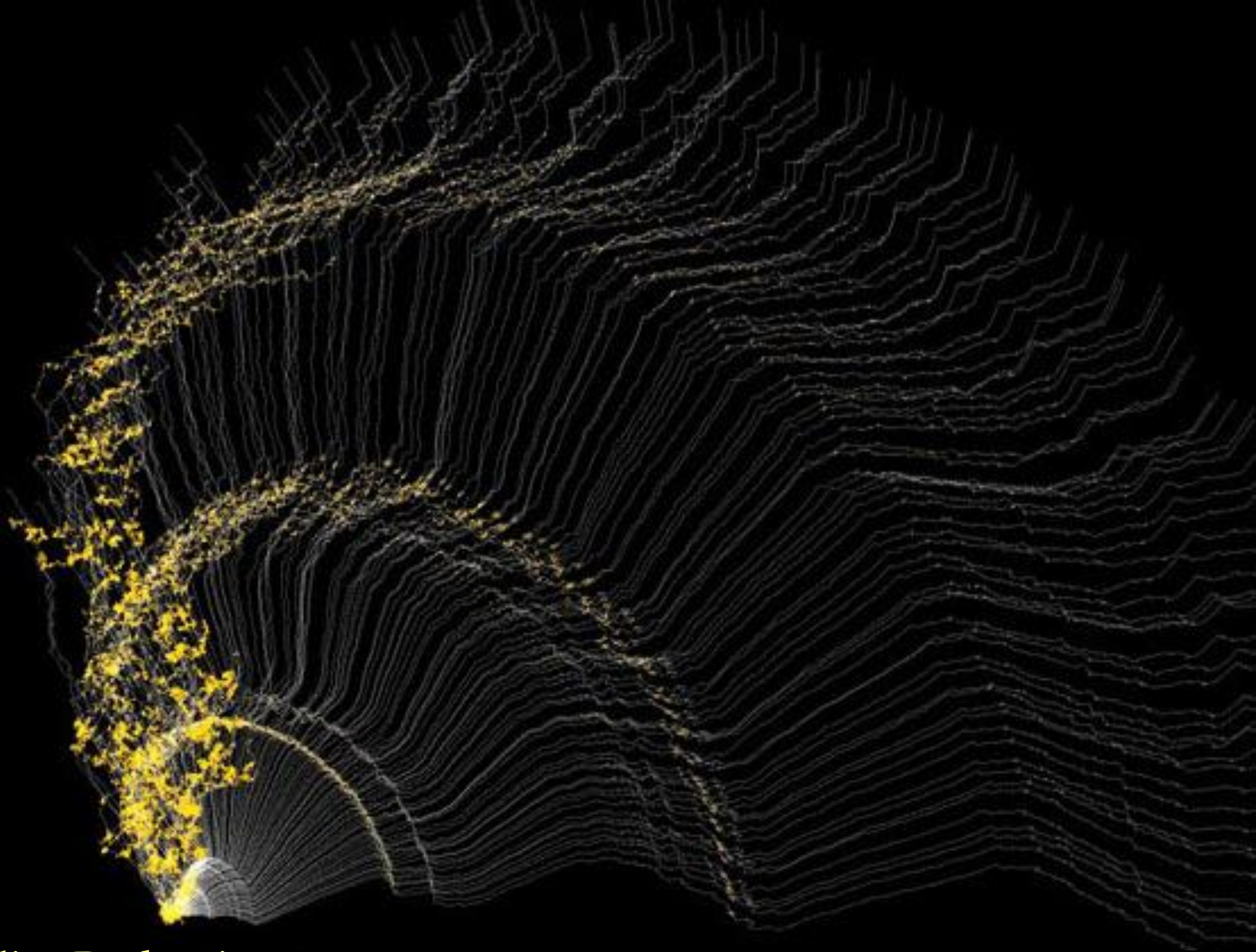


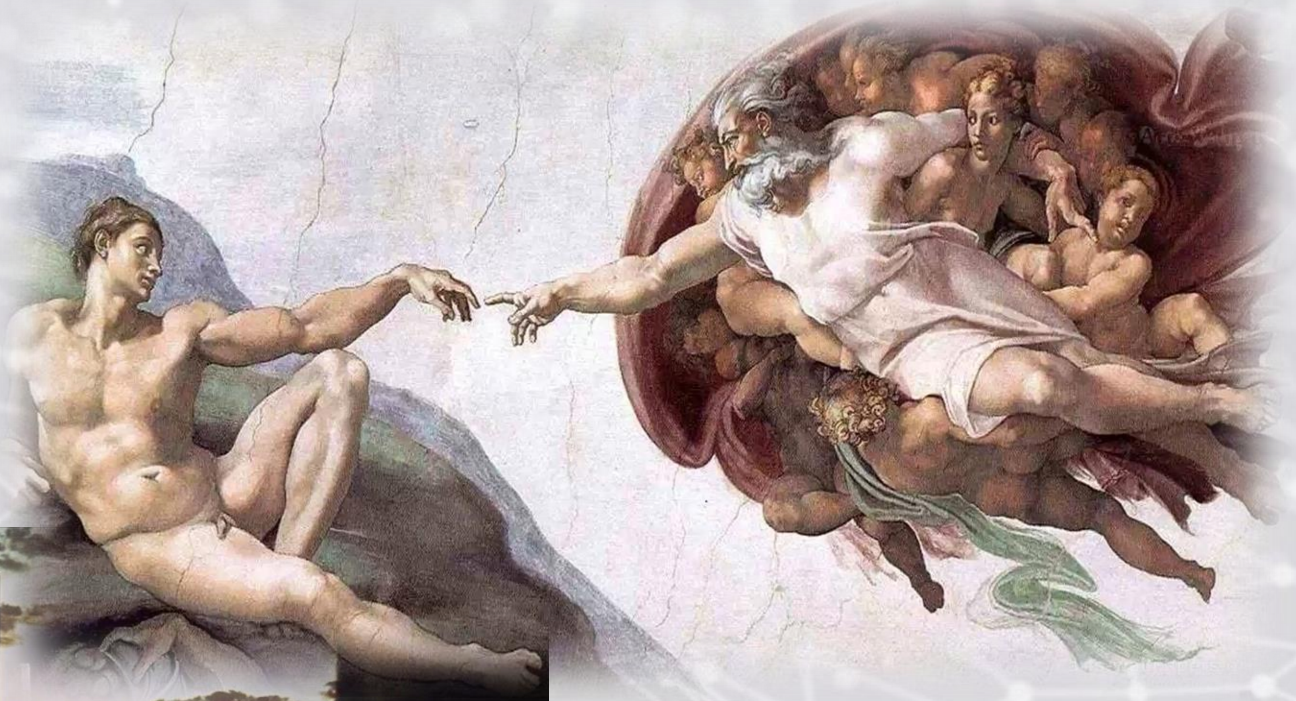
Dimensionality Reduction:
CS 445/545

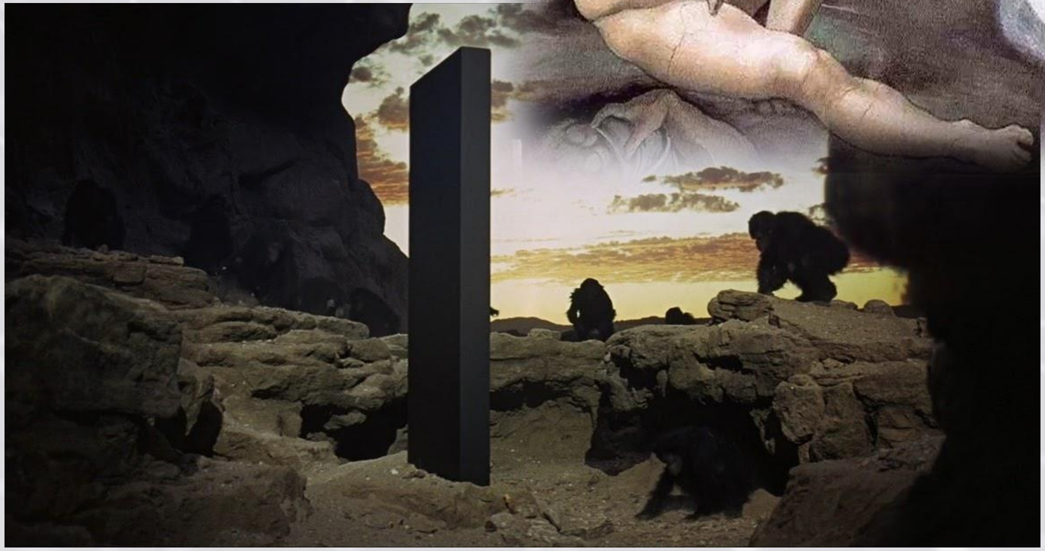
Outline

- "Big data" and motives for dimensionality reduction
- **LDA** (linear discriminant analysis)
- **PCA** (principal component analysis)
- **SVD** (singular value decomposition)

Big Data/High-Dimensional Data

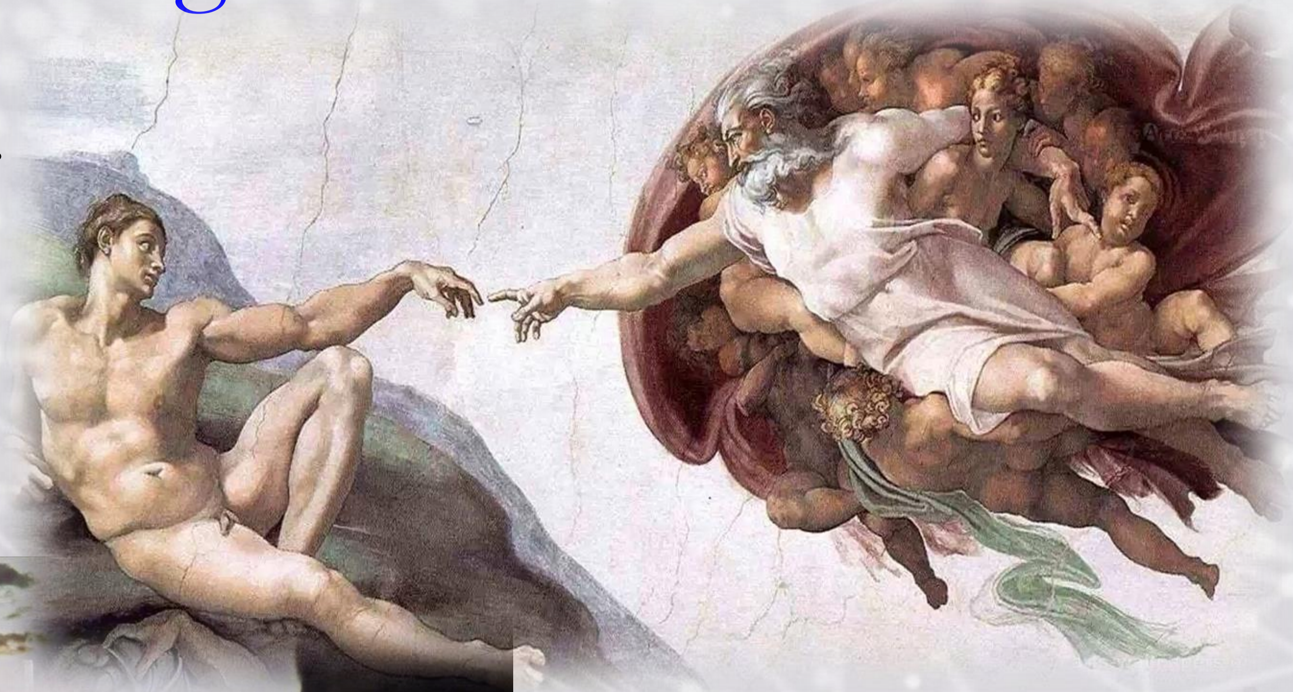
In the beginning…

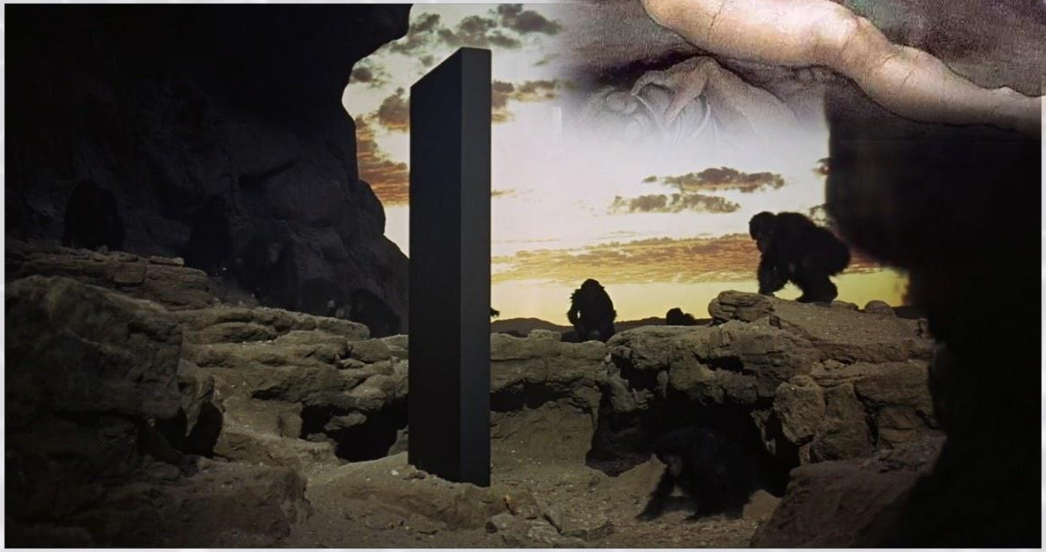




Big Data/High-Dimensional Data

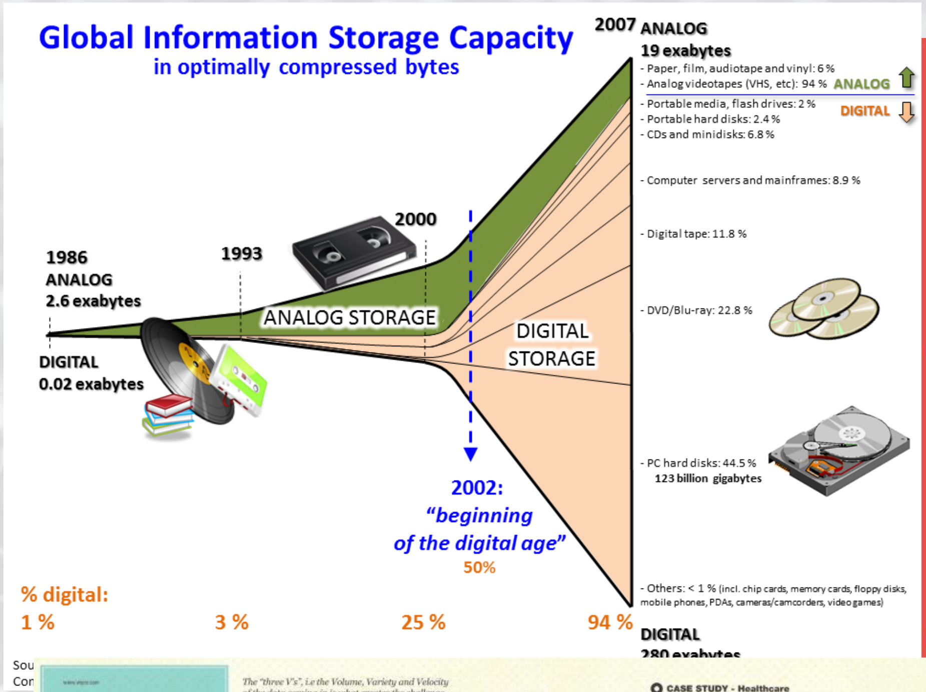
In the beginning…

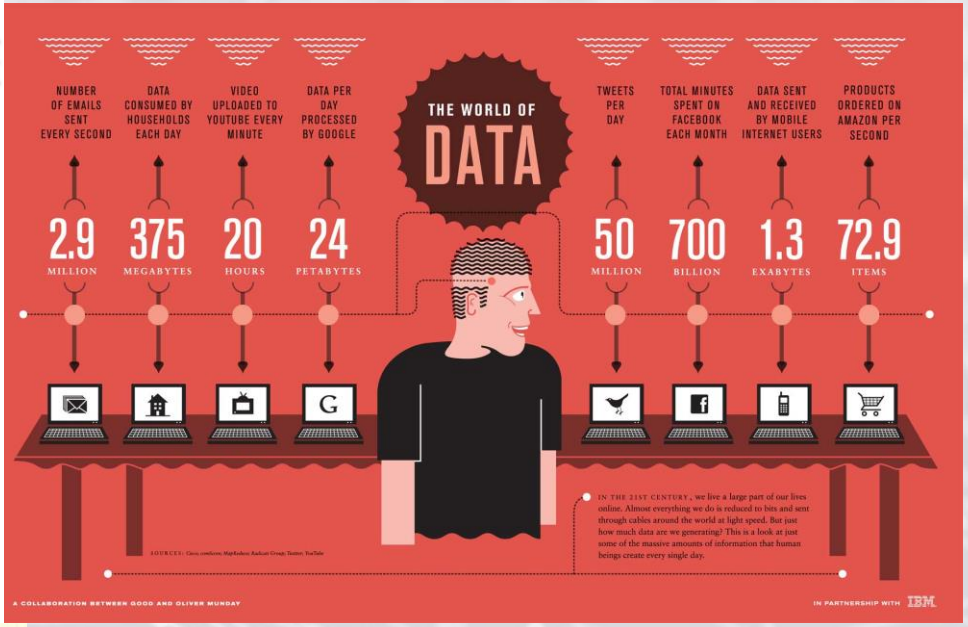


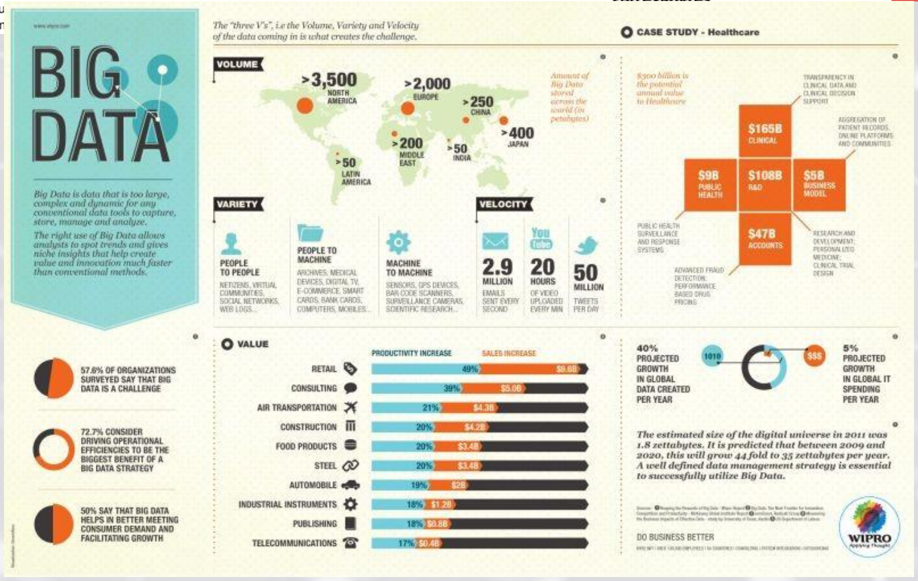


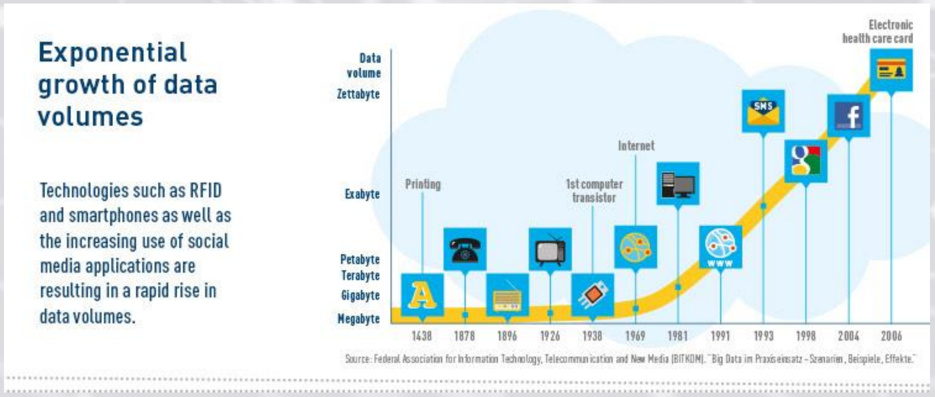
There was *small data*…

Big Data/High-Dimensional Data

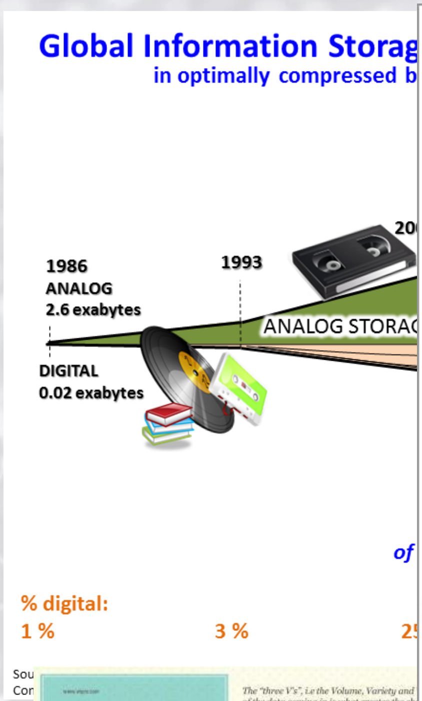



THE WORLD OF DATA
NUMBER OF EMAILS SENT EVERY SECOND
2.9 MILLION
DATA CONSUMED BY HOUSEHOLDS EACH DAY
375 MEGABYTES
VIDEO UPLOADED TO YOUTUBE EVERY MINUTE
20 HOURS
DATA PER DAY PROCESSED BY GOOGLE
24 PETABYTES
TWEETS PER DAY
50 MILLION
TOTAL MINUTES SPENT ON FACEBOOK EACH MONTH
700 BILLION
DATA SENT AND RECEIVED BY MOBILE INTERNET USERS
1.3 EXABYTES
PRODUCTS ORDERED ON AMAZON PER SECOND
72.9 ITEMS
G
IN THE 21ST CENTURY, we live a large part of our lives online. Almost everything we do is reduced to bits and sent through cables around the world at light speed. But just how much data are we generating? This is a look at just some of the massive amounts of information that human beings create every single day.
A COLLABORATION BETWEEN GOOD AND OLIVER MUNDAY
IN PARTNERSHIP WITH IBM

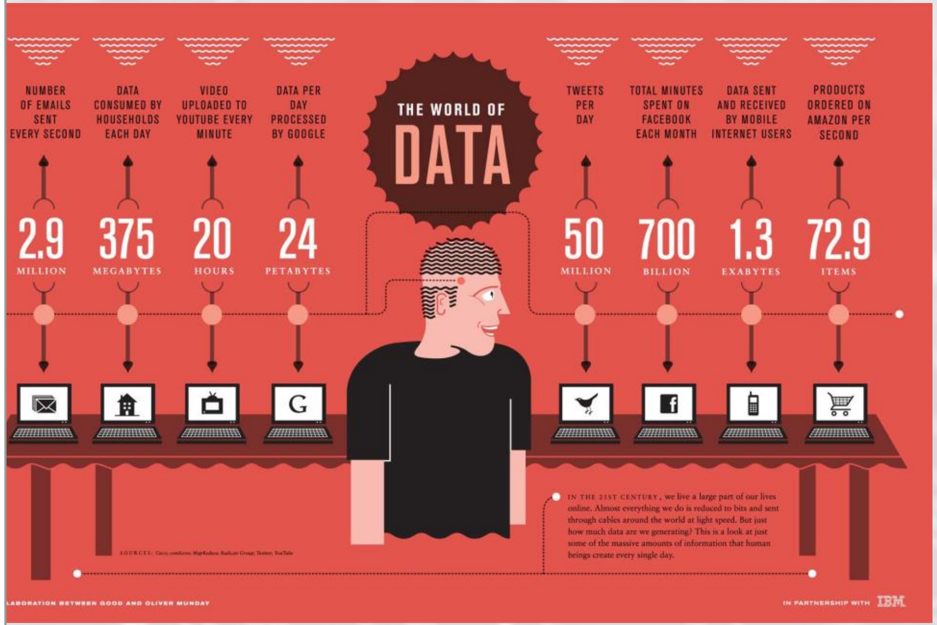


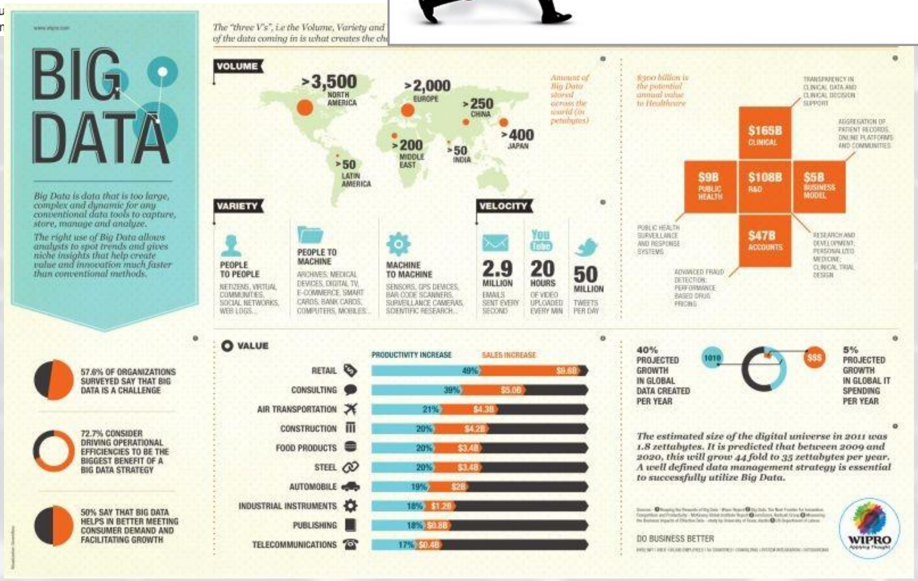


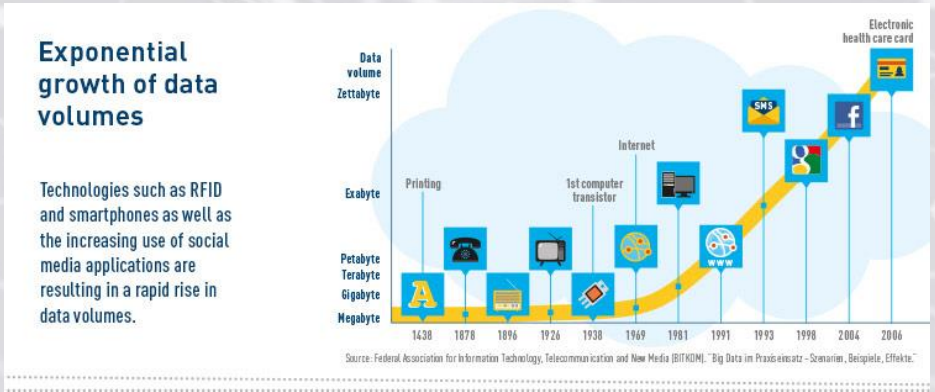
Big Data/High-Dimensional Data



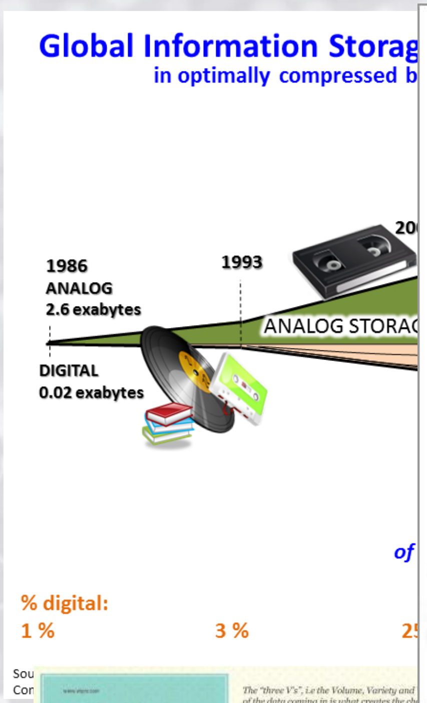
HBR.ORG
Harvard Business Review
OCTOBER 2012
46 The Big Idea
The True Measures Of Success
Michael J. Mauboussin
84 International Business
10 Rules for Managing Global Innovation
Keeley Wilson and Yves L. Doz
93 Leadership
What Ever Happened To Accountability?
Thomas E. Ricks
GETTING CONTROL OF
BIG DATA
How vast new streams of information are changing the art of management
PAGE 59


THE WORLD OF DATA
NUMBER OF EMAILS SENT EVERY SECOND
2.9 MILLION
DATA CONSUMED BY HOUSEHOLDS EACH DAY
375 MEGABYTES
VIDEO UPLOADED TO YOUTUBE EVERY MINUTE
20 HOURS
DATA PER DAY PROCESSED BY GOOGLE
24 PETABYTES
TWEETS PER DAY
50 MILLION
TOTAL MINUTES SPENT ON FACEBOOK EACH MONTH
700 BILLION
DATA SENT AND RECEIVED BY MOBILE INTERNET USERS
1.3 EXABYTES
PRODUCTS ORDERED ON AMAZON PER SECOND
72.9 ITEMS
IN THE 21ST CENTURY, we live a large part of our lives online. Almost everything we do is reduced to bits and sent through cables around the world at light speed. But just how much data are we generating? This is a look at just some of the massive amounts of information that human beings create every single day.
LABORATION BETWEEN GOOD AND OLIVER MUNDAY
IN PARTNERSHIP WITH IBM

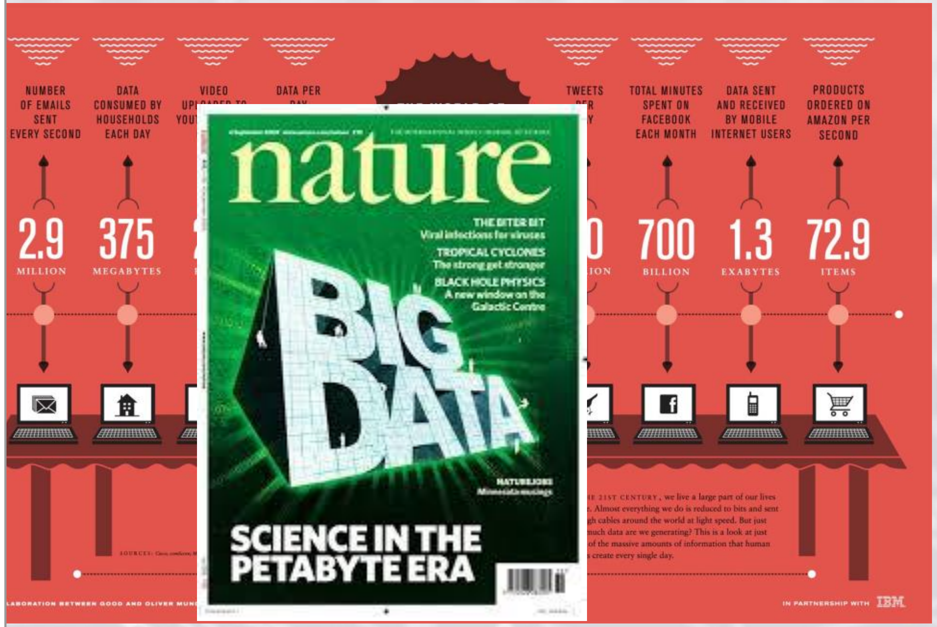




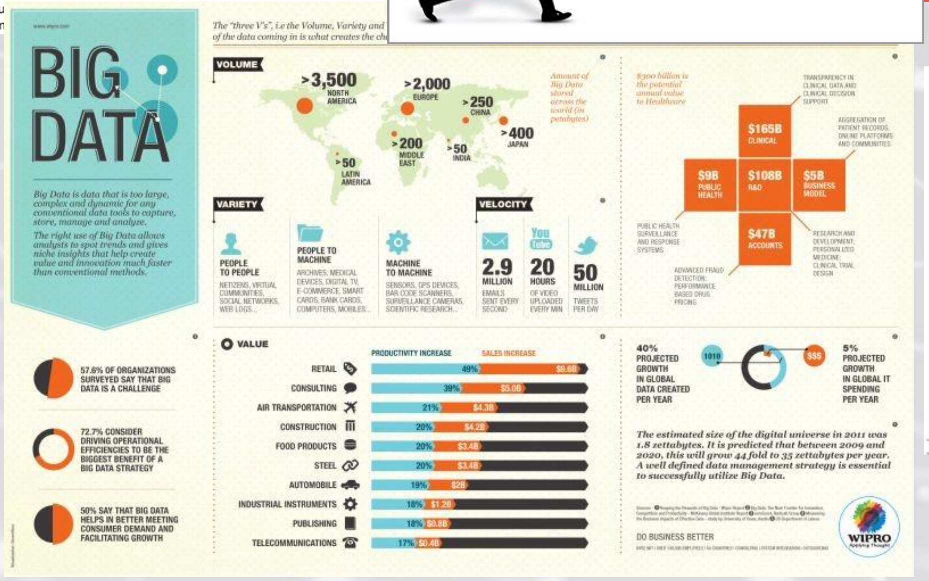
Big Data/High-Dimensional Data

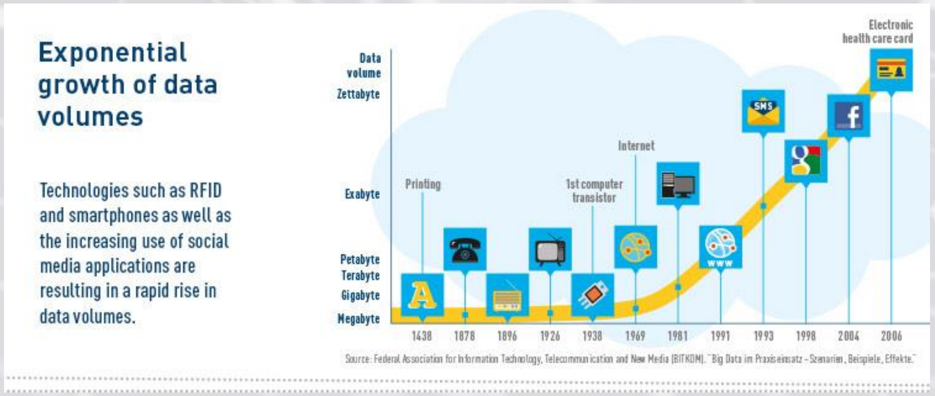


HBR.ORG
Harvard Business Review
OCTOBER 2012
46 The Big Idea
The True Measures Of Success
Michael J. Mauboussin
84 International Business
10 Rules for Managing Global Innovation
Keeley Wilson and Yves L. Doz
93 Leadership
What Ever Happened To Accountability?
Thomas E. Ricks
GETTING CONTROL OF
BIG DATA
How vast new streams of information are changing the art of management
PAGE 59

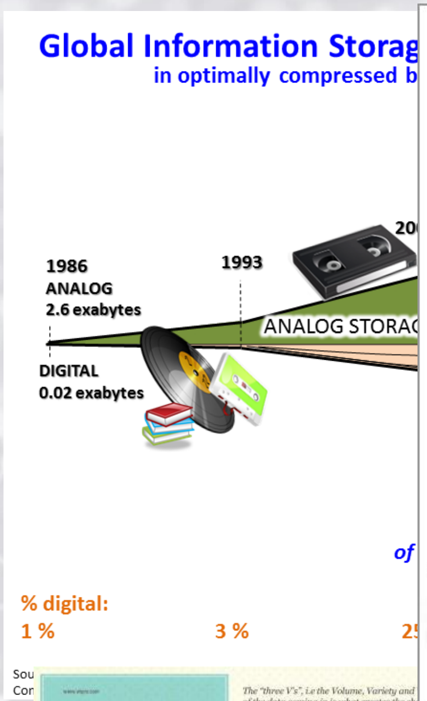

NUMBER OF EMAILS SENT EVERY SECOND
2.9 MILLION
DATA CONSUMED BY HOUSEHOLDS EACH DAY
375 MEGABYTES
VIDEO UPLOADED TO YOU
DATA PER DAY
TWEETS PER DAY
TOTAL MINUTES SPENT ON FACEBOOK EACH MONTH
700 BILLION
DATA SENT AND RECEIVED BY MOBILE INTERNET USERS
1.3 EXABYTES
PRODUCTS ORDERED ON AMAZON PER SECOND
72.9 ITEMS
IN PARTNERSHIP WITH IBM

nature
THE BITTER BIT
Viral infections for viruses
TROPICAL CYCLONES
The strong get stronger
BLACK HOLE PHYSICS
A new window on the Galactic Centre
BIG DATA
SCIENCE IN THE PETABYTE ERA


BIG DATA
Big Data is data that is too large, complex and dynamic for any conventional data tools to capture, store, manage and analyze.
The right use of Big Data allows analysts to spot trends and gives niche insights that help create value and innovation much faster than conventional methods.
The "three V's", i.e the Volume, Variety and
of the data coming in is what creates the ch
VOLUME
>3,500 NORTH AMERICA
>2,000 EUROPE
>250 CHINA
>400 JAPAN
>200 MIDDLE EAST
>50 INDIA
>50 LATIN AMERICA
VARIETY
PEOPLE TO PEOPLE
PEOPLE TO MACHINE
MACHINE TO MACHINE
VELOCITY
2.9 MILLION EMAILS SENT EVERY SECOND
20 HOURS OF VIDEO UPLOADED EVERY MIN
50 MILLION TWEETS PER DAY
$165B CLINICAL
$9B PUBLIC HEALTH
$108B R&D
$5B BUSINESS MODEL
$47B ACCOUNTS
57.6% OF ORGANIZATIONS SURVEYED SAY THAT BIG DATA IS A CHALLENGE
72.7% CONSIDER DRIVING OPERATIONAL EFFICIENCIES TO BE THE BIGGEST BENEFIT OF A BIG DATA STRATEGY
50% SAY THAT BIG DATA HELPS IN BETTER MEETING CONSUMER DEMAND AND FACILITATING GROWTH
VALUE
PRODUCTIVITY INCREASE
SALES INCREASE
RETAIL 49% $9.6B
CONSULTING 39% $5.0B
AIR TRANSPORTATION 21% $4.3B
CONSTRUCTION 20% $4.2B
FOOD PRODUCTS 20% $3.4B
STEEL 20% $3.4B
AUTOMOBILE 19% $2B
INDUSTRIAL INSTRUMENTS 18% $1.2B
PUBLISHING 18% $0.8B
TELECOMMUNICATIONS 17% $0.4B
40% PROJECTED GROWTH IN GLOBAL DATA CREATED PER YEAR
5% PROJECTED GROWTH IN GLOBAL IT SPENDING PER YEAR
The estimated size of the digital universe in 2011 was 1.8 zettabytes. It is predicted that between 2009 and 2020, this will grow 44 fold to 35 zettabytes per year. A well defined data management strategy is essential to successfully utilize Big Data.
DO BUSINESS BETTER
WIPRO



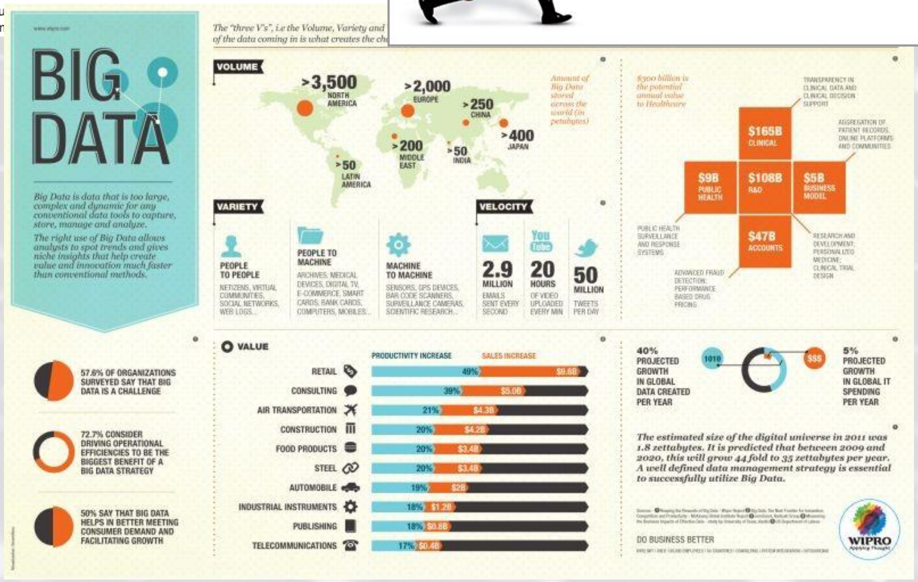
Big Data/High-Dimensional Data



HBR.ORG
Harvard Business Review
OCTOBER 2012
46 The Big Idea
The True Measures Of Success
Michael J. Mauboussin
84 International Business
10 Rules for Managing Global Innovation
Keeley Wilson and Yves L. Doz
93 Leadership
What Ever Happened To Accountability?
Thomas E. Ricks
GETTING CONTROL OF
BIG DATA
How vast new streams of information are changing the art of management
PAGE 59

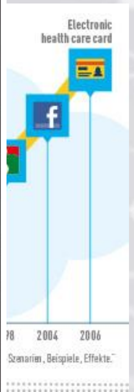
NUMBER OF EMAILS SENT EVERY SECOND
2.9 MILLION
DATA CONSUMED BY HOUSEHOLDS EACH DAY
375 MEGABYTES
VIDEO UPLOADED TO YOUTUBE EVERY MINUTE
20 HOURS
TOTAL MINUTES SPENT ON FACEBOOK EACH MONTH
700 BILLION
DATA SENT AND RECEIVED BY MOBILE INTERNET USERS
1.3 EXABYTES
PRODUCTS ORDERED ON AMAZON PER SECOND
72.9 ITEMS
IN PARTNERSHIP WITH IBM

nature
THE BITER BIT
Viral infections for viruses
TROPICAL CYCLONES
The strong get stronger
BLACK HOLE PHYSICS
A new window on the Galactic Centre
BIG DATA
SCIENCE IN THE PETABYTE ERA



INSIDE: A 14-PAGE SPECIAL REPORT ON FINANCIAL TECHNOLOGY
The Economist
How to fix America's inner cities
The self-service economy
Time to open up Indonesia
Inside the anti-bribery business
Why humans cause heatwaves
Artificial Intelligence
The promise and the peril

The Economist
Obama the warrior
Misgoverning Argentina
The economic shift from West to East
Genetically modified crops blossom
The right to eat cats and dogs
The data deluge
AND HOW TO HANDLE IT: A 14-PAGE SPECIAL REPORT



Big Data/High-Dimensional Data

Global Information Storag
in optimally compressed b
1986
ANALOG
2.6 exabytes
DIGITAL
0.02 exabytes
1993
ANALOG STORAG
% digi
1 %
Harvard Business Review
OCTOBER 2012
The True Measures Of Success
10 Rules for Managing Global Innovation
What Ever Happened To Accountability?
GETTING CONTROL OF BIG DATA
nature
BIG DATA
SCIENCE IN THE PETABYTE ERA
NUMBER OF EMAILS SENT EVERY SECOND
2.9 MILLION
DATA CONSUMED BY HOUSEHOLDS EACH DAY
375 MEGABYTES
TOTAL MINUTES SPENT ON FACEBOOK EACH MONTH
700 BILLION
DATA SENT AND RECEIVED BY MOBILE INTERNET USERS
1.3 EXABYTES
PRODUCTS ORDERED ON AMAZON PER SECOND
72.9 ITEMS
IN PARTNERSHIP WITH IBM
TIME
A tale of two Libyas
YOUR DATA FOR SALE
Everything about you is being tracked—get over it
BY JOEL STEIN
$165B
$9B
$108B
$5B
$47B
WIPRO
The Economist
Artificial Intelligence
The promise and the peril
The Economist
The data deluge
AND HOW TO HANDLE IT: A 14-PAGE SPECIAL REPORT
Electronic health care card

Big Data/High-Dimensional Data

Global Information Storag
in optimally compressed b
1986 ANALOG 2.6 exabytes
1993
DIGITAL 0.02 exabytes
ANALOG STORAG
Harvard Business Review
OCTOBER 2012
GETTING CONTROL OF BIG DATA
nature
BIG DATA
NUMBER OF EMAILS SENT EVERY SECOND 2.9 MILLION
DATA CONSUMED BY HOUSEHOLDS EACH DAY 375 MEGABYTES
TOTAL MINUTES SPENT ON FACEBOOK EACH MONTH 700 BILLION
DATA SENT AND RECEIVED BY MOBILE INTERNET USERS 1.3 EXABYTES
PRODUCTS ORDERED ON AMAZON PER SECOND 72.9 ITEMS
TIME
YOUR DATA FOR SALE
Everything about you is being tracked—get over it
BY JOEL STEIN
The Economist
Artificial Intellig
The promise and the peril
Pipeline | Volume 11 | Issue 3
Pipeline
The Unstructured Data Structure
GET SMARTER! NEXT-GEN ANALYSIS
BIG DATA TO THE RESCUE
THE BIG DATA HUBRIS
BIG DATA & ANALYTICS
EXPERT TIPS for Call Center CEM
Box Fresh: Real-time Contextual Awareness
Big Data Takes the C-suite
Big Data Infrastructure Demands
Machine Learning and Big Data
WIPRO

Big Data/High-Dimensional Data

Global Information Storag
in optimally compressed b
1986
ANALOG
2.6 exabytes
DIGITAL
0.02 exabytes
1993
ANALOG STORAG
Harvard Business Review
OCTOBER 2012
The True Measures Of Success
10 Rules for Managing Global Innovation
What Ever Happened To Accountability?
GETTING CONTROL OF BIG DATA
NUMBER OF EMAILS SENT EVERY SECOND
2.9 MILLION
DATA CONSUMED BY HOUSEHOLDS EACH DAY
375 MEGABYTES
TOTAL MINUTES SPENT ON FACEBOOK EACH MONTH
700 BILLION
DATA SENT AND RECEIVED BY MOBILE INTERNET USERS
1.3 EXABYTES
PRODUCTS ORDERED ON AMAZON PER SECOND
72.9 ITEMS
nature
BIG DATA
TIME
A tale of two Libyas
YOUR DATA FOR SALE
Everything about you is being tracked—get over it
BY JOEL STEIN
GLOBAL FINANCE
WORLD'S BEST SUBCUSTODIANS 2015
BIG DATA
The Future Is Now
Pipeline | Volume 11 | Issue 3
Pipeline
The Unstructured Data Structure
GET SMARTER! NEXT-GEN ANALYSIS
BIG DATA TO THE RESCUE
THE BIG DATA HUBRIS
BIG DATA & ANALYTICS
EXPERT TIPS for Call Center CEM
Box Fresh: Real-time Contextual Awareness
Big Data Takes the C-suite
Big Data Infrastructure Demands
Machine Learning and Big Data

Big Data/High-Dimensional Data

MIT Technology Review
SOCIAL MEDIA AND BIG DATA IN POLITICS
Harvard Business Review
GETTING CONTROL OF BIG DATA
nature
BIG DATA
SCIENCE IN THE PETABYTE ERA
COSMOS
IS data THE NEW GOD?
TIME
YOUR DATA FOR SALE
Everything about you is being tracked—get over it
BY JOEL STEIN
GLOBAL FINANCE
BIG DATA
The Future Is Now
Pipeline
The Unstructured Data Structure
GET SMARTER! NEXT-GEN ANALYSIS
BIG DATA TO THE RESCUE
THE BIG DATA HUBRIS
BIG DATA & ANALYTICS
EXPERT TIPS for Call Center CEM
Box Fresh: Real-time Contextual Awareness
Big Data Takes the C-suite
Big Data Infrastructure Demands
Machine Learning and Big Data
NUMBER OF EMAILS SENT EVERY SECOND
2.9 MILLION
DATA CONSUMED BY HOUSEHOLDS EACH DAY
375 MEGABYTES

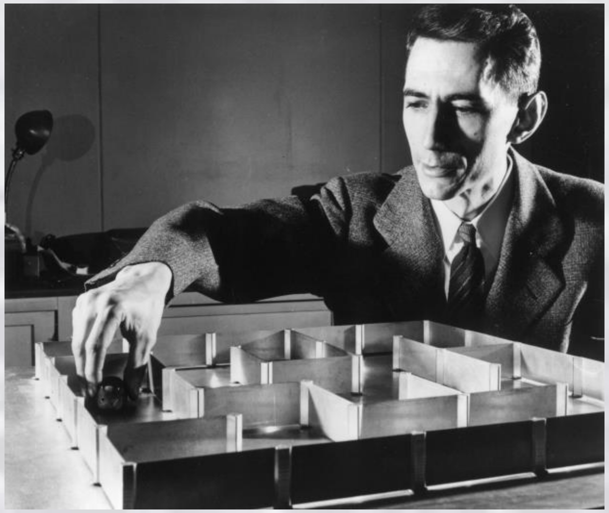
Big Data/High-Dimensional Data

MIT Technology Review
SOCIAL MEDIA AND BIG DATA IN POLITICS
Harvard Business Review
GETTING CONTROL OF BIG DATA
Computer
BIG DATA
COSMOS
IS data THE NEW GOD?
MIT Technology Review
BIG DATA WILL SAVE POLITICS
HARVARD MAGAZINE
Making sense of "Big Data"
BIG DATA
NEWS
FAKE
SPECIAL ISSUE Computational Propaganda and Political Big Data
THE BIG DATA HUBRIS
ANALYTICS
Machine Learning and Big Data
Big Data Takes the C-suite
Big Data Infrastructure Demands

Big Data/High-Dimensional Data

BIG DATA...
SO HOT RIGHT NOW
MIT Technology Review
SOCIAL MEDIA AND BIG DATA IN POLITICS
Harvard Business Review
Computer
COSMOS
IS data THE NEW GOD?
BIG DATA WILL SAVE POLITICS
GET SMARTER! NEXT-GEN ANALYSIS
THE BIG DATA HUBRIS
ANALYTICS
Big Data Takes the C-suite
Big Data Infrastructure Demands
Machine Learning and Big Data
of "Big Data"

Aside: Is the Basis of the Universe Information?



• In the 1940s, "the father of the digital age," Claude Shannon, formalized the notion of *information* through *entropy*.

$$H = -\sum_i p_i \log p_i$$

• The theoretical physicist John Archibald Wheeler later stated that The universe had (3) parts:

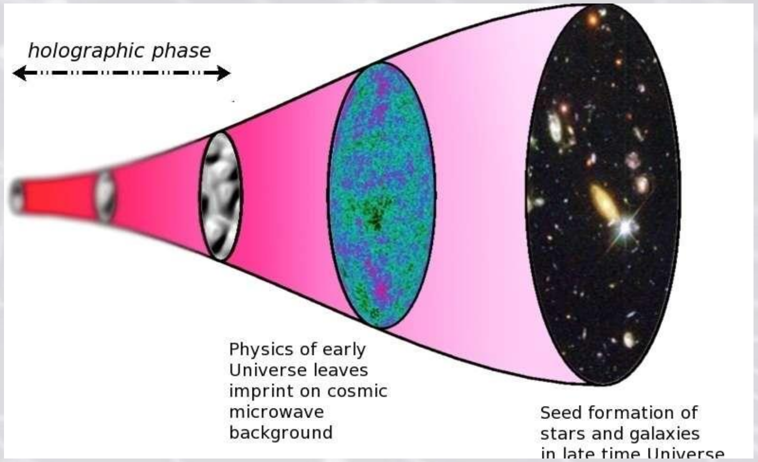
(1) Everything is particles

(2) Everything is fields

(3) Everything is information

Aside: Is the Basis of the Universe Information?

- A study in 2017 revealed substantial evidence that **we live in a holographic universe**.
- In this view, we might be caught inside a giant hologram; the cosmos is a projection, much like a 3D simulation.


holographic phase
Physics of early Universe leaves imprint on cosmic microwave background
Seed formation of stars and galaxies in late time Universe

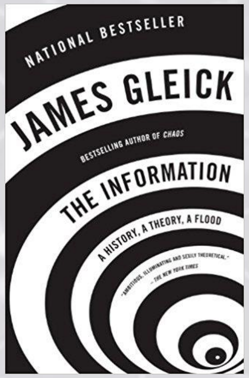
- If the nature of reality is in fact <u>reducible to information itself</u>, that implies a conscious mind on the receiving end, to interpret and comprehend it.

https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.118.041301

Aside: Is the Basis of the Universe Information?

- Wheeler himself believed in a *participatory universe*, where consciousness holds a central role.

- It is possible that information theory may in the future help bridge the gap between general relativity and quantum mechanics, or aid in our understanding of dark matter.

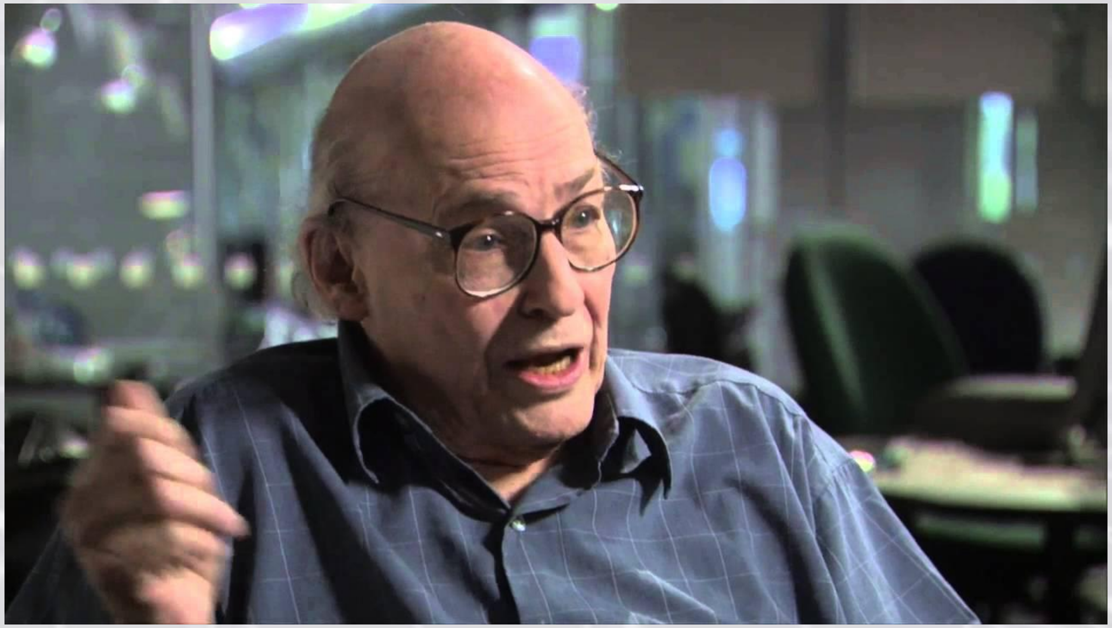


"The universe is a physical system that contains and processes information in a systematic fashion and that can do everything a computer can do." – Seth Lloyd MIT

Susskind: "On the world as hologram" https://www.youtube.com/watch?v=2DIl3Hfh9tY

Aside: Is the Basis of the Universe Information?

- Just for fun…here is a short conversation with Minsky on the question of whether information is a basic building block of reality.



https://www.closertotruth.com/series/information-fundamental

Introduction

- Most traditional statistical techniques (e.g. regression/classification) were developed in *low-dimensional* settings (i.e. $n >> p$ where n is the data size and p is the number of features).

- Over the last several decades, new technologies have drastically changed the way that data are collected (see "big data age"). Consequently, it is now commonplace to work with data with a very large number of features (i.e. $p >> n$).

- While p can be extremely large, the number of observations n is often limited due to cost, sample availability, or other considerations.

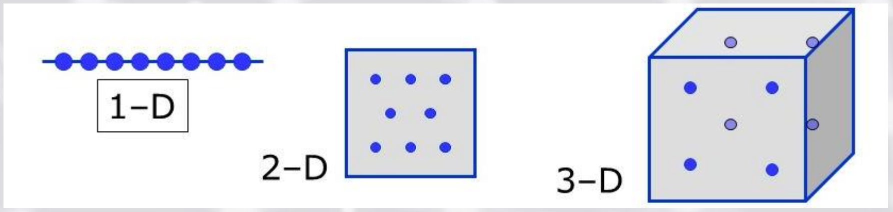
Introduction

- Data containing more features than observations are typically referred to as ***high-dimensional***.

- Issues pertaining to the *bias-variance tradeoff* and *overfitting* are commonly exacerbated in high dimensions.

- With a large number of features, statistical models (e.g. regression) can become too flexible and hence overfit the data.

- Recall the *curse of dimensionality*, which poses two fundamental, associated problems: (1) "neighborhoods" become very large (this is problematic in particular for kernel and clustering methods), (2) we need a much larger data set to adequately "fill" the space for predictive modeling, etc.



Interpretability in High Dimensions

- In high-dimensional settings we need to be **cautious about how we interpret our results** – that is to say if they can be reasonably interpreted at all.

- Of course, it is oftentimes adequate, depending on the application, to treat a machine learning model as a *mere* predictive "black box" (e.g. statistical arbitrage, government work).

- Conversely, if we want to say that the features in our model directly impact the outcomes we observe (note: in ML we almost never use the c-word – viz., variables *caused* observed effect) we need to be alert to ***multicollinearity***.

- In high dimensions, it is very likely that some of our model variables are mutually correlated. This means we can never know exactly which variables (if any) are truly predictive of the outcome. Moreover, we can rarely identify the optimal set of features for a given phenomenon of interest.

Interpretability in High Dimensions

- The "first rule" of data science and ML: **one can always add more and more features to achieve zero classification/predictive error**, a perfect correlation coefficient value, etc.

- In the end, however, ***this is a useless model***. We always need to report results on an independent test or validation set.

- In 2008, Hinton *et al*, developed a non-linear dimensionality technique known as ***t-SNE*** (*t-distributed stochastic neighbor embedding*) that is particularly well-suited for embedding high-dimensional data into 2 or 3 dimensions, which can be visualized with a scatter plot.

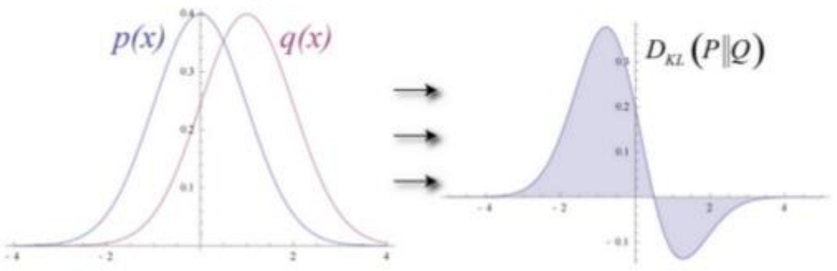
- Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that **similar objects are modeled by nearby points** and dissimilar objects are modeled by distant points.

t-SNE: H-D Data Visualization

- First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked, whilst dissimilar points have an extremely small probability of being picked.

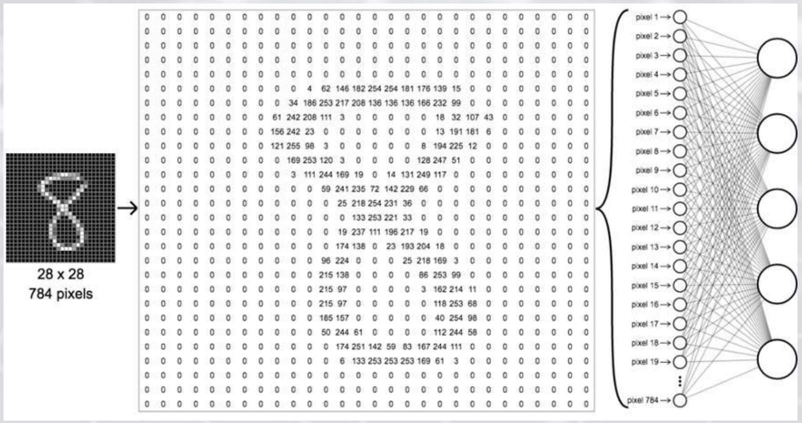
- Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the ***KL divergence*** (a standard measure of "distance" between probability distributions) between the two distributions with respect to the locations of the points in the map.

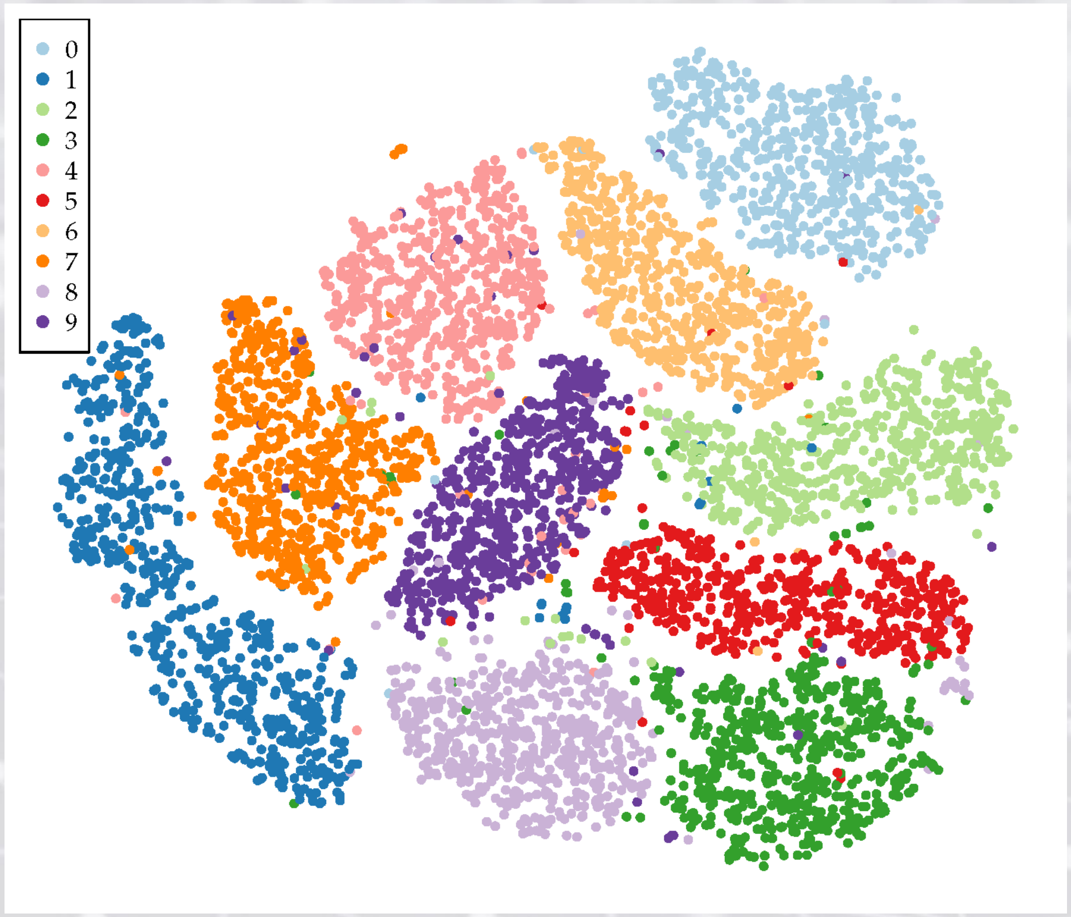
$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)}$$



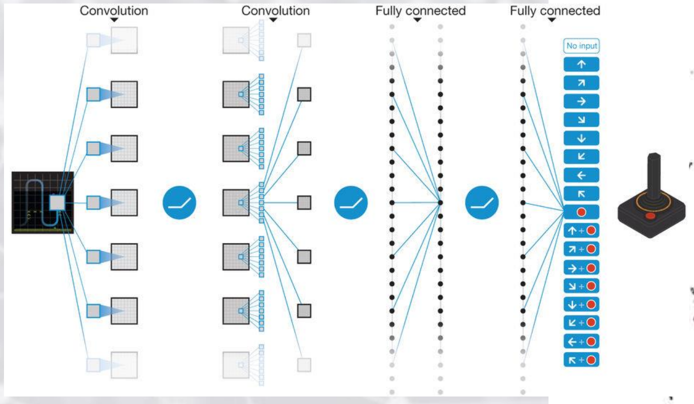
$$\begin{aligned} D_{\mathrm{KL}}(P\|Q) &= -\textstyle\sum_x p(x)\log q(x) &+& \textstyle\sum_x p(x)\log p(x) \\ &= H(P,Q) &-& H(P) \end{aligned}$$

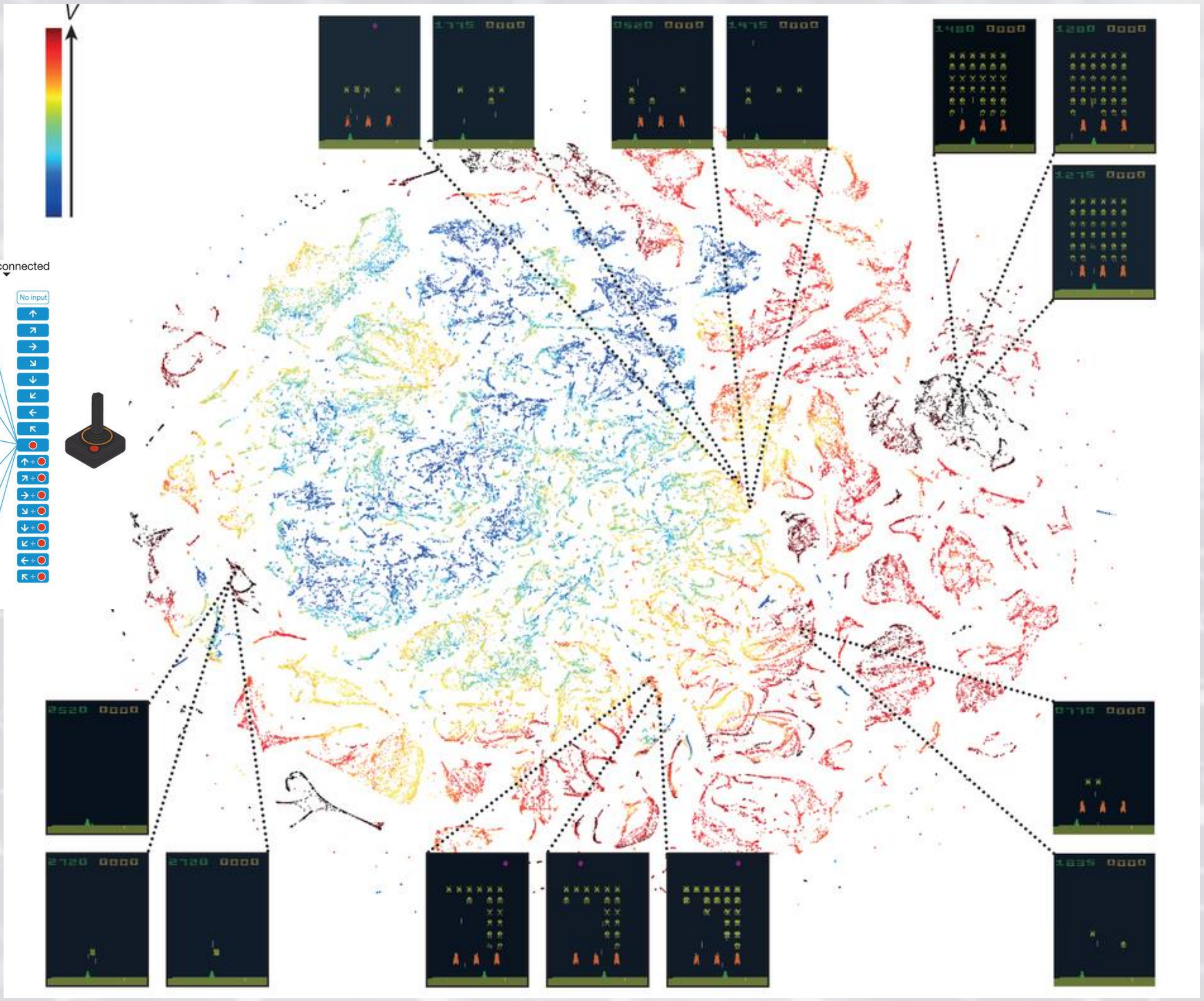
t-SNE for MNIST





t-SNE for Atari! (Deepmind)





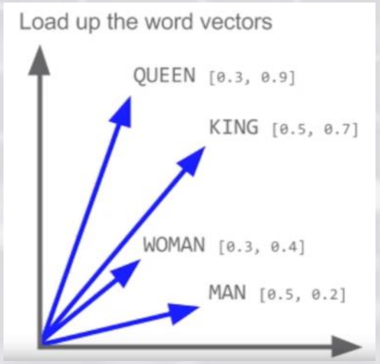
Word2vec (2013)

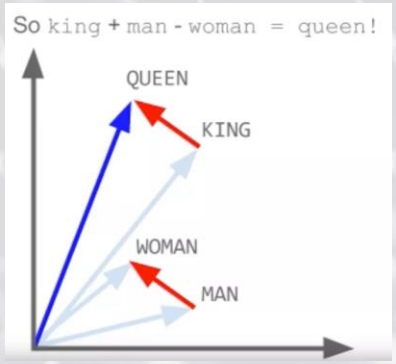
- **Word2vec** is a group of related models (Google) that are used to produce word embeddings.

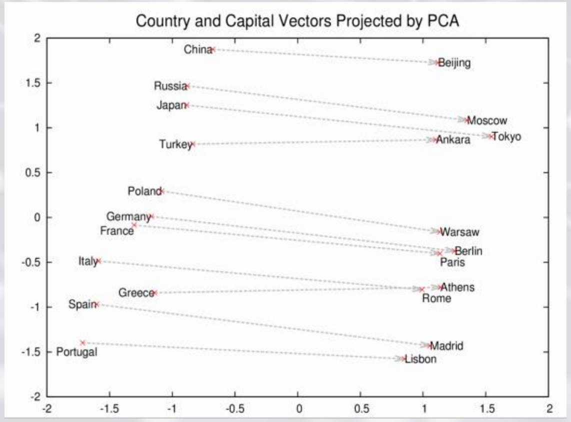
- These models are shallow, two-layer neural network that are trained to reconstruct linguistic contexts of words.

- Word2vec takes as its input a large corpus of text and produces a vector space (usually of high dimensions), with each unique word in the corpus being assigned a corresponding vector in the space.

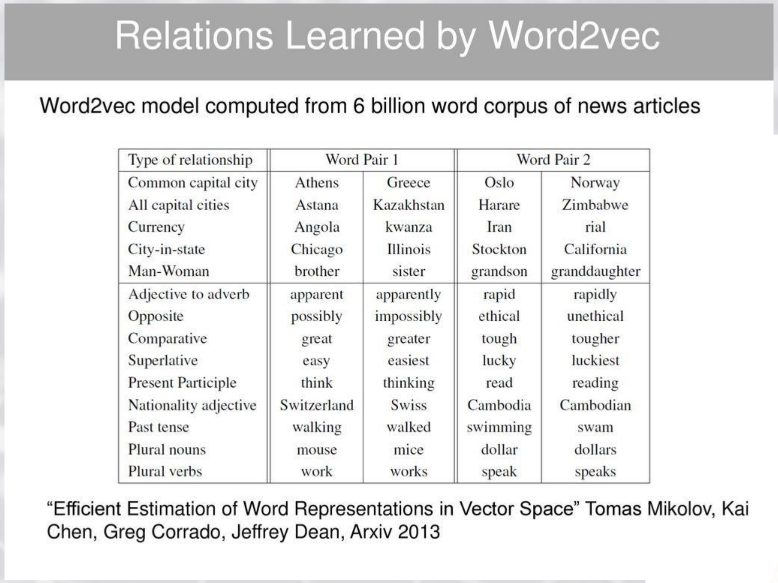
- Word vectors are positioned in the vector space such that **words that share common contexts in the corpus are located in close proximity to one another in the space**.







t-SNE for word2vec

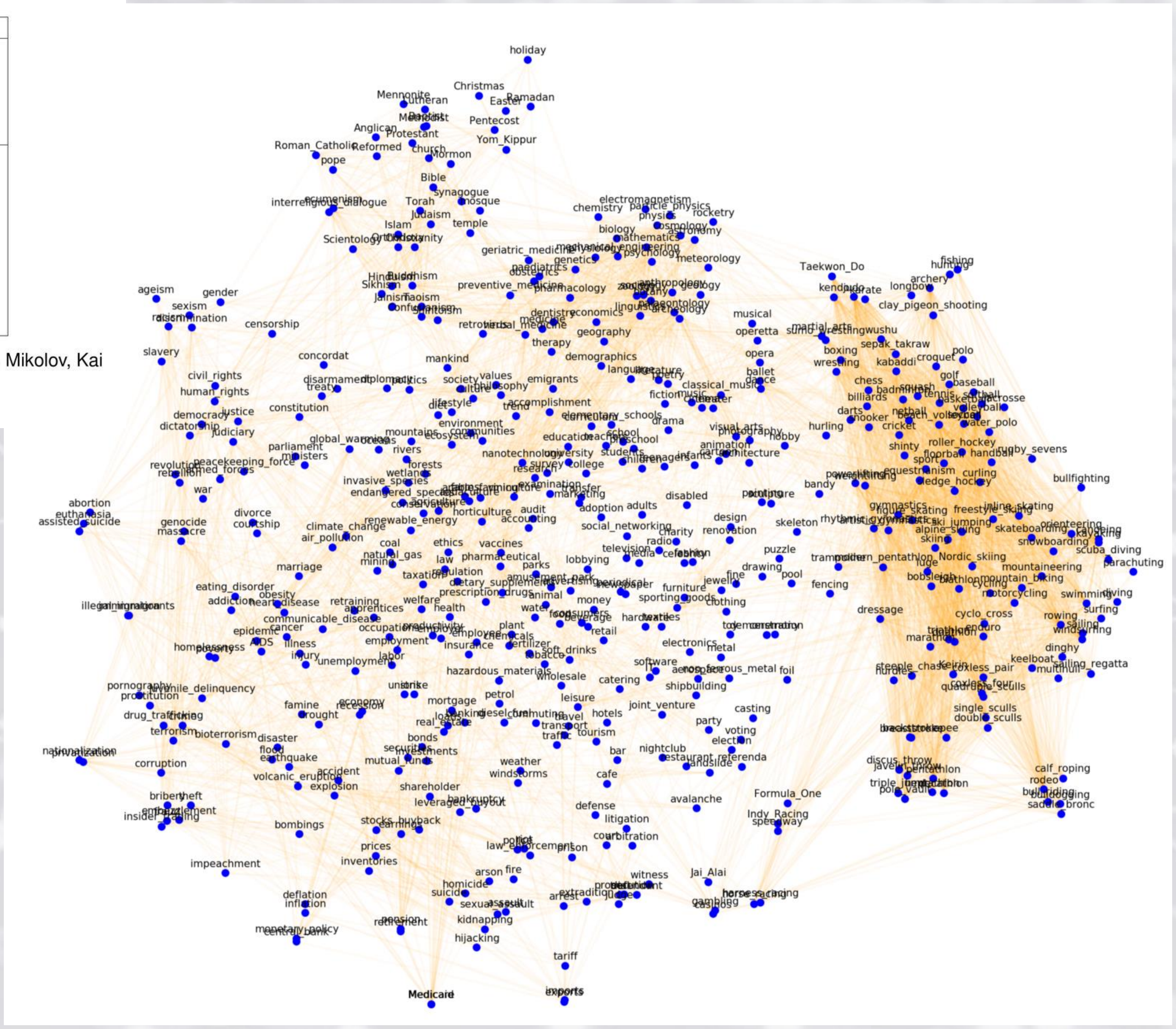


Relations Learned by Word2vec

Word2vec model computed from 6 billion word corpus of news articles

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

"Efficient Estimation of Word Representations in Vector Space" Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013



Dimensionality Reduction

- In general: the higher the number of dimensions we have, the more training data we need.
- Additionally, computational cost is generally an explicit function of dimensionality.
- Dimensionality reduction can also **remove noise** in a data set, which can, in turn, significantly improve the results of a learning algorithm.
- These are perhaps the strongest reasons why dimensionality reduction is useful (in addition to improving visualization/interpretability).

In general, there are (3) common ways to perform dimensionality reduction:

(1) Feature selection – determine whether the features available are actually useful, i.e. are they correlated with the output variables.

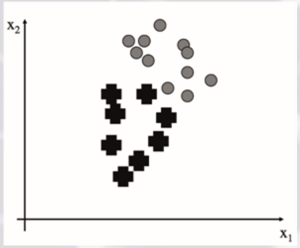
(2) Feature derivation – means deriving new features from old ones, generally by applying transforms to the data set that change the coordinate system axes (e.g., by moving or rotating); this is usually achieved through matrix multiplication.

(3) Clustering – group together similar data points to see whether this allows fewer features to be used.

LDA

- We will consider linear discriminant analysis (LDA) in an **unsupervised setting** (that is to say the data has no target label), due to Fisher (1936).

- Consider the case of **two data classes**; we can compute various summary statistics for these data, including μ, the mean of the entire set, as well as μ_1 and μ_2, the means of each class, respectively; and the covariance of each class: $\sum_j (x_j - \mu)(x_j - \mu)^T$



***Note**: We use the standard notation for column vectors, so that $\mathbf{w}^T\mathbf{w}$ is equivalent to the dot product of **w** with itself, whereas $\mathbf{w}\mathbf{w}^T$ yields an nxn matrix, for n-dimensional **w**.

LDA

- The principal insight of LDA is that the **covariance matrix** can tell us about the ***scatter within a dataset***, which is the amount of spread extant within the data.

- The way to find the *scatter* is to multiply the covariance by p_c, **the probability of the given class** (that is, the number of data points there are in the class divided by the total number).

- Adding the values of this for all of the classes gives us a measure called the **within-class scatter of the data set**:

$$S_W = \sum_{\text{classes } c} \sum_{j \in c} p_c \left(x_j - \mu_c\right)\left(x_j - \mu_c\right)^T$$

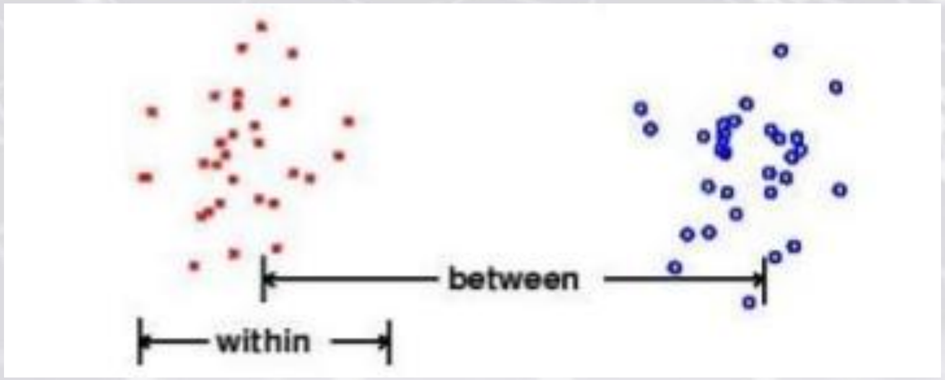
LDA

- **Within-class scatter of the data set:** $S_W = \sum_{classes\ c} \sum_{j \in c} p_c (x_j - \mu_c)(x_j - \mu_c)^T$

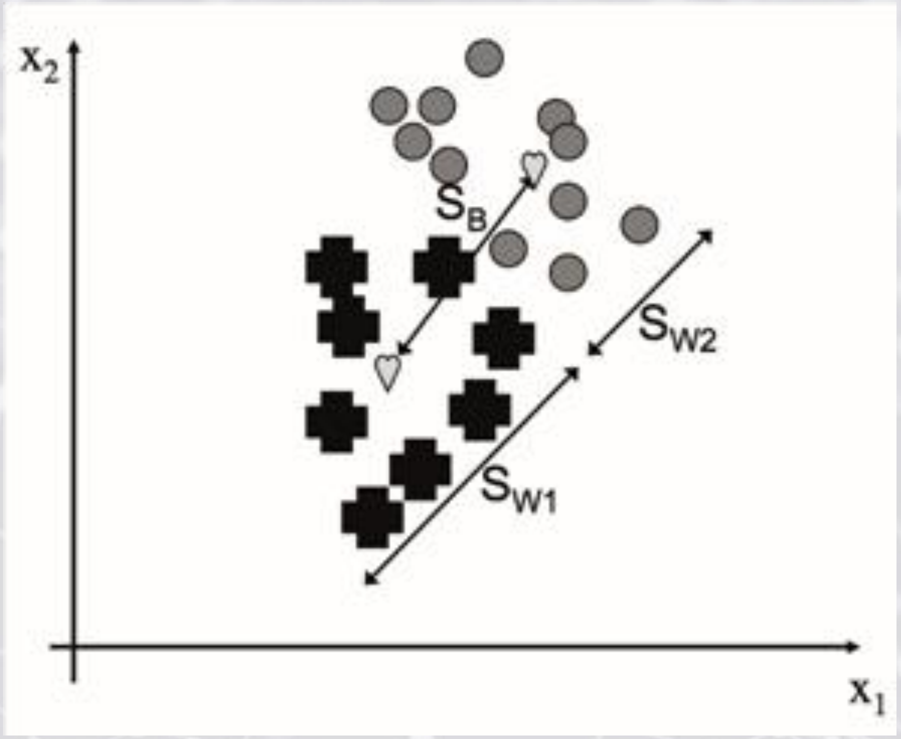
- If the data is **easy to separate into classes**, then this within-class scatter should be small, so that each class is tightly clustered together.

- Conversely, in order to separate data, we also want the **distance between classes to be large**. This quantity is known as the **between-classes scatter**:

$$S_B = \sum_{classes\ c} (\mu_c - \mu)(\mu_c - \mu)^T$$



LDA



$$S_W = \sum_{classes\ c} \sum_{j \in c} p_c \left(x_j - \mu_c\right)\left(x_j - \mu_c\right)^T \qquad S_B = \sum_{classes\ c} \left(\mu_c - \mu\right)\left(\mu_c - \mu\right)^T$$

LDA

- "Good separation" means S_W should be small and S_B should be large; accordingly we wish to make the ratio: **S_B/S_W as large as possible**.

- Regarding dimensionality reduction: we want **S_B/S_W** to be large when we reduce the number of dimensions of our data.

- Recall that the **projection** of a data point (i.e. a vector) **x** onto another vector **w** can be written as $z = \mathbf{w}^T\mathbf{x}$.

- So let's consider the between and within scatter quantities projected onto some vector **w**:

why?

$$\sum_{classes\ c}\sum_{j\in c} p_c\left(w^T\left(x_j-\mu_c\right)\right)\left(w^T\left(x_j-\mu_c\right)\right)^T = w^T S_W w$$

$$\sum_{classes\ c}\left(w^T\left(\mu_c-\mu\right)\right)\left(w^T\left(\mu_c-\mu\right)^T\right) = w^T S_B w$$

LDA

- Thus, the ratio of projected between-class and within-class scatter is:

$$\frac{w^T S_B w}{w^T S_W w}$$

How do we find the maximum value of this expression with respect to **w**?

LDA

- Thus, the ratio of projected between-class and within-class scatter is:

$$\frac{w^T S_B w}{w^T S_W w}$$

How do we find the maximum value of this expression with respect to **w**?

Take the derivative with respect to **w** and set it equal to zero (i.e. find the “critical points”).

LDA

- Thus, the ratio of projected between-class and within-class scatter is:

$$\frac{w^T S_B w}{w^T S_W w}$$

How do we find the maximum value of this expression with respect to **w**?

Take the derivative with respect to **w** and set it equal to zero (i.e. find the "critical points").

- This yields:

Why?

$$\frac{S_B w\left(w^T S_W w\right) - S_W w\left(w^T S_B w\right)}{\left(w^T S_W w\right)^2} = 0$$

LDA

- Thus, the ratio of projected between-class and within-class scatter is:

$$\frac{w^T S_B w}{w^T S_W w}$$

How do we find the maximum value of this expression with respect to **w**?

Take the derivative with respect to **w** and set it equal to zero (i.e. find the "critical points").

- This yields:

Why?

$$\frac{S_B w\left(w^T S_W w\right) - S_W w\left(w^T S_B w\right)}{\left(w^T S_W w\right)^2} = 0$$

Calculus!

$$\left(\frac{u}{v}\right)' = \frac{u'v - uv'}{v^2}$$

FYI: Matrices have analogous "derivative" rules (see *Matrix Calculus*):

$\frac{\partial}{\partial \mathbf{x}}\left[\mathbf{x}^T A \mathbf{x}\right] = \mathbf{x}^T\left(A + A^T\right)$, etc.

LDA

- After some simplification and solving a sub-problem known as the generalized eigenvalue problem, we arrive at Fisher's solution in the 2-class case:

$$w^* = \arg\max \left[\frac{w^T S_B w}{w^T S_W w} \right] = S_W^{-1}(\mu_1 - \mu_2)$$

Let's recap the LDA framework and what w* represents, now that we have the solution:

(*) LDA seeks to <u>reduce dimensionality while preserving as much of the class discriminatory information as possible</u>.

(*) We seek to obtain a scalar y by projecting each datum **x** onto a line: y = $\mathbf{w}^T\mathbf{x}$.

(*) Of all possible lines, <u>w* represents the one that maximizes the separability of the scalars</u>.

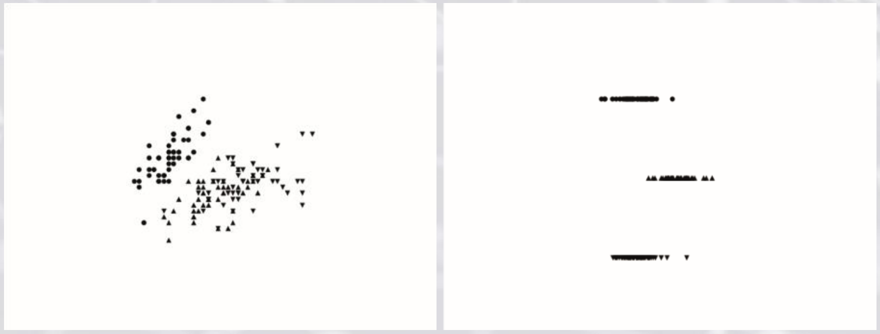


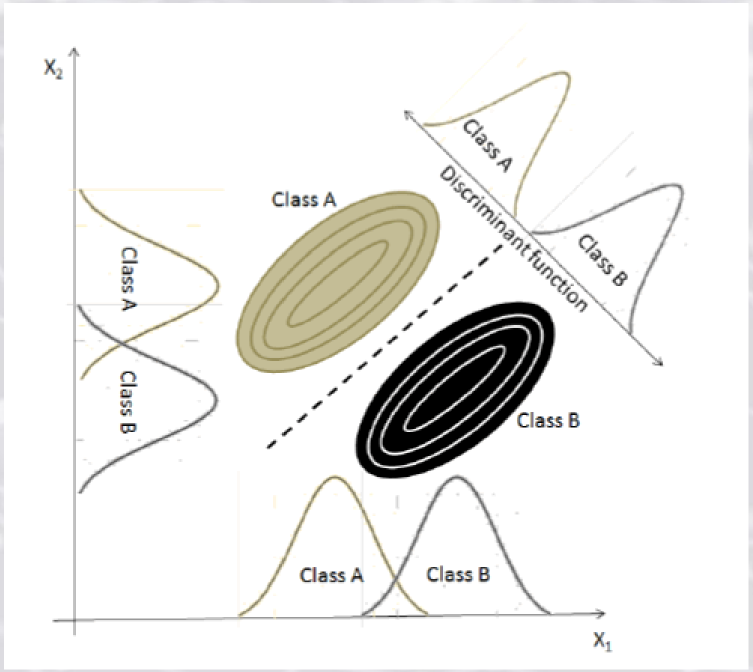
FIGURE 6.5 Plot of the iris data showing the three classes *left:* before and right: after LDA has been applied.

LDA

(*) In the figure: the two classes A and B appear overlapped along both X_1 and X_2 directions.

(*) However, they are perfectly separated along the discriminant function w*.

(*) Projecting the data onto this discriminant function renders a perfect separation of the two classes.



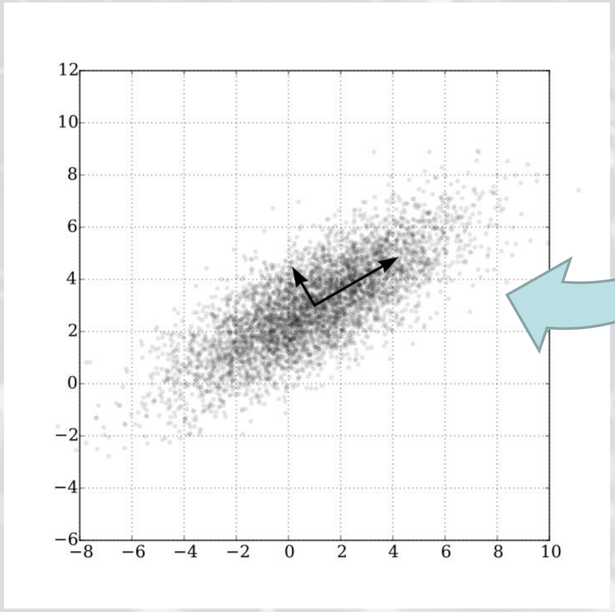
$$w^* = \arg\max\left[\frac{w^T S_B w}{w^T S_W w}\right] = S_W^{-1}(\mu_1 - \mu_2)$$

* Note that there are a variety of different forms of LDA, including the use of Bayes' theorem for posterior classification, and QDA (quadratic discriminant analysis), where the discriminant function is parabolic.

PCA

- Like LDA, **principal component analysis** (PCA) amounts to computing a transformation of a data set in order to identify a (useful) lower-dimensional set of axes.

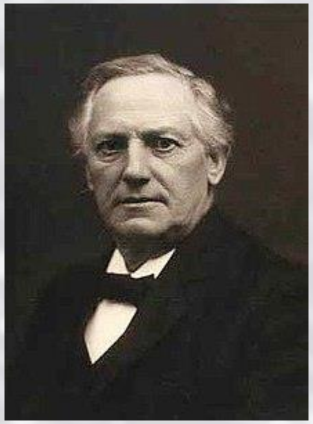
Essential idea: PCA generates a particular set of coordinate axes (usually in fewer dimensions than the original data) **that capture the maximum variability in the data**; furthermore, these new coordinate axes are **orthogonal** (which is to say they are uncorrelated).



PCA: (Aside) Gram-Schmidt

- One of the quintessential results in Linear Algebra is the **Gram-Schmidt algorithm** (*NB*: this is the same Gram per the "Gram matrix").

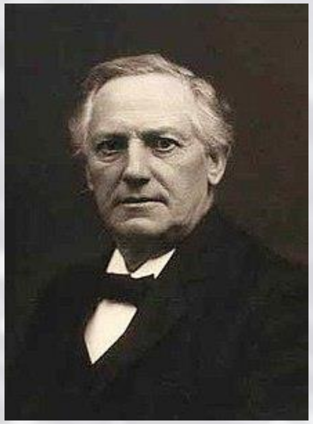
The G-S algorithm takes a basis for, say, an *inner product space* (viz., a vector space equipped with an inner product, such as the *Euclidean space* R^n) and returns an ***orthonormal basis*** for the same space.


Gram

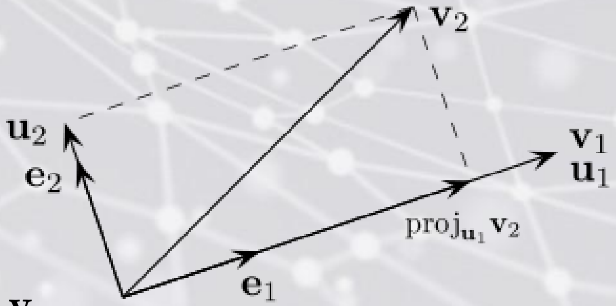
PCA: (Aside) Gram-Schmidt

- One of the quintessential results in Linear Algebra is the **Gram-Schmidt algorithm** (*NB*: this is the same Gram per the "Gram matrix").

The G-S algorithm takes a basis for, say, an *inner product space* (viz., a vector space equipped with an inner product, such as the *Euclidean space* R^n) and returns an ***orthonormal basis*** for the same space.


Gram

(*) Given a basis set, each step of the G-S algorithm amounts to <u>iteratively subtracting off the orthogonal projection of the current vector from each of the previous vectors in the process</u>.



(*) In the diagram, the set $\{\mathbf{v}_1, \mathbf{v}_2\}$ is the original basis set;

(1) Normalize $\mathbf{v}_1$: $e_1 = v_1/||v_1||$

(2) Project $\mathbf{v}_2$ onto $\mathbf{v}_1$

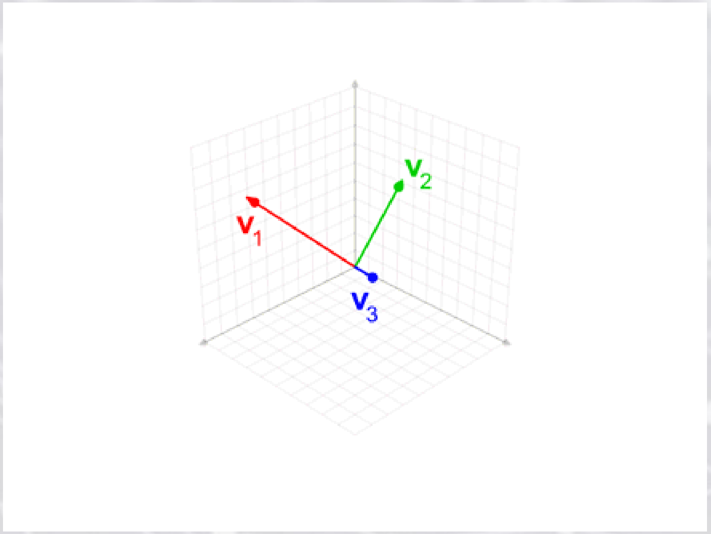
(3) Define $\mathbf{u}_2$ as the difference of $\mathbf{v}_2$ and the projection of $\mathbf{v}_2$ onto $\mathbf{v}_1$

(4) Normalize $\mathbf{u}_2$: $e_2 = u_2/||u_2||$ (5) the set $\{e_1, e_2\}$ is an **orthonormal basis**.

PCA: (Aside) Gram-Schmidt

The G-S algorithm takes a basis for, say, an *inner product space* (viz., a vector space equipped with an inner product, such as the *Euclidean space R^n^*) and returns an ***orthonormal basis*** for the same space.

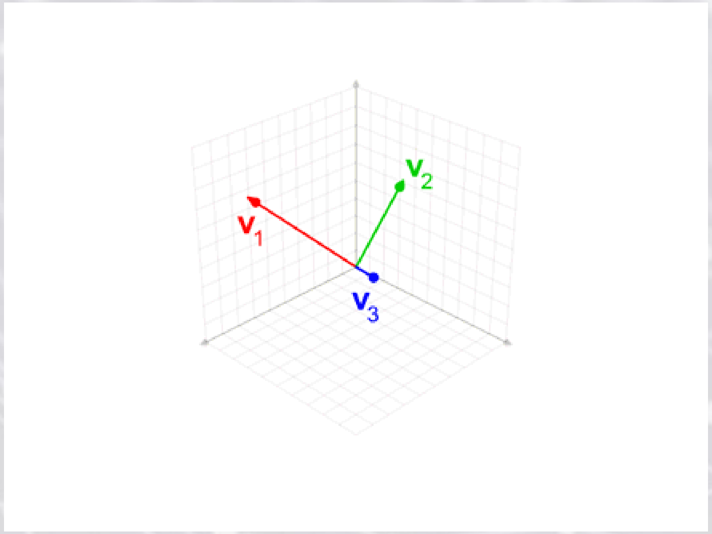
$$
\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{v}_2 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_2), & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{v}_3 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_3), & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
\mathbf{u}_4 &= \mathbf{v}_4 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_4) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_4) - \mathrm{proj}_{\mathbf{u}_3}(\mathbf{v}_4), & \mathbf{e}_4 &= \frac{\mathbf{u}_4}{\|\mathbf{u}_4\|} \\
&\ \ \vdots & &\ \ \vdots \\
\mathbf{u}_k &= \mathbf{v}_k - \sum_{j=1}^{k-1} \mathrm{proj}_{\mathbf{u}_j}(\mathbf{v}_k), & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}.
\end{aligned}
$$



PCA: (Aside) Gram-Schmidt

The G-S algorithm takes a basis for, say, an *inner product space* (viz., a vector space equipped with an inner product, such as the *Euclidean space* R^n) and returns an ***orthonormal basis*** for the same space.

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{v}_2 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_2), & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{v}_3 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_3), & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
\mathbf{u}_4 &= \mathbf{v}_4 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_4) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_4) - \mathrm{proj}_{\mathbf{u}_3}(\mathbf{v}_4), & \mathbf{e}_4 &= \frac{\mathbf{u}_4}{\|\mathbf{u}_4\|} \\
&\ \ \vdots & &\ \ \vdots \\
\mathbf{u}_k &= \mathbf{v}_k - \sum_{j=1}^{k-1} \mathrm{proj}_{\mathbf{u}_j}(\mathbf{v}_k), & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}.
\end{aligned}$$

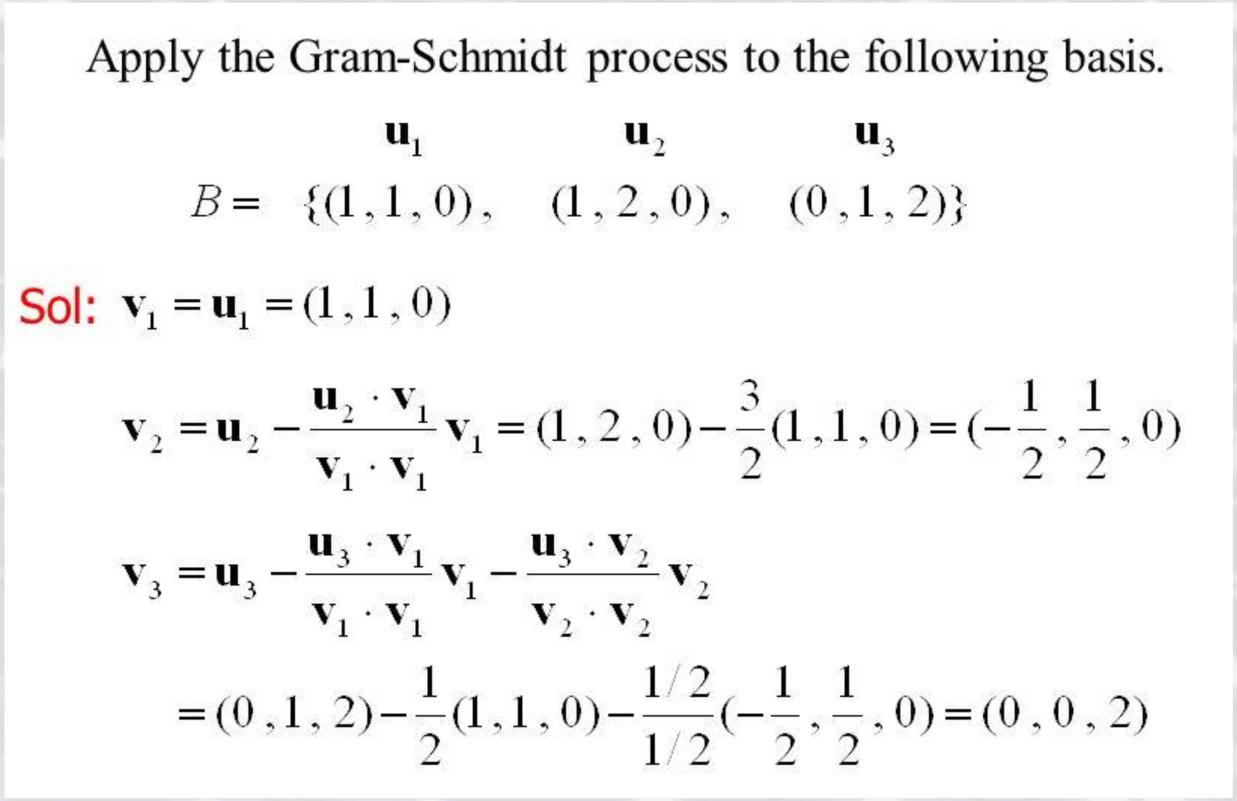


(*) How is this useful? G-S algorithm is used to render the **QR factorization** of a matrix.

(*) PCA employs a similar strategy to generate the "principal components" of a data set.

PCA: (Aside) Gram-Schmidt

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{v}_2 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_2), & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{v}_3 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_3), & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
\mathbf{u}_4 &= \mathbf{v}_4 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_4) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_4) - \mathrm{proj}_{\mathbf{u}_3}(\mathbf{v}_4), & \mathbf{e}_4 &= \frac{\mathbf{u}_4}{\|\mathbf{u}_4\|} \\
&\vdots & &\vdots \\
\mathbf{u}_k &= \mathbf{v}_k - \sum_{j=1}^{k-1} \mathrm{proj}_{\mathbf{u}_j}(\mathbf{v}_k), & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}.
\end{aligned}$$



Apply the Gram-Schmidt process to the following basis.

$$B = \{\underset{\mathbf{u}_1}{(1,1,0)},\ \underset{\mathbf{u}_2}{(1,2,0)},\ \underset{\mathbf{u}_3}{(0,1,2)}\}$$

Sol: $\mathbf{v}_1 = \mathbf{u}_1 = (1,1,0)$

$$\mathbf{v}_2 = \mathbf{u}_2 - \frac{\mathbf{u}_2 \cdot \mathbf{v}_1}{\mathbf{v}_1 \cdot \mathbf{v}_1}\mathbf{v}_1 = (1,2,0) - \frac{3}{2}(1,1,0) = (-\frac{1}{2}, \frac{1}{2}, 0)$$

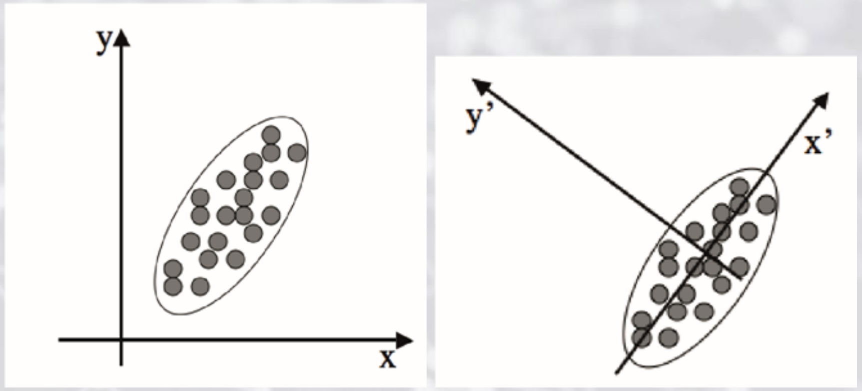
$$\mathbf{v}_3 = \mathbf{u}_3 - \frac{\mathbf{u}_3 \cdot \mathbf{v}_1}{\mathbf{v}_1 \cdot \mathbf{v}_1}\mathbf{v}_1 - \frac{\mathbf{u}_3 \cdot \mathbf{v}_2}{\mathbf{v}_2 \cdot \mathbf{v}_2}\mathbf{v}_2$$

$$= (0,1,2) - \frac{1}{2}(1,1,0) - \frac{1/2}{1/2}(-\frac{1}{2}, \frac{1}{2}, 0) = (0,0,2)$$

PCA

PCA generates a particular set of coordinate axes **that capture the maximum variability in the data**; furthermore, these new coordinate axes are **orthogonal.**

The figure shows two versions of the same data set.



- In the **first image**, the data are arranged in an ellipse that runs at 45° axes; while in **the second**, the axes have been moved so that the data now runs along the x-axis and is centered on the origin.

- Key idea: the potential for dimensionality reduction rests in the fact that the *y* dimension now does not demonstrate much variability – and so it might be possible to ignore it and simply use the *x* axis values alone for learning, etc.

(*) In fact, applying this dimensionality reduction often has the nice effect of removing some of the noise in the data.

PCA

Q: How do we choose the new axes?

A: With PCA, the *principal component* is the direction in the data with the largest variation.

$\left\{ \right.$

- The algorithm first **centers the data** (by subtracting off the mean), and then **chooses the direction with the largest variation** and places an axis in that direction.

- For subsequent steps, the algorithm then looks at the variation that remains and **finds another axis that is orthogonal to the first and covers as much of the remaining variation as possible**.

Rinse and repeat…

PCA

Q: How do we choose the new axes?

A: With PCA, the *principal component* is the direction in the data with the largest variation.

- The algorithm first **centers the data** (by subtracting off the mean), and then **chooses the direction with the largest variation** and places an axis in that direction.

- For subsequent steps, the algorithm then looks at the variation that remains and **finds another axis that is orthogonal to the first and covers as much of the remaining variation as possible**.

Rinse and repeat…

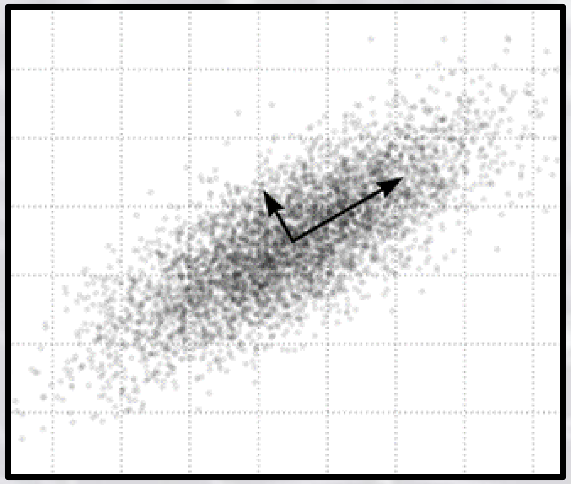
(*) The end result is that all the variation is along the axes of the coordinate set, and so the **covariance matrix of the transformed data is diagonal** (since each new variable is uncorrelated with every variable except itself).

(*) Because some of the axes generated in this process have very little variation, **we can typically remove them** without drastically affecting the variability in the data (note the implicit assumption: variation in data features equates to useful information for classification/inference).

PCA

Let's formally work out the PCA algorithm.

- Suppose we have a data matrix **X** of dimension **n by m** (*n* is the number of training instances, *m* is the dimension of each datum).

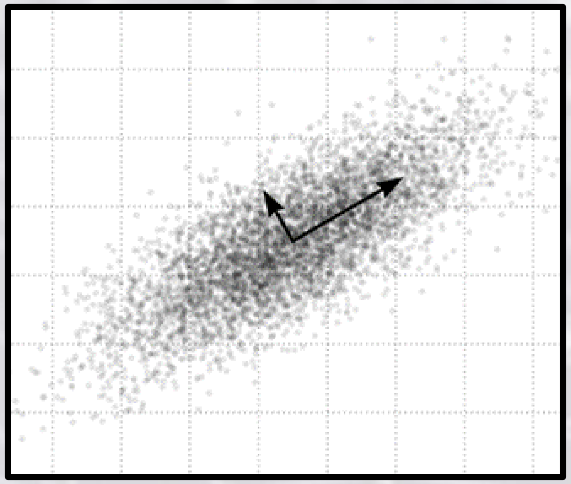


<u>Goal for PCA</u>: rotate the data so that we render a coordinate system so the new axes are uncorrelated (i.e. orthogonal) and can be ranked according to maximum variation.

PCA

Let's formally work out the PCA algorithm.

- Suppose we have a data matrix **X** of dimension **n by d** (*n* is the number of training instances, *d* is the dimension of each datum).

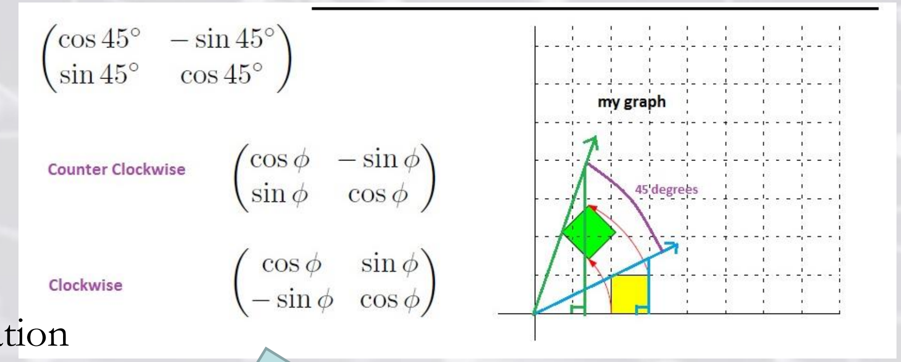


<u>**Goal for PCA**</u>: rotate the data so that we render a coordinate system so the new axes are uncorrelated (i.e. orthogonal) and can be ranked according to maximum variation.

Using standard techniques from linear algebra, we can express the <u>rotation of the data matrix X</u> as:

$$Y = P^T X$$

Where P is a "rotation matrix" with the natural property that $P^T = P^{-1}$ (why?).



$$\begin{pmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{pmatrix}$$

Counter Clockwise $\begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}$

Clockwise $\begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}$

Simple rotation matrix example

PCA

$$Y = P^T X$$

- Recall that for PCA, we want the covariance of the transformed matrix Y to be diagonal.

- That is to say, we want to find a matrix P where:

$$\text{cov}(Y) = \text{cov}(P^T X) = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix}$$

PCA

$$Y = P^T X$$

- Recall that for PCA, <u>we want the covariance of the transformed matrix *Y* to be diagonal</u>.

- That is to say, we want to find a matrix *P* where:

$$\mathrm{cov}(Y) = \mathrm{cov}(P^T X) = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix}$$

- Let's relate cov(Y) and cov(X), so that we can more easily solve for P.

$$\mathrm{cov}(Y) = E\left[YY^T \right]$$

(By definition of covariance; recall the data is centered)

PCA

$$Y = P^T X$$

- Recall that for PCA, we want the covariance of the transformed matrix Y to be diagonal.

- That is to say, we want to find a matrix P where:

$$\mathrm{cov}(Y) = \mathrm{cov}(P^T X) = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix}$$

- Let's relate cov(Y) and cov(X), so that we can more easily solve for P.

$$\mathrm{cov}(Y) = E\left[YY^T\right]$$

$$= E\left[\left(P^T X\right)\left(P^T X\right)^T\right]$$

PCA

$$Y = P^T X$$

- Recall that for PCA, we want the covariance of the transformed matrix *Y* to be diagonal.

- That is to say, we want to find a matrix *P* where:

$$\mathrm{cov}(Y) = \mathrm{cov}(P^T X) = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix}$$

- Let's relate cov(Y) and cov(X), so that we can more easily solve for P.

$$\mathrm{cov}(Y) = E\left[YY^T\right]$$

$$= E\left[\left(P^T X\right)\left(P^T X\right)^T\right]$$

$$= E\left[\left(P^T X\right)\left(X^T P\right)\right] \quad \text{Why?}$$

PCA

$$Y = P^T X$$

- Recall that for PCA, we want the covariance of the transformed matrix Y to be diagonal.

- That is to say, we want to find a matrix P where:

$$\mathrm{cov}(Y) = \mathrm{cov}(P^T X) = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix}$$

- Let's relate cov(Y) and cov(X), so that we can more easily solve for P.

$$\begin{aligned} \mathrm{cov}(Y) &= E\left[YY^T\right] \\ &= E\left[\left(P^T X\right)\left(P^T X\right)^T\right] \\ &= E\left[\left(P^T X\right)\left(X^T P\right)\right] \\ &= P^T E\left[XX^T\right]P \\ &= P^T \,\mathrm{cov}\left(X\right)P \end{aligned}$$

Why?

PCA

- In summary: $\operatorname{cov}(Y) = P^T \operatorname{cov}(X) P$

- This tells us that:

$$P \operatorname{cov}(Y) = PP^T \operatorname{cov}(X) P = \operatorname{cov}(X) P$$

Why?

PCA

- In summary: $\operatorname{cov}(Y) = P^T \operatorname{cov}\left(X\right) P$

- This tells us that:

$$P \operatorname{cov}(Y) = PP^T \operatorname{cov}\left(X\right) P = \operatorname{cov}\left(X\right) P$$

$$P^T = P^{-1}$$

(*) Remember that **cov(Y) is a diagonal matrix**; if we write the matrix P as a set of column vectors: $P = \left[\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_N}\right]$, then we have:

$$P \operatorname{cov}(Y) = \left[\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_N}\right] \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix} = \left[\lambda_1 \mathbf{p_1}, \lambda_2 \mathbf{p_2}, \ldots, \lambda_N \mathbf{p_N}\right]$$

PCA

- In summary: $\operatorname{cov}(Y) = P^T \operatorname{cov}\left(X\right) P$ and

$$P\operatorname{cov}(Y) = [\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_N}] \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix} = [\lambda_1\mathbf{p_1}, \lambda_2\mathbf{p_2}, \ldots, \lambda_N\mathbf{p_N}]$$

(*) If we write $\boldsymbol{\lambda} = \left(\lambda_1, \lambda_2, \ldots, \lambda_N\right)^T$ *and* $\mathbf{Z} = \operatorname{cov}\left(\mathbf{X}\right)$, we then arrive at the fundamental equation:

PCA

- In summary: $\operatorname{cov}(Y) = P^T \operatorname{cov}(X) P$ and

$$P\operatorname{cov}(Y) = [\mathbf{p_1}, \mathbf{p_2}, \dots, \mathbf{p_N}] \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix} = [\lambda_1\mathbf{p_1}, \lambda_2\mathbf{p_2}, \dots, \lambda_N\mathbf{p_N}]$$

(*) If we write $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_N)^T$ *and* $\mathbf{Z} = \operatorname{cov}(\mathbf{X})$, we then arrive at the fundamental equation:

$$\boxed{\underbrace{\boldsymbol{\lambda}\mathbf{p_i} = \mathbf{Z}\mathbf{p_i}}_{P\operatorname{cov}(Y) = \operatorname{cov}(X)P} \ \textit{for each } \mathbf{p_i}}$$

PCA

- In summary: $\text{cov}(Y) = P^T \text{cov}(X) P$ and

$$P\,\text{cov}(Y) = [\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_N}] \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix} = [\lambda_1 \mathbf{p_1}, \lambda_2 \mathbf{p_2}, \ldots, \lambda_N \mathbf{p_N}]$$

(*) If we write $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_N)^T$ *and* $\mathbf{Z} = \text{cov}(\mathbf{X})$, we then arrive at the fundamental equation:

$$\boldsymbol{\lambda}\mathbf{p_i} = \mathbf{Z}\mathbf{p_i} \quad \textit{for each } \mathbf{p_i}$$

This is the *i*th column of cov(X)P

This is the *i*th column of Pcov(Y)

PCA

- In summary: $\operatorname{cov}(Y) = P^T \operatorname{cov}(X) P$ and

$$P\operatorname{cov}(Y) = [\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_N}] \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix} = [\lambda_1\mathbf{p_1}, \lambda_2\mathbf{p_2}, \ldots, \lambda_N\mathbf{p_N}]$$

(*) If we write $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_N)^T$ *and* $\mathbf{Z} = \operatorname{cov}(\mathbf{X})$, we then arrive at the fundamental equation:

$$\boldsymbol{\lambda}\mathbf{p_i} = \mathbf{Z}\mathbf{p_i} \quad \textit{for each } \mathbf{p_i}$$

This is the *i*th column of cov(X)P

This is the *i*th column of Pcov(Y)

(*) Do you recognize anything “special” about the structure of these vectors?

PCA

- In summary: $\operatorname{cov}(Y) = P^T \operatorname{cov}(X) P$ and

$$P \operatorname{cov}(Y) = [\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_N}] \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{pmatrix} = [\lambda_1 \mathbf{p_1}, \lambda_2 \mathbf{p_2}, \ldots, \lambda_N \mathbf{p_N}]$$

(*) If we write $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_N)^T$ *and* $\mathbf{Z} = \operatorname{cov}(\mathbf{X})$, we then arrive at the fundamental equation:

$$\boldsymbol{\lambda}\mathbf{p_i} = \mathbf{Z}\mathbf{p_i} \; \textit{for each } \mathbf{p_i}$$

The $\mathbf{p_i}$ vectors are **<u>eigenvectors</u>** of **Z**=cov(**X**)!

PCA

- In summary: we need to solve $\lambda \mathbf{p_i} = \mathbf{Z}\mathbf{p_i}$ *for each* $\mathbf{p_i}$, where each $\mathbf{p_i}$ vector is an eigenvector of **Z**=cov(**X**).

(*) **Useful fact (1)**: all eigenvectors of a square symmetric matrix A are orthogonal to each other.

PCA

- In summary: we need to solve $\lambda \mathbf{p}_\mathbf{i} = \mathbf{Z}\mathbf{p}_\mathbf{i}$ *for each* $\mathbf{p}_\mathbf{i}$, where each $\mathbf{p}_i$ vector is an eigenvector of **Z**=cov(**X**).

(*) **Useful fact (1)**: all eigenvectors of a square symmetric matrix A are orthogonal to each other.

(*) **Useful fact (2)**: **Z**=cov(**X**) is a square symmetric matrix (as is true for *any* covariance matrix).

PCA

- In summary: we need to solve $\lambda \mathbf{p_i} = \mathbf{Z}\mathbf{p_i} \ \textit{for each}\ \mathbf{p_i}$, where each $\mathbf{p_i}$ vector is an eigenvector of **Z**=cov(**X**).

(*) **Useful fact (1)**: all eigenvectors of a square symmetric matrix A are orthogonal to each other.

(*) **Useful fact (2)**: **Z**=cov(**X**) is a square symmetric matrix (as is true for *any* covariance matrix).

(*) If we put (1) and (2) together, this means that the solution to the PCA problem (i.e. solving for the matrix P) boils down to determining the ***eigendecomposition*** (also called the *spectral decomposition*) – which is guaranteed to exist – of cov(**X**).

$$\mathbf{Z} = \mathrm{cov}\left(\mathbf{X}\right) = \mathbf{EDE}^T$$

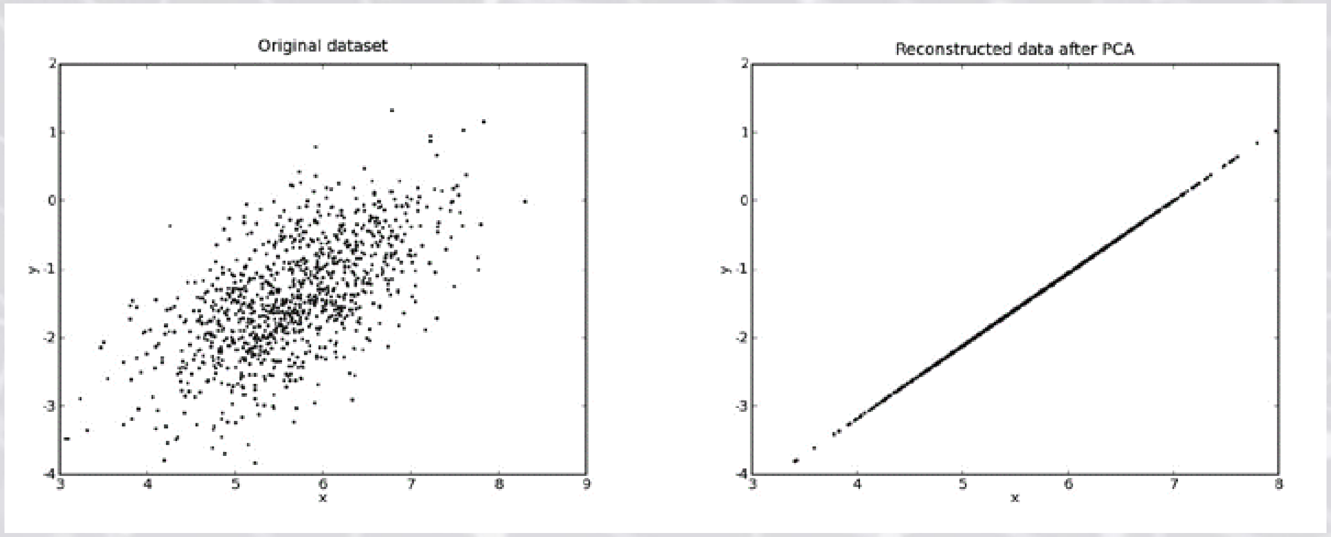
Where **D** is the diagonal matrix of eigenvalues for **Z**, and **E** is the corresponding (orthogonal) matrix of eigenvectors.

PCA

$$\mathbf{Z} = \operatorname{cov}(\mathbf{X}) = \mathbf{EDE}^{T}$$

- Note: In the eigendecomposition for cov(**X**), the dimensions with large eigenvalues have lots of variation and are therefore useful dimensions.

- In order to perform a **dimensionality reduction** on our data set, we can therefore throw away dimensions for which the eigenvalues are very small (usually smaller than some chosen parameter).



PCA

- Here is the **PCA algorithm**:

(1) Write *N* data points $x_i=(x_{1i},x_{2i},\ldots,x_{Mi})$ as row vectors.

(2) Put these vectors into the data matrix **X** (of size *N* x *M*).

(3) Center the data by subtracting off the mean of each column, place into matrix **B**.

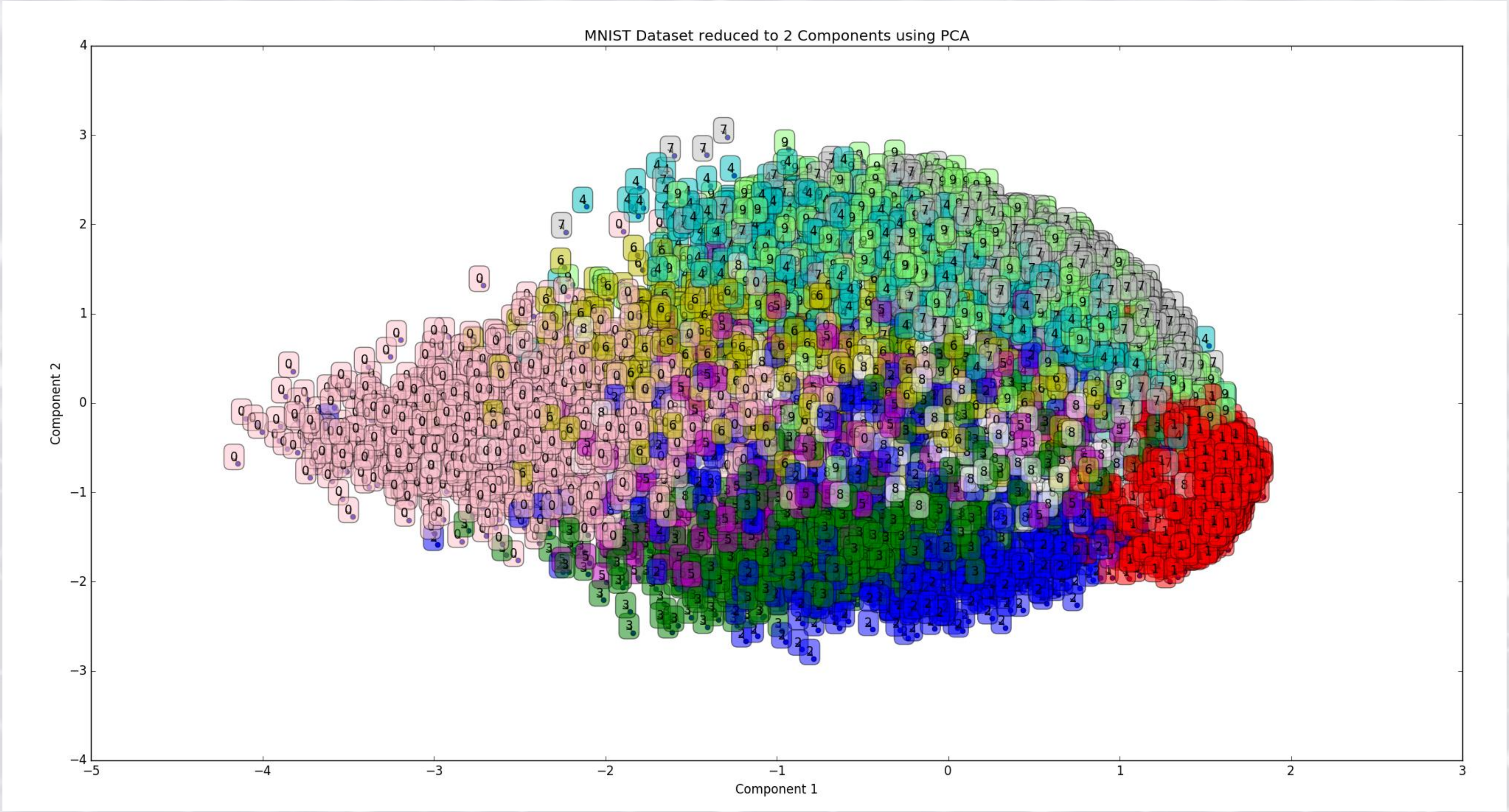
(4) Computer the covariance matrix: $\mathbf{C} = \frac{1}{N}\mathbf{B}\mathbf{B}^T$

(5) Computer the *eigenvalues* and *eigenvectors* of **C**, so: $\mathbf{C} = \mathbf{V}\mathbf{D}\mathbf{V}^T$
where **D** is the diagonal matrix of eigenvalues; V is the matrix of corresponding eigenvectors.

(6) Sort of the columns of D into order of decreasing eigenvalues, and apply the same order to the columns of V.

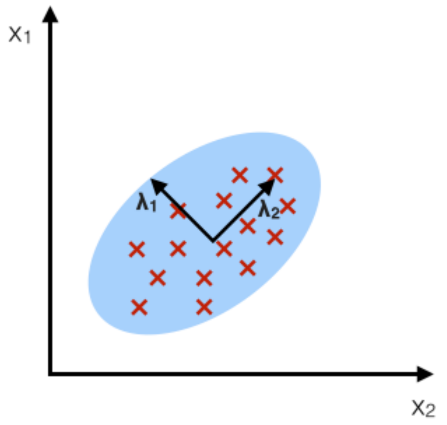
(7) Reject those with eigenvalues less than some given threshold, leaving *L* dimensions in the data.

PCA for MNIST

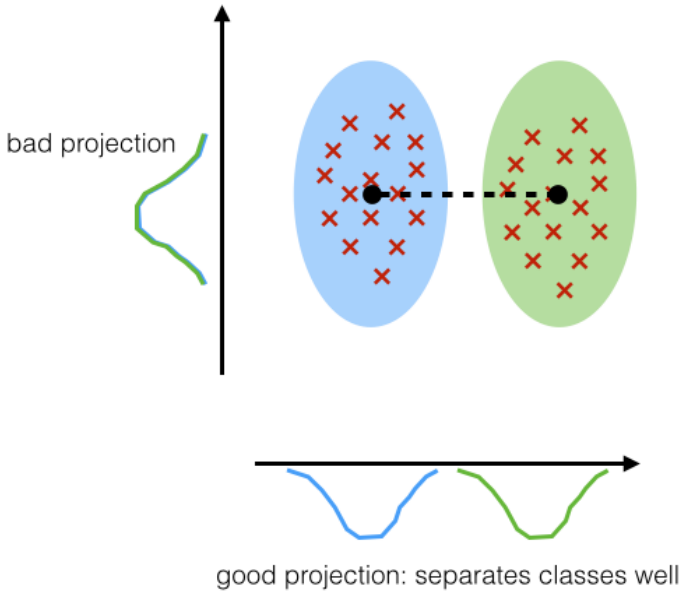


PCA vs. LDA

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



Extending PCA

Q: What strong assumptions did we make about the surface for the directions of maximum variation with PCA?

Extending PCA

Q: What strong assumptions did we make about the surface for the directions of maximum variation with PCA?

A: We assumed these surfaces of maximum variation are **straight lines** (this is a strong assumption!)

Q: How can are "break" the linear restriction for PCA?

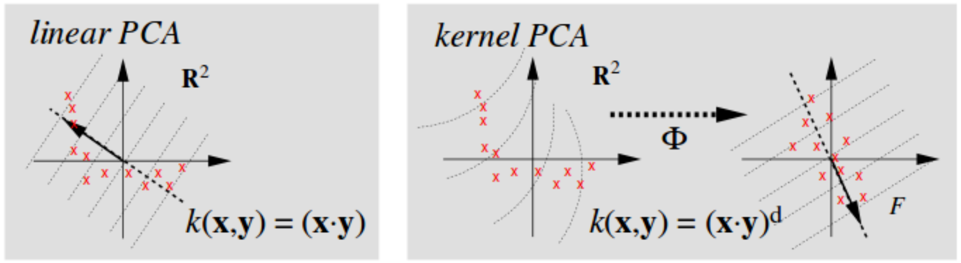
Extending PCA

Q: What strong assumptions did we make about the surface for the directions of maximum variation with PCA?

A: We assumed these surfaces of maximum variation are **straight lines** (this is a strong assumption!)

Q: How can are break the linear restriction for PCA?

A: **"Kernelize" PCA**!



linear PCA
$\mathbf{R}^2$
$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})$
kernel PCA
$\mathbf{R}^2$
Φ
$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^d$
F

Fig. 1. Basic idea of kernel PCA: by using a nonlinear kernel function k instead of the standard dot product, we implicitly perform PCA in a possibly high–dimensional space F which is nonlinearly related to input space. The dotted lines are contour lines of constant feature value.

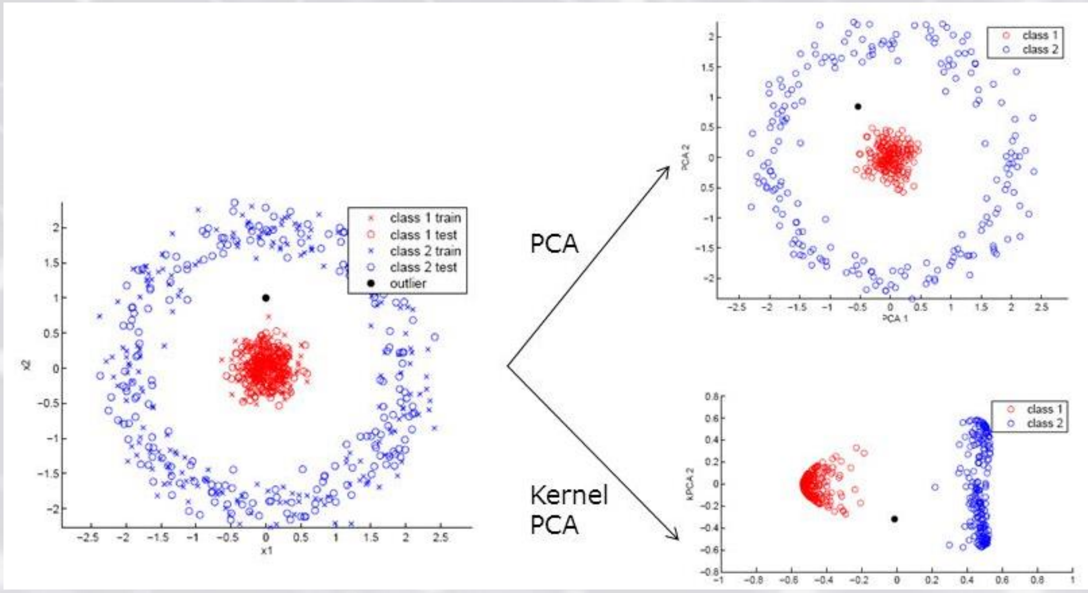
Kernel PCA

(*) All we have to do is express the covariance matrix **C** (recall this was the covariance of the data matrix X after centering) in terms of a kernel transformation:

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}\Phi\left(x_n\right)\bullet\Phi\left(x_n\right)^T$$

(*) Next we compute the *eigendecomposition* of **C** and use the eigenvectors with the largest associated eigenvalues for PCA.

(*) Recall (from SVM lecture) that by using a kernel function we implicitly perform a dot product in a larger dimensional feature space (this is the crux of the **kernel trick**), with the upshot of enhanced expressiveness.



PCA & Auto-encoders

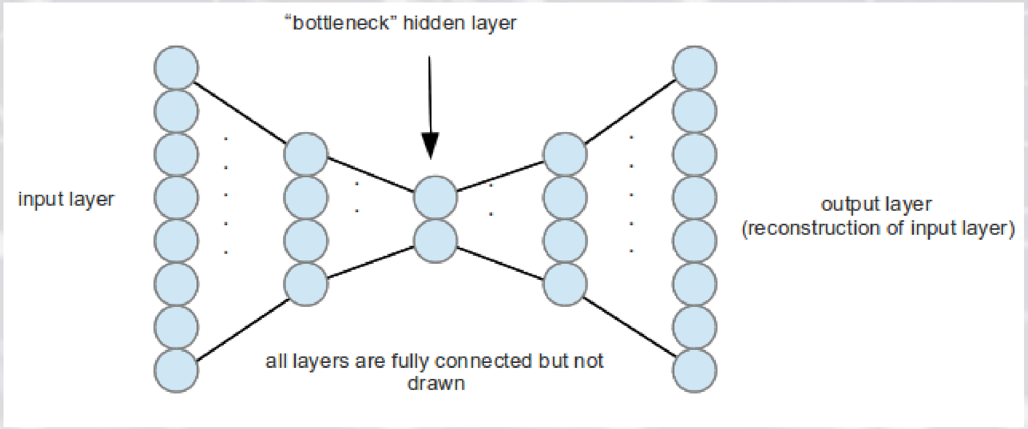
- Note that PCA is intimately connected with MLPs.

(*) An MLP can perform (non-linear) PCA using what is called an *auto-associator* (more commonly: **auto-encoder**).

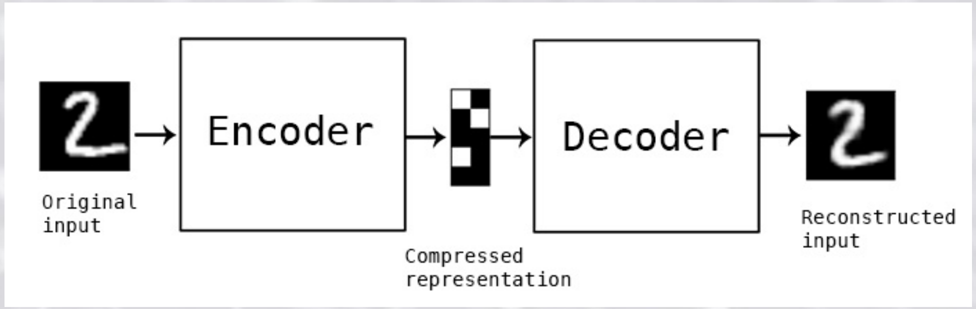
(*) If we train the MLP **where the output equals the input**, we are asking the network to learn a data "reconstruction" process; we therefore train to *minimize the reconstruction error*.

(*) Usually the hidden layers are smaller in dimension than the output/input layers so that they form a **compression "bottleneck"**.

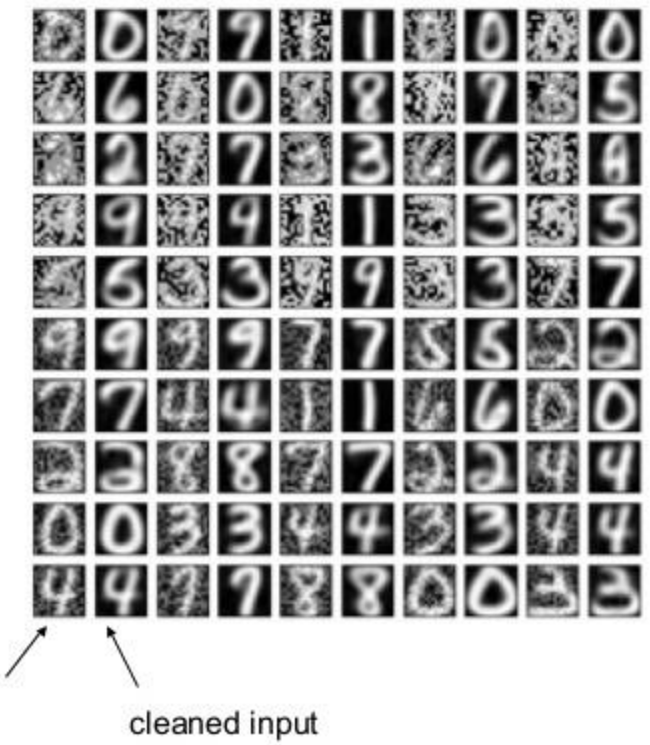
(*) The activations at the hidden layers (i.e. the feature vectors) encode a *dimensionality reduction* of the data.


"bottleneck" hidden layer
input layer
output layer
(reconstruction of input layer)
all layers are fully connected but not drawn

PCA & Auto-encoders: Image Denoising



The image shows how a "denoising" autoencoder may be used to generate correct input from corrupted input.



SVD

- Of the vast array of different matrix factorizations used in applied math, the **singular value decomposition** (SVD) is one of the most common and useful (we show its conceptual connection with PCA in the subsequent slides).

- Recall that a *symmetric matrix A* admits of an eigendecomposition. Notably, if A is **not** symmetric, then there is no guarantee that it has an eigendecomposition – so not every matrix can be factored in this way.

When it came to our previous discussion of PCA, we had $A=XX^T$.

Now, is *A* symmetric?

SVD

- Of the vast array of different matrix factorizations used in applied math, the **singular value decomposition** (SVD) is one of the most common and useful (we show its conceptual connection with PCA in the subsequent slides).

- Recall that a *symmetric matrix* ***A*** admits of an eigendecomposition. Notably, if **A** is **not** symmetric, then there is no guarantee that it has an eigendecomposition – so not every matrix can be factored in this way.

When it came to our previous discussion of PCA, we had $\mathbf{A}=\mathbf{X}\mathbf{X}^T$.

Now, is A symmetric? **Yes**: $\mathbf{A}^T=(\mathbf{X}\mathbf{X}^T)^T=(\mathbf{X}^T)^T\mathbf{X}^T=\mathbf{X}\mathbf{X}^T=\mathbf{A}$.

The PCA algorithm was crucially reliant on the fact that **A** was symmetric and thus it has an eigendecomposition.

SVD

- Conversely, *every* matrix has a singular value decomposition!

Definition: For an *m × n* matrix **A**, the *singular values* of **A** are the square roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$. They are denoted:

$$\sigma_1, \ldots, \sigma_n$$

It is conventional to arrange the singular values in decreasing order, whence: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$

Example:

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

has eigenvalues $\lambda_1=3$ and $\lambda_2=1$. Consequently, the singular values of **A** are:

$$\sigma_1 = \sqrt{\lambda_1} = \sqrt{3}$$

$$\sigma_2 = \sqrt{\lambda_2} = 1$$

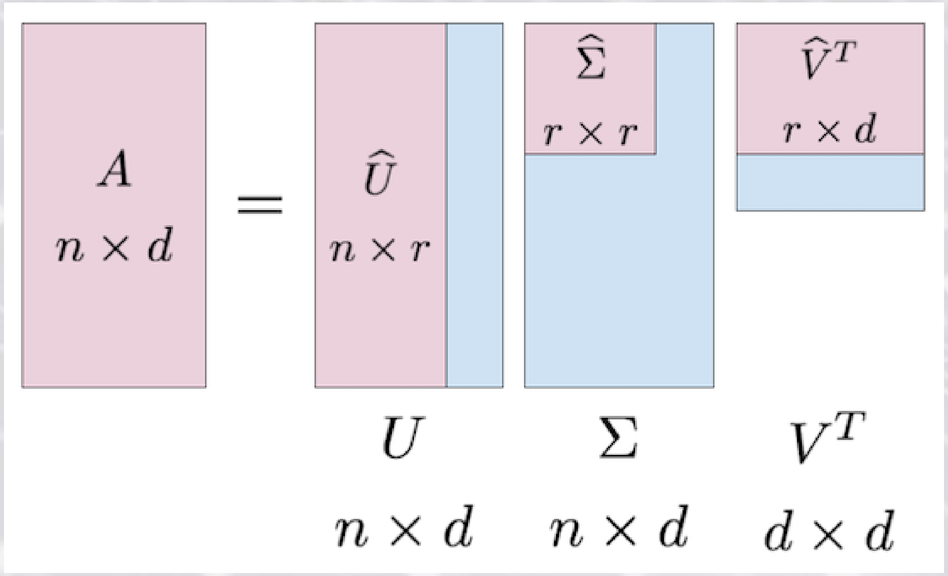
SVD

- Definition: Let A be an *m x n* matrix with singular values, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ and $\sigma_{r+1} = \sigma_{r+2} = \ldots = \sigma_n = 0$. Then there exist an *m x n* orthogonal matrix **U**, and *n x n* orthogonal matrix **V**, and an *m x n diagonal* matrix **Σ** of the form:

$$\boxed{A = U\Sigma V^T}$$

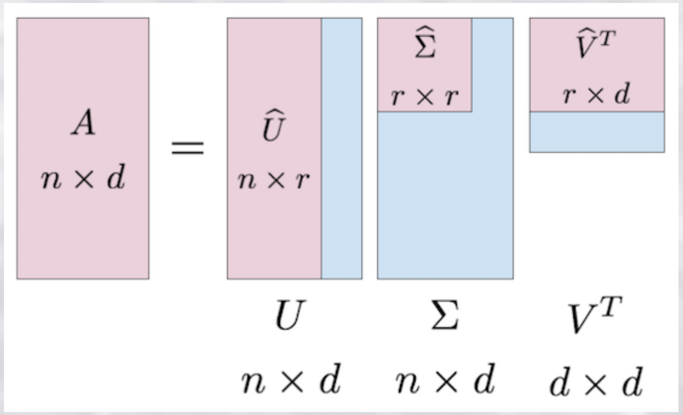
Note: the columns of **U** are called *left singular vectors* of **A**, and the columns of **V** are called *right singular vectors* of **A**. The matrices **U** and **V** are not uniquely determined by **A**.

(*) *NB*: **rank(A) = r**.



SVD

$$A = U\Sigma V^T$$



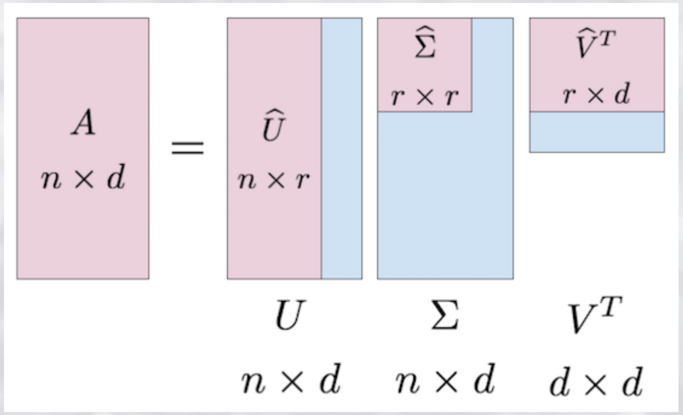
Example:

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

SVD

$$A = U\Sigma V^T$$



Example:

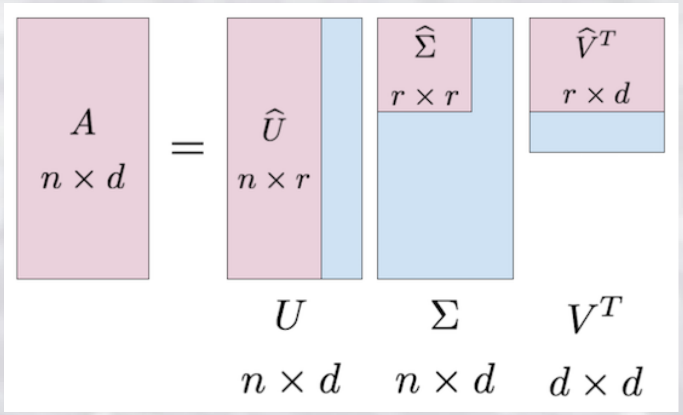
$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$eigenvalues\left(A^T A\right): \lambda_1 = 2, \lambda_2 = 1, \lambda_3 = 0$$

SVD

$$A = U\Sigma V^T$$



Example:

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

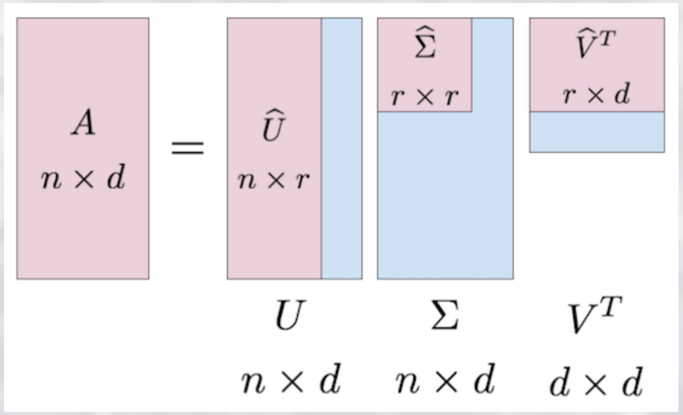
$$eigenvalues\left(A^T A\right): \lambda_1 = 2, \lambda_2 = 1, \lambda_3 = 0 \qquad eigenvectors\left(A^T A\right): \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$

These vectors are orthogonal, so now we normalize them:

$$V = \begin{bmatrix} 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 1/\sqrt{2} & 0 & 1/\sqrt{2} \\ 0 & 1 & 0 \end{bmatrix}, \Sigma = \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

SVD

$$A = U\Sigma V^T$$



Example: $A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$eigenvalues\left(A^T A\right): \lambda_1 = 2, \lambda_2 = 1, \lambda_3 = 0$$

$$eigenvectors\left(A^T A\right): \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$

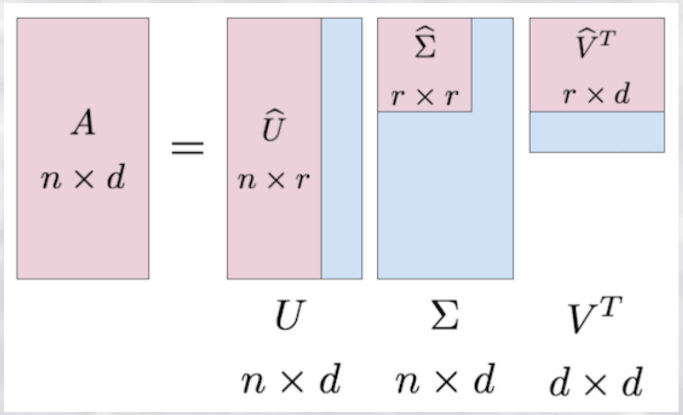
$$V = \begin{bmatrix} 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 1/\sqrt{2} & 0 & 1/\sqrt{2} \\ 0 & 1 & 0 \end{bmatrix}, \Sigma = \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

To find **U** we compute:

$$u_1 = \frac{1}{\sigma_1} A v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad u_2 = \frac{1}{\sigma_2} A v_2 = \frac{1}{1} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

SVD

$$A = U\Sigma V^T$$



Example:

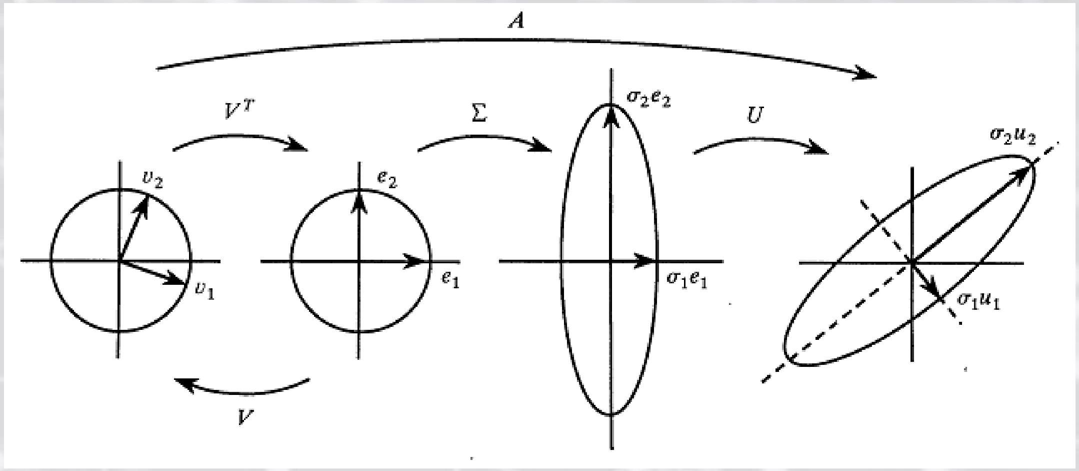
$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \end{bmatrix} = U\Sigma V^T$$

SVD

Geometric Interpretation: In general, **Σ** can be regarded as a scaling matrix, and **U**, **V** can be viewed as rotation matrices.

$$A = U\Sigma V^T$$

Thus the expression **UΣV** can be intuitively interpreted as a **composition of three successive geometrical transformations**: a rotation or reflection, a scaling and another rotation or reflection.



As shown in the figure, the singular values can be interpreted as the **semiaxes of an ellipse in 2D**. This concept can be generalized to *n*-dimensional *Euclidean space*, with the singular values of any $n \times n$ square matrix being viewed as the semiaxes of an *n*-dimensional ellipsoid.

As in PCA, these coordinate axes provide a natural framework for determining a dimensionality reduction scheme that captures maximal variation.

SVD: Outer Product Form

- SVD factorization yields a useful method for "low rank" approximations/dimensionality reduction of data.

Theorem: For a given SVD decomposition of an $m \times n$ matrix **A**, we can express **A** in the so-called **outer product form**:

$$\boxed{\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \ldots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T}$$

Where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ denote the singular values of **A**; **u** and **v** are the corresponding *left singular* and *right singular vectors*.

(*) Note that the **condition number** of a matrix **A** is defined as the ratio of the largest and the smallest singular values of **A**. Matrices with large condition numbers are called **ill-conditioned** (this has a significant impact on the stability of many different kinds of numerical algorithms in linear algebra).

$$cond(A) = \frac{\sigma_{\max}}{\sigma_{\min}}$$

SVD: Outer Product Form

$$\boxed{\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \ldots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T}$$

Example:

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \end{bmatrix} = U\Sigma V^T$$

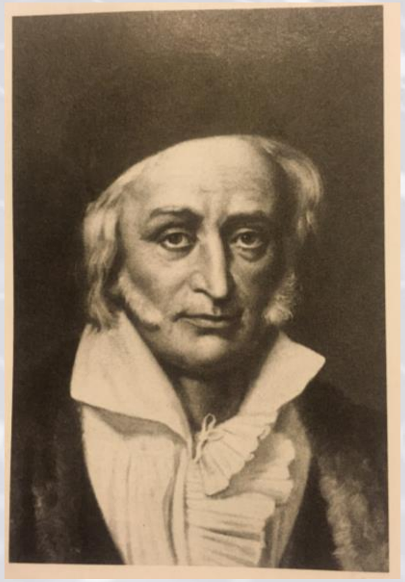
$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \sqrt{2} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \end{bmatrix} + 1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

SVD: Outer Product Form for Image Compression

• Consider the task of compressing a grayscale image of dimension *340 x 280*; each pixel is in the range [0, 255].

• We can store this image in a *340 x 280* dimension matrix, but transmitting and manipulating these 95,200 numbers is very expensive.

• Let's use SVD for efficient image compression. Recall that the small singular values in the SVD of a matrix correspond with "less informative" data features.



SVD: Outer Product Form for Image Compression

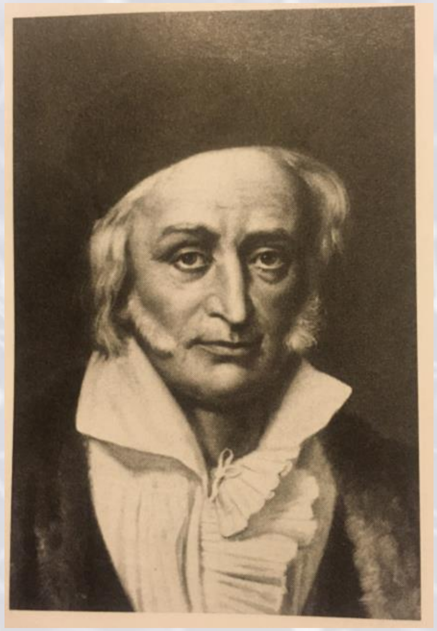
- Suppose we have the SVD of A expressed in outer product form:

$$\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \ldots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T$$

- For the original *340 x 280* image shown, we have r = 280 (why?).

- Define: $\mathbf{A}_k = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \ldots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T, \; k \le r$

  as the *k-rank approximation to **A***.



SVD: Outer Product Form for Image Compression

- Suppose we have the SVD of A expressed in outer product form:

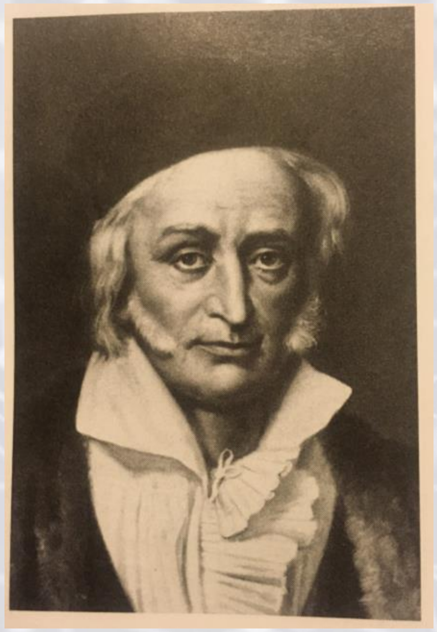
$$\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \ldots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T$$

- For the original *340 x 280* image shown, we have r = 280 (why?).

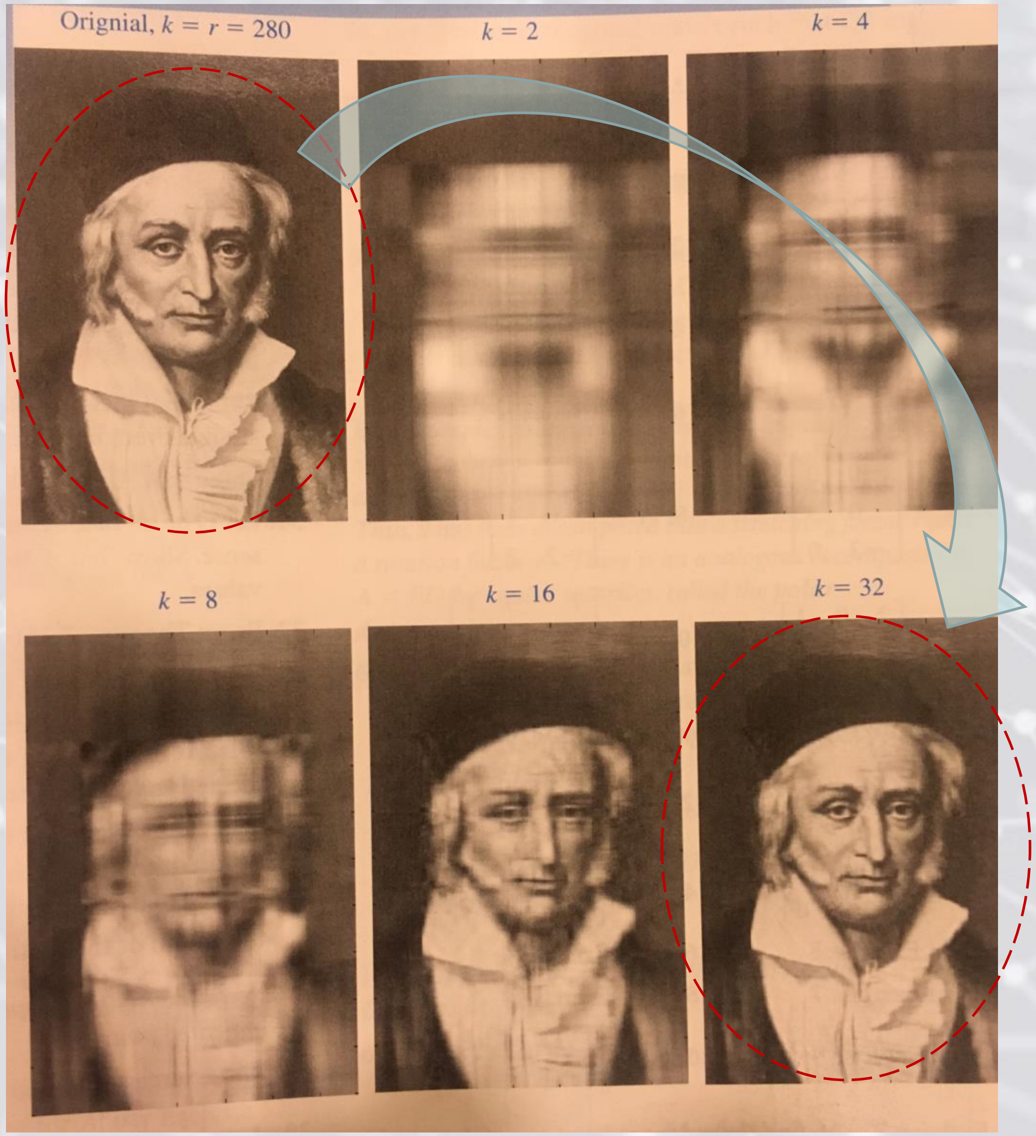
- Define: $\mathbf{A}_k = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \ldots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T, \; k \le r$

  as the *k-rank approximation to* ***A***.

(*) If for example, we use a **k = 20** *rank approximation* for **A** (i.e. we use the largest 20 singular values), the storage/ computational overhead is <u>reduced from 95,200 numbers to 12,420</u>!



SVD: Outer Product Form for Image Compression



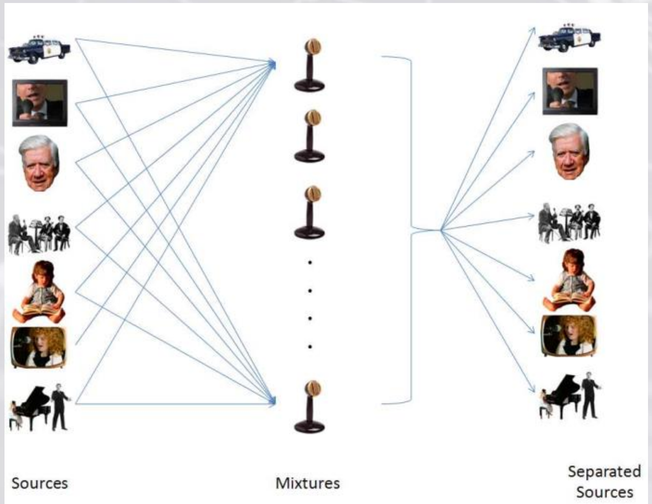
$$\mathbf{A}_k = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \ldots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T, \; k = 32$$

(*) Here, using the SVD-based, low-rank approximation to **A**, the fidelity of the image is very strong – even after discarding roughly 85% of the image data!

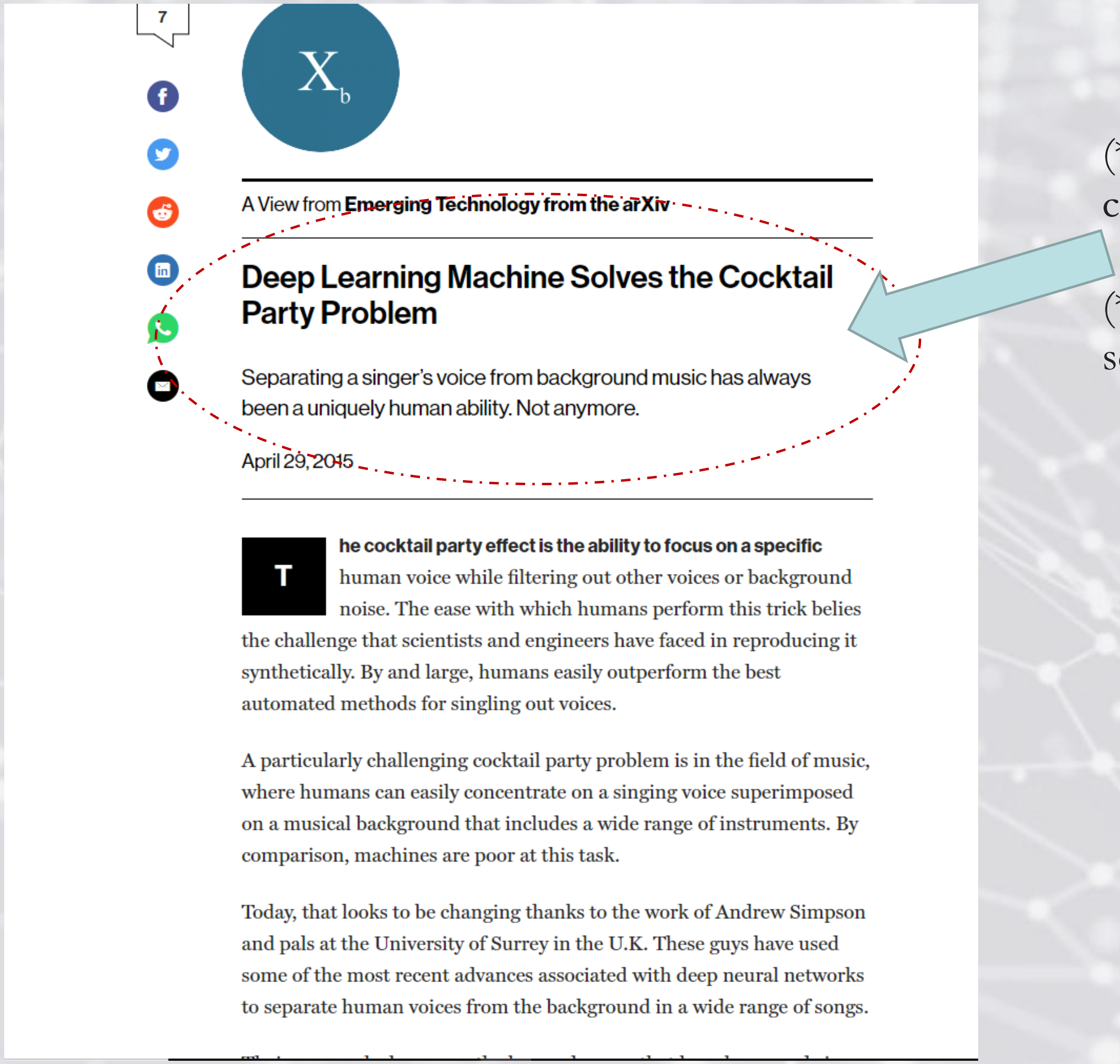
SVD: Audio Denoising & the "cocktail party problem"

- The **cocktail party effect** is the phenomenon of the brain's ability to focus one's auditory attention (an effect of selective attention in the brain) on a particular stimulus while filtering out a range of other stimuli.

- Listeners have the ability to both <u>segregate different stimuli into different streams</u>, and subsequently decide which streams are most pertinent to them. Thus, it has been proposed that one's sensory memory subconsciously siphons through all stimuli, and when an important word or phrase with high meaning appears, it stands out to the listener.
- This effect is what allows most people to "tune into" a single voice and "tune out" all others.



SVD: Audio Denoising & the "cocktail party problem"



7

A View from **Emerging Technology from the arXiv**

Deep Learning Machine Solves the Cocktail Party Problem

Separating a singer's voice from background music has always been a uniquely human ability. Not anymore.

April 29, 2015

The cocktail party effect is the ability to focus on a specific human voice while filtering out other voices or background noise. The ease with which humans perform this trick belies the challenge that scientists and engineers have faced in reproducing it synthetically. By and large, humans easily outperform the best automated methods for singling out voices.

A particularly challenging cocktail party problem is in the field of music, where humans can easily concentrate on a singing voice superimposed on a musical background that includes a wide range of instruments. By comparison, machines are poor at this task.

Today, that looks to be changing thanks to the work of Andrew Simpson and pals at the University of Surrey in the U.K. These guys have used some of the most recent advances associated with deep neural networks to separate human voices from the background in a wide range of songs.

(*) Recent research with DNNs claims to have solved the problem.

(*) Another approach: use SVD to separate signal from noise!

PCA & SVD

Q: How do PCA and SVD relate?

- In fact, we can use SVD to perform PCA.

- Given a data matrix **X**, perform the SVD decomposition:

$$X = U\Sigma V^T$$

PCA & SVD

Q: How do PCA and SVD relate?

- In fact, we can use SVD to perform PCA.

- Given a data matrix **X**, perform the SVD decomposition:

$$X = U\Sigma V^T$$

- Next, we compute the covariance **C** of **X** (where the data is assumed centered):

$$C = XX^T = U\Sigma V^T \left(U\Sigma V^T\right)^T = U\Sigma V^T V\Sigma^T U^T$$

PCA & SVD

- Given a data matrix **X**, perform the SVD decomposition:

$$X = U\Sigma V^T$$

- Next, we compute the covariance **C** of **X** (where the data is assumed centered):

$$C = XX^T = U\Sigma V^T \left(U\Sigma V^T\right)^T = U\Sigma V^T V \Sigma^T U^T$$

$$= U\Sigma\left(V^T V\right)\Sigma U^T$$

Why?

PCA & SVD

- Given a data matrix **X**, perform the SVD decomposition:

$$X = U\Sigma V^T$$

- Next, we compute the covariance **C** of **X** (where the data is assumed centered):

$$C = XX^T = U\Sigma V^T \left(U\Sigma V^T\right)^T = U\Sigma V^T V \Sigma^T U^T$$

$$= U\Sigma\left(V^T V\right)\Sigma U^T = U\Sigma\left(V^T V\right)\Sigma U^T$$

Why?

PCA & SVD

- Given a data matrix **X**, perform the SVD decomposition:

$$X = U\Sigma V^T$$

- Next, we compute the covariance **C** of **X** (where the data is assumed centered):

$$C = XX^T = U\Sigma V^T \left(U\Sigma V^T\right)^T = U\Sigma V^T V \Sigma^T U^T$$

$$= U\Sigma\left(V^T V\right)\Sigma U^T = U\Sigma\left(V^T V\right)\Sigma U^T = U\Sigma^2 U^T$$

Why?

PCA & SVD

- Given a data matrix **X**, perform the SVD decomposition:

$$X = U\Sigma V^T$$

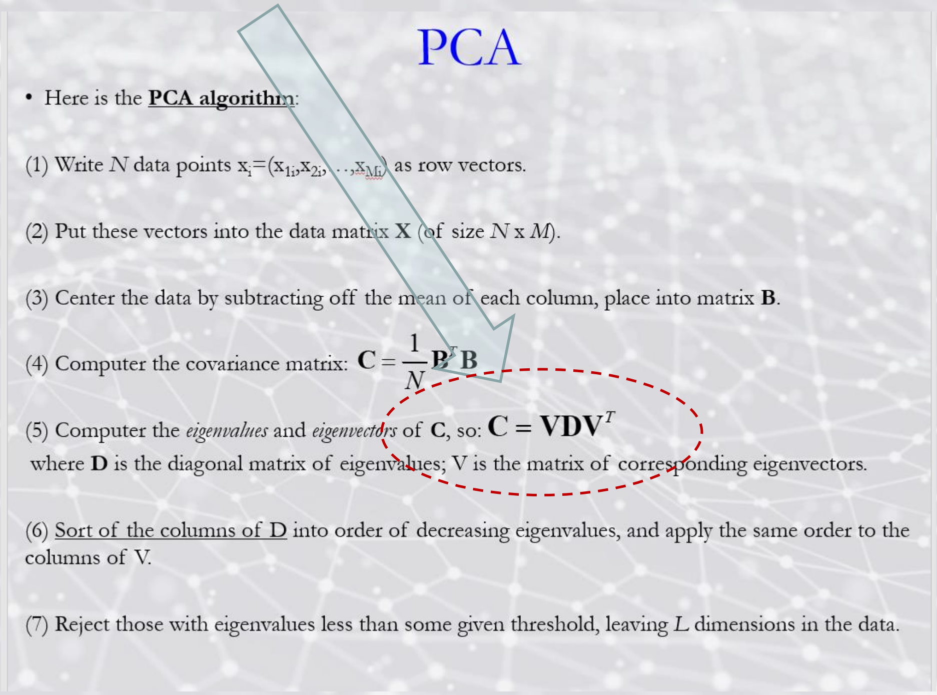
- Next, we compute the covariance **C** of **X** (where the data is assumed centered):

$$C = XX^T = U\Sigma V^T\left(U\Sigma V^T\right)^T = U\Sigma V^T V\Sigma^T U^T$$

$$= U\Sigma\left(V^T V\right)\Sigma U^T = U\Sigma\left(V^T V\right)\Sigma U^T = U\Sigma^2 U^T$$

(*) Recall that with the PCA *eigendecomposition* algorithm we generate the of C.

(*) This is precisely what we've done here, where: D=Σ^2, which confirms that the singular values of X are indeed equivalent to the square root of the eigenvalues of XX^T.



PCA

- Here is the **PCA algorithm**:

(1) Write N data points $x_i=(x_{1i},x_{2i},\ldots,x_{Mi})$ as row vectors.

(2) Put these vectors into the data matrix **X** (of size N x M).

(3) Center the data by subtracting off the mean of each column, place into matrix **B**.

(4) Computer the covariance matrix: $\mathbf{C} = \frac{1}{N}\mathbf{B}^T\mathbf{B}$

(5) Computer the *eigenvalues* and *eigenvectors* of **C**, so: $\mathbf{C} = \mathbf{VDV}^T$
where **D** is the diagonal matrix of eigenvalues; V is the matrix of corresponding eigenvectors.

(6) Sort of the columns of D into order of decreasing eigenvalues, and apply the same order to the columns of V.

(7) Reject those with eigenvalues less than some given threshold, leaving L dimensions in the data.

Fin