CSCI 5408 Data Analytics: DM and DW Tech
(Mar 30, Week 12)

- Ass6 Due: Apr 11
  - Read Assignment 6 & 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 18-20
  - 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

Probable model vs. Probable class

- DT induction:
  - What is the most probable classification model (hypothesis) given the training data?
    - Hypotheses: classification models
    - Number of hypotheses: <u>too many</u>

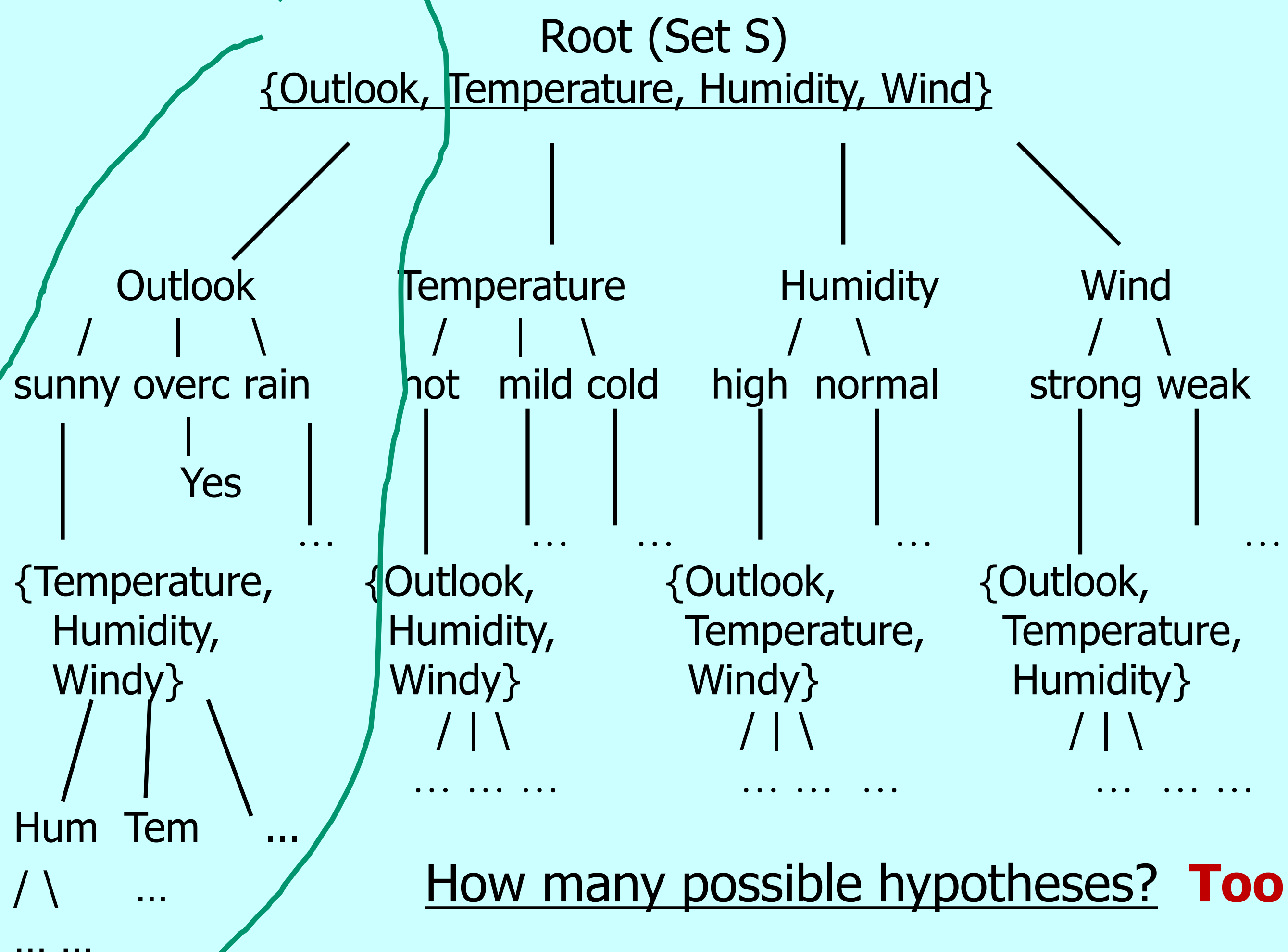
- Probabilistic prediction:
  - What is the most probable classification (class) of the new instance given the training data?
    - Hypotheses: classes of a new instance
    - Number of hypotheses: <u>few</u>

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  ...   ...
/ \
... ...
```

DT induction: the most probable classification model

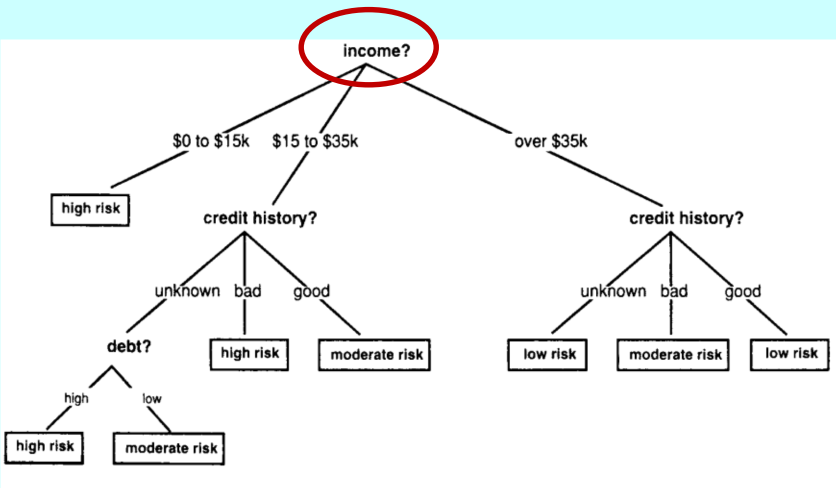


How many possible hypotheses? **Too many!**

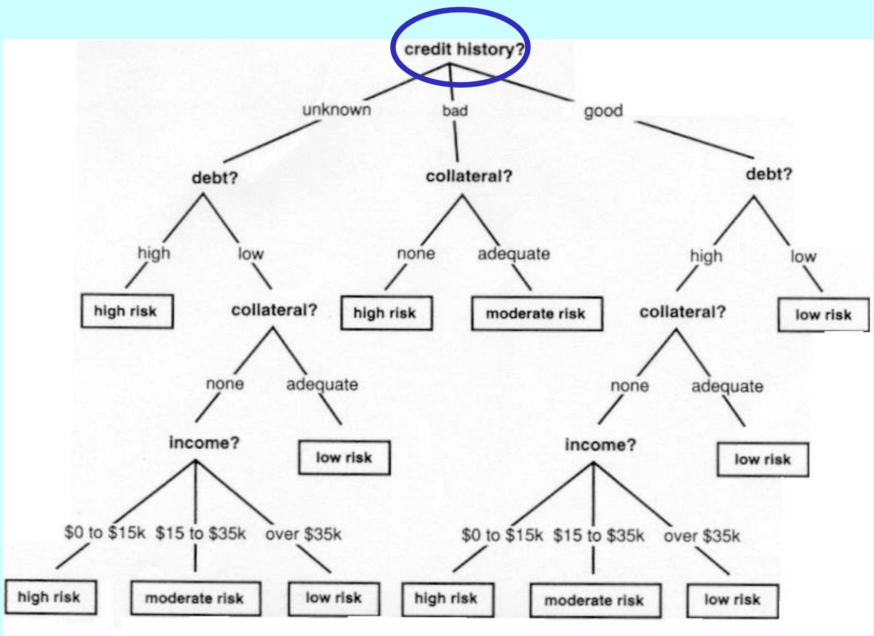
Which DT is better, and Why?

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: …

Bayesian classification: predict the most probable class of a new instance

- **How many hypotheses (classes) for a new instance?**
  - Play tennis: {yes, no}
  - Loan risk: {high, moderate, low}
  - Wage: {< $50k, ≥ $50k}
- **Inference rule:**
  - Bayes formula
- **Probabilistic prediction:**
  - Evaluate class hypotheses of a new instance, weighted by their posterior probabilities using Bayes rule.
  - The rule provides a standard of optimal decision making.

Bayesian Classification

- General properties:
  - Given a training set, the most probable classification of the new instance is obtained by choosing the hypothesis with <u>maximum posterior probability</u> of Bayesian rule
  - Incremental: Each training example can incrementally increase/decrease the probability (i.e. prior knowledge) that a hypothesis is correct

- Naïve Bayes Classifier:
  - One highly practical method: In some domains, its performance has been shown to be comparable to that of decision tree and neural network methods
  - Widely used for <u>text classification</u>

- **Classification by DT Induction approach:**
  - Mining knowledge: Find a probable model by searching hypothesis space
    - Hypotheses: classification models
    - Number of hypotheses: many
  - Classify each new case by the decision rules of DT

- **Classification by Bayesian approach:**
  - Mining knowledge: Calculate prior probabilities of the training data about target classes
  - Classify each new case by directly calculating the probable class based on the Bayes rule and the prior probabilities matching the case (i.e. comparing between few class hypotheses)

Review Bayes Theorem

- **Bayes's Rule:** (Thomas Bayes, 1702-1761)
  - If you have a hypothesis **h**, and an evidence **e** which supports the hypothesis, then

$$\boxed{P(h|e) = \frac{P(e|h)\,P(h)}{P(e)},} \qquad P(e) = P(e|h)P(h) + P(e|\sim h)P(\sim h)$$

Where:

P(h|e): a posterior probability in that probability **h** is true given evidence **e**.

P(h), P(e|h), P(e): prior probabilities (initial knowledge)

P(h): a probability that **h** is true overall.
P(e|h): a probability of observing evidence **e** when h is true.
P(e): a probability that **e** true overall.

- **Bayesian classification:**
  - To reason: e => h is to calculate P(h|e)

Illustration of Bayes' Rule Application

- **Example of medical diagnosis**

E.g., h = "John has malaria", and e = "John has a high fever."

General knowledge:

1. P(h): probability that a person has malaria in a particular season and region.
2. P(e|h): probability that a person has a high fever, given that he has malaria.
3. P(e): probability that a person has a high fever.

Given the general knowledge: $P(h) = 0.0001$, $P(e \mid h) = 0.75$, $P(e \mid \sim h) = 0.14$.

$P(e) = (0.75)(0.0001) + (0.14)(0.9999) = 0.14006$, and

P(h|e) $= P(e|h)P(h) / P(e) = (0.5)(0.0001) / 0.14006 = 0.0005354$, which is about 0.0005.

On the other hand, if John did not have the fever:

P(h|~e) $= P(\sim e|h)P(h) / P(\sim e) = (1 - 0.75)(0.0001) / (1 - 0.14006) = 0.0000029$

- **Bayesian classification:**

It provides a quantitative approach to weighing the evidence supporting alternative hypotheses.

E.g., Given the data set below, classify the input instance (sunny, cool, high, strong, ?) for the concept PlayTennis.

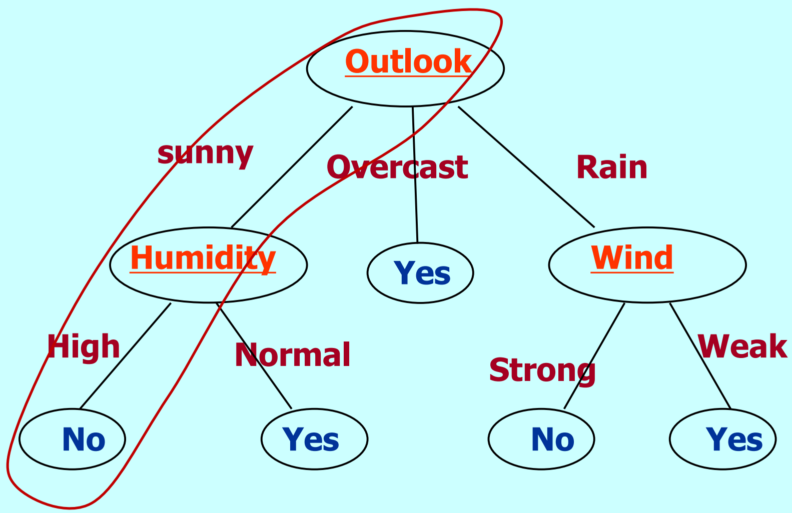
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes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

Compare Two Classification Methods:

Given a training dataset and a new input instance for classification:

Outlook	**Temperature**	**Humidity**	**Wind**	**Play-tennis**
sunny	cool	high	strong	?

ID3 (Model based)



Naïve Bayes Classifier (Instance based)

$$c_{NB} = \underset{j \in \{yes,\, no\}}{\operatorname{argmax}} P(c_j) \prod_{i=1}^{4} P(a_i \mid c_j)$$

P(yes|**e**) = 9/14 * 2/9 * 3/9 * 3/9 * 3/9 = 0.0053 = 20.5%
P(no|**e**) = 5/14 * 3/5 * 1/5 * 4/5 * 3/5 = 0.0206 = 79.5%

Bayesian Classification

- c_{MAP} = argmax P(c_j | a1,a1,..,ak) ← **P(h|e)=P(e|h) P(h)**

  = argmax P(a1,a1,..,ak | c_j) P(c_j)

Where C = {c1,c2, ..., cm}, the classes of the target attribute;
x = {a1,a2, ..., ak}, the values of the example x's attributes
(not including the target attribute).

E.g., The classifier for the target with classes {yes, no} :

$$\boxed{c_{MAP} = \underset{c_j \in \{yes,no\}}{\text{argmax}}\ P(c_j)\ \ P(a1,a2,a3,a4\ |c_j)}$$

Input instance = {Outlook=sunny, Temperature= cool,

Humidity=high, Wind= strong, PlayTennis=?}

Assumptions for Naive Bayes classifier

- Assumption 1:
  - The quantities of interest are governed by probability distributions and that optimal decisions can be made by reasoning about these probabilities together with observed data
    - Bayes' theorem provides a probability based reasoning solution for classification:

      $c_{MAP} = \operatorname{argmax} P(c_j \mid a1,a1,..,ak) = P(a1,a1,...,ak \mid c_j) * P(c_j)$

- Assumption 2:
  - Attributes are independent each other
  - Naive Bayes classifier:

$$c_{MAP} = \underset{c_j}{\operatorname{argmax}}\, P(c_j) * \prod_{i=1}^{n} P(a_i \mid c_j)$$

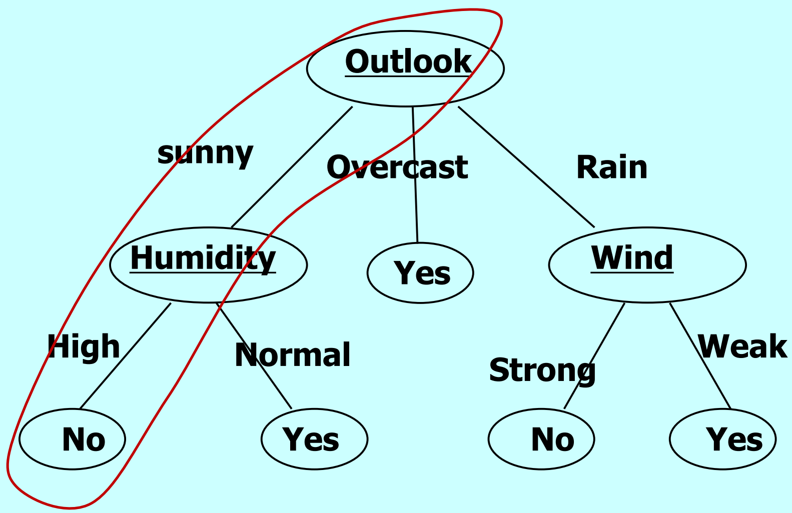
Recap: ID3 vs. Naïve-Bayes Methods for Classification

Day	Outlook	Tempe.	Humid.	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	weak	yes
14	rain	mild	high	strong	no

Given a training dataset and a new input instance for classification:

Outlook	Tempe.	Humid.	Wind	Play-tennis
sunny	cool	high	strong	?

ID3 (Model based)



Naïve Bayes (Instance based)

$$c_{NB} = \underset{j \in \{yes,\, no\}}{\operatorname{argmax}} P(c_j) \prod_{i=1}^{4} P(a_i \mid c_j)$$

P(yes|**e**) = 9/14 * 2/9 * 3/9 * 3/9 * 3/9 = 0.0053 = 20.5%
P(no|**e**) = 5/14 * 3/5 * 1/5 * 4/5 * 3/5 = 0.0206 = 79.5%

Use Naive Bayes classifier for the new input instance:

(sunny, cool, high, strong, ?)

Day	Outlook	Tempe.	Humid.	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	weak	yes
14	rain	mild	high	strong	no

Apply Naive Bayes classifier:

$$\boxed{c_{NB} = \underset{j \in \{yes,no\}}{\text{argmax}}\, P(c_j) \prod_{i=1}^{4} P(a_i \mid c_j)}$$

c_{yes} = P(Yes) * P(Outlook=sunny|yes) * P(Temp=cool|yes) *
P(Hum=high|yes) * P(Wind=strong|yes)

c_{no} = P(No) * P(Outlook=sunny|no) * P(Temp=cool|no) *
P(Hum=high|no) * P(Wind=strong|no)

How to calculate prior probabilities?

E.g., P(Outlook=sunny| Yes) = 2/9

P(Outlook=sunny| No) = 3/5

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	weak	yes
14	rain	mild	high	strong	no

The prior knowledge in the training data about the target "Play-tennis":

	Outlook			Temperature			Humidity			Wind			Play-tennis	
		yes	no		yes	no		yes	no		yes	no	yes	no
Counts	sunny:	2	3	hot:	2	2	high:	3	4	weak:	6	2	9	5
	overca:	4	0	mild:	4	2	normal:	6	1	strong:	3	3		
	rainy:	3	2	cool:	3	1								
Probabilities	sunny:	2/9	3/5	hot:	2/9	2/5	high:	3/9	4/5	weak:	6/9	2/5	9/14	5/14
	overca:	4/9	0/5	mild:	4/9	2/5	normal:	6/9	1/5	strong:	3/9	3/5		
	rainy:	3/9	2/5	cool:	3/9	1/5								

E.g., Prior probabilities for the new instance:

(sunny, cool, high, strong, ?)

$P(c_{yes}) = 9/14 = .64$, $P(c_{no}) = 5/14 = .36$

$P(sunny \mid c_{yes}) = 2/9 = .22$, $P(sunny \mid c_{no}) = 3/5 = .60$

$P(cool \mid c_{yes}) = 3/9 = .33$, $P(cool \mid c_{no}) = 1/5 = .20$

$P(high \mid c_{yes}) = 3/9 = .33$, $P(high \mid c_{no} = 4/5 = .80$

$P(strong \mid c_{yes}) = 3/9 =.33$, $P(strong \mid c_{no}) = 3/5 =.60$

The prior probabilities from the training set:

Outlook			**Temperature**			**Humidity**			**Wind**			**PlayTennis**	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny:	2/9	3/5	hot:	2/9	2/5	high:	3/9	4/5	week:	6/9	2/5	9/14	5/14
overca:	4/9	0/5	mild:	4/9	2/5	normal:	6/9	1/5	strong:	3/9	3/5		
rainy:	3/9	2/5	cool:	3/9	1/5								

Prior probabilities for the new data:

(sunny, cool, high, strong, ?)

P(c_yes) = 9/14 = .64,	P(c_no) = 5/14 = .36
P(sunny | c_yes) = 2/9 = .22,	P(sunny | c_no) = 3/5 = .60
P(cool | c_yes) = 3/9 =.33,	P(cool | c_no) = 1/5 = .20
P(high | c_yes) = 3/9 = .33,	P(high | c_no) = 4/5 = .80
P(strong | c_yes) = 3/9 =.33,	P(strong | c_no) = 3/5 =.60

Calculate Posterior Probabilities:

c_yes = P(c_yes) * P(sunny|yes) * P(cool|yes) * P(high|yes) * P(strong|yes)
= .64 * .22 * .33 * .33 * .33 = **.0053**

c_no = P(c_no) * P(sunny|no) * P(cool|no) * P(high|no) * P(strong|no)
= .36 * .6 * .2 * .8 * .6 = **.0206**

c_yes = .0053
c_no = .0206

Thus, the Naive Bayes classifier assigns the target value PlayTennis = no.

Classification for the input data:

(Outlook=sunny, Temp=cool, Hum=high, Wind=strong, Pay-tennis=?)

- The new input day:

Outlook	Temperature	Humidity	Wind	Play-tennis
sunny	cool	high	strong	?

- **The inference:**

  likelihood of yes = 9/14 * 2/9 * 3/9 * 3/9 * 3/9 = 0.0053

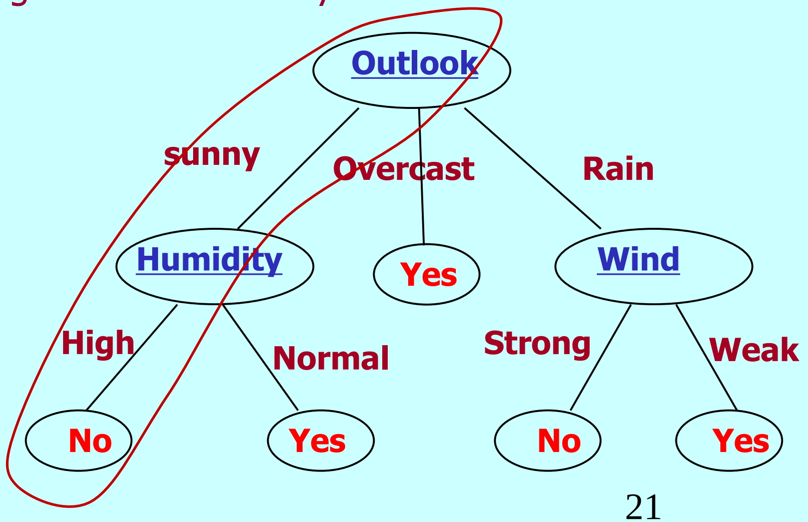
  likelihood of no = 5/14 * 3/5 * 1/5 * 4/5 * 3/5 = 0.0206

The numbers can be turned into probabilities by normalizing them so that they sum to 1:

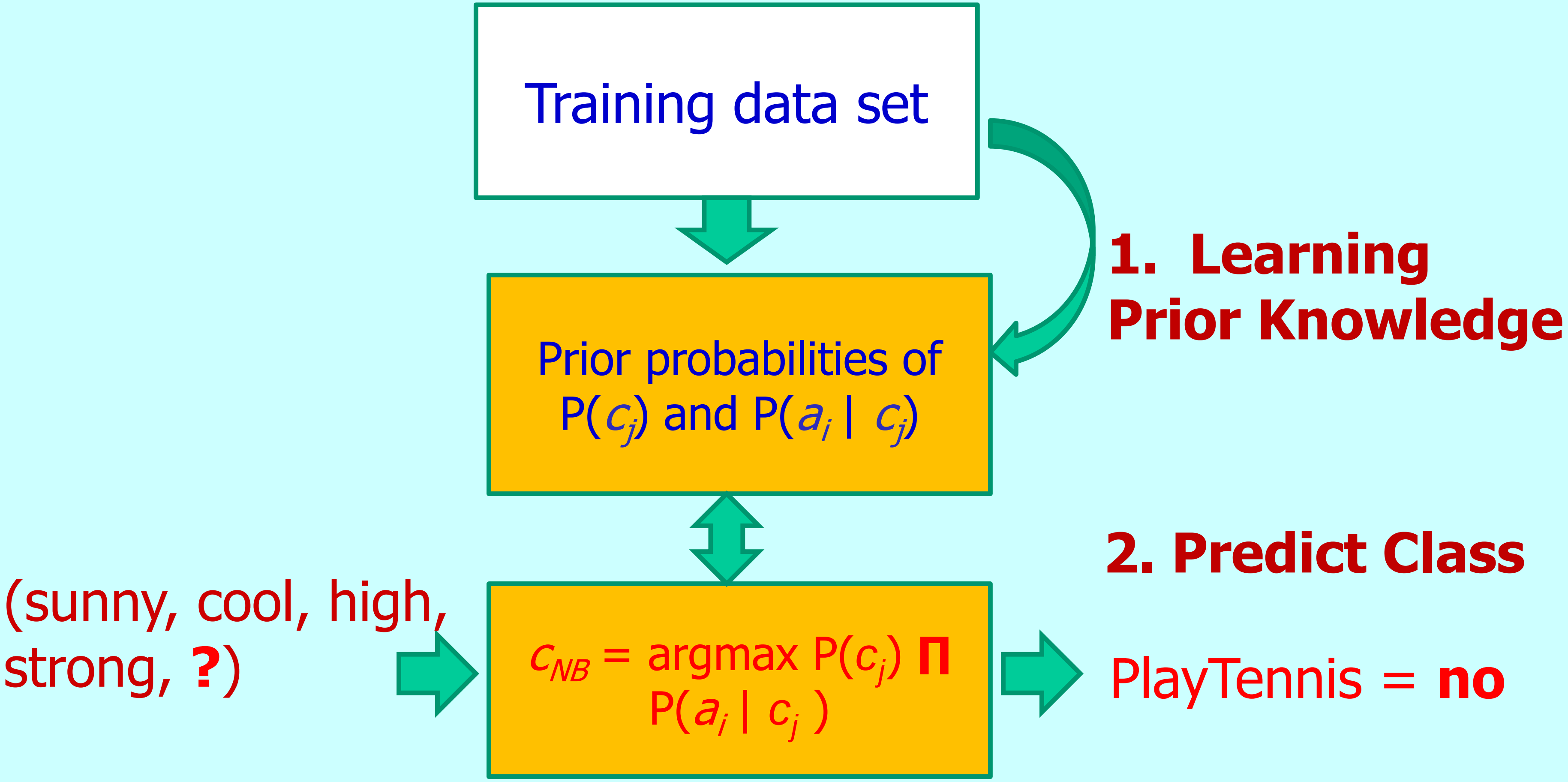
$$\text{Probability of Yes} = \frac{0.0053}{0.0053 + 0.0206} = 20.5\%$$

$$\text{Probability of No} = \frac{0.0206}{0.0053 + 0.0206} = 79.5\%$$

Predication: (sunny, cool, high, strong, No)



System Architecture



Any problem with the method?

Outlook	Temperature	Humidity	Windy	Playtennis
overcast	hot	high	false	no
overcast	cool	normal	true	no
overcast	cool	normal	true	no
rain	cool	normal	true	yes
overcast	cool	normal	true	no
overcast	cool	normal	true	no
overcast	cool	normal	true	no

Observation: None of the tuples have outlook = sunny, this leads to P(sunny|yes) = 0 and P(sunny|no) = 0. E.g., Many of the tuples in data1 have outlook = sunny, but C_yes = C_no = 0. Any problem?

Use Naive Bayes classifier for the new case:
(sunny, cool, high, strong, ?)

Apply Naive Bayes classifier:

$$c_{NB} = \underset{j \in \{yes,no\}}{\operatorname{argmax}} P(c_j) \prod_{i=1}^{4} P(a_i \mid c_j)$$

c_{yes} = P(c_{yes}) * P(sunny|yes) * P(cool|yes) * P(high|yes) * P(strong|yes)

c_{no} = P(c_{no}) * P(sunny|no) * P(cool|no) * P(high|no) * P(strong|no)

0 ?

M-estimate of Probability (for handling missing values in training data)

Problem: P(outlook=sunny | yes) = 0 or P(outlook=sunny | no) = 0 ?

When this probability estimate is zero, this probability term will dominate the Bayes classifier by this zero value.

Solution: Instead of using n_c/n for the estimate, we use

$$\textbf{M-estimate} = \frac{n_c + mp}{n + m}$$

n: the number of examples with a particular class, e.g. "yes", of the training set, n = 9.

n_c: the total number of examples with outlook=sunny in yes class, e.g. 0 in this case.

m: a constant determining how to weight p.

p: <u>a prior estimation for an attribute value, 1/k,</u> there k is the number of possible values, e.g. p = .33 for the attribute "outlook", m = 1, M-estimate = .033.

Observations on Naïve Bayesian Classifier

- **Strengths:**
  - Easy to implement
  - Robust to isolated noise points
  - Handle missing values by ignoring the instance during probability estimate calculations
  - Robust to irrelevant attributes, and <u>high dimensional data</u>
  - Good results obtained in some application domains, e.g. text classification
- **Limitations**
  - Assumption: <u>attributes independent</u>, therefore may loss accuracy for some applications.
    - Practically, dependencies exist among variables
      - E.g., population data: age, education, wage, etc.
      - E.g., medical diagnoses data (symptoms): fever, cough, etc.
  - <u>Dependencies among factors cannot be modeled by Naïve Bayesian Classifier</u>
- **How to deal with these dependencies?**
  - Bayesian Belief Networks (Optional reading)

Text Classification

- **What is text classification?**
  - Consider a classification problem in which the instances are text documents:
    - Training data
    - Target classes
    - Prediction: label an unseen text to one of the classes
- **Application examples:**
  - Spam filtering
  - Text document categorization (place e-articles into categories), such as news articles, tweets, etc.
  - Classify survey comments into positive or negative class

Dataset: Structured vs. Unstructured

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	weak	yes
14	rain	mild	high	strong	no

Structured:

- Objects
- Attributes
- Target values

Solution Idea:
Unstructured Text Data -> Structured Data

"*Our approach to representing arbitrary text documents is disturbingly simple: given a text document, such as this paragraph, we <u>define an attribute for each word position</u> in the document and <u>define the value of that attribute to be the English word found in that position</u>. Thus, the current paragraph would be described by 111 attribute values, corresponding to the 111 word positions. The value of the first attribute is the word "Our," the value of the second attribute is the word "approach," and so on. Notice that long text documents will require a larger number of attributes than short documents. As we shall see, this will not cause us any trouble.*"

How to make text be structured for classification

- **Text documents are unstructured data**
  - In comparing with structured data, such as formatted data in a table
    - With fixed number of attributes, and well defined domain for each attribute, etc.
    - What about text documents? - Structure, object unit, etc.
- **How to automatically process large quantities of unstructured text data?**
  - People are good at processing small quantities of unstructured text (We have intelligence)
  - Computers are good at processing large quantities of structured text

Main ideas for text classification

- **Make text data be structured**
  - We need to find ways to structure text so that it can be understood by computer:
    - Treat each text document as an object
    - Treat word position as attribute
    - Treat words as domain values
- **Apply a classification method which can handle high dimension data, such as NB method**
  - How to represent text in terms of attributes?
  - How to calculate prior probabilities for using NB?

Convert Text into Structured Data

- **Idea: Treat each word position as an attribute, and each word as a value:**
  - "*Our approach to representing arbitrary text documents is disturbingly simple: given a text document, such as this paragraph, we define an attribute for each word position in the document and define the value of that attribute to be the English word found in that position. Thus, the current paragraph would be described by 111 attribute values, corresponding to the 111 word positions. The value of the first attribute is the word "Our," the value of the second attribute is the word "approach," and so on. Notice that long text documents will require a larger number of attributes than short documents. As we shall see, this will not cause us any trouble.*"
- **E.g.,** a1="our", a2= "approach", ..., a111="trouble"

Attribute 1 has the value "our" positioned at the word location 1.

Apply NB Method for Text Classification

- Apply NB for classify the example text:

$$c_{NB} = \underset{c_j \in \{like, dislike\}}{\operatorname{argmax}} P(c_j) * \prod_{i=1}^{111} P(a_i \mid c_j)$$

$$= \underset{c_j \in \{like, dislike\}}{\operatorname{argmax}} P(c_j) * P(a_1 = \text{"Our"} \mid c_j) * P(a_2 = \text{"approach"} \mid c_j) \ldots\ldots * P(a_{111} = \text{"trouble"} \mid c_j)$$

- How to calculate the prior probabilities?
  - Training data set
  - Calculation method

Example

- **Training set:** 1000 documents = 700 (dislike) + 300 (like)
- **Classifier:**

$$c_{NB} = \underset{c_j \in \{like, dislike\}}{\operatorname{argmax}} P(c_j) \prod_{i=1} P(a_i \mid c_j)$$

Where, $P(c_j)$ and $P(a_i \mid c_j)$ are prior probabilities from the training data.

- **Estimating** $P(c_j)$**:**
  - P(like) = 300/1000 = 0.3
  - P(dislike) = 700/1000 = 0.7

How many prior probabilities?

- **How many prior probabilities of** $P(a_i = w_k \mid c_j)$?

Where w_k is the kth word in the English vocabulary.

E.g., $P(a_1 =$"our" $\mid$ dislike).

- **Estimation on how many to calculate:**
  - $P(a_i = w_k \mid c_j)$: **a function of three variables,** a_i , w_k , c_j
  - We must estimate one such probability term for each combination of **word positions** of the text (I), **English words** (K), and **target values** (C):

$$\boxed{f(I, K, C) = I * K * C}$$

Estimation

- E.g., *"Our approach to representing arbitrary text documents is disturbingly simple: given a text document, such as this paragraph, we define an attribute for each word position in the document and define the value of that attribute to be the English word found in that position. Thus, the current paragraph would be described by 111 attribute values, corresponding to the 111 word positions. The value of the first attribute is the word "Our," the value of the second attribute is the word "approach," and so on. Notice that long text documents will require a larger number of attributes than short documents. As we shall see, this will not cause us any trouble."*
- **Estimation: *C * I * K***

  C = 2, Target values in the example.

  I = 111, Text positions in the example.

  K = 50,000, Distinct words in the English vocabulary.

  Total # of probabilities: 2 * 111 * 50,000 ≈ 10 million !!

- Is there any room for reducing the number of probability terms?

Make Effective Estimation

- Fortunately, we can make a reasonable assumption that reduces the number of probabilities that must be estimated

Assumption:

  - The probability of encountering a specific word w_k, (e.g., “our") is independent of the specific word position being considered (e.g., a_1 versus a_{85}), where k in the kth position in the vocabulary

  - This can be expressed by

    $P(a_i = w_k \mid c_j) = P(a_m = w_k \mid c_j)$ for all i, j, k, m

    E.g., a_1 = “our" | dislike) = P(a_{85} =“our“ | dislike)

- In fact this assumption is consistent with the previous assumption “attributes are independent”, and identically distributed given the target classification:

  - $C * I * K$ -> $C * K$

- **Assumption: P(a_i = w_k | c_j) = P(a_m= w_k | c_j)**
  - In fact this assumption is consistent with the previous assumption "attributes are independent", and identically distributed given the target classification:

    C * I * K => C * K

  - Accordingly, we estimate the entire set of probabilities: $P(a_1 = w_k \mid c_j)$, $P(a_2 = w_k \mid c_j)$ ... by the single position-independent probability $\boldsymbol{P(w_k \mid c_j)}$ which we will use regardless of the word position
    - The net effect is we now require only 2 * 50000 distinct terms of the form $P(w_k \mid c_j)$ comparing with 2* 111 * 50000

Prior Probability Estimation Formula

A commonly used estimator:

$$\boxed{P(w_k \mid c_j) = \frac{n_k + 1}{n + Vocabulary}}$$

- n, the <u>total word positions</u> in all training examples whose target values is c_j
- n_k, the <u>frequency</u> of w_k among these n word positions
- *Vocabulary*, the total number of <u>distinct words</u> found within the training data

NB Algorithm for calculating prior probabilities

Learn_NB_text (Examples, C)

/* Examples is a set of text documents along with their target values. C is the set of all possible target values. The function learns the probability terms $P(w_k \mid c_j)$, describing the probability that a randomly drawn word from document in class c_j will be the English word w_k. It also learns the class prior probabilities $P(c_j)$.

*/

(Next page)

1. Collect the set of all distinct and useful words occurring in any document of the training set: *Vocabulary*

2. Calculate $P(c_j)$ and $P(w_k | c_j)$ probability terms

- For each c_j in C do

- $docs_j \leftarrow$ the subset of documents from training set for which the target value is c_j
- $$P(c_j) = \frac{docs_j}{Examples}$$
- $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$
- $n \leftarrow$ total number of word positions in $Text_j$
- for each word w_k in *Vocabulary*
  - $n_k \leftarrow$ number of times word w_k occurs in $Text_j$
  - $$P(w_k | c_j) = \frac{n_k + 1}{n + Vocabulary}$$

NB Text Classifier

- **Classify_NB_text (doc)**

// Return the estimated target value for the input document doc, a_i denotes the word found in the ith position within doc.

- positions ← all word positions in doc that contain tokens found in Vocabulary
- Return c_NB, where $c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(a_i \mid c_j)$

Case Study: E-news Categorization

- Joachims T. (1996), "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization", (Computer Science Technical Report CMU-CS-96-118), Carnegie Mellon University (http://www.cs.cornell.edu/People/tj/publications/joachims_97a.pdf)
- **Application:** classify usenet news articles: 20 electronic newsgroups were considered.
- 1,000 articles were collected from each newsgroup, forming a data set of 20,000 documents.
- **Training set and test set:** two-thirds of these 20,000 documents as training examples, and performance was measured over the remaining third.

  For each newsgroup, the training uses 667 articles. The target contains 20 labels.

How Effective Is the Algorithm ?

- **Vocabulary:** Only a subset of the words occurring in the documents were included

  as the value of the Vocabulary variable in the algorithm:

  The 100 most frequently words were removed (these include words such as "the" and "of", etc), and any word fewer than three times was also removed. The resulting vocabulary contained approximately **38,500** words.

- **Result:**

  Given 20 possible newsgroups, we would expect random guessing to achieve a classification accuracy of approximately 5%. The accuracy of the program: **89%**.

Limitation of NB Method

- **Issue with NB's assumption**
  - The independence assumption states that the word probability contributes to the text classification has not association with its location in the text (i.e. words are independent each other)
  - This assumption is clearly inaccurate. In practice, however, the naive Bayes algorithm performs remarkably well when classify text by topic despite the incorrectness of this independence assumption.

- **Problem:** for non-topic orientated text classification
  - When classify documents not by topic, but by overall sentiment, e.g. determining whether a review is positive or negative, the Naïve algorithm does not perform well.
  - MCS/MACS research: Knowledge based sentiment text classification

    (Doc/Theses/MCSthesisXu03.pdf, MACSprojectYonas04.pdf)

Review Questions

1. How different a classification task is done by DT induction and by Naïve Bayes classifier? (*Give 3 differences.)
2. What are the two assumptions for using NB classifier?
3. Why Naïve Bayes algorithm is more suitable to high dimensional data?
4. What is text classification? What is the basic idea to convert unstructured text data into structured for classification?
5. How to estimate the number of prior probabilities which need to be calculated for text classification?
6. Why Naïve Bayes algorithm is more suitable for text data mining? What is its limitation?