



# CS 540 Introduction to Artificial Intelligence

## **Statistics & Linear Algebra Review**

Yingyu Liang  
University of Wisconsin-Madison  
**Sept 21, 2021**

Based on slides by Fred Sala

# Review: Bayesian Inference

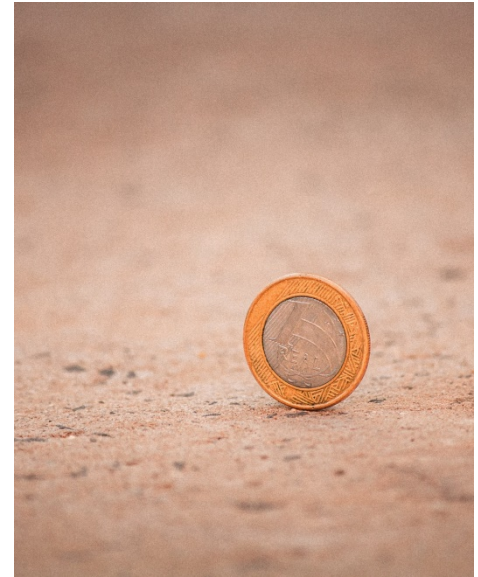
- Conditional Prob. & Bayes:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

- $H$ : some class we'd like to infer from evidence
  - Need to plug in prior, likelihood, etc.
  - How to estimate?

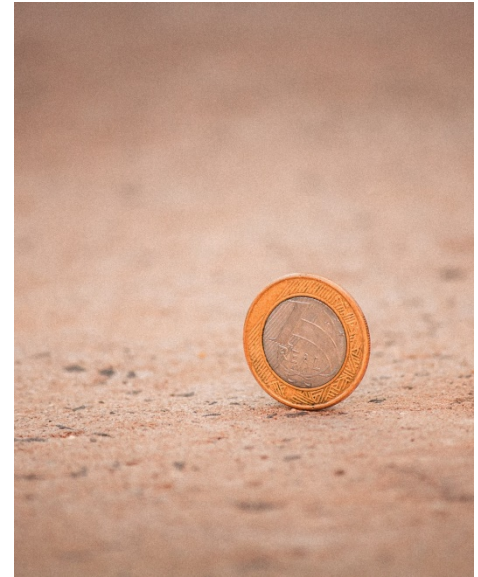
# Samples and Estimation

- Usually, we don't know the distribution ( $P$ )
  - Instead, we see a bunch of samples
- Typical statistics problem: **estimate parameters** from samples
  - Estimate probability  $P(H)$
  - Estimate the mean  $E[X]$
  - Estimate parameters  $P_{\theta}(X)$



# Samples and Estimation

- Typical statistics problem: **estimate parameters** from samples
  - Estimate probability  $P(H)$
  - Estimate the mean  $E[X]$
  - Estimate parameters  $P_{\theta}(X)$
- Example: Bernoulli with parameter  $p$ 
  - Mean  $E[X]$  is  $p$



# Examples: Sample Mean

- Bernoulli with parameter  $p$
- See samples  $x_1, x_2, \dots, x_n$ 
  - Estimate mean with **sample mean**

$$\hat{\mathbb{E}}[X] = \frac{1}{n} \sum_{i=1}^n x_i$$

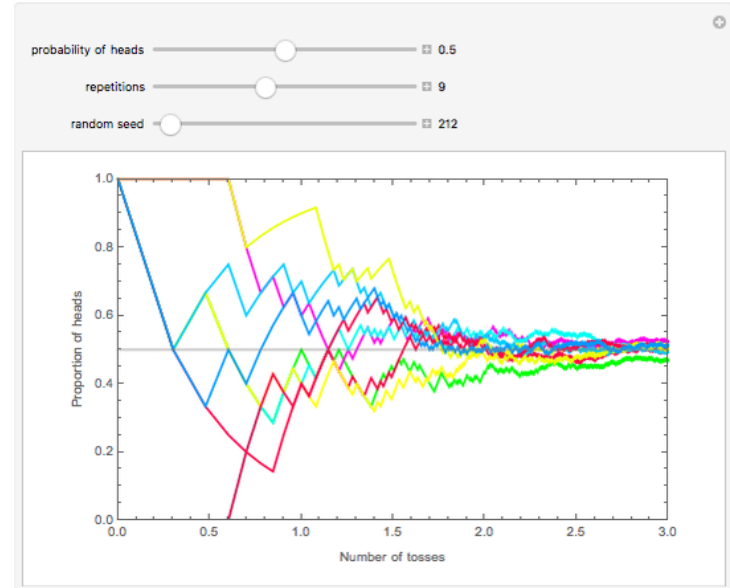
- No different from counting heads



# Estimation Theory

- How do we know that the sample mean is a good estimate of the true mean?
  - Law of large numbers
  - Central limit theorems
  - Concentration inequalities

$$P(|\mathbb{E}[X] - \hat{\mathbb{E}}[X]| \geq t) \leq \exp(-2nt^2)$$



Wolfram Demo

# Linear Algebra: What is it good for?

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

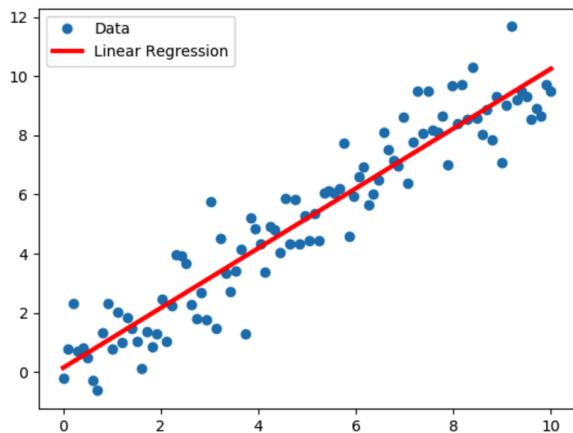




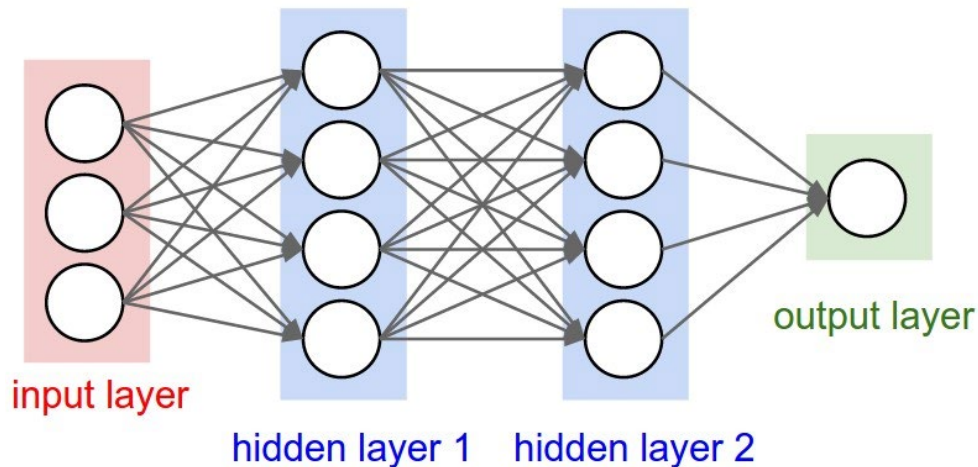
# In AI/ML Context

## Building blocks for **all models**

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



Hieu Tran



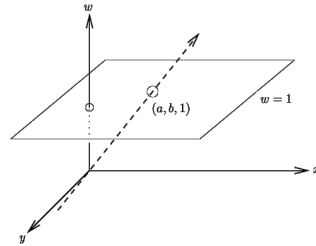
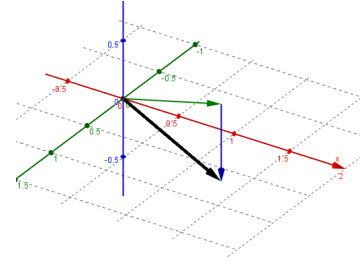
Stanford CS231n



# Basics: Vectors

## Vectors

- Many interpretations
  - Physics: magnitude + direction
  - Point in a space
  - List of values (represents information)

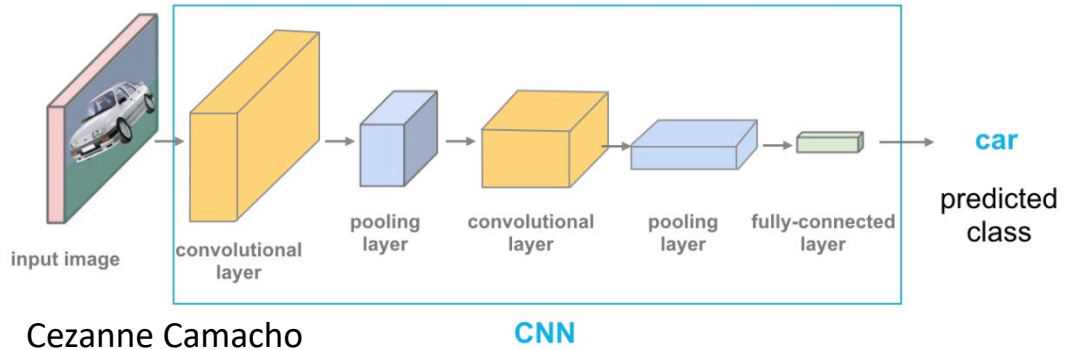


$$\mathbf{x} =$$

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

# 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!



# 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
  - Indexing!  $A \in \mathbb{R}^{c \times d}$
  - Dimensions: #rows x #columns

# Basics: Transposition

- Transposes: flip rows and columns
  - Vector: standard is a column. Transpose: row
  - Matrix: go from  $m \times n$  to  $n \times m$

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

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

# Vector Operations

- Addition, Scalar Multiplication
- Inner product (e.g., dot product)

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

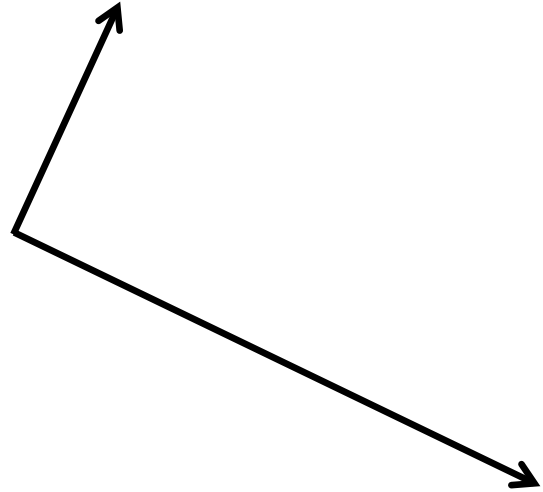
- Outer product

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

# 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

- Addition, scalar multiplication
- Matrix-Vector multiply
  - linear transformation: plug in vector, get another vector
  - Each entry in  $Ax$  is the inner product of a row of  $A$  with  $x$

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



# Matrix & Vector Operations

Ex: feedforward neural networks. Input  $x$ .

- Output of layer  $k$  is

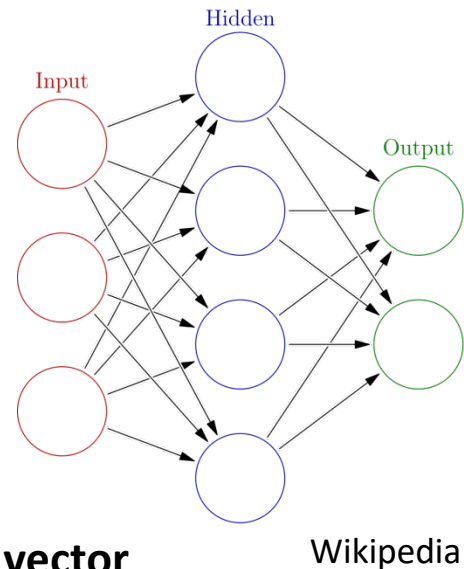
$$f^{(k)}(x) = \sigma(W_k^T f^{(k-1)}(x))$$

nonlinearity

↑  
Output of layer  $k$ : vector

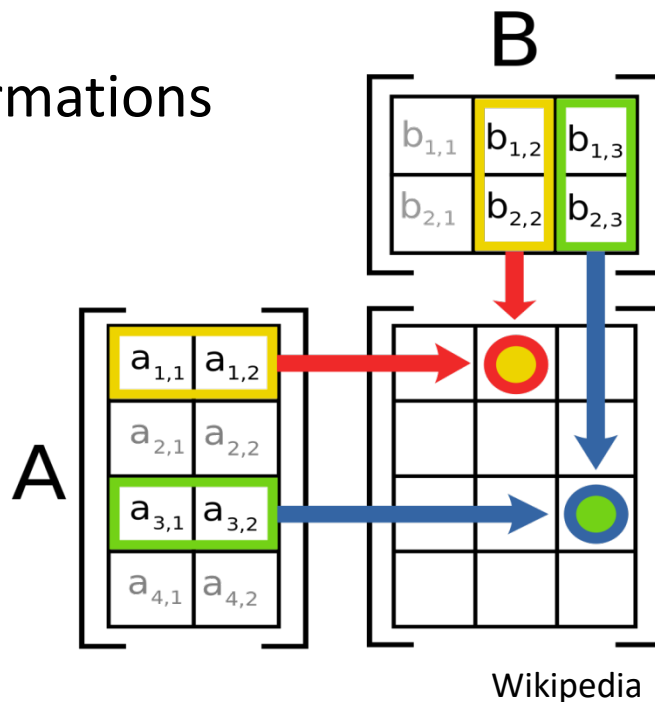
↑  
Weight **matrix** for layer  $k$ :  
Note: linear transformation!

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



# Matrix & Vector Operations

- Matrix multiplication
  - “Composition” of linear transformations
  - **Not commutative** (in general)!
  - Lots of interpretations



# More on Matrices: Identity

- Identity matrix:
  - Like “1”
  - Multiplying by it gets back the same matrix or vector
  - Rows & columns are the “**standard basis vectors**”  $e_i$

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

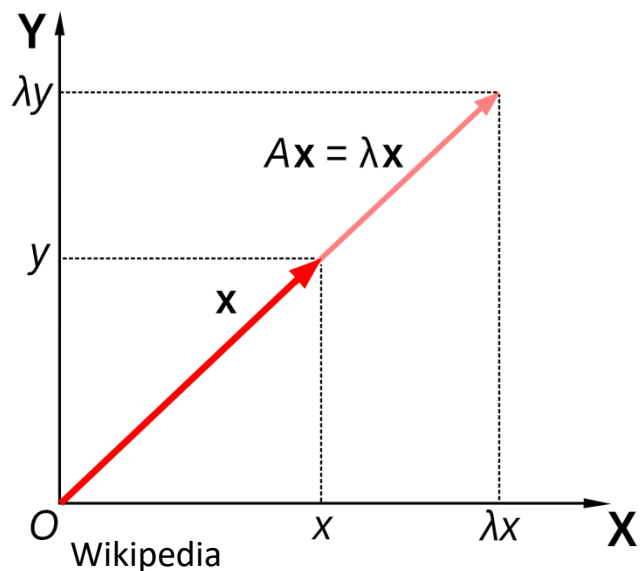
# More on Matrices: Inverses

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

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

# Eigenvalues & Eigenvectors

- For a square matrix  $A$ , solutions to  $Av = \lambda v$ 
  - $v$  (nonzero) is a vector: **eigenvector**
  - $\lambda$  is a scalar: **eigenvalue**
  - Intuition:  $A$  is a linear transformation;
  - Can stretch/rotate vectors;
  - E-vectors: only stretched (by e-vals)



# Dimensionality Reduction

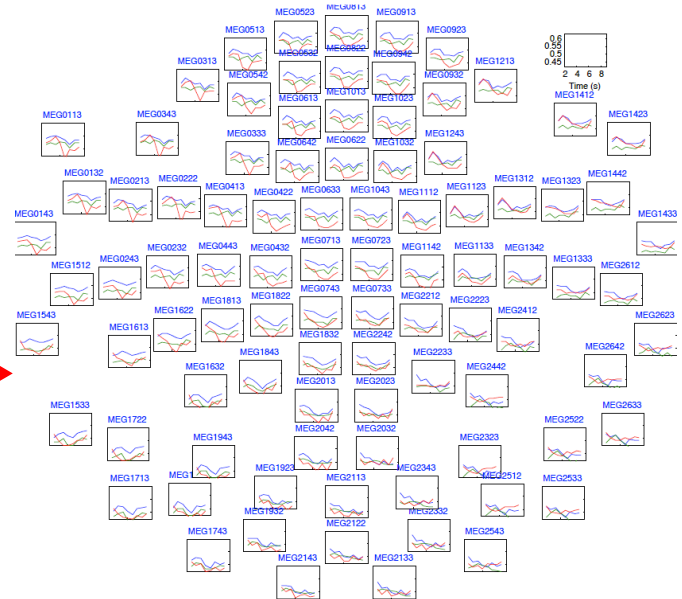
- Vectors used to store features
  - Lots of data -> lots of features!
- Document classification
  - Each doc: thousands of words, etc.
- Netflix surveys: 480189 users x 17770 movies

	movie 1	movie 2	movie 3	movie 4	movie 5	movie 6
Tom	5	?	?	1	3	?
George	?	?	3	1	2	5
Susan	4	3	1	?	5	1
Beth	4	3	?	2	4	2

# Dimensionality Reduction

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

- Or any image





# Dimensionality Reduction

Reduce dimensions

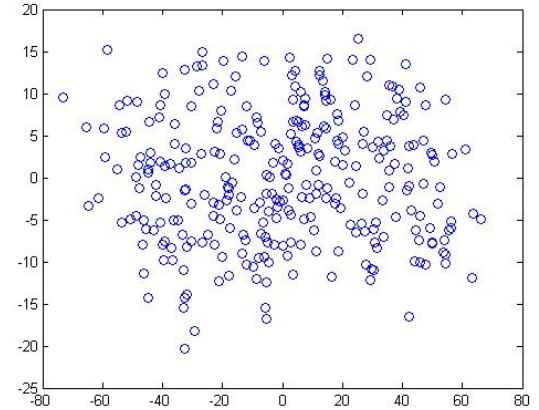
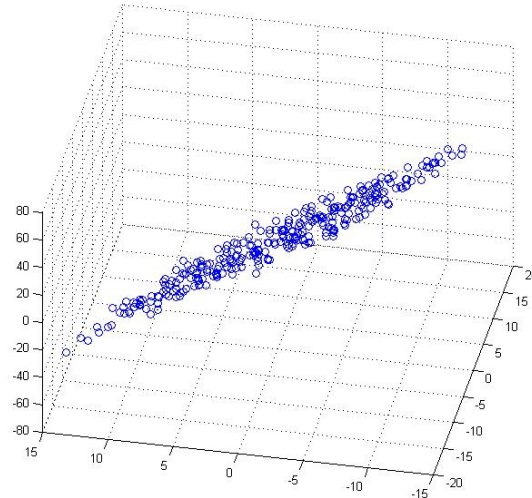
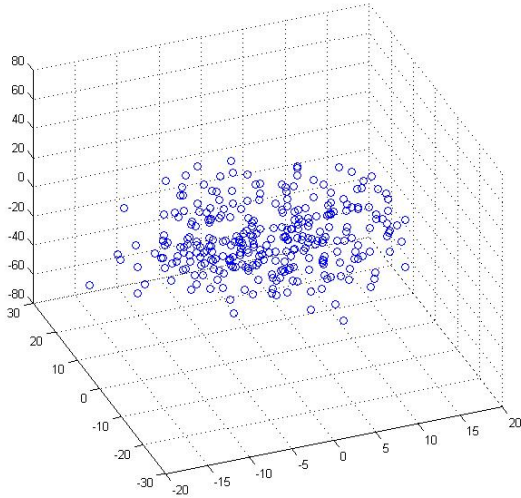
- Why?
  - Lots of features redundant
  - Storage & computation costs



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

# Compression

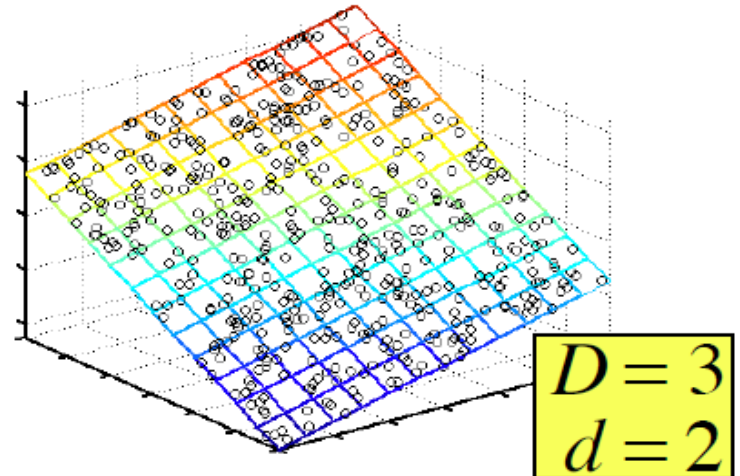
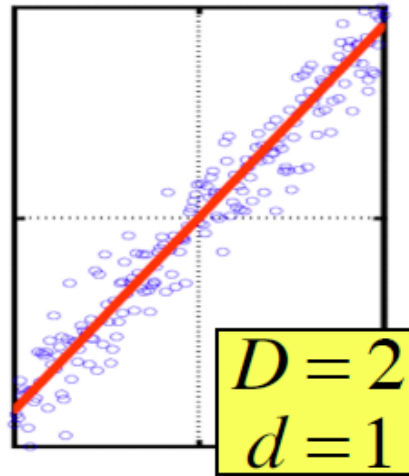
## Examples: 3D to 2D



Andrew Ng

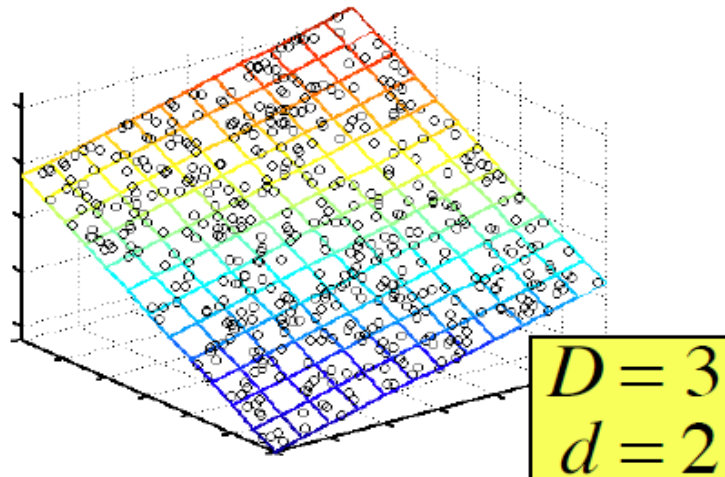
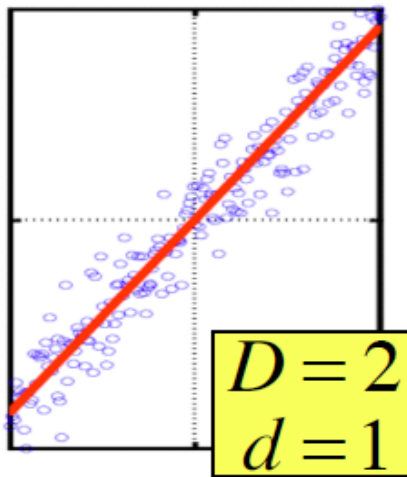
# Principal Components Analysis (PCA)

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



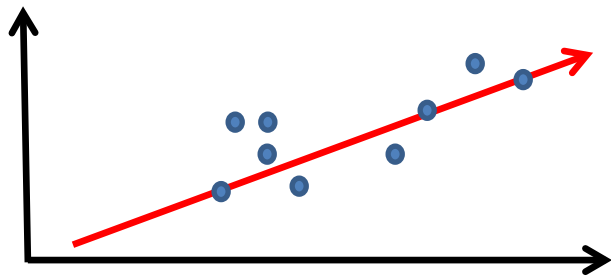
# Principal Components Analysis (PCA)

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



# Principal Components Analysis (PCA)

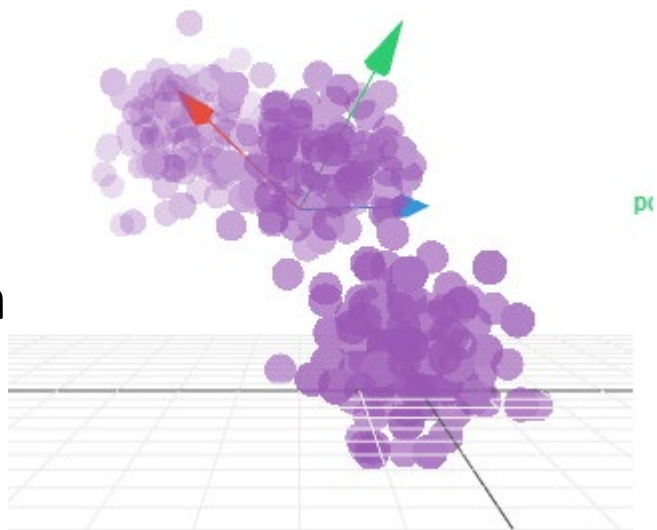
- From 2D to 1D:
  - Find a  $v_1 \in \mathbb{R}^d$  so that we maximize “variability”
  - IE,



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

# Principal Components Analysis (PCA)

- From  $d$  dimensions to  $r$  dimensions
  - Sequentially get  $v_1, v_2, \dots, v_r \in \mathbb{R}^d$
  - Orthogonal!
  - Still maximize “variability”
  - The vectors are the **principal compon**



Victor Powell

# PCA Setup

- **Inputs**

- Data:  $x_1, x_2, \dots, x_n, x_i \in \mathbb{R}^d$

- Can arrange into  $X \in \mathbb{R}^{n \times d}$

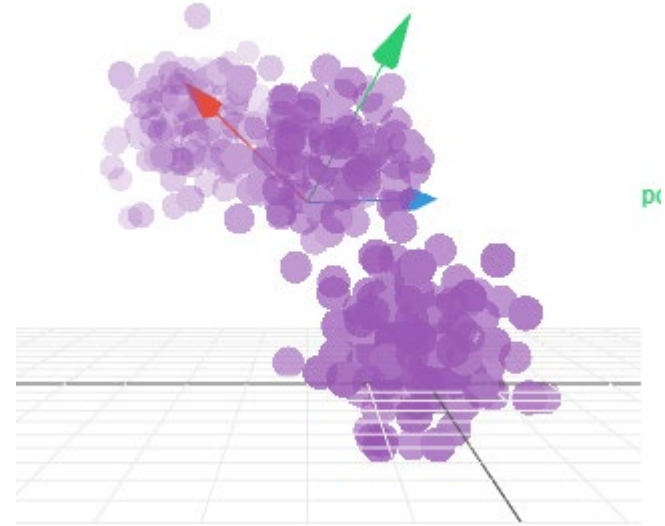
- **Centered!**

$$\frac{1}{n} \sum_{i=1}^n x_i = 0$$

- **Outputs**

- Principal components  $v_1, v_2, \dots, v_r \in \mathbb{R}^d$

- Orthogonal!



Victor Powell



# PCA Goals

- Want directions/components (unit vectors) so that
  - Projecting data maximizes variance
  - What's variance of the projections?  $\sum_{i=1}^n \langle x_i, v \rangle^2 = \|Xv\|^2$
- Do this **recursively**
  - Get orthogonal directions  $v_1, v_2, \dots, v_r \in \mathbb{R}^d$

# PCA First Step

- First component,

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

- Same as getting

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

# 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

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

**Deflation**



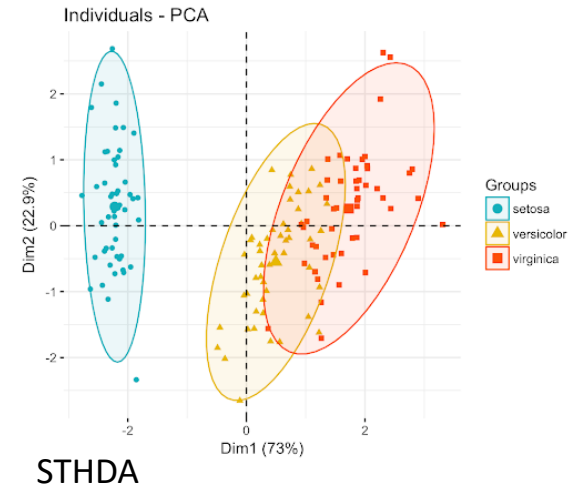
# PCA Interpretations

- The  $v$ 's are eigenvectors of  $X^T X$  (**Gram matrix**)
  - Show via Rayleigh quotient
- $X^T X$  (proportional to) sample covariance matrix
  - When data is 0 mean!
  - I.e., PCA is eigendecomposition of sample covariance
- Nested subspaces  $\text{span}(v_1), \text{span}(v_1, v_2), \dots,$



# Lots of Variations

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



# Application: Image Compression

- Start with image; divide into 12x12 patches

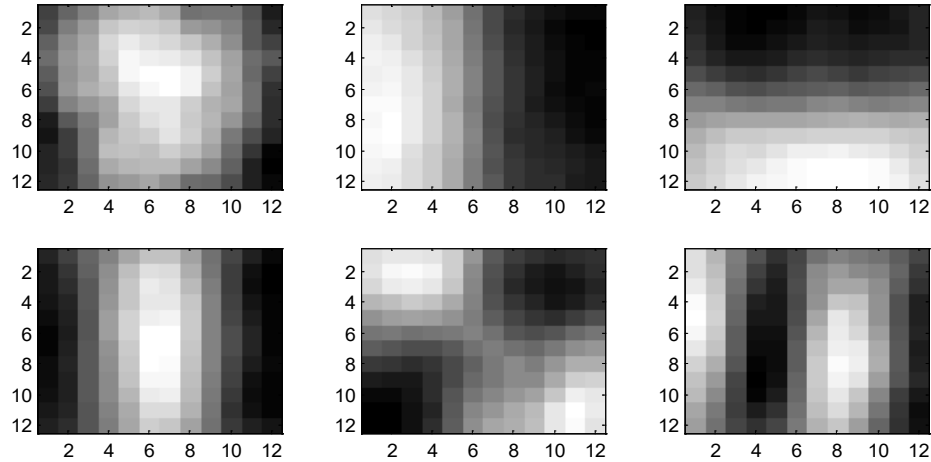
- I.E., 144-D vector

- **Original image:**



# Application: Image Compression

- 6 most important components (as an image)





# Application: Image Compression

- Project to 6D,



Compressed



Original