CS 540 Introduction to Artificial Intelligence
Linear Algebra & PCA

Yudong Chen
University of Wisconsin-Madison

Sep 14, 2021

Announcements

- **HW1 Due next Tuesday**
- Class roadmap:

Tuesday, Sep 14	Probability
Thursday, Sep 16	**Linear Algebra and PCA**
Tuesday, Sep 21	Statistics and Math Review
Thursday, Sep 23	Introduction to Logic
Tuesday, Sep 28	Natural Language Processing

Fundamentals

From Last Time

- Conditional Prob. & Bayes:

$$P(H|E_1, E_2, \ldots, E_n) = \frac{P(E_1, \ldots, E_n|H)P(H)}{P(E_1, E_2, \ldots, E_n)}$$

- Has more evidence.
  - Likelihood is hard---but **conditional independence assumption**

$$P(H|E_1, E_2, \ldots, E_n) = \frac{P(E_1|H)P(E_2|H)\cdots, P(E_n|H)P(H)}{P(E_1, E_2, \ldots, E_n)}$$

Classification

- Expression

$$P(H|E_1, E_2, \ldots, E_n) = \frac{P(E_1|H)P(E_2|H)\cdots, P(E_n|H)P(H)}{P(E_1, E_2, \ldots, E_n)}$$

- *H*: some class we'd like to infer from evidence
  - We know prior *P(H)*
  - Estimate $P(E_i|H)$ from data! ("training")
  - Very similar to envelopes problem. **Part of HW2**

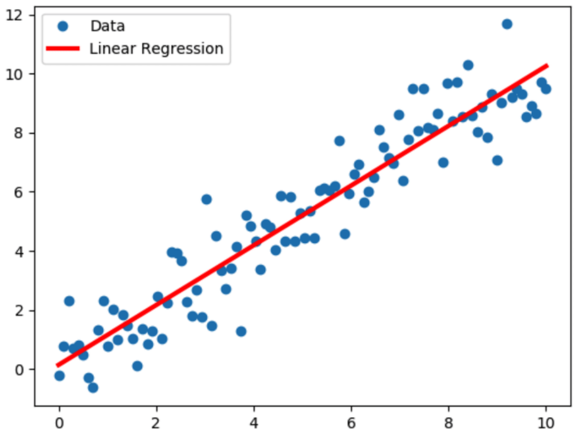
Linear Algebra: What is it good for?

- Everything is a **function**
  - With multiple inputs and outputs
- Linear functions
  - Simple, tractable
- Study of linear functions

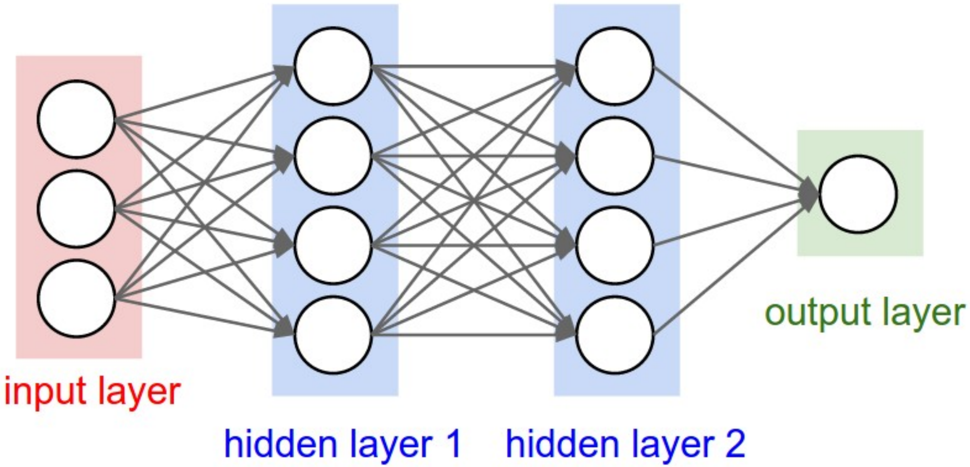
In AI/ML Context

Building blocks for **all models**

- E.g., linear regression; part of neural networks



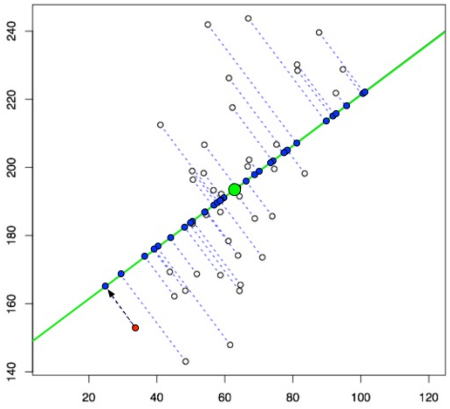
Hieu Tran



Stanford CS231n

Outline

- Basics: vectors, matrices, operations
- Dimensionality reduction
- Principal Components Analysis (PCA)

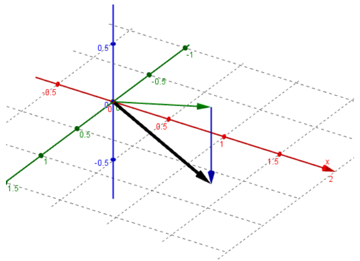


Lior Pachter

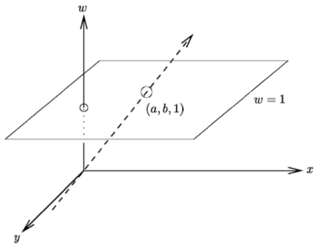
Basics: Vectors

Vectors

- Many interpretations
  - Physics: magnitude + direction



  - Point in a space

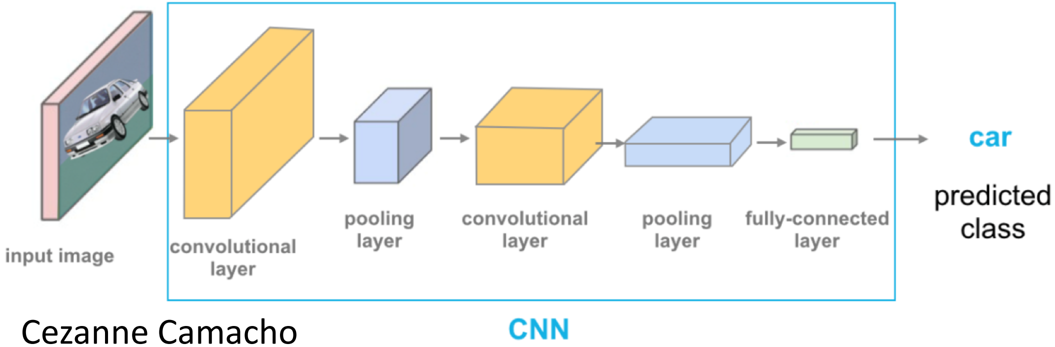


  - List of values (represents information)

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

Basics: **Vectors**

- Dimension
  - Number of values $x \in \mathbb{R}^d$
  - Higher dimensions: richer but more complex
- AI/ML: often use **very high dimensions**:
  - Ex: images!



Cezanne Camacho

Basics: **Matrices**

- Again, many interpretations
  - Represent linear transformations
  - Apply to a vector, get another vector
  - Also, list of vectors

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

- Not necessarily square
  - Dimension: $A \in \mathbb{R}^{c \times d}$
  - Indexing: A_{ij}

Basics: Transposition

- Transposes: flip rows and columns
  - Vector: standard is a column. Transpose: row vector
  - Matrix: go from *m x n* to *n x m*

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad x^T = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix} \quad A^T = \begin{bmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \\ A_{13} & A_{23} \end{bmatrix}$$

Matrix & Vector **Operations**

- Vectors
  - Addition: component-wise
    - Commutative: $x + y = y + x$
    - Associative: $x + y + z = x + (y + z)$

$$x + y = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ x_3 + y_3 \end{bmatrix}$$

  - Scalar Multiplication
    - Uniform stretch / scaling

$$cx = \begin{bmatrix} cx_1 \\ cx_2 \\ cx_3 \end{bmatrix}$$

Matrix & Vector **Operations**

- **Vector products**
  - Inner product (e.g., dot product)

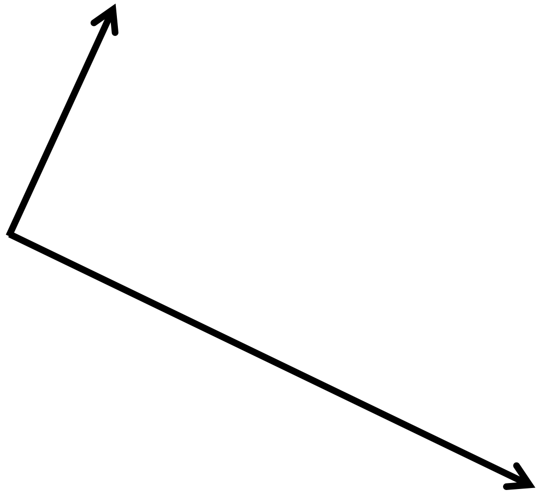
$$< x, y > := x^T y = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = x_1 y_1 + x_2 y_2 + x_3 y_3$$

  - Outer product

$$xy^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & x_1 y_3 \\ x_2 y_1 & x_2 y_2 & x_2 y_3 \\ x_3 y_1 & x_3 y_2 & x_3 y_3 \end{bmatrix}$$

Matrix & Vector **Operations**

- Inner product defines "orthogonality"
  - If $\langle x, y \rangle = 0$



- Vector norms: "size"

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Matrix & Vector **Operations**

- Matrices:
  - Addition: Component-wise
  - **Commutative!** + Associative

$$A + B = \begin{bmatrix} A_{11} + B_{11} & A_{12} + B_{12} \\ A_{21} + B_{21} & A_{22} + B_{22} \\ A_{31} + B_{31} & A_{32} + B_{32} \end{bmatrix}$$

  - Scalar Multiplication
  - "Stretching" the linear transformation

$$cA = \begin{bmatrix} cA_{11} & cA_{12} \\ cA_{21} & cA_{22} \\ cA_{31} & cA_{32} \end{bmatrix}$$

Matrix & Vector **Operations**

- Matrix-Vector multiplication
  - I.e., linear transformation; plug in vector, get another vector
  - Each entry in Ax is the inner product of a row of A with x

$$Ax = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + \ldots + A_{1n}x_n \\ A_{21}x_1 + A_{22}x_2 + \ldots + A_{2n}x_n \\ \vdots \\ A_{n1}x_1 + A_{n2}x_2 + \ldots + A_{nn}x_n \end{bmatrix}$$

Matrix & Vector **Operations**

Ex: feedforward neural networks. Input x.

- Output of layer k is

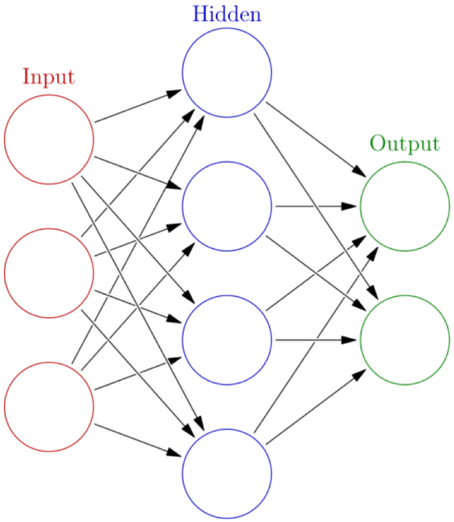
nonlinearity

$$f^{(k)}(x) = \sigma(W_k^T f^{(k-1)}(x)))$$

Output of layer k: vector

Weight **matrix** for layer k:
Note: linear transformation!

Output of layer k-1: **vector**

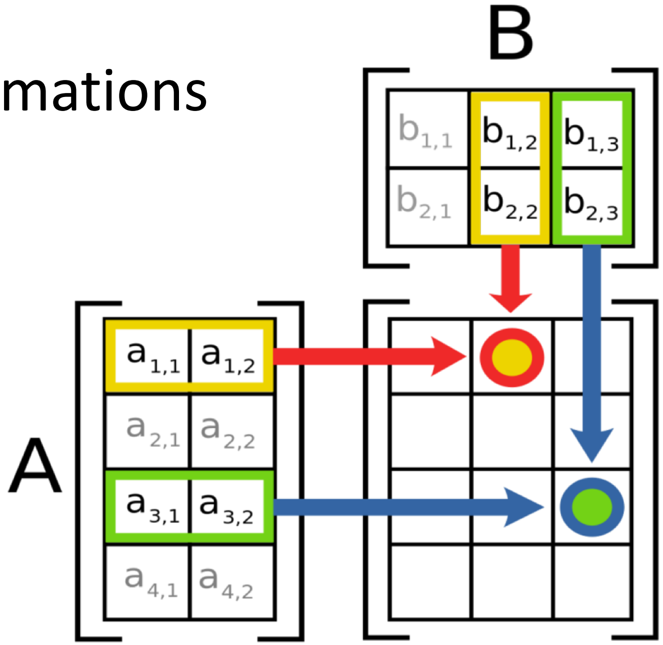


Wikipedia

Matrix & Vector **Operations**

- Matrix multiplication
  - "Composition" of linear transformations
  - **Not commutative** (in general)!

  - Lots of interpretations



Wikipedia

More on Matrix Operations

- Identity matrix:
  - Like "1"
  - Multiplying by it gets back the same matrix or vector
  - Columns are the "**standard basis vectors**" e_i
    - e_i denotes a vector with 1 in the i-th position, and 0 elsewhere

$$I = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Break & Quiz

- **Q 1.1**: What is $\begin{bmatrix} 1 & 2 \\ 3 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix}$?
- A. $[-1\ 1\ 1]^T$
- B. $[2\ 1\ 1]^T$
- C. $[1\ 3\ 1]^T$
- D. $[1.5\ 2\ 1]^T$

Break & Quiz

- **Q 1.2**: Given matrices $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{d \times m}, C \in \mathbb{R}^{p \times n}$

What are the dimensions of BAC^T

- A. *n x p*
- B. *d x p*
- C. *d x n*
- D. Undefined

Break & Quiz

- **Q 1.3**: A and B are matrices, neither of which is the identity matrix. Is $AB = BA$?

- A. Never
- B. Always
- C. Sometimes

More on Matrices: Inverses

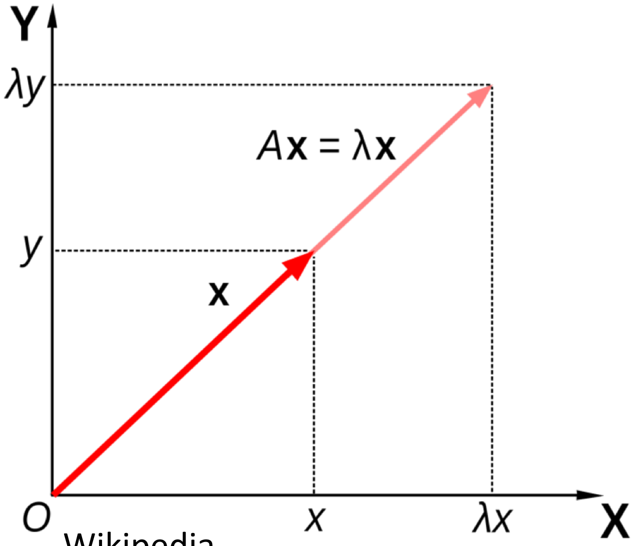
- If for *A* there is a *B* such that $AB = BA = I$
  - Then *A* is invertible/nonsingular, B is the inverse of A
  - Some matrices are **not** invertible!
  - Usual notation: A^{-1}
  - Only talk about inverse for **square** matrices

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

Eigenvalues & Eigenvectors

- For a square matrix A, solutions to $Av = \lambda v$
  - v (nonzero) is a vector: **eigenvector**
  - λ is a scalar: **eigenvalue**

  - Intuition: A is a linear transformation
  - In general can stretch/rotate vectors
  - E-vectors: only stretched (by e-vals)



Wikipedia

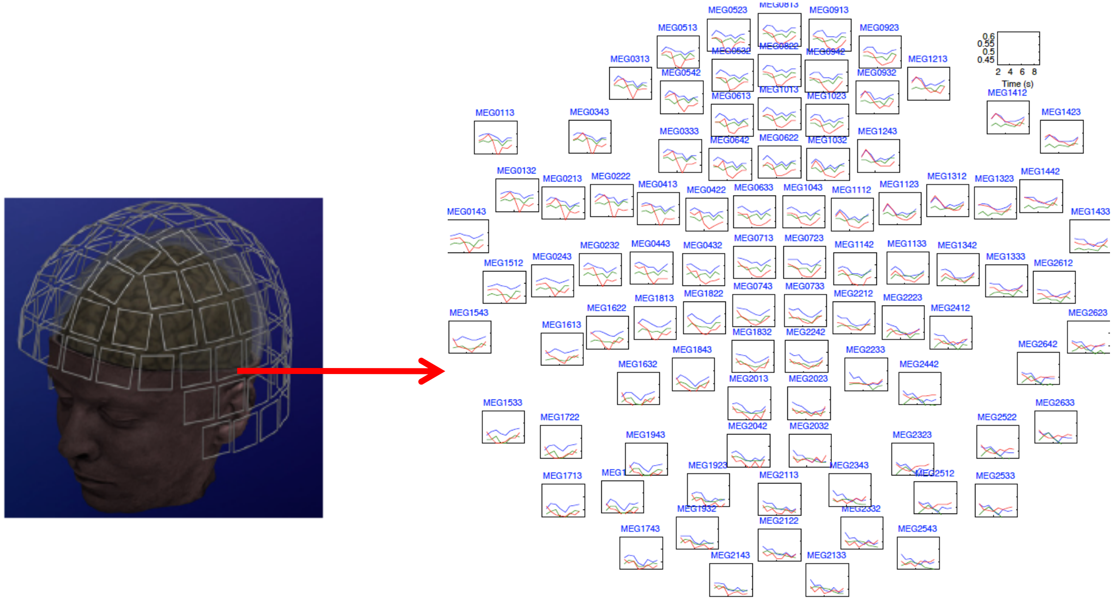
Dimensionality Reduction

- Vectors used to store features
  - Lots of data -> lots of features!

- Ex: Document classification
  - Each doc: thousands of words/millions of bigrams, etc

Dimensionality Reduction

Ex: MEG Brain Imaging: 120 locations x 500 time points x 20 objects



Dimensionality Reduction

Reduce dimensions

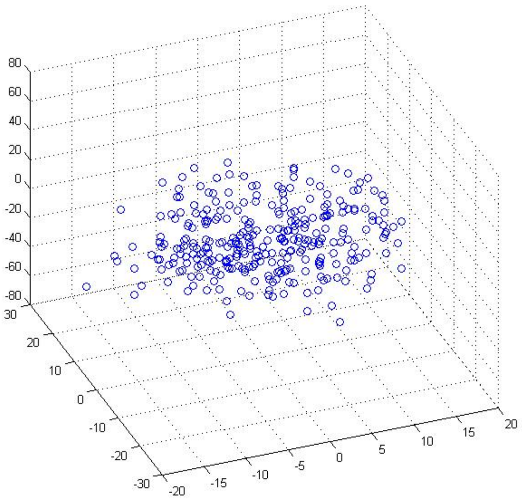
- Why?
  - Lots of features redundant
  - Storage & computation costs

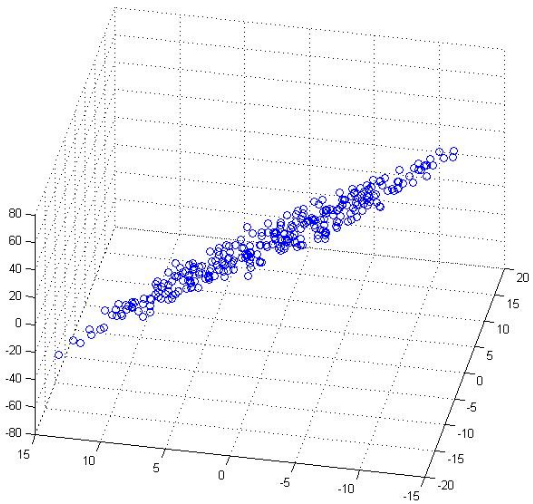
CreativeBloq

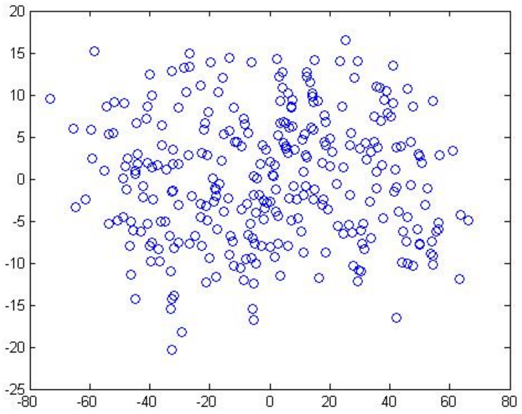
- Goal: take $x \in \mathbb{R}^d \rightarrow x \in \mathbb{R}^r$ for $r << d$
  - But, minimize information loss

Compression

Examples: 3D to 2D







Andrew Ng

Break & Quiz

Q 2.1: What is the inverse of

$$A = \begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix}$$

A. : $A^{-1} = \begin{bmatrix} -3 & 0 \\ 0 & -2 \end{bmatrix}$

B. : $A^{-1} = \begin{bmatrix} 0 & \frac{1}{3} \\ \frac{1}{2} & 0 \end{bmatrix}$

C. Undefined / *A* is not invertible

Break & Quiz

Q 2.2: What are the eigenvalues of $A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

A. -1, 2, 4
B. 0.5, 0.2, 1.0
C. 0, 2, 5
D. 2, 5, 1

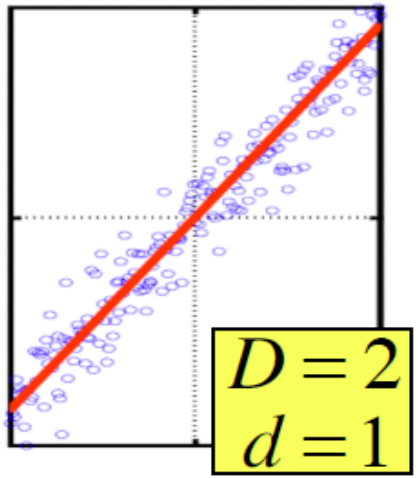
Break & Quiz

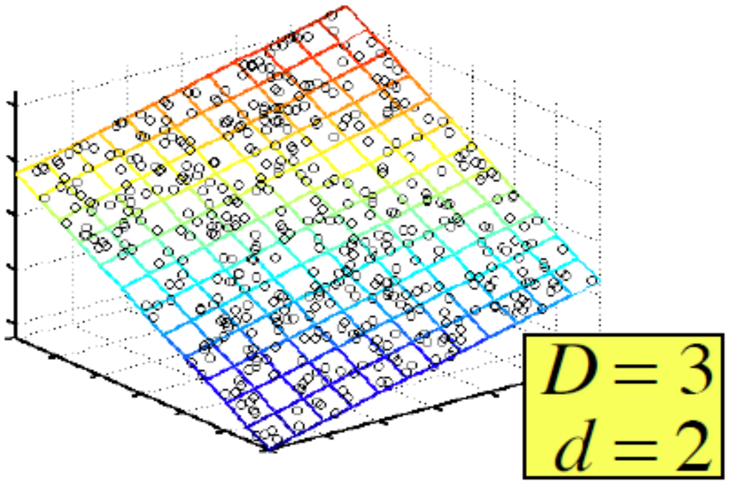
Q 2.3: Suppose we are given a dataset with n=10000 samples with 100-dimensional binary feature vectors. Our storage device has a capacity of 50000 bits. What’s the lower compression ratio we can use?

A. 20X
B. 100X
C. 5X
D. 1X

Principal Components Analysis (**PCA**)

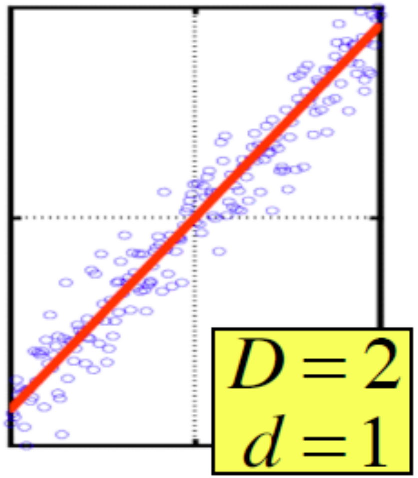
- A type of dimensionality reduction approach
  - For when data is **approximately lower dimensional**

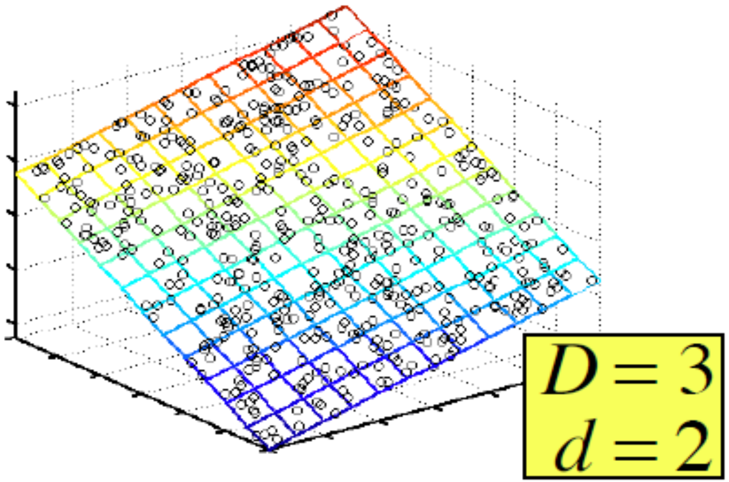




Principal Components Analysis (**PCA**)

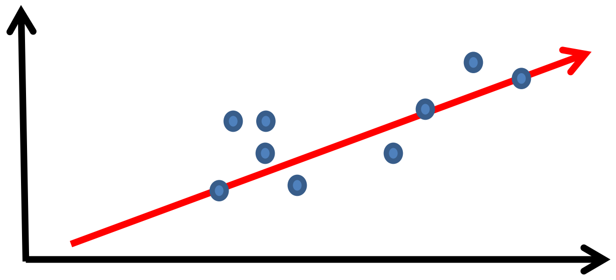
- Goal: find **axes** of a subspace
  - Will project to this subspace; want to preserve data





Principal Components Analysis (**PCA**)

- From 2D to 1D:
  - Find a $v_1 \in \mathbb{R}^d$ so that we maximize "variability" along v_1



  - New representations are along this vector (1D!)

Principal Components Analysis (**PCA**)

- From *d* dimensions to *r* dimensions:
  - Sequentially get $v_1, v_2, \ldots, v_r \in \mathbb{R}^d$ (the axes)
  - Orthogonal!
  - Still minimize the projection error
    - Equivalent to **“maximizing variability”**

  - The vectors are the **principal components**

PCA Setup

- **Inputs**
  - Data: $x_1, x_2, \ldots, x_n, \;\; x_i \in \mathbb{R}^d$
  - Can arrange into $X \in \mathbb{R}^{n \times d}$
  - **Centered**! $\frac{1}{n}\sum_{i=1}^{n} x_i = 0$
- **Outputs**
  - Principal components $v_1, v_2, \ldots, v_r \in \mathbb{R}^d$
  - Orthogonal!

PCA Goals

- Want directions/components (unit vectors) so that
  - Projecting data maximizes variance

$$\sum_{i=1}^{n} \langle v, x_i \rangle^2 = \|Xv\|^2$$

- Do this **recursively**
  - Get orthogonal directions $v_1, v_2, \ldots, v_r \in \mathbb{R}^d$

PCA First Step

- First component,

$$v_1 = \arg\max_{\|v\|=1} \sum_{i=1}^{n} \langle v, x_i \rangle^2$$

- Same as

$$v_1 = \arg\max_{\|v\|=1} \|Xv\|^2$$

PCA Recursion

- Once we have *k-1* components, next?

$$\hat{X}_k = X - \sum_{i=1}^{k-1} X v_i v_i^T$$

- Then do the same thing

Deflation

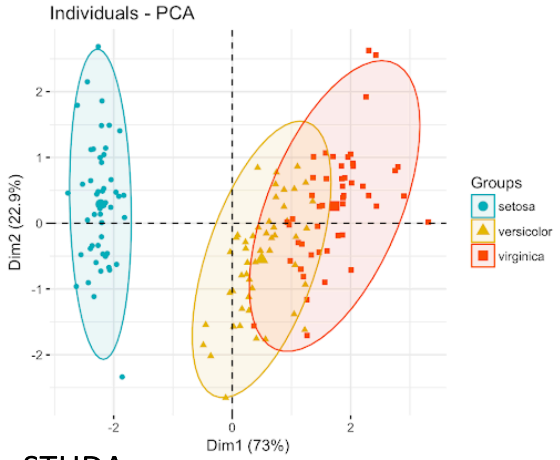
$$v_k = \arg\max_{\|v\|=1} \|\hat{X}_k v\|^2$$

PCA Interpretations

- The v's are eigenvectors of X^TX
- X^TX (proportional to) sample covariance matrix
  - When data is 0 mean!
  - I.e., PCA is eigendecomposition of sample covariance
- Nested subspaces *span(v1), span(v1,v2),…,*

Lots of Variations

- PCA, Kernel PCA, ICA, CCA
  - Unsupervised techniques to extract structure from high dimensional dataset
- Used for:
  - **Visualization**
  - Efficiency
  - Noise removal
  - Downstream machine learning use



STHDA

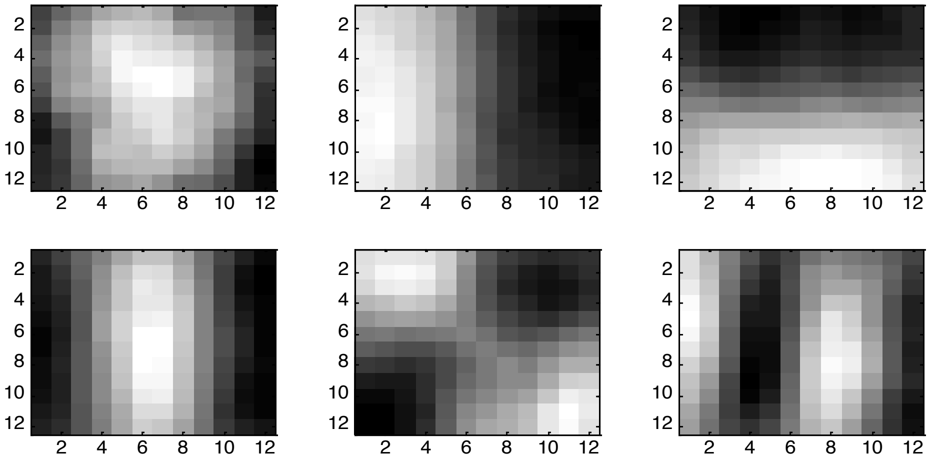
Application: Image Compression

- **Original image:**

- Divide into 12x12 patches
  - Each patch is a 144-D vector

Application: Image Compression

- 6 most important components (as an image)



Application: Image Compression

- Project to 6D,

Compressed

Original