day,

will the one enjoy the sport?

Classical Targeting Problems

- Classification problem involving nominal data.
- Discrete
- No natural notion of similarity
- No order, in general

- Another Example: Fruit
  - Color: red, green, yellow, …
  - Size: small, medium, big
  - Shape: round, thin
  - Taste: sweet, sour



Representation

- Lists of attributes instead of vectors of real numbers.

e.g.

- EnjoySport:
  - 6 tuples on *Sky, AirTemp, Humidity,Wind,Water, Forecast*
  - {Sunny, Warm, Normal, Strong, Warm, Same}
- Fruit:
  - 4-tuple on *color*, *size*, *shape*, *taste*
    {red, round, sweet, small}

Basic Concepts

- Given:
  - Instance Space X e.g. possible days, each described by the attributes *Sky, AirTemp, Humidity,Wind,Water, Forecast*
  - Hypothesis Class H e.g.
    if (Temp = cold AND humidity = high) *then* play tennis = no.
  - Training Examples D Positive and negative examples of the Target Function C $<x_1, c(x_1)>, \ldots\ldots, <x_m, c(x_m)>$
- Determine: A hypothesis $h \in H$ *such that*

$$h(x) = c(x) \text{ for all } x \in X$$

Basic Concepts

- Typically X is exponentially or infinitely larger, so in general we can never be sure that $h(x)=c(x)$ for all $x \in X$
- Instead, settle for a good approximation, e.g. $h(x)=c(x)$ for all $x \in D$

Suppose: n binary attributes/features （e.g. true/false, warm/cold）

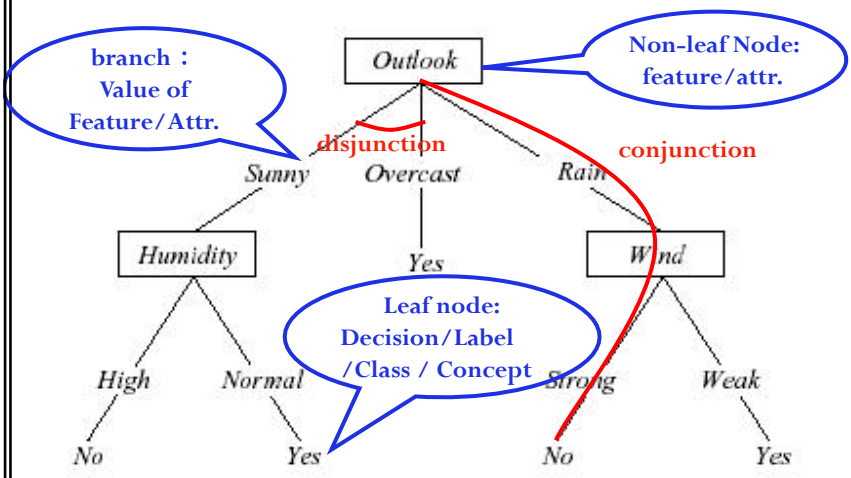
- **Instance Space X：2^n elements**
- **Concept (Hypothesis) Space H：at most 2^{2^n} elements (why?)**

Training examples

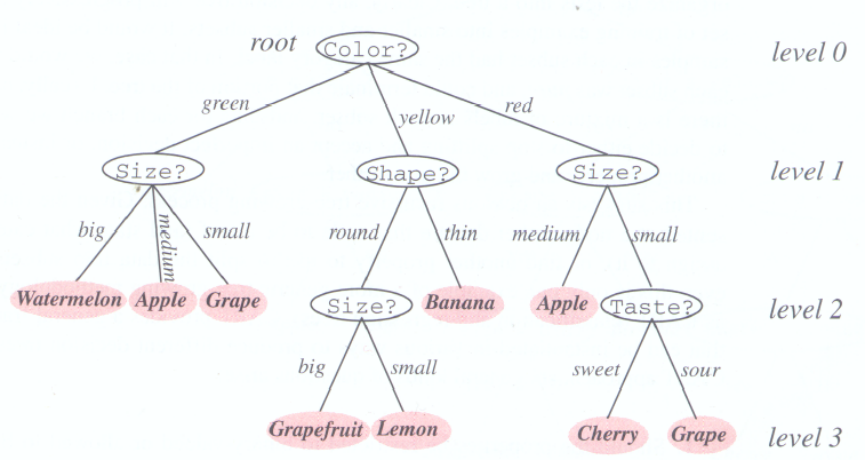
Sky	Temp	Humid	Wind	Water	Forecst	Enjoy
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes
			…			…

- Banana: yellow, thin, medium, sweet
- Watermelon: green, round, big, sweet
- Banana: yellow, thin, medium, sweet
- Grape: green, round, small, sweet
- Grape: red, round, small, sour
- …

Decision tree – Concepts



Example 2: Fruits



Decision tree – Milestones

- In 1966, first proposed by Hunt
- In 1970's~1980's
  - CART by Friedman, Breiman
  - ID3 by Quinlan
- Since 1990's
  - Comparative study (Mingers, Dietterich, Quinlan, etc)
  - Most popular DTree algorithm: C4.5 by Quinlan in 1993

Classical Decision Tree Algorithms

CART (classification and regression trees)

A general framework:

- Create or grow a decision tree using training data
- Decision tree will progressively split the set of training examples into smaller and smaller subsets
- Stop splitting if each subset is pure
- Or accept an imperfect decision

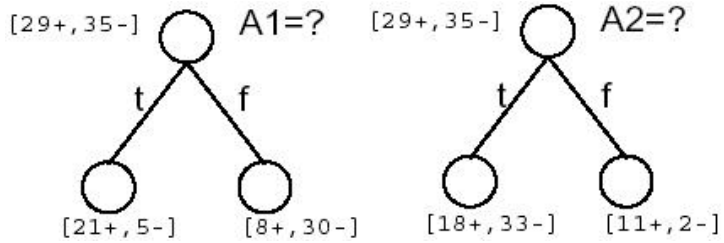
Many DTree algorithms follow this framework, including ID3, C4.5, etc.

Classical DTree Alg. – ID3

Outlook

- Top-down, greedy search
- Recursive algorithm
- Main Cycle:
  - A : the best decision attribute for the next step
  - Assign A as decision attribute for node
  - For each value of A (v_i), create new descendant of node
  - Sort training examples to leaf nodes
  - If training examples perfectly classified, Then RETURN, Else drill down to new leaf nodes

ID3 Q1: Which attribute is the best one?

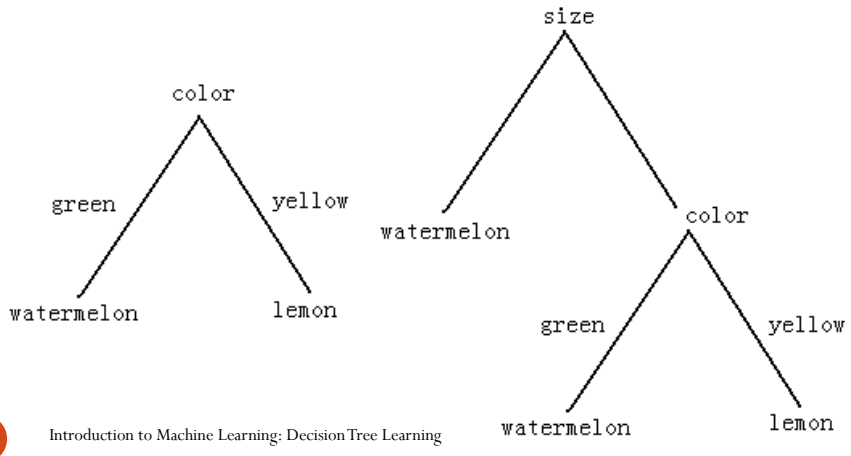


Outlook, Humidity, Wind, ….. ?

Query selection and node impurity

- Fundamental principle: **simplicity**

  We prefer decisions that lead to a simple, compact tree with few nodes



Query selection and node impurity

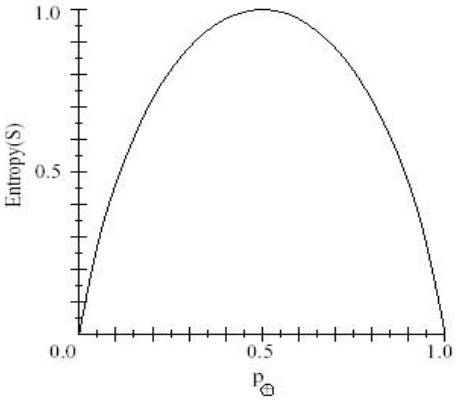
- Fundamental principle: **simplicity**
  - We prefer decisions that lead to a simple, compact tree with few nodes
- We seek a property query T at each node N that makes the data reaching the immediate descendent nodes as "pure" as possible
- Purity – Impurity

How to measure impurity?

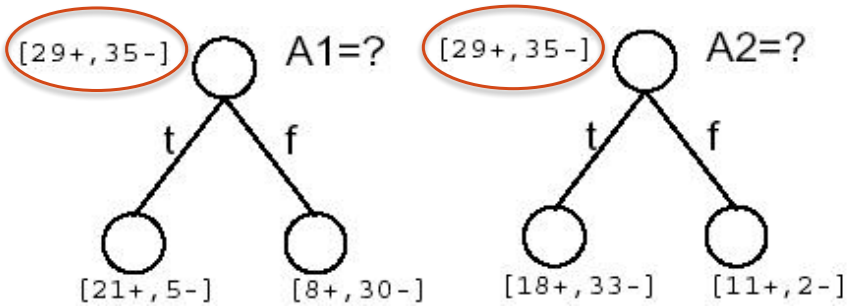
Entropy impurity (is frequently used)

$$Entropy(N) = -\sum_j P(w_j)\log_2 P(w_j)$$

- Define: 0log0=0
- In information theory, entropy measures the **purity/impurity** of information, or the uncertainty of information
- Uniform distribution – Maximum value of entropy



Entropy



$$Entropy(S) = -\frac{29}{64}\times\log_2\frac{29}{64} - \frac{35}{64}\times\log_2\frac{35}{64} = 0.993$$

Besides Entropy Impurity

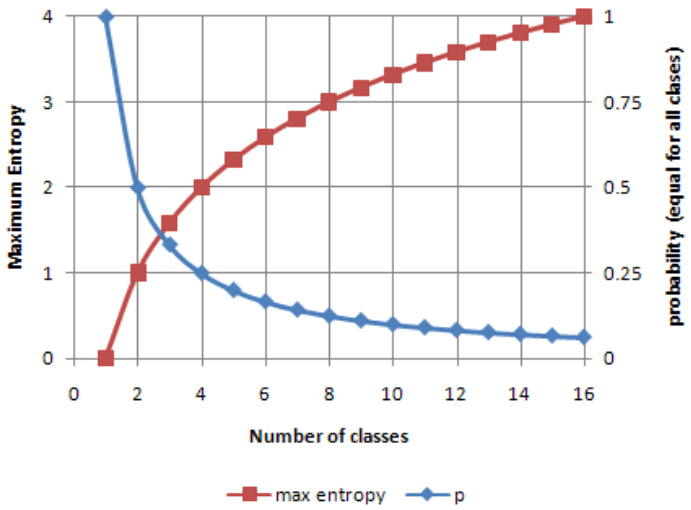
- Gini impurity (Duda prefers Gini impurity)

$$i(N) = \sum_{i\neq j} P(w_i)P(w_j) = 1 - \sum_j P^2(w_j)$$

- Misclassification impurity

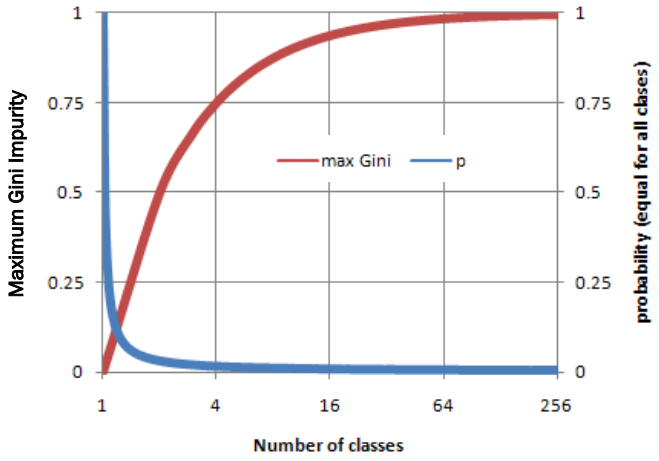
$$i(N) = 1 - \max_j P(w_j)$$

Impurity (Entropy)



Impurity (Gini)

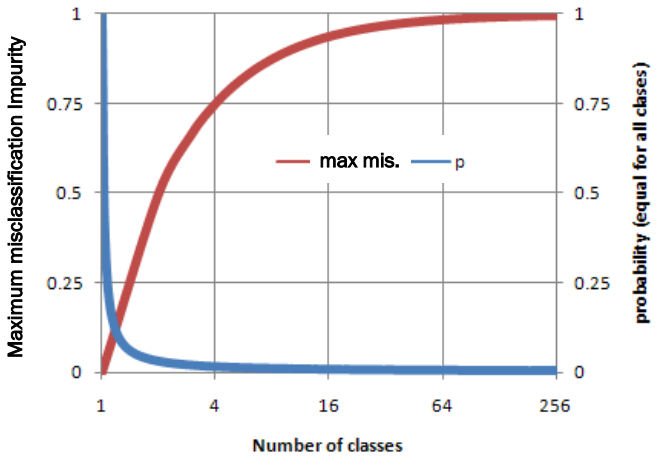
Maximum Gini impurity happens at $1-n*(1/n)^2=1-1/n$



Impurity (Misclassification)

Maximum Gini impurity happens at $1-n*(1/n)^2=1-1/n$

For n classes, Maximum Misclassification impurity = Maximum Gini impurity



Measuring the change of impurity ΔI(N) — Information Gain (IG), for example

- Expected reduction in entropy due to sorting on A

$$Gain(S,A) \equiv \underline{Entropy(S)} - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

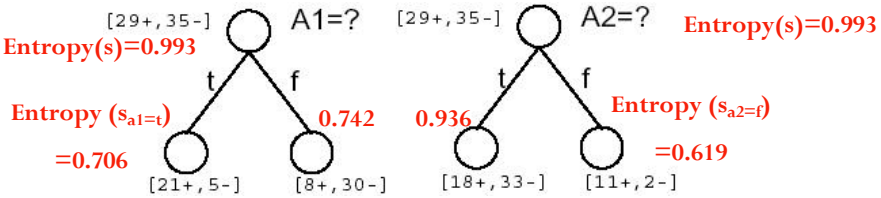
Entropy of Original S

Expected entropy after sorting on A

Information Gain, IG

- Expected reduction in entropy due to sorting on A

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



Entropy(s)=0.993

Entropy ($s_{a1=t}$) =0.706

0.742

0.936

Entropy ($s_{a2=f}$) =0.619

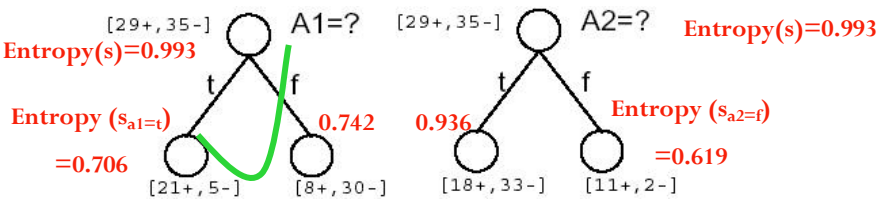
Entropy(s)=0.993

$$Entropy(S) = -\frac{29}{64}\times\log_2\frac{29}{64} - \frac{35}{64}\times\log_2\frac{35}{64} = 0.993$$

Information Gain, IG

- Expected reduction in entropy due to sorting on A

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



Entropy(s)=0.993

Entropy ($s_{a1=t}$) =0.706

0.742

0.936

Entropy ($s_{a2=f}$) =0.619

Entropy(s)=0.993

$$Gain(S,A_1) = 0.993 - (\frac{26}{64}\times 0.706 + \frac{38}{64}\times 0.742) = 0.266, \quad Gain(S,A_2) = 0.121$$

ID3 Q2: when to RETURN (stop splitting) ?

- "If training examples perfectly classified"
- Condition 1: if all the data in the current subset has the same output class, then stop
- Condition 2: if all the data in the current subset has the same input value, then stop

Possible condition 3: if all the attributes' IG scores are 0, then stop

A good idea ?

ID3 Q2 : when to RETURN (stop splitting) ?

- y=a XOR b

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

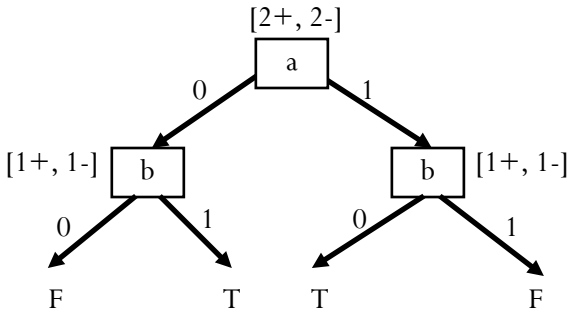
Information Gain:

Attr	value	probability	IG
a	0	50%	0
	1	50%	
b	0	50%	0
	1	50%	

- According to the proposed condition 3, No attribute could be chosen even at the first step.

ID3 Q2 : when to RETURN ?

- If we ignore the proposed condition 3



There're ONLY 2 conditions for stopping splitting in ID3 :

- The same output class or The same input value

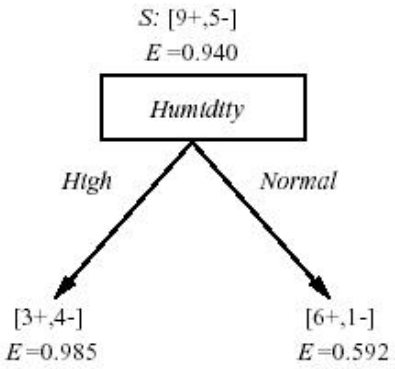
Discussion: If they have same input but diff. output, what does it mean?

ID3 example: training samples

High: 3+,4 -; Normal: 6+,1- Total: 9+, 5- ;

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

ID3 example: feature selection



Gain(S,Humidity)

=0.940－(7/14)*0.985-
(7/14)*0.592

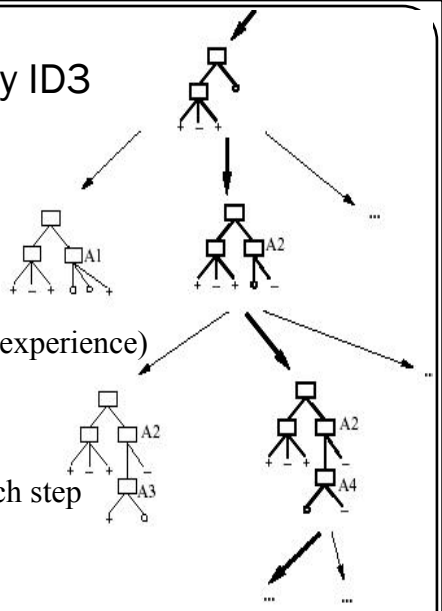
=0.151

Gain(S,Outlook)=0.246

Gain(S,Wind)=0.048

Gain(S,Temperature)=0.029

Hypothesis space search by ID3



- Hypothesis space is complete
  - Target function surely in there
- Output a single hypothesis
  - Can't play over 20 questions (by experience)
- No back tracking
  - Local minima…
- Use all the data in the subset for each step
  - Statistically-based search choices
  - Robust to noisy data

(to be continued…)