# 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?

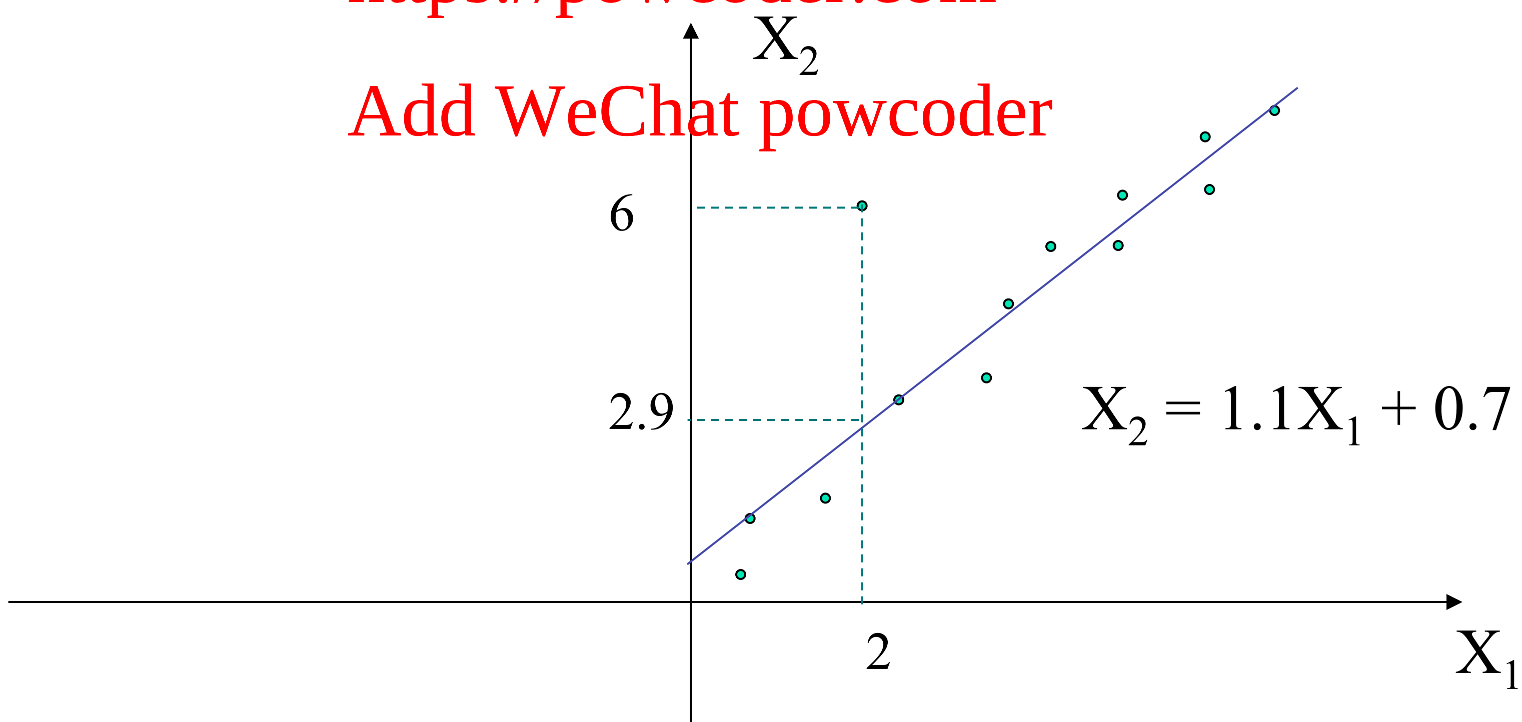
To be discussed in discretization

- 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

Also see other transformations later in the Clustering part

# 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

MinMaxScaler

- min-max normalization

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



StandardScaler; sklearn uses biased variance estimate

- z-score normalization



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



- normalization by decimal scaling

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

In scikit-learn, they are called Scaling.
Normalization means converting vectors to unit vectors.

- Data reduction

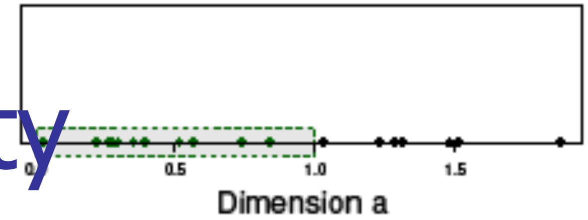
# Data Reduction Strategies

- 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

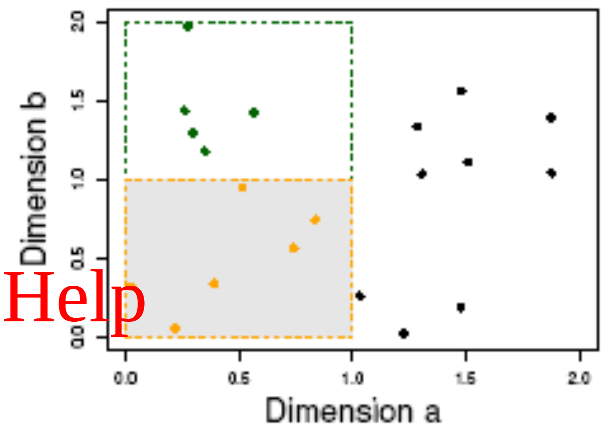
- It is common for many datasets to contain many features
  - More is better at data capturing/creation
    - 561 features for human activity recognition using smartphone dataset: https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones
    - GIST: 128 dimensional feature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

# The Curse of Dimensionality

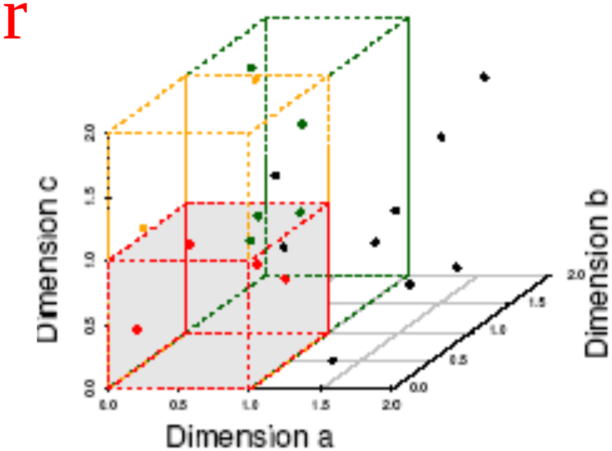



- Data in only one dimension is relatively packed
- Adding a dimension “stretches” the  points across that dimension, making them further apart
- Adding more dimensions will make the points further apart—**high dimensional data is extremely sparse ➔ hard to learn**
- **Distance measure tends to become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






(b) 6 Objects in One Unit Bin



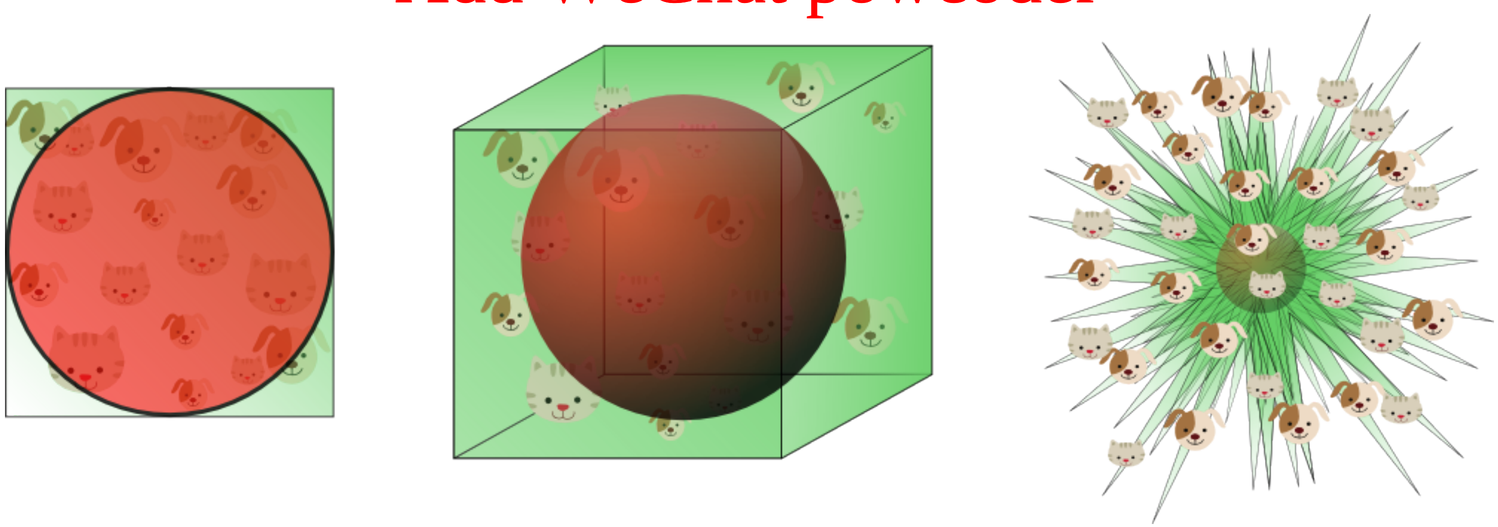

(c) 4 Objects in One Unit Bin

(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)





http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

# Goals

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

# Dimensionality reduction

- Dataset **X** consisting of **n** points in a **d**-dimensional space
- Data point $\mathbf{x}_i \in \mathbf{R}^d$ (**d**-dimensional real vector):

$$\mathbf{x}_i = [\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots, \mathbf{x}_{id}]^T$$



- 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, ...



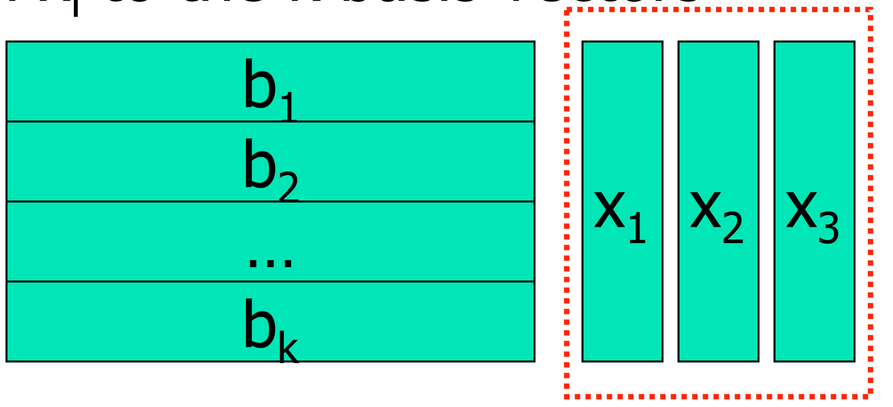
  - Step-wise feature elimination:
    - Repeatedly eliminate the worst feature
  - Best combined feature selection and elimination:
  - Optimal branch and bound:
    - Use feature elimination and backtracking

# Principal Component Analysis (**PCA**)

- Original dataset: *N* *d*-dimensional vectors $X = \{x_i\}_{i=1..n}$
  - Find $k \le d$ **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)
  - $x_i' = [b_1 \ldots b_k]^T x_i$
  - $X' = [b_1 \ldots b_k]^T X$ (en masse)

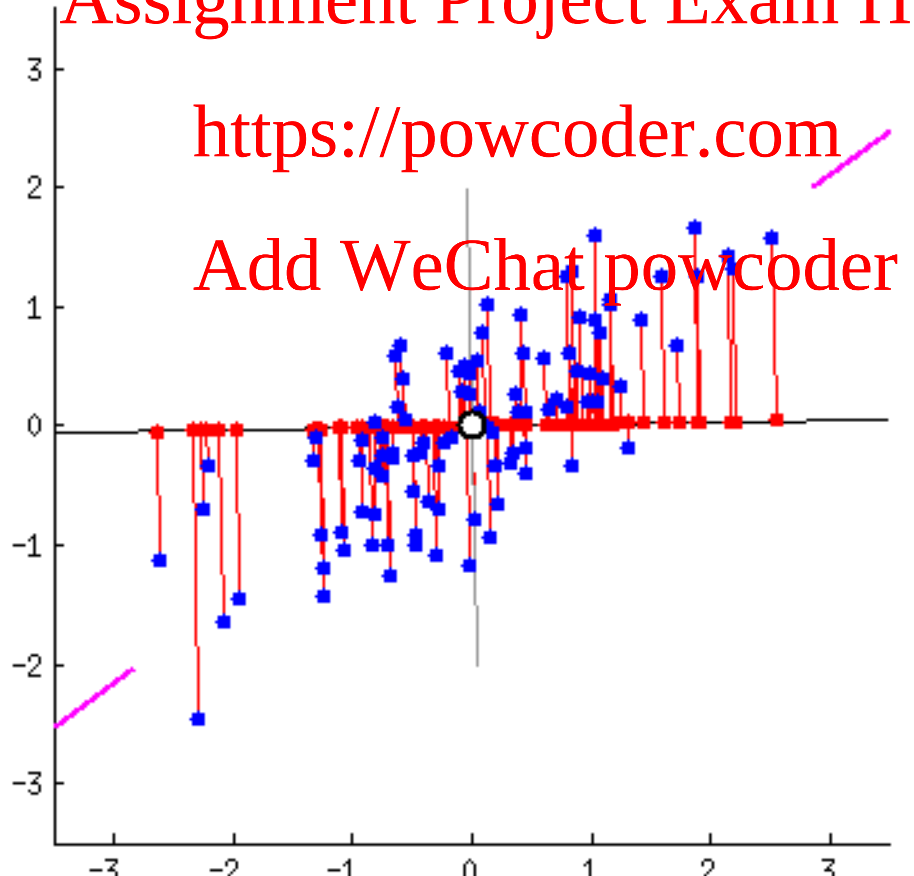



- Closed related to Singular Vector Decomposition (**SVD**)

# 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 $R^d$ there exists $F: R^d \rightarrow R^k$ 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 ε in (0, ½], δ > 0, there is a random linear mapping F: $R^d \rightarrow R^k$ with $k = O(\varepsilon^{-2} \log \delta^{-1})$ such that for any unit vector x in Rd,



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

- Take $\delta = n^{-2}$, so $k = O(\varepsilon^{-2} \log(n))$, and then for 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 ½.
  - 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.

$$F(x) = \begin{bmatrix} U_{*1} \\ U_{*1} \\ \cdots \\ U_{*k} \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} y_1 & y_2 & \ldots & y_k \end{bmatrix}$$



**Quick proof:**

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

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

**Concentration bound of chisquared distribution:**

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

# Approximating Inner Product

- 20 news groups
- Origin dim: 5000

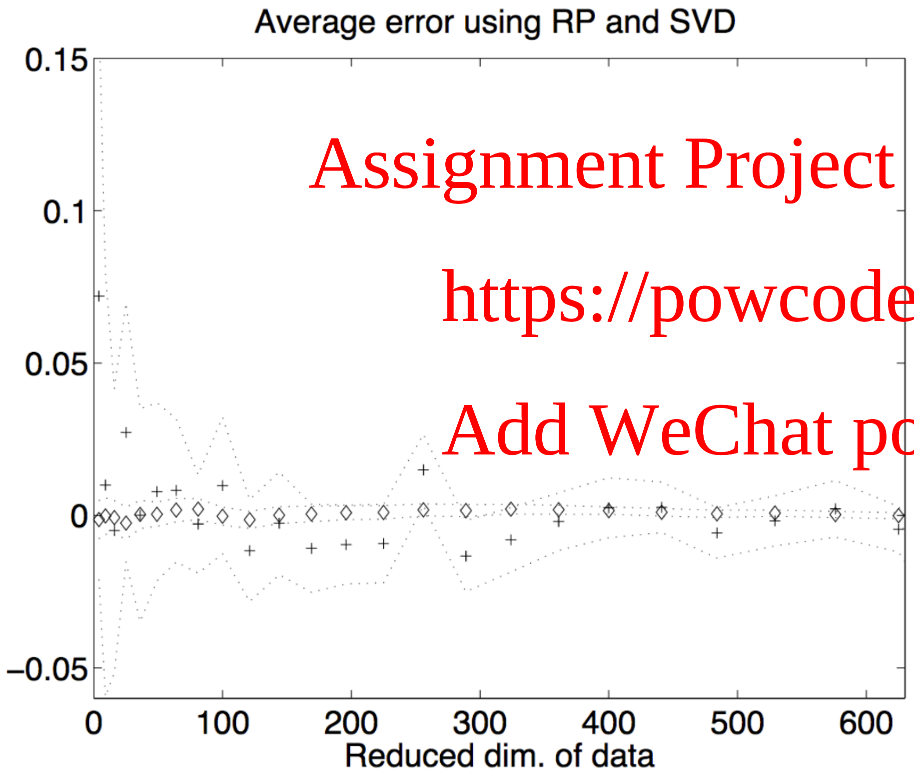





**Figure 4: The error produced by RP (+) and SVD ($\diamond$) on text document data, with 95% confidence intervals over 100 pairs of document vectors.**

# Non-linear Dimensionality Reduction

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



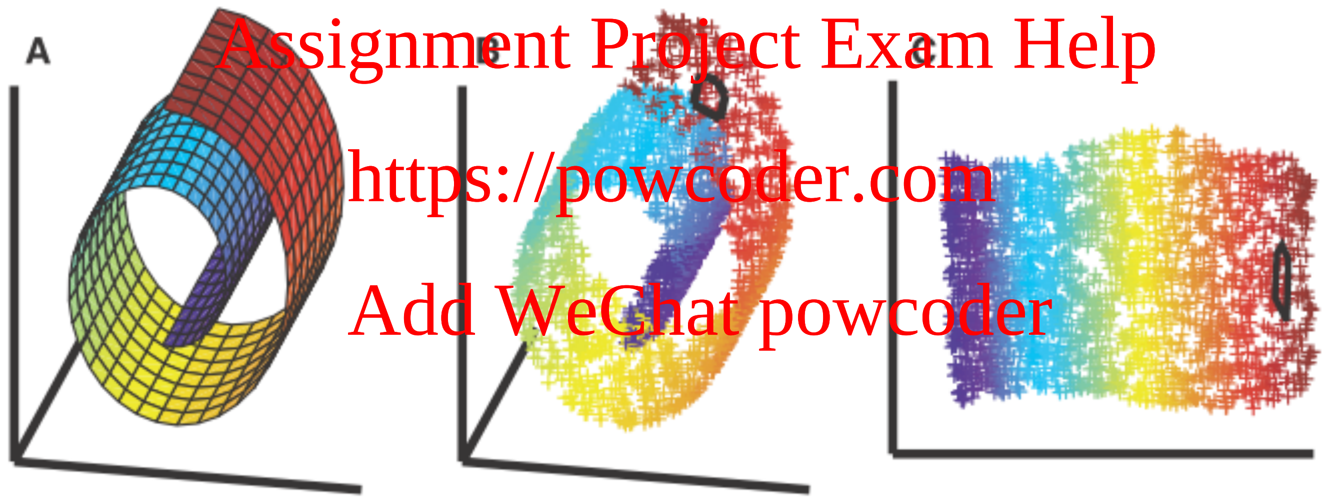
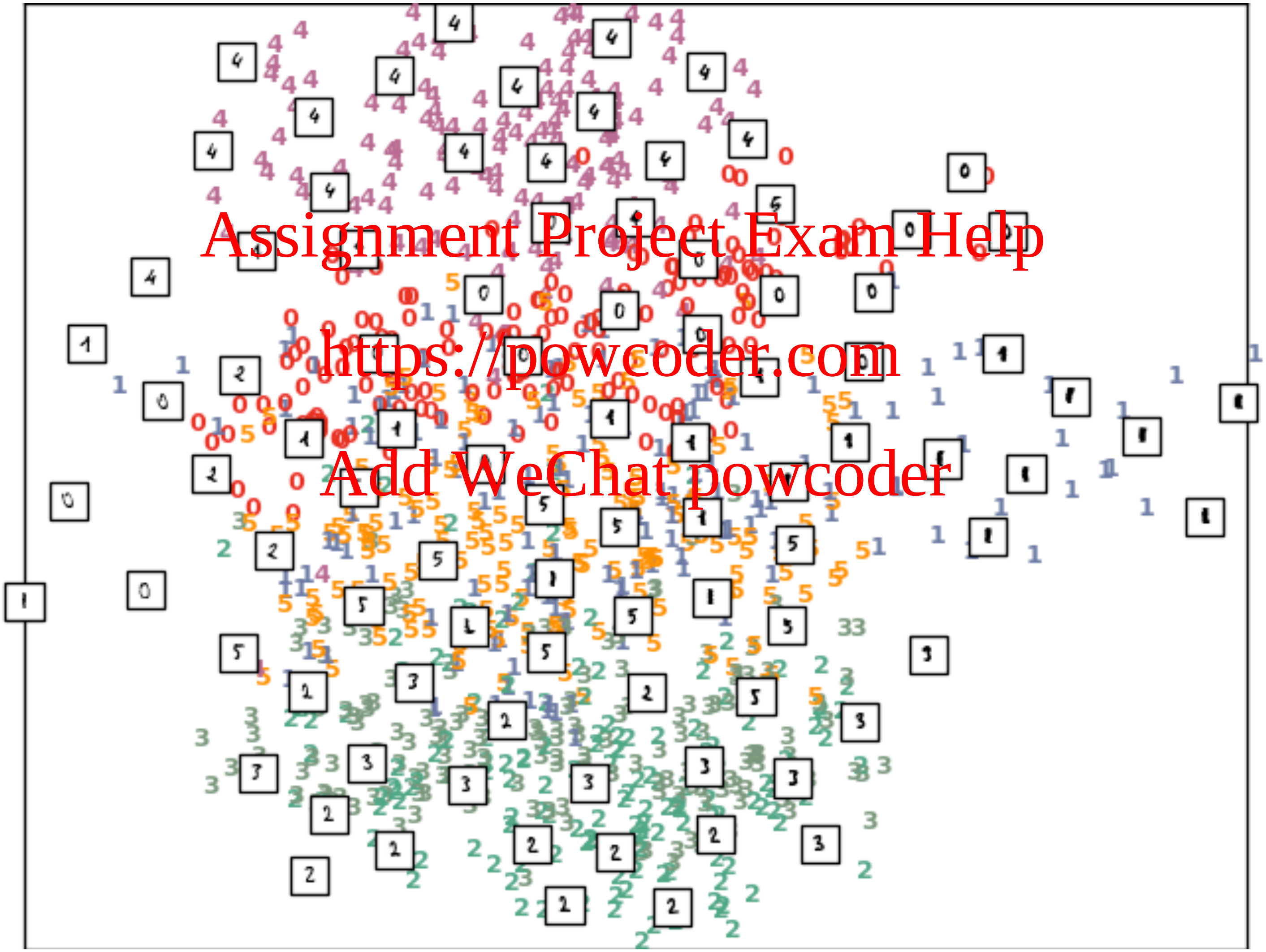


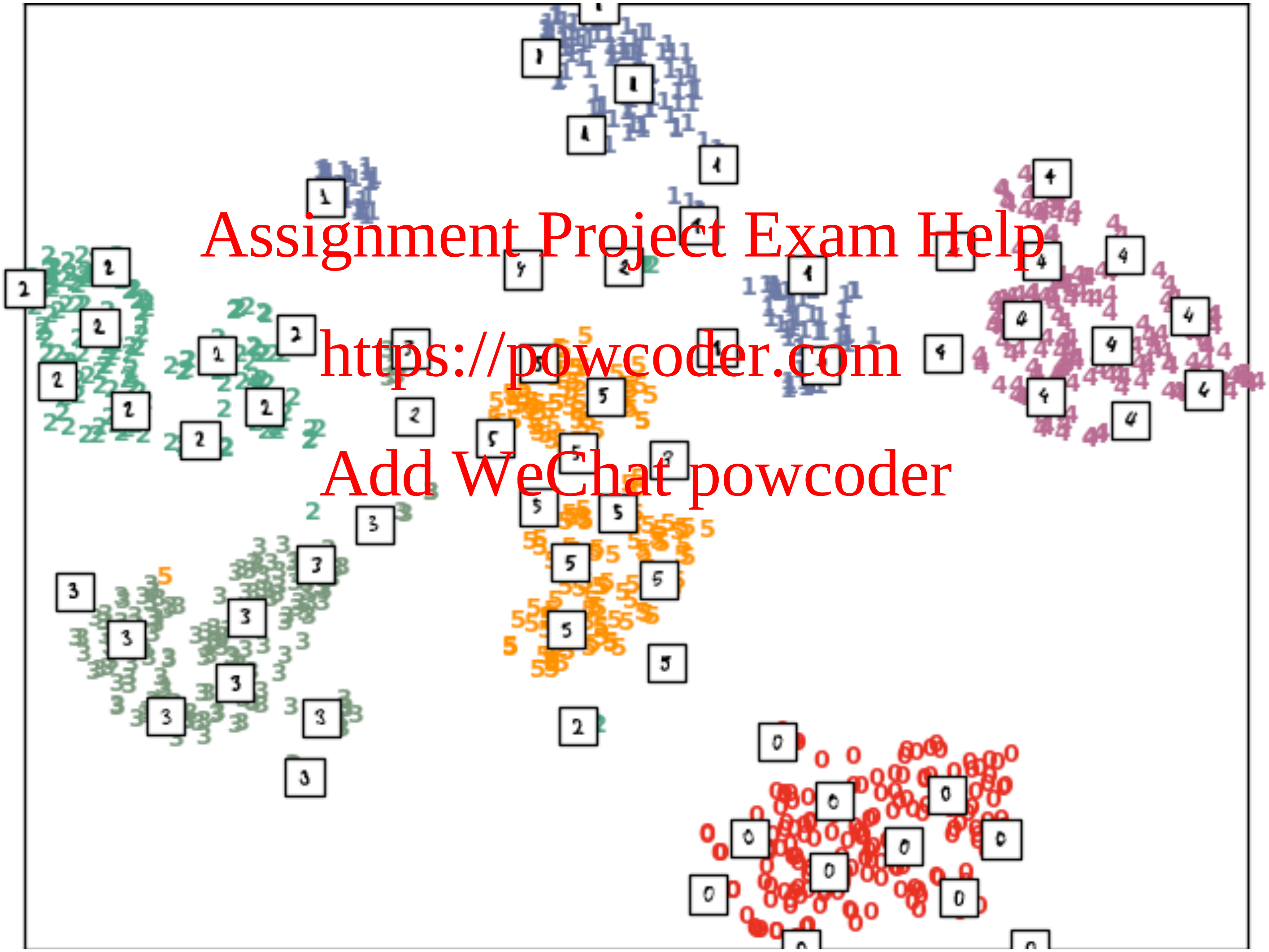

**Fig. 1.** The problem of nonlinear dimensionality reduction, as illustrated (*10*) for three-dimensional data (B) sampled from a two-dimensional manifold (**A**). An unsupervised learning algorithm must discover the global internal coordinates of the manifold without signals that explicitly indicate 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)

# 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

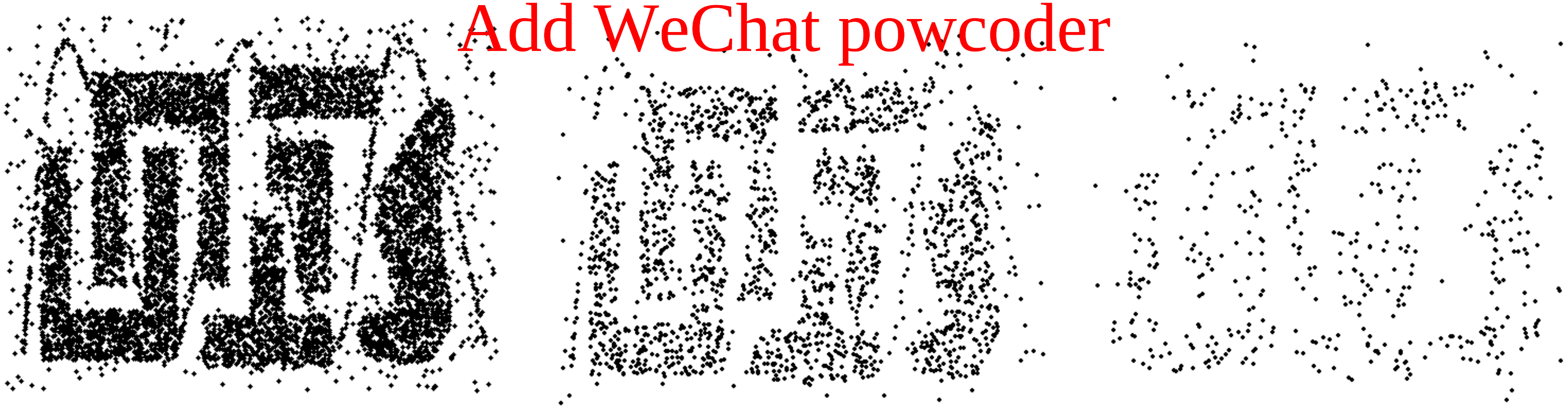
# Numerosity Reduction

- 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
- cost(bin) = SSE($[x_1, \ldots, x_m]$) = $\sum_{i=1}^{m} (x_i - E(x_i))^2$
- cost of B bins = sum(cost($bin_1$), ..., cost($bin_B$))



- Problem: find the B-1 bin boundaries such that the cost of the resulting bins is minimized



  - Alg( $\{x_1, \ldots, x_n\}$, 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 **sorted**($\{x_1, \ldots, x_n\}$) if necessary

For simplicity, we use integers, rather than real numbers, for the examples here.

# Recursive Formulation

- Observation

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



- Example
  - n=4, B=3

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





$(7,9,13,5)_3$

2

$(7)_1+(9,13,5)_2$

$0 + 8$

$(7,9)_1+(13,5)_2$

$(1^2+1^2) + 0$

$(7)_1+(9)_1+(13,5)_1$

$0 + (4^2+4^2)$

$(7)_1+(9,13)_1+(5)_1$

$(2^2+2^2) + 0$

$(7,9)_1+(13)_1+(5)_1$

$0 + 0$

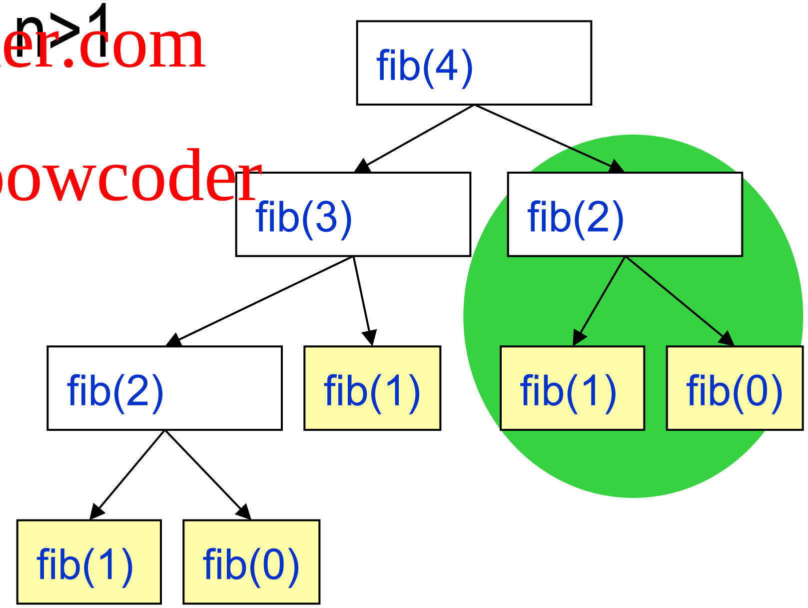
what about $(7,9,\ 13)_1+(5)_2$

# Problem Caused by Overlapping Sub-problems

- Consider calculating Fibonacci function
  - fib(0)=0
  - fib(1)=1
  - **fib(n) = fib(n-1) + fib(n-2)** for n>1
- Naïve D&C implementation is in efficient

Boundary Condition



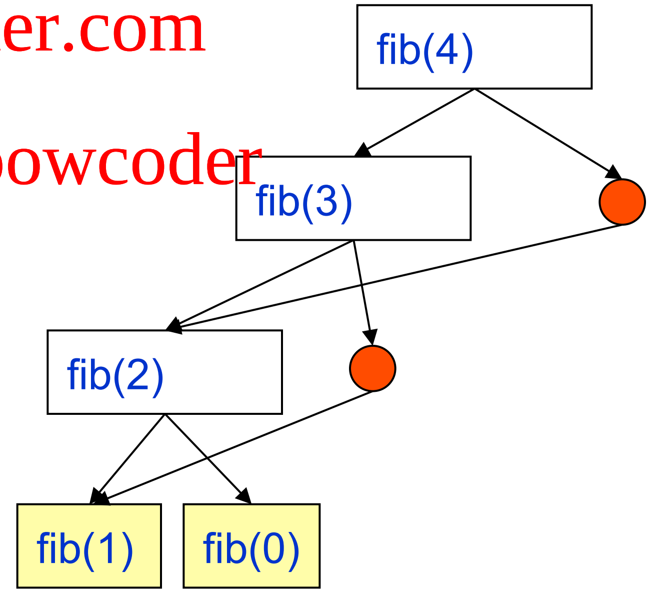


# Memoization

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



| Sub-problem | solution |
|---|---|
| fib(4) | 3 |
| fib(3) | 2 |
| fib(2) | 1 |
| fib(1) | 1 |
| fib(0) | 0 |

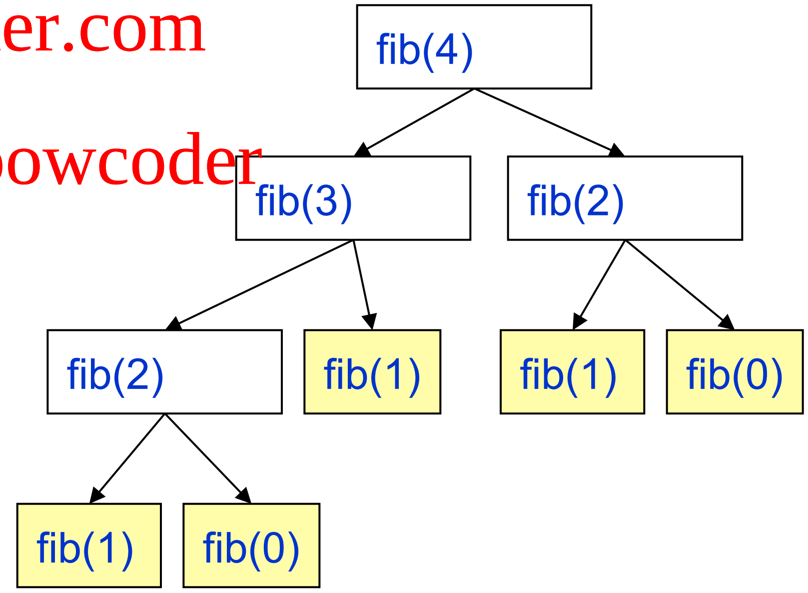


# Dynamic Programming

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



| Sub-problem | solution |
|---|---|
| | |
| | |

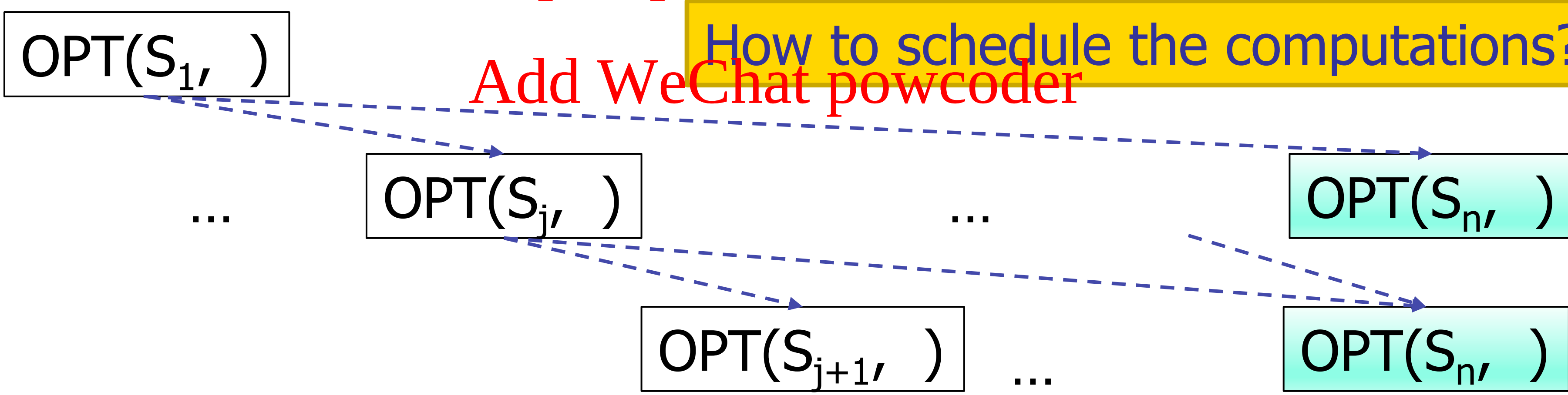


# 2D Dynamic Programming

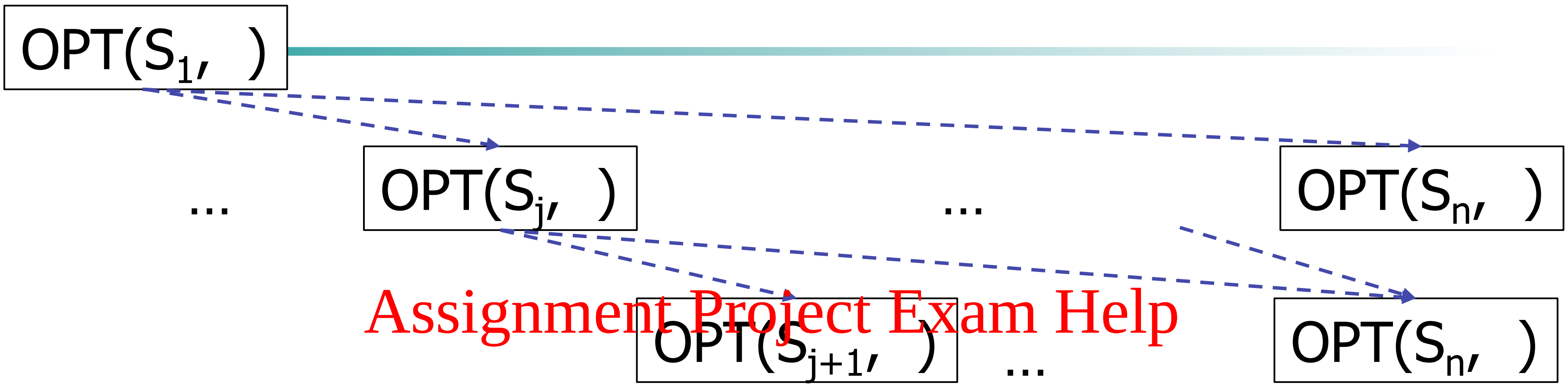
- **OPT**$(x[1..\ n], B) = \min_{i \text{ in } [n]}\{SSE(x[1\ ..\ i]) +$ **OPT**$(x[i+1\ ..\ n], B-1)\}$
- **OPT**$(S_1, B) = \min_{i \text{ in } [e]}\{\ldots + $ **OPT**$(S_{i+1}, B-1)\}$
- Goal:



How to schedule the computations?



# Pseudocode



# Example

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

| B | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|
| 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

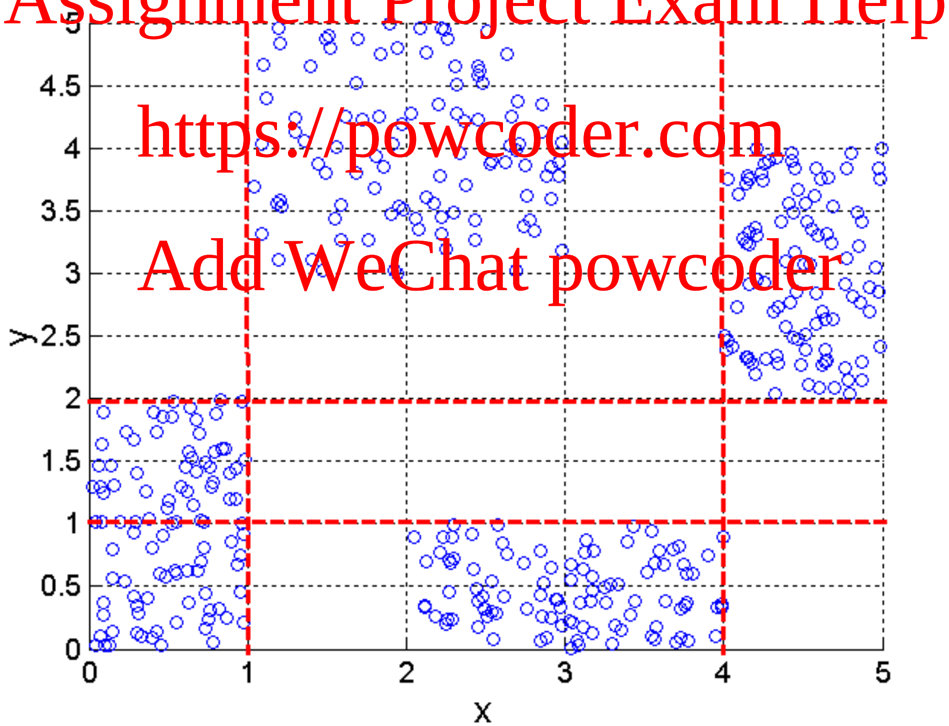
| x[1] | x[2] | X[3] | x[4] |
|---|---|---|---|
| 7 | 9 | 13 | 5 |

| gap(1-2) | gap(2-3) | gap(3-4) |
|---|---|---|
| 2 | 4 | 8 |

$(7,9)_1 \quad (13)_1 \quad (5)_1$

# Discretization via Clustering

- Can consider multiple attributes together




**An example where univariate discretization does not work well**

# Supervised Discretization Methods

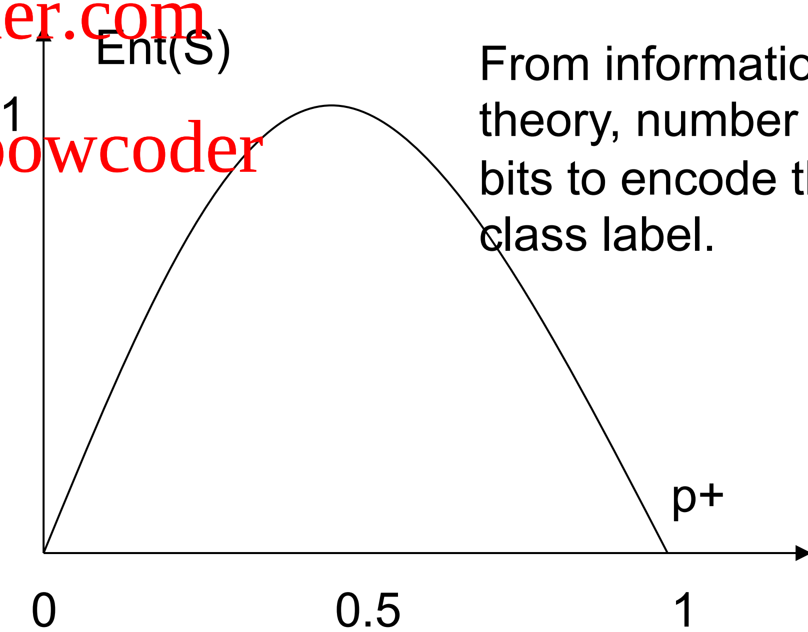
- MDLPC [Fayyad & Irani, 1993]

# Entropy measures uncertainty

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

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

- Consider drawing a random sample from S. What can you tell about its label?
  - If Ent(S) = 0:
  - If Ent(S) = log(m):




From information theory, number of bits to encode the class label.

Can be generalized to n-ary split

# 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?
  - Assignment Project Exam Help

    https://powcoder.com
  - Add WeChat powcoder
- 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?

These functions are also used in Decision Trees.

# 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



$$E(T;S) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2)$$



- 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.,

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

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

Not needed. Just for the sake of completeness.

# 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?



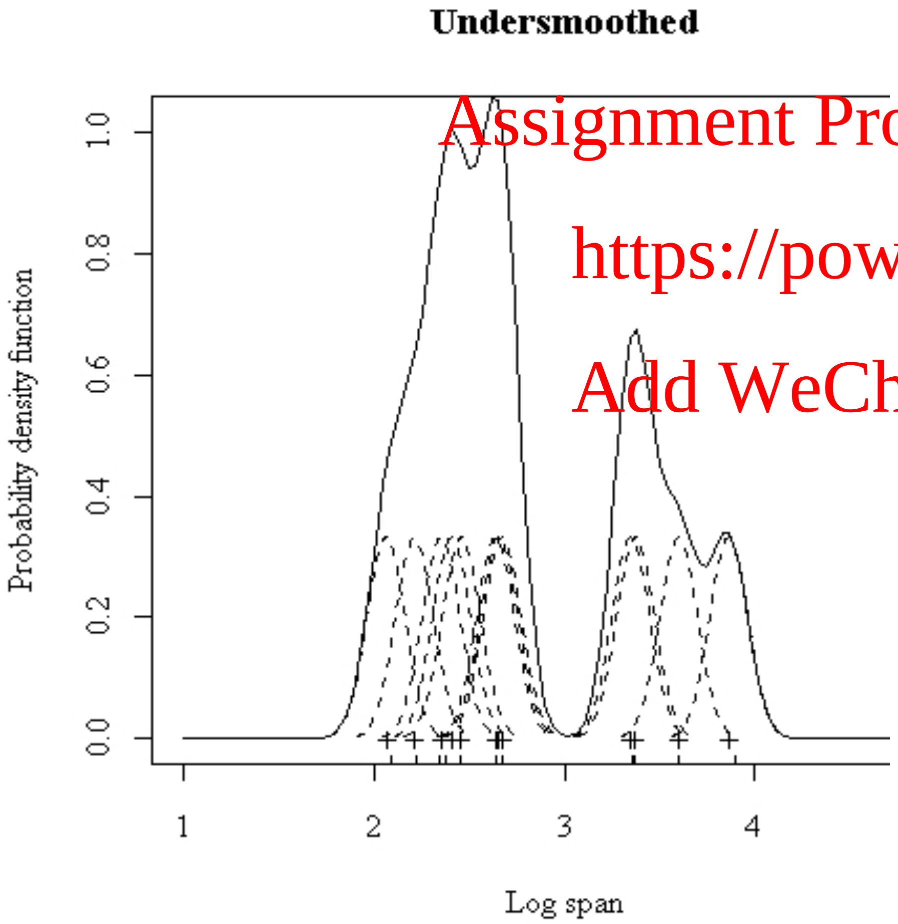
- Naive estimation
  - P(x = 3) = 4/7    P(x = 4) = 0 / 7
- Assume x follows the Poisson distribution
  - MLE estimation of $\lambda$
  - MAP estimation with a prior (typically Gamma)

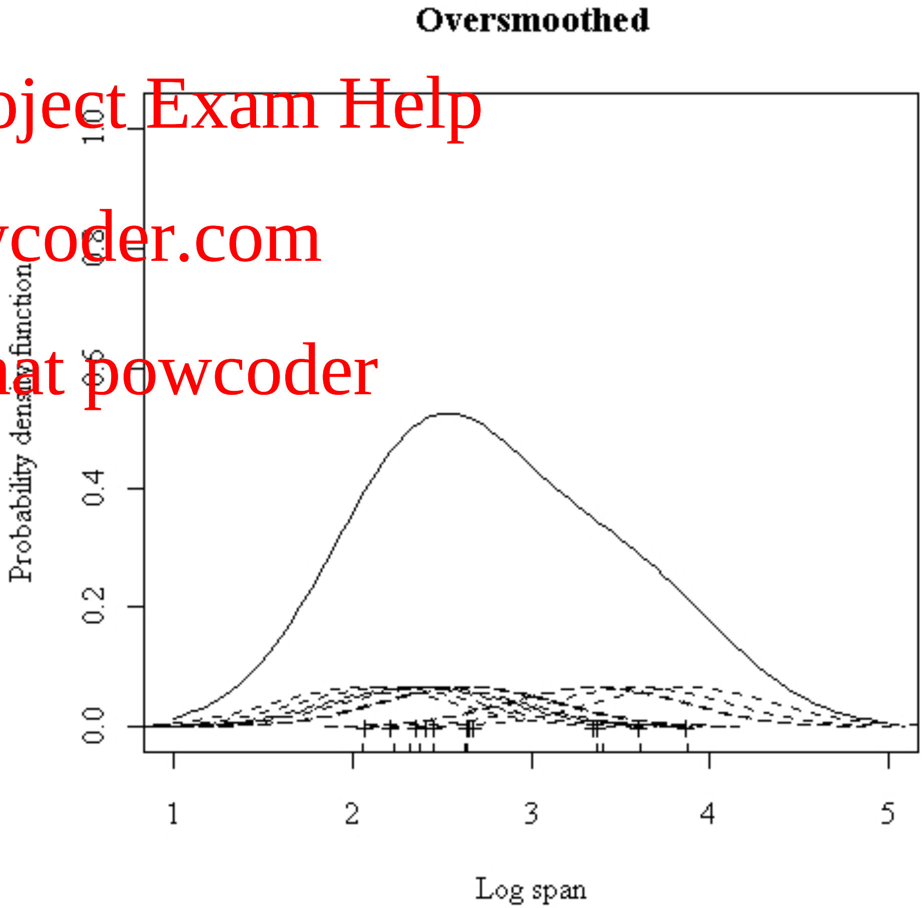
$$P(x;\lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$$

# 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


Undersmoothed
Probability density function
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
Log span


Oversmoothed
Probability density function
0.0
0.2
0.4
1
2
3
4
5
Log span

# 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 !=1
    - $v_j = 1$, otherwise
  - scikit-learn: OneHotEncoder
- Disadvantages:
  - Ignores similarity between values
- Embedding-based methods can learn better (real) vectors

- Case Study
  - c.f., TUN_datacleansing.ppt