

IS 733: Data Mining Spring 2021

Exam Review

Nirmalya Roy

Department of Information Systems

University of Maryland Baltimore County

Exam

- When: Tuesday (5/04) 7:10pm – 9:10 pm
- Where: In Class Online

- Open book, Open notes
- Comprehensive

- Materials for preparation:
 - Lecture slides
 - Homework assignments
 - Textbook

Course Overview

- Introduction to data mining
 - Applications, the data mining process, data mining ethics
- Know your data
 - Instances and attributes, plotting and visualization
- Data preprocessing
 - Data cleaning, integration, transformation, reduction, discretization
 - Principal components analysis
- Data Mining Tutorial using Python
 - Google Colab
- Data Warehousing
 - OLAP vs OLTP, data cubes

Course Overview

- Knowledge representation
 - Linear models, trees, rules
- Supervised learning
 - Decision trees, decision rules
 - Naive Bayes, logistic regression, support vector machines
- Evaluation of supervised learning
 - Hold-out method, cross validation, ROC curves
- Ensemble methods
 - Bagging, boosting, random forests, stacking
- Unsupervised learning
 - Association rule learning, K-means, hierarchical clustering

Course Overview

- Recommender systems
 - Content filtering
 - Collaborative filtering

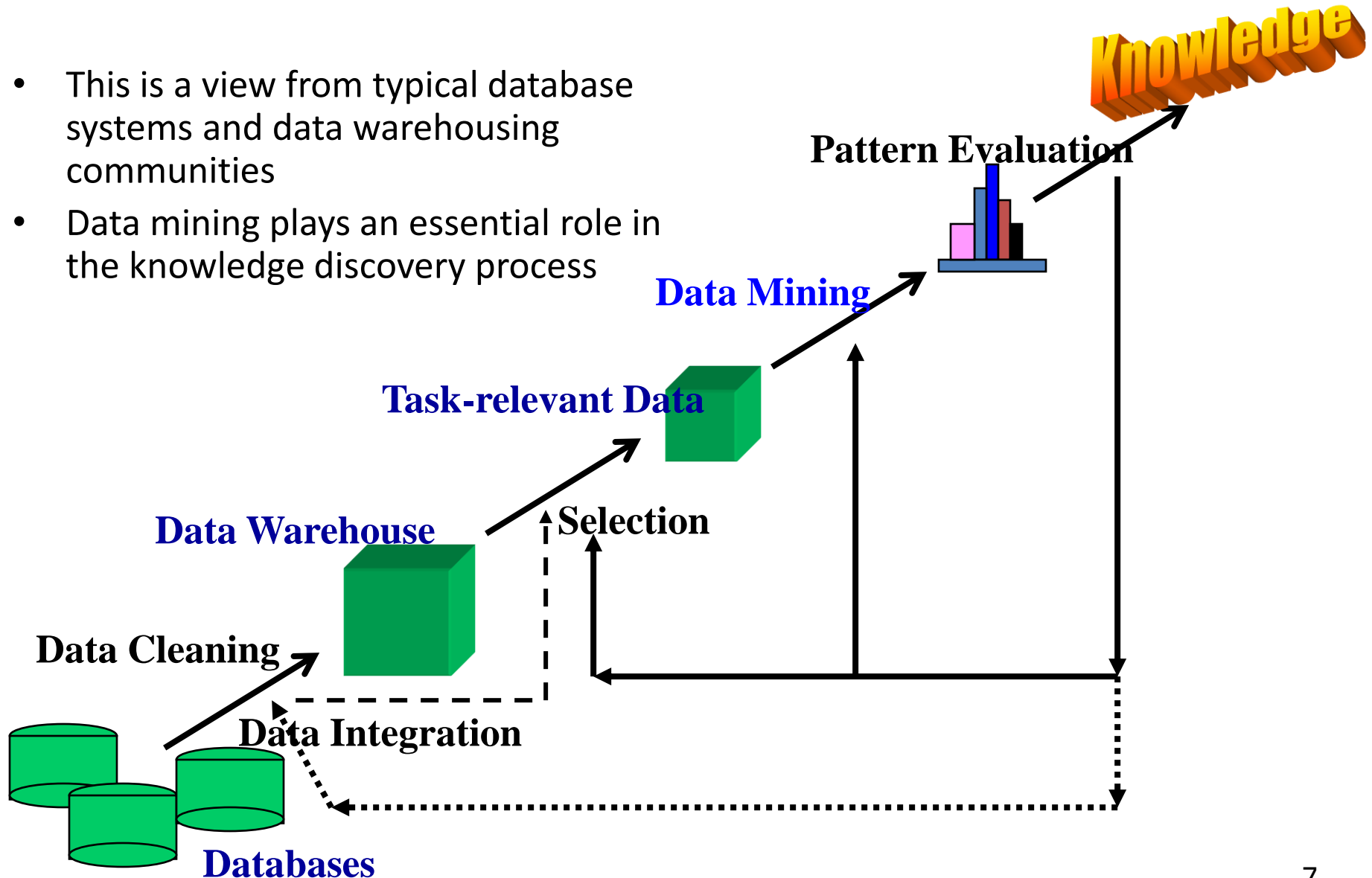
Week 1

- Course overview
- Introduction to data mining
 - Applications
 - The data mining process
 - Data mining ethics

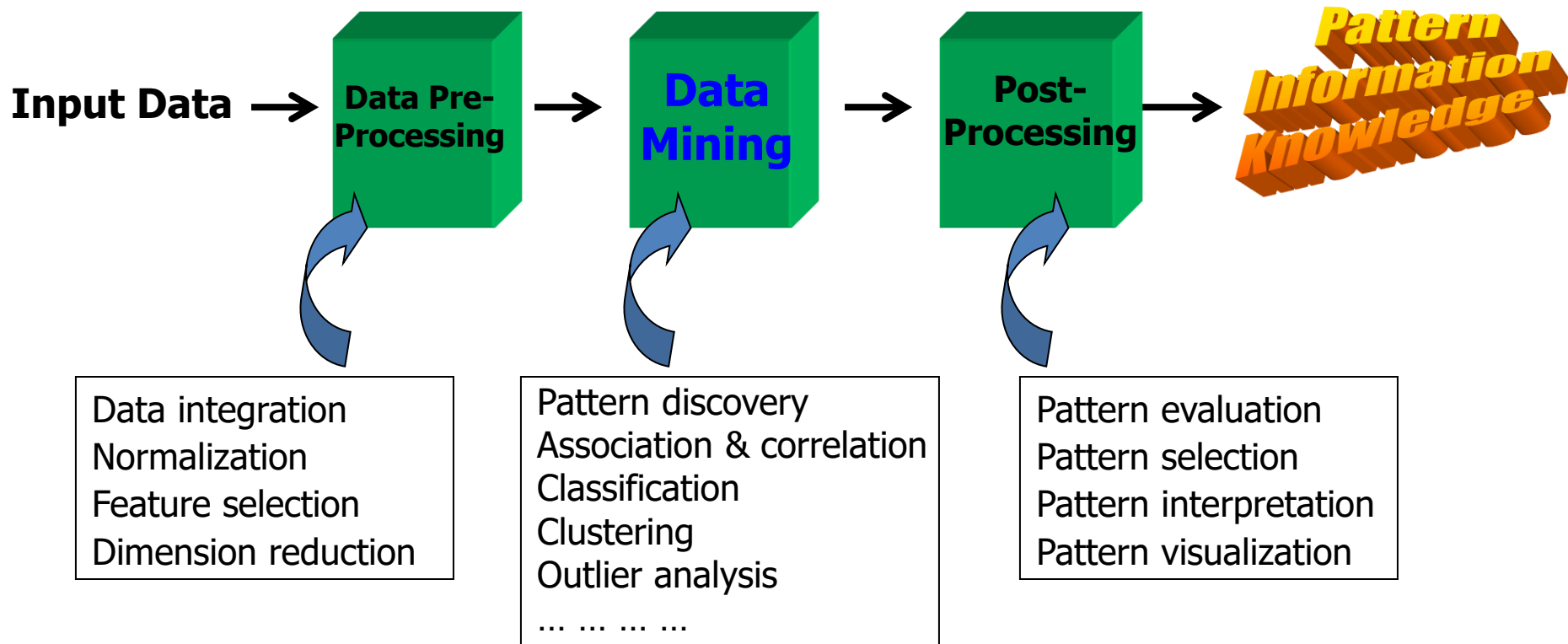


Knowledge Discovery (KDD) Process

- This is a view from typical database systems and data warehousing communities
- Data mining plays an essential role in the knowledge discovery process

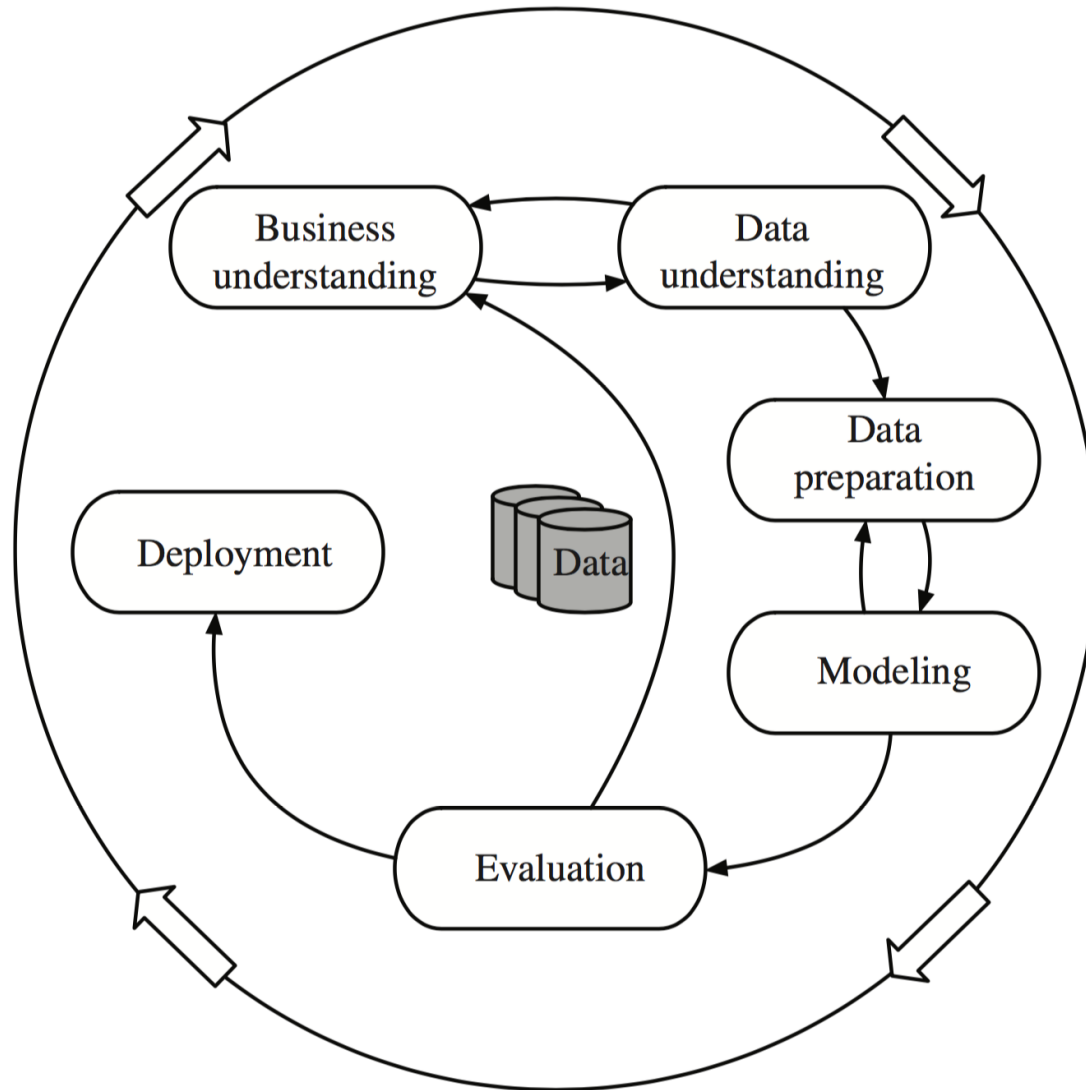


KDD Process: A Typical View from ML and Statistics



- This is a view from typical machine learning and statistics communities

In reality, data mining is an iterative process



Classification

- **Classification and label prediction**
 - Construct models (functions) based on some training examples
 - Describe and distinguish classes or concepts for future prediction
 - E.g., classify countries based on (climate), or classify cars based on (gas mileage)
 - Predict some unknown class labels
- **Typical methods**
 - Decision trees, naïve Bayesian classification, support vector machines, neural networks, rule-based classification, pattern-based classification, logistic regression, ...
- **Typical applications**
 - Credit card fraud detection, direct marketing, classifying stars, diseases, web-pages, ...

The weather problem

- Conditions for playing a certain game

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

Examples

Features / attributes

Class label
(what we want to predict)

The weather problem

- Conditions for playing a certain game

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

If outlook = sunny and humidity = high then play = no

If outlook = rainy and windy = true then play = no

If outlook = overcast then play = yes

If humidity = normal then play = yes

If none of the above then play = yes

Classifying iris flowers

	Sepal length	Sepal width	Petal length	Petal width	Type
1	5.1	3.5	1.4	0.2	Iris setosa
2	4.9	3.0	1.4	0.2	Iris setosa
...					
51	7.0	3.2	4.7	1.4	Iris versicolor
52	6.4	3.2	4.5	1.5	Iris versicolor
...					
101	6.3	3.3	6.0	2.5	Iris virginica
102	5.8	2.7	5.1	1.9	Iris virginica
...					

```
If petal length < 2.45 then Iris setosa
If petal width < 2.10 then Iris versicolor
...
```

Fielded applications

- The result of learning—or the learning method itself—is deployed in practical applications
 - Processing loan applications
 - Screening images for oil slicks
 - Marketing and sales...
 - Electricity supply forecasting
 - Diagnosis of machine faults
 - Separating crude oil and natural gas
 - Reducing banding in rotogravure printing
 - Finding appropriate technicians for telephone faults
 - Scientific applications: biology, astronomy, chemistry
 - Automatic selection of TV programs
 - Monitoring intensive care patients

Processing loan applications (American Express)

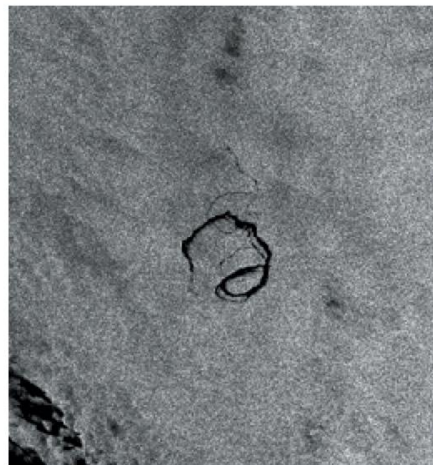
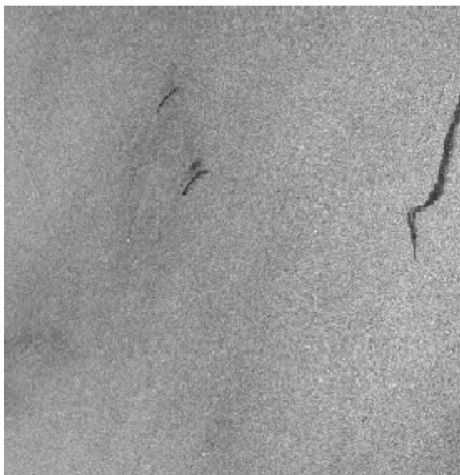
- Given: questionnaire with financial and personal information
- Question: should money be lent?
- Simple statistical method covers 90% of cases
- Borderline cases referred to loan officers
- But: 50% of accepted borderline cases defaulted!
- Solution: reject all borderline cases?
 - No! Borderline cases are most active customers

Enter machine learning

- 1000 training examples of borderline cases
- 20 attributes:
 - age
 - years with current employer
 - years at current address
 - years with the bank
 - other credit cards possessed,...
- Learned rules: correct on 70% of cases
 - human experts only 50%
- Rules could be used to explain decisions to customers

Screening images

- Given: radar satellite images of coastal waters
- Problem: detect oil slicks in those images
- Oil slicks appear as dark regions with changing size and shape
- Not easy: lookalike dark regions can be caused by weather conditions (e.g. high wind)
- Expensive process requiring highly trained personnel



Enter machine learning

- Extract dark regions from normalized image
- Attributes:
 - size of region
 - shape, area
 - intensity
 - sharpness and jaggedness of boundaries
 - proximity of other regions
 - info about background
- Constraints:
 - Few training examples—oil slicks are rare!
 - Unbalanced data: most dark regions aren't slicks
 - Regions from same image form a batch
 - Requirement: adjustable false-alarm rate

Marketing and sales

- Market basket analysis
 - Association techniques find groups of items that tend to occur together in a transaction (used to analyze checkout data)



Diaper

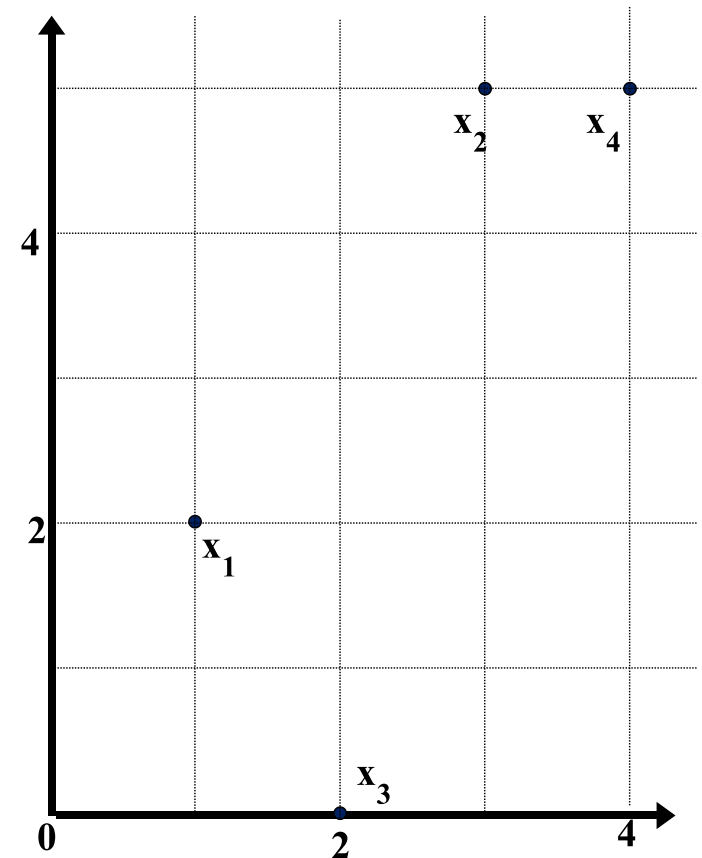


Beer

Week 2

- Know your data

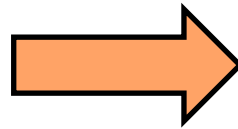
- Instances and attributes
- plotting and visualization



Data Mining

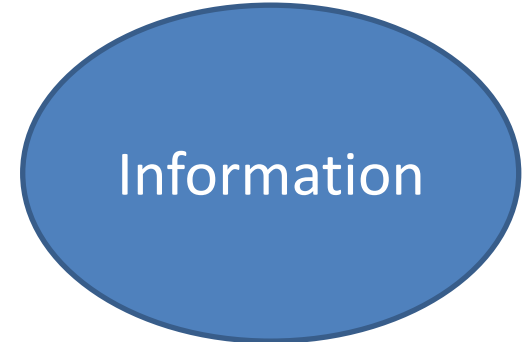


Complicated, noisy,
high-dimensional



Data Mining

Find patterns



Explore, understand, predict

Summary of what the next slides will cover

Input to data mining: concepts, instances, attributes

- Components of the input for learning
- What's a **concept**?
 - Classification, association, clustering, numeric prediction
- What's in an **example**?
 - Relations, flat files, recursion
- What's in an **attribute**?
 - Nominal, ordinal, interval, ratio
- **Preparing the input**
 - ARFF, sparse data, attributes, missing and inaccurate values, unbalanced data, getting to know your data

Components of the input

- **Concepts**: kinds of things that can be learned
 - Aim: intelligible and operational **concept description**
- **Instances**: the individual, independent examples of a concept to be learned
 - A.k.a. **examples, data points, samples, data objects**
 - More complicated forms of input with dependencies between examples are possible
- **Attributes**: measuring aspects of an instance
 - A.k.a. features, dimensions, covariates, variables
 - We will focus on nominal and numeric ones

What's a concept?

- **Concept**: thing to be learned
- **Concept description**: output of the learning scheme
- Styles of learning:
 - **Classification learning**:
predicting a discrete class
 - **Association learning**:
detecting associations between features
 - **Clustering**:
grouping similar instances into clusters
 - **Numeric prediction**:
predicting a numeric quantity

Classification learning

- Example problems: weather data, contact lenses, irises, labor negotiations
- **Classification learning** is *supervised*
 - Scheme is provided with actual outcome
- Outcome is called the *class* of the example
- Measure success on fresh data for which class labels are known (*test data*)
- In practice success is often measured subjectively

Classification example: the weather problem

- Conditions for playing a certain game

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

**Instances
a.k.a.
examples**

**Attributes
a.k.a. features**

**Class label
(what we want to
predict)**

Association learning

- Can be applied if no class is specified and any kind of structure is considered “interesting”
- Difference to 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, such as minimum coverage and minimum accuracy

Association rule mining example: Marketing and sales

- Market basket analysis
 - Association techniques find groups of items that tend to occur together in a transaction
(used to analyze checkout data)



Diaper



Beer

Clustering

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

	Sepal length	Sepal width	Petal length	Petal width	Type
1	5.1	3.5	1.4	0.2	Iris setosa
2	4.9	3.0	1.4	0.2	Iris setosa
...					
51	7.0	3.2	4.7	1.4	Iris versicolor
52	6.4	3.2	4.5	1.5	Iris versicolor
...					
101	6.3	3.3	6.0	2.5	Iris virginica
102	5.8	2.7	5.1	1.9	Iris virginica
...					

Numeric prediction

- Variant of classification learning where “class” is **numeric** (also called “**regression**”)
- Learning is supervised
 - Scheme is being provided with target value
- Measure success on test data

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

What's in an example?

- **Instance:** specific type of example
 - Things 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
- Rather restricted form of input
 - No relationships between objects
- Most common form in practical data mining

Types of Data Sets

- Record

- Relational records
- Data matrix, e.g., numerical matrix, crosstabs
- Document data: text documents: term-frequency vector
- Transaction data

- Graph and network

- World Wide Web
- Social or information networks
- Molecular Structures

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

- Ordered

- Video data: sequence of images
- Temporal data: time-series
- Sequential Data: transaction sequences
- Genetic sequence data

- Spatial, image and multimedia:

- Spatial data: maps
- Image data:
- Video data:

<i>TID</i>	<i>Items</i>
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Data Matrix

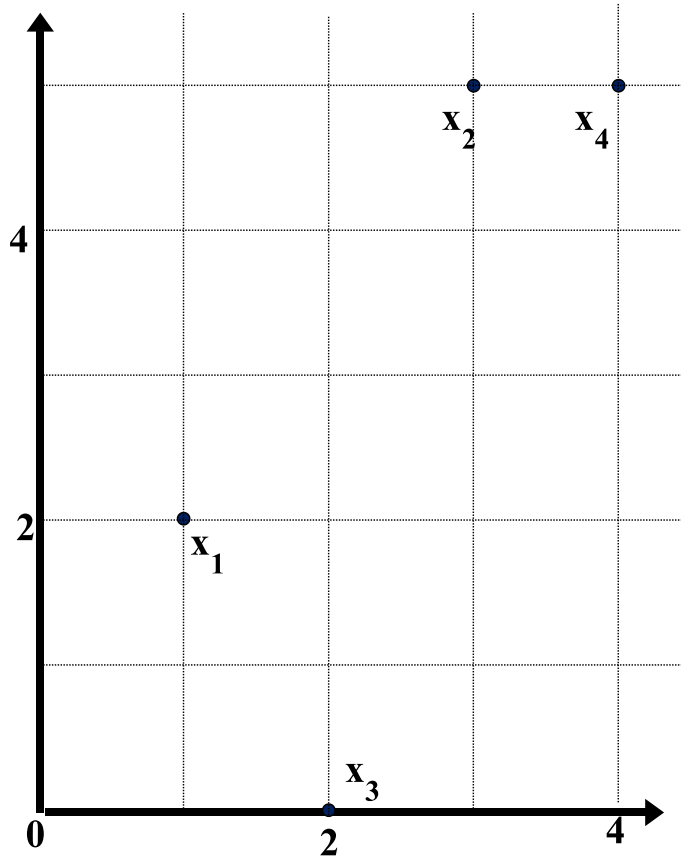
- Data matrix

- n instances with p attributes

	Attributes				
Instances	x_{11}	\dots	x_{1f}	\dots	x_{1p}
	\dots	\dots	\dots	\dots	\dots
	x_{i1}	\dots	x_{if}	\dots	x_{ip}
	\dots	\dots	\dots	\dots	\dots
	x_{n1}	\dots	x_{nf}	\dots	x_{np}

Need to convert data set to this form to apply standard machine learning algorithms!

Example: Data Matrix



Data Matrix

point	attribute1	attribute2
$x1$	1	2
$x2$	3	5
$x3$	2	0
$x4$	4	5

Attributes

- **Attribute** (or **dimensions, features, variables**): a data field, representing a characteristic or feature of a data object.
 - *E.g., customer_ID, name, address*
- Types:
 - Nominal
 - Binary
 - Numeric: quantitative
 - Interval-scaled
 - Ratio-scaled

Week 3

- Data preprocessing
 - Data cleaning, integration, transformation, reduction, discretization.

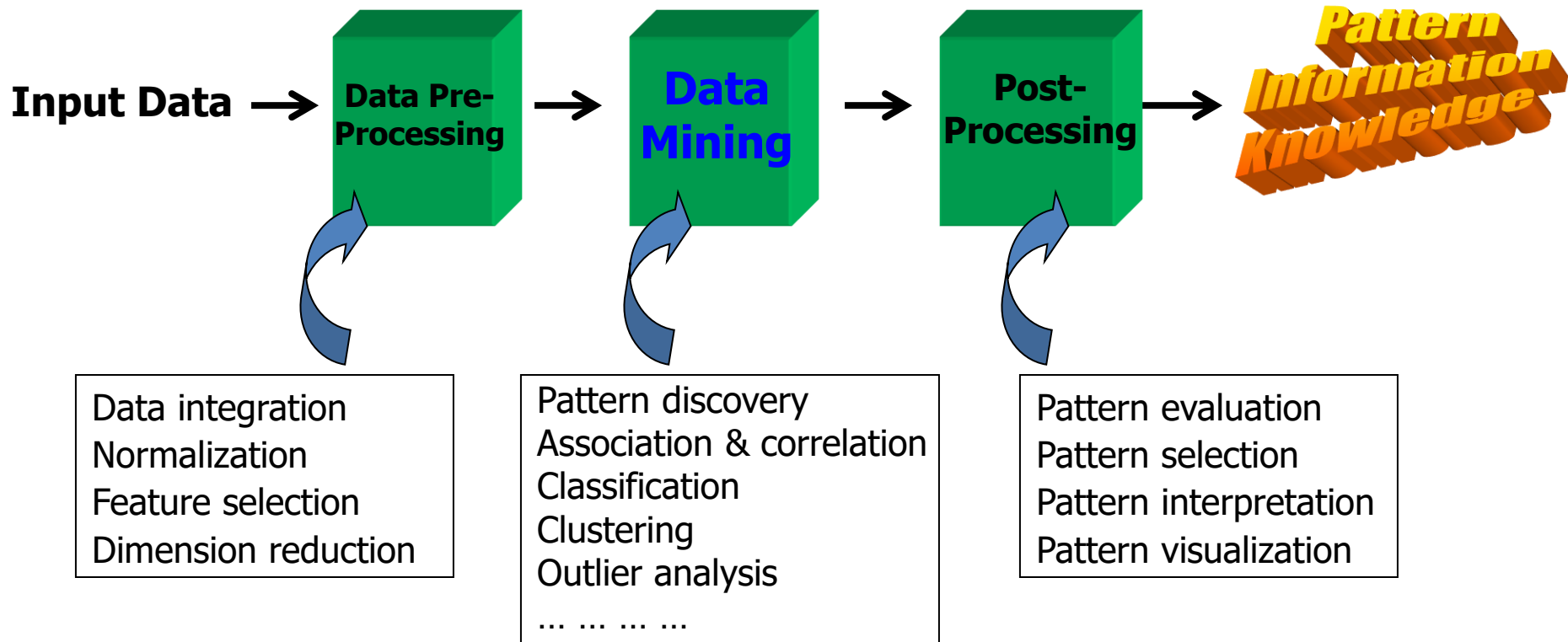


Learning outcomes

By the end of the lesson, you should be able to:

- **Explain** and **employ** several strategies for “getting to know your data”
- **Discuss** common reasons for real-world data being noisy, and strategies for **data cleaning**
- **Identify** redundancy from **data integration** using the **correlation coefficient** and **covariances**
- **Apply** appropriate **data transformations** and **feature construction** techniques to improve the performance of data mining algorithms, and **justify your choices**

KDD Process: A Typical View from ML and Statistics



How do we know which of these methods we should perform?

Getting to know your data

- It's pretty hard to know what methods to use, if you don't know anything about your data
- An initial exploration of your data can help you decide how to proceed



Getting to know your data

- **Inspecting your data “by hand” is a best practice**
- Take the time to **open up your data file and have a look.**
You might be surprised at what you find!
 - You may notice obvious issues with the data, e.g., duplicate records / attributes, nonsensical values, useless attributes,...
 - Too much data to inspect manually?
Take a sample!

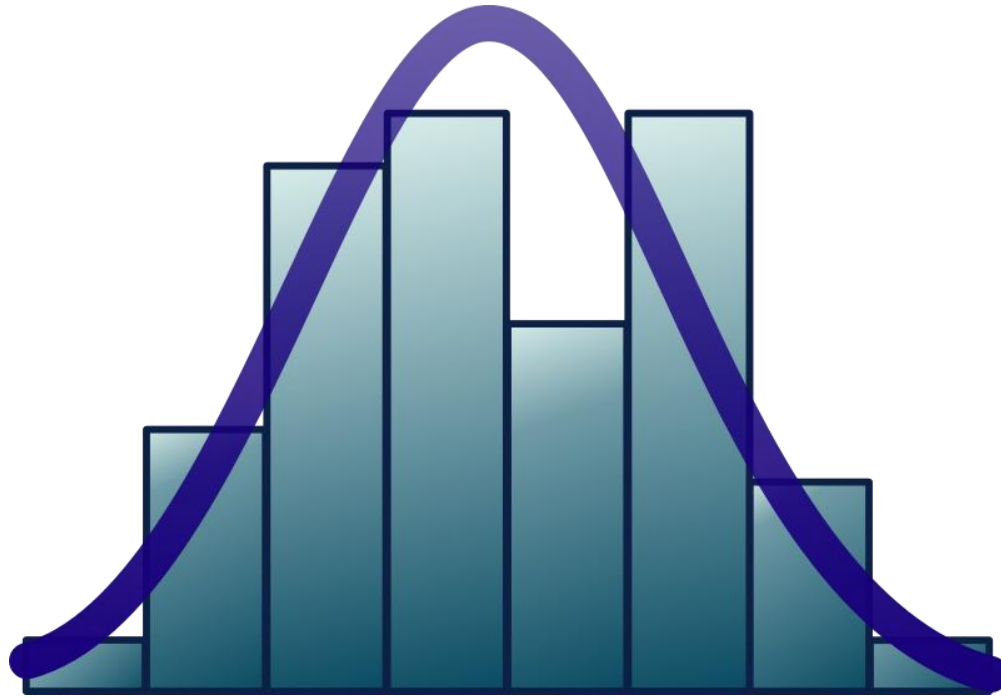


Getting to know your data

- Simple **visualization tools** and **summary statistics** are very useful
 - Make some plots, calculate summary statistics, then think:
 - Is the distribution consistent with background knowledge?
(You may need to consult domain experts)
 - Any obvious outliers?
 - Are some variables heavily correlated with each other?



Getting to know your data



Histogram and best-fit normal distribution for sepal lengths of Iris Versicolor flowers

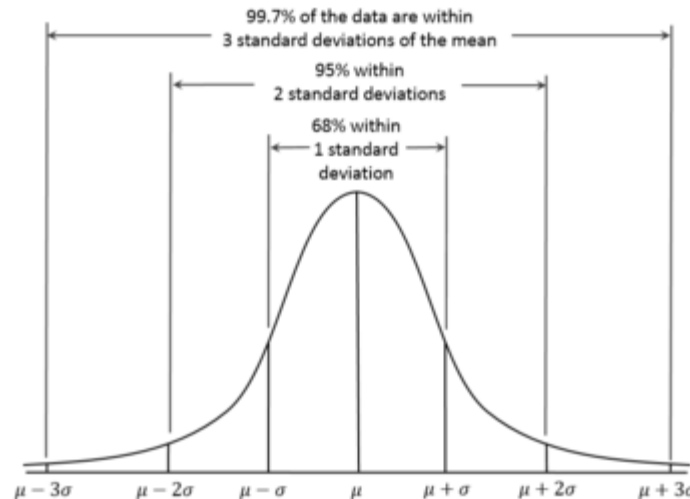
Measuring the Dispersion of Data

- Variance and standard deviation

- **Variance:**
$$\sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i - \mu)^2$$

- **Standard deviation** σ is the square root of variance σ^2

For a normal distribution:



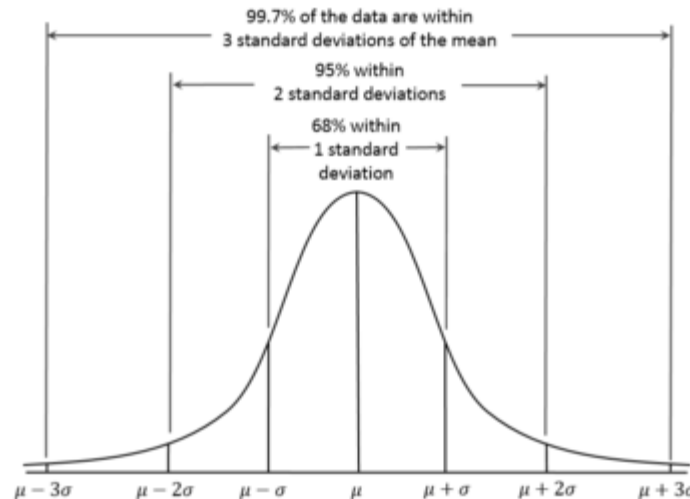
Measuring the Dispersion of Data

- Variance and standard deviation

- **Variance:**
$$\sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i - \mu)^2 = \frac{1}{N} \sum_{i=1}^n x_i^2 - \mu^2$$

- **Standard deviation** σ is the square root of variance σ^2

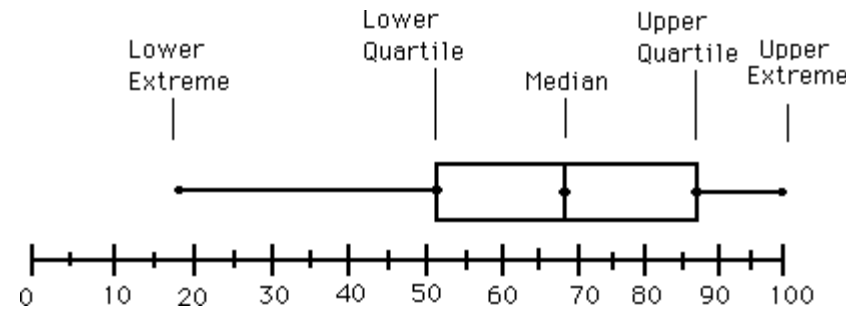
For a normal distribution:



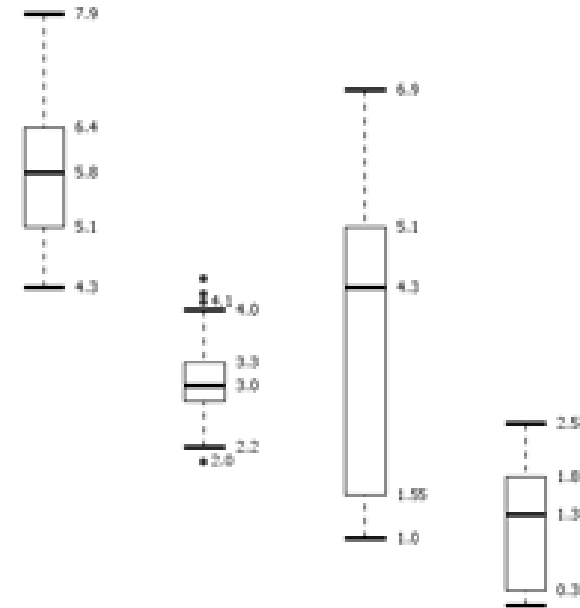
Measuring the Dispersion of Data

- Quartiles, outliers and boxplots
 - **Quartiles:** Q_1 (25th percentile), Q_3 (75th percentile)
 - **Inter-quartile range:** $IQR = Q_3 - Q_1$
 - **Five number summary:** min, Q_1 , median, Q_3 , max
 - **Boxplot:** ends of the box are the quartiles; median is marked; add whiskers, and plot outliers individually
 - **Outlier:** usually, a value higher/lower than $1.5 \times IQR$

Boxplot Analysis

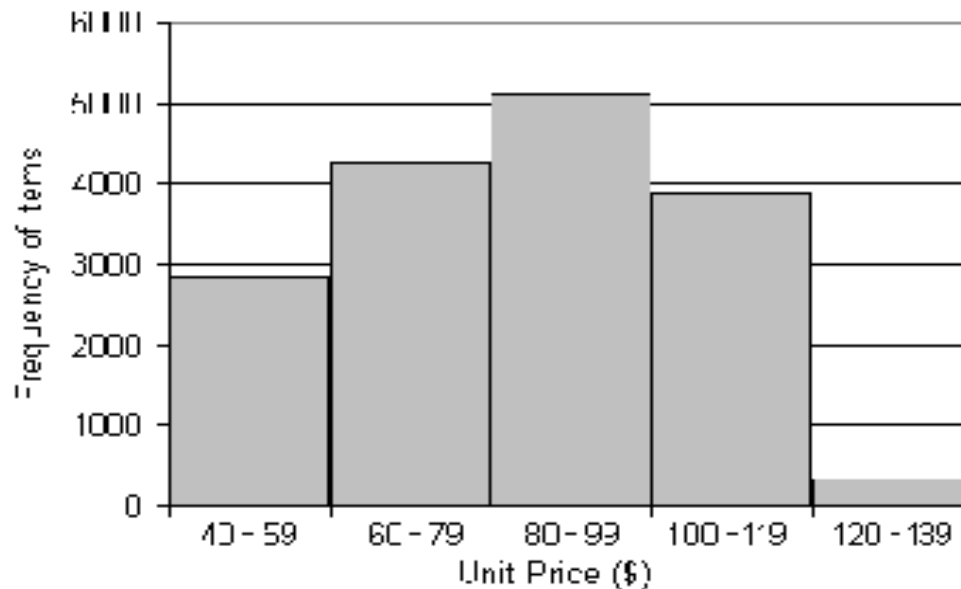


- **Five-number summary** of a distribution
 - Minimum, Q1, Median, Q3, Maximum
- **Boxplot**
 - Data is represented with a box
 - The ends of the box are at the first and third quartiles, i.e., the height of the box is IQR
 - The median is marked by a line within the box
 - Whiskers: two lines outside the box extended to Minimum and Maximum
 - Outliers: points beyond a specified outlier threshold, plotted individually

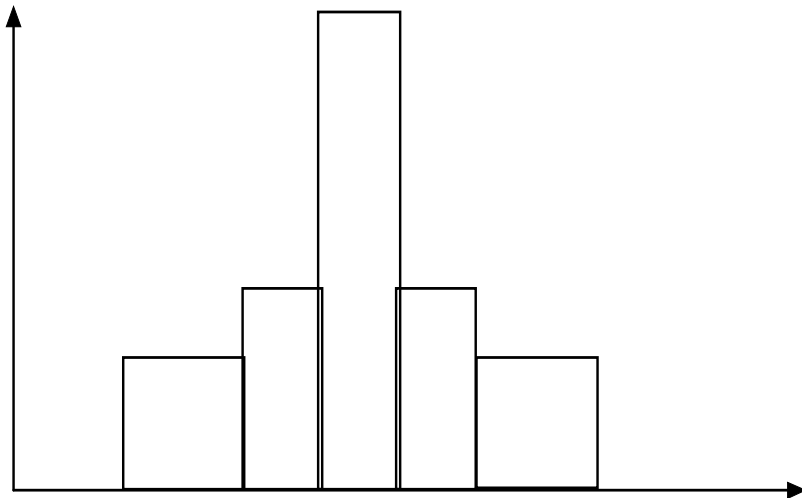
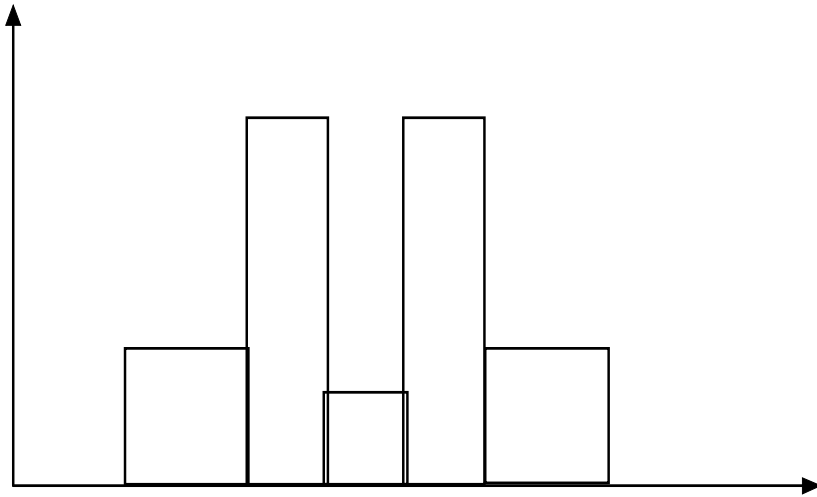


Histogram Analysis

- Graph displays of basic statistical class descriptions
 - Frequency histograms
 - A univariate graphical method
 - Consists of a set of rectangles that reflect the counts or frequencies of the classes present in the given data



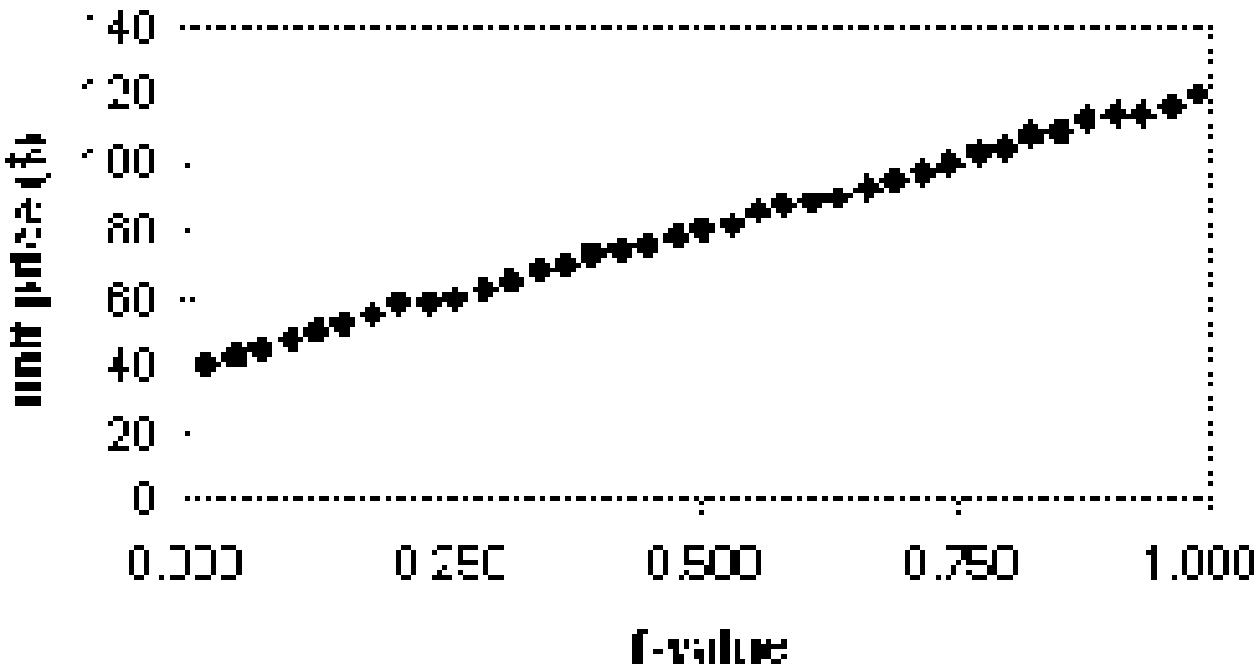
Histograms Often Tell More than Boxplots



- The two histograms shown in the left may have the same boxplot representation
 - The same values for: min, Q1, median, Q3, max
- But they have rather different data distributions

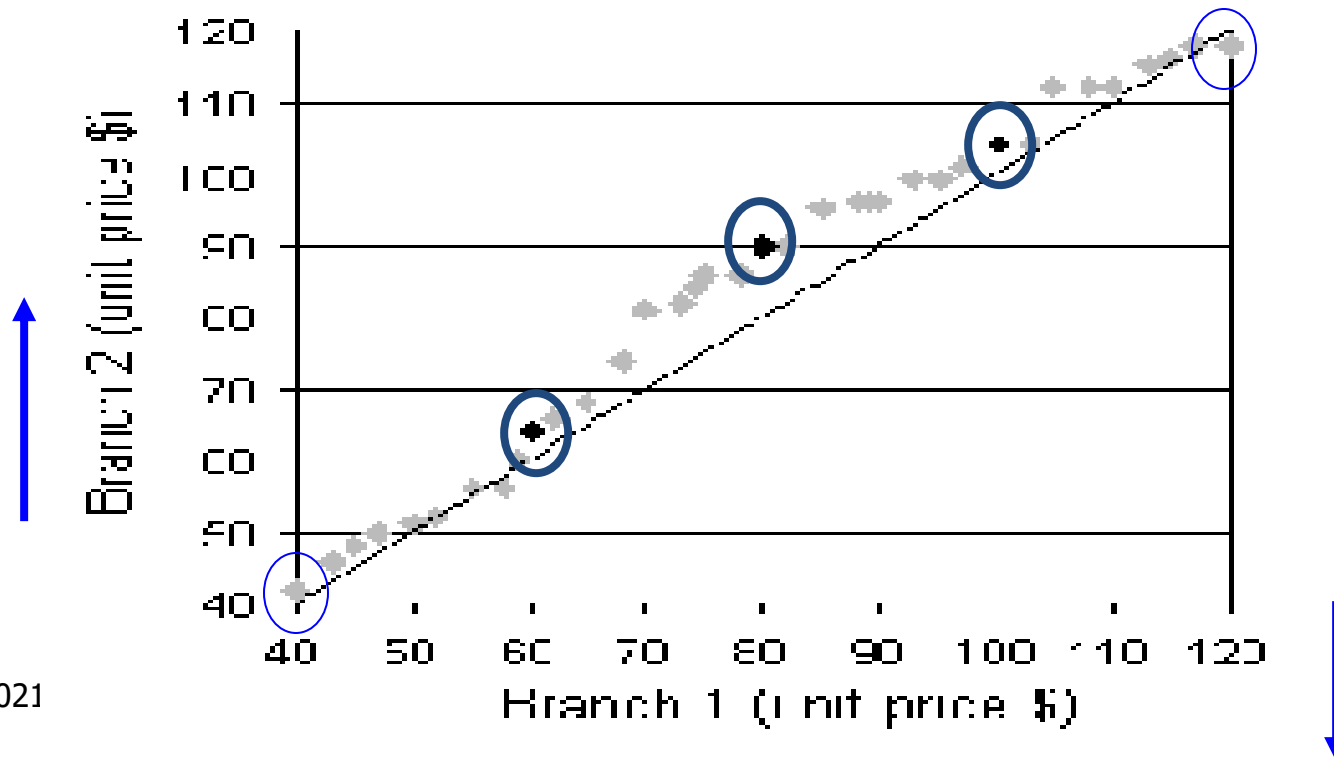
Quantile Plot

- Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)
- Plots **quantile** information
 - For a data x_i data sorted in increasing order, f_i indicates that approximately $100 f_i$ % of the data are below or equal to the value x_i



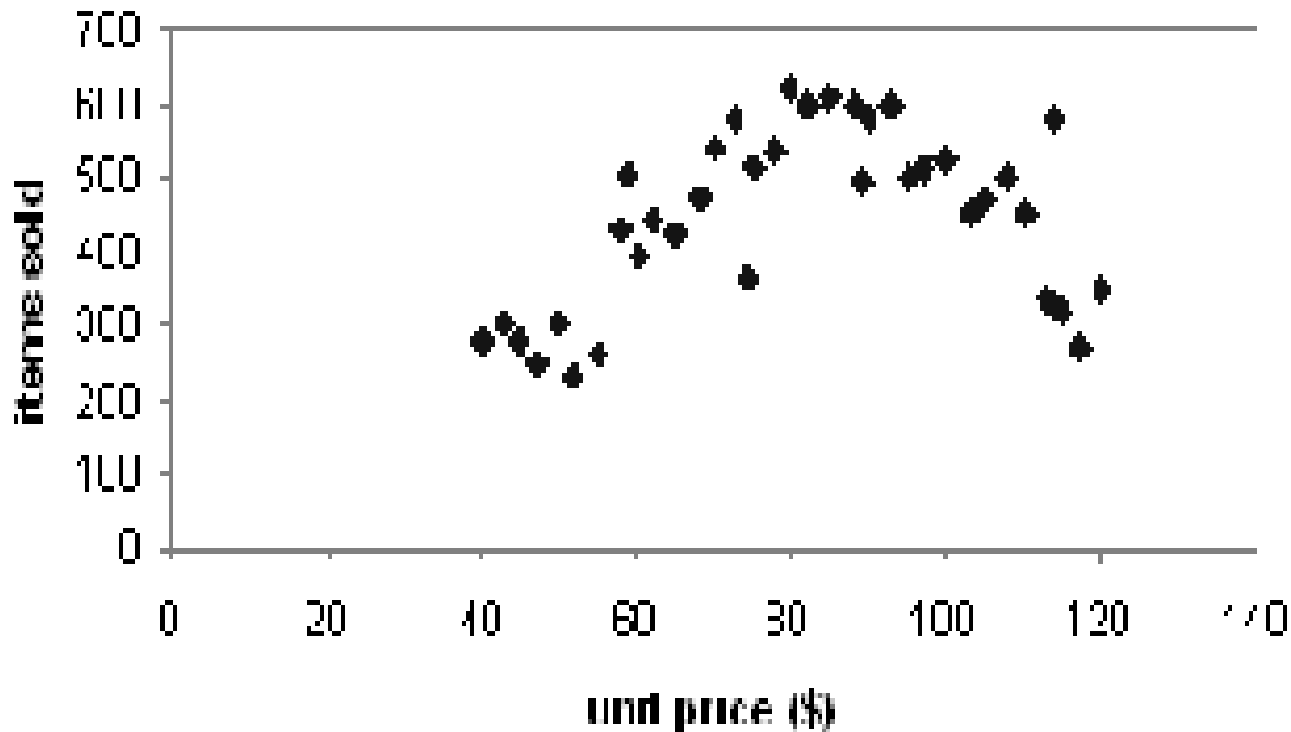
Quantile-Quantile (Q-Q) Plot

- Graphs the quantiles of one univariate distribution against the corresponding quantiles of another
- Allows the user to view whether there is a shift in going from one distribution to another



Scatter plot

- Provides a first look at bivariate data to see clusters of points, outliers, etc
- Each pair of values is treated as a pair of coordinates and plotted as points in the plane



Summary: Simple statistical plots

- **Boxplot:** graphic display of five-number summary
- **Histogram:** x-axis are values, y-axis repres. frequencies
- **Quantile plot:** each value x_i is paired with f_i indicating that approximately $100 f_i \%$ of data are $\leq x_i$
- **Quantile-quantile (q-q) plot:** graphs the quantiles of one univariate distribution against the corresponding quantiles of another
- **Scatter plot:** each pair of values is a pair of coordinates and plotted as points in the plane

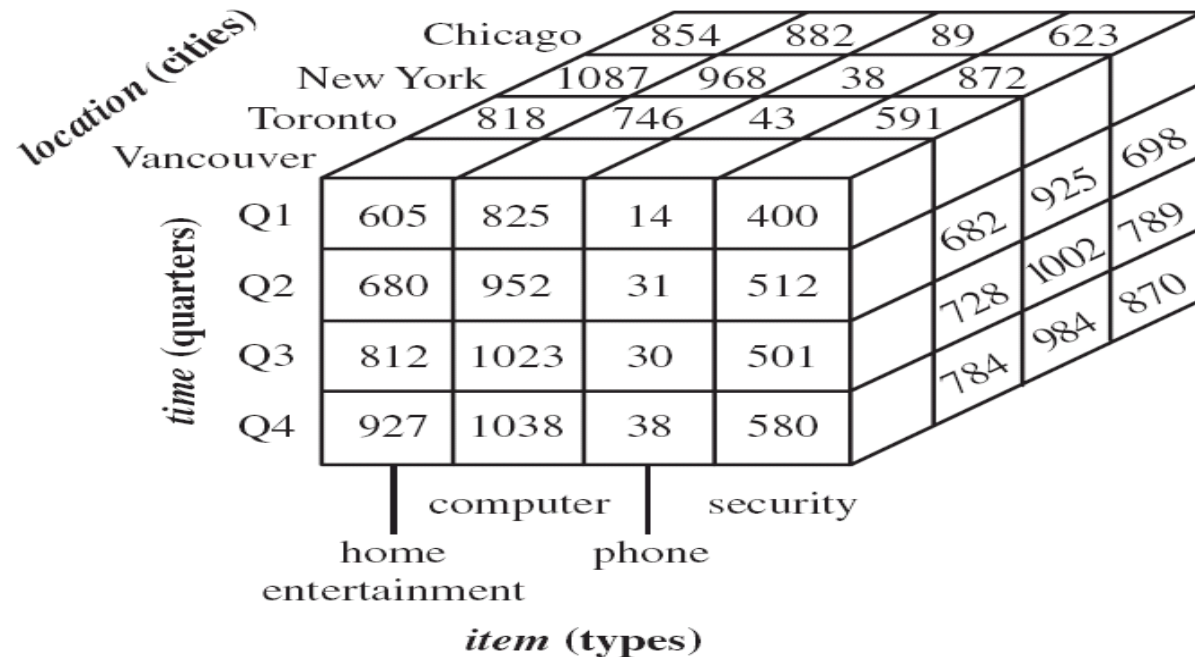
Week 4

- Data Mining Tutorial Python
- Google Colab

Week 5

■ Data warehousing

- OLAP vs OLTP
- Data cubes



Learning outcomes

By the end of the lesson, you should be able to:

- **Enumerate** the key differences between **OLTP** and **OLAP**, and the **corresponding benefits** from the use of a **data warehouse**
- **Compute** the number of **cuboids** in a **data cube**
- **Select** appropriate **OLAP operations** to obtain relevant information from a data warehouse

Data warehousing

Basic concept

What is a Data Warehouse?

- Defined in many different ways, but not rigorously.
 - A decision support database that is maintained **separately** from the organization's operational database
 - Support **information processing** by providing a solid platform of consolidated, historical data for analysis.
- “A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process.” —W. H. Inmon
- Data warehousing:
 - The process of constructing and using data warehouses

Data Warehouse—Subject-Oriented

- Organized around major subjects, such as **customer, product, sales**
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing
- Provide **a simple and concise** view around particular subject issued by **excluding data that are not useful in the decision support process**

Data Warehouse—Integrated

- Constructed by integrating multiple, heterogeneous data sources
 - relational databases, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied
 - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
 - E.g., Hotel price: currency, tax, breakfast covered, etc.
 - When data is moved to the warehouse, it is converted

Data Warehouse—Time Variant

- The time horizon for the data warehouse is significantly longer than that of operational systems
 - Operational database: current value data
 - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)
- Every key structure in the data warehouse
 - Contains an element of time, explicitly or implicitly
 - But the key of operational data may or may not contain “time element”

Data Warehouse—Nonvolatile

- A *physically separate store* of data transformed from the operational environment
- Operational *update of data does not occur* in the data warehouse environment
 - Does not require transaction processing, recovery, and concurrency control mechanisms
 - Requires only two operations in data accessing:
 - *initial loading of data* and *access of data*

Data Warehouse vs. Operational DBMS

- OLAP (on-line analytical processing)
 - Major task of data warehouse system
 - Data analysis and decision making
- OLTP (on-line transaction processing)
 - Major task of traditional relational DBMS
 - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.

OLTP vs. OLAP

	OLTP	OLAP
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
#users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response

Why a Separate Data Warehouse?

- High performance for both systems
 - DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
 - Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation
- Different functions and different data:
 - [missing data](#): Decision support requires historical data which operational DBs do not typically maintain
 - [data consolidation](#): DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
 - [data quality](#): different sources typically use inconsistent data representations, codes and formats which have to be reconciled
- Note: There are more and more systems which perform OLAP analysis directly on relational databases

Data Warehouse Usage

- Three kinds of data warehouse applications
 - Information processing
 - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
 - Analytical processing
 - multidimensional analysis of data warehouse data
 - supports basic OLAP operations, slice-dice, drilling, pivoting
 - Data mining
 - knowledge discovery from hidden patterns
 - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools

Data warehousing

Data Cubes and OLAP Operations

From Tables and Spreadsheets to Data Cubes

- A **data warehouse** is based on a **multidimensional data model** which views data in the form of a data cube
- A data cube, such as **sales**, allows data to be modeled and viewed in multiple dimensions
 - **Dimension tables**, such as **item** (**item_name**, **brand**, **type**), or **time**(**day**, **week**, **month**, **quarter**, **year**)
 - **Fact table** contains **measures** (such as **dollars_sold**) and keys to each of the related dimension tables
- In data warehousing literature, an n-D base cube is called a **base cuboid**. The top most 0-D cuboid, which holds the highest-level of summarization, is called the **apex cuboid**. The lattice of cuboids forms a **data cube**.

Flat view of Data

Data about items

home entertainment	computer	phone	security
605	825	14	400

Data about Quarters

Q1	605
Q2	680
Q3	812
Q4	927

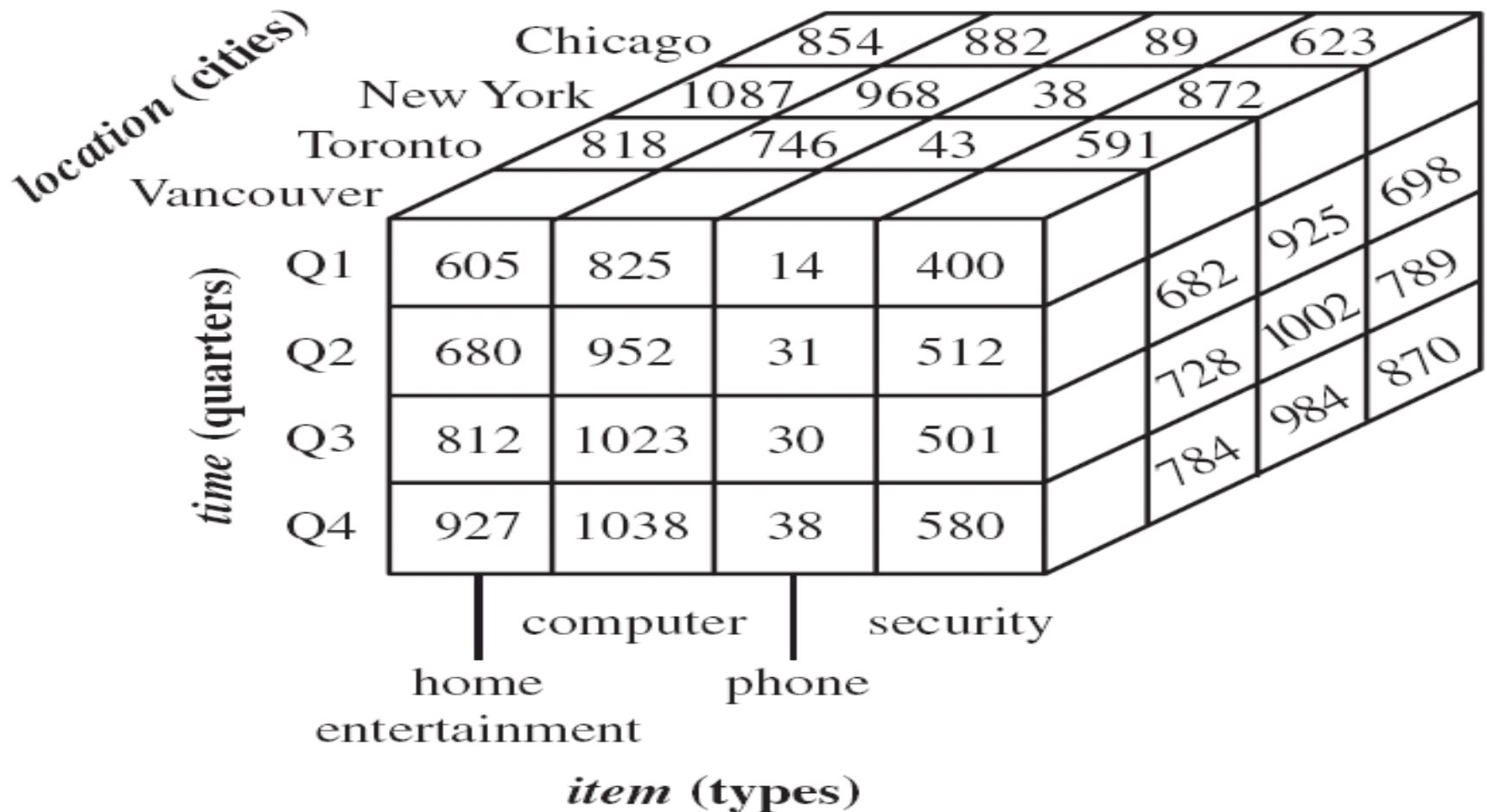
Multidimensional data

<i>time</i> (quarter)	<i>location</i> = “Vancouver”			
	<i>item</i> (type)			
	home entertainment	computer	phone	security
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

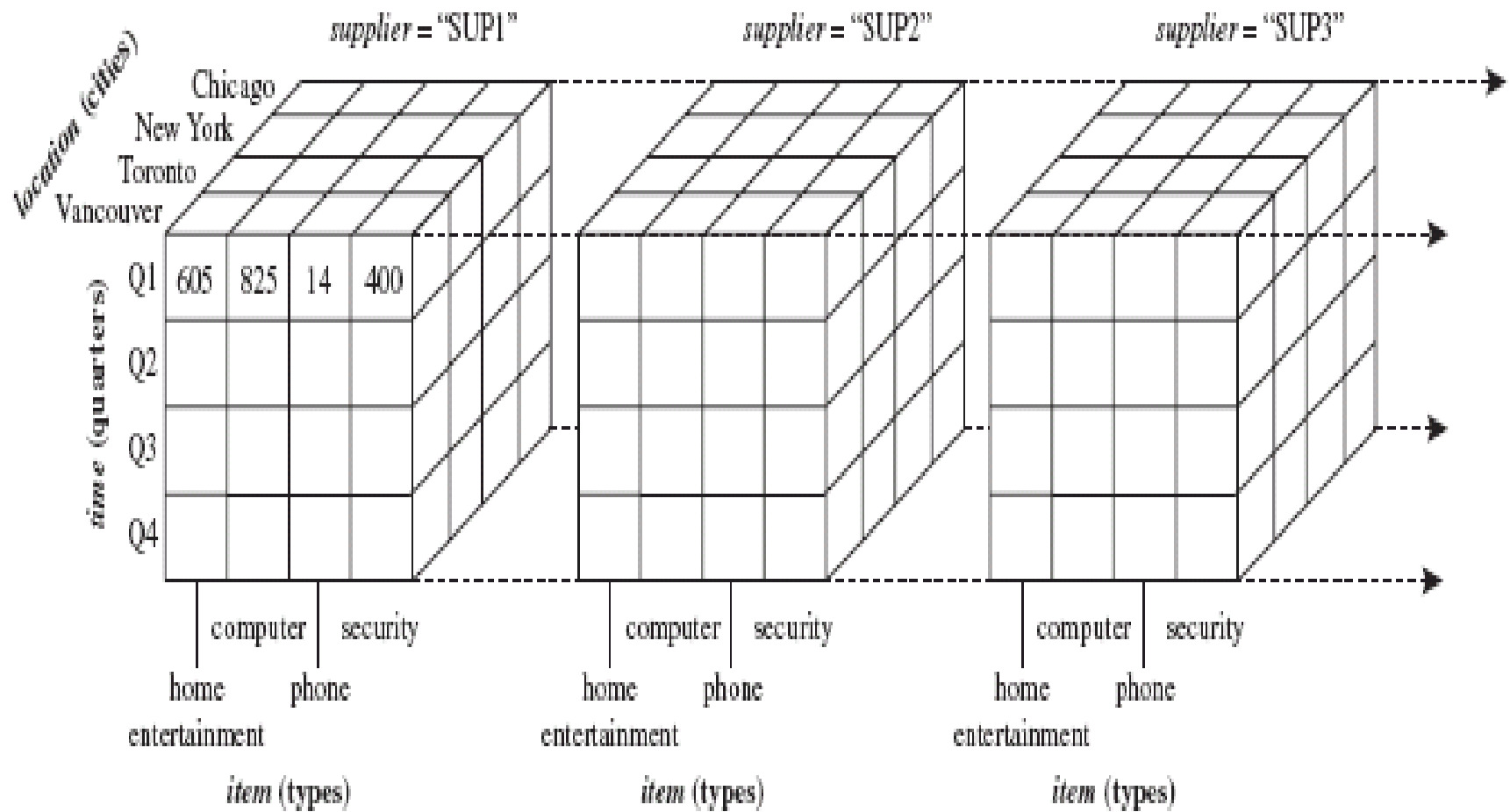
Data for multiple locations

	<i>location</i> = "Chicago"				<i>location</i> = "New York"				<i>location</i> = "Toronto"				<i>location</i> = "Vancouver"			
<i>t</i> <i>l</i> <i>m</i> <i>e</i>	<i>item</i>				<i>item</i>				<i>item</i>				<i>item</i>			
	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

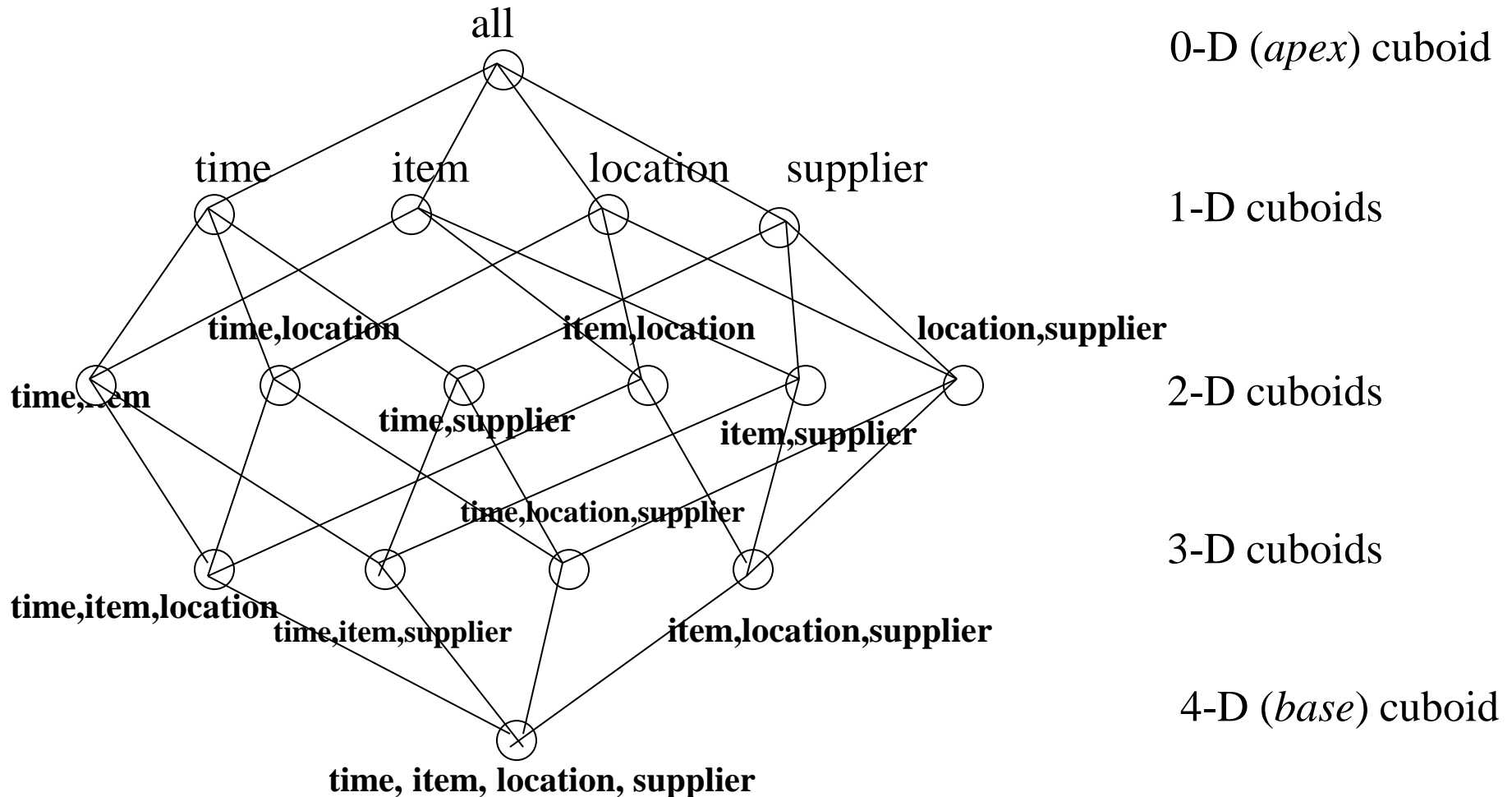
Multidimensional view -3D



Multidimensional view -4D

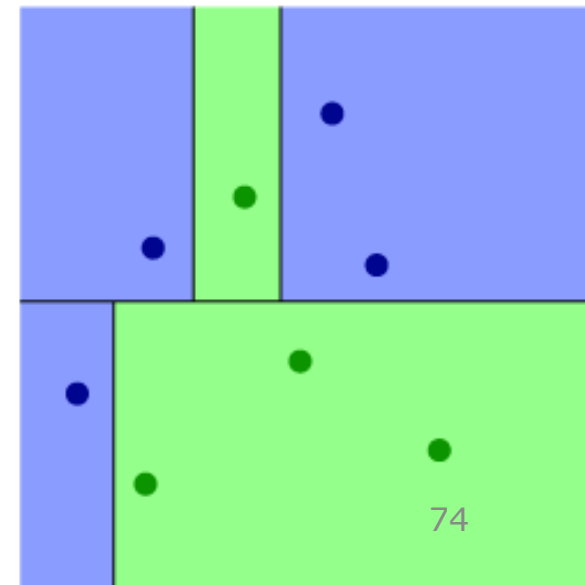
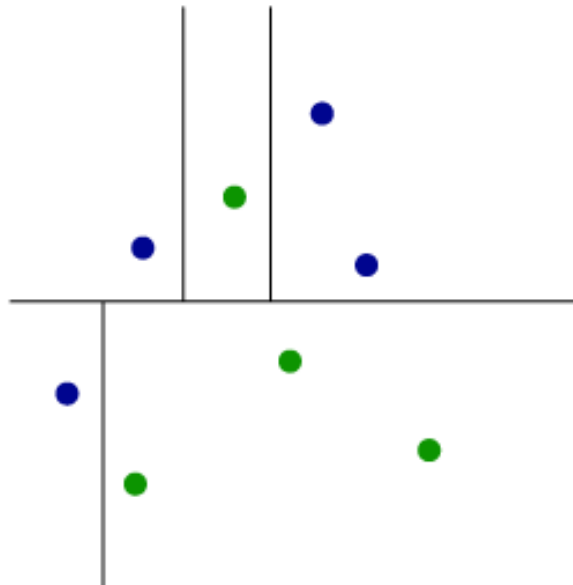


Data Cube: A Lattice of Cuboids



Week 6

- Knowledge representation
 - Linear models
 - Trees
 - Rules
 - Nearest neighbors



Learning outcomes

By the end of the lesson, you should be able to:

- **Discuss** the differences between **decision tree** and **decision rule** learning algorithms
- **Calculate** the **information gain** and **gain ratio** splitting criteria used by decision tree induction algorithms
- **Perform** the steps of the training algorithms for the **PRISM** rule learner, given a small dataset

Supervised learning

- Typical applications:
 - Credit/loan approval
 - Medical diagnosis: if a tumor is cancerous or benign
 - Fraud detection: if a transaction is fraudulent
 - Web page categorization: which category it is



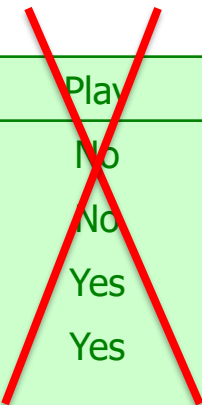
Supervised vs. Unsupervised Learning

- Supervised learning (e.g. classification)
 - Supervision: The training data are accompanied by **labels**, e.g. the **class** of the observations
 - The goal is to predict the labels for new, unseen data

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...

Supervised vs. Unsupervised Learning

- Unsupervised learning (e.g. clustering)
 - The class labels of training data are **unknown**
 - We aim to find patterns in the data without class labels, e.g.
 - clustering the data into groups
 - finding associations between the attributes



Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...

Prediction Problems:

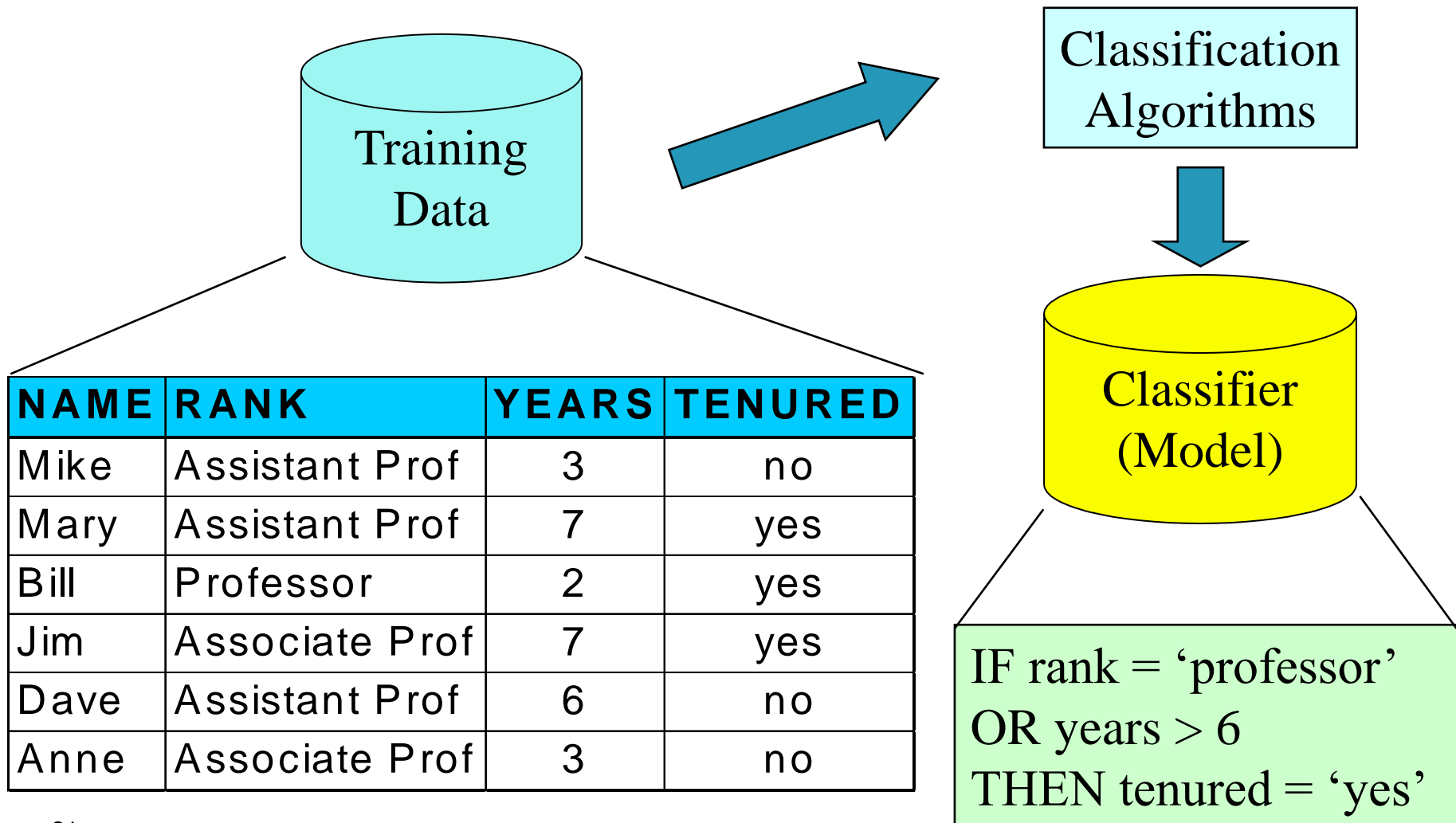
Classification vs. Numeric Prediction

- **Classification**
 - predicts categorical class labels (discrete or nominal)
 - classifies data (constructs a model) based on the training set and the values (**class labels**) in a classifying attribute and uses it in classifying new data
- **Numeric Prediction**
 - models continuous-valued functions, i.e., predicts unknown or missing values

Classification—A Two-Step Process

- Step 1, Model construction:
 - **Given the data, build a model of it!**
 - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class label attribute**
 - The set of tuples used for model construction is **training set**
 - The model is represented as either:
 - classification rules, decision trees, linear model, etc.

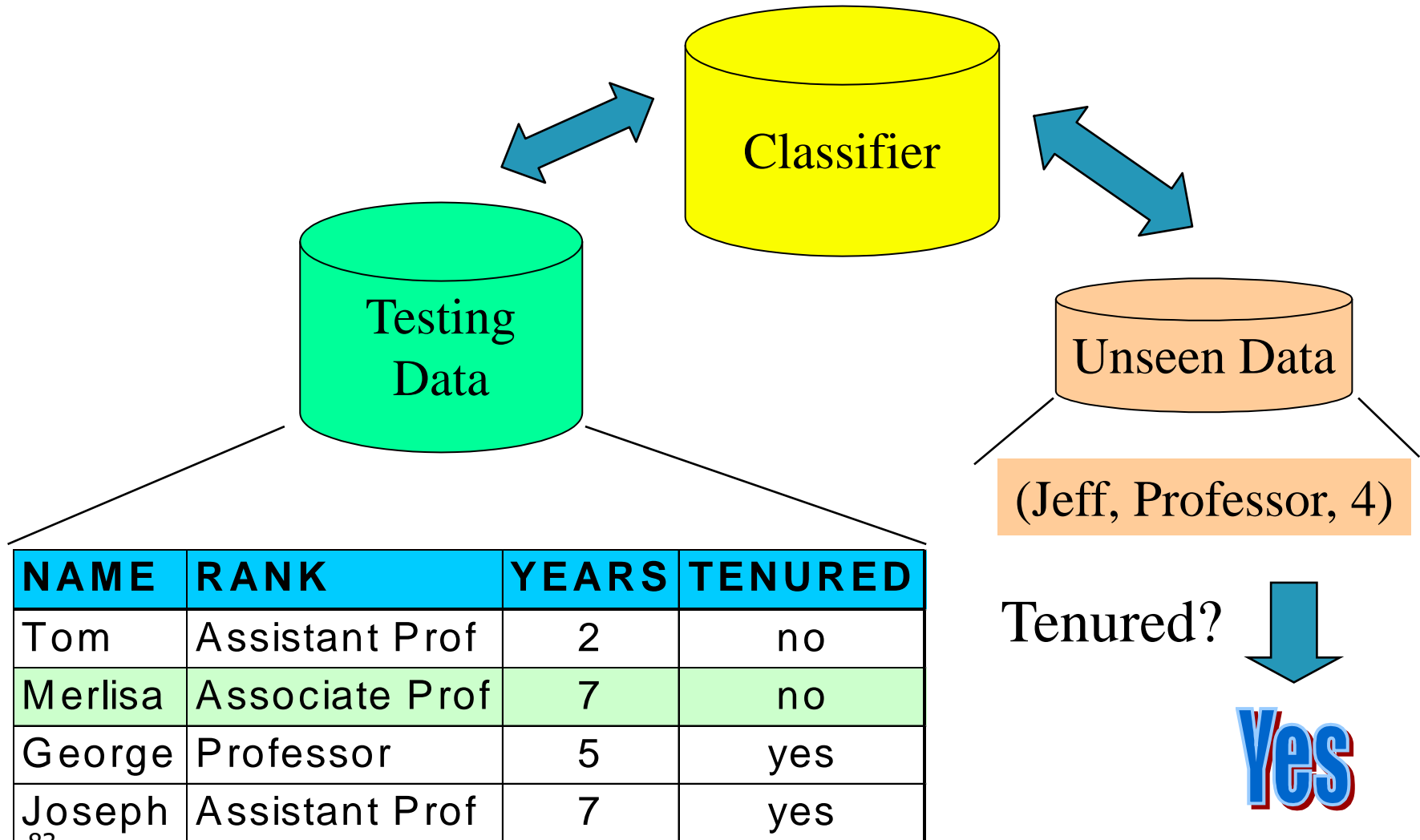
Process (1): Model Construction



Classification—A Two-Step Process

- Step 2, Model usage: for classifying future or unknown objects
 - Estimate accuracy of the model
 - The known label of test sample is compared with the classified result from the model
 - Accuracy rate is the percentage of test set samples that are correctly classified by the model
 - Test set is independent of training set (otherwise overfitting)
 - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation set

Process (2): Using the Model in Prediction



Elements of a Classification Algorithm

- **Representation**
- **Objective function for training**
- **Optimization algorithm**

Elements of a Classification Algorithm

- **Representation**
 - Rules, trees, linear model, instance-based, probabilistic model,...
 - Representation is important for interpretation. Flexibility of representation trades overfitting versus underfitting (bias-variance trade-off).
- **Objective function for training**
- **Optimization algorithm**

Elements of a Classification Algorithm

- **Representation**

- Rules, trees, linear model, instance-based, probabilistic model,...
- Representation is important for interpretation. Flexibility of representation trades overfitting versus underfitting (bias-variance trade-off).

- **Objective function for training**

- E.g. Error rate, accuracy, loss, on training data
- May include **regularization**, encouraging simpler models to prevent overfitting (e.g. via some penalty in the objective function)

- **Optimization algorithm**

Elements of a Classification Algorithm

- **Representation**

- Rules, trees, linear model, instance-based, probabilistic model,...
- Representation is important for interpretation. Flexibility of representation trades overfitting versus underfitting (bias-variance trade-off).

- **Objective function for training**

- E.g. Error rate, accuracy, loss, on training data
- May include **regularization**, encouraging simpler models to prevent overfitting (e.g. via some penalty in the objective function)

- **Optimization algorithm**

- Aims to find “best” classifier according to objective function
- Search algorithm can also avoid overfitting
 - Only consider simple classifiers, or search these first and stop early

Elements of a Classification Algorithm

- **Representation**

- Rules, trees, linear model, instance-based, probabilistic model,...
- Representation is important for interpretation. Flexibility of representation trades overfitting versus underfitting (bias-variance trade-off).

- **Objective function for training**

- E.g. Error rate, accuracy, loss, on training data
- May include **regularization**, encouraging simpler models to prevent overfitting (e.g. via some penalty in the objective function)

- **Optimization algorithm**

- Aims to find “best” classifier according to objective function
- Search algorithm can also avoid overfitting
 - Only consider simple classifiers, or search these first and stop early

- **The whole game: select the above so that the method generalizes well, is computationally efficient, and is interpretable (if desired)**

Simplicity first

- Simple algorithms often work very well!
- There are many kinds of simple structure, e.g.:
 - One attribute does all the work
 - All attributes contribute equally & independently
 - Logical structure with a few attributes suitable for tree
 - A weighted linear combination of the attributes
 - Strong neighborhood relationships based on distance
 - ...
- Success of method depends on the domain

Simplest possible classifier: ZeroR

- Ignore the input! Predict the **majority class**, according to the training data
- **Representation**
 - One rule with no antecedent: *If (true), predict Class=C*
- **Objective function for training**
 - Error on training data
- **Search algorithm**
 - Calculate class frequencies.
Select the one with the highest frequency

Inferring rudimentary rules: 1R

- 1R rule learner:
 - A set of rules that all test **one particular attribute** - the one that yields the lowest classification error
 - Equivalent to a 1-level decision tree

Outlook

Sunny → No

Overcast → Yes

Rainy → Yes

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...

Inferring rudimentary rules: 1R

- Basic version (assumes nominal attributes):
 - For each attribute
 - Make one branch for each value of the attribute
 - **To each branch, assign the most frequent class value** of the instances pertaining to that branch
 - Calculate the error rate of the rules
 - The **Error rate** equals the proportion of instances that do not belong to the majority class of their corresponding branch
 - Choose attribute with lowest **error rate**

1R Rule Learner

- **Representation**
 - Rules (tests on one attribute)
Equivalently – a 1-level decision trees
- **Objective function for training**
 - Error rate on training data
- **Search algorithm**
 - Consider each attribute, create rules predicting class for each attribute value.
 - Pick the attribute that performs best on training data.

Evaluating the weather attributes

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

* indicates a tie

Evaluating the weather attributes

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

* indicates a tie

Dealing with numeric attributes

- Idea: discretize numeric attributes into sub ranges (intervals)
- How to divide each attribute's overall range into intervals?
 - Sort instances according to attribute's values
 - Place breakpoints where (majority) class changes
 - This minimizes the total classification error
- Example: *temperature* from weather data

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

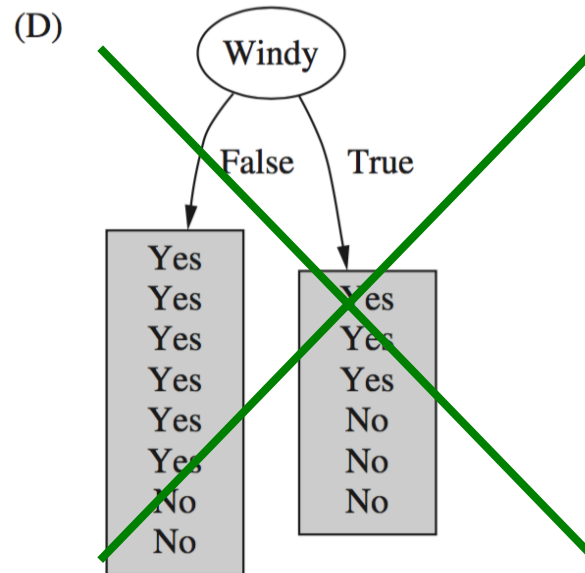
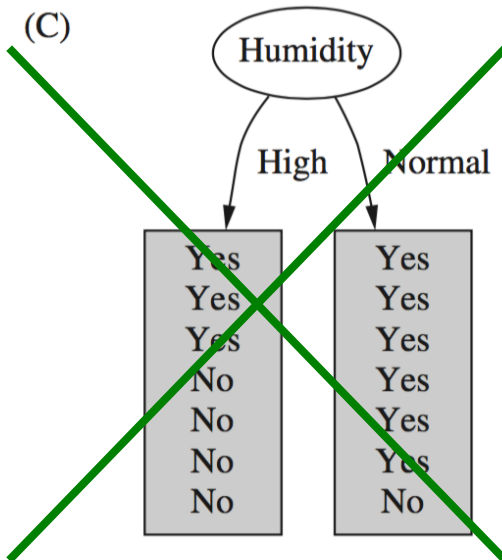
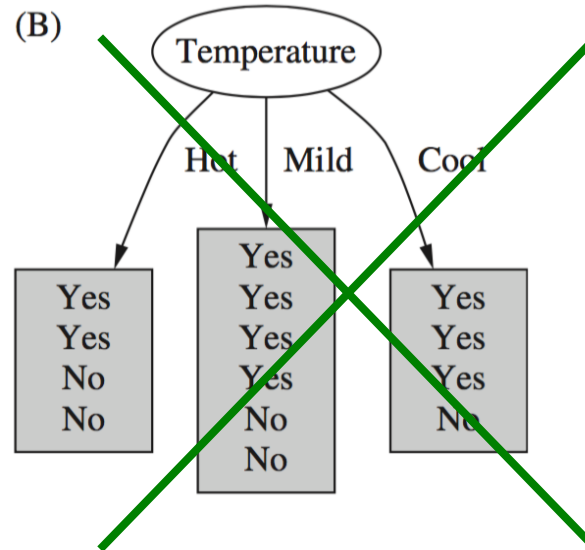
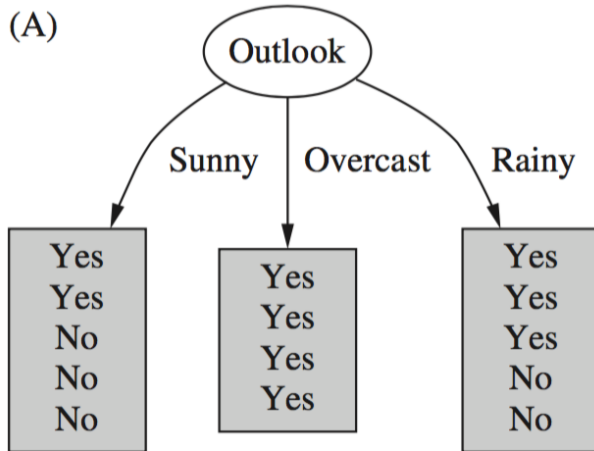
Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...

Constructing decision trees

- Strategy: top down learning using recursive *divide-and-conquer* process
 - First: select attribute for root node
Create branch for each possible attribute value
 - Then: split instances into subsets
One for each branch extending from the node
 - Finally: repeat recursively for each branch, using only instances that reach the branch
- Stop if all instances have the same class

(For numeric attributes, compare to a constant value, e.g. $x_i > c$?)

Which attribute to select?



Criterion for attribute selection

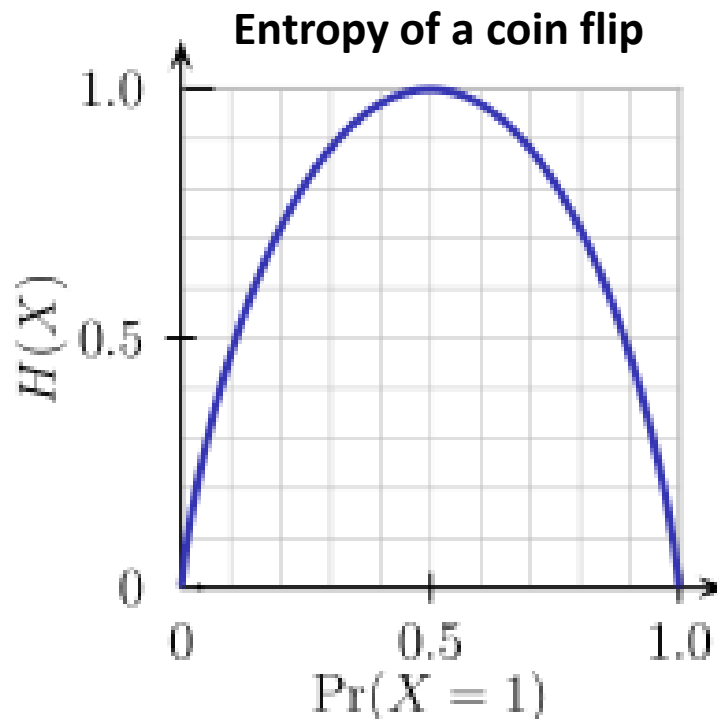
- Which is the best attribute?
 - Want to get the smallest tree
 - Heuristic: choose the attribute that produces the “purest” nodes
- Popular selection criterion: **information gain**
 - Information gain increases with the average purity of the subsets
- Strategy: amongst attributes available for splitting, choose attribute that gives greatest information gain
- Information gain requires measure of *impurity*
- Impurity measure that it uses is the **entropy** of the class distribution, which is a measure from information theory

Wishlist for an impurity measure

- Properties we would like to see in an impurity measure:
 - When node is pure, measure should be zero
 - When impurity is maximal (i.e., all classes equally likely), measure should be maximal
 - Measure should ideally obey *multistage property* (i.e., decisions can be made in several stages, without affecting the overall impurity score):
- It can be shown that **entropy is the only function that satisfies all three properties!**

Brief Review of Entropy

- Entropy (Information Theory)
 - A measure of uncertainty associated with a random variable



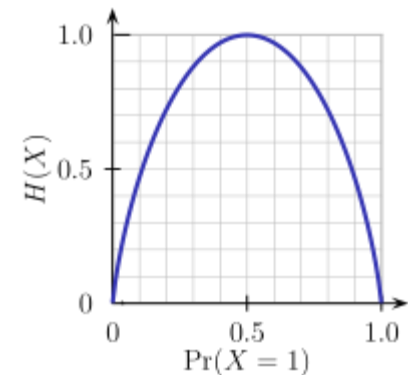
Brief Review of Entropy

- Entropy (Information Theory)
 - A measure of uncertainty associated with a random variable
 - Calculation: For a discrete random variable Y taking m distinct values $\{y_1, \dots, y_m\}$,
 - $H(Y) = -\sum_{i=1}^m p_i \log(p_i)$, where $p_i = P(Y = y_i)$

Brief Review of Entropy

- Entropy (Information Theory)

- A measure of uncertainty associated with a random variable
- Calculation: For a discrete random variable Y taking m distinct values $\{y_1, \dots, y_m\}$,
 - $H(Y) = -\sum_{i=1}^m p_i \log(p_i)$, where $p_i = P(Y = y_i)$
- Interpretation:
 - Higher entropy => higher uncertainty
 - Lower entropy => lower uncertainty



Information Gain

- Select the attribute with the **highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i , estimated by $|C_{i,D}|/|D|$
- **Expected information** (entropy) needed to classify a tuple in D :

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

Information Gain

- Select the attribute with the **highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i , estimated by $|C_{i,D}|/|D|$
- **Expected information** (entropy) needed to classify a tuple in D :

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

- **Information** needed (after using A to split D into v partitions) to classify D :

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

Information Gain

- Select the attribute with the **highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i , estimated by $|C_{i,D}|/|D|$
- **Expected information** (entropy) needed to classify a tuple in D :

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

- **Information** needed (after using A to split D into v partitions) to classify D :

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

- **Information gained** by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Example: attribute *Outlook*

- *Outlook = Sunny* :

$$\text{Info}([2, 3]) = 0.971 \text{ bits}$$

- *Outlook = Overcast* :

$$\text{Info}([4, 0]) = 0.0 \text{ bits}$$

- *Outlook = Rainy* :

$$\text{Info}([3, 2]) = 0.971 \text{ bits}$$

- Expected information for attribute:

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

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
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

Computing information gain

- Information gain: information before splitting – information after splitting

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

- Information gain for attributes from weather data:

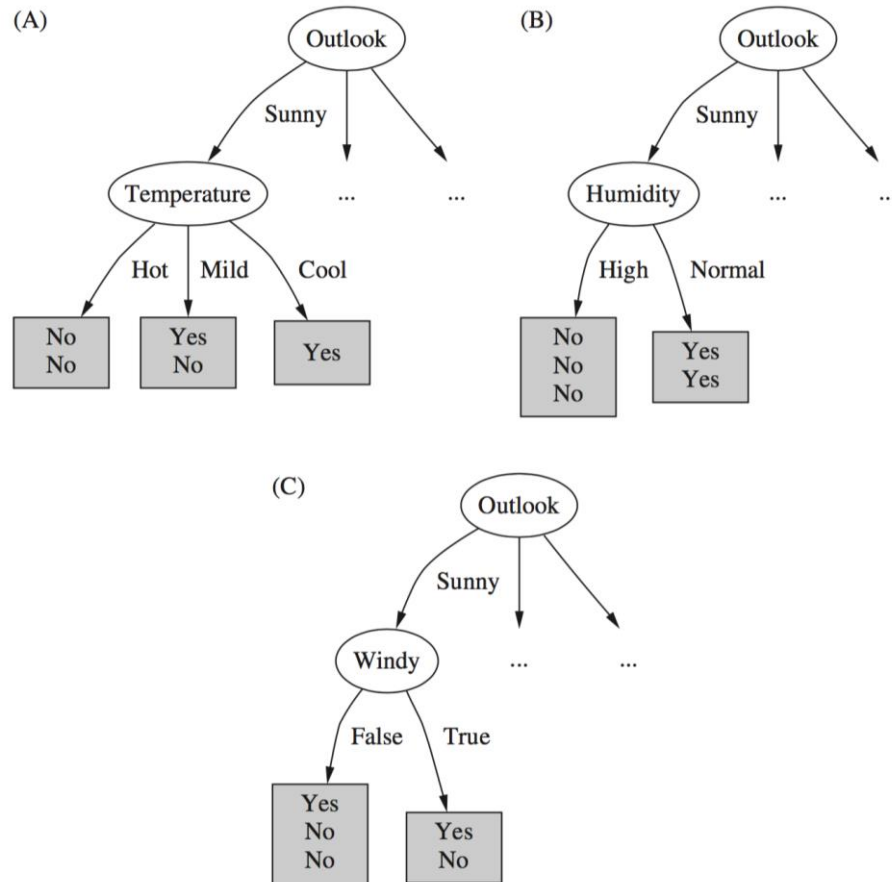
$$\text{Gain}(\textit{Outlook}) = 0.247 \text{ bits}$$

$$\text{Gain}(\textit{Temperature}) = 0.029 \text{ bits}$$

$$\text{Gain}(\textit{Humidity}) = 0.152 \text{ bits}$$

$$\text{Gain}(\textit{Windy}) = 0.048 \text{ bits}$$

Continuing to split

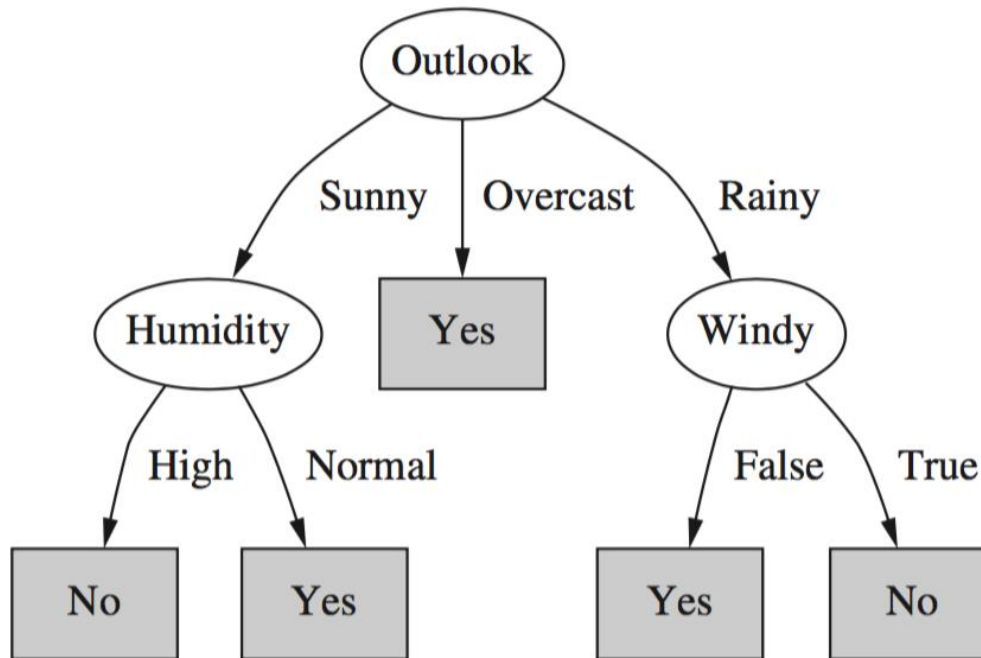


$\text{Gain}(\text{Temperature}) = 0.571 \text{ bits}$

$\text{Gain}(\text{Humidity}) = 0.971 \text{ bits}$

$\text{Gain}(\text{Windy}) = 0.020 \text{ bits}$

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

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

Gain ratio

- **Intrinsic information (split info):** 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

$$\text{SplitInfo}(A) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \log_2 \frac{|D_j|}{|D|}$$

Computing the gain ratio

- Example: intrinsic information of ID code

$$\text{SplitInfo(ID)} = - \sum_{j=1}^v \frac{1}{14} \log_2 \frac{1}{14}$$

- 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:	0.247
0.940–0.693	
Split info:	1.577
info([5,4,5])	
Gain ratio:	0.156
0.247/1.577	

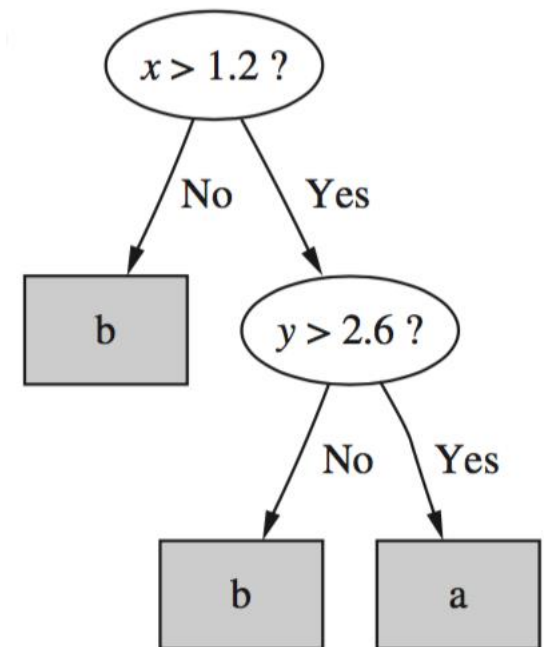
All gain ratios for the weather data

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

“Outlook” is still the chosen attribute to split on

Week 7

- Supervised learning
 - Decision trees
 - Decision rules
 - Ethical thinking:
fairness in classification



ID3 Decision Tree Classifier

- **Representation**
 - Decision tree
- **Objective function for training**
 - Information gain of each split
- **Search algorithm**
 - Recursively construct tree, greedily selecting attributes to split on via information gain

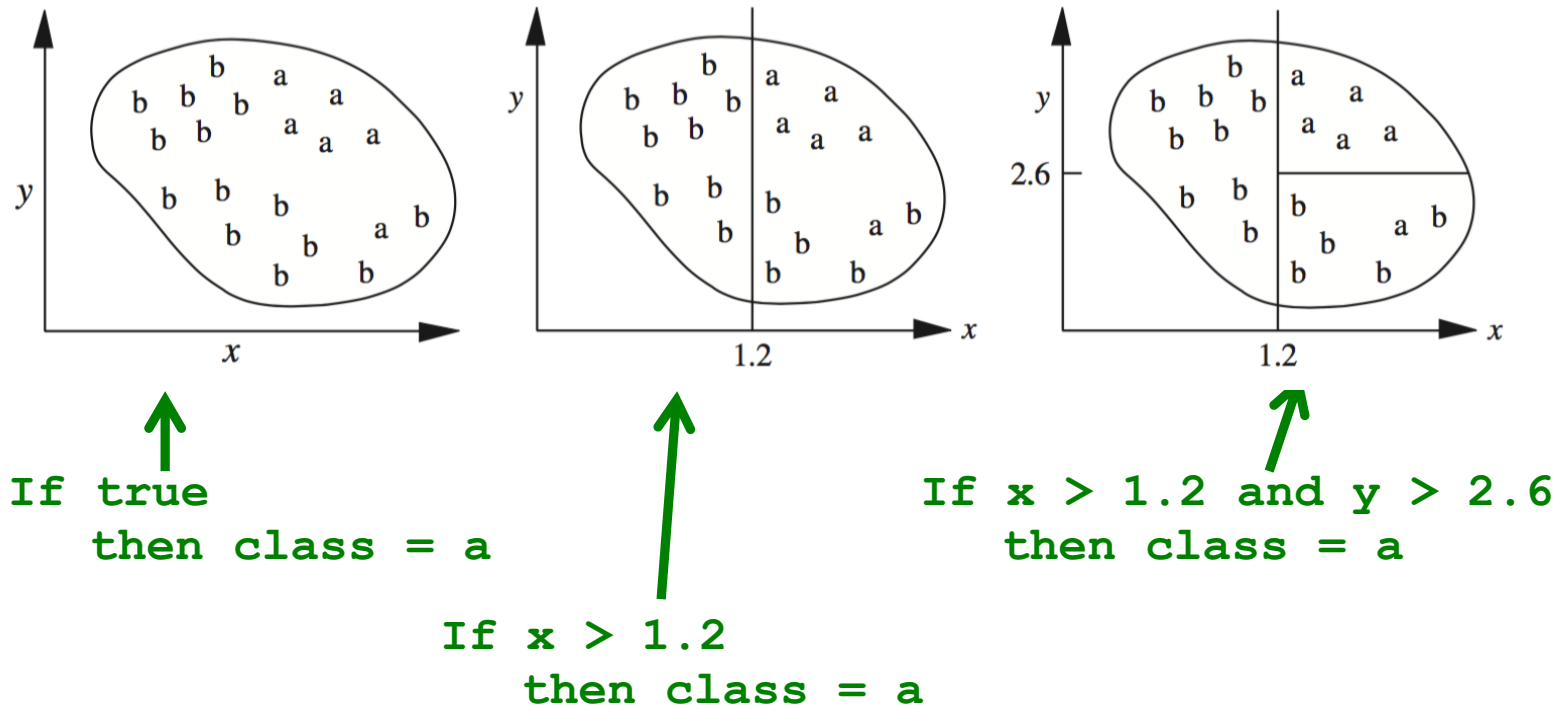
Covering algorithms

- We've seen how to learn decision trees.
What about decision rules?
- We can always convert a decision tree into a rule set
 - However, the rule set will be overly complex
- **How do we learn a smaller, more interpretable set of rules?**

Covering algorithms

- Instead, we can generate rule set directly
 - One approach: for each class in turn, find rule set that covers all instances in it
(excluding instances not in the class)
- Called a *covering* approach:
 - At each stage of the algorithm, a rule is identified that “covers” some of the instances

Example: generating a rule



- Possible rule set for class "b":

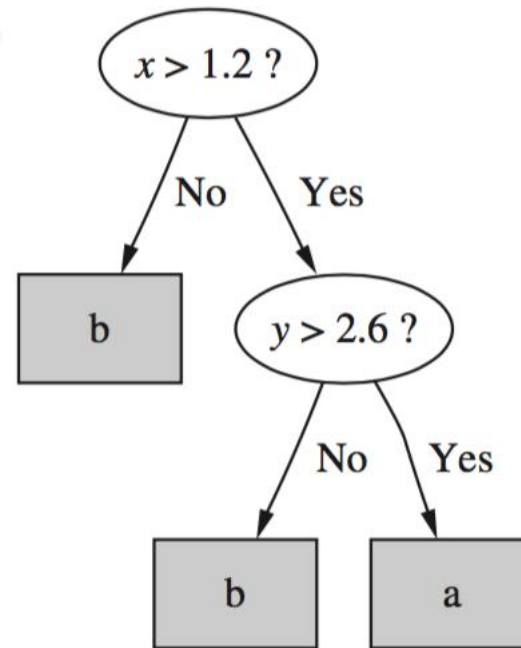
If $x \leq 1.2$ then class = b

If $x > 1.2$ and $y \leq 2.6$ then class = b

- Could add more rules, get "perfect" rule set

Rules vs. trees

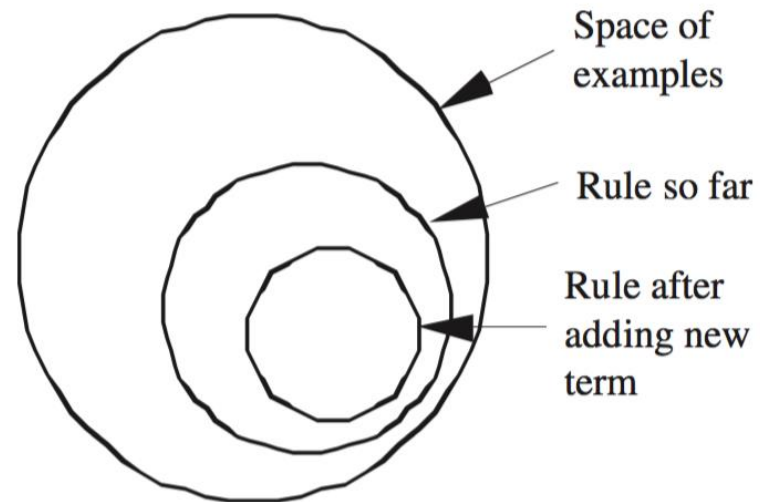
- Corresponding decision tree:
(produces exactly the same predictions)



- But: rule sets *can* be more perspicuous when decision trees suffer from replicated subtrees
- Also: in multiclass situations, covering algorithm concentrates on one class at a time whereas decision tree learner takes all classes into account

Simple covering algorithm

- Basic idea: **generate 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 rule's coverage:



Example: contact lens data

- Rule we seek:

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

- Possible tests:

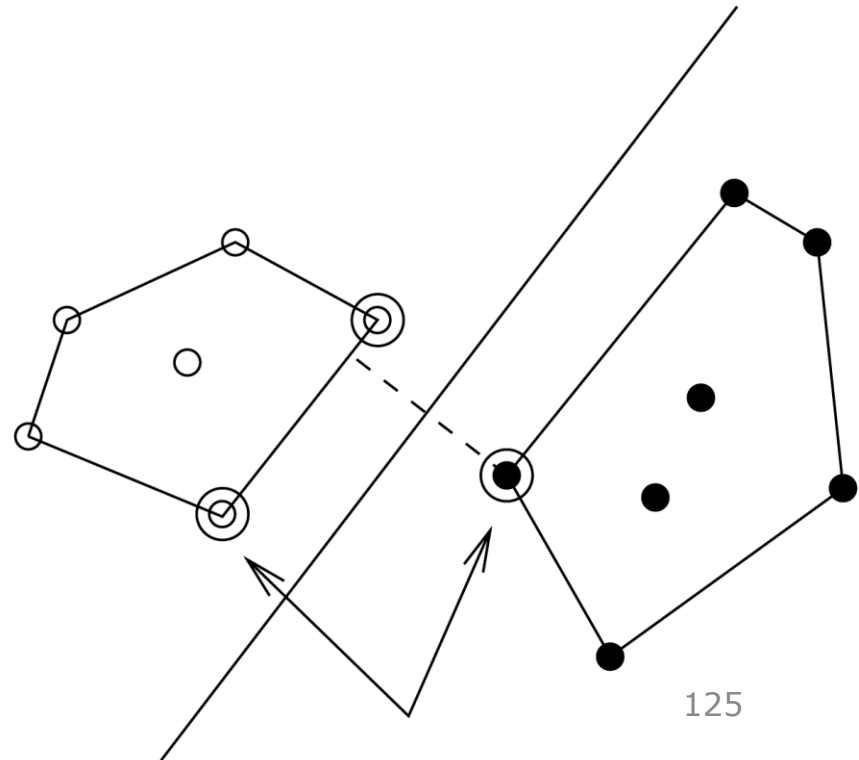
Age = Young	2/8
Age = Pre-presbyopic	1/8
Age = Presbyopic	1/8
Spectacle prescription = Myope	3/12
Spectacle prescription = Hypermetrope	1/12
Astigmatism = no	0/12
Astigmatism = yes	4/12
Tear production rate = Reduced	0/12
Tear production rate = Normal	4/12

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
 - Select test that maximizes the ratio p/t
- We are finished when $p/t = 1$ or the set of instances cannot be split any further

Week 9

- Supervised learning (continued)
 - Naive Bayes
 - logistic regression
 - support vector machines.



Learning outcomes

By the end of the lesson, you should be able to:

- **Perform** the steps of the training algorithms for the **naïve Bayes classifier**, given a small dataset
- **Explain** the assumptions made by naïve Bayes models
- **Discuss** how **logistic regression** and **support vector machines (SVMs)** extend the simple linear regression model into powerful techniques for classification

Bayes' rule

- **Our goal:**

To infer the probability of a hypothesis H , given evidence E .

This probability distribution is called the **posterior distribution**, $Pr(H|E)$

- **We have:**

Prior beliefs about H , encoded as a probability distribution,
the **prior distribution** $Pr(H)$

A model for how the data were generated, given the hypothesis, the
likelihood $Pr(E|H)$

Bayes' rule

- Bayes' rule allows us to **combine observed evidence and prior beliefs** to obtain the posterior,

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

$$Pr(H|E) = \frac{Pr(E|H)Pr(H)}{Pr(E)}$$

A normalizing constant

Deconstructing Bayes' rule

$$Pr(H|E) = \frac{Pr(H, E)}{Pr(E)}$$

Definition of conditional probability

$$Pr(H, E) = Pr(E|H)Pr(H)$$

Product rule

$$Pr(H|E) = \frac{Pr(E|H)Pr(H)}{Pr(E)}$$

Plug in equation for joint

$$Pr(E) = \sum_H Pr(E, H)$$

Sum rule

Example of Bayes Theorem

■ Given:

- A doctor knows that meningitis causes stiff neck 50% of the time
- Prior probability of any patient having meningitis is $1/50,000$
- Prior probability of any patient having stiff neck is $1/20$

- ## ■ If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M | S) = \frac{P(S | M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Bayesian Classifiers

- Consider each attribute and class label as random variables
- Given a record with attributes (A_1, A_2, \dots, A_n)
 - Goal is to predict class C
 - Specifically, we want to find the value of C that maximizes $P(C | A_1, A_2, \dots, A_n)$
- Can we estimate $P(C | A_1, A_2, \dots, A_n)$ directly from data?

Bayesian Classifiers

- Approach:

- compute the posterior probability $P(C \mid A_1, A_2, \dots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \dots A_n) = \frac{P(A_1 A_2 \dots A_n \mid C) P(C)}{P(A_1 A_2 \dots A_n)}$$

- Choose value of C that maximizes $P(C \mid A_1, A_2, \dots, A_n)$
- Equivalent to choosing value of C that maximizes $P(A_1, A_2, \dots, A_n \mid C) P(C)$

- How to estimate $P(A_1, A_2, \dots, A_n \mid C)$?

Logistic Regression

- **Representation**
 - Linear model. Logistic function maps a linear function to probabilities.
- **Objective function for training**
 - Log probability of class labels under the model, $\log \Pr(C|A)$.
- **Search algorithm**
 - Iteratively reweighted least squares, or gradient-based methods

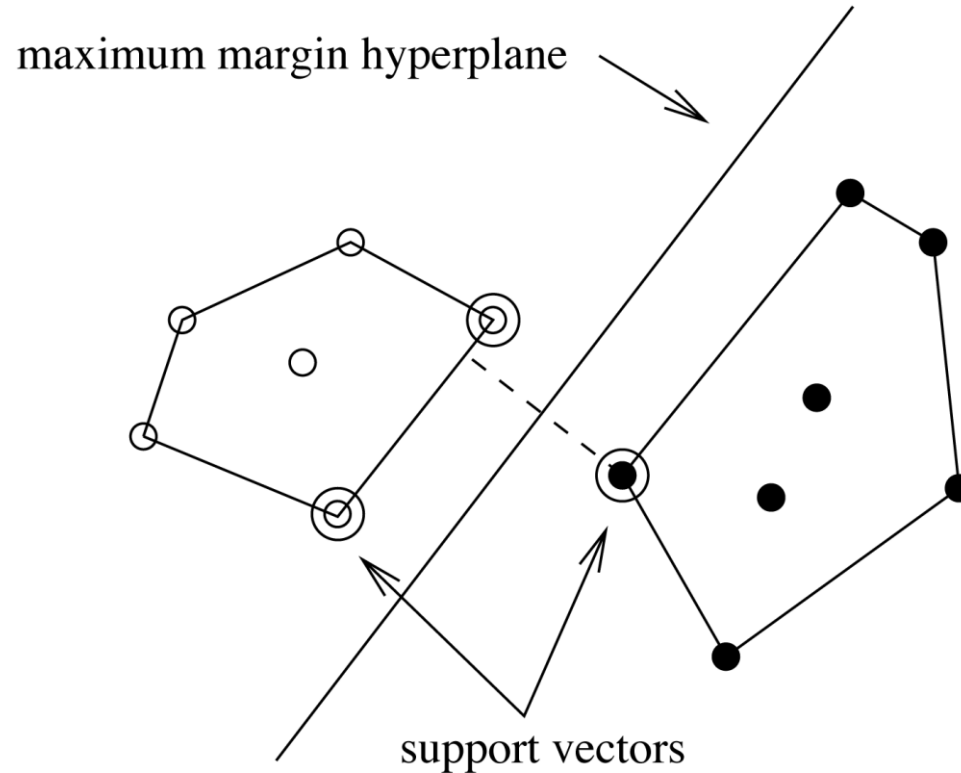
Relationship between naïve Bayes and logistic regression

- Both naïve Bayes and logistic regression are trained to maximize the log-likelihood
- Naïve Bayes: a **generative** classifier
 - Models the attributes as well as the class
 - Maximize $\log P(C,A) = \log P(A|C) + \log P(C) = \log P(C|A) + \log P(A)$
- Logistic regression: A **discriminative** classifier
 - Models the class given the attributes, but not attributes
 - Maximize $\log P(C|A)$
- Naïve Bayes, trained to maximize $\log P(C|A)$, is equivalent to logistic regression (Both with Gaussian assumption, or discrete data. **Proof on bonus slide at end**)
- Naïve Bayes and logistic regression are a “generative-discriminative pair”

Support vector machines

- **Support vector machines** are algorithms for learning linear classifiers
- Resilient to overfitting because they learn a particular linear decision boundary:
 - The **maximum margin hyperplane**
- They can also **learn non-linear classifiers**, using a certain “trick”
 - Use a mathematical trick to avoid creating “pseudo-attributes”
 - The nonlinear space is created implicitly

The maximum margin hyperplane



- The instances closest to the maximum margin hyperplane are called *support vectors*

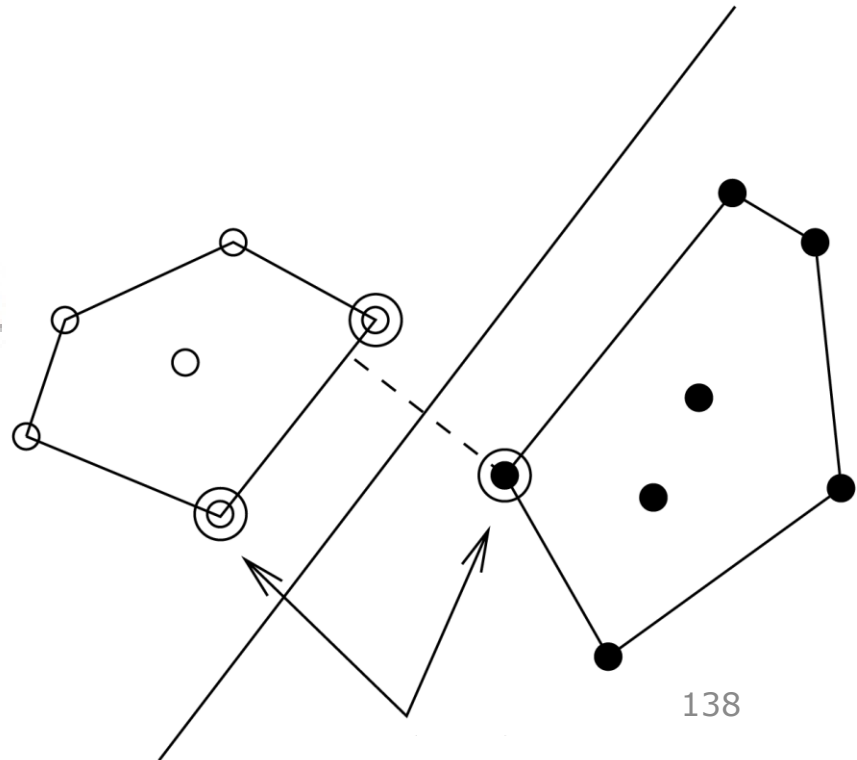
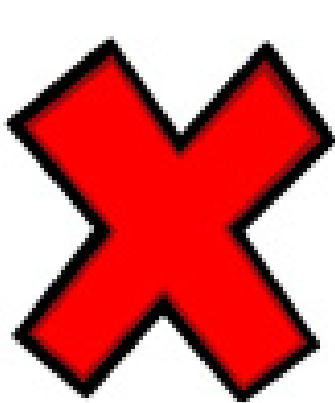
Support Vector Machines (SVMs)

- **Representation**
 - Linear model. The “kernel trick” implicitly maps instances to higher-dimensional spaces, leading to non-linear decision boundaries
- **Objective function for training**
 - The margin (distance from hyperplane to closest instances for each class)
- **Search algorithm**
 - Quadratic optimization, often solved by special-purpose algorithms
- **Support Vector Machine (SVM) Tutorial:**
 - Learning SVMs from examples

Week 10

■ Evaluation of supervised learning

- Hold-out method
- Cross validation
- ROC curves



Learning outcomes

By the end of the lesson, you should be able to:

- **Explain** why classification accuracy on the training set may not be a good indicator of a classifier's performance
- **Describe** the steps of several evaluation methodologies for classifiers
- **Outline** valid strategies for hyperparameter selection which avoid “peeking” at the test data
- **Select** appropriate experimental methodologies when evaluating machine learning methods

Training and testing

- How predictive is the model we have learned?
- Natural performance measure for classification problems: **error rate**
 - *Success*: instance's class is predicted correctly
 - *Error*: instance's class is predicted incorrectly
- **Classification accuracy**: Percent of the whole set of instances that the classifier got right
- **Error rate**: proportion of errors made over the whole set of instances

Training and testing

- ***Resubstitution error (a.k.a. training error):***
error rate obtained by evaluating model on training data
- This measures the model's performance on data that it got to see ahead of time.
- When it is deployed, it won't get to see the data that it has to predict on!

Evaluation: the key to success

- Error on the training data is ***not*** necessarily a good indicator of performance on future data
 - Otherwise 1-NN would be the optimal classifier!
 - Our model is potentially fitting to noisy, spurious patterns that occur in the training set, possibly by chance.
 - There is no guarantee that the error rate on the training data will correspond to its error rate “in the wild”

Resubstitution error is (hopelessly) optimistic!

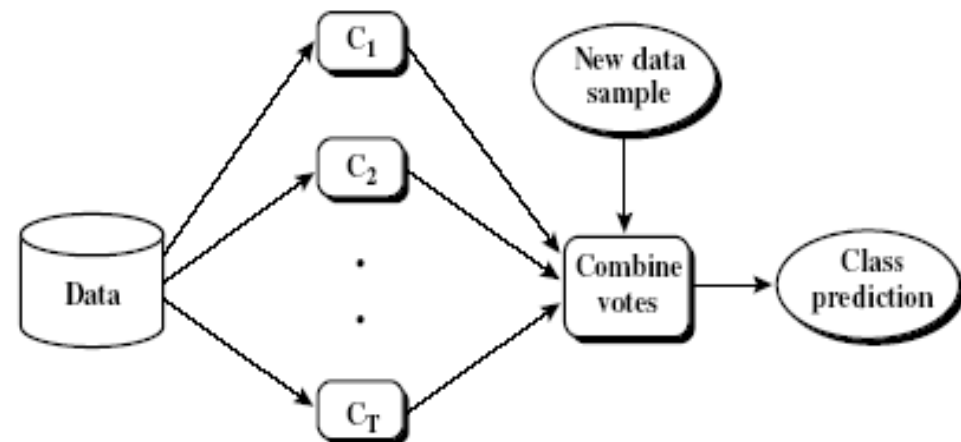
Training and testing

- **Test set**: independent instances that have played no part in formation of classifier
 - Assumption: both training data and test data are representative samples of the underlying problem
- Be careful to avoid a situation where test and training data could differ in nature
 - Example: classifiers built using customer data from two different towns *A* and *B*
 - To estimate performance of classifier from town *A* in completely new town, test it on data from *B*

Week 11

■ Ensemble methods

- Bagging
- Boosting
- Random forests
- Stacking

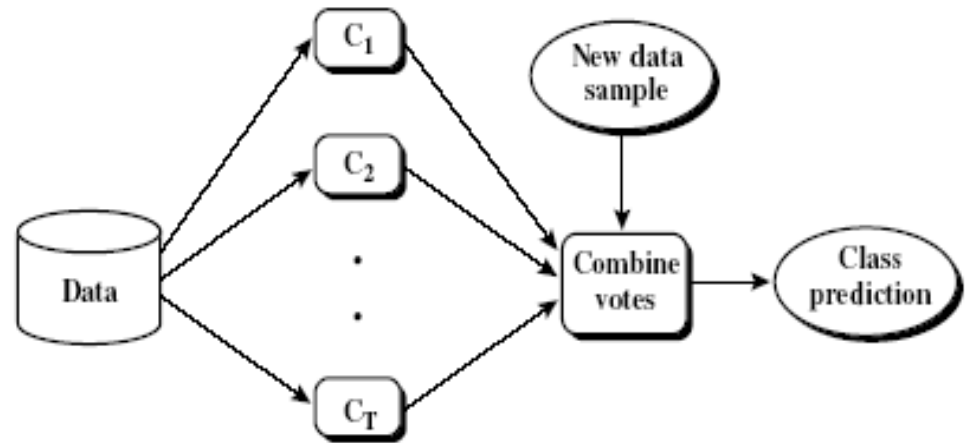


Learning outcomes

By the end of the lesson, you should be able to:

- **Describe**, and **compare and contrast** the main ensemble methods:
bagging, boosting, random forests, stacking
- **Discuss** why these methods may improve classification performance
- **Explain**, at an intuitive level, the **bias-variance trade-off**

Ensemble Methods



- **Ensemble methods**
 - Use a combination of models to increase accuracy
 - Combine a series of k learned models, M_1, M_2, \dots, M_k , with the aim of creating an improved model M^*
- Popular ensemble methods
 - **Bagging**: averaging the prediction over a collection of classifiers
 - **Boosting**: weighted vote with a collection of classifiers
 - **Stacking**: combining a set of heterogeneous classifiers

Combining multiple models

- Basic idea:
build different “experts”, let them vote
- Advantage:
 - often improves predictive performance
- Disadvantage:
 - usually produces output that is very hard to analyze
 - but: there are approaches that aim to produce a single comprehensible structure

Bagging

- **Combining predictions by voting/averaging**
 - Each model receives **equal weight**
- *“Idealized” version:*
 - Sample several training sets of size n
(instead of just having one training set of size n)
 - Build a classifier for each training set
 - Combine the classifiers’ predictions
- If learning scheme is *unstable*,
bagging almost always improves performance
 - **Unstable learner:** small change in training data can make big change in model (e.g., when learning decision trees)

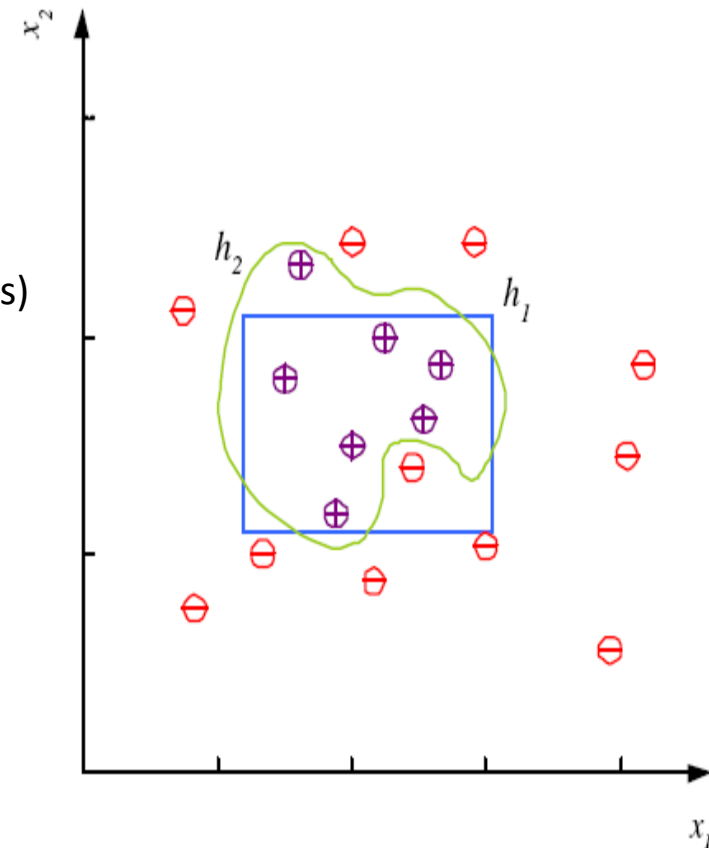
Bias-variance decomposition

- The ***bias-variance decomposition*** is used to analyze how much restriction to a single training set affects performance
- Assume we have the idealized ensemble classifier discussed on the previous slide
- We can decompose the expected error of any individual ensemble member as follows:
 - *Bias* = expected error of the ensemble classifier on new data
 - *Variance* = component of the expected error due to the particular training set being used to build our classifier
 - Total expected error = bias + variance

Noise and Model Complexity

Use the simpler one because

- Simpler to use (lower computational complexity)
 - Easier to check if a point is inside/outside a rectangle
- Easier to train (lower space complexity)
 - Fewer parameters
 - More bias (rigid: more likely will not change its hypothesis)
 - Less variance (less ability of learner to change its hypothesis)
 - May fail if indeed the underlying class is not that simple
- Easier to explain (more interpretable)
 - Defining intervals on the two attributes
- Generalizes better (lower variance - Occam's razor)
 - Better discriminator than wiggly shape in presence of mislabel/noise though with higher error
 - Simpler explanations are more plausible



Noise

- Sources
 - Incorrect feature values
 - Imprecision in recoding the input attributes
 - Incorrect class labels
 - teacher noise
 - Hidden or latent features
 - additional attributes
- Impact
 - Overfitting: Trying too hard to fit h to the noise

Recap: “Idealized version” of Bagging

- **Combining predictions by voting/averaging**
 - Each model receives **equal weight**
- *“Idealized” version:*
 - Sample several training sets of size n
(instead of just having one training set of size n)
 - Build a classifier for each training set
 - Combine the classifiers’ predictions

More on bagging

- The idealized version of bagging improves performance because it eliminates the *variance* component of the error
- **Problem:** we only have one dataset!
 - We can't actually implement "idealized bagging" in practice

More on bagging

- **Solution:** generate new datasets of size n by **sampling from the original dataset *with replacement***
- This is what ***bagging*** algorithm does
- Even though the datasets are all dependent, bagging often reduces variance, and, thus, error
 - Can be applied to numeric prediction and classification
 - Can help a lot if the data is noisy
 - Usually, the more classifiers the better, with diminishing returns

Boosting

- Bagging can easily be parallelized because ensemble members are created independently
- Boosting is an alternative approach which is trained sequentially instead of in parallel
 - Later classifiers in the ensemble are informed by earlier ones
- Also uses voting/averaging
 - But: weights models according to performance

Boosting

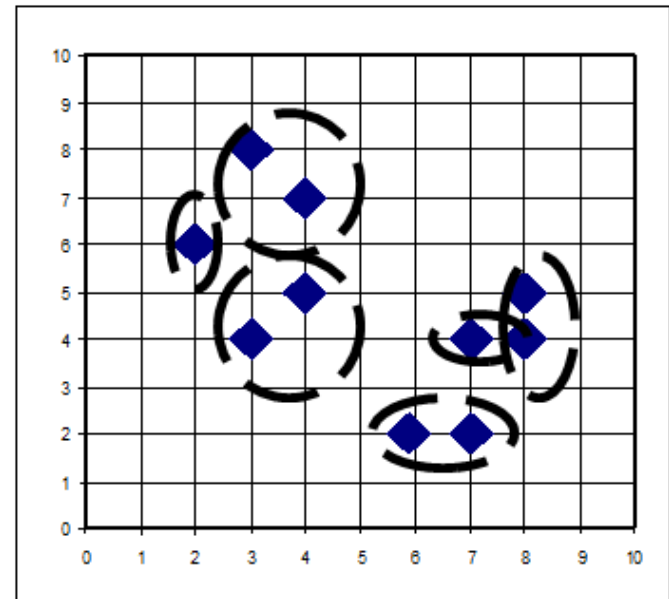
- Boosting is Iterative: new models are influenced by performance of previously built ones
 - Encourage new model to become an “expert” for instances misclassified by earlier models
 - Intuitive justification: models should be **experts** that **complement each other**
- Many variants of boosting exist. We will focus on AdaBoost.M1

Stacking

- Question: how to build a *heterogeneous* ensemble consisting of different types of models (e.g., decision tree and neural network)
 - Problem: models can be vastly different in accuracy
- Idea: to combine predictions of base learners, do *not* just vote, instead, use *meta learner*
 - In stacking, the base learners are also called *level-0 models*
 - Meta learner is called *level-1 model*
 - Predictions of base learners are input to meta learner
- Base learners are usually different learning schemes

Week 12

- Unsupervised learning
 - Association rule learning
 - K-means
 - Hierarchical clustering

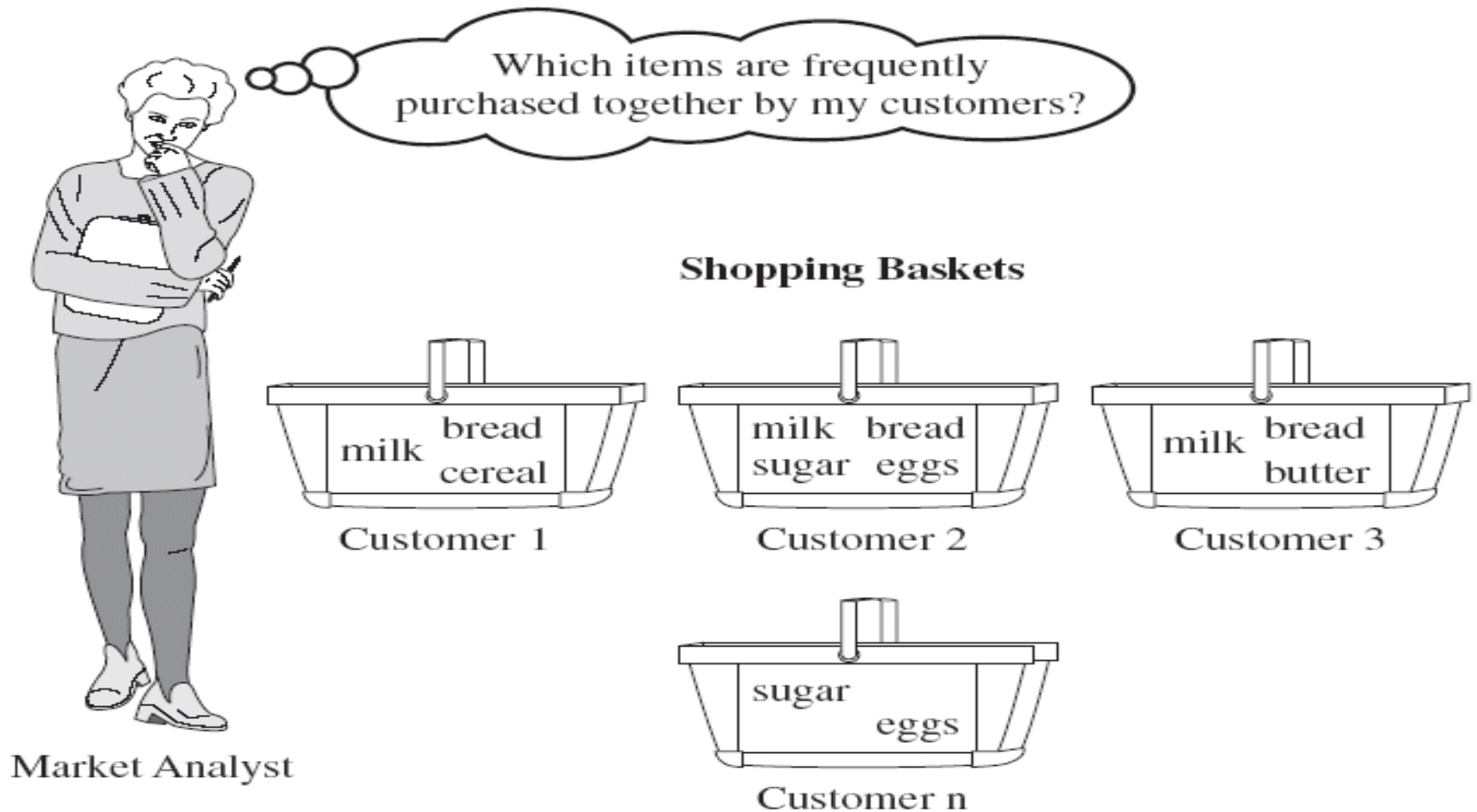


Learning outcomes

By the end of the lesson, you should be able to:

- **Describe** the steps of the Apriori algorithm for association rule mining
- **Implement** the **K-means** algorithm
- **Explain** the steps of the main **hierarchical clustering** algorithms
- **Make appropriate algorithmic choices** for clustering

Frequent Pattern Analysis and Association Rule Mining



Frequent Pattern Analysis and Association Rule Mining

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of **frequent itemsets** and **association rule mining**
- Motivation: Finding inherent regularities in data
 - What products were often purchased together?— Beer and diapers?!
 - What are the subsequent purchases after buying a PC?
- Applications
 - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Apriori algorithm for association rules

- **Representation**
 - Association rules
- **Objective function for training**
 - Coverage and accuracy of rules
- **Search algorithm**
 - Construct frequent item sets of increasing size, and of sufficient coverage, based on smaller item sets. From each item set, construct rules of increasing consequent size, and of sufficient accuracy, based on smaller sets of consequents.

What is Clustering?

- Cluster: A collection of data objects
 - similar (or related) to one another within the same group
 - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation*, ...)
 - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- **Unsupervised learning**: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical applications
 - As a **stand-alone tool** to get insight into data distribution
 - As a **preprocessing step** for other algorithms

Clustering

- Clustering techniques apply when there is no class to be predicted: they perform unsupervised learning
- Aim: divide instances into “natural” groups
- Clusters can be:
 - disjoint vs. overlapping
 - deterministic vs. probabilistic
 - flat vs. hierarchical
- We will look at a classic clustering algorithm called *k-means*
- *k-means* clusters are disjoint, deterministic, and flat

Clustering for Data Understanding and Applications

- Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- Information retrieval: document clustering
- Land use: Identification of areas of similar land use in an earth observation database
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earthquake epicenters should be clustered along continent faults
- Climate: understanding earth climate, find patterns of atmospheric and ocean
- Economic Science: market research

Clustering as a Preprocessing Tool (Utility)

- Summarization:
 - Preprocessing for regression, PCA, classification, and association analysis
- Compression:
 - Image processing: vector quantization
- Finding K-nearest Neighbors
 - Localizing search to one or a small number of clusters
- Outlier detection
 - Outliers are often viewed as those “far away” from any cluster

Partitioning Algorithms: Basic Concept

- Partitioning method: Partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where c_i is the centroid or medoid of cluster C_i)

$$E = \sum_{i=1}^k \sum_{p \in C_i} \|p - c_i\|^2$$

- Given k , find a partition of k clusters that optimizes the chosen partitioning criterion
 - Global optimal: exhaustively enumerate all partitions
 - Heuristic methods: *k-means* and *k-medoids* algorithms
 - *k-means* (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
 - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The k -means algorithm

- **Step 1:** Choose k random cluster centers
- **Step 2:** Assign each instance to its closest cluster center based on Euclidean distance
- **Step 3:** Recompute cluster centers by computing the average (aka *centroid*) of the instances pertaining to each cluster
- **Step 4:** If cluster centers have moved, go back to Step 2

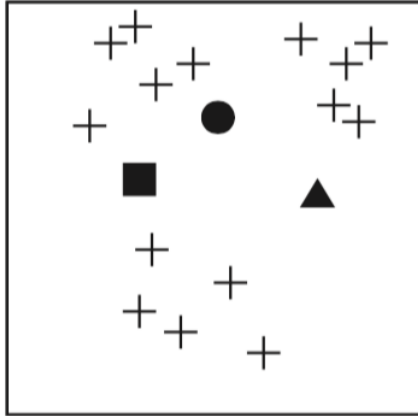
The k -means algorithm

- Equivalent termination criterion: stop when assignment of instances to cluster centers has not changed
- This algorithm minimizes the squared Euclidean distance of the instances from their corresponding cluster centers:

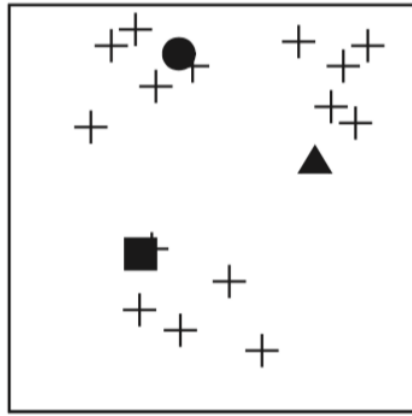
$$E = \sum_{i=1}^k \sum_{p \in C_i} \| p - c_i \|^2$$

The k -means algorithm: example

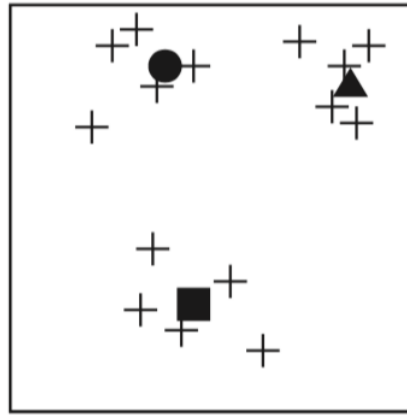
Initial step



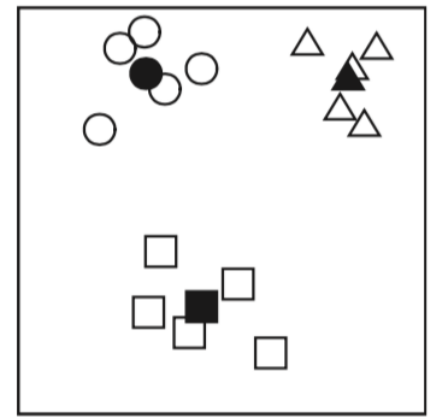
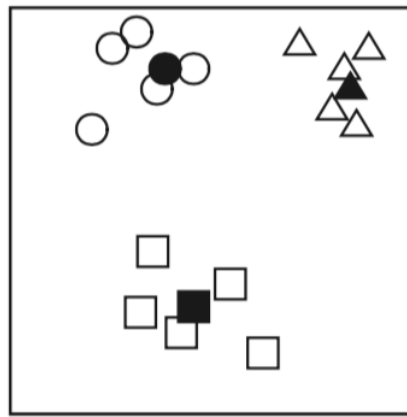
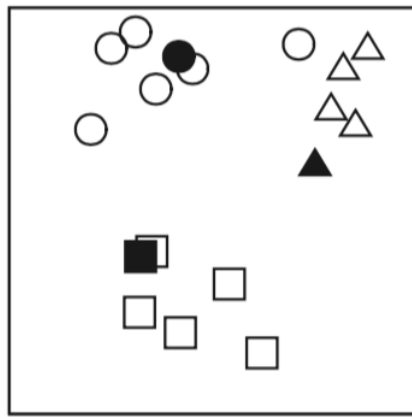
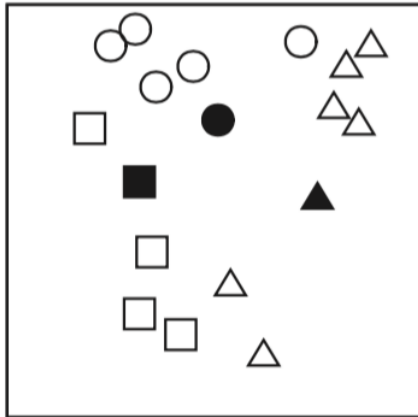
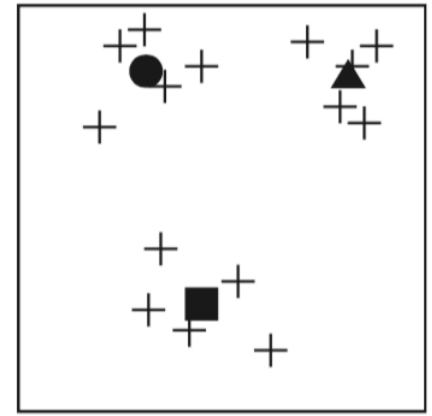
Step 1



Step 2



Final step

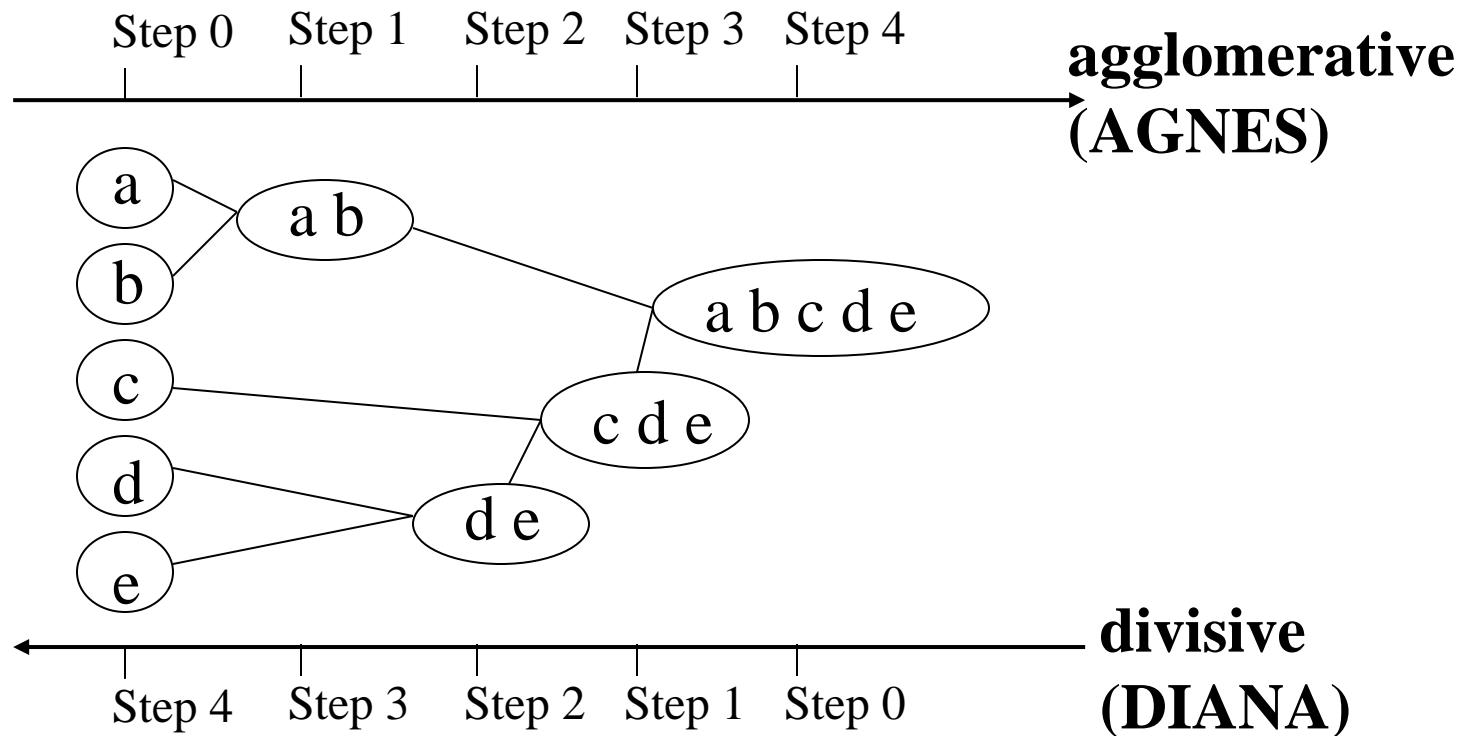


Hierarchical clustering

- Bisecting k -means performs hierarchical clustering in a top-down manner
- Standard hierarchical clustering performs clustering in a bottom-up manner; it performs *agglomerative* clustering:
 - First, make each instance in the dataset into a trivial mini-cluster
 - Then, find the two closest clusters and merge them; repeat
 - Clustering stops when all clusters have been merged into a single cluster

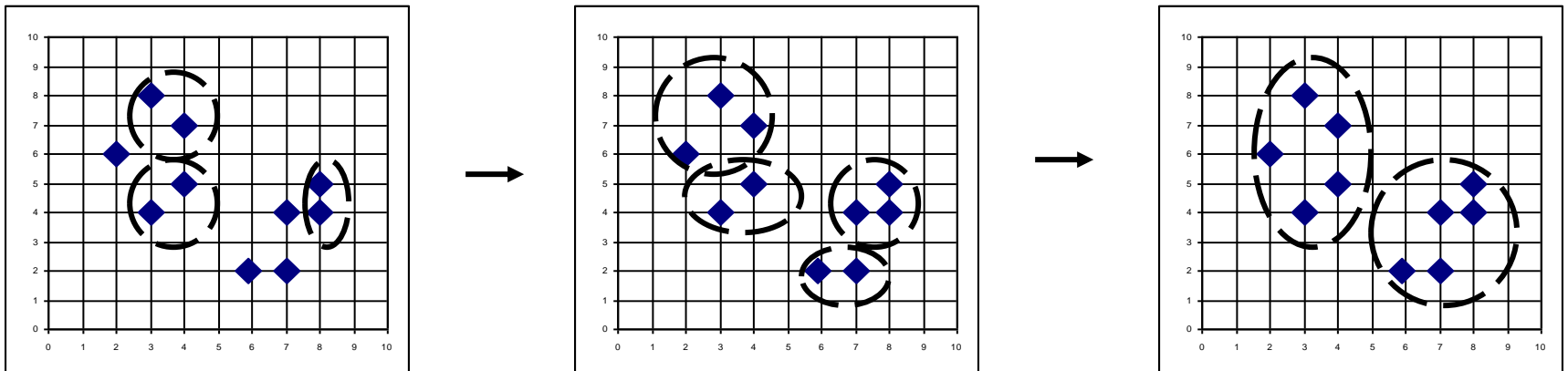
Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters k as an input, but needs a termination condition



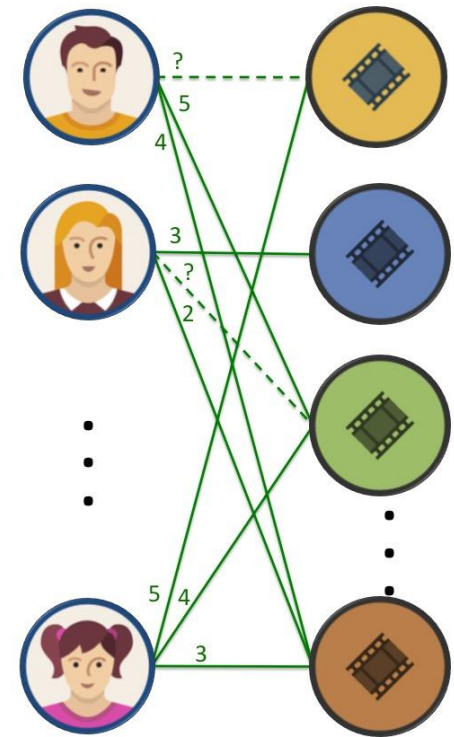
AGNES (Agglomerative Nesting)

- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical packages, e.g., Splus
- Use the **single-link** method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



Week 13

- Recommender systems
 - Content filtering
 - Collaborative filtering



Learning outcomes

By the end of the lesson, you should be able to:

- **Compare and contrast** the main **content-based filtering** and **collaborative filtering methods**
- **Explain** the intuition behind **neighborhood-based collaborative filtering** algorithms
- **Outline** the steps of the popular training algorithms for **matrix factorization collaborative filtering** methods: **stochastic gradient descent**, and **alternating least squares**, and **discuss** their advantages/disadvantages
- **Apply** these methods in real-world scenarios, making sensible choices of methods

Content-Based Filtering

- Try to recommend similar items to what the user has liked in the past
- Extract features based on content of the items (**item profiles**)
- Build feature vectors for users based on content features (**user profiles**)
- Recommend items that are similar to user profile



Collaborative Filtering

- Makes use of **other users**, or **other items**, to predict ratings “**collaboratively**”
- Predictions based on *other ratings*
- **Neighborhood-based** methods
 - Find similar users or similar items, predict that the target rating will be similar
- **Matrix factorization / latent factor** methods
 - **Factorize the ratings matrix**, find **latent vector** representations for users and items based on the factorization

Neighborhood-Based CF Methods

- User-user neighborhood-based CF
 - **Find K-nearest neighbor users** according to ratings
 - Represent **users** by their **rows of the ratings matrix**
 - Only consider *users who have rated the item s* in question
 - Aggregate their ratings for an item to predict rating

$$f(u, s) = \frac{1}{\sum_{u' \in neighbors(u)} sim(\mathbf{u}, \mathbf{u}')} \sum_{u' \in neighbors(u)} sim(\mathbf{u}, \mathbf{u}') f(u', s)$$

$$sim(\mathbf{u}, \mathbf{u}') = \frac{\mathbf{u} \cdot \mathbf{u}'}{\|\mathbf{u}\| \|\mathbf{u}'\|}$$

- Other aggregation functions possible

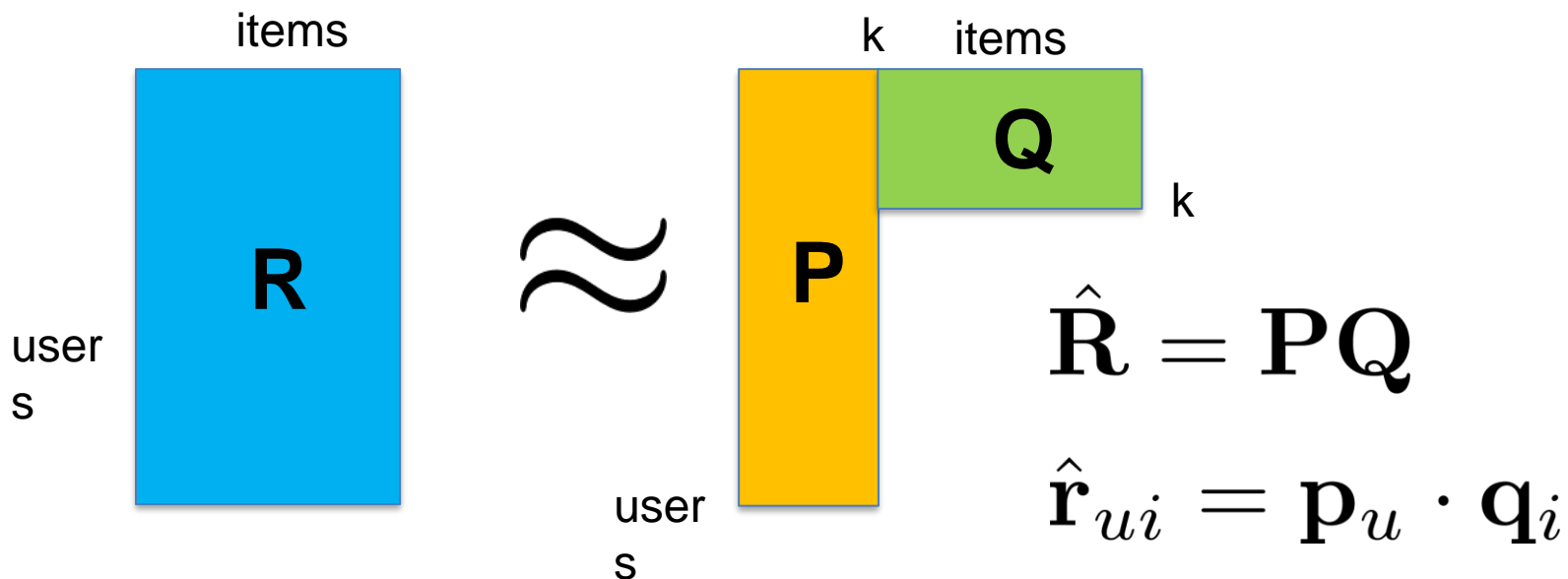
Neighborhood-Based CF Methods

- Item-item neighborhood-based CF
 - Find **K-nearest neighbor items** according to ratings
 - represent **items** by their **columns of the ratings matrix**
 - Only consider *items rated by user u in question*
 - Aggregate their ratings for an item to predict rating

$$f(u, s) = \frac{1}{\sum_{s' \in neighbors(s)} sim(s, s')} \sum_{s' \in neighbors(s)} sim(s, s') f(u, s') \quad sim(s, s') = \frac{s \cdot s'}{\|s\| \|s'\|}$$

Matrix Factorization CF (Latent Factor Models)

- Use a **low-rank factorization model** to impute the unobserved ratings
- This represents users and items with vector-valued representations: “**latent factors**”



Matrix Factorization CF

- Objective function: regularized squared error

$$\text{obj}(\mathbf{Q}, \mathbf{P}) = \sum_{(u,i) \in \text{training}} (r_{ui} - \mathbf{p}_u \cdot \mathbf{q}_i)^2 + \lambda \left(\sum_i \|\mathbf{q}_i\|^2 + \sum_u \|\mathbf{p}_u\|^2 \right)$$

Squared error

Regularization
penalty

Controls trade-off between squared
error and regularization penalty

Generalizes SVD to matrices with missing entries. SVD also minimizes squared error, but assumes all entries of matrix are known.

Learning algorithms: Stochastic Gradient Descent

Monday, December 11, 2006

Netflix Update: Try This at Home

- This approach was popularized for CF by a **blog post**!
- By Simon Funk (a pseudonym)
- This blog post is still *recommended reading*
- Author was 3rd on Netflix Prize leaderboard.
He made a big impact by sharing his methods



[Followup to [this](#)]

Ok, so here's where I tell all about how I (now we) got to be tied for third place on the [netflix prize](#). And I don't mean a sordid tale of computing in the jungle, but rather the actual math and methods. So yes, after reading this post, you too should be able to rank in the top ten or so.

Ur... yesterday's top ten anyway.

<http://sifter.org/~simon/journal/20061211.html>

Learning algorithms:

Stochastic Gradient Descent

- Compute the **gradient** of the error with respect to **one rating**
 - take a step downhill (opposite direction of the gradient).
 - Loop over all ratings in the training set, and repeat.
- Prediction error:
$$e_{ui} \stackrel{def}{=} r_{ui} - q_i^T p_u$$
- SGD updates:
$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$

SGD: Pros and Cons

- **Pros:**
 - Simple to implement
 - Fast execution time
- **Cons:**
 - Does not exploit parallelism
 - Slower for *dense matrices*, e.g. implicit feedback (a randomized strategy could mitigate this)

Learning Algorithms: Alternating Least-Squares

- Until converged:
 - Fix \mathbf{P} , solve for \mathbf{Q}
 - Fix \mathbf{Q} , solve for \mathbf{P}
- These are **least squares** problems similar to linear regression. Can be solved in closed form (requires a matrix inverse for each value)
- Each iteration can be **performed in parallel**. Useful when *target matrix is dense*, e.g. with implicit feedback

Which Algorithm Would You Use? SGD or ALS?

1. You are building a recommender system for Amazon.com which recommends products leveraging not only ratings, but also which products were viewed or mouse-overed
2. You aim to recommend high-end jewellery items based on feedback after purchases. You only have one server machine

Scenario Question: Recommending Textbooks on Amazon

- Suppose you work for Amazon, and are designing a recommender system specifically for **textbooks**. You will serve all of their customers who are interested in purchasing such books (**millions of users** and **hundreds of thousands of items**), and will have access to all of their computational resources and data.
 - What **methods** and **algorithms** will you use?
 - How will you **scale up** to this scenario?
 - You have access to **ratings, content, temporal information**, and many kinds of **implicit feedback**. How will you leverage these?

1-16 of 704,422 results for "textbooks"

Sort by Rel

☐ **prime** | FREE One-Day

Get FREE One-Day Shipping on qualifying orders over \$35

Show results for

AmazonFresh

☐ **fresh**

Books

- Schools & Teaching
- Medical & Surgical Nursing
- General Surgery
- Astronomy
- Academic Development
- Counseling
- [See more](#)

Kindle Store

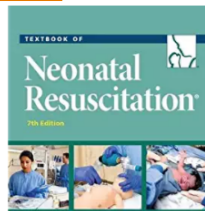
- Pediatrics
- Perinatology & Neonatology
- Kindle eBooks
- Biology
- Human Resources & Personnel



Textbooks – Rent, Buy, Sell

[Textbooks](#) | [Textbook Rentals](#) | [eTextbooks](#) | [eTextbook Rentals](#) | [Sell Your Books](#)

Best Seller



Textbook of Neonatal Resuscitation May 6, 2016
by American Academy of Pediatrics and American Heart Association

Paperback

\$47⁶⁷ ~~\$69.95~~

FREE Shipping on eligible orders

Only 20 left in stock - order soon.

More Buying Choices

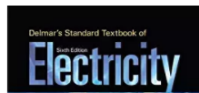
\$31.80 (62 used & new offers)

Kindle Edition

\$54⁶⁵

★★★★★ ▾ 117

Trade in yours for an Amazon Gift Card up to \$24.78



Delmar's Standard Textbook of Electricity Jan 1, 2015
by Stephen Herman

Hardcover

\$46⁵⁵ to rent **prime** | FREE One-Day

★★★★★ ▾ 57

Trade in yours for an Amazon Gift Card up to \$61.00

Scenario Question: Finding pets

- You are developing a smartphone app called **PetFinder** which uses data mining to help to reunite pet owners with their pets that have gone missing. Users can upload photos of their missing pets, demographic data such as gender and breed, plain-text descriptions, approximate locations, and so on, as well as similar information for stray pets that they encounter on the streets. You can also access any other publicly available data such as street maps. Your task is to design data mining analyses and/or data-driven artificial intelligence (AI) systems to help the app succeed in its pet-finding mission.

Think through your overall strategy before you start to answer. Design a coherent overall approach across your answers, rather than separate unrelated answers. Be sure to clearly justify your choices.

Building blocks to design a data-driven AI system

Design a data mining analysis which uses a **supervised learning** method. For your method, **specify** and **explain** each of the following items:

- (a) The type of analysis you perform (e.g., classification, regression, recommender system etc).
- (b) How will the output or internal representation of your method be informative?
- (c) What features/attributes will you collect and use?
- (d) What preprocessing steps will you take, and why will these be expected to be beneficial?
- (e) Which machine learning algorithm(s) will you use? Outline the steps of the algorithms.
- (f) How will you ensure that the algorithm will scale up to handle a dataset of this size?
- (g) Why are the algorithm(s) you have selected more appropriate than other alternatives?
- (h) How will you select any hyper-parameter values for your method, such as regularization parameters, number of clusters, etc.?
- (i) How will you evaluate your method? Describe your evaluation procedure in detail

Design a data-driven AI system

- Design another analysis for the scenario in Question which uses an unsupervised learning method (e.g., association rule mining, clustering, ...).

Week 14

Exam review

1. Multiple Choice Questions
 2. Short & Medium Type Questions
 3. Scenario Type Problems
-
- When: Tuesday (5/04) 7:10 pm – 8:10 pm (2 hours)
 - Where: Online In Class

Conclusion

Please take a few minutes to complete the online course evaluations.

Thank you for taking IS 733: Data Mining.

Good Luck!