CSCI 5408 Data Analytics: DM and DW Tech
(Mar 28, Week 12)

- Ass5 Due: Mar 28 (Today)
- Ass6 Due: Apr 11
  - Read Assignment 6.pdf
  - Ass6-Tutorial: Mar 29 (Tomorrow)
  - Read Ass6-Tutorial slides
  - Help Hours: Fri, 1:00-2:30 PM, CS 233
- Final Exam: Apr 20, 3:30-5:30 PM
- Write answers for review questions
- Reading:
  - Lectures 18 & 19
  - Text 3rd: 8.1-8.3, or 2nd: 6.1, 6.2-4, 6.6, 6.16

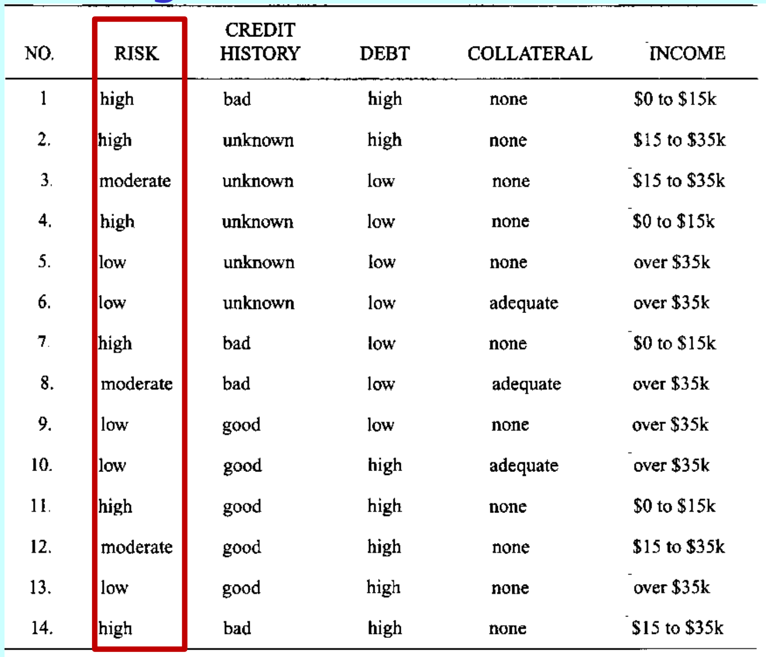
5. Classification DM

(Text 3rd: 8.1-8.3 / 2nd: 6.1, 6.2-4, 6.6, 6.16)

- Classification problem overview
- General issues of classification DM
- Mining classification model by decision tree induction
- Bayesian classification
- Text classification
- Other classification methods
- Summary

Recap: DT Classification

Training Data Set D:



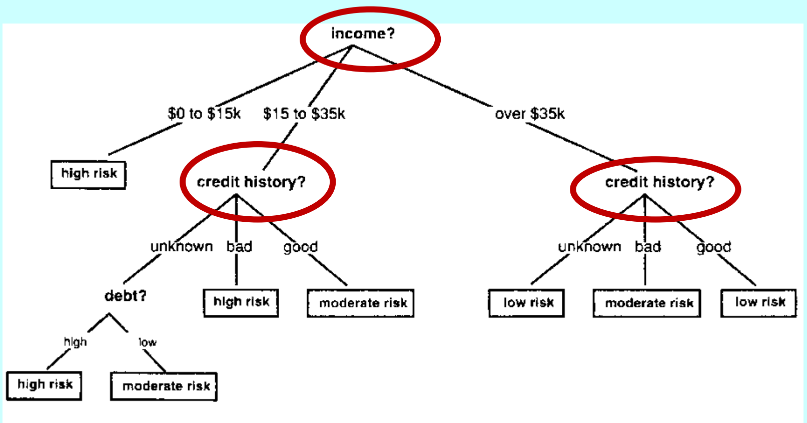
NO.	RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
1	high	bad	high	none	$0 to $15k
2.	high	unknown	high	none	$15 to $35k
3.	moderate	unknown	low	none	$15 to $35k
4.	high	unknown	low	none	$0 to $15k
5.	low	unknown	low	none	over $35k
6.	low	unknown	low	adequate	over $35k
7.	high	bad	low	none	$0 to $15k
8.	moderate	bad	low	adequate	over $35k
9.	low	good	low	none	over $35k
10.	low	good	high	adequate	over $35k
11.	high	good	high	none	$0 to $15k
12.	moderate	good	high	none	$15 to $35k
13.	low	good	high	none	over $35k
14.	high	bad	high	none	$15 to $35k

Data from credit history of loan applications

DT:

Decision node: ⬭

Leave node: ▭



A simplified decision tree for credit risk assessment.

Predication by the learned classifier (DT):

- Input: <RISK=?, CREDIT HIS=good, DEBT=low, COLLATERAL=unknown, INCOME=$30k>
- Output: RISK= *moderate*

DT Induction: Divide & Conquer

```
Input:
    $D$      //Training data
Output:
    $T$      //Decision Tree
DTBuild Algorithm:
             //Simplistic algorithm to illustrate naive approach to building DT
    $T = \emptyset$;
    Determine best splitting criterion;
    $T$ = Create root node node and label with splitting attribute;
    $T$ = Add arc to root node for each split predicate and label;
    for each arc do
        $D$ = Database created by applying splitting predicate to $D$;
        if stopping point reached for this path then
            $T'$ = Create leaf node and label with appropriate class;
        else
            $T' = DTBuild(D)$;
        $T$ = Add $T'$ to arc;
```

Choose attribute to split

Split

Test

Choose attribute to split

Three Major Steps of ID3

- Choosing an attribute for splitting
  - Calculate information gain for choosing the best attribute
- Splitting
  - Multi-way dividing based on the distinct values of a selected attribute
- Testing
  - End, or call again at each move (added additional condition: majority voting)

ID3 Algorithm

```
ID3(Examples, Target, Attributes)
    // Examples - the training tuples. Target - the attribute whose value is to be predicated by the tree.
       Attributes - the list of other attributes available for evaluation.
       It returns a decision tree that correctly classifies the given Examples.
· Create a Root node for the tree          // The current set of Examples
· If all Examples are positive, Return the single-node tree Root, with label = "+"
· If all Examples are negative, Return the single-nod tree Root, with label = "-"
· If Attributes list is empty, Return the single-node tree Root, with label = most
  common value of Target in Examples   // Use majority voting and provide confidence measure
· Otherwise Begin
    - A ← the attribute from Attributes that best classifies Examples measured by Gain(Examples, A).
    - The decision attribute for Root ← A
    - For each possible value, v_i, of A:
          - Add a new tree branch below Root,  corresponding to the test A = v_i
          - Let Examples_v_i, be the subset of Examples that have value v_i  for A
          - Test conditions for stopping partitioning for creating a leaf node
              - If all samples for a given node belong to the same class, create a leaf node
              - If no remaining attributes for further partitioning – majority voting
                for classifying the leaf node
              - If no samples left, then below this new branch add a leaf node with label = most
                  common value of Target_attribute in Examples
          - Else below this new branch add the subtree:
            ID3(Examples_(v_i), Target_attribute, Attributes - {A})
· End
· Return Root
```

Induction/DT construction: testing->choose->split

1. Test *D:*

D (Outlook, Temperature, Humidity, Wind, PlayTennis)

+: X3,X4,X5,X7,X9,X11,X12,X13 Yes for PlayTennis

--: X1,X2,X6,X8,X14 No for PlayTennis

2. Choose:

3. Split:

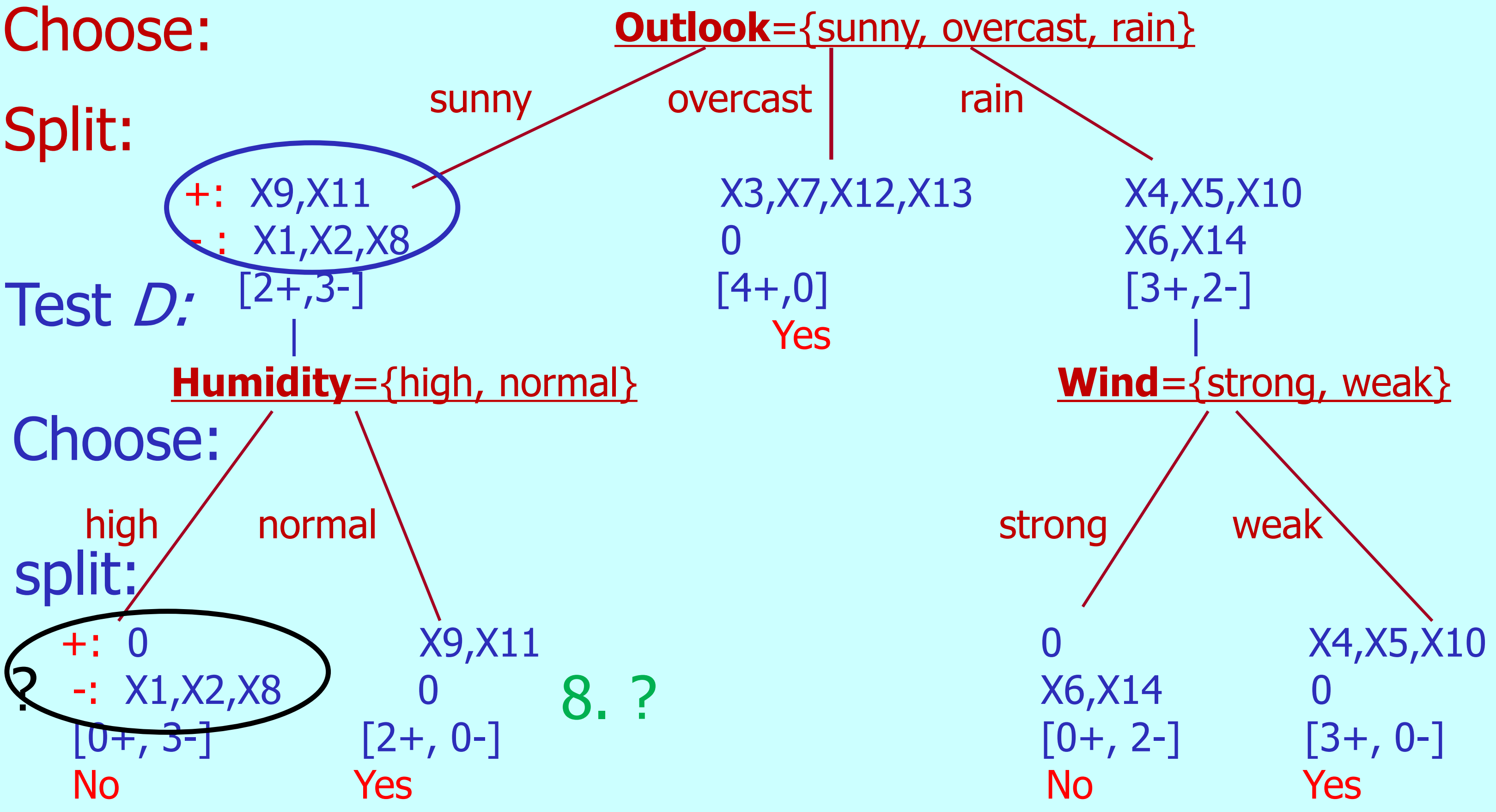
4. Test *D:*

5. Choose:

6. split:

7. ?

8. ?



Main Issues of DT Induction

1. How to construct the tree:
   – Greedy search
2. How to choose an attribute to split:
   – Information gain (ID3)
3. How to determine when to stop splitting:
   – i.e. Reached a leave node when: single class, no attribute left (majority votes), no sample left (track back to previous node).

Determine the Best Split

- **Which attribute is the best classifier?**
  - How well a given attribute split the records for classification
  - We want to select the best one to split
- **What is a good quantitative measure?**
  - **Information Gain** (a probability property)
- ID3 uses information gain to select among the candidate attributes at each step while growing the tree

How to measure the information contained in an attribute for classification of the target concept?

How to Measure Information?

- Information in general always associates with a question and a message for answering the question

- Information measure is to quantify the outcome of some expectation from the message for answering the question

How to quantify information?

- **In information theory, information is measured in bits**
  - One bit of information is enough to answer a yes or no question about which one has no idea, such as coin flipping.

- **How to derive the quantity of information?**
  - Information is defined in terms of the number of bits necessary to encode the factor by which <u>the probability of the particular outcome has been increased:</u>

Q(message) = P2(outcome_after) - P1(outcome_before)

Q, the quantity of information contained in a message.

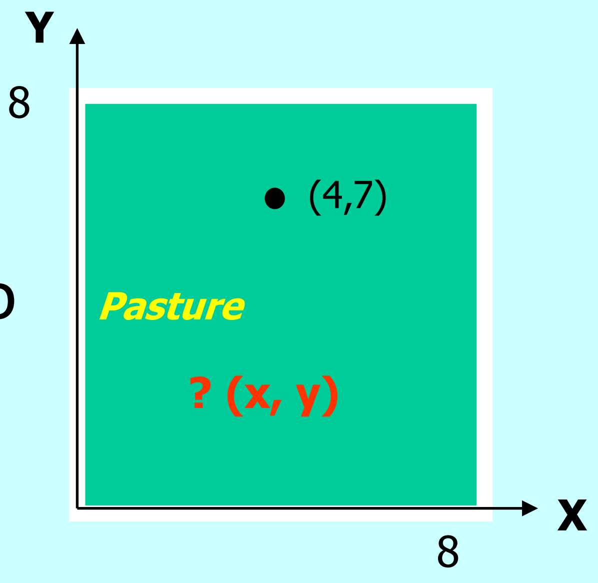
P1, the probability of outcome before receiving a message.

P2, the probability of outcome after receiving the message.

"Information"

Information measure

- an illustration example:



Suppose a missing cow has strayed into a pasture represented as an 8 x 8 array of "cells".

Question: Where is the cow?

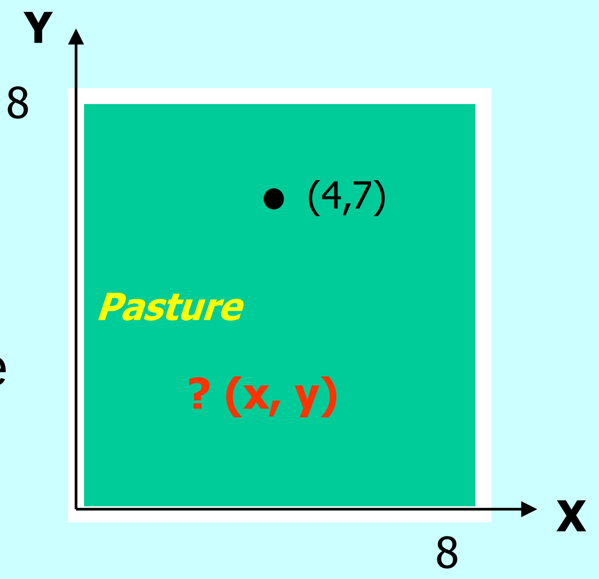
Outcome: the probability of finding the cow.

We want to get the **information about <u>finding the cow.</u>**

Answer 1: Nobody knows.

Answer 2: The cow is in cell (4, 7).

What is the probability of outcome expectation?



Outcome1: Without knowing which cell it is in, the information about <u>where is a missing cow</u>:

P1 (find-the-cow_before) = 1/64

Outcome2: If some one sends a **message** telling which cell the cow is in, e.g. in **<u>cell (4,7)</u>** then we acquired some information.

<u>How much information we received about find the cow?</u>

P2 (find-the-cow_after) = 1

The information has been gone from 1/64 to 1 for a-factor-of 64 increase.

Information Measure:

The information has been gone from 1/64 to 1 for a-factor-of 64 increase.

The quantity of information in a message can be formally defined in terms of the number of bits necessary to encode the factor by which the probability of the particular outcome has been increased:

E.g., log2 (64) = 6 bits.

Information: *The increase of outcome expectation*

The information of finding the cow contained in the message "*The crow is in cell (4,7)”:*

Q(message) = P2(after) - P1(before)

Information received = $\log_2 P2 - \log_2 P1$

$= \log_2 (P2 / P1)$ // $P1 = 1/64$, $P2 = 1$

$= \log_2 64$

$= 6$ bits

What is the probability of an unseen record being "Yes" class based on the training dataset without asking any question?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	sunny	hot	high	weak	No
2	sunny	hot	high	strong	No
3	overcast	hot	high	weak	Yes
4	rain	mild	high	weak	Yes
5	rain	cool	normal	weak	Yes
6	rain	cool	normal	strong	No
7	overcast	cool	normal	strong	Yes
8	sunny	mild	high	weak	No
9	sunny	cool	normal	weak	Yes
10	rain	mild	normal	weak	Yes
11	sunny	mild	normal	strong	Yes
12	overcast	mild	high	strong	Yes
13	overcast	hot	normal	false	Yes
14	rain	mild	high	strong	No

Information measure for classification: Class Purity

E.g., Let’s look at classification for the concept PlayTennis for answer (i.e. class) “Yes”:

- For expectation: PlayTennis = Yes
  - P1_yes(before) = 9/14
    - The probability of a unseen record being “Yes” class without asking any question.
    - Similarly for “No” P1_no (before) = 5/14
  - For P2_yes(after) = ?

Which question should be asked from {Outlook, Temperature, Humidity, Wind}?

Which question should be asked from the give attributes (regarding class purity)?

- **P2_yes(after) =?**

The questions may be asked like "what is outlook?“, or “what is Temperature?”, etc. from the list of the attributes {Outlook Temperature Humidity Wind}.

- **E.g., What is Outlook?**
  - Then the message can be from

    P2_yes(Outlook=sunny) = 2/5,

    P2_yes(Outlook= overcast) = 4/4,

    P2_yes(Outlook= rain) = 3/5.

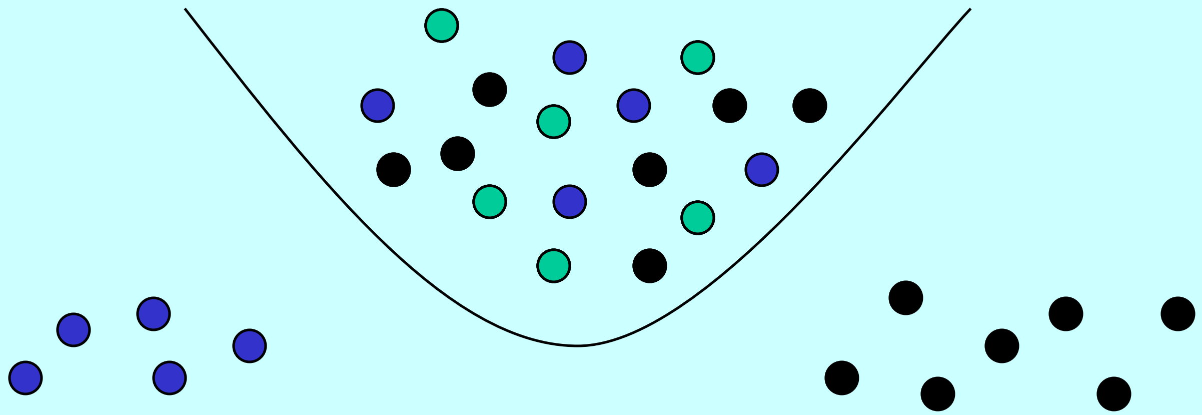
- **P2_no(after) =?**
  - …

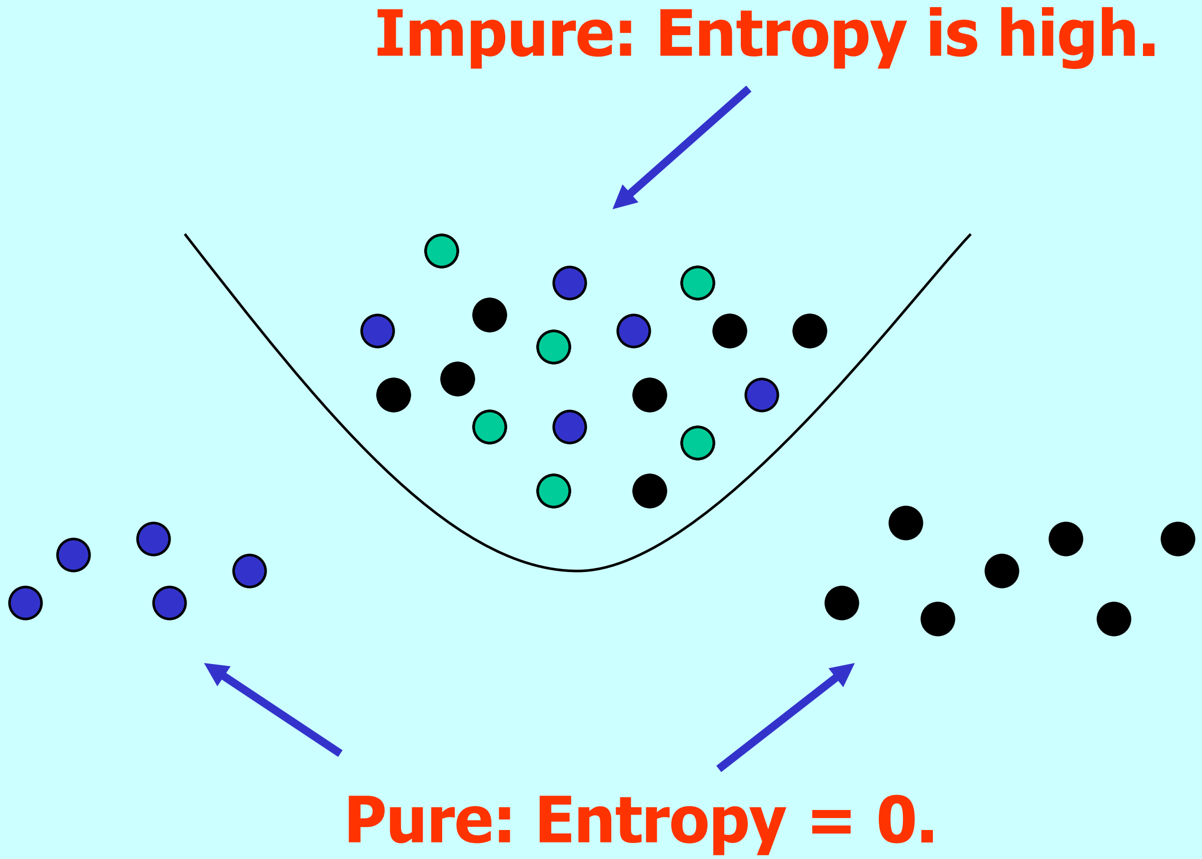
How to quantify the information about containing both “Yes” and “No” classes? <u>**Entropy.**</u>

What is Entropy?

- A borrowed notion from physics
  - In thermodynamics, entropy is a measure of the disorder in a system of particles. Entropy is a measure of how much confusion, uncertainty, or disorder there is in a situation/system.
- In information theory
  - Entropy characterizes the impurity (or uncertainty) of an arbitrary set of data
  - For classification, entropy is used to estimate how close a data set to a single class.

Entropy: a measure of impurity of data




Impure: Entropy is high.
Pure: Entropy = 0.

Expected Information: The increase of outcome expectation

The quantity of the information, "The crow is in cell (4,7)", about finding the cow is therefore:

Information_quantity(message) = P2(after) - P1(before)

Information received = $\log_2$ P2 - $\log_2$ P1

= $\log_2$ (P2 / P1) = $\log_2$ (1 / (1/64))

= $\log_2$ 64 = 6 bits

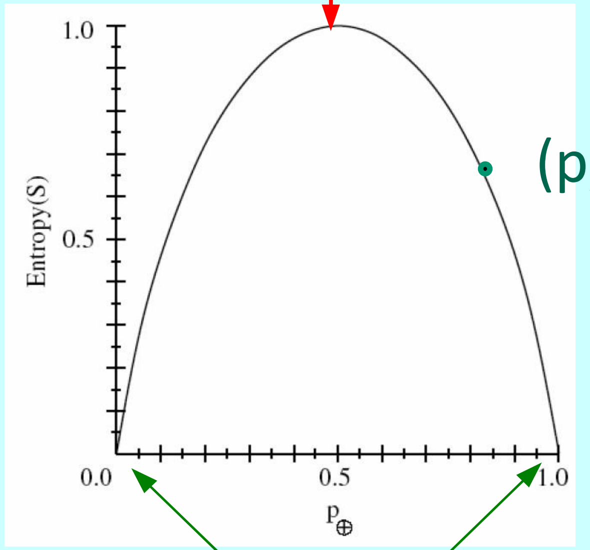
In general, when we learn the answer to a question where the probability of a particular answer was *p*, the quantity of information received is "$\log_2 (1/p)$" or "$-\log_2 (p)$".

Answers for multiple expectations

- Where is the cow for the farm problem?
  - The probability of finding the cow: p, which is for a single event/expectation.
- Application Weather & Play-Tennis?
  - **Multiple expectations:** If the probability of a training example belongs to “Yes” is “p”, then the probability it belongs to “No” should be “1-p”.
  - We need to find a measure function to evaluate multiple expectations
  - **What is the best question:** We have multiple questions to choose for asking, such as if temperature is … then…; or if outlook is… then…, etc.

Entropy Property for Binary Classes

maximal value at equal class distribution



(p, 1-p)

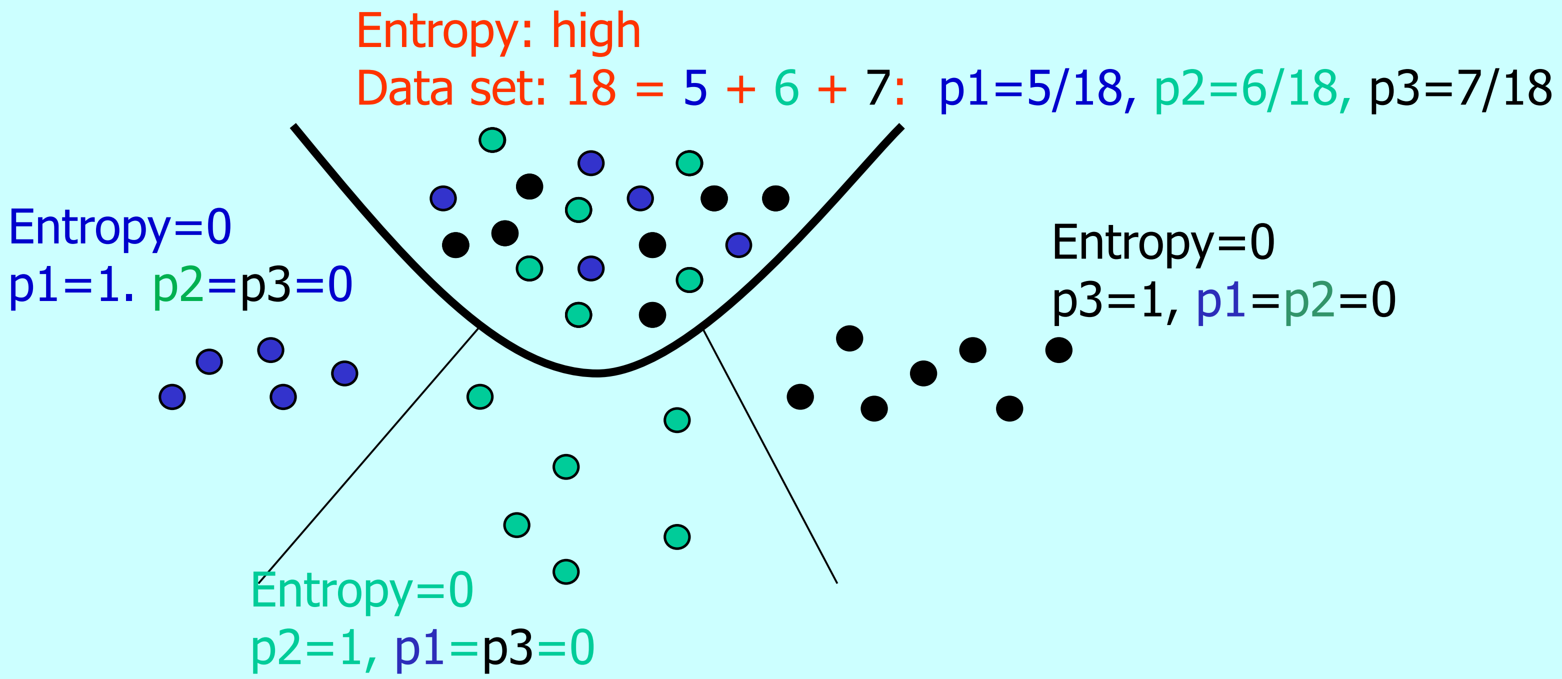
minimal value if only one class left in *S*

Entropy is a measure of both events (i.e. expectations, such as *yes* & *now*):
$-p*\log_2 p - (1-p)*\log_2(1-p)$, or

$$p*\log_2(1/p) + (1-p)*\log_2(1/(1-p))$$

yes *no*

Entropy for data sets with 3 expectations:



Entropy (5, 6, 7) = - (5/18)$\log_2$(5/18) - (6/18)$\log_2$(6/18) – (7/18)$\log_2$(7/18)
= 0.970

Entropy(5, 0, 0) = 0

Entropy(0, 6, 0) = 0

Entropy(0, 0, 7) = 0

Entropy Calculation

- Entropy formula:

$$\boxed{Entropy(S) = - \sum_{i=1}^{n} p_i \times \log p_i}$$

  - Where S, is the data set, p_i, stands for a subset of S belonging to a class of the target; n, is the number of target classes
  - $p_1, p_2, \ldots, p_n$ are probabilities and $p_1 + p_2 + \ldots + p_n = 1$

- E.g., For PlayTennis dataset:
  - Entropy (9+,5-) = - (9/14)$\log_2$(9/14) - (5/14)$\log_2$(5/14) = 0.940
  - Entropy is maximized for $p_1 = p_2 = \ldots = p_n = 1/n$, and entropy is minimized (zero) for $p_i = 1$ for some i, and $p_i = 0$ for all $j \neq i$
  - For classification, entropy can measure how close is a set of records to a single class of a given target concept.

E.g., PlayTennis dataset:

```
E.g.,   +: X3,X4,X5,X7,X9,X11,X12,X13     Yes for PlayTennis
        -:  X1,X2,X6,X8,X14                No  for PlayTennis
                              Outlook
                   ________/     |      \________
              sunny          overcast              rain
               /                 |                    \
  +:  X9,X11                  X3,X7,X12,X13         X4,X5,X6
  - :  X1,X2,X8                0                    X10,X14
      [2+,3-]                 [4+,0]                [3+,2-]
        |                     Yes                     |
```

Entropy(S_sunny) = **Entropy(2+,3-)** = $-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)$ = **0.971**
Entropy(S_overcast) = **Entropy(4+,0-)** = $-(4/4)\log_2(4/4) - (0/4)\log_2(0/4)$ = **0**
Entropy(S_rain) = **Entropy(3+,2-)** = $-(3/5)\log_2(3/5) - (2/5)\log_2(2/5)$ = **0.971**

To measure overall partition result:

Expected Entropy = $EE(Outlook) = \sum S_j/S \; Entropy\,(S_j)$

Can you justify "Outlook" is the best pick?
- Try the necessary computation for comparison.

Recap Attribute Evaluation Method-Information Gain

Information gain measures the <u>expected reduction</u> in <u>Entropy</u>:

$$Gain(S, A) = Entropy(S) - \sum^{k} S_j/S \; Entropy(S_j)$$

– Entropy formula: $Entropy(S) = -\sum^{n}_{i=1} p_i \times \log p_i$

Gain (S,A), is the information about the purity change of the examples about the classification on the target using attribute A.

Where k is the number of different values of the attribute currently under consideration, and j, ranges from 1 to k.

A good partition will produce a small expected entropy, in turn, the Gain will be bigger.

E.g., Information Gain for Outlook:

E.g., The set S has 14 examples: i.e. **S = 14 = [9+,5-]** for the target PlayTennis = {yes,no}. So we have

Entropy(S) = Entropy (9+,5-) = - (9/14)$\log_2$(9/14) - (5/14)$\log_2$(5/14) = 0.940

The atttribute **Outlook** = {sunny, overcast, rain} partitions the set into subsets:

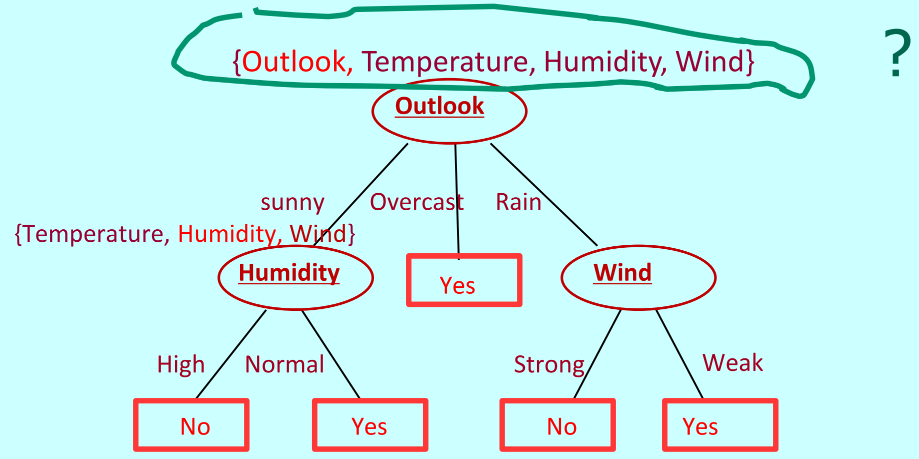
S_sunny = 5 = [2+,3-]
S_overcast = 4 = [4+,0-]
S_rain = 5 = [3+,2-]

Gain(S,Outlook) = Entropy(S) - (5/14)*Entropy(S_sunny)
- (4/14)*Entropy(S_overcast)
- (5/14)*Entropy(S_rain)
= **0.246**

Where,

Entropy(S) =	Entropy(9+,5-) =	-(9/14)log_2(9/14)	- (5/14)log_2(5/14)	= 0.940
Entropy(S_sunny) =	Entropy(2+,3-) =	-(2/5)log_2(2/5)	- (3/5)log_2(3/5)	= 0.971
Entropy(S_overcast) =	Entropy(4+,0-) =	-(4/4)log_2(4/4)	- (0/4)log_2(0/4)	= 0
Entropy(S_rain) =	Entropy(3+,2-) =	-(3/5)log_2(3/5)	- (2/5)log_2(2/5)	= 0.971

Choose which Attribute?



The information gain calculations for all four attributes:

Gain(S,Outlook) = 0.246
Gain(S,Humidity) = 0.151
Gain(S,Windy) = 0.048
Gain(S,Temprature) = 0.029

The attribute Outlook may lead to a best partition of S in terms of the target PlayTennis.

The partitioned subsets are overall more pure (more homogenous) than other partitions (the best classifier).

What is ID3's inductive bias?

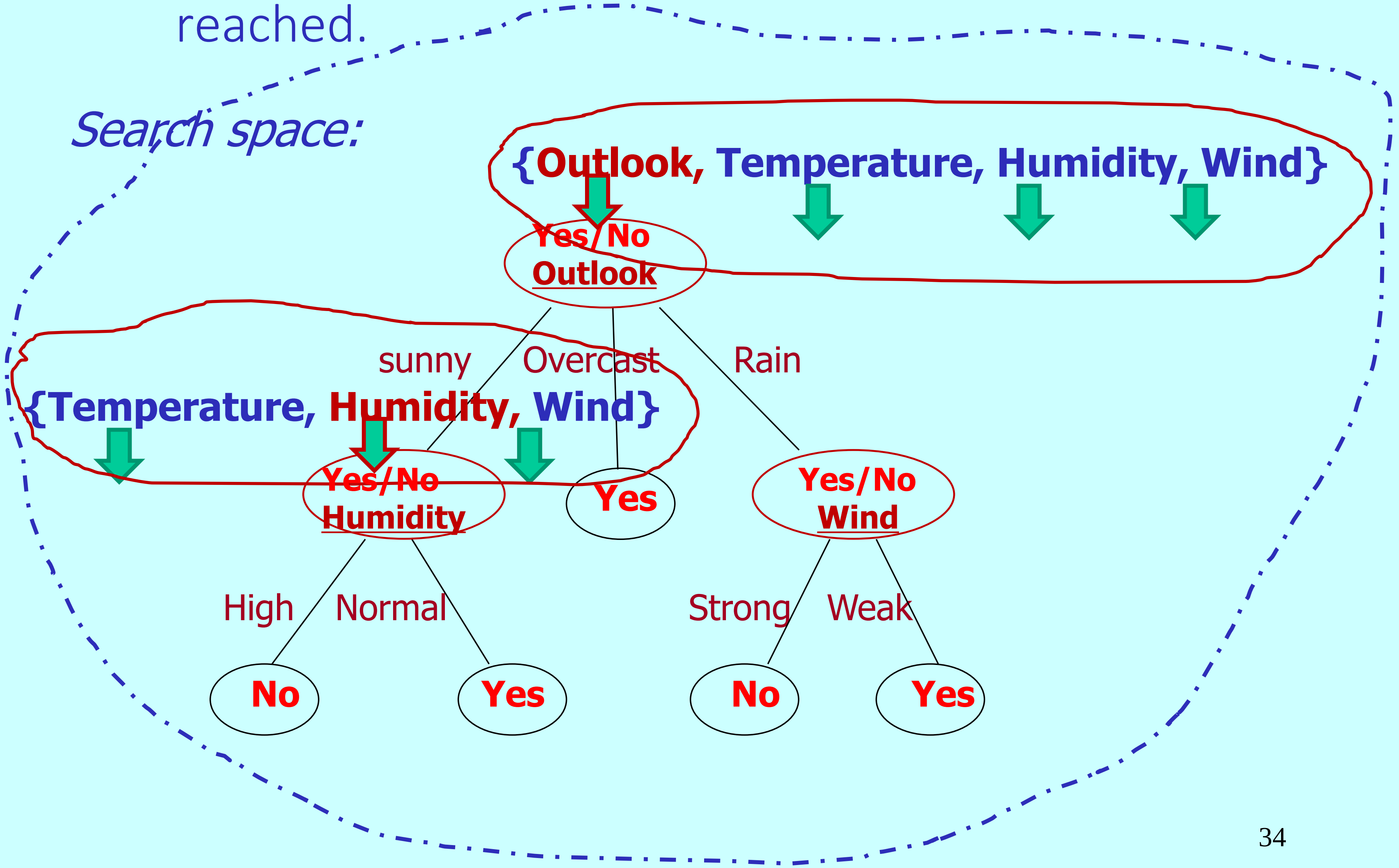
- **Approximate inductive bias of ID3:**
  - ID3 chooses the first acceptable tree it encounters (greedy search strategy)
  - Shorter trees are preferred over longer trees
    - Trees that place highest information gain attributes close to the root are preferred over those that do not.
  - It has a tendency of choosing an attribute with more distinct values than an attribute with fewer values

What is ID3's inductive bias?

- **Approximate inductive bias of ID3:**
  - ID3 chooses the first acceptable tree it encounters (greedy search strategy)
  - Shorter trees are preferred over longer trees
    - Trees that place highest information gain attributes close to the root are preferred over those that do not.
  - It has a tendency of choosing an attribute with more distinct values than an attribute with fewer values

Any potential problem?

Greedy search: choose the best direction at each move using depth-first exploration if goal is not reached.

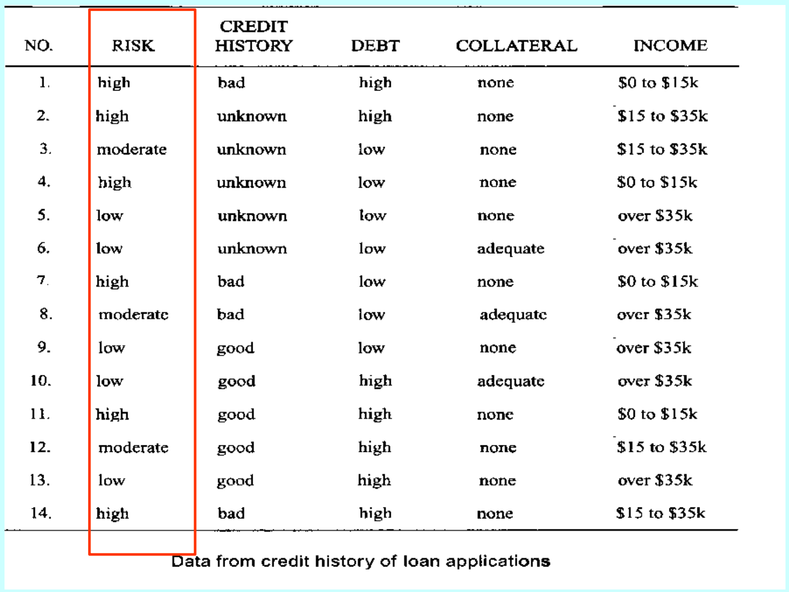


Hypothesis Space Search in DT Induction

The search space: all possible decision trees.

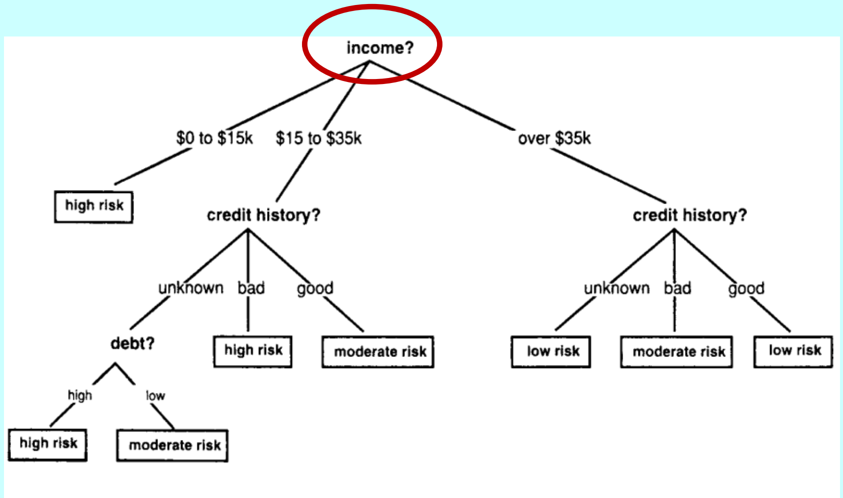
```
                         Root (Set S)
             {Outlook, Temperature, Humidity, Wind}
             ----------------------------------------------
                /             |               |           \
               /              |               |            \
          Outlook       Temperature      Humidity       Wind
         /    |    \        /   |   \         /   \          /   \
sunny overc rain  hot mild cold   high  normal   strong weak
  |        |      |     |      |      |        |        |       |      |
  |        o      |     |      |      |        |        |       |      |
  |               ...   |      ...    ...      |        ...     |      ...
{Temperature,   {Outlook,        {Outlook,          {Outlook,
   Humidity,       Humidity,        Temperature,      Temperature,
   Windy}          Windy}          Windy}             Humidity}
    /   |   \         / | \            / | \              / | \
   /    |    \       … … …            … …  …              …  … …
Hum  ...     ...
/ \
… …
```

Which DT is better, and Why?

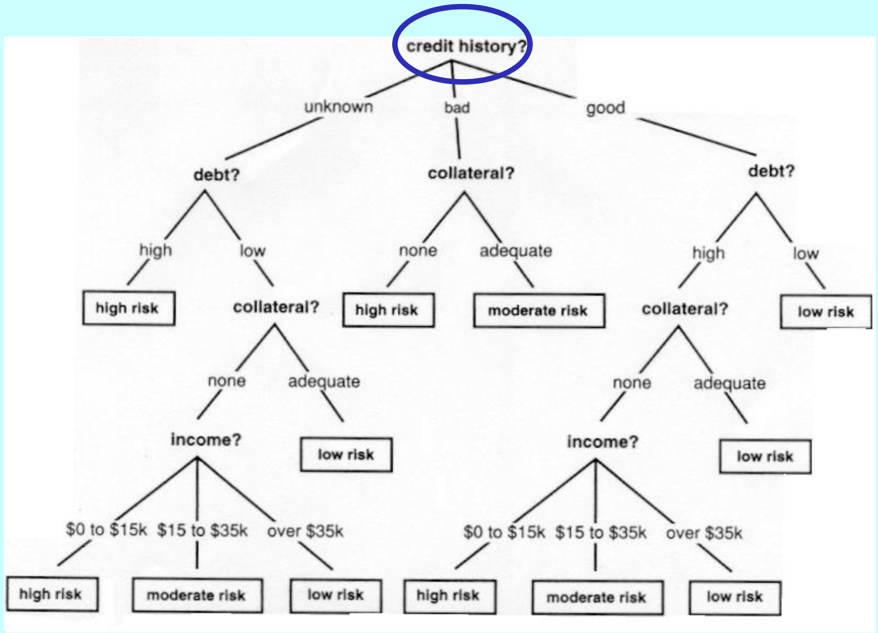


NO.	RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
1.	high	bad	high	none	$0 to $15k
2.	high	unknown	high	none	$15 to $35k
3.	moderate	unknown	low	none	$15 to $35k
4.	high	unknown	low	none	$0 to $15k
5.	low	unknown	low	none	over $35k
6.	low	unknown	low	adequate	over $35k
7.	high	bad	low	none	$0 to $15k
8.	moderate	bad	low	adequate	over $35k
9.	low	good	low	none	over $35k
10.	low	good	high	adequate	over $35k
11.	high	good	high	none	$0 to $15k
12.	moderate	good	high	none	$15 to $35k
13.	low	good	high	none	over $35k
14.	high	bad	high	none	$15 to $35k

Data from credit history of loan applications



A simplified decision tree for credit risk assessment.



- Number of rules: 8 vs. 12
- Depth of the tree: 3 vs. 4
- Ave examples of leaf nodes: …

DT Quality Evaluation

1. The average depth of the tree branches.
2. The number of leave nodes.
3. The average examples covered by a leave node.

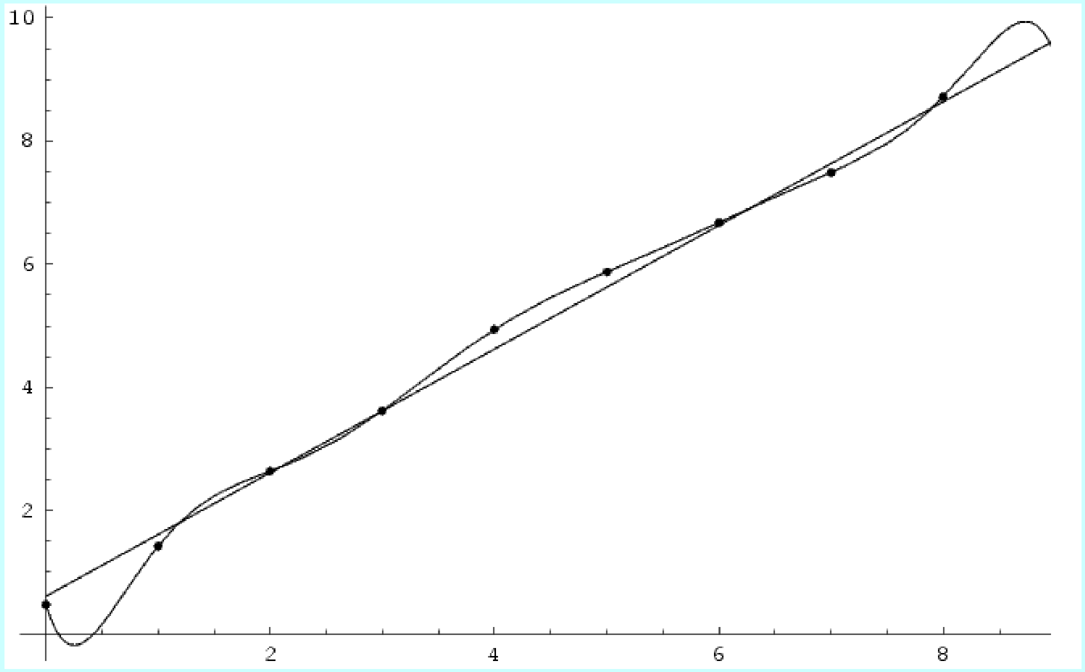
Issue of ID3 Inductive Bias

- **There is a natural bias in the information gain measure that favors attributes with many distinct values over those with few distinct values**
  - It tends to prefer attributes with many values, i.e. many divisions (subsets), and thus may lead to **over-fitting**
  - E.g. If attribute ID is used as an attribute, then what can happen?
    - In the extreme, an attribute that has a unique value for each tuple in the training set would be the best because there would be only one tuple (and thus one class) for each division
  - E.g., Compare the data sets and ID3 results: Ass/Ass4-demo: adult1 and adult2

What is overfitting?

- O**verfitting** is about a statistics model describes random error or noise instead of the underlying relationship.
- **Overfitting** generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.
- A model which has been **over fit** will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data.

Which function to choose?



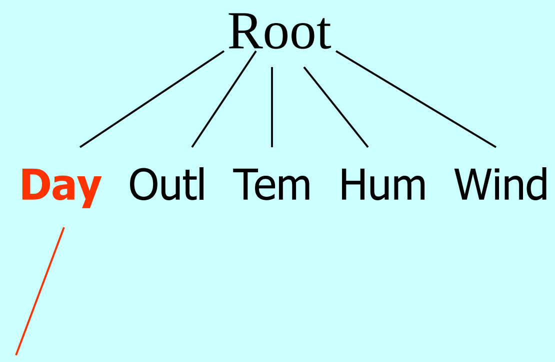
- Noisy (roughly linear) data is fitted to both *linear* and *polynomial* functions. Although the polynomial function passes through each data point, and the linear function through few, the linear version is a better fit. If the regression curves were used to predict the data, the overfit would do much worse.

What happened if add Day into the process?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	sunny	hot	high	weak	No
2	sunny	hot	high	strong	No
3	overcast	hot	high	weak	Yes
4	rain	mild	high	weak	Yes
5	rain	cool	normal	weak	Yes
6	rain	cool	normal	strong	No
7	overcast	cool	normal	strong	Yes
8	sunny	mild	high	weak	No
9	sunny	cool	normal	weak	Yes
10	rain	mild	normal	weak	Yes
11	sunny	mild	normal	strong	Yes
12	overcast	mild	high	strong	Yes
13	overcast	hot	normal	false	Yes
14	rain	mild	high	strong	No

Overfitting Example

As an extreme example, consider the attribute Day, which has a very large number of possible values.



Gain (S, Day) = E(S) – EE(Day)

= E(S) - ***0***

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	sunny	hot	high	weak	No
2	sunny	hot	high	strong	No
3	overcast	hot	high	weak	Yes
4	rain	mild	high	weak	Yes
5	rain	cool	normal	weak	Yes
6	rain	cool	normal	strong	No
7	overcast	cool	normal	strong	Yes
8	sunny	mild	high	weak	No
9	sunny	cool	normal	weak	Yes
10	rain	mild	normal	weak	Yes
11	sunny	mild	normal	strong	Yes
12	overcast	mild	high	strong	Yes
13	overcast	hot	normal	false	Yes
14	rain	mild	high	strong	No

- If add the attribute Day to the data, it would have the highest information gain
- This is because Day along perfectly predicts the target attribute over the training data.
- Obviously, Day is not a useful predictor despite the fact that it perfectly separates the training data.

How to deal with overfitting: Alternative measures for selecting attribute

- Many alternatives have been proposed, such as use gain ratio, which consider the probability of each attribute value, i.e take into account the cardinality of each division, as opposed to Gain
- **GainRatio(S,A)** = Gain(S, A) / Entropy (S1/S,…,Sv/S)
  - where (S1,…,Sv) is the subsets divided based on the values of A, and the split ignores the target classes
  - **E.g., Outlook = {sunny, overcast, rain} partitions the set into subsets:**
    S_sunny = [2+,3-] = 5
    S_overcast = [4+,0-] = 4
    S_rain = [3+,2-] = 5

GainRatio(S, Outlook) = Gain(S, Outlook) / Entropy (5/14, 4/14, 5/14)

GainRatio is an adjusted value of gain.

  - **The measure Gain Ratio has been used in C4.5 algorithm**

Handling noise and missing values

- ID3 uses available training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis
  - ID3 can easily handle noisy training data by modifying its **termination criterion** to accept hypotheses that imperfectly fit the training data
- ID3 can be modified to handling missing value in that
  - A missing value of an examples's attribute A may be filled based on a value that has highest frequency among various values of A
  - A more safe method is to detect all missing values in advance and either recovered the correct values or removed the associated tuples.

Tree Pruning

- To overcome over-fitting the data, the mining result (decision tree) may need to be further processed for pruning, i.e. make the rules
  more general (e.g. short trees).
- **Pre-pruning:**

  During the process of constructing the tree, stop split the subset of training samples at a given node according to threshold of some statistical measures.
- **Post-pruning:**

  Remove branches from a "fully growth" tree according the statistical significance of distributed classes.
  - E.g., C4.5 algorithm uses subtree replacement method. A subtree is replaced by a leaf node if this replacement result is within an allowed error rate

A Comparison of DT Algorithms

Algorithms	**ID3**	**C4.5**	**C5.0**	**CART**
Type of data	Categorical	Continuous and Categorical	Continuous and Categorical, dates, times, timestamps	continuous and nominal attributes data
Speed	Low	Faster than ID3	Highest	Average
Pruning	No	Pre-pruning	Pre-pruning	Post pruning
Boosting	Not supported	Not supported	Supported	Supported
Missing Values	Can’t deal with	Can’t deal with	Can deal with	Can deal with
Attribution Evaluation Formula	Information Gain	Informaiton gain ratio	Same as C4.5	Use Gini diversity index

(http://ijarcce.com/upload/2015/december-15/IJARCCE%2059.pdf)

Review Questions

1. What are the major technical issues of DT Induction approach?
2. How to quantify information contained in a message?
3. What are the definitions of entropy, expected entropy and information gain?
4. How Information Gain is applied in ID3 algorithm?
5. What is entropy and information gain? How to use information gain for choosing an attribute?
6. What is ID3's induction bias, explain why?
7. What are the technical options of overcoming the bias problem for possible improvement?