# CSE 590
# Data Science Fundamentals

# Data Preparation And reduction I

Klaus Mueller

Computer Science Department
Stony Brook University and SUNY Korea

| Lecture | Topic | Projects |
|---|---|---|
| 1 | Intro, schedule, and logistics | |
| 2 | Data Science components and tasks | |
| 3 | Data types | Project #1 out |
| 4 | Introduction to R, statistics foundations | |
| 5 | Introduction to D3, visual analytics | |
| 6 | Data preparation and reduction | |
| 7 | Data preparation and reduction | Project #1 due |
| 8 | Similarity and distances | Project #2 out |
| 9 | Similarity and distances | |
| 10 | Cluster analysis | |
| 11 | Cluster analysis | |
| 12 | Pattern miming | Project #2 due |
| 13 | Pattern mining | |
| 14 | Outlier analysis | |
| 15 | Outlier analysis | Final Project proposal due |
| 16 | Classifiers | |
| 17 | Midterm | |
| 18 | Classifiers | |
| 19 | Optimization and model fitting | |
| 20 | Optimization and model fitting | |
| 21 | Causal modeling | |
| 22 | Streaming data | Final Project preliminary report due |
| 23 | Text data | |
| 24 | Time series data | |
| 25 | Graph data | |
| 26 | Scalability and data engineering | |
| 27 | Data journalism | |
| | Final project presentation | Final Project slides and final report due |

# Data in the Real World is Dirty

Incomplete

- can lack attribute values
- can lack certain attributes of interest
- may contain only aggregate data

Noisy

- has errors or outliers

Inconsistent

- may have discrepancies in codes or names

No quality data, no quality mining results!

# Data Preparation Tasks

Data cleaning

- fill in **missing values**
- smooth **noisy data**
- identify or **remove outliers**
- **resolve inconsistencies**

Data reduction

- **obtain reduced volume**, but get same/similar analytical results
- data discretization (for numerical data)
- data aggregation (summarization)
- **data transformation/normalization**
- **dimensionality reduction**
- data compression/generalization

# Data Integration

Data integration/fusion

- multiple databases
- data cubes
- files
- notes

Produces new opportunities

- can gain more comprehensive insight (value > sum of parts)
- but watch out for *synonymy and polysemy*
- attributes with different labels may have the same meaning
  – “comical” and “hilarious”
- attributes with the same label may have different meaning
  – “jaguar” can be a cat or a car

# Missing Values

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
- other reasons

# Missing Values – What To Do

Ignore the tuple

- usually done when class label is missing

Fill in the missing value manually

- can be tedious and even infeasible

Use a global constant to fill in the missing value:

- e.g., “unknown” or a new class

Use the attribute mean to fill in the missing value

- better: only use samples of the same class for better fit

Use the most probable value to fill in the missing value:

- inference-based: regression, Bayesian formula, decision tree

# Missing Data – Example

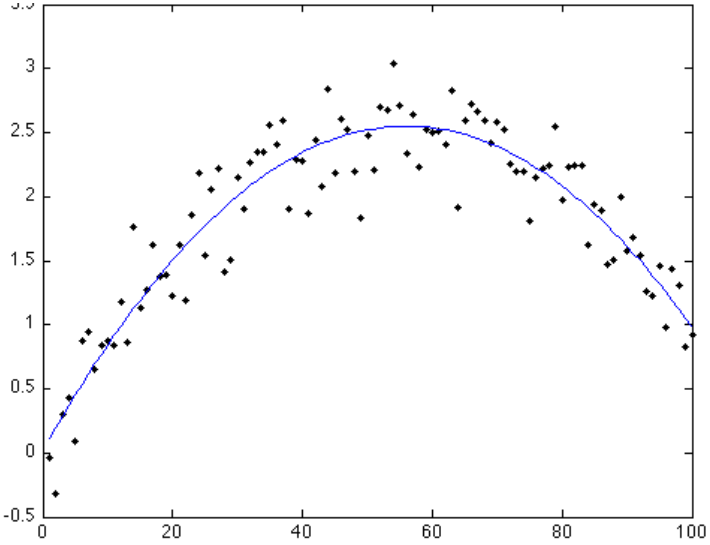
Fill missing values using aggregate functions

- average
- probabilistic estimates on global value distribution

| Age | Income | Team | Gender |
|---|---|---|---|
| 23 | 24,200 | Red Sox | M |
| 39 | $?_1$ | Yankees | F |
| 45 | 45,390 | $?_2$ | F |

- $?_1$: put the average income, or put the most probable income based on the fact that the person is 39 years old
- $?_2$: put the most frequent team

# Noisy Data




Noise = Random error in a measured variable

- faulty data collection instruments
- data entry problems
- data transmission problems
- technology limitation
- inconsistency in naming convention

Other data problems which require data cleaning

- duplicate records
- incomplete data
- inconsistent data

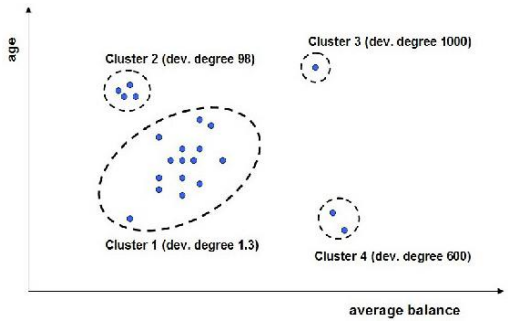
# Noisy Data – What To Do

## Binning method

- first sort data and partition into (equi-depth) bins
- then smooth by bin means,  bin median, bin boundaries, etc.

## Clustering

- detect and remove outliers

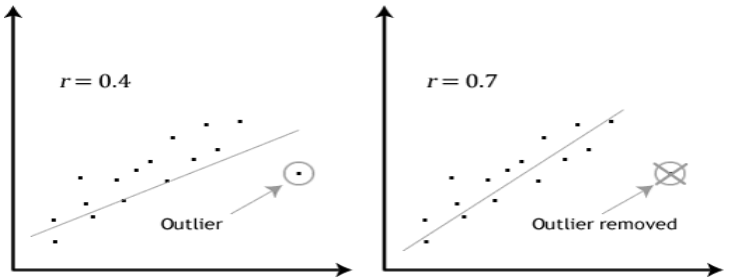



## Semi-automated method

- combined computer and human inspection
- detect suspicious values and check manually (need visualization)

## Regression

- smooth by fitting the data to a regression function



# Example For Binning

Sorted data by price ($): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

Partition into (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

# Noise removal

A word of caution

- an outlier may not be noise
- it may be an anomaly that is very valuable
- look for noise statistics
- outlier from the noise statistics may be important data

# Resolve Inconsistencies

Inconsistencies in naming conventions or data codes

- e.g., 2/5/2002 could be 2 May 2002 or 5 Feb 2002

Redundant data

- duplicate tuples, which were received twice should be removed

# Data Transformation

Can help reduce influence of extreme values

Variance reduction:

- often very useful when dealing with skewed data (e.g. incomes)
- square root, reciprocal, logarithm, raising to a power
- Logit: transforms probabilities from 0 to 1 to real-line

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

# Data Normalization

Sometimes we like to have all variables on the same scale

- min-max normalization

$$v' = \frac{v - min}{max - min}$$

- standardization / z-score normalization

$$v' = \frac{v - \overline{v}}{\sigma_v}$$

- standardization is less sensitive to outliers

# The Need for Data Reduction

Purpose

- reduce the data to a size that can be feasibly stored
- reduce the data so a mining algorithm can be feasibly run

Alternatives

- buy more storage
- buy more computers or faster ones
- develop more efficient algorithms (look beyond O-notation)

In practice, all of this is happening at the same time

- but the growth of data and complexities is faster
- and so data reduction is important

# Data Reduction

Sampling

- random, stratified, Monte Carlo, importance
- redundancy sampling
- reservoir sampling for streaming data


The CURE algorithm

- well-scattered points

Data summarization

- binning (already discussed)
- clustering (see future a lecture)
- dimension reduction (see next lecture)

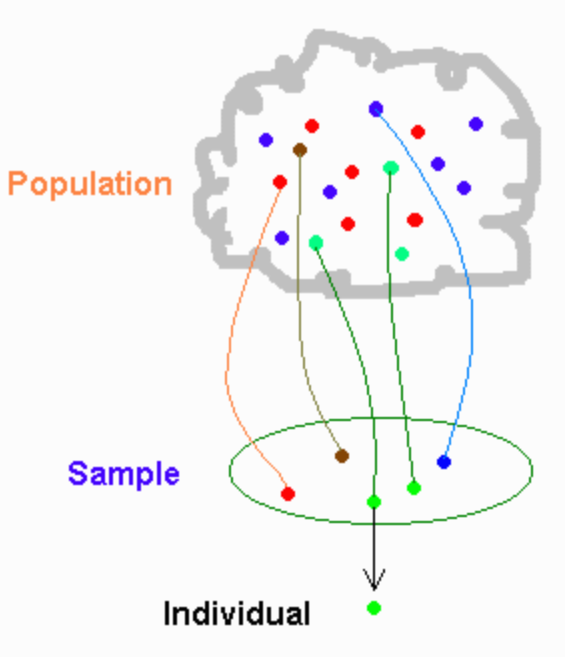
# Sampling

The goal

- pick a representative subset of the data

Random sampling

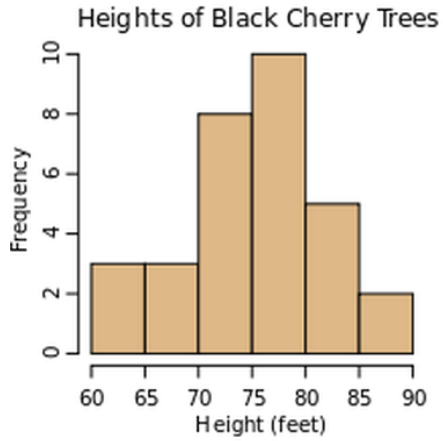
- pick sample points at random
- will work if the points are distributed uniformly
- this is usually not the case
- outliers will likely be missed
- so the sample will not be representative

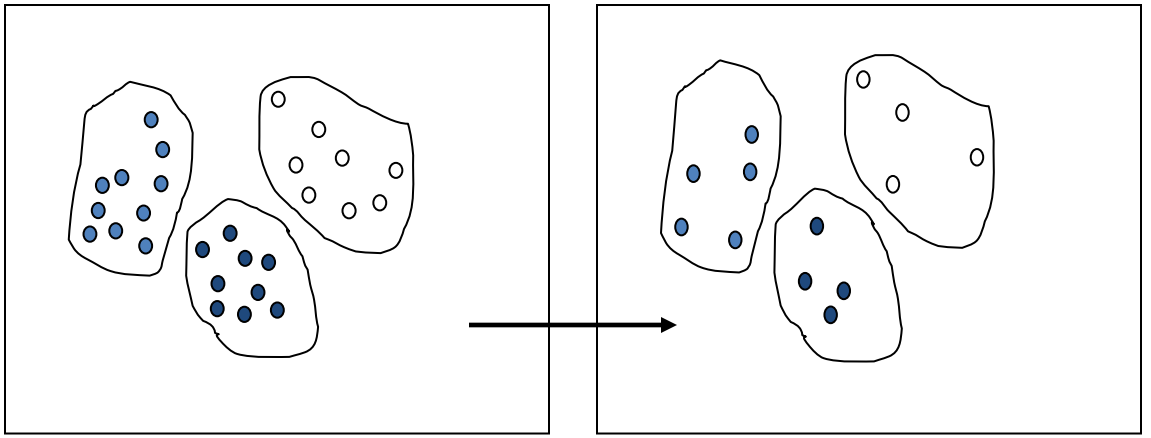


# Adaptive Sampling

Pick the samples according to some knowledge of the data distribution

- create a binning of some sort (outliers will form bins as well)
- also called *strata* (stratified sampling)
- the size of each bin represents its percentage in the population
- it guides the number of samples – bigger bins get more samples



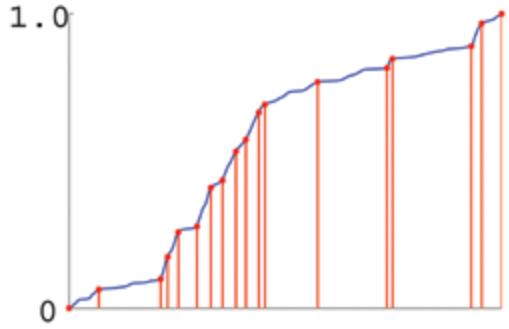
sampling rate ~ bin height

sampling rate ~ cluster size

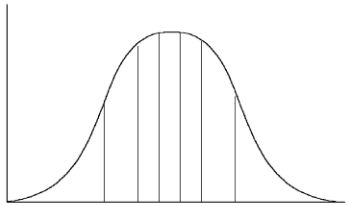
# Importance Sampling

Estimate the statistical properties of a distribution

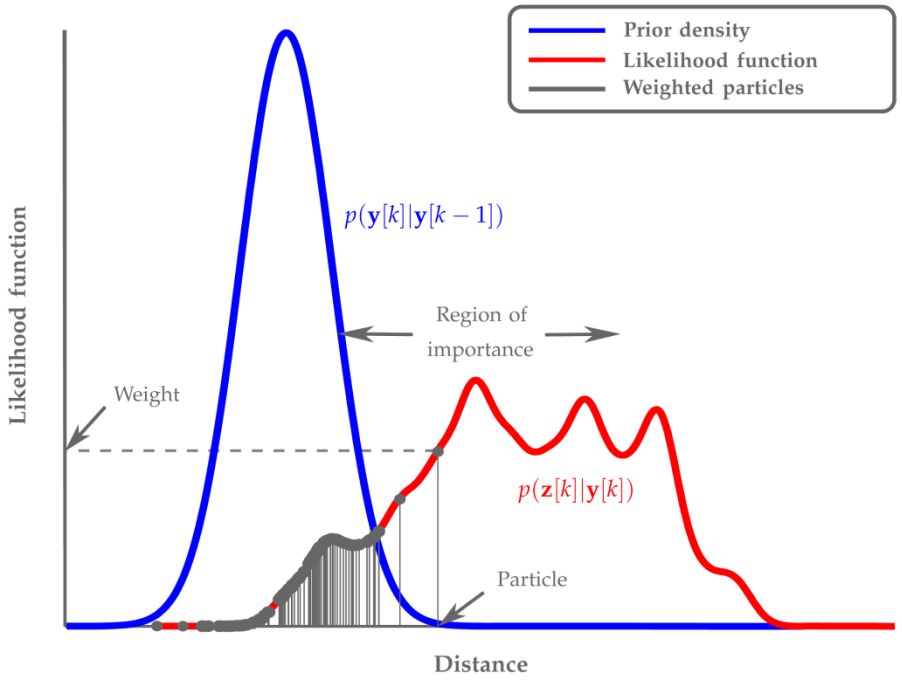
- then sample the distribution according to this distribution
- define the importance




sample in high slopes



sample in high densities




sample according to a user-defined function

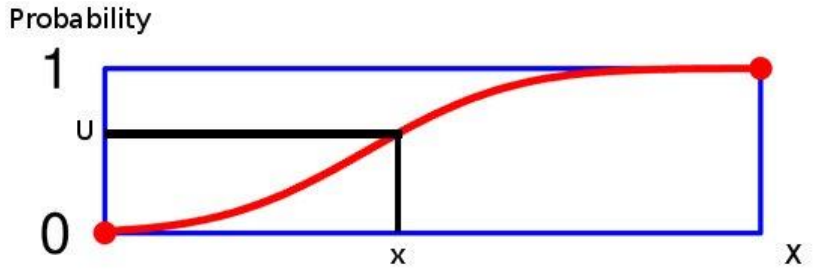
# The Inversion Method

Easy way to making your own sampling algorithm

- find the cumulative distribution function (CDF) of your desired probability density function (PDF)

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

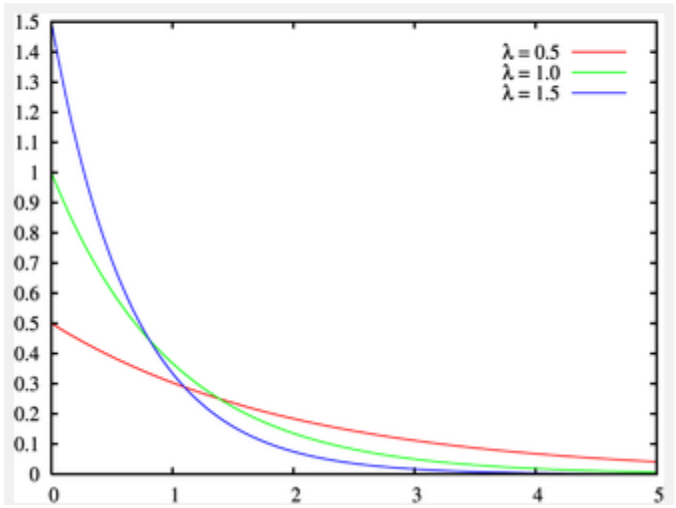
- if f(x) has an inverse then we can use the inversion method to create a sampling method




- generate a random u-value between [0.1] and look up the x-value
- region with higher f(x) have a steeper CDF and get sampled more

# Inversion Method Example

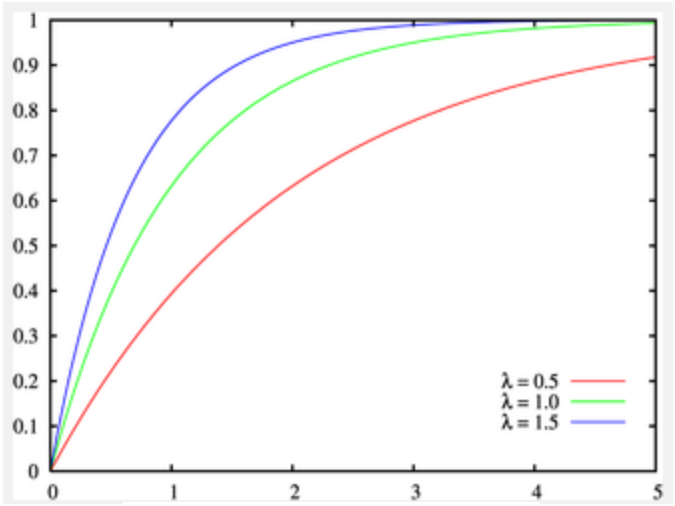
$$f(x) = \lambda e^{-\lambda x}$$




The cumulative probability function of this PDF is:

$$\int_{-\infty}^{x} f(t)dt = F(x) = 1 - e^{-\lambda x}$$

u




$$F^{-1}(u) = -\frac{\ln(u)}{\lambda}$$

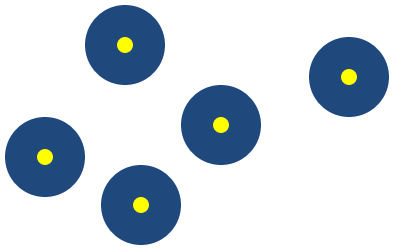
# Redundancy Sampling

Eliminate redundant attributes

- eliminate correlated attributes
  - km vs. miles
  - a + b + c = d → can eliminate ‘c’ (or ‘a’ or ‘b’

Eliminate redundant data

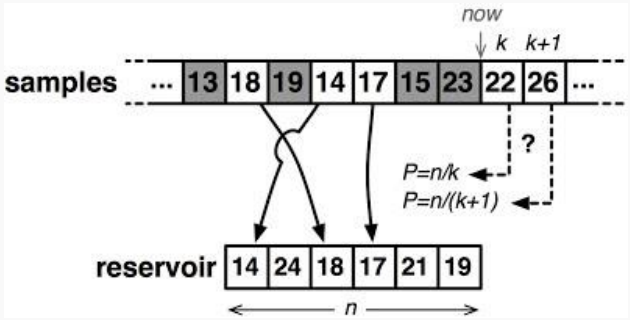
- cluster the data with small ranges
- only keep the cluster centroids
- store size of clusters along to keep importance



# Reservoir Sampling

```
/*
  S has items to sample, R will contain the result
*/
ReservoirSample(S[1..n], R[1..k])
  // fill the reservoir array
  for i = 1 to k
      R[i] := S[i]

  // replace elements with gradually decreasing probability
  for i = k+1 to n
    j := random(1, i)   // important: inclusive range
    if j <= k
        R[j] := S[i]
```




Probabilities

- $k/i$ for the $i^{th}$ sample to go into the reservoir
- $1/k \cdot k/i = 1/i$ for the $j^{th}$ reservoir element to be replaced
- $k/n$ for all elements in the reservoir after $n$ has been reached
- can be shown via induction

A good algorithm to use for streaming data when $n$ is growing

# Sampling of Well-Scattered Points

Used in the CURE high-dimensional clustering algorithm

- S. Guha, R. Rajeev, and K. Shim. "CURE: an efficient clustering algorithm for large databases." *ACM SIGMOD,* 27(2): 73-84, 1998

Algorithm

- initialize the point set *S* to empty
- pick the point farthest from the mean as the first point for *S*
- then iteratively pick points that are furthest from the points in S collected so far

Complexity is $O(m \cdot n^2)$

- *n* is the total number of points, *m* is the number of desired points
- can find arbitrarily shaped clusters and preserve outliers, too
- need some good data structures to run efficiently: kd-tree, heap