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

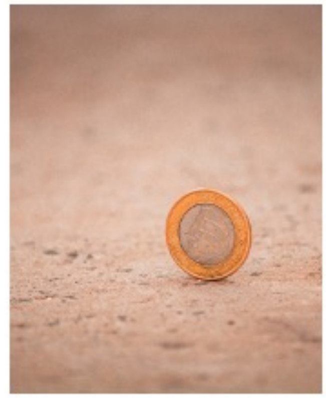
- Conditional Prob. & Bayes:

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

- *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 $P_\theta(X)$

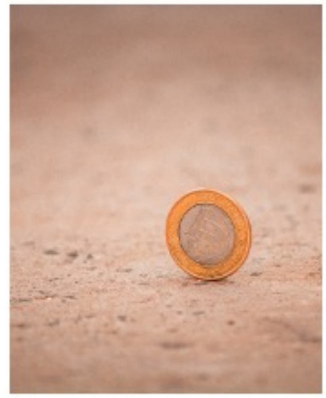


Probability theory tells us how to handle probabilities. But the probability terms (like those on the right hand of Bayes' rule) are unknown; in practice, we need to estimate them using data.

This falls in statistics: how to estimate various properties of the distribution given samples from the distribution.

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 $x_1, x_2, \ldots, x_n$
  - Estimate mean with **sample mean**

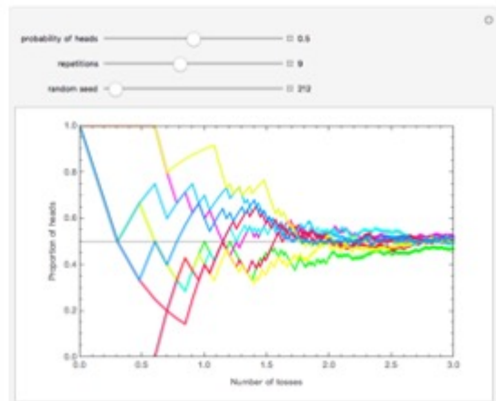
$$\hat{\mathbb{E}}[X] = \frac{1}{n}\sum_{i=1}^{n} x_i$$

  - 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

Theory about how close the estimate is to the truth:

1. Law of large numbers: the sample mean tends to the true mean in the infinity limit
2. CLT: the average of independent random variables looks like the normal distribution in the infinity limit
3. Concentration inequalities: quantitative analysis of the error for finite samples (instead of the infinity limit)

Break & Quiz

Q 2.1: You see samples of *X* given by [0,1,1,2,2,0,1,2]. Empirically estimate $E[X^2]$

A. 9/8
B. 15/8
C. 1.5
D. There aren't enough samples to estimate $E[X^2]$

Break & Quiz

Q 2.1: You see samples of X given by [0,1,1,2,2,0,1,2]. Empirically estimate $E[X^2]$

A. 9/8
B. 15/8
C. 1.5
D. There aren't enough samples to estimate $E[X^2]$

(0+1+1+4+4+0+1+4)/8 = 15/8

Break & Quiz

Q 2.2: You are empirically estimating *P(X)* for some random variable *X* that takes on 100 values. You see 50 samples. How many of your *P(X=a)* estimates might be 0?

A. None.
B. Between 5 and 50, exclusive.
C. Between 50 and 100, inclusive.
D. Between 50 and 99, inclusive.

Break & Quiz

Q 2.2: You are empirically estimating *P(X)* for some random variable *X* that takes on 100 values. You see 50 samples. How many of your *P(X=a)* estimates might be 0?

A. None.
B. Between 5 and 50, exclusive.
C. Between 50 and 100, inclusive.
D. Between 50 and 99, inclusive.

In one extreme, all samples have the same value. Then the estimated probability for all the other 99 values will be 0.

In the other extreme, all samples have different values. Then the estimated probability for the other 50 values will be 0.

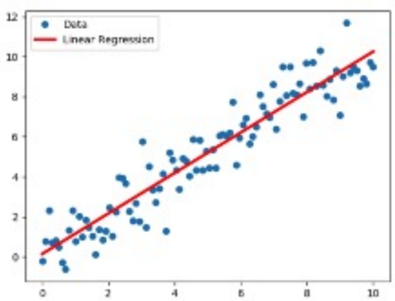
Linear Algebra: What is it good for?

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

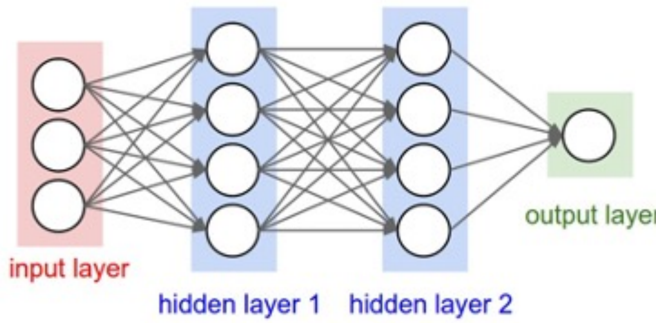
In AI/ML Context

Building blocks for **all models**

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



Hieu Tran

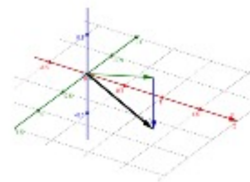


Stanford CS231n

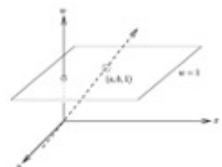
Basics: **Vectors**

Vectors

- Many interpretations
  - Physics: magnitude + direction



  - Point in a space

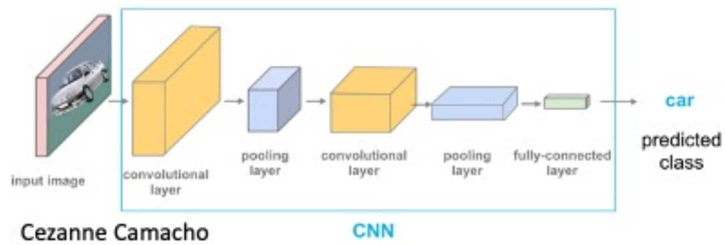


  - List of values (represents information)

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

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
  - 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 x n* to *n x m*

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

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

A vector is usually regarded as a column vector (ie, a matrix of dimension d by 1).

Vector **Operations**

- Addition, Scalar Multiplication
- 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

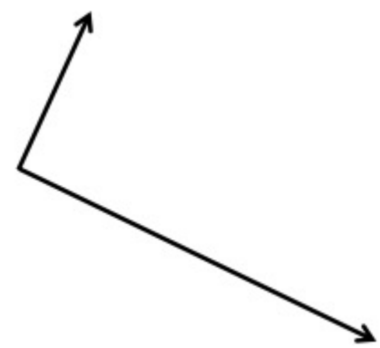
$$xy^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & x_1 y_3 \\ x_2 y_1 & x_2 y_2 & x_2 y_3 \\ x_3 y_1 & x_3 y_2 & x_3 y_3 \end{bmatrix}$$

Vector **Operations**

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

- Vector norms: "size"

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



If two vectors have 0 inner product then we say they are orthogonal.

Note that the slide shows the l2 norm (Euclidean norm) of the vector. There exists other norms.

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*

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

Matrix & Vector **Operations**

Ex: feedforward neural networks. Input *x*.

- Output of layer k is

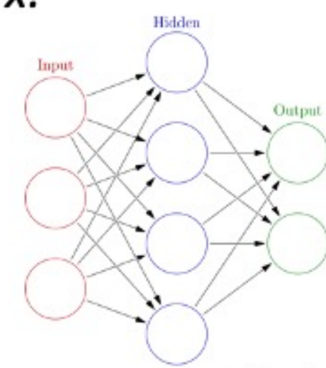
nonlinearity

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

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

Output of layer k: vector

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



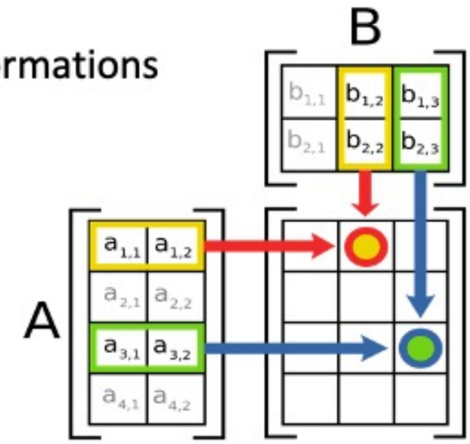
Wikipedia

See more details in the lectures on neural networks.

Matrix & Vector **Operations**

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

  - Lots of interpretations



Wikipedia

Matrix C=AB, then the entry of C at row i and column j is the inner product of the row i of A and the column j of B.

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

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

For any matrix A that can be multiplied with I (ie, of the same dimension as I), we have A = AI = IA.

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!

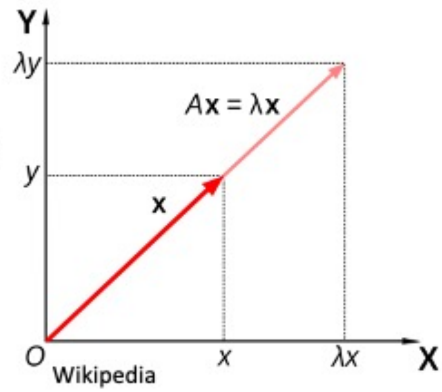
  - Usual notation: A^{-1}

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

Eigenvalues & Eigenvectors

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

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



Eigenvectors are those directions along which A only stretch (by a scaling factor = eigenvalue) but not rotate.

Dimensionality Reduction

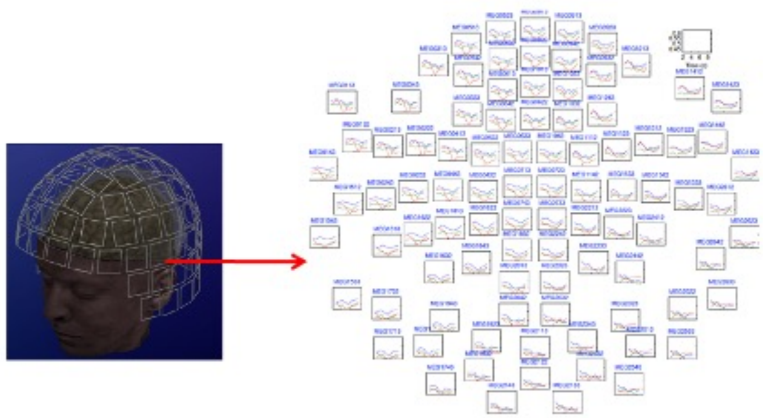
- Vectors used to store features
  - Lots of data -> lots of features!
- Document classification
  - Each doc: thousands of words, etc.
- Netflix surveys: 480189 users x 17770 movies

	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

CreativeBloq

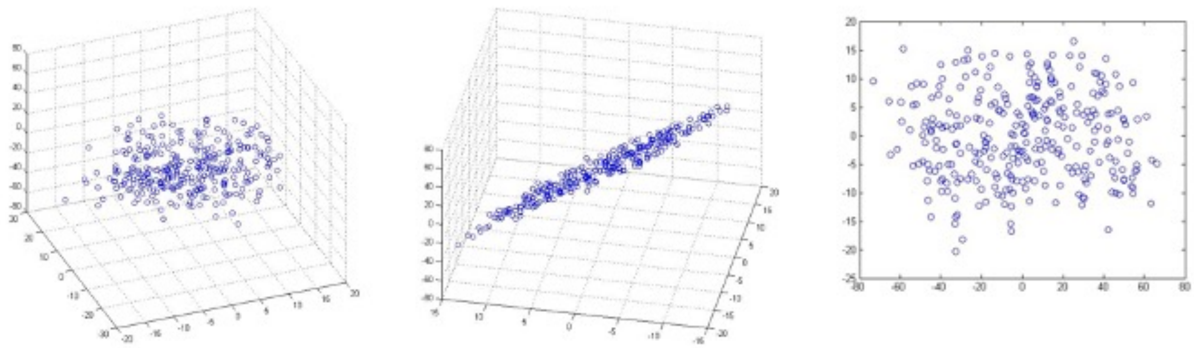
- Goal: take $x \in \mathbb{R}^d \rightarrow x \in \mathbb{R}^r$ for $r << d$
  - But, minimize information loss

Dimensionality Reduction: take the original data point, map it to a point in a lower dimension.

Would like to minimize information loss. The "information loss" can vary, and different definitions of information loss lead to different reduction techniques.

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 2.1: 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 2.2: 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

Let e_i denote the basis vectors.

Clearly:
A e_1 = 2 e_1
A e_2 = 5 e_2
A e_3 = 1 e_1
So the eigenvalues are 2 5 1, and the corresponding eigenvectors are e_1 e_2 and e_3.

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

Break & Quiz

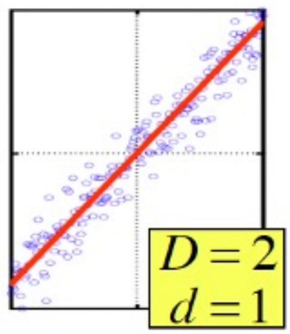
Q 2.3: Suppose we are given a dataset with n=10000 samples with 100-dimensional binary feature vectors. Our storage device has a capacity of 50000 bits. What's the lower compression ratio we can use?

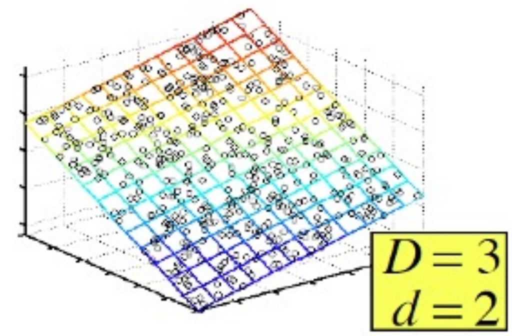
A. 20X
B. 100X
C. 5X
D. 1X

If we use 5X or 1X, then we need 200,000 or 1000,000 bits.
If we use 20X, then we need 50,000 bits.

Principal Components Analysis (**PCA**)

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

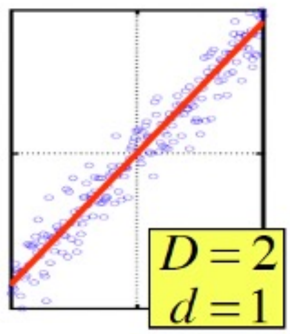


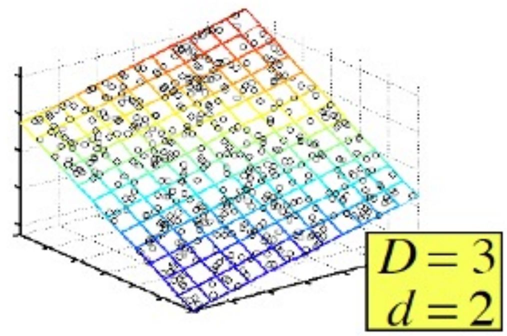


PCA is a classic dimensionality reduction method. It’s good for the case when the data points are close to a low dimensional subspace.

Principal Components Analysis (**PCA**)

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

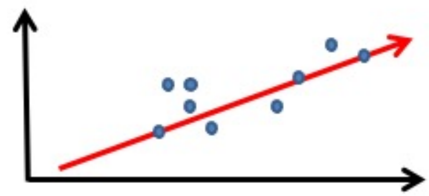




Suppose the ideal case when the data points are in dim D, but close to a low dimensional subspace with dim d. PCA is going to find the d axes of that subspace, then project to that subspace to get a lower dimensional representation (ie, represent the projection using the axes of the subspace and thus get a representation of dim d).

Principal Components Analysis (**PCA**)

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



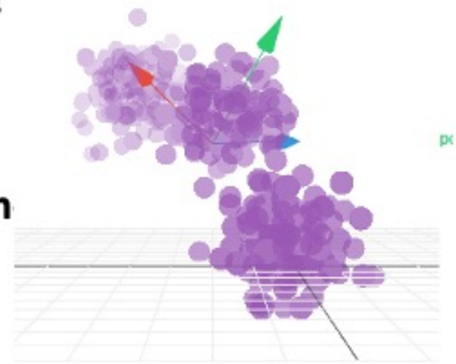
  - New representations are along this vector (1D!)

How to find the axes in the general case? We need to define a metric measuring the quality of the axes.

Consider the case mapping to 1 dimension, ie, would like to find one axis. Then intuitively, we would like the projected data on that axis to be spread out.

Principal Components Analysis (**PCA**)

- From *d* dimensions to *r* dimensions
  - Sequentially get $v_1, v_2, \ldots, v_r \in \mathbb{R}^d$
  - Orthogonal! Also, of unit length
  - Still maximize "variability"
  - The vectors are the **principal compon**

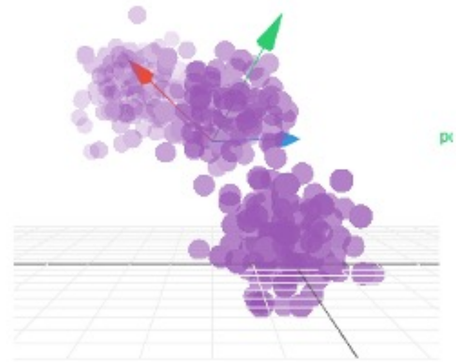


Victor Powell

Consider the case mapping to r dimension, ie, would like to find r axes. Then we can define them sequentially, still wanting to maximize the variance of the projected data. We further need them to be orthogonal to each other (so that they form the axes of a subspace). Usually, we also require them to be of unit length (ie, the Euclidean norm = 1), so that they are directions (unit length vectors).

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! Also, of unit length



Victor Powell

Formal definition.

We assume the data are centered (ie, average=0). This is to simplify the computation later on.
If not centered, we can compute the average of the data, and subtract that from all the data points. Then we can get a centered dataset.

We can stack the data points as rows in a matrix, then we get a matrix X. In other words, the i-th row of X is the point x_i.

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 $v_1, v_2, \ldots, v_r \in \mathbb{R}^d$

Formal definition continued.

The quality or score of the directions are defined as the variance of the projections. How to compute the variance?
1. Let v be a direction (ie, a unit-length vector). Then the length of the projection of a point x_i on v is just the inner product <x_i, v>. This is also the representation of x_i's projection using the axis v.
2. The mean of the projections are \sum_i <x_i, v> = <\sum_i x_i, v> = <0, v> = 0. Note that this is where we use the assumption that the data are centered, and this simplifies the computation.
3. Then the variance is \sum_i (<x_i, v> - mean of projections)^2 = \sum_i <x_i, v>^2. This is exactly the squared Euclidean norm of Xv, where X is a matrix obtained by stacking x_i as the i-th row.

Given the math expression of the variance of the projections, we can then compute the principle components recursively.

PCA First Step

- **First component,**

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

- **Same as getting**

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

Formal definition continued.

The first component is just the direction (unit-length vector) that maximizes the variance of the projection, among all directions.

PCA Recursion

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

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

Deflation

- Then do the same thing

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

Formal definition continued.

The second component is just the direction that maximizes the variance of the projection, among all directions orthogonal to the already found first component.
The third component is just the direction that maximizes the variance of the projection, among all directions orthogonal to the already found first and second components.
…
This defines the principal components.

To compute the k-th component given the first (k-1) components, we can remove the part of the data along the already found components.
That is, from each point x_j, subtract its projection on the first (k-1) components.

1. Recall that the length of the projection on v_i is <x_j, v_i> = x_i^T v_i, so the vector form of the projection is <x_j, v_i> v_i = x_j^T v_i v_i^T
2. Then we compute x_j^T – \sum_{i=1}^k x_j^T v_i v_i^T
3. In matrix form, it's just X - \sum_{i=1}^k Xv_i v_i^T. This is called deflation.

Then (the direction that maximizes the variance of the projection of the deflated data) will be (the direction that maximizes the variance of the projection of the original data among all directions orthogonal to the already found k-1 components).

This thus gives a recursive method to compute the principal components.

PCA Interpretations

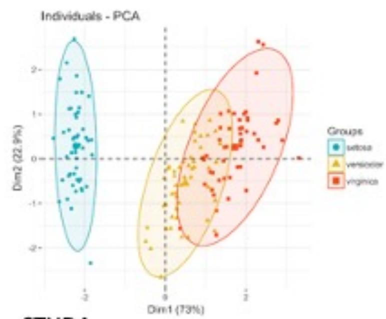
- The v's are eigenvectors of X^TX **(Gram matrix)**
  - Show via Rayleigh quotient
- X^TX (proportional to) sample covariance matrix
  - When data is 0 mean!
  - I.e., PCA is eigendecomposition of sample covariance
- Nested subspaces *span(v1), span(v1,v2),…,*

It can be proved that the first k principal components are just the top k eigenvectors of the Gram matrix with the largest eigenvalues. (This course doesn't require understanding this.)
When the data are centered, the Gram matrix is just (proportional to) the sample covariance matrix.

This gives a method to compute the principal components:
1. Center the data
2. Compute the Gram matrix
3. Find the top k eigenvectors of the Gram matrix

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



STHDA

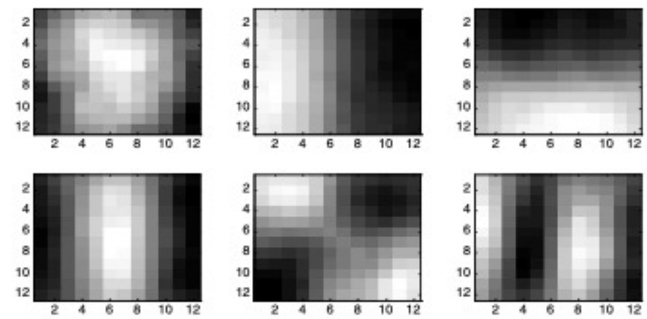
Application: Image Compression

- Start with image; divide into 12x12 patches
  - I.E., 144-D vector
  - **Original image:**

We divide the images into 12x12 patches, and flatten the patches into 144-dim vectors. These vectors are our dataset.

Application: Image Compression

- 6 most important components (as an image)



We compute the principal components (which are 144-dim vectors). We convert them back to 12x12 patches and visualize them.

Application: Image Compression

- Project to 6D,

Compressed

Original

We can compute the projection of the original patches to the found 6 principal components, ie, <x_i, v_i> v_i^T.
This is the compressed image.

Break & Quiz

Q 3.1: What is the projection of $[1\ 2]^T$ onto $[0\ 1]^T$?

- A. $[1\ 2]^T$
- B. $[-1\ 1]^T$
- C. $[0\ 0]^T$
- D. $[0\ 2]^T$

Break & Quiz

Q 3.1: What is the projection of $[1\ 2]^T$ onto $[0\ 1]^T$?

- A. $[1\ 2]^T$
- B. $[-1\ 1]^T$
- C. $[0\ 0]^T$
- **D. $[0\ 2]^T$**

Compute <[1,2], [0,1]> [0, 1] = [0, 2]

Break & Quiz

Q 3.2: We wish to run PCA on 10-dimensional data in order to produce *r*-dimensional representations. Which is the most accurate?

- A. $r \leq 3$
- B. $r < 10$
- C. $r \leq 10$
- D. $r \leq 20$

Break & Quiz

Q 3.2: We wish to run PCA on 10-dimensional data in order to produce r-dimensional representations. Which is the most accurate?

- A. $r \leq 3$
- B. $r < 10$
- **C. $r \leq 10$**
- D. $r \leq 20$

The number of principal components is smaller or equal to the original dimension.