Mid term 1

- **Exam:**
  - 75-min exam on Oct 16 (lecture time and location)
  - In-person closed-book
  - Can bring a cheatsheet: 1 handwritten piece of paper (letter size, two sides)
  - No need for calculator
- **Practice exam:**
  - Exam and solutions posted on course website (under the files tab)
  - Will go over during the review for mid term 1 (Oct 14)
- **Mid term 1** covers:
  - All the modules we have learned so far including K-Means and PCA (this week)

Lecture 12: Unsupervised Learning (Part 2)

CIS 4190/5190

Fall 2024

Principal Component Analysis

Dimensionality Reduction

- **Goal:** Learn a mapping from $x \in \mathbb{R}^d$ to $x \in \mathbb{R}^{d'}$, with $d' \ll d$

- We may want to reduce the number of features for several reasons:
  - Reduce the complexity of our learning problem
  - Remove colinear/correlated features
  - Visualize the features

Learning Good Features

	LotFrontage	LotArea	Street	LotShape	Utilities	LandSlope	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	ExterQual	ExterCond	BsmtQual	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	...	SaleCondition_Abnorml
0	65.0	8450	2	4	4	3	7	5	2003	2003	196.0	4	3	4	0	6	706	1	...	0
1	80.0	9600	2	4	4	3	6	8	1976	1976	0.0	3	3	4	3	5	978	1	...	0
2	68.0	11250	2	3	4	3	7	5	2001	2002	162.0	4	3	4	1	6	486	1	...	0
3	60.0	9550	2	3	4	3	7	5	1915	1970	0.0	3	3	3	0	5	216	1	...	1
4	84.0	14260	2	3	4	3	8	5	2000	2000	350.0	4	3	4	2	6	655	1	...	0
5	85.0	14115	2	3	4	3	5	5	1993	1995	0.0	3	3	4	0	6	732	1	...	0
6	75.0	10084	2	4	4	3	8	5	2004	2005	186.0	4	3	5	2	6	1369	1	...	0
7	0.0	10382	2	3	4	3	7	6	1973	1973	240.0	3	3	4	1	5	859	4	...	0
8	51.0	6120	2	4	4	3	7	5	1931	1950	0.0	3	3	3	0	1	0	1	...	1
9	50.0	7420	2	4	4	3	5	6	1939	1950	0.0	3	3	3	0	6	851	1	...	0
10	70.0	11200	2	4	4	3	5	5	1965	1965	0.0	3	3	3	0	3	906	1	...	0
11	85.0	11924	2	3	4	3	9	5	2005	2006	286.0	5	3	5	0	6	998	1	...	0
12	0.0	12968	2	2	4	3	5	6	1962	1962	0.0	3	3	3	0	5	737	1	...	0
13	91.0	10652	2	3	4	3	7	5	2006	2007	306.0	4	3	4	2	1	0	1	...	0
14	0.0	10920	2	3	4	3	6	5	1960	1960	212.0	3	3	3	0	4	733	1	...	0
15	51.0	6120	2	4	4	3	7	8	1929	2001	0.0	3	3	3	0	1	0	1	...	0
16	0.0	11241	2	3	4	3	6	7	1970	1970	180.0	3	3	3	0	5	578	1	...	0
17	72.0	10791	2	4	4	3	4	5	1967	1967	0.0	3	3	0	0	0	0	0	...	0
18	66.0	13695	2	4	4	3	5	5	2004	2004	0.0	3	3	3	0	6	646	1	...	0
19	70.0	7560	2	4	4	3	5	6	1958	1965	0.0	3	3	3	0	2	504	1	...	1
20	101.0	14215	2	3	4	3	8	5	2005	2006	380.0	4	3	5	2	1	0	1	...	0
21	57.0	7449	2	4	4	3	7	7	1930	1950	0.0	3	3	3	0	1	0	1	...	0
22	75.0	9742	2	4	4	3	8	5	2002	2002	281.0	4	3	4	0	1	0	1	...	0
23	44.0	4224	2	4	4	3	5	7	1976	1976	0.0	3	3	4	0	6	840	1	...	0

227 features

Data from: De Cock. Journal of Statistics Education 19(3), 2011

Data Visualization

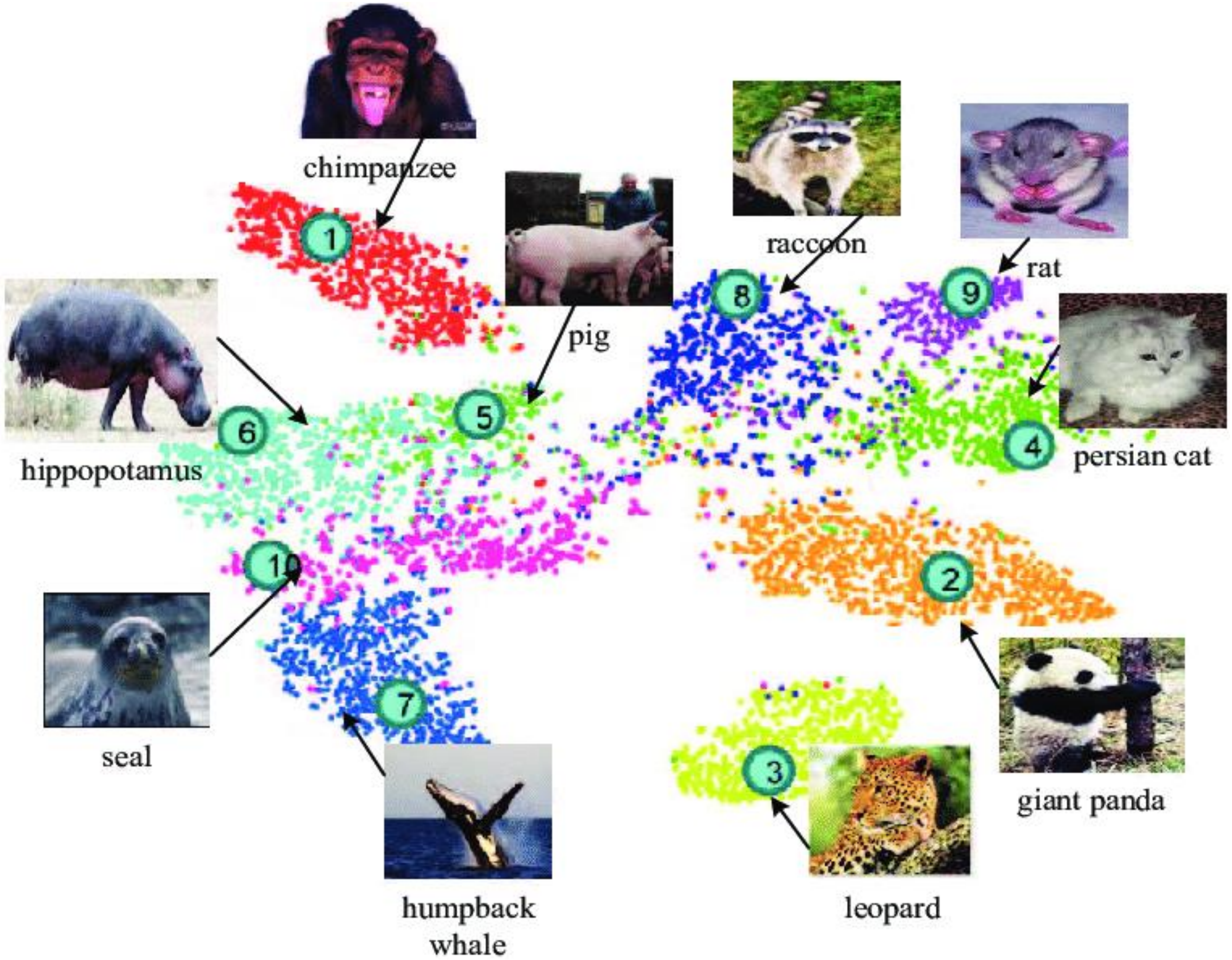


Image: https://arxiv.org/pdf/1703.08893.pdf

Dimensionality Reduction

- We can write each input x as

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + x_d \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

projections original axes

- We aim to approximate x using a new basis $\{v_i\}_i$ (of unit norm):

$$x \approx \tilde{f}(x) = f(x)_1 v_1 + f(x)_2 v_2 + \cdots + f(x)_{d'} v_{d'}$$

Representation vs. Approximation

- We **approximate** x as follows:

$$x \approx \tilde{f}(x) = f(x)_1 v_1 + f(x)_2 v_2 + \cdots + f(x)_{d'} v_{d'} \in \mathbb{R}^d$$

- The corresponding **representation** is

$$f(x) = [f(x)_1 \quad f(x)_2 \quad \cdots \quad f(x)_{d'}] \in \mathbb{R}^{d'}$$

Dimensionality Reduction

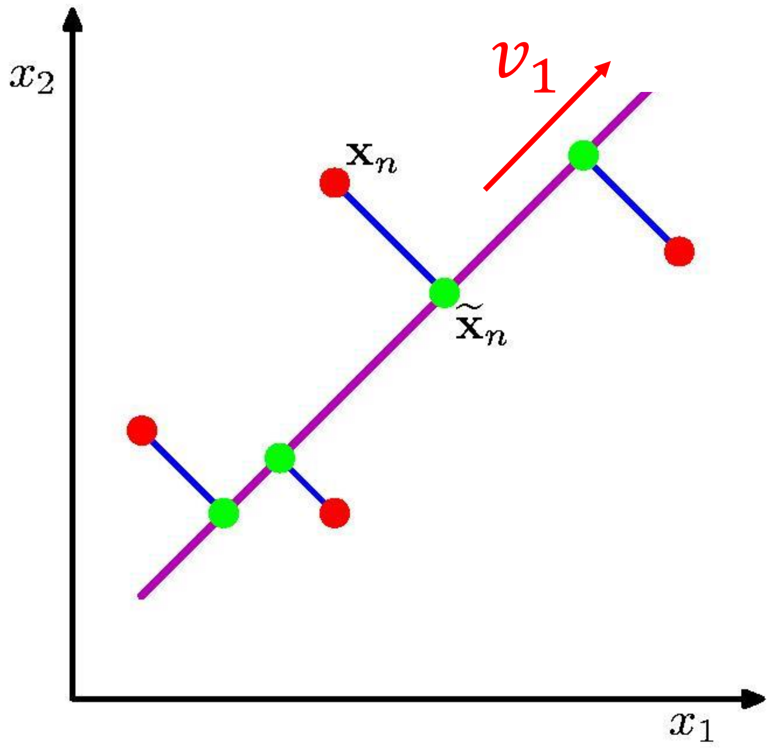
- **Loss function:** Minimize MSE of projected vectors

$$L(f; Z) = \frac{1}{n} \sum_{i=1}^{n} \left\| x_i - \tilde{f}(x_i) \right\|_2^2$$

1D Case

- **Simplest case:** If $d' = 1$, then we want $x \approx f(x)_1 v_1$

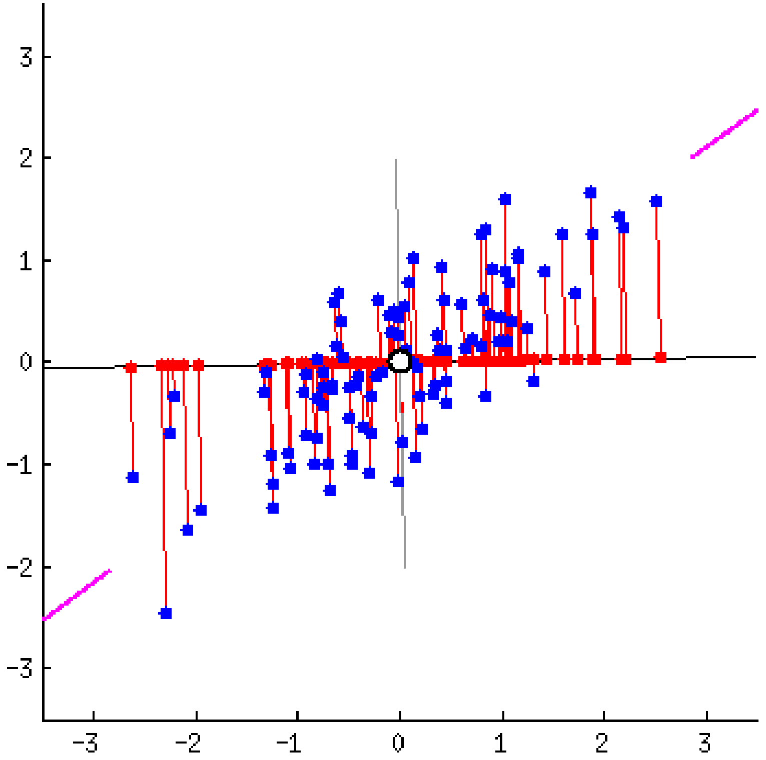
- Given v_1, we can take $f(x)_1 = x^\top v_1$
  - Minimizes $\|x - f(x)_1 v_1\|_2^2$
  - Then, we have $\tilde{f}(x) = (x^\top v_1) v_1$
  - i.e., orthogonal projection
  - Assuming $\|v_1\|_2 = 1$



1D Case

- **Simplest case:** If $d' = 1$, then we want $x \approx f(x)_1 v_1$

- Given v_1, we can take $f(x)_1 = x^\top v_1$
  - Minimizes MSE of $\|x - f(x)_1 v_1\|$
  - Then, we have $\tilde{f}(x) = (x^\top v_1) v_1$
  - i.e., orthogonal projection
  - Assuming $\|v_1\|_2 = 1$
- How do we pick v_1 ?



(fig: stats.stackexchange)

1D Case

- In this case, the loss is

$$L(v_1; Z) = \frac{1}{n}\sum_{i=1}^{n}\left\|x_i - (x_i^\top v_1)v_1\right\|_2^2$$

- Can be shown to be equivalent to maximizing variance:

$$L(v_1; Z) = -\mathrm{Var}\left(\{x_i^\top v_1\}_i\right)$$

- If variance of projection on v_1 is low, v_1 is not informative about x_i

1D Case

- Need a way to minimize $L(v_1; Z)$
- The **covariance matrix** is

$$C = \mathbb{E}[xx^\top] = \mathbb{E}\begin{bmatrix} x_1x_1 & \cdots & x_1x_d \\ \vdots & \ddots & \vdots \\ x_dx_d & \cdots & x_dx_d \end{bmatrix}$$

- Given v_1, we have $\mathrm{Var}(x^\top v_1) = v_1^\top C v_1$
- Thus, $L(v_1; Z) = -\mathrm{Var}(x^\top v_1) = -v_1^\top C v_1$

1D Case

- The principal components analysis (PCA) algorithm computes

$$v_1^* = \min_{v_1} L(v_1; Z) = \max_{v_1} v_1^\top C v_1$$

- **Theorem:** Solution is $v_1^* = \text{TopEigenvector}(C)$
  - That is, eigenvector corresponding to the largest eigenvalue
  - **Recall:** If $Cv = \lambda v$, then v is an eigenvector corresponding to eigenvalue λ

1D Case

- In practice, use **empirical covariance matrix**

$$\hat{C} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top = \frac{1}{n}\sum_{i=1}^{n} \begin{bmatrix} x_{i,1}x_{i,1} & \cdots & x_{i,1}x_{i,d} \\ \vdots & \ddots & \vdots \\ x_{i,d}x_{i,d} & \cdots & x_{i,d}x_{i,d} \end{bmatrix}$$

- **Algorithm:** Compute eigenvectors + eigenvalues of $\hat{C}$ and return the (unit) eigenvector corresponding to the largest eigenvalue
  - Sign of eigenvector doesn't matter

General Case

$\text{PCA}(Z)$:

$$Z \leftarrow \{\, x - \text{Mean}(Z) \mid x \in Z \,\}$$

$$C \leftarrow \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top$$

$\textbf{for } j \in \{1, \ldots, d'\}$:

$$v_j \leftarrow \text{Eigenvector}(C, j)$$

$\textbf{return } f : x \mapsto [x^\top v_1 \quad \cdots \quad x^\top v_{d'}]^\top$

General Case

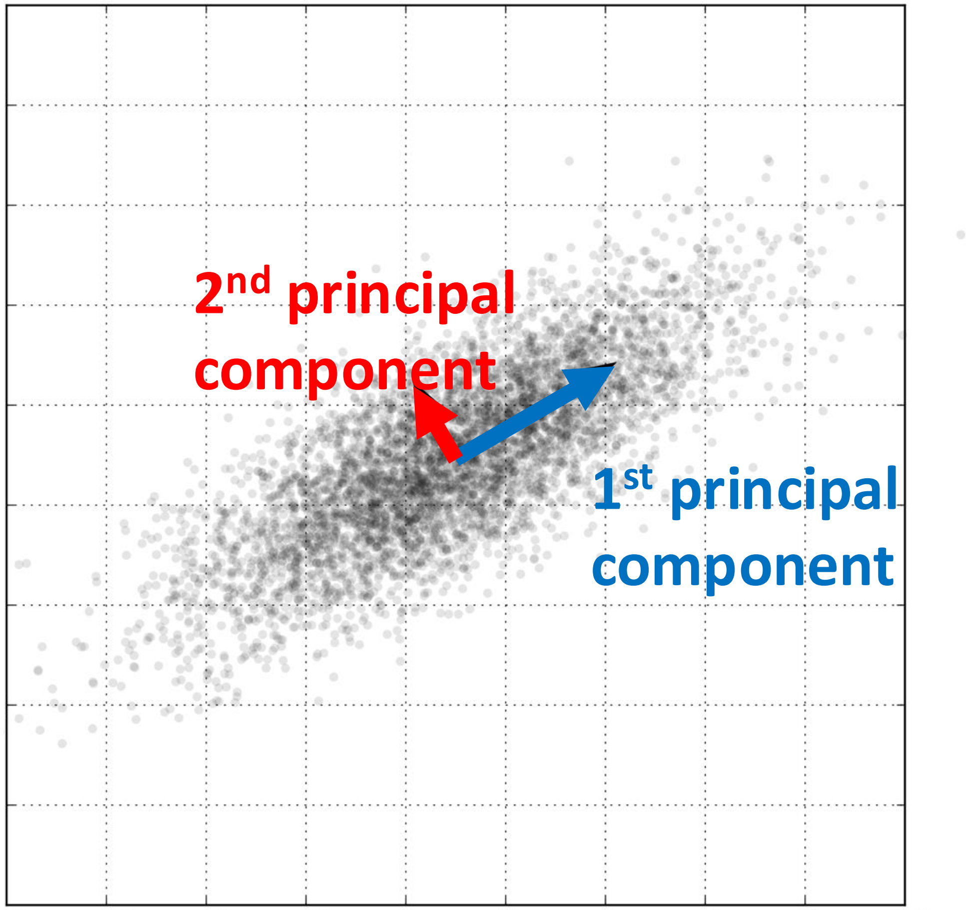
- Resulting function is

$$f(x) = \begin{bmatrix} x^\top v_1 \\ \vdots \\ x^\top v_{d'} \end{bmatrix} = \begin{bmatrix} v_1^\top \\ \vdots \\ v_{d'}^\top \end{bmatrix} x = Vx$$

PCA on a 2D Gaussian Dataset

- The vectors v_j are called **principal components**
  - Mutually orthogonal
  - Largest directions of variation

- Subtract mean to ensure vectors originate from the mean



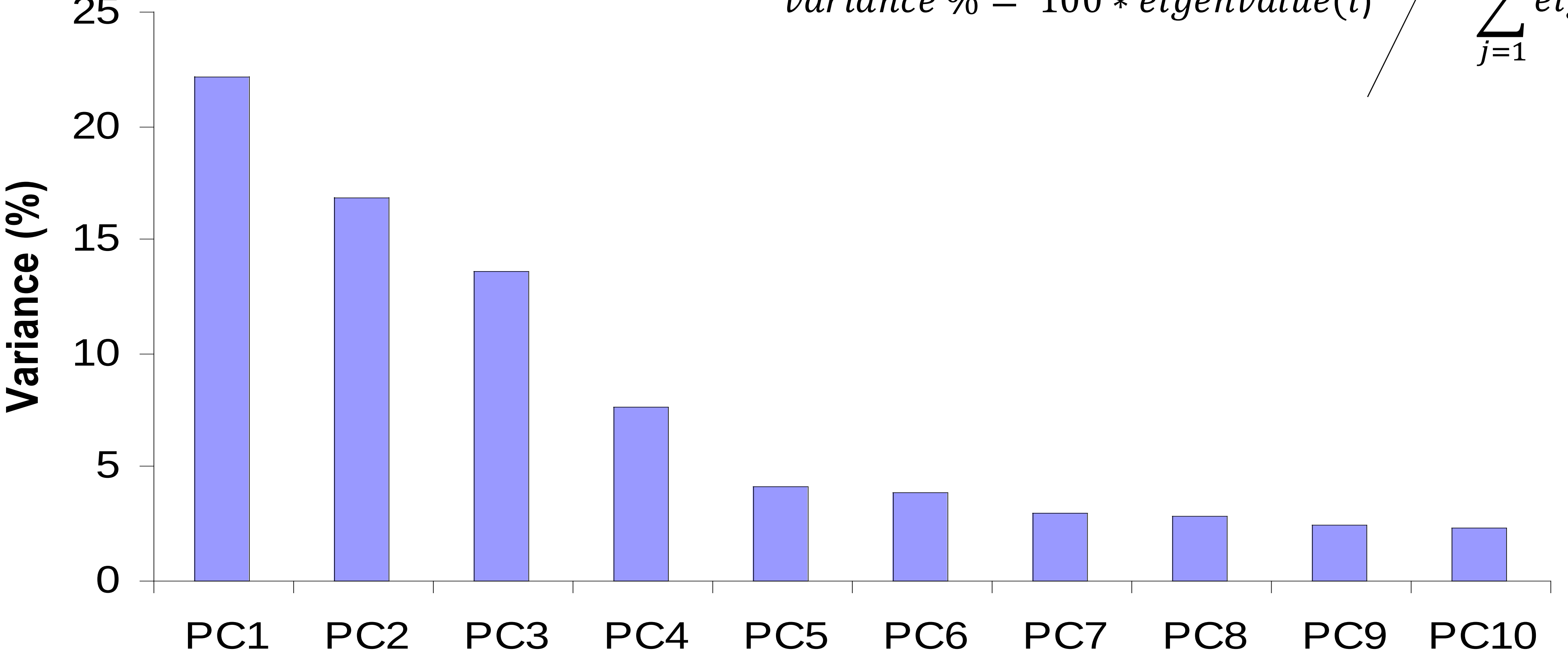
Dimensionality Reduction

- Taking $d' = d$ is just a change of basis
  - Linear regression does not change, but other algorithms may be affected

- Taking $d' \ll d$ reduce dimensionality of data while removing the smallest possible amount of information
  - In a linear sense

Dimensionality Reduction

$$variance\,\% = 100 * eigenvalue(i) \Big/ \sum_{j=1}^{d} eigenvalue(j)$$

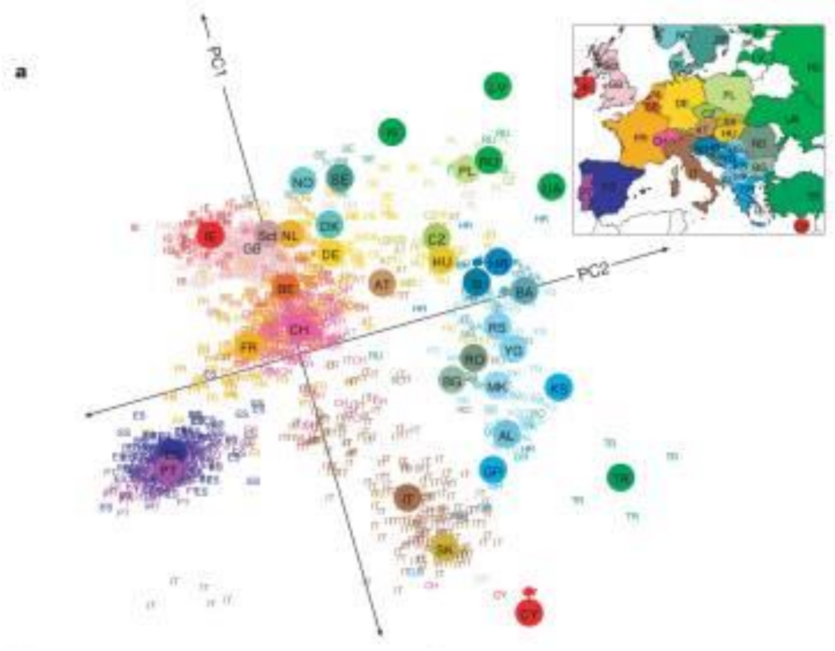


Based on slide by Barnabás Póczos, UAlberta

Applications

- Can use $f(x)$ as the feature map
  - First examples of “learned features”
  - Form of regularization
  - Forms the basis for important modern deep learning algorithms

- Can be used to visualize high-dimensional data



Novembre et al., Genes mirror geography within Europe. Nature 2009.

Eigenfaces

(1000 64×64 images)

https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

Eigenfaces

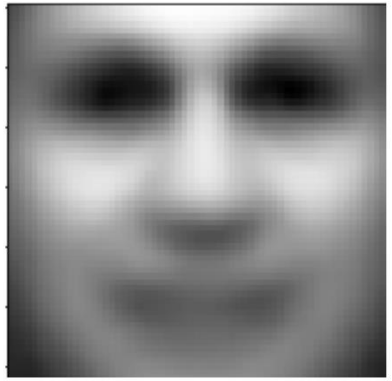


Figure #5: mean face

https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

Eigenface Projections

$d = 4096$

$d' = 1000$

Eigenface Projections

$d = 4096$

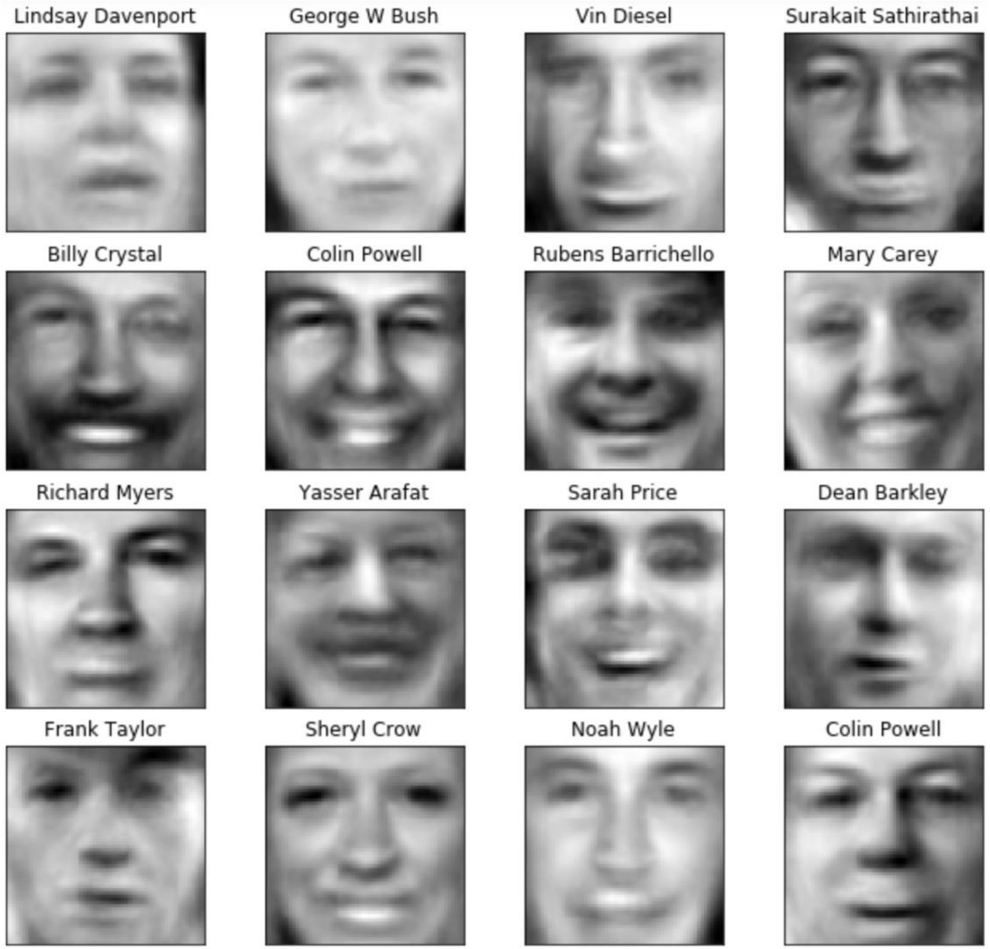
$d' = 250$

Eigenface Projections

$d = 4096$

$d' = 100$

Eigenface Projections



$$d' = 50$$

MNIST Digit Dataset

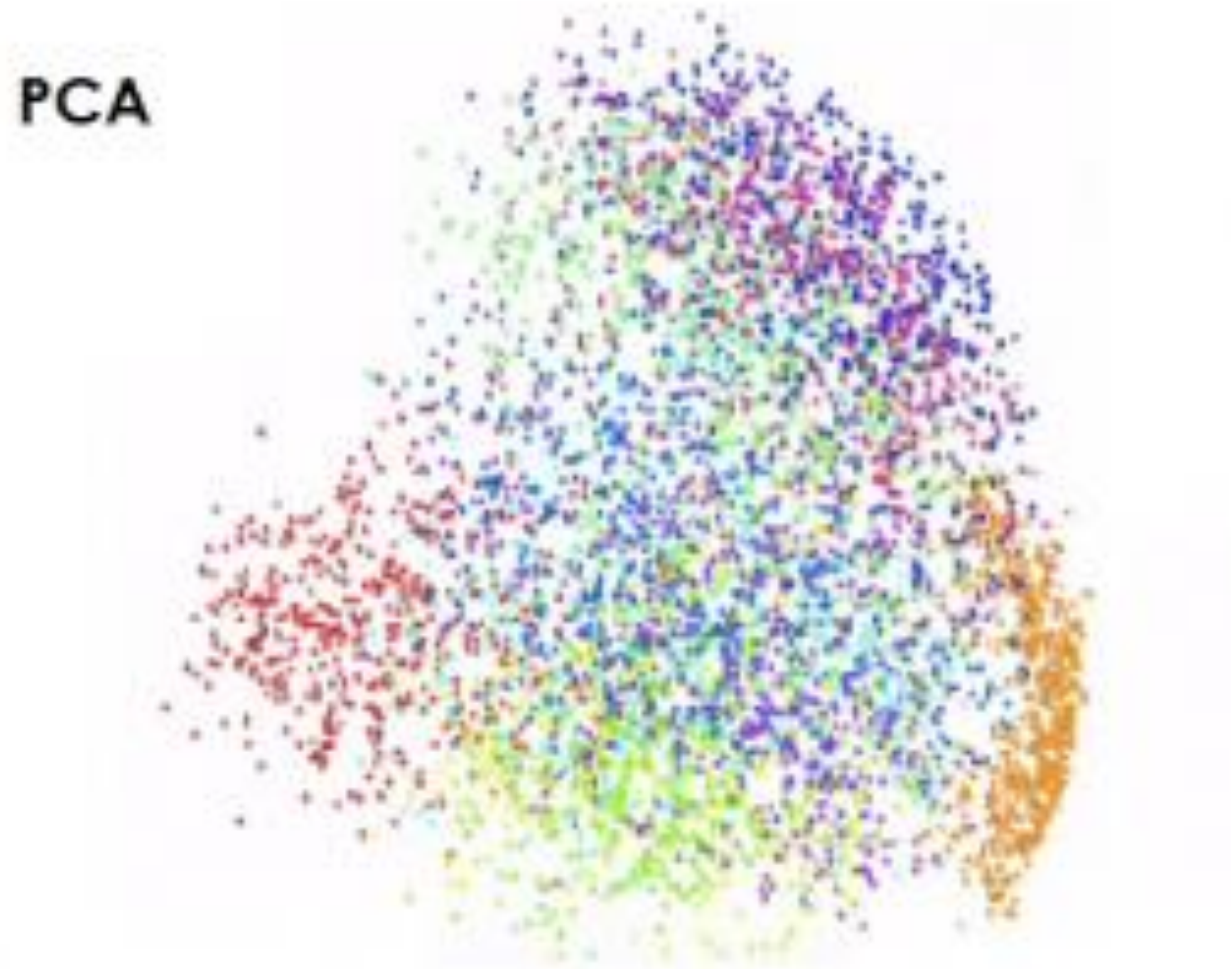


Fig: Laurens van der Maaten

Nonlinear Dimensionality Reduction

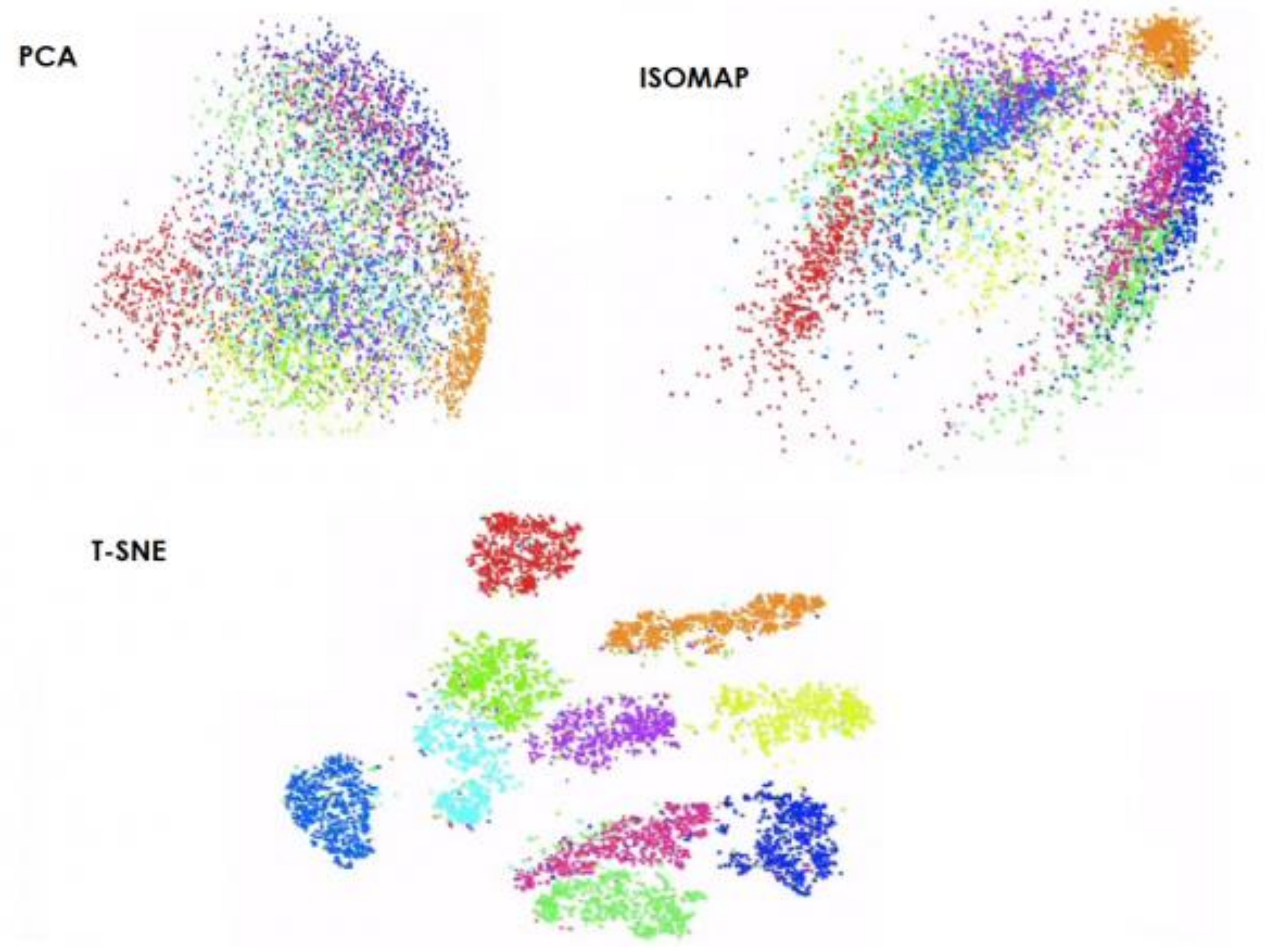


Fig: Laurens van der Maaten

Nonlinear Dimensionality Reduction

- **PCA benefits**
  - Projected representation of data can be approximate data in original space
  - Easy to optimize
  - No hyperparameters (except d')

- **Deep learning based approaches**
  - Nonlinear PCA is the basis of the autoencoder
  - Fundamental algorithm for feature learning that is still widely used