10-605/10-805: Machine Learning with Large Datasets

FALL 2020

Principal Component Analysis

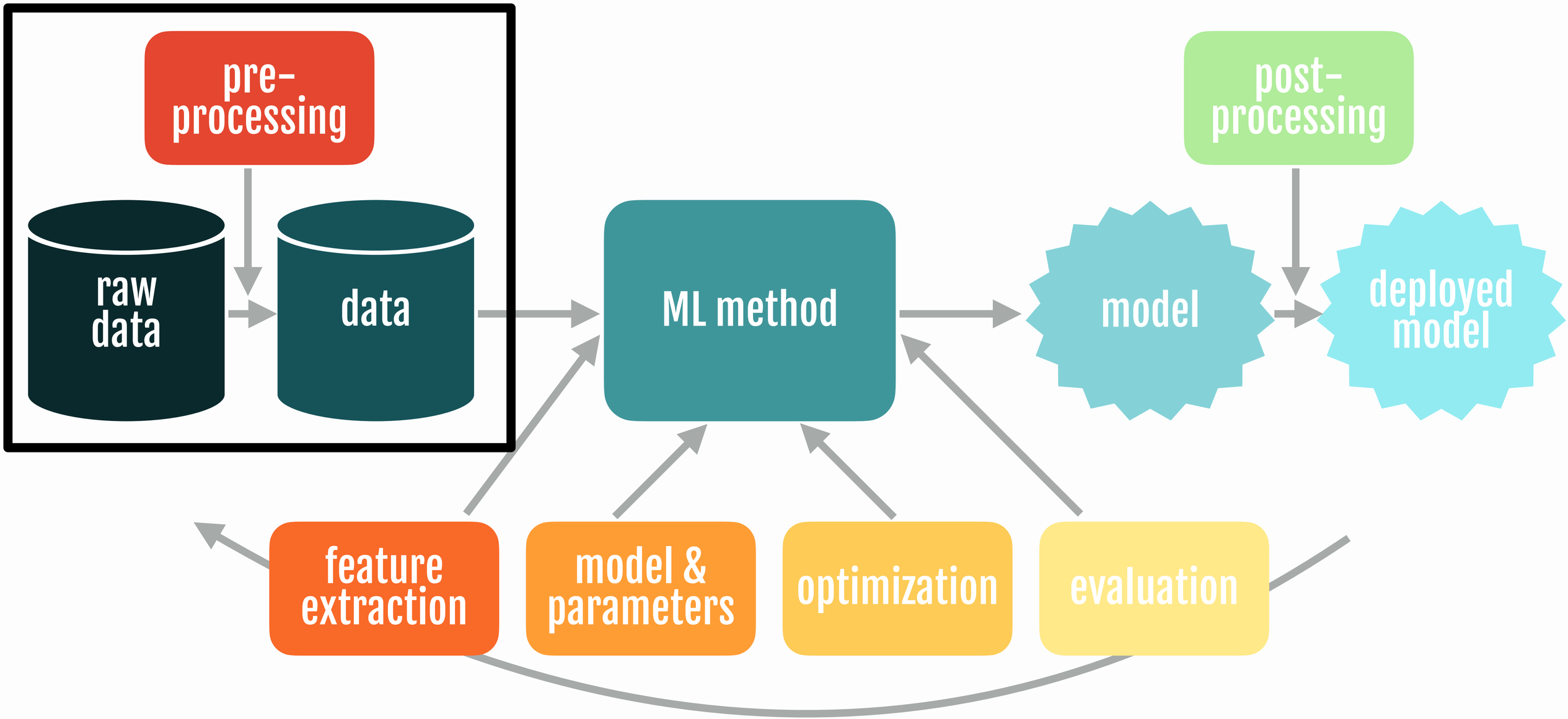
Announcements

- **HW1**: due *Tuesday September 15*, right before class (1:30pm ET)
  - get started if you haven't already!
  - please use piazza, office hours, recitation for questions
    - but don't provide direct answers on these forums

- **Questions about Spark/Databricks?**
  - Check out last Friday's recitation
  - This Friday's recitation will also cover additional details on Spark

RECALL

Machine learning pipeline



Key course topics

Data preparation
- Data cleaning
- Data summarization
- Visualization
- Dimensionality reduction

Training
- Distributed ML
- Large-scale optimization
- Scalable deep learning
- Efficient data structures
- Hyperparameter tuning

Inference
- Hardware for ML
- Techniques for low-latency inference (compression, pruning, distillation)

Infrastructure / Frameworks
- Apache Spark
- TensorFlow
- AWS / Google Cloud / Azure

Advanced topics
- Federated learning
- Neural architecture search
- Productionizing ML

Examples of data pre-processing

- ETL (extract, transform, load):
  - e.g., load data from text files into DataFrame
- Clean data
  - e.g., handle missing data, duplicated data, formatting issues, errors in data
- Label data [*see: upcoming guest lecture!*]
- Understand data:
  - Summarize data
  - Explore data
  - Visualize data
  - *Note: this is a critical step before applying a machine learning method*

Outline

1. Motivation: Data visualization
2. Principal Component Analysis (PCA)
3. Johnson-Lindenstrauss

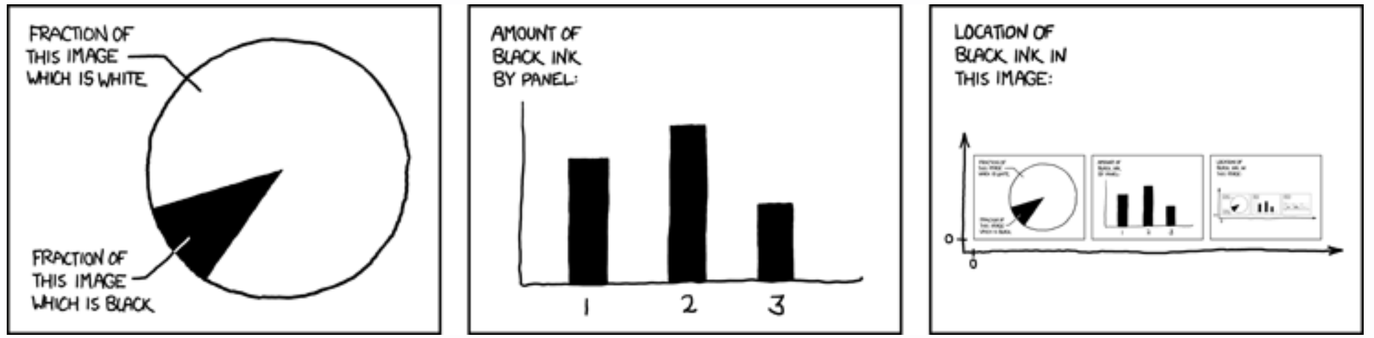
QUESTION FOR YOU

Suppose I have some data …

What questions do you have about the data?

- How was the data generated?
- What does the data *look like*? (What kind of data is it? What format is it in?)
- How much data is there?
- What do you want to do with the data?
- What patterns/trends exist in the data?
- What values does the data take (average, max, min, etc)?
- How do the features/labels relate to each other?

Why visualize data?



XKCD

- Gain insight about patterns in data (trends, relationships, biases, groups)
- Discover errors/issues with data (clean data)
- Figure out what questions to ask about data
- Determine how to model data

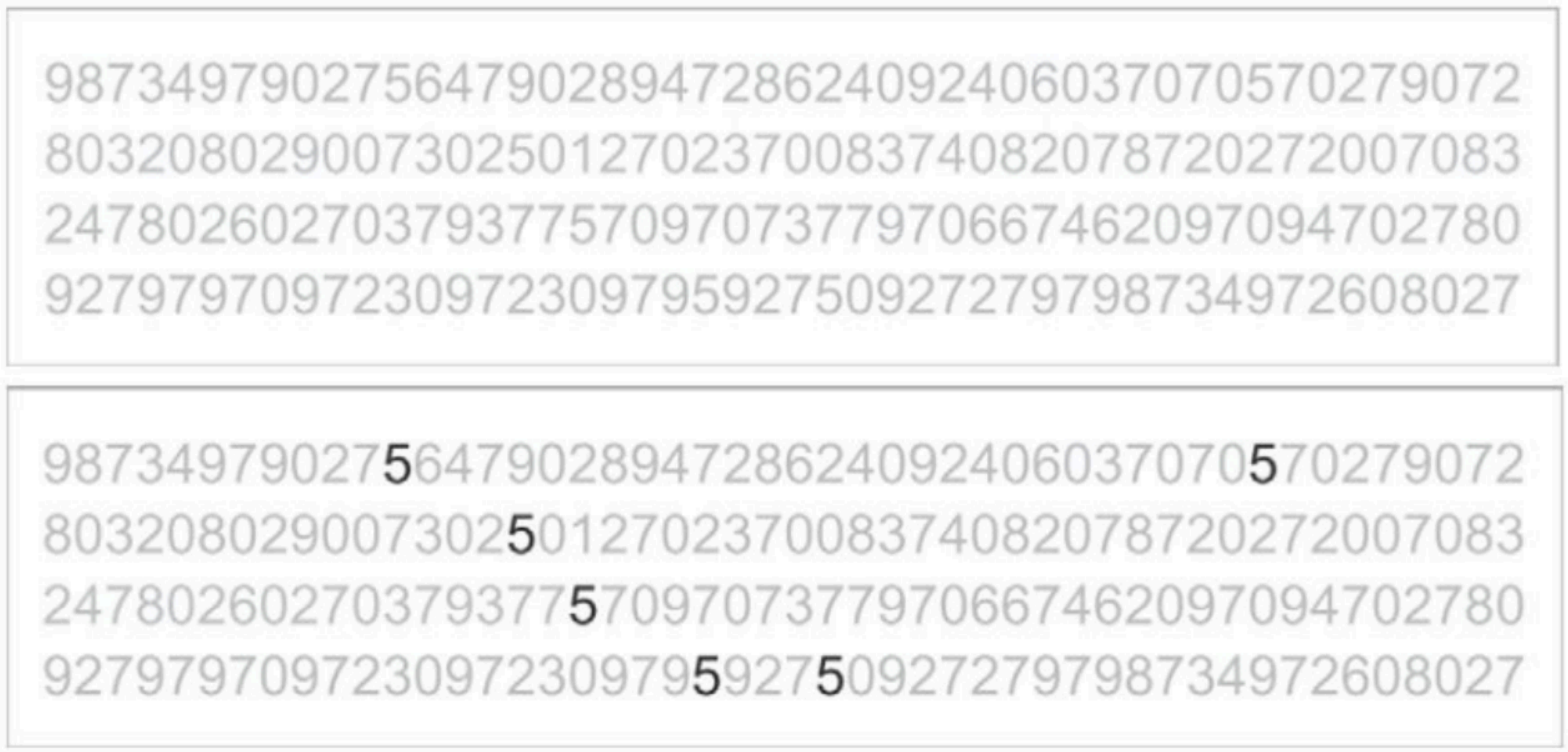
Note: visualization in ML can also be used to:

- Evaluate training (e.g., track convergence)
- Evaluate model performance
- Interpret / inspect model behavior
- Communicate results

EXAMPLE #1

Highlight/inspect data

How many 5s?

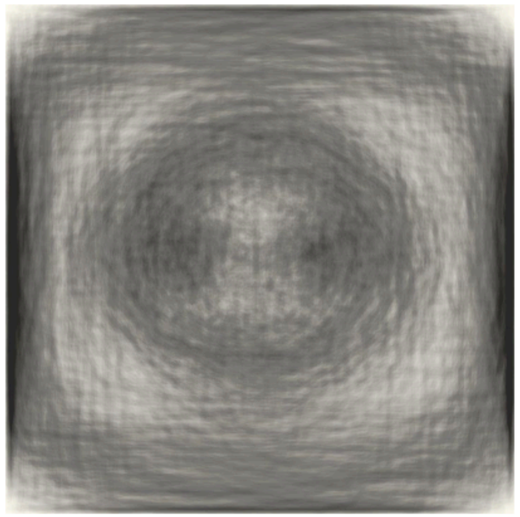


Viegas & Wattenberg, 2018

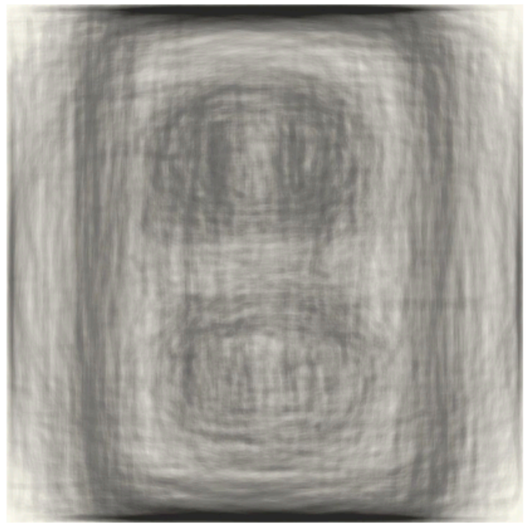
EXAMPLE #2

Group data

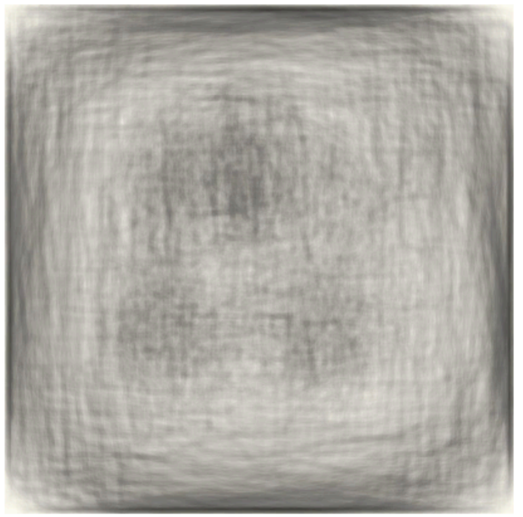
Quick, draw!: *https://quickdraw.withgoogle.com/*



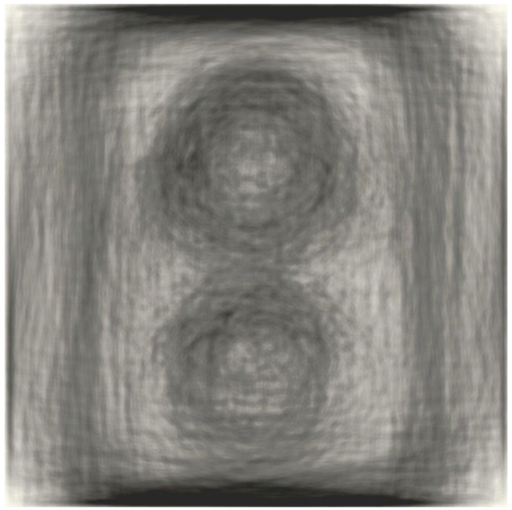
Germany



Japan



Malaysia

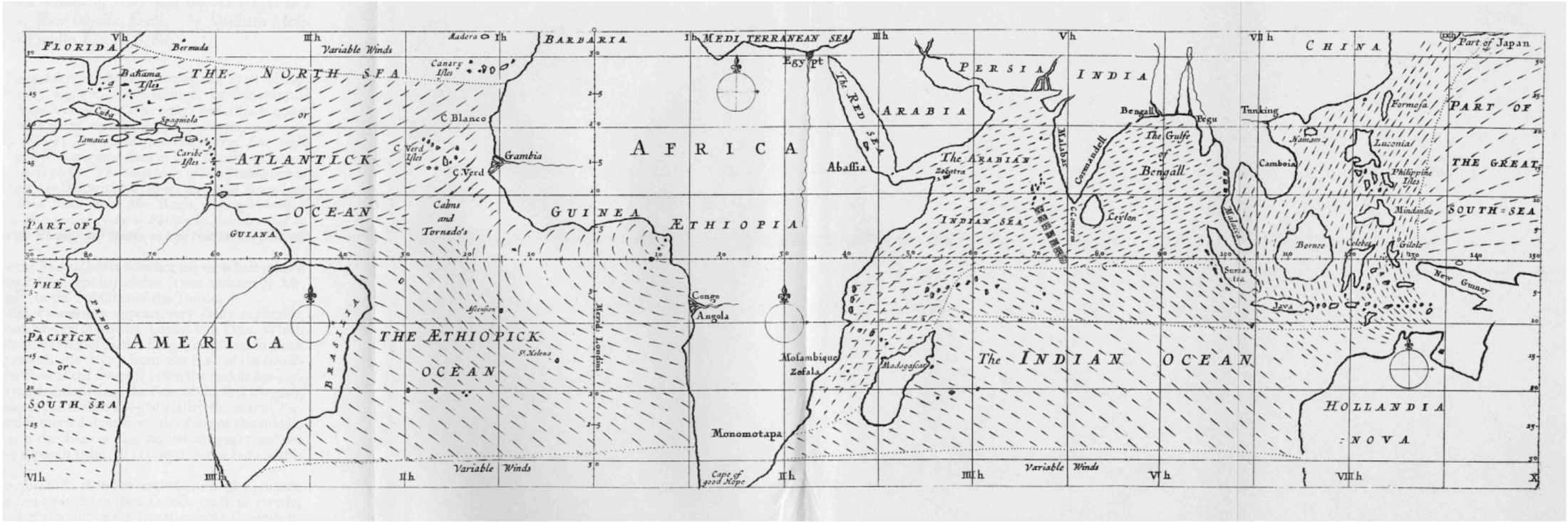


Sweden

EXAMPLE #3

Understand scale

Visualizing wind magnitude & patterns: http://hint.fm/wind/

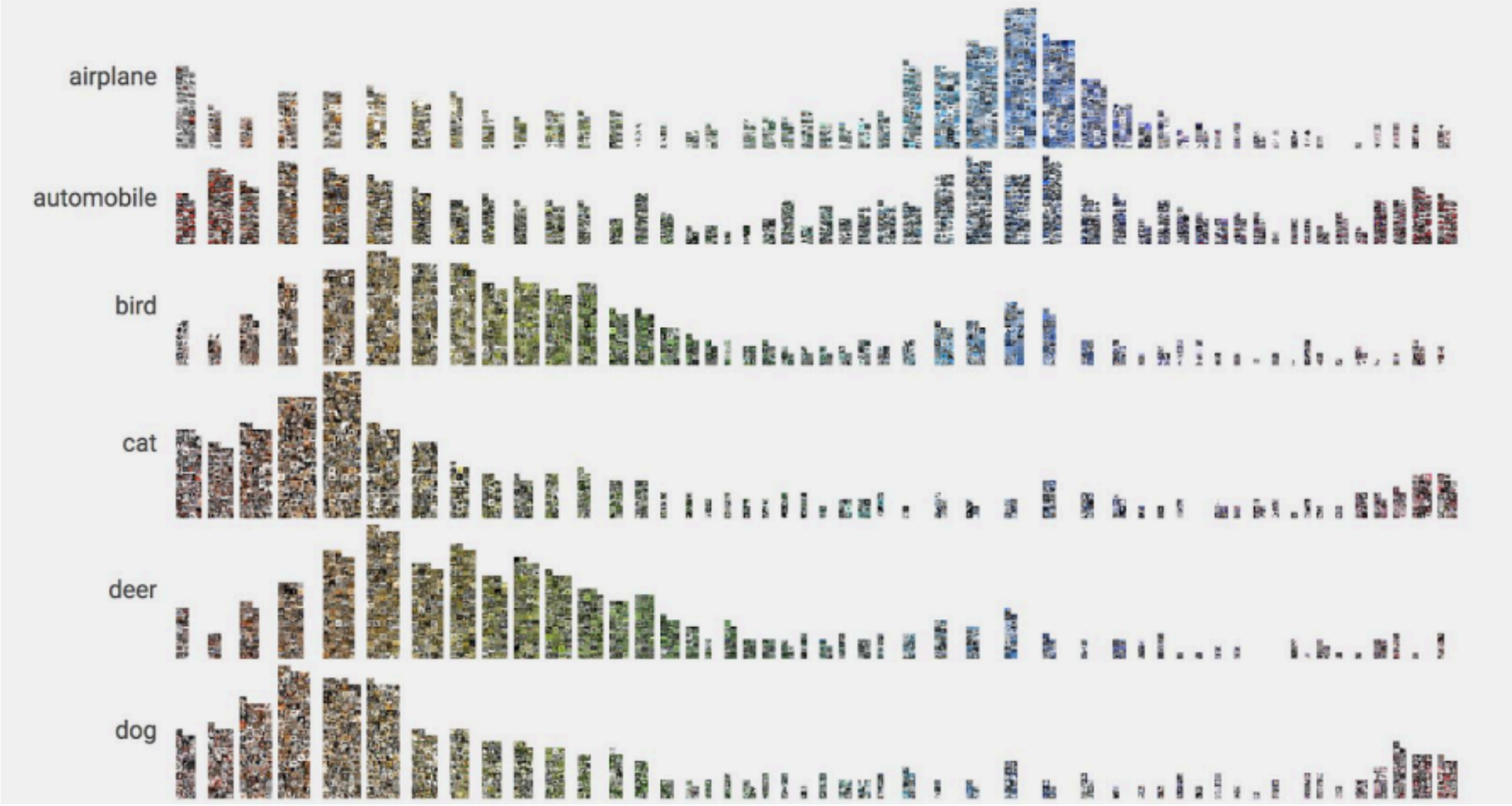


Edmund Halley, 1686

Air quality: nytimes.com/interactive/2019/12/02/climate/air-pollution-compare-ar-ul

EXAMPLE #4

Understand variation

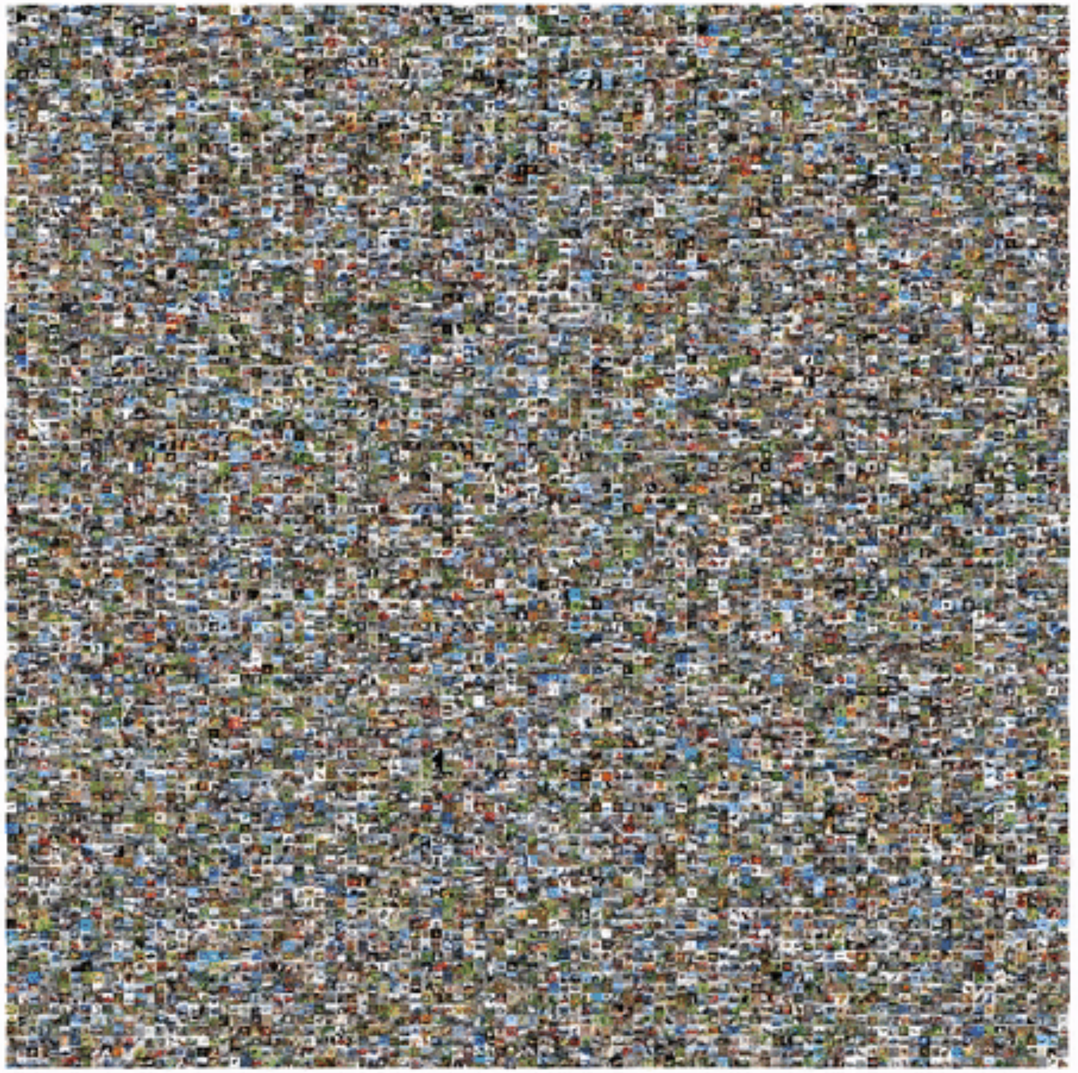


Facets, CIFAR-10

EXAMPLE #5

Zoom & filter

- Facets: https://pair-code.github.io/facets/
- Identifying incorrect labels in CIFAR-10 Dataset

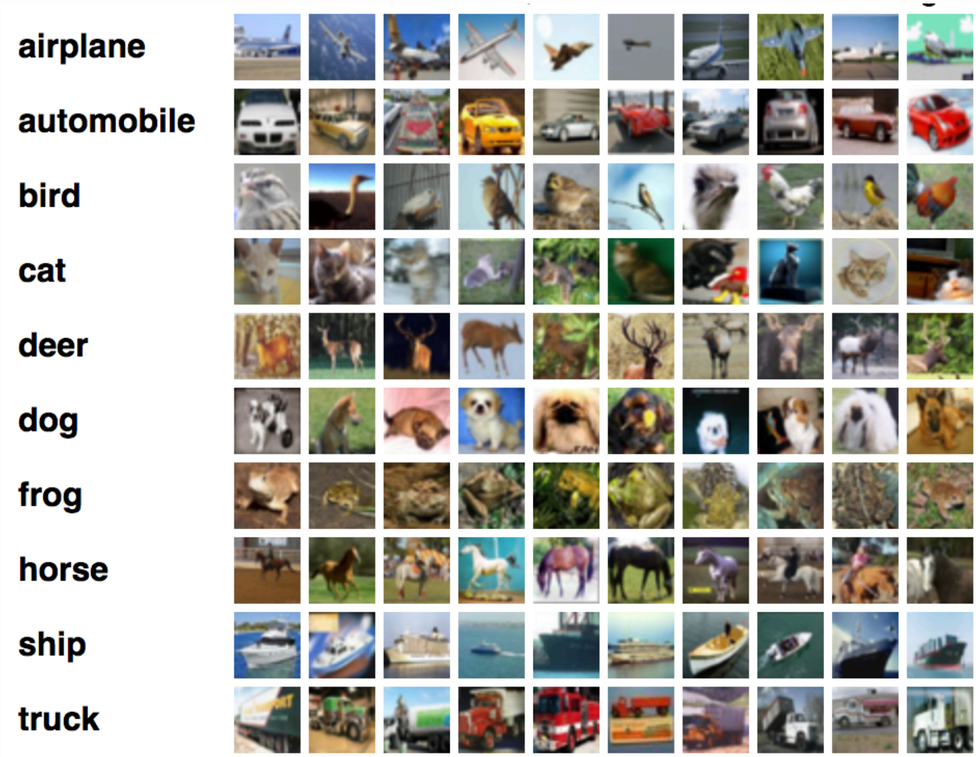


Examples of common visualizations

- View a small sample
- Table of summary statistics
- Box plot
- Histogram
- Scatter plot

Examples of common visualizations

- **View a small sample**
- Table of summary statistics
- Box plot
- Histogram
- Scatter plot




```
print('\n'.join(shakespeareRDD
                .zipWithIndex()  # to (line, lineNum)
                .map(lambda kv: '{0}: {1}'.format(kv[1], kv[0]))  # to 'lineNum: line'
                .take(15)))
```

Examples of common visualizations

- View a small sample
- **Table of summary statistics**
- Box plot
- Histogram
- Scatter plot

```
1 df = sqlContext.table("power_plant")
2 display(df.describe())
```

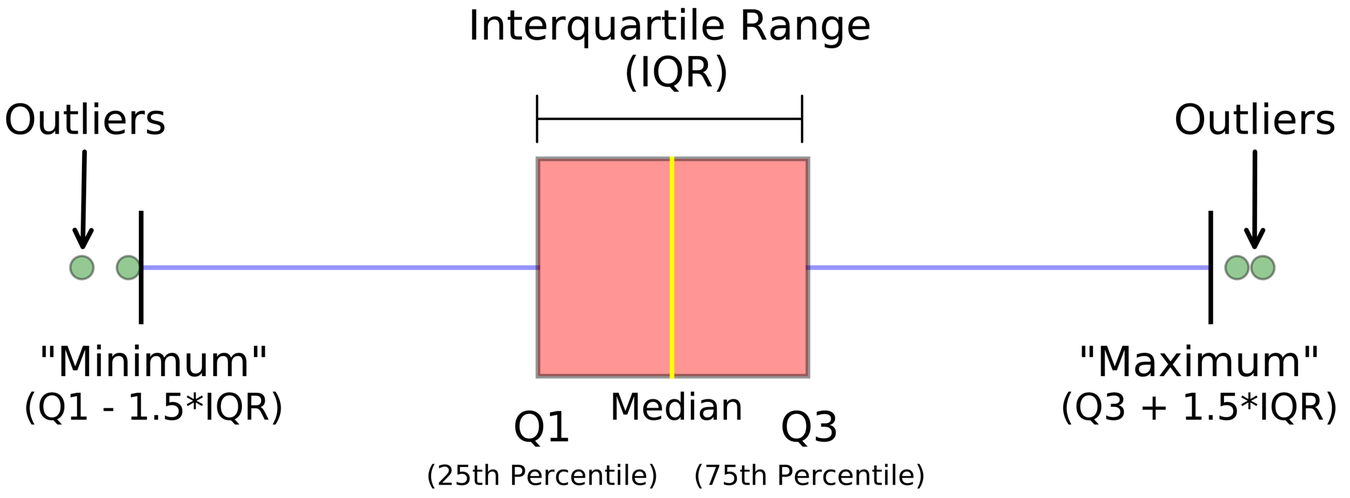
▸ (1) Spark Jobs

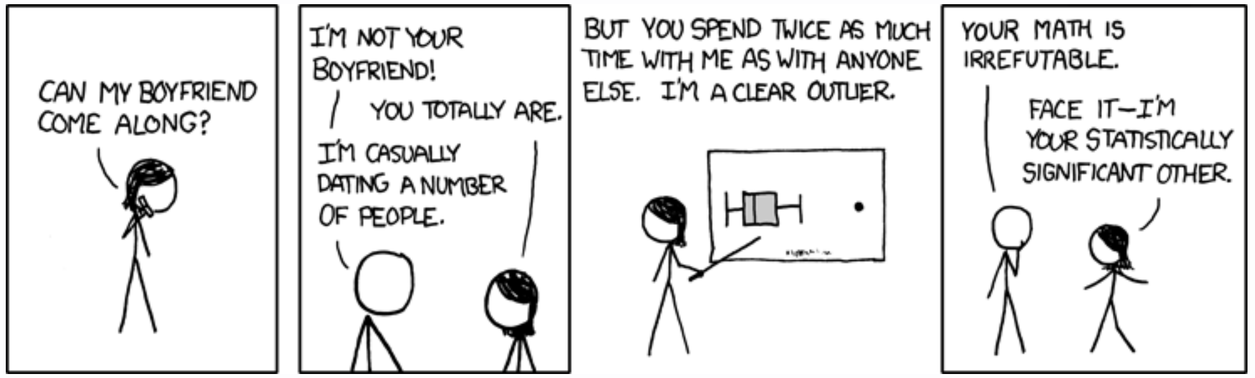
▸ df: pyspark.sql.dataframe.DataFrame = [AT: double, V: double ... 3 more fields]

summary	AT	V	AP
count	47840	47840	47840
mean	19.651231187290996	54.30580372073594	1013.2590781772572
stddev	7.452161658340004	12.707361709685806	5.938535418520816
min	1.81	25.36	992.89
max	37.11	81.56	1033.3

Examples of common visualizations

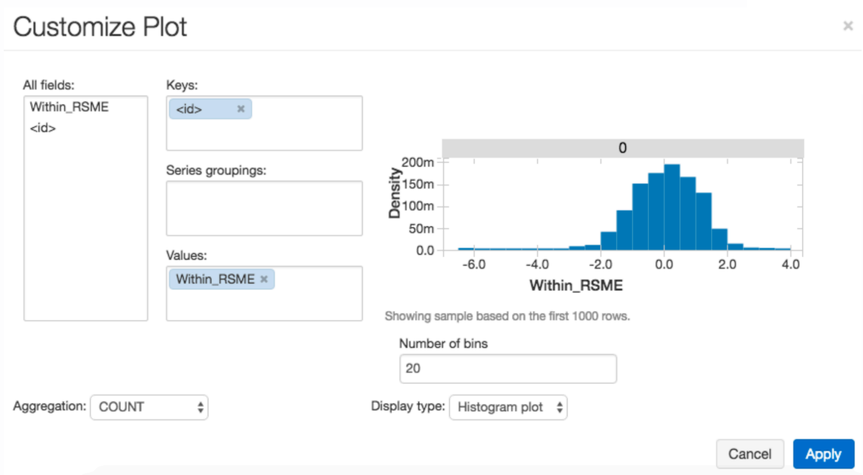
- View a small sample
- Table of summary statistics
- **Box plot**
- Histogram
- Scatter plot

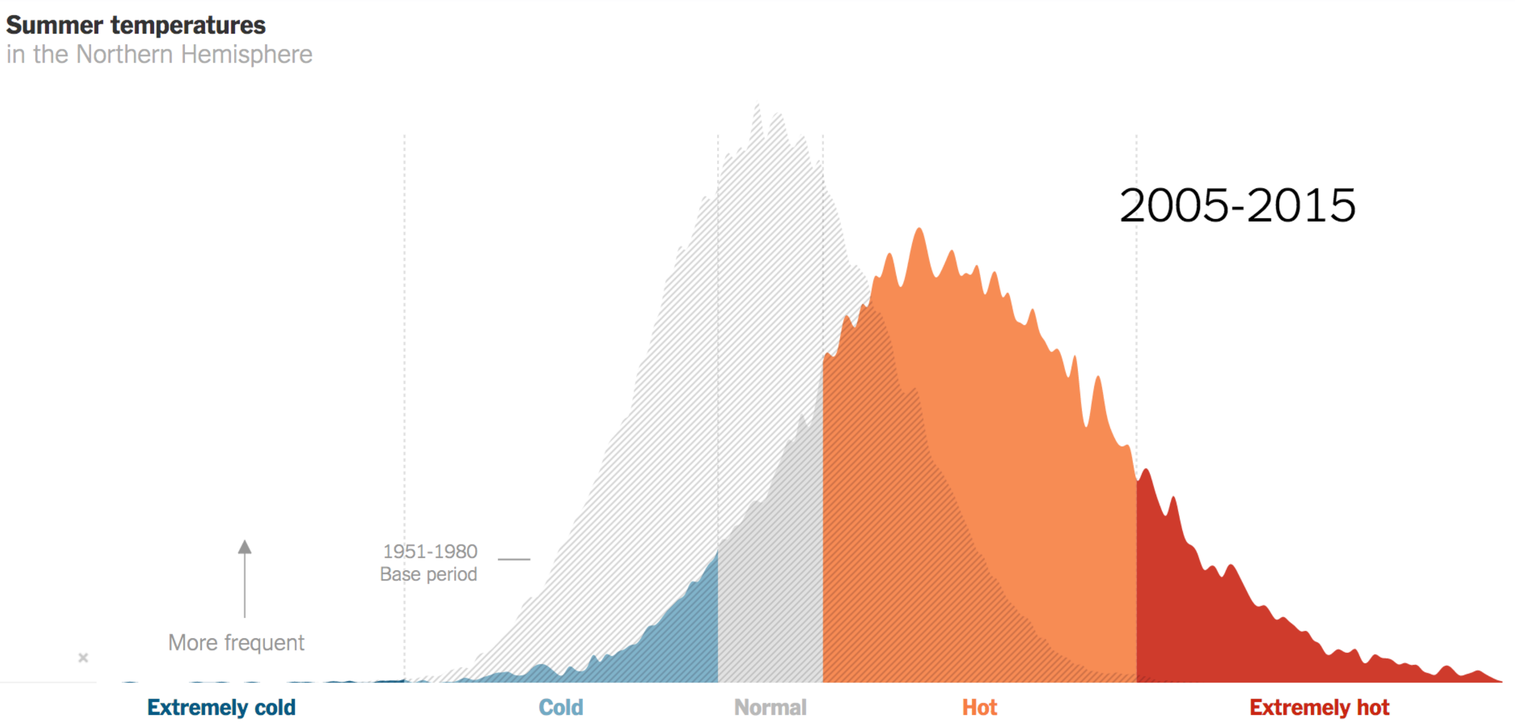




Examples of common visualizations

- View a small sample
- Table of summary statistics
- Box plot
- **Histogram**
- Scatter plot

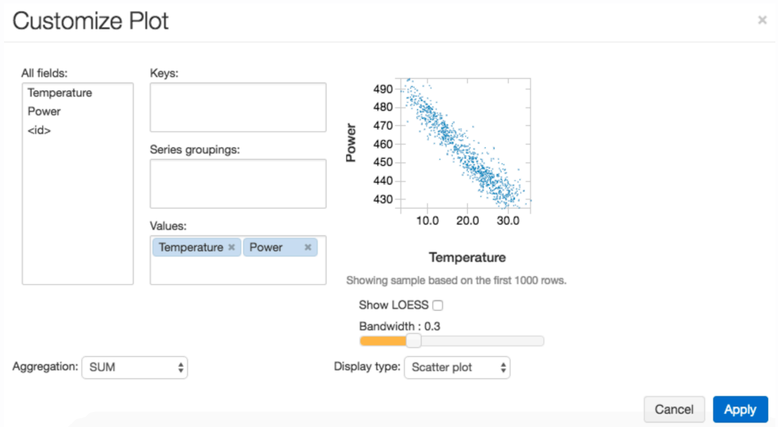


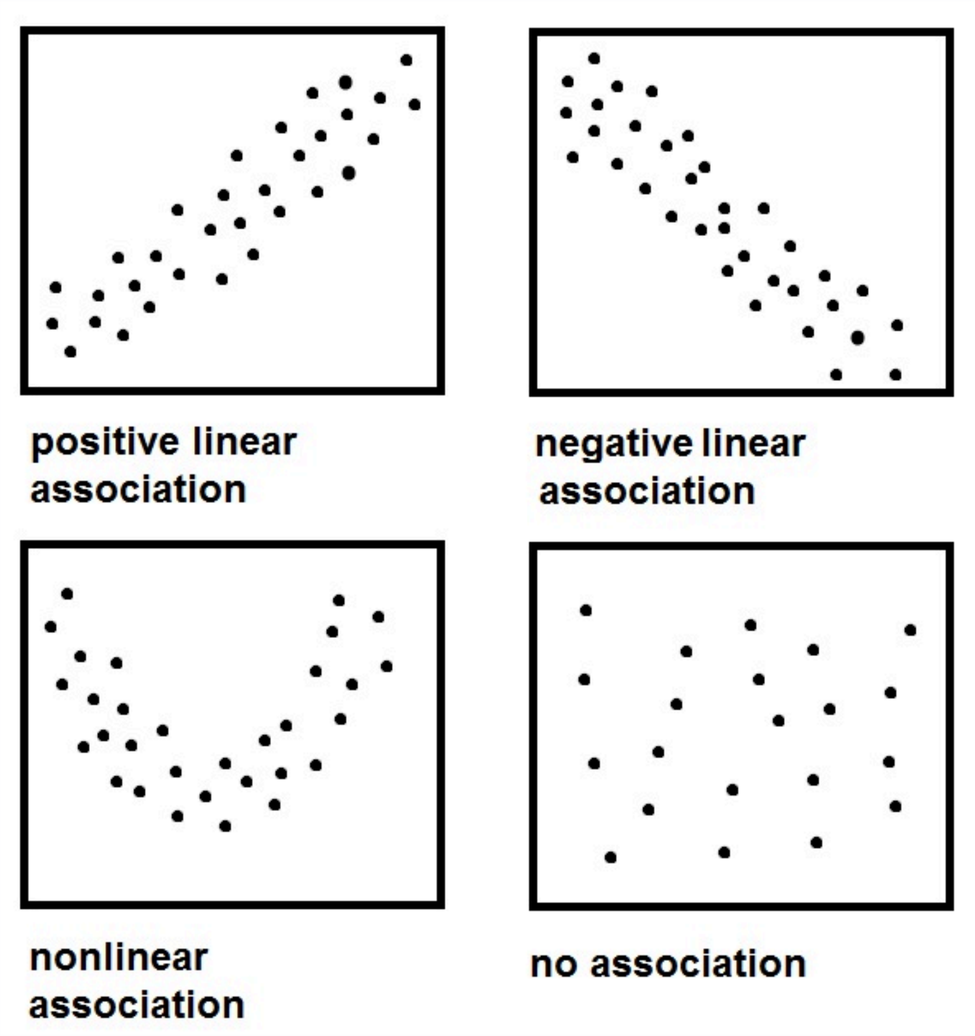


NYT

Examples of common visualizations

- View a small sample
- Table of summary statistics
- Box plot
- Histogram
- **Scatter plot**





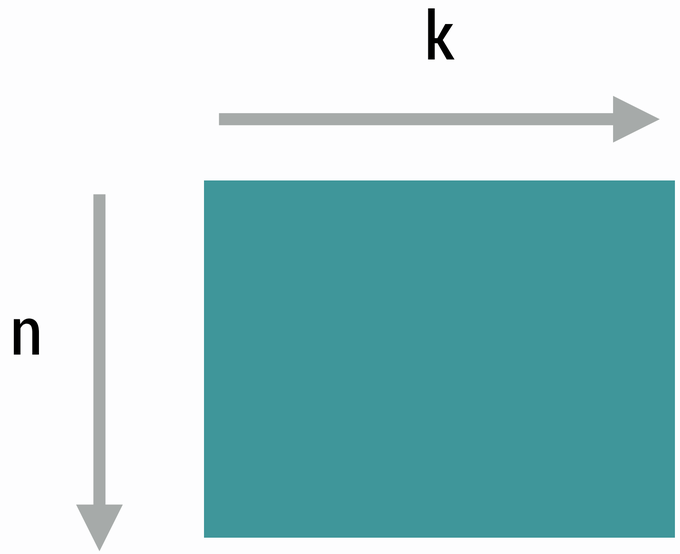
What's difficult about visualizing large datasets?

Large *k* (features)

- Difficult to visualize in high dimensions
- Key idea: dimensionality reduction
  - e.g., PCA, t-SNE

Large *n* (observations)

- Can be expensive, dense/complex
- Key ideas: distribute computation, subsample



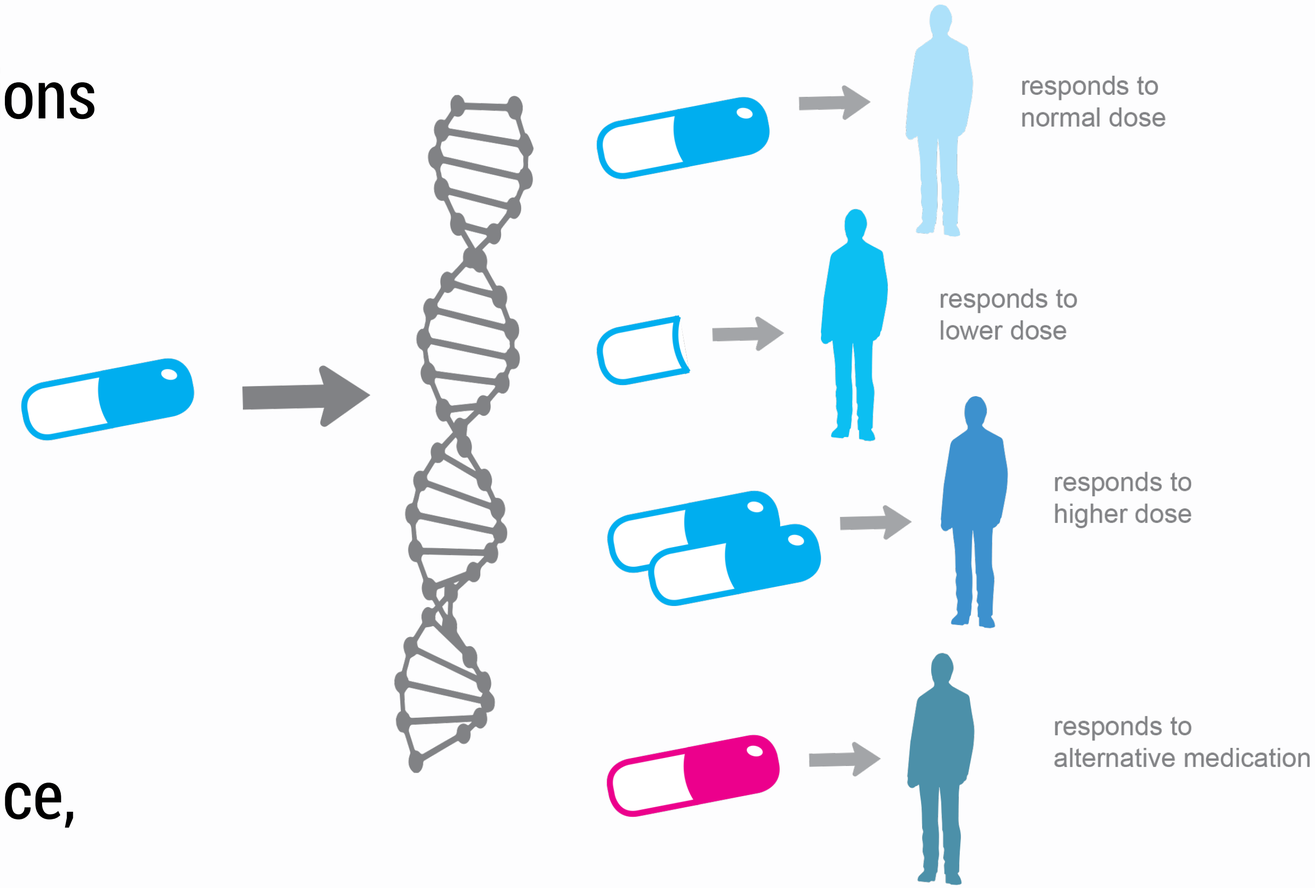
RECALL

High-dimensional data

k is large (e.g., hundreds to millions of features)

Some examples:

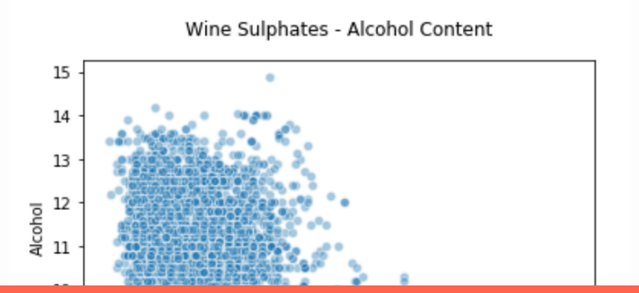
- images
- video
- audio
- text
- health data (genome sequence, medical history)



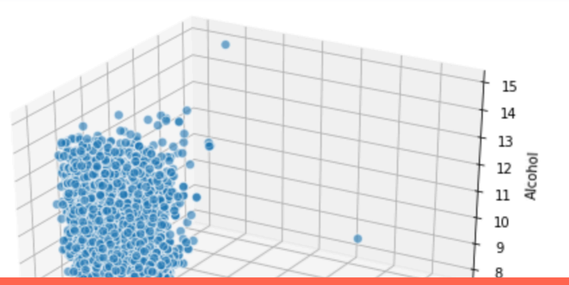
How to visualize high-dimensional data?

one heuristic: use additional cues such as hue, depth, size, shape …

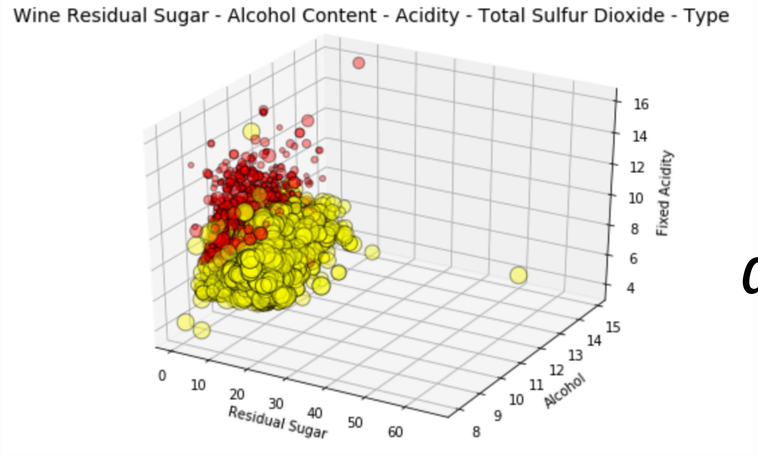
2 dimensions



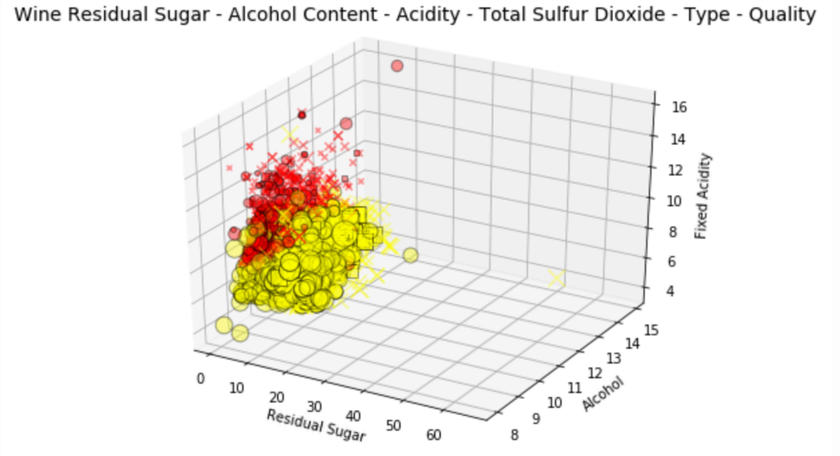
3 dimensions



issue: not scalable



5 dimensions



6 dimensions

Dimensionality Reduction

Goal: Find a more compact representation of our data

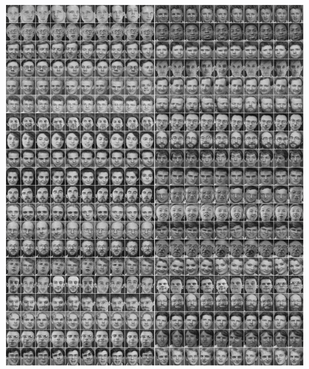
Useful for:

- Pre-processing for visualization or supervised learning
- Discovering structure or patterns in data
- Efficient use of resources (time, memory, communication)

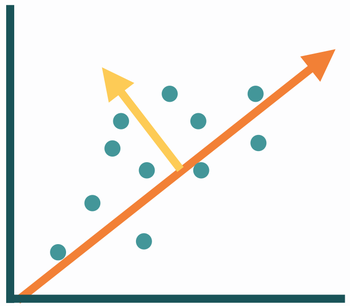
RECALL

Embedding

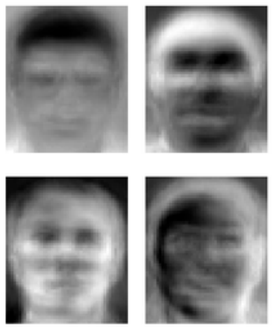
How to reduce size of dataset?



input: large dataset {x}



find: sources of variation



return: representation {z}

Outline

1. Motivation: Data visualization
2. Principal Component Analysis (PCA)
3. Johnson-Lindenstrauss

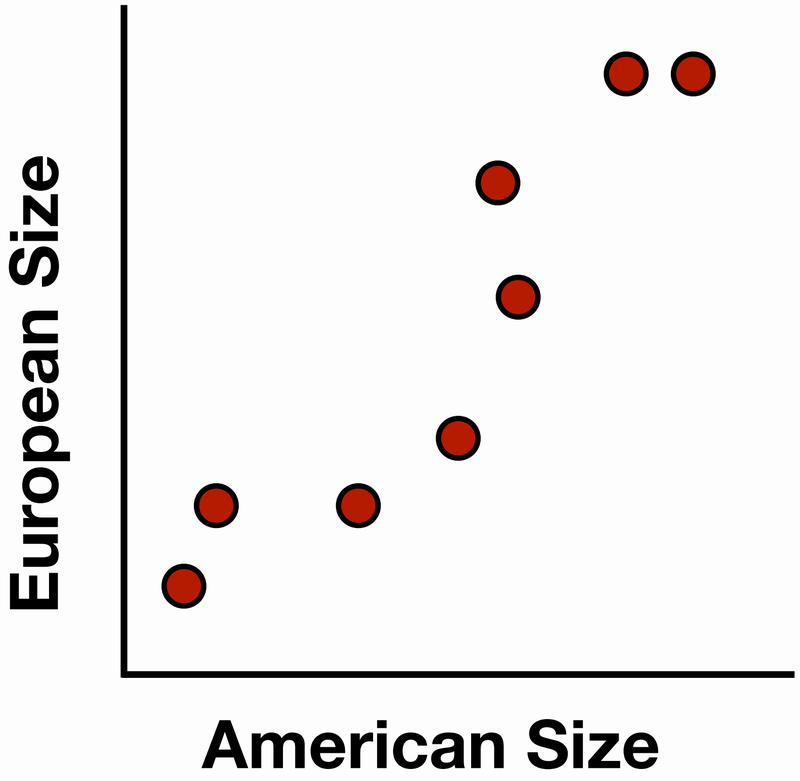
Toy example: Shoe sizes

Shoes often labeled in terms of their European and American shoe sizes

- We expect sizes to be correlated (but maybe not perfect, due to some errors/noise)

How can we find a simpler, more compact representation of this data?

Pick a direction & project into one dimension



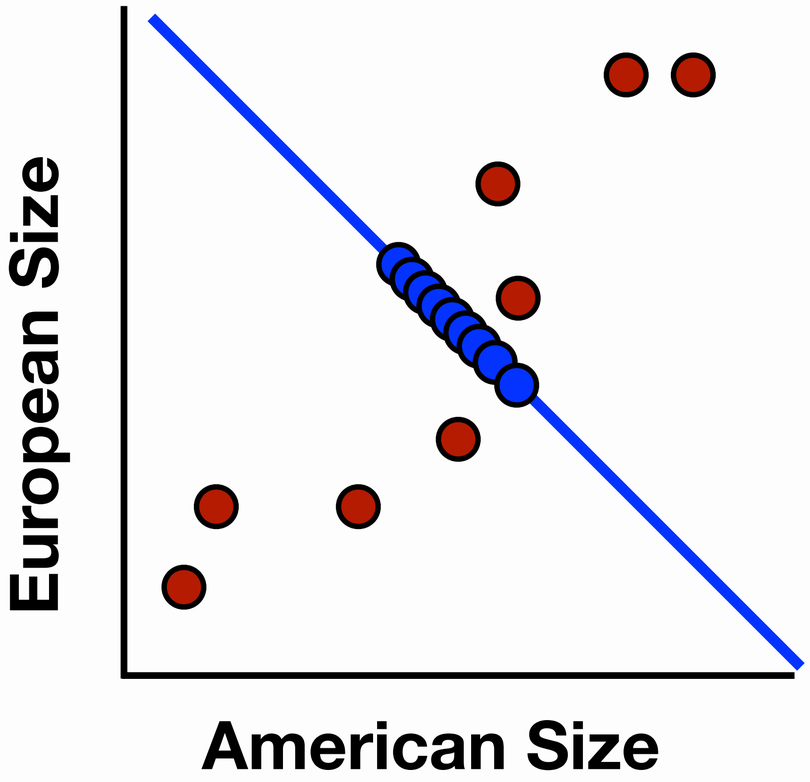
Toy example: Shoe sizes

Shoes often labeled in terms of their European and American shoe sizes

- We expect sizes to be correlated (but maybe not perfect, due to some errors/noise)

How can we find a simpler, more compact representation of this data?

Pick a direction & project into one dimension



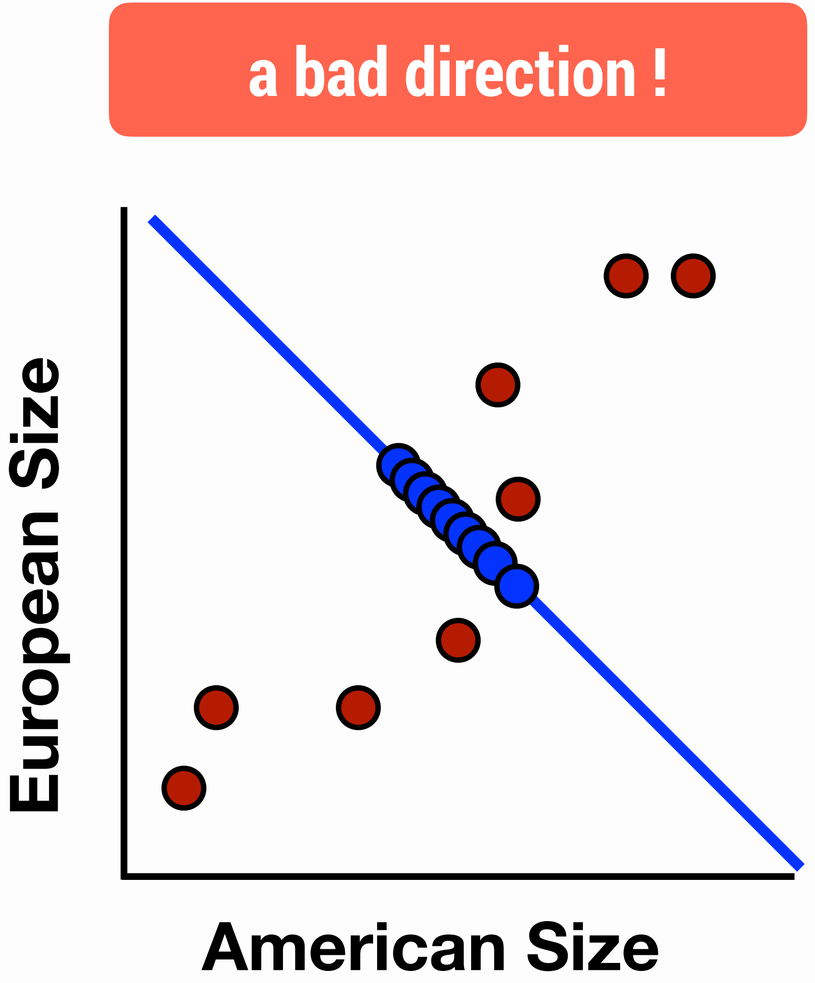
Toy example: Shoe sizes

Shoes often labeled in terms of their European and American shoe sizes

- We expect sizes to be correlated (but maybe not perfect, due to some errors/noise)

How can we find a simpler, more compact representation of this data?

Pick a direction & project into one dimension



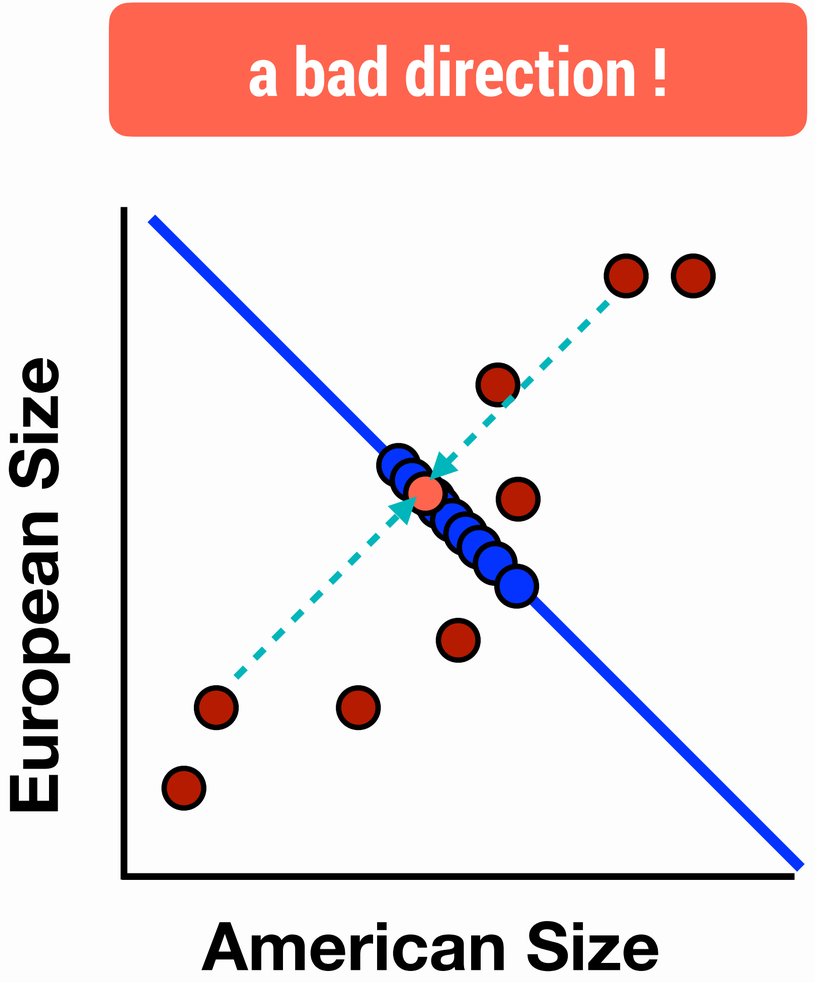
Toy example: Shoe sizes

Shoes often labeled in terms of their European and American shoe sizes

- We expect sizes to be correlated (but maybe not perfect, due to some errors/noise)

How can we find a simpler, more compact representation of this data?

Pick a direction & project into one dimension



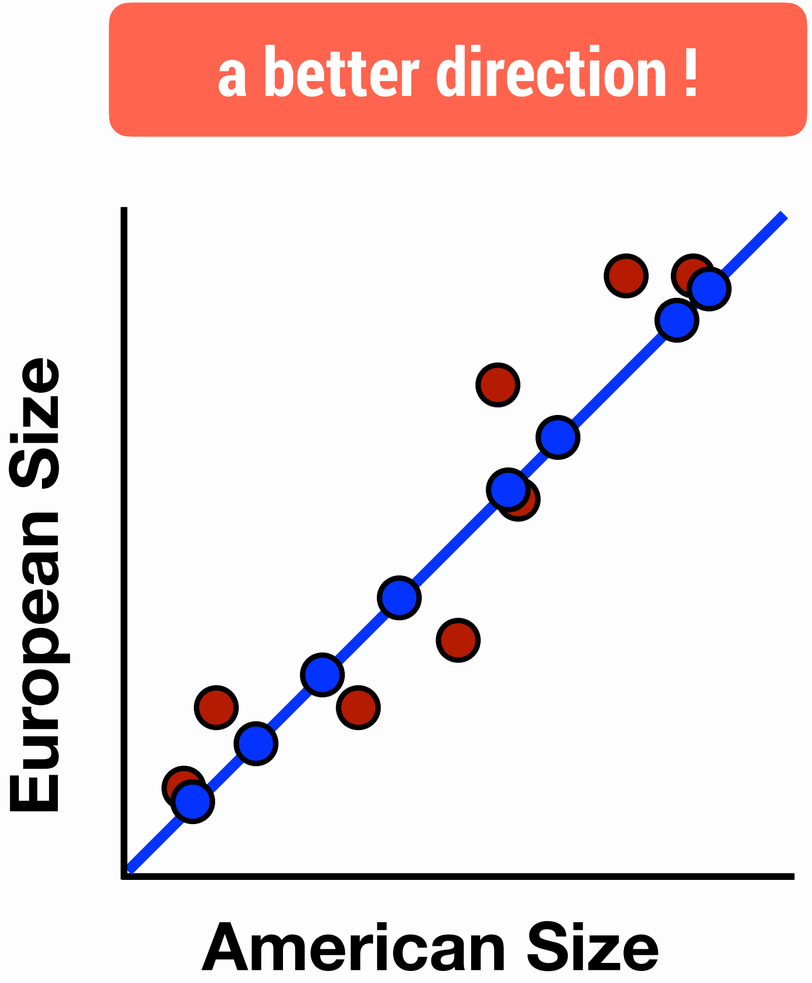
Toy example: Shoe sizes

Shoes often labeled in terms of their European and American shoe sizes

- We expect sizes to be correlated (but maybe not perfect, due to some errors/noise)

How can we find a simpler, more compact representation of this data?

Pick a direction & project into one dimension

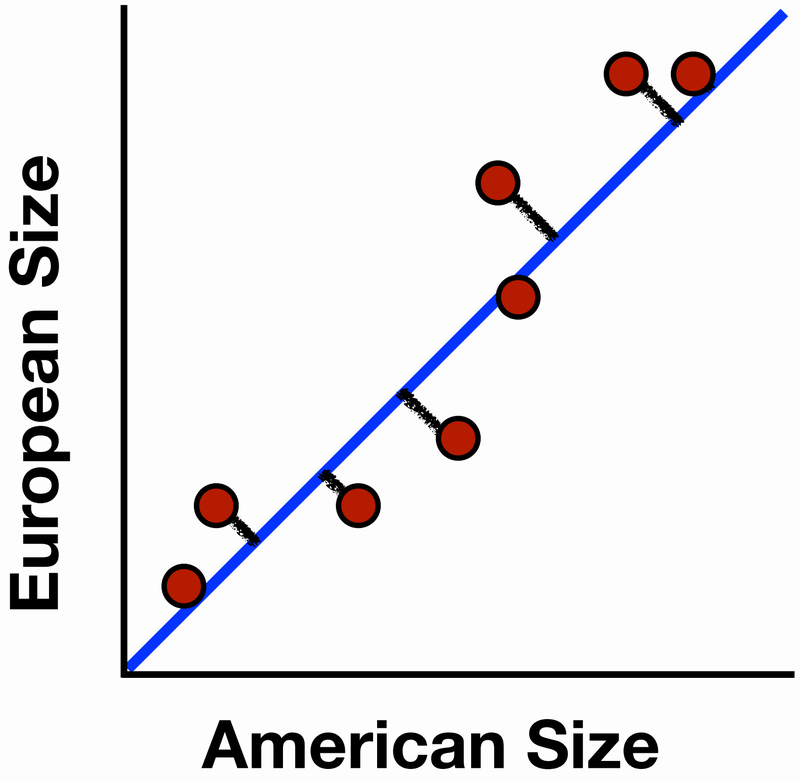


How to find the ‘best’ direction?

Goal: Minimize reconstruction error

- i.e., Find the direction that minimizes the Euclidean distance between the original points and their projections

This is the key idea behind PCA

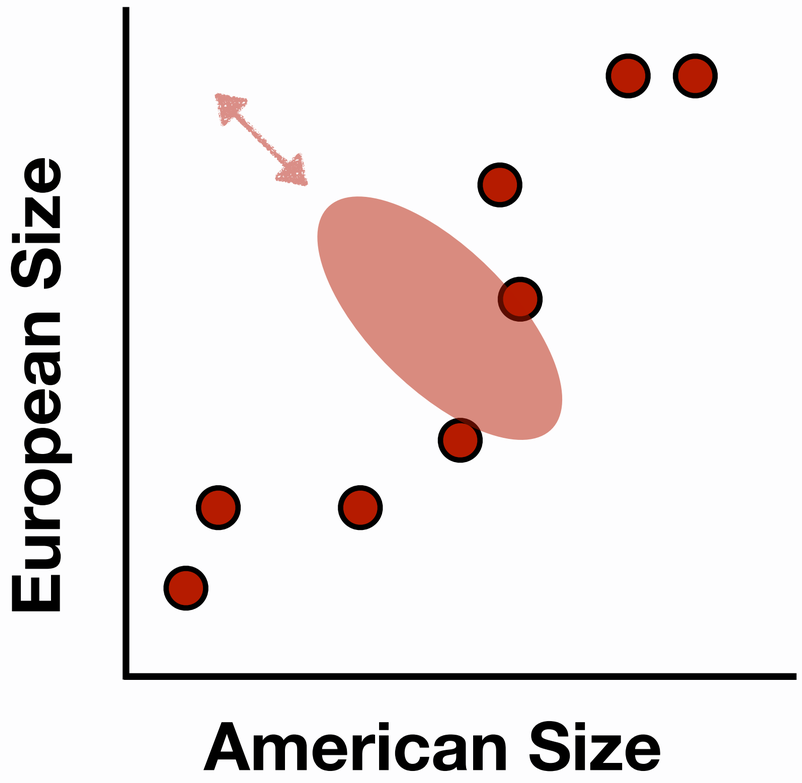


PCA

Another interpretation: Maximize variance

To identify patterns we want to study variation across observations

Can we find a compact representation that best captures this variation?

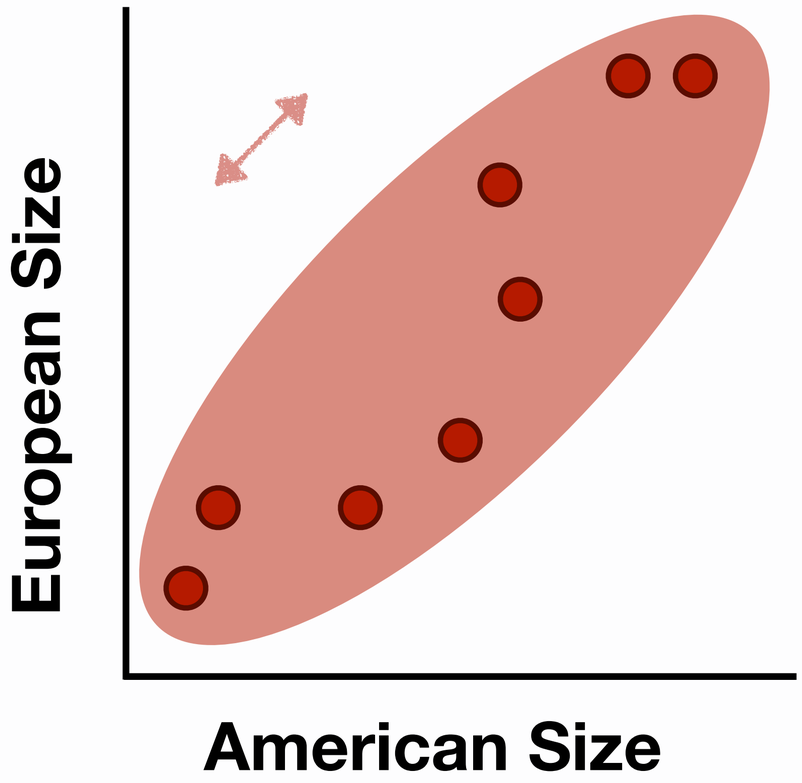


PCA

Another interpretation: Maximize variance

To identify patterns we want to study variation across observations

Can we find a compact representation that best captures this variation?



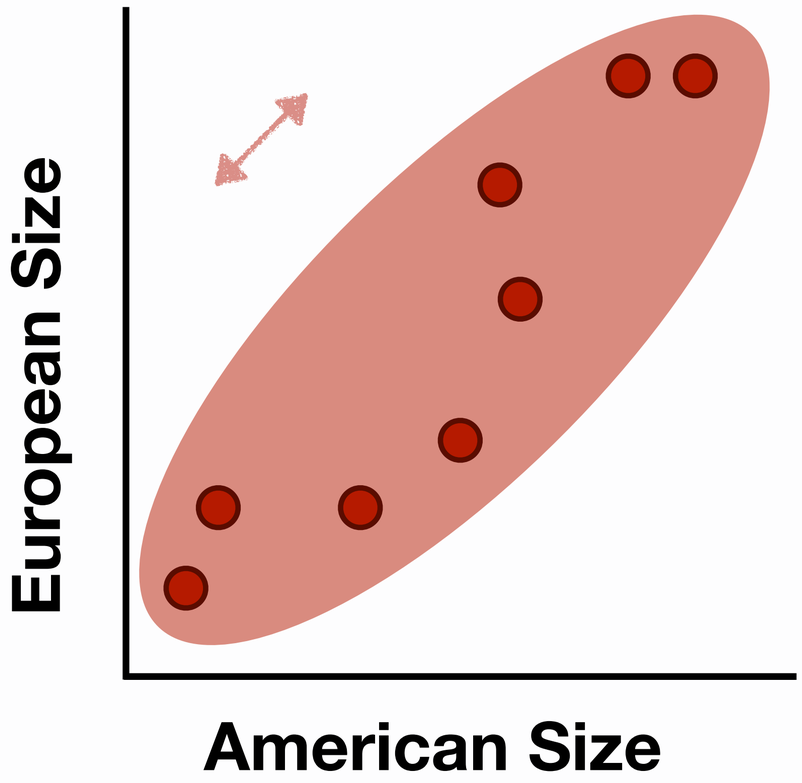
PCA

Another interpretation: Maximize variance

To identify patterns we want to study variation across observations

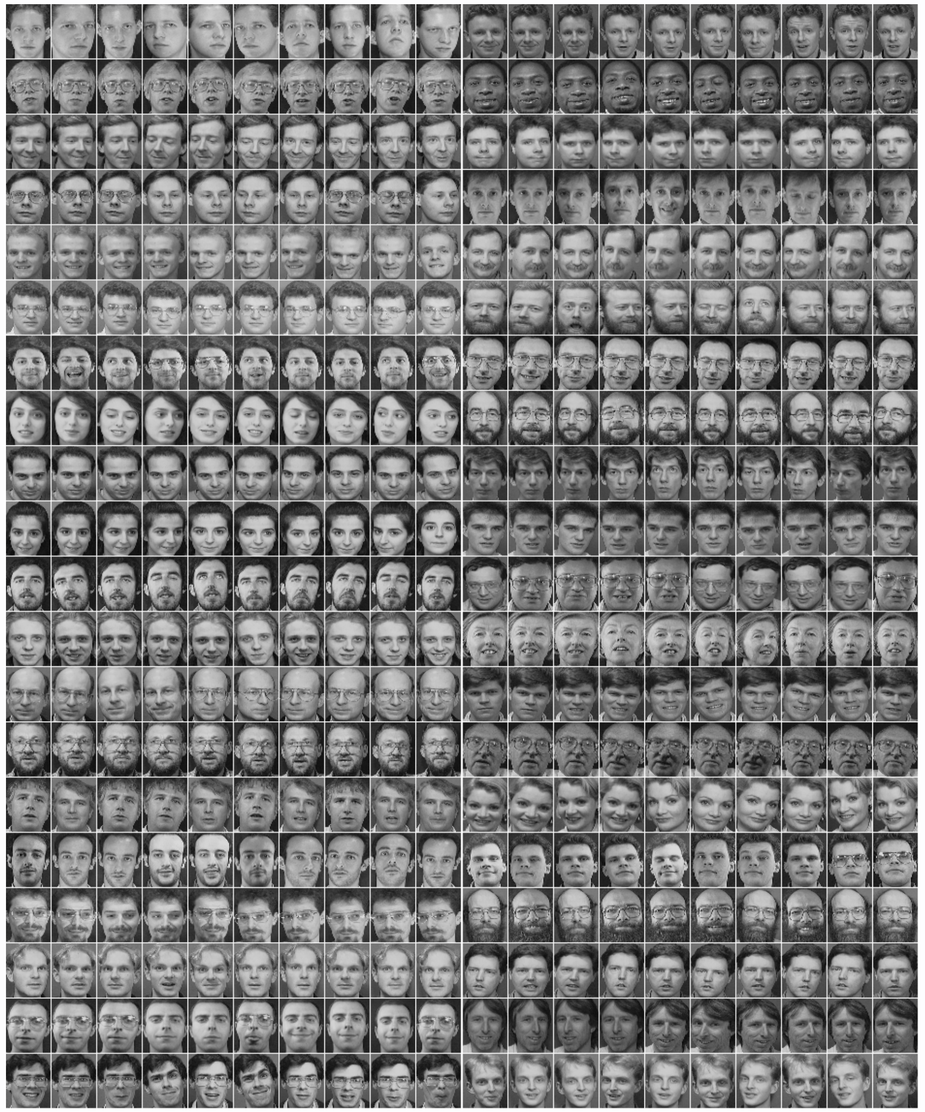
Can we find a compact representation that best captures this variation?

PCA finds the direction of *maximal variance*

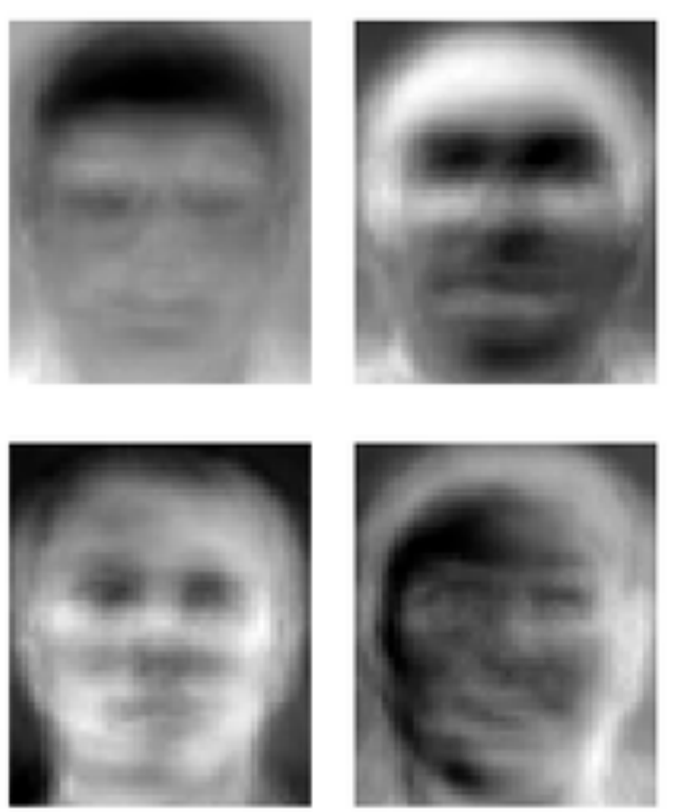


EXAMPLE

Eigenfaces



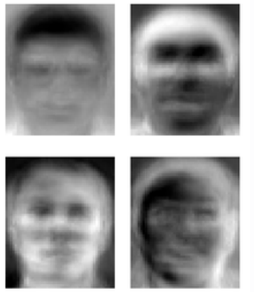
Top 4 eigenvectors (‘eigenfaces’) from PCA



The directions of greatest variability

EXAMPLE

Eigenfaces



Can be used to (approximately) reconstruct original data


≈ 0.55

+ 0.30

+ 0.10

+ 0.05



≈ 0.40

+ 0.05

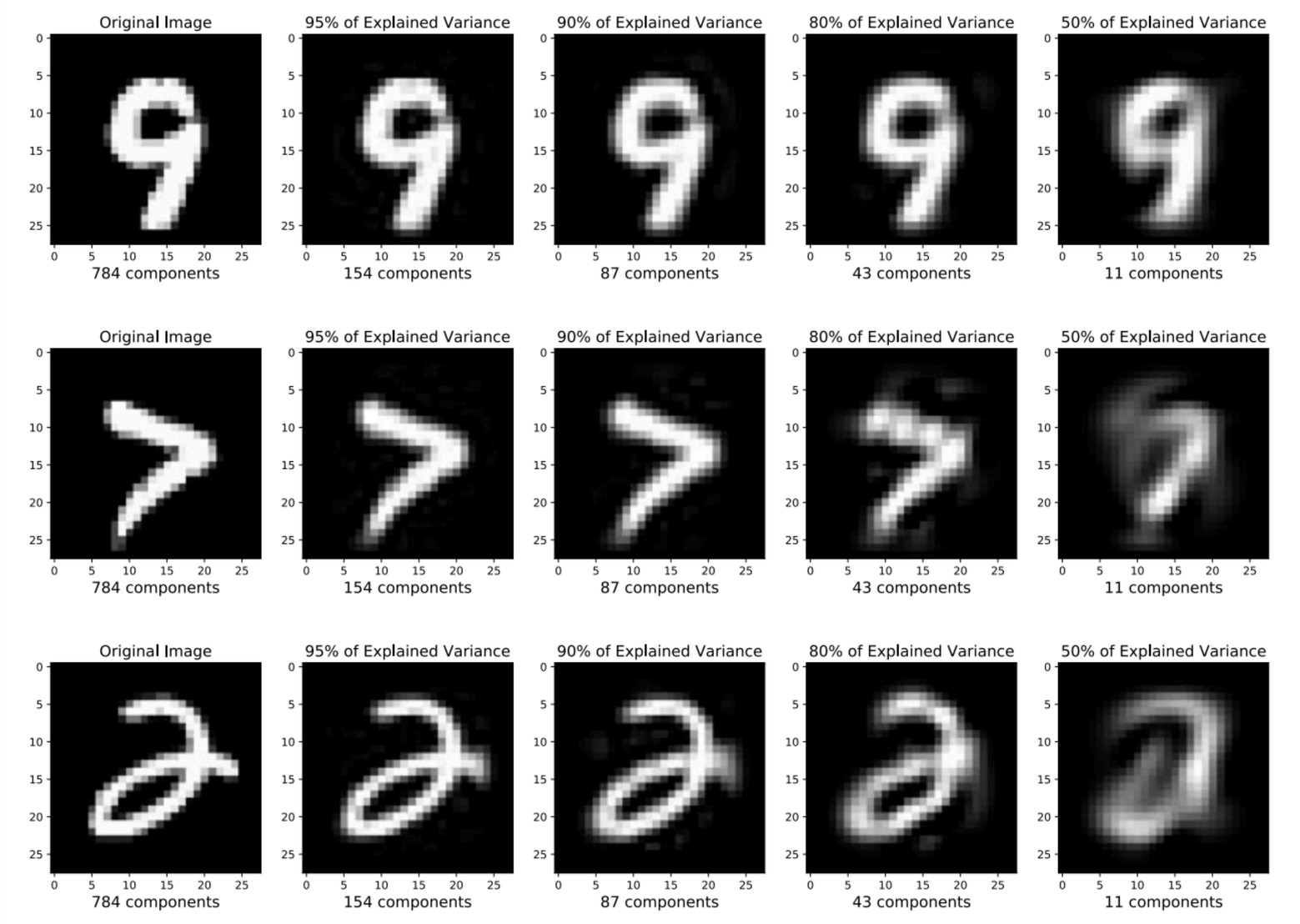
+ 0.10

+ 0.45


EXAMPLE

MNIST

- Full image: 784 pixels
- Use PCA to reduce to XX components
- See how much is lost in reconstruction

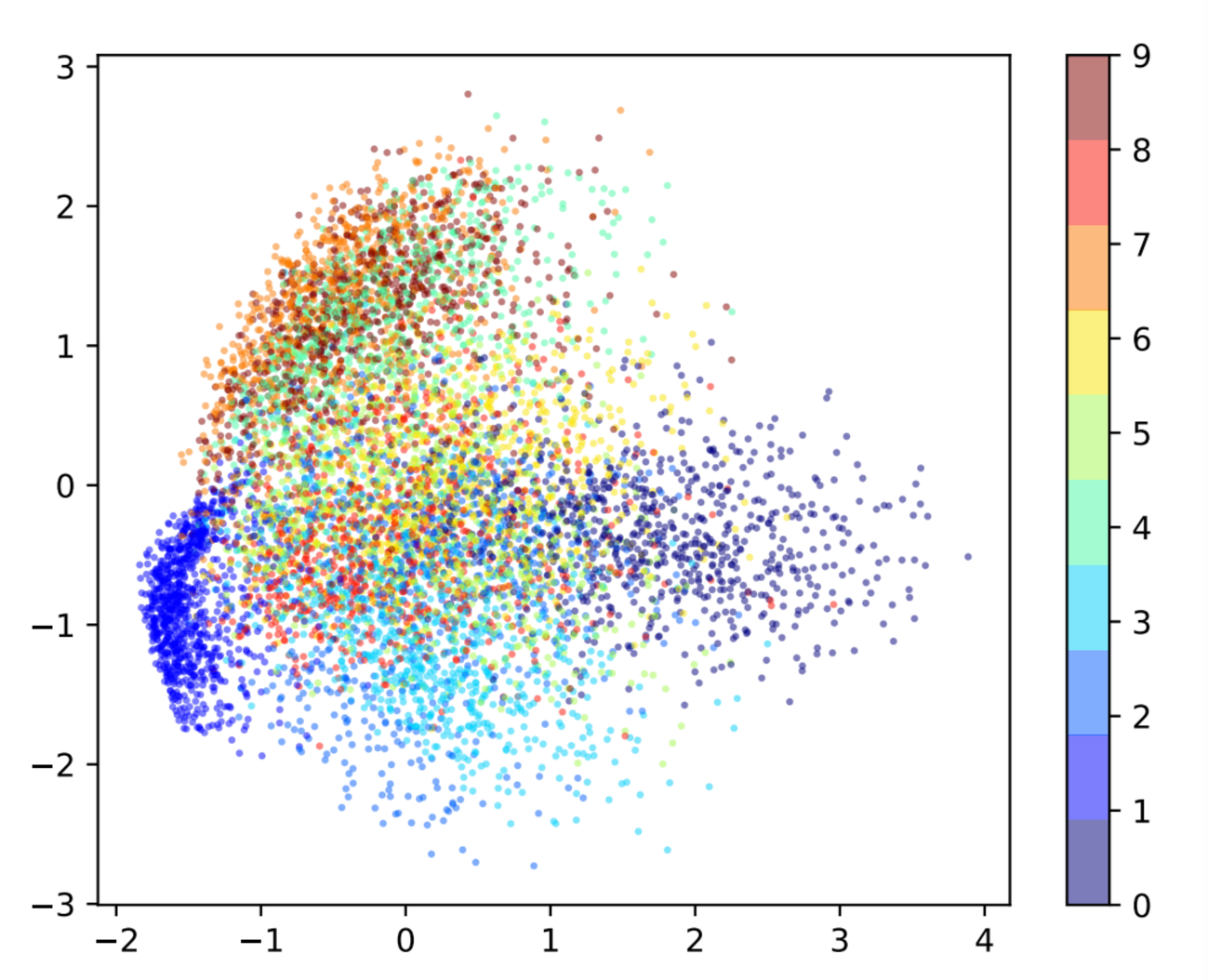


Credit: Matt Gormley

EXAMPLE

MNIST: 2 dimensions

- Use PCA to reduce to 2 dimensions
- Plot data and color code by label [0-9]

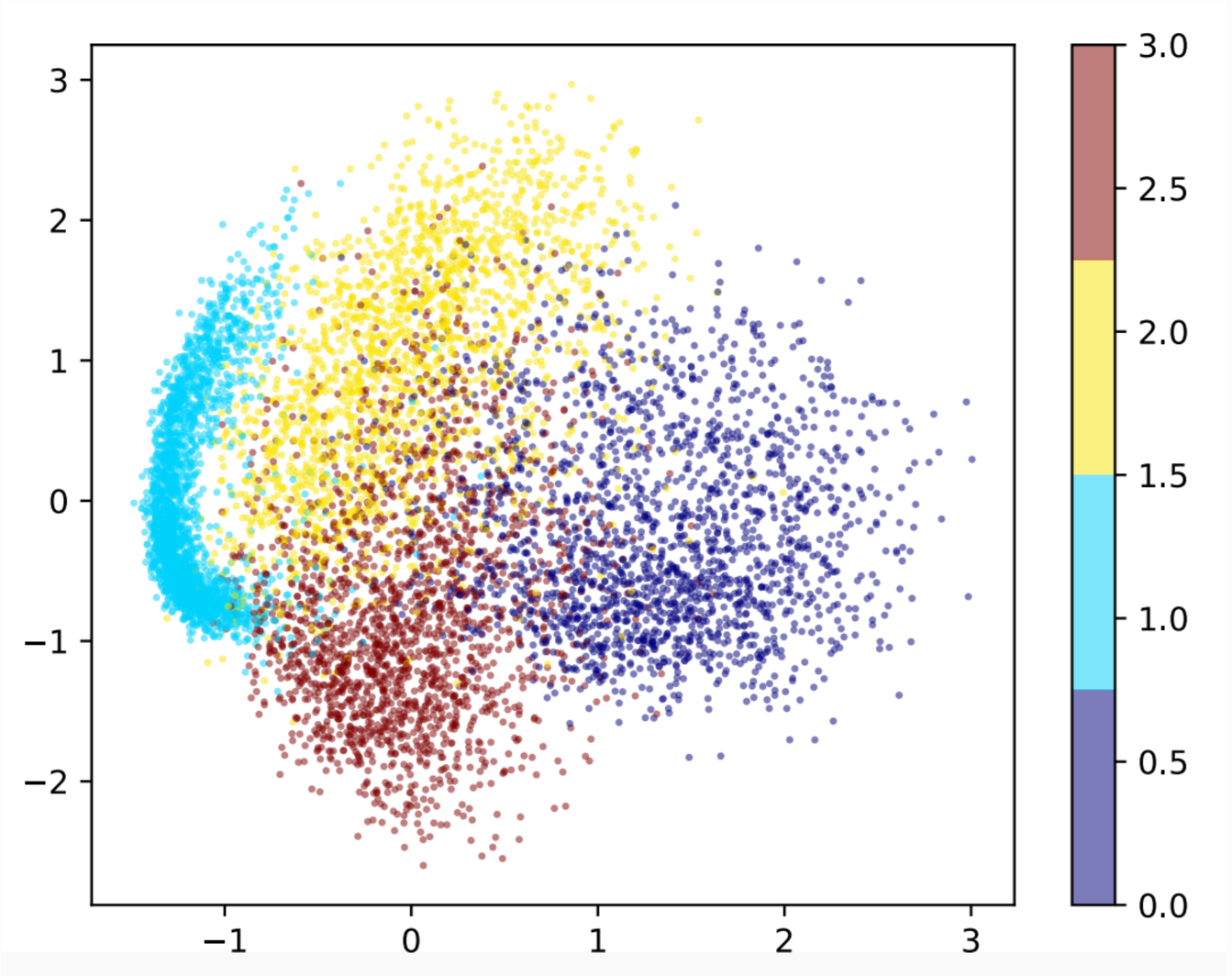


Credit: Matt Gormley

EXAMPLE

MNIST: 2 dimensions

- Use PCA to reduce to 2 dimensions
- Plot *subset* of data (0,1,2,3) and color code by label [0-3]



Credit: Matt Gormley

HOW DOES PCA WORK?

PCA Assumptions & Solution

PCA formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times k$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times r$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $k \times r$ (columns contain r principal components)
- Variance constraints

Linearity assumption ($\mathbf{Z} = \mathbf{XP}$) simplifies problem

$$\begin{bmatrix} \\ \mathbf{Z} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{X} \\ \\ \end{bmatrix} \begin{bmatrix} \\ \mathbf{P} \\ \\ \end{bmatrix}$$

PCA formulation

Given *n* training points with *k* features:

- $\mathbf{X} \in \mathbb{R}^{n \times k}$: matrix storing points
- $x_j^{(i)}$: *j*th feature for *i*th point
- μ_j : mean of *j*th feature

variance of 1st feature $\sigma_1^2 = \frac{1}{n} \sum_{i=1}^{n} \left(x_1^{(i)} - \mu_1 \right)^2$

variance of 1st feature assuming zero mean $\sigma_1^2 = \frac{1}{n} \sum_{i=1}^{n} \left(x_1^{(i)} \right)^2$

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero → uncorrelated
- Large magnitude → (anti) correlated / redundant
- $\sigma_{12} = \sigma_1^2 = \sigma_2^2$ → features are the same

Covariance of 1st and 2nd features (assuming zero mean)

$$\sigma_{12} = \frac{1}{n} \sum_{i=1}^{n} x_1^{(i)} x_2^{(i)}$$

Covariance matrix

Covariance matrix generalizes this idea for many features

$k \times k$ covariance matrix with zero mean features

$$\mathbf{C_X} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$$

- ith diagonal entry equals variance of ith feature
- ijth entry is covariance between ith and jth features
- Symmetric (makes sense given definition of covariance)

variance: $$\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_1^{(i)}\right)^2$$

covariance: $$\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}x_1^{(i)}x_2^{(i)}$$

$$\begin{bmatrix} 2 & -1 & -1 \\ 3 & 2 & -5 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ -1 & 2 \\ -1 & -5 \end{bmatrix} = \begin{bmatrix} & \\ & \end{bmatrix}$$

$\mathbf{X}^\top$ $\mathbf{X}$ $\mathbf{X}^\top\mathbf{X}$

dividing by *n* yields covariance matrix

variance: $$\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_1^{(i)}\right)^2$$

covariance: $$\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}x_1^{(i)}x_2^{(i)}$$

$$\begin{bmatrix} 2 & -1 & -1 \\ 3 & 2 & -5 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ -1 & 2 \\ -1 & -5 \end{bmatrix} = \begin{bmatrix} 6 & 9 \\ 9 & 38 \end{bmatrix}$$

$\mathbf{X}^\top$ $\mathbf{X}$ $\mathbf{X}^\top\mathbf{X}$

dividing by *n* yields covariance matrix

PCA formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times k$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times r$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $k \times r$ (columns contain r principal components)
- variance / covariance constraints

What constraints make sense in reduced representation?

- No feature correlation, i.e., all off-diagonals in $\mathbf{C_Z}$ are zero
- Rank-ordered features by variance, i.e., sorted diagonals of $\mathbf{C_Z}$

PCA formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times k$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times r$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $k \times r$ (columns contain r principal components)
- variance / covariance constraints

$\mathbf{P}$ equals the top k eigenvectors of $\mathbf{C_X}$

$$\begin{bmatrix} \\ \mathbf{Z} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{X} \\ \\ \end{bmatrix} \begin{bmatrix} \\ \mathbf{P} \\ \\ \end{bmatrix}$$

PCA solution

All covariance matrices have an eigendecomposition

- $\mathbf{C_X} = \mathbf{U}\Lambda\mathbf{U}^\top$ (eigendecomposition for real symmetric matrix)
- $\mathbf{U}$ is $k \times k$ (column are eigenvectors, sorted by their eigenvalues)
- Λ is $k \times k$ (diagonals are eigenvalues, off-diagonals are zero)

The k eigenvectors are orthonormal directions of max variance

- Associated eigenvalues equal variance in these directions
- 1st eigenvector is direction of max variance (variance is λ_1)

in the HW, we'll use the `eigh` function from `numpy.linalg`

Orthogonal and orthonormal vectors

Orthogonal vectors are **perpendicular** to each other

- Equivalently, their dot product equals zero
- $\mathbf{a}^\top \mathbf{b} = 0$ and $\mathbf{d}^\top \mathbf{b} = 0$, but **c** isn't orthogonal to others

$$\mathbf{a} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top \qquad \mathbf{b} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top \qquad \mathbf{c} = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top \qquad \mathbf{d} = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$$

Orthonormal vectors are orthogonal and have unit norm

- **a** are **b** are orthonormal, but **b** are **d** are not orthonormal

PUTTING IT ALL TOGETHER …

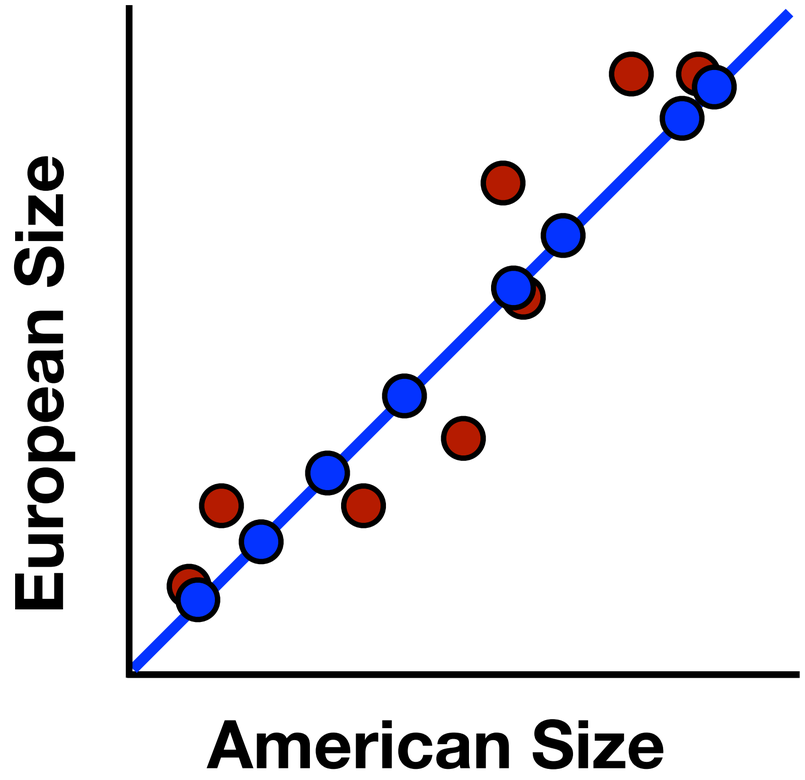
PCA iterative algorithm

r = 1: Find direction of max variance, project onto this direction

- Locations along this direction are the new 1D representation

More generally, for *i* in {1, …, *r*}:

- Find direction of max variance that is *orthonormal* to previously selected directions, project onto this direction
- Locations along this direction are the *i*th feature in new representation



Choosing *r*

How should we pick the dimension of the new representation?

Visualization: Pick top 2 or 3 dimensions for plotting purposes

Other analyses: Capture 'most' of the variance in the data

- Recall that eigenvalues are variances in the directions specified by eigenvectors, and that eigenvalues are sorted
- Fraction of retained variance: $\frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{k} \lambda_i}$

Can choose *r* such that we retain some fraction of the variance, e.g., 95%

Other practical tips

PCA assumptions (linearity, orthogonality) not always appropriate

- Extensions to PCA with different underlying assumptions, e.g., Kernel PCA, ICA
- Other nonlinear dimensionality reduction techniques (LLE, Isomap, t-SNE, …)

Centering is crucial, i.e., we must preprocess data so that all features have zero mean before applying PCA

PCA results dependent on scaling of data

- Data is sometimes rescaled in practice before applying PCA

[ADDITIONAL DETAILS]

PCA Derivation

Eigendecomposition

All covariance matrices have an eigendecomposition

- $\mathbf{C_X} = \mathbf{U}\Lambda\mathbf{U}^\top$ (eigendecomposition for real symmetric matrix)
- $\mathbf{U}$ is *k* × *k* (column are eigenvectors, sorted by their eigenvalues)
- Λ is *k* × *k* (diagonals are eigenvalues, off-diagonals are zero)

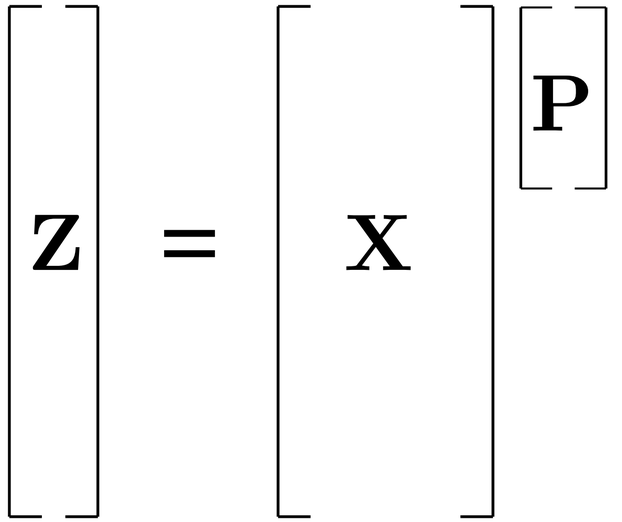
Eigenvector / Eigenvalue equation: $\mathbf{C_x u} = \lambda \mathbf{u}$

- By definition $\mathbf{u}^\top \mathbf{u} = 1$ (unit norm)
- Example: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \implies$ eigenvector: $\mathbf{u} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$, eigenvalue: $\lambda = 1$

PCA formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times k$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times r$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $k \times r$ (columns contain r principal components)
- variance / covariance constraints



PCA formulation, *r* = 1

PCA: find one-dimensional representation of raw data

- $\mathbf{X}$ is $n \times k$ (raw data)
- $\mathbf{z} = \mathbf{X}\mathbf{p}$ is $n \times 1$ (reduced representation, PCA 'scores')
- $\mathbf{p}$ is $k \times 1$ (columns are r principal components)
- variance constraint

$$\sigma_{\mathbf{z}}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(z^{(i)}\right)^2 = \frac{1}{n}||\mathbf{z}||_2^2$$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

$$\sigma_{\mathbf{z}}^2 = \frac{1}{n}||\mathbf{z}||_2^2$$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C}_{\mathbf{x}} \mathbf{p}$ where $||\mathbf{p}||_2 = 1$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

Relationship between Euclidean distance and dot product

Definition: $\mathbf{z} = \mathbf{X}\mathbf{p}$

Transpose property: $(\mathbf{X}\mathbf{p})^\top = \mathbf{p}^\top \mathbf{X}^\top$; associativity of multiply

Definition: $\mathbf{C}_\mathbf{X} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$

$$\begin{aligned}
\sigma_{\mathbf{z}}^2 &= \frac{1}{n}||\mathbf{z}||_2^2 \\
&= \frac{1}{n}\mathbf{z}^\top \mathbf{z} \\
&= \frac{1}{n}(\mathbf{X}\mathbf{p})^\top (\mathbf{X}\mathbf{p}) \\
&= \frac{1}{n}\mathbf{p}^\top \mathbf{X}^\top \mathbf{X}\mathbf{p} \\
&= \mathbf{p}^\top \mathbf{C}_\mathbf{X}\mathbf{p}
\end{aligned}$$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C}_\mathbf{x}\mathbf{p}$ where $||\mathbf{p}||_2 = 1$

Connection to eigenvectors

Recall eigenvector / eigenvalue equation: $\mathbf{C_x u} = \lambda \mathbf{u}$

- By definition $\mathbf{u}^\top \mathbf{u} = 1$, and thus $\mathbf{u}^\top \mathbf{C_x u} = \lambda$
- But this is the expression we're optimizing, and thus maximal variance achieved when $\mathbf{p}$ is top eigenvector of $\mathbf{C_X}$

Similar arguments can be used for *r* > 1

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C_x p}$ where $||\mathbf{p}||_2 = 1$

HW1 PREVIEW

Sparse PCA

$$\max_{\mathbf{p}} \quad \mathbf{p}^\top C_X \mathbf{p}$$

$$\textit{subject to} \quad \|\mathbf{p}\|_2 = 1, \ \|\mathbf{p}\|_0 \leq s$$

- If *s*=*k*, this reduces to normal PCA
- Additional constraint limits the number of non-zero elements in principal components
- Why is this useful?
  - interpretability (which input features are most informative?)
  - pre-processing (can shift focus to specific features)

PCA Recap

Will spend more time on PCA later in the course

- Implement PCA in HW2
- & discuss distributed PCA next week

Next lecture: one potential issue with PCA: linearity assumption

$$[x_1, x_2, \ldots, x_k] \rightarrow [f_1(x_1, x_2, \ldots, x_k), \ldots, f_m(x_1, x_2, \ldots, x_k)]$$

PCA assumes the functions f() are *linear*

Outline

1. Motivation: Data visualization
2. Principal Component Analysis (PCA)
3. Johnson-Lindenstrauss

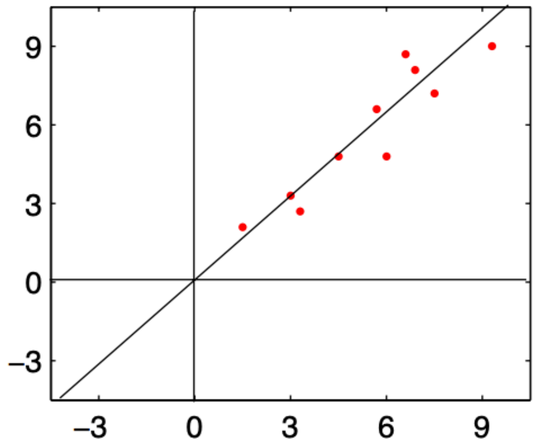
A CHEAPER APPROACH

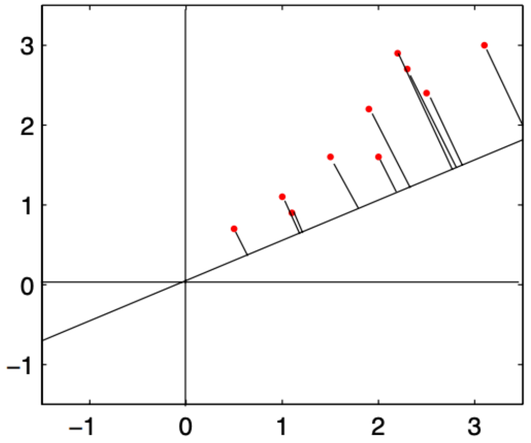
Random projections

Don't bother calculating directions as in PCA … just choose them randomly!

$$Z_{n\times r} = \frac{1}{\sqrt{r}} X_{n\times k} P_{k\times r} \quad with \quad P_{i,j} \sim N(0,1)$$

PCA: directions of projection are data-dependent →





← random projections: directions of projection are *independent* of data

Why random projections?

- Data is so high dimensional that it's too expensive to compute principal components
- You don't have access to the entire dataset at once, e.g., streaming data

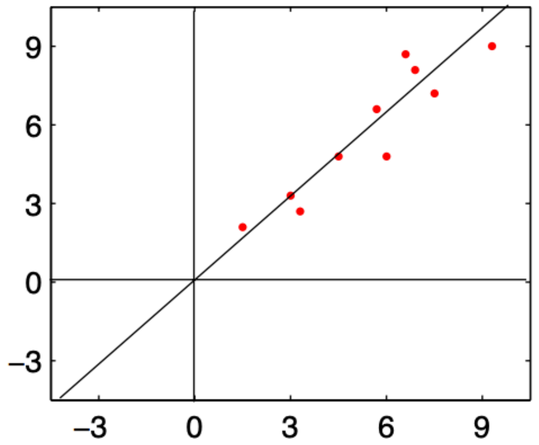
A CHEAPER APPROACH

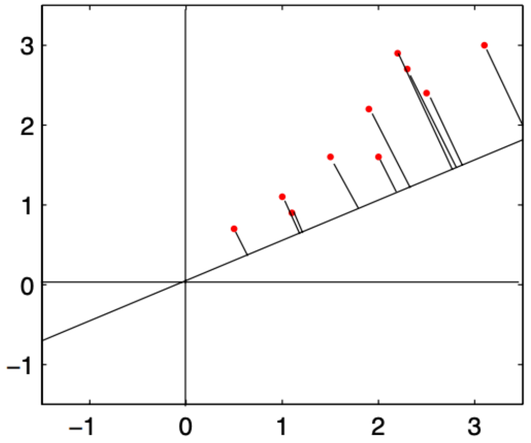
Random projections

Don't bother calculating directions as in PCA … just choose them randomly!

$$Z_{n\times r} = \frac{1}{\sqrt{r}} X_{n\times k} P_{k\times r} \quad with \quad P_{i,j} \sim N(0,1)$$

PCA: directions of projection are data-dependent →



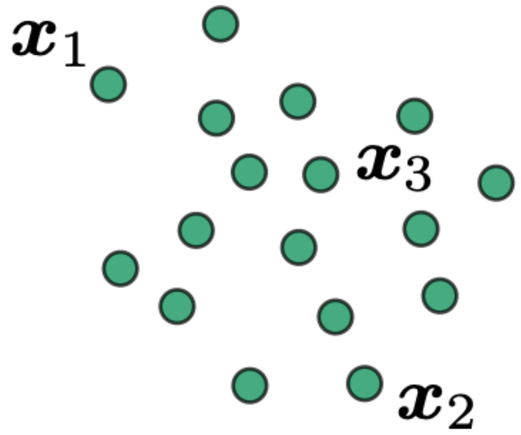


← random projections: directions of projection are *independent* of data

question: will this work? if so, how well will it work?

[IN PICTURES]

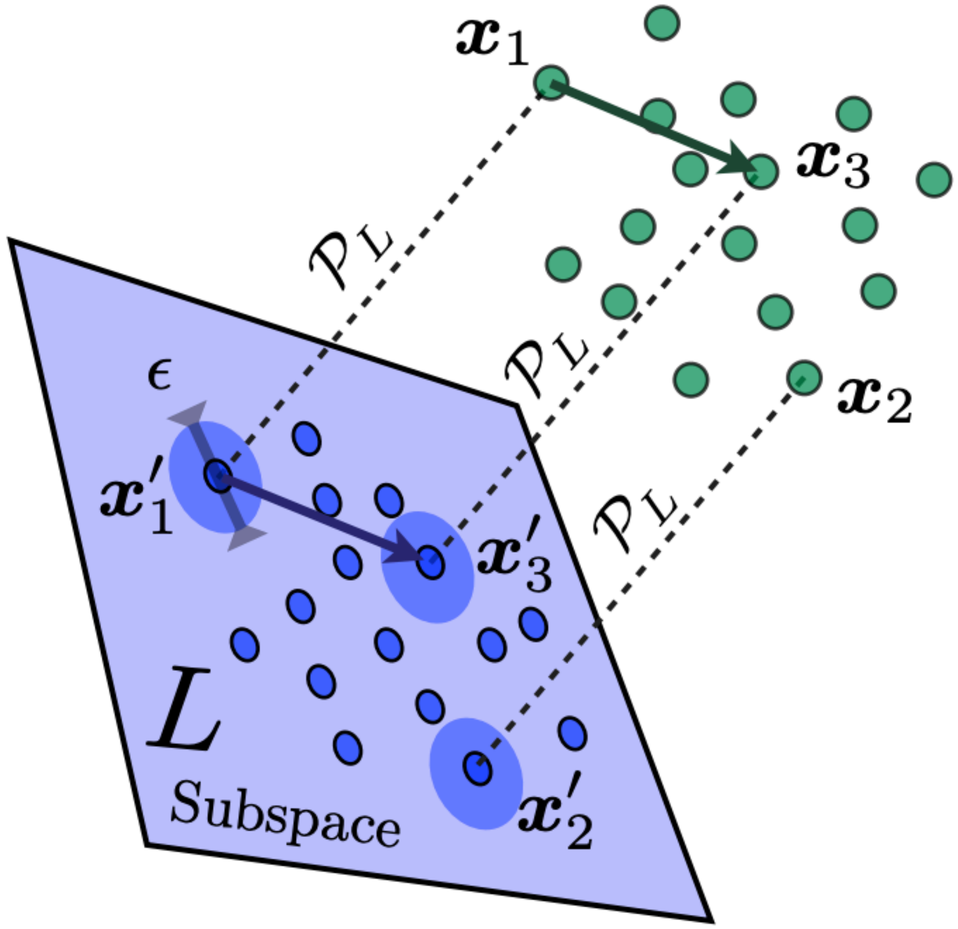
Johnson-Lindenstrauss Lemma



suppose we have *n* points,
each of dimension *k*

[IN PICTURES]

Johnson-Lindenstrauss Lemma

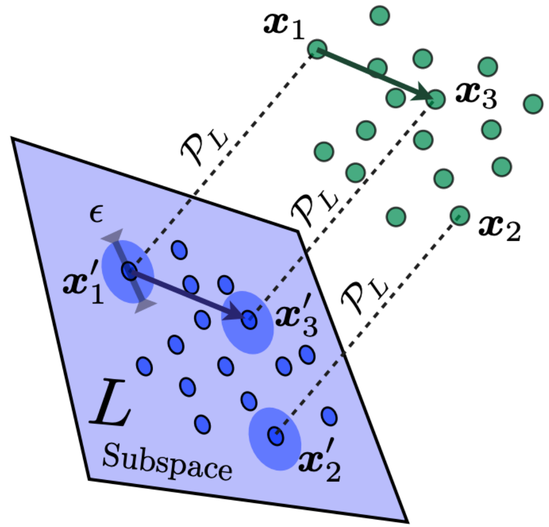


suppose we have *n* points, each of dimension *k*

if we project these points onto a subspace, L, of dimension *r*, will the distances between each point be (approximately) preserved? i.e.,

$$\|x_i - x_j\| \simeq (1+\epsilon)\|x'_i - x'_j\|$$

Johnson-Lindenstrauss Lemma



Given $0 < \varepsilon < 1$ and $x_1, \ldots x_n \in \mathbb{R}^k$, if r is a natural number such that:

$$r \geq 4\ln(n)(\epsilon^2/2 - \epsilon^3/3)^{-1} \ ,$$

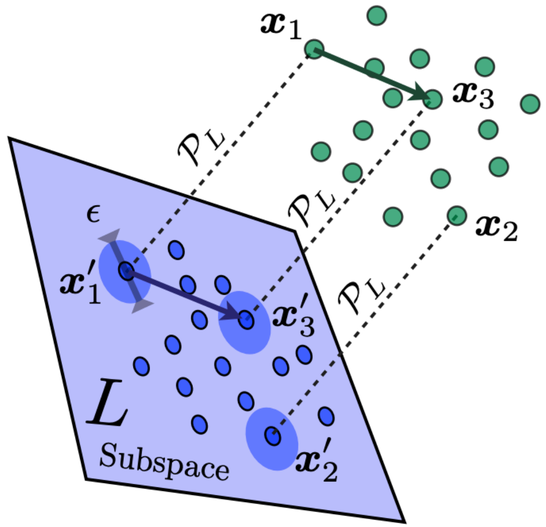
then there exists a *linear map*, $f: \mathbb{R}^k \rightarrow \mathbb{R}^r$ such that:

$$(1-\epsilon)\|x_i - x_j\|^2 \leq \|f(x_i) - f(x_j)\|^2 \leq (1+\epsilon)\|x_i - x_j\|^2 \quad \forall i,j \in [n]$$

Key ideas:

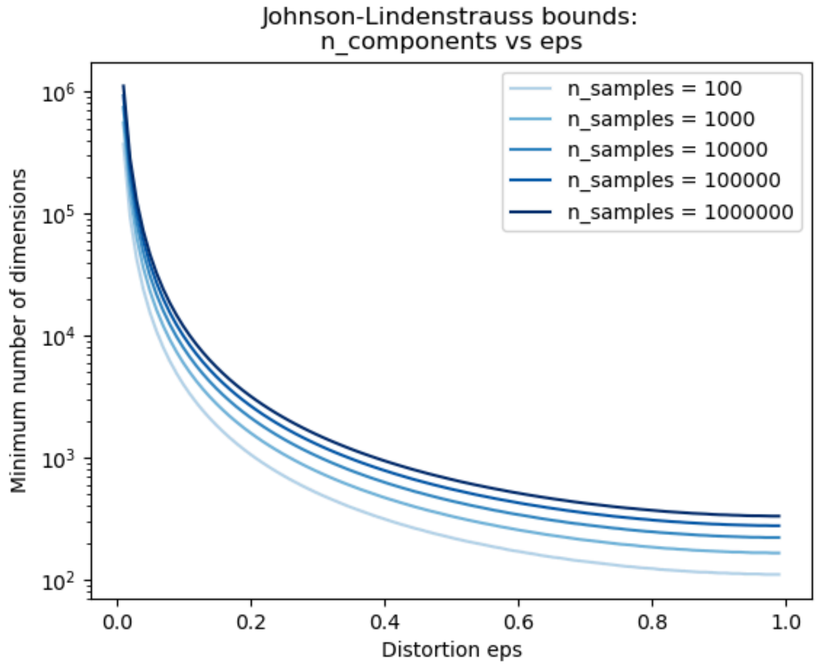
- construct f using random projections (as in slide 65)
- show that mean value of distance will be equal in lower dim, and use concentration of measure (i.e., random quantity will be close to mean with high probability)
- note: does not depend on k !

Johnson-Lindenstrauss Lemma



Rough formula for *r,* size of lower dimensional subspace:

Given the size of the dataset, *n*, to provide a random project within some desired error ε, we should set the minimum dimension of the subspace to be: $r > \ln(n)/\epsilon^2$



scikit-learn

PCA vs. Random Projections

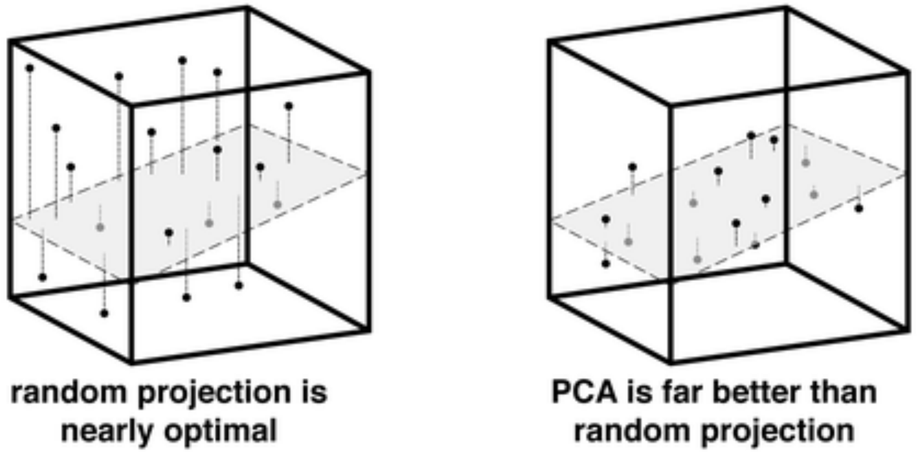


fig. 1 : randomly projecting from $\mathbb{R}^3$ to $\mathbb{R}^2$

PCA can provide better projections

- data-dependent, mutually orthogonal

However, PCA is more expensive

- PCA: $O(nk^2 + k^3 + nkr)$ vs. RP: $O(nkr)$
- May be infeasible for problems with very high dimension (k), or in online settings

Johnson-Lindenstrauss

- Helps to characterize performance of random projections
- Provides data-independent guarantees about dimensionality reduction

Additional Resources

- NeurIPS tutorial on visualization, Fernanda Viégas & Martin Wattenberg (2018): https://www.youtube.com/watch?v=ze08gwVPaXk
- A Tutorial on Principal Component Analysis, Jonathon Shlens (2014): https://arxiv.org/pdf/1404.1100.pdf
- An Elementary Proof of a Theorem of Johnson and Lindenstrauss (2002) https://cseweb.ucsd.edu/~dasgupta/papers/jl.pdf