Multiplication and Division

Multiplication: Unsigned V/s Signed

1001	(9)		1001	(−7)	
× 0011	(3)		× 0011	(3)	
00001001	1001 × 2^0		11111001	(−7) × 2^0	= (−7)
00010010	1001 × 2^1		11110010	(−7) × 2^1	= (−14)
00011011	(27)		11101011	(−21)	

(a) Unsigned integers (b) Twos complement integers

Figure 10.11 Comparison of Multiplication of Unsigned and Twos Complement Integers

Multiplication: Booth's Algorithm

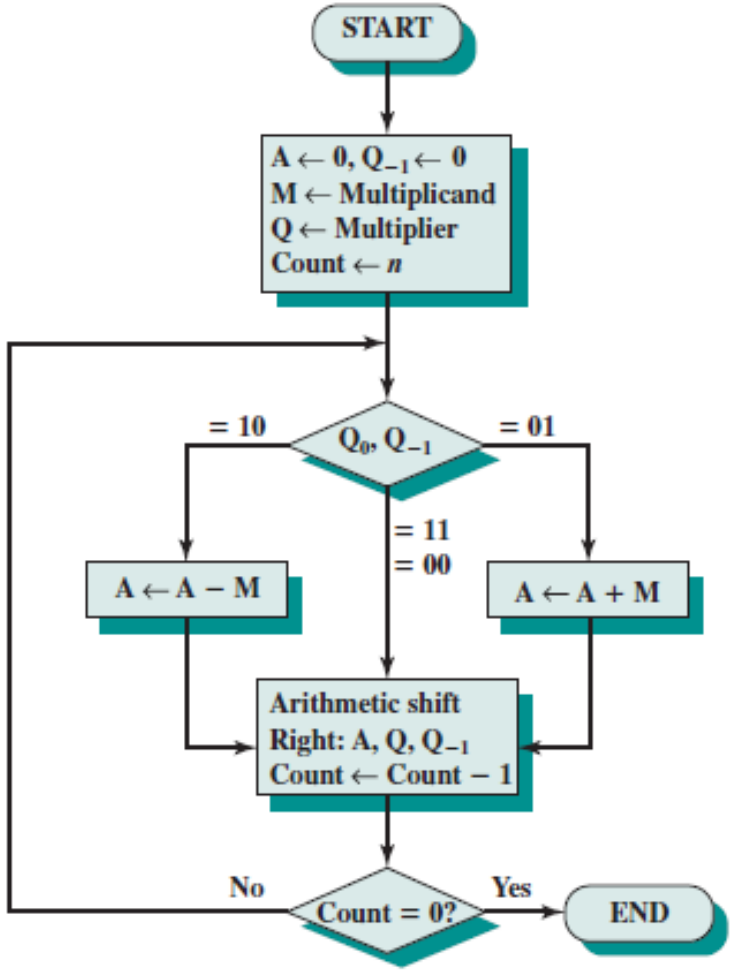


Figure 10.12 Booth's Algorithm for Twos Complement Multiplication

Multiplication: Booth's Algorithm

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial values	
1001	0011	0	0111	A ← A − M	First cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second cycle
0101	0100	1	0111	A ← A + M	Third cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth cycle

Figure 10.13 Example of Booth's Algorithm (7 × 3)

Booth's algorithm: 13X-6

- M=13=01101 Q=-6=11010 -M=10011

A	Q	Q_{-1}	M	
00000	11010	0	01101	Initial values
00000	01101	0	01101	Shift Right-1st cycle
10011 10011 *11001*	01101 *10110*	0 *1*	01101 01101	A=A-M [Subtract M from A(Adding 2's complement of M)] *Shift Right- 2nd cycle*
01101 00110 *00011*	10110 *01011*	1 *0*	01101	A=A+M Shift Right- 3rd cycle
10011 10110 *11011*	01011 *00101*	0 *1*	01101 01101	A=A-M [Subtract M from A(Adding 2's complement of M)] Shift Right- 4th cycle
11101	**10010**	1	01101	Shift Right 5th cycle
Taking 2's complement	**0001001110→78 Product=-78**			

Booth's algorithm: 23X29

- M=23=010111 Q=29=011101 -M=101001

A	Q	Q_{-1}	M	
000000	011101	0	010111	Initial values
101001 101001 110100	 011101 101110	 0 1	 010111	 A=A-M Shift Right-1st cycle
010111 001011 000101	 101110 110111	 1 0	 010111	 A=A+M *Shift Right- 2nd cycle*
101001 101110 110111	 110111 011011	 0 1	 010111	 A=A-M Shift Right- 3rd cycle
111011	101101	1	010111	Shift Right- 4th cycle
111101	110110	1	010111	Shift Right 5th cycle
010111 010100 001010	 110110 011011	 1 0	 010111	 A=A+M *Shift Right- 6th cycle*
	001010011011→667 Product=667			

Multiplication: Booth's Algorithm

0111			0111		
× 0011	(0)		× 1101	(0)	
11111001	1–0		11111001	1–0	
0000000	1–1		0000111	0–1	
000111	0–1		111001	1–0	
00010101	(21)		11101011	(–21)	

(a) (7) × (3) = (21)

(b) (7) × (–3) = (–21)

1001			1001		
× 0011	(0)		× 1101	(0)	
00000111	1–0		00000111	1–0	
0000000	1–1		1111001	0–1	
111001	0–1		000111	1–0	
11101011	(–21)		00010101	(21)	

(c) (–7) × (3) = (–21)

(d) (–7) × (–3) = (21)

Figure 10.14 Examples Using Booth's Algorithm

Division

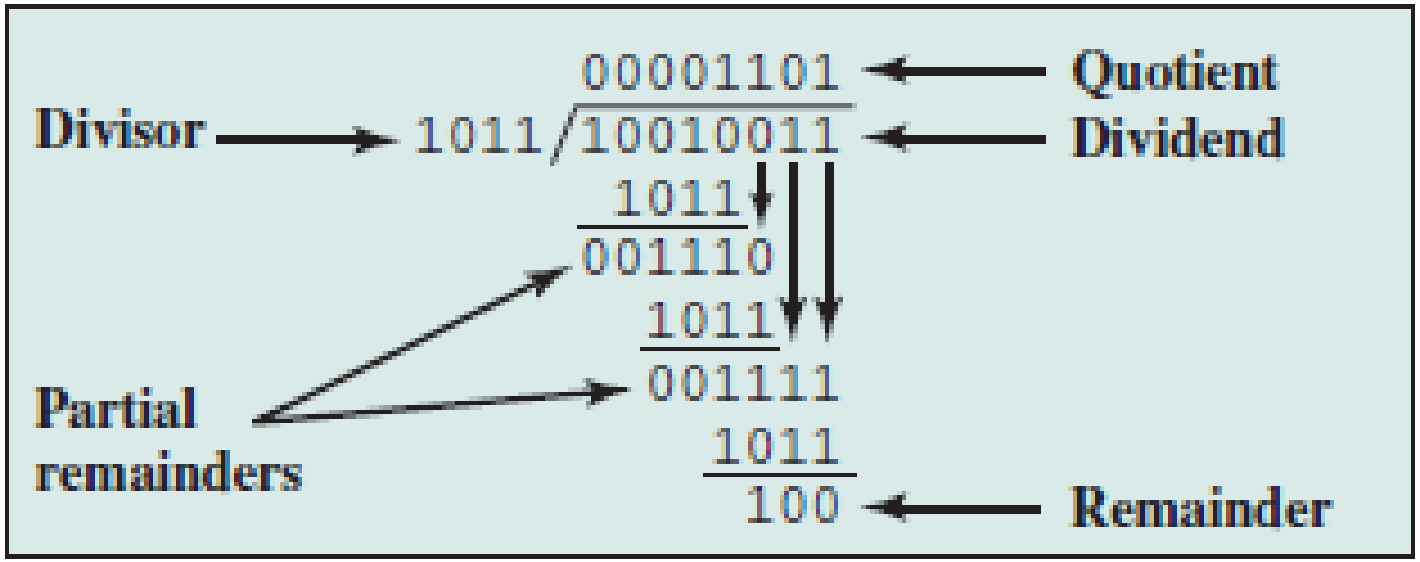


Figure 10.15 Example of Division of Unsigned Binary Integers

Division

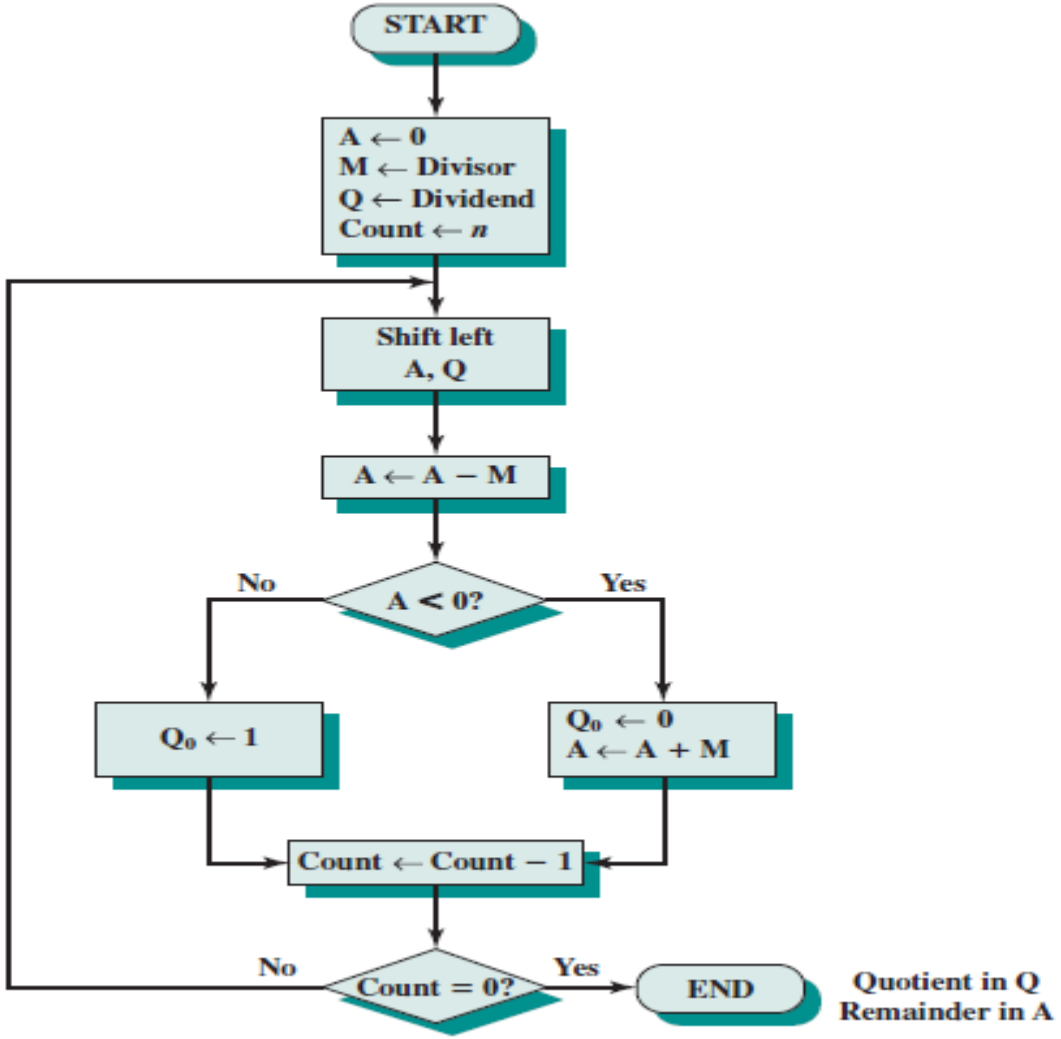


Figure 10.16 Flowchart for Unsigned Binary Division

Division

A	Q	
0000	0111	Initial value
0000	1110	Shift
1101		Use twos complement of 0011 for subtraction
1101		Subtract
0000	1110	Restore, set $Q_0 = 0$
0001	1100	Shift
1101		
1110		Subtract
0001	1100	Restore, set $Q_0 = 0$
0011	1000	Shift
1101		
0000	1001	Subtract, set $Q_0 = 1$
0001	0010	Shift
1101		
1110		Subtract
0001	0010	Restore, set $Q_0 = 0$

Figure 10.17 Example of Restoring Twos Complement Division (7/3)

Division: Algorithm

- Assumption: divisor V and the dividend D are positive and that |V|< |D|.
- If |V|= |D|, then the quotient =1 and the remainder=0.
- If |V|> |D|, then Q=0 and R=D. The algorithm can be summarized as follows:

1. Load the twos complement of the divisor into the M register; that is, the M register contains the negative of the divisor. Load the dividend into the A, Q registers. The dividend must be expressed as a 2n-bit positive number. Thus, for example, the 4-bit 0111 becomes 00000111.
2. Shift A, Q left 1 bit position.
3. Perform A=A-M. This operation subtracts the divisor from the contents of A.
4.
    a. If the result is nonnegative (most significant bit of A=0), then set Q0=1
    b. If the result is negative (most significant bit of A=1), then set Q0=0, and restore the previous value of A.
5. Repeat steps 2 through 4 as many times as there are bit positions in Q.
6. The remainder is in A and the quotient is in Q.

Division: Example

- Divide 8 by 3; M=-3=11101; Q=01000

A	Q	
00000	01000	Initial values
00000 11101 11101 + 00011 00000	10000 10000 10000	Shift Left Subtract Divisor(Add 2's complement of divisor) Set Q0=0 Restore 1st Cycle
00001 11101 11110 + 00011 00001	00000 00000 00000	Shift Left Subtract Divisor(Add 2's complement of divisor) Set Q0=0 Restore 2nd Cycle
00010 11101 11111+ 00011 00010	00000 00000 00000	Shift Left Subtract Divisor(Add 2's complement of divisor) Set Q0=0 Restore 3rd Cycle
00100 11101 00001	00000 00001	Shift Left Subtract Set Q0=1 4th cycle
00010 11101 11111 + 00011 00010	00010 00010 00010	Shift left Subtract Divisor(Add 2's complement of divisor) Set Q0=0 Restore 5th Cycle

Division

- Consider the following examples of integer division with all possible combinations of signs of D and V:

  D = 7 V = 3 ➔ Q = 2 R = 1

  D = 7 V = -3 ➔ Q = -2 R = 1

  D = -7 V = 3 ➔ Q = -2 R = -1

  D = -7 V = -3 ➔ Q = 2 R = -1

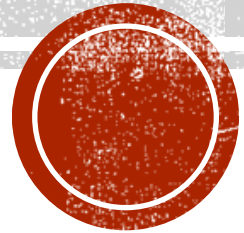
- (-7)/(3) and (7)/(-3) produce different remainders.
- The magnitudes of Q and R are unaffected by the input signs
- The signs of Q and R are easily derivable from the signs of D and V.
  - sign(R) = sign(D)
  - sign(Q) = sign(D) * sign(V).
- One way to do twos complement division is to convert the operands into unsigned values and, at the end, to account for the signs by complementation where needed.
- This is the method of choice for the restoring division algorithm.

Exercise

1. Given x and y in twos complement notation i.e., x=0101 and y=1010,compute the product p=x*y with Booth's algorithm
2. Use the Booth algorithm to multiply 23 (multiplicand) by 29 (multiplier), where each number is represented using 6 bits
3. Divide 145 by 13 in binary twos complement notation, using 12-bit words. Use the restoring division algorithm

IMPROVING THE EFFICIENCY OF APRIORI

"How can we further improve the efficiency of Apriori-based mining?" Many variations of the Apriori algorithm have been proposed that focus on improving the efficiency of the original algorithm. Several of these variations are summarized here.



HASH-BASED TECHNIQUE

A hash-based technique can be used to reduce the size of the candidate k-itemsets, C_k, for k >1.

For example, when scanning each transaction in the database to generate the frequent 1-itemsets, L_1, we can generate all the 2-itemsets for each transaction, hash (i.e., map) them into the different buckets of a hash table structure, and increase the corresponding bucket counts (Figure in next slide). A 2-itemset with a corresponding bucket count in the hash table that is below the support threshold cannot be frequent and thus should be removed from the candidate set. Such a hash-based technique may substantially reduce the number of candidate k-itemsets examined (especially when k=2).

Create hash table H_2 using hash function $h(x, y) = ((order\ of\ x) \times 10 + (order\ of\ y))\ mod\ 7$ →

H_2

bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3} {I1, I3}

Hash table, H_2, for candidate 2-itemsets. This hash table was generated by scanning Table 6.1's transactions while determining L_1. If the minimum support count is, say, 3, then the itemsets in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in C_2.

TRANSACTION REDUCTION

A transaction that does not contain any frequent k-itemsets cannot contain any frequent (k +1)-itemsets. Therefore, such a transaction can be marked or removed from further consideration because subsequent database scans for j-itemsets, where j >k, will not need to consider such a transaction.

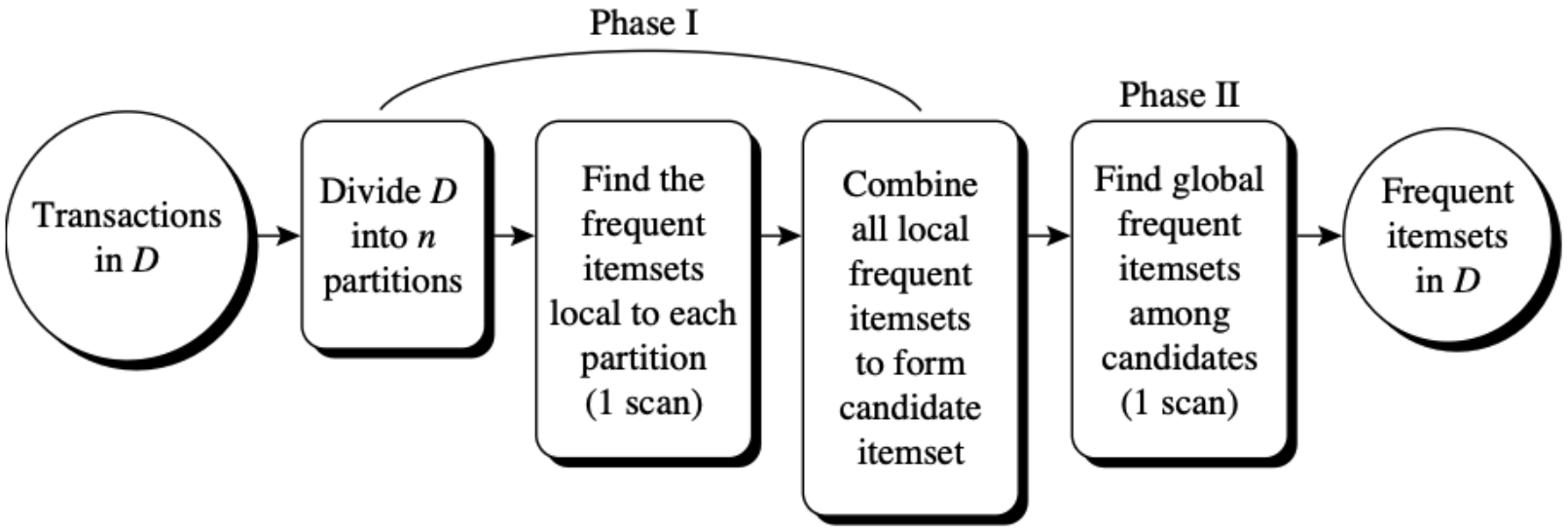
PARTITIONING`

A partitioning technique can be used that requires just two database scans to mine the frequent itemsets (Figure in next slide). It consists of two phases. In phase I, the algorithm divides the transactions of D into n nonoverlapping partitions. If the minimum relative support threshold for transactions in D is min sup, then the minimum support count for a partition is min sup ×the number of transactions in that partition. For each partition, all the local frequent itemsets (i.e., the itemsets frequent within the partition) are found.

A local frequent itemset may or may not be frequent with respect to the entire database, D. However, any itemset that is potentially frequent with respect to D must occur as a frequent itemset in at least one of the partitions.

Therefore, all local frequent itemsets are candidate itemsets with respect to D. The collection of frequent itemsets from all partitions forms the global candidate itemsets with respect to D. In phase II, a second scan of D is conducted in which the actual support of each candidate is assessed to determine the global frequent itemsets. Partition size and the number of partitions are set so that each partition can fit into main memory and therefore be read only once in each phase.

PARTITIONING



Mining by partitioning the data.

SAMPLING

The basic idea of the sampling approach is to pick a random sample S of the given data D, and then search for frequent itemsets in S instead of D. In this way, we trade off some degree of accuracy against efficiency. The S sample size is such that the search for frequent itemsets in S can be done in main memory, and so only one scan of the transactions in S is required overall. Because we are searching for frequent itemsets in S rather than in D, it is possible that we will miss some of the global frequent itemsets. To reduce this possibility, we use a lower support threshold than minimum support to find the frequent itemsets local to S (denoted L^S).

The rest of the database is then used to compute the actual frequencies of each itemset in LS. A mechanism is used to determine whether all the global frequent itemsets are included in L^S. If L^S actually contains all the frequent itemsets in D, then only one scan of D is required.

Otherwise, a second pass can be done to find the frequent itemsets that were missed in the first pass. The sampling approach is especially beneficial when efficiency is of utmost importance such as in computationally intensive applications that must be run frequently.

DYNAMIC ITEMSET COUNTING

A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points. In this variation, new candidate itemsets can be added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately before each complete database scan.

The tech-nique uses the count-so-far as the lower bound of the actual count. If the count-so-far passes the minimum support, the itemset is added into the frequent itemset collectionand can be used to generate longer candidates. This leads to fewer database scans thanwith Apriori for finding all the frequent itemsets.

Data Mining:

Concepts and Techniques

(3rd ed.)

— Chapter 3 —

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign &
Simon Fraser University

Data Reduction Strategies

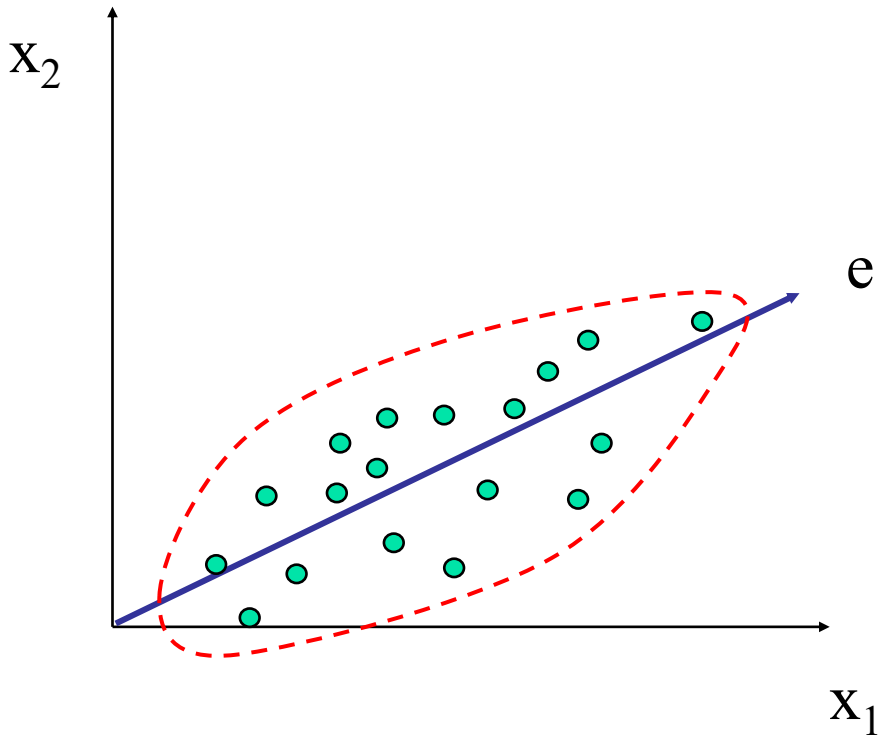
- **Data reduction**: Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Why data reduction? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run on the complete data set.
- Data reduction strategies
  - Dimensionality reduction, e.g., remove unimportant attributes
    - Wavelet transforms
    - Principal Components Analysis (PCA)
    - Feature subset selection, feature creation
  - Numerosity reduction (some simply call it: Data Reduction)
    - Regression and Log-Linear Models
    - Histograms, clustering, sampling
    - Data cube aggregation
  - Data compression

Data Reduction 1: Dimensionality Reduction

- **Curse of dimensionality**
  - When dimensionality increases, data becomes increasingly sparse
  - Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
  - The possible combinations of subspaces will grow exponentially
- **Dimensionality reduction**
  - Avoid the curse of dimensionality
  - Help eliminate irrelevant features and reduce noise
  - Reduce time and space required in data mining
  - Allow easier visualization
- **Dimensionality reduction techniques**
  - Wavelet transforms
  - Principal Component Analysis
  - Supervised and nonlinear techniques (e.g., feature selection)

Principal Component Analysis (PCA)

- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space, resulting in dimensionality reduction. We find the eigenvectors of the covariance matrix, and these eigenvectors define the new space



Principal Component Analysis (Steps)

- Given *N* data vectors from *n*-dimensions, find $k \leq n$ orthogonal vectors (*principal components*) that can be best used to represent data
  - Normalize input data: Each attribute falls within the same range
  - Compute *k* orthonormal (unit) vectors, i.e., *principal components*
  - Each input data (vector) is a linear combination of the *k* principal component vectors
  - The principal components are sorted in order of decreasing "significance" or strength
  - Since the components are sorted, the size of the data can be reduced by eliminating the *weak components*, i.e., those with low variance (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)
- Works for numeric data only

Attribute Subset Selection

- Another way to reduce dimensionality of data
- Redundant attributes
  - Duplicate much or all of the information contained in one or more other attributes
  - E.g., purchase price of a product and the amount of sales tax paid
- Irrelevant attributes
  - Contain no information that is useful for the data mining task at hand
  - E.g., students' ID is often irrelevant to the task of predicting students' GPA

Heuristic Search in Attribute Selection

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

Attribute Creation (Feature Generation)

- Create new attributes (features) that can capture the important information in a data set more effectively than the original ones
- Three general methodologies
  - Attribute extraction
    - Domain-specific
  - Mapping data to new space (see: data reduction)
    - E.g., Fourier transformation, wavelet transformation, manifold approaches (not covered)
  - Attribute construction
    - Combining features (see: discriminative frequent patterns in Chapter 7)
    - Data discretization

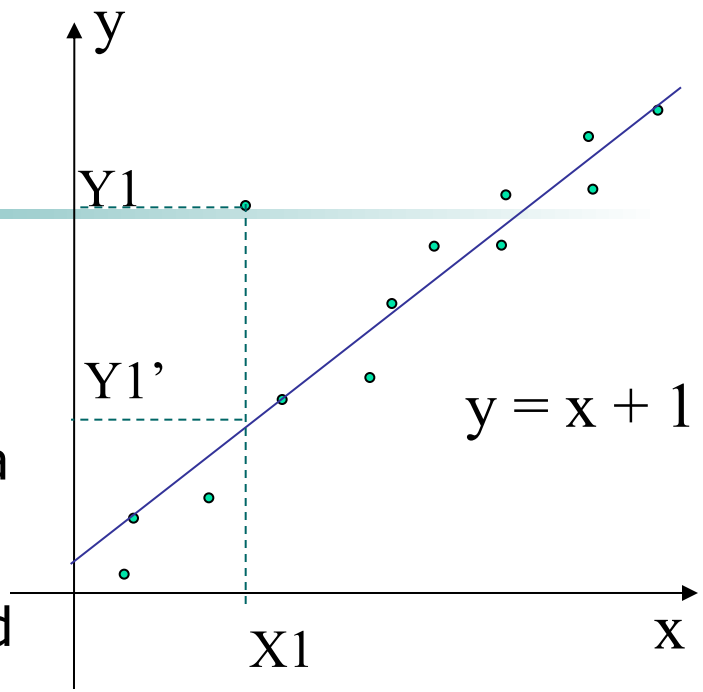
Data Reduction 2: Numerosity Reduction

- Reduce data volume by choosing alternative, *smaller forms* of data representation
- **Parametric methods** (e.g., regression)
  - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
  - Ex.: Log-linear models—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, clustering, sampling, …

Parametric Data Reduction: Regression and Log-Linear Models

- **Linear regression**
  - Data modeled to fit a straight line
  - Often uses the least-square method to fit the line
- **Multiple regression**
  - Allows a response variable Y to be modeled as a linear function of multidimensional feature vector
- **Log-linear model**
  - Approximates discrete multidimensional probability distributions

Regression Analysis



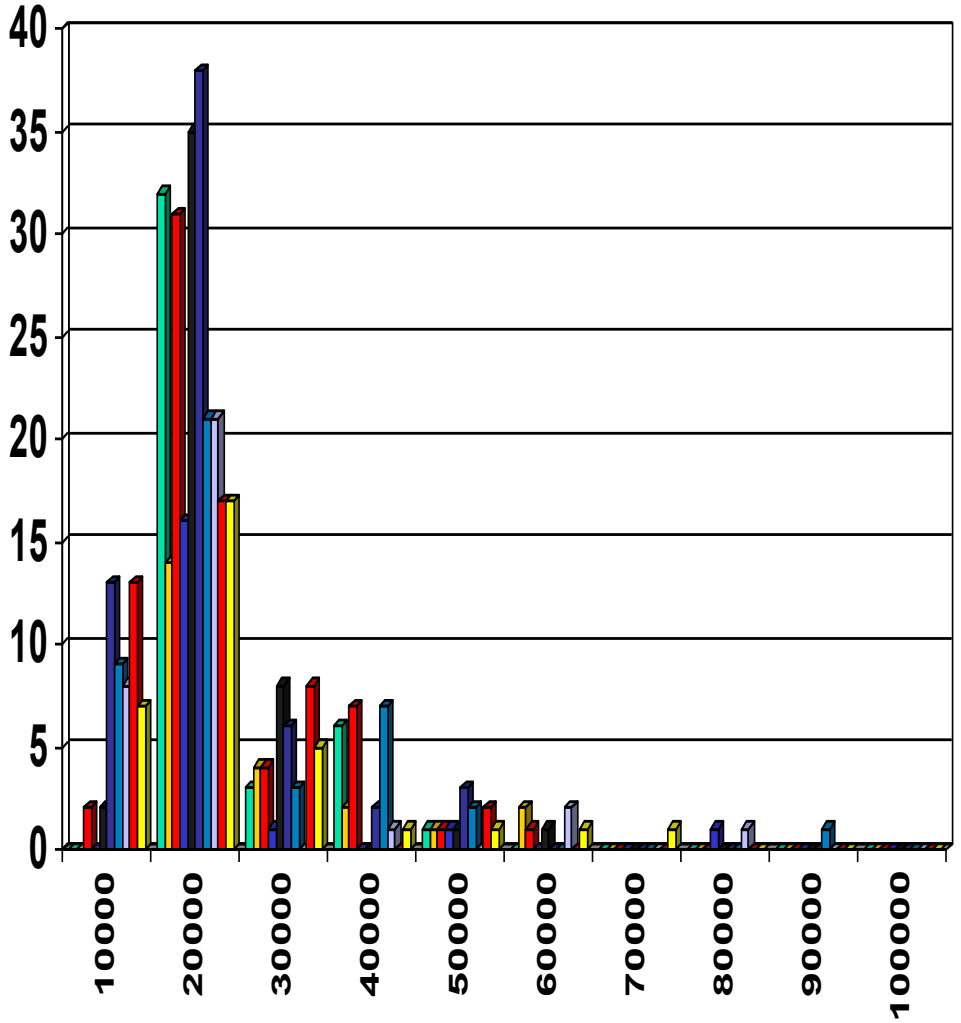
- Regression analysis: A collective name for techniques for the modeling and analysis of numerical data consisting of values of a ***dependent variable*** (also called ***response variable*** or *measurement*) and of one or more *independent variables* (aka. ***explanatory variables*** or ***predictors***)
- The parameters are estimated so as to give a "**best fit**" of the data
- Most commonly the best fit is evaluated by using the ***least squares method***, but other criteria have also been used
- Used for prediction (including forecasting of time-series data), inference, hypothesis testing, and modeling of causal relationships

Regress Analysis and Log-Linear Models

- Linear regression: $Y = w X + b$
  - Two regression coefficients, w and b, specify the line and are to be estimated by using the data at hand
  - Using the least squares criterion to the known values of Y_1, Y_2, …, X_1, X_2, ….
- Multiple regression: $Y = b_0 + b_1 X_1 + b_2 X_2$
  - Many nonlinear functions can be transformed into the above
- Log-linear models:
  - Approximate discrete multidimensional probability distributions
  - Estimate the probability of each point (tuple) in a multi-dimensional space for a set of discretized attributes, based on a smaller subset of dimensional combinations
  - Useful for dimensionality reduction and data smoothing

Histogram Analysis

- Divide data into buckets and store average (sum) for each bucket
- Partitioning rules:
  - Equal-width: equal bucket range
  - Equal-frequency (or equal-depth)



Clustering

- Partition data set into clusters based on similarity, and store cluster representation (e.g., centroid and diameter) only
- Can be very effective if data is clustered but not if data is "smeared"
- Can have hierarchical clustering and be stored in multi-dimensional index tree structures
- There are many choices of clustering definitions and clustering algorithms
- Cluster analysis will be studied in depth in Chapter 10

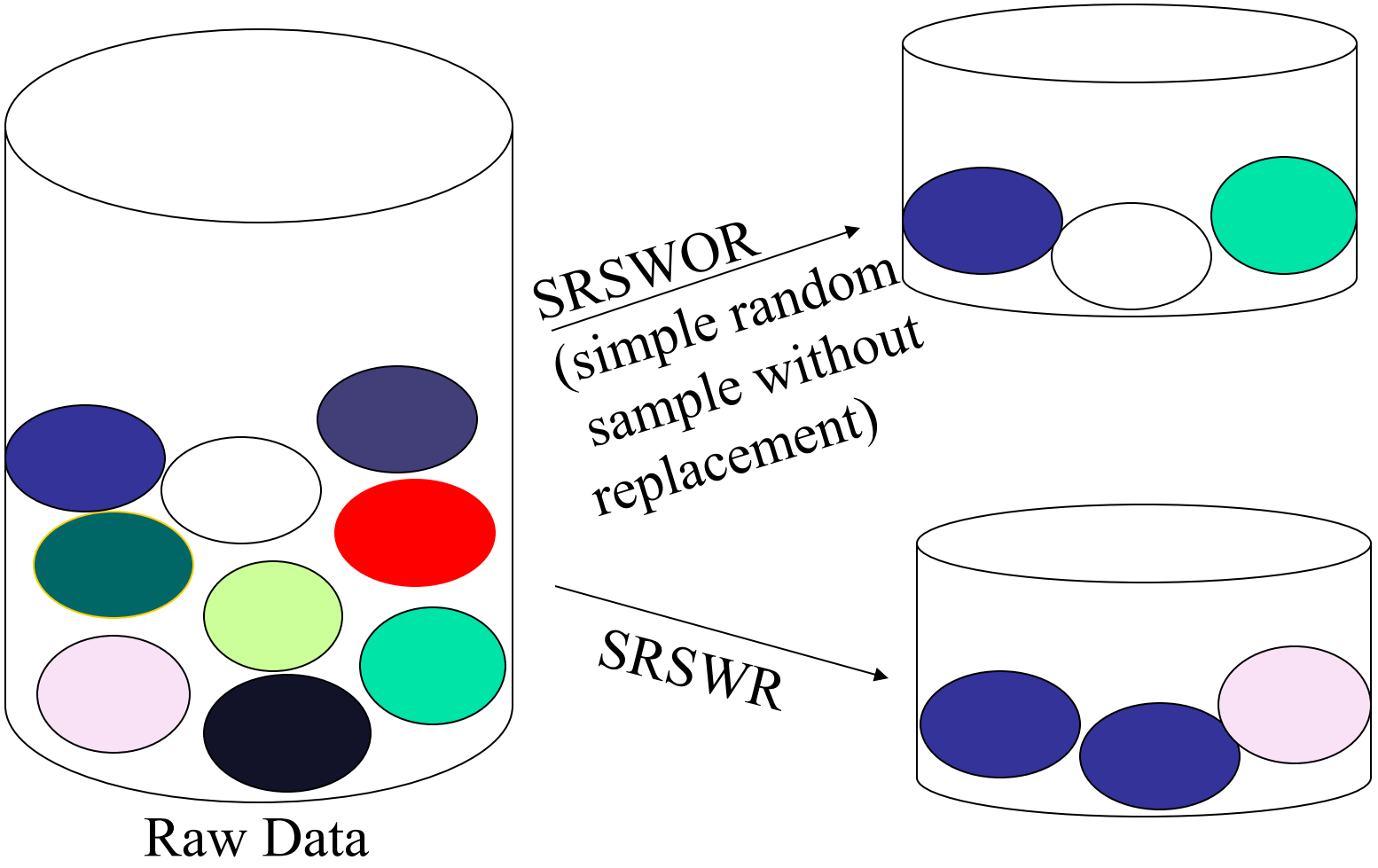
Sampling

- Sampling: obtaining a small sample *s* to represent the whole data set *N*
- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Key principle: Choose a representative subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
  - Develop adaptive sampling methods, e.g., stratified sampling:
- Note: Sampling may not reduce database I/Os (page at a time)

Types of Sampling

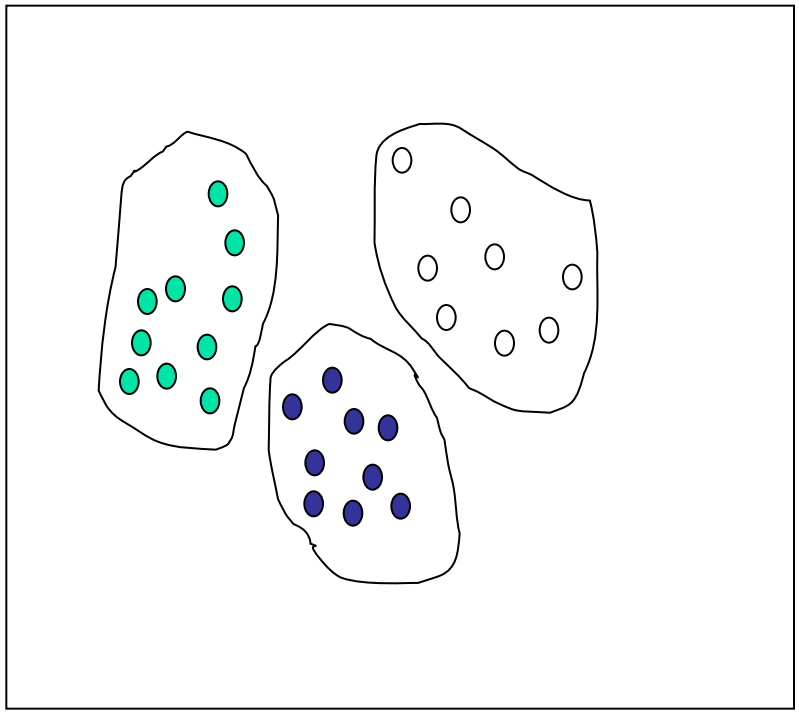
- **Simple random sampling**
  - There is an equal probability of selecting any particular item
- **Sampling without replacement**
  - Once an object is selected, it is removed from the population
- **Sampling with replacement**
  - A selected object is not removed from the population
- **Stratified sampling:**
  - Partition the data set, and draw samples from each partition (proportionally, i.e., approximately the same percentage of the data)
  - Used in conjunction with skewed data

Sampling: With or without Replacement

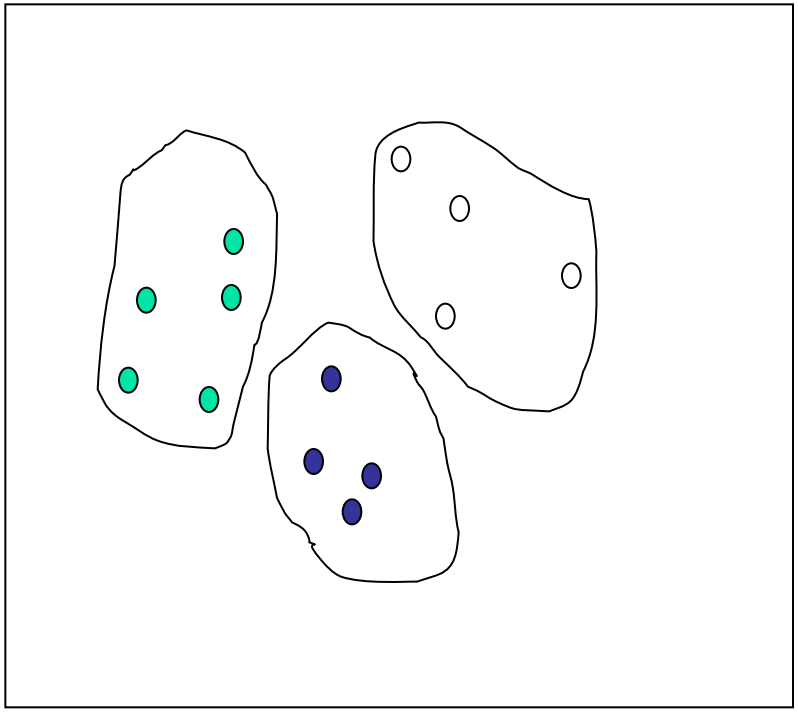


Sampling: Cluster or Stratified Sampling

Raw Data



Cluster/Stratified Sample



Data Cube Aggregation

- The lowest level of a data cube (base cuboid)
  - The aggregated data for an individual entity of interest
  - E.g., a customer in a phone calling data warehouse
- Multiple levels of aggregation in data cubes
  - Further reduce the size of data to deal with
- Reference appropriate levels
  - Use the smallest representation which is enough to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible

Data Transformation

- A function that maps the entire set of values of a given attribute to a new set of replacement values s.t. each old value can be identified with one of the new values
- Methods
  - Smoothing: Remove noise from data
  - Attribute/feature construction
    - New attributes constructed from the given ones
  - Aggregation: Summarization, data cube construction
  - Normalization: Scaled to fall within a smaller, specified range
    - min-max normalization
    - z-score normalization
    - normalization by decimal scaling
  - Discretization: Concept hierarchy climbing

Normalization

- **Min-max normalization**: to [new_min$_A$, new_max$_A$]

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

  - Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,000 is mapped to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$

- **Z-score normalization** (μ: mean, σ: standard deviation):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  - Ex. Let μ = 54,000, σ = 16,000. Then $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$

- **Normalization by decimal scaling**

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

Examples

Example: Student Test Scores

Normalizing test scores from different subjects with varying scales to a [0,1] range:

Subject	Original Score	Min	Max	Normalized Score
Math	85	45	100	0.73
Physics	42	20	50	0.73
History	78	30	90	0.80
English	65	40	80	0.63

Notice how the normalized scores allow for fair comparison across subjects with different scales.

Examples

Example: Student Heights (cm)

Normalizing student heights using z-score normalization:

Given: Mean (μ) = 170 cm, Standard Deviation (σ) = 10 cm

Student	Height (cm)	Z-Score	Interpretation
A	155	-1.5	1.5 SD below mean
B	168	-0.2	0.2 SD below mean
C	172	0.2	0.2 SD above mean
D	190	2.0	2.0 SD above mean

Note: In a normal distribution, approximately 68% of values fall within ±1 SD, 95% within ±2 SD, and 99.7% within ±3 SD from the mean.

Decimal Scaling Normalization

Example: Revenue Data (in millions)

Given data: -10, 201, 301, -401, 501, 601, 701

Step 1: Maximum absolute value = 701
Step 2: Smallest j where max(|v'|) < 1 is j = 3 ($10^3 = 1000$)
Step 3: Divide all values by $10^3 = 1000$

Original Value	Normalized Value ($v/10^3$)
-10	-0.01
201	0.201
301	0.301
-401	-0.401
501	0.501
601	0.601
701	0.701

Discretization

- Three types of attributes
  - Nominal—values from an unordered set, e.g., color, profession
  - Ordinal—values from an ordered set, e.g., military or academic rank
  - Numeric—real numbers, e.g., integer or real numbers
- Discretization: Divide the range of a continuous attribute into intervals
  - Interval labels can then be used to replace actual data values
  - Reduce data size by discretization
  - Supervised vs. unsupervised
  - Split (top-down) vs. merge (bottom-up)
  - Discretization can be performed recursively on an attribute
  - Prepare for further analysis, e.g., classification

Data Discretization Methods

- Typical methods: All the methods can be applied recursively
  - Binning
    - Top-down split, unsupervised
  - Histogram analysis
    - Top-down split, unsupervised
  - Clustering analysis (unsupervised, top-down split or bottom-up merge)
  - Decision-tree analysis (supervised, top-down split)
  - Correlation (e.g., χ^2) analysis (unsupervised, bottom-up merge)

Simple Discretization: Binning

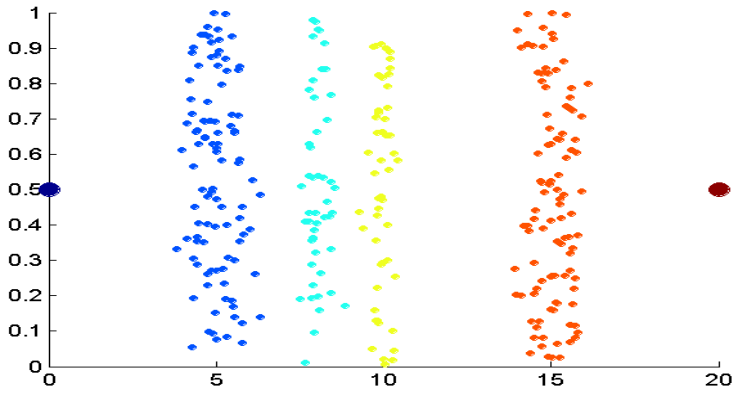
- Equal-width (distance) partitioning
  - Divides the range into *N* intervals of equal size: uniform grid
  - if *A* and *B* are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
  - The most straightforward, but outliers may dominate presentation
  - Skewed 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

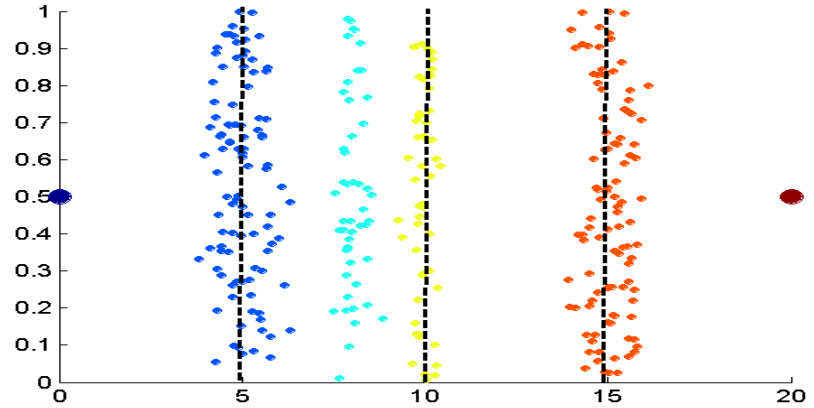
Binning Methods for Data Smoothing

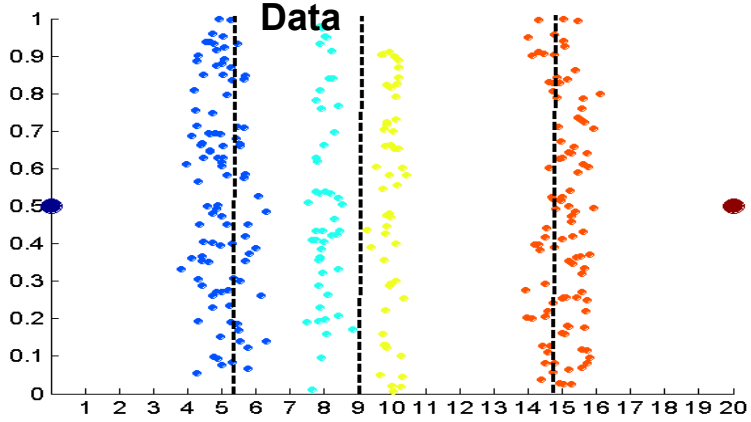
- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

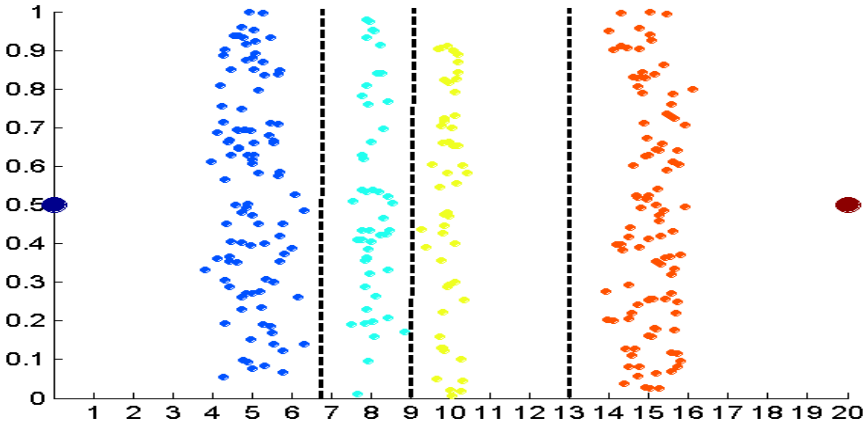
* Partition into equal-frequency (**equi-depth**) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
* Smoothing by **bin means**:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
* Smoothing by **bin boundaries**:
  - Bin 1: 4, 4, 4, 15
  - Bin 2: 21, 21, 25, 25
  - Bin 3: 26, 26, 26, 34

Discretization Without Using Class Labels (Binning vs. Clustering)









Equal frequency (binning)

K-means clustering leads to better results

Discretization by Classification & Correlation Analysis

- Classification (e.g., decision tree analysis)
  - Supervised: Given class labels, e.g., cancerous vs. benign
  - Using *entropy* to determine split point (discretization point)
  - Top-down, recursive split
- Correlation analysis (e.g., Chi-merge: χ^2-based discretization)
  - Supervised: use class information
  - Bottom-up merge: find the best neighboring intervals (those having similar distributions of classes, i.e., low χ^2 values) to merge
  - Merge performed recursively, until a predefined stopping condition

Concept Hierarchy Generation

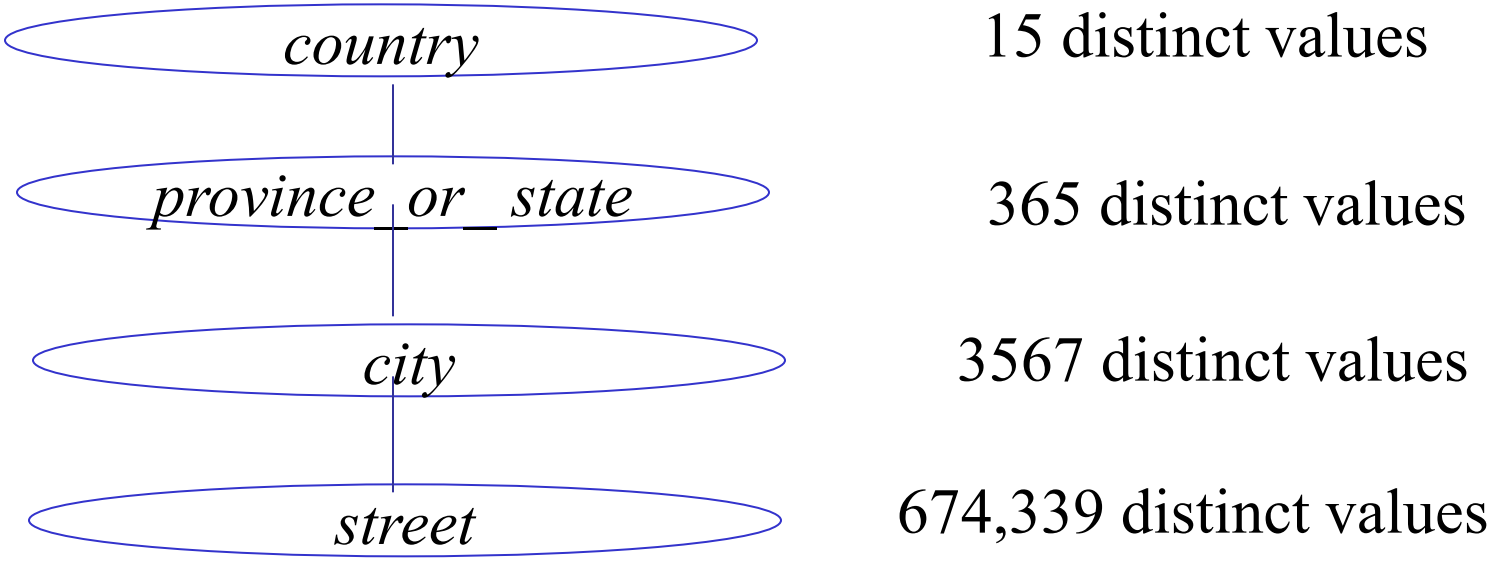
- **Concept hierarchy** organizes concepts (i.e., attribute values) hierarchically and is usually associated with each dimension in a data warehouse
- Concept hierarchies facilitate drilling and rolling in data warehouses to view data in multiple granularity
- Concept hierarchy formation: Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for *age*) by higher level concepts (such as *youth, adult*, or *senior*)
- Concept hierarchies can be explicitly specified by domain experts and/or data warehouse designers
- Concept hierarchy can be automatically formed for both numeric and nominal data. For numeric data, use discretization methods shown.

Concept Hierarchy Generation for Nominal Data

- Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
  - *street* < *city* < *state* < *country*
- Specification of a hierarchy for a set of values by explicit data grouping
  - {Urbana, Champaign, Chicago} < Illinois
- Specification of only a partial set of attributes
  - E.g., only *street* < *city*, not others
- Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
  - E.g., for a set of attributes: {*street, city, state, country*}

Automatic Concept Hierarchy Generation

- Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set
  - The attribute with the most distinct values is placed at the lowest level of the hierarchy
  - Exceptions, e.g., weekday, month, quarter, year



Summary

- **Data quality**: accuracy, completeness, consistency, timeliness, believability, interpretability
- **Data cleaning**: e.g. missing/noisy values, outliers
- **Data integration** from multiple sources:
  - Entity identification problem
  - Remove redundancies
  - Detect inconsistencies
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation