SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Established in collaboration with MIT

Features, Data, Text Processing

Soujanya Poria

Adapted from "Introduction to Data Mining" by Tan, Steinbach, Karpatne and Kumar

50.038 Computational data science

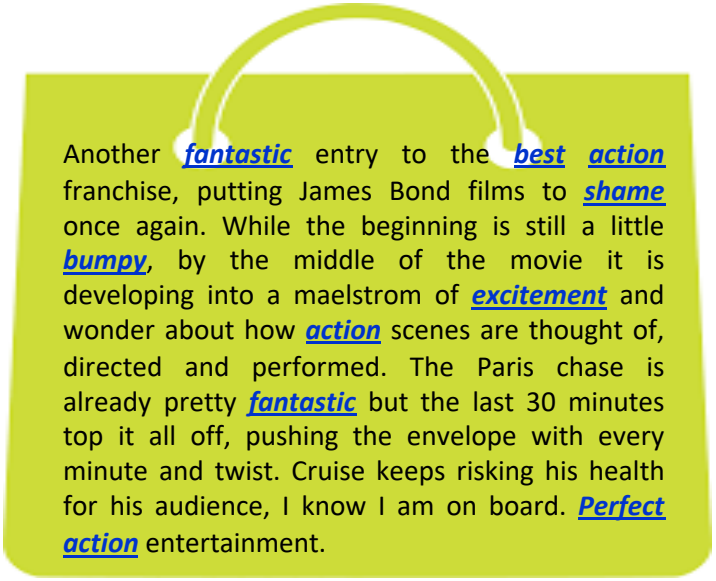
Objectives

- Understand the properties and different types of features
- Understand the characteristics and possible issues with datasets
- Understand the various data pre-processing steps
- Understand basic NLP and textual feature extraction

Features

Examples of Features

- Home Type, Marital Status, Income Level
- Uni-grams, Bi-grams, Tri-grams



ID	House	Marital Status	Income Level	Eat KFC
1	HDB	Single	High	***No***
2	Private	Married	Medium	***No***
3	Private	Single	Low	***No***
4	HDB	Married	High	***No***
5	Private	Divorced	Low	***Yes***

Properties of Features

- Distinctness: $= \neq$
  - E.g., Cat $\neq$ Dog
- Order: $< > \leq \geq$
  - E.g., A+ $>$ B-
- Meaningful differences: $+ \ -$
  - E.g., 08 Oct 2018 is three days after 05 Oct 2018
- Meaningful ratios: $\times \div$
  - E.g., Tom (18 years) is two times older than John (9 years)

Types of Features

- Nominal
  - Property: Distinctness
  - Examples: gender, eye colour, postal codes
- Ordinal
  - Properties: Distinctness and ordered
  - Examples: school level (primary/secondary), grades
- Interval
  - Properties: Distinctness, ordered and meaningful differences
  - Examples: calendar dates, temperatures (Celsius or Fahrenheit)
- Ratio
  - Properties: Distinctness, ordered and meaningful differences/ratios
  - Examples: length, time, counts

Types of Features

	Attribute Type	Transformation	Comments
Categorical Qualitative	Nominal	Any permutation of values	If all employee ID numbers were reassigned, would it make any difference?
	Ordinal	An order preserving change of values, i.e., *new_value = f(old_value)* where *f* is a monotonic function	An attribute encompassing the notion of good, better best can be represented equally well by the values {1, 2, 3} or by { 0.5, 1, 10}.
Numeric Quantitative	Interval	*new_value = a * old_value + b* where a and b are constants	Thus, the Fahrenheit and Celsius temperature scales differ in terms of where their zero value is and the size of a unit (degree).
	Ratio	*new_value = a * old_value*	Length can be measured in meters or feet.

Discrete VS Continuous

- Nominal, Ordinal, Interval and Ratio features can be represented by discrete or continuous values
- Discrete values (including Binary)
  - Finite or countable set of values
  - Typically represented as integers
  - E.g., course ID, postal codes
- Continuous values
  - Real values
  - Typically represented as floats
  - E.g., temperature, weight, height

Exercise 1: Feature Types

- Categorize the following features (state your assumptions):

Feature	Binary, Discrete, or Continuous?	Nominal, Ordinal, Interval, or Ratio?
Postal code		
Gender		
Height / Weight		
Student ID		
Grading system		
Date		

Exercise 1: Feature Types

- Categorize the following features (state your assumptions):

Feature	Binary, Discrete, or Continuous?	Nominal, Ordinal, Interval, or Ratio?
Postal code	Discrete	Nominal
Gender		
Height / Weight		
Student ID		
Grading system		
Date		

Feature Types

- Categorize the following features (state your assumptions):

Feature	Binary, Discrete, or Continuous?	Nominal, Ordinal, Interval, or Ratio?
Postal code	Discrete	Nominal
Gender	Binary	Nominal
Height / Weight	Continuous	Ratio
Student ID	Discrete	Nominal, Ordinal (if ID assigned by sequence)
Grading system	Binary (P/F), Discrete (A+,..,F), Continuous (Scores)	Ordinal, Ratio (Scores)
Date	Discrete (MM/YY), Continuous (time)	Interval

Good question!

- What is the feature type of – “the number of students at SUTD”?
  - Discrete
- You may wonder why is so?
  - A continuous variable can take any value i.e., float or decimal but number of students at SUTD can only take integer values. As the domain is only integer, hence it is discrete variable.

Data

Dataset Characteristics

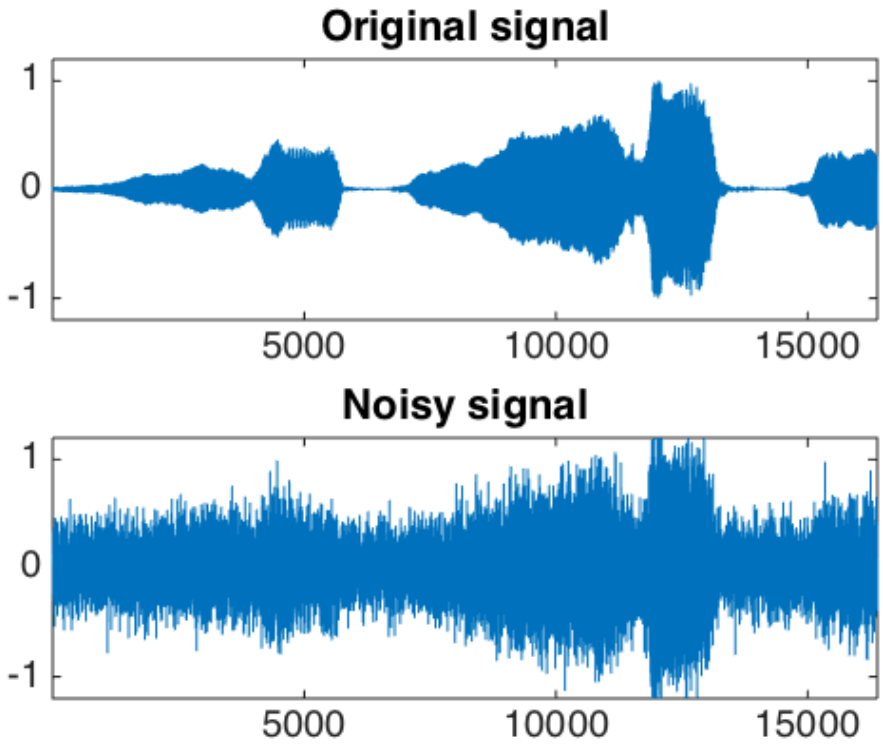
- Dimensionality (number of features)
  - Challenges of high-dimensional data, “Curse of dimensionality”
- Sparsity
  - E.g., In bag-of-words, most words will be zero (not used)
  - Advantage for computing time and space
- Resolution
  - Patterns depend on the scale
  - E.g., travel patterns on scale of hours, days, weeks

Possible Issues with Dataset

- Low quality dataset/features lead to poor models
  - E.g., a classifier build with poor data/features may incorrectly diagnose a patient as being sick when he/she is not
- Possible issues with dataset/features
  - Noise
  - Outliers
  - Missing values
  - Duplicate data
  - Wrong/Inconsistent data

Noise

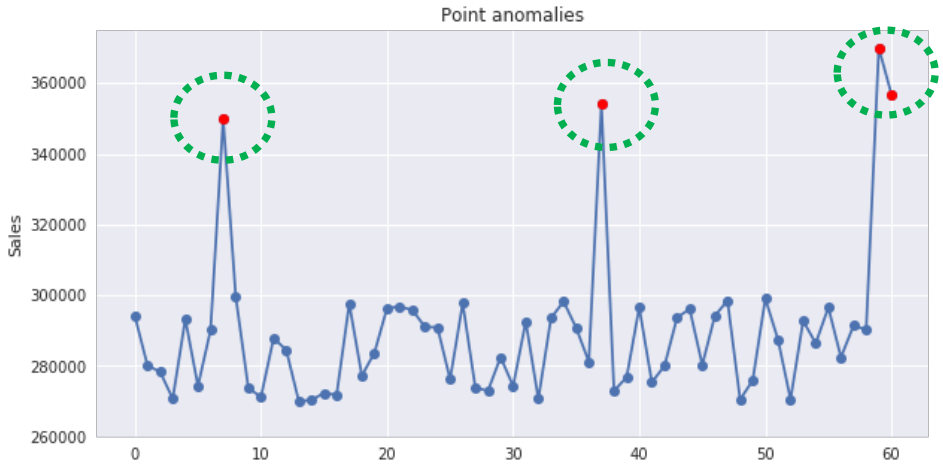
- For features, noise refers to random error/variance in original values
  - E.g., recording of a concert with background noise
  - E.g., check-in data on social media with GPS errors



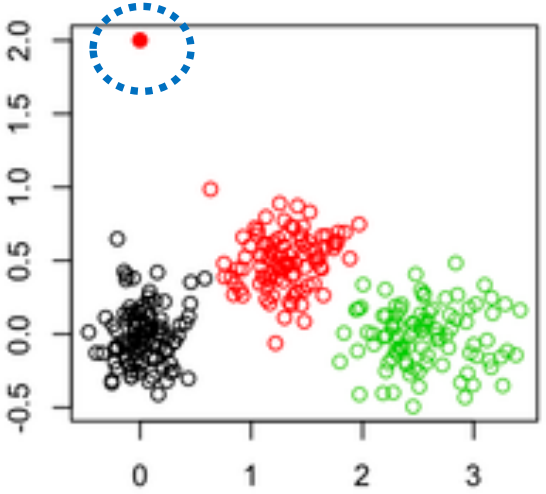
http://www.numerical-tours.com/matlab/audio_1_processing/

Outliers

- **Anomalous objects**: Observations with characteristics that are considerably different than most other observations in the data set
- **Anomalous values**: Feature values that are unusual with respect to typical values for that feature



https://thingsolver.com/anomaly-detection/



http://i.stack.imgur.com/tRDGU.png

Noise VS Outliers

- What other examples of noise and outliers can you think of?
- What are the differences between noise and outliers?
- What do we want to do with them (noise and outliers)?

Noise VS Outliers

- Noise
  - Due to random errors/variance in the data collection/measurement process
  - E.g., blurry images, noisy recordings
  - Want to remove them (noise reduction/removal)
- Outlier
  - Due to interesting events, which may have good/bad consequences
  - E.g., sudden increase in web traffic, large and odd online purchases
  - Want to identify/detect them (anomaly detection)

Missing Values

- Reasons for missing values
  - Incomplete data collection, e.g., people not providing annual income
  - Features not applicable to certain observations, e.g., annual income not applicable to children

- Types of missing values
  - Missing completely at random (MCAR)
    - Missing values are a completely random subset, e.g., data collection/survey is randomly lost
  - Missing at Random (MAR)
    - Missing values related to some other features, e.g., older adults not providing annual income
  - Missing Not at Random (MNAR)
    - Missing values related to unobserved features, e.g., not knowing age and income

Missing Values

- What to do with missing values?
  - Eliminate observations or variables
    - Ok for MCAR values, may not be ok for MAR and MNAR values
    - Need to understand the effects of this elimination
  - Estimate missing values
    - E.g., using averages in a time series or spatial data
  - Ignore the missing value during analysis
    - E.g., KNN using features with values

Duplicate Data

- Data set may include data objects that are duplicates, or almost duplicates of one another
  - Major issue when merging data from heterogeneous sources
- Examples:
  - Same person with multiple email addresses
- Data cleaning
  - Process of dealing with duplicate data issues

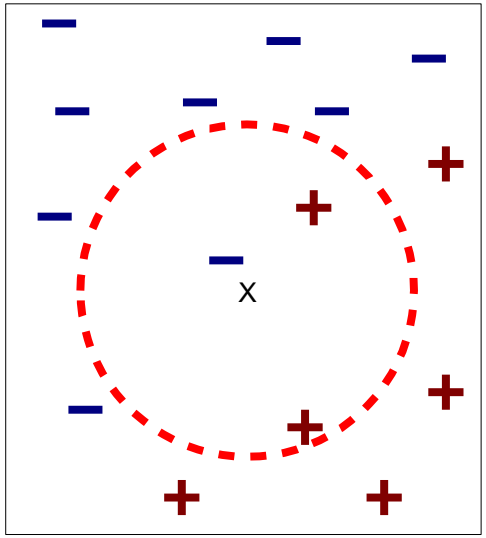
Wrong/Inconsistent Data

- Features may contain wrong or inconsistent values
  - E.g., user-provided street name and postal code not matching
  - E.g., negative values for weight, height, age, etc

- Ways to overcome wrong/inconsistent data
  - More stringent data collection
    - E.g., drop-down list for specific data input
  - Detect potentially wrong data values
    - E.g., allowable range for specific features
  - Correction of wrong/inconsistent values
    - E.g., correct postal code based on block number and street name

Classification Problems and Dataset

- Consider a dataset with the issues of noise, outliers, duplicate observations, missing values and wrong/inconsistent data.
- What would be the possible problems of applying the k-nearest neighbors (KNN) algorithm on this dataset? *Note: KNN assigns an observation to the class label of the k-nearest neighbor with majority voting*



Classification Problems and Dataset

- Consider a dataset with the issues of noise, outliers, duplicate observations, missing values and wrong/inconsistent data.
- What would be the possible problems of applying the k-nearest neighbors (KNN) algorithm on this dataset? *Note: KNN assigns an observation to the class label of the k-nearest neighbor with majority voting*
  - **Noise/Outliers**: If k-value is too small, may be overly sensitive to noise/outliers
  - **Duplicate observations**: K-nearest neighbors may be all duplicates
  - **Missing/wrong/inconsistent**: Distance measure may be inaccurate

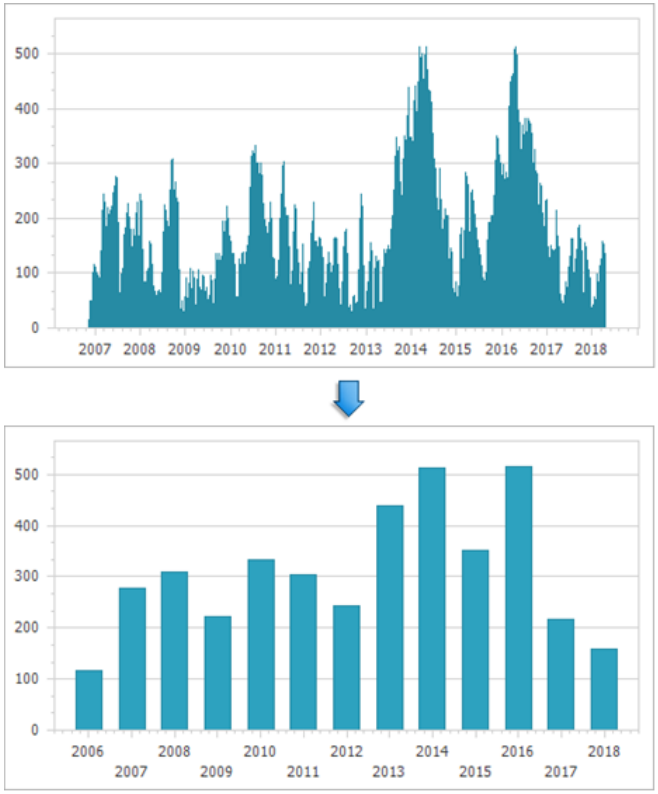
Data Preprocessing

- Aggregation
- Sampling
- Dimensionality Reduction
- Feature subset selection
- Feature creation
- Discretization and Binarization

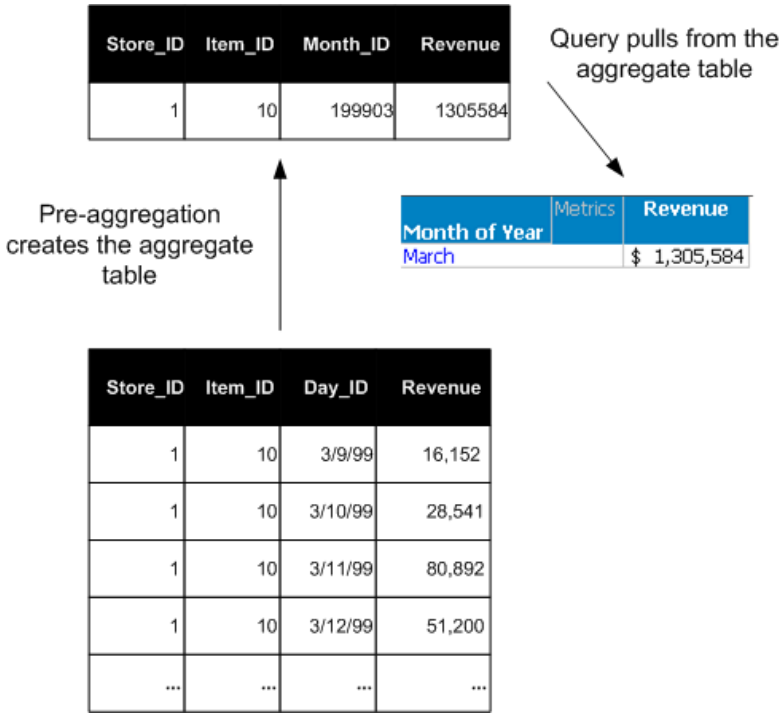
Aggregation

- Combining two or more features (or observations) into a single feature (or observation)
- Purpose
  - Data reduction
    - Reduce the number of features or observations
  - Change of scale
    - Cities aggregated into regions, states, countries, etc.
    - Days aggregated into weeks, months, or years
  - More “stable” data
    - Aggregated data tends to have less variability

Aggregation



https://documentation.devexpress.com



https://www2.microstrategy.com

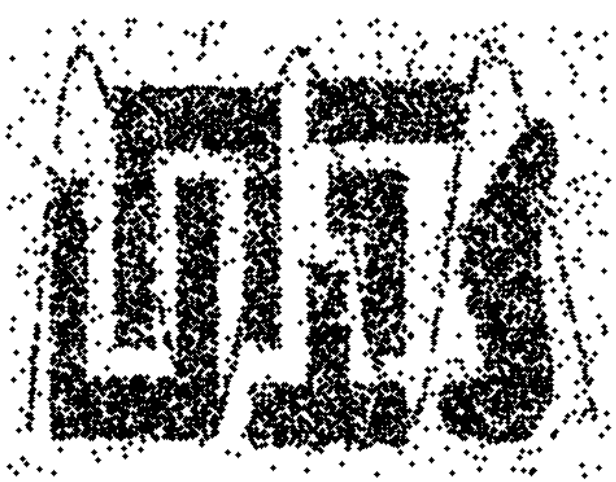
Data Sampling

- Sampling is the main technique employed for data reduction.
  - It is often used for both the preliminary investigation of the data and the final data analysis.

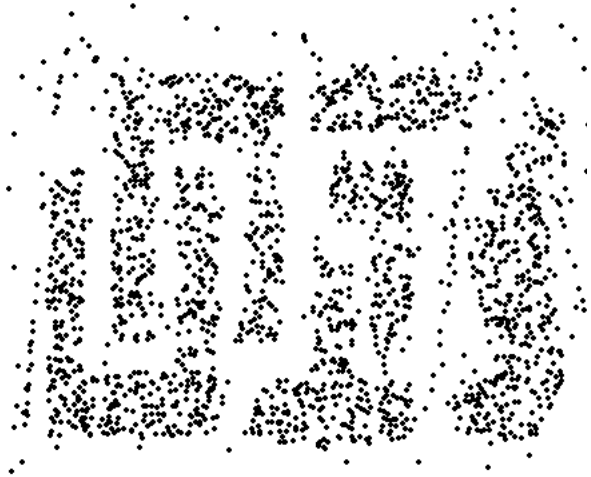
- Why use data sampling?
  - Expensive or time-consuming to ***obtain/collect*** entire set of relevant data
    - E.g., random survey instead of census on entire population
  - Expensive or time-consuming to ***process*** entire set of relevant data
    - Less of an issue these days with distributed computing

Data Sampling

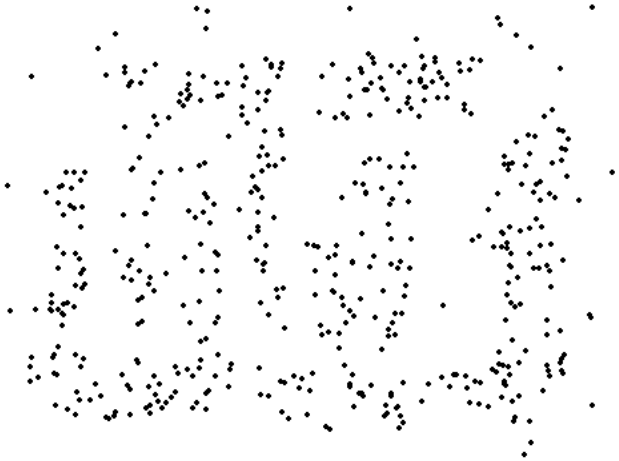
- The key principle for effective sampling is the following:
  - Using a sample will work almost as well as using the entire data set, if the sample is representative
  - A sample is representative if it has approximately the same properties (of interest) as the original set of data



8000 points



2000 Points



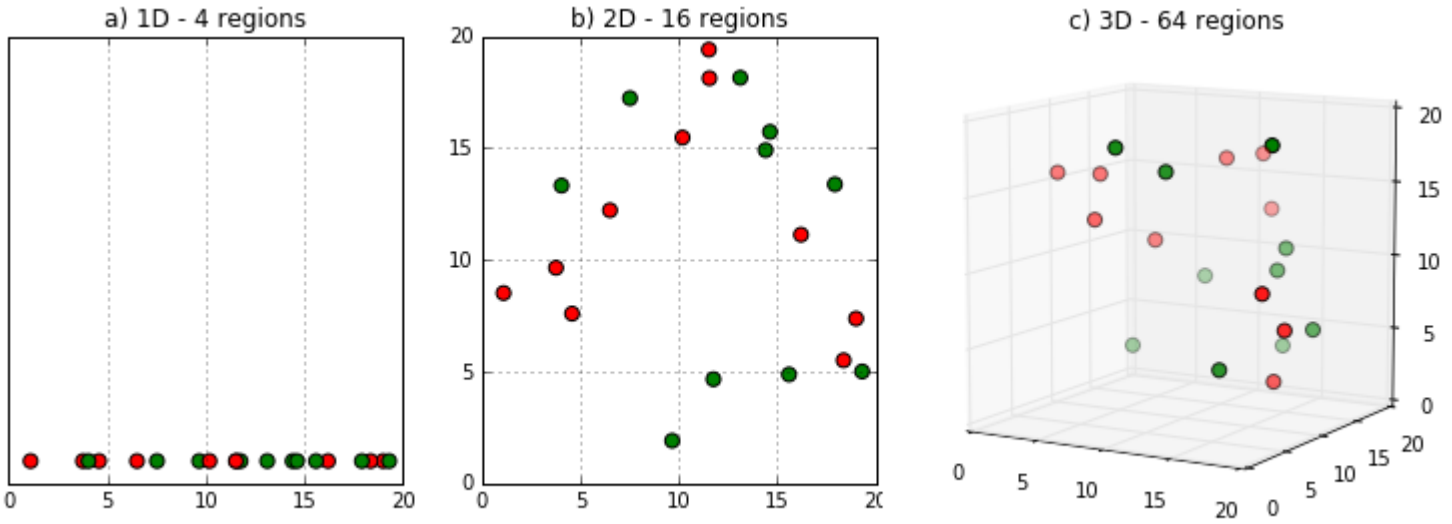
500 Points

Types of Sampling

- Simple Random Sampling
  - There is an equal probability of selecting any particular item
  - Sampling without replacement
    - As each item is selected, it is removed from the population
  - Sampling with replacement
    - Objects are not removed from the population as they are selected for the sample.
    - In sampling with replacement, the same object can be picked up more than once
- Stratified sampling
  - Split the data into several partitions; then draw random samples from each partition

Curse of Dimensionality

- As the number of features (dimensions) increases, more data (observations) is needed for an accurate classifier (model)
  - Why is this so? Data gets increasingly sparse in the space it occupies



Dimensionality Reduction

- Purpose:
  - Avoid curse of dimensionality
  - Reduce amount of time and memory required by data mining algorithms
  - Allow data to be more easily visualized
  - May help to eliminate irrelevant features or reduce noise
- Techniques
  - Data projection from high-dimensional to low-dimensional space
    - Linear algebra techniques, e.g., Principal Components Analysis, Singular Value Decomposition
  - Feature selection
    - More later

Motivation

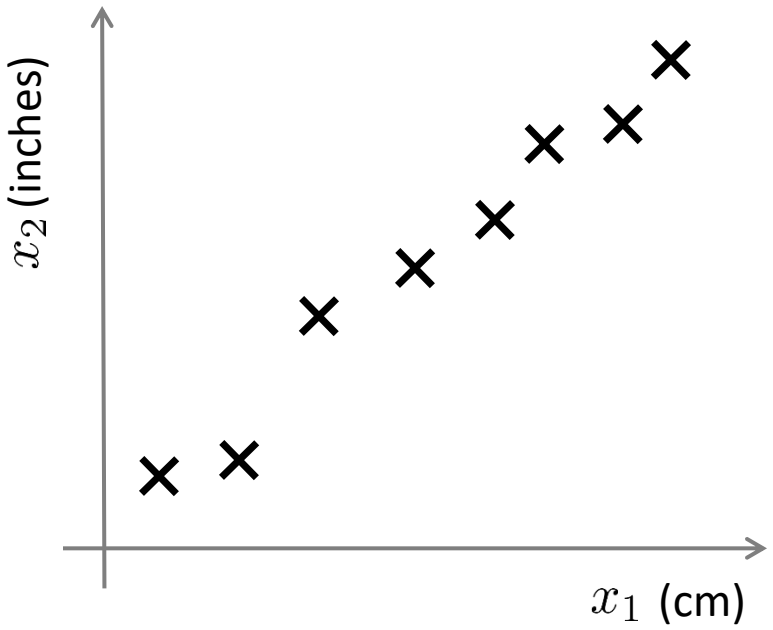
- Clustering
  - One way to summarize a complex real-valued data point with a single categorical variable
- Dimensionality reduction
  - Another way to simplify complex high-dimensional data
  - Summarize data with a lower dimensional real valued vector

Motivation

- Clustering
  - One way to summarize a complex real-valued data point with a single categorical variable
- Dimensionality reduction
  - Another way to simplify complex high-dimensional data
  - Summarize data with a lower dimensional real valued vector

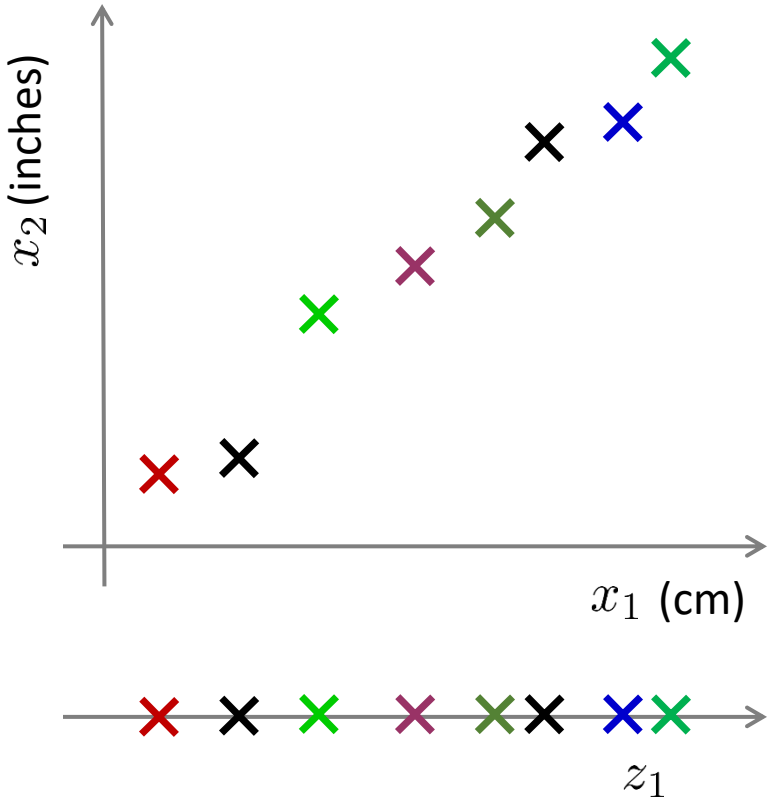
- Given data points in *d* dimensions
- Convert them to data points in $r<d$ dimensions
- With minimal loss of information

Data Compression



Reduce data from
2D to 1D

Data Compression



Reduce data from 2D to 1D

$$x^{(1)} \rightarrow z^{(1)}$$

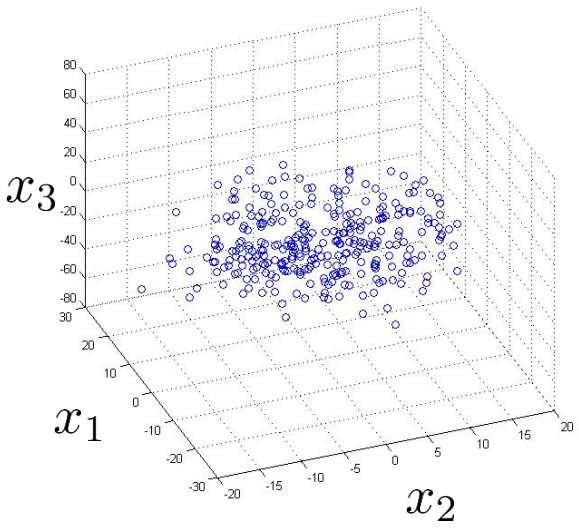
$$x^{(2)} \rightarrow z^{(2)}$$

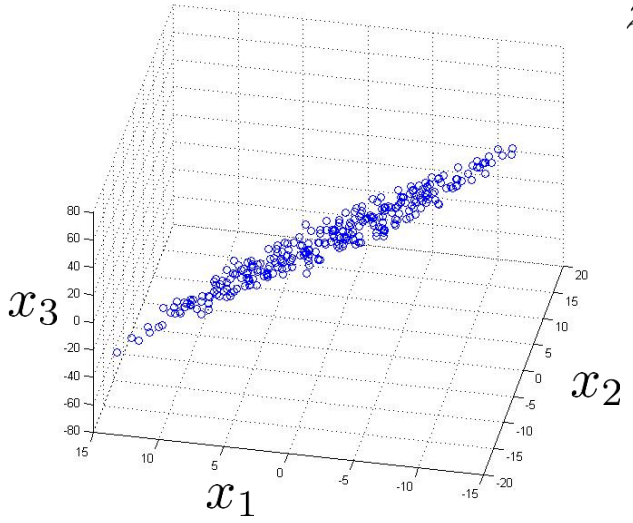
$$\vdots$$

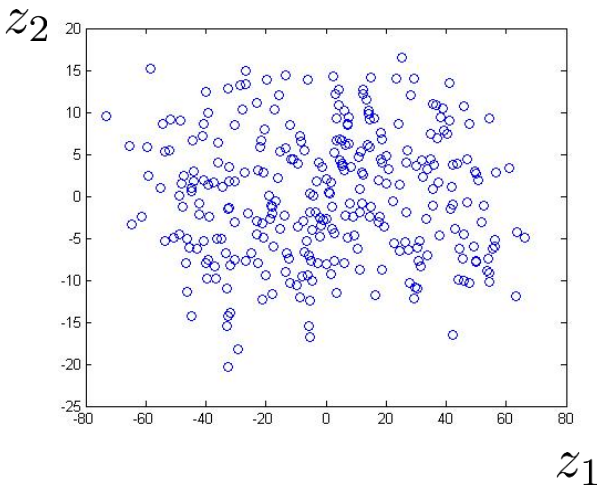
$$x^{(m)} \rightarrow z^{(m)}$$

Data Compression

Reduce data from 3D to 2D

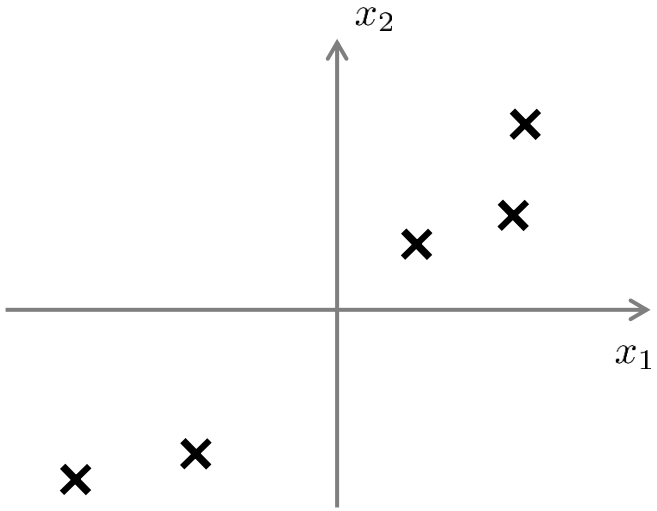


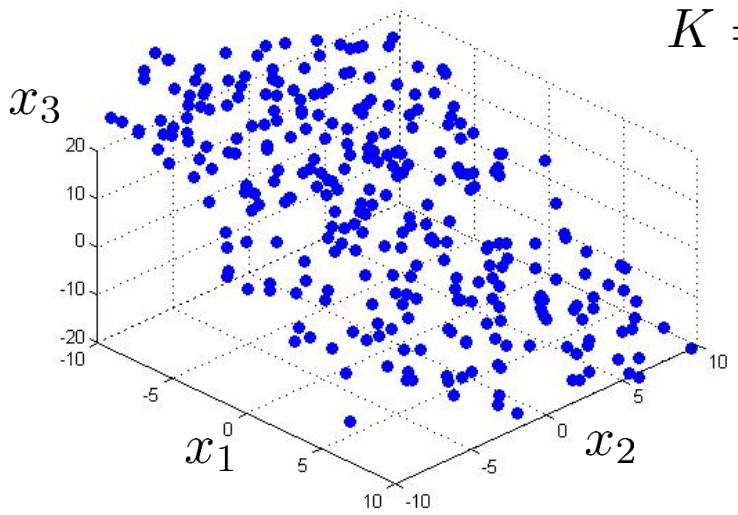




Principal Component Analysis (PCA) problem formulation

$3D \rightarrow 2D$
$K = 2$





Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

Principal Component Analysis

Goal: Find r-dim projection that best preserves variance

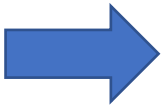
1. Compute mean vector μ and covariance matrix Σ of original points
2. Compute eigenvectors and eigenvalues of Σ
3. Select top r eigenvectors
4. Project points onto subspace spanned by them:

$$y = A(x - \mu)$$

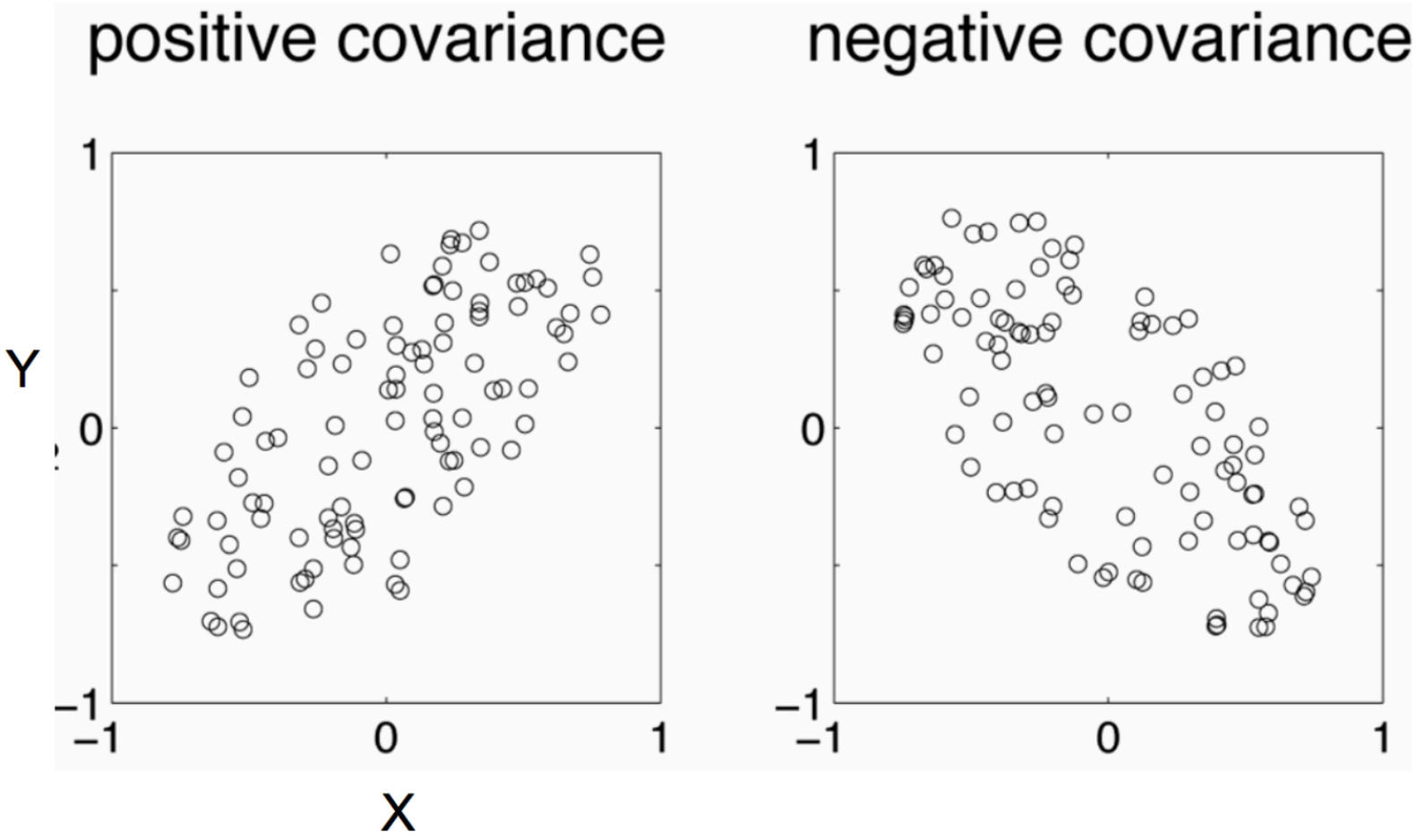
where y is the new point, x is the old one,
and the rows of A are the eigenvectors

Covariance

- Variance and Covariance:
  - Measure of the "spread" of a set of points around their center of mass(mean)
- Variance:
  - Measure of the deviation from the mean for points in one dimension
- Covariance:
  - Measure of how much each of the dimensions vary from the mean with **respect to each other**



- **Covariance is measured between two dimensions**
- **Covariance sees if there is a relation between two dimensions**
- **Covariance between one dimension is the variance**



Positive: Both dimensions increase or decrease together

Negative: While one increase the other decrease

Covariance

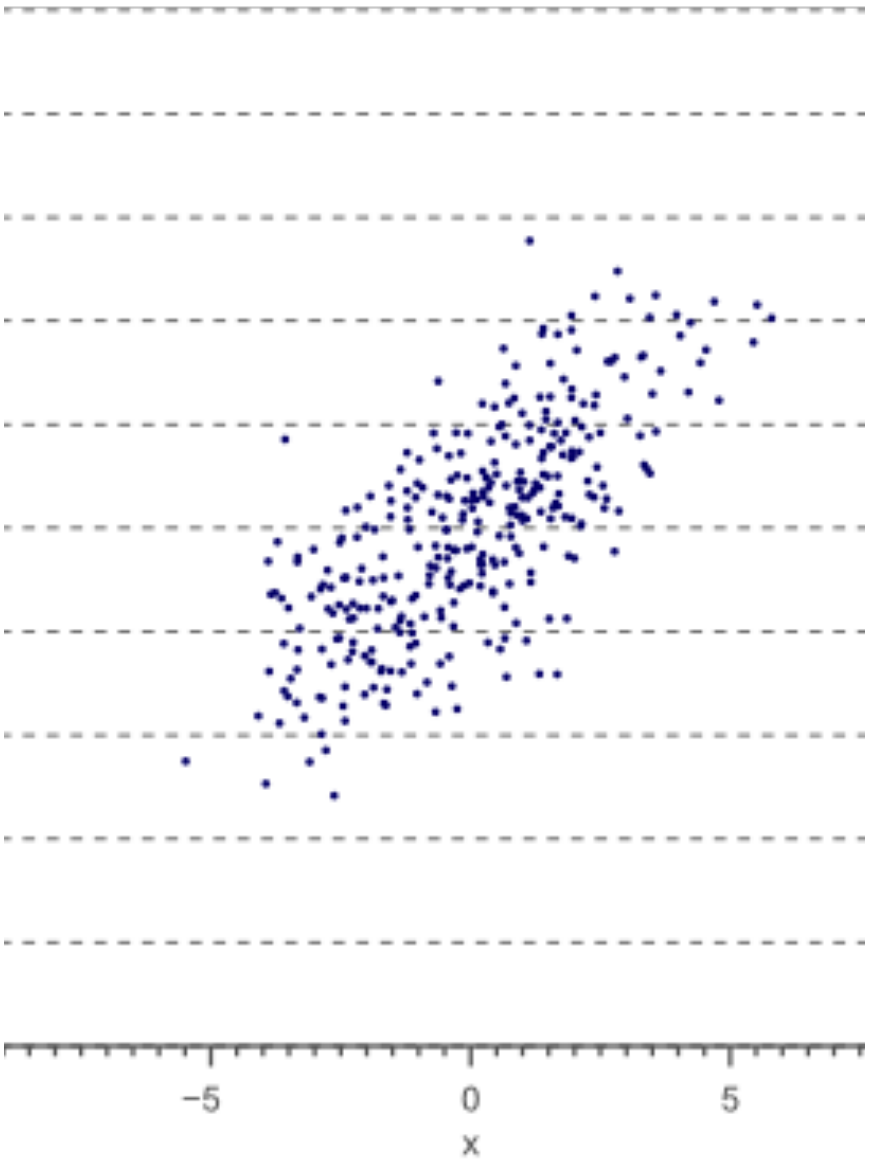
- Used to find relationships between dimensions in high dimensional data sets

$$q_{jk} = \frac{1}{N} \sum_{i=1}^{N} (X_{ij} - E(X_j))(X_{ik} - E(X_k))$$

The Sample mean

More on covariance

- For this data, we could calculate the variance var(x,x) in the x-direction and the variance var(y,y) in the y-direction. However, the horizontal spread and the vertical spread of the data does not explain the clear diagonal correlation. Figure clearly shows that on average, if the x-value of a data point increases, then also the y-value increases, resulting in a positive correlation.



Eigenvector and Eigenvalue

$$Ax = \lambda x$$

A: Square Matirx

λ: Eigenvalue or characteristic value

X: Eigenvector or characteristic vector

- ***The zero vector can not be an eigenvector***
- ***The value zero can be eigenvalue***

Eigenvector and Eigenvalue

$$\mathbf{Ax = \lambda x}$$

A: Square Matirx

λ: Eigenvector or characteristic vector

X: Eigenvalue or characteristic value

Example

$$Show\ x = \begin{bmatrix} 2 \\ 1 \end{bmatrix} is\ an\ eigenvector\ for\ A = \begin{bmatrix} 2 & -4 \\ 3 & -6 \end{bmatrix}$$

$$Solution: Ax = \begin{bmatrix} 2 & -4 \\ 3 & -6 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$But\ \ for\ \ \lambda = 0,\ \ \lambda x = 0\begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$Thus, x\ is\ an\ eigenvector\ of\ A, and\ \lambda = 0\ is\ an\ eigenvalue.$$

Eigenvector and Eigenvalue

$Ax = \lambda x$ → $Ax - \lambda x = 0$

$(A - \lambda I)x = 0$

If we define a new matrix B: → $B = A - \lambda I$

$Bx = 0$

If B has an inverse: → $x = B^{-1}0 = 0$ ✗ BUT! an eigenvector cannot be zero!!

→ x will be an eigenvector of A if and only if B does not have an inverse, or equivalently det(B)=0 :

$$\det(A - \lambda I) = 0$$

Eigenvector and Eigenvalue

Example 1: Find the eigenvalues of $A = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix}$

$$|\lambda I - A| = \begin{vmatrix} \lambda - 2 & 12 \\ -1 & \lambda + 5 \end{vmatrix} = (\lambda - 2)(\lambda + 5) + 12$$

$$= \lambda^2 + 3\lambda + 2 = (\lambda + 1)(\lambda + 2)$$

two eigenvalues: –1, – 2

Note: The roots of the characteristic equation can be repeated. That is, $\lambda_1 = \lambda_2 = \ldots = \lambda_k$. If that happens, the eigenvalue is said to be of multiplicity k.

Example 2: Find the eigenvalues of $A = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$

$$|\lambda I - A| = \begin{vmatrix} \lambda - 2 & -1 & 0 \\ 0 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 2 \end{vmatrix} = (\lambda - 2)^3 = 0$$

λ = 2 is an eigenvector of multiplicity 3.

Principal Component Analysis

Goal: Find r-dim projection that best preserves variance

1. Compute mean vector μ and covariance matrix Σ of original points
2. Compute eigenvectors and eigenvalues of Σ
3. Select top r eigenvectors
4. Project points onto subspace spanned by them:

$$y = A(x - \mu)$$

where y is the new point, x is the old one,
and the rows of A are the eigenvectors

Principal Component Analysis

Input: $\mathbf{x} \in \mathbb{R}^D$: $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

Set of basis vectors: $\mathbf{u}_1, \ldots, \mathbf{u}_K$

Summarize a *D* dimensional vector *X* with *K* dimensional feature vector *h(x)*

$$h(\mathbf{x}) = \begin{bmatrix} \mathbf{u}_1 \cdot \mathbf{x} \\ \mathbf{u}_2 \cdot \mathbf{x} \\ \cdots \\ \mathbf{u}_K \cdot \mathbf{x} \end{bmatrix}$$

Principal Component Analysis

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$$

Basis vectors are orthonormal

$$\mathbf{u}_i^T \mathbf{u}_j = 0$$
$$||\mathbf{u}_j|| = 1$$

New data representation *h(x)*

$$z_j = \mathbf{u}_j \cdot \mathbf{x}$$
$$h(\mathbf{x}) = [z_1, \ldots, z_K]^T$$

Principal Component Analysis

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$$

New data representation *h(x)*

$$h(\mathbf{x}) = \mathbf{U}^T \mathbf{x}$$

$$h(\mathbf{x}) = \mathbf{U}^T (\mathbf{x} - \mu_0)$$

Empirical mean of the data

$$\mu_0 = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

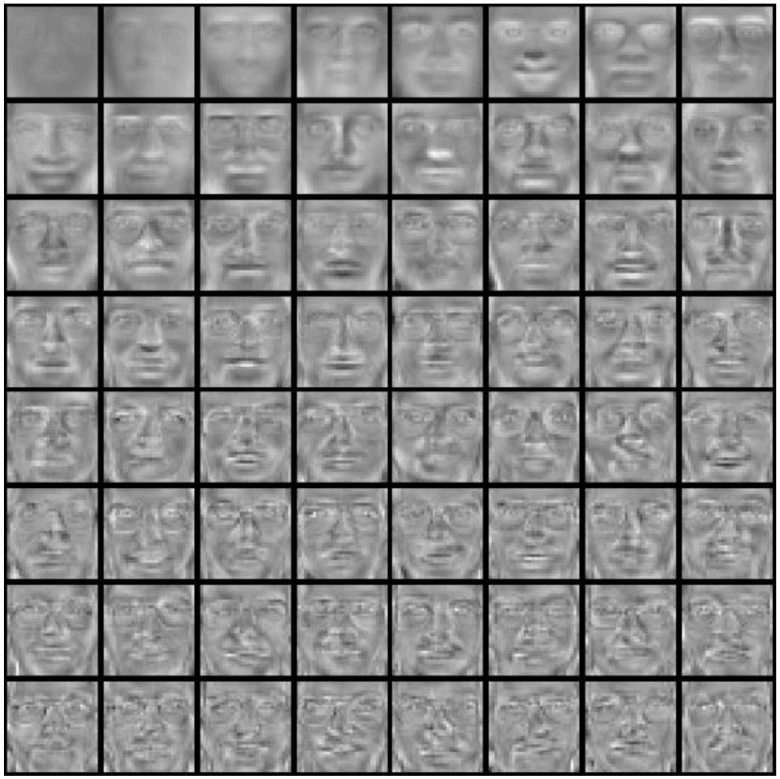
The space of all face images

- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image = 10,000 dimensions
  - Slow and lots of storage
- But very few 10,000-dimensional vectors are valid face images
- We want to effectively model the subspace of face images

Eigenfaces example

Mean: μ

Top eigenvectors: $u_1,\ldots u_k$



slide by Derek Hoiem

More about eigenfaces

- We have initially n number of images each having b*c pixels.
- The value of each pixel can be one feature. So, we represent each image in b*c dimensional space. We can call it d where d = b*c
- Now after applying PCA we get p principal components. Hence, we have p*d dimensional eigenvectors where d = b*c. We can plot these p principal components/eigenvectors. We call these eigenfaces.

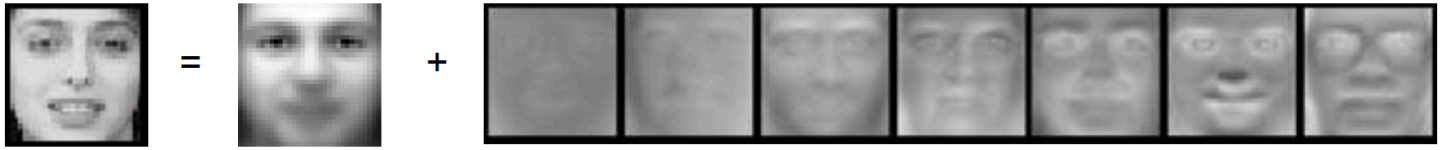
- We will do this in lab

Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x}-\mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x}-\mu)]$$
$$= \quad w_1, \ldots, w_k$$

- Reconstruction:



$\hat{x} = \mu + w_1u_1+w_2u_2+w_3u_3+w_4u_4+ \ldots$

Lecture note 1

2018

1. Input matrix is X which is n x d dimensional

2. From we first mean center X and get X'

$x' = X - \mu$

3. Calculate co-variance of X'. In matrix operation, it is $X'^T X$. Call it Z

$Z = X'^T X$ → verify yourself that this is co-variance

$X' \rightarrow n \times d$

$X'^T \rightarrow d \times n$

$Z \rightarrow d \times d$

4. Do eigendecomposition of Z. Use SVD to do it.

So, Z can be written as

$Z = P D P^{-1}$ | P → eigenvectors, D → diagonal matrix with eigenvalues on the diagonal

Lecture note 1

2018

1. Input matrix is X which is n x d dimensional

2. From we first mean center X and get X'

$x' = X - \mu$

3. Calculate co-variance of X'. In matrix operation, it is $X'^T X$. Call it Z

$Z = X'^T X$ → verify yourself that this is co-variance

$X' \rightarrow n \times d$

$X'^T \rightarrow d \times n$

$Z \rightarrow d \times d$

4. Do eigendecomposition of Z. Use SVD to do it.

So, Z can be written as

$Z = P D P^{-1}$ | P → eigenvectors, D → diagonal matrix with eigenvalues on the diagonal

Lecture note 3

2018

Now the new features will be—

$X_{new} = X' P^*$

$X' \rightarrow n \times d$

$P^* \rightarrow d \times r$

$X_{new} \rightarrow n \times r$

Reconstruction:

We can go back to the original feature space $n \times d$.

How?

Ans:- reverse the steps shown above

$X_{recon} = X_{new} P^{*T} + mean$

This refers to the figures in the slide

$X_{new} \rightarrow n \times r$

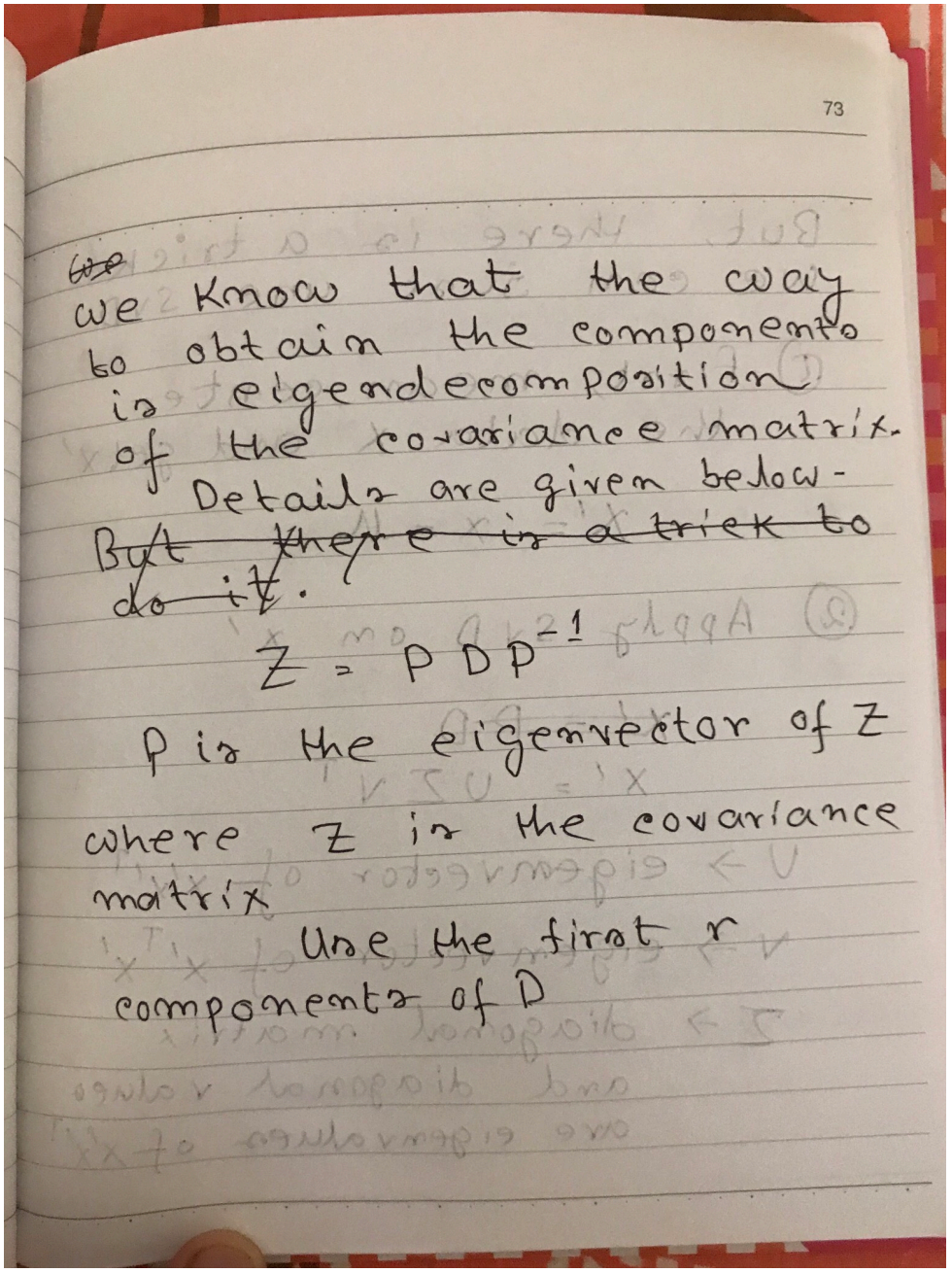
$P^* \rightarrow d \times r$

$P^{*T} \rightarrow r \times d$

$X_{recon} \rightarrow \frac{n \times d}{\text{original dimension}}$

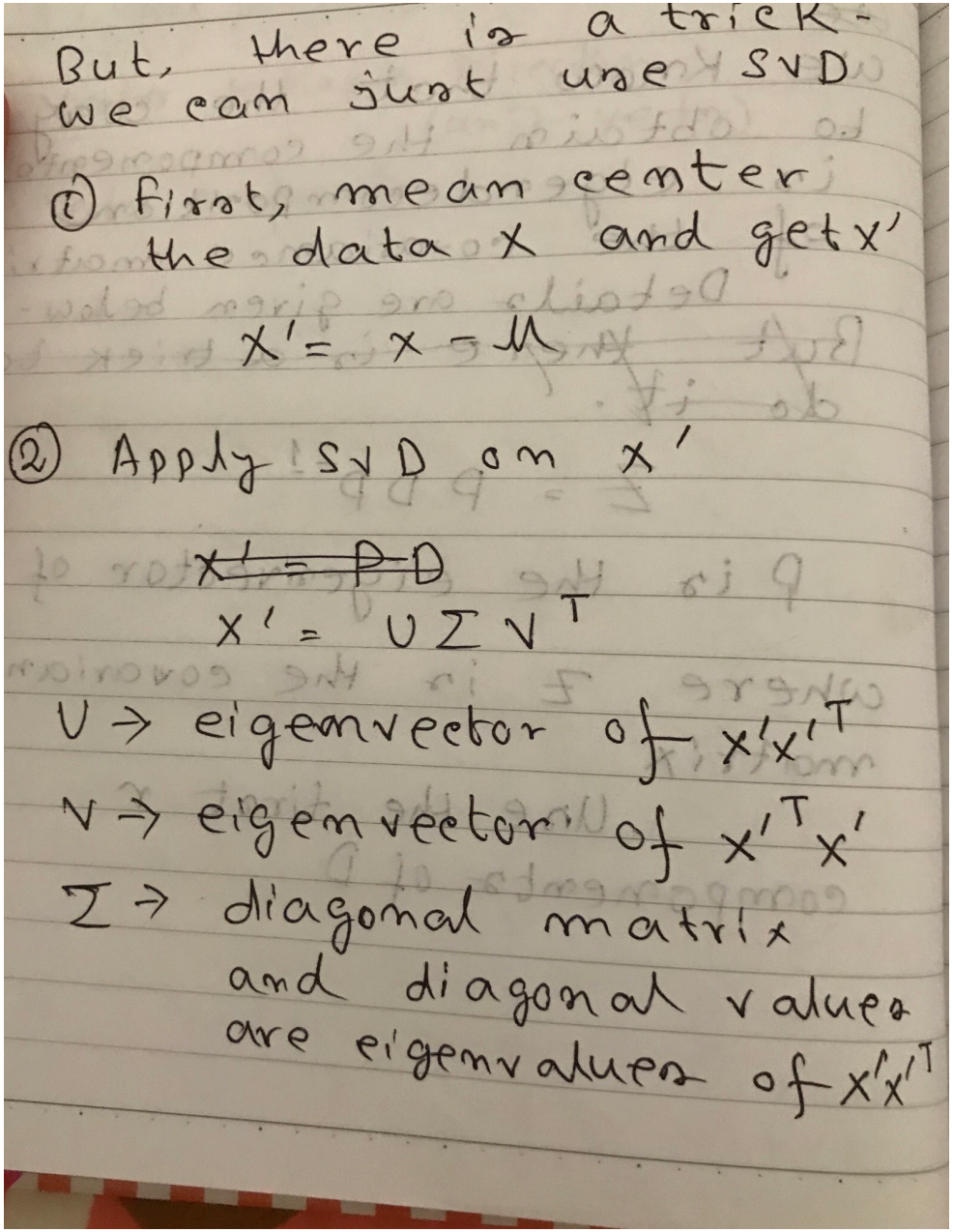
PCA using SVD

2018



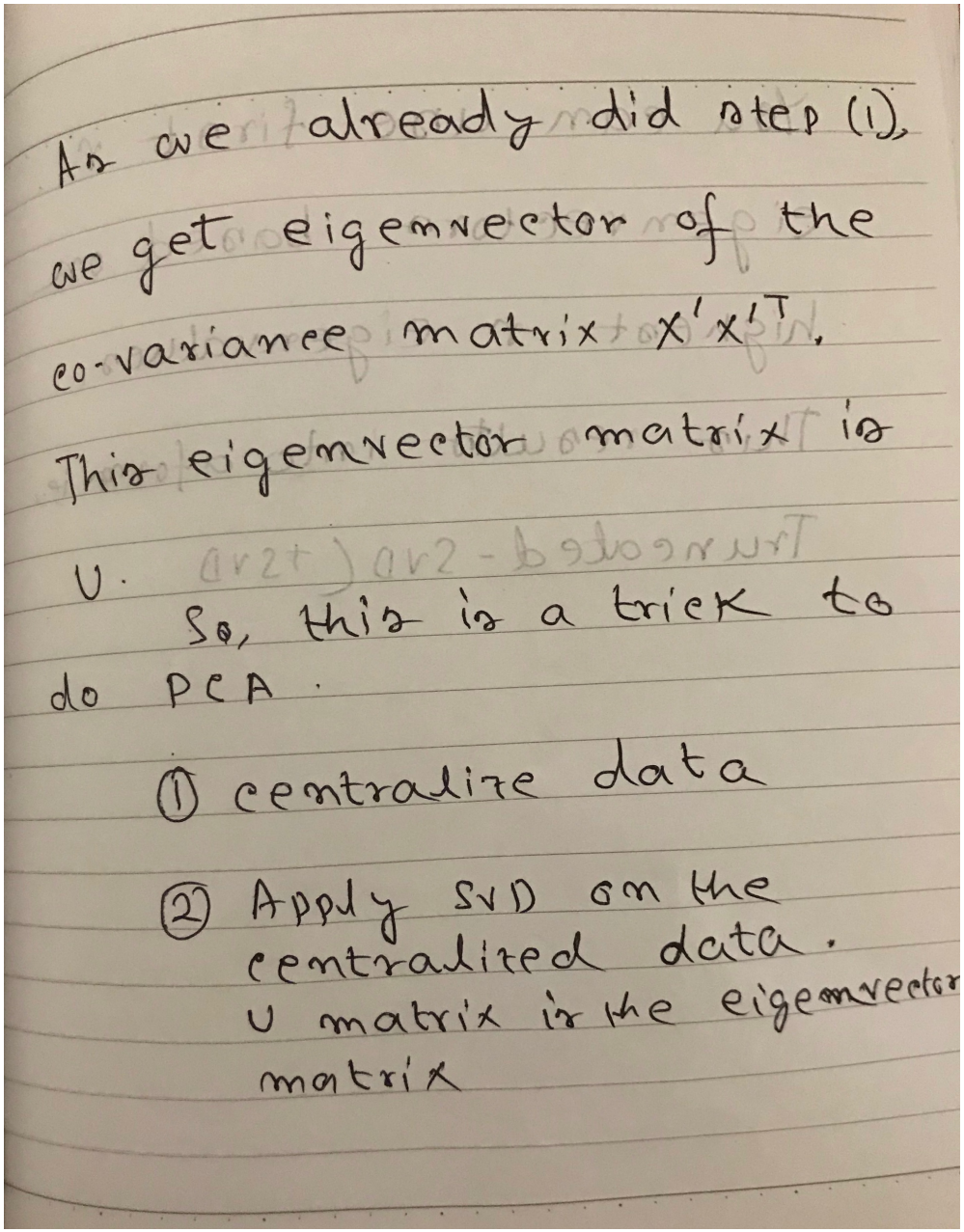
PCA using SVD

2018



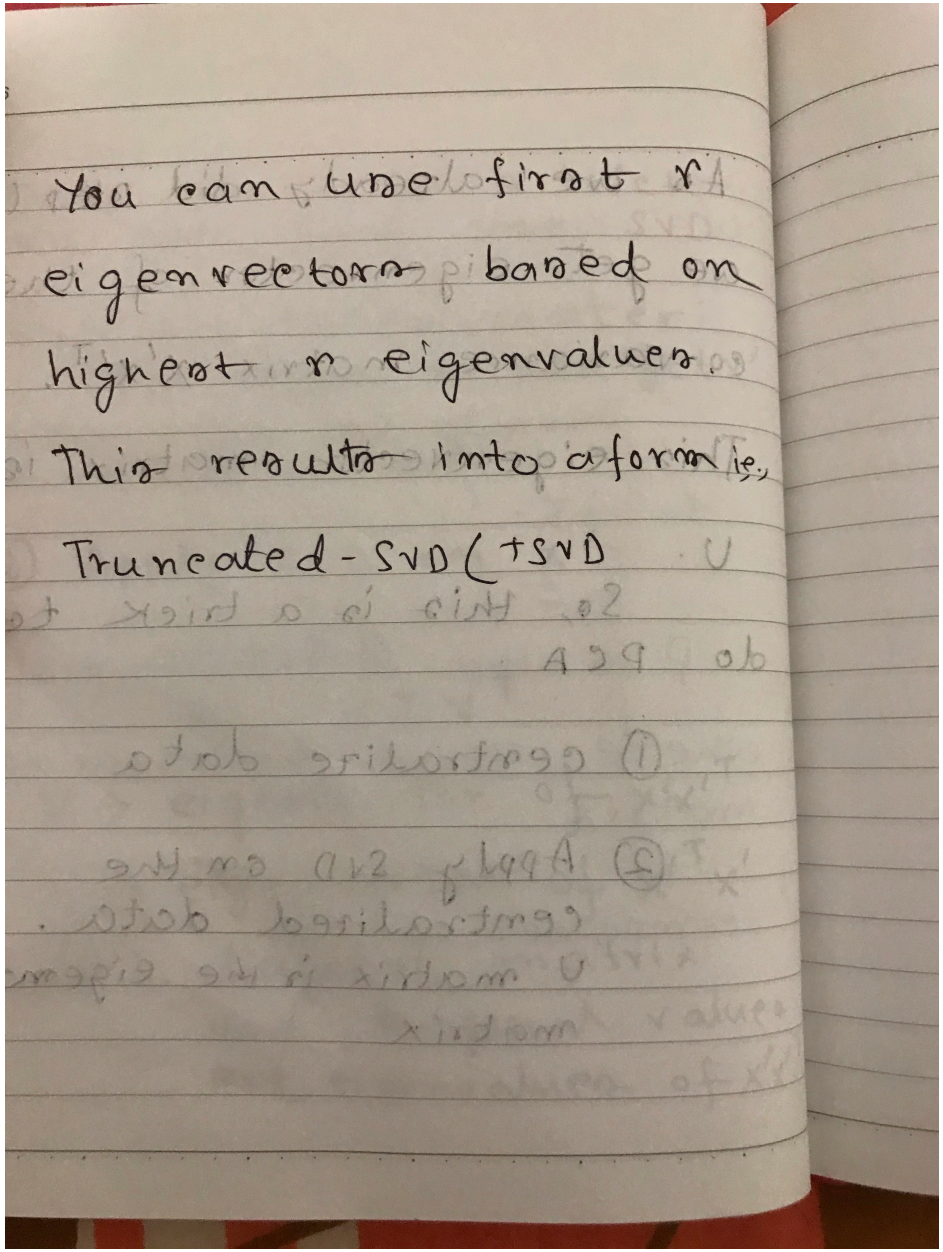
PCA using SVD

2018



PCA using SVD

2018



Reconstruction

P = 4

P = 200

P = 400

After computing eigenfaces using 400 face images from ORL face database

Reconstruction

- This reconstruction is blurred and **lossy** because we just use p principal components.
- If you use all the principal components then you can reconstruct the image without loosing any information.

- We will compute PCA and visualize in lab.

Application: Image compression

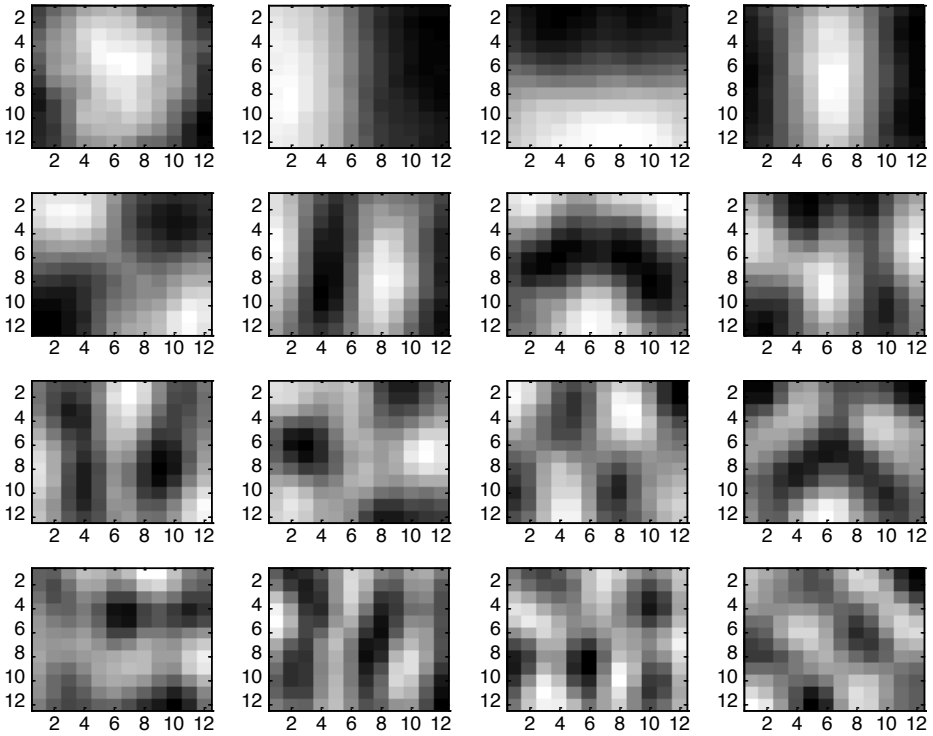
Original Image

- Divide the original 372x492 image into patches:
  - Each patch is an instance that contains 12x12 pixels on a grid
- View each as a 144-D vector

PCA compression: 144D → 60D

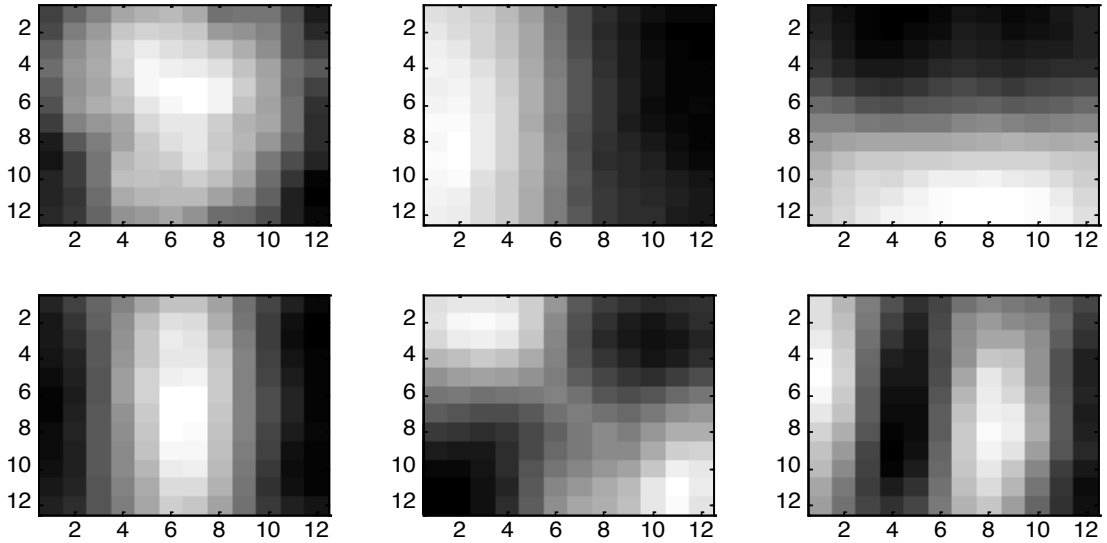
PCA compression: 144D → 16D

16 most important eigenvectors



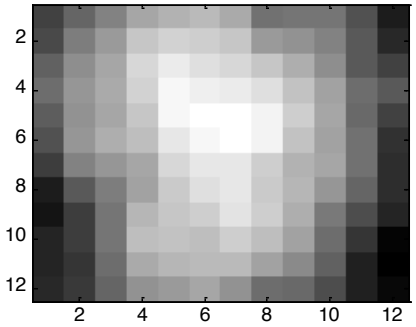
PCA compression: 144D → 6D

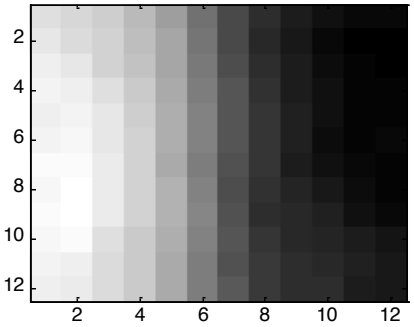
6 most important eigenvectors

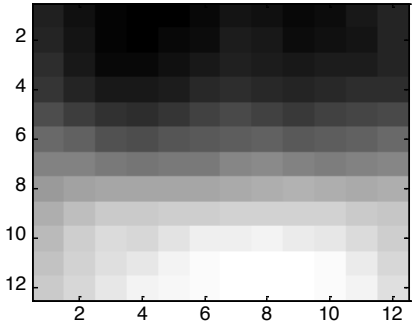


PCA compression: 144D → 3D

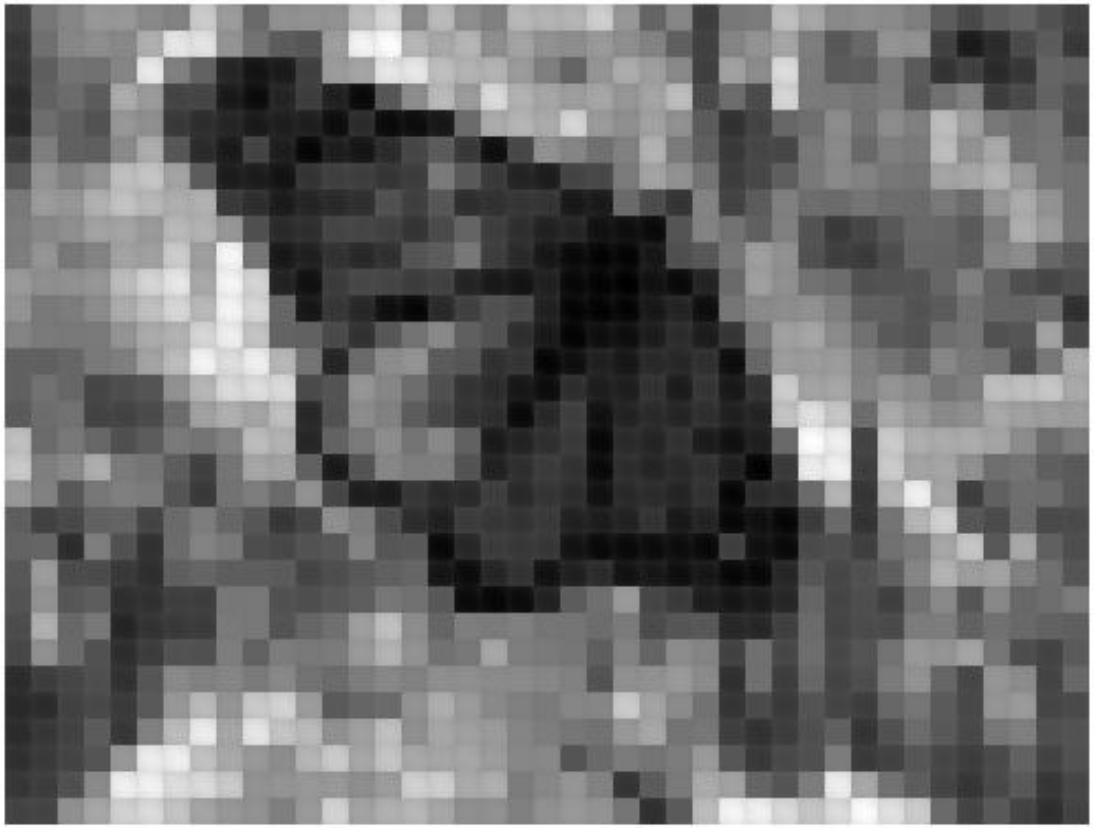
3 most important eigenvectors







PCA compression: 144D → 1D



Dimensionality reduction

- PCA (Principal Component Analysis):
  - Find projection that maximize the variance
- Multidimensional Scaling:
  - Find projection that best preserves inter-point distances
- LDA(Linear Discriminant Analysis):
  - Maximizing the component axes for class-separation
- …
- …

Feature Subset Selection

- Another way to reduce dimensionality of data
- Redundant features
  - Duplicate much information contained in one or more features
  - E.g., income level, CPF contributions and income tax
- Irrelevant features
  - Contain no useful information for the data science task at hand
  - E.g., NRIC for predicting a person's chances of falling sick
- Many techniques developed, especially for classification
  - Will explore some of them in the labs

Feature Selection Approaches

- Embedded Approaches
  - Feature selection embedded as part of the classification algorithm
  - E.g., Selection of features when building decision trees
- Filter Approaches
  - Independent feature selection process, before applying the algorithm
  - E.g., filtering features based on their correlation with class labels
- Wrapper Approaches
  - Search for best feature subset for a specific algorithm as a black box
  - E.g., recursive feature elimination

Filter VS Wrapper

- Compare and contrast between filter and wrapper approaches for feature selection.

Filter VS Wrapper

- Compare and contrast between filter and wrapper approaches for feature selection.

Filter Approach	Wrapper Approach
Not tagged to any algorithm	Tagged to a specific algorithm, but usually performs well for that algorithm
Less computationally expensive	Computationally expensive

Feature Creation

- Create new attributes that can capture the important information in a data set much more efficiently than the original attributes

- Three general methodologies:
  - Feature extraction
    - E.g., extracting edges from images
  - Feature construction
    - E.g., dividing mass by volume to get density
  - Mapping data to new space
    - E.g., Fourier and wavelet analysis

Discretization

- Converting a continuous attribute into an ordinal attribute
  - A potentially infinite number of values are mapped into a small number of categories
  - Many classification algorithms work best if both the independent and dependent variables have only a few values
  - E.g., instead of using (continuous) total savings, we have (discrete) savings level like low, medium, high

Person	Savings
A	100k
B	120k
C	200k
D	210k
E	20k

Person	Savings
A	Mid
B	Mid
C	High
D	High
E	Low

Binarization

- Mapping a categorical attribute into one or more binary variables
- Typically used for association analysis
  - E.g., market basket analysis: buy broccoli and oil → buy garlic

Person	Item
Tom	Fish
Tom	Milk
Mary	Fish
Mary	Chicken
John	Fruit

Person	Fish	Milk	Chicken	Fruit
Tom	1	1	0	0
Mary	1	0	1	0
John	0	0	0	1

Text processing

From Text to Tokens to Terms

- **Tokenization** = segmenting text into tokens:
- **token** = a sequence of characters, in a particular document at a particular position.

Simple Tokenization

- Analyze text into a sequence of discrete tokens (words).
- Sometimes punctuation (e-mail), numbers (1999), and case (Republican vs. republican) can be a meaningful part of a token.
  - However, frequently they are not.
- Simplest approach is to ignore all numbers and punctuation and use only case-insensitive unbroken strings of alphabetic characters as tokens.
- More careful approach:
  - Separate ? ! ; : “ ‘ [ ] ( ) < >
  - Care with . - why? when?
  - Care with … ??

Tokenization

- Apostrophes are ambiguous:
  - possessive constructions:
    - the book's cover => the book s cover
  - contractions:
    - he's happy => he is happy
    - aren't => are not
  - quotations:
    - 'let it be' => let it be
- Whitespaces in proper names or collocations:
  - San Francisco => San_Francisco
    - how do we determine it should be a single token?

Tokenization

- Hyphenations:
  - co-education => co-education
  - state-of-the-art => state of the art? state_of_the_art?
  - lowercase, lower-case, lower case => lower_case
  - Hewlett-Packard => Hewlett_Packard? Hewlett Packard?
- Period
  - Abbreviations: Mr., Dr.
  - Acronyms: U.S.A.
  - File names: a.out

Tokenization

- Numbers
  - 3/12/91
  - Mar. 12, 1991
  - 55 B.C.
  - B-52
  - 100.2.86.144
- Unusual strings that should be recognized as tokens:
    - C++, C#, B-52, C4.5,M*A*S*H.

Tokenization

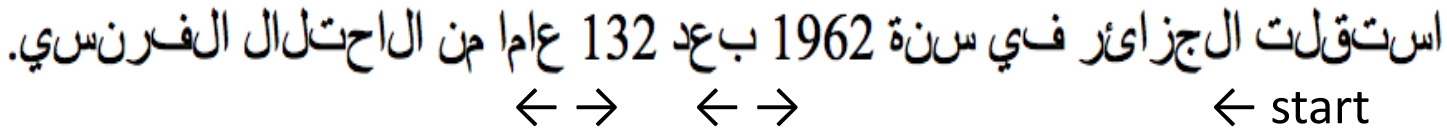
- Tokenizing HTML
  - Should text in HTML tags not typically seen by the user be included as tokens?
    - Words appearing in URLs.
    - Words appearing in “meta text” of images.
  - Simplest approach is to exclude all HTML tag information (between “<“ and “>”) from tokenization.

Tokenization is Language Dependent

- Need to know the language of the document/query:
  - **Language Identification**, based on classifiers trained on short character subsequences as features, is highly effective.
- French (reduced definite article, postposed clitic, pronouns):
    - l'ensemble, un ensemble, donne-moi.
- German (compound nouns), need *compound splitter*:
    - Computerlinguistik
    - Lebensversicherungsgesellschaftsangestellte*r*
    - *(*life insurance company employee)
  - Compound Splitting for German:
    - usually implemented by finding segments that match against dictionary entries.

Tokenization is Language Dependent

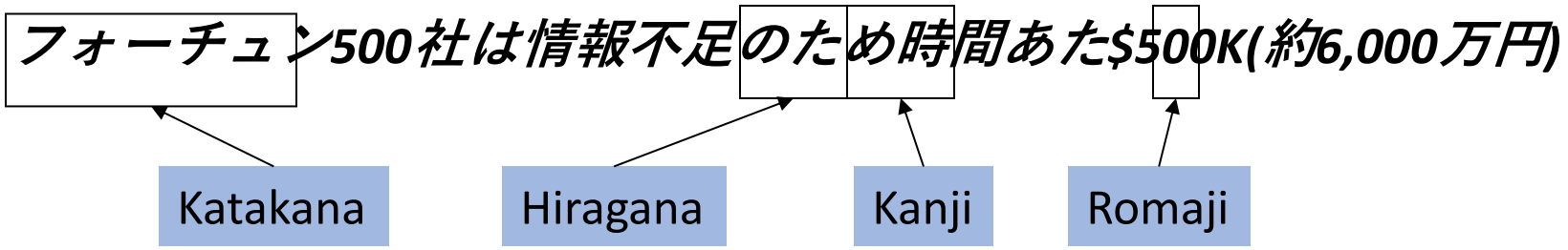
- Arabic and Hebrew:
  - Written right to left, but with certain items like numbers written left to right.
  - Words are separated, but letter forms within a word form complex ligatures



Algeria achieved its independence in 1962 after 132 years of French occupation.

Tokenization is Language Dependent

- Chinese and Japanese have no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats



End-user can express query entirely in hiragana!

Word Tokenization in Chinese

- Also called **Word Segmentation**
- Chinese words are composed of characters
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:
  - Maximum Matching (also called Greedy)

Stopwords

- It is typical to *exclude* high-frequency words (e.g., function words: “a”, “the”, “in”, “to”; pronouns: “I”, “he”, “she”, “it”).
- Stopwords are language dependent
- For efficiency, store strings for stopwords in a hashtable to recognize them in constant time.
  - E.g., simple Python dictionary

Normalization

- **Token Normalization** = reducing multiple tokens to the same canonical term, such that matches occur despite superficial differences.
- Before: tokens stored and indexed in a normalized form
- Now:
  1. Create equivalence classes, named after one member of the class:
     - {anti-discriminatory, antidiscriminatory}
     - {U.S.A., USA}
  2. Maintain relations between unnormalized tokens:
     - can be extended with lists of synonyms (car, automobile).
       1. Index unnormalized tokens, a query term is expanded into a disjunction of multiple postings lists.
       2. Perform expansion during index construction.

Normalization

- Accents and diacritics in French:
  - ***résumé*** vs. ***resume*.**
- Umlauts in German:
  - ***Tuebingen*** vs. ***Tübingen***

- Most important criterion:
  - How do users like to write their queries for these words?
    - Even in languages that standardly have accents, users often may not type them:
    - Often best to normalize to a de-accented term
      - ***Tuebingen, Tübingen, Tubingen*** => ***Tubingen***

Normalization

- **Case-Folding** = reduce all letters to lower case:
  - allow Automobile at beginning of sentences to match automobile.
  - allow user-typed ferrari to match Ferrari in documents.
  - but may lead to unintended matches:
    - the Fed vs. fed.
    - Bush, Black, General Motors, Associated Press, ...
- **Heuristic** = lowercase only some tokens:
  - words at beginning of sentences.
  - all words in a title where most words are capitalized.
- **Truecasing** = use a classifier to decide when to fold:
  - trained on many heuristic features.

Normalization

- British vs. American spellings:
  - colour vs. color.
- Multiple formats for dates, times:
  - 09/30/2013 vs. Sep 30, 2013.
- Asymmetric expansion:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***

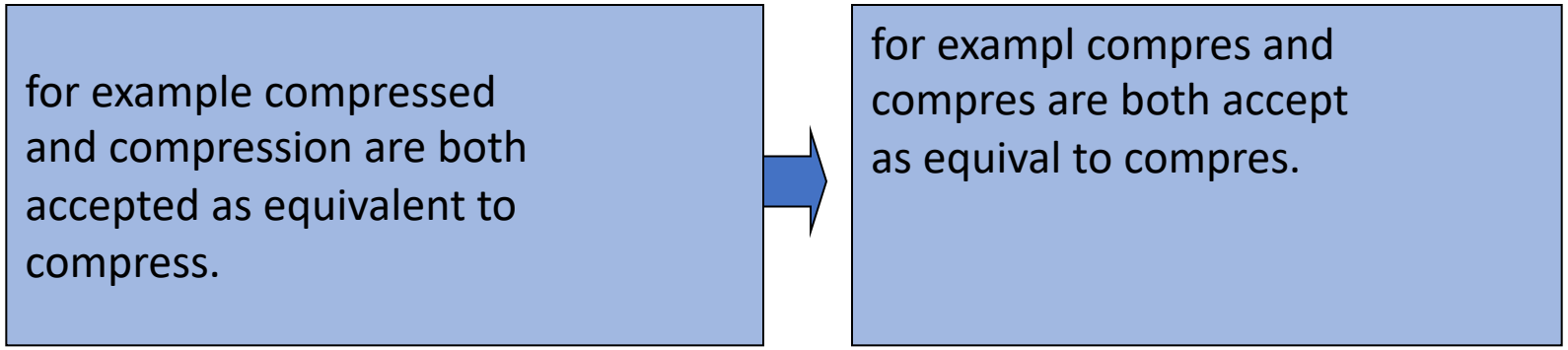
Lemmatization

- Reduce inflectional/variant forms to base form
- Direct impact on vocabulary size
- E.g.,
  - *am, are, is* $\rightarrow$ *be*
  - *car, cars, car's, cars'* $\rightarrow$ *car*
- *the boy's cars are different colors* $\rightarrow$ *the boy car be different color*

- How to do this?
  - Need a list of grammatical rules + a list of irregular words
  - Children $\rightarrow$ child, spoken $\rightarrow$ speak …
  - Practical implementation: use WordNet's morphstr function
    - Python: NLTK.stem

Stemming

- Reduce tokens to "root" form of words to recognize morphological variation
  - "computer", "computational", "computation" all reduced to same token "compute"
- Correct morphological analysis is language specific and can be complex
- Stemming "blindly" strips off known affixes (prefixes and suffixes) in an iterative fashion
- Stemming was frequently used in IR



Porter Stemmer

- Simple procedure for removing known affixes in English without using a dictionary
- Can produce unusual stems that are not English words:
  - “computer”, “computational”, “computation” all reduced to same token “comput”
- May conflate (reduce to the same token) words that are actually distinct
- Does not recognize all morphological derivations

Typical rules in Porter

- *sses* $\rightarrow$ *ss*
- *ies* $\rightarrow$ *i*
- *ational* $\rightarrow$ *ate*
- *tional* $\rightarrow$ *tion*

- Official Porter stemmer website https://tartarus.org/martin/PorterStemmer/
  - Provides Python ready to use implementations

Porter Stemmer Errors

- Errors of “comission”:
  - organization, organ $\rightarrow$ organ
  - police, policy $\rightarrow$ polic
  - arm, army $\rightarrow$ arm
- Errors of “omission”:
  - cylinder, cylindrical
  - create, creation
  - Europe, European

Other stemmers

- Other stemmers exist, e.g., Lovins stemmer
  - http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm
  - Single-pass, longest suffix removal (about 250 rules)
- Stemming is language- and often application-specific:
  - open source and commercial plug-ins.
- Does it improve IR performance?
  - mixed results for English: improves recall, but hurts precision.
    - operative (dentistry) ⇒ oper
  - definitely useful for languages with richer morphology:
    - Spanish, German, Finish (30% gains).

Text representation

Natural Language Processing

Some basic terms:

- Syntax: the allowable structures in the language: sentences, phrases, affixes (-ing, -ed, -ment, etc.).
- Semantics: the meaning(s) of texts in the language.
- Part-of-Speech (POS): the category of a word (noun, verb, preposition etc.).
- Bag-of-words (BoW): a featurization that uses a vector of word counts (or binary) ignoring order.
- N-gram: for a fixed, small N (2-5 is common), an n-gram is a consecutive sequence of words in a text.

Bag of words Featurization

Assuming we have a dictionary mapping words to a unique integer id, a bag-of-words featurization of a sentence could look like this:

Sentence: The cat sat on the mat

word id's: 1 12 5 3 1 14

The BoW featurization would be the vector:

Vector 2, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

position 1 3 5 12 14

In practice this would be stored as a sparse vector of (id, count)s:

(1,2),(3,1),(5,1),(12,1),(14,1)

Note that the original word order is lost, replaced by the order of id's.

Document Collection

- A collection of *n* documents can be represented in the vector space model by a term-document matrix.
- An entry in the matrix corresponds to the "weight" of a term in the document; zero means the term has no significance in the document or it simply doesn't exist in the document.

	T_1	T_2	....	T_t
D_1	w_{11}	w_{21}	...	w_{t1}
D_2	w_{12}	w_{22}	...	w_{t2}
:	:	:		:
:	:	:		:
D_n	w_{1n}	w_{2n}	...	w_{tn}

N-grams

Because word order is lost, the sentence meaning is weakened. This sentence has quite a different meaning but the same BoW vector:

Sentence:	The	mat	sat	on	the	cat
word id s:	1	14	5	3	1	12

BoW featurization:

Vector 2, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

But word order is important, especially the order of nearby words.

N-grams capture this, by modeling tuples of consecutive words.

N-grams

Sentence: The cat sat on the mat

2-grams: the-cat, cat-sat, sat-on, on-the, the-mat

Notice how even these short n-grams "make sense" as linguistic units. For the other sentence we would have different features:

Sentence: The mat sat on the cat

2-grams: the-mat, mat-sat, sat-on, on-the, the-cat

We can go still further and construct 3-grams:

Sentence: The cat sat on the mat

3-grams: the-cat-sat, cat-sat-on, sat-on-the, on-the-mat

Which capture still more of the meaning:

Sentence: The mat sat on the cat

3-grams: the-mat-sat, mat-sat-on, sat-on-the, on-the-cat

N-grams Features

Typically, its advantages to use multiple n-gram features in machine learning models with text, e.g.

unigrams + bigrams (2-grams) + trigrams (3-grams).

The unigrams have higher counts and are able to detect influences that are weak, while bigrams and trigrams capture strong influences that are more specific.

e.g. “the white house” will generally have very different influences from the sum of influences of “the”, “white”, “house”.

N-grams size

N-grams pose some challenges in feature set size.

If the original vocabulary size is $|V|$, the number of 2-grams is $|V|^2$

While for 3-grams it is $|V|^3$

Luckily natural language n-grams (including single words) have a **power law** frequency structure. This means that most of the n-grams you see are common. A dictionary that contains the most common n-grams will cover most of the n-grams you see.

N-grams size

Because of this you may see values like this:

- Unigram dictionary size: 40,000
- Bigram dictionary size: 100,000
- Trigram dictionary size: 300,000

With coverage of > 80% of the features occurring in the text.

N-gram Language Models

N-grams can be used to build statistical models of texts.

When this is done, they are called n-gram language models.

An n-gram language model associates a probability with each n-gram, such that the sum over all n-grams (for fixed n) is 1.

You can then determine the overall likelihood of a particular sentence:

The cat sat on the mat

Is much more likely than

The mat sat on the cat

Skip-grams

We can also analyze the meaning of a particular word by looking at the contexts in which it occurs.

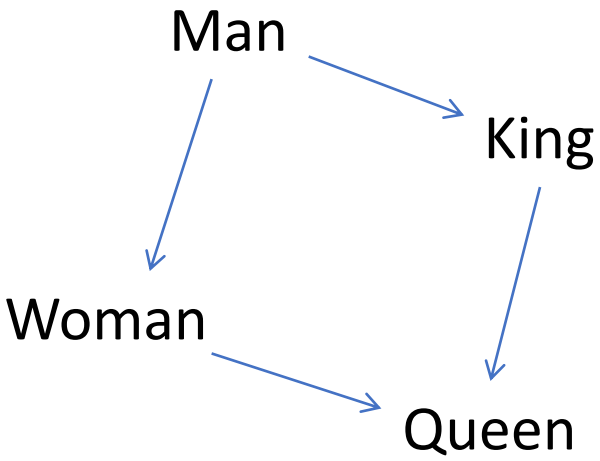
The context is the set of words that occur near the word, i.e. at displacements of …,-3,-2,-1,+1,+2,+3,… in each sentence where the word occurs.

A skip-gram is a set of non-consecutive words (with specified offset), that occur in some sentence.

We can construct a BoSG (bag of skip-gram) representation for each word from the skip-gram table.

Skip-grams

Then with a suitable embedding (DNN or linear projection) of the skip-gram features, we find that word meaning has an algebraic structure:



Man + (King – Man) + (Woman – Man) = Queen

Tomáš Mikolov et al. (2013). "Efficient Estimation of Word Representations in Vector Space"

Term Weights: Term Frequency

- More frequent terms in a document are more important, i.e. more indicative of the topic.

  f_{ij} = frequency of term *i* in document *j*

- May want to normalize *term frequency* (*tf*) by dividing by the frequency of the most common term in the document:

  $tf_{ij} = f_{ij} / max_i\{f_{ij}\}$

Term Weights: Inverse Document Frequency

- Terms that appear in many *different* documents are *less* indicative of overall topic.

  df_i = document frequency of term i

  = number of documents containing term i

  idf_i = inverse document frequency of term i,

  = $\log_2 (N/ df_i)$

  (N: total number of documents)

- An indication of a term's *discrimination* power.
- Log used to dampen the effect relative to *tf*.

TF-IDF Weighting

- A typical combined term importance indicator is *tf-idf weighting*:

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij} \log_2 (N/ df_i)$$

- A term occurring frequently in the document but rarely in the rest of the collection is given high weight.
- Many other ways of determining term weights have been proposed.
- Experimentally, *tf-idf* has been found to work well.

Exercise 1

- **Exercise #1**
- Modify the cities table by adding a new boolean column that is True if and only if *both* of the following are True:
- The city is named after a saint.
- The city has an area greater than 50 square miles.
- **Note:** Boolean Series are combined using the bitwise, rather than the traditional boolean, operators. For example, when performing *logical and*, use & instead of and.
- **Hint:** "San" in Spanish means "saint."

Exercise 2

Create 2 dataframes in pandas based on the information below and calculate tf-idf score for each word in document d1 and d2. The function you try to design is tfidf(w,d) where w can be any word in the vocabulary and d is either d1 and d2.

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

TF-IDF computation

2018

Tf-idf computation

$tf("this", d_1) = \frac{1}{2}$

"a" is the most common word in d_1 and

#('#('a', d_1) = 2

Another way to normalize is to divide by the count of all words in the document.

In that case,

$tf("this", d_1) = \frac{1}{5}$

$tf("this", d_2) = \frac{1}{3}$

$idf("this", D) = \log\left(\frac{2}{2}\right)$

$= \log 1$

$= 0$

$tfidf("this", d_1, D) = 0.5 \times 0 = 0$

$tfidf("this", d_2, D) = 0.34 \times 0 = 0$

Similarly,

tfidf(

TF-IDF computation

2018

Similarly,

$tf(\text{"example"}, d_1) = \frac{0}{2} = 0$

$tf(\text{"example"}, d_2) = \frac{3}{3} = 1$

$idf(\text{"example"}, D) = \log\left(\frac{2}{1}\right)$

$= \log 2$

$= 0.301$

$tfidf(\text{"example"}, d_1, D) = 0 \times 0.301$

$= 0$

$tfidf(\text{"example"}, d_2, D) = 1 \times 0.301$

$= 0.301$