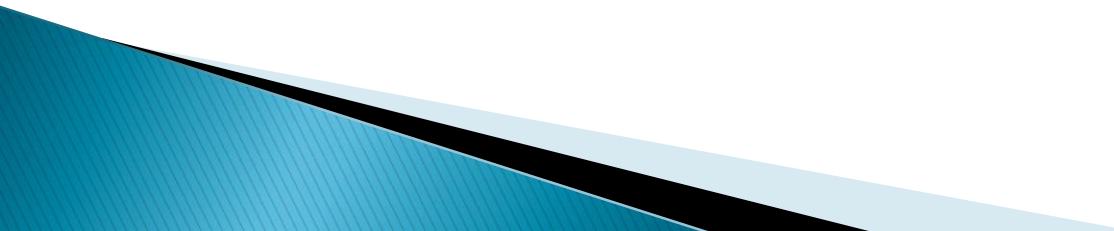


Agenda

- ▶ Data Pre-processing
 - Recap
 - Finish Slides
 - Quiz
 - ▶ Feature Engineering
 - Highlight Google Colabs
 - Slides
 - Quiz
 - ▶ Project Proposal Reviews
- 

Feature Engineering

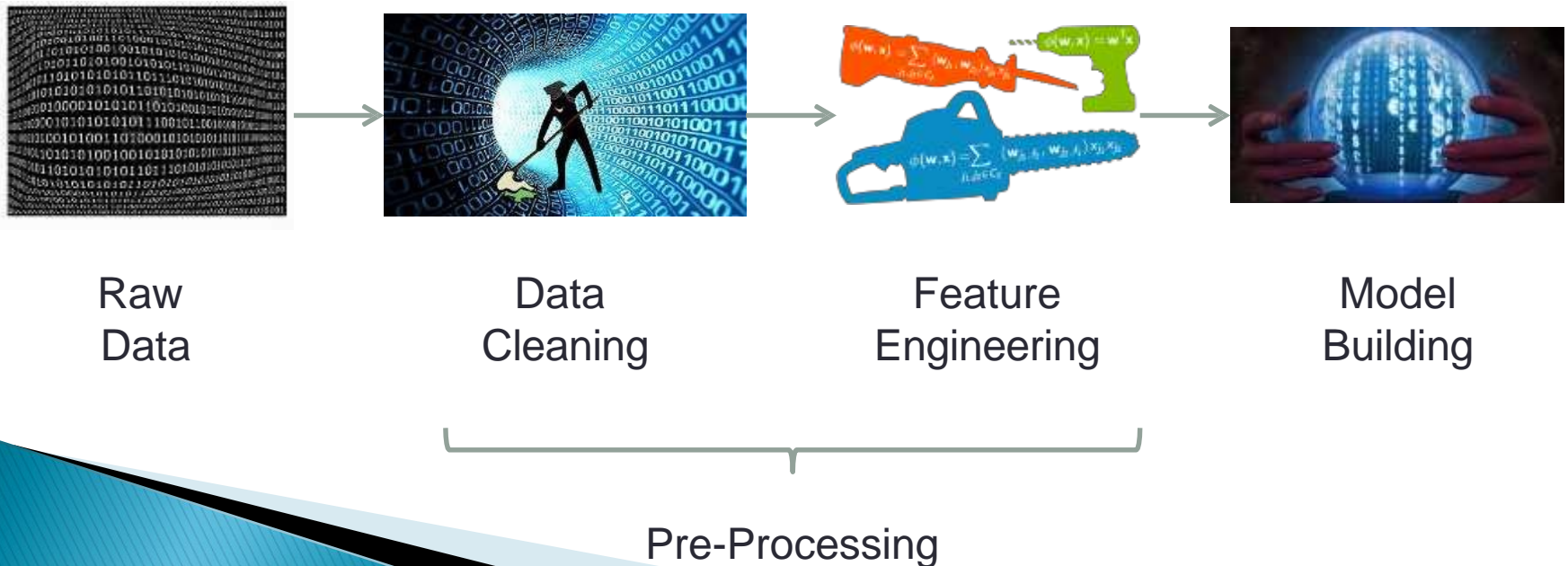
Andrew McCarren

With thanks to
David Epstein Socure

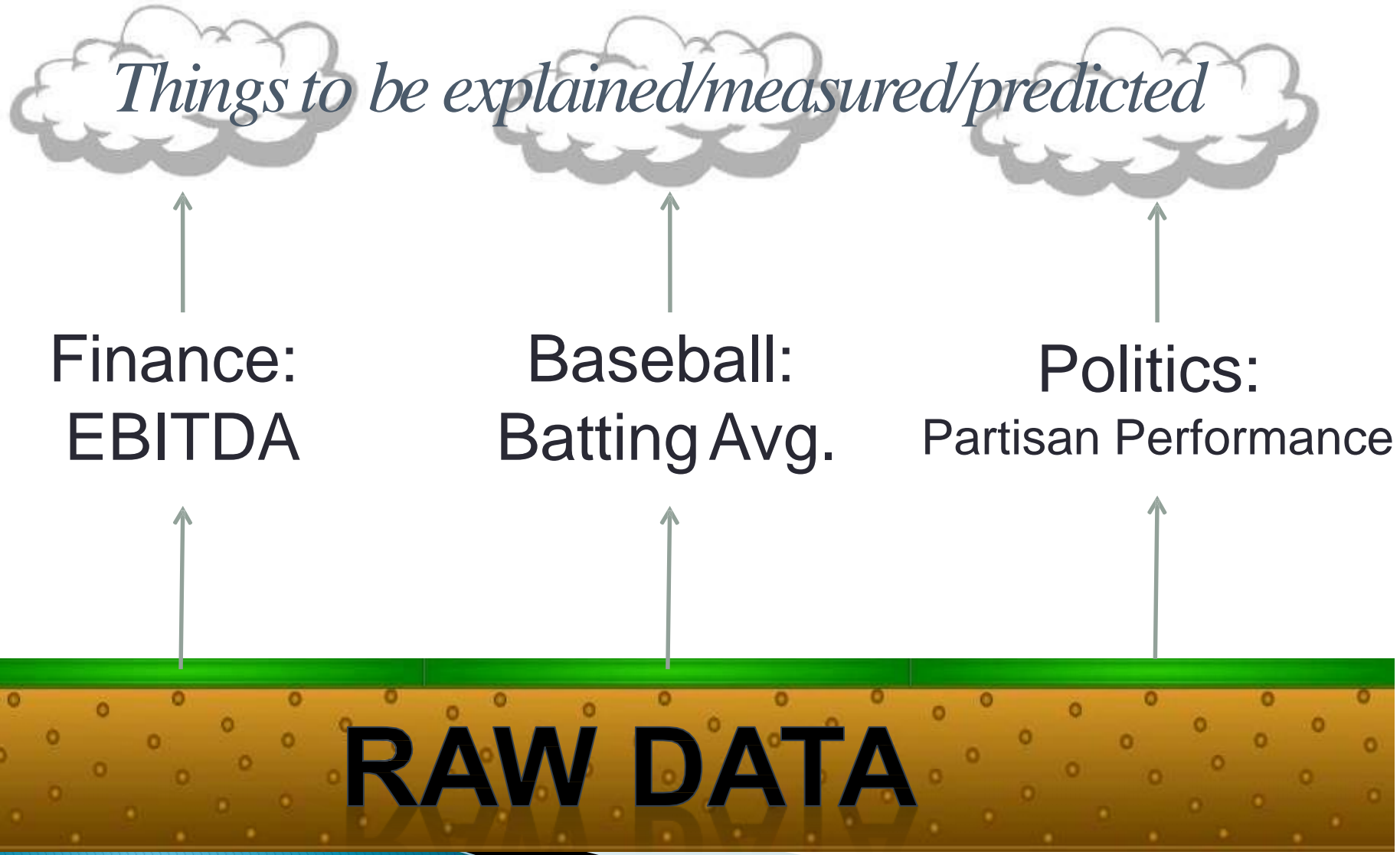
Feature Engineering

- (My) Definition: Transforming data to create model inputs.

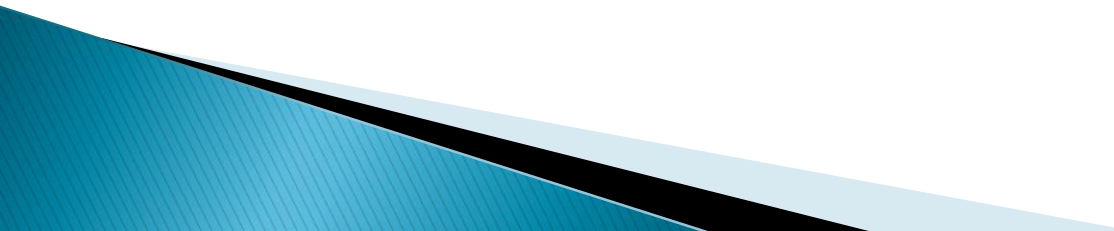
Data Workflow



Features are engineered everywhere



The Big Questions

- Seen in this light, FE is ubiquitous (as all truly important concepts are)
 - Any time you construct an intermediate variable, you're doing FE
 - Two questions naturally arise:
 1. How do you construct “good” features?
 2. What are the limits on this process?
 - I'll answer the second one first, because it's easier....
- 

Limits on Feature Engineering

- In medical studies, social science, financial analysis, etc., two main problems emerge
- Eating up degrees of freedom: relatively small data sets
 - # of respondents in survey
 - # of patients in trial
 - # of elections to Congress
 - If your data lives in an $N \times K$ matrix, you want to make sure that K is small relative to N
- Relevance to hypothesis testing, emphasis on explanation
 - You generally start with an equation defining the relationship between the key independent and dependent variables
 - Other variables enter your model as controls, not really interested in their functional form

Limits on Feature Engineering

◀ In most modern data science applications, neither is an issue

- We start with lots of data, and
- Care more about prediction than explanation

◀ So why not add in lots of extra variables?

- Think of your data not as what goes into your model, but a starting point for the creation of new variables, which can then be combined...



Limits to Feature Engineering

- First, adding many correlated predictors can decrease model performance
 - Adding an x4 term to above example actually reduces model fit:

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.648123	0.628178	1.032	0.336513
x	-0.009230	0.103740	-0.089	0.931593
x2	0.866738	0.139087	6.232	0.000432 ***
x4	0.004852	0.005361	0.905	0.395572


Residual standard error: 1.088 on 7 degrees of freedom

Multiple R-squared: 0.9902, Adjusted R-squared: 0.986

F-statistic: 236.1 on 3 and 7 DF, p-value: 2.15e-07

- More variables make models less interpretable
- Models have to be generalizable to other data
 - Too much feature engineering can lead to overfitting
 - Close connection between feature engineering and cross-validation

How To Select a “Good” Set of Features

- This is the open-ended question in the field
 - Separate (but related to) the question of feature selection – which variables to retain in a model.
 - You can use some metrics to tell which features are useful, one at a time, like Pearson correlation coefficient
 - But this can't tell you which set of features works best together
 - This is an NP complete problem, clearly computationally hard
 - Many new data analysis services include automated feature engineering as part of their packages
 - But if there are features you want in your model, it's best to add them in explicitly, rather than depend on these generators.
- 

A “Middle Theory” of FE

- Start with a reasonable-sized set of features
 - Include features suggested by domain knowledge
 - Test these out individually, build from the bottom up
- Number and type of features depend on model used
 - Can include more features if models does some feature selection
 - **Lasso regression**, e.g., logit with $||L_1||$ regularization (but not $||L_2||$ ridge)
 - GBM (Gradient Boosting Models) with backward pruning (but not random forests)
 - Stepwise regression, with either forward or backward feature selection
 - Some models are invariant to monotonic variable transformations
 - Tree-based approaches divide variables into two groups at each branch
- So, no perfect answer. But there are some standard techniques every data scientist should have in their bag of tricks.

Watch this!

Non-numeric to numeric

1. Count # of times each value appears
2. One-hot encoding

Zip Code	Count
10024	4
63105	2
94304	1
06443	3
10024	4
63105	2
06443	3
10024	4
10024	4
06443	3

Non-numeric to numeric

3. The “hash trick”

string	h(string)
“The”	36
“quick”	8
“brown”	92
“fox”	14
“jumps”	75
“over”	25
“the”	36
“lazy”	44
“dog”	21

Single Variable Transformations

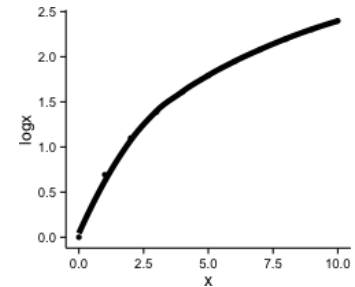
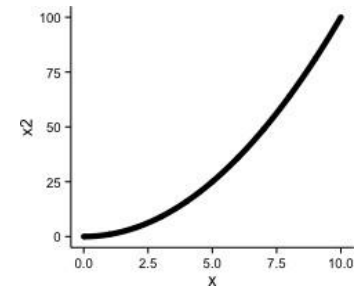
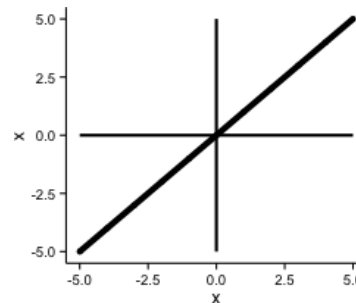
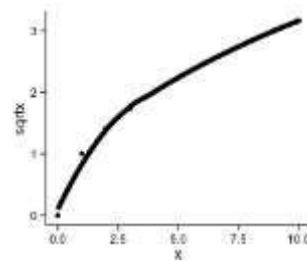
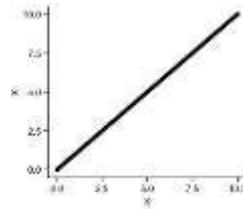
x

x^2

\sqrt{x}

$\text{Log}(x)$

scaling



Two-variable combinations

1. Add: Sum similar-scaled variables

Q1	Q2	Q3	Q4	Total
33	88	51	81	251
11	11	72	30	124
15	36	70	55	176
70	82	8	50	209
99	56	35	86	276
7	20	10	71	107
65	0	25	74	164
96	25	2	89	211
60	29	56	92	238
63	50	96	61	269

Two-variable combinations

2. Subtract: Difference relative to baseline

ViewerID	MovieID	Date	Rating	Days Since First Rating
44972004	8825	1/1/13	5	0
44972004	0471	2/1/13	4	31
44972004	3816	3/1/13	5	59
44972004	8243	4/1/13	3	90
44972004	2855	5/1/13	5	120
44972004	9923	6/1/13	2	151
44972004	1023	7/1/13	4	181
44972004	8306	8/1/13	3	212
44972004	2771	9/1/13	2	243
44972004	5281	10/1/13	2	273

Two-variable combinations

3. Multiply: Interactive effects

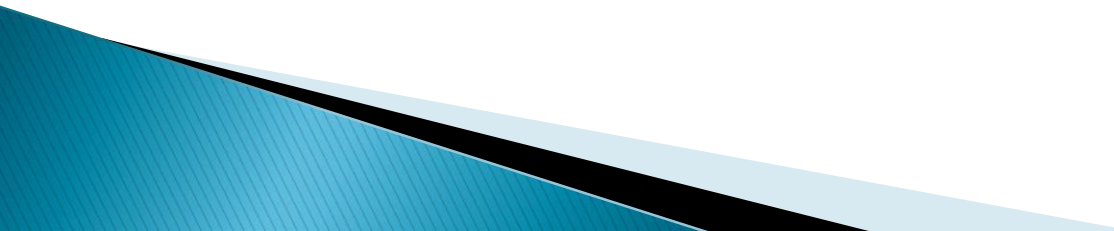
ViewerID	Country	Domestic	DSFR	Dom*DSFR
8825	US	1	38	38
0471	CA	0	277	0
3816	FR	0	187	0
8243	US	1	33	33
2855	US	1	87	87
9923	GB	0	42	0
1023	IT	0	192	0
8306	CA	0	365	0
2771	US	1	505	505
5281	FI	0	49	0

Two-variable combinations

4. Divide: Scaling/Normalizing

Country	GDP	Population	GDP/Capit a
US	159849	275	581
CA	731812	111	6593
FR	826320	90	9181
IT	573494	80	7169
IR	609223	22	27692
GB	717673	60	11961
NE	605257	15	40350
MX	687944	124	5548
RU	203319	402	506
FI	744983	40	18625

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?

- ▶ Binning method:
 - first sort data and partition into (equi-depth)s
 - then 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
- ▶ Regression
 - smooth by fitting the data into regression functions
- Moving Average/Exponential Smoothing etc.
- Wavelett Analysis/Fourier Transforms

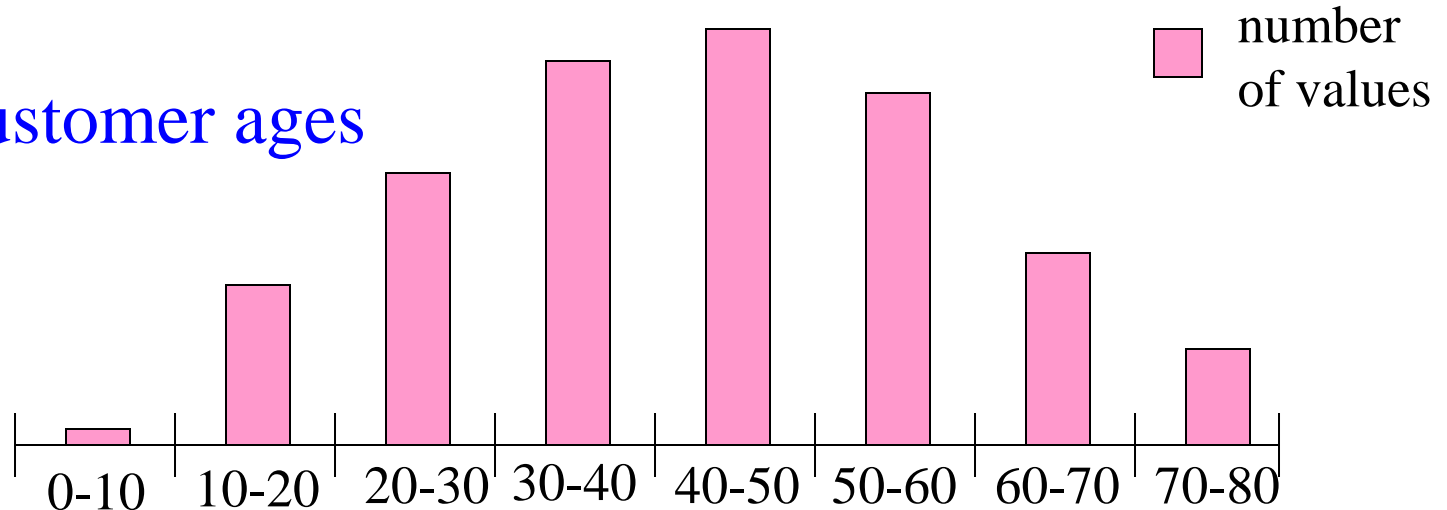
Simple Discretization Methods: Binning

- ▶ **Equal-width (distance) partitioning:**
 - It 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:**
 - It divides the range into N intervals, each containing approximately same number of samples
 - Good data scaling
 - Managing categorical attributes can be tricky.

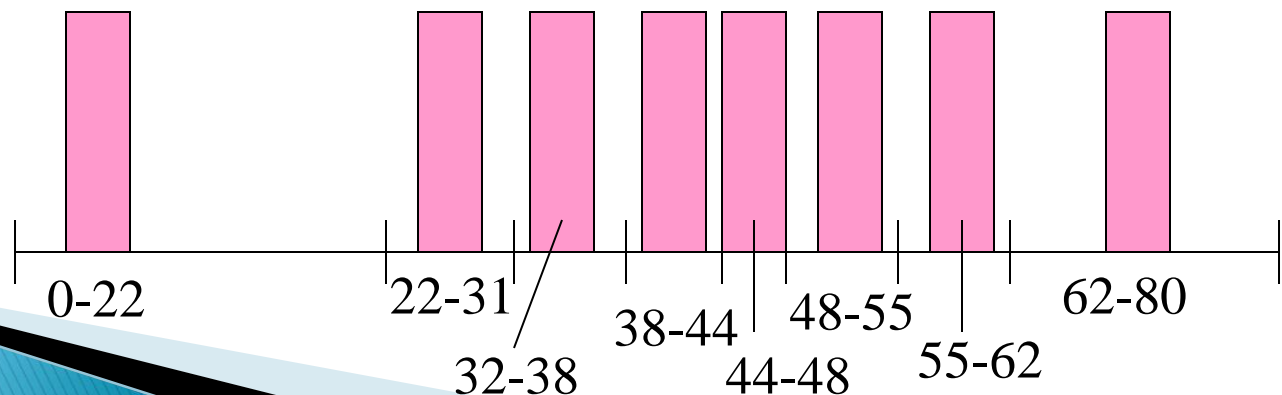
Simple Discretization Methods: Binning

Example: customer ages

Equi-width
binning:



Equi-depth
binning:



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 (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

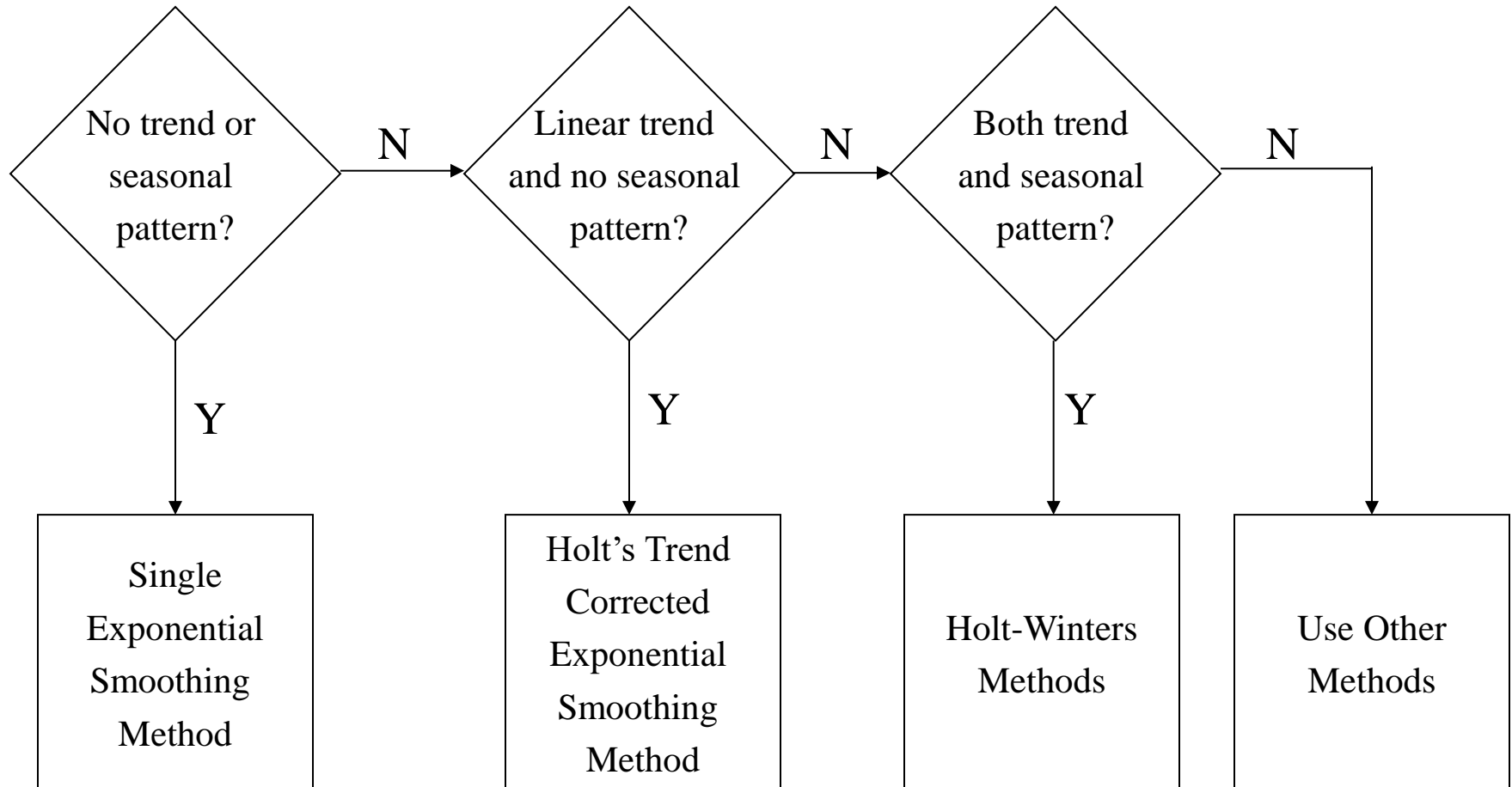
Motivation of Exponential Smoothing

- ▶ Simple moving average method assigns equal weights ($1 / k$) to all k data points.
- ▶ Arguably, recent observations provide more relevant information than do observations in the past.
- ▶ So we want a weighting scheme that assigns decreasing weights to the more distant observations.

Exponential Smoothing

- ▶ Exponential smoothing methods give larger weights to more recent observations, and the weights decrease exponentially as the observations become more distant.
- ▶ These methods are most effective when the parameters describing the time series are changing SLOWLY over time.

Data vs Methods



Simple Exponential Smoothing

- ▶ The Simple Exponential Smoothing method is used for forecasting a time series when there is no trend or seasonal pattern, but the mean (or level) of the time series y_t is slowly changing over time.
- ▶ NO TREND model

$$y_t = \beta_o + \varepsilon_t$$

Procedures of Simple Exponential Smoothing Method

- ▶ **Step 1:** Compute the initial estimate of the mean (or level) of the series at time period $t = 0$

$$\ell_0 = \bar{y} = \frac{\sum_{t=1}^n y_t}{n}$$

- ▶ **Step 2:** Compute the updated estimate by using the smoothing equation

$$\ell_T = \alpha y_T + (1 - \alpha)\ell_{T-1}$$

where α is a smoothing constant between 0 and 1.

Procedures of Simple Exponential Smoothing Method

Note that

$$\begin{aligned}\ell_T &= \alpha y_T + (1-\alpha)\ell_{T-1} \\ &= \alpha y_T + (1-\alpha)[\alpha y_{T-1} + (1-\alpha)\ell_{T-2}] \\ &= \alpha y_T + (1-\alpha)\alpha y_{T-1} + (1-\alpha)^2 \ell_{T-2} \\ &= \alpha y_T + (1-\alpha)\alpha y_{T-1} + (1-\alpha)^2 \alpha y_{T-2} + \dots + (1-\alpha)^{T-1} \alpha y_1 + (1-\alpha)^T \ell_0\end{aligned}$$

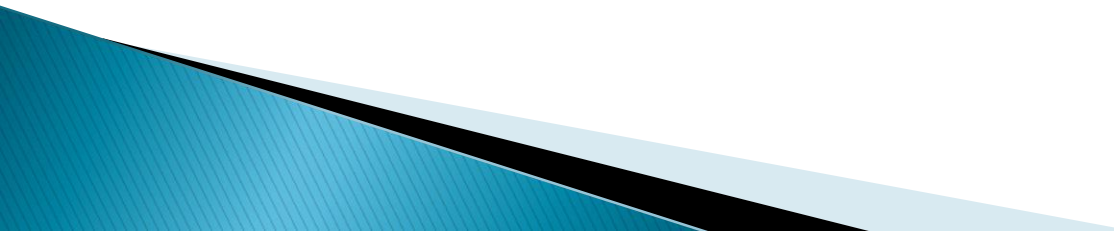
The coefficients measuring the contributions of the observations decrease exponentially over time.

Better explanation

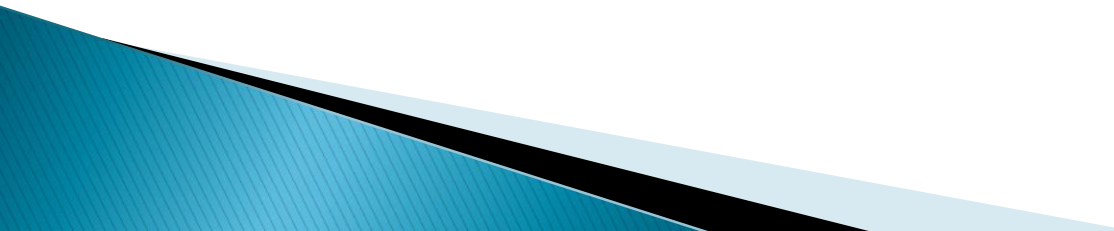
Quiz

- ▶ Smoothing methods help us handle Noisy Data. Which statements are correct.
 - 1. Exponential smoothing gives larger weights to the most recent data.
 - 2. Exponential smoothing methods are most effective when the parameters describing the time series are not changing slowly.
 - 3. Moving average only uses the last k data points and gives them an equal weight.

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
- 

Reasons for using Transformations

- ▶ Convenience
 - ▶ Reducing skewness
 - ▶ Equal spreads
 - ▶ Linear relationships
 - ▶ Additive relationships
- 

Data Transformation: Normalization

- ▶ min-max normalization

$$v' = \frac{v - \min_A}{\max_A - \min_A} (\text{new_max}_A - \text{new_min}_A) + \text{new_min}_A$$

- ▶ z-score normalization

$$v' = \frac{v - \text{mean}_A}{\text{stand_dev}_A}$$

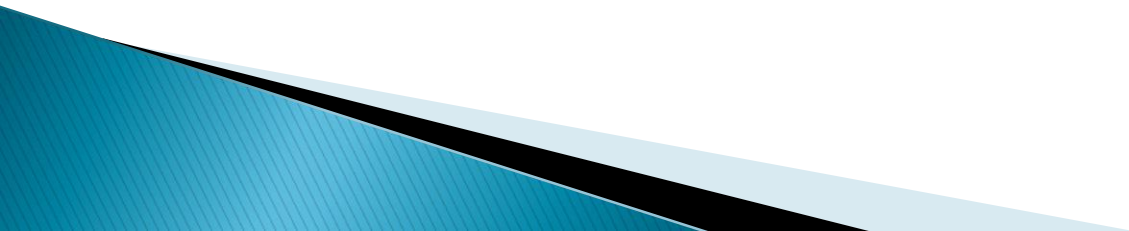
- ▶ normalization by decimal scaling

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


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)/ Multiple Correspondence analysis (MCA)
 - 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)



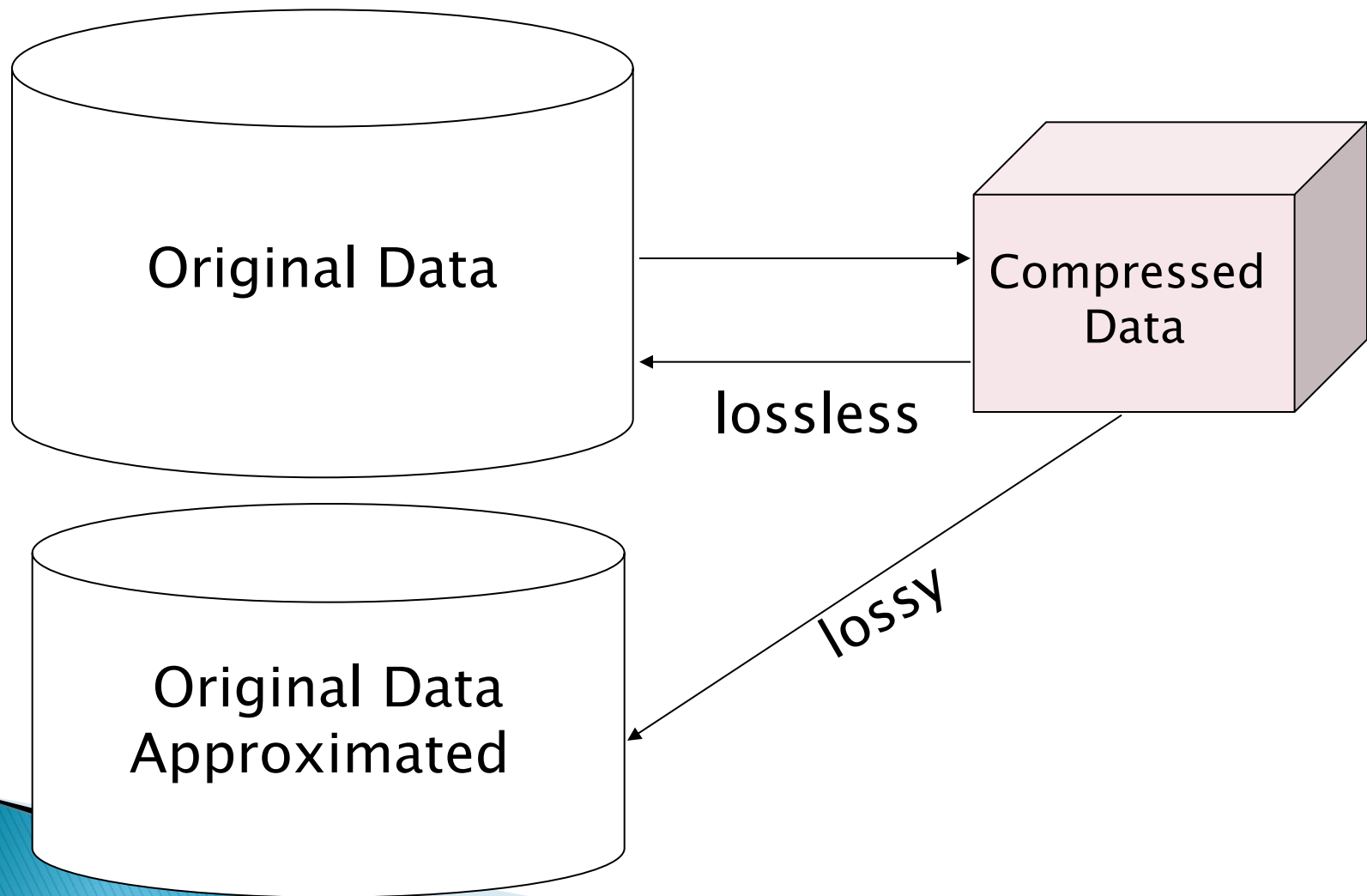
Data Reduction Strategies

- ▶ Warehouse may store terabytes of data: Complex data analysis/mining may take a very long time to run on the complete data set
 - ▶ Data reduction
 - Obtains a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
 - ▶ Data reduction strategies
 - Data cube aggregation
 - Dimensionality reduction
 - Numerosity reduction
 - Discretization and concept hierarchy generation
- 

Dimensionality Reduction

- ▶ 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
 - ▶ Used methods such as Multivariate techniques to reduce the number of components.
- 

Data Compression

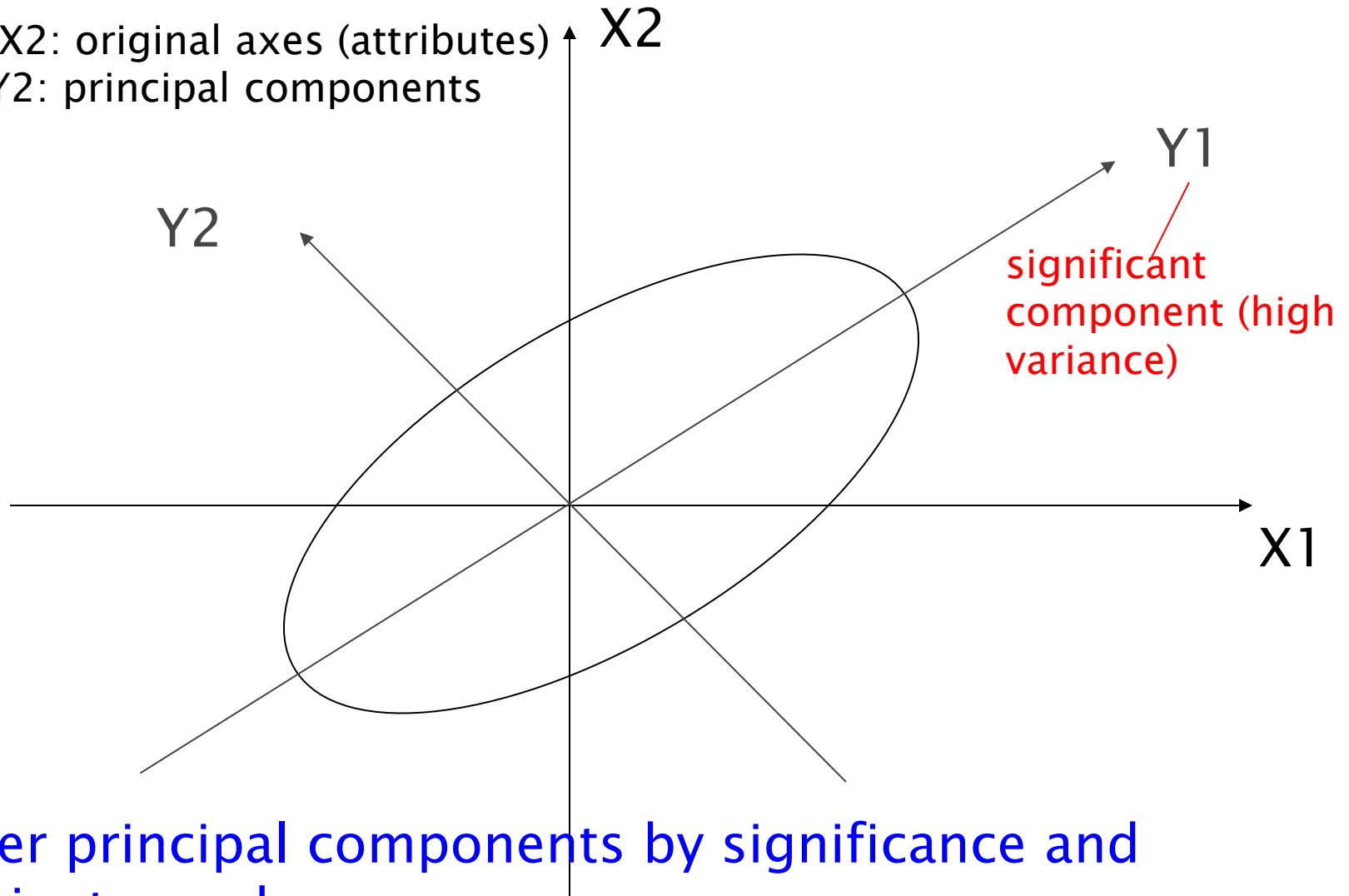


Principal Component Analysis or Karhunen–Loeve (K–L) method

- ▶ Given N data vectors from k -dimensions, find $c \leq k$ orthogonal vectors that can be best used to represent data
 - The original data set is reduced to one consisting of N data vectors on c principal components (reduced dimensions)
- ▶ Each data vector is a linear combination of the c principal component vectors
- ▶ Works for numeric data only
- ▶ Used when the number of dimensions is large

Principal Component Analysis

X_1, X_2 : original axes (attributes)
 Y_1, Y_2 : principal components



Order principal components by significance and eliminate weaker ones

Numerosity Reduction:

Reduce the volume of data

▶ Parametric methods

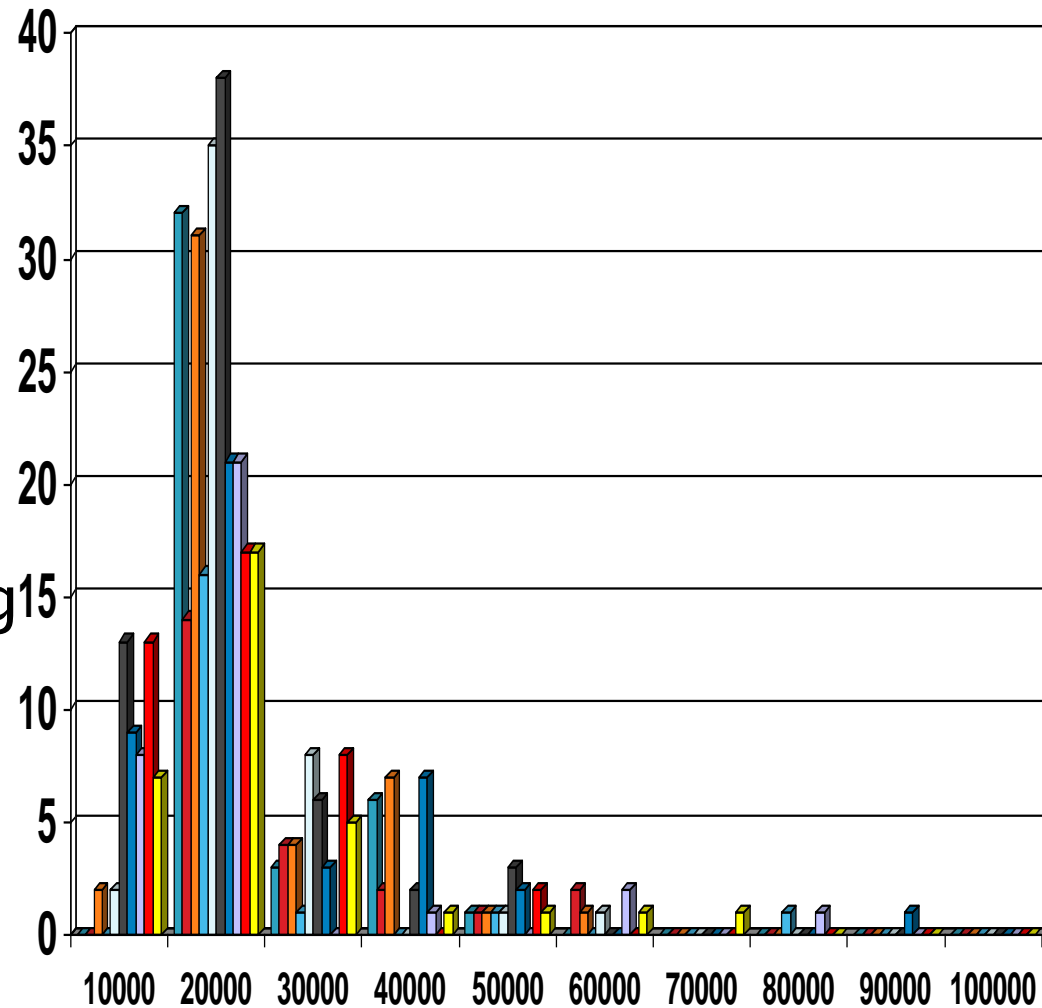
- Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
- Log-linear models: obtain value at a point in m -D space as the product on appropriate marginal subspaces

▶ Non-parametric methods

- Do not assume models
- Major families: histograms, clustering, sampling

Histograms

- ▶ A popular data reduction technique
- ▶ Divide data into buckets and store average (or sum) for each bucket
- ▶ Can be constructed optimally in one dimension using dynamic programming
- ▶ Related to quantization problems.



Discretization

- ▶ Three types of attributes:
 - Nominal — values from an unordered set
 - Ordinal — values from an ordered set
 - Continuous — real numbers
- ▶ Discretization:
 - divide the range of a continuous attribute into intervals
 - why?
 - Some classification algorithms only accept categorical attributes.
 - Reduce data size by discretization
 - Prepare for further analysis

Discretization and Concept hierarchy

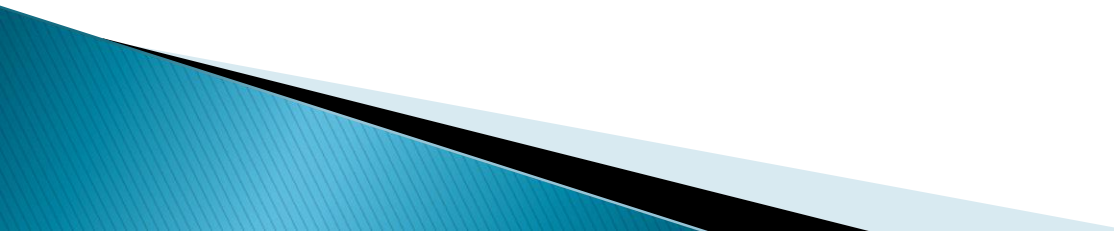
▶ Discretization

- 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.

▶ Concept hierarchies

- reduce the data by collecting and replacing low level concepts (such as numeric values for the attribute age) by higher level concepts (such as young, middle-aged, or senior).

Discretization and concept hierarchy generation for numeric data

- ▶ Binning/Smoothing (see sections before)
 - ▶ Histogram analysis (see sections before)
 - ▶ Clustering analysis
 - ▶ Entropy-based discretization (Expected Info)
 - ▶ Segmentation by natural partitioning
- 

Entropy (expected information)

- ▶ When discretising a data series we need to consider the information we get from the new categories
- ▶ How can we get the right splits in a series that will give us the most information

Entropy: $Ent(S_1) = -\sum_{i=1}^m p_i \log_2(p_i)$

Calculate the Entropy?

	INCOME < 50K	Income > 50K
Age >=25	4	6

	INCOME < 50K	Income > 50K
Age <25	9	1

$$Ent(S_1) = -\sum_{i=1}^m p_i \log_2(p_i)$$

Entropy-Based Discretization

Entropy: $Ent(S_1) = -\sum_{i=1}^m p_i \log_2(p_i)$

- ▶ Given a set of samples S , if S is partitioned into two intervals S_1 and S_2 using boundary T , the **information** $I(S,T)$ after partitioning is

$$I(S,T) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2)$$

- ▶ The boundary that maximizes the information gain 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.,

$$Ent(S) - I(T, S) > \delta$$

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

[Check this site](#)

Calculate info gain for first split?

HOURS STUDIED	A ON TEST
4	N
5	Y
8	N
12	Y
15	Y

- Split on Y/N without classification on Hours studied.
- Then first split is between 4 & 5
- Calculate total Entropy for A and not A
- Calculate $\text{info}(S, T=4.5)$
- Iterate until you are satisfied.

Calculations

New Split on Hours spent

	A ON TEST	LOWER THAN A
≤ 4.5	0	1
> 4.5	3	1

$$Entropy(D_{\leq 4.5}) = -\left(\frac{1}{1}\log_2(1) + 0\log_2(0)\right) = 0 + 0 = 0$$

$$Entropy(D_{> 4.5}) = -\left(\frac{3}{4}\log_2\left(\frac{3}{4}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right) = .311 + .5 = .811$$

$$Info_a(D_{new}) = \frac{1}{5}(0) + \frac{4}{5}(.811) = .6488$$

Segmentation by natural partitioning

- Users often like to see numerical ranges partitioned into relatively uniform

The 3–4–5 rule

can be used to segment numerical data into relatively uniform, “natural” intervals.

- If an interval covers 3, 6, 7 or 9 distinct values at the **most significant digit**, partition the range into 3 equiwidth intervals for 3,6,9 or 2–3–2 for 7
- If it covers 2, 4, or 8 distinct values at the **most significant digit**, partition the range into 4 equiwidth intervals
- If it covers 1, 5, or 10 distinct values at the **most significant digit**, partition the range into 5 equiwidth intervals

The rule can be recursively applied for the resulting intervals

Example

- ▶ Suppose that profit data values for year 2017 for a company range from $-3,51,976$ to $+4,70,00,896$.
- ▶ For practical purpose of avoiding noise, extremely high or extremely low values are not considered. So first we need to smooth out our data. Let's discard bottom 5% and top 5% values.
- ▶ Suppose after discarding above data new values for $LOW = -159876$ and $HIGH = 1838761$.
- ▶ Most Significant Digit or MSD is at million position, see highlighted digit : -159876 and 1838761 .

Example Contd

- ▶ Next step is to *round down* LOW and *round up* HIGH to MSD that million position.
 - So LOW = -1000000 and HIGH = 2000000. -1000000 is nearest down million to -159876 and 2000000 is nearest up million to 1838761.
- ▶ Now let's identify range of this interval.
 - Range = HIGH - LOW that is $2000000 - (-1000000) = 3000000$. We consider only MSD here which is 3.
- ▶ Now that we know range MSD = 3, we can apply rule #1.

Example Contd

- ▶ Rule #1 says that we can divide this interval into three equal size intervals:
 - Interval 1 : -1000000 to 0
 - Interval 2 : 0 to 1000000
 - Interval 3 : 1000000 to 2000000
- ▶ You should be thinking of how 0 can be part of multiple intervals? You're right! We should represent it as follows:
 - Interval 1 : $(-1000000 \dots 0]$
 - Interval 2 : $(0 \dots 1000000]$
 - Interval 3 : $(1000000 \dots 2000000]$
 - Here $(a \dots b]$ denotes range that excludes a but includes b . $(,]$ is notation for half-open interval.

Summary

- ▶ Data preparation or preprocessing is a big issue for both data warehousing and data mining
- ▶ Descriptive data summarization is need for quality data preprocessing
- ▶ Data preparation includes
 - Data cleaning and data integration
 - Data reduction and feature selection
 - Discretization
- ▶ A lot a methods have been developed but data preprocessing still an active area of research

The End