

---

# COMP9318: Data Warehousing and Data Mining

— L1: Introduction —

# Chapter 1. Introduction

---

- Motivation: Why data mining?
- What is data mining?
- Data Mining: On what kind of data?
- Data mining functionality
- Are all the patterns interesting?
- Classification of data mining systems
- Major issues in data mining

# *Necessity Is the Mother of Invention*

---

- Data explosion problem
  - Automated data collection tools and mature database technology lead to tremendous amounts of data accumulated and/or to be analyzed in databases, data warehouses, and other information repositories
- We are drowning in data, but starving for knowledge!

***Who could be expected to digest millions of records, each having tens or hundreds of fields?***

- Solution: Data warehousing and data mining
  - Data warehousing and on-line analytical processing
  - Mining interesting knowledge (rules, regularities, patterns, constraints) from data in large databases

# Evolution of Database Technology

---

- 1960s:
  - Data collection, database creation, IMS and network DBMS
- 1970s:
  - Relational data model, relational DBMS implementation
- 1980s:
  - RDBMS, advanced data models (extended-relational, OO, deductive, etc.)
  - Application-oriented DBMS (spatial, scientific, engineering, etc.)
- 1990s:
  - Data mining, data warehousing, multimedia databases, and Web databases
- 2000s
  - Stream data management and mining
  - Data mining with a variety of applications
  - Web technology and global information systems



# What Is Data Mining?

- Data mining (knowledge discovery from data)
  - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) patterns or knowledge from **huge amount** of data
  - Data mining: a misnomer?
- Alternative names
  - Knowledge discovery (mining) in databases (KDD), knowledge extraction, data/pattern analysis, data archeology, data dredging, information harvesting, business intelligence, etc.
- Watch out: Is everything “data mining”?
  - (Deductive) query processing.
  - Expert systems or small ML/statistical programs



# Why Data Mining?—Potential Applications

---

- Data analysis and decision support
  - Market analysis and management
    - Target marketing, customer relationship management (CRM), market basket analysis, cross selling, market segmentation
  - Risk analysis and management
    - Forecasting, customer retention, improved underwriting, quality control, competitive analysis
  - Fraud detection and detection of unusual patterns (outliers)
- Other Applications
  - Text mining (news group, email, documents) and Web mining
  - Stream data mining
  - DNA and bio-data analysis

# Market Analysis and Management

---

- Where does the data come from?
  - Credit card transactions, loyalty cards, discount coupons, customer complaint calls, plus (public) lifestyle studies
- Target marketing
  - Find clusters of “model” customers who share the same characteristics: interest, income level, spending habits, etc.
  - Determine customer purchasing patterns over time
- Cross-market analysis
  - Associations/co-relations between product sales, & prediction based on such association
- Customer profiling
  - What types of customers buy what products (clustering or classification)
- Customer requirement analysis
  - identifying the best products for different customers
  - predict what factors will attract new customers
- Provision of summary information
  - multidimensional summary reports
  - statistical summary information (data central tendency and variation)

# Corporate Analysis & Risk Management

---

- Finance planning and asset evaluation
  - cash flow analysis and prediction
  - contingent claim analysis to evaluate assets
  - cross-sectional and time series analysis (financial-ratio, trend analysis, etc.)
- Resource planning
  - summarize and compare the resources and spending
- Competition
  - monitor competitors and market directions
  - group customers into classes and a class-based pricing procedure
  - set pricing strategy in a highly competitive market

# Fraud Detection & Mining Unusual Patterns

---

- Approaches: Clustering & model construction for frauds, outlier analysis
- Applications: Health care, retail, credit card service, telecomm.
  - Auto insurance: ring of collisions
  - Money laundering: suspicious monetary transactions
  - Medical insurance
    - Professional patients, ring of doctors, and ring of references
    - Unnecessary or correlated screening tests
  - Telecommunications: phone-call fraud
    - Phone call model: destination of the call, duration, time of day or week. Analyze patterns that deviate from an expected norm
  - Retail industry
    - Analysts estimate that 38% of retail shrink is due to dishonest employees
  - Anti-terrorism

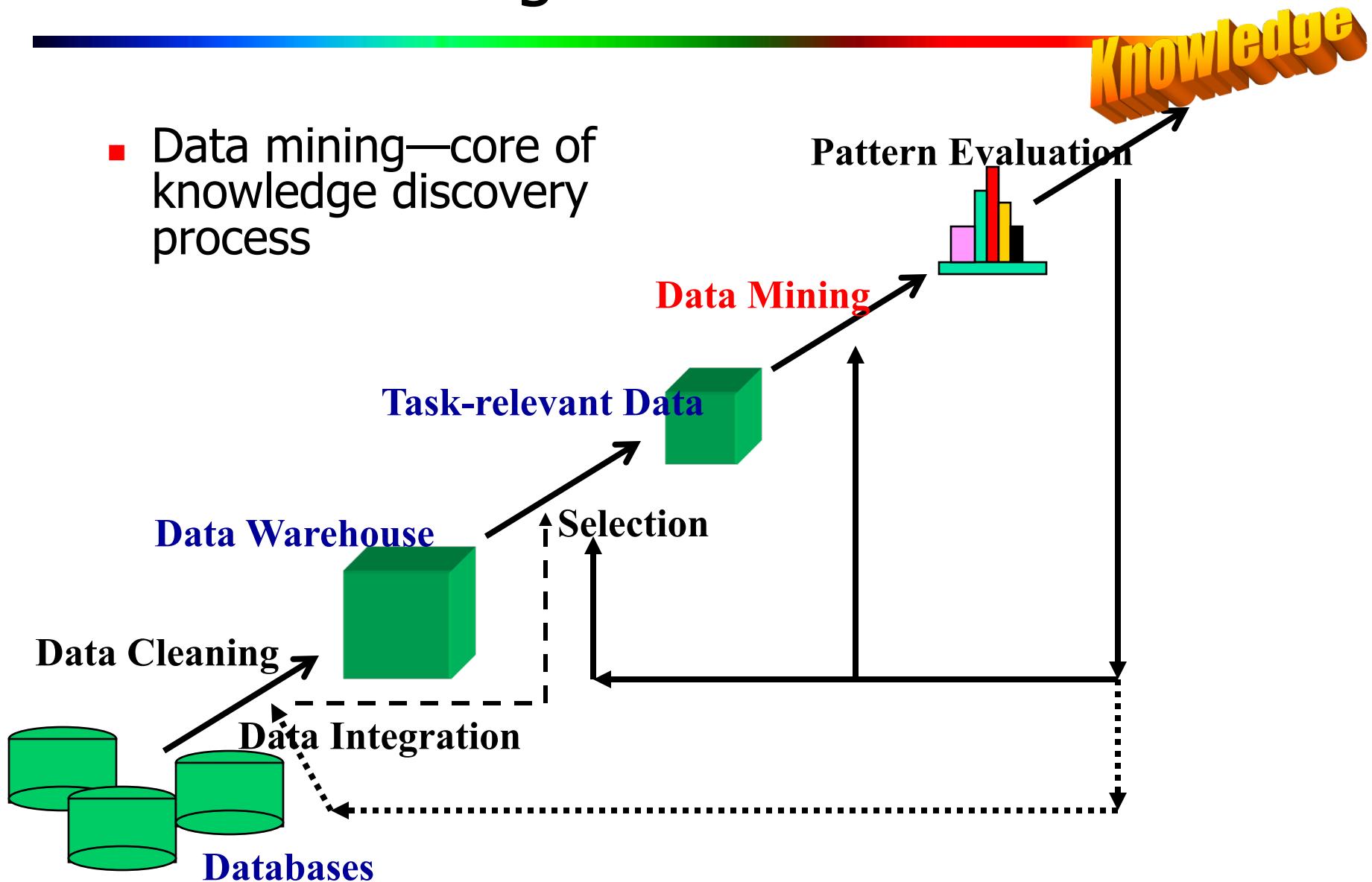
# Other Applications

---

- Sports
  - IBM Advanced Scout analyzed NBA game statistics (shots blocked, assists, and fouls) to gain competitive advantage for New York Knicks and Miami Heat
- Astronomy
  - JPL and the Palomar Observatory discovered 22 quasars with the help of data mining
- Internet Web Surf-Aid
  - IBM Surf-Aid applies data mining algorithms to Web access logs for market-related pages to discover customer preference and behavior pages, analyzing effectiveness of Web marketing, improving Web site organization, etc.

# Data Mining: A KDD Process

- Data mining—core of knowledge discovery process



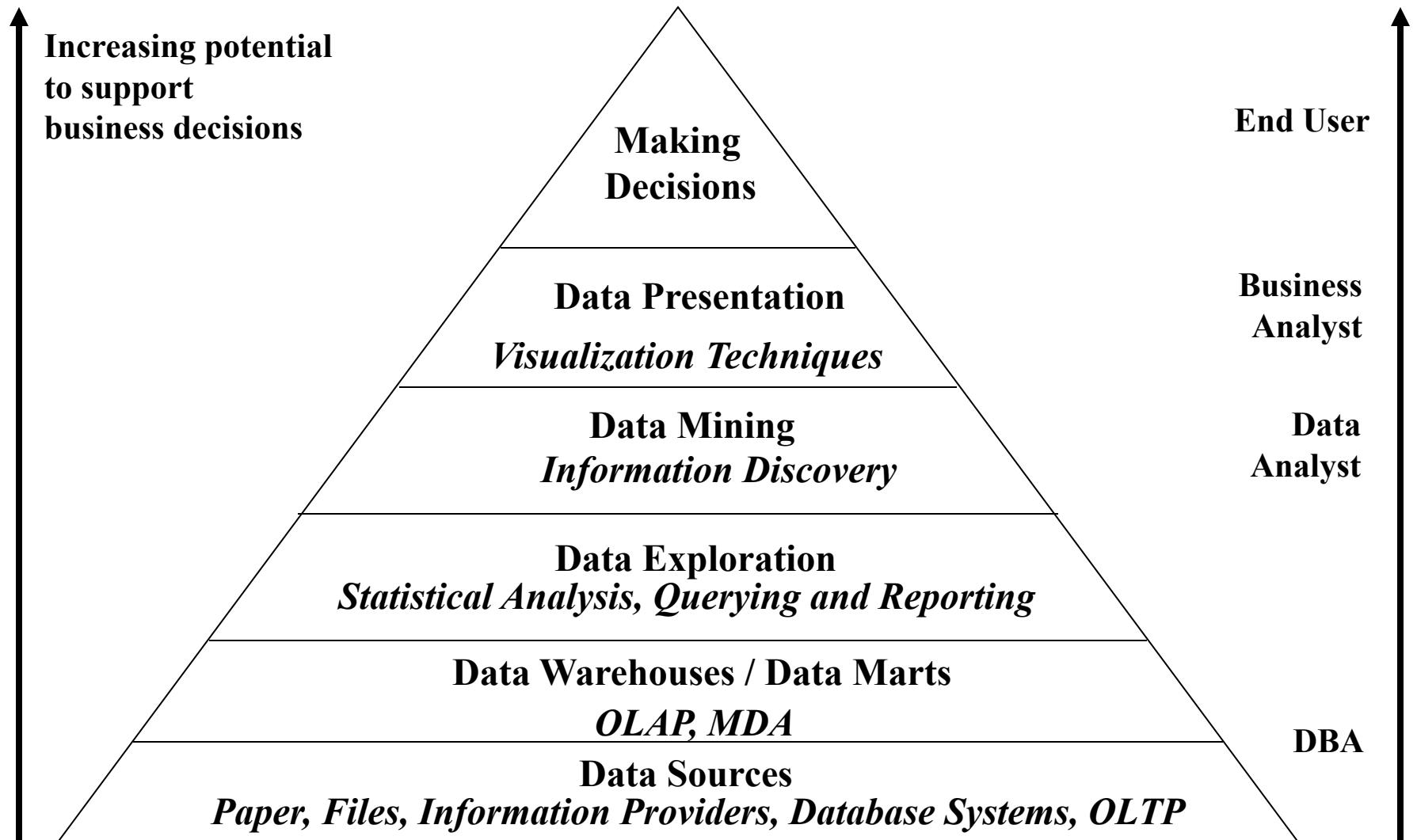
# Steps of a KDD Process

---

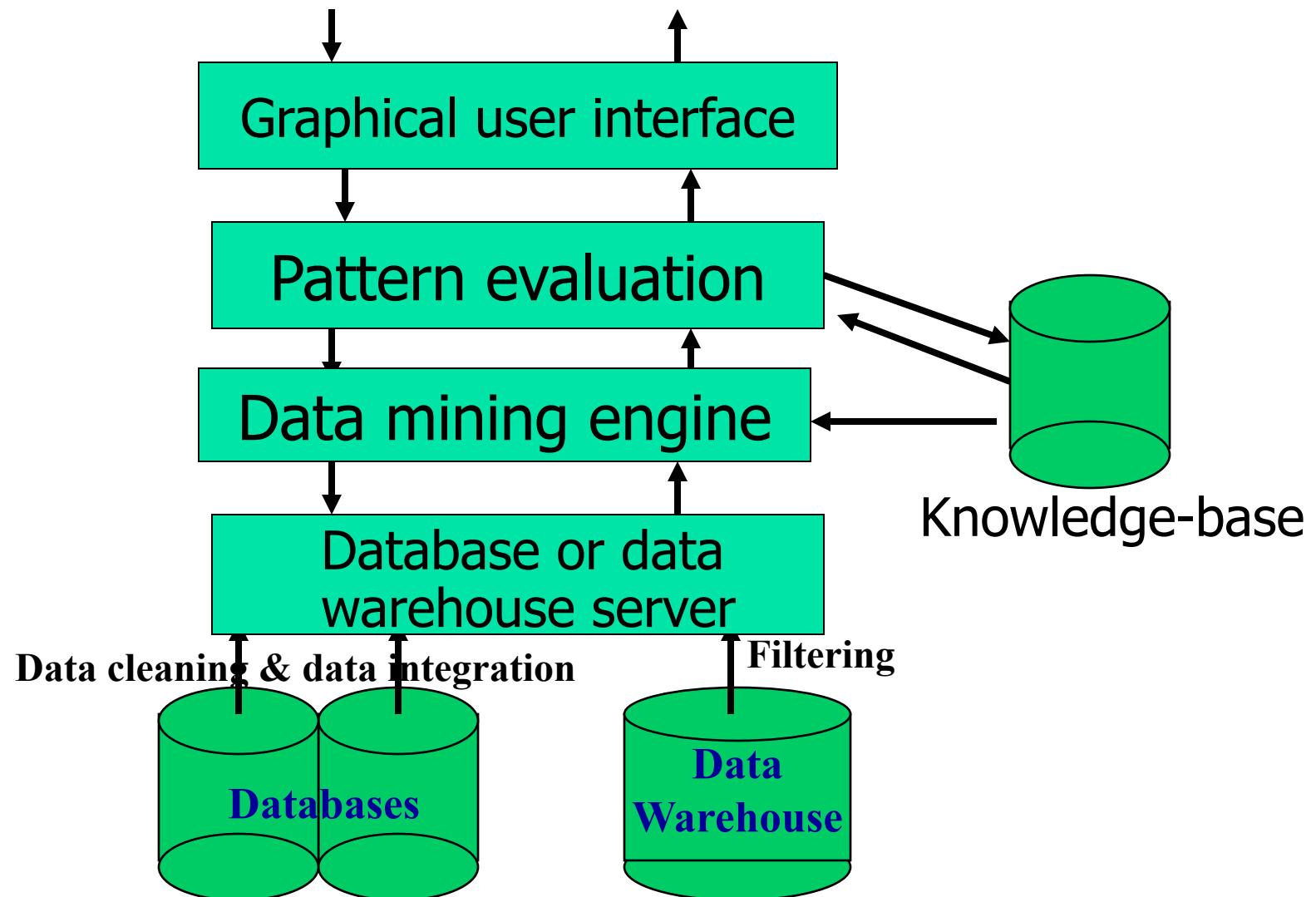
- Learning the application domain
  - relevant prior knowledge and goals of application
- Creating a target data set: data selection
- Data cleaning and preprocessing: (*may take 60% of effort!*)
- Data reduction and transformation
  - Find useful features, dimensionality/variable reduction, invariant representation.
- Choosing functions of data mining
  - summarization, classification, regression, association, clustering.
- Choosing the mining algorithm(s)
- Data mining: search for patterns of interest
- Pattern evaluation and knowledge presentation
  - visualization, transformation, removing redundant patterns, etc.
- Use of discovered knowledge

# Data Mining and *Business Intelligence*

---



# Architecture: Typical Data Mining System



# Data Mining: On What Kinds of Data?

---

- Relational database
- Data warehouse
- Transactional database
- Advanced database and information repository
  - Object-relational database
  - Spatial and temporal data
  - Time-series data
  - Stream data
  - Multimedia database
  - Heterogeneous and legacy database
  - Text databases & WWW

# Data Mining Functionalities

---

- Concept description: Characterization and discrimination
  - Generalize, summarize, and contrast data characteristics, e.g., dry vs. wet regions
- Association (correlation and causality)
  - Diaper → Beer [0.5%, 75%]
- Classification and Prediction
  - Construct models (functions) that describe and distinguish classes or concepts for future prediction
    - E.g., classify countries based on climate, or classify cars based on gas mileage
  - Presentation: decision-tree, classification rule, neural network
  - Predict some unknown or missing numerical values

# Data Mining Functionalities (2)

---

- Cluster analysis
  - Class label is unknown: Group data to form new classes, e.g., cluster houses to find distribution patterns
  - Maximizing intra-class similarity & minimizing interclass similarity
- Outlier analysis
  - Outlier: a data object that does not comply with the general behavior of the data
  - Noise or exception? No! useful in fraud detection, rare events analysis
- Trend and evolution analysis
  - Trend and deviation: regression analysis
  - Sequential pattern mining, periodicity analysis
  - Similarity-based analysis
- Other pattern-directed or statistical analyses

# Are All the “Discovered” Patterns Interesting?

---

- Data mining may generate thousands of patterns: Not all of them are interesting
  - Suggested approach: Human-centered, query-based, focused mining
- **Interestingness measures**
  - A pattern is **interesting** if it is easily understood by humans, valid on new or test data with some degree of certainty, potentially useful, novel, or validates some hypothesis that a user seeks to confirm
- **Objective vs. subjective interestingness measures**
  - **Objective**: based on **statistics and structures of patterns**, e.g., support, confidence, etc.
  - **Subjective**: based on **user’s belief** in the data, e.g., unexpectedness, novelty, actionability, etc.

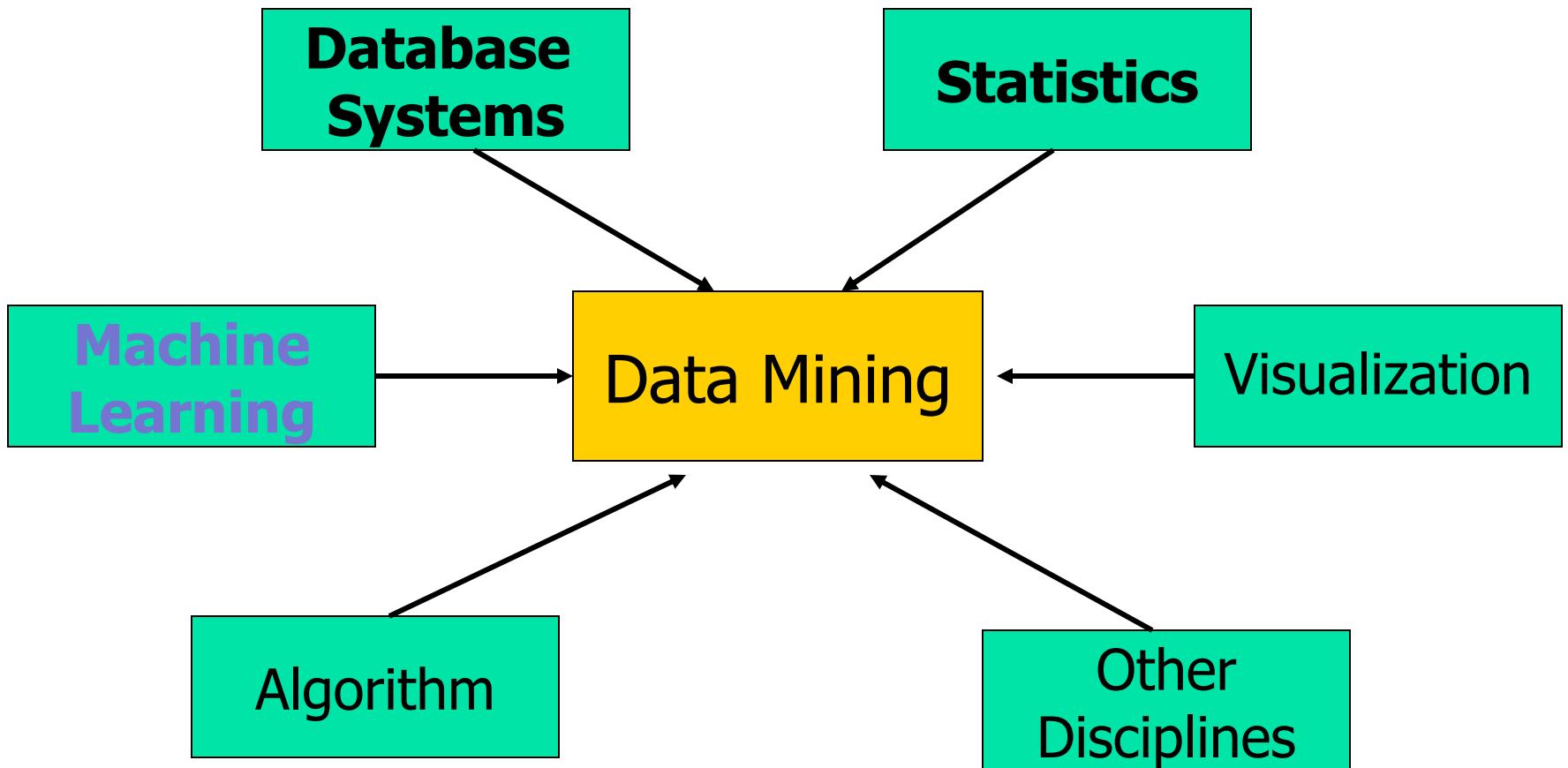
# Can We Find All and Only Interesting Patterns?

---

- Find all the interesting patterns: Completeness
  - Can a data mining system find all the interesting patterns?
  - Heuristic vs. exhaustive search
  - Association vs. classification vs. clustering
- Search for only interesting patterns: An optimization problem
  - Can a data mining system find only the interesting patterns?
  - Approaches
    - First generate all the patterns and then filter out the uninteresting ones.
    - Generate only the interesting patterns—mining query optimization

# Data Mining: Confluence of Multiple Disciplines

---



# Data Mining: Classification Schemes

---

- General functionality
  - Descriptive data mining
  - Predictive data mining
- Different views, different classifications
  - Kinds of data to be mined
  - Kinds of knowledge to be discovered
  - Kinds of techniques utilized
  - Kinds of applications adapted

# *Multi-Dimensional View of Data Mining*

---

- **Data to be mined**

- Relational, data warehouse, transactional, stream, object-oriented/relational, active, spatial, time-series, text, multi-media, heterogeneous, legacy, WWW

- **Knowledge to be mined**

- Characterization, discrimination, association, classification, clustering, trend/deviation, outlier analysis, etc.
  - Multiple/integrated functions and mining at multiple levels

- **Techniques utilized**

- Database-oriented, data warehouse (OLAP), machine learning, statistics, visualization, etc.

- **Applications adapted**

- Retail, telecommunication, banking, fraud analysis, bio-data mining, stock market analysis, Web mining, etc.

# Major Issues in Data Mining

---

- Mining methodology
  - Mining different kinds of knowledge from diverse data types, e.g., bio, stream, Web
  - Performance: efficiency, effectiveness, and scalability
  - Pattern evaluation: the interestingness problem
  - Incorporation of background knowledge
  - Handling noise and incomplete data
  - Parallel, distributed and incremental mining methods
  - Integration of the discovered knowledge with existing one: knowledge fusion
- User interaction
  - Data mining query languages and ad-hoc mining
  - Expression and visualization of data mining results
  - Interactive mining of knowledge at multiple levels of abstraction
- Applications and social impacts
  - Domain-specific data mining & invisible data mining
  - Protection of data security, integrity, and privacy

# Summary

---

- Data mining: discovering interesting patterns from large amounts of data
- A natural evolution of database technology, in great demand, with wide applications
- A KDD process includes data cleaning, data integration, data selection, transformation, data mining, pattern evaluation, and knowledge presentation
- Mining can be performed in a variety of information repositories
- Data mining functionalities: characterization, discrimination, association, classification, clustering, outlier and trend analysis, etc.
- Data mining systems and architectures
- Major issues in data mining

# A Brief History of Data Mining Society

---

- 1989 IJCAI Workshop on Knowledge Discovery in Databases (Piatetsky-Shapiro)
  - Knowledge Discovery in Databases (G. Piatetsky-Shapiro and W. Frawley, 1991)
- 1991-1994 Workshops on Knowledge Discovery in Databases
  - Advances in Knowledge Discovery and Data Mining (U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, 1996)
- 1995-1998 International Conferences on Knowledge Discovery in Databases and Data Mining (KDD'95-98)
  - Journal of Data Mining and Knowledge Discovery (1997)
- 1998 ACM SIGKDD, SIGKDD'1999-2001 conferences, and SIGKDD Explorations
- More conferences on data mining
  - PAKDD (1997), PKDD (1997), SIAM-Data Mining (2001), (IEEE) ICDM (2001), etc.

1. [DBLP](#)
2. [Google](#)
3. [Citeseer](#)
4. [DL@lib](#)

# Where to Find References?

---

## ■ Data mining and KDD

- Conferences: ACM-SIGKDD, IEEE-ICDM, SIAM-DM, PKDD, PAKDD, etc.
- Journal: Data Mining and Knowledge Discovery, KDD Explorations

## ■ Database systems

- Conferences: ACM-SIGMOD, ACM-PODS, VLDB, IEEE-ICDE, EDBT, ICDT, DASFAA
- Journals: ACM-TODS, IEEE-TKDE, JIIS, J. ACM, VLDBJ, etc.

## ■ AI & Machine Learning

- Conferences: Machine learning (ML), AAAI, IJCAI, COLT (Learning Theory), etc.
- Journals: Machine Learning, Artificial Intelligence, etc.

## ■ Statistics

- Conferences: Joint Stat. Meeting, etc.
- Journals: Annals of statistics, etc.

## ■ Visualization

- Conference proceedings: CHI, ACM-SIGGraph, etc.
- Journals: IEEE Trans. visualization and computer graphics, etc.

# Recommended Reference Books

---

- I. H. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, 2001
- **C. C. Aggarwal, Data Mining: The Textbook, Springer, 2015**
- **J. Leskovec, A. Rajaraman, and J. Ullman, Mining of Massive Datasets (v2.1), Cambridge University Press, 2014.**
- Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, Learning From Data. AMLBook, 2012.
- **J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2001**
- **D. J. Hand, H. Mannila, and P. Smyth, Principles of Data Mining, MIT Press, 2001**
- **T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer-Verlag, 2001**
- **T. M. Mitchell, Machine Learning, McGraw Hill, 1997**
- **P-N. Tan, M. Steinbach, and V. Kumar, Introduction to Data Mining,. Addison-Wesley, 2005**
- S. M. Weiss and N. Indurkhya, Predictive Data Mining, Morgan Kaufmann, 1998

# Jai's Project (COMP9318, 2016s2)

---

- Problem
  - <http://kentandlime.com.au/>, a startup company helping male customers to stay in fashion but out of the shops.
  - Status-quo:
    - Ask questions, and stylists makes a list of recommended items, and send them to customers
    - If happy, customers pay for the product.
  - Recommendation is the key!
- Challenges
  - Dirty data
  - Not an easy/typical recommendation system settings
  - Customer feedbacks
  - Real-time recommendations

<http://www.news.com.au/lifestyle/fashion/fashion-trends/fashions-most-unlikely-trend-would-you-buy-clothes-chosen-for-you/news-story/8634b5f06f608b9f2fd7c27758f9c10a>

# Solutions - Highlight

---

- Use domain-knowledge and quick evaluations to guide the whole process
- Data preprocessing
  - Data source: CRM (profile) + NoSQL DB (transactions)
  - Missing data: e.g., due to schema changes
  - Data normalization: A's XL = B's L
  - Data noise: k-means / binning
  - Data selection: remove sparse columns/rows
- Feature engineering
  - weight-to-height ratio

# Solutions – Highlight /2

---

- Product class clustering and prediction
- Collaborative filtering with smoothing and weighting
- Content-based recommendation (solve the cold start problem)
- Incorporate customer feedbacks
- Association rule mining:
  - LSShirts\_1, Shorts\_2 → Socks\_3
- Ensemble of the above
  
- Plus many engineering efforts

# Results

---

- Test set:
  - Classification rate: 74%, on par with humans
- Deployed to production on 18-24 Nov 2016:
  - Customers rejecting on average 2.36 items out of a basket of 10-12 items ➔ (76.4%, 80.3%)
  - Latency: 2.3s
- Future work identified
  - e.g., seasonality
- Check Jai's presentation slides for more details.

---

# COMP9318: Data Warehousing and Data Mining

— L2: Data Warehousing and OLAP —

---

- Why and What are Data Warehouses?

# Data Analysis Problems

---

- The same data found in many different systems
  - Example: customer data across different departments
  - The same concept is defined differently
- Heterogeneous sources
  - Relational DBMS, OnLine Transaction Processing (OLTP)
  - Unstructured data in files (e.g., MS Excel) and documents (e.g., MS Word)

# Data Analysis Problems (Cont'd)

---

- Data is suited for operational systems
  - Accounting,billing,etc.
  - Do not support analysis across business functions
- Data quality is bad
  - Missing data, imprecise data, different use of systems
- Data are “volatile”
  - Data deleted in operational systems (6months)
  - Data change over time – no historical information

# Solution: 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 issues 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”.

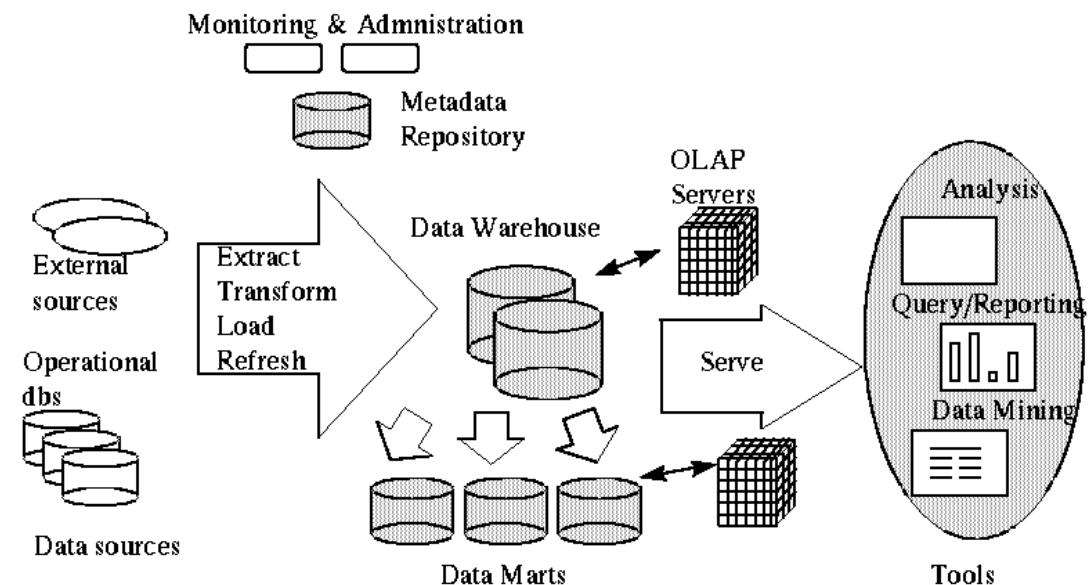
# Data Warehouse—Non-Volatile

---

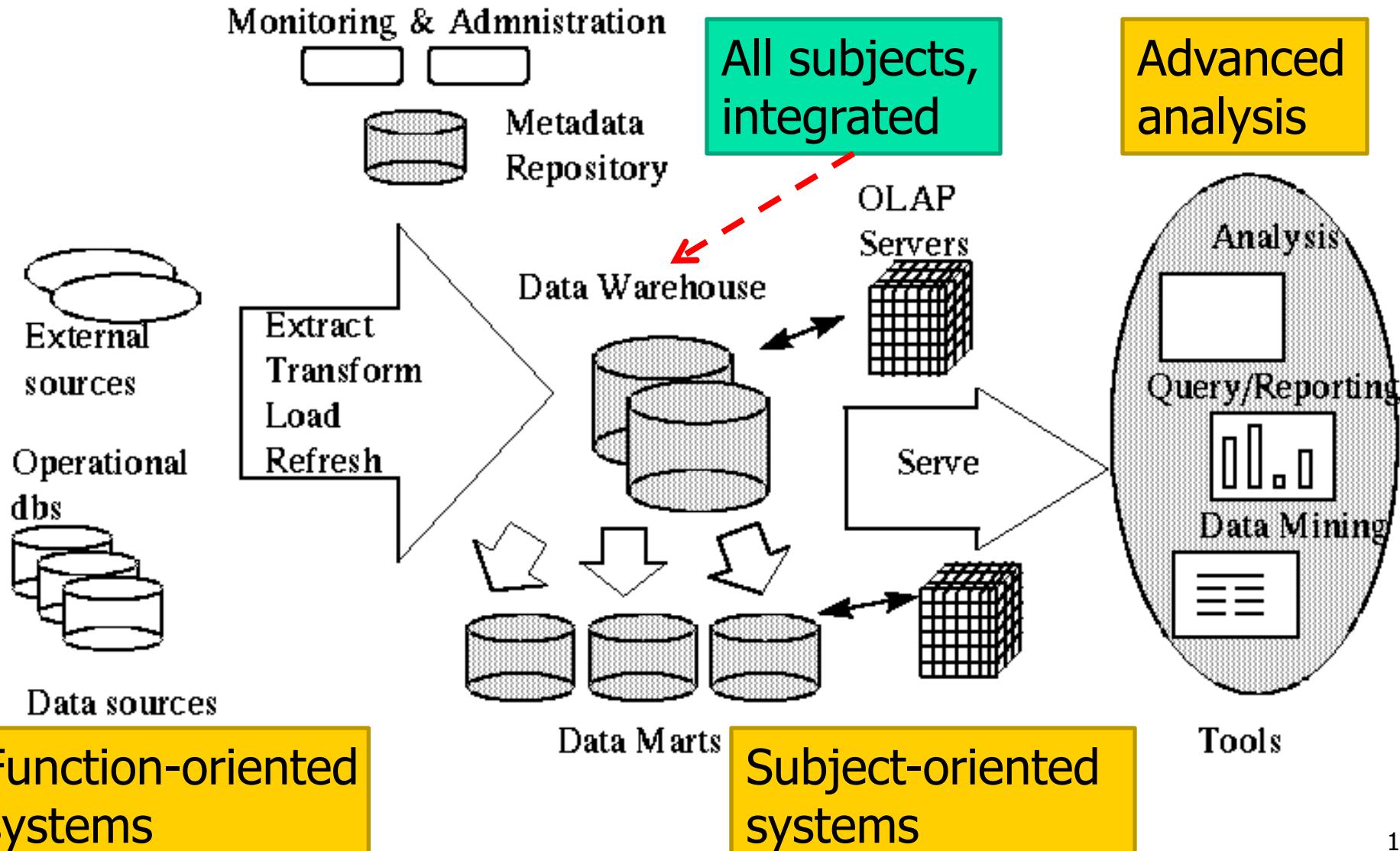
1. A **physically separate store** of data transformed from the operational environment.
2. Operational **update of data does not occur** in the data warehouse environment.
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data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

# Data Warehouse Architecture

- Extract data from operational data sources
  - clean, transform
- Bulk load/refresh
  - warehouse is offline
- OLAP-server provides multidimensional view
- Multidimensional-olap (Essbase, oracle express)
- Relational-olap (Redbrick, Informix, Sybase, SQL server)



# Data Warehouse Architecture



# Why 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

# Why OLAP Servers?

---

- Different workload:
  - OLTP (on-line transaction processing)
    - Major task of traditional relational DBMS
    - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.
  - OLAP (on-line analytical processing)
    - Major task of data warehouse system
    - Data analysis and decision making
- Queries hard/infeasible for OLTP, e.g.,
  - Which **week** we have the largest sales?
  - Does the sales of **dairy products** increase over time?
  - Generate a **spread sheet** of total sales by state and by year.
- Difficult to represent these queries by using SQL ← **Why?**

# OLTP vs. OLAP

---

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

# Comparisons

	<b>Databases</b>	<b>Data Warehouses</b>
Purpose	Many purposes; Flexible and general	One purpose: Data analysis
Conceptual Model	ER	Multidimensional
Logical Model	(Normalized) Relational Model	(Denormalized) Star schema / Data cube/cuboids
Physical Model	Relational Tables	ROLAP: Relational tables MOLAP: Multidimensional arrays
Query Language	SQL (hard for analytical queries)	MDX (easier for analytical queries)
Query Processing	B+-tree/hash indexes, Multiple join optimization, Materialized views	Bitmap/Join indexes, Star join, Materialized data cube

---

- The Multidimensional Model

# The Multidimensional Model

---

- A data warehouse is based on a multidimensional data model which views data in the form of a **data cube**, which is a multidimensional generalization of 2D spread sheet.
- Key concepts:
  - **Facts**: the subject it models
    - Typically transactions in this course; other types includes snapshots, etc.
    - Measures: numbers that can be aggregated
    - Dimensions: context of the measure
  - Hierarchies:
    - Provide contexts of different granularities (aka. grains)
- Goals for dimensional modeling:
  - Surround facts with as much relevant context (dimensions) as possible ← **Why?**

# Supermarket Example

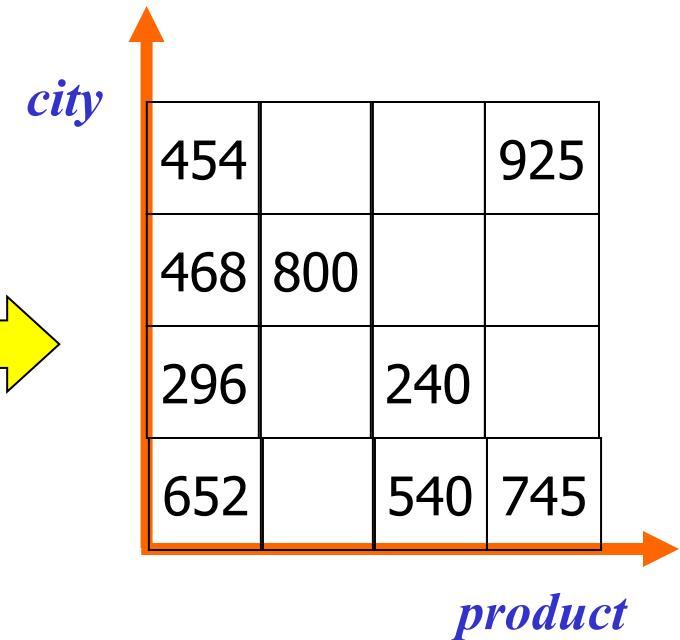
---

- Subject: analyze total sales and profits
- Fact: Each Sales **Transaction**
  - Measure: Dollars\_Sold, Amount\_Sold, Cost
  - Calculated Measure: Profit
- Dimensions:
  - Store
  - Product
  - Time

# Visualizing the Cubes

- A valid **instance** of the model is a data cube

total Sales		product			
city	NY	\$454	-	-	\$925
	LA	\$468	\$800	-	-
	SD	\$296	-	\$240	-
	SF	\$652	-	\$540	\$745



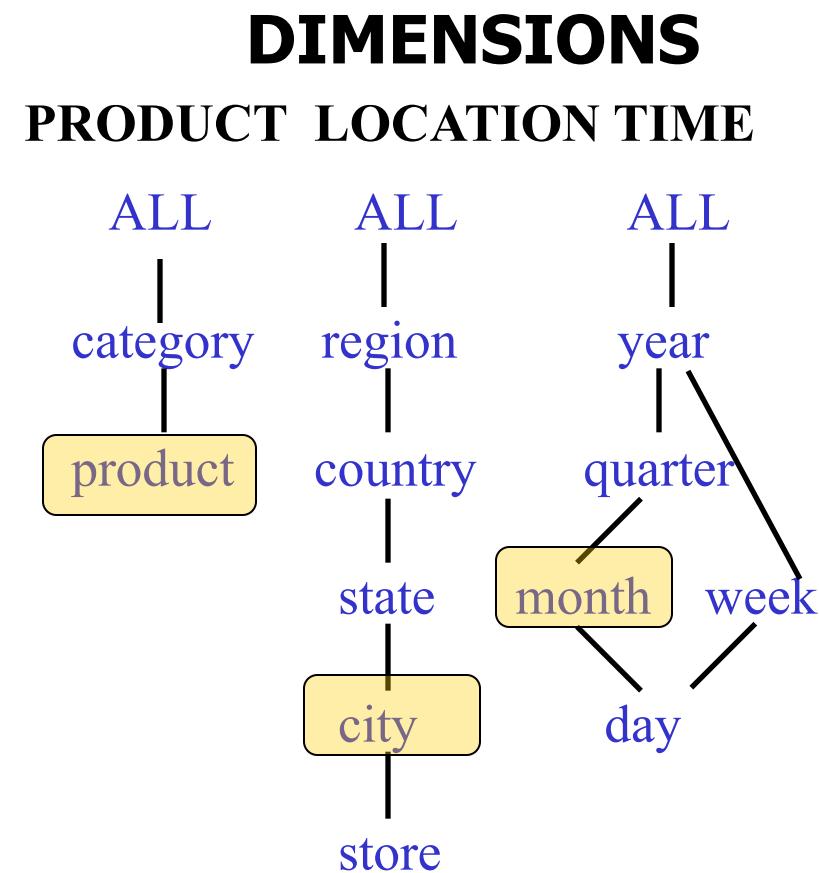
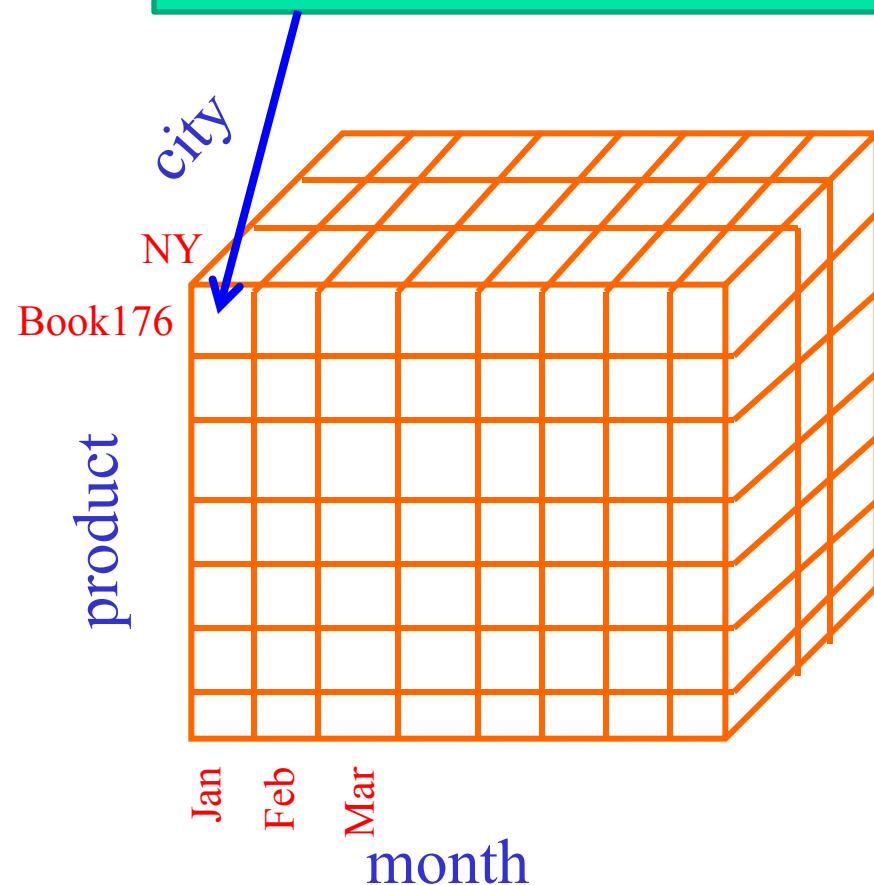
**Concepts:** cell, fact (=non-empty cell), measure, dimensions

Q: How to generalize it to 3D?

# 3D Cube and Hierarchies

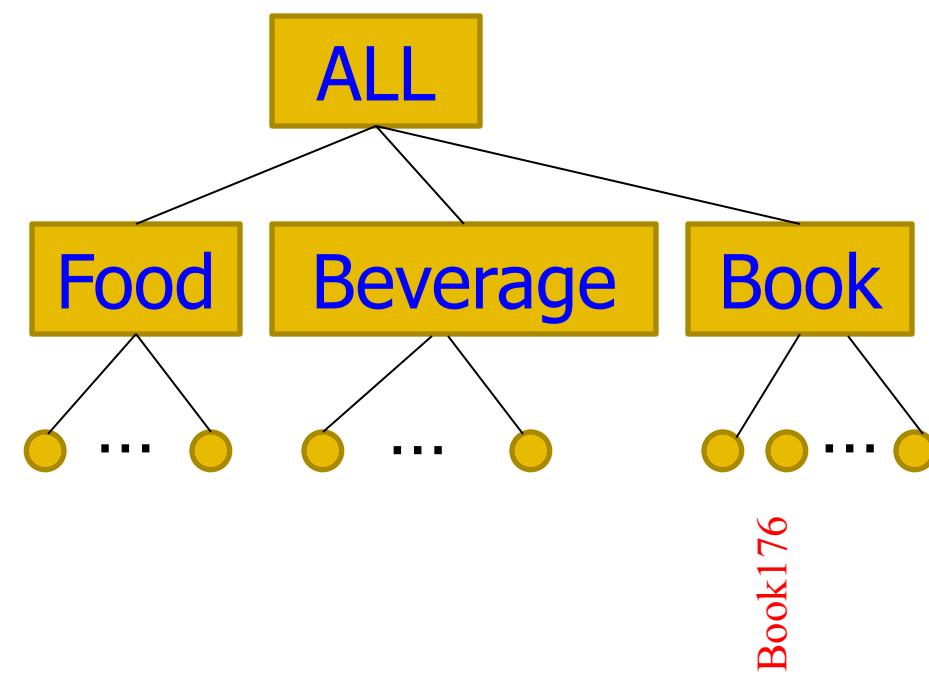
**Concepts:** hierarchy (a tree of dimension values), level

Sales of book176 in NY in Jan can be found in this cell

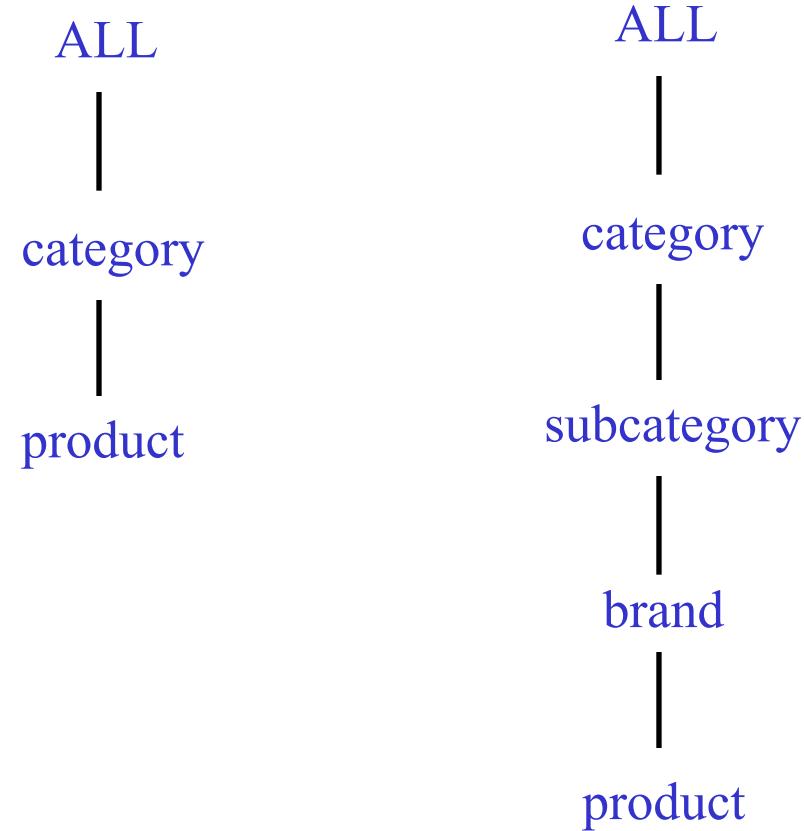


# Hierarchies

**Concepts:** hierarchy (a tree of dimension values), level

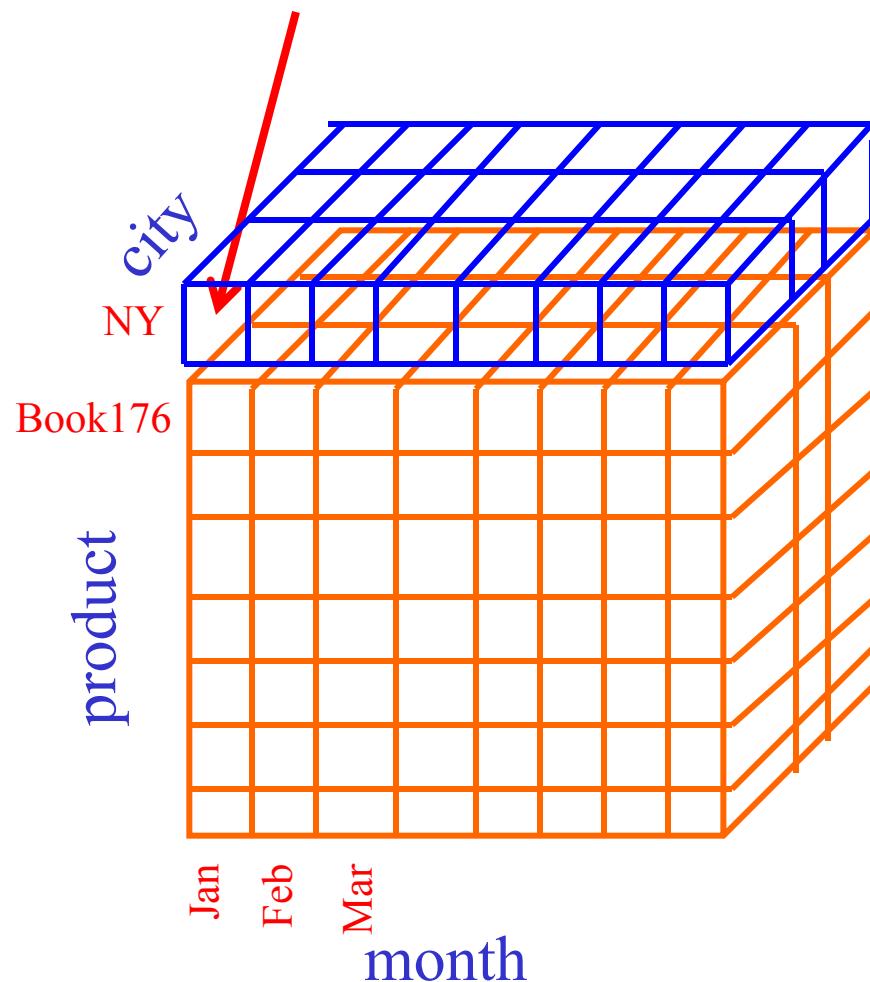


Which design is better? Why?

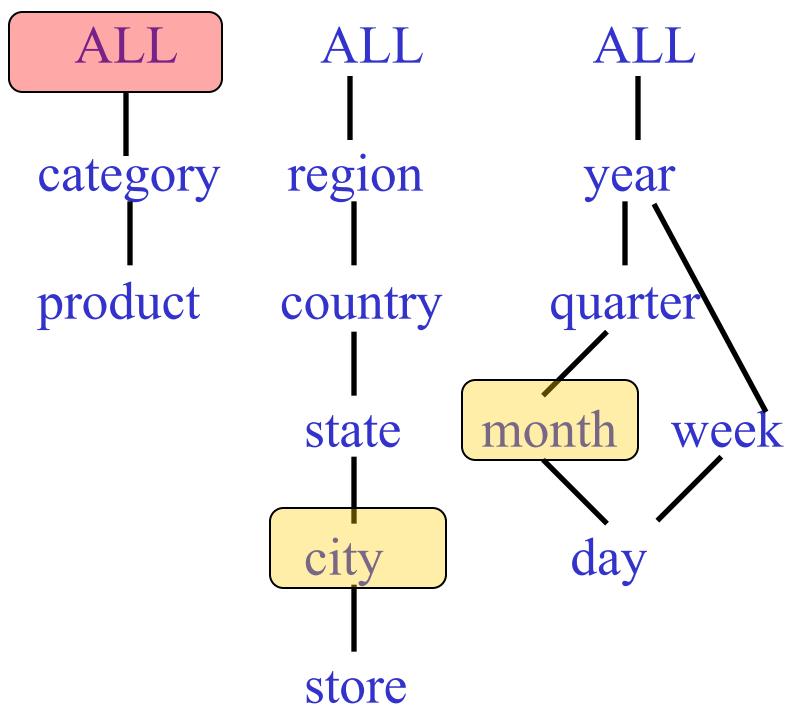


# The (city, moth) Cuboid

Sales of ALL\_PROD in NY in Jan

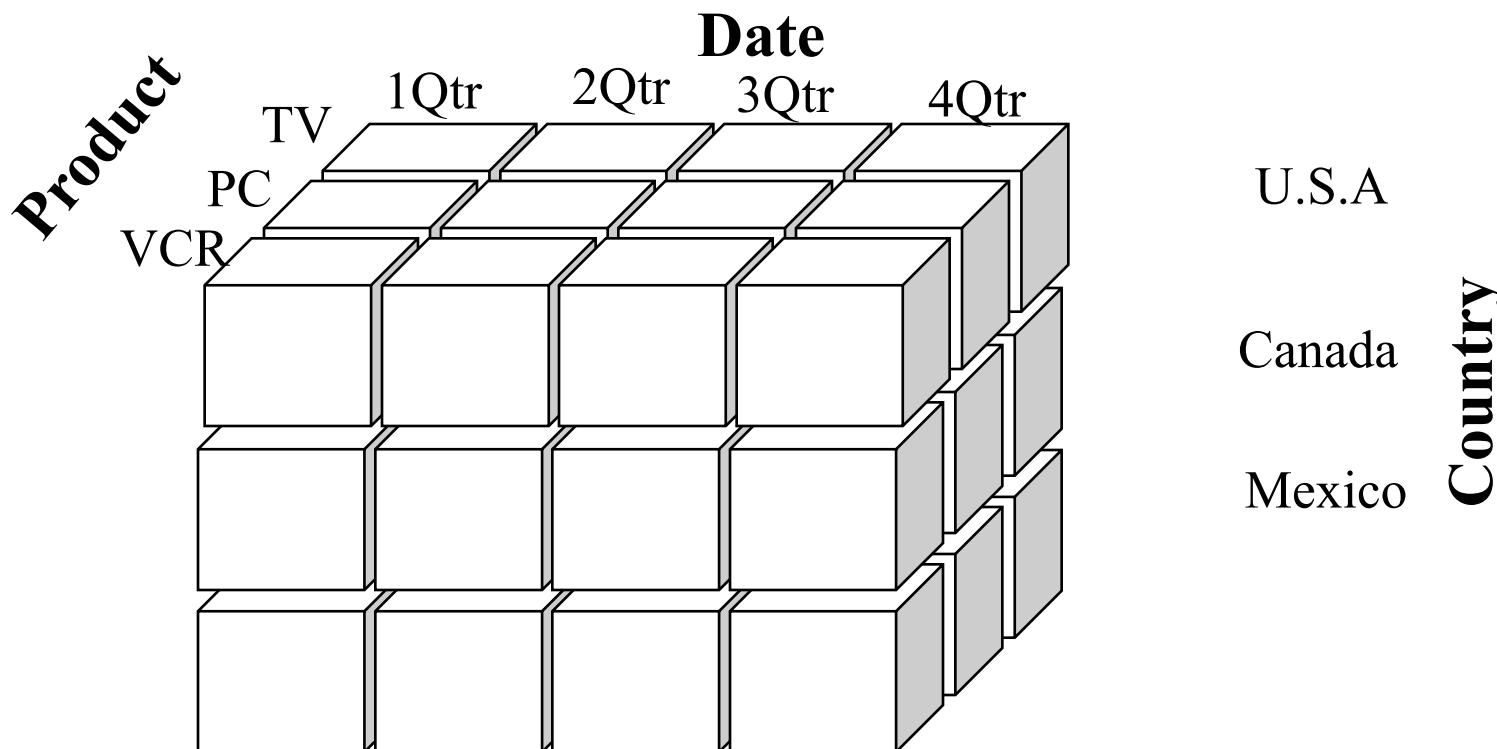


**DIMENSIONS**  
PRODUCT LOCATION TIME



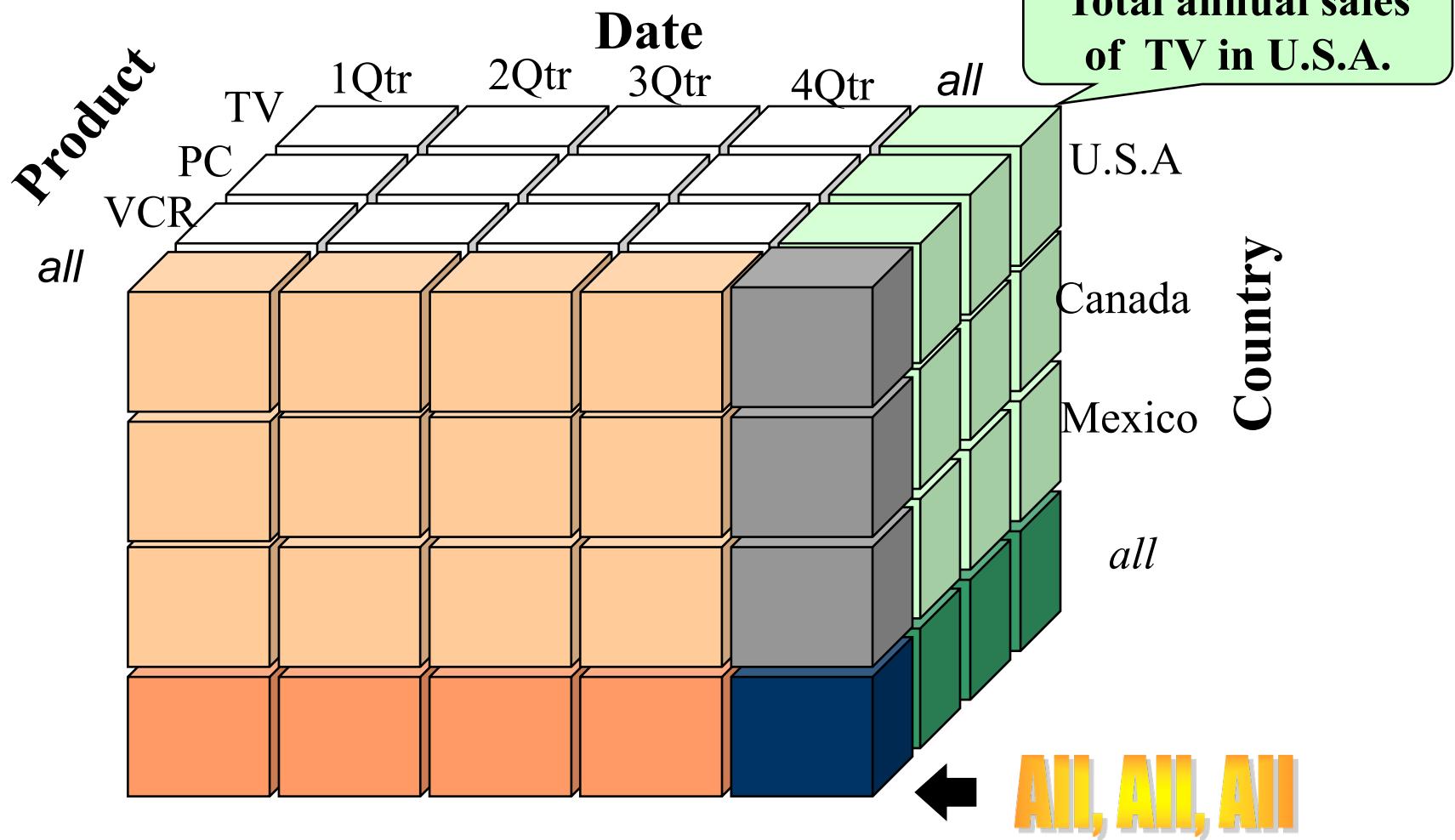
# All the Cuboids

Assume: no other non-ALL levels on all dimensions.

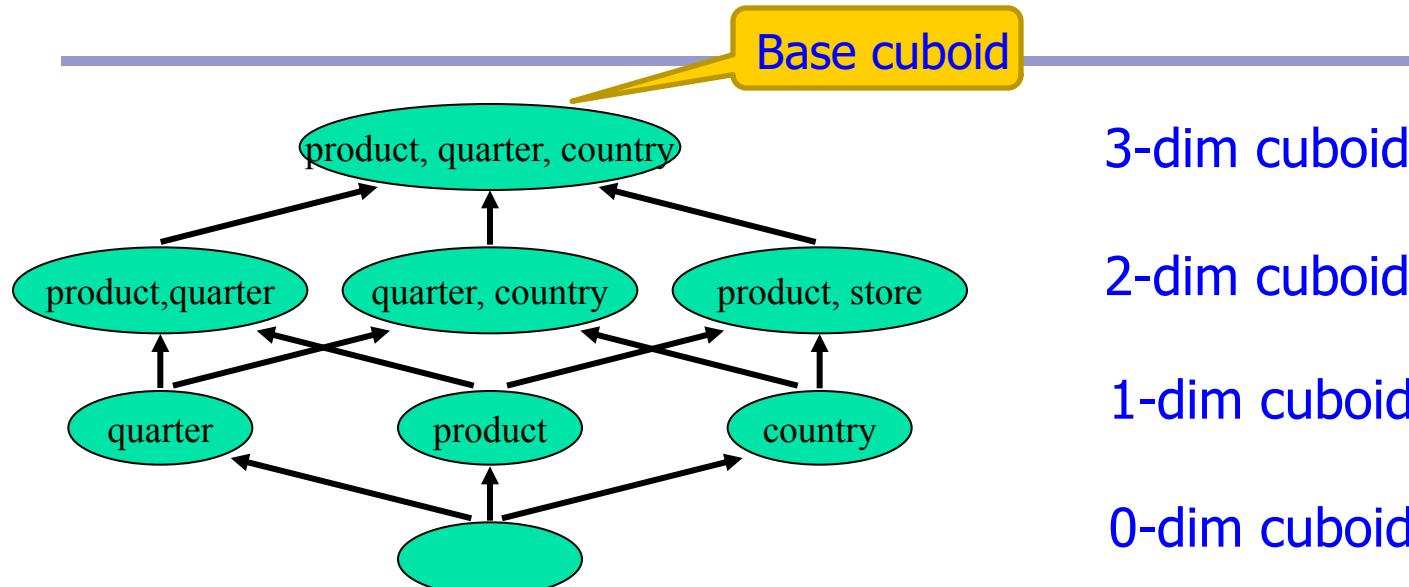


## All the Cuboids /2

Assume: no other non-ALL levels on all dimensions.



# Lattice of the cuboids



- n-dim cube can be represented as  $(D_1, D_2, \dots, D_d)$ , where  $D_i$  is the set of allowed values on the i-th dimension.
  - if  $D_i = L_i$  (a particular level), then  $D_i =$  all descendant dimension values of  $L_i$ .
  - ALL can be omitted and hence reduces the effective dimensionality
- A complete cube of d-dimensions consists of  $\prod_{i=1}^d (n_i + 1)$  cuboids, where  $n_i$  is the number of levels (excluding ALL) on i-th dimension.
  - They collectively form a lattice.

# Properties of Operations

---

- All operations are closed under the multidimensional model
  - i.e., both input and output of an operation is a cube
- So that they can be composed

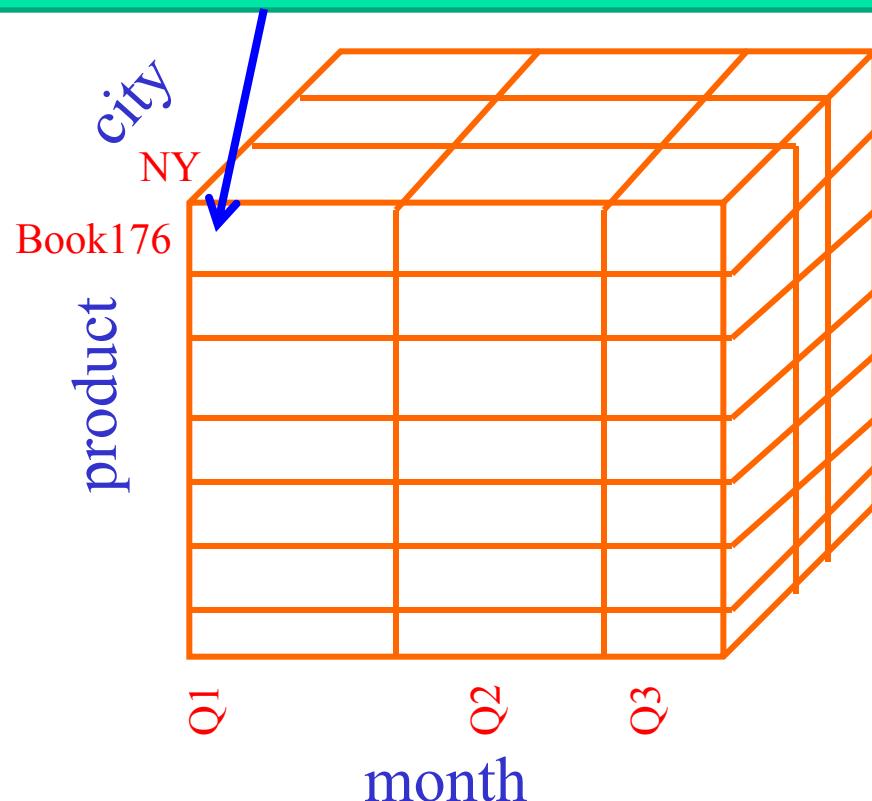
Q: What's the analogy in the Relational Model?

# Common OLAP Operations

- **Roll-up:** move up the hierarchy

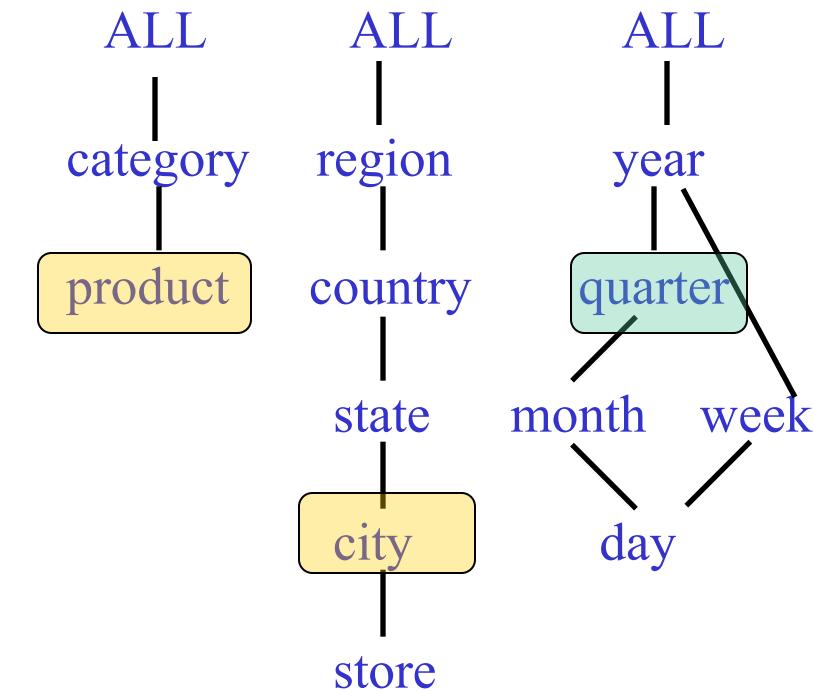
Q: what should be its value?

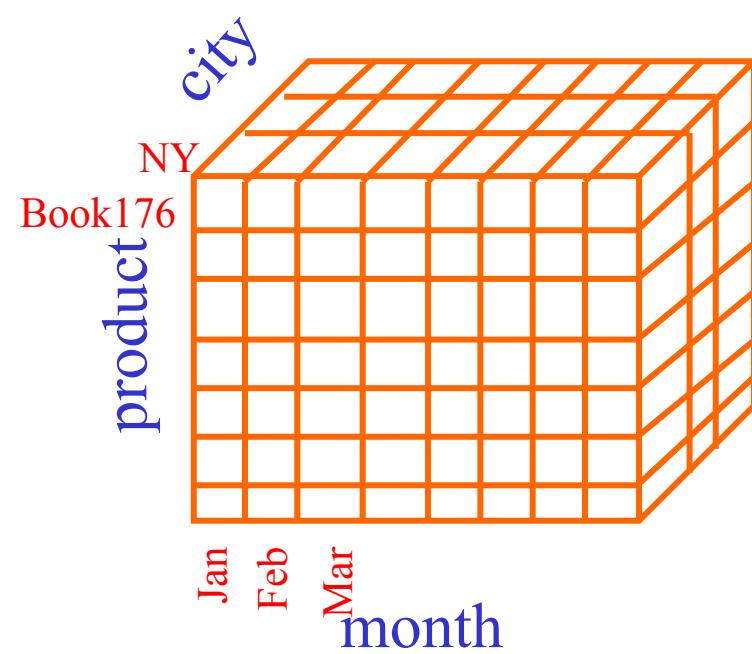
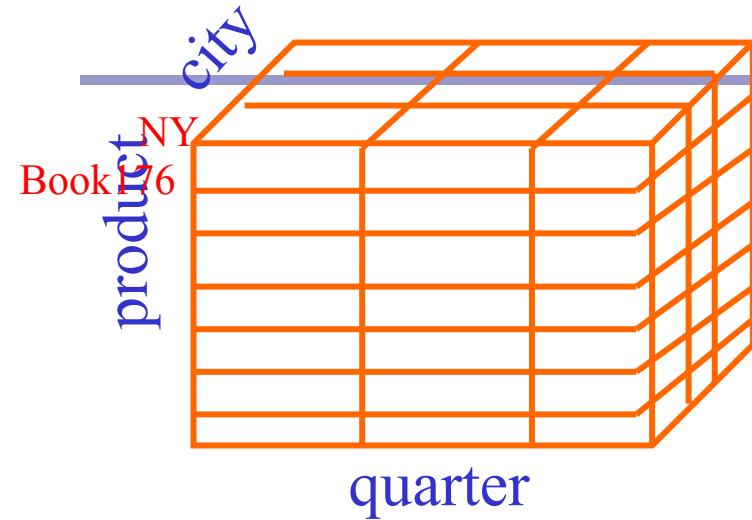
Sales of book176 in NY in Q1 here



## DIMENSIONS

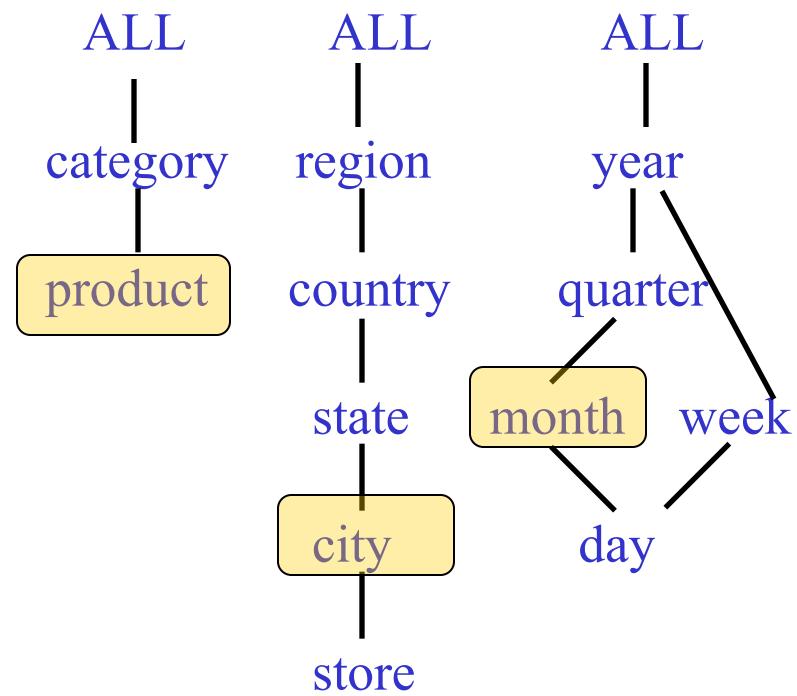
PRODUCT LOCATION TIME





## **DIMENSIONS**

**PRODUCT   LOCATION   TIME**



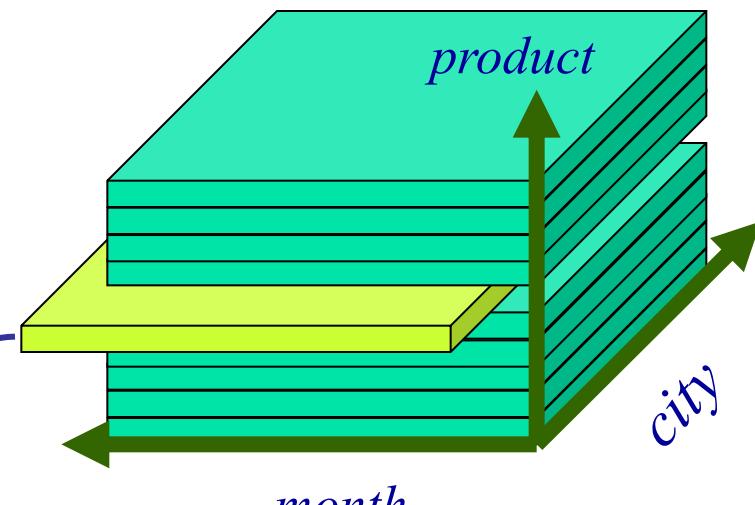
# Common OLAP Operations

---

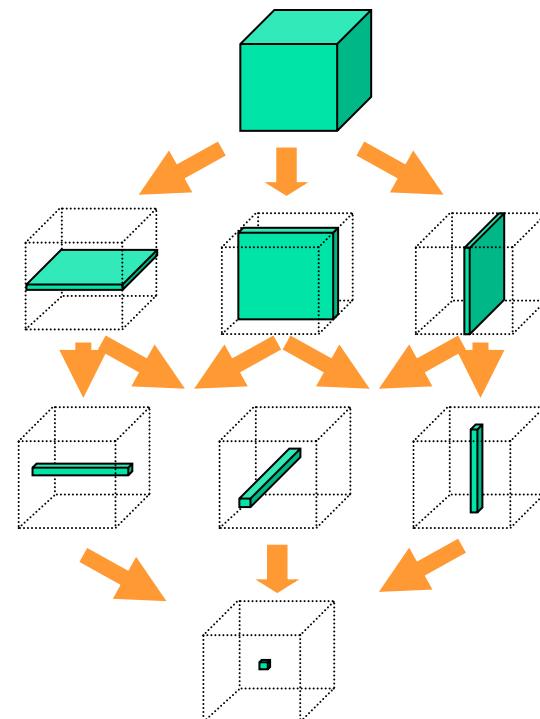
- **Drill-down:** move down the hierarchy
  - more fine-grained aggregation

# Slice and Dice Queries

- Slice and Dice: select and project on one or more dimension values



*Product = Book176*

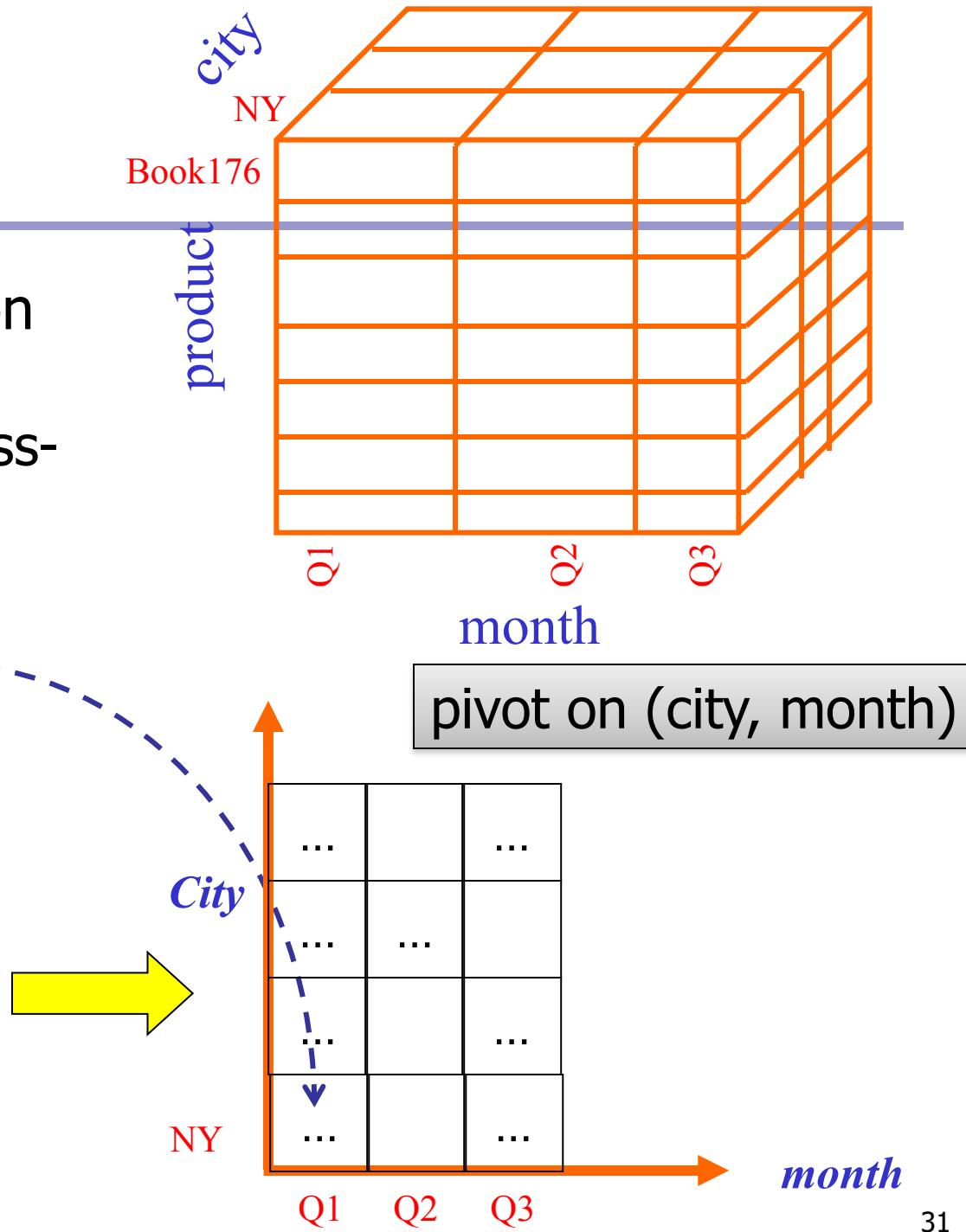


The output cube has smaller dimensionality than the input cube

# Pivoting

- Pivoting: aggregate on selected dimensions
  - usually 2 dims (cross-tabulation)

Sales (of all products) in NY in Q1  
 $=\text{sum}(\text{???})$



# A Reflective Pause

---

- Let's review the definition of data cubes again.
- **Key message:**
  - Disentangle the “object” from its “representation” or “implementation”

# Modeling Exercise 1: Monthly Phone Service Billing

The screenshot shows the Telstra website homepage with a blue header bar. The header includes the Telstra logo, a search bar, and navigation links for Telstra Home, Shop Online, My Account, Products & Services, Help Centre, and About Telstra.

**My Account** section:

- [Login to My Account](#)

**Quick Links** section:

- [Check my WebMail](#)
- [Pay my bill](#)
- [Find a mobile](#)
- [Move home](#)
- [Activate Pre-Paid](#)
- [Search FAQs](#)
- [Switch to Telstra](#)
- [Find a Telstra or T\[life\] shop](#)
- [Discover Next G™](#)
- [Contact Us](#)

**Latest News** section (partially visible):

**Whereis® Mobile - FREE to browse^ on your Next G™ mobile**

**RECOMMENDED FOR RURAL HANDHELD COVERAGE**

**For Next G™ handset coverage questions call the 1800 888 888 hotline**

**Personal** section:

- [Mobile](#)
- [BigPond® Internet](#)
- [Bundle & Save](#)
- [Home Phone](#)
- [FOXTEL AUSTAR from Telstra](#)

**Business** section:

- [Small & Medium Business](#)
- [Enterprise & Government](#)

Theme: analyze the income/revenue of Telstra

# Solution

---

- FACT
- MEASURE
- DIMENSIONS



---

- The Logical Model

# Logical Models

---

- Two main approaches:
  - Using relational DB technology:
    - Star schema, Snowflake schema, Fact constellation
  - Using multidimensional technology:
    - Just as multidimensional data cube

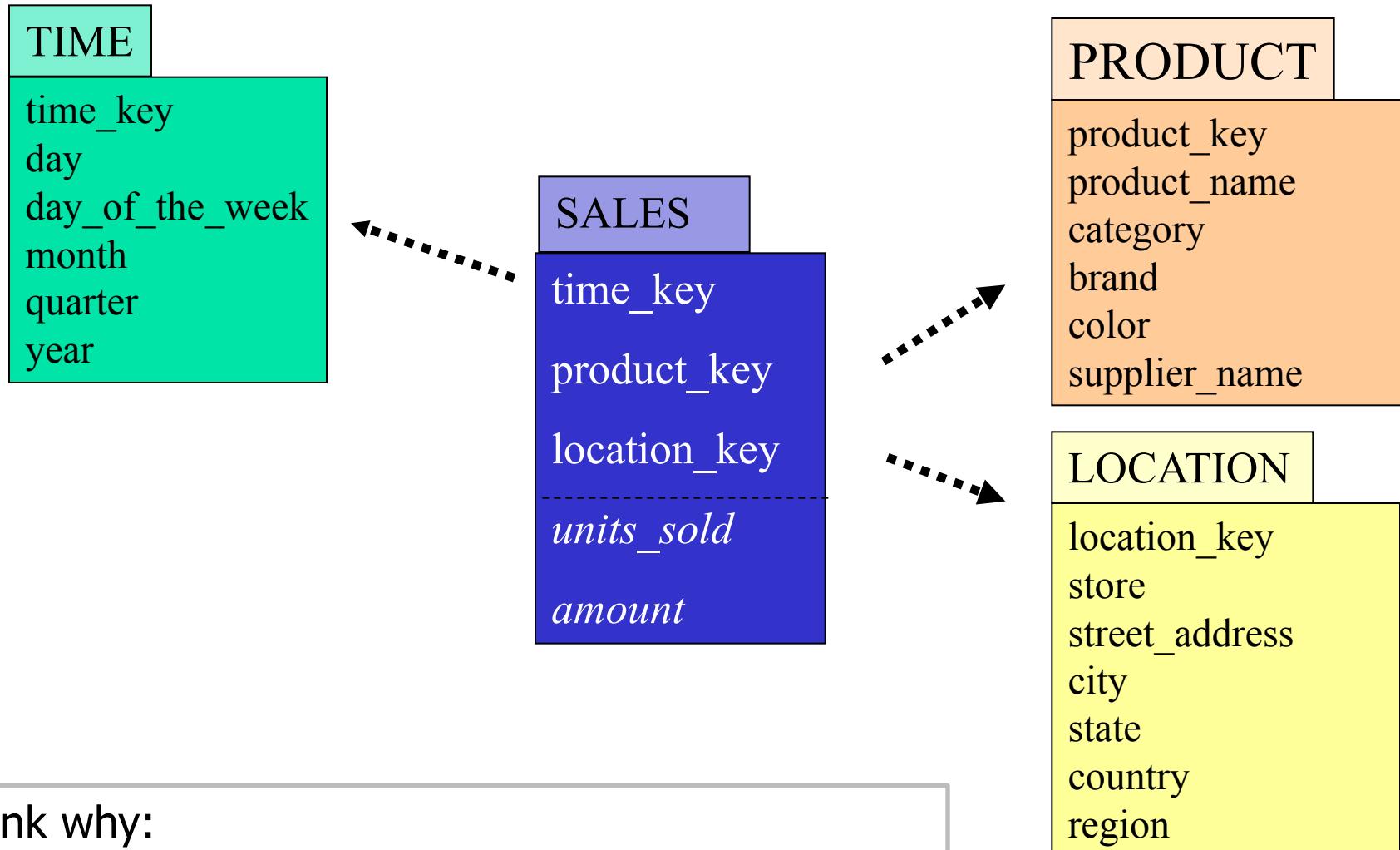
# Universal Schema → Star Schema

- Many data warehouses adopt a star schema to represent the multidimensional model
- Each dimension is represented by a dimension-table
  - LOCATION (location\_key, store, street\_address, city, state, country, region)
  - dimension tables are not normalized
- Transactions are described through a fact-table
  - each tuple consists of a pointer to each of the dimension-tables (foreign-key) and a list of measures (e.g. sales \$\$\$)

The universal schema for supermarket

Store	City	State	Prod	Brand	Category	\$Sold	#Sold	Cost
S136	Syd	NSW	76Ha	Nestle	Biscuit	40	10	18
S173	Melb	Vic	76Ha	Nestle	Biscuit	20	5	11

# The Star Schema



Think why:

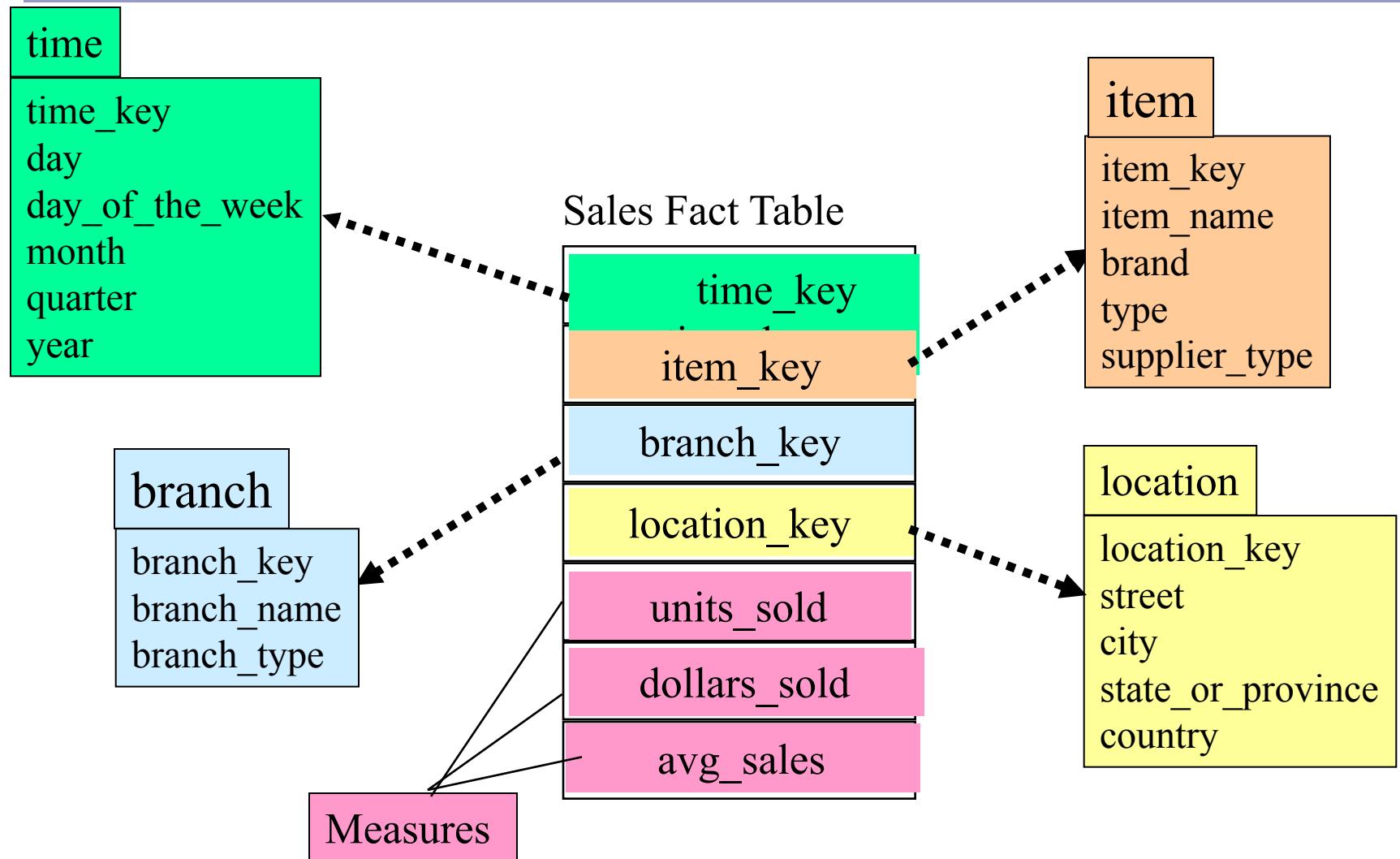
- (1) Denormalized **once** from the universal schema
- (2) Controlled **redundancy**

# Typical Models for Data Warehouses

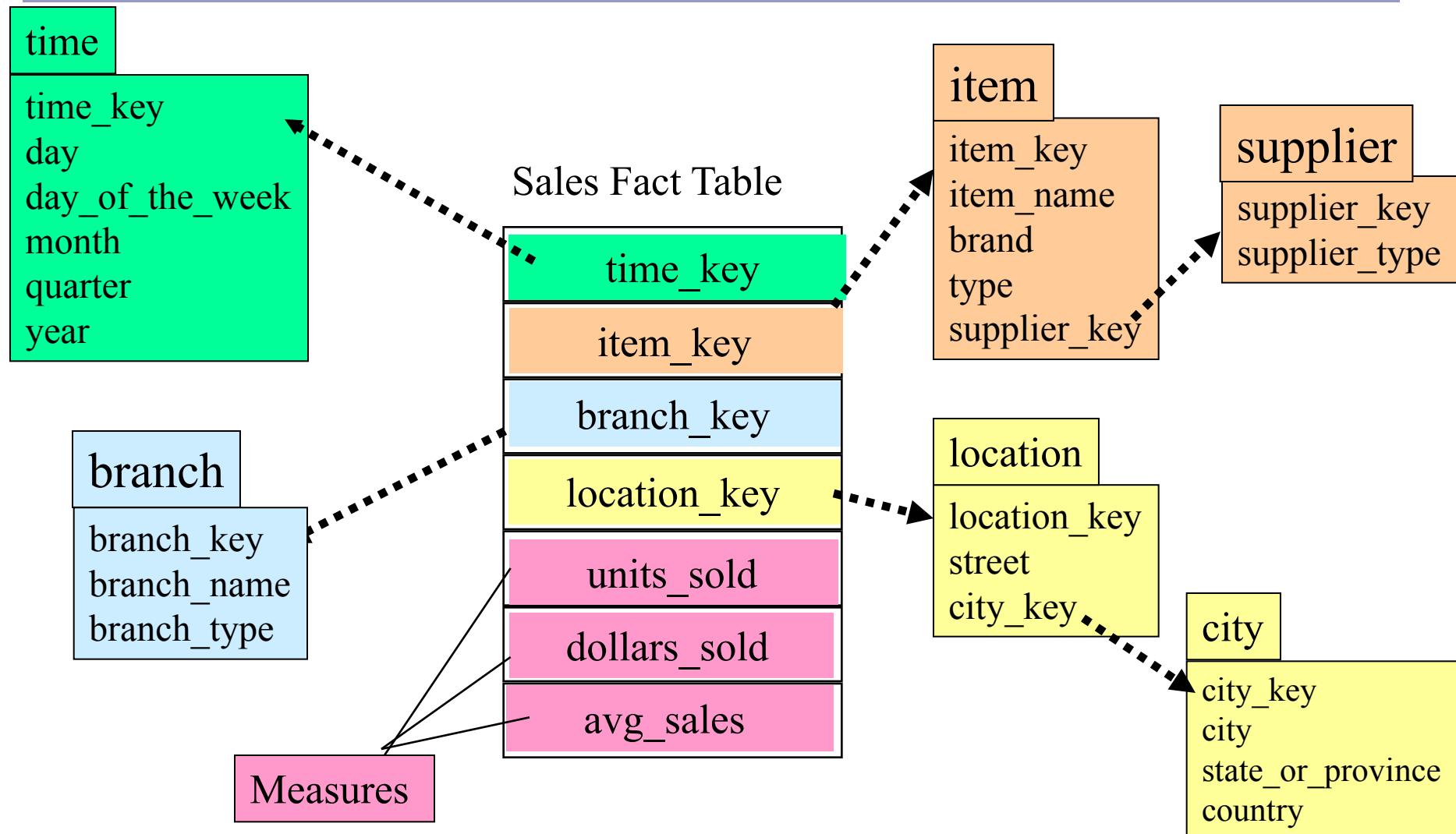
---

- Modeling data warehouses: dimensions & measures
  - Star schema: A fact table in the middle connected to a set of dimension tables
  - Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
  - Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

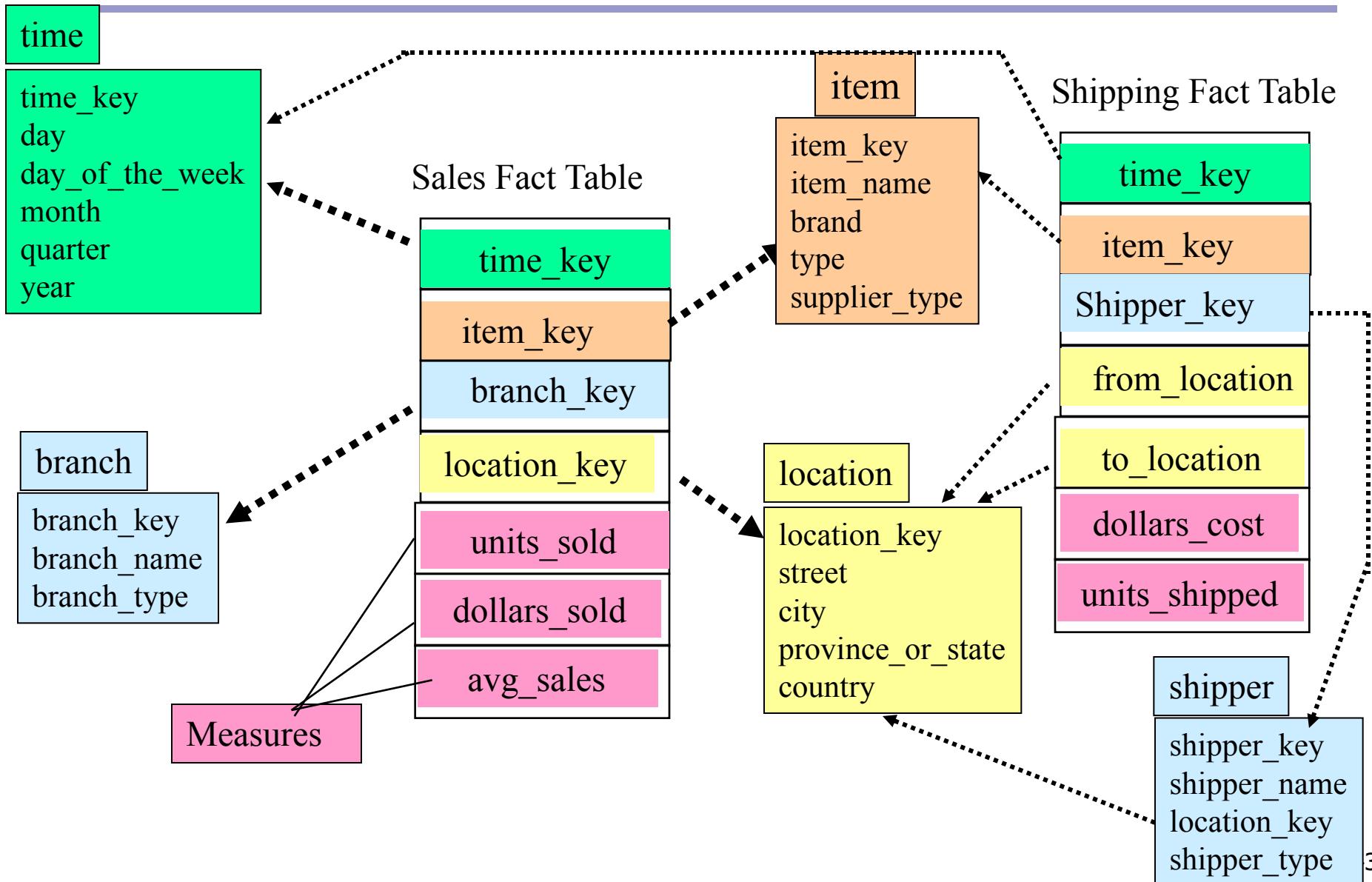
# Example of Star Schema



# Example of Snowflake Schema



# Example of Fact Constellation



# Advantages of Star Schema

---

- Facts and dimensions are clearly depicted
  - dimension tables are relatively static, data is loaded (append mostly) into fact table(s)
  - easy to comprehend (and write queries)

*“Find total sales per product-category in our stores in Europe”*

```
SELECT PRODUCT.category, SUM(SALES.amount)  
FROM SALES, PRODUCT, LOCATION  
WHERE SALES.product_key = PRODUCT.product_key  
AND SALES.location_key = LOCATION.location_key  
AND LOCATION.region = “Europe”  
GROUP BY PRODUCT.category
```

Operations: Slice (Loc.Region.Europe) + Pivot (Prod.category)

---

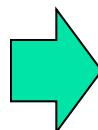
- Query Language

# Query Language

- Two approaches:
  - Using relational DB technology: SQL (with extensions such as **CUBE/PIVOT/UNPIVOT**)
  - Using multidimensional technology: **MDX**

```
SELECT PRODUCT.category,  
SUM(SALES.amount)  
FROM SALES, PRODUCT, LOCATION  
WHERE SALES.product_key =  
PRODUCT.product_key  
AND SALES.location_key =  
LOCATION.location_key  
AND LOCATION.region="Europe"  
GROUP BY PRODUCT.category
```

```
SELECT  
{[PRODUCT].[category]} on ROWS,  
{[MEASURES].[amount]} on COLUMNS  
FROM [SALES]  
WHERE ([LOCATION].[region].[Europe])
```



---

- Physical Model + Query Processing Techniques

# Physical Model + Query Processing Techniques

---

- Two main approaches:
  - Using relational DB technology: ROLAP
  - Using multidimensional technology: MOLAP
- Hybrid: HOLAP
  - Base cuboid: ROLAP
  - Other cuboids: MOLAP

# Q1: Selection on low-cardinality attributes

TIME
time_key
day
day_of_the_week
month
quarter
year

$S_{\text{region}=\text{"Europe"}}$

SALES
time_key
customer_key
location_key
-----
units_sold
amount

CUSTOMER
customer_key
customer_name
region
type

LOCATION
location_key
store
street_address
city
state
country
region

JOIN

JOIN

JOIN

- Ignoring the final GROUP BY for now
- Omitting the Product dimension

# Indexing OLAP Data: Bitmap Index

---

## (1) BI on dimension tables

- Index on an attribute (column) with low distinct values
- Each distinct values, v, is associated with a n-bit vector (n = #rows)
  - The  $i$ -th bit is set if the  $i$ -th row of the table has the value v for the indexed column
- Multiple BIs can be efficiently combined to enable optimized scan of the table

**Custom**

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

**BI on Customer.Region**

v	bitmap
Asia	1 0 1 0 0
Europe	0 1 0 0 1
America	0 0 0 1 0

# Indexing OLAP Data: Bitmap Index /2

## (1) Bitmap join index (BI on Fact Table Joined with Dimension tables)

- Conceptually, perform a join, map each dimension value to the bitmap of corresponding fact table rows.

-- ORACLE SYNTAX –

```
CREATE BITMAP INDEX sales_cust_region_bjix
  ON  sales(customer.cust_region)
  FROM sales, customer
 WHERE sales.cust_id = customers.cust_id;
```

# Indexing OLAP Data: Bitmap Index /3

Sales

time	customer	loc	Sale
101	C1	100	1
173	C1	200	2
208	C2	100	3
863	C3	200	5
991	C1	100	8
1001	C2	200	13
1966	C4	100	21
2017	C5	200	34

Customer

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

BI on Sales(Customer.Region)

v	bitmap
Asia	11011000
Europe	00100101
America	00000010

# Q2: Selection on high-cardinality attributes

TIME
time_key
day
day_of_the_week
month
quarter
year



SALES
time_key
customer_key
location_key
-----
units_sold
amount



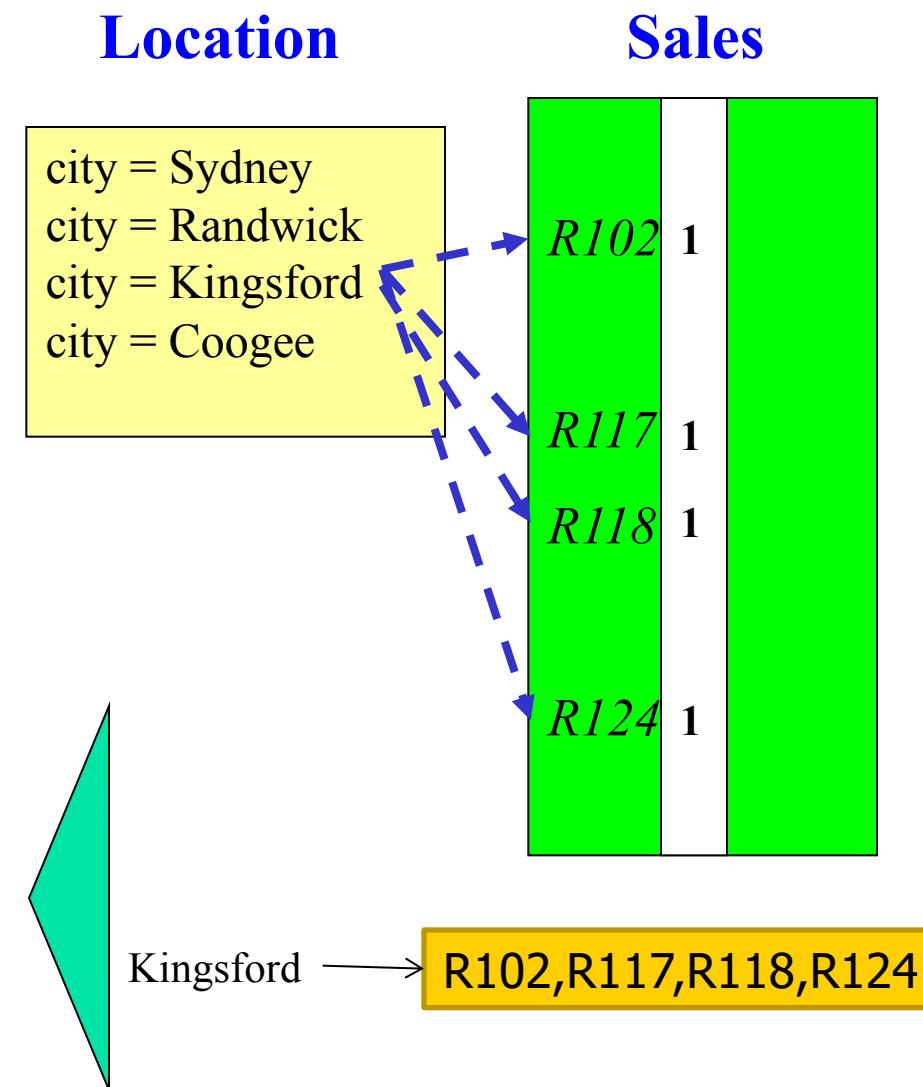
CUSTOMER
customer_key
customer_name
region
type

LOCATION
location_key
store
street_address
city
state
country
region

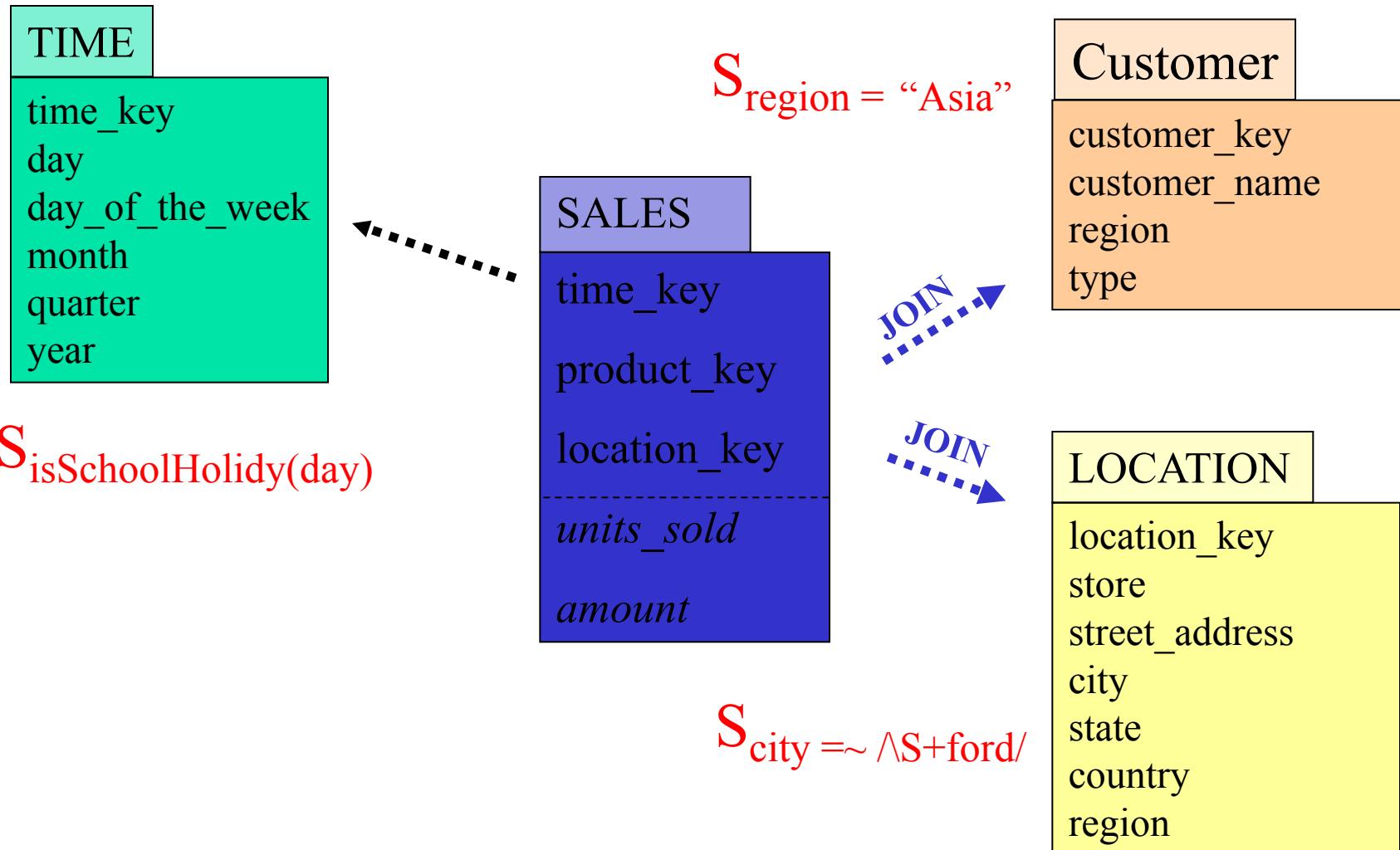
$S_{\text{city}=\text{"Kingsford"}}$

# Indexing OLAP Data: Join Indices

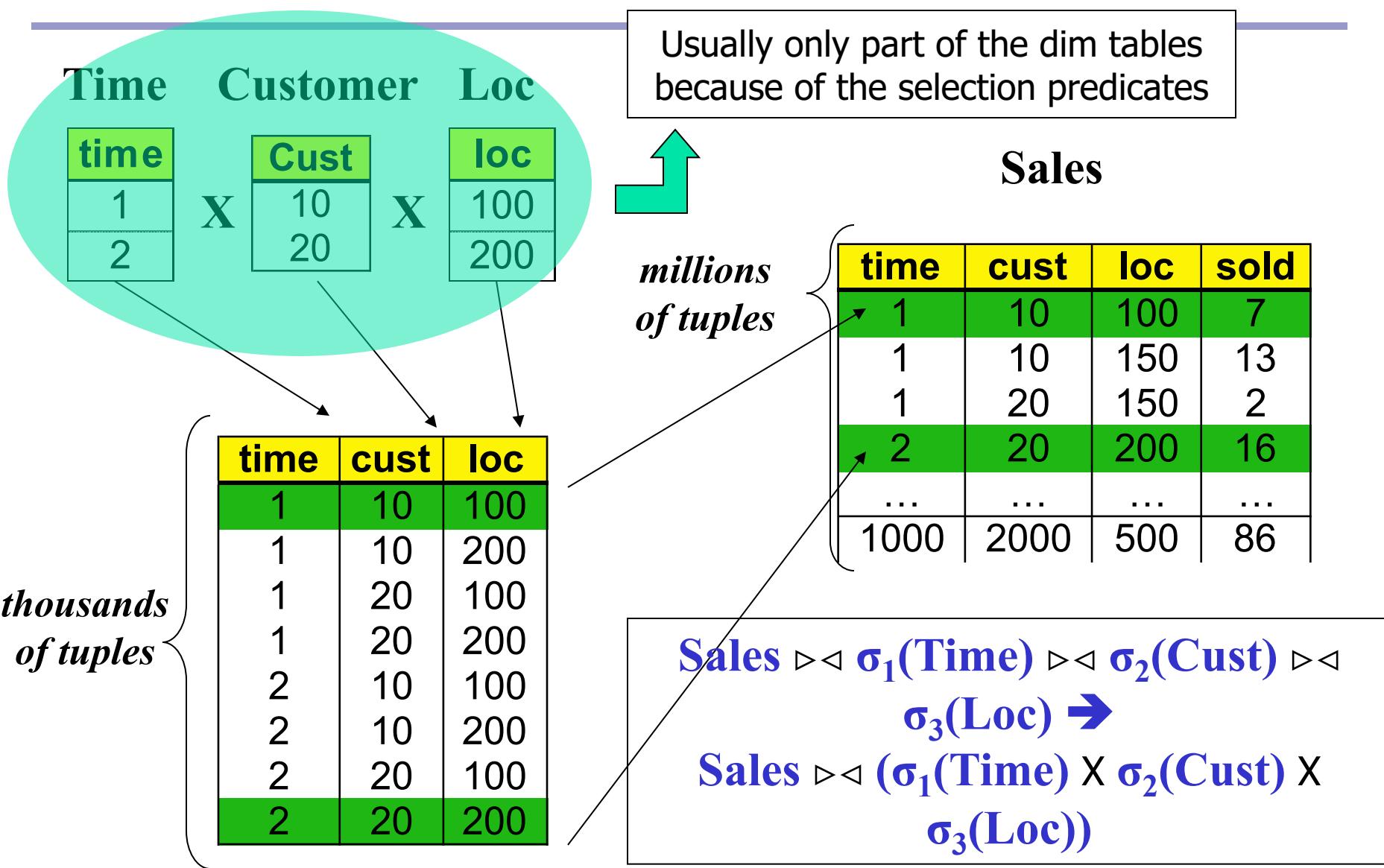
- Join index relates the values of the dimensions of a star schema to rows in the fact table.
  - a join index on *city* maintains for each distinct city a list of ROW-IDs of the tuples recording the sales in the city
- Join indices can span multiple dimensions OR
  - can be implemented as bitmap-indexes (per dimension)
  - use bit-op for multiple-joins



# Q3: Arbitrary selections on Dimensions

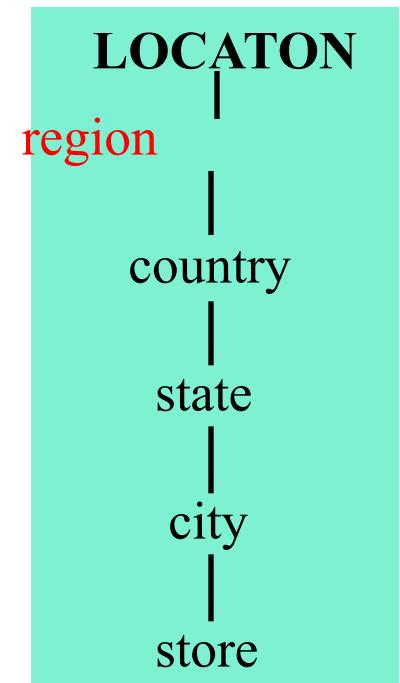


# Star Query and Star Join (Cont.)



# Q4: Coarse-grain Aggregations

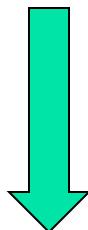
- *“Find total sales per customer type in our stores in Europe”*
  - Join-index will prune  $\frac{3}{4}$  of the data (uniform sales), but the remaining  $\frac{1}{4}$  is still large (several millions transactions)
    - Index is unclustered
- High-level aggregations are expensive!!!!
  - ⇒ Long Query Response Times
  - ⇒ Pre-computation is necessary
  - ⇒ Pre-computation is most beneficial



# Cuboids = GROUP BYs

---

- Multidimensional aggregation = selection on corresponding cuboid

$$\text{GB}_{(\text{type}, \text{city})}(\text{Sales} \bowtie \sigma_1(\text{Time}) \bowtie \sigma_2(\text{Cust}) \bowtie \sigma_3(\text{Loc}))$$


$\sigma_1$  selects some Years,  $\sigma_2$  selects some Brands,  
 $\sigma_3$  selects some Cities,

$$\text{GB}_{(\text{type}, \text{city})}(\sigma_{1'2'3'}, \text{Cuboid}(\text{Year}, \text{Type}, \text{City}))$$

- Materialize some/all of the cuboids
  - A complex decision involving cuboid sizes, query workload, and physical organization

# Two Issues

---

- How to store the materialized cuboids?
- How to compute the cuboids efficiently?

# CUBE BY in ROLAP

Sales		Product				
		1	2	3	4	ALL
Store	1	454	-	-	925	1379
	2	468	800	-	-	1268
	3	296	-	240	-	536
	4	652	-	540	745	1937
	ALL	1870	800	780	1670	5120

4 Group-bys here:  
**(store,product)**  
**(store)**  
**(product)**  
**0**

- Need to write 4 queries!!!
- Compute them independently

Store	Product_key	sum(amount)
1	1	454
1	4	925
2	1	468
2	2	800
3	1	296
3	3	240
4	1	625
4	3	240
4	4	745
1	ALL	1379
2	ALL	1268
3	ALL	536
4	ALL	1937
ALL	1	1870
ALL	2	800
ALL	3	780
ALL	4	1670
ALL	ALL	5120

**SELECT LOCATION.store, SALES.product\_key, SUM (amount)**

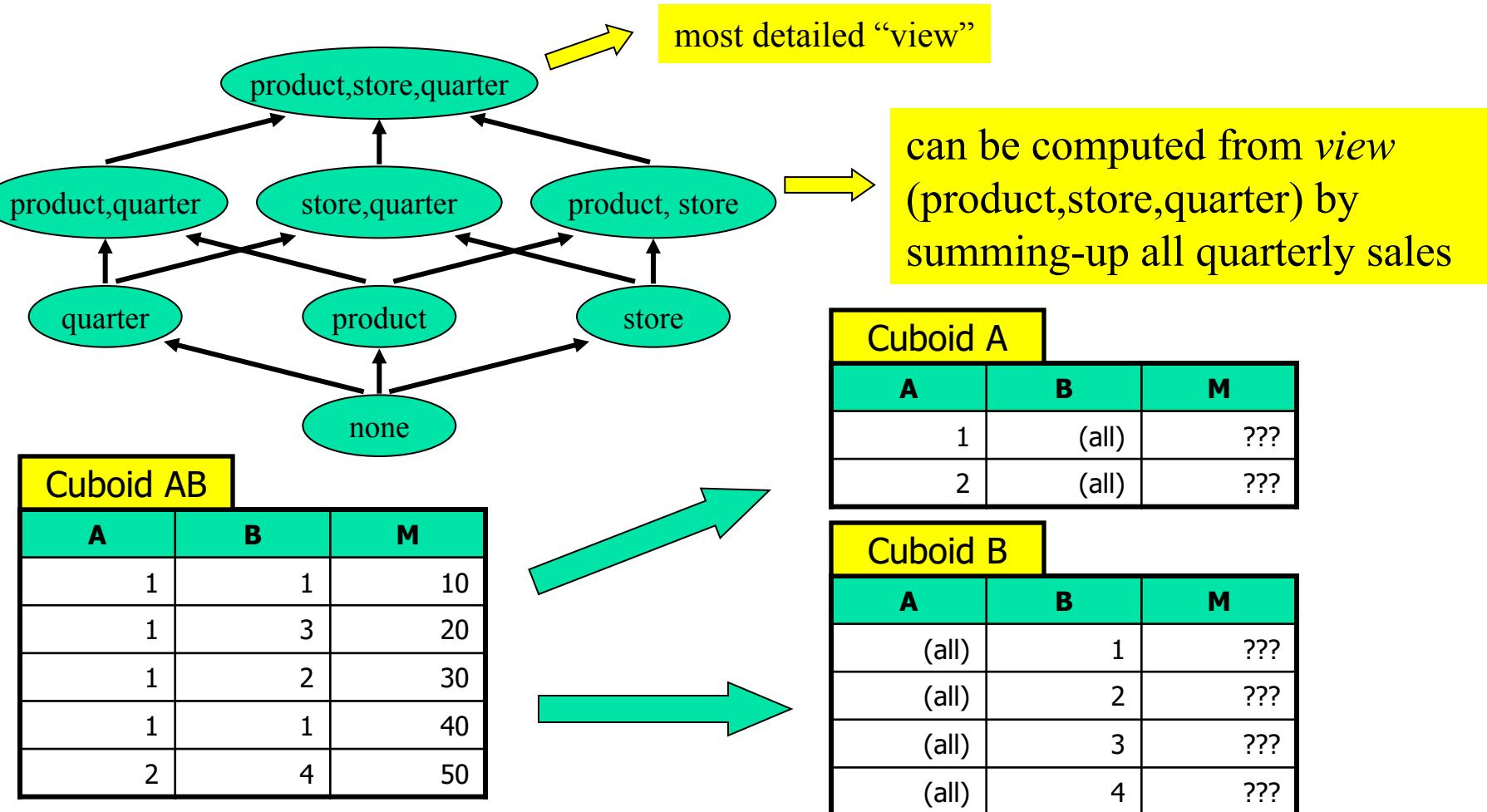
**FROM SALES, LOCATION**

**WHERE SALES.location\_key=LOCATION.location\_key**

**CUBE BY SALES.product\_key, LOCATION.store**

# Top-down Approach

- Model dependencies among the aggregates:



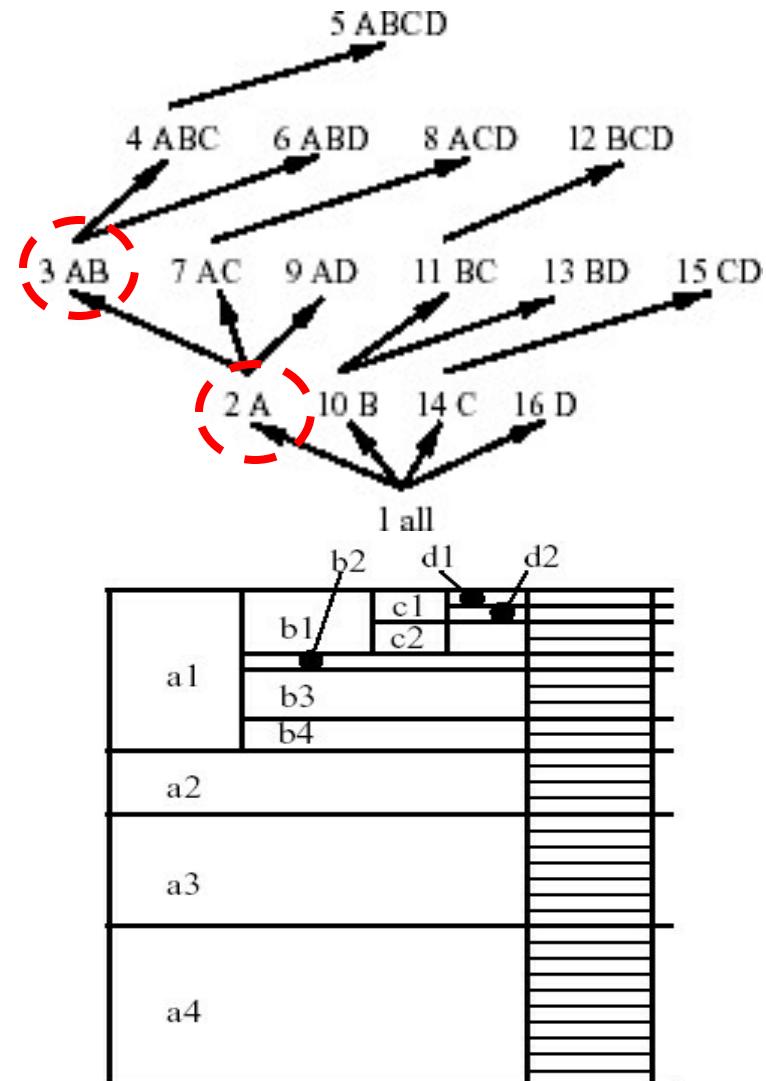
# Bottom-Up Approach (BUC)

- BUC (Beyer & Ramakrishnan, SIGMOD'99)
- Ideas
  - Compute the cube from bottom up
  - Divide-and-conquer
- A simpler recursive version:
  - BUC-SR

A	B	...
1	1	...
1	3	...
1	2	...
1	1	...
2	...	...



■ ■ ■



# Understanding Recursion /1

---

- Powerful computing/problem-solving techniques
- Examples
  - Factorial:
    - $f(n) = 1$ , if  $n = 1$
    - $f(n) = f(n-1) * n$ , if  $n \geq 1$
  - Quick sort:
    - $\text{Sort}([x]) = [x]$
    - $\text{Sort}([x_1, \dots, \textcolor{orange}{pivot}, \dots x_n]) = \text{sort}[\textcolor{blue}{ys}] ++ \text{sort}[\textcolor{red}{zs}], \text{ where}$

$$f(0) = 0! = \\ ???$$

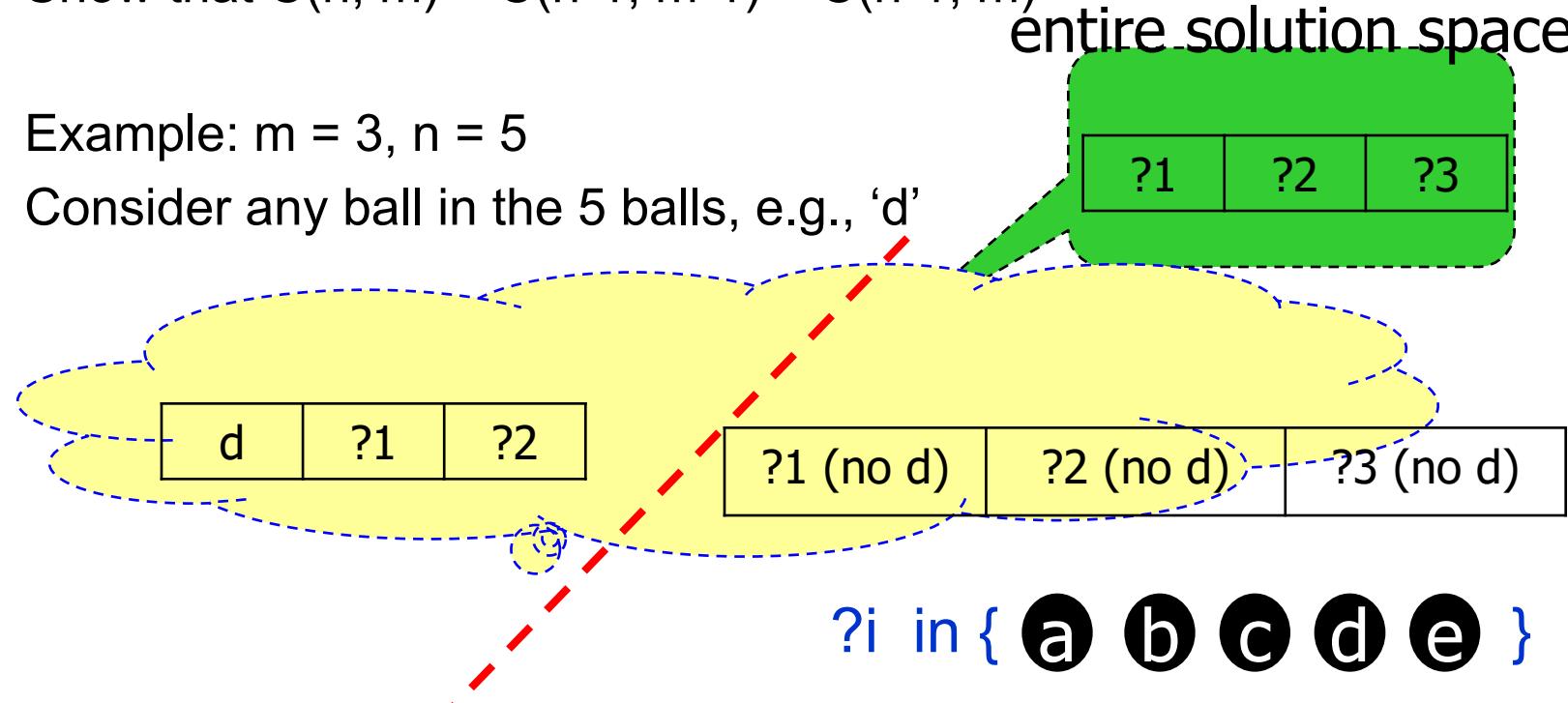
$ys = [ x \mid x \textcolor{blue}{in} xi, x \leq \textcolor{orange}{pivot} ]$

$zs = [ x \mid x \textcolor{blue}{in} xi, x > \textcolor{orange}{pivot} ]$

List comprehension  
in Haskell or  
python

# Understanding Recursion /2

- Let  $C(n, m)$  be the number of ways to select  $m$  balls from  $n$  numbered balls
- Show that  $C(n, m) = C(n-1, m-1) + C(n-1, m)$
- Example:  $m = 3, n = 5$
- Consider any ball in the 5 balls, e.g., 'd'



# Key Points

---

- Sub-problems need to be “**smaller**”, so that a simple/trivial boundary case can be reached
- Divide-and-conquer
  - There may be multiple ways the entire solution space can be divided into **disjoint** sub-spaces, each of which can be conquered **recursively**.

# Geometric Intuition /1

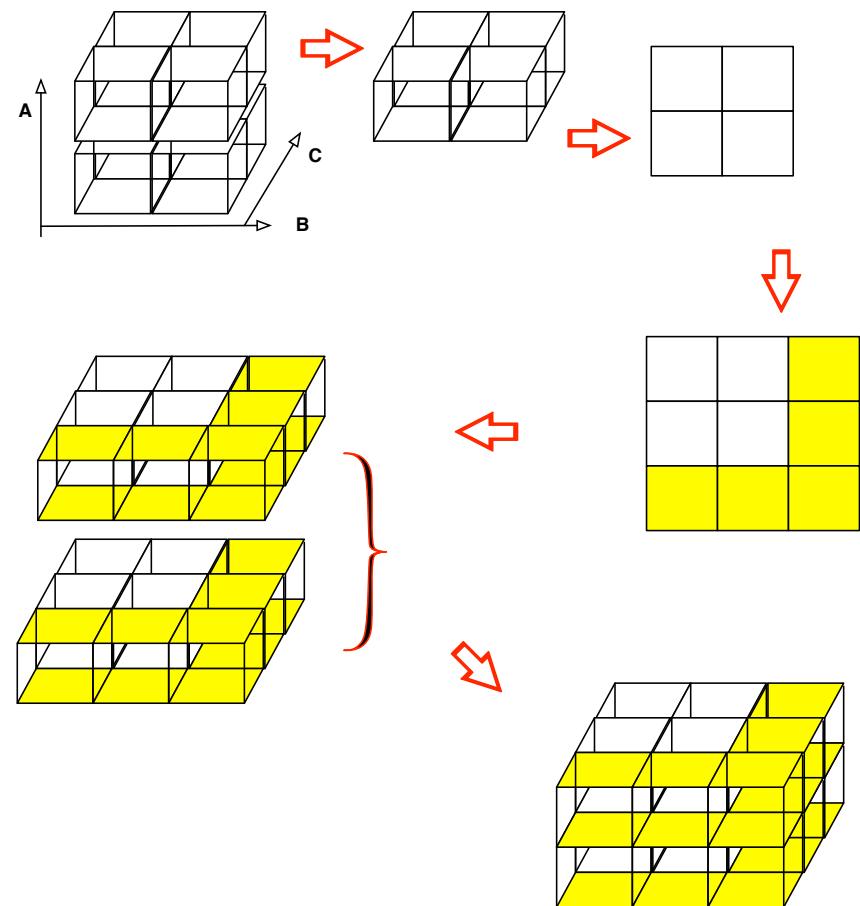
- Reduce Cube(in 2D) to Cube(in 1D)

	b1	b2	b3	
a1	M11	M12	M13	[Step 1]
a2	M21	M22	M23	[Step 1]
	[Step 2]	[Step 2]	[Step 2]	[Step 3]

[a1] ×	b1	b2	b3	
[a2] ×	M11	M12	M13	[Step 1]
[*] ×	M21	M22	M23	[Step 1]
	[Step 2]	[Step 2]	[Step 2]	[Step 3]

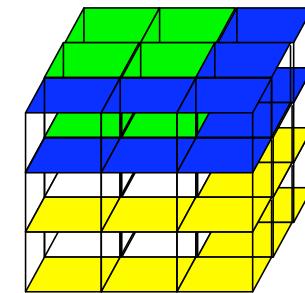
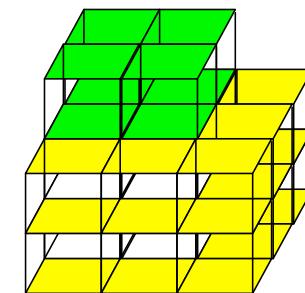
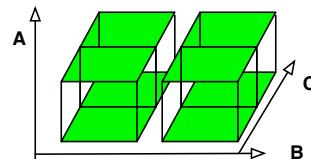
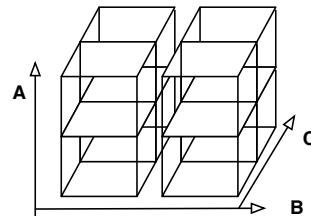
# Geometric Intuition /2

- Reduce Cube(in 3D) to Cube(in 2D)



# Geometric Intuition /3

- Reduce Cube(in 3D) to Cube(in 2D)



# BUC-SR (Simple Recursion)\*

- BUC-SR(data, dims)
  - If (dims is empty)
    - Output (sum(data))
  - Else
    - Dims = [dim1, rest\_of\_dims]
    - For each distinct value  $v$  of dim1
      - slice\_v = slice of data on “dim1 = v”
      - BUC-SR(slice\_v, rest\_of\_dims)
    - data' = Project(data, rest\_of\_dims)
    - BUC-SR(data', rest\_of\_dims)

Boundary case:  
data is essentially a list  
of measure values

General case:  
1)Slice on dim1. Call  
BUC-SR recursively for  
each slice  
  
2)Project out dim1, and  
call BUC-SR on it  
recursively

# Example

	1	2	3	*
1	30	30	40	100
2	50			50
*	80	30	40	150

Input

$\{r_1-r_4\}, B$

B	M
1	30
2	30
3	40
*	100
3	40
2	30
1	20
1	10
B	M

Internal Output

B M

1	30
2	30
3	40
*	100

A B M

1	1	30
1	2	30
1	3	40
1	*	100

A B M

2	1	50
2	*	50

B M

1 50

\*

B M

1 50

\*

B M

1 80

2 30

3 40

\*

150

A B M

*	1	80
*	2	30
*	3	40
*	*	150

r1  
r2  
r3  
r4  
r5

$\{r_1-r_5\}, AB$

A	B	M
1	1	10
1	1	20
1	2	30
1	3	40
2	1	50

$\{r_1'-r_5'\}, B$

B	M
1	80
2	30
3	40
*	150
3	40
2	30
1	20
1	10
B	M

# Try a 3D-Cube by Yourself

---

[{r1-r5}, ABC]

	A	B	C	M
r1	1	1	1	10
r2	1	1	2	20
r3	1	2	1	30
r4	1	3	1	40
r5	2	1	1	50

# MOLAP

---

- (Sparse) array-based multidimensional storage engine
- Pros:
  - small size (esp. for dense cubes)
  - fast in indexing and query processing
- Cons:
  - scalability
  - conversion from relational data

# Multidimensional Array

$$f(\text{time}, \text{item}) = 4 * \text{time} + \text{item}$$

time	item	dollars_sold
Q1	home entertainment	605
Q2	home entertainment	680
Q3	home entertainment	812
Q4	home entertainment	927
Q1	computer	825
Q2	computer	952
Q3	computer	1023
Q4	computer	1038
Q1	phone	14
Q2	phone	31
Q3	phone	30
Q4	phone	38
Q1	security	400
Q2	security	512
Q3	security	501
Q4	security	580



Step 1

Mappings

time	value
Q1	0
Q2	1
Q3	2
Q4	3

item	value
home entertainment	0
computer	1
phone	2
security	3

time	item	dollars_sold	offset
0	0	605	0
1	0	680	4
2	0	812	8
3	0	927	12
0	1	825	1
1	1	952	5
2	1	1023	9
3	1	1038	13
0	2	14	2
1	2	31	6
2	2	30	10
3	2	38	14
0	3	400	3
1	3	512	7
2	3	501	11
3	3	580	15

# Multidimensional Array

Step 3': If **sparse**

Step 3: If **dense**, only need to store sorted slots

offset	dollars_sold
0	605
1	825
2	14
3	400
4	680
5	952
6	31
7	512
8	812
9	1023
10	30
11	501
12	927
13	1038
14	38
15	580



- Think: how to decode a slot?
- Multidimensional array is typically sparse
  - Use sparse array (i.e., offset + value)
  - Could use chunk to further reduce the space
- Space usage:
  - $(d+1)*n*4$  vs  $2*n*4$
- HOLAP:
  - Store all non-base cuboid in MD array
  - Assign a value for ALL

Dense MD array
605
825
14
400
680
952
31
512
812
1023
30
501
927
1038
38
580

# Maths Preliminaries

Wei Wang @ CSE, UNSW

February 27, 2019

# Introduction

- This review serves two purposes:
  - Recap relevant maths contents that you may have learned a long time ago (probably not in a CS course and rarely used in any CS course).
  - More importantly, present it in a way that is useful (i.e., giving semantics/motivations) for understanding maths behind Machine Learning.
- Contents
  - Linear Algebra

# Note

- You've probably learned Linear Algebra from matrix/system of linear equations, etc. We will review key concepts in LA from the perspective of **linear transformations** (think of it as *functions* for now). This perspective provides **semantics and intuition** into most of the ML models and operations.
  - Here we emphasize more on intuitions; We deliberately skip many concepts and present some contents in an informal way.
- It is a great exercise for you to view related maths and ML models/operations in this perspective *throughout* this course!

# A Common Trick in Maths I

## Question

Calculate  $2^{10}$ ,  $2^{-1}$ ,  $2^{\ln 5}$  and  $2^{4-3i}$ ?

- Properties:

- $f_a(n) = f_a(n - 1) * a$ , for  $n \geq 1$ ;  $f_a(0) = 1$ .
- $f(u) * f(v) = f(u + v)$ .
- $f(x) = y \Leftrightarrow \ln(y) = x \ln(a) \Leftrightarrow f(x) = \exp\{x \ln a\}$ .
- $e^{ix} = \cos(x) + i \cdot \sin(x)$ .

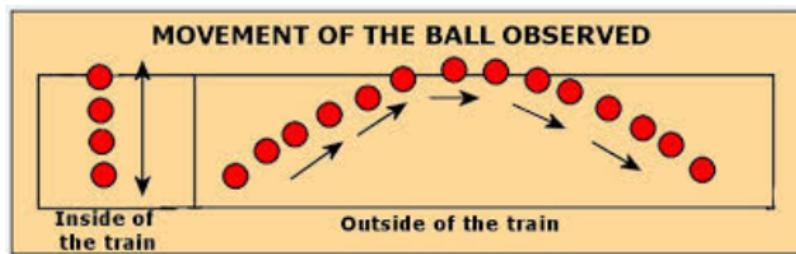
- The trick:

- Same in Linear algebra

# Objects and Their Representations

## Goal

- We need to study the objects
- On one side:
  - A good representation helps (a lot)!
- On the other side:
  - Properties of the objects should be independent of the representation!



# Basic Concepts I

## Algebra

- a set of objects
- two operations and their identity objects (aka. *identity element*):
  - addition (+); its identity is **0**.
  - *scalar* multiplication ( $\cdot$ ); its identity is **1**.
- constraints:
  - Closed for both operations
  - Some nice properties of these operations:
    - Communicative:  $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$ .
    - Associative:  $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$ .
    - Distributive:  $\lambda(\mathbf{a} + \mathbf{b}) = \lambda\mathbf{a} + \lambda\mathbf{b}$ .

# Basic Concepts II

**Think:** *What about subtraction and division?*

## Tips

Always use analogy from algebra on integers ( $\mathbb{Z}$ ) and algebra on Polynomials ( $\mathcal{P}$ ).

## Why these constraints are natural and useful?

# Basic Concepts III

Representation matters?

Consider even geometric vectors:  $\mathbf{c} = \mathbf{a} + \mathbf{b}$

What if we represent vectors by a column of their coordinates?

What if by their polar coordinates?

## Notes

- Informally, the objects we are concerned with in this course are **(column) vectors**.
- The set of all  $n$ -dimensional real vectors is called  $\mathbb{R}^n$ .

# (Column) Vector

## Vector

- A  $n$ -dimensional vector,  $\mathbf{v}$ , is a  $n \times 1$  matrix. We can emphasize its shape by calling it a *column* vector.
  - A *row vector* is a transposed column vector:  $\mathbf{v}^\top$ .

## Operations

- Addition:  $\mathbf{v}_1 + \mathbf{v}_2 =$
- (Scalar) Multiplication:  $\lambda\mathbf{v}_1 =$

# Linearity I

## Linear Combination: Generalization of Univariate Linear Functions

- Let  $\lambda_i \in \mathbb{R}$ , given a set of  $k$  vectors  $\mathbf{v}_i$  ( $i \in [k]$ ), a linear combination of them is

$$\lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 + \dots + \lambda_k \mathbf{v}_k = \sum_{i \in [k]} \lambda_i \mathbf{v}_i$$

- Later, this is just  $\mathbf{V}\lambda$ , where

$$\mathbf{V} = \begin{bmatrix} | & | & | & | \\ v_1 & v_2 & \dots & v_k \\ | & | & | & | \end{bmatrix} \qquad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \dots \\ \lambda_k \end{bmatrix}$$

- Span: All linear combination of a set of vectors is the *span* of them.
- Basis: The minimal set of vectors whose span is exactly the whole  $\mathbb{R}^n$ .

# Linearity II

- Benefit: every vector has a **unique** decomposition into basis.  
**Think: Why uniqueness is desirable?**

## Examples

- Span of  $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$  and  $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$  is  $\mathbb{R}^2$ . They are also the basis.
- Span of  $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ ,  $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$  is  $\mathbb{R}^2$ . But one of them is *redundant*.  
**Think: Who?**
  - Decompose  $\begin{bmatrix} 4 \\ 6 \end{bmatrix}$

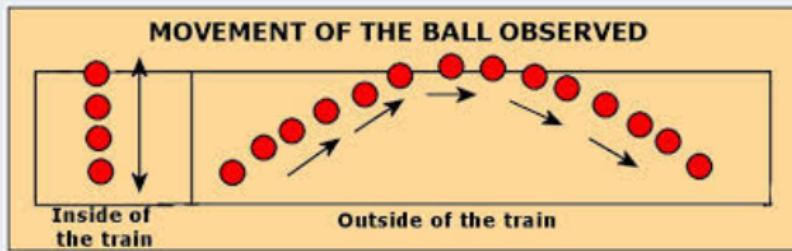
# Linearity III

## Exercises

- What are the (natural) basis of all (univariate) Polynomials of degrees up to  $d$ ?
- Decompose  $3x^2 + 4x - 8$  into *the* linear combination of  $2$ ,  $2x - 3$ ,  $x^2 + 1$ .

$$3x^2 + 4x - 7 = 3(x^2 + 1) + 2(2x - 3) + (-2)(2).$$

- The “same” polynomial is mapped to two different vectors under two different bases. **Think: Any analogy?**



# Matrix I

## Linear Transformation

- is a “nice” linear function that maps a vector in  $\mathbb{R}^n$  to another vector in  $\mathbb{R}^m$ .

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \xrightarrow{f} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

- The general form:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \xrightarrow{f} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \implies \begin{aligned} y_1 &= M_{11}x_1 + M_{12}x_2 \\ y_2 &= M_{21}x_1 + M_{22}x_2 \\ y_3 &= M_{31}x_1 + M_{32}x_2 \end{aligned}$$

# Matrix II

## Nonexample

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \xrightarrow{f} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \implies \begin{aligned} y_1 &= \alpha x_1^2 + \beta x_2 \\ y_2 &= \gamma x_1^2 + \theta x_1 + \tau x_2 \\ y_3 &= \cos(x_1) + e^{x_2} \end{aligned}$$

## Why Only Linear Transformation?

- Simple and nice properties:
  - $(f_1 + f_2)(x) = f_1(x) + f_2(x)$
  - $(\lambda f)(x) = \lambda \cdot f(x)$
  - What about  $f(g(x))$ ?
- Useful



# Matrix I

## Definition

- A  $m \times n$  matrix corresponds to a *linear transformation* from  $\mathbb{R}^n$  to  $\mathbb{R}^m$ 
  - $f(\mathbf{x}) = \mathbf{y} \implies \mathbf{M}\mathbf{x} = \mathbf{y}$ , where matrix-vector multiplication is defined as:  $y_i = \sum_k M_{ik} \cdot x_k$
  - $\mathbf{M}_{\text{outDim} \times \text{inDim}}$
  - *Transformation* or *Mapping* emphasizes more on the mapping between two sets, rather than the detailed specification of the mapping; the latter is more or less the *elementary* understanding of a *function*. These are all specific instances of *morphism* in category theory.

## Semantic Interpretation

# Matrix II

- Linear combination of columns of  $\mathbf{M}$ :

$$\begin{bmatrix} | & | & | & | \\ M_1 & M_2 & \dots & M_n \\ | & | & | & | \end{bmatrix} \begin{bmatrix} | \\ x \\ | \end{bmatrix} = \begin{bmatrix} | & | & | & | \\ M_1 & M_2 & \dots & M_n \\ | & | & | & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

$$\mathbf{y} = x_1 \mathbf{M}_{\bullet 1} + \dots + x_n \mathbf{M}_{\bullet n}$$

- Example:

$$\begin{bmatrix} 1 & 2 \\ -4 & 9 \\ 25 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 10 \end{bmatrix} = 1 \begin{bmatrix} 1 \\ -4 \\ 25 \end{bmatrix} + 10 \begin{bmatrix} 2 \\ 9 \\ 1 \end{bmatrix} = \begin{bmatrix} 21 \\ 86 \\ 35 \end{bmatrix}$$

# Matrix III

$$\begin{bmatrix} 1 & 2 \\ -4 & 9 \end{bmatrix} \begin{bmatrix} 1 \\ 10 \end{bmatrix} = 1 \begin{bmatrix} 1 \\ -4 \end{bmatrix} + 10 \begin{bmatrix} 2 \\ 9 \end{bmatrix} = \begin{bmatrix} 21 \\ 86 \end{bmatrix}$$

**Think:** What does  $\mathbf{M}$  do for the last example?

- Rotation and scaling
- When  $\mathbf{x}$  is also a matrix,

$$\begin{bmatrix} 1 & 2 \\ -4 & 9 \\ 25 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 10 & 20 \end{bmatrix} = \begin{bmatrix} 21 & 42 \\ 86 & 172 \\ 35 & 70 \end{bmatrix}$$

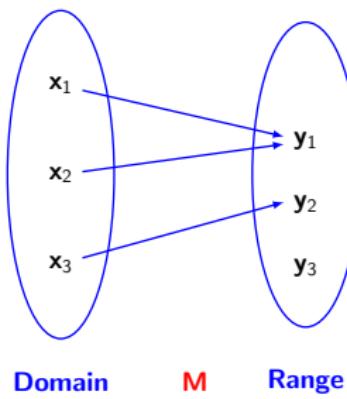
# System of Linear Equations I

$$\begin{aligned}y_1 &= M_{11}x_1 + M_{12}x_2 \\y_2 &= M_{21}x_1 + M_{22}x_2 \\y_3 &= M_{31}x_1 + M_{32}x_2\end{aligned}\implies \begin{bmatrix}y_1 \\ y_2 \\ y_3\end{bmatrix} = \begin{bmatrix}M_{11} & M_{12} \\ M_{21} & M_{22} \\ M_{31} & M_{32}\end{bmatrix} \begin{bmatrix}x_1 \\ x_2\end{bmatrix}$$

$\mathbf{y} = \mathbf{M}\mathbf{x}$

- Interpretation: find a vector in  $\mathbb{R}^2$  such that its image (under  $\mathbf{M}$ ) is exactly the given vector  $\mathbf{y} \in \mathbb{R}^3$ .
- How to solve it?

# System of Linear Equations II



The above transformation is *injective*, but not *surjective*.

# A Matrix Also Specifies a (Generalized) Coordinate System

|

Yet another interpretation

- $\mathbf{y} = \mathbf{M}\mathbf{x} \implies \mathbf{I}\mathbf{y} = \mathbf{M}\mathbf{x}$ .
- The vector  $\mathbf{y}$  wrt standard coordinate system,  $\mathbf{I}$ , is the same as  $\mathbf{x}$  wrt the coordinate system defined by **column** vectors of  $\mathbf{M}$ . **Think: why columns of  $\mathbf{M}$ ?**

# A Matrix Also Specifies a (Generalized) Coordinate System

II

## Example for polynomials

$$\begin{aligned}
 \mathbf{I}: & \begin{array}{l} \text{for } 1 \\ \text{for } x \\ \text{for } x^2 \end{array} \quad \left[ \begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right] \quad \Rightarrow \quad \mathbf{M}: \begin{array}{l} \text{for } 3 \\ \text{for } x-1 \\ \text{for } 2x^2+5x-4 \end{array} \quad \left[ \begin{array}{ccc} 3 & -1 & -4 \\ 0 & 1 & 5 \\ 0 & 0 & 2 \end{array} \right] \\
 & \text{Let } \mathbf{x} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix} \quad \Rightarrow \quad \mathbf{Mx} = \mathbf{I} \begin{bmatrix} -7 \\ 13 \\ 6 \end{bmatrix}
 \end{aligned}$$

# Exercise I

- What if  $\mathbf{y}$  is given in the above example?
- What does the following mean?

$$\begin{bmatrix} 3 & -1 & -4 \\ 0 & 1 & 5 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3 & -1 & -4 \\ 0 & 1 & 5 \\ 0 & 0 & 2 \end{bmatrix}$$

- Think about representing polynomials using the basis:  
 $(x - 1)^2, x^2 - 1, x^2 + 1.$

# Inner Product

THE binary operator – some kind of “similarity”

- Type signature: vector  $\times$  vector  $\rightarrow$  scalar:  $\langle \mathbf{x}, \mathbf{y} \rangle$ .
  - In  $\mathbb{R}^n$ , usually called *dot product*:  $\mathbf{x} \cdot \mathbf{y} \stackrel{\text{def}}{=} \mathbf{x}^\top \mathbf{y} = \sum_i x_i y_i$ .
  - For certain functions,  $\langle f, g \rangle = \int_a^b f(t)g(t) dt$ .  $\Rightarrow$  leads to the Hilbert Space
- Properties / definitions for  $\mathbb{R}$ :
  - conjugate symmetry:  $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle$
  - linearity in the first argument:  $\langle a\mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = a\langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$
  - positive definitiveness:  $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$ ;  $\langle \mathbf{x}, \mathbf{x} \rangle \Leftrightarrow \mathbf{x} = \mathbf{0}$ ;
- Generalizes many geometric concepts to vector spaces: angle (orthogonal), projection, norm
  - $\langle \sin nt, \sin mt \rangle = 0$  within  $[-\pi, \pi]$  ( $m \neq n$ )  $\Rightarrow$  they are orthogonal to each other.
- $\mathbf{C} = \mathbf{A}^\top \mathbf{B}$ :  $C_{ij} = \langle A_i, B_j \rangle$ 
  - Special case:  $\mathbf{A}^\top \mathbf{A}$ .

# Eigenvalues/vectors and Eigen Decomposition

“Eigen” means “characteristic of” (German)

- A (right) **eigenvector** of a square matrix  $\mathbf{A}$  is  $\mathbf{u}$  such that  $\mathbf{Au} = \lambda\mathbf{u}$ .
- Not all matrices have eigenvalues. Here, we only consider “good” matrices. Not all eigenvalues need to be distinct.
- Traditionally, we normalize  $\mathbf{u}$  (such that  $\mathbf{u}^\top \mathbf{u} = 1$ ).
- We can use all eigenvectors of  $\mathbf{A}$  to construct a matrix  $\mathbf{U}$  (as columns). Then  $\mathbf{AU} = \mathbf{U}\Lambda$ , or equivalently,  $\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^{-1}$ . This is the **Eigen Decomposition**.
  - We can interpret  $\mathbf{U}$  as a transformation between two coordinate systems. **Note** that vectors in  $\mathbf{U}$  are not necessarily orthogonal.
  - $\Lambda$  as the scaling on each of the directions in the “new” coordinate system.

# Similar Matrices

## Similar Matrix

- Let  $\mathbf{A}$  and  $\mathbf{B}$  be two  $n \times n$  matrix.  $\mathbf{A}$  is **similar** to  $\mathbf{B}$  (denoted as  $\mathbf{A} \sim \mathbf{B}$ ) if there exists an invertible  $n \times n$  matrix  $\mathbf{P}$  such that  $\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{B}$ .
- Think: What does this mean?**
  - Think of  $\mathbf{P}$  as a *change of basis* transformation.
  - Relationship with the Eigen decomposition.
- Similar matrices have the same value wrt many properties (e.g., rank, trace, eigenvalues, determinant, etc.)

# SVD

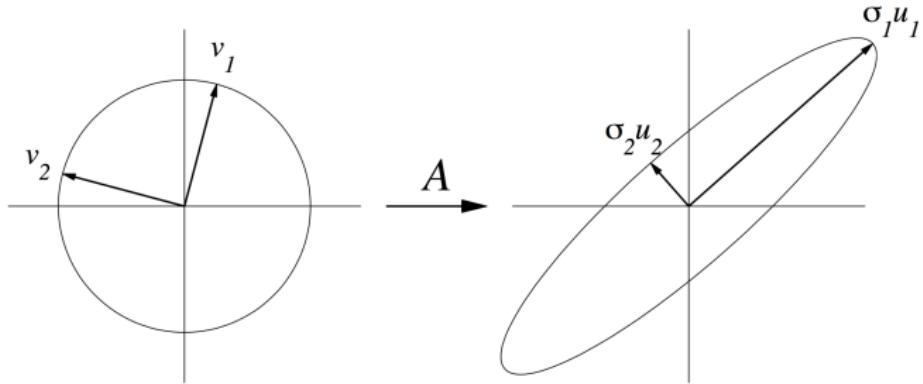
## Singular Vector Decomposition

- Let  $\mathbf{M}$  be  $n \times d$  ( $n \geq d$ ).
- Reduced SVD:  $\mathbf{M} = \hat{\mathbf{U}} \hat{\Sigma} \hat{\mathbf{V}}^\top$  exists for any  $\mathbf{M}$ , such that
  - $\hat{\Sigma}$  is a diagonal matrix with diagonal elements  $\sigma_i$  (called *singular values*) in decreasing order
  - $\hat{\mathbf{U}}$  consists of an (incomplete) set of basis vectors  $\mathbf{u}_i$  (*left singular vectors* in  $\mathbb{R}^n$ ) ( $n \times d$ : original space as  $\mathbf{M}$ )
  - $\hat{\mathbf{V}}$  consists of a set of basis vectors  $\mathbf{v}_i$  (*right singular vectors* in  $\mathbb{R}^d$ ) ( $d \times d$ : reduced space)
- Full SVD:  $\mathbf{M} = \mathbf{U} \Sigma \mathbf{V}^\top$ :
  - Add the remaining  $(n - d)$  basis vectors to  $\hat{\mathbf{U}}$  (thus becomes  $n \times n$ ).
  - Add the  $n - d$  rows of  $\mathbf{0}$  to  $\hat{\Lambda}$  (thus becomes  $n \times d$ ).

# Geometric Illustration of SVD

## Geometric Meaning

- $\mathbf{M}v_i = \sigma_i u_i$



# Graphical Illustration of SVD I

Figure: Reduced SVD vs Full SVD

$$\begin{array}{c}
 \boxed{\mathbf{M}} = \boxed{\hat{\mathbf{U}}} \boxed{\hat{\Sigma}} \boxed{\mathbf{V}^T} \\
 (n \times d) \qquad (n \times d) \qquad (d \times d) \qquad (d \times d)
 \end{array}$$
  

$$\begin{array}{c}
 \boxed{\mathbf{M}} = \boxed{\mathbf{U}} \boxed{\Sigma} \boxed{\mathbf{V}^T} \\
 (n \times d) \qquad (n \times n) \qquad (n \times d) \qquad (d \times d)
 \end{array}$$

# Graphical Illustration of SVD II

Meaning:

- Columns of  $\mathbf{U}$  are the basis of  $\mathbb{R}^n$
- Rows of  $\mathbf{V}^\top$  are the basis of  $\mathbb{R}^d$

# SVD Applications I

## Relationship between Singular Values and Eigenvalues

- What are the eigenvalues of  $\mathbf{M}^\top \mathbf{M}$ ?
- Hint:  $\mathbf{M} = \mathbf{U} \Sigma \mathbf{V}^\top$  and  $\mathbf{U}$  and  $\mathbf{V}$  are unitary (i.e., rotations)

- Related to *PCA (Principle Component Analysis)*

# References and Further Reading I

- Gaussian Quadrature:

<https://www.youtube.com/watch?v=k-yUdqRXijo>

- Linear Algebra Review and Reference.

<http://cs229.stanford.edu/section/cs229-linalg.pdf>

- Scipy LA tutorial. <https://docs.scipy.org/doc/scipy/reference/tutorial/linalg.html>

- We Recommend a Singular Value Decomposition.

<http://www.ams.org/samplings/feature-column/fcarc-svd>

---

# COMP9318: Data Warehousing and Data Mining

— L3: Data Preprocessing and Data Cleaning —

---

- Why preprocess the data?

# Why Data Preprocessing?

---

- Data in the real world is dirty
  - **incomplete**: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., occupation=""
  - **noisy**: containing errors or outliers
    - e.g., Salary="-10"
  - **inconsistent**: containing discrepancies in codes or names
    - e.g., Age="42" Birthday="03/07/1997"
    - e.g., Was rating "1,2,3", now rating "A, B, C"
    - e.g., discrepancy between duplicate records

# Why Is Data Dirty?

---

- Incomplete data comes from
  - n/a data value when collected
  - different consideration between the time when the data was collected and when it is analyzed.
  - human/hardware/software problems
- Noisy data comes from the process of data
  - collection
  - entry
  - transmission
- Inconsistent data comes from
  - Different data sources
  - Functional dependency violation

# Why Is Data Preprocessing Important?

---

- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
    - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
  - Data warehouse needs consistent integration of quality data
- Data extraction, cleaning, and transformation comprises the **majority** of the work of building a data warehouse. — Bill Inmon
- Also a critical step for data mining.

# Major Tasks in Data Preprocessing

---

- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data integration
  - Integration of multiple databases, data cubes, or files
- Data transformation
  - Normalization and aggregation
- Data reduction
  - Obtains reduced representation in volume but produces the same or similar analytical results
- Data discretization & Data Type Conversion

---

- Data cleaning

# Data Cleaning

---

- Importance
  - “Data cleaning is one of the three biggest problems in data warehousing”—Ralph Kimball
  - “Data cleaning is the number one problem in data warehousing”—DCI survey
- Data cleaning tasks
  - Fill in missing values
  - Identify outliers and smooth out noisy data
  - Correct inconsistent data
  - Resolve redundancy caused by data integration

# Missing Data

---

- Data is not always available
  - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- Missing data may need to be inferred.
  - Many algorithms need a value for all attributes
  - Tuples with missing values may have different true values

# How to Handle Missing Data?

---

- Ignore the tuple: usually done when class label is missing (assuming the tasks in classification—not effective when the percentage of missing values per attribute varies considerably.)
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant : e.g., “unknown”, a new class?!
  - the attribute mean
  - the attribute mean for all samples belonging to the same class: smarter
  - the most probable value: inference-based such as Bayesian formula or decision tree

# Noisy Data

---

- Noise: random error or variance in a measured variable
- Incorrect attribute values may due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - technology limitation
  - inconsistency in naming convention
- Other data problems which requires data cleaning
  - duplicate records
  - incomplete data
  - inconsistent data

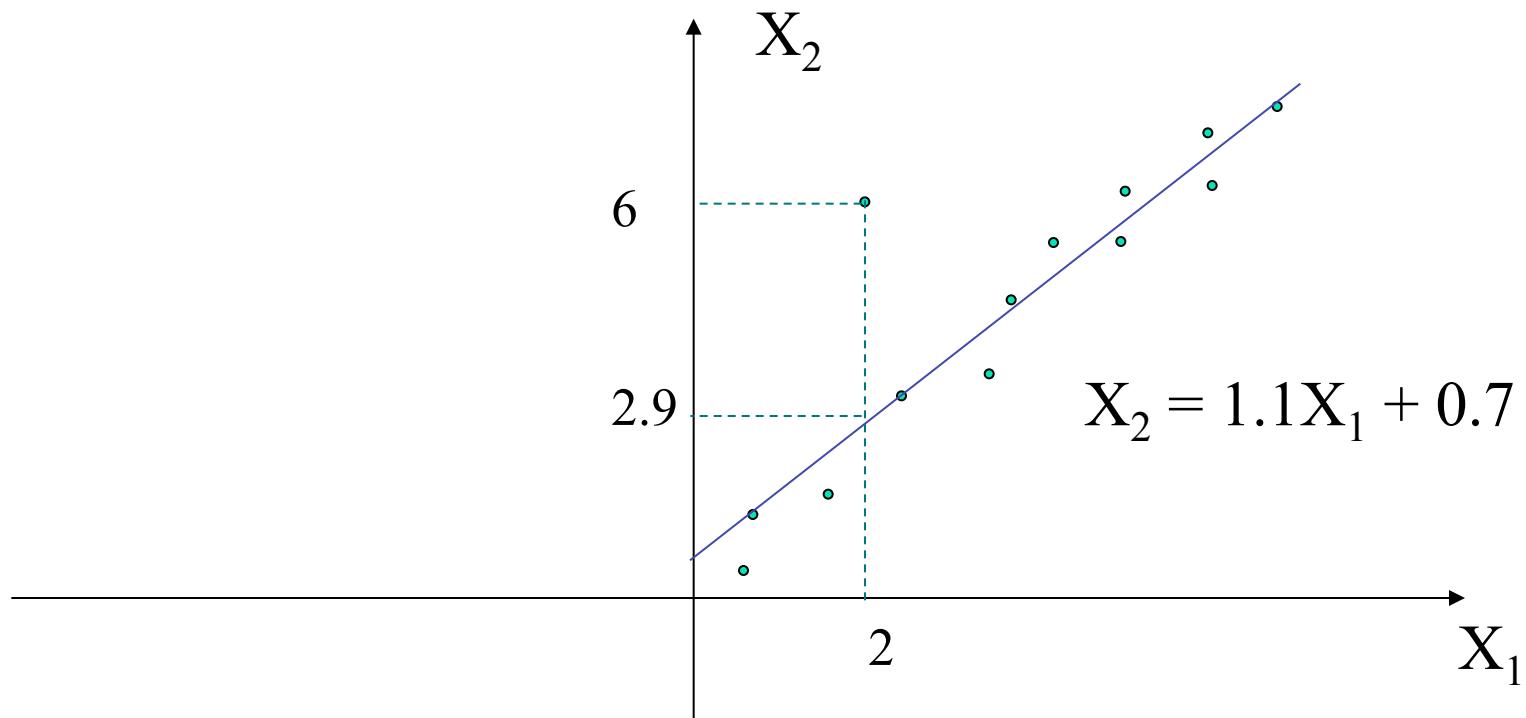
# How to Handle Noisy Data?

To be  
discussed in  
discretization

- Binning method:
  - first sort data and partition into (equi-depth) bins
  - then one can **smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.**
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)
- Regression
  - smooth by fitting the data into regression functions

# Regression

Suburb	#Residents	Usage	Charge
Kingsford	2	1502	3047
Kensington	3	987	265.6
Maroubra	1	568	198.3
...	...	...	...



- 
- Data integration and transformation

# Data Integration

---

- Data integration:
  - combines data from multiple sources into a coherent store
- Schema integration
  - integrate metadata from different sources
  - Entity identification problem: identify real world entities from multiple data sources, e.g., A.cust-id = B.cust-#
- Detecting and resolving data value conflicts
  - for the same real world entity, attribute values from different sources are different
  - possible reasons: different representations, different scales, e.g., metric vs. British units

# Example

---

- Data source 1:
  - Book(bid, title, isbn)
  - Author(aid, fname, lname, birthdate)
  - Writes(bid, aid, order)
- Data source 2:
  - Book(isbn, title, year, author1, author2, ..., author10)
- Data source 3:
  - Author(name, bornInYear, description, book1, book2, ..., book5)

# Handling Redundancy in Data Integration

---

- Redundant data occur often when integration of multiple databases
  - The same attribute may have different names in different databases
  - One attribute may be a “derived” attribute in another table, e.g., annual revenue
- Redundant data may be able to be detected by **correlational analysis**
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

# Data Transformation

---

- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling
- Attribute/feature construction
  - New attributes constructed from the given ones

# Data Transformation: Normalization

- min-max normalization

MinMaxScaler

$$v' = \frac{v - \min_A}{\max_A - \min_A} (new\_max_A - new\_min_A) + new\_min_A$$

- z-score normalization

StandardScaler; sklearn uses biased variance estimate

$$v' = \frac{v - \mu}{\sigma}, \text{ where } \mu \equiv \widehat{\text{mean}}(A) \text{ and } \sigma \equiv \widehat{\text{stddev}}(A)$$

- normalization by decimal scaling

$$v' = \frac{v}{10^j} \quad \text{Where } j \text{ is the smallest integer such that } \max(|v'|) < 1$$

In scikit-learn, they are called Scaling.

Normalization means converting vectors to unit vectors.

---

- Data reduction

# Data Reduction Strategies

---

- Modern datasets may be very large
  - Ratings of millions of customers on millions of items
  - Many ML algorithms have high time and space complexities.
  - Even learned models could be very large.
  - E.g., learned word embeddings (300 dims) for 1M words → at least 1.2GB memory
- Data reduction
  - Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results
- **Data reduction strategies**
  - Dimensionality reduction—remove unimportant attributes
  - Data Compression
  - Numerosity reduction—fit data into models
  - Discretization and concept hierarchy generation

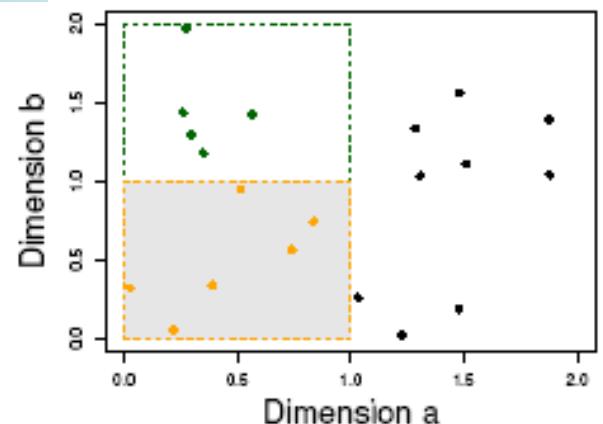
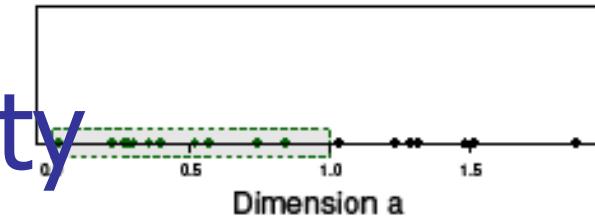
# High-dimensional Features

---

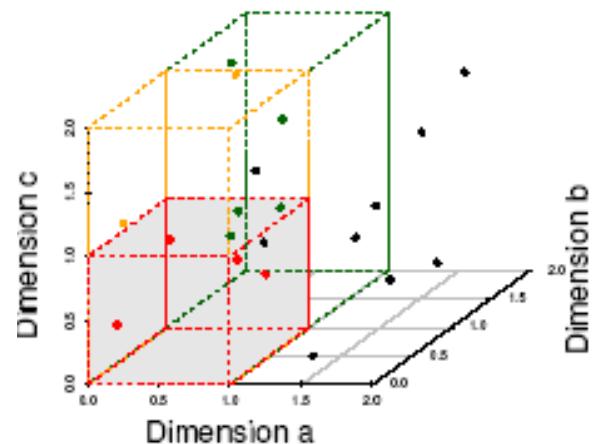
- It is common for many datasets to contain many features
  - More is better at data capturing/creation
    - 561 features for human activity recognition using smartphone dataset:  
<https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones>
    - GIST: 128 dimensional feature
  - Mandated by some model
    - A document is converted into a high-dimensional feature vector. #dims = |vocabulary|

# The Curse of Dimensionality

- Data in only one dimension is relatively packed
- Adding a dimension “stretches” the points across that dimension, making them further apart
- Adding more dimensions will make the points further apart—**high dimensional data is extremely sparse → hard to learn**
- **Distance measure tends to become meaningless**



(b) 6 Objects in One Unit Bin

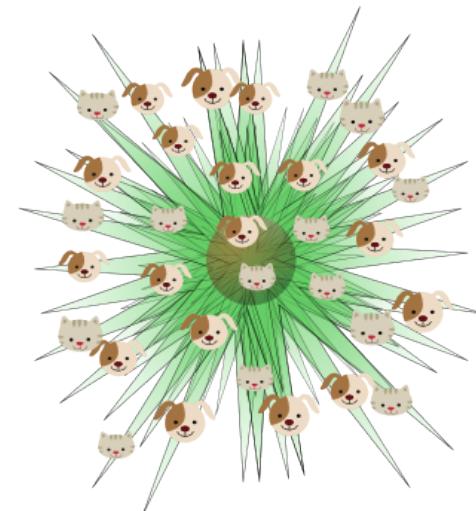
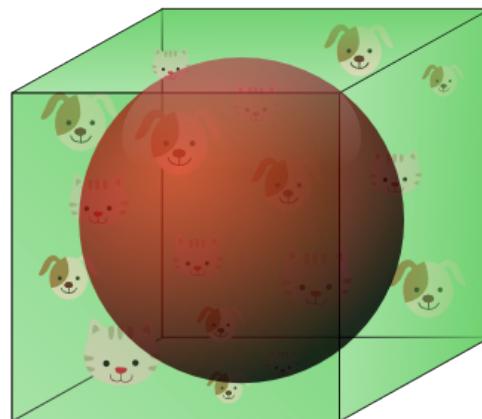
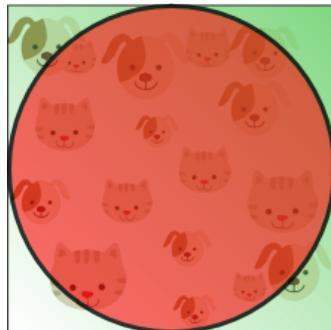


(c) 4 Objects in One Unit Bin

(graphs from Parsons et al. KDD Explorations 2004)

# High-dimensional space

- High-dimensional space is totally different from low-dimensional space (e.g., 3D)
- Many counter-intuitive facts about the high-dimensional space
  - Two random vectors are almost surely orthogonal
  - Random sample  $n$  points within a unit hypercube → most points are on a thin layer of the surface (annulus)



# Goals

---

- Reduce dimensionality of the data, yet still maintain the meaningfulness of the data

# Dimensionality reduction

---

- Dataset  $\mathbf{X}$  consisting of  $n$  points in a  $d$ -dimensional space
- Data point  $\mathbf{x}_i \in \mathbb{R}^d$  ( $d$ -dimensional real vector):  
$$\mathbf{x}_i = [\mathbf{x}_{i1}, \mathbf{x}_{i2}, \dots, \mathbf{x}_{id}]^T$$
- Dimensionality reduction methods:
  - **Feature selection:** choose a subset of the features
  - **Feature extraction:** create new features by combining new ones

# Feature Selection

---

- **Feature** selection (i.e., attribute subset selection):
  - Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features
  - reduce # of patterns in the patterns, easier to understand
- Heuristic methods (due to exponential # of choices):
  - step-wise forward selection
  - step-wise backward elimination
  - combining forward selection and backward elimination
  - decision-tree induction

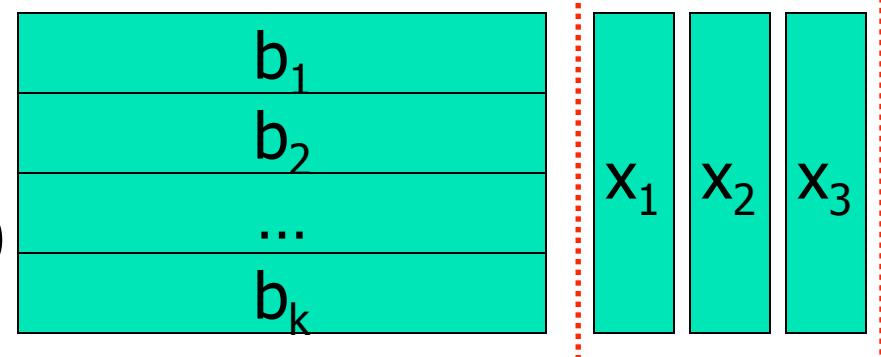
# Heuristic Feature Selection Methods

---

- There are  $2^d$  possible sub-features of  $d$  features
- Several heuristic feature selection methods:
  - Best single features under the **feature independence assumption**: choose by significance tests.
  - Best step-wise feature selection:
    - The best single-feature is picked first
    - Then next best feature condition to the first, ...
  - Step-wise feature elimination:
    - Repeatedly eliminate the worst feature
  - Best combined feature selection and elimination:
  - Optimal branch and bound:
    - Use feature elimination and backtracking

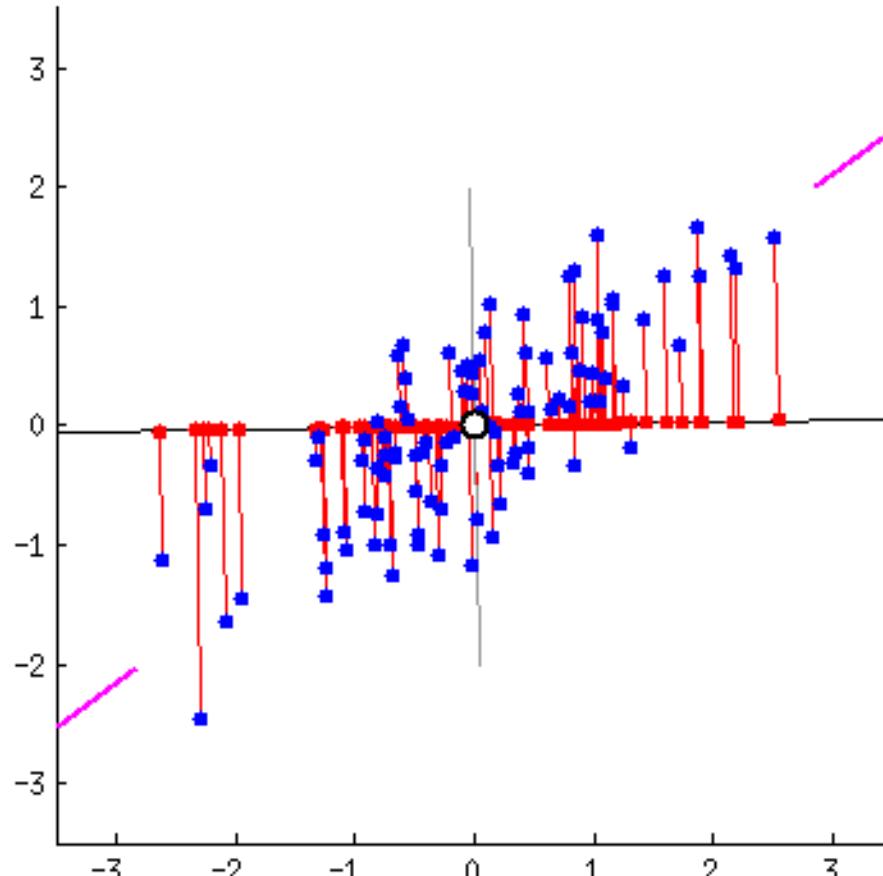
# Principal Component Analysis (PCA)

- Original dataset:  $N$   $d$ -dimensional vectors  $X = \{x_i\}_{i=1..n}$ 
  - Find  $k \leq d$  **orthogonal basis vectors** that can be **best** used to represent data
  - Preserves maximum “information” (i.e., variance under the orthogonal constraint) if projected onto these  $k$  basis vectors
- Reduced data set: Project each  $x_i$  to the  $k$  basis vectors (aka., principal components)
  - $x'_i = [b_1 \dots b_k]^T x_i$
  - $X' = [b_1 \dots b_k]^T X$  (en masse)
- Closed related to Singular Vector Decomposition (**SVD**)



# Projection

- $b^T x$ : projection of  $x$  onto the basis vector  $b$
- What about  $x' = B^T x$ , where  $B$  consists of another set of  $d$ -dim basis vectors?



# JL Lemma

---

- Johnson-Lindenstrauss Flattening Lemma '84:
  - Given  $\varepsilon > 0$ , and an integer  $n$ , let  $k$  be a positive integer such that  $k \geq k_0 = O(\varepsilon^{-2} \log n)$ . For every set  $X$  of  $n$  points in  $\mathbb{R}^d$  there exists  $F: \mathbb{R}^d \rightarrow \mathbb{R}^k$  such that for all  $x_i, x_j$  in  $X$ 
$$\|F(x_i) - F(x_j)\|^2 \in (1 \pm \varepsilon) \|x_i - x_j\|^2$$
- What is the intuitive interpretation of the Lemma?

# Distributional JL Lemma

- Given  $\varepsilon$  in  $(0, \frac{1}{2}]$ ,  $\delta > 0$ , there is a random linear mapping  $F: \mathbb{R}^d \rightarrow \mathbb{R}^k$  with  $k = O(\varepsilon^{-2} \log \delta^{-1})$  such that for any unit vector  $x$  in  $\mathbb{R}^d$ ,

$$\Pr[\|F(x)\|^2 \in 1 \pm \varepsilon] \geq 1 - \delta$$

- Take  $\delta = n^{-2}$ , so  $k = O(\varepsilon^{-2} \log(n))$ , and then for all  $x_i, x_j \in X$ ,

$$\Pr[\|F(x_i) - F(x_j)\|^2 \in (1 \pm \varepsilon) \|x_i - x_j\|^2] \geq 1 - \frac{1}{n^2}$$

- Hence, by a simple union bound, the same statement holds for all  $\binom{n}{2}$  pairs from  $X$  simultaneously with probability at least  $\frac{1}{2}$ .
  - There exists a deterministic linear mapping which is an approximate isometry. ← Why/How?

# Explicit Mapping

- $F(x) = k^{-1/2} * Ux$ , where  $U_{ij} \sim N(0, 1)$ , i.e., i.i.d. samples from the standard Gaussian distribution.

$$F(x) = \begin{matrix} U_{*1} \\ U_{*1} \\ \dots \\ U_{*k} \end{matrix} \quad x = [y_1 | y_2 | \dots | y_k]$$

Quick proof:

$$y_j = \langle U_{*j}, x \rangle = \sum_{i=1}^d x_i U_{ij} \sim \mathcal{N}(0, \|x\|^2)$$

$$\|y\|^2 \sim \|x\|^2 \cdot \chi_k^2 \quad \longrightarrow \quad \begin{aligned} E[\|y\|^2] &= \|x\|^2 \cdot k \\ Var[\|y\|^2] &= 2k \end{aligned}$$

Concentration bound of chisquared distribution:

$$\text{if } z \sim \chi_k^2 \quad \longrightarrow \quad Pr\left[ \left| \frac{z}{k} - 1 \right| < \varepsilon \right] \geq 1 - \exp\left(-\frac{3}{16}k\varepsilon^2\right)$$

# Approximating Inner Product

- 20 news groups
- Origin dim: 5000

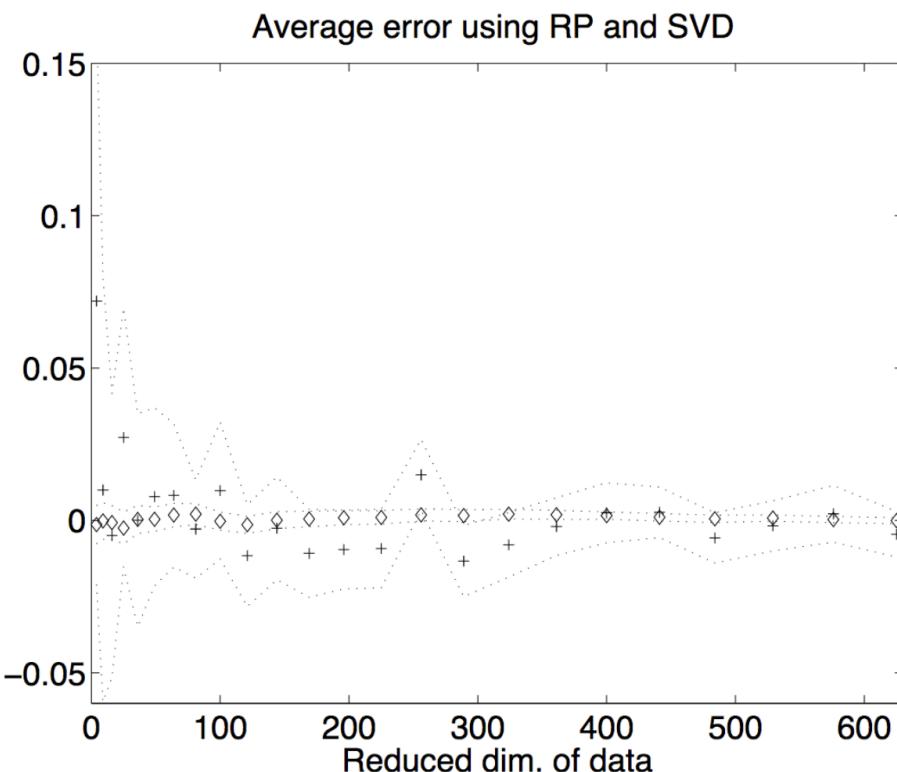
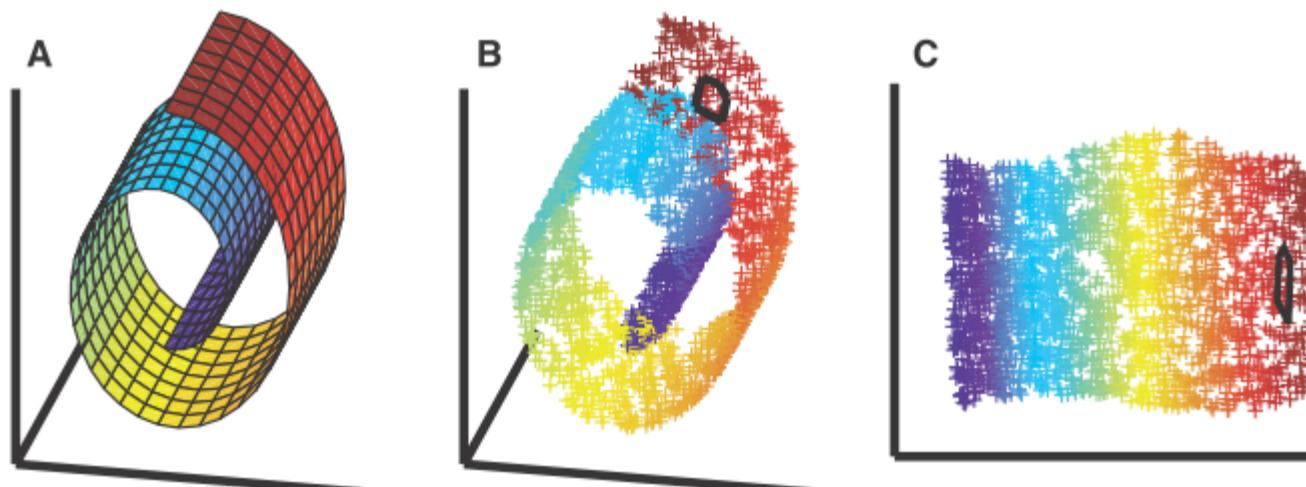


Figure 4: The error produced by RP (+) and SVD (◊) on text document data, with 95% confidence intervals over 100 pairs of document vectors.

# Non-linear Dimensionality Reduction

- There are many advanced **non-linear** dimensionality reduction methods
  - Hypothesis: real high-dimensional data live in a

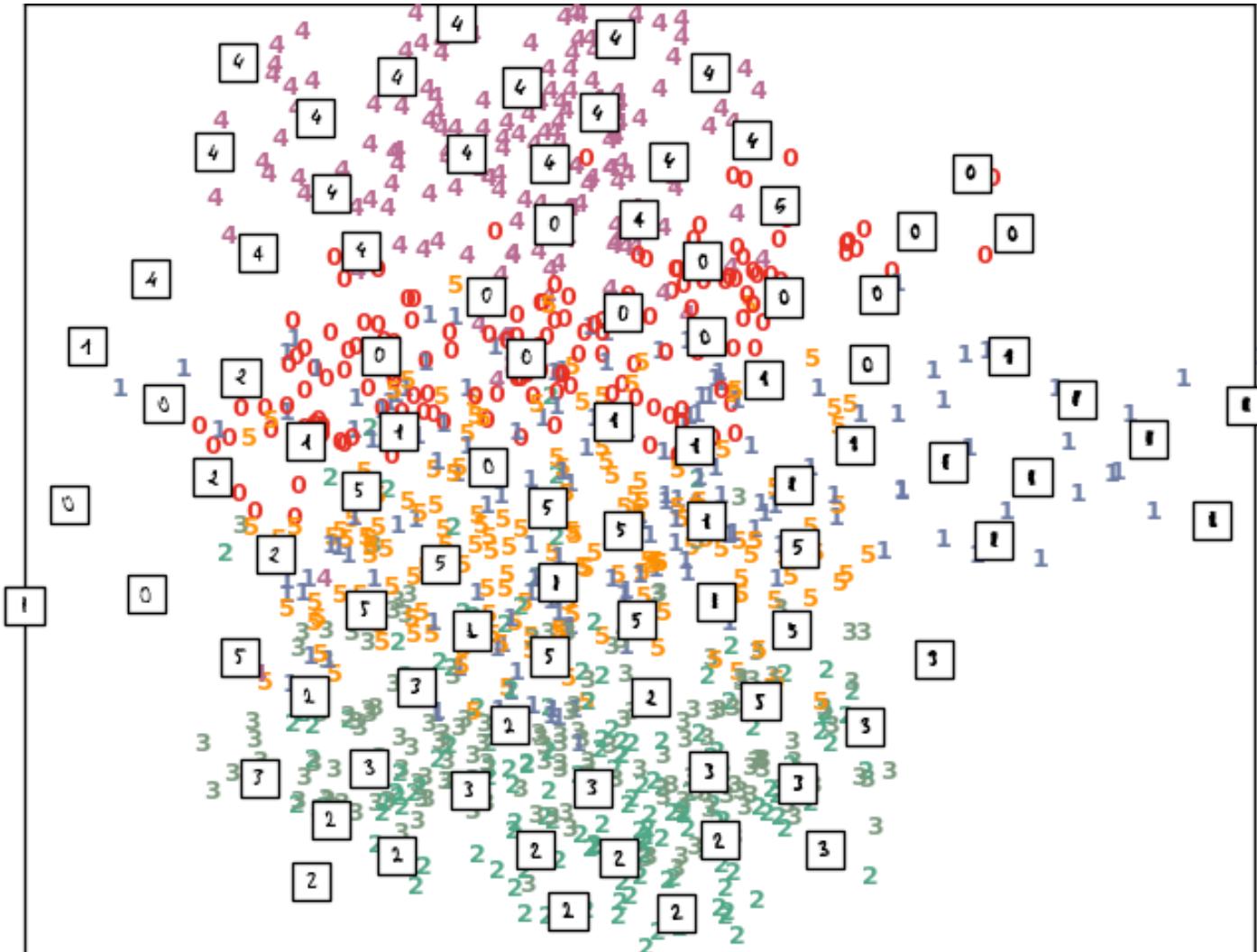


**Fig. 1.** The problem of nonlinear dimensionality reduction, as illustrated (10) for three-dimensional data (B) sampled from a two-dimensional manifold (A). An unsupervised learning algorithm must discover the global internal coordinates of the manifold without signals that explicitly indicate how the data should be embedded in two dimensions. The color coding illustrates the neighborhood-preserving mapping discovered by LLE; black outlines in (B) and (C) show the neighborhood of a single point. Unlike LLE, projections of the data by principal component analysis (PCA) (28) or classical MDS (2) map faraway data points to nearby points in the plane, failing to identify the underlying structure of the manifold. Note that mixture models for local dimensionality reduction (29), which cluster the data and perform PCA within each cluster, do not address the problem considered here: namely, how to map high-dimensional data into a single global coordinate system of lower dimensionality.

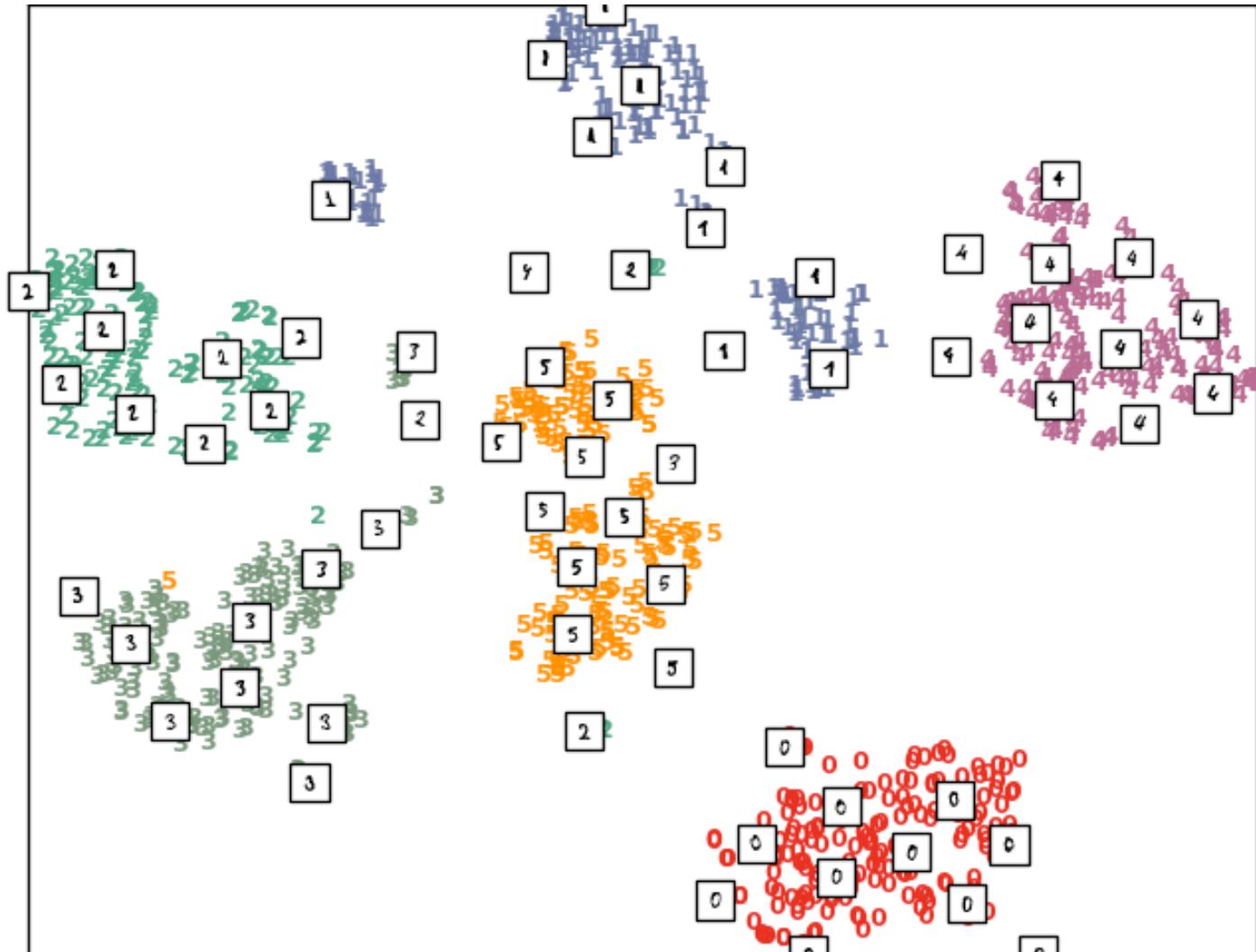
# Digits dataset (d = 64, Class = 0..5)

0	1	2	3	4	5	0	1	2	3	4	5	0	5
5	5	0	4	1	3	5	1	0	0	2	2	2	0
4	4	1	5	0	5	2	2	0	0	1	3	2	1
3	4	4	0	5	3	1	5	4	4	2	2	5	5
2	3	4	5	0	1	2	3	4	5	0	1	2	3
0	4	4	1	3	5	1	0	0	2	2	1	0	1
1	5	0	5	2	2	0	0	1	3	2	1	3	1
0	5	3	4	1	5	4	4	2	2	1	5	5	4
5	0	4	2	3	3	5	0	4	2	3	4	5	0
3	5	1	0	0	2	2	2	0	1	2	3	3	3
5	2	2	0	0	4	3	2	4	3	1	4	3	1
3	1	5	4	4	2	2	2	5	5	4	4	1	5
0	1	2	3	4	5	0	1	2	3	4	5	0	1
5	1	0	0	1	2	2	0	1	2	3	3	3	4
1	2	0	0	1	3	2	1	4	3	1	3	1	4
1	5	4	4	2	2	2	5	5	4	4	0	1	5
2	3	4	5	0	1	2	3	4	5	0	5	5	0
0	0	1	2	2	0	1	2	3	3	3	4	4	1
0	0	1	3	2	1	4	3	1	3	1	4	3	1
4	4	2	2	1	5	5	4	4	0	0	1	2	3

# PCA (time = 0.01s)



# t-SNE (time = 5.69s)



# Data Compression

---

- String compression
  - There are extensive theories and well-tuned algorithms
  - Typically lossless
  - But only limited manipulation is possible without expansion
- Audio/video compression
  - Typically lossy compression, with progressive refinement
  - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
  - Typically short and vary slowly with time

# Numerosity Reduction

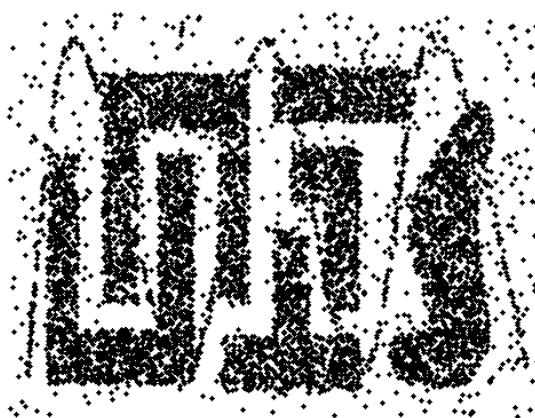
---

- Parametric methods
  - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
  - Log-linear analysis: obtain value at a point in m-D space as the product on appropriate marginal subspaces
- Non-parametric methods
  - Do not assume models
  - Major families: histograms (binning), **clustering**, **sampling**

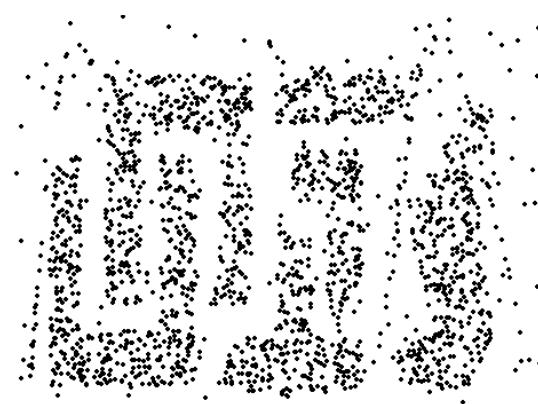
# Random Sampling

---

- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
  - For approximately evaluating models/parameters, etc.
  - Then run the “best” model/parameters on large dataset



8000 points



2000 Points



500 Points

# Other Sampling Methods

---

- Simple random sampling may have very poor performance in the presence of skew
- Adaptive sampling methods
  - Stratified sampling:
    - Approximate the percentage of each class (or subpopulation of interest) in the overall database
    - Used in conjunction with skewed data
- Sketch/synopsis based methods
  - E.g., count-min sketch
    - A simple and versatile data structure to remember the frequency of elements approximately

- 
- Conversion of data types:
    - Discretization
    - Kernel density estimation

# Discretization

---

- Three types of simple attributes:
  - Nominal/categorical — values from an unordered set
    - Profession: clerk, driver, teacher, ...
  - Ordinal — values from an ordered set
    - WAM: HD, D, CR, PASS, FAIL
  - Continuous — real numbers, including Boolean values
- Other types:
  - Array
  - String
  - Objects

# Discrete values → Continuous values

---

- Here we focus on
  - Continuous values → discrete values
    - Removes noise
    - Some ML methods only work with discrete valued features
    - Reduce the number of distinct values on features, which may improve the performance of some ML models
    - Reduce data size
  - Discrete values → continuous values
    - Smooth the distribution
    - Reconstruct probability density distribution from samples, which helps generalization

# Discretization

---

- Discretization
  - reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values
- Methods
  - Binning/Histogram analysis
  - Clustering analysis
  - **Entropy-based discretization**

# Simple Discretization Methods: Binning

---

- **Equal-width** (distance) partitioning:
  - Divides the range into  $N$  intervals of equal size: uniform grid
  - if  $A$  and  $B$  are the lowest and highest values of the attribute, the width of intervals will be:  $W = (B - A)/N$ .
  - The most straightforward, but outliers may dominate presentation
  - Skewed data is not handled well.
- **Equal-depth** (frequency) partitioning:
  - Divides the range into  $N$  intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky.

# Optimal Binning Problem

- After binning, the educated guess or the smoothed value is  $E(x_i)$ , where  $x_i$  are all the values in the same bin
- $\text{cost}(\text{bin}) = \text{SSE}([x_1, \dots, x_m]) = \sum_{i=1}^m (x_i - E(x_i))^2$
- cost of B bins =  $\text{sum}(\text{cost}(\text{bin}_1), \dots, \text{cost}(\text{bin}_B))$
- Problem: find the B-1 bin boundaries such that the cost of the resulting bins is minimized
  - $\text{Alg}(\{x_1, \dots, x_n\}, B)$
  - Optimal Binning: Solve the problem optimally in  $O(B*n^2)$  time and  $O(n^2)$  space.
  - MaxDiff: Solve the problem heuristically in  $O(n*\log(n))$  time and  $O(n)$  space.
  - Note: both algorithms do not sort input data
    - Send in  $\text{sorted}(\{x_1, \dots, x_n\})$  if necessary

# Recursive Formulation

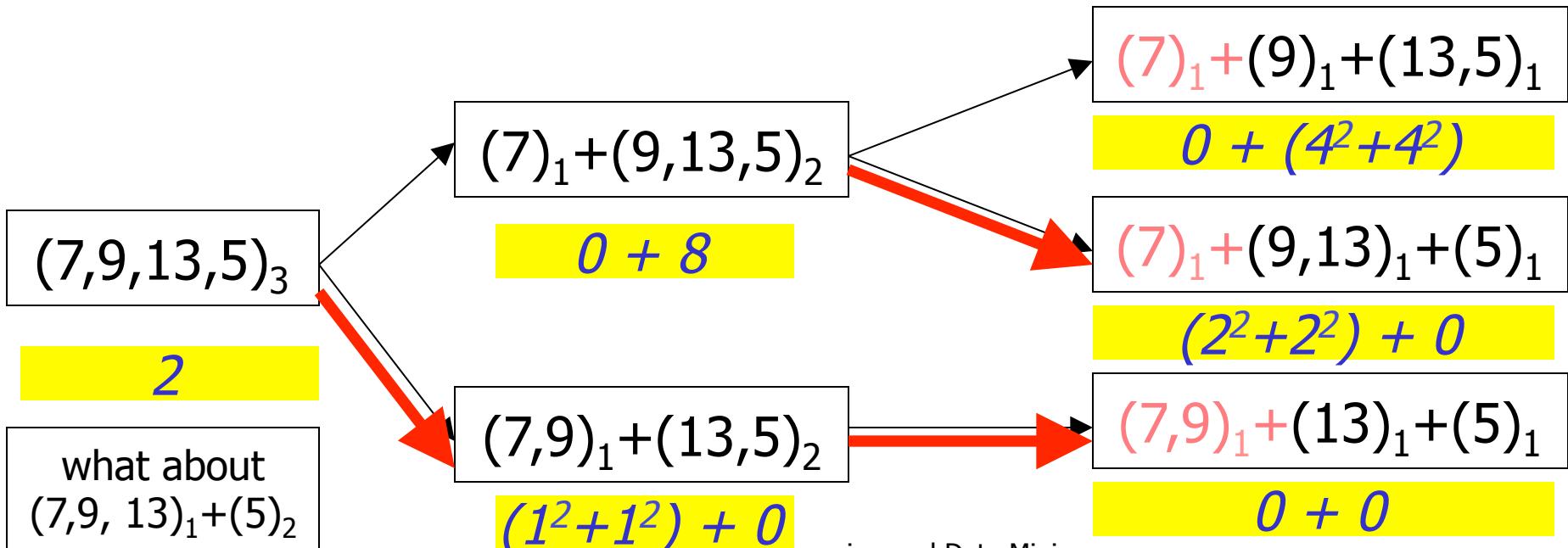
- Observation

$$\text{OPT}(x[1.. n], B) = \min_{i \text{ in } [n]} \{ \text{SSE}(x[1 .. i]) + \text{OPT}(x[i+1 .. n], B-1) \}$$

- Example

- n=4, B=3

x[1]	x[2]	X[3]	x[4]
7	9	13	5



# Problem Caused by Overlapping Sub-problems

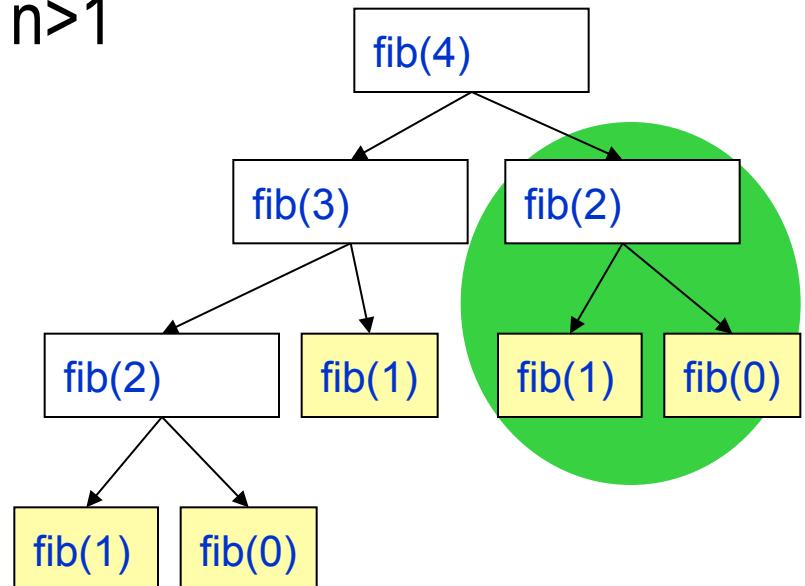
- Consider calculating Fibonacci function

- $\text{fib}(0)=0$
- $\text{fib}(1)=1$

Boundary Condition

- $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$ , for  $n > 1$

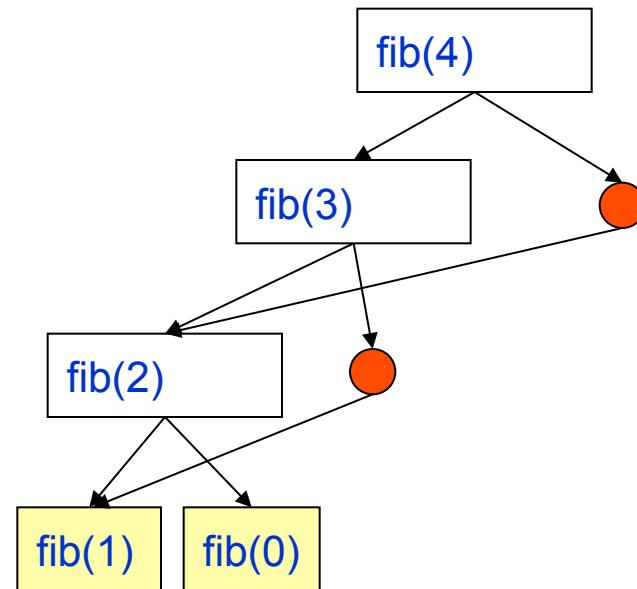
- Naïve D&C implementation is  
in efficient



# Memoization

- Remember solutions of all the sub-problems
- Trade space for time

Sub-problem	solution
$\text{fib}(4)$	3
$\text{fib}(3)$	2
$\text{fib}(2)$	1
$\text{fib}(1)$	1
$\text{fib}(0)$	0

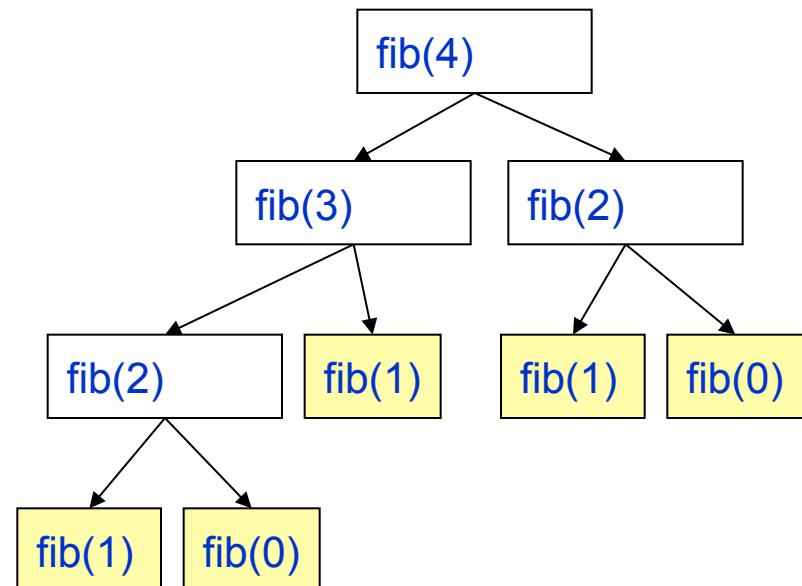


# Dynamic Programming

## Ideas

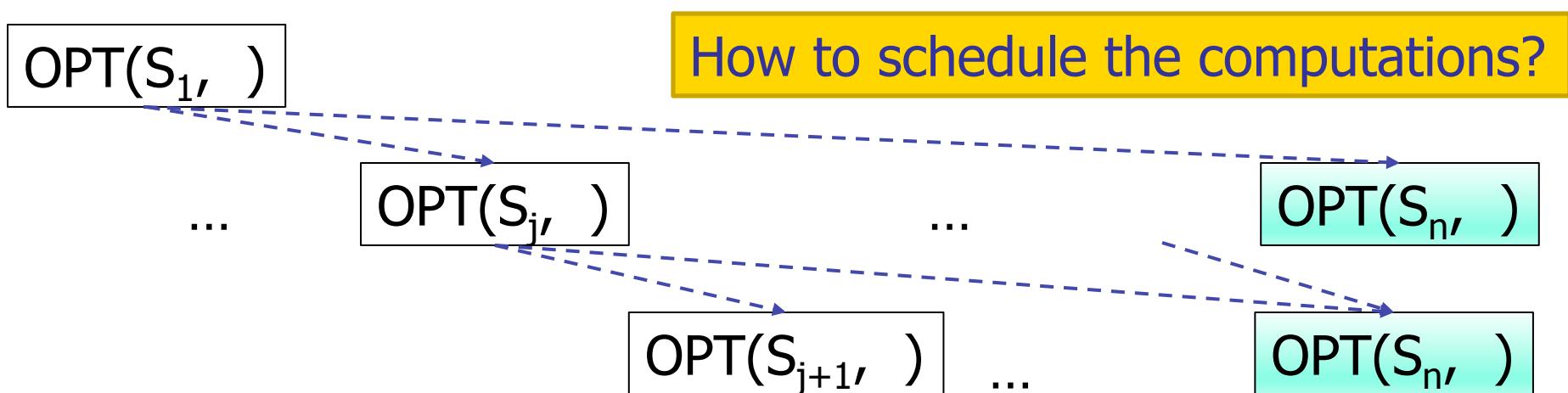
- Ensure all needed recursive calls are already computed and memorized → a good schedule of computation
- (Optional) Reuse space to store previous recursive call results

Sub-problem	solution

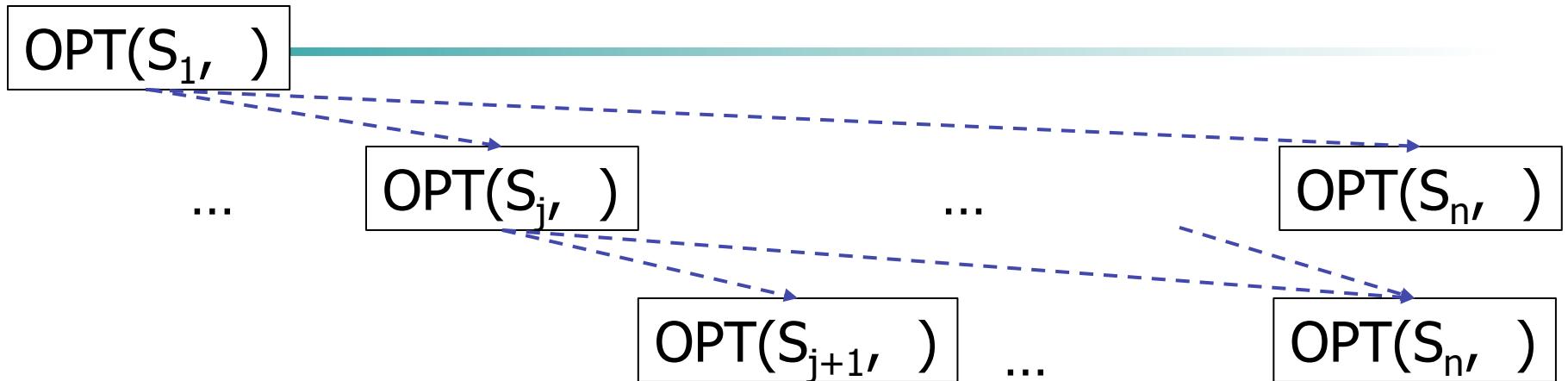


# 2D Dynamic Programming

- $\text{OPT}(x[1.. n], B) = \min_{i \in [n]} \{ \text{SSE}(x[1 .. i]) + \text{OPT}(x[i+1 .. n], B-1) \}$
- $\text{OPT}(S_1, B) = \min_{i \in [n]} \{ \dots + \text{OPT}(S_{i+1}, B-1) \}$
- Goal:



# Pseudocode



# Example

---

- $X = [7, 9, 13, 5]$ ,  $B = 3$

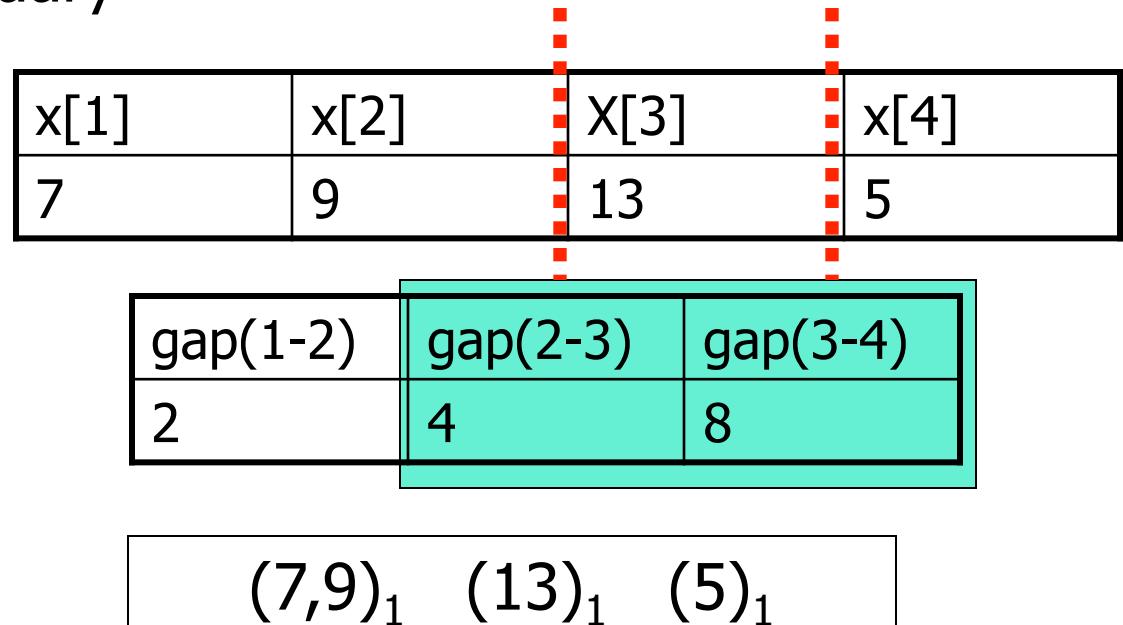
B	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>
1			32	0
2		??	0	-
3	***		-	-

- ( $B=2, S_2$ )
  - What's the problem?
  - How to calculate it?

# MaxDiff

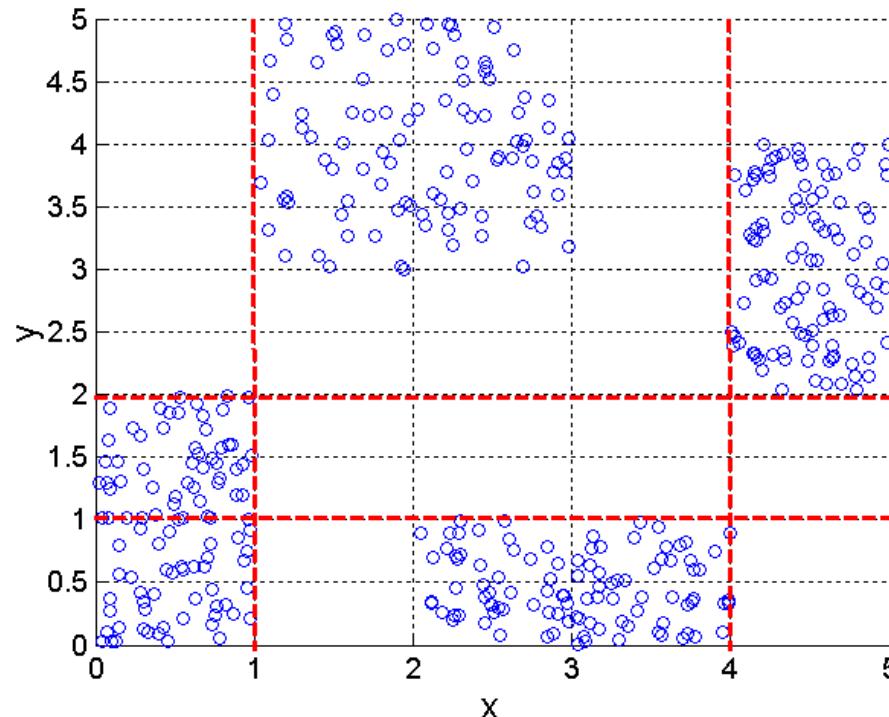
- Complexity of the DP algorithm:
  - $O(n^2*B)$  running time!
- Consider a heuristic method: MaxDiff
  - Idea: use the top-( $B-1$ ) max “gaps” in the data as the bin/bucket boundary
  - Example:

$n=4, B=3$



# Discretization via Clustering

- Can consider multiple attributes together



An example where univariate discretization does not work well

# Supervised Discretization Methods

---

- MDLPC [Fayyad & Irani, 1993]

# Entropy measures uncertainty

- Two classes:

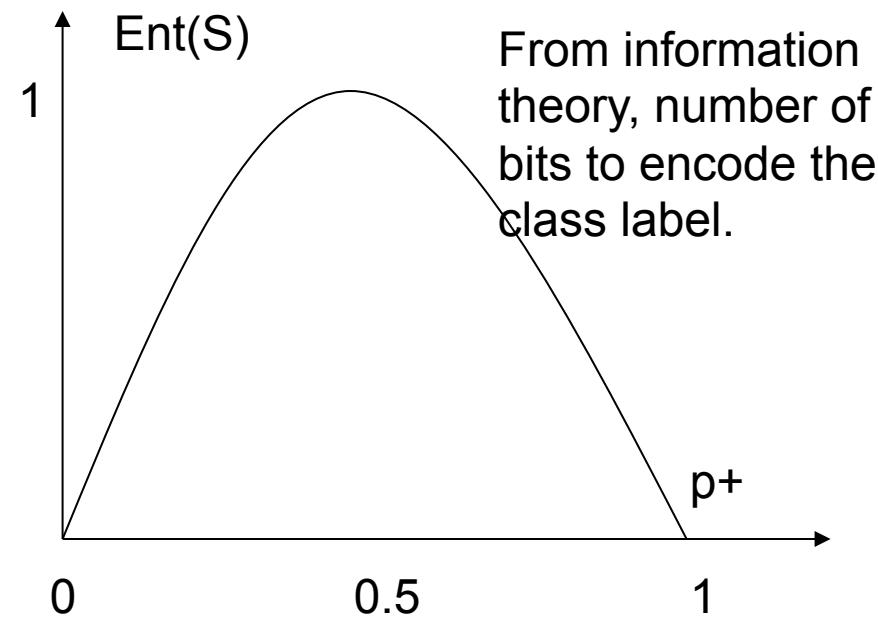
- Give a set  $S$  of instances with binary classes  $\{+,-\}$ .  
Let the proportions of + and - be  $p^+$  and  $p^-$ .
  - $\text{Ent}(S) = - (p^+ \log_2 p^+ + p^- \log_2 p^-)$  (Note:  $\log 0 \equiv 0$ )

- $m$  classes:

$$\text{Ent}(S) = - \sum_{i=1}^m p_i \log p_i$$

- Consider drawing a random sample from  $S$ . What can you tell about its label?

- If  $\text{Ent}(S) = 0$ :
  - If  $\text{Ent}(S) = \log(m)$ :



# Entropy After Splitting T

- Split S into two subsets: S1 and S2.
- What about the label of a random sample given that you know which subset it is drawn from?
  - 
  -
- Define  $E(T; S) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2)$   
 $Gain(T) = Ent(S) - E(T; S)$
- Intuitive meaning of Gain?

# Entropy-Based Discretization: MDLPC

- Given a set of samples  $S$ , if  $S$  is partitioned into two intervals  $S_1$  and  $S_2$  using boundary  $T$ , the entropy after partitioning is
- The boundary that minimizes the entropy function over all possible boundaries is selected as a binary discretization.
- The process is recursively applied to partitions obtained until some stopping criterion is met, e.g.,

$$\text{before } Ent(S) - \text{after } E(T, S) > \delta$$

- Experiments show that it may reduce data size and improve classification accuracy

# Stopping Condition of MDLPC

- Stop when

$$Gain(T; S) < \frac{\log_2(N - 1)}{N} + \frac{\Delta(T; S)}{N}$$

$$\Delta(T; S) = \log_2(3^{|S|} - 2) - |S| \cdot Gain(T; S)$$

# Comments

---

- Understanding the underlying data distribution is important
  - e.g., via visualization
- Supervised methods usually works well
- More advanced methods exist
  
- After learning some ML models, you need to rethink
  - Why discretization?
  - Why different method/parameters affects the model performance?

# Continuous values → Discrete values

---

- After repeating the experiments (e.g., measuring customers arriving 3-4pm), we observed the following random variable  $x_i$  (e.g., #customers):
  - $x_i = 2, 3, 3, 3, 3, 1, 5$
  - What's the probability to see  $x = 3$  in a new experiment? What about  $x = 4$ ?
- Naive estimation
  - $P(x = 3) = 4/7 \quad P(x = 4) = 0 / 7$
- Assume  $x$  follows the Poisson distribution
  - MLE estimation of  $\lambda$
  - MAP estimation with a prior (typically Gamma)
$$P(x; \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$$

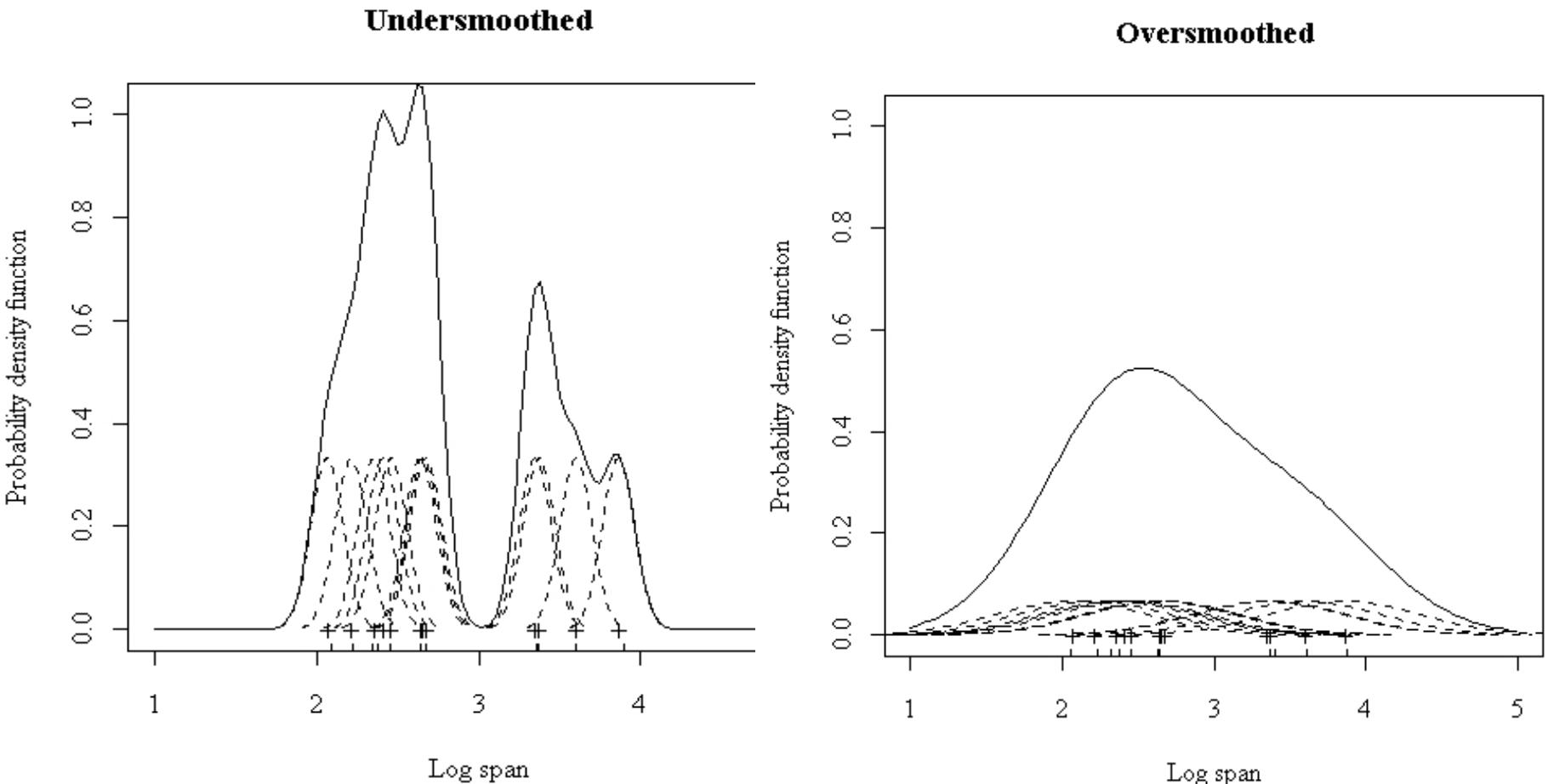
# Non-parametric Estimation

---

- Kernel density estimation (KDE)
  - Let  $\{x_i\}_{i=1:n}$  be  $n$  i.i.d. sample of an unknown  $f(x)$
  - We can estimate  $f(x)$  as 
$$f(x; h) = \frac{1}{n} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$
    - $K(z)$  controls the weight given to  $f(x_i)$  to **influence**  $f(x)$ 
      - May think  $K(x, x_i)$  as measuring their similarity
    - $h$  is the bandwidth parameter
  - Gaussian kernel: 
$$K(x; h) \propto \exp\left(-\frac{x^2}{2h^2}\right)$$

# Impact of $h$

---



# Categorical Values

---

- One hot encoding is widely used in ML
  - Let there be m distinct values for the attribute
  - The i-th (category) value is converted into a m-dimensional binary vector v, where
    - $v_j = 0$ , if  $j \neq i$
    - $v_j = 1$ , otherwise
  - scikit-learn: OneHotEncoder
- Disadvantages:
  - Ignores similarity between values
- Embedding-based methods can learn better (real) vectors

- 
- Case Study
    - c.f., TUN\_datacleansing.ppt

---

# COMP9318: Data Warehousing and Data Mining

— L7: Classification and Prediction —

---

- Problem definition and preliminaries

# Classification vs. 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
- Prediction (aka. Regression):
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical Applications
  - credit approval
  - target marketing
  - medical diagnosis
  - treatment effectiveness analysis

# Classification and Regression

---

- Given a new **object**  $o$ , map it to a **feature vector**  $\mathbf{x} = (x_1, x_2, \dots, x_d)^\top$
- Predict the output (**class label**)  $y \in \mathcal{Y}$ 
  - Binary classification:  $\mathcal{Y} = \{-1, +1\}$  Sometimes,  $\{0, 1\}$
  - Multi-class classification:  $\mathcal{Y} = \{1, 2, \dots, C\}$
- Learn a classification **function**:  $f(\mathbf{x}) : \mathbb{R}^d \mapsto \mathcal{Y}$
- Regression:  $f(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}$

# Examples of Classification Problem

- Text categorization:

**Doc:** Months of campaigning  
and weeks of round-the-clock  
efforts in Iowa all came down to  
a final push Sunday, ...

**Topic:** {  
Politics  
Sport}

- Input object: a document = a sequence of words
- Input features  $\mathbf{x}$  = word frequencies
  - freq(democrats)=2, freq(basketball)= 0, ...
  - $\mathbf{x} = [1, 2, 0, \dots]^T$
- Class label:  $y$ 
  - 'Politics':  $y = +1$
  - 'Sport':  $y = -1$

# Examples of Classification Problem

---

- Image Classification:



Which images are birds,  
which are not?

- Input object: an image = a matrix of RGB values
- Input features  $\mathbf{x}$  = Color histogram
  - $\text{pixel\_count(red)} = 1004$ ,  $\text{pixel\_count(blue)} = 23000$
  - $\mathbf{x} = [1004, 23000, \dots]^T$
- Class label  $y$ 
  - ‘bird image’:  $y = +1$
  - ‘non-bird image’:  $y = -1$

# Supervised Learning

---

- How to find  $f()$ ?
- In supervised learning, we are given a set of **training examples**:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \dots, n\}$$

- Identical independent distribution (i.i.d) assumption
  - A critical assumption for machine learning theory

# Machine Learning Terminologies

- Supervised learning has input labelled data
  - $\# \text{instances} \times \# \text{attributes}$  matrix/table
  - $\# \text{attributes} = \# \text{features} + 1$ 
    - 1 (usu. the last attribute) is for the class attribute
- Labelled data split into 2 or 3 disjoint subsets
  - Training data  $\frac{\# \text{correctly\_classified}}{\# \text{training\_instances}}$  → Build a model
  - Validation/development data → Select/refine the model
  - Testing data  $\frac{\# \text{correctly\_classified}}{\# \text{testing\_instances}}$ 
    - Testing/generalization error =  $1.0 - \frac{\# \text{correctly\_classified}}{\# \text{testing\_instances}}$
- We mainly discuss binary classification here
  - i.e.,  $\# \text{labels} = 2$  → Evaluate the model

- 
- Overview of the whole process (not including cross-validation)

# Classification—A Two-Step Process

---

- **Model construction:** describing a set of predetermined classes
  - 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 classification rules, decision trees, or mathematical formulae
- **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 over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

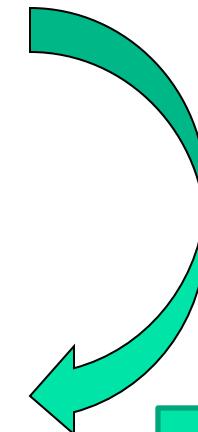
# Classification Process (1): Preprocessing & Feature Engineering

EID	Name	Title	EMP_Date	Track
101	Mike	Assistant Prof	2013-01-01	3-year contract
102	Mary	Assistant Prof	2009-04-15	Continuing
103	Bill	Scientia Professor	2014-02-03	Continuing
110	Jim	Associate Prof	2009-03-14	Continuing
121	Dave	Associate Prof	2009-12-02	1-year contract
234	Anne	Professor	2013-03-21	Future Fellow
188	Sarah	Student Officer	2008-01-17	Continuing

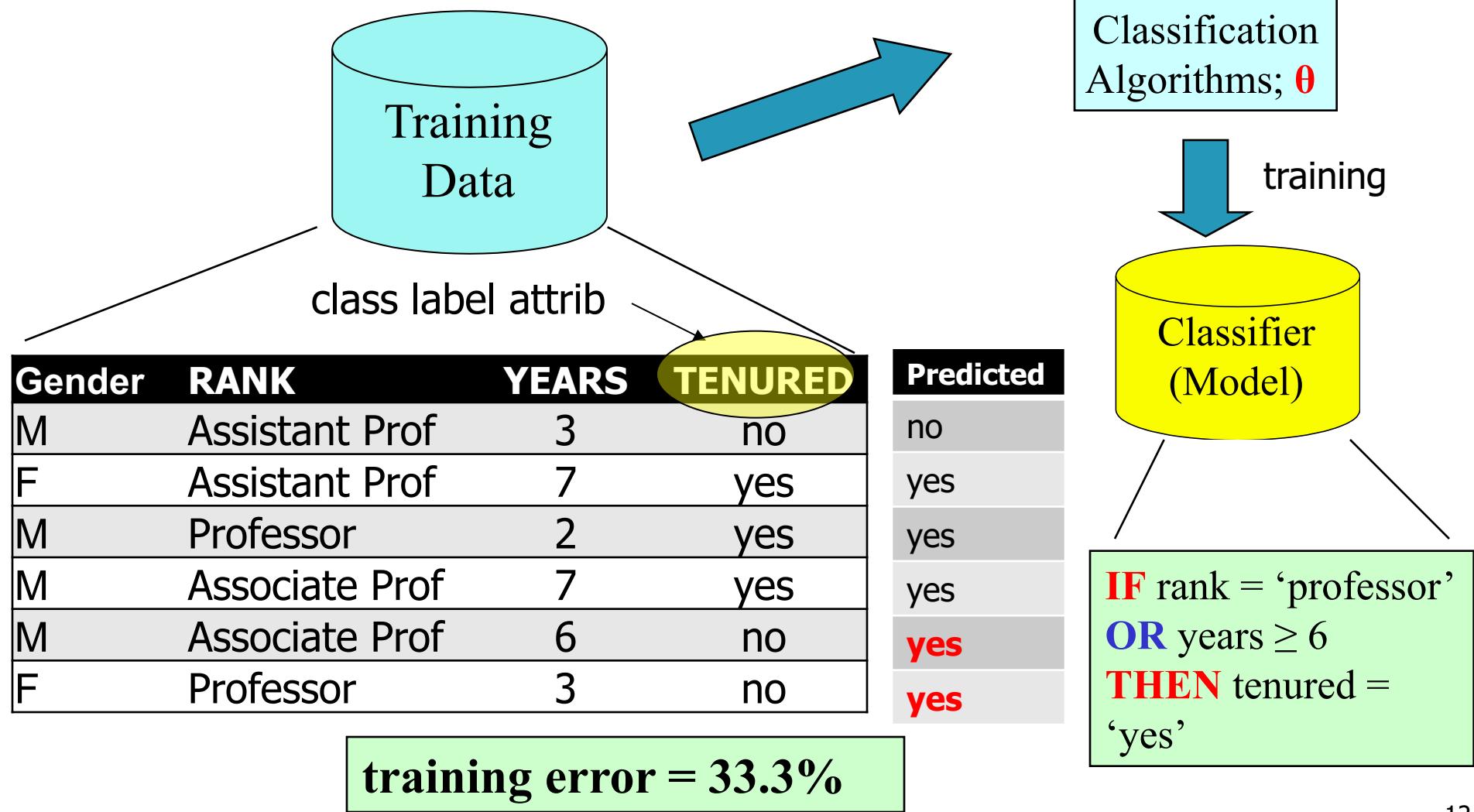
Raw  
Data

Gender	RANK	YEARS	TENURED
M	Assistant Prof	3	no
F	Assistant Prof	7	yes
M	Professor	2	yes
M	Associate Prof	7	yes
M	Associate Prof	6	no
F	Professor	3	no

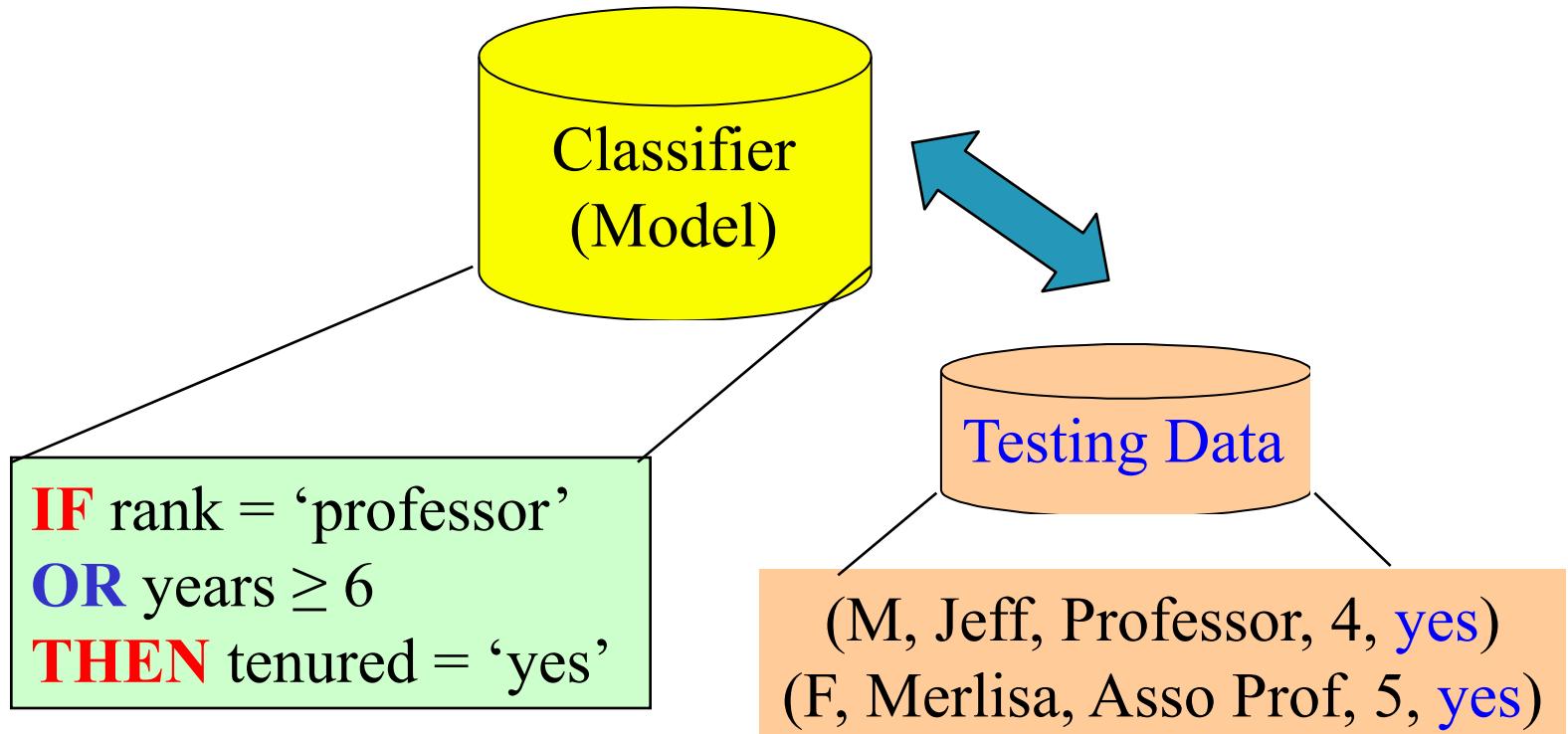
Training  
Data



# Classification Process (2): Training

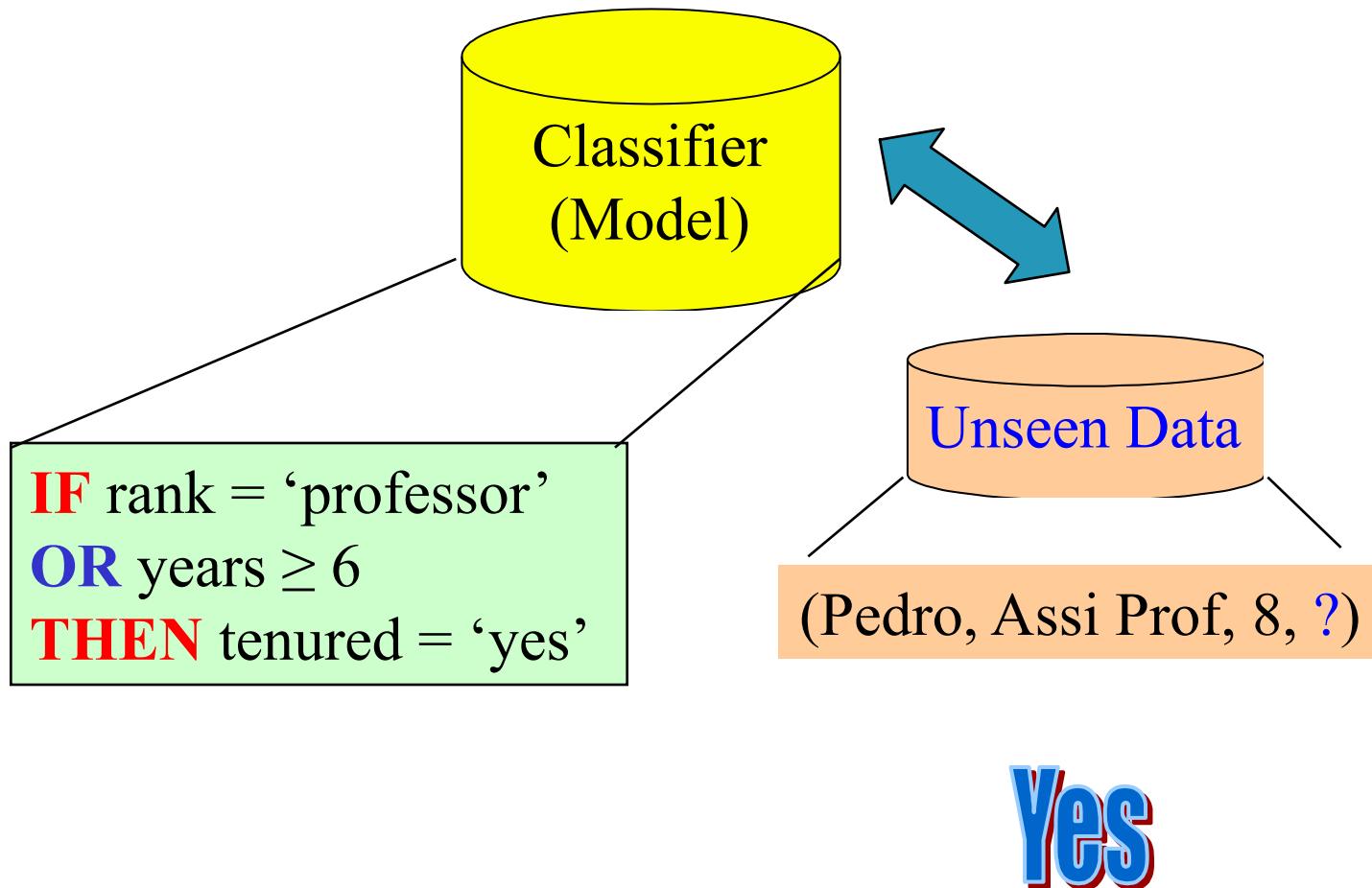


# Classification Process (3): Evaluate the Model on Testing Data



**testing error = 50%**

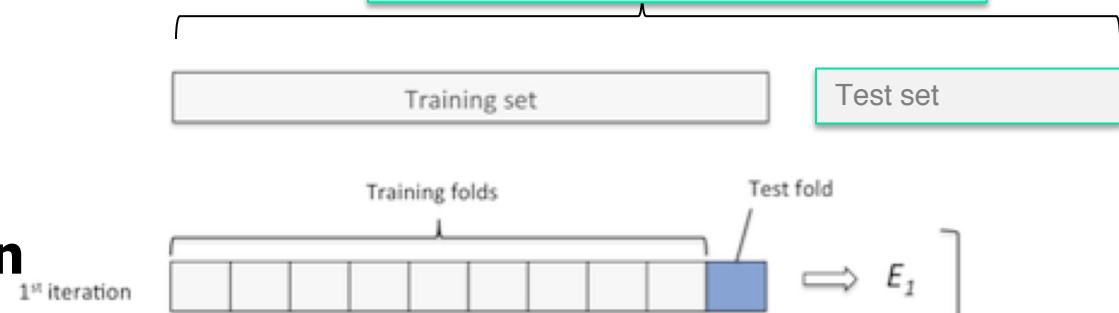
# Classification Process (4): Use the Model in Production



# How to judge a model?

- Based on training error or testing error?
  - Testing error
    - Otherwise, this is a kind of data scooping → **overfitting**
- What if there are multiple models to choose from?
  - Further split a “test/development set” from the training set
- Can we trust the error values on the development set?
  - Need “large” dev set
  - → less data for training
  - **k-fold cross-validation**
  - k=n: leave-one-out

All the labelled data



10-fold CV

# Exercise: Problem definition and Feature Engineering

---

- How to formulate the following into ML problems?  
What kind of resources do you need? What are the features you think may be most relevant?
  1. Predict the sale trend of a particular product in the next month
  2. Design an algorithm to produce better top-10 ranking results for queries

# Supervised vs. Unsupervised Learning

---

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

- 
- Decision Tree classifier
    - ID3
    - Other variants

# Training Dataset

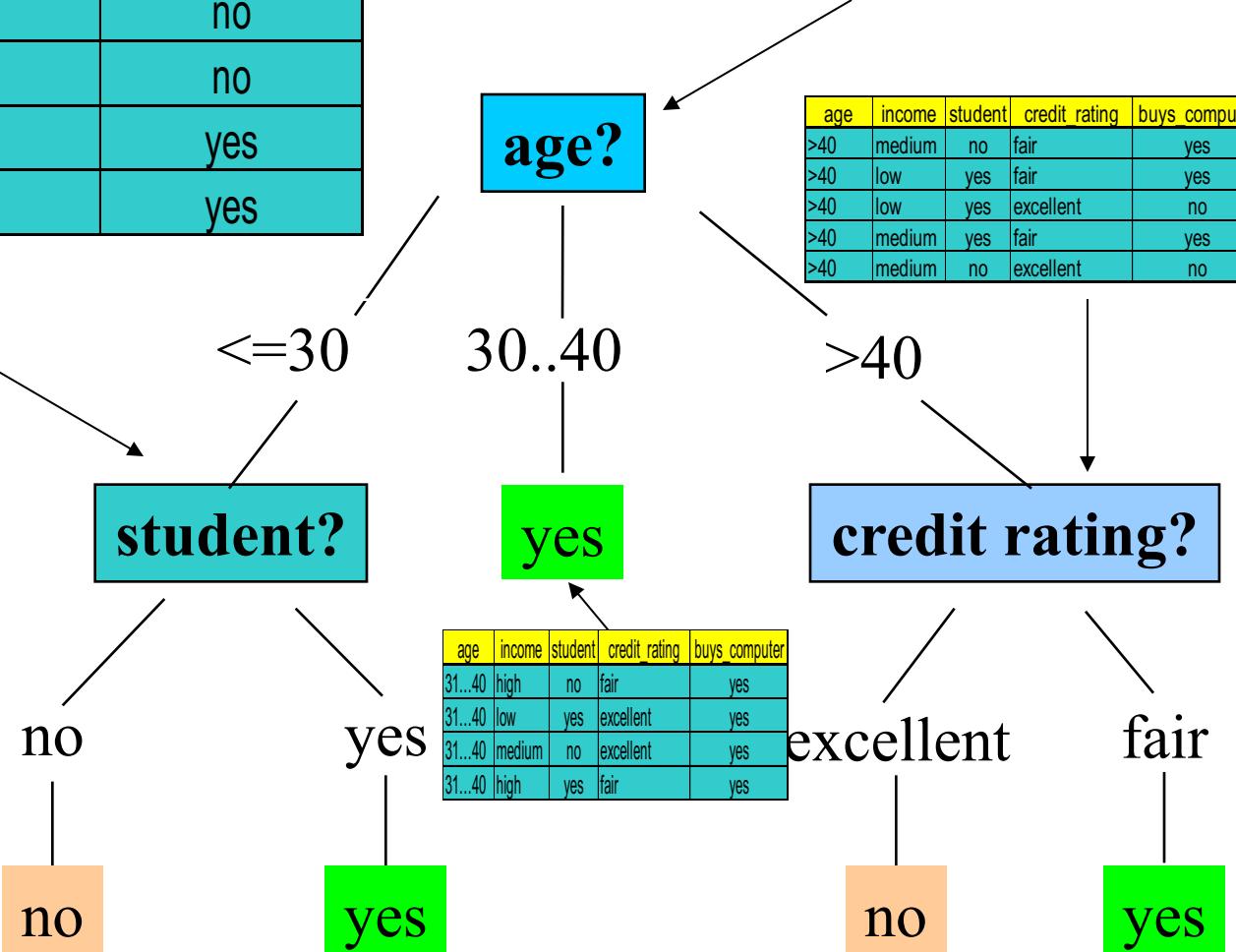
This follows an example from Quinlan's ID3

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

# Output: A Decision Tree for Computer Purchase

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

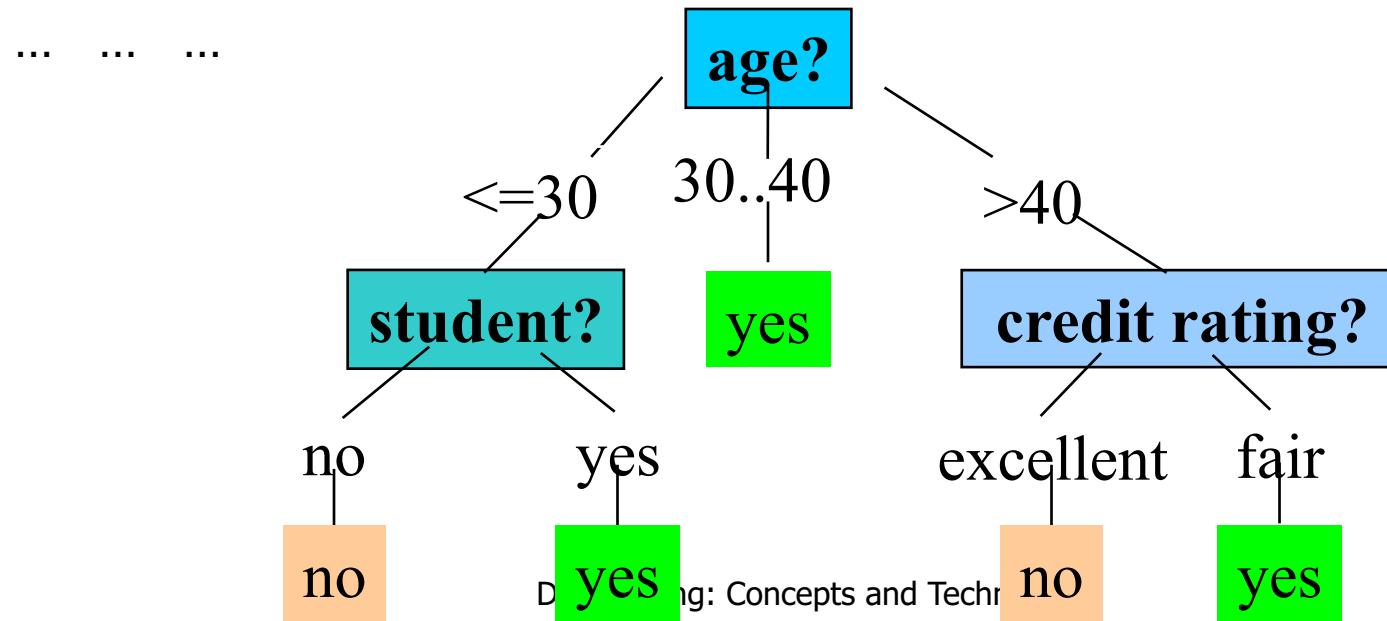


# Extracting Classification Rules from Trees

- Represent the knowledge in the form of **IF-THEN** rules
  - Rules are easier for humans to understand
- One rule is created for each path from the root to a leaf
  - Each attribute-value pair along a path forms a conjunction
  - The leaf node holds the class prediction
  - Example

**IF**  $age = "<=30"$  **AND**  $student = "no"$  **THEN**  $buys\_computer = "no"$

**IF**  $age = "<=30"$  **AND**  $student = "yes"$  **THEN**  $buys\_computer = "yes"$



Exercise: Write down the pseudo-code of the induction algorithm

## Algorithm for Decision Tree Induction

---

- Basic algorithm (a greedy algorithm)
  - Input: Attributes are categorical (if continuous-valued, they are **discretized** in advance)
  - Overview: Tree is constructed in a **top-down recursive divide-and-conquer manner**
    - At start, all the training examples are at the root
    - Samples are partitioned recursively based on selected *test-attributes*
    - *Test-attributes* are selected on the basis of a heuristic or statistical measure (e.g., **information gain**)
- Three conditions for stopping partitioning (i.e., boundary conditions)
  - There are no samples left, **OR**
  - All samples for a given node belong to the same class, **OR**
  - There are no remaining attributes for further partitioning (**majority voting** is employed for classifying the leaf)

# Decision Tree Induction Algorithm

---

# Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- S contains  $s_i$  tuples of class  $C_i$  for  $i = \{1, \dots, m\}$
- information measures info required to classify any arbitrary tuple

$$I(S_1, S_2, \dots, S_m) = - \sum_{i=1}^m \frac{s_i}{S} \log_2 \frac{s_i}{S}$$

- entropy of attribute A with values  $\{a_1, a_2, \dots, a_v\}$

$$E(A) = \sum_{j=1}^v \frac{s_{1j} + \dots + s_{mj}}{S} I(s_{1j}, \dots, s_{mj})$$

- information gained by branching on attribute A

$$Gain(A) = I(S_1, S_2, \dots, S_m) - E(A)$$

# Attribute Selection by Information Gain Computation

- Class P: buys\_computer = "yes"
- Class N: buys\_computer = "no"
- $I(p, n) = I(9, 5) = 0.940$
- Compute the entropy for *age*:

age	$p_i$	$n_i$	$I(p_i, n_i)$
$\leq 30$	2	3	0.971
$30 \dots 40$	4	0	0
$>40$	3	2	0.971

age	income	student	credit_rating	buys_computer
$\leq 30$	high	no	fair	no
$\leq 30$	high	no	excellent	no
$31 \dots 40$	high	no	fair	yes
$>40$	medium	no	fair	yes
$>40$	low	yes	fair	yes
$>40$	low	yes	excellent	no
$31 \dots 40$	low	yes	excellent	yes
$\leq 30$	medium	no	fair	no
$\leq 30$	low	yes	fair	yes
$>40$	medium	yes	fair	yes
$\leq 30$	medium	yes	excellent	yes
$31 \dots 40$	medium	no	excellent	yes
$31 \dots 40$	high	yes	fair	yes
$>40$	medium	no	excellent	no

$$E(\text{age}) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0)$$

$$+ \frac{5}{14} I(3,2) = 0.694$$

$\frac{5}{14} I(2,3)$  means " $\text{age} \leq 30$ " has 5 out of 14 samples, with 2 yes's and 3 no's. Hence

$$\text{Gain}(\text{age}) = I(p, n) - E(\text{age}) = 0.246$$

Similarly,

$$\text{Gain}(\text{income}) = 0.029$$

$$\text{Gain}(\text{student}) = 0.151$$

$$\text{Gain}(\text{credit\_rating}) = 0.048$$

income	$p_i$	$n_i$	$I(p_i, n_i)$
high	2	2	1
medium	4	2	0.918

Q: what's the extreme/worst case?

# Other Attribute Selection Measures and Splitting Choices

---

- Gini index (CART, IBM IntelligentMiner)
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - May need other tools, such as clustering, to get the possible split values
  - Can be modified for categorical attributes
- Induces binary split => binary decision trees

# *Gini* Index (IBM IntelligentMiner)

- If a data set  $T$  contains examples from  $n$  classes, gini index,  $gini(T)$  is defined as

$$gini(T) = 1 - \sum_{j=1}^n p_j^2 = \sum_j p_j \cdot (1 - p_j)$$

where  $p_j$  is the **relative** frequency of class  $j$  in  $T$ .

- If a data set  $T$  is split into two subsets  $T_1$  and  $T_2$  with sizes  $N_1$  and  $N_2$  respectively, the *gini* index of the split data contains examples from  $n$  classes, the *gini* index  $gini(T)$  is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the **smallest**  $gini_{split}(T)$  is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

# Case I: Numerical Attributes

Age	Car	Class
20	...	Y
20	...	N
20	...	N
25	...	N
25	...	Y
30	...	Y
30	...	Y
30	...	Y
40	...	Y
40	...	Y

Split value

Cut=22.5	Y	N
<	1	2
>=	6	1

$$Gini_{split}(S) = \frac{3}{10}Gini(1,2) + \frac{7}{10}Gini(6,1) = 0.30$$

22.5  
27.5

Cut=27.5	Y	N
<	2	3
>=	5	0

$$Gini_{split}(S) = \frac{5}{10}Gini(2,3) + \frac{5}{10}Gini(5,0) = 0.24$$

...

...

$$\dots Gini(S) = 1 - \sum p_j^2$$

$$Gini_{split}(S) = \frac{n_1}{n} Gini(S_1) + \frac{n_2}{n} Gini(S_2)$$

Exercise: compute gini indexes for other splits

# Case II: Categorical Attributes

count matrix

attrib list for Car

Age	Car	Class
20	M	Y
30	M	Y
25	T	N
30	S	Y
40	S	Y
20	T	N
30	M	Y
25	M	Y
40	M	Y
20	S	N



	Class=Y	Class=N
M	5	0
T	0	2
S	2	1

Need to consider all possible splits !

	Y	N
{M, T}	5	2
{S}	2	1

	Y	N
{M, S}	7	1
{T}	0	2

	Y	N
{T, S}	2	3
{M}	5	0

$$\begin{aligned} \text{Gini}_{\text{split}}(S) &= \\ 7/10 * \text{Gini}(5,2) + \\ 3/10 * \text{Gini}(2,1) &= \\ 0.42 \end{aligned}$$

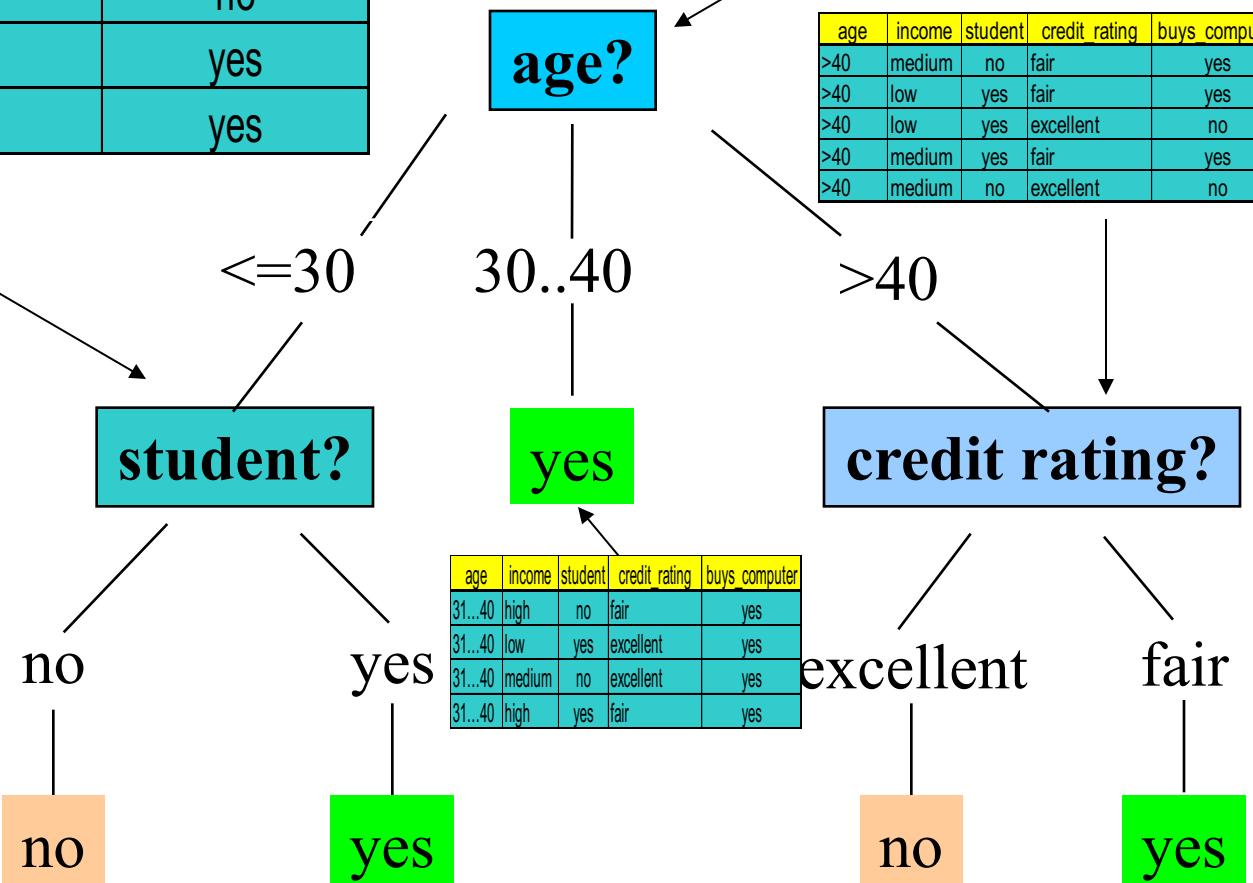
$$\begin{aligned} \text{Gini}_{\text{split}}(S) &= \\ 8/10 * \text{Gini}(7,1) + \\ 2/10 * \text{Gini}(0,2) &= \\ 0.18 \end{aligned}$$

$$\begin{aligned} \text{Gini}_{\text{split}}(S) &= \\ 5/10 * \text{Gini}(2,3) + \\ 5/10 * \text{Gini}(5,0) &= \\ 0.24 \end{aligned}$$

# ID3

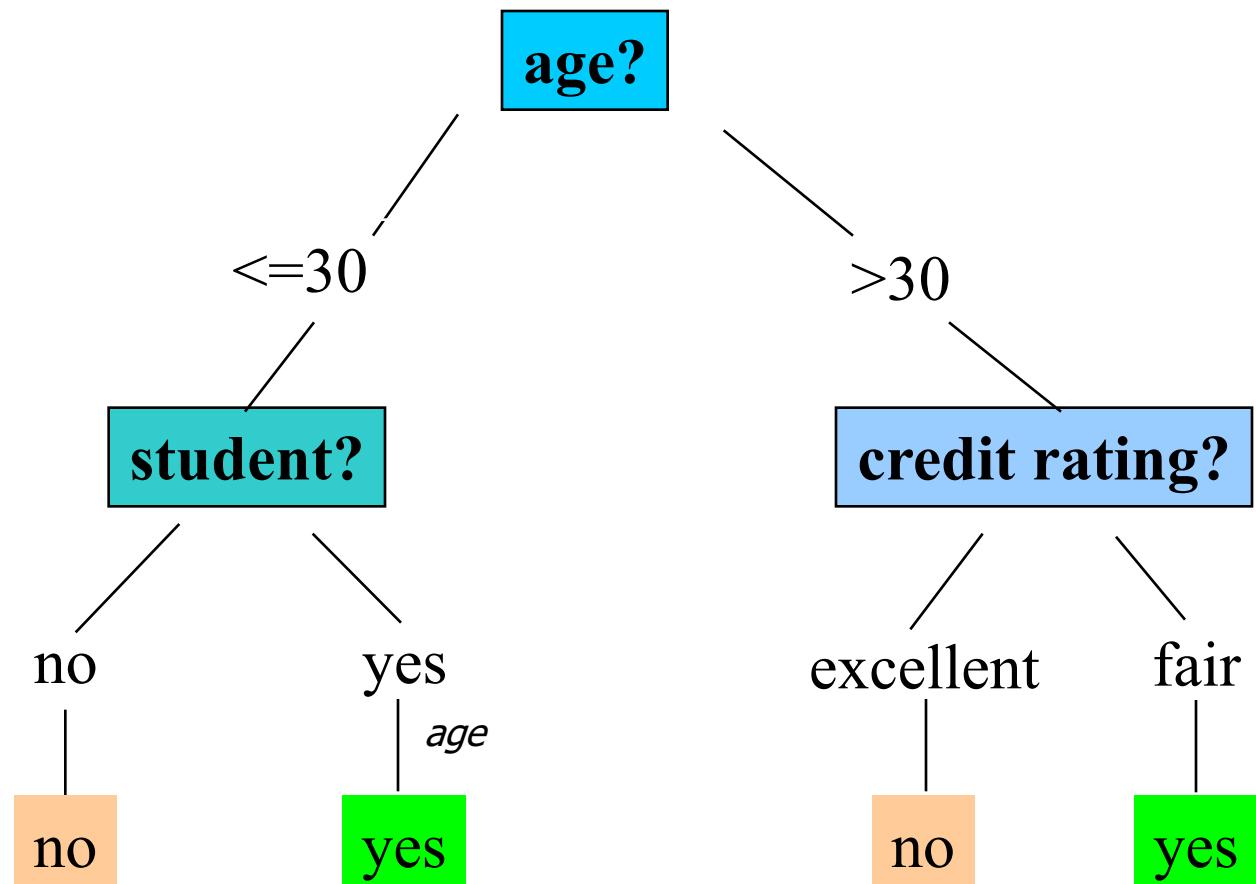
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no



# CART/SPRINT

Illustrative of  
the shape only



# Avoid Overfitting in Classification

---

- **Overfitting:** An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”



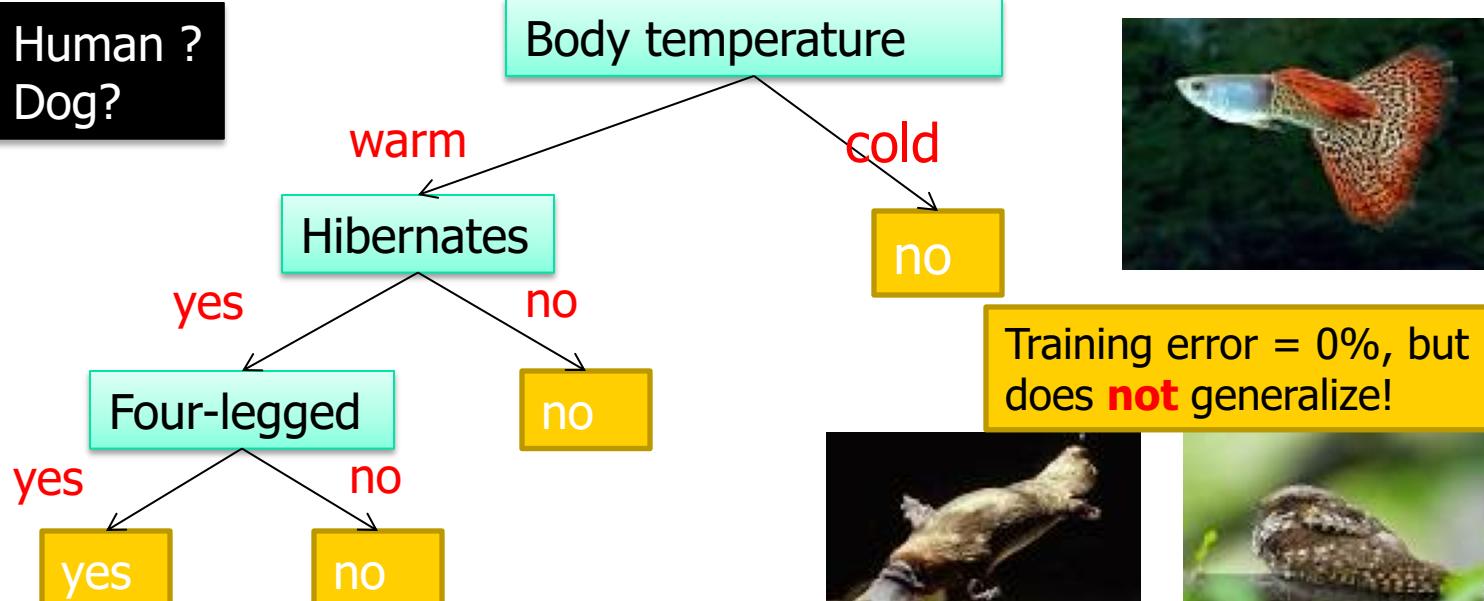
- Lack of representative samples
- Existence of noise

# Overfitting Example /1

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Is Mammal?
Salamander	Cold-blooded	No	Yes	Yes	No
Guppy	Cold-blooded	Yes	No	No	No
Eagle	Warm-blooded	No	No	No	No
Poorwill	Warm-blooded	No	No	Yes	No
Platypus	Warm-blooded	No	Yes	Yes	Yes



Human ?  
Dog?





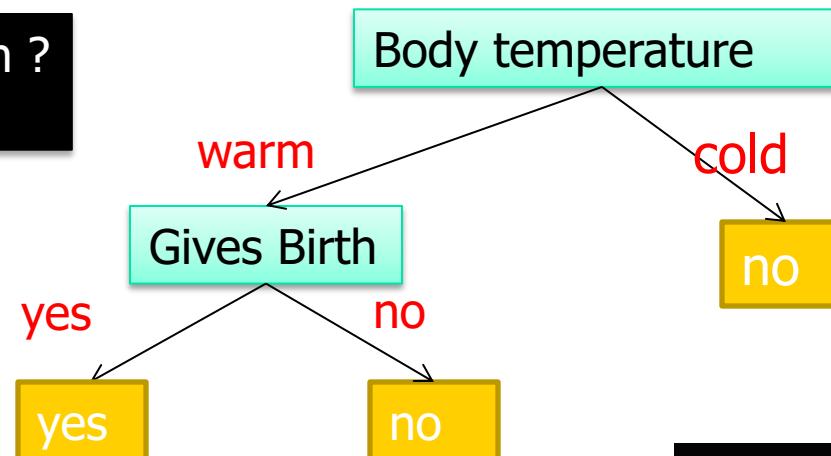
- Lack of representative samples
- Existence of noise

# Overfitting Example /2

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Is Mammal?
Salamander	Cold-blooded	No	Yes	Yes	No
Guppy	Cold-blooded	Yes	No	No	No
Eagle	Warm-blooded	No	No	No	No
Poorwill	Warm-blooded	No	No	Yes	No
Platypus	Warm-blooded	No	Yes	Yes	Yes

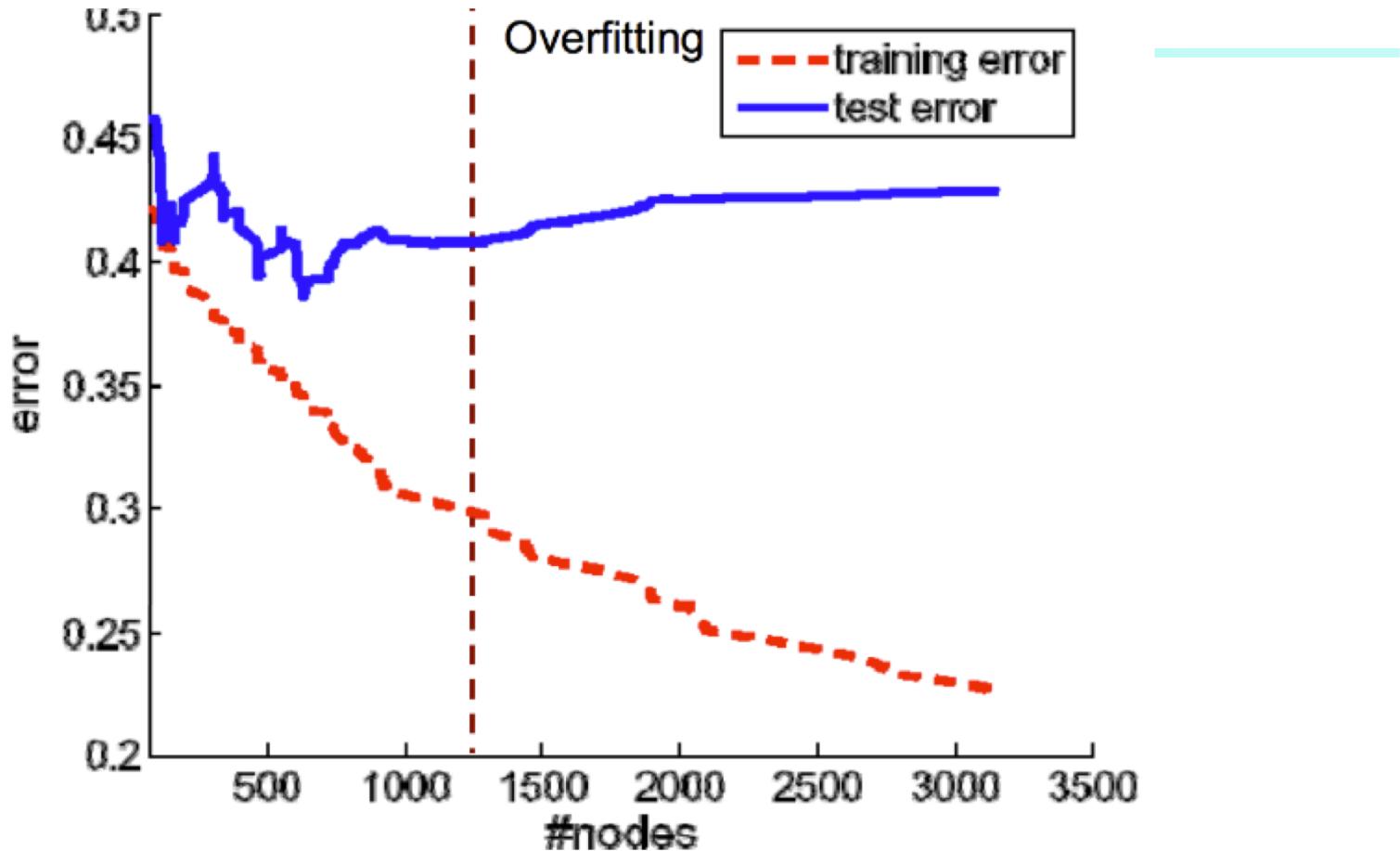


Human ?  
Dog?



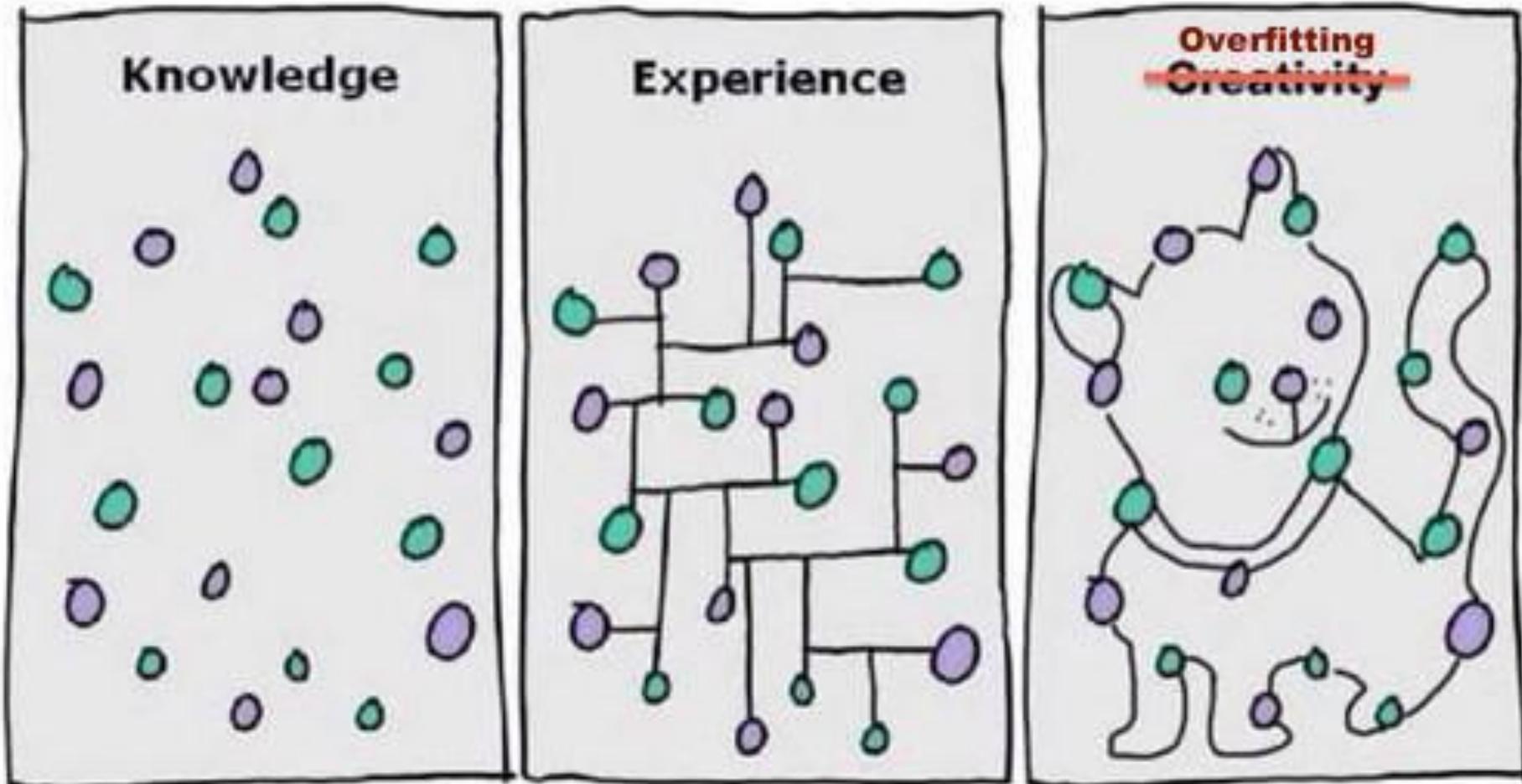
Training error = 20%, but **generalizes!**

# Overfitting



- Overfitting: model too complex → training error keep decreasing, but testing error increases
- Underfitting: model too simple → both training and testing has large errors.

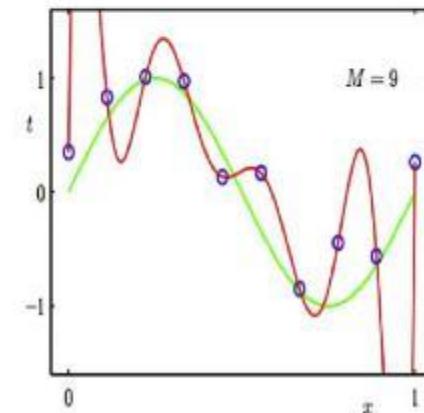
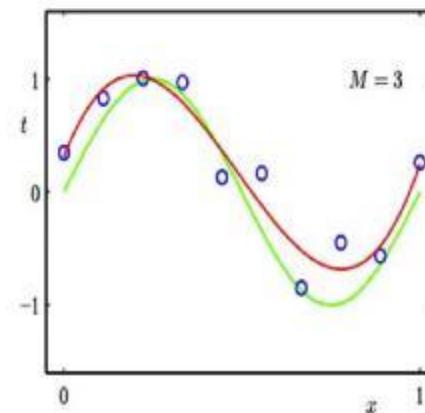
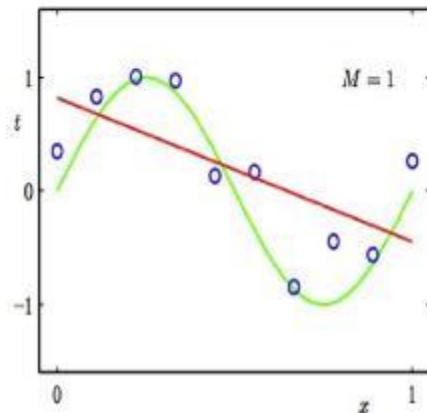
# Overfitting



# Overfitting examples in Regression & Classification

## Under- and Over-fitting examples

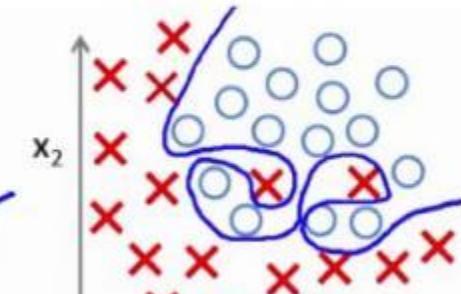
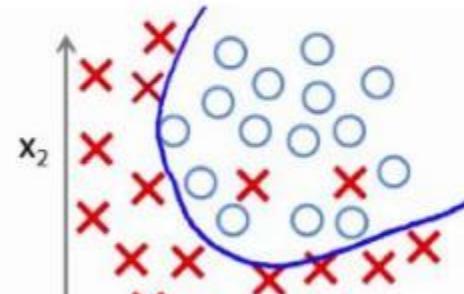
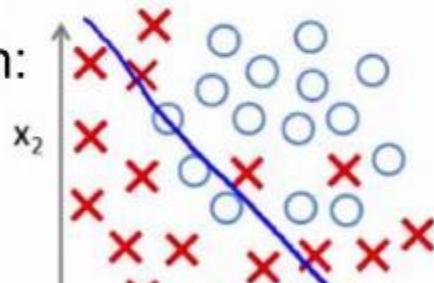
Regression:



predictor too inflexible:  
cannot capture pattern

predictor too flexible:  
fits noise in the data

Classification:



# DT Pruning Methods

---

- Use a separate **validation set**
- Estimation of generalization/test errors
- Use all the data for training
  - but apply a **statistical test** (e.g., chi-square) to estimate whether expanding or pruning a node may improve the entire distribution
- Use minimum description length (MDL) principle
  - halting growth of the tree when the encoding is minimized

# Pessimistic Post-pruning

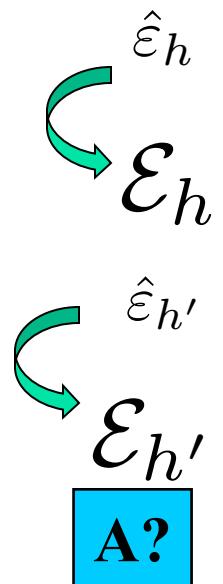
Better estimate of generalization error in C4.5:  
*Use the upper 75% confidence bound from the training error of a node, assuming a binomial distribution*

- Observed on the training data
  - $e(t)$ : #errors on a leaf node  $t$  of the tree  $T$
  - $e(T) = \sum_{t \in T} e(t)$
- What's the **generalization errors** (i.e., errors on testing data) on  $T$ ?
  - Use pessimistic estimates
  - $e'(t) = e(t) + 0.5$
  - $E'(t) = e(T) + 0.5N$ , where  $N$  is the number of leaf nodes in  $T$
- What's the generalization errors on  $\text{root}(T)$  only?
  - $E'(\text{root}(T)) = e(T) + 0.5$
- Post-pruning from bottom-up
  - If generalization error reduces after pruning, replace sub-tree by a leaf node
  - Use majority voting to decide the class label

# Example

Class = Yes	20
Class = No	10

$$\text{Error} = 10/30$$



- Training error before splitting on  $A = 10/30$
- Pessimistic error =  $(10+0.5)/30$
- Training error after splitting on  $A = 9/30$
- Pessimistic error =  $(9 + 4*0.5)/30 = 11/30$

Prune the subtree at  $A$

Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

# Classification in Large Databases

---

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why decision tree induction in data mining?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods

# Enhancements to basic decision tree induction

---

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals, **one-hot encoding**, or specialized DT learning algorithms
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

---

- Bayesian Classifiers

# Bayesian Classification: Why?

---

- Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- Probabilistic prediction: **Predict multiple hypotheses**, weighted by their probabilities
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

# Bayesian Theorem: Basics

---

- Let  $X$  be a data sample whose class label is **unknown**
- Let  $h$  be a **hypothesis** that  $X$  belongs to class C
- For classification problems, determine  $P(h|X)$ : the probability that the hypothesis holds given the observed data sample X
- $P(h)$ : **prior** probability of hypothesis h (i.e. the initial probability before we observe any data, reflects the background knowledge)
- $P(X)$ : probability that sample data is observed
- $P(X|h)$  : probability of observing the sample X, given that the hypothesis holds

# Bayesian Theorem

---

- Given training data  $X$ , *posteriori probability of a hypothesis*  $h$ ,  $P(h|X)$  follows the Bayes theorem

$$P(h | X) = \frac{P(X|h)P(h)}{P(X)}$$

- Informally, this can be written as  
posterior = likelihood x prior / evidence
- MAP (**maximum posteriori**) hypothesis

$$h_{\text{MAP}} = \arg \max_{h \in H} P(h | X) = \arg \max_{h \in H} P(X|h)P(h)$$

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

# Training dataset

## Hypotheses:

$C_1$ : buys\_computer= 'yes'

$C_2$ : buys\_computer= 'no'

Data sample

$X = (\text{age} \leq 30,$

**Income**=medium,

**Student**=yes,

**Credit\_rating**=

Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
30...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

# Sparsity Problem

$$h_{\text{MAP}} = \arg \max_{h \in H} P(X|h)P(h)$$

- Maximum likelihood Estimate of  $P(h)$ 
  - Let  $p$  be the probability that the class is  $C_1$
  - Consider a training example  $x$  and its label  $y$
  - $L(x) = p^y(1-p)^{1-y}$
  - $L(X) = \prod_{x \text{ in } X} L(x)$  Data likelihood
  - $I(X) = \log(L(X)) = \sum_{x \text{ in } X} y \log(p) + (1-y) \log(1-p)$  Log Data likelihood
  - To maximize  $I(X)$ , let  $dI(X)/dp = 0 \rightarrow p = (\sum y)/n$
  - $P(C_1) = 9/14, P(C_2) = 5/14$
- ML estimate of  $P(X|h) = ?$ 
  - Requires  $O(2^d)$  training examples, where  $d$  is the #features.

Curse of dimensionality

# Naïve Bayes Classifier

---

- Use a model
  - Assumption: attributes are **conditionally independent**:
- $$P(X | C_i) = \prod_{k=1}^n P(x_k | C_i)$$
- The product of occurrence of say 2 elements  $x_1$  and  $x_2$ , given the current class is  $C$ , is the product of the probabilities of each element taken separately, given the same class  $P([x_1, x_2], C) = P(x_1, C) * P(x_2, C)$
- No dependence relation between attributes given the class
- Greatly reduces the computation cost, only count the class distribution → Only need to estimate  $P(x_k | C_i)$

# Naïve Bayesian Classifier: Example

- Compute  $P(X|C_i)$  for each class

$X=(\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$

$$P(\text{age} = \text{"<30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$$

$$P(\text{age} = \text{"<30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$$

$$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$$

$$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$$

$$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$$

$$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$$

$$P(X | C_i) : P(X | \text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$

$$P(X | \text{buys\_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X | C_i) * P(C_i) : P(X | \text{buys\_computer} = \text{"yes"}) * P(\text{buys\_computer} = \text{"yes"}) = 0.028$$

$$P(X | \text{buys\_computer} = \text{"no"}) * P(\text{buys\_computer} = \text{"no"}) = 0.007$$

$X$  belongs to class "buys\_computer=yes"

likelihood

# The Need for Smoothing

---

- $\Pr[X_i = v_j \mid C_k]$  could still be 0, if not observed in the training data
  - makes  $\Pr[C_k \mid X] = 0$ , regardless of other likelihood values of  $\Pr[C_t = v_t \mid C_k]$
- Add-1 Smoothing
  - reserve a small amount of probability for unseen probabilities
  - (conditional) probabilities of observed events have to be **adjusted** to make the total probability equals 1.0

# Add-1 Smoothing

---

- $\Pr[X_i = v_j \mid C_k] = \text{Count}(X_i = v_j, C_k) / \text{Count}(C_k)$
- $\Pr[X_i = v_j \mid C_k] = [\text{Count}(X_i = v_j, C_k) + 1] / [\text{Count}(C_k) + B]$
- What's the right value for B?
  - make  $\sum_{v_j} \Pr[X_i = v_j \mid C_k] = 1$
  - Explain the above constraint
    - $\Pr[X_i \mid C_k]$ : Given  $C_k$ , the (conditional) probability of  $X_i$  taking a specific value
    - $X_i$  must take one of the values, hence summing over all possible value for  $X_i$ , the probabilities should sum up to 1.0
  - $B = \text{dom}(X_i)$ , i.e., # of values  $X_i$  can take

# Smoothing Example

no instance of age  $\leq 30$  in the "No" class



Class:

C1:`buys_computer='yes'`

C2:`buys_computer='no'`

Data sample

$X = (\text{age} \leq 30, \text{Income}=\text{medium}, \text{Student}=\text{yes}, \text{Credit\_rating}=\text{Fair})$

age	income	student	credit_rating	buys_computer
30...40	high	no	fair	no
30...40	high	no	excellent	no
30...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
30...40	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

B=3

B=3

B=2

B=2

# Consider the “No” Class

- Compute  $P(X|C_i)$  for the “No” class

**X=(age<=30 , income =medium, student=yes, credit\_rating=fair)**

$$P(\text{age}=<30 \mid \text{buys\_computer}=\text{"no"}) = (0 +1) / (5 +3) = 0.125$$

$$P(\text{income}=\text{"medium"} \mid \text{buys\_computer}=\text{"no"}) = (2 +1) / (5 +3) = 0.375$$

$$P(\text{student}=\text{"yes"} \mid \text{buys\_computer}=\text{"no"}) = (1 +1) / (5 +2) = 0.286$$

$$P(\text{credit\_rating}=\text{"fair"} \mid \text{buys\_computer}=\text{"no"}) = (2 +1) / (5 +2) = 0.429$$

$$\mathbf{P(X|Ci)} : P(X|\text{buys\_computer}=\text{"no"}) = 0.125 \times 0.375 \times 0.286 \times 0.429 = 0.00575$$

$$\mathbf{P(X|Ci)*P(Ci)} : P(X|\text{buys\_computer}=\text{"no"}) * P(\text{buys\_computer}=\text{"no"}) = 0.00205$$

Probabilities	Without Smoothing	With Smoothing
$\Pr[<=30 \mid \text{No}]$	0 / 5	1 / 8
$\Pr[30..40 \mid \text{No}]$	3 / 5	4 / 8
$\Pr[>=40 \mid \text{No}]$	2 / 5	3 / 8

# How to Handle Numeric Values

---

- Need to model the distribution of  $\Pr[X_i | C_k]$
- Method 1:
  - Assume it to be Gaussian  $\rightarrow$  Gaussian Naïve Bayes
  - $\Pr[X_i = v_j | C_k] = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(v_j - \mu_{ik})^2}{2\sigma_i^2}\right)$
- Method 2:
  - Use binning to discretize the feature values

# Text Classification

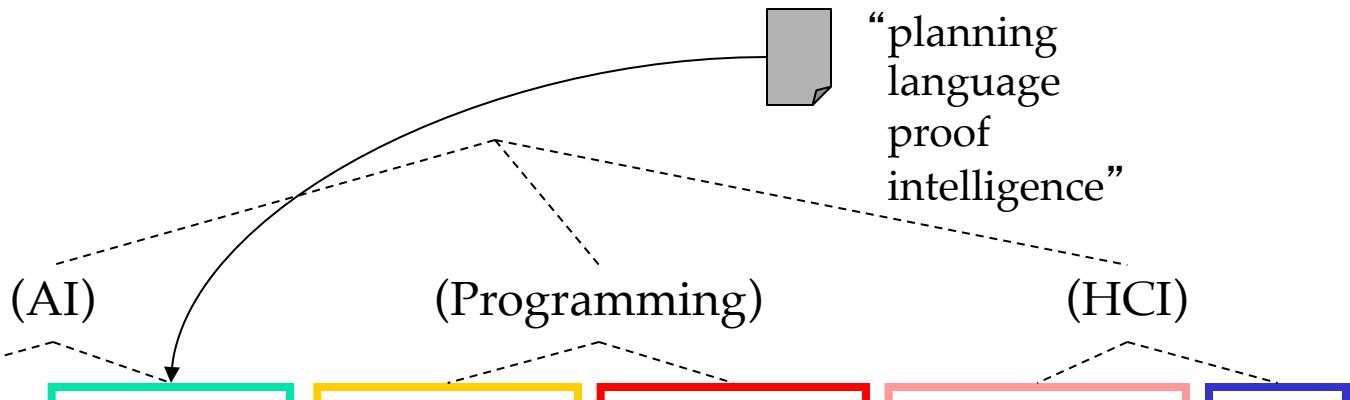
---

- NB has been widely used in **text classification** (aka., text categorization)
- Outline:
  - Applications
  - Language Model
  - Two Classification Methods

Based on “Chap 13: Text classification & Naive Bayes”  
in Introduction to Information Retrieval  
• <http://nlp.stanford.edu/IR-book/>

# Document Classification

*Test  
Data:*



*Classes:*

*Training  
Data:*

learning	planning	programming	garbage	...
<u>intelligence</u>	temporal	semantics	collection	...
algorithm	reasoning	<u>language</u>	memory	
reinforcement	plan	<u>proof</u> ...	optimization	
network...	<u>language</u> ...		region...	

(Note: in real life there is often a hierarchy, not present in the above problem statement; and also, you get papers on ML approaches to Garb. Coll.)

# More Text Classification Examples:

## Many search engine functionalities use classification

---

Assign labels to each document or web-page:

- Labels are most often topics such as Yahoo-categories  
e.g., "*finance*," "*sports*," "*news>world>asia>business*"
- Labels may be genres  
e.g., "*editorials*" "*movie-reviews*" "*news*"
- Labels may be opinion on a person/product  
e.g., "*like* ", "*hate* ", "*neutral* "
- Labels may be domain-specific  
e.g., "*interesting-to-me*": "*not-interesting-to-me*"  
e.g., "*contains adult language*": "*doesn't*"  
e.g., *language identification*: *English, French, Chinese, ...*  
e.g., *search vertical*: *about Linux versus not*  
e.g., "*link spam*": "*not link spam*"

# Challenge in Applying NB to Text

---

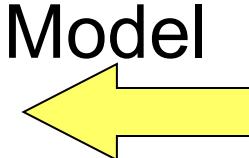
- Need to model  $P(\text{text} \mid \text{class})$ 
  - e.g.,  $P(\text{"a b c d"} \mid \text{class})$
- Extremely sparse → Need a model to help
- 1<sup>st</sup> method:
  - Based on a statistical language model
    - Turns out to be the **bag** of words model if using unigrams
    - → multinomial NB
- 2<sup>nd</sup> method:
  - View a text as a **set** of tokens → a Boolean vector in  $\{0, 1\}^{|V|}$ , where  $V$  is the vocabulary.
  - → Bernoulli NB

`sklearn.naive_bayes.MultinomialNB`

`sklearn.naive_bayes.BernoulliNB`

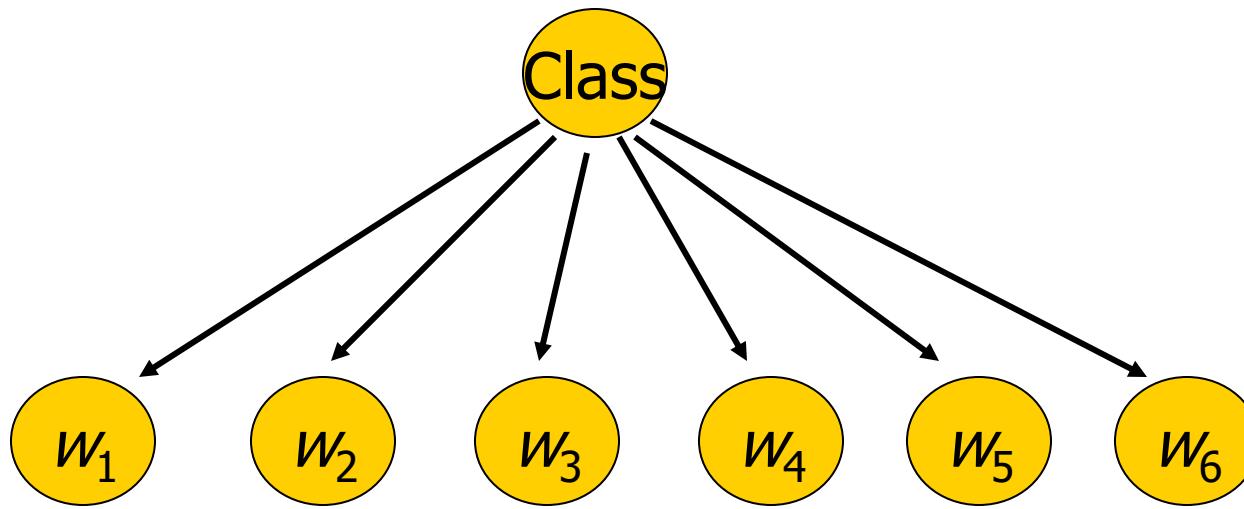
# Unigram and higher-order Language models in Information Retrieval

---

- $P("a\ b\ c\ d") =$ 
    - $P(a) * P(b | a) * P(c | a\ b) * P(d | a\ b\ c)$
  - Unigram model: 0-th order Markov Model
    - $P(d | a\ b\ c) = P(d)$
  - Bigram Language Models: 1<sup>st</sup> order Markov Model
    - $P(d | a\ b\ c) = P(d | c)$
    - $P(a\ b\ c\ d) = P(a) * P(b | a) * P(c | b) * P(d | c)$
  - The same with class-conditional probabilities,  
i.e.,  $P("a\ b\ c\ d" | C)$
- 

Easy.  
Effective!

# Naïve Bayes via a class conditional language model = multinomial NB



- Effectively, the probability of each class is done as a class-specific unigram language model

Probabilistic Graphical Model: arrow means conditional dependency.

# Using **Multinomial** Naive Bayes Classifiers to Classify Text: Basic method

$$P(C_j \mid w_1 w_2 w_3 w_4) \propto P(C_j) P(w_1 w_2 w_3 w_4 \mid C_j)$$

an example  
text of 4  
tokens

$$\propto P(c_j) \prod_{i=1}^4 P(w_i \mid C_j)$$

unigram  
model

$$\propto P(c_j) \prod_{i=1}^{|V|} P(x_i \mid C_j)^{\theta_i}$$

bag of word;  
multinomial

- Essentially, the classification is *independent* of the positions of the words
  - Use same parameters for each position
  - Result is **bag** of words model, i.e. text  $\equiv \{x_i : \theta_i\}_{i=1}^{|V|}$

e.g., “to be or not to be”

# Naïve Bayes: Learning

---

- From training corpus, extract  $V = \text{Vocabulary}$
- Calculate required  $P(c_j)$  and  $P(x_k | c_j)$  terms
  - For each  $c_j$  in  $C$  do
    - $docs_j \leftarrow$  subset of documents for which the target class is  $c_j$
    - $P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$
  - $Text_j \leftarrow$  single document containing all  $docs_j$
  - for each word  $x_k$  in  $\text{Vocabulary}$ 
    - $n_k \leftarrow$  number of occurrences of  $x_k$  in  $Text_j$
    - $P(x_k | c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |\text{Vocabulary}|}$

# Naïve Bayes: Classifying

---

- positions  $\leftarrow$  all word positions in current document which contain tokens found in *Vocabulary*
- Return  $c_{NB}$ , where

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in positions} P(x_i | c_j)$$

# Naive Bayes: Time Complexity

---

- **Training Time:**  $O(|D|L_d + |C||V|)$   
where  $L_d$  is the average length of a document in  $D$ .
  - Assumes  $V$  and all  $D_i$ ,  $n_i$ , and  $n_{ij}$  pre-computed in  $O(|D|L_d)$  time during one pass through all of the data.
  - Generally just  $O(|D|L_d)$  since usually  $|C||V| < |D|L_d$
- **Test Time:**  $O(|C| L_t)$   
where  $L_t$  is the average length of a test document.
- Very efficient overall, linearly proportional to the time needed to just read in all the data.

Why?

# Underflow Prevention: log space

---

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since  $\log(xy) = \log(x) + \log(y)$ , it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

$$c_{NB} = \operatorname{argmax}_{c_j \in C} \log P(c_j) + \sum_{i \in positions} \log P(x_i | c_j)$$

- Note that model is now just max of sum of weights...

# Bernoulli Model

---

- $V = \{a, b, c, d, e\} = \{x_1, x_2, x_3, x_4, x_5\}$
- Feature functions  $f_i(\text{text}) = \text{if text contains } x_i$
- The feature functions extract a vector of  $\{0, 1\}^{|V|}$  from any text
- Apply NB directly

"a b c d"
"d e b"



a	b	c	d	e	C
1	1	1	1	0	+
0	1	0	1	1	-

# Two Models /1

---

- Model 1: Multinomial = Class conditional unigram
  - One feature  $X_i$  for each word pos in document
    - feature's values are all words in dictionary
  - Value of  $X_i$  is the word in position  $i$
  - Naïve Bayes assumption:
    - Given the document's topic, word in one position in the document tells us nothing about words in other positions
  - Second assumption:
    - Word appearance does not depend on position

$$P(X_i = w | c) = P(X_j = w | c)$$

for all positions  $i, j$ , word  $w$ , and class  $c$

- Just have one multinomial feature predicting all words

# Two Models /2

---

- Model 2: Multivariate Bernoulli
  - One feature  $X_w$  for each word in dictionary
  - $X_w = \text{true}$  in document  $d$  if  $w$  appears in  $d$
  - Naive Bayes assumption:
    - Given the document's topic, appearance of one word in the document tells us nothing about chances that another word appears
- This is the model used in the binary independence model in classic probabilistic relevance feedback in hand-classified data (Maron in IR was a very early user of NB)

# Parameter estimation

---

- Multivariate Bernoulli model:

$$\hat{P}(X_w = t \mid c_j) = \begin{matrix} \text{fraction of documents of topic } c_j \\ \text{in which word } w \text{ appears} \end{matrix}$$

- Multinomial model:

$$\hat{P}(X_i = w \mid c_j) = \begin{matrix} \text{fraction of times in which} \\ \text{word } w \text{ appears} \\ \text{across all documents of topic } c_j \end{matrix}$$

- Can create a mega-document for topic  $j$  by concatenating all documents in this topic
- Use frequency of  $w$  in mega-document

# Classification

---

- Multinomial vs Multivariate Bernoulli?
- Multinomial model is almost always more effective in text applications!
  - See results figures later
- See *IIR* sections 13.2 and 13.3 for worked examples with each model

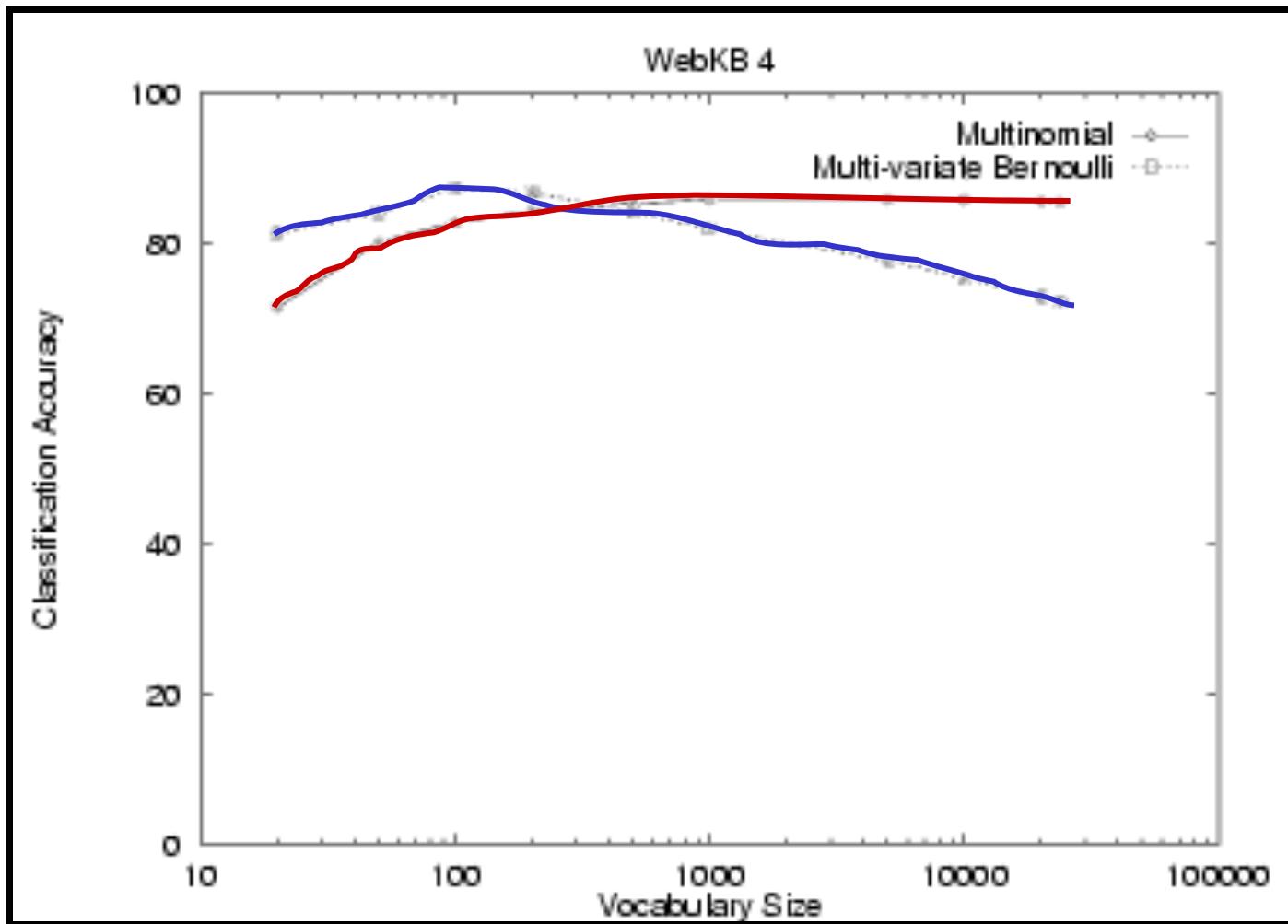
# Feature selection for NB

---

- In general feature selection is *necessary* for multivariate Bernoulli NB.
- Otherwise you suffer from noise, multi-counting
- “Feature selection” really means something different for multinomial NB. It means dictionary truncation
  - The multinomial NB model only has 1 feature
- This “feature selection” normally isn’t needed for multinomial NB, but may help a fraction with quantities that are badly estimated

# NB Model Comparison: WebKB

---

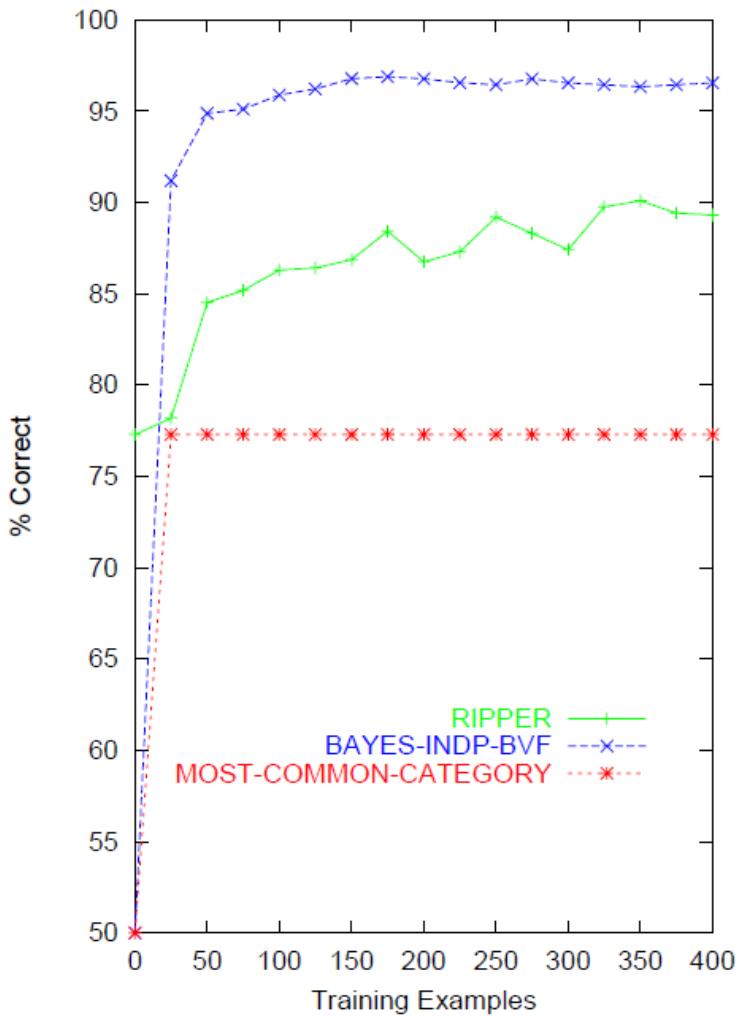


Faculty		Students		Courses	
associate	0.00417	resume	0.00516	homework	0.00413
chair	0.00303	advisor	0.00456	syllabus	0.00399
member	0.00288	student	0.00387	assignments	0.00388
ph	0.00287	working	0.00361	exam	0.00385
director	0.00282	stuff	0.00359	grading	0.00381
fax	0.00279	links	0.00355	midterm	0.00374
journal	0.00271	homepage	0.00345	pm	0.00371
recent	0.00260	interests	0.00332	instructor	0.00370
received	0.00258	personal	0.00332	due	0.00364
award	0.00250	favorite	0.00310	final	0.00355

Departments		Research Projects		Others	
departmental	0.01246	investigators	0.00256	type	0.00164
colloquia	0.01076	group	0.00250	jan	0.00148
epartment	0.01045	members	0.00242	enter	0.00145
seminars	0.00997	researchers	0.00241	random	0.00142
schedules	0.00879	laboratory	0.00238	program	0.00136
webmaster	0.00879	develop	0.00201	net	0.00128
events	0.00826	related	0.00200	time	0.00128
facilities	0.00807	arpa	0.00187	format	0.00124
eople	0.00772	affiliated	0.00184	access	0.00117
postgraduate	0.00764	project	0.00183	begin	0.00116

# Naïve Bayes on spam email

---

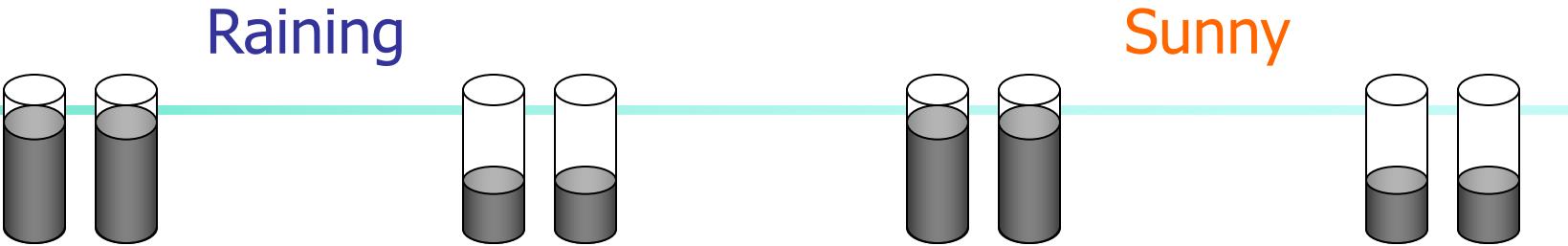


# Violation of NB Assumptions

---

- **Conditional independence**
- “Positional independence”
- Examples?

# Observations



$$P(+,+,\text{r}) = 3/8 \quad P(-,-,\text{r}) = 1/8$$

$$P(+,+,\text{s}) = 1/8 \quad P(-,-,\text{s}) = 3/8$$

- Two identical sensors at the same location
- Equivalent training dataset
- Note:  $P(s_1|C) = P(s_2|C)$  no matter what

$$P(s_i = + | r) =$$

$$P(s_i = - | r) =$$

$$P(r) =$$

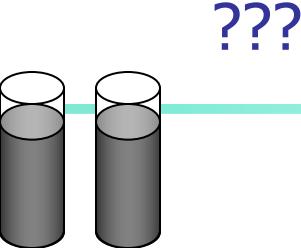
$$P(s) =$$

$$P(s_i = + | s) =$$

$$P(s_i = - | s) =$$

<b>S1</b>	<b>S2</b>	<b>Class</b>
+	+	r
+	+	r
+	+	r
-	-	r
+	+	s
-	-	s
-	-	s
-	-	s

# Prediction



- $P(r | ++) \propto$
- $P(s | ++) \propto$

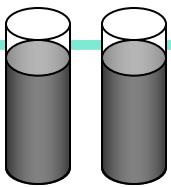
S1	S2	Class
+	+	???

$$P(s_i = + | r) = 3/4$$
$$P(s_i = - | r) = 1/4$$

$$P(r) = 1/2$$
$$P(s) = 1/2$$

$$P(s_i = + | s) = 1/4$$
$$P(s_i = - | s) = 3/4$$

???



Problem: Posterior probability estimation not accurate

Reason: Correlation between features

Fix: Use logistic regression classifier

- $P(r | ++) \propto 9/32$
- $P(s | ++) \propto 1/32$



- $P(r | ++) = 9/10$
- $P(s | ++) = 1/10$

$$P(s_i = + | r) = 3/4$$
$$P(s_i = - | r) = 1/4$$

$$P(s_i = + | s) = 1/4$$
$$P(s_i = - | s) = 3/4$$

$$P(r) = 1/2$$
$$P(s) = 1/2$$

S1	S2	Class
+	+	r
+	+	r
+	+	r
-	-	r
+	+	s
-	-	s
-	-	s
-	-	s

# Naïve Bayes Posterior Probabilities

---

- Classification results of naïve Bayes (the class with maximum posterior probability) are usually fairly accurate.
- However, due to the inadequacy of the conditional independence assumption, the actual posterior-probability numerical estimates are not.
  - Output probabilities are commonly very close to 0 or 1.
- Correct estimation  $\Rightarrow$  accurate prediction, but correct probability estimation is **NOT** necessary for accurate prediction (just need right ordering of probabilities)

# Naïve Bayesian Classifier: Comments

---

- Advantages :
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence , therefore loss of accuracy
  - Practically, dependencies exist among variables
  - E.g., hospitals: patients: Profile: age, family history etc  
Symptoms: fever, cough etc., Disease: lung cancer, diabetes etc
  - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- Better methods?
  - **Bayesian Belief Networks**
  - Logistic regression / maxent

- 
- (Linear Regression and) Logistic Regression Classifier
    - See LR slides

---

- Other Classification Methods

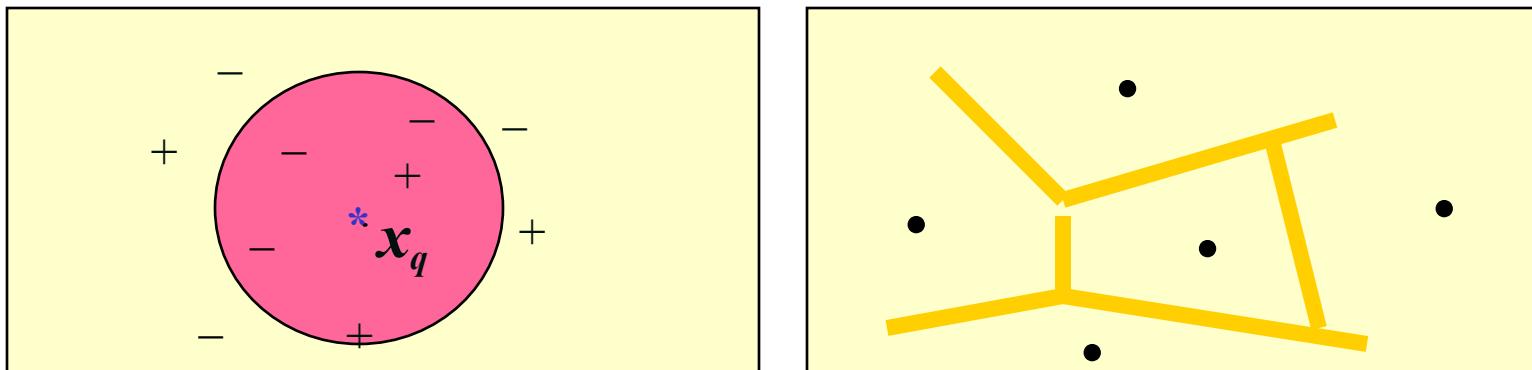
# Instance-Based Methods

---

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - $k$ -nearest neighbor approach
    - Instances represented as points in a Euclidean space.

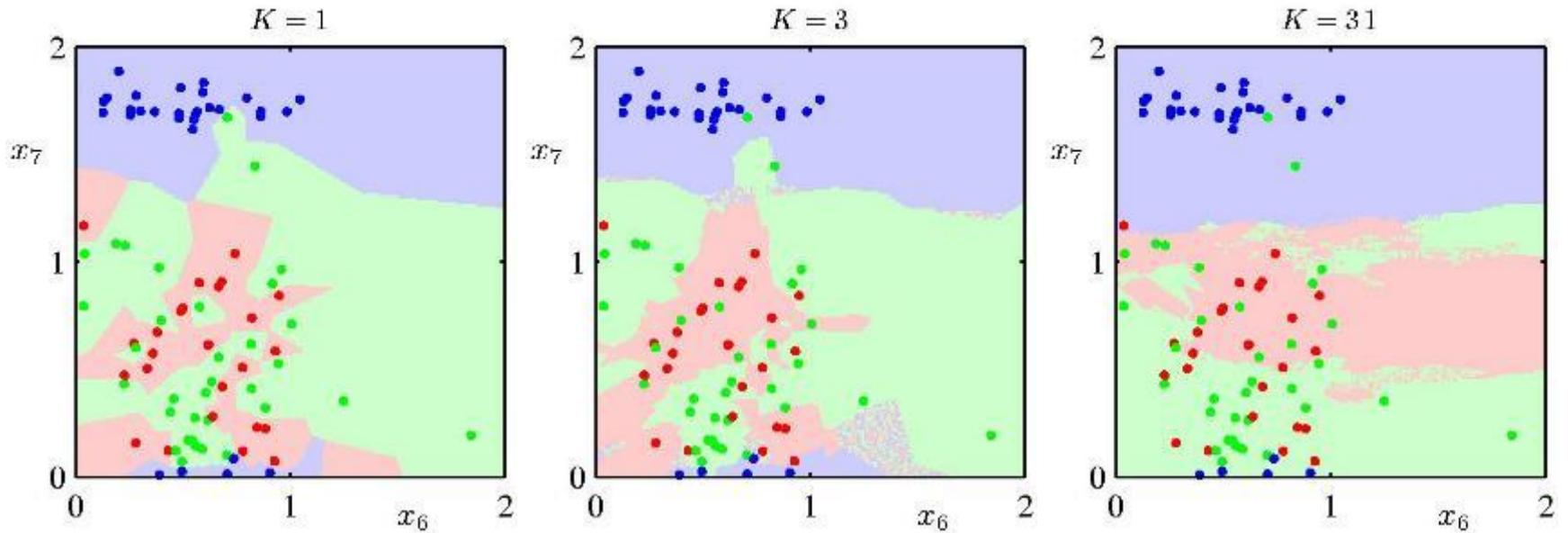
# The $k$ -Nearest Neighbor Algorithm

- All instances correspond to points in the  $n$ -D space.
- The nearest neighbor are defined in terms of Euclidean distance.
- The target function could be discrete- or real- valued.
- For discrete-valued, the  $k$ -NN returns the most common value among the  $k$  training examples nearest to  $x_q$ .
- Voronoi diagram: the *decision boundary* induced by 1-NN for a typical set of training examples.



# Effect of $k$

---



- $k$  acts as a smoother

# Discussion on the $k$ -NN Algorithm

---

- The  $k$ -NN algorithm for continuous-valued target functions
  - Calculate the mean values of the  $k$  nearest neighbors
- **Distance-weighted** nearest neighbor algorithm
  - Weight the contribution of each of the  $k$  neighbors according to their distance to the query point  $x_q$ 
    - giving greater weight to closer neighbors
  - Similarly, for real-valued target functions
- Robust to noisy data by averaging  $k$ -nearest neighbors
- Curse of dimensionality:
  - kNN search becomes very expensive in high dimensional space
    - High-dimensional indexing methods, e.g., **LSH**
  - distance between neighbors could be dominated by irrelevant attributes.
    - To overcome it, axes stretch or elimination of the least relevant attributes.

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

# Remarks on Lazy vs. Eager Learning

---

- Instance-based learning: lazy evaluation
- Decision-tree and Bayesian classification: eager evaluation
- Key differences
  - Lazy method may consider query instance  $x_q$  when deciding how to generalize beyond the training data  $D$
  - Eager method cannot since they have already chosen global approximation when seeing the query
- Efficiency: Lazy - less time training but more time predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

# Logistic Regression and MaxEnt

Wei Wang @ CSE, UNSW

March 13, 2019

# Generative vs. Discriminative Learning

- Generative models:

$$\Pr[y \mid x] = \frac{\Pr[x \mid y]\Pr[y]}{\Pr[x]}$$
$$\propto \Pr[x \mid y]\Pr[y] = \Pr[x, y]$$

- The key is to model the **generative** probability:  $\Pr[x \mid y]$ .
- Example: Naive Bayes.
- Discriminative models:
  - models  $\Pr[y \mid x]$  directly as  $g(x; \theta)$ .
  - Example: Decision tree, Logistic Regression.
- Instance-based Learning.
  - Example:  $k$ NN classifier.

# Linear Regression

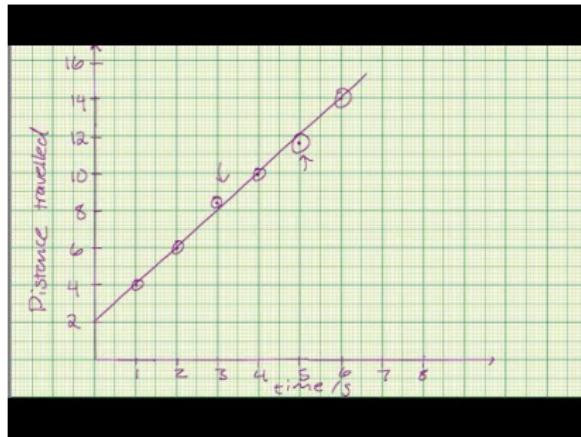


Figure: Linear Regression

## Task

- Input:  $(x^{(i)}, y^{(i)})$  pairs ( $1 \leq i \leq n$ )
- Preprocess: let  $\mathbf{x}^{(i)} = [1 \quad x^{(i)}]^\top$
- Output: The best  $\mathbf{w} = [w_0 \quad w_1]^\top$  such that  $\hat{y} = \mathbf{w}^\top \mathbf{x}$  **best** explains the observations

The criterion for “best”:

- Individual error:  $\epsilon_i = \hat{y}^{(i)} - y^{(i)}$
- Sum squared error:  $\ell = \sum_{i=1}^n \epsilon_i^2$

Find  $\mathbf{w}$  such that  $\ell$  is minimized.

# Minimizing a Function

Taylor Series of  $f(x)$  at point  $a$

$$f(x) = \sum_{n=0}^{+\infty} \frac{f^{(n)}(a)}{n!} (x - a)^n \quad (1)$$

$$= f(a) + f'(a) \cdot (x - a) + \frac{f''(a)}{2} (x - a)^2 + o((x - a)^2) \quad (2)$$

- Intuitively,  $f(x)$  is almost  $f(a) + f'(a) \cdot (x - a)$  for all  $a$  if it is close to  $x$ .
- If  $f(x)$  has local minimum  $x^*$ , then
  - $f'(x^*) = 0$ , and
  - $f''(x^*) > 0$ .

Minimum of the local minima is the global minimum if it is smaller than the function values at all the boundary points.

- Intuitively,  $f(x)$  is almost  $f(a) + \frac{f''(a)}{2} (x - a)^2$  if  $a$  is close to  $x^*$ .

# Find the Least Square Fit for Linear Regression

$$\begin{aligned}\frac{\partial \ell}{\partial w_j} &= \sum_{i=1}^n 2\epsilon_i \frac{\partial \epsilon_i}{\partial w_j} = \sum_{i=1}^n 2\epsilon_i \frac{\partial \mathbf{w}^\top \mathbf{x}^{(i)}}{\partial w_j} \\ &= \sum_{i=1}^n 2\epsilon_i x_j^{(i)} = 2 \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)}) x_j^{(i)}\end{aligned}$$

By setting the above to 0, this essentially requires, **for all  $j$**

$$\sum_{i=1}^n \hat{y}^{(i)} x_j^{(i)} = \sum_{i=1}^n y^{(i)} x_j^{(i)}$$

what the model predicts

what the data says

# Find the Least Square Fit for Linear Regression

In the simple 1D case, we have only two parameters in  $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$

$$\sum_{i=1}^n (w_0 + w_1 x_1^{(i)}) x_0^{(i)} = \sum_{i=1}^n y^{(i)} x_0^{(i)}$$

$$\sum_{i=1}^n (w_0 + w_1 x_1^{(i)}) x_1^{(i)} = \sum_{i=1}^n y^{(i)} x_1^{(i)}$$

Since  $x_0^{(i)} = 1$ , they are essentially

$$\sum_{i=1}^n (w_0 + w_1 x_1^{(i)}) \cdot 1 = \sum_{i=1}^n y^{(i)} \cdot 1$$

$$\sum_{i=1}^n (w_0 + w_1 x_1^{(i)}) \cdot x_1^{(i)} = \sum_{i=1}^n y^{(i)} \cdot x_1^{(i)}$$

## Example

Using the same example in [https://en.wikipedia.org/wiki/Linear\\_least\\_squares\\_\(mathematics\)](https://en.wikipedia.org/wiki/Linear_least_squares_(mathematics))

$$\mathbf{X} = \begin{bmatrix} \vdots & (\mathbf{x}^{(1)})^\top & \vdots \\ \vdots & (\mathbf{x}^{(2)})^\top & \vdots \\ \vdots & (\mathbf{x}^{(3)})^\top & \vdots \\ \vdots & (\mathbf{x}^{(4)})^\top & \vdots \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 6 \\ 5 \\ 7 \\ 10 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \quad \begin{bmatrix} 6 \\ 5 \\ 7 \\ 10 \end{bmatrix} \quad = \quad \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \quad \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \end{bmatrix}$$

# Generalization to $m$ -dim

- Easily generalizes to more than 2-dim:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_m^{(1)} \\ 1 & \dots & \dots & \dots \\ 1 & x_1^{(i)} & \dots & x_m^{(i)} \\ 1 & \dots & \dots & \dots \\ 1 & x_1^{(n)} & \dots & x_m^{(n)} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(i)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- How to perform polynomial regression for one dimensional  $x$ ?
  - $\hat{y} = w_0 + w_1x + w_2x^2 \dots + w_mx^m$ .
  - Let  $x_j^{(i)} = (x_1^{(i)})^j \implies$  Polynomial least square fitting  
(<http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html>)

High-level idea:

- Any  $\mathbf{w}$  is possible, but some  $\mathbf{w}$  is most likely.
- $P(y^{(i)} | \hat{y}^{(i)}) = f_i(\mathbf{w})$
- Assuming independence of training examples, the likelihood of the training dataset is  $\prod_i f_i(\mathbf{w})$ .
- We shall choose the  $\mathbf{w}^*$  that **maximizes** the likelihood.
  - Maximum likelihood estimation (MLE)
  - If we also incorporate some prior on  $\mathbf{w}$ , this becomes Maximum Posterior Estimation (MAP)
    - If we assume some Gaussian prior on  $\mathbf{w}$ , this will add a  $\ell_2$  regularization term to the objective function.
- Many models and their variants can be deemed as different ways of estimating  $P(y^{(i)} | \hat{y}^{(i)})$

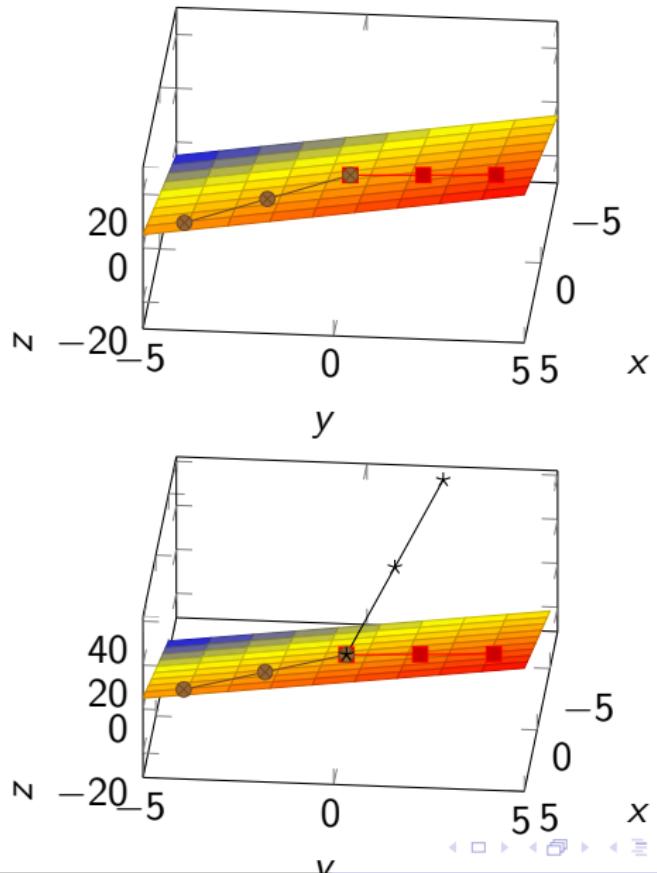
Find  $\mathbf{w}$  such that  $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2$  is minimized.

- What is  $\mathbf{X}\mathbf{w}$  when  $\mathbf{X}$  is fixed?
  - It is the hyperplane spanned by the  $d$  column vectors of  $\mathbf{X}$ .
- $\mathbf{y}$  in general is a vector outside the hyperplane. So the minimum distance is achieved when  $\mathbf{X}\mathbf{w}^*$  is exactly the projection of  $\mathbf{y}$  on the hyperplane. This means (denote  $i$ -th column of  $\mathbf{X}$  as  $X_i$ )

$$\left. \begin{array}{l} X_1^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \\ X_2^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \\ \dots \\ X_d^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \end{array} \right\} \Rightarrow \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$$

- $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^+ \mathbf{y}$       ( $\mathbf{X}^+$ : pseudo inverse of  $\mathbf{X}$ )

# Illustration



# Logistic Regression

Special case:  $y^{(i)} \in \{0, 1\}$ .

- Not appropriate to directly regress  $y^{(i)}$ .
- Rather, **model**  $y^{(i)}$  as the observed outcome of a Bernoulli trial with an unknown parameter  $p_i$
- How to model  $p_i$ 
  - We assume that  $p_i$  depends on  $\mathbf{x} \triangleq \mathbf{X}_{i\bullet} \implies$  rename  $p_i$  to  $p_{\mathbf{x}}$ .
  - Still hard to estimate  $p_{\mathbf{x}}$  reliably.
    - MLE:  $p_{\mathbf{x}} = \mathbf{E}[y = 1 | \mathbf{x}]$
    - What can we say about  $p_{\mathbf{x}+\epsilon}$  when  $p_{\mathbf{x}}$  is given?
- Answer: we impose a linear relationship between  $p_{\mathbf{x}}$  and  $\mathbf{x}$ 
  - What about a simple linear model  $p_{\mathbf{x}} = \mathbf{w}^T \mathbf{x}$  for some  $\mathbf{w}$ ?  
(Note: all points share the same parameter  $\mathbf{w}$ )
  - Problem: mismatch of the domains:  $\mathbf{x}$  vs  $p_{\mathbf{x}}$
  - Solution: mean function / inverse of link function:  
$$g^{-1} : \mathfrak{R} \rightarrow \text{params}$$

- Solution: Link function  $g(\text{parameters}) \rightarrow \Re$

$$g(p) = \text{logit}(p) \triangleq \log \frac{p}{1-p} = \mathbf{w}^\top \mathbf{x} \quad (3)$$

- Equivalently, solve for  $p$ .

$$p = \frac{e^{\mathbf{w}^\top \mathbf{x}}}{1 + e^{\mathbf{w}^\top \mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}} = \sigma(\mathbf{w}^\top \mathbf{x}) \quad (4)$$

Where  $\sigma(z) = \frac{1}{1+\exp(-z)}$ .

Recall that  $p_x = \mathbf{E}[y = 1 | \mathbf{x}]$ .

- Decision boundary is  $p \geq 0.5$ .
  - Equivalent to whether  $\mathbf{w}^\top \mathbf{x} \geq 0$ . Hence, LR is a linear classifier.

# Learning the Parameter $\mathbf{w}$

- Consider a training data point  $\mathbf{x}^{(i)}$ .
  - Recall that the conditional probability ( $\Pr[y^{(i)} = 1 | \mathbf{x}^{(i)}]$ ) computed by the model is denoted by the shorthand notation  $p$  (which is a function of  $\mathbf{w}$  and  $\mathbf{x}^{(i)}$ ).
  - The likelihood of  $\mathbf{x}^{(i)}$  is  $\begin{cases} p & , \text{ if } y^{(i)} = 1 \\ 1 - p & , \text{ otherwise} \end{cases}$ , or equivalently,  
$$p^{y^{(i)}} (1 - p)^{1 - y^{(i)}}.$$
- Hence, the likelihood of the whole training dataset is

$$L(\mathbf{w}) = \prod_{i=1}^n p(\mathbf{x}^{(i)})^{y^{(i)}} (1 - p(\mathbf{x}^{(i)}))^{1 - y^{(i)}}.$$

- Log-likelihood is (assume  $\log \triangleq \ln$ )

$$\ell(\mathbf{w}) = \sum_{i=1}^n y^{(i)} \log p(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - p(\mathbf{x}^{(i)})) \quad (5)$$

# Learning the Parameter $\mathbf{w}$

- To maximize  $\ell$ , notice that it is concave. So take its partial derivatives

$$\begin{aligned}\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} &= \sum_{i=1}^n \left( y^{(i)} \frac{1}{p(\mathbf{x}^{(i)})} \frac{\partial p(\mathbf{x}^{(i)})}{\partial \mathbf{w}_j} + (1 - y^{(i)}) \frac{1}{1 - p(\mathbf{x}^{(i)})} \frac{\partial (1 - p(\mathbf{x}^{(i)}))}{\partial \mathbf{w}_j} \right) \\ &= \sum_{i=1}^n \left( \mathbf{x}^{(i)}_j y^{(i)} - \mathbf{x}^{(i)}_j p(\mathbf{x}^{(i)}) \right)\end{aligned}$$

- and set them to 0 essentially means, for all  $j$

$$\sum_{i=1}^n \hat{y}^{(i)} \cdot \mathbf{x}^{(i)}_j = \sum_{i=1}^n p(\mathbf{x}^{(i)}) \mathbf{x}^{(i)}_j = \sum_{i=1}^n y^{(i)} \cdot \mathbf{x}^{(i)}_j$$

what the model predicts

what the data says

# Understand the Equilibrium

- Consider one dimensional  $x$ . The above condition is simplified to

$$\sum_{i=1}^n p^{(i)} x^{(i)} = \sum_{i=1}^n y^{(i)} x^{(i)}$$

- The RHS is essentially the sum of  $x$  values **only** for the training data in class  $Y = 1$ .
- The LHS says: if we use our learned model to assign a probability (of belonging to the class  $Y = 1$ ) for **every** training data, the LHS is the expected sum of  $x$  values.
- If this is still abstract, think of an example.

- There is no closed-form solution to maximize  $\ell$ .
- Use the *Gradient Ascent* algorithm to maximize  $\ell$ .
- There are faster algorithms.

# (Stochastic) Gradient Ascent

- $\mathbf{w}$  is initialized to some random value (e.g.,  $\mathbf{0}$ ).
- Since the gradient gives the *steepest* direction to increase a function's value, we move a small step towards that direction, i.e.,

$$w_j \leftarrow w_j + \alpha \frac{\partial \ell(\mathbf{w})}{\partial w_j} \quad , \text{ or}$$

$$w_j \leftarrow w_j + \alpha \sum_{i=1}^n (y^{(i)} - p(\mathbf{x}^{(i)})) \mathbf{x}^{(i)}_j$$

where  $\alpha$  (*learning rate*) is usually a small constant, or decreasing over the epochs.

- Stochastic version: using the gradient on a **randomly** selected training instance, i.e.,

$$w_j \leftarrow w_j + \alpha (y^{(i)} - p(\mathbf{x}^{(i)})) \mathbf{x}^{(i)}_j$$

# Newton's Method

- Gradient Ascent moves to the “right” direction a tiny step at a time. Can we find a good step size?
- Consider 1D case: **minimize**  $f(x)$  and the current point is  $a$ .
  - $f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2}(x - a)^2$  for  $x$  near  $a$ .
  - To minimize  $f(x)$ , take  $\frac{\partial f(x)}{\partial x} = 0$ , i.e.,

$$\frac{\partial f(x)}{\partial x} = 0$$

$$\Leftrightarrow f'(a) \cdot 1 + \frac{f''(a)}{2} \cdot 2(x - a) \cdot 1 = f'(a) + f''(a)(x - a) = 0$$

$$\Leftrightarrow x = a - \frac{f'(a)}{f''(a)}$$

- Can be applied to multiple dimension cases too  $\Rightarrow$  need to use  $\nabla$  (gradient) and Hess (Hessian).

- Regularization is another method to deal with **overfitting**.
  - It is designed to penalize large values of the model parameters.
  - Hence it *encourages* simpler models, which are less likely to overfit.
- Instead of optimizing for  $\ell(\mathbf{w})$ , we optimize  $\ell(\mathbf{w}) + \lambda R(\mathbf{w})$ .
  - $\lambda$  is a hyper-parameter that controls the strength of regularization.
    - It is usually determined by cross validating with a list of possible values (e.g., 0.001, 0.01, 0.1, 1, 10, ...)
    - Grid search: [http://scikit-learn.org/stable/modules/grid\\_search.html](http://scikit-learn.org/stable/modules/grid_search.html)
    - There are alternative methods.
  - $R(\mathbf{w})$  quantifies the “size” of the model parameters. Popular choices are:
    - $L_2$  regularization (**Ridge LR**)  $R(\mathbf{w}) = \|\mathbf{w}\|_2$
    - $L_1$  regularization (**Lasso LR**)  $R(\mathbf{w}) = \|\mathbf{w}\|_1$
    - $L_1$  regularization is more likely to result in sparse models.

# Generalizing LR to Multiple Classes

- LR can be generalized to multiple classes  $\Rightarrow$  MaxEnt.

$$\Pr[c \mid \mathbf{x}] \propto \exp\left(\mathbf{w}_c^\top \mathbf{x}\right) \implies \Pr[c \mid \mathbf{x}] = \frac{\exp\left(\mathbf{w}_c^\top \mathbf{x}\right)}{Z}$$

- $Z$  is the normalization constant.
- Let  $c^*$  be the last class in  $C$ , then  $\mathbf{w}_{c^*} = \mathbf{0}$ .
- Derive LR from MaxEnt
- Both belong to *exponential* or *log-linear* classifiers.

- Andrew Ng's note:  
<http://cs229.stanford.edu/notes/cs229-notes1.pdf>
- Cosma Shalizi's note: <http://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf>
- Tom Mitchell's book chapter: <https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf>

# COMP9318: HIDDEN MARKOV MODEL

Wei Wang, University of New South Wales

# Outline

2

- Markov Model
- Hidden Markov Model
  - Definition and basic problems
  - Decoding
  - Proj1

# Applications

3

- On-line handwriting recognition
- Speech recognition
- Gesture recognition
- Language modeling
- Motion video analysis and tracking
- Protein sequence/gene sequence alignment
- Stock price prediction
- ...

# What's HMM?

4

- Hidden Markov Model
  - Hidden
  - Markov

# Markov Model

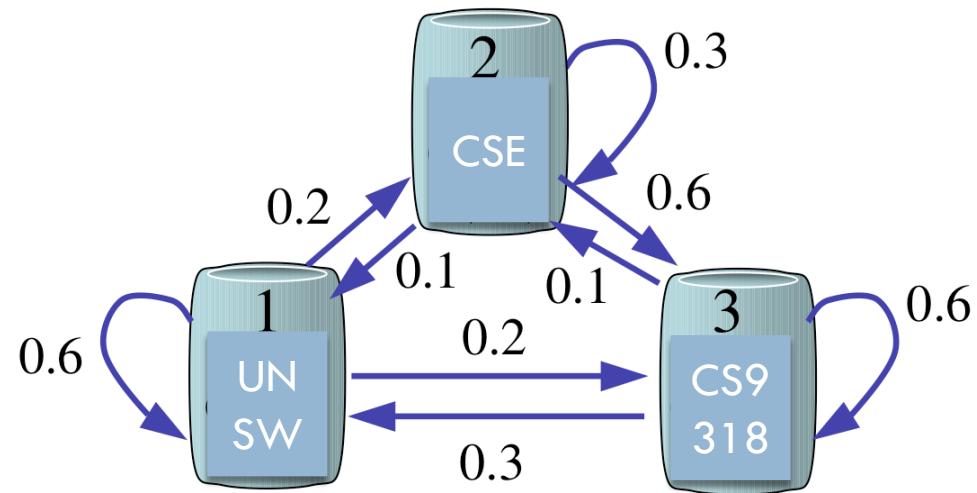
5

- The model (and some notations):
  - States:  $Q: \{q_0, q_1, \dots, q_{N-1}\}$
  - State sequence:  $X = \{x_i\}$ ; each  $x_i$  takes a value in  $Q$
  - State transition probabilities:  $A_{u \rightarrow v}$
  - Initial state distribution:  $\pi$
- Markov assumption (order = 1)
  - $\Pr[x_{i+1} | x_0, x_1, \dots, x_i] = \Pr[x_{i+1} | x_i]$
  - Limited memory

# Example

6

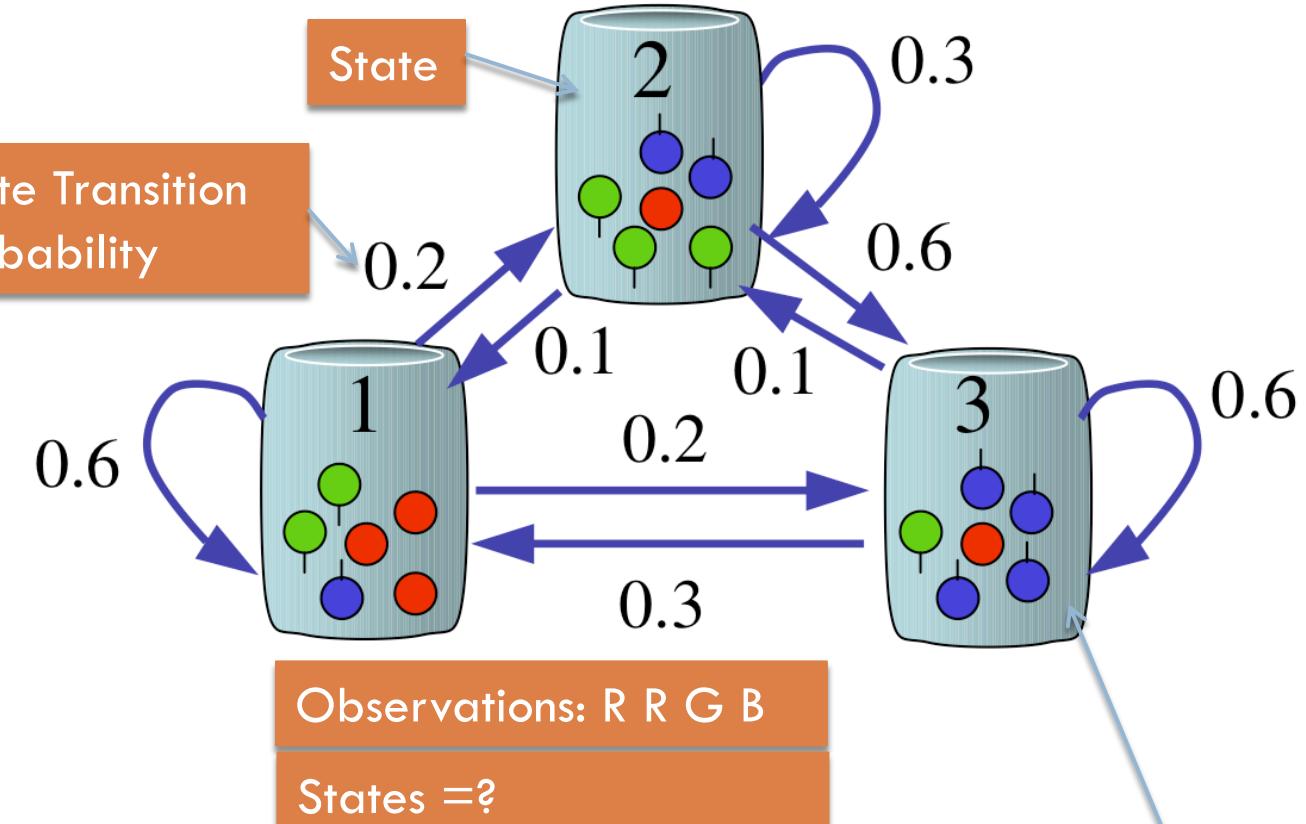
- Google's PageRank:
  - States: webpages
  - State sequence: sequence of webpages one visited
  - State transition probabilities:
    - $A_{u \rightarrow v} = \frac{\text{\#-outlinks-from-page-u-to-v}}{\text{\#-out-links-at-page-u}}$
    - (actually a bit more complex)
  - Initial state distribution:  $\pi$  = uniform on all pages
- Markov assumption (order = 1)
  - $\Pr[x_{i+1} | x_0, x_1, \dots, x_i] = \Pr[x_{i+1} | x_i]$
  - Randomly click an out link on page i



# Sequence Probability

7

- What's the probability of the state sequence being  $Q = q_0 \ q_1 \ \dots \ q_T$  ?
- Chain rule + Markov assumption
  - $\Pr[Q \mid \text{model}=\lambda] = \Pr[q_0 \mid \lambda]$ 
    - \*  $\Pr[q_1 \mid q_0, \lambda]$
    - \*  $\Pr[q_2 \mid q_0, q_1, \lambda] * \dots$
    - \*  $\Pr[q_T \mid q_0, q_1, \dots, q_{T-1}, \lambda]$
  - =  $\Pr[q_0 \mid \lambda] * (\Pr[q_1 \mid q_0, \lambda] * \Pr[q_2 \mid q_1, \lambda] * \dots * \Pr[q_T \mid q_{T-1}, \lambda])$

Example Hidden:

- States are hidden
- However, each state emits a symbol according to a distribution  $B(u \rightarrow \alpha)$

 Additional notations

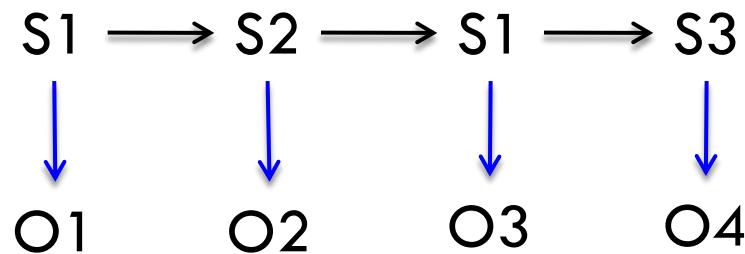
- Symbols: 0, 1, 2, ..., M-1
- Observed symbol sequence:  $O_0, O_1, \dots, O_{T-1}$

Symbol Emission Probability  
(Green = 1/6;  
Red = 1/6; Blue = 4/6)

# The Generative Process

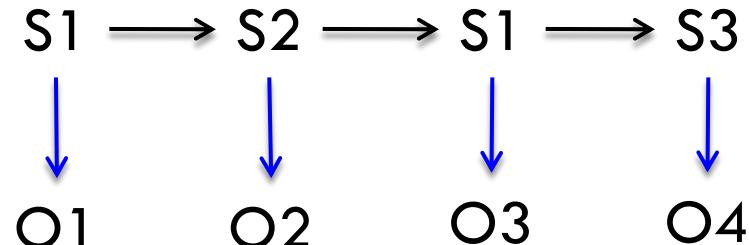
9

- Loop
  - Pick the next state the transit to
  - Transit to the chosen state, and generate an output symbol
- All according to the pmf of the distributions



# 3 Problems

10



## □ P1: Model Evaluation Problem

- What's the probability of seeing this **observation sequence**, given the HMM model  $\lambda$ ?
- Compute  $\Pr[O_0, O_1, \dots, O_{T-1} | \lambda]$

Forward algorithm

## □ P2: Decoding Problem

- What is the most likely state sequence (Q) corresponding to this **observation sequence**, given the HMM model  $\lambda$ ?
- Argmax<sub>Q</sub>  $\Pr[Q=q_0, q_1, \dots, q_{T-1} | O_0, O_1, \dots, O_{T-1}, \lambda]$

Viterbi algorithm

proj1

## □ P3: Learning the model

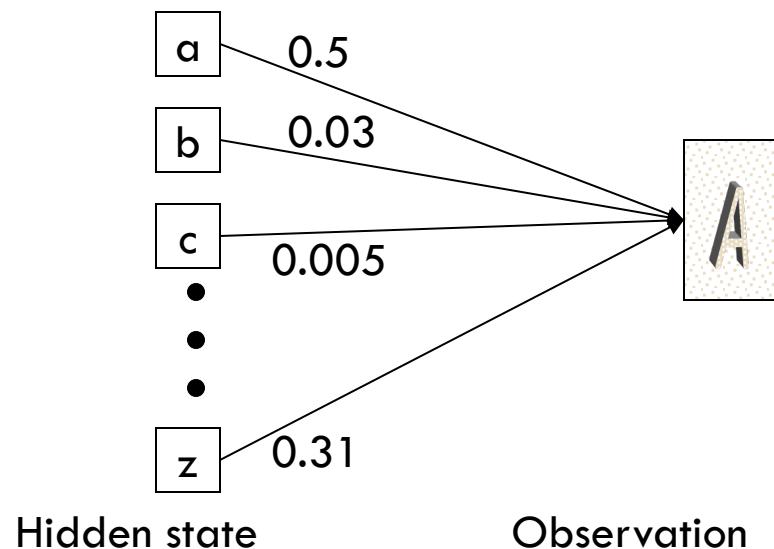
- What is the most likely parameters that generates this **observation sequence**?

Baum-Welch algorithm

# Application: Typed word recognition

11

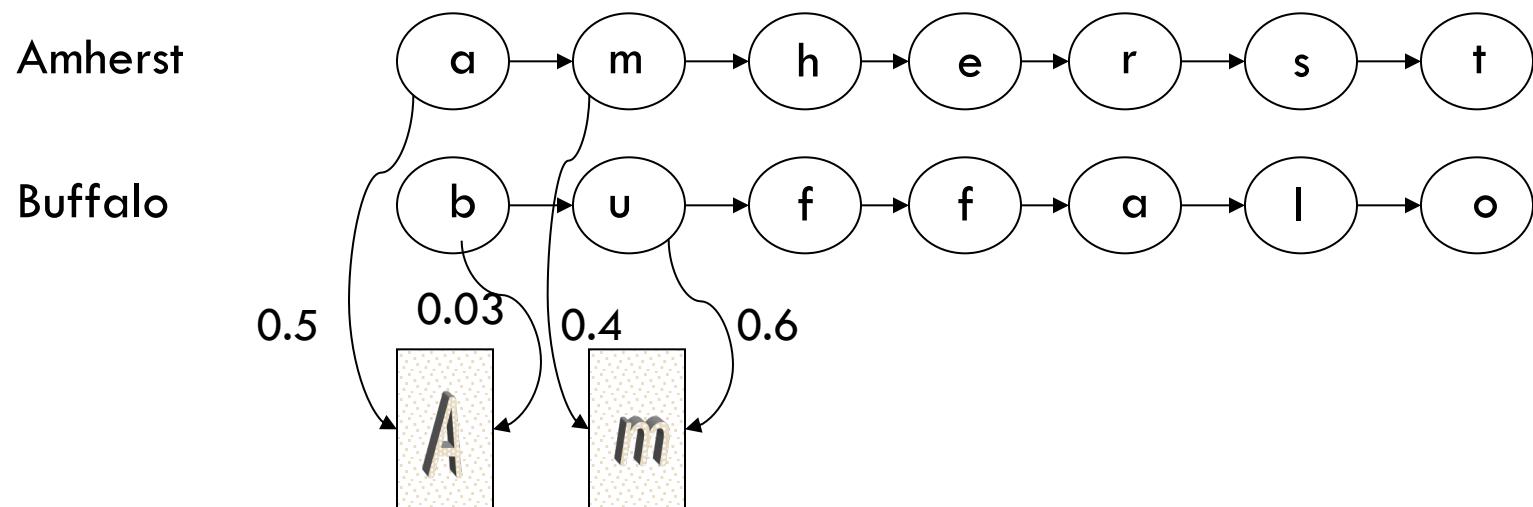
- Assume all chars are separated
- Character recognizer outputs probability of the image being particular character,  $P(\text{image} | \text{char})$ .
- There are infinite number of observations though



# Casting into the Evaluation Problem

12

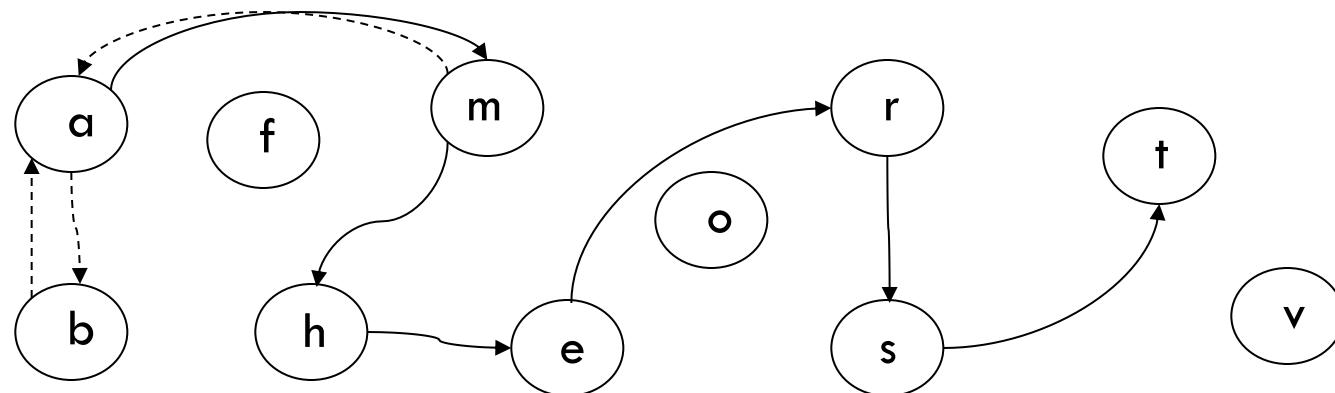
- Assume the lexicon is given
- Construct **separate** HMM models for each lexicon word
- Pick the model whose generation probability is the maximum



# The Other Approach

13

- Construct a **single** HMM models for all lexicon words
- Pick the best state sequence (= char sequence) whose generation probability is the maximum
- This is actually the **decoding problem**



# Decoding Problem (P2)

14

- Naïve algorithm:
  - Enumerate all possible state sequence and *evaluate their probability of generating the observations (next slide)*
  - Pick the one whose resulting probability is the highest
  - Problem: time complexity =  $O(N^T * T)$
- Viterbi: Dynamic programming-based method
  - **Attempt:** if we “magically” know best state sequence for RRG, can we know what’s the best state sequence for RRGB?
    - No. (Give an counter example)
  - **Remedy:** best state sequence for RRGB must come from the best state sequence ending at **some** state for the last observation. We don’t know which, but we can compute all.

# Joint Probability

Such joint probability is useful for several inference problems on HMM

15

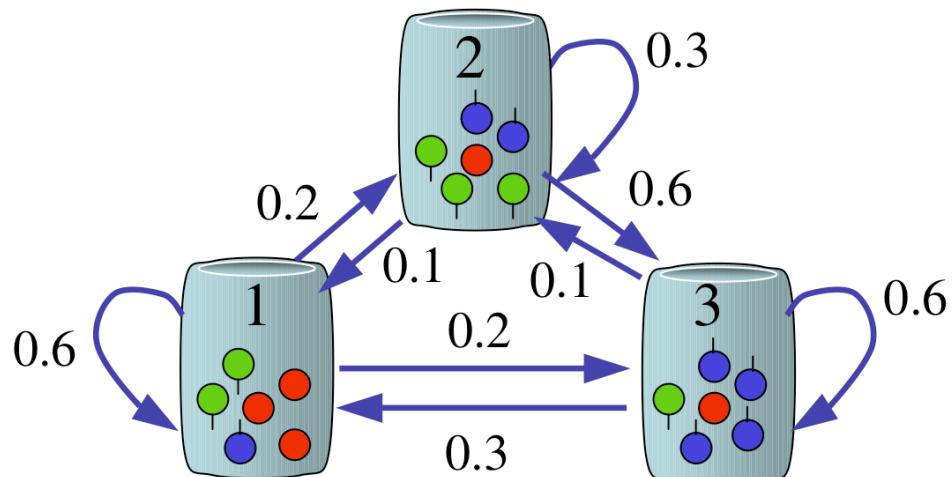
All conditioned on  $\lambda$

- $\Pr[\mathbf{O}_0, \mathbf{O}_1, \dots, \mathbf{O}_{T-1}, \mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_{T-1} | \lambda] = (\text{I}) * (\text{II})$
- (I)  $\Pr[\mathbf{O}_0, \mathbf{O}_1, \dots, \mathbf{O}_{T-1} | \mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_{T-1}, \lambda] = \Pr[\mathbf{O}_0 | \mathbf{q}_0, \lambda] * \Pr[\mathbf{O}_1 | \mathbf{q}_1, \lambda] * \dots * \Pr[\mathbf{O}_{T-1} | \mathbf{q}_{T-1}, \lambda]$
- (II)  $\Pr[\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_{T-1} | \lambda] = \Pr[\mathbf{q}_0 | \lambda] * \Pr[\mathbf{q}_1 | \mathbf{q}_0, \lambda] * \dots * \Pr[\mathbf{q}_{T-1} | \mathbf{q}_{T-2}, \lambda]$
- Example ( $\pi[\mathbf{q}_i] = 1/3$ )

States: 1 1 2 3

Observations: R R G B

- (I)  $= (3/6) (3/6) (3/6) (4/6)$
- (II)  $= (1/3) (0.6) (0.2) (0.6)$



# Viterbi Algorithm

16

- Define  $\delta[O_t \rightarrow q_i]$  as the best probability of any state sequence such that the symbol at timestamp  $t$ , denoted as  $O_t$ , corresponds to state  $q_i$
- Recursive formula:

$$\delta[O_t \rightarrow q_i] = \max_{u \in [0, N-1]} (\delta[O_{t-1} \rightarrow q_u] * A[q_u \rightarrow q_i] * B[q_i \rightarrow O_t])$$

- Boundary condition:

$$\delta[O_0 \rightarrow q_i] = \pi[q_i] * B[q_i \rightarrow O_0]$$

# Viterbi Algorithm

17

- Define  $\delta[O_t \rightarrow q_i]$  as the best probability of any state sequence such that the symbol  $O_t$  corresponds to state  $q_i$
- Recursive formula:

$$\delta[O_t \rightarrow q_i] = \max_{u \in [0, N-1]} (\delta[O_{t-1} \rightarrow q_u] * A[q_u \rightarrow q_i] * B[q_i \rightarrow O_t])$$

- Boundary condition:

$$\delta[O_0 \rightarrow q_i] = \pi[q_i] * B[q_i \rightarrow O_0]$$

- Easy to find the computing order in DP is from  $O_0$  to  $O_{T-1}$ , within which we loop over all the states.

# Example of Viterbi Algorithm

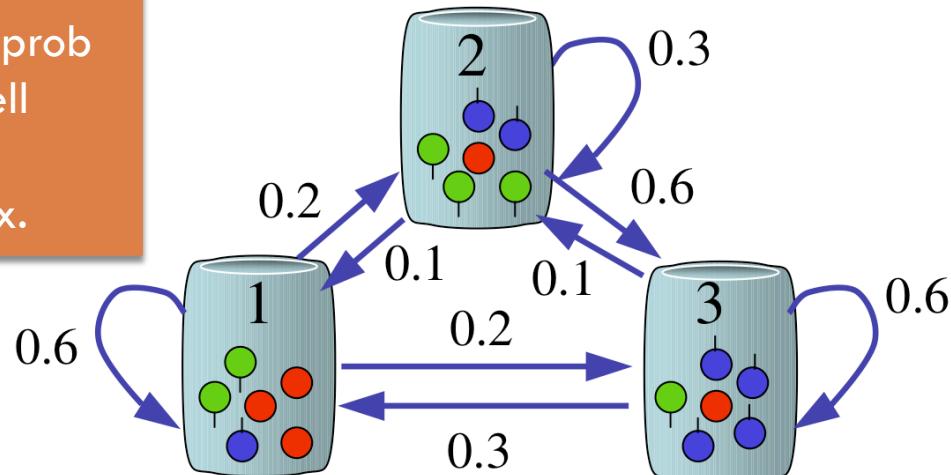
18

State\Symbol	R	R	G	B
<b>State = 1</b>	$(1/3)*(3/6)=1/6$	$1/20$	??	
<b>State = 2</b>	$(1/3)*(1/6)=1/18$	?		
<b>State = 3</b>	$(1/3)*(1/6)=1/18$	???		

e.g., for the cell (State = 1, 2nd R), it considers:

1. Prev state is 1: prob =  $(1/6)*0.6*(3/6)$
2. Prev state is 2: prob =  $(1/18)*0.1*(3/6)$
3. Prev state is 3: prob =  $(1/18)*0.3*(3/6)$

Max of the three options is the first one with prob value of  $(1/20)$ , hence the value in the cell (and  $\delta[O_1 \rightarrow q_0]$ )  
Also “remembers” which prev state is the max.



# Example of Viterbi Algorithm

19

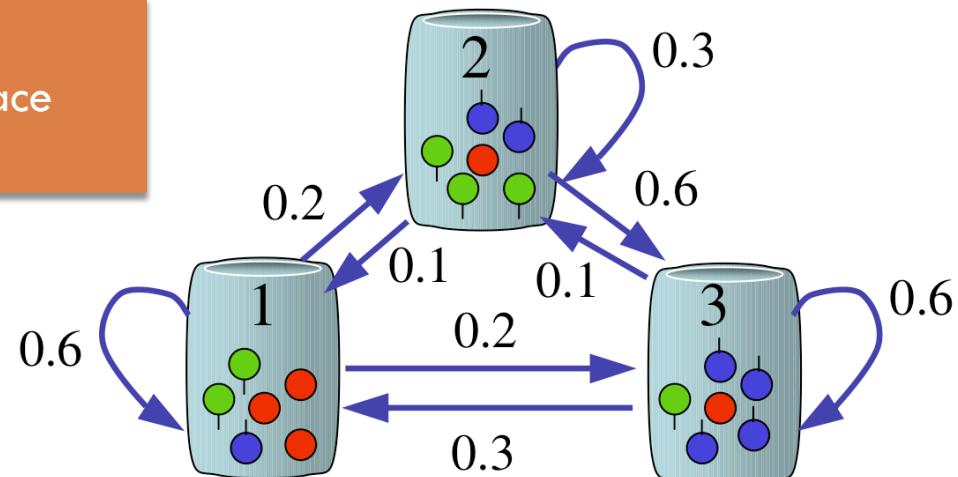
State \ Symbol	R	R	G	B
State = 1	$(1/3)*(3/6)=1/6$	1/20	1/100	1/1000
State = 2	$(1/3)*(1/6)=1/18$	1/180	1/200	1/1500
State = 3	$(1/3)*(1/6)=1/18$	1/180 (all)	1/600	1/500

Tracing back, we know the best state sequence in terms of the generative probability of the observed symbol sequence is RRGB

Time complexity:  $O(T*N^2)$

Space complexity:  $O(T*N)$ , as we need to trace back

All computation of probabilities should be performed in the log space to avoid underflow. E.g.,  $\log(p1*p2) = \log(p1) + \log(p2)$



# A Brief Introduction to Proj1

20

- **Input:**
  - An HMM model
  - A test file; each line is an address to be parsed
- **Output:**
  - Top-k parsed results (i.e., state sequences) and their corresponding log-probability score
- **Notes**
  - Special states: BEGIN and END
  - Add-1 smoothing
  - Tokenization of the address line

# Address Parsing Example

21

- States: {BEGIN, ST#, STNM, STTYP, CITY, STATE, PSTCD}
- Symbols: ASCII strings
- Observed symbol sequence:
  - ▣ begin 221 Anzac Parade **Kingsford** NSW 2032 end
    - begin
    - ST#
    - STNM
    - STTYP
    - CITY
    - STATE
    - PSTCD
    - end
  - ▣ What's the most likely state sequence?
- Enables us to perform advanced tasks, such as deduplication and advanced queries
  - ▣ begin 10 **Kingsford** St, Fairy Meadow, NSW 2519 end
    - STNM

# Smoothing

22

- Emission probabilities
  - Let Symbols = {a, b, c, d, ..., z}
  - Without smoothing,  $\Pr[S \rightarrow x] = \#(S, x) / \#(S)$
  - Hence if  $\#(S, x) = 0$ , the probability is 0.
  - With add-1 smoothing,  $\Pr[S \rightarrow x] = [\#(S, x) + 1] / [\#(S) + |\text{Symbols}| + 1]$ 
    - Denominator needs +1 for Out-of-vocabulary (OOV) symbol
- State transition probabilities
  - With add-1 smoothing,  $\Pr[S_1 \rightarrow S_2] = [\#(S_1, S_2) + 1] / [\#(S_1) + |\text{States}|]$
- Special procedure to handle the BEGIN/END states (and its impact)

# References

23

- Section 5 in “A Revealing Introduction to Hidden Markov Models” by Mark Stamp.
- Sung-jung Cho, “Introduction to Hidden Markov Model and Its Application”
- Ankur Jain, “Hidden Markov Models”

# Hidden Markov Model

Wei Wang @ CSE, UNSW

April 3, 2019

- $P(S_{1:k} \mid O_{1:k}) \stackrel{\text{def}}{=} P(S_1, \dots, S_k \mid O_1, \dots, O_k)$
- What do we need?
  - $\arg \max P(S_{1:k} \mid O_{1:k})$
  - Can we derive even  $P(S_1 \mid O_1)$ ?

- $P(S_{1:k} | O_{1:k}) \stackrel{\text{def}}{=} P(S_1, \dots, S_k | O_1, \dots, O_k)$

- What do we need?

- $\arg \max P(S_{1:k} | O_{1:k})$
- Can we derive even  $P(S_1 | O_1)$ ?
- Need to use the Bayes rule:

$$P(S_{1:k} | O_{1:k}) = \frac{P(O_{1:k} | S_{1:k}) \cdot P(S_{1:k})}{P(O_{1:k})}$$

- Because of the “arg max”, we have

$$\begin{aligned} \arg \max_{S_{1:k}} P(S_{1:k} | O_{1:k}) &= \arg \max_{S_{1:k}} P(O_{1:k} | S_{1:k}) \cdot P(S_{1:k}) \\ &= \arg \max_{S_{1:k}} P(O_{1:k}, S_{1:k}) \end{aligned}$$

- Since we can compute the probabilities, the naive method is to consider all  $O(N^k)$  state sequences, where  $N$  is the number of states.

- A common trick (for Markov Chains) is to check if we can use recursion (thanks for the abundant conditional independences given by the Markov assumption).

Note:

$$\begin{aligned} P(O_{1:k+1} \mid S_{1:k+1}) &= P(O_{k+1} \mid O_{1:k}, S_{1:k+1}) \cdot P(O_{1:k} \mid S_{1:k+1}) \\ &= P(O_{k+1} \mid S_{k+1}) \cdot P(O_{1:k} \mid S_{1:k}) \end{aligned}$$

$$P(S_{1:k+1}) = P(S_{1:k}) \cdot P(S_{k+1} \mid S_k)$$

Then

$$\begin{aligned} &P(O_{1:k+1}, S_{1:k+1}) \\ &= P(O_{k+1} \mid S_{k+1}) \cdot P(O_{1:k} \mid S_{1:k}) \cdot P(S_{1:k}) \cdot P(S_{k+1} \mid S_k) \\ &= P(O_{k+1} \mid S_{k+1}) \cdot P(S_{k+1} \mid S_k) \cdot P(O_{1:k}, S_{1:k}) \end{aligned}$$

$$\begin{aligned} P(O_{1:k+1}, S_{1:k+1}) &= P(O_{1:k}, S_{1:k}) \cdot \left( P(O_{k+1} | S_{k+1}) \cdot P(S_{k+1} | S_k) \right) \\ &= P(O_{1:k}, S_{1:k}) \cdot f(S_k, S_{k+1}) \end{aligned}$$

- Note that the value of  $S_k$  is important when considering applying “max”.
- Define  $\delta(k, v) \stackrel{\text{def}}{=} \max_{S_{1:k-1}} P(O_{1:k}, S_{1:k-1}, v)$ , we have the recurrence:

$$\delta(k + 1, v) = \max_u (\delta(k, u) \cdot f(u, v))$$

with the boundary condition:

$$\delta(1, v) = P(O_1, v)$$

- Then

$$\max_{S_{1:k+1}} P(O_{1:k+1}, S_{1:k+1}) = \max_v \max_{S_{1:k}} P(O_{1:k+1}, S_{1:k}, v) = \max_v \delta(k + 1, v)$$

- The arg max solution can be obtained via backtracking or some additional data structure.

# **SUPPORT VECTOR MACHINE**

Mainly based on

<https://nlp.stanford.edu/IR-book/pdf/15svm.pdf>

# Overview

---

- SVM is a huge topic
  - Integration of MMDS, IIR, and Andrew Moore's slides here
- Our foci:
  - Geometric intuition → Primal form
    - Alternative interpretation from Empirical Risk Minimization point of view.
  - Understand the final solution (in the dual form)
    - Generalizes well to Kernel functions
  - SVM + Kernels

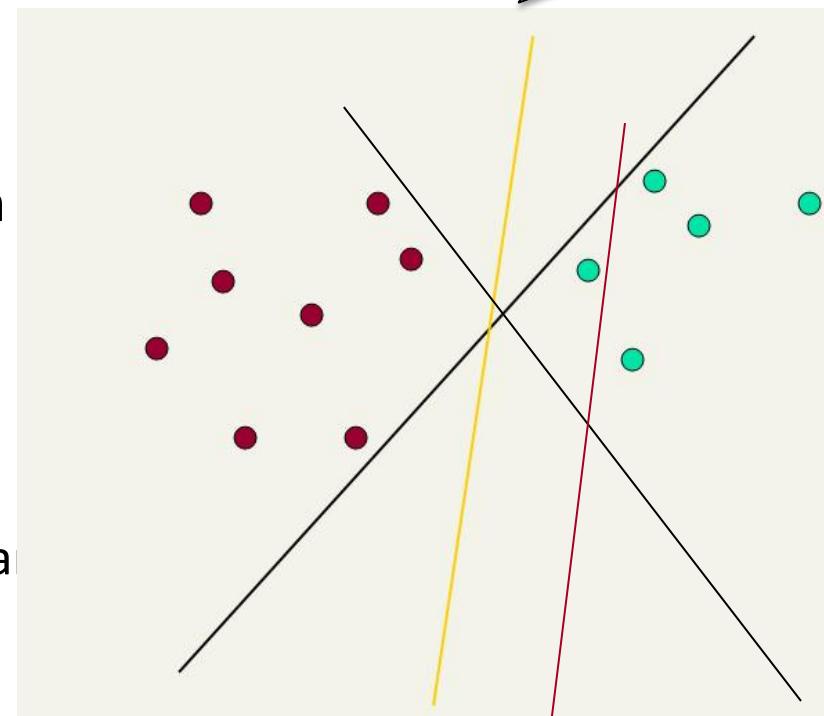
$$\mathbf{w}^T \mathbf{x}_i + b = 0$$

# Linear classifiers: Which Hyperplane?



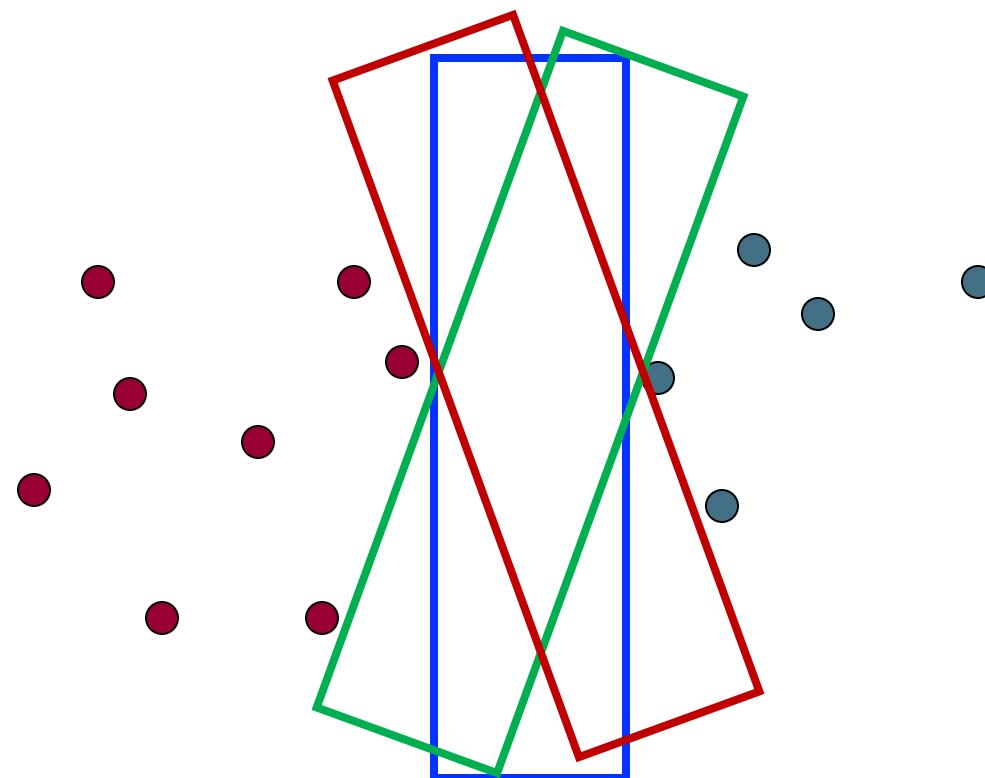
- Lots of possible solutions for  $a, b, c$ .
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
  - E.g., perceptron
- Support Vector Machine (SVM) finds an optimal\* solution.
  - Maximizes the distance between the hyperplane and the “difficult points” close to decision boundary
  - One intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions

This line represents the decision boundary:  
 $ax + by - c = 0$



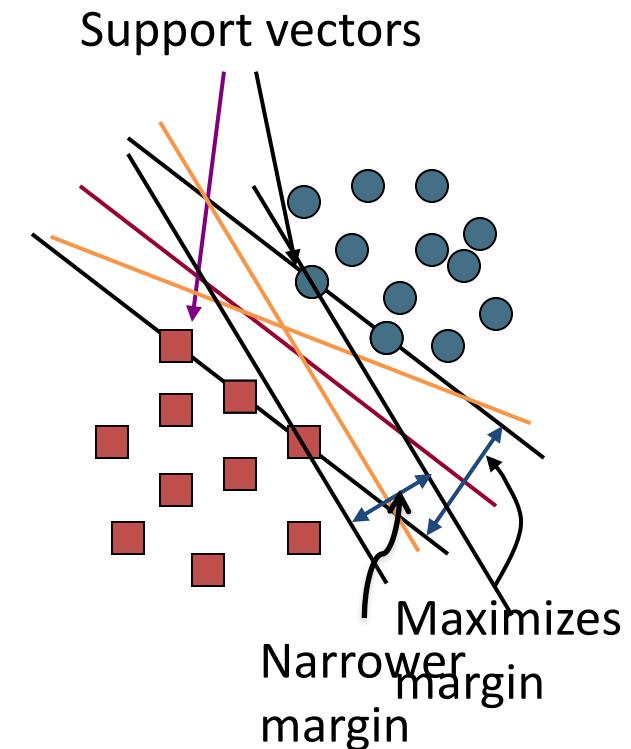
# Another intuition

- If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased



# Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
  - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Solving SVMs is a *quadratic programming* problem
- Seen by many as the most successful current text classification method\*



\*but other discriminative methods often perform very similarly

# Maximum Margin: Formalization

- $\mathbf{w}$ : decision hyperplane normal vector
- $\mathbf{x}_i$ : data point  $i$
- $y_i$ : class of data point  $i$  (+1 or -1)
- Classifier is:  $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T \mathbf{x}_i + b)$
- **Functional margin** of  $\mathbf{x}_i$  is:  $y_i (\mathbf{w}^T \mathbf{x}_i + b)$ 
  - But note that we can increase this margin simply by scaling  $\mathbf{w}$ ,  $b$ ....
- Functional margin of dataset is twice the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin

NB: Not 1/0

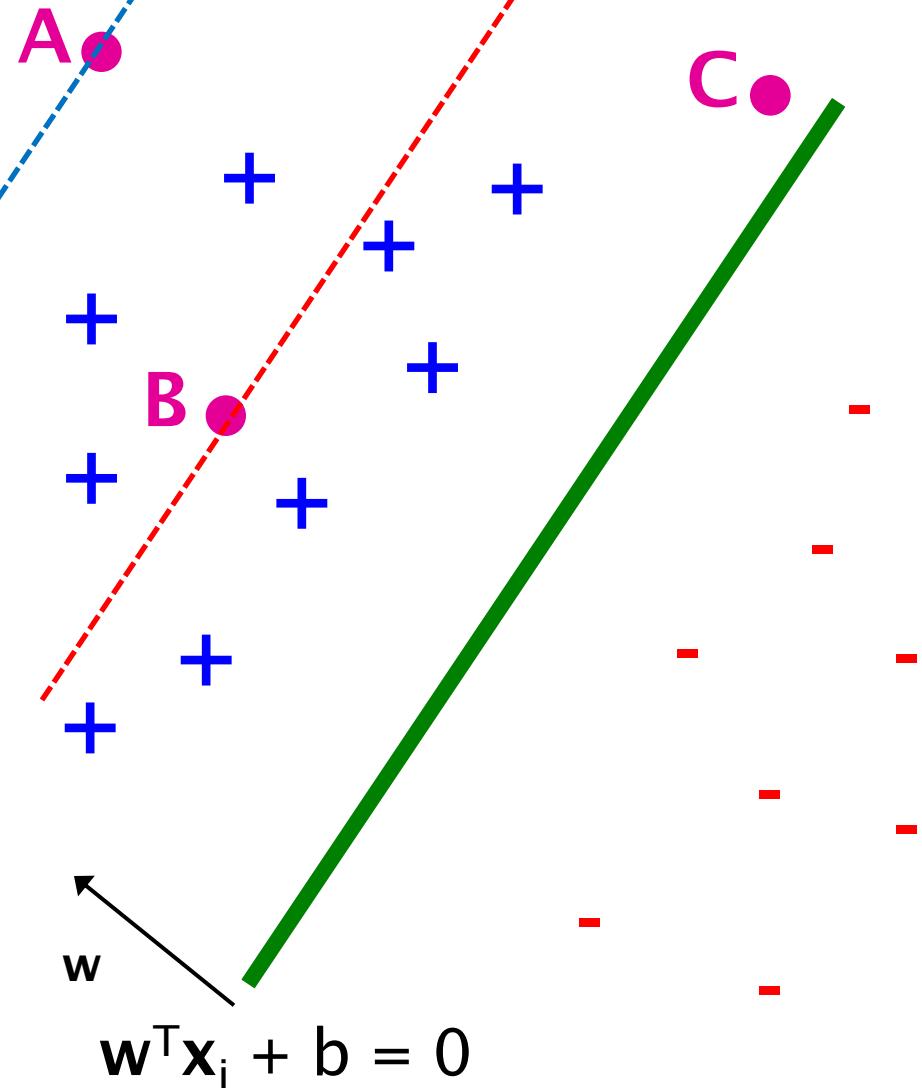


NB: a common  
trick

$$\mathbf{w}^T \mathbf{x}_i + b = 7.4$$

$$\mathbf{w}^T \mathbf{x}_i + b = 3.7$$

## Largest Margin

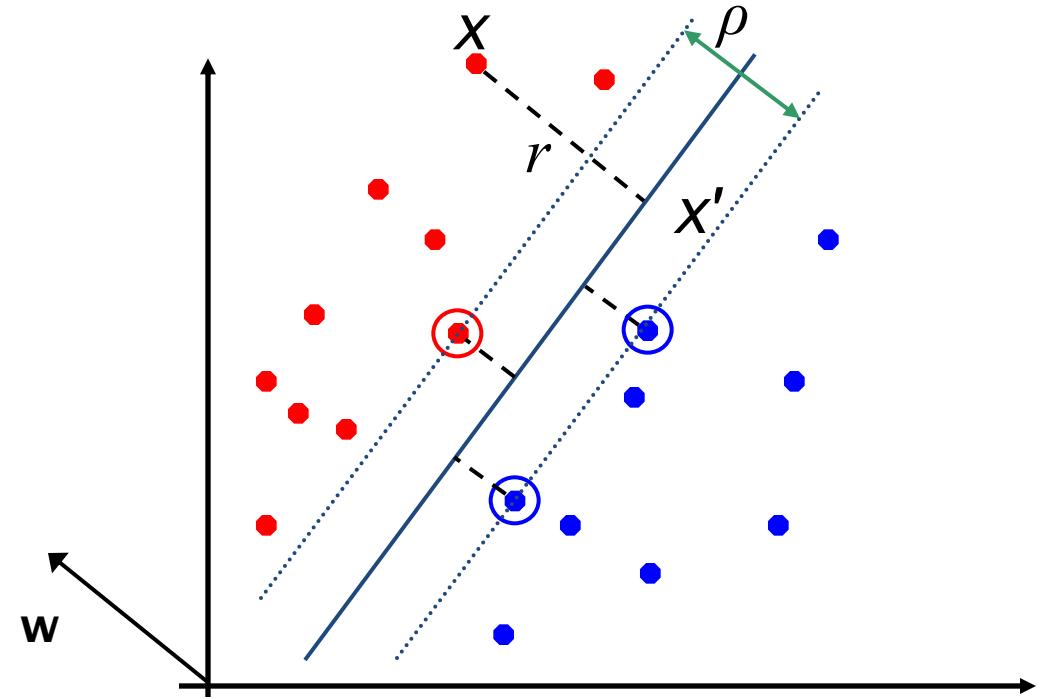


- Distance from the separating hyperplane corresponds to the “confidence” of prediction
- Example:**
  - We are more sure about the class of A and B than of C

# Geometric Margin

- Distance from example to the separator is
- Examples closest to the hyperplane are ***support vectors***.
- Margin  $\rho$**  of the separator is the width of separation between support vectors of classes.

$$r = y \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$$



## Algebraic derivation of finding $r$ :

Dotted line  $\mathbf{x}' - \mathbf{x}$  is perpendicular to decision boundary so parallel to  $\mathbf{w}$ . Unit vector is  $\mathbf{w}/|\mathbf{w}|$ , so line is  $r\mathbf{w}/|\mathbf{w}|$ , for some  $r$ .

$$\mathbf{x}' = \mathbf{x} - yr\mathbf{w}/|\mathbf{w}|.$$

$$\mathbf{x}' \text{ satisfies } \mathbf{w}^T \mathbf{x}' + b = 0.$$

$$\text{So } \mathbf{w}^T(\mathbf{x} - yr\mathbf{w}/|\mathbf{w}|) + b = 0$$

$$\text{Recall that } |\mathbf{w}| = \sqrt{\mathbf{w}^T \mathbf{w}}.$$

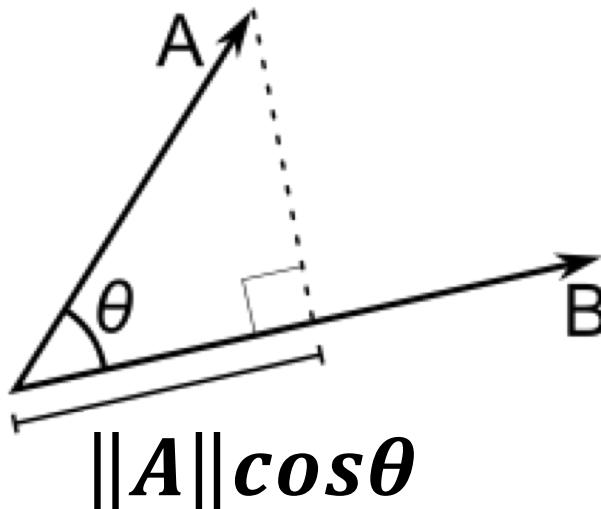
$$\text{So } \mathbf{w}^T \mathbf{x} - yr|\mathbf{w}| + b = 0$$

So, solving for  $r$  gives:

$$r = y(\mathbf{w}^T \mathbf{x} + b)/|\mathbf{w}|$$

# Help from Inner Product

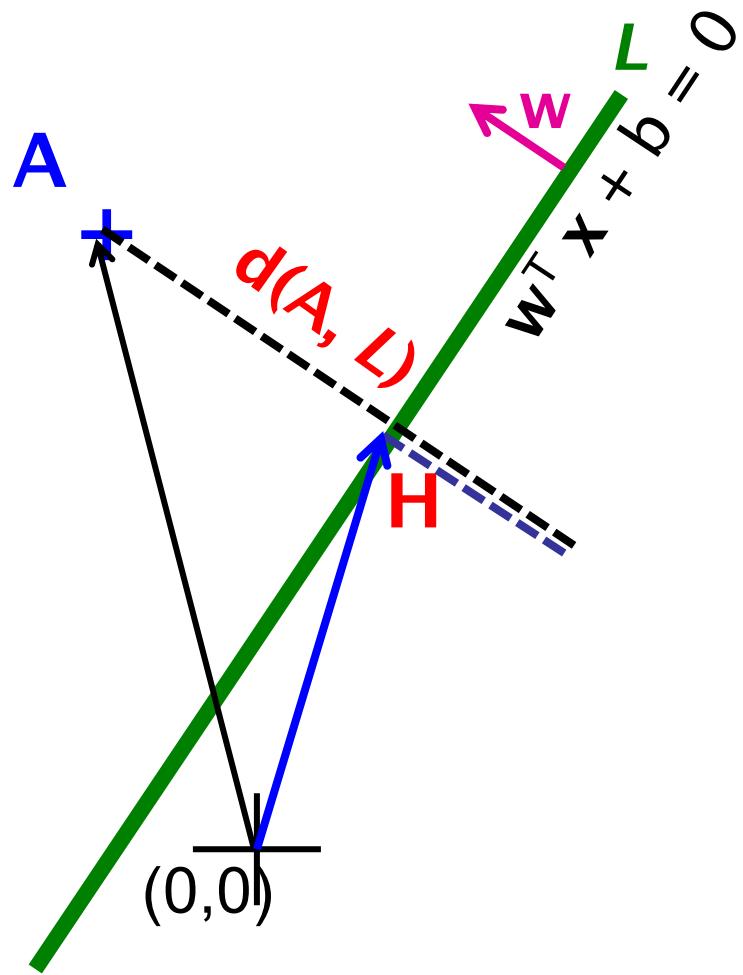
- Remember: Dot product / inner product



$$\langle A, B \rangle = \|A\| \|B\| \cos \theta$$

- (Or use an algebraic derivation similar to prev slide)
- A's projection onto B =  $\langle A, B \rangle / \langle B, B \rangle$

# Derivation of the Geometric Margin



- $d(A, L) = \langle A, w \rangle / \langle w, w \rangle$   
-  $\langle H, w \rangle / \langle w, w \rangle$
- Note that  $H$  is on the line  $L$ , so  
 $\langle w, H \rangle + b = 0$
- Therefore  $= (\langle A, w \rangle + b) / \langle w, w \rangle$

# Linear SVM Mathematically

## The linearly separable case

---

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set  $\{(\mathbf{x}_i, y_i)\}$

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality
- Then, since each example's distance from the hyperplane is

$$r = y \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

# Derivation of $\rho$

- Hyperplane

$$\mathbf{w}^T \mathbf{x} + b = 0$$

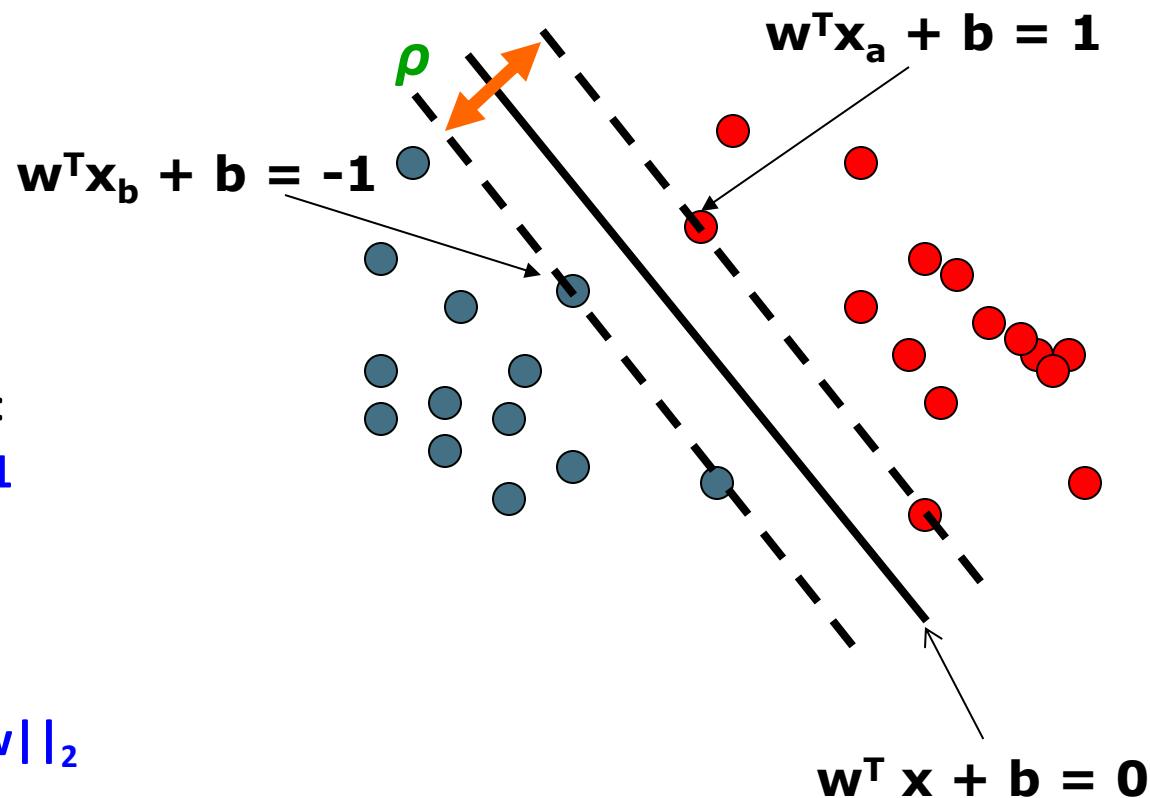
- Extra scale constraint:

$$\min_{i=1,\dots,n} |\mathbf{w}^T \mathbf{x}_i + b| = 1$$

- This implies:

$$\mathbf{w}^T (\mathbf{x}_a - \mathbf{x}_b) = 2$$

$$\rho = \| \mathbf{x}_a - \mathbf{x}_b \|_2 = 2 / \| \mathbf{w} \|_2$$



# Solving the Optimization Problem

Find  $\mathbf{w}$  and  $b$  such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$  is minimized;

and for all  $\{(\mathbf{x}_i, y_i)\}$ :  $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- This is now optimizing a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)
- The solution involves constructing a *dual problem* where a *Lagrange multiplier*  $\alpha_i$  is associated with every constraint in the primary problem:

Find  $\alpha_1 \dots \alpha_N$  such that

$Q(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$  is maximized and

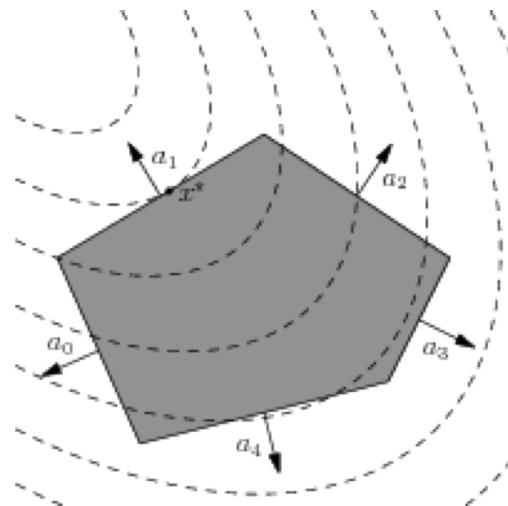
$$(1) \quad \sum \alpha_i y_i = 0$$

$$(2) \quad \alpha_i \geq 0 \text{ for all } \alpha_i$$

# Geometric Interpretation

Find  $\mathbf{w}$  and  $b$  such that  
 $\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$  is minimized;  
and for all  $\{(\mathbf{x}_i, y_i)\}$ :  $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- What are fixed and what are variables?
- **Linear** constraint(s): Where can the variables be?
- **Quadratic** objective function:



# The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \quad b = y_k - \mathbf{w}^T \mathbf{x}_k \text{ for any } \mathbf{x}_k \text{ such that } \alpha_k \neq 0$$

- Each non-zero  $\alpha_i$  indicates that corresponding  $\mathbf{x}_i$  is a support vector.
- Then the scoring function will have the form:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

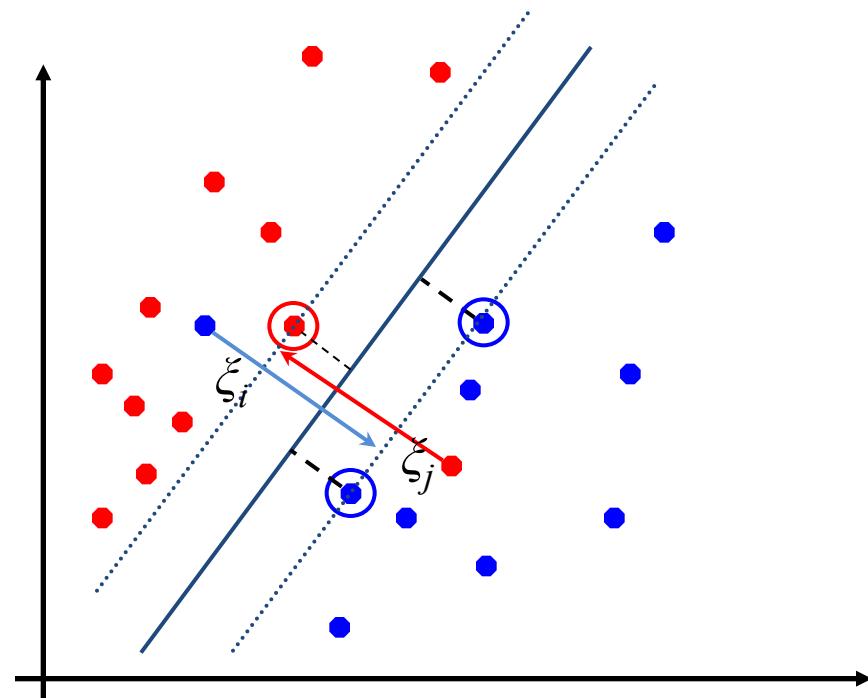
$$f(\mathbf{x}) = \sum_{sv \in SV} y_{sv} \cdot \alpha_{sv} \cdot \langle \mathbf{x}_{sv}, \mathbf{x} \rangle + b$$

*Q: What are the model parameters? What does f(x) mean intuitively?*

- Classification is based on the sign of  $f(\mathbf{x})$
- Notice that it relies on an **inner product** between the test point  $\mathbf{x}$  and the support vectors  $\mathbf{x}_i$ 
  - We will return to this later.
- Also *keep in mind* that solving the optimization problem involved computing the inner products  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$  between all pairs of training points.

# Soft Margin Classification

- If the training data is not linearly separable, *slack variables*  $\xi_i$  can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane “far” from each class (large margin)



# Soft Margin Classification

[Optional]

## Mathematically

---

- The old formulation:

Find  $\mathbf{w}$  and  $b$  such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

- The new formulation incorporating slack variables:

Find  $\mathbf{w}$  and  $b$  such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for all } i$$

- Parameter  $C$  can be viewed as a way to control overfitting
  - A regularization term

# Alternative View

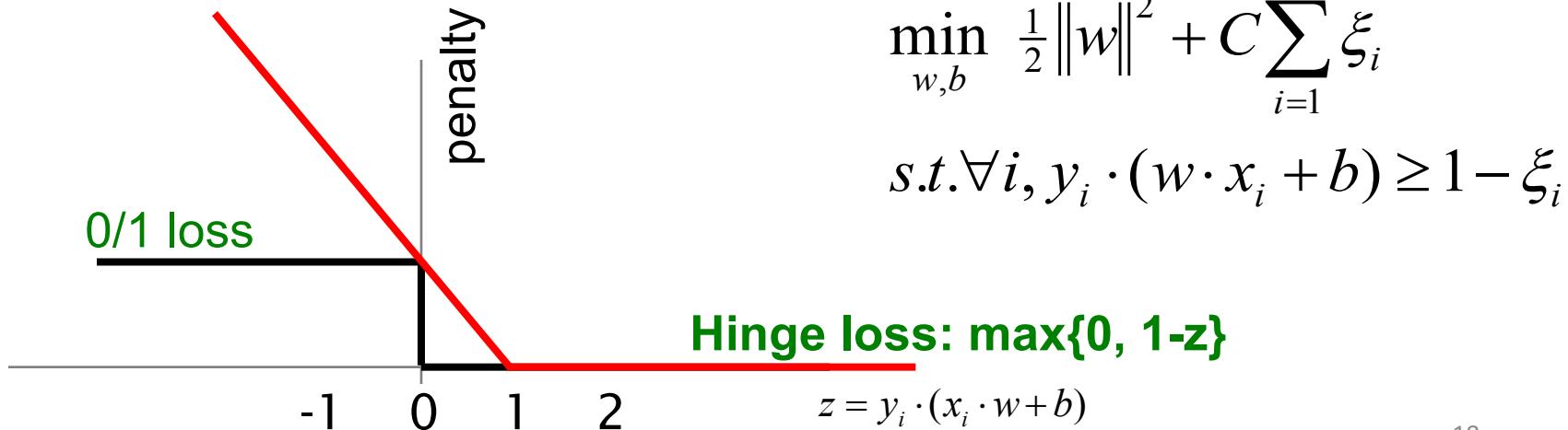
- SVM in the “natural” form

$$\arg \min_{w,b} \underbrace{\frac{1}{2} w \cdot w}_{\text{Margin}} + C \cdot \sum_{i=1}^n \max \{0, 1 - y_i (w \cdot x_i + b)\}$$

↑  
Empirical loss  $L$  (how well we fit training data)

Hyper-parameter related to regularization

- SVM uses “Hinge Loss”:



**[Optional]**

# Soft Margin Classification – Solution

- The dual problem for soft margin classification:

Find  $\alpha_1 \dots \alpha_N$  such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$  is maximized and

- (1)  $\sum \alpha_i y_i = 0$
- (2)  $0 \leq \alpha_i \leq C$  for all  $\alpha_i$

- Neither slack variables  $\xi_i$ , nor their Lagrange multipliers appear in the dual problem!
- Again,  $\mathbf{x}_i$  with non-zero  $\alpha_i$  will be support vectors.
- Solution to the dual problem is:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

$$b = y_k(1 - \xi_k) - \mathbf{w}^T \mathbf{x}_k \text{ where } k = \operatorname{argmax}_{k'} \alpha_{k'}$$

w is not needed explicitly for classification!

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

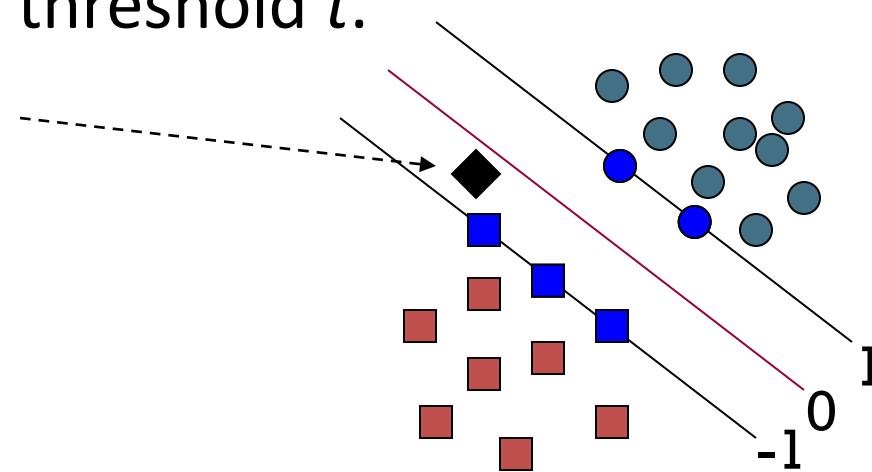
# Classification with SVMs

- Given a new point  $\mathbf{x}$ , we can score its projection onto the hyperplane normal:
  - i.e., compute score:  $\mathbf{w}^T \mathbf{x} + b = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$ 
    - Decide class based on whether  $<$  or  $> 0$
  - Can set confidence threshold  $t$ .

Score  $> t$ : yes

Score  $< -t$ : no

Else: don't know



# Linear SVMs: Summary

- The classifier is a *separating hyperplane*.
- The most “important” training points are the support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points  $\mathbf{x}_i$  are **support vectors** with non-zero Lagrangian multipliers  $\alpha_i$ .
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

Find  $\alpha_1 \dots \alpha_N$  such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$  is maximized and

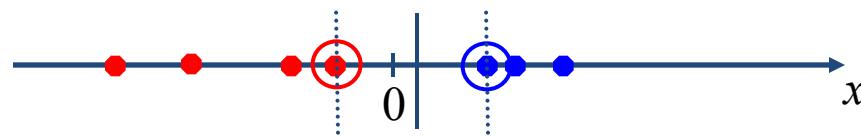
$$(1) \sum \alpha_i y_i = 0$$

$$(2) 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

$$f(\mathbf{x}) = \sum_{sv \in SV} y_{sv} \cdot \alpha_{sv} \cdot \langle \mathbf{x}_{sv}, \mathbf{x} \rangle + b$$

# Non-linear SVMs

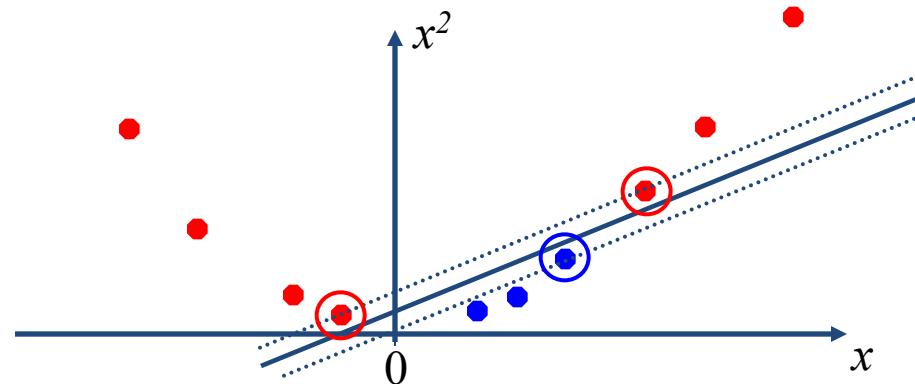
- Datasets that are linearly separable (with some noise) work out great:



- But what are we going to do if the dataset is just too hard?



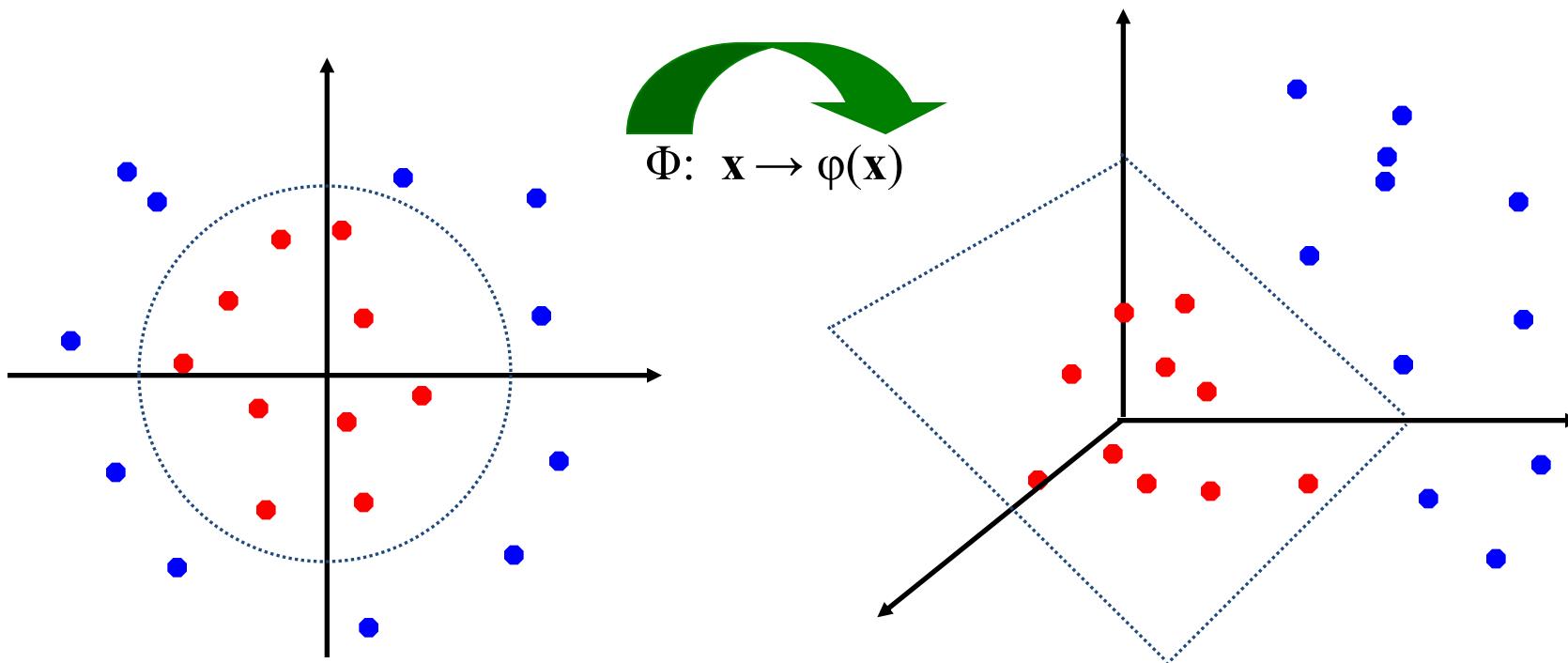
- How about ... mapping data to a higher-dimensional space:



c.f., polynomial regression

# Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



# The “Kernel Trick”

---

- The linear classifier relies on an inner product between vectors  
$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$
- If every datapoint is mapped into high-dimensional space via some transformation  $\Phi$ :  $\mathbf{x} \rightarrow \phi(\mathbf{x})$ , the inner product becomes:  
$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$
- A *kernel function* is some function that corresponds to an inner product in **some** expanded feature space.
  - Usually, no need to construct the feature space explicitly.

**What about**  $K(\mathbf{u}, \mathbf{v}) = (1 + \mathbf{u}^\top \mathbf{v})^3$  ?

# Example

$$\begin{aligned} K(\mathbf{u}, \mathbf{v}) &= (1 + \mathbf{u}^\top \mathbf{v})^2 \\ &= 1 + 2\mathbf{u}^\top \mathbf{v} + (\mathbf{u}^\top \mathbf{v})^2 \\ &= 1 + 2\mathbf{u}^\top \mathbf{v} + \left( \sum_i \mathbf{u}_i \mathbf{v}_i \right)^2 \end{aligned}$$

$$\begin{aligned} &= 1 + 2\mathbf{u}^\top \mathbf{v} + \left( \sum_i \mathbf{u}_i^2 \mathbf{v}_i^2 + \sum_{i \neq j} \mathbf{u}_i \mathbf{u}_j \mathbf{v}_i \mathbf{v}_j \right) \\ &= \phi(\mathbf{u})^\top \phi(\mathbf{v}) \end{aligned}$$

**O(d<sup>2</sup>) new cross-term features**

$$\phi(\mathbf{u}) = [1 \quad \sqrt{2}\mathbf{u}_1 \quad \dots \quad \sqrt{2}\mathbf{u}_d \quad \mathbf{u}_1^2 \quad \dots \quad \mathbf{u}_d^2 \quad \mathbf{u}_1 \mathbf{u}_2 \quad \dots \quad \mathbf{u}_{d-1} \mathbf{u}_d]^\top$$

$$\phi(\mathbf{v}) = [1 \quad \sqrt{2}\mathbf{v}_1 \quad \dots \quad \sqrt{2}\mathbf{v}_d \quad \mathbf{v}_1^2 \quad \dots \quad \mathbf{v}_d^2 \quad \mathbf{v}_1 \mathbf{v}_2 \quad \dots \quad \mathbf{v}_{d-1} \mathbf{u}_d]^\top$$

Linear

Non-linear

Non-linear + feature combination

# Why feature combinations?

---

- Examples:
  - Two categorical features (age & married) encoded as one-hot encoding → combination = conjunction rules
    - e.g.,  $1[\text{age in } [30, 40] \text{ AND married} = \text{FALSE}]$
  - [..., eagerness-for-travel, income, ...] → combination indicates how much to spend on travel
  - NLP, feature vector =  $1[w \in x]$  → combination indicates two word co-occurrence (where phrase/multi-word expression (MWE) is just a special case)
- $\mathbf{x} \rightarrow \varphi(\mathbf{x})$ , then a linear model in the new feature space is just  $\mathbf{w}^T \varphi(\mathbf{x}) + b$ 
  - each feature combination will be assigned a weight  $w_i$
  - irrelevant features combinations will get 0 weight

# Inner product in an **infinite** dimensional space!

[Optional]

RBF kernel:

$$e^{-\gamma \|x_i - x_j\|^2} = e^{-\gamma(x_i - x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2}$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2} \left( 1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \dots \right)$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2} \left( 1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}} x_i \cdot \sqrt{\frac{2\gamma}{1!}} x_j + \sqrt{\frac{(2\gamma)^2}{2!}} x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}} x_j^2 + \dots \right)$$

$$= \phi(x_i)^T \phi(x_j) \quad , \text{ where}$$

$$\phi(x) = e^{-\gamma x^2} \left[ 1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \sqrt{\frac{(2\gamma)^3}{3!}} x^3, \dots \right]^T$$

# String Kernel

---

- $K(s_1, s_2)$  should evaluate the similarity between the two strings
  - Without this,  $\text{sim}(\text{"actor"}, \text{"actress"}) = 0$
- Intuition:
  - consider all **substrings** as (binary) “features”
  - inner product in that “enhanced” feature space means the number of common substrings the two share.
- Variants:
  - (more complex): consider subsequences (with possibly gap penalty)
  - (simpler): consider all k-grams, and use Jaccard
    - $\text{bigrams(actor)} = \{\text{ac, ct, to, or}\}$
    - $\text{bigrams(actress)} = \{\text{ac, ct, tr, re, es, ss}\}$
    - $\text{Jaccard(actor, actress)} = 2/8$

# Kernels

- Why use kernels?

- Make non-separable problem separable.
  - Map data into better representational space
  - Can be learned to capture some notion of “similarity”

- Common kernels

- Linear
  - Polynomial  $K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^d$ 
    - Gives feature combinations
  - Radial basis function (infinite dimensional space)

$$K(\mathbf{x}_i, \mathbf{x}_j; \sigma) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

- Text classification:

- Usually use linear or polynomial kernels

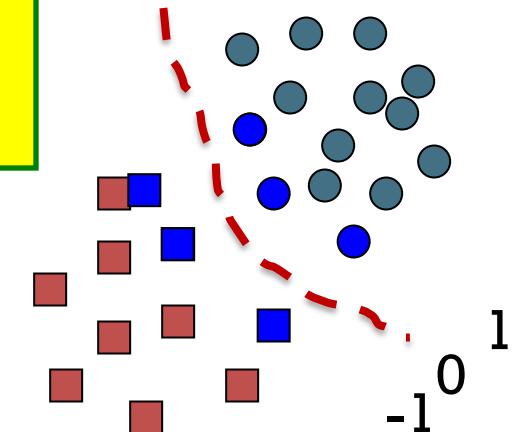
# Classification with SVMs **with Kernels**

- Given a new point  $\mathbf{x}$ , we can compute its score

i.e.,

$$\sum_{sv \in SV} \alpha_{sv} K(\mathbf{x}_{sv}, \mathbf{x}) + b$$

- Decide class based on whether  $<$  or  $> 0$
- E.g., in scikit-learn



- linear:  $\langle x, x' \rangle$ .
- polynomial:  $(\gamma \langle x, x' \rangle + r)^d$ .  $d$  is specified by keyword `degree`,  $r$  by `coef0`.
- rbf:  $\exp(-\gamma \|x - x'\|^2)$ .  $\gamma$  is specified by keyword `gamma`, must be greater than 0.
- sigmoid ( $\tanh(\gamma \langle x, x' \rangle + r)$ ), where  $r$  is specified by `coef0`.

# Pros and Cons of the SVM Classifier

---

- Pros
  - High accuracy
  - Fast classification speed
  - Works with cases where #features > #samples
  - Can adapt to different objects (due to Kernel)
    - Any  $K(u, v)$  can be used if symmetric, continuous, and positive semi-definite.
    - Or explicitly engineer the feature space.
- Cons
  - Training does not scale well with number of training samples ( $O(d*n^2)$  or  $O(d*n^3)$ )
  - Hyper-parameters need tuning

# Resources for today's lecture

---

- Christopher J. C. Burges. 1998. A Tutorial on Support Vector Machines for Pattern Recognition
- S. T. Dumais. 1998. Using SVMs for text categorization, IEEE Intelligent Systems, 13(4)
- S. T. Dumais, J. Platt, D. Heckerman and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. *CIKM '98*, pp. 148-155.
- Yiming Yang, Xin Liu. 1999. A re-examination of text categorization methods. 22nd Annual International SIGIR
- Tong Zhang, Frank J. Oles. 2001. Text Categorization Based on Regularized Linear Classification Methods. *Information Retrieval* 4(1): 5-31
- Trevor Hastie, Robert Tibshirani and Jerome Friedman. *Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, New York.
- T. Joachims, *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.
- Fan Li, Yiming Yang. 2003. A Loss Function Analysis for Classification Methods in Text Categorization. *ICML* 2003: 472-479.
- Tie-Yan Liu, Yiming Yang, Hao Wan, et al. 2005. Support Vector Machines Classification with Very Large Scale Taxonomy, *SIGKDD Explorations*, 7(1): 36-43.
- ‘Classic’ Reuters-21578 data set: <http://www.daviddlewis.com/resources/testcollections/reuters21578/>

---

# COMP9318: Data Warehousing and Data Mining

— L8: Clustering —

---

- What is Cluster Analysis?

# What is Cluster Analysis?

---

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Cluster analysis
  - Grouping a set of data objects into clusters
- Clustering *belongs to unsupervised classification*: no predefined classes
- Typical applications
  - As a **stand-alone tool** to get insight into data distribution
  - As a **preprocessing step** for other algorithms

# General Applications of Clustering

---

- Pattern Recognition
- Spatial Data Analysis
  - create thematic maps in GIS by clustering feature spaces
  - detect spatial clusters and explain them in spatial data mining
- Image Processing
- Economic Science (especially market research)
- WWW
  - Document classification
  - Cluster Weblog data to discover groups of similar access patterns

# Examples of Clustering Applications

---

- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Land use: Identification of areas of similar land use in an earth observation database
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults

# What Is Good Clustering?

---

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.

# Requirements of Clustering in Data Mining

---

- Scalability
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- High dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

# Chapter 8. Cluster Analysis

---

- Preliminaries

# Typical Inputs

Key component for clustering:  
the dissimilarity/similarity  
metric:  $d(i, j)$

- Data matrix
  - N objects, each represented by a m-dimensional feature vector

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1m} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{if} & \dots & x_{im} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nf} & \dots & x_{nm} \end{bmatrix}$$

$n \times m$

- Dissimilarity matrix
  - A square matrix giving distances between all pairs of objects.
  - If similarity functions are used → similarity matrix

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

$n \times n$

# Comments

---

- The definitions of distance functions are usually very different for interval-scaled, boolean, categorical, ordinal and ratio variables.
- Weights should be associated with different variables based on applications and data semantics, or appropriate preprocessing is needed.
- There is a separate “quality” function that measures the “goodness” of a cluster.
- It is hard to define “similar enough” or “good enough”
  - the answer is typically highly subjective.

# Type of data in clustering analysis

---

- Interval-scaled variables:
- Binary variables:
- Nominal, ordinal, and ratio variables:
- Variables of mixed types:

# Interval-valued variables

---

- Standardize data
  - Calculate the *mean absolute deviation*:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|)$$

where

$$m_f = \frac{1}{n}(x_{1f} + x_{2f} + \dots + x_{nf})$$

- Calculate the standardized measurement (*z-score*)

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

- Using mean absolute deviation is more **robust** than using standard deviation

# Similarity and Dissimilarity Between Objects

---

- Distances are normally used to measure the similarity or dissimilarity between two data objects
- A popular choice is the *Minkowski distance, or the  $L_p$  norm of difference vector*

$$d(i, j) = \|\mathbf{x}_i - \mathbf{x}_j\|_p, \quad \text{where } \|\mathbf{z}\|_p = \left( \sum_{i=1}^m |z_i|^p \right)^{1/p}$$

- Special cases:
  - if  $p = 1$ ,  $d$  is the **Manhattan distance**
  - if  $p = 2$ ,  $d$  is the **Euclidean distance**
  - if  $p = \infty$ ,  $\|\mathbf{x}_i - \mathbf{x}_j\|_\infty = \max_{k=1}^m |\mathbf{x}_{ik} - \mathbf{x}_{jk}|$

# Similarity and Dissimilarity Between Objects (Cont.)

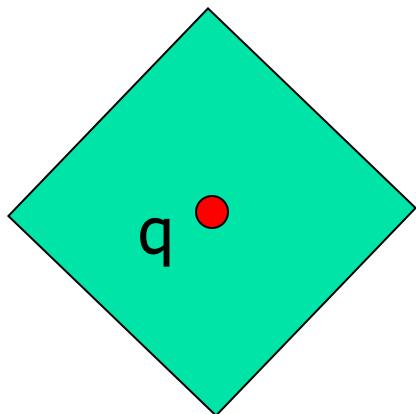
- Other similarity/distance functions:
  - Mahalanobis distance
  - Jaccard, Dice, cosine similarity, Pearson correlation coefficient
- Metric distance
  - Properties
    - $d(i,j) \geq 0$
    - $d(i,i) = 0$
    - $d(i,j) = d(j,i)$
    - $d(i,j) \leq d(i,k) + d(k,j)$

*common to all distance functions*

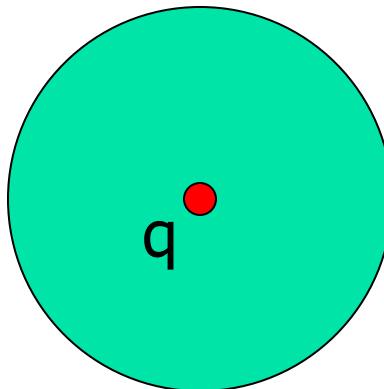
positiveness  
symmetry  
reflexivity

triangular inequality

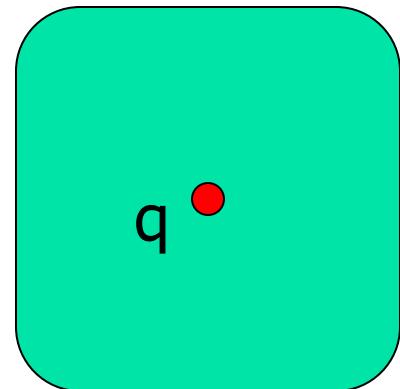
# Areas within a unit distance from q under different $L_p$ distances



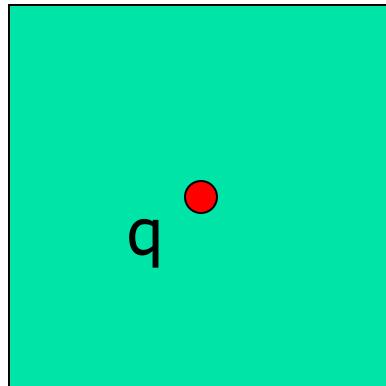
$L_1$



$L_2$



$L_8$



$L_\infty$

# Binary Variables

Obj	Vector Representation
i	[0, 1, 0, 1, 0, 0, 1, 0]
j	[0, 0, 0, 0, 1, 0, 1, 1]

- A contingency table for binary data

		Object <i>j</i>		<i>sum</i>
		1	0	
Object <i>i</i>	1	<i>a</i>	<i>b</i>	<i>a+b</i>
	0	<i>c</i>	<i>d</i>	<i>c+d</i>
<i>sum</i>		<i>a+c</i>	<i>b+d</i>	<i>p</i>

- Simple matching coefficient (invariant, if the binary variable is *symmetric*):  $d(i, j) = \frac{b+c}{a+b+c+d}$
- Jaccard coefficient (noninvariant if the binary variable is *asymmetric*):  $d(i, j) = \frac{b+c}{a+b+c}$

# Dissimilarity between Binary Variables

$$d(i, j) = \frac{b+c}{a+b+c}$$

## Example

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

- gender is a symmetric attribute
- the remaining attributes are asymmetric binary
- let the values Y and P be set to 1, and the value N be set to 0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

# Nominal Variables

---

- A generalization of the binary variable in that it can take more than 2 states, e.g., red, yellow, blue, green
- Method 1: Simple matching
  - $m$ : # of matches,  $p$ : total # of variables

$$d(i, j) = \frac{p - m}{p}$$

- Method 2: One-hot encoding
  - creating a new binary variable for each of the  $M$  nominal states

# Ordinal Variables

---

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
  - replace  $x_{if}$  by their rank  $r_{if} \in \{1, \dots, M_f\}$
  - map the range of each variable onto  $[0, 1]$  by replacing  $i$ -th object in the  $f$ -th variable by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

- compute the dissimilarity using methods for interval-scaled variables

# Ratio-Scaled Variables

---

- Ratio-scaled variable: a positive measurement on a nonlinear scale, approximately at exponential scale, such as  $Ae^{Bt}$  or  $Ae^{-Bt}$
- Methods:
  - treat them like interval-scaled variables—*not a good choice!* (why?—the scale can be distorted)
  - apply logarithmic transformation
$$y_{if} = \log(x_{if})$$
  - treat them as continuous ordinal data treat their rank as interval-scaled

---

- A Categorization of Major Clustering Methods

# Major Clustering Approaches

---

- Partitioning algorithms: Construct various partitions and then evaluate them by some criterion
- Hierarchy algorithms: Create a hierarchical decomposition of the set of data (or objects) using some criterion
- Graph-based algorithms: Spectral clustering
- Density-based: based on connectivity and density functions
- Grid-based: based on a multiple-level granularity structure
- Model-based: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other

---

- Partitioning Methods

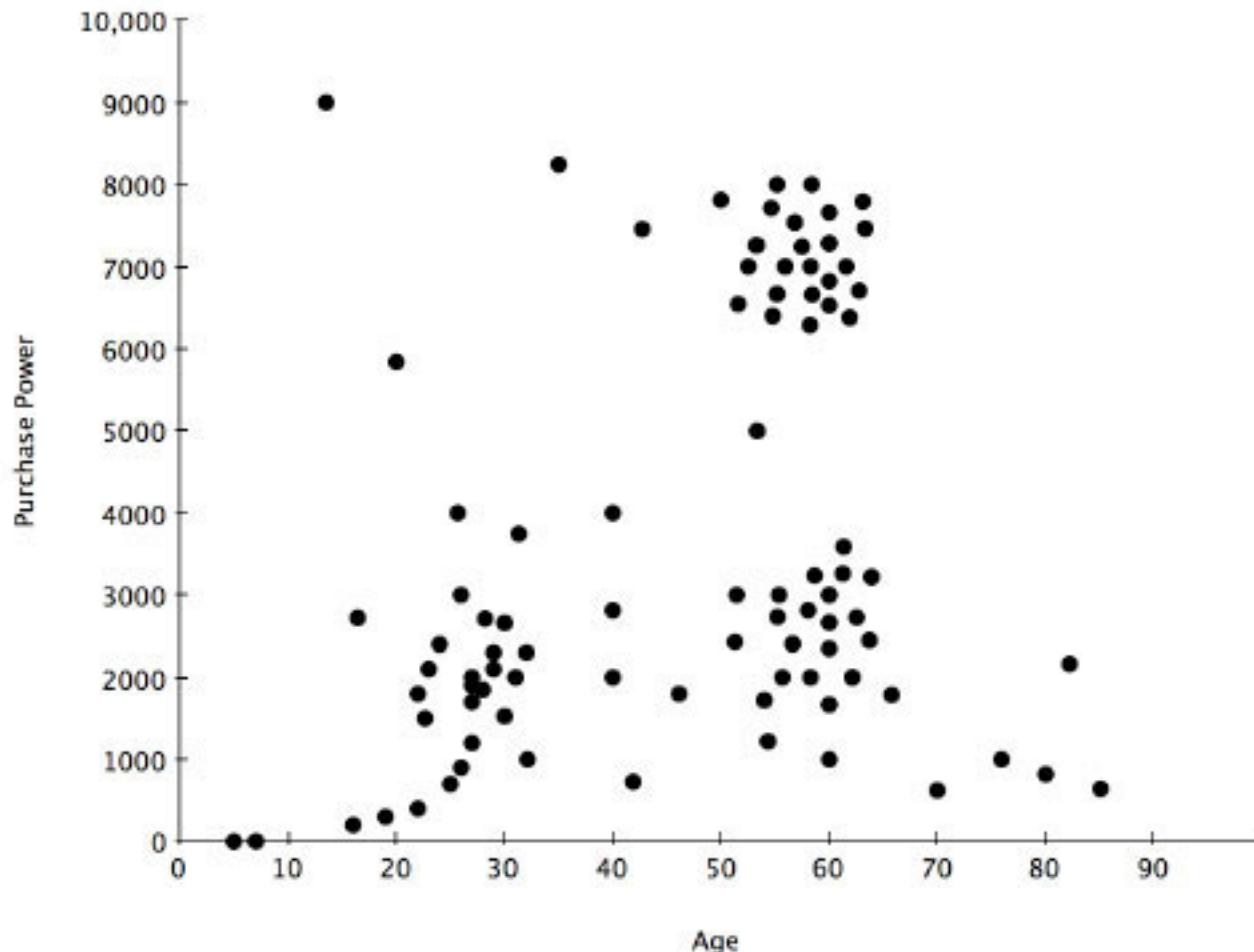
# Partitioning Algorithms: Problem Definition

---

- Partitioning method: Construct a “**good**” partition of a database of  $n$  objects into a set of  $k$  clusters
  - Input: a  $n \times m$  data matrix
- How to measure the “goodness” of a given partitioning scheme?
  - Cost of a cluster,  $\text{cost}(C_i) = \sum_{x_j \in C_i} \|x_j - \text{center}(C_i)\|_2^2$ 
    - Note:  $L_2$  distance used
    - Analogy with binning?
    - How to choose the center of a cluster?
      - Centroid (i.e., Avg) of  $x_j \rightarrow$  Minimizes  $\text{cost}(C_i)$
  - Cost of  $k$  clusters: sum of  $\text{cost}(C_i)$

# Example (2D)

---



# Partitioning Algorithms: Basic Concept

---

- It's an optimization problem!
  - Global optimal:
    - NP-hard (for a wide range of cost functions)
    - Requires exhaustively enumerate all  $\left\{ \begin{smallmatrix} n \\ k \end{smallmatrix} \right\} = \Theta\left(\frac{k^n}{k!}\right)$  partitions
      - Stirling numbers of the second kind
  - Heuristic methods:
    - k-means: an instance of the EM (expectation-maximization) algorithm
    - Many variants

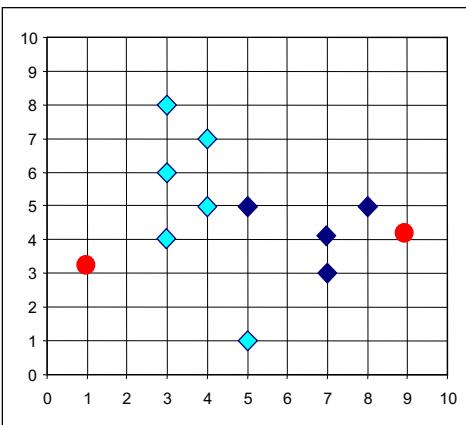
# The *K-Means* Clustering Method

---

- Lloyds Algorithm:
  1. Initialize  $k$  centers randomly
  2. While stopping condition is not met
    - i. Assign each object to the cluster with the nearest center
    - ii. Compute the new center for each cluster.
- Stopping condition =?
- What are the final clusters?

# The *K*-Means Clustering Method

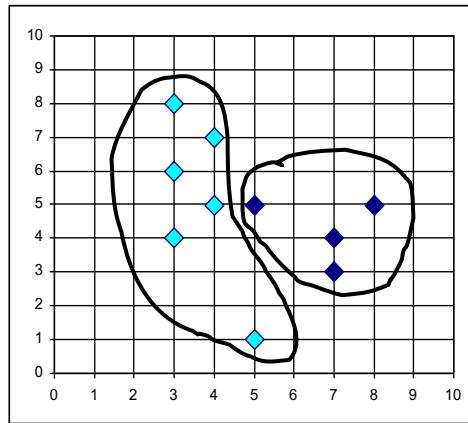
## ■ Example



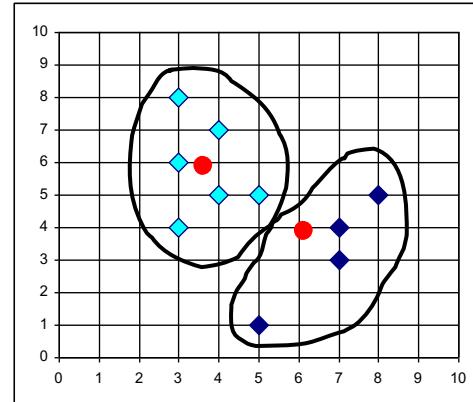
K=2

Arbitrarily choose K object as initial cluster center

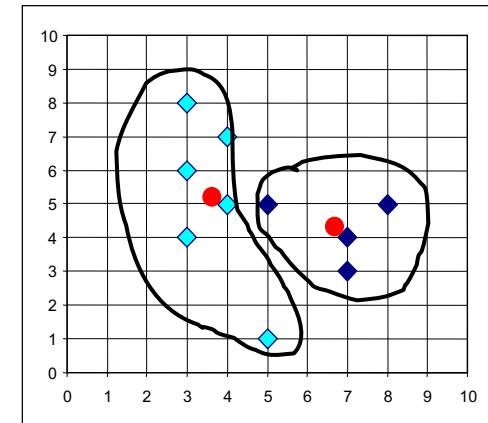
Assign each objects to most similar center



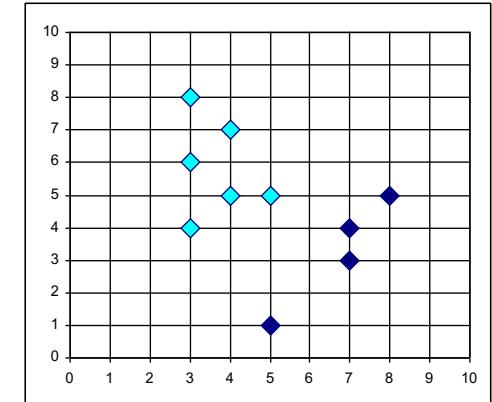
Update the cluster means



Update the cluster means



reassign



reassign

# Comments on the *K-Means* Method

---

- Strength: *Relatively efficient*:  $\mathcal{O}(tkn)$ , where  $n$  is # objects,  $k$  is # clusters, and  $t$  is # iterations. Normally,  $k, t \ll n$ .
  - Comparing: PAM:  $O(k(n-k)2)$ , CLARA:  $O(ks2 + k(n-k))$
- Comment:
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
  - No guarantee on the quality. Use k-means++.
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - Need to specify  $k$ , the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

# Variations of the *K-Means* Method

---

- A few variants of the *k-means* which differ in
  - Selection of the initial  $k$  means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes* (Huang'98)
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method

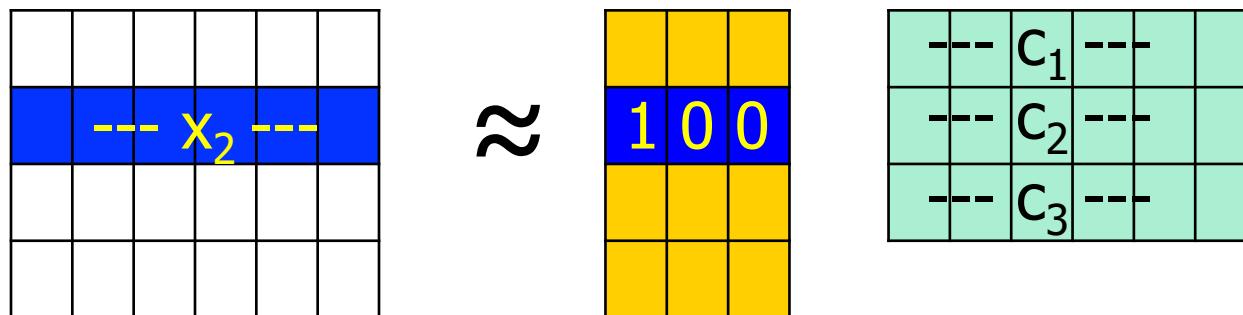
# **k-Means++** [Arthur and Vassilvitskii, SODA 2007]

---

- A simple initialization routine that guarantees to find a solution that is  $O(\log k)$  competitive to the optimal  $k$ -means solution.
- Algorithm:
  1. Find first center uniformly at random
  2. For each data point  $x$ , compute  $D(x)$  as the distance to its nearest center
  3. Randomly sample one point as the new center, with probabilities proportional to  $D^2(x)$
  4. Goto 2 if less than  $k$  centers
  5. Run the normal  $k$ -means with the  $k$  centers

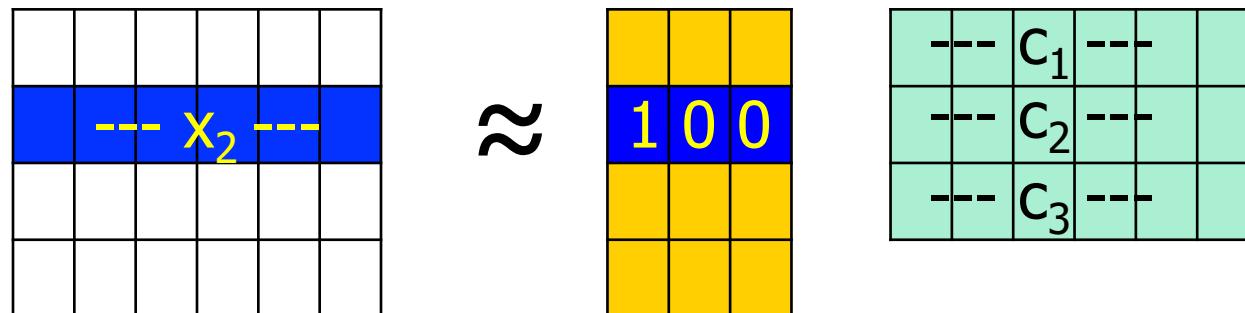
# k-means: Special Matrix Factorization

- $X^{n \times d} \approx U^{n \times k} V^{k \times d}$
- **Loss function:**  $\| X - UV \|_F^2$ 
  - Squared Frobenius norm
- **Constraints:**
  - Rows of  $U$  must be a one-hot encoding
- Alternative view
  - $X_{j,*} \approx U_{j,*} V \rightarrow X_{j,*}$  can be explained as a “special” linear combination of rows in  $V$



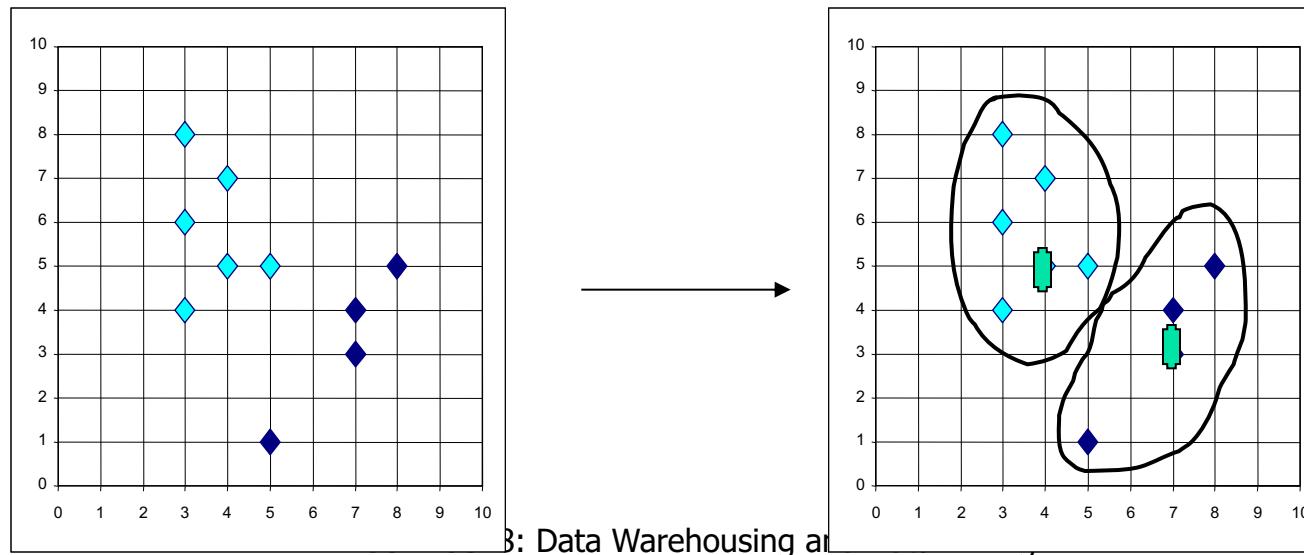
# Expectation Maximization Algorithm

- $X^{n \times d} \approx U^{n \times k} V^{k \times d}$
- **Loss function:**  $\| X - UV \|_F^2$
- Finding the best  $U$  and  $V$  simultaneously is hard, but
- Expectation step:
  - Given  $V$ , find the best  $U \rightarrow$  easy
- Maximization step:
  - Given  $U$ , find the best  $V \rightarrow$  easy
- Iterate until converging at a local minima.



# What is the problem of k-Means Method?

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data.
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster.



# K-medoids (PAM)

---

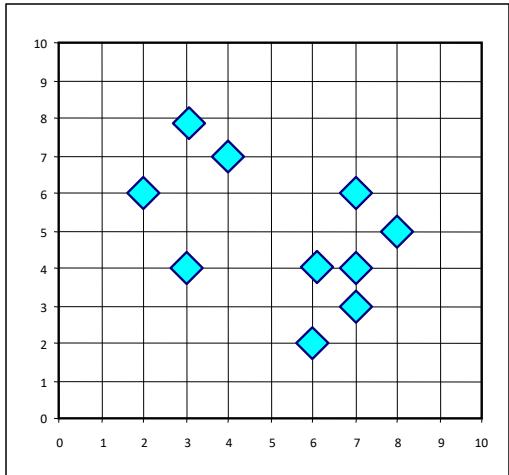
- *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by **one** of the objects in the cluster

# The *K-Medoids* Clustering Method

---

- Find *representative* objects, called medoids, in clusters
- *PAM* (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- *CLARA* (Kaufmann & Rousseeuw, 1990)
- *CLARANS* (Ng & Han, 1994): Randomized sampling
- Focusing + spatial data structure (Ester et al., 1995)

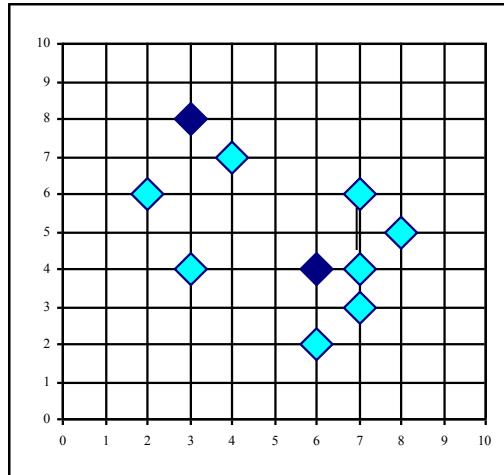
# Typical k-medoids algorithm (PAM)



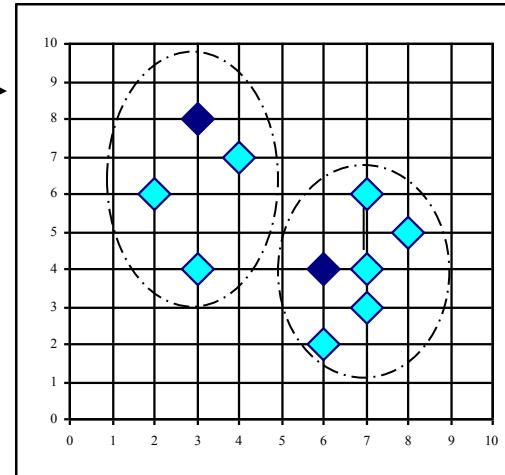
K=2

**Do loop**  
**Until no change**

Arbitrary choose k object as initial medoids

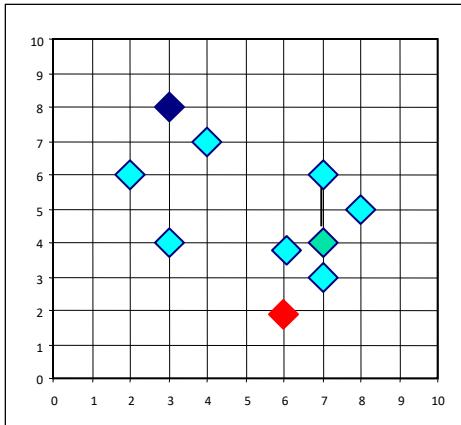


Assign each remaining object to nearest medoids

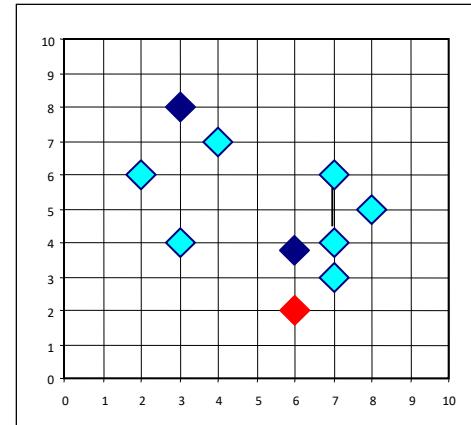


Total Cost = 20  
For each nonmedoid object,  $O_a$

Swapping  $O$  and  $O_a$   
If quality is improved.



Compute total cost of swapping



# PAM (Partitioning Around Medoids) (1987)

---

- PAM (Kaufman and Rousseeuw, 1987), built in Splus
- Use real object to represent the cluster
  - Select  $k$  representative objects arbitrarily
  - For each pair of non-selected object  $h$  and selected object  $i$ , calculate the total swapping cost  $TC_{ih}$
  - For each pair of  $i$  and  $h$ ,
    - If  $TC_{ih} < 0$ ,  $i$  is replaced by  $h$
    - Then assign each non-selected object to the most similar representative object
  - repeat steps 2-3 until there is no change

# What is the problem with PAM?

---

- PAM is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean
- PAM works efficiently for small data sets but does not **scale well** for large data sets.

- $O(k(n-k)^2)$  for each iteration

where n is # of data, k is # of clusters

→ Sampling based method,

CLARA(Clustering LARge Applications)

# Gaussian Mixture Model for Clustering

---

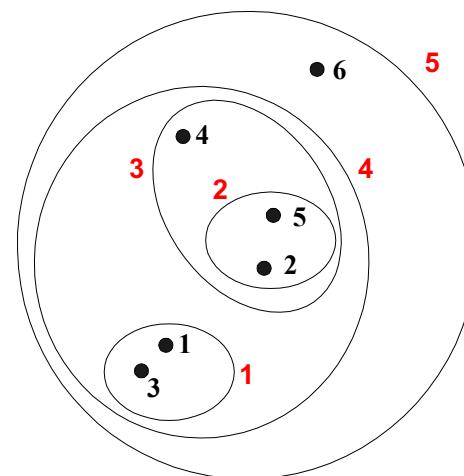
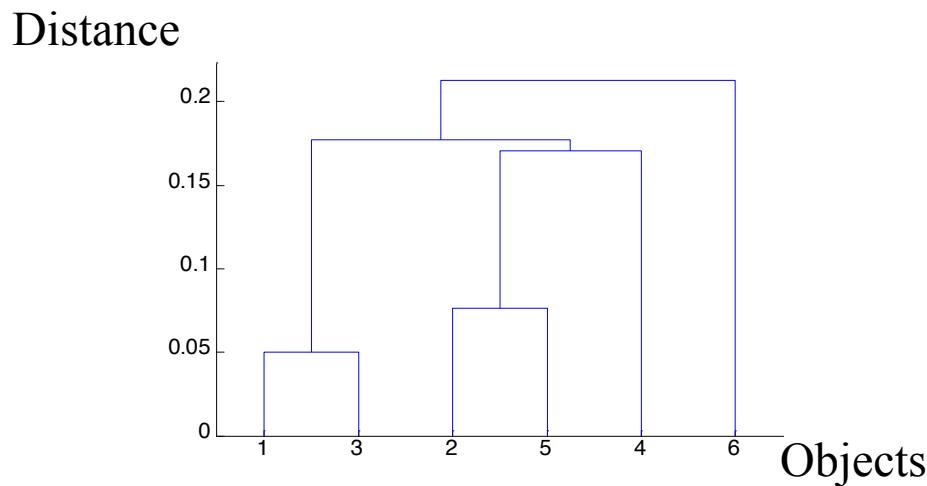
- $k$ -means can be deemed as a special case of the EM algorithm for GMM
- GMM
  - allows “soft” cluster assignment:
    - model  $\Pr(C \mid x)$
  - also a good example of
    - Generative model
    - Latent variable model
  - Use the Expectation-Maximization (EM) algorithm to obtain a local optimal solution

---

- Hierarchical Methods

# Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a **dendrogram**
  - A tree like diagram that records the sequences of merges or splits
  - A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.



# Strengths of Hierarchical Clustering

---

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, **phylogeny** reconstruction, ...)

# Hierarchical Clustering

---

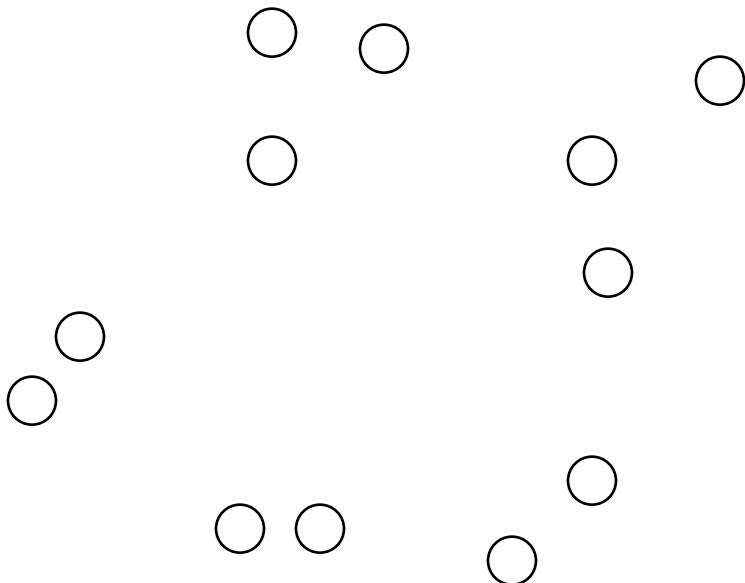
- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, **merge the closest pair of clusters** until only one cluster (or k clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

# Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix (i.e., matrix of pair-wise distances)
  2. Let each data point be a cluster
  3. **Repeat**
    4. Merge the two **closest** clusters
    5. **Update** the proximity matrix
    6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two **clusters** ← different from that of two **points**
  - Different approaches to defining the distance between clusters distinguish the different algorithms

# Starting Situation

- Start with clusters of individual points and a proximity matrix



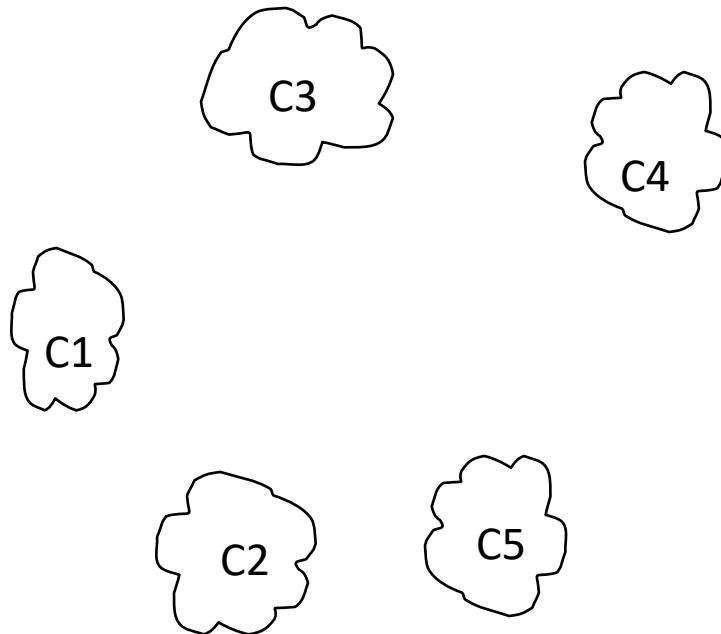
	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						

Proximity Matrix

p1    p2    p3    p4    ...    p9    p10    p11    p12

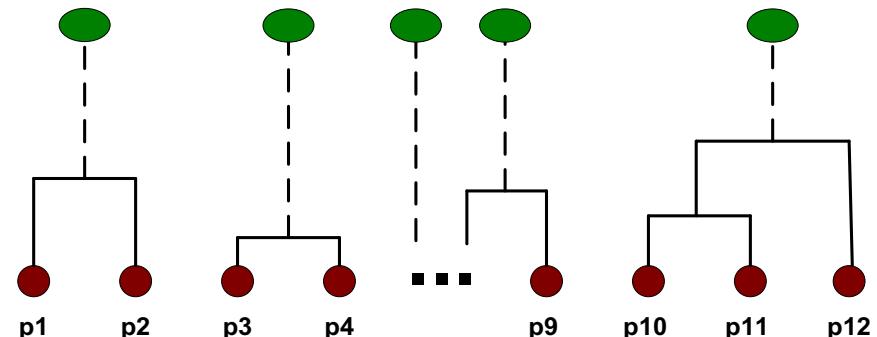
# Intermediate Situation

- After some merging steps, we have some clusters



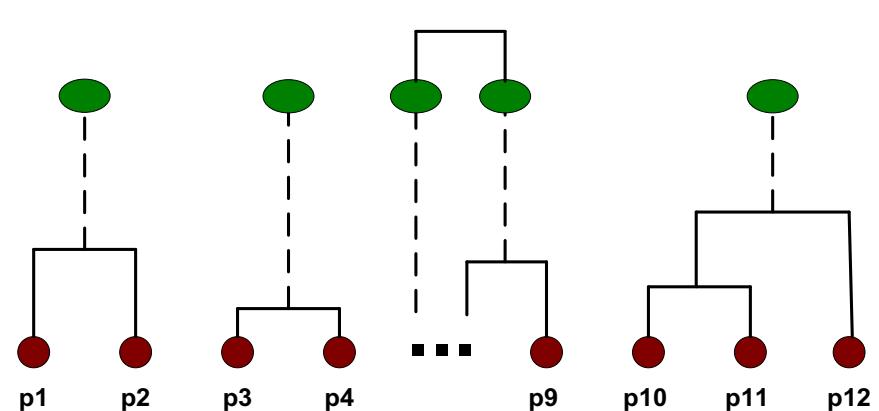
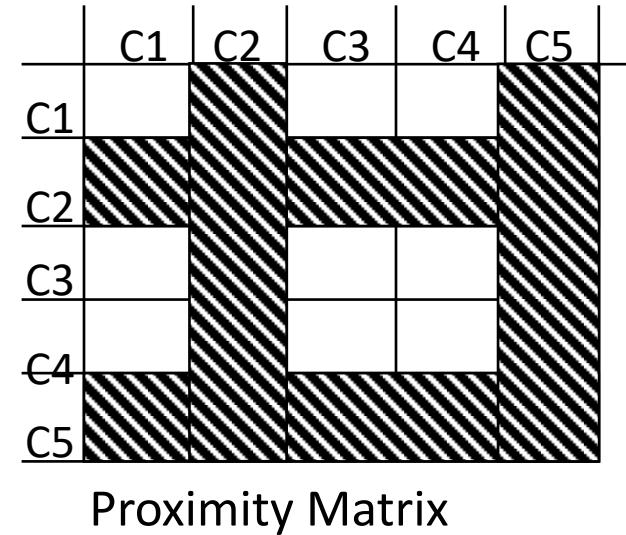
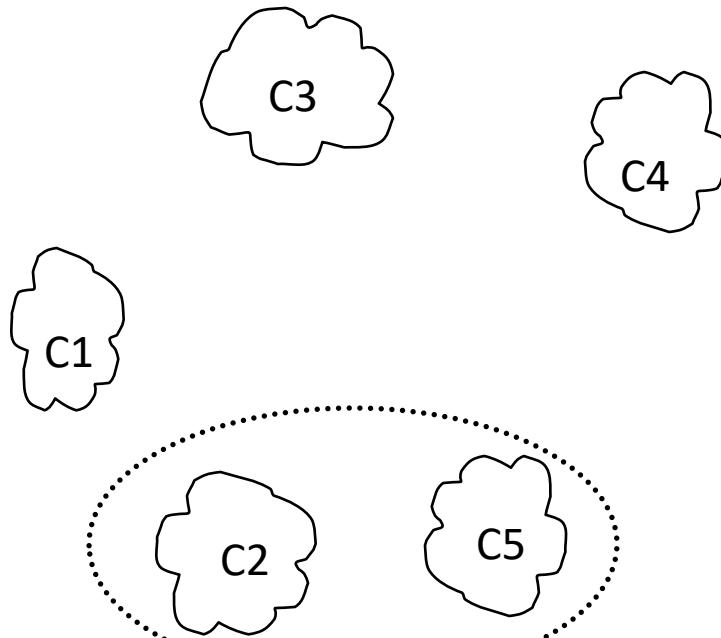
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



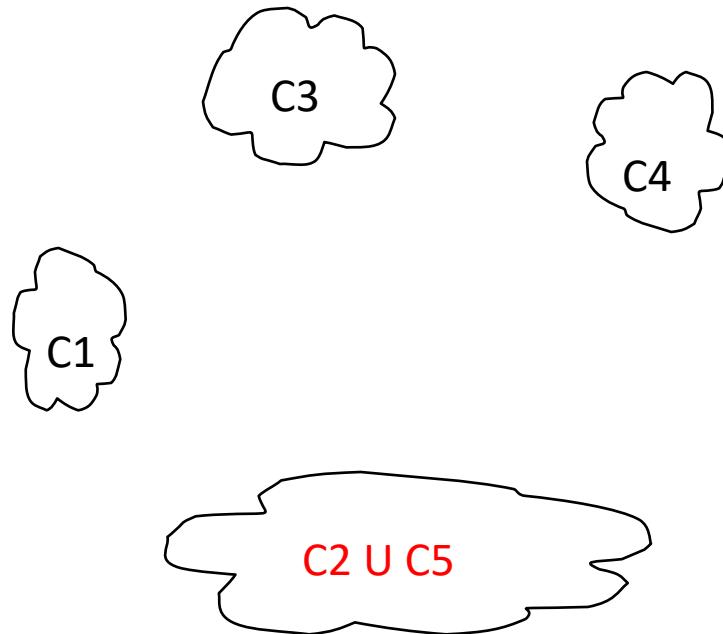
# Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



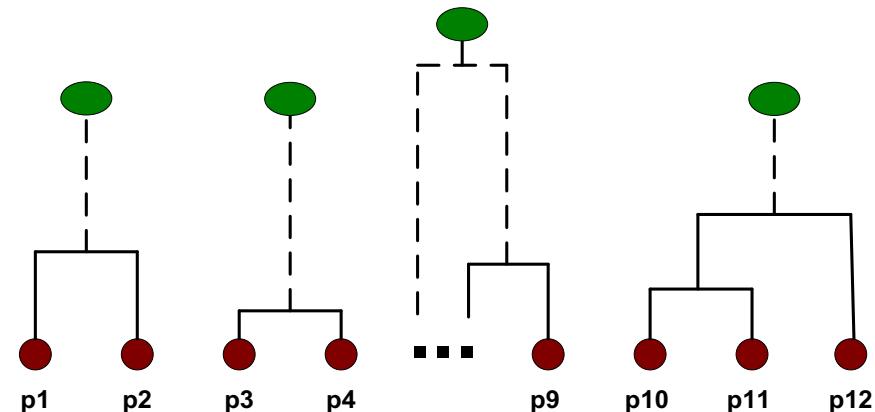
# After Merging

- The question is “How do we update the proximity matrix?”

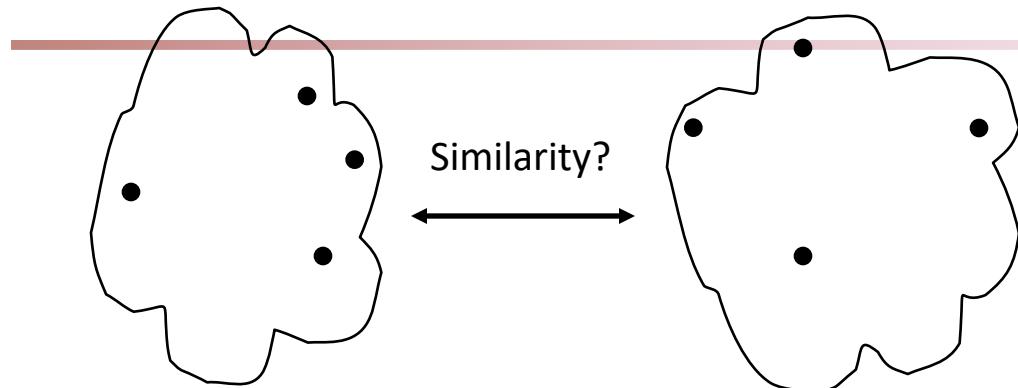


		C1	C5	C3	C4
C1	C1	?			
	C2 U C5	?	?	?	?
C3		?			
C4		?			

Proximity Matrix



# How to Define Inter-Cluster Distance

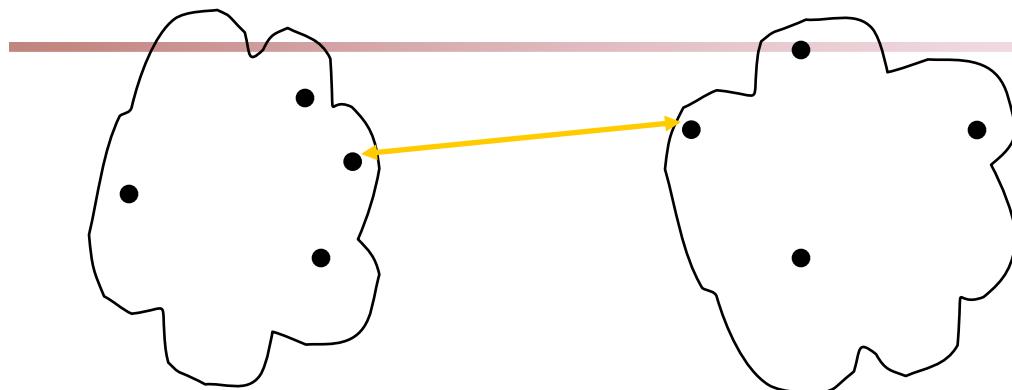


	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						

- MIN
- MAX
- Centroid-based
- **Group Average**
- Other methods driven by an objective function
  - Ward's Method uses squared error

Proximity Matrix

# How to Define Inter-Cluster Similarity

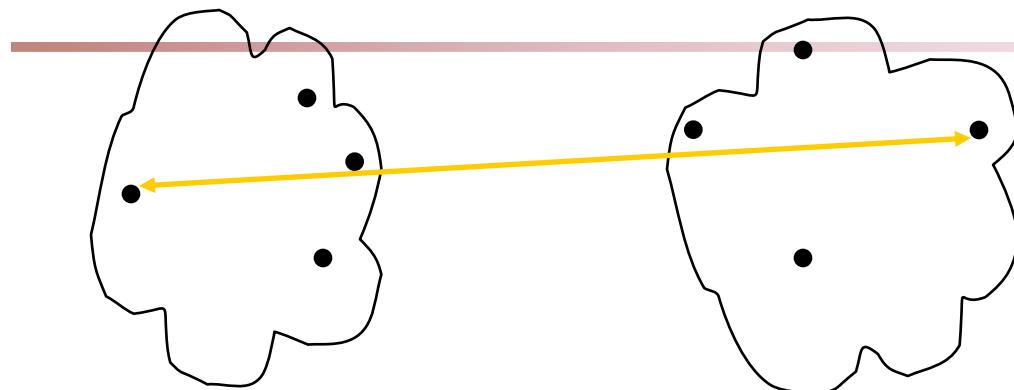


- MIN
- MAX
- Centroid-based
- Group Average
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

# How to Define Inter-Cluster Similarity

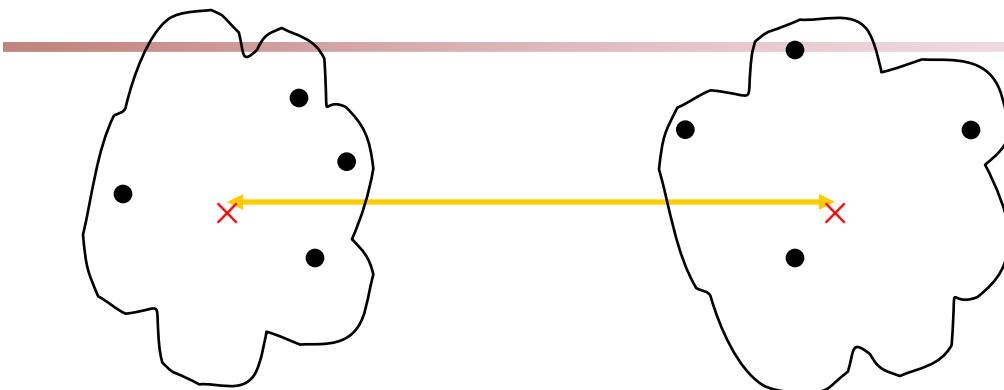


- MIN
- MAX
- Centroid-based
- Group Average
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

# How to Define Inter-Cluster Similarity

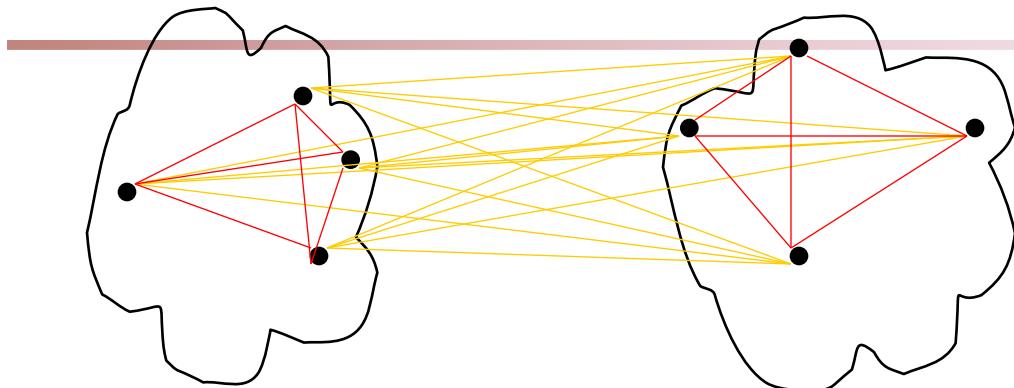


- MIN
- MAX
- **Centroid-based**
- Group Average
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						

Proximity Matrix

# How to Define Inter-Cluster Similarity



- MIN
- MAX
- Centroid-based
- **Group Average**
- Other methods driven by an objective function
  - Ward's Method uses squared error

Note: not simple avg distance between the clusters

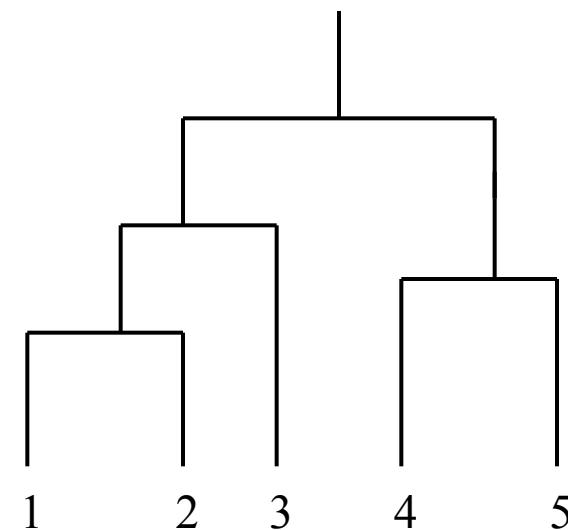
	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

# Cluster Similarity: MIN or Single Link/LINK

- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - i.e.,  $\text{sim}(C_i, C_j) = \min(\text{dissim}(p_x, p_y))$  //  $p_x \in C_i, p_y \in C_j$   
 $= \max(\text{sim}(p_x, p_y))$
  - Determined by **one** pair of points, i.e., by one link in the proximity graph.

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00



$$\text{sim}(C_i, C_j) = \max(\text{sim}(p_x, p_y))$$

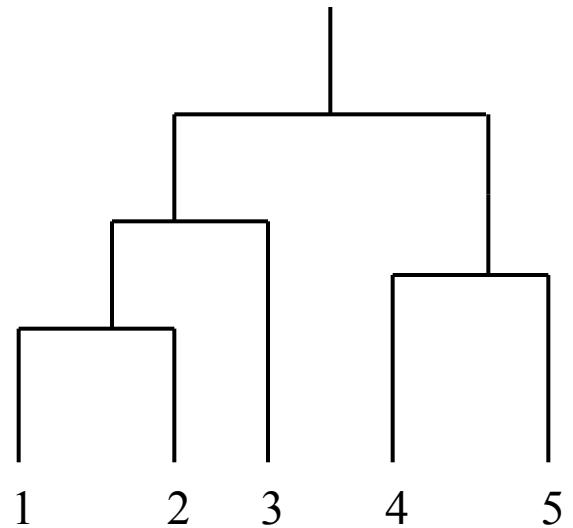
# Single-Link Example

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00

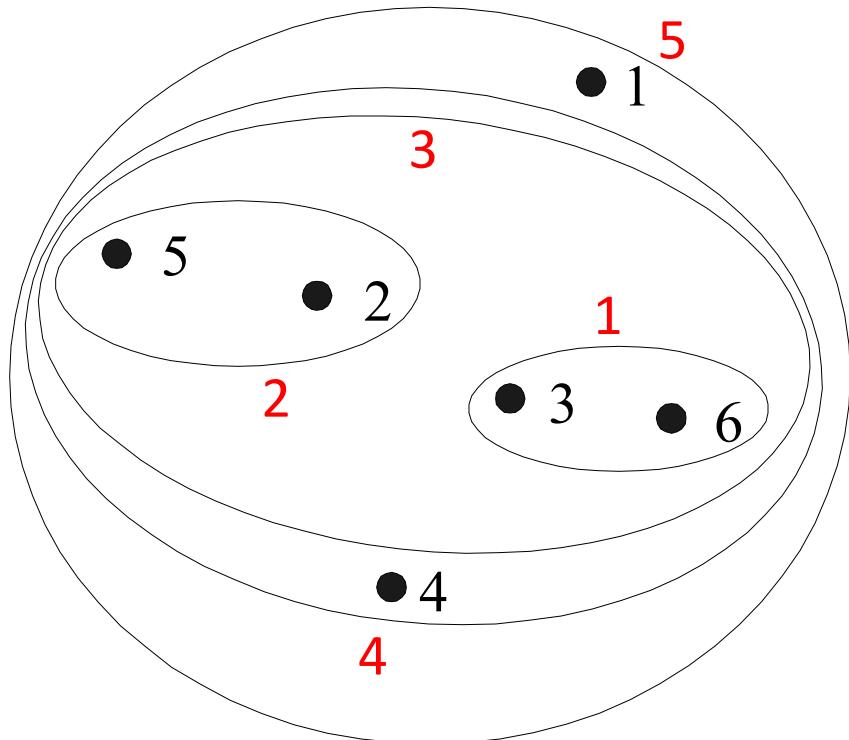
	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2		1.00	0.70	0.60	0.50
P3			1.00	0.40	0.30
P4				1.00	0.80
P5					1.00

	12	P3	P4	P5
12	1.00	0.70	0.65	0.50
P3		1.00	0.40	0.30
P4			1.00	0.80
P5				1.00

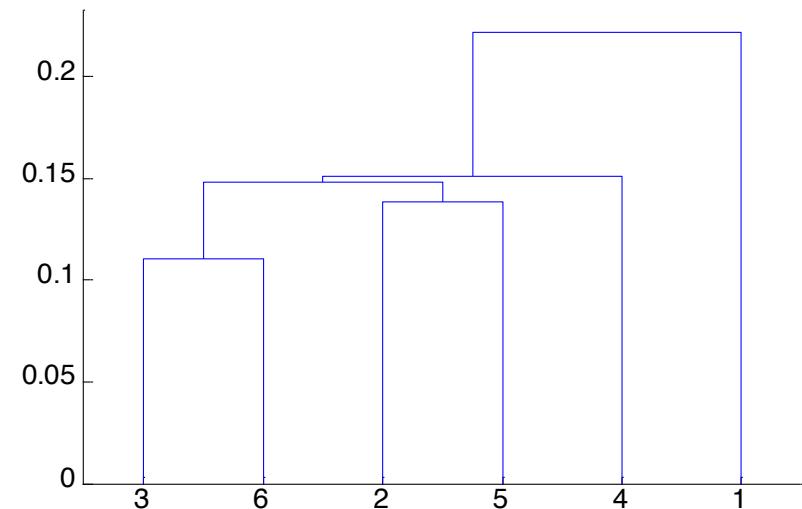
	12	P3	45
12	1.00	0.70	0.65
P3		1.00	0.40
45			1.00



# Hierarchical Clustering: MIN



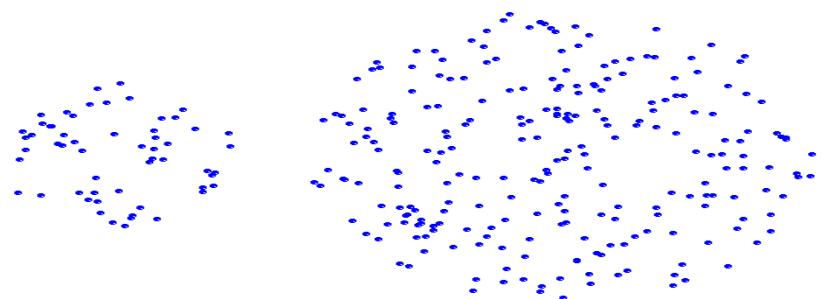
Nested Clusters



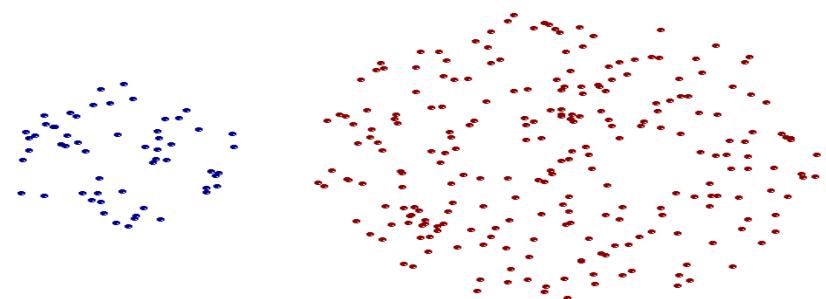
Dendrogram

# Strength of MIN

---



Original Points

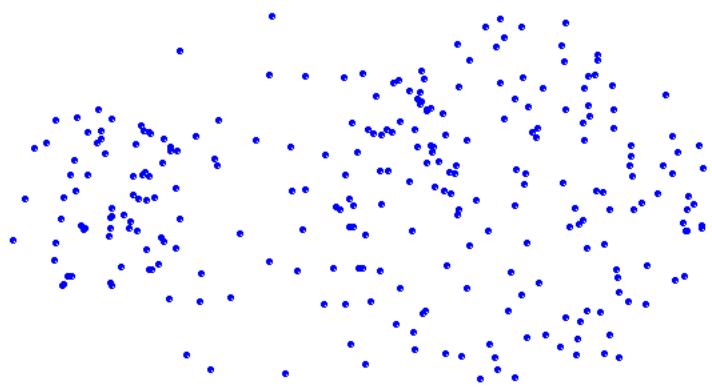


Two Clusters

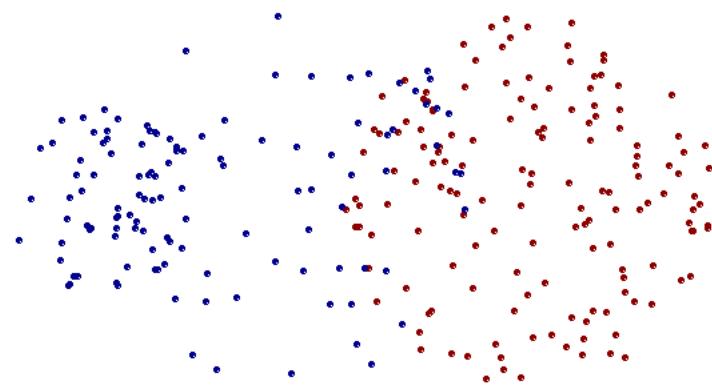
- Can handle non-elliptical shapes

# Limitations of MIN

---



Original Points



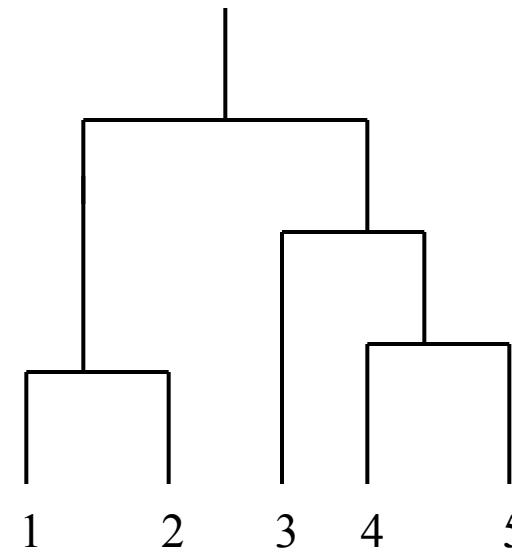
Two Clusters

- Sensitive to noise and outliers

# Cluster Similarity: MAX or Complete Link (CLINK)

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - i.e.,  $\text{sim}(C_i, C_j) = \max(\text{dissim}(p_x, p_y))$  //  $p_x \in C_i, p_y \in C_j$   
 $= \min(\text{sim}(p_x, p_y))$
  - Determined by **all** pairs of points in the two clusters

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00



$$\text{sim}(C_i, C_j) = \min(\text{sim}(p_x, p_y))$$

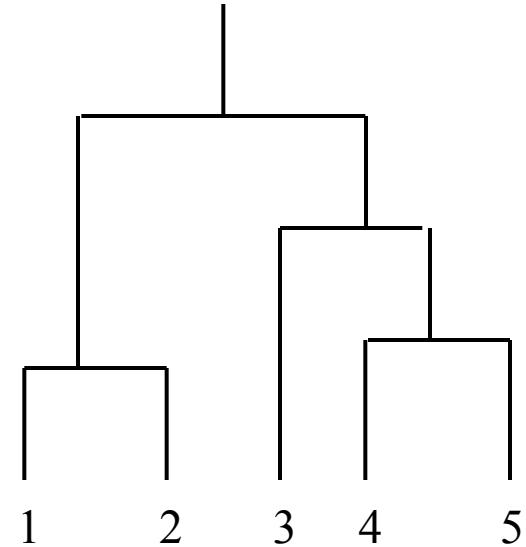
# Complete-Link Example

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00

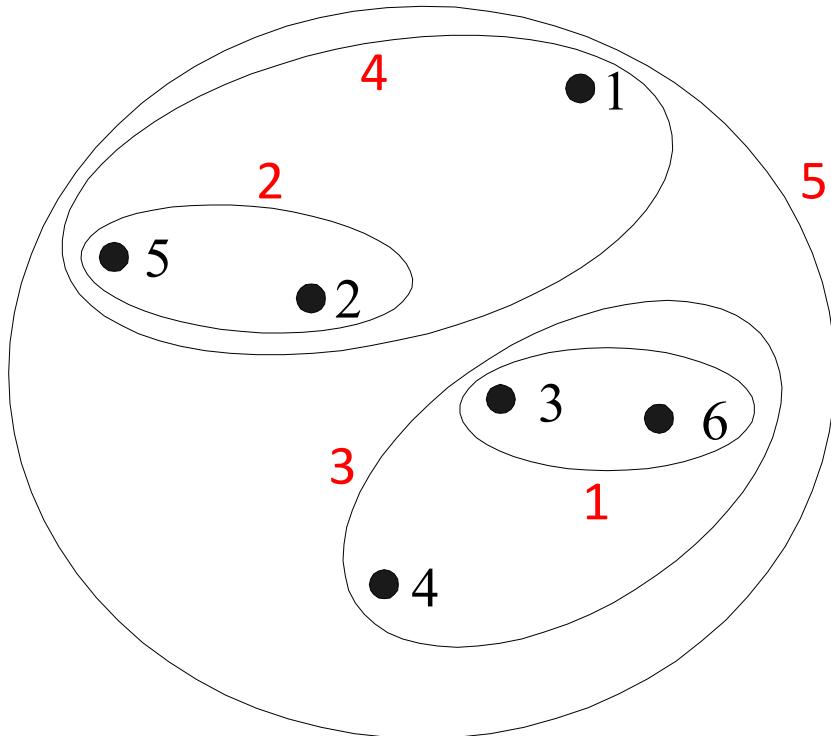
	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2		1.00	0.70	0.60	0.50
P3			1.00	0.40	0.30
P4				1.00	0.80
P5					1.00

	12	P3	P4	P5
12	1.00	0.10	0.60	0.20
P3		1.00	0.40	0.30
P4			1.00	0.80
P5				1.00

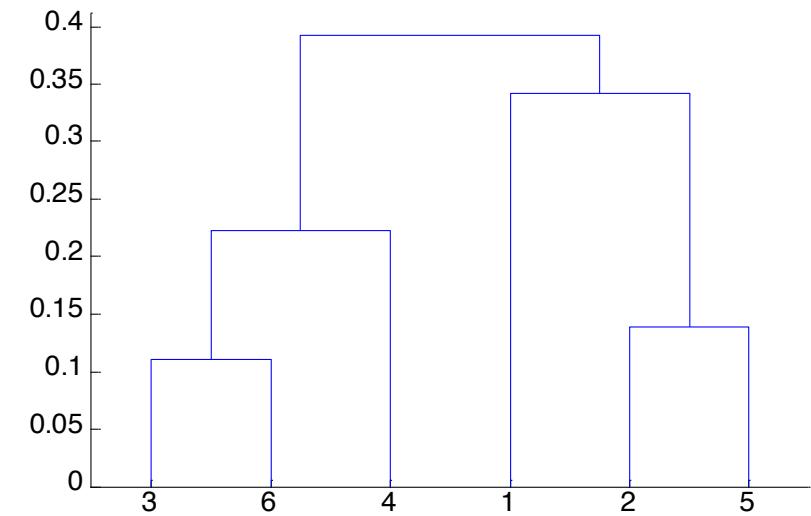
	12	P3	45
12	1.00	0.10	0.20
P3		1.00	0.30
45			1.00



# Hierarchical Clustering: MAX



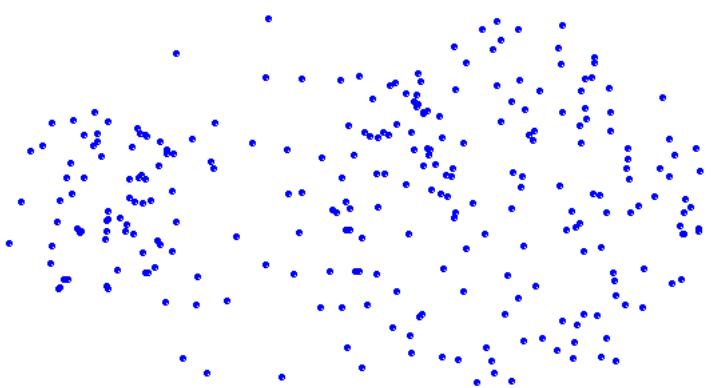
Nested Clusters



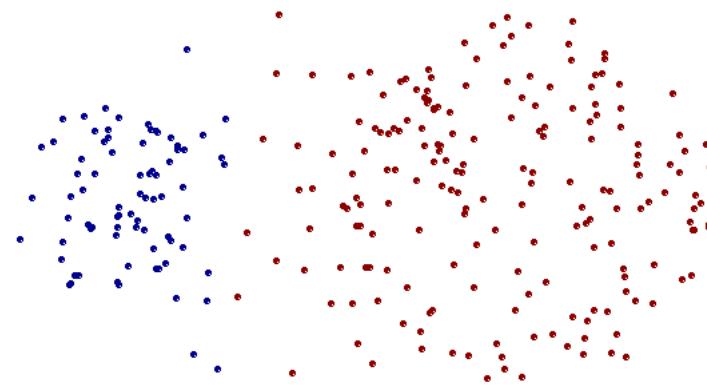
Dendrogram

# Strength of MAX

---



Original Points

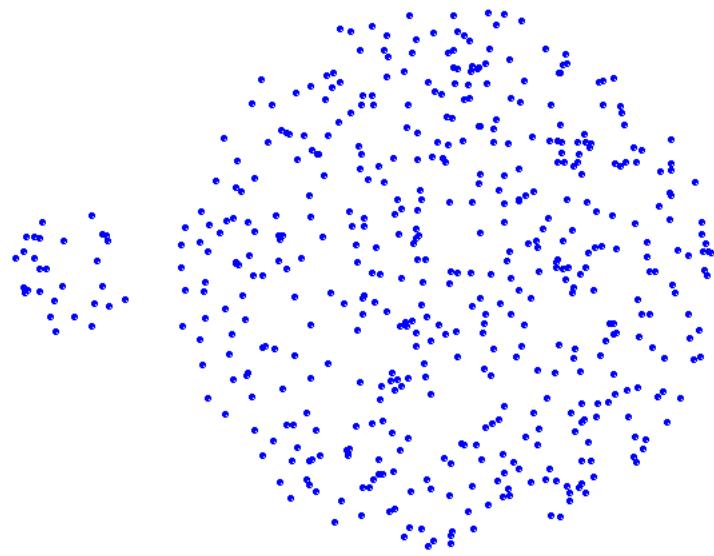


Two Clusters

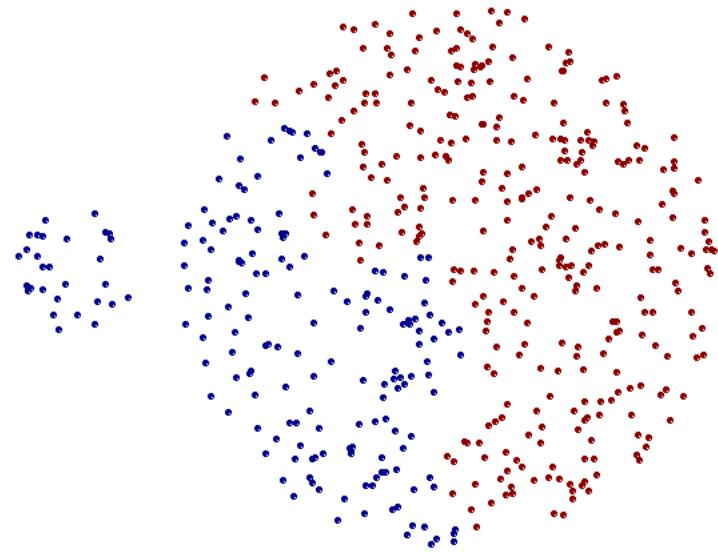
- Less susceptible to noise and outliers

# Limitations of MAX

---



Original Points



Two Clusters

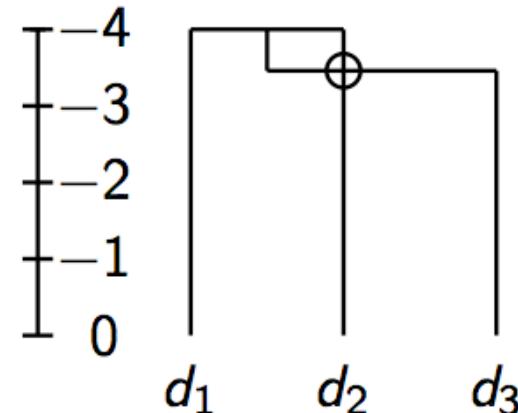
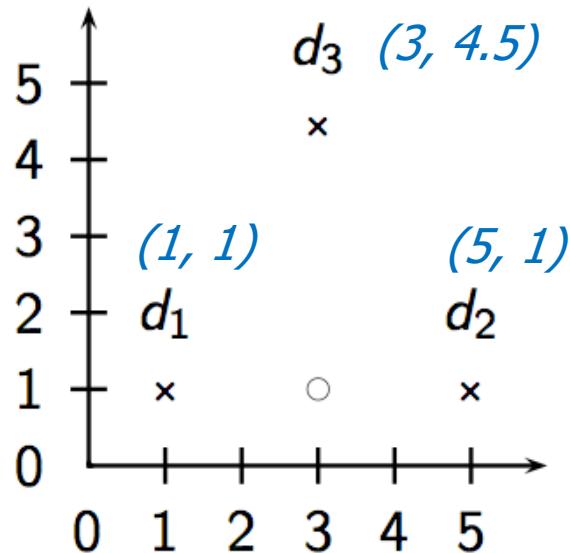
- Tends to break large clusters
- Biased towards globular clusters

# Cluster Similarity: Group Average

- GAAC (Group Average Agglomerative Clustering)
- Similarity of two clusters is the average of pair-wise similarity between points in the two clusters.

$$\text{similarity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i, p_j \in \text{Cluster}_i \cup \text{Cluster}_j \\ p_j \neq p_i}} \text{similarity}(p_i, p_j)}{(|\text{Cluster}_i| + |\text{Cluster}_j|) * (|\text{Cluster}_i| + |\text{Cluster}_j| - 1)}$$

- Why not using simple average distance? This method guarantees that no *inversions* can occur.



$$\text{sim}(C_i, C_j) = \text{avg}(\text{sim}(p_i, p_j))$$

# Group Average Example

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00

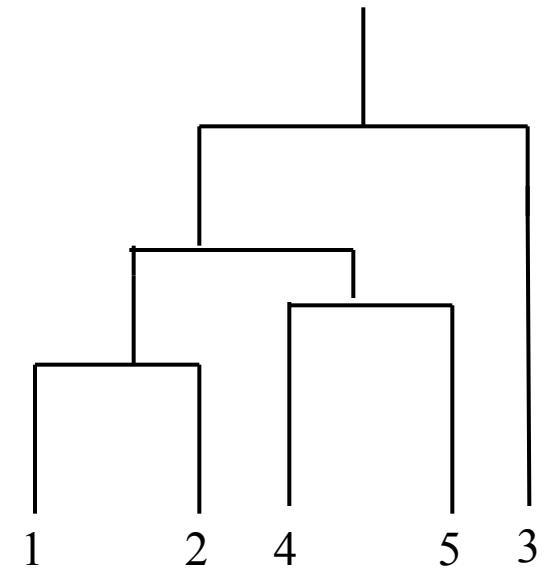
	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2		1.00	0.70	0.60	0.50
P3			1.00	0.40	0.30
P4				1.00	0.80
P5					1.00

	12	P3	P4	P5
12	1.00	0.567	0.717	0.533
P3		1.00	0.40	0.30
P4			1.00	0.80
P5				1.00

	12	P3	45
12	1.0	0.567	0.608
P3		1.00	0.5
45			1.00

$$\text{Sim}(12,3)=2*(0.1+0.7+0.9)/6 = 0.5666666$$

$$\text{Sim}(12,45)=2*(0.9+0.65+0.2+0.6+0.5+0.8)/12 = 0.608$$



# Hierarchical Clustering: Centroid-based and Group Average

---

- Compromise between Single and Complete Link
  
- Strengths
  - Less susceptible to noise and outliers
  
- Limitations
  - Biased towards globular clusters

# More on Hierarchical Clustering Methods

---

- Major weakness of agglomerative clustering methods
  - do not scale well: time complexity of at least  $O(n^2)$ , where  $n$  is the number of total objects
  - can never undo what was done previously
- Integration of hierarchical with distance-based clustering
  - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - CURE (1998): selects well-scattered points from the cluster and then shrinks them towards the center of the cluster by a specified fraction
  - CHAMELEON (1999): hierarchical clustering using dynamic modeling

# Spectral Clustering

---

- See additional slides.

# Spectral Clustering

Wei Wang @ CSE, UNSW

April 8, 2019

# Quadratic Form

- Let  $\mathbf{A}$  be some  $n \times n$  matrix.
- What is  $\mathbf{Ax}$ ? What's the **type** of the output? What may  $\mathbf{x}$  represent?
  - Some numeric assignment to  $\{1, 2, \dots, n\}$  (i.e., think of  $x_i$  as  $x(i)$ ).
  - E.g., what if  $x_i \in \{0, 1\}$ ?  $x_i \in [0, 1]$ ?  $x_i \in \mathbb{R}$ ?
- What is  $\mathbf{x}^\top \mathbf{Ax}$ ? What's the **type** of the output? Why it is called a quadratic form?

- Let  $\mathbf{A}$  be some  $n \times n$  matrix.
- What is  $\mathbf{Ax}$ ? What's the **type** of the output? What may  $\mathbf{x}$  represent?
  - Some numeric assignment to  $\{1, 2, \dots, n\}$  (i.e., think of  $x_i$  as  $x(i)$ ).
  - E.g., what if  $x_i \in \{0, 1\}$ ?  $x_i \in [0, 1]$ ?  $x_i \in \mathbb{R}$ ?
- What is  $\mathbf{x}^\top \mathbf{Ax}$ ? What's the **type** of the output? Why it is called a quadratic form?
  - $\mathbf{x}^\top \mathbf{Ax} = \sum_{i,j} A_{ij} \cdot (x_i x_j)$

Exercise:

- Rewrite  $f_1(\mathbf{x}) = (3x_1 - 2x_2) + 4x_3^2$  into a quadratic form.
- Rewrite  $f_2(\mathbf{x}) = (3x_1 - 2)^2 + (x_2 + x_1)^2$  into a quadratic form.

# Unnormalized Graph Laplacian /1

- (See the example graph later) Let  $\mathbf{A}$  is the adjacency matrix of a “normal” (unweighted) *undirected* graph  $G$ .  $\mathbb{V}$  are the vertices of  $G$  and  $\mathbb{E}$  are the edges of  $G$ 
  - An edge between  $v_i$  and  $v_j$  is modelled as  $(i, j)$  and  $(j, i)$ , i.e.,  $A_{ij} = A_{ji} = 1$ .
  - $A_{ii} = \underline{\hspace{2cm}}?$
  - Write out  $A$  for the example graph.
  - How many edges are there in the example graph? 16

# Unnormalized Graph Laplacian /2

- What is  $\mathbf{x}^\top \mathbf{I}_n \mathbf{x}$ ?
- What is  $\mathbf{x}^\top \mathbf{D}\mathbf{x}$ , where  $\mathbf{D} = \text{Diag}(d_1, d_2, \dots, d_n)$  and  $d_i = \deg(v_i) = \sum_{(i,j) \in \mathbb{E}} w_{ij}$ ?
- What is  $\mathbf{x}^\top \mathbf{A}\mathbf{x}$ ?
- Now what about  $2(\mathbf{x}^\top \mathbf{D}\mathbf{x} - \mathbf{x}^\top \mathbf{A}\mathbf{x})$  ?

# Unnormalized Graph Laplacian /3

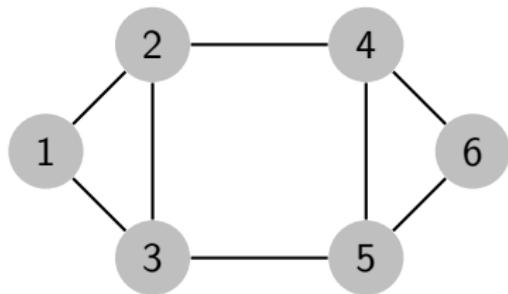
- $\mathbf{x}^\top \mathbf{I} \mathbf{x} = \sum_i x_i x_i$
- $\mathbf{x}^\top \mathbf{D} \mathbf{x} = \sum_i d_i \cdot x_i x_i = \sum_e x_i x_i$ .
- $\mathbf{x}^\top \mathbf{A} \mathbf{x} = \sum_{(i,j) \in \mathbb{E}} x_i x_j = \sum_e x_i x_j$
- $2(\mathbf{x}^\top \mathbf{D} \mathbf{x} - \mathbf{x}^\top \mathbf{A} \mathbf{x}) = \sum_e (2x_i^2 - 2x_i x_j) = \sum_e x_i^2 + \sum_e x_j^2 - \sum_e 2x_i x_j = \sum_e (x_i - x_j)^2$

## Example

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \frac{1}{2} \cdot \sum_{e_{ij} \in \mathbb{E}} (x_i - x_j)^2 \quad , \text{ where } \mathbf{L} = \mathbf{D} - \mathbf{A}.$$

- $\ell_2$  differences between assignments on the two ends of an edge, summed over all edges.

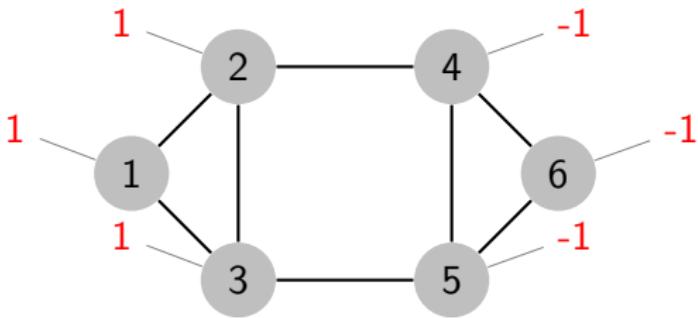
# Example



	$n_1$	$n_2$	$n_3$	$n_4$	$n_5$	$n_6$
$n_1$						
$n_2$						
$n_3$						
$n_4$						
$n_5$						
$n_6$						

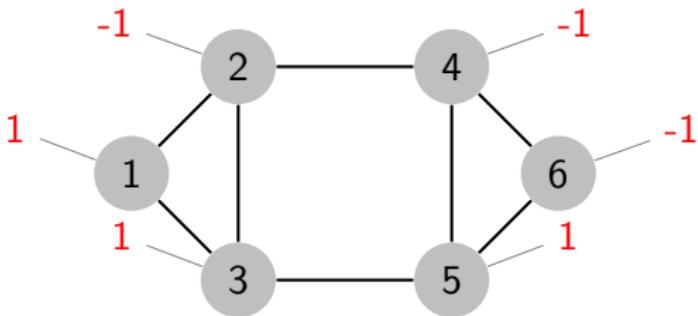
- $\mathbf{1}_n$  is the one vector.
- $\mathbf{L}\mathbf{1}_n = \quad \quad \quad (\text{NB: } \mathbf{L}^\top = \mathbf{L}) \quad \quad \Rightarrow \quad \lambda_1 = 0, v_1 = \mathbf{1}_n$
- $\mathbf{x}^\top \mathbf{L} \mathbf{x} =$

# Binary $\mathbf{x}$ induces a Clustering /1



- $\mathbf{x} =$
- $\mathbf{x}^\top \mathbf{L} \mathbf{x} =$

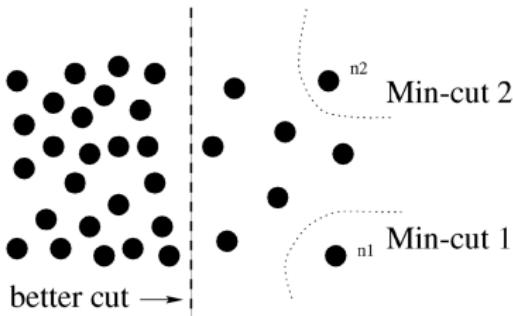
# Binary $\mathbf{x}$ induces a Clustering /2



- $\mathbf{x} =$
- $\mathbf{x}^\top \mathbf{L} \mathbf{x} =$
- $\mathbf{x}^\top \mathbf{x} =$

# Min Cut vs. Normalized Cut

- Min cuts are not *always* desirable.
  - Biased towards cutting small sets of isolated nodes.



- Cut:  $\text{cut}(A, B) = \sum_{v_i \in A, v_j \in B} w_{ij}.$
- Normalized cut:

$$\text{ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{vol}(A)} + \frac{\text{cut}(A, B)}{\text{vol}(B)},$$

where  $\text{vol}(A) = \sum_{v_i \in A} d_i = \sum_{v_i \in A, v_j \in \mathbb{V}} w_{i,j}.$

# Connection to $L$

$$ncut(A, B) = cut(A, B) \left( \frac{1}{vol(A)} + \frac{1}{vol(B)} \right)$$

- Let  $x_i = \frac{1}{vol(A)}$  if  $v_i \in A$ , and  $= \frac{-1}{vol(B)}$  otherwise.
- $\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_e w_{ij} (x_i - x_j)^2 = 0 + \sum_{v_i \in A, v_j \in B} \left( \frac{1}{vol(A)} + \frac{1}{vol(B)} \right)^2$
- $\mathbf{x}^\top \mathbf{D} \mathbf{x} = (\mathbf{x}^\top \mathbf{D}) \mathbf{x} = \sum_E d_i x_i^2 = \sum_{v_i \in A} \frac{d_i}{vol(A)^2} + \sum_{v_j \in B} \frac{d_j}{vol(B)^2} = \frac{1}{vol(A)} + \frac{1}{vol(B)}$

$$ncut(A, B) = \frac{\mathbf{x}^\top \mathbf{L} \mathbf{x}}{\mathbf{x}^\top \mathbf{D} \mathbf{x}}$$

$$\text{Minimize } ncut(A, B) = \frac{\mathbf{x}^\top \mathbf{Lx}}{\mathbf{x}^\top \mathbf{Dx}} \quad \text{Subject to} \quad x_i \in \left\{ \frac{1}{vol(A)}, \frac{-1}{vol(B)} \right\}$$

- NP-hard to optimize under the discrete constraint.
- Relaxation: grow the feasible region of  $\mathbf{x}$  and find the minimum value within the enlarged region.
  - allow  $\mathbf{x}$  to be a real vector?
  - Yes, but too large.
  - This gives the constraint:  $\mathbf{x}^\top \mathbf{D}\mathbf{1} = 0$  or equivalently  $\mathbf{x}^\top \mathbf{D} \perp \mathbf{1}$   
(You can verify this by plugging in any discrete vectors)
- Solution: the second smallest eigenvector of the generalized eigen value problem  $\mathbf{Lx} = \lambda \mathbf{Dx}$ .
- Normalized Laplacian:

$$\mathbf{L}' = \mathbf{D}^{-\frac{1}{2}} (\mathbf{D} - \mathbf{W}) \mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$$

- Algorithm  $SC\_recursive\_bin\_cut(data, k)$ 
  - Construct the weighted graph  $G$
  - Construct the **special** graph laplacian  $L$  for  $G$ .
  - Compute the smallest non-zero eigenvector for  $L$ . This is the new representation of vertices in a new **1-dimensional** space (i.e., **embedding**).
  - Cluster the vertices in the embedding space according to the objective function.
  - For each cluster, recursively call the algorithm if more clusters are needed.

- Algorithm  $SC\_k\_way\_cut(data, k)$ 
  - Construct the weighted graph  $G$
  - Construct the **special** graph laplacian  $L$  for  $G$ .
  - Compute the smallest  $t$  non-zero eigenvector for  $L$ . This is the new representation of vertices in a new  **$t$ -dimensional** space (i.e., **embedding**).
  - Cluster the vertices in the embedding space using another clustering algorithm (e.g.,  $k$ -means)

# Notes on the Algorithms

- How to construct the weighted graph if only  $n$  objects are given?
  - Be based on the similarity or distance among objects.
  - E.g.,  $w_{ij} = \exp\left(\frac{\|f(o_i) - f(o_j)\|}{2\sigma^2}\right)$  where  $f(o)$  is the feature vector of object  $o$ . One can also induce a sparse graph if one caps the raw weights by a threshold.
- Which Laplacian to use?
  - Unnormalized graph laplacian  $\mathbf{L} = \mathbf{D} - \mathbf{W}$ .
  - Normalized graph laplacian  $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$ .

# Comments on Spectral Clustering

- Pros:
  - Usually better quality than other methods.
  - Can be thought of (non-linear) dimensionality reduction or embedding.
  - Freedom to construct a (sparse)  $G$  to preserve local similarity/connectivity.
  - Only requires some similarity measure.
  - Could be more efficient than  $k$ -means for high-dimensional sparse vectors (esp. if  $k$ -means is not fully optimized for such case).
- Cons:
  - Still need to determine  $k$
  - Assumes clusters are of similar sizes.
  - Does not scale well with large datasets; but more scalable variants exist.
  - One of the relaxations of the original NP-hard problem – may not be the tightest relaxation.

---

# COMP9318: Data Warehousing and Data Mining

— L6: Association Rule Mining —

---

- Problem definition and preliminaries

# What Is Association Mining?

---

- Association rule mining:
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
  - **Frequent pattern**: pattern (set of items, sequence, etc.) that occurs frequently in a database [AIS93]
- Motivation: finding regularities in data
  - What products were often purchased together? — Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

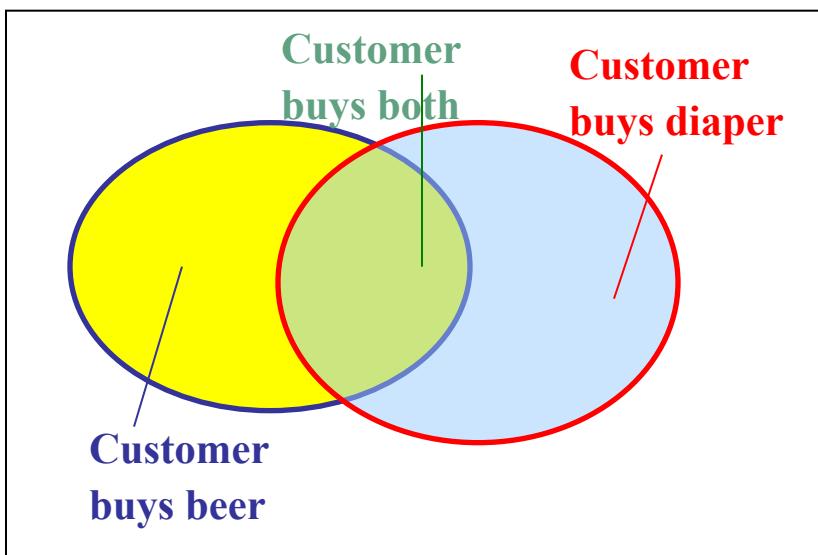
# Why Is Frequent Pattern or Association Mining an Essential Task in Data Mining?

---

- Foundation for many essential data mining tasks
  - Association, correlation, causality
  - Sequential patterns, temporal or cyclic association, partial periodicity, spatial and multimedia association
  - Associative classification, cluster analysis, iceberg cube, fascicles (semantic data compression)
- Broad applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis
  - **Web log** (click stream) **analysis**, DNA sequence analysis, etc. c.f., google's spelling suggestion

# Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	{ A, B, C }
20	{ A, C }
30	{ A, D }
40	{ B, E, F }



- Itemset  $X = \{x_1, \dots, x_k\}$ 
  - **Shorthand:**  $x_1 x_2 \dots x_k$
- Find all the rules  $X \rightarrow Y$  with min confidence and support
  - **support, s, probability** that a transaction contains  $X \cup Y$
  - **confidence, c, conditional probability** that a transaction having  $X$  also contains  $Y$ .

Let  $\text{min\_support} = 50\%$ ,

$\text{min\_conf} = 70\%:$

$$\text{sup}(AC) = 2$$

$$A \rightarrow C (50\%, 66.7\%)$$

$$C \rightarrow A (50\%, 100\%)$$

frequent itemset

association rule

# Mining Association Rules—an Example

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

Min. support 50%  
Min. confidence 50%

Frequent pattern	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

For rule  $A \rightarrow C$ :

$$\text{support} = \text{support}(\{A\} \cup \{C\}) = 50\%$$

$$\text{confidence} = \text{support}(\{A\} \cup \{C\}) / \text{support}(\{A\}) = 66.6\%$$

major computation challenge: calculate the support of itemsets

← The **frequent itemset mining** problem

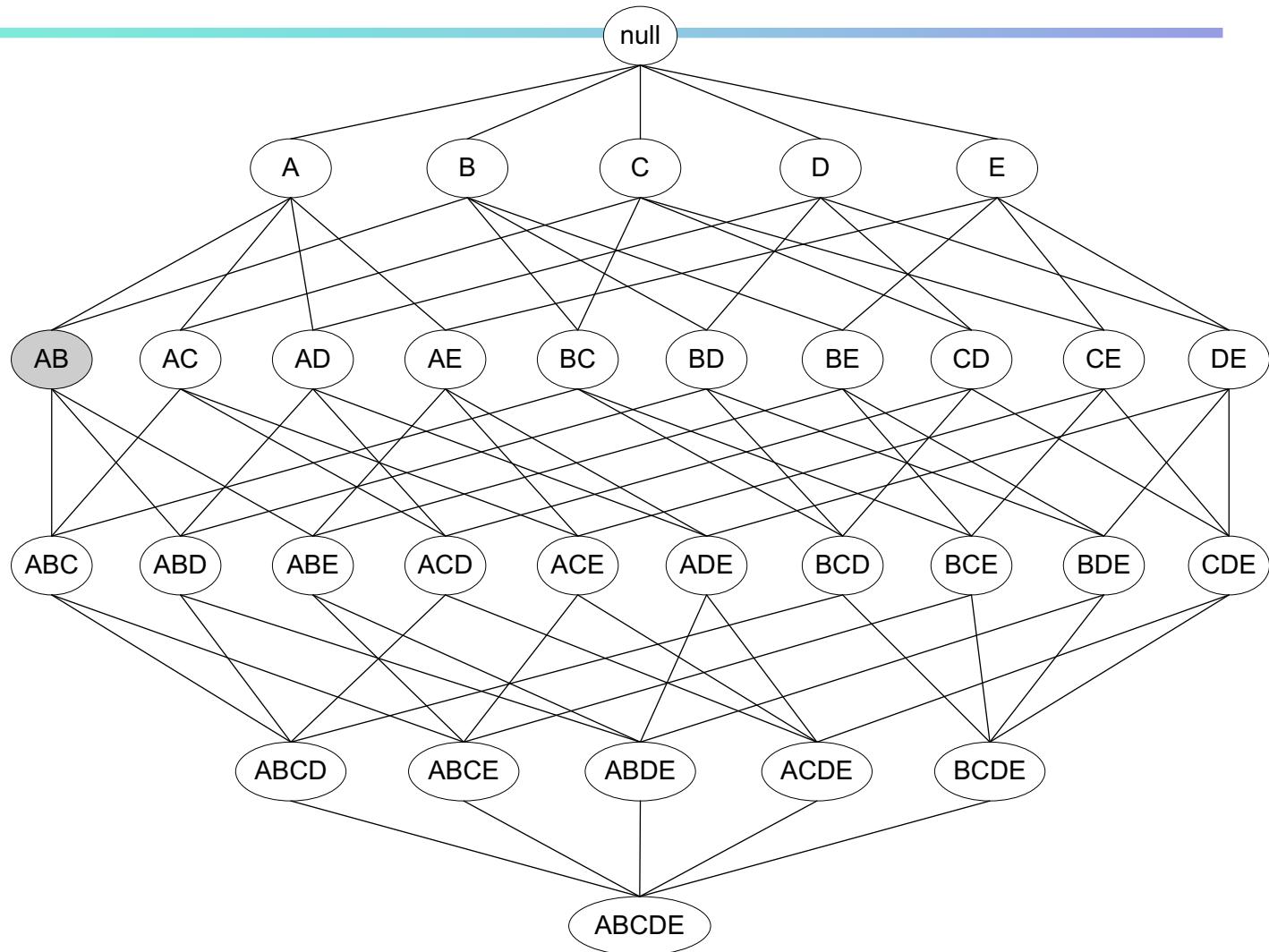
- 
- Algorithms for scalable mining of (single-dimensional Boolean) association rules in transactional databases

# Association Rule Mining Algorithms

Candidate Generation  
& Verification

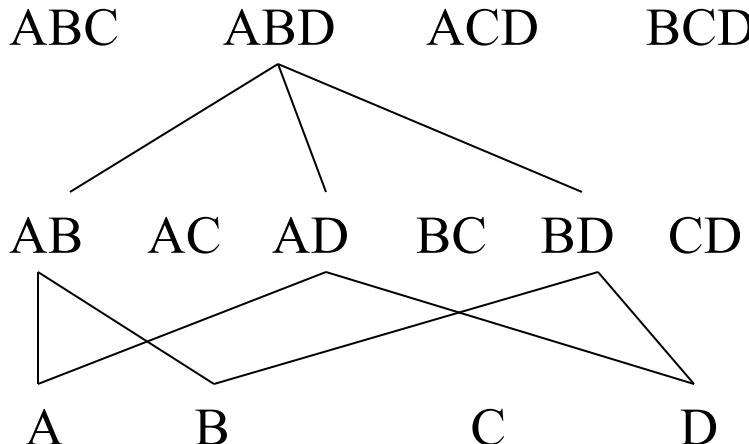
- Naïve algorithm
  - Enumerate all possible itemsets and check their support against *min\_sup*
  - Generate all association rules and check their confidence against *min\_conf*
- The Apriori property
  - Apriori Algorithm
  - FP-growth Algorithm

# All Candidate Itemsets for $\{A, B, C, D, E\}$

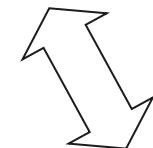


# Apriori Property

- A *frequent* (used to be called *large*) *itemset* is an itemset whose support is  $\geq \text{min\_sup}$ .
- Apriori property (downward closure): any **subsets** of a frequent itemset are also frequent itemsets
- Aka the **anti-monotone** property of support



“any **supersets** of an **infrequent** itemset are also **infrequent** itemsets”

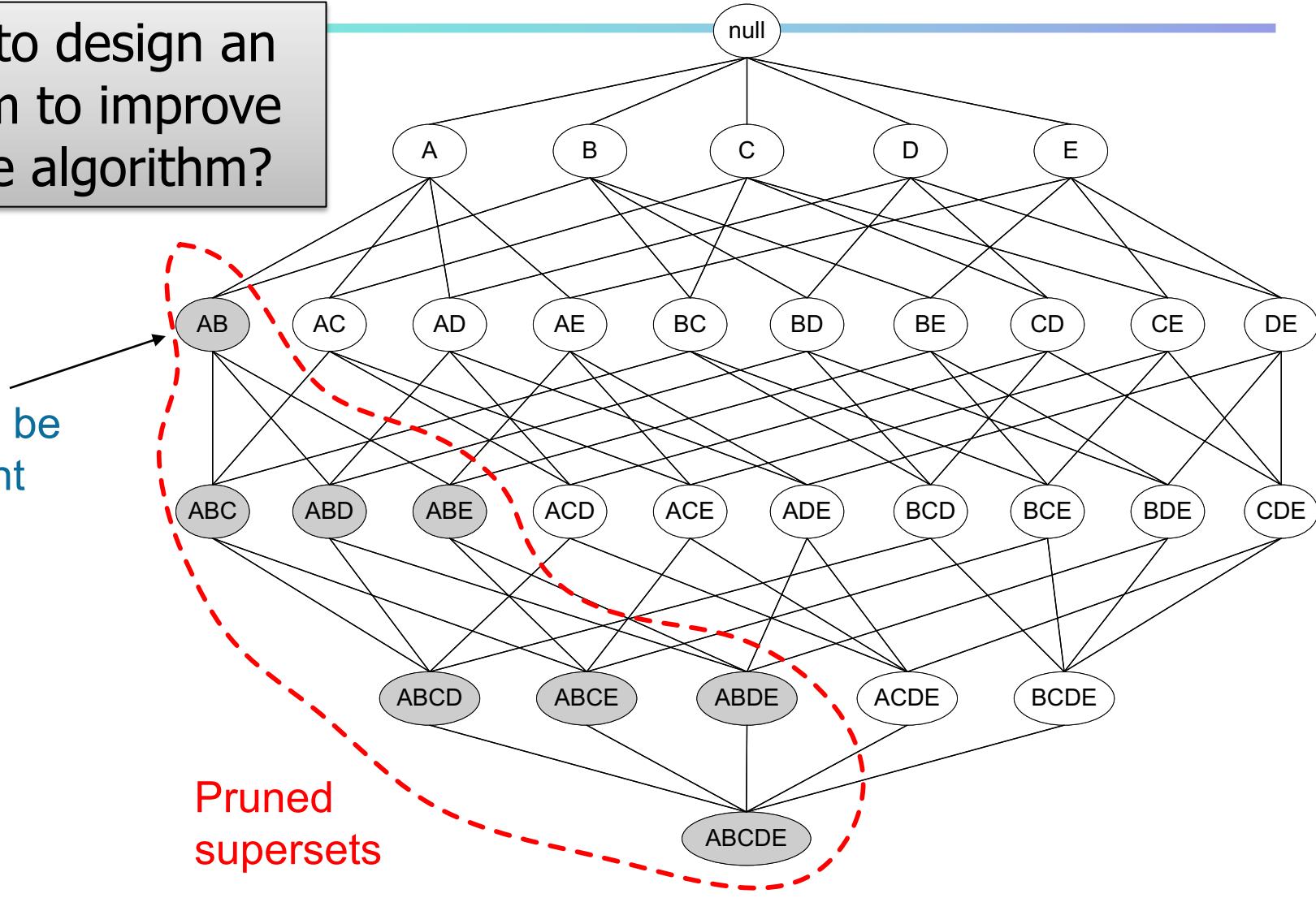


# Illustrating Apriori Principle

Q: How to design an algorithm to improve the naïve algorithm?

Found to be Infrequent

Pruned supersets



# Apriori: A Candidate Generation-and-test Approach

---

- **Apriori pruning principle:** If there is **any** itemset which is infrequent, its superset should not be generated/tested!
- Algorithm [Agrawal & Srikant 1994]
  1.  $C_k \leftarrow$  Perform level-wise candidate generation (from singleton itemsets)
  2.  $L_k \leftarrow$  Verify  $C_k$  against  $L_k$
  3.  $C_{k+1} \leftarrow$  generated from  $L_k$
  4. Goto 2 if  $C_{k+1}$  is not empty

# The Apriori Algorithm

---

- Pseudo-code:

$C_k$ : Candidate itemset of size  $k$

$L_k$ : frequent itemset of size  $k$

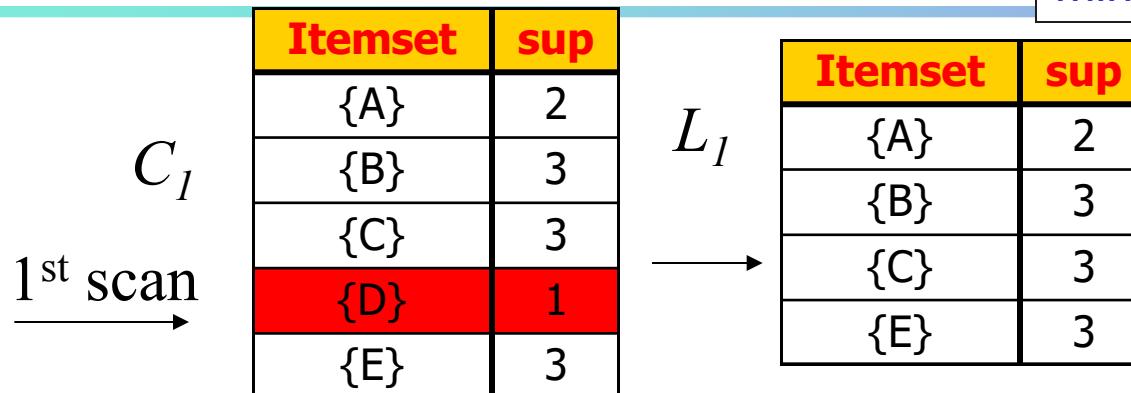
```
 $L_1 = \{\text{frequent items}\};$ 
for ( $k = 1; L_k \neq \emptyset; k++$ ) do begin
     $C_{k+1} = \text{candidates generated from } L_k;$ 
    for each transaction  $t$  in database do begin
        increment the count of all candidates in  $C_{k+1}$ 
        that are contained in  $t$ 
    end
     $L_{k+1} = \text{candidates in } C_{k+1} \text{ with min\_support}$ 
end
return  $\bigcup_k L_k;$ 
```

# The Apriori Algorithm—An Example

$\text{minsup} = 50\%$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E



$L_2$

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

$C_2$

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

$C_2$

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

$2^{\text{nd}}$  scan

$C_3$

Itemset
{B, C, E}

$L_3$

Itemset	sup
{B, C, E}	2

$3^{\text{rd}}$  scan

# Important Details of Apriori

---

1. How to generate candidates?
  - Step 1: self-joining  $L_k$  (**what's the join condition? why?**)
  - Step 2: pruning
2. How to count supports of candidates?

## Example of Candidate-generation

- $L_3 = \{abc, abd, acd, ace, bcd\}$
- Self-joining:  $L_3 * L_3$ 
  - $abcd$  from  $abc$  and  $abd$
  - $acde$  from  $acd$  and  $ace$
- Pruning:
  - $acde$  is removed because  $ade$  is not in  $L_3$
- $C_4 = \{abcd\}$

# Generating Candidates in SQL

- Suppose the items in  $L_{k-1}$  are listed in an order
- Step 1: self-joining  $L_{k-1}$

```
insert into  $C_k$ 
select  $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$ 
from  $L_{k-1} p, L_{k-1} q$ 
where  $p.item_1=q.item_1, \dots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} <$ 
       $q.item_{k-1}$ 
```

- Step 2: pruning

```
forall itemsets  $c$  in  $C_k$  do
    forall ( $k-1$ )-subsets  $s$  of  $c$  do
        if ( $s$  is not in  $L_{k-1}$ ) then delete  $c$  from  $C_k$ 
```

# Derive rules from frequent itemsets

---

- Frequent itemsets  $\neq$  association rules
- One more step is required to find association rules
- For each frequent itemset  $X$ ,  
For each **proper nonempty** subset  $A$  of  $X$ ,
  - Let  $B = X - A$
  - $A \rightarrow B$  is an association rule if
    - Confidence ( $A \rightarrow B$ )  $\geq min\_conf$ ,  
where support ( $A \rightarrow B$ ) = support (AB), and  
confidence ( $A \rightarrow B$ ) = support (AB) / support (A)

# Example – deriving rules from frequent itemsets

---

- Suppose 234 is frequent, with supp=50%
  - Proper nonempty subsets: 23, 24, 34, 2, 3, 4, with supp=50%, 50%, 75%, 75%, 75%, 75% respectively
  - These generate these association rules:
    - 23 => 4, confidence=100%
    - 24 => 3, confidence=100%
    - 34 => 2, confidence=67% =  $(N * 50\%) / (N * 75\%)$
    - 2 => 34, confidence=67%
    - 3 => 24, confidence=67%
    - 4 => 23, confidence=67%
    - All rules have support = 50%

Q: is there any optimization (e.g., pruning) for this step?

# Deriving rules

---

- To recap, in order to obtain  $A \rightarrow B$ , we need to have  $\text{Support}(AB)$  and  $\text{Support}(A)$
- This step is not as time-consuming as frequent itemsets generation
  - Why?
- It's also easy to speedup using techniques such as parallel processing.
  - How?
- Do we really need candidate generation for deriving association rules?
  - Frequent-Pattern Growth (FP-Tree)

# Bottleneck of Frequent-pattern Mining

---

- Multiple database scans are **costly**
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset  $i_1 i_2 \dots i_{100}$ 
    - # of scans: **100**
    - # of Candidates:  $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1$
- Bottleneck: candidate-generation-and-test

*Can we avoid candidate generation **altogether**?*

---

- FP-growth

# No Pain, No Gain

	<u>Java</u>	<u>Lisp</u>	<u>Scheme</u>	<u>Python</u>	<u>Ruby</u>
Alice	X				X
Bob				X	X
Charlie	X			X	X
Dora		X	X		

minsup = 1

- Apriori:

- $L_1 = \{J, L, S, P, R\}$
- $C_2 = \text{all the } ({}^5_2) \text{ combinations}$ 
  - Most of  $C_2$  do not contribute to the result
  - There is no way to tell because

# No Pain, No Gain

	<u>Java</u>	<u>Lisp</u>	<u>Scheme</u>	<u>Python</u>	<u>Ruby</u>
Alice	X				X
Bob				X	X
Charlie	X			X	X
Dora		X	X		

## Ideas:

- Keep the **support set** for each frequent itemset
- DFS

minsup = 1

$J \rightarrow JL?$

$J \rightarrow ???$

Only need to look at support set for J

{A, C}

J

$\emptyset$

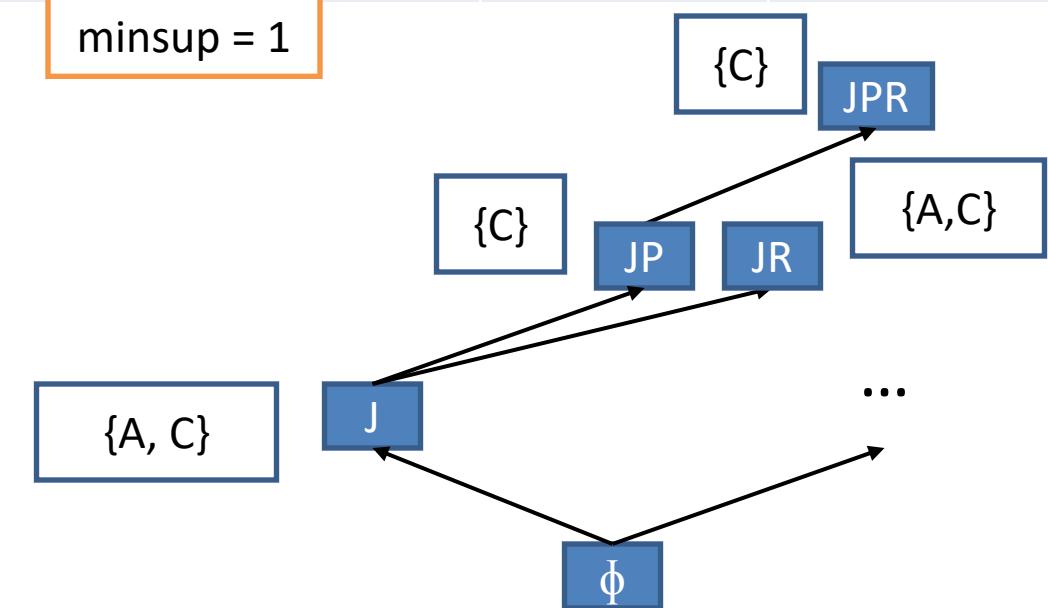
# No Pain, No Gain

	<u>Java</u>	<u>Lisp</u>	<u>Scheme</u>	<u>Python</u>	<u>Ruby</u>
Alice	X				X
Bob				X	X
Charlie	X			X	X
Dora		X	X		

## Ideas:

- Keep the support set for each frequent itemset
- DFS

minsup = 1



# Notations and Invariants

---

- ConditionalDB:
  - $DB|p = \{t \in DB \mid t \text{ contains itemset } p\}$
  - $DB = DB|\emptyset$  (i.e., conditioned on nothing)
  - Shorthand:  $DB|px = DB|(p \cup x)$
- $\text{SupportSet}(p \cup x, DB) = \text{SupportSet}(x, DB|p)$ 
  - $\{x \mid x \bmod 6 = 0 \wedge x \in [100]\} =$   
 $\{x \mid x \bmod 3 = 0 \wedge x \in \text{even}([100])\}$
- A FP-tree|p is equivalent to a DB|p
  - One can be converted to another
  - Next, we illustrate the alg using conditionalDB

# FP-tree Essential Idea /1

- Recursive algorithm again!

- Freq**Itemsets**(DB|p):

- $X = \text{FindFrequentItems}(DB|p)$

easy task, as  
only items (not  
itemsets) are  
needed

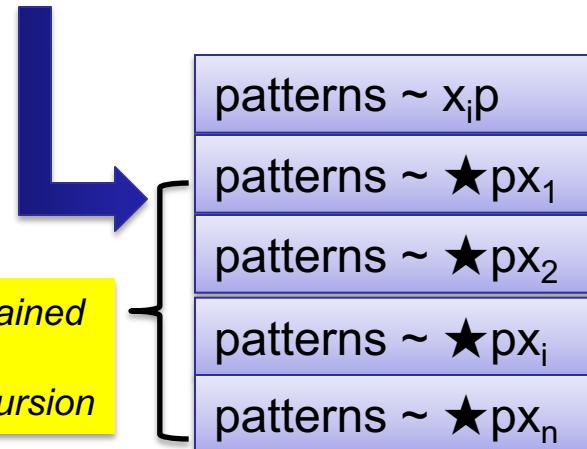
all frequent itemsets in  
DB|p belong to one of  
the following  
categories:

output  $\{ (x \ p) \mid x \in X \}$

- Foreach  $x$  in  $X$

- $DB^*|px = \text{GetConditionalDB}^+(DB^*|p, x)$
- 
- $\text{FreqItemsets}(DB^*|px)$

*obtained  
via  
recursion*



# No Pain, No Gain

DB J	Java	Lisp	Scheme	Python	Ruby
Alice	X				X
Charlie	X			X	X

minsup = 1

- Freq**Itemsets**(DB|J):
  - $\{P, R\} \leftarrow \text{FindFrequentItems}(DB|J)$
  - Output  $\{JP, JR\}$
  - Get  $DB^*|JP$ ; Freq**Itemsets**( $DB^*|JP$ )
  - Get  $DB^*|JR$ ; Freq**Itemsets**( $DB^*|JR$ )
  - // Guaranteed no other frequent itemset in DB|J

# FP-tree Essential Idea /2

- Freq**Itemsets**(DB|p):

- If boundary condition, then ...
- X = FindFrequent**Items**(DB|p)
- [optional] DB\*|p = PruneDB(DB|p, X)  
output { (x p) | x ∈ X }
- Foreach **x** in X
  - DB\*|px = GetConditionalDB+(DB\*|p, x)
  - [optional] if DB\*|px is degenerated, then powerset(DB\*|px)
  - Freq**Itemsets**(DB\*|px)

Also output each item in X (appended with the conditional pattern)

Remove items not in X; potentially reduce # of transactions ( $\emptyset$  or dup). Improves the efficiency.

Also gets rid of items already processed before x → *avoid duplicates*

# Lv 1 Recursion

- $\text{minsup} = 3$

F	C	A	D	G	I	M	P
A	B	C	F	L	M	O	
B	F	H	J	J	O	W	
B	C	K	S	P			
A	F	C	E	L	P	M	N

DB

 $X = \{F, C, A, B, M, P\}$ 

Output: F, C, A, B, M, P

F	C	A	M	P
F	C	A	B	M
F	B			
C	B	P		
F	C	A	M	P

DB\*

F	C	A	M	P
C	B	P		
F	C	A	M	P

DB\*|P

DB\*|M (sans P)

DB\*|B (sans MP)

DB\*|A (sans BMP)

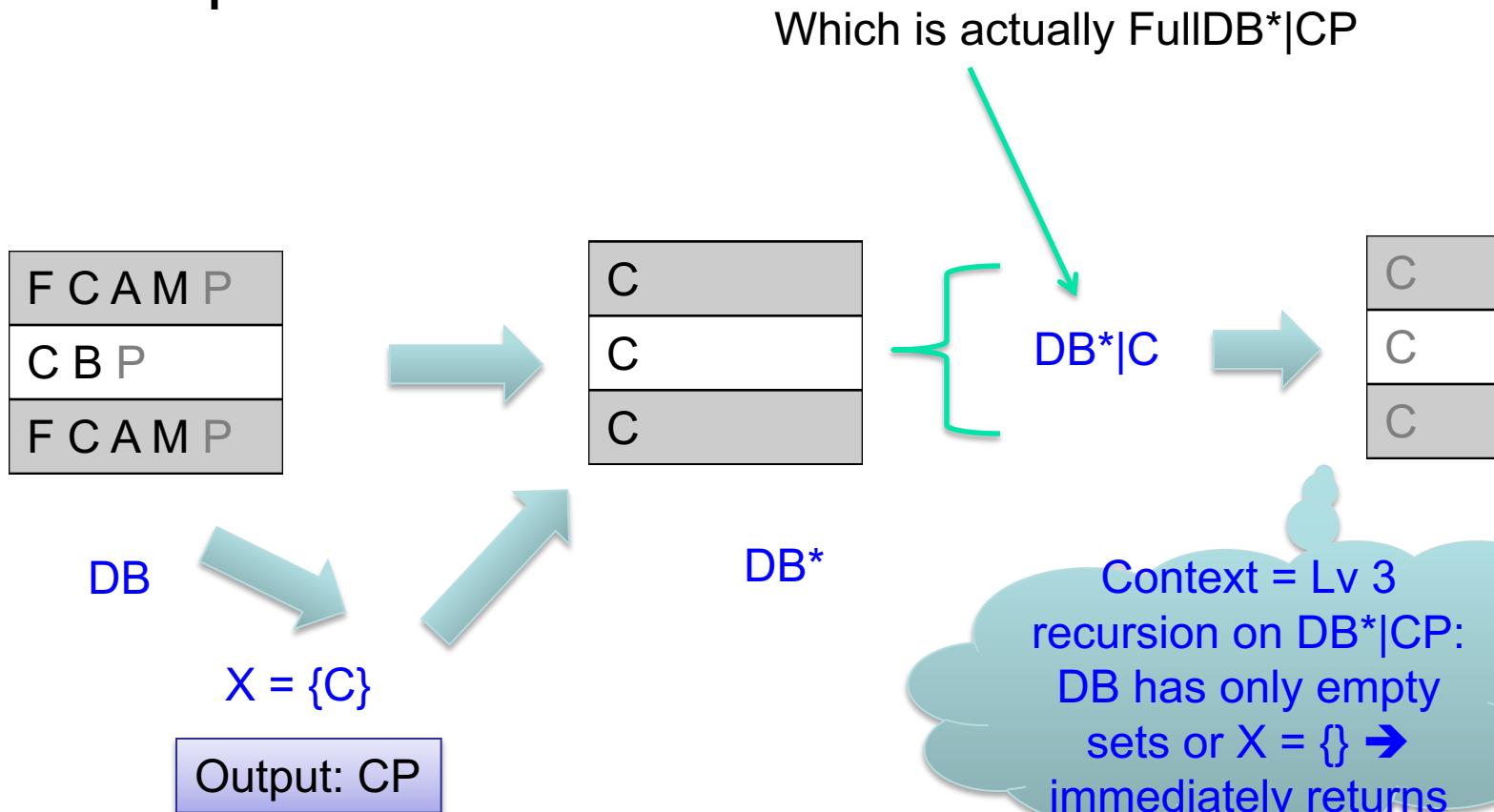
DB\*|C (sans ABMP)

DB\*|F (sans CABMP)

F	C	A
F	C	A
F	C	A

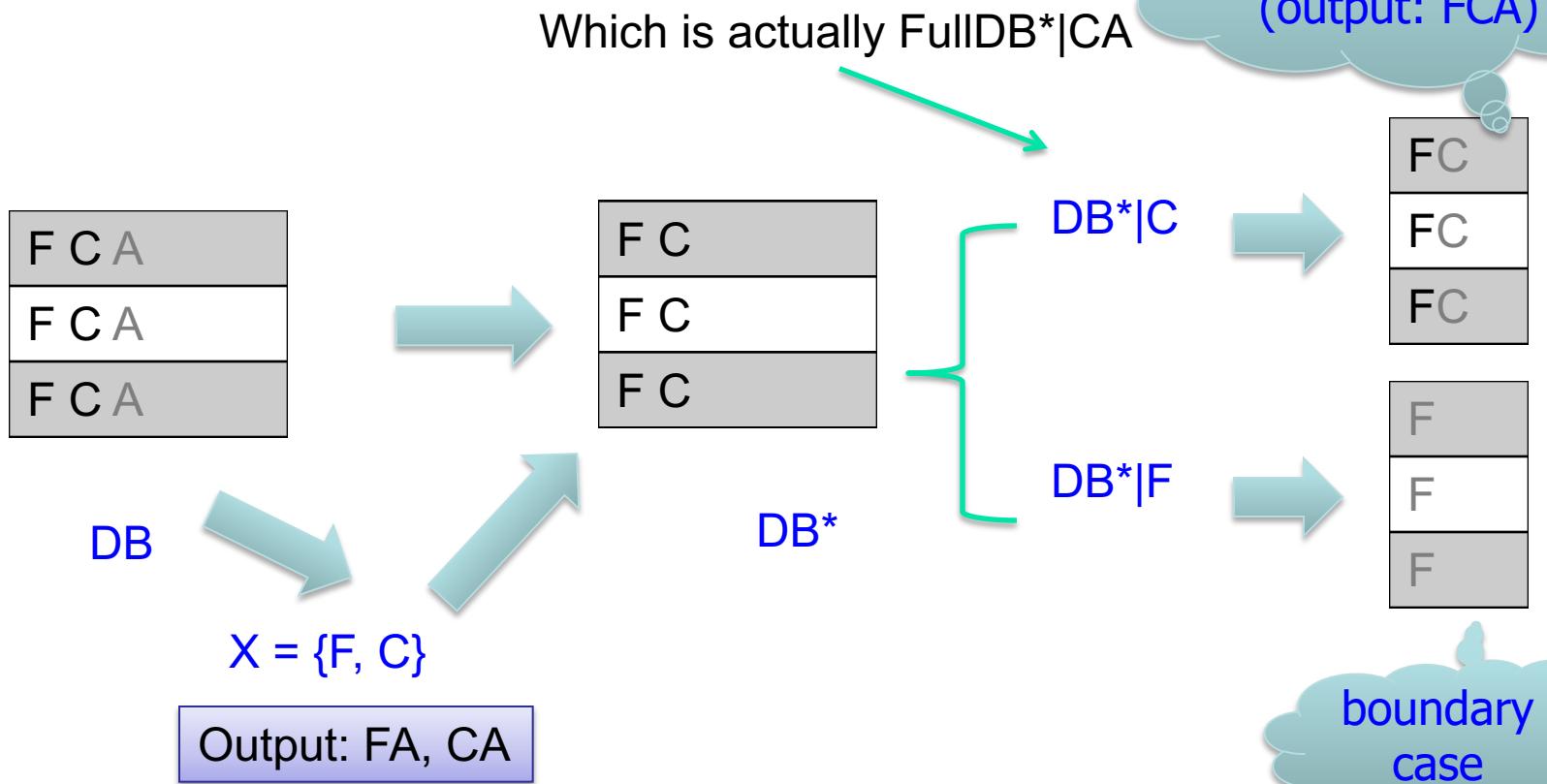
# Lv 2 Recursion on DB\*|P

- $\text{minsup} = 3$



# Lv 2 Recursion on DB\*|A (sans ...)

- $\text{minsup} = 3$



# Different Example: Lv 2 Recursion on DB\*|P

- $\text{minsup} = 2$

F	C	A	M	P
F	C	B	P	
F	A	P		



F	C	A
F	C	
F	A	

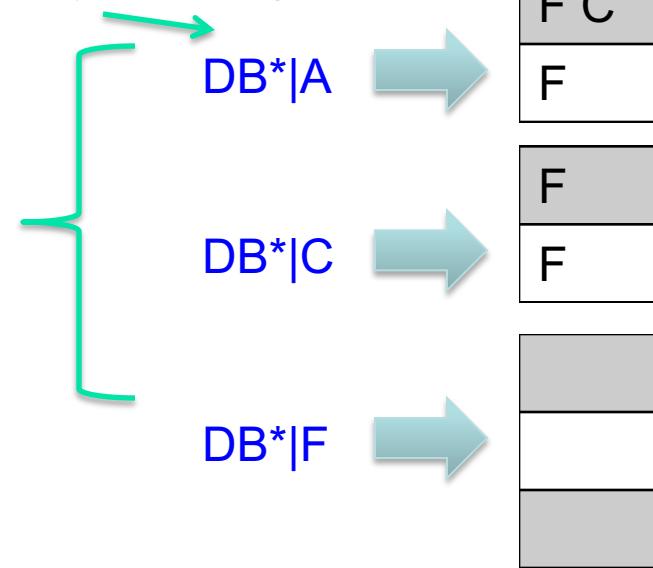
DB\*

DB

$X = \{F, C, A\}$

Output: FP, CP, AP

Which is actually FullDB\*|AP



Output: FAP

$X = \{F\}$

F
F

F	C
F	

F
F


# I will give you back the FP-tree

---

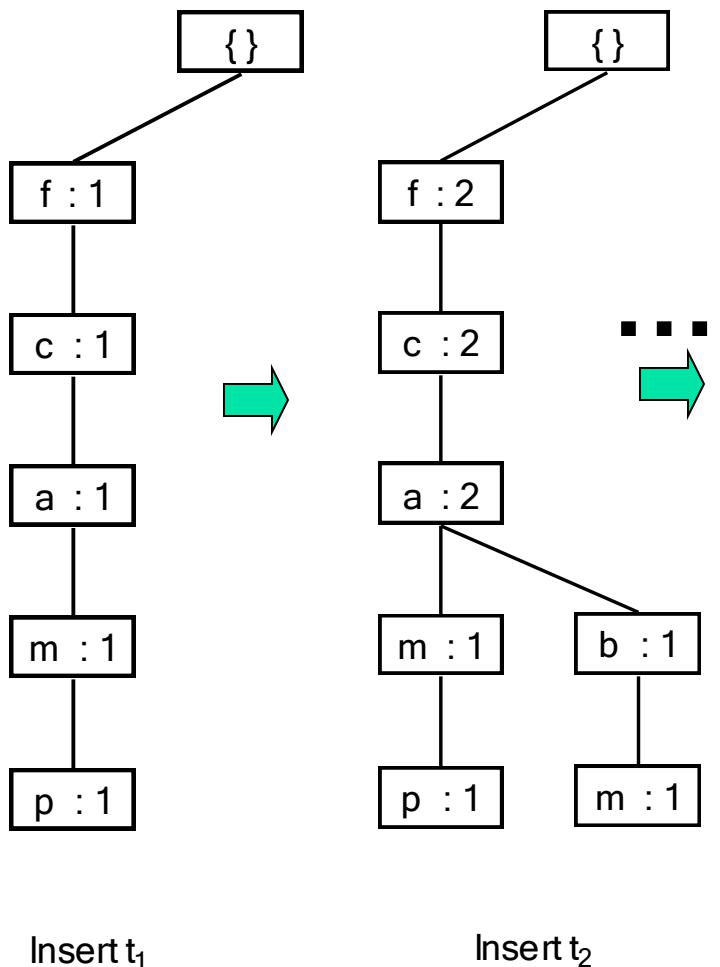
- An FP-tree tree of DB consists of:
  - A fixed **order** among items in DB
  - A prefix, **threaded** tree of **sorted** transactions in DB
  - Header table: (item, freq, ptr)
- When used in the algorithm, the input DB is always pruned (c.f., PruneDB())
  - Remove infrequent items
  - Remove infrequent items in every transaction

# FP-tree Example

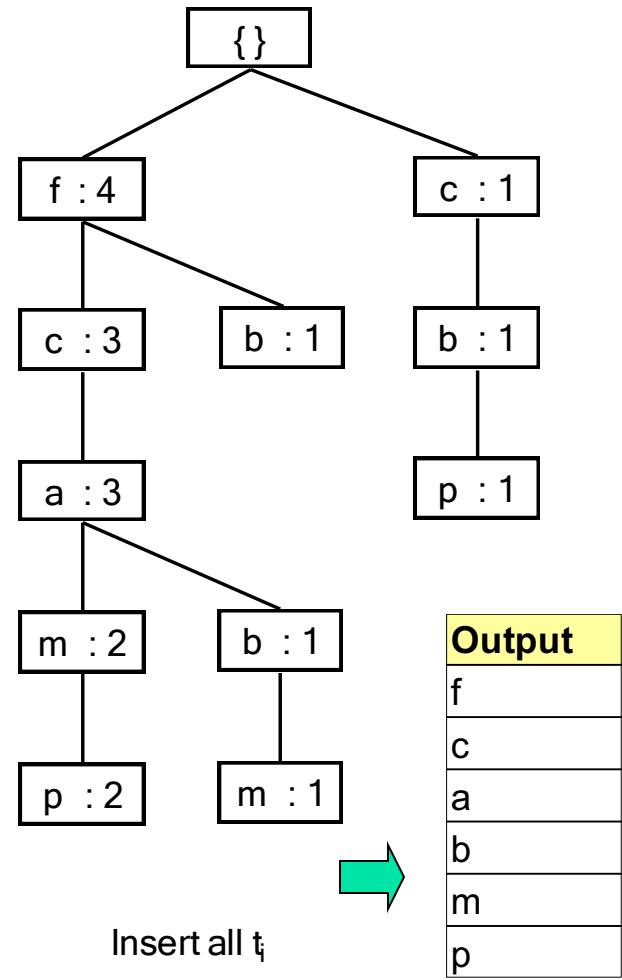
minsup = 3

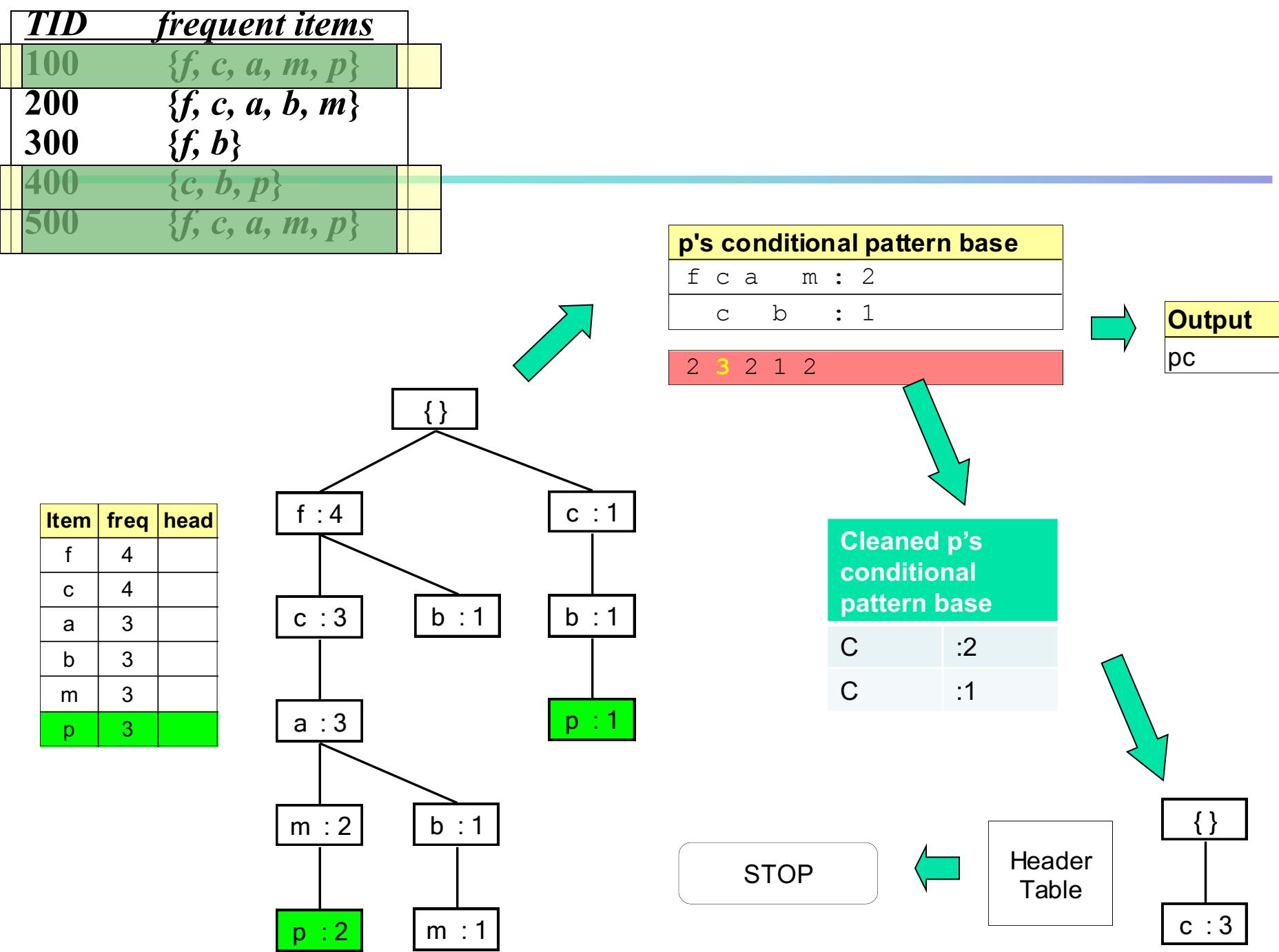
<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

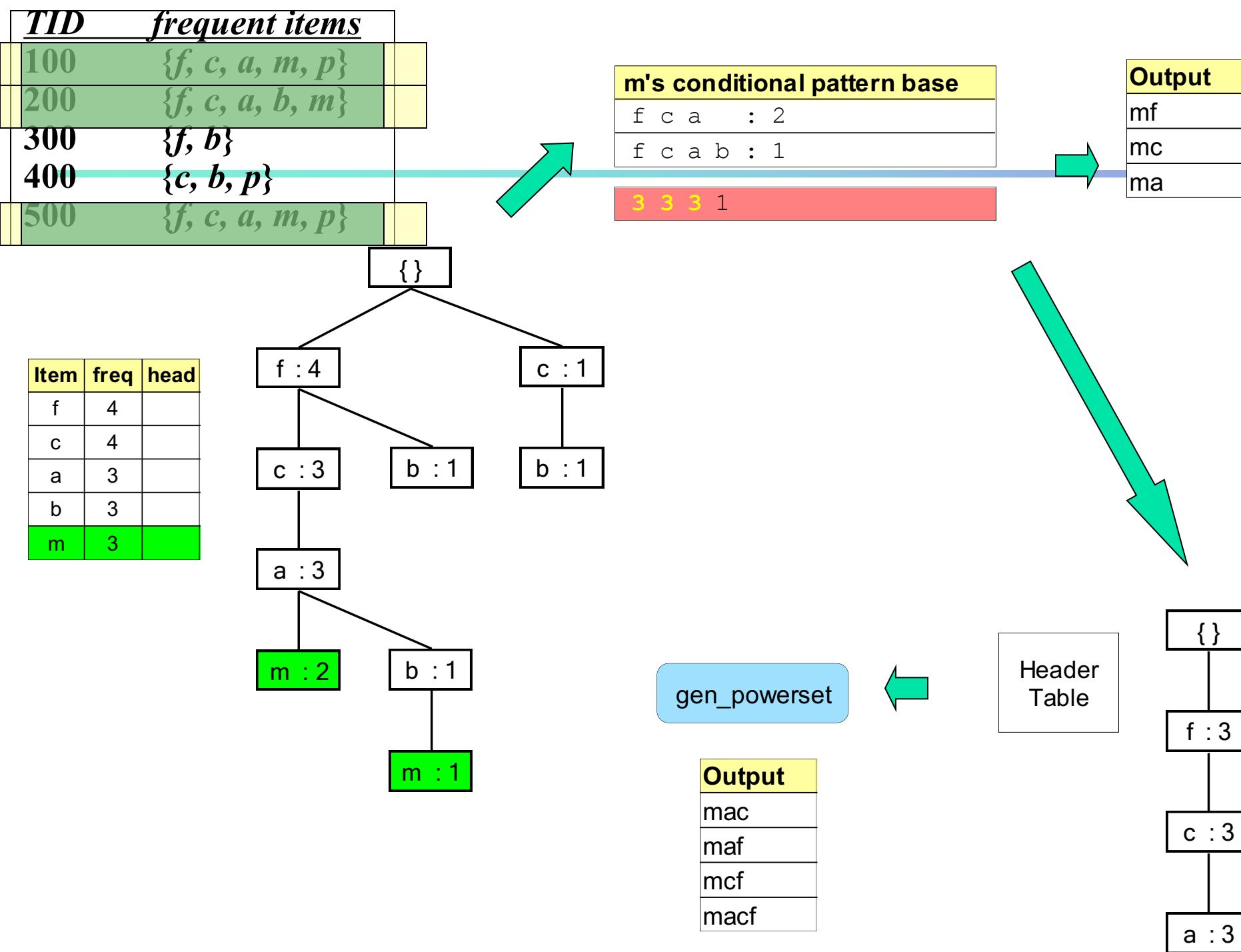
<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}



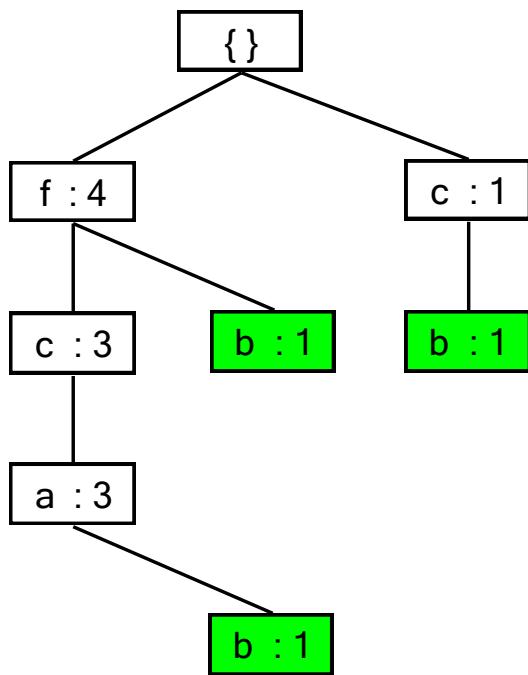
Item	freq	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	







Item	freq	head
f	4	
c	4	
a	3	
b	3	



b's conditional pattern base

---

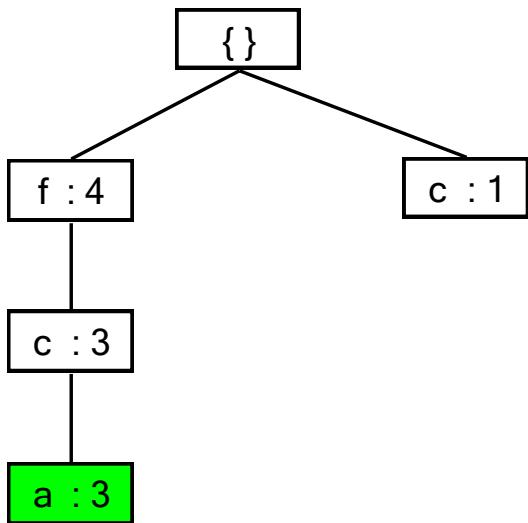
f	c	a	:	1
f			:	1
	c		:	1

---

2 2 1

STOP

Item	freq	head
f	4	
c	4	
a	3	



**a's conditional pattern base**

f	c	:	3
3 3			

**Output**

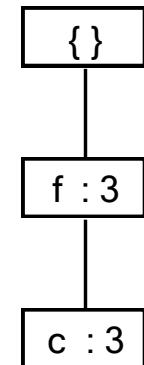
af
ac

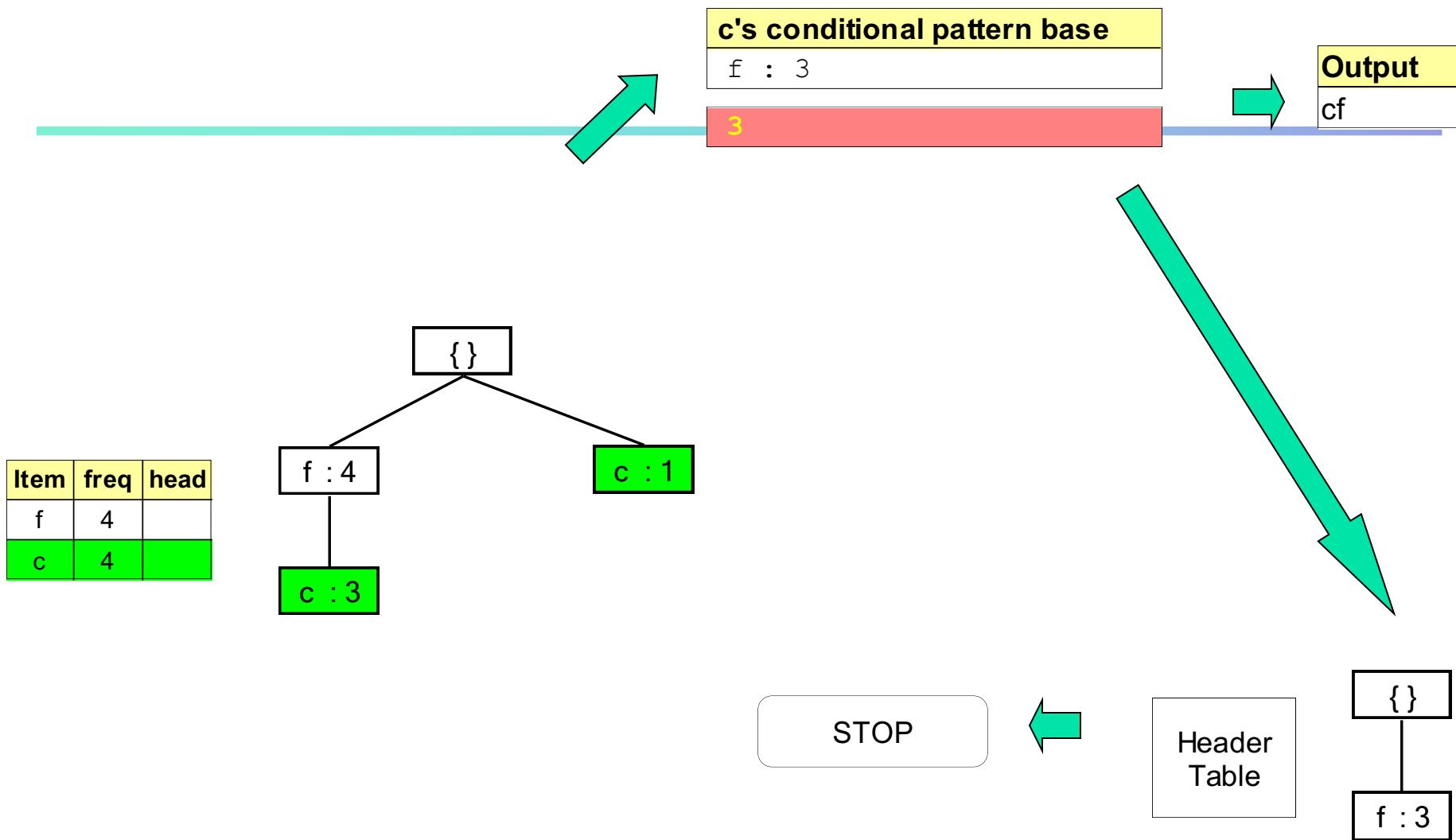
gen\_powerset

Header Table

**Output**

acf
-----







STOP



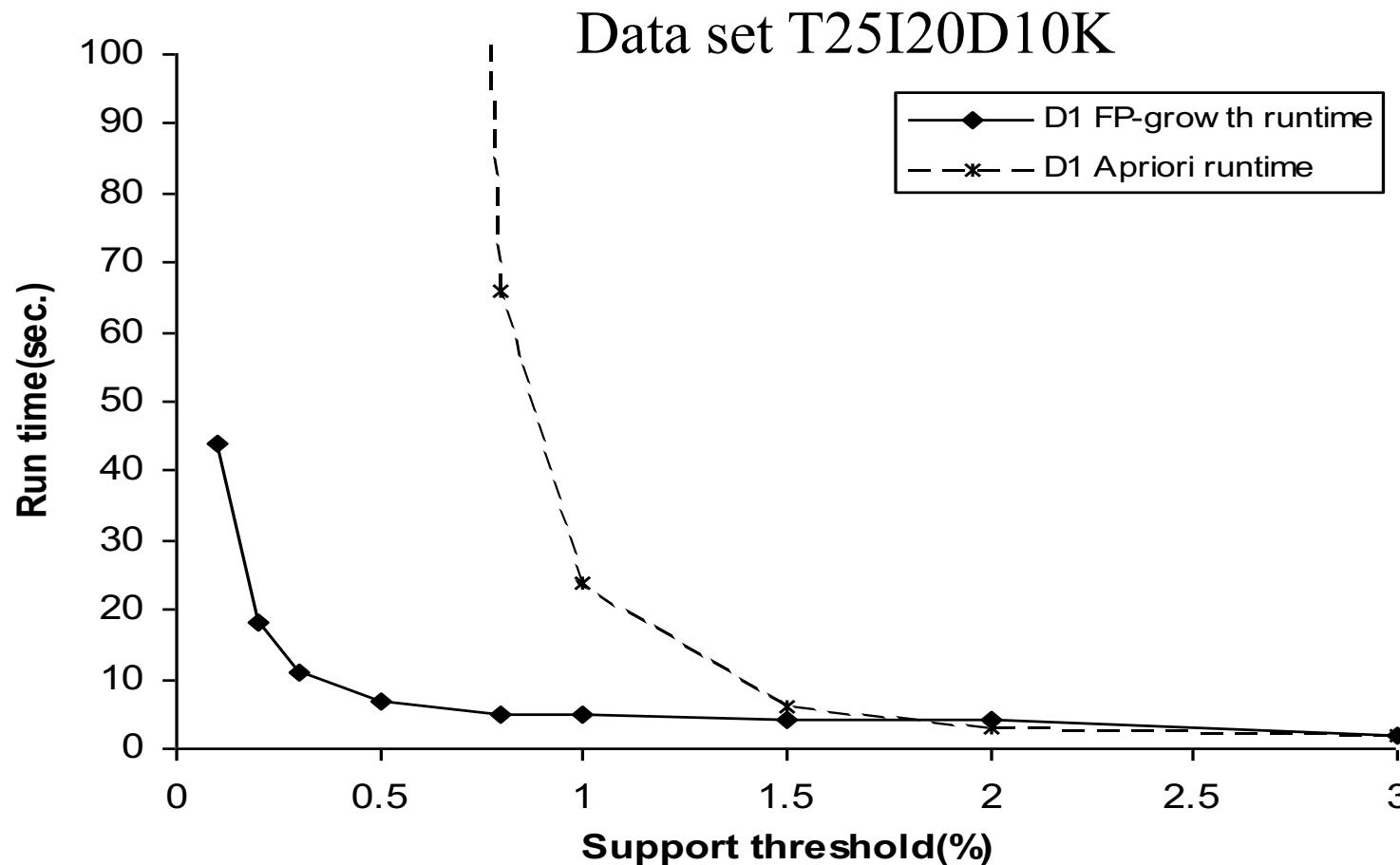
{ }

f : 4

Item	freq	head
f	4	

# FP-Growth vs. Apriori: Scalability With the Support Threshold

---



# Why Is FP-Growth the Winner?

---

- Divide-and-conquer:
  - decompose both the mining task and DB according to the frequent patterns obtained so far
  - leads to focused search of smaller databases
- Other factors
  - no candidate generation, no candidate test
  - compressed database: FP-tree structure
  - no repeated scan of entire database
  - basic ops—counting local freq items and building sub FP-tree, no pattern search and matching