Natural Language Processing

Machine Learning - Basic Algorithms

Dr. Sampath Deegalla
PhD, PhLic (Stockholm University, Sweden),
BSc Eng (University of Peradeniya), AMIESL, MIEEE, MIEEE-CS
sampathd@sltc.ac.lk

Input

Outline

- Basic types of Learning in Data Mining
- Types of input to the learning algorithm
  - Concept, Example, Attribute
- Preparing the input
  - Missing values, Inaccurate values, getting to know data

Basic types of Learning

- Classification (Supervised Learning):
  - predicting a particular class value (discrete)
  - E.g. Weather problem
- Association learning:
  - detecting associations between features
- Clustering (Unsupervised Learning):
  - grouping similar instances into clusters
- Numeric prediction:
  - predicting a numeric quantity
  - E.g. CPU performance

Classification learning

- Example
  - Weather data to predict play/not play
- Classification learning is supervised
  - Scheme is being provided with actual outcome
  - Outcome is called the class of the example
- success can be measured on fresh data for which class labels are known (test data)

Association learning

- Example
  - supermarket basket analysis - what items are bought together (e.g. milk + cereal)
- Can be applied if no class is specified and any kind of structure is considered "interesting"
- Difference with classification learning:
  - Can predict any attribute's value, not just the class, and more than one attribute's value at a time
  - Hence: far more association rules than classification rules
  - Thus: constraints are necessary
    - Minimum coverage and minimum accuracy

Clustering

- Example
  - customer grouping
- Finding groups of items that are similar
- Clustering is unsupervised
  - The class of an example is not known
- Success often measured subjectively

Numeric prediction

- Classification learning, but "class" is numeric
- Learning is supervised
  - Scheme is being provided with target value
- Measure success on test data

Outlook	Temperature	Humidity	Windy	Windy
Sunny	Hot	High	False	5
Sunny	Hot	High	True	0
Overcast	Hot	High	False	55
Rainy	Mild	Normal	False	40
...	...	...	...	...

What's a concept?

- Concept: thing to be learned
- Concept description: output of learning scheme
  - In the form of rule set, decision tree, ...
- Success rate on test data $\rightarrow$ How well the concept has been learned

What's in an example?

- Instance: specific type of example
  - Thing to be classified, associated, or clustered
  - Individual, independent example of target concept
  - Characterized by a predetermined set of attributes
- Input to learning scheme: set of instances/dataset
  - Represented as a single relation/flat file

Generating a flat file

- Process of flattening a file is called denormalization
  - Several relations are joined together to make one
- Possible with any finite set of finite relations
- Denormalization may produce fake regularities that reflect structure of database
  - Example: “supplier” predicts “supplier address”

What's in an attribute?

- Each instance is described by a fixed predefined set of features, its “attributes”
- But: number of attributes may vary in practice
  - Possible solution: “irrelevant value” flag
- Related problem: existence of an attribute may depend of value of another one
  - E.g. spouse name $\rightarrow$ married/single
- Possible attribute types :
  - Nominal, ordinal, interval and ratio

Nominal quantities

- Values are distinct symbols
  - Values themselves serve only as labels or names
  - Nominal comes from the Latin word for name
- Example: attribute “outlook” from weather data
  - Values: “sunny”,“overcast”, and “rainy”
- No relation is implied among nominal values
  - no ordering or distance measure
- Only equality tests can be performed

Ordinal quantities

- Impose order on values
- But: no distance between values defined
- Example: attribute “temperature” in weather data
  - Values: “hot” $>$ “mild” $>$ “cool”
- Note: addition and subtraction don’t make sense
- Example rule:
  - temperature $<$ hot $\rightarrow$ play $=$ yes
- Distinction between nominal and ordinal not always clear (e.g. attribute “outlook”)
- Note: Distinction between nominal and ordinal is not always straightforward

Interval quantities (Numeric)

- Interval quantities are not only ordered but measured in fixed and equal units
- Example 1: attribute “temperature” expressed in degrees Fahrenheit
- Example 2: attribute “year”
- Difference of two values makes sense
  - E.g. 46F , 48F and 1939, 1945
- Sum or product doesn’t make sense
  - Zero point is not defined! (year 0)

Ratio quantities

- Ratio quantities are ones for which the measurement scheme defines a zero point
- Example: attribute “distance”
  - Distance between an object and itself is zero
- Ratio quantities are treated as real numbers
  - All mathematical operations are allowed
- But: is there an “inherently” defined zero point?
  - Answer depends on scientific knowledge (e.g. Fahrenheit knew no lower limit to temperature)

Attribute types used in practice

- Most schemes accommodate just two levels of measurement: nominal and ordinal
- Nominal attributes are also called “categorical”, ”enumerated”, or “discrete”
- Special case: dichotomy (“boolean” attribute)
- Ordinal attributes are called “numeric”, or “continuous”
  - But: “continuous” implies mathematical continuity

Preparing the input

- Problem: different data sources (e.g. sales department, customer billing department, . . .)
  - Differences: styles of record keeping, conventions, time periods, data aggregation, primary keys, errors
  - Data must be assembled, integrated, cleaned up
  - “Data warehouse”: consistent point of access
- Denormalization relational data are not the only issue
- External data may be required (“overlay data”)
- Critical: type and level of data aggregation

The Weather Data

Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
...				

- Instances, Examples, Objects – Represent the rows of the table.
- Attributes, Features – Represent the columns ’Outlook’, ’Temperature’, ’Humidity’, and ’Windy’.
- Class Attribute – Represented by the column ’Play’.

Missing Values

- Frequently indicated by out-of-range entries
  - Types: unknown, unrecorded, irrelevant
  - Reasons:
    - malfunctioning equipment
    - changes in experimental design
    - collation of different datasets
    - measurement not possible
- Missing value may have significance in itself (e.g., missing test in a medical examination)
  - Most schemes assume that is not the case
    $\Rightarrow$ "missing" may need to be coded as an additional value

Inaccurate Values

- Reason: data has not been collected for mining it
- Result: errors and omissions that don't affect original purpose of data (e.g., age of customer)
- Typographical errors in nominal attributes ⇒ values need to be checked for consistency
- Typographical and measurement errors in numeric attributes ⇒ outliers need to be identified
- Errors may be deliberate (e.g., wrong zip/postal codes)
- Other problems: duplicates, scale data

Getting to Know the Data

- Simple visualization tools are very useful
  - Nominal attributes: histograms (Distribution consistent with background knowledge?)
  - Numeric attributes: graphs (Any obvious outliers?)
- 2-D and 3-D plots show dependencies
- Need to consult domain experts
- Too much data to inspect? Take a sample!

Basic Algorithms

Outline

- Basic Algorithms
  - IR
  - Decision Trees
  - Decision Rules (PRISM)
  - Nearest Neighbor
  - Naïve Bayes

OneR

Simplicity first

- Simple algorithms often work very well!
- There are many kinds of simple structure, eg:
  - One attribute does all the work
  - All attributes contribute equally & independently
  - A weighted linear combination might do
  - Instance-based: use a few prototypes
  - Use simple logical rules
- Success of method depends on the domain

Inferring rudimentary rules

- 1R: learns a 1-level decision tree
  - i.e., rules that all test one particular attribute
- Basic version
  - One branch for each value
  - Each branch assigns most frequent class
  - Error rate: proportion of instances that don't belong to the majority class of their corresponding branch
  - Choose attribute with lowest error rate (assumes nominal attributes)

ZeroR and OneR Algorithms

- **ZeroR**: Outputs the majority class.
- **OneR**: Learns a 1-level decision tree.
  - Rules that test one particular attribute.
  - Basic version:
    - One branch for each attribute value.
    - Each branch assigns the most frequent class.
  - Error rate is the proportion of instances that don't belong to the majority class of their corresponding branch.
  - Choose the attribute with the lowest error rate.
- The basic version of OneR assumes nominal attributes.

The Weather Data

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Output of ZeroR

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
		...		

Play $\rightarrow$ Yes

Pseudo-code for 1R

```
For each attribute,
    For each value of the attribute, make a rule:
        Count how often each class appears
        Find the most frequent class
        Make the rule assign that class to
        .. this attribute-value
    Calculate the error rate of the rules
Choose the rules with the smallest error rate
```

- Note: “missing” is treated as a separate attribute value

Evaluating the weather attributes

Attribute	Rules	Errors	Total errors
Outlook	Sunny $\rightarrow$ No	2/5	4/14
	Overcast $\rightarrow$ Yes	0/4	
	Rainy $\rightarrow$ Yes	2/5	
Temp			
Humidity			
Windy			

Evaluating the weather attributes (cont.)

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No Overcast → Yes Rainy → Yes	2/5 0/4 2/5	4/14
Temp	Hot → No* Mild → Yes Cool → Yes	2/4 2/6 1/4	5/14
Humidity	High → No Normal → Yes	3/7 1/7	4/14
Windy	False → Yes True → No*	2/8 3/6	5/14

[1]* indicates a tie

Output of OneR

Outlook	Sunny → No Overcast → Yes Rainy → Yes

Dealing with numeric attributes

- Discretizing Numeric Attributes
- Divide each attribute's range into intervals.
  - Sort instances according to the attribute's values.
  - Place breakpoints where the class changes (the majority class).
  - This minimizes the total error.

Weather Data with Some Numeric Attributes

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

- Example: **temperature from weather data**

64 Yes | 65 No | 68 Yes 69 Yes 70 Yes | 71 No 72 No 72 Yes | 75 Yes 75 Yes | 80 No | 81 Yes 83 Yes | 85 No

The problem of overfitting

- This procedure is very sensitive to noise.
  - One instance with an incorrect class label will probably produce a separate interval.
- Simple solution: enforce minimum number of instances in majority class per interval.

Discretization Example

64 Yes || 65 No || 68 Yes 69 Yes 70 Yes || 71 No 72 No 72 Yes || 75 Yes 75 Yes || 80 No || 81 Yes 83 Yes || 85 No

Example (with min = 3):

64 Yes | 65 No | 68 Yes 69 Yes 70 Yes || 71 No 72 No 72 Yes | 75 Yes 75 Yes || 80 No | 81 Yes 83 Yes | 85 No

Final result for temperature attribute

64 Yes 65 No 68 Yes 69 Yes 70 Yes | 71 No 72 No 72 Yes 75 Yes 75 Yes || 80 No 81 Yes 83 Yes 85 No

Evaluating the weather attributes (cont.)

Attribute	Rules	Errors	Total errors
Outlook	Sunny $\rightarrow$ No	2/5	4/14
	Overcast $\rightarrow$ Yes	0/4	
	Rainy $\rightarrow$ Yes	2/5	
Temp	$\leq 77.5 \rightarrow$ Yes	3/10	5/14
	$> 77.5 \rightarrow$ No*	2/4	
Humidity	$\leq 82.5 \rightarrow$ Yes	1/7	3/14
	≥ 82.5 and $\leq 95.5 \rightarrow$ No	2/6	
	$\leq 95.5 \rightarrow$ Yes	0/1	
Windy	False $\rightarrow$ Yes	2/8	5/14
	True $\rightarrow$ No*	3/6	

[1]* indicates a tie

Output of OneR

Humidity $\leq 82.5 \rightarrow$ Yes
> 82.5 and $\leq 95.5 \rightarrow$ No
$> 95.5 \rightarrow$ Yes

Decision Trees

Decision Trees

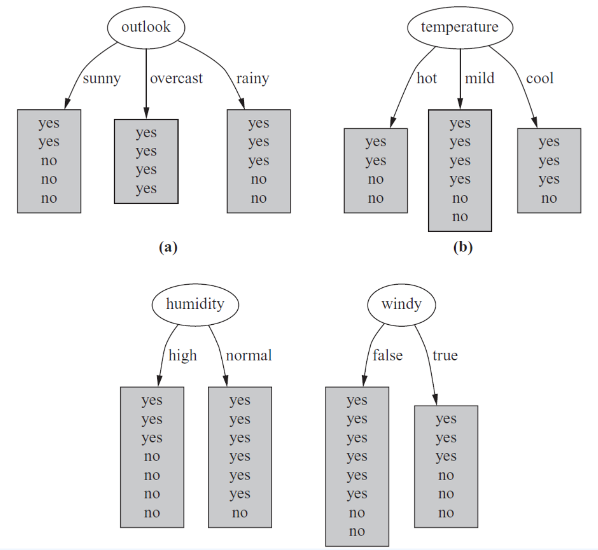
What is a Decision Tree?

Learns a tree-like structure, where each node represents an attribute (or a test) and each branch represents a possible value (or outcome of the test). The leaves of the tree represent the final class.

How to Build a Tree Automatically from Data

- Divide and conquer approach (Top-Down Induction of Decision Trees).
- Select an attribute to place as the root node.
- Make one branch for each possible value.
- Repeat the process for each branch.
- Stop if all instances at a node have the same classification.

Which attribute to select?



Criterion for Attribute Selection

- **Objective:** Find the best attribute to create the smallest decision tree.
- **Heuristic:** Choose the attribute that produces the "purest" nodes.
- **Popular Selection Criterion:** Information Gain.
  - Information gain increases with the average purity of the subsets.
- **Strategy:** Amongst attributes available for splitting, choose the attribute that gives the greatest information gain.
- **Information Gain Requirement:** A measure of impurity.
- **Impurity Measure:** Entropy of the class distribution (from information theory).

Computing Information

- We have a probability distribution: the class distribution in a subset of instances.
- The expected information required to determine an outcome (i.e., class value) is the distribution's entropy.
- Formula for computing the entropy:

$$\text{Entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \cdots - p_n \log p_n$$

- Using base-2 logarithms, entropy gives the information required in expected bits.
- Entropy is maximal when all classes are equally likely and minimal when one of the classes has probability 1.

Information Gain

- Expected Information

$$\text{Info}(D) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

- Expected information after partitioned by attribute A

$$\text{Info}_A(D) = \sum_{i=1}^{v} \frac{|D_i|}{|D|} \times \text{Info}(D_i)$$

- Information Gain

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

Example: Attribute Outlook

- Outlook = Sunny:

$$\begin{aligned}\text{Info}([2,3]) = \text{entropy}\left(\frac{2}{5}, \frac{3}{5}\right) &= -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) \\ &= 0.971 \text{ bits}\end{aligned}$$

- Outlook = Overcast: Info([4,0]) = 0.0 bits

$$\begin{aligned}\text{Info}([4,0]) &= \text{entropy}(1,0) = -1\log_2(1) - 0\log_2(0)^{1} \\ &= 0.0 \text{ bits}\end{aligned}$$

- Outlook = Rainy: Info([3,2]) = 0.971 bits
- Expected information for attribute:

$$\begin{aligned}\text{info}([2,3],[4,0],[3,2]) &= \frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971 \\ &= 0.693 \text{ bits}\end{aligned}$$

[1]Note: log(0) is not defined, but we evaluate 0*log(0) as zero

Computing Information Gain

- Information gain = information before splitting - information after splitting.
- Example:

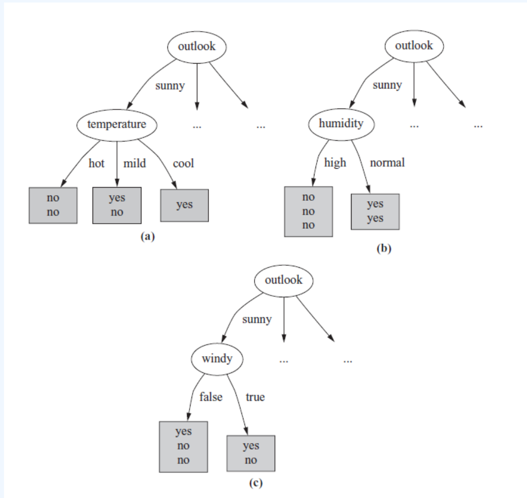
$$\begin{aligned} \text{Gain}(\text{Outlook}) &= \text{Info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) \\ &= 0.940 - 0.693 \\ &= 0.247 \text{ bits} \end{aligned}$$

- Information gain for attributes from weather data:

$$\begin{aligned} \text{Gain}(\text{Outlook}) &= 0.247 \text{ bits} \\ \text{Gain}(\text{Temperature}) &= 0.029 \text{ bits} \\ \text{Gain}(\text{Humidity}) &= 0.152 \text{ bits} \\ \text{Gain}(\text{Windy}) &= 0.048 \text{ bits} \end{aligned}$$

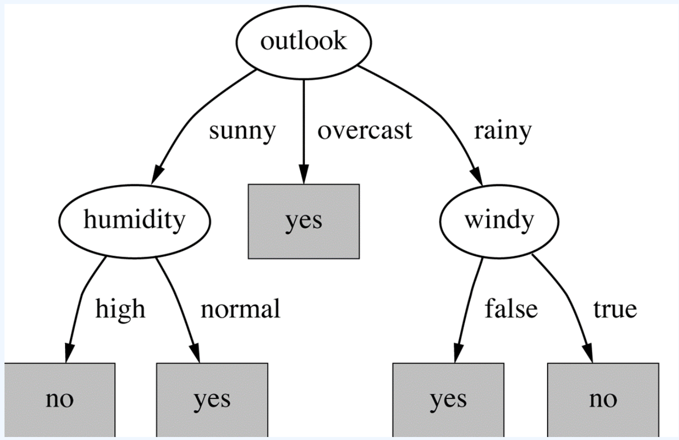
- Select the attribute with the highest gain ratio
- Problems: highly branching attributes in Information Gain, Example, ID code

Continuing to Split



- Gain(Temperature) = 0.571 bits
- Gain(Humidity) = 0.971 bits
- Gain(Windy) = 0.020 bits

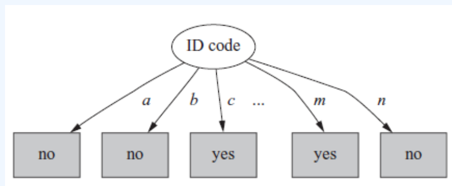
Final decision tree



- Note: not all leaves need to be pure; sometimes identical instances have different classes
  - Splitting stops when data cannot be split any further

Highly-branching attributes

- Problematic: attributes with a large number of values (extreme case: ID code)
- Subsets are more likely to be pure if there is a large number of values
  - Information gain is biased towards choosing attributes with a large number of values
  - This may result in overfitting (selection of an attribute that is non-optimal for prediction)



- In the above, all (single-instance) subsets have entropy zero!
- This means the information gain is maximal for this ID code attribute

Gain Ratio

- Gain ratio is a modification of the information gain that reduces its bias towards attributes with many values.
- Gain ratio takes number and size of branches into account when choosing an attribute.
- It corrects the information gain by taking the intrinsic information of a split into account.
- Intrinsic information: entropy of the distribution of instances into branches.
- Measures how much info do we need to tell which branch a randomly chosen instance belongs to.

Computing the Gain Ratio

- Example: intrinsic information of ID code

$$\frac{1}{14}(\text{info}([0,1])+\text{info}([0,1])+\text{info}([1,0])+\cdots+\text{info}([1,0])+\text{info}([0,1]))$$

- Value of attribute should decrease as intrinsic information gets larger.
- The gain ratio is defined as the information gain of the attribute divided by its intrinsic information.
- Example (outlook at root node):
  - Gain: 0.940-0.693 = 0.247
  - Split info: info([5,4,5]) = 1.577
  - Gain ratio: 0.247/1.577 = 0.156

All Gain Ratios for the Weather Data

Outlook	Temperature
Info: 0.693 Gain: 0.940-0.693 = 0.247 Split info: info([5,4,5]) = 1.577 Gain ratio: 0.247/1.577 = 0.157	Info: 0.911 Gain: 0.940-0.911 = 0.029 Split info: info([4,6,4]) = 1.557 Gain ratio: 0.029/1.557 = 0.019
Humidity	Windy
Info: 0.788 Gain: 0.940-0.788 = 0.152 Split info: info([7,7]) = 1.000 Gain ratio: 0.152/1 = 0.152	Info: 0.892 Gain: 0.940-0.892 = 0.048 Split info: info([8,6]) = 0.985 Gain ratio: 0.048/0.985 = 0.049

More on the Gain Ratio

- “Outlook” still comes out top.
- However: “ID code” has greater gain ratio.
- Standard fix: ad hoc test to prevent splitting on that type of identifier attribute.
- Problem with gain ratio: it may overcompensate.
- May choose an attribute just because its intrinsic information is very low.
- Standard fix: only consider attributes with greater than average information gain.
- Both tricks are implemented in the well-known C4.5 decision tree learner.

Gini Index

- The **Gini** index is used in CART.

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2 \tag{1}$$

- It considers a binary split for each attribute. **Gini** after partitioned by attribute A:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{2}$$

- The reduction of impurity:

$$\Delta Gini(A) = Gini(D) - Gini_A(D) \tag{3}$$

Real Machine Learning Algorithms – Decision Trees C4.5

- 1. Handling Numeric Attributes
- 2. Handling Missing Values
- 3. Generalizing the Model

Decision Trees

- Extending ID3:
  - to permit numeric attributes: *straightforward*
  - to deal sensibly with missing values: *trickier*
  - stability for noisy data: *requires pruning mechanism*
    - **Problem:** ID3 performs well on the training set and does not generalize well on the independent test sets
- End result: C4.5 (Quinlan)
  - Best-known and (probably) most widely-used learning algorithm
  - Commercial successor: C5.0 (now available freely under GNU General Public License)

Numeric Attributes

- **Standard method:** binary splits
  - E.g. temp < 45
- Unlike nominal attributes, every attribute has many possible split points
- **Solution is straightforward extension:**
  - Evaluate information gain (or other measure) for every possible split point of attribute
  - Choose “best” split point
  - Information gain for best split point is info gain for attribute
- Computationally more demanding

Weather Data with Some Numeric Attributes

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

Example

- Split on temperature attribute:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- E.g. temperature < 71.5: yes/4, no/2
  temperature $\geq$ 71.5: yes/5, no/3
- Expected information for temperature attribute:

$$\text{info}([4,2],[5,3]) = \frac{6}{14} \times \text{Info}([4,2]) + \frac{8}{14} \times \text{Info}([5,3])$$
$$= 0.939 \text{ bits}$$

- Place split points halfway between values
- Can evaluate all split points in one pass!

Missing Values

- Treat missing values as another possible value of the attribute (if significant)
- Ignore all the instances with missing values (lose information)
- Split instances with missing values into pieces:
  - A piece going down a branch receives a weight proportional to the popularity of the branch: weights sum to 1
  - Info gain works with fractional instances: use sums of weights instead of counts
  - During classification, split the instance into pieces in the same way: merge probability distribution using weights

Pruning

- Prevent overfitting to noise in the data
- "Prune" the decision tree
- Two strategies:
  - **Postpruning (backward pruning)**: take a fully-grown decision tree and discard unreliable parts
  - **Prepruning (forward pruning)**: stop growing a branch when information becomes unreliable (decide on when to stop developing subtrees)
- Postpruning preferred in practice – prepruning can "stop early"

Prepruning

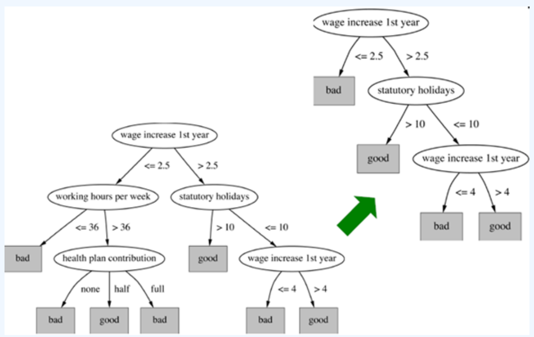
- Based on statistical significance test
  - Stop growing the tree when there is no *statistically significant association between any attribute and the class* at a particular node
- Pre-pruning may stop the growth process prematurely: *early stopping*
- Prepruning is faster than postpruning

Postpruning

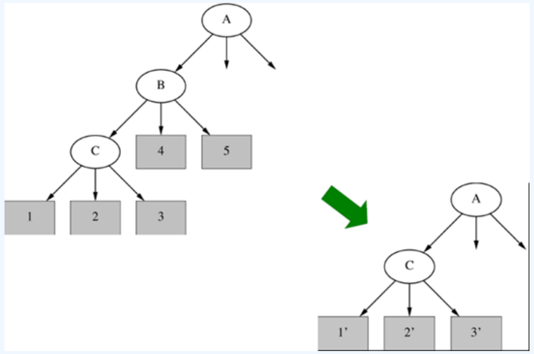
- First, build full tree and then, prune it
  - Fully-grown tree shows all attribute interactions
- Problem: some subtrees might be due to chance effects
- Two pruning operations:
  - *Subtree replacement*
  - *Subtree raising*
- Possible strategies:
  - error estimation
  - significance testing
  - MDL principle

Subtree replacement

- The idea is to select some subtrees and replace them with single leaves.
- Bottom-up: Consider replacing a tree only after considering all its subtrees



Subtree raising



- Delete node
- Redistribute instances
- Slower than subtree replacement

Discussion

- Top-down induction of decision trees: ID3, algorithm developed by Ross Quinlan.
- Gain ratio just one modification of this basic algorithm.
- C4.5 tree learner deals with numeric attributes, missing values, noisy data.
- Similar approach: CART tree learner.
- Uses Gini index rather than entropy to measure impurity.
- There are many other attribute selection criteria! (But little difference in accuracy of result).
  - Information Gain
  - Gain Ratio
  - Gini index (CART trees – used in scikit-learn)
  - Chi-Square

Decision Rules (PRISM)

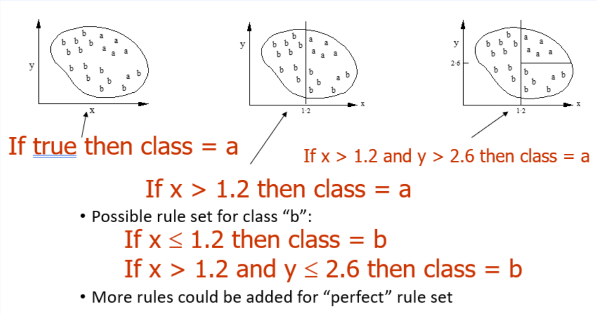
Generating Rules

- Decision tree can be converted into a rule set
- **Straightforward conversion:**
  - each path to the leaf becomes a rule – makes an overly complex rule set
- **More effective conversions are not trivial**
  - *(e.g., C4.8 tests each node in root-leaf path to see if it can be eliminated without loss in accuracy)*

Covering Algorithms

- Strategy for generating a rule set directly: for each class in turn, find a rule set that covers all instances in it (excluding instances not in the class)
- This approach is called a *covering* approach because at each stage a rule is identified that covers some of the instances

Example: generating a rule

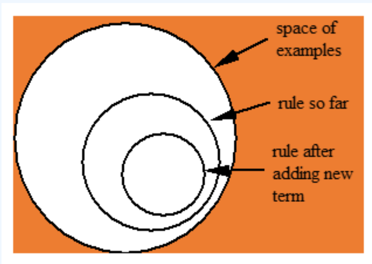


A Simple Covering Algorithm

- Generates a rule by adding tests that maximize the rule's accuracy
- Similar to situation in decision trees: problem of selecting an attribute to split on
  - But: decision tree inducer maximizes overall purity
- Each new test reduces the rule's coverage

A Simple Covering Algorithm

- Generates a rule by adding tests that maximize the rule's accuracy
- Similar to situation in decision trees: problem of selecting an attribute to split on
  - But: decision tree inducer maximizes overall purity
- Each new test reduces the rule's coverage



Selecting a Test

- **Goal:** maximize accuracy
  - t = total number of instances covered by rule
  - p = positive examples of the class covered by rule
  - $t - p$ = number of errors made by rule
  - $\Rightarrow$ Select test that maximizes the ratio p/t
- We are finished when $p/t = 1$ or the set of instances can't be split any further

Contact Lens Data

Age	Spectacle Prescription	Astigmatism	Tear Production Rate	Recommended Lenses
young	myope	no	reduced	none
young	myope	no	normal	soft
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

Example: Contact Lens Data, 1

- **Rule we seek:**

```
If ?
  then recommendation = hard
```

- **Possible tests:**
  - Age = Young 2/8
  - Age = Pre-presbyopic
  - Age = Presbyopic
  - Spectacle prescription = Myope
  - Spectacle prescription = Hypermetrope
  - Astigmatism = no
  - Astigmatism = yes
  - Tear production rate = Reduced
  - Tear production rate = Normal

Example: Contact Lens Data, 2

- **Rule we seek:**

```
If ?
  then recommendation = hard
```

- **Possible tests:**
  - Age = Young 2/8
  - Age = Pre-presbyopic 1/8
  - Age = Presbyopic 1/8
  - Spectacle prescription = Myope 3/12
  - Spectacle prescription = Hypermetrope 1/12
  - Astigmatism = no 0/12
  - Astigmatism = yes 4/12
  - Tear production rate = Reduced 0/12
  - Tear production rate = Normal 4/12

Modified Rule and Resulting Data

- **Rule with best test added:**

```
If astigmatism = yes
then recommendation = hard
```

- **Instances covered by modified rule:**

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	Hard
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

Further Refinement, 1

- **Current state:**

```
If astigmatism = yes
  and ?
  then recommendation = hard
```

- **Possible tests:**
  - Age = Young 2/4
  - Age = Pre-presbyopic
  - Age = Presbyopic
  - Spectacle prescription = Myope
  - Spectacle prescription = Hypermetrope
  - Tear production rate = Reduced
  - Tear production rate = Normal

Further Refinement, 2

- **Current state:**

```
If astigmatism = yes
  and ?
  then recommendation = hard
```

- **Possible tests:**
  - Age = Young 2/4
  - Age = Pre-presbyopic 1/4
  - Age = Presbyopic 1/4
  - Spectacle prescription = Myope 3/6
  - Spectacle prescription = Hypermetrope 1/6
  - Tear production rate = Reduced 0/6
  - Tear production rate = Normal 4/6

Modified Rule and Resulting Data

- **Rule with best test added:**

```
If astigmatism = yes
  and tear production rate = normal
  then recommendation = hard
```

- **Instances covered by modified rule:**

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Normal	None

Further Refinement, 3

- **Current state:**

```
If astigmatism = yes
  and tear production rate = normal
  and ?
  then recommendation = hard
```

- **Possible tests:**
  - Age = Young
  - Age = Pre-presbyopic
  - Age = Presbyopic
  - Spectacle prescription = Myope
  - Spectacle prescription = Hypermetrope

Further Refinement, 4

- **Current state:**

```
If astigmatism = yes
  and tear production rate = normal
  and ?
  then recommendation = hard
```

- **Possible tests:**
  - Age = Young 2/2
  - Age = Pre-presbyopic 1/2
  - Age = Presbyopic 1/2
  - Spectacle prescription = Myope 3/3
  - Spectacle prescription = Hypermetrope 1/3
- Tie between the first and the fourth test
  - We choose the one with greater coverage

The Result

- **Final rule:**

```
If astigmatism = yes
  and tear production rate = normal
  and spectacle prescription = myope
  then recommendation = hard
```

- **Second rule for recommending "hard lenses":**
  (built from instances not covered by the first rule)

```
If age = young and astigmatism = yes
  and tear production rate = normal
  then recommendation = hard
```

- These two rules cover all "hard lenses":
  - Process is repeated with other two classes

Pseudo-code for PRISM

```
For each class C
  Initialize E to the instance set
  While E contains instances in class C
    Create a rule R with an empty left-hand side that predicts class C
    Until R is perfect (or there are no more attributes to use) do
      For each attribute A not mentioned in R, and each value v,
        Consider adding the condition A = v to the left-hand side of R
        Select A and v to maximize the accuracy p/t
          (break ties by choosing the condition with the largest p)
        Add A = v to R
    Remove the instances covered by R from E
```


Separate and Conquer Algorithms

- **PRISM is a separate-and-conquer algorithm**
  - Identify a rule that covers many instances in the class
  - Separate out the covered instances
  - Continue with the remaining instances

Nearest Neighbor

Instance-based Representation

- Simplest form of learning: *rote learning*
  - Training instances are searched for instance that most closely resembles new instance
  - The instances themselves represent the knowledge
  - Also called *instance-based* learning
- Similarity function defines what's "learned"
- Instance-based learning is *lazy learning*
- Methods:
  - *nearest-neighbor*
  - *k-nearest-neighbor*

The Distance Function

- Simplest case: one numeric attribute
  - Distance is the difference between the two attribute values involved (or a function thereof)
- Several numeric attributes: normally, Euclidean distance is used and attributes are normalized
- Nominal attributes: distance is set to 1 if values are different, 0 if they are equal
- Are all attributes equally important?
  - Weighting the attributes might be necessary

Instance-based Learning: Nearest Neighbor Algorithm

- Distance function defines what's learned
- Most instance-based schemes use Euclidean distance:

$$\sqrt{(a_1^{(1)} - a_1^{(2)})^2 + (a_2^{(1)} - a_2^{(2)})^2 + \ldots + (a_k^{(1)} - a_k^{(2)})^2}$$

- $a^{(1)}$ and $a^{(2)}$: two instances with k attributes
- Taking the square root is not required when comparing distances
- Other popular metric: city-block (Manhattan) metric
  - Adds differences without squaring them:

$$|a_1^{(1)} - a_1^{(2)}| + |a_2^{(1)} - a_2^{(2)}| + \ldots + |a_k^{(1)} - a_k^{(2)}|$$

Distance Function (cont.)

- Minkowski distance

$$\left(\sum_{i=1}^{k} \left| a_i^{(1)} - a_i^{(2)} \right|^p \right)^{\frac{1}{p}}$$

- When $p = 1$, Manhattan distance
- When $p = 2$, Euclidean distance

Example: Dissimilarity Matrices

- **Points and Attributes:**

Point	Attribute 1	Attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5

Dissimilarity Matrices:

Manhattan (L$_1$):

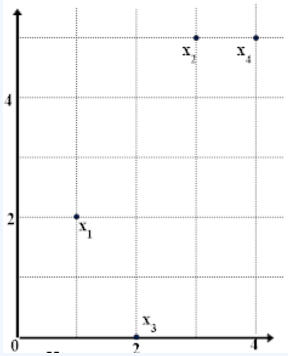
L	x1	x2	x3	x4
x1	0	5	3	6
x2	5	0	6	1
x3	3	6	0	7
x4	6	1	7	0

Euclidean (L$_2$):

L2	x1	x2	x3	x4
x1	0	3.61	2.24	4.24
x2	3.61	0	5.10	1
x3	2.24	5.10	0	5.39
x4	4.24	1	5.39	0

Example

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Dissimilarity Matrices

Manhattan (L$_1$)

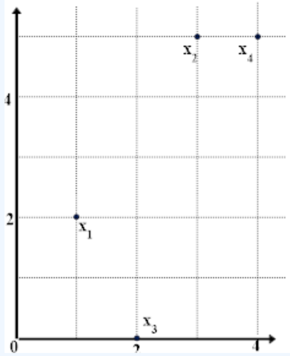
L	x1	x2	x3	x4
x1	0			
x2		0		
x3			0	
x4				0

Euclidean (L$_2$)

L2	x1	x2	x3	x4
x1	0			
x2		0		
x3			0	
x4				0

Example

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Dissimilarity Matrices

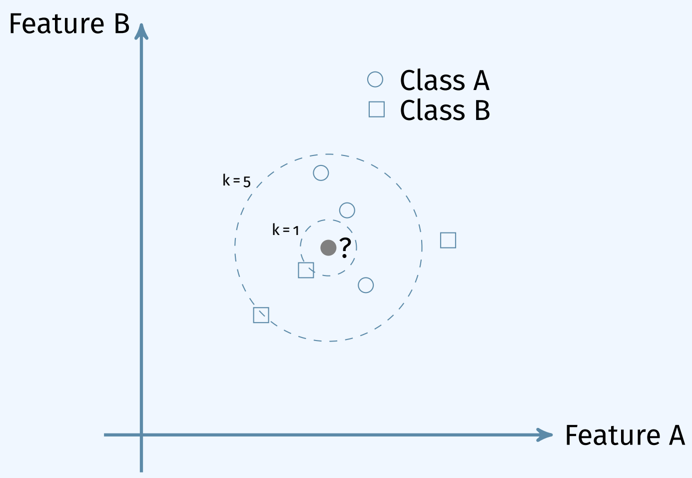
Manhattan (L$_1$)

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean (L$_2$)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Nearest Neighbor Classification



Normalization and Other Issues

- Different attributes are measured on different scales → need to be *normalized*:

$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i} \quad \text{or} \quad a_i = \frac{v_i - \mathrm{Avg}(v_i)}{\mathrm{StDev}(v_i)}$$

- v_i: the actual value of attribute i
- Nominal attributes: distance either 0 or 1
- Common policy for missing values: assumed to be maximally distant (given normalized attributes)

Discussion of Nearest Neighbor

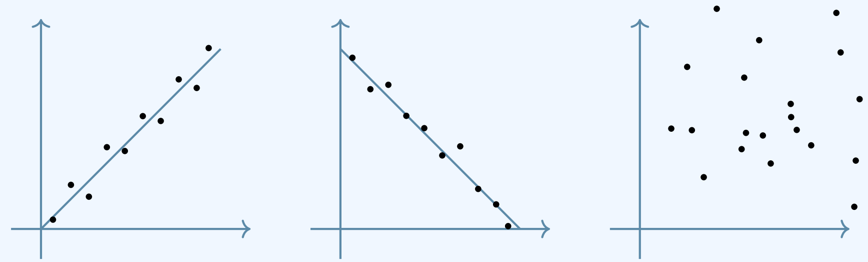
- Often very accurate
- … but slow:
  - Simple version scans entire training data to derive a prediction
  - Remedy: use of kD-tree and Ball-trees
- Assumes all attributes are equally important
  - Remedy: attribute selection or weights
- Possible remedies against noisy instances:
  - Take a majority vote over the k nearest neighbors
  - Removing noisy instances from dataset (difficult!)
- Statisticians have used k-NN since early 1950s

Statistical Modeling

Statistical Modeling

- "Opposite" of 1R: use all the attributes
- Two assumptions: Attributes are
  - *equally important*
  - *statistically independent* (given the class value)
    - I.e., knowing the value of one attribute says nothing about the value of another (if the class is known)
- Independence assumption is almost never correct!
- But . . . this scheme works well in practice

Types of Correlation



Positive Correlation Negative Correlation No Correlation

Naïve Bayes: Discussion

- Naïve Bayes works surprisingly well (even if independence assumption is clearly violated)
- Why? Because classification doesn't require accurate probability estimates *as long as* maximum probability is assigned to the correct class
- However: adding too many redundant attributes will cause problems (e.g., identical attributes)
- Note also: many numeric attributes are not normally distributed (→ *kernel density estimators*)

Probabilities for Weather Data

Outlook			Temperature			Humidity			Windy			Play	
	Yes	No		Yes	No		Yes	No		Yes	No	Yes	No
Sunny	2	3	Hot	2	2	High			False			9	5
Overcast	4	0	Mild	4	2	Normal			True				
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot			High			False			9/14	5/14
Overcast	4/9	0/5	Mild			Normal			True				
Rainy	3/9	2/5	Cool										

Probabilities for Weather Data

Outlook			Temperature			Humidity			Windy			Play	
	Yes	No		Yes	No		Yes	No		Yes	No	Yes	No
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Probabilities for Weather Data

- A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Likelihood of the two classes:

- For "yes" = $\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$
- For "no" = $\frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206$

Conversion into a probability by normalization:

- $P(\text{"yes"}) = \frac{0.0053}{0.0053+0.0206} = 0.205$
- $P(\text{"no"}) = \frac{0.0206}{0.0053+0.0206} = 0.795$

Bayes's Rule

- Probability of event H given evidence E:

$$\Pr(H \mid E) = \frac{\Pr(E \mid H) \cdot \Pr(H)}{\Pr(E)}$$

- *A priori* probability of H:
  - Probability of event *before* evidence is seen ($\Pr(H)$)
- *A posteriori* probability of H:
  - Probability of event *after* evidence is seen ($\Pr(H \mid E)$)

Weather Data Example

Evidence E**:**

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Probability of class "yes":

$$\begin{aligned}\Pr(yes \mid E) &= \Pr[Outlook = Sunny \mid yes] \\ &\times \Pr[Temperature = Cool \mid yes] \\ &\times \Pr[Humidity = High \mid yes] \\ &\times \Pr[Windy = True \mid yes] \\ &\times \frac{\Pr[yes]}{\Pr[E]} \\ &= \frac{\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}}{\Pr[E]}\end{aligned}$$

Predict the class value of the following day

A new day:

Outlook	Temp.	Humidity	Windy	Play
Overcast	Cool	High	True	?

The "Zero-Frequency Problem"

- What if an attribute value doesn't occur with every class value?
  - *e.g.*, "Outlook = Overcast" for class "no"
  - Probability will be zero! $\Pr(\text{Outlook = Overcast} \mid \text{yes}) = 0$
  - A *posteriori* probability will also be zero! $\Pr(\text{yes} \mid E) = 0$
  - (No matter how likely the other values are!)
- Remedy: add 1 to the count for every attribute value–class combination (Laplace estimator)
- Result: probabilities will never be zero!
  (Also: stabilizes probability estimates)

Laplace Estimator for Weather Data

Outlook			**Temperature**			**Humidity**			**Windy**			**Play**	
	Yes	No		Yes	No		Yes	No		Yes	No	Yes	No
Sunny	2	3 + 1	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0 + 1	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2 + 1	Cool	3	1								
Sunny	2/9	4/8	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	1/8	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	3/8	Cool	3/9	1/5								

Missing Values

- **Training**: instance is not included in frequency count for attribute value–class combination
- **Classification**: attribute will be omitted from calculation
- **Example:**

Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	True	?

$$\text{Likelihood of class "yes"}: = \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0238$$

$$\text{Likelihood of class "no"}: = \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0343$$

$$P(\text{yes}) = \frac{0.0238}{0.0238 + 0.0343} = 0.41 \quad (41\%)$$

$$P(\text{no}) = \frac{0.0343}{0.0238 + 0.0343} = 0.59 \quad (59\%)$$

Classifying a New Day

- **A new day:**

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

Numeric Attributes

- Usual assumption: attributes have a normal or Gaussian probability distribution (given the class)
- The *probability density function* for the normal distribution is defined by two parameters:
  - Sample mean μ:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$

  - Standard deviation σ:

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2}$$

- Then the density function $f(x)$ is:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Statistics for Weather Data

Outlook			Temperature		Humidity		Windy			Play	
	Yes	No	Yes	No	Yes	No		Yes	No	Yes	No
Sunny	2	3	64, 68,	65, 71,	65, 70,	70, 85,	False	6	2	9	5
Overcast	4	0	69, 70,	72, 80,	70, 75,	90, 91,	True	3	3		
Rainy	3	2	72, ...	85, ...	80, ...	95, ...					
Sunny	2/9	3/5	$\mu = 73$	$\mu = 75$	$\mu = 79$	$\mu = 86$	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	$\sigma = 6.2$	$\sigma = 7.9$	$\sigma = 10.2$	$\sigma = 9.7$	True	3/9	3/5		

Example density value:

$$f(\text{temperature} = 66 \mid \text{yes}) = \frac{1}{\sqrt{2\pi}6.2} e^{-\frac{(66-73)^2}{2\times 6.2^2}} = 0.0340$$

$$f(\text{humidity} = 90 \mid \text{yes}) = \frac{1}{\sqrt{2\pi}10.2} e^{-\frac{(90-79)^2}{2\times 10.2^2}} = 0.0221$$

Classifying a New Day

- A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

$$\text{Likelihood of “yes”} = \frac{2}{9} \times 0.0340 \times 0.0221 \times \frac{3}{9} \times \frac{9}{14} = 0.000036$$

$$\text{Likelihood of “no”} = \frac{3}{5} \times 0.0291 \times 0.0380 \times \frac{3}{5} \times \frac{5}{14} = 0.000136$$

$$P(\text{“yes”}) = \frac{0.000036}{0.000036 + 0.000136} = 20.9\%$$

$$P(\text{“no”}) = \frac{0.000136}{0.000036 + 0.000136} = 79.1\%$$

Note: Missing values during training are not included in calculation of mean and standard deviation.

Probability Densities

- Probability densities $f(x)$ can be greater than 1; hence, they are not probabilities.
  - However, they must integrate to 1: the area under the probability density curve must be 1.
- Approximate relationship between probability and probability density can be stated as:

$$P(x - \varepsilon/2 \leq X \leq x + \varepsilon/2) \approx \varepsilon f(x)$$

  assuming ε is sufficiently small.
- When computing likelihoods, we can treat densities just like probabilities.

Multinomial Naïve Bayes I

- Version of naïve Bayes used for document classification using *bag of words* model.
- $n_1, n_2, \ldots, n_k$: number of times word i occurs in the document.
- $P_1, P_2, \ldots, P_k$: probability of obtaining word i when sampling from documents in class H.
- Probability of observing a particular document E given class H (based on *multinomial distribution*):

$$P(E \mid H) = N! \times \prod_{i=1}^{k} \frac{P_i^{n_i}}{n_i!}$$

- Note: this expression ignores the probability of generating a document of the right length.
  - This probability is assumed to be constant for all classes.

Multinomial Naïve Bayes II

- Suppose dictionary has two words, *yellow* and *blue*
- Suppose $P(\text{yellow} \mid H) = 75\%$ and $P(\text{blue} \mid H) = 25\%$
- Suppose E is the document *"blue yellow blue"*
- Probability of observing document:

$$P(\{\text{blue yellow blue}\} \mid H) = 3! \times \frac{0.75^1}{1!} \times \frac{0.25^2}{2!} = \frac{27}{64}$$

- Suppose there is another class H' with:

$$P(\text{yellow} \mid H') = 10\%, \quad P(\text{blue} \mid H') = 90\%$$

$$P(\{\text{blue yellow blue}\} \mid H') = 3! \times \frac{0.1^1}{1!} \times \frac{0.9^2}{2!} = \frac{243}{1000}$$

- Need to take prior probability of class into account using Bayes' rule
- Factorials do not need to be computed: they cancel out
- Underflows can be prevented by using logarithms

Naïve Bayes: Discussion

- Naïve Bayes works surprisingly well (even if independence assumption is clearly violated)
- Why? Because classification doesn't require accurate probability estimates *as long as maximum probability is assigned to correct class*