EECS 16B : Module 3 / Lecture 5

Announcements:

- 0.25 EC point for each lecture you attend for rest of the term.
  - links.eecs16b.org/lecture-ec
- Lab Design Contest: See Ed post (EC opportunities!)
- Course Evaluations: 1.5 EC pts. with ≥ 80% response rate.

Last time:

- PCA motivation / introduction

$$A_\ell = \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^T \quad \left(\text{Note: } A = \sum_{j=1}^{r} \sigma_j \vec{u}_j \vec{v}_j^T\right)$$

"effective" rank of data matrix $A = \ell \ll r$ typically

- Young-Eckart Theorem:

$$\min \|A - B\|_F^{2*} \quad \text{s.t. } B \text{ has rank } \ell \quad \text{is solved by } B = A_\ell = \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^T$$

$*\left(\|X\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} |X_{ij}|^2\right)$ e.g. $\left\| \begin{bmatrix} 1 & 3 & 2 \\ 2 & 0 & -1 \end{bmatrix} \right\|_F = 19$

Today:

- Wrap up PCA
  - Derive $\ell = 1$ case: $A_1 = \sigma_1 \vec{u}_1 \vec{v}_1^T$ is best Rank-1 approx. to A.
- Linearization for control
  - Linear approx. to non-linear functions
  - Equilibrium / operating points
  - Examples

See Note 16

PCA concept

Movie recommendation problem (Netflix app.)

	U_1 User 1	U_2 User 2	U_3 User 3	...	U_{1000} User 1000
V_1 Video 1	75	80	20		50
V_2 Video 2	30	20	85		70
⋮				q_{ij}	
V_{100} Video 100	90	95	30		20

Q - matrix

q_{ij} : rating of user j for video i.

- 100 videos (movies) / 1000 users: users who rate (score) the movies.
- Goal: learn the "low-dimensional" structure underlying this "big" data matrix

Say every movie is represented by 4 attributes: (features)

SCORE for VIDEO i : $[a_i \quad b_i \quad c_i \quad d_i]$

- a_i: "action" level
- b_i: "bloodiness" or horror level
- c_i: "comedy" level
- d_i: "drama" level

User j: "sensitivity" or "affinity" for components, i.e. how much user j "likes" or "dislikes" the attributes or "features."

V_j : $[s_{aj} \quad s_{bj} \quad s_{cj} \quad s_{dj}]$

q_{ij} is the inner product between Movie i's feature vector & User j's sensitivity to features vector

$$q_{ij} = s_{aj} \cdot a_i + s_{bj} b_i + s_{cj} \cdot c_i + s_{dj} \cdot d_i$$

e.g. $q_{ij} = 80(0.2) + 0(0.1) + 77(0) + 20(0.7) = \boxed{30}$

$\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{100} \end{bmatrix}$ action scores of all videos $\in \mathbb{R}^{100}$

Similarly, for $\vec{b}$, $\vec{z}$, $\vec{d}$.

$S_a = \begin{bmatrix} s_{a1} \\ s_{a2} \\ \vdots \\ s_{a1000} \end{bmatrix}^T$ "action" sensitivity vector for users $\in \mathbb{R}^{1000}$

Similarly, for $\vec{s_b}$, $\vec{s_c}$, $\vec{s_d}$

Consider

$$\vec{a} \cdot \vec{S_a}^T$$ ("outer product")

$$= \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{100} \end{bmatrix} \begin{bmatrix} s_{a1} & s_{a2} & \cdots & s_{a1000} \end{bmatrix}$$

$$= \begin{bmatrix} a_1 s_{a1} & a_1 s_{a2} & \cdots & a_1 s_{a1000} \\ a_2 s_{a1} & & & \vdots \\ \vdots & & & \\ a_{100} s_{a1} & \cdots & & a_{100} s_{a1000} \end{bmatrix}$$

$$Q = \vec{a} \cdot \vec{s_a}^T + \vec{b} \cdot \vec{s_b}^T + \vec{c} \cdot \vec{s_c}^T + \vec{d} \cdot \vec{s_d}^T$$

$$Q = U \Sigma V^T = \sum_{i=1}^{r} \sigma_i \vec{u_i} \vec{v_i}^T$$

r: rank of Q

The k principal components of Q
- along the columns: $\vec{u_1}, \vec{u_2}, \ldots \vec{u_k}$
- along the rows: $\vec{v_1}, \vec{v_2}, \ldots \vec{v_k}$

$(m=100,\ n=1000)$

$$Q = \begin{bmatrix} | & | & & | \\ \vec{q_1} & \vec{q_2} & \cdots & \vec{q_n} \\ | & | & & | \end{bmatrix} \quad ; \quad Q = \begin{bmatrix} \vec{m_1} \\ \vec{m_2} \\ \vdots \\ \vec{m_m} \end{bmatrix}$$

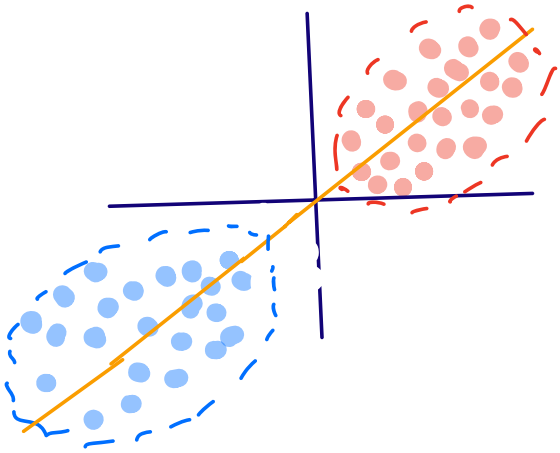
($U_1, U_2, \ldots U_{1000}$ / $V_1, V_2, \ldots V_{100}$)

- We assume data is organized by columns:
  - i.e. each data point is a 100-dim. vector $\vec{q_j}$ denoting U_j's ratings for the 100 videos $V_1, V_2, \ldots V_{100}$

GOAL: Find the first principal component vector (direction) that is most informative about the data set.

The hope is that if we pick the right PC direction, the high-dim. data set will be projected to a much (low-dim.) representation that unveils distinct "clusters" in the dataset!

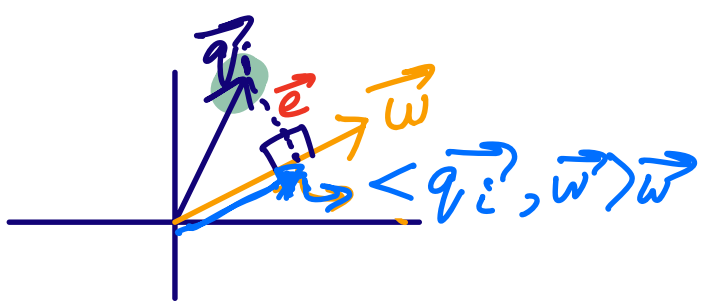
look in this direction



In typical apps, "clusters" will emerge from a low-dim. representation of the data using PCA.

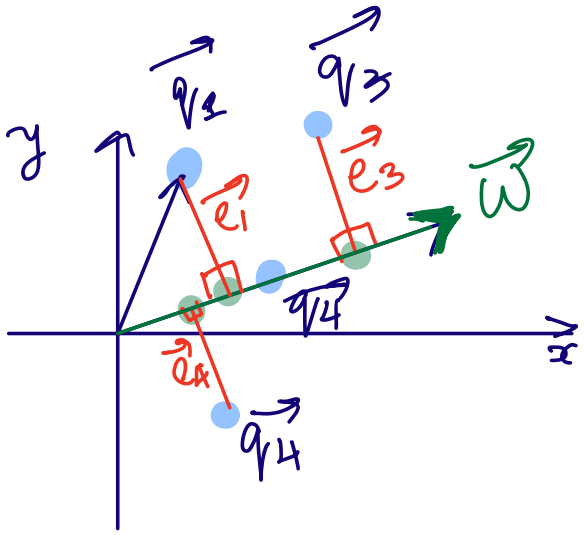
→ We want a $\vec{w}$ such that when $\vec{q}_i$ is projected onto $\vec{w}$, the error is minimized.

Let us normalize: $\|\vec{w}\| = 1$



Projection of $\vec{q_i}$ onto $\vec{w}$: $\langle \vec{q_i}, \vec{w} \rangle \vec{w}$

(since $\|\vec{w}\| = 1$)



Squared-norm of error due to i^{th} data point : $\| \vec{q_i} - \langle \vec{q_i}, \vec{w} \rangle \vec{w} \|^2$

$\vec{e_i}$

Summing error over all points:

$$\min_{\vec{w}} \sum_{i=1}^{n} \| \vec{q_i} - \langle \vec{q_i}, \vec{w} \rangle \vec{w} \|^2$$

$(\|\vec{w}\| = 1)$

$\vec{e_i}$

→ Find the $\vec{w}^*$ that achieves the minimum error.

Can also write as:

$$\vec{w}^* = \underset{\vec{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \| \underbrace{\vec{q}_i - \langle \vec{q}_i, \vec{w} \rangle \vec{w}}_{\vec{e}_i} \|^2$$

$$\text{s.t. } \|\vec{w}\| = 1$$

$$\|\vec{e}_i\|^2 = \left(\vec{q}_i - \langle \vec{q}_i, \vec{w} \rangle \vec{w}\right)^T \left(\vec{q}_i - \langle \vec{q}_i, \vec{w} \rangle \vec{w}\right)$$

$$= \vec{q}_i^T \vec{q}_i - \langle \vec{q}_i, \vec{w} \rangle \vec{w}^T \vec{q}_i - \langle \vec{q}_i, \vec{w} \rangle \vec{q}_i^T \vec{w} + \langle \vec{q}_i, \vec{w} \rangle^2 \underbrace{\vec{w}^T \vec{w}}_{\|\vec{w}\|^2 = 1}$$

$$= \|\vec{q}_i\|^2 - 2\langle \vec{q}_i, \vec{w} \rangle^2 + \langle \vec{q}_i, \vec{w} \rangle^2$$

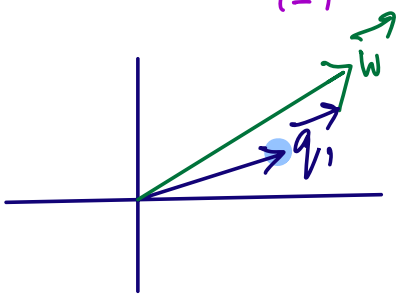
$$= \|\vec{q}_i\|^2 - \langle \vec{q}_i, \vec{w} \rangle^2$$

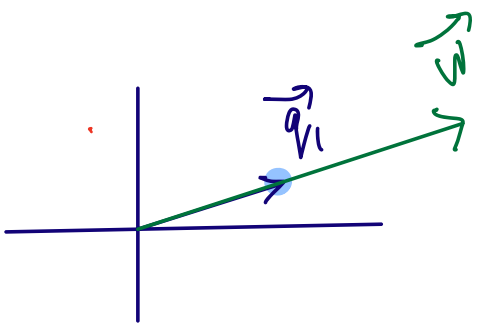
$$\|\vec{e}_i\|^2 = \|\vec{q}_i\|^2 - \langle \vec{q}_i, \vec{w} \rangle^2$$

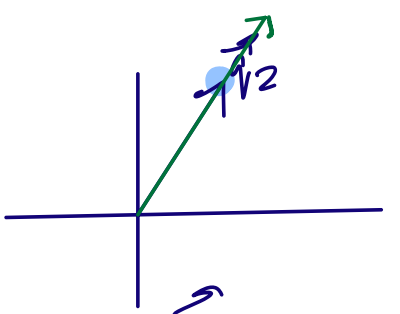
$$\vec{w}^* = \underset{\vec{w}}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \underbrace{\|\vec{q}_i\|^2}_{\text{does not depend on } \vec{w}} - \sum_{i=1}^{n} \langle \vec{q}_i, \vec{w} \rangle^2 \right)$$

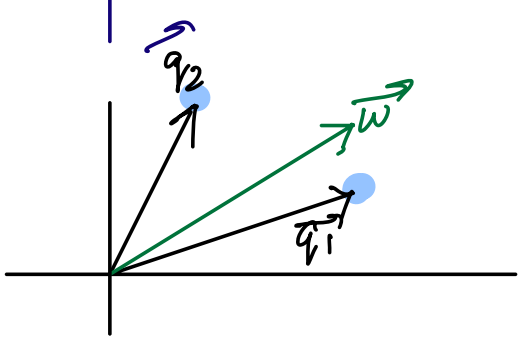
$$\vec{w}^* = \underset{\vec{w}}{\operatorname{argmin}} -\sum_{i=1}^{n} \langle \vec{q}_i, \vec{w} \rangle = \underset{\vec{w}}{\operatorname{avg\,max}} \sum_{i=1}^{n} \langle \vec{q}_i, \vec{w} \rangle$$
$$\text{s.t. } \|\vec{w}\| = 1 \qquad \text{s.t. } \|\vec{w}\| = 1$$

Choose a $\vec{w}$ s.t. $\sum_{i=1}^{n} \langle \vec{q}_i, \vec{w} \rangle^2$ is largest.

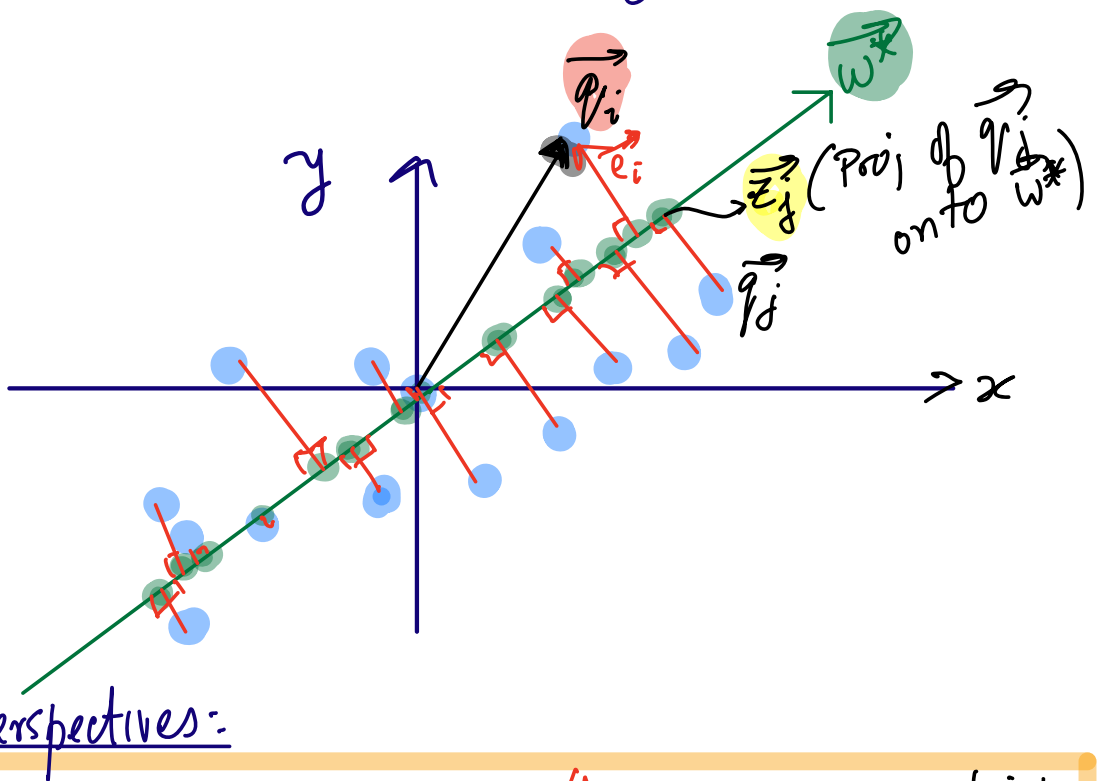








Projection of data-points onto $\vec{w}$



2 perspectives:

- $\vec{w}^*$ minimizes the sum of projection errors of the data-points

$$\vec{w}^* = \underset{\vec{w}\ (\|\vec{w}\|=1)}{\operatorname{argmax}} \sum_{i=1}^{n} \langle \vec{q}_i, \vec{w} \rangle^2$$

- $\vec{w}^*$ maximizes the "spread" of the data as measured by the sum of the squared magnitudes of the projection of the data pts

(Assumes the data-pts. are "mean-removed")

(a.k.a. the direction of maximum variance → a probability concept (out of scope for 16B)

$$\left(\sum_{j=1}^{n} \|\vec{z}_j\|^2 \text{ is maximum} \right)$$

$$\vec{w}^* = \underset{\vec{w}}{\operatorname{argmax}} \sum_{i=1}^{n} \langle \vec{w}_i, \vec{q} \rangle \langle \vec{q}, \vec{w}_i \rangle \qquad \langle \vec{w}_i, \vec{q} \rangle^2$$

$$(\text{s.t. } \|\vec{w}\| = 1)$$

$$= \underset{\vec{w}}{\operatorname{argmax}} \sum_{i=1}^{n} \vec{w}^T \vec{q}_i \vec{q}_i^T \vec{w}$$

outer product

$$\vec{w}^* = \underset{\vec{w}}{\operatorname{argmax}} \left[\vec{w}^T Q Q^T \vec{w} \right]$$

$$S = U \Lambda U^T$$

Recall: $\sum \vec{u}_i \vec{v}_i^T = U V^T$

(where $\Lambda = \operatorname{diag}(\sigma_1^2, \sigma_2^2, \ldots \sigma_m^2)$ and U is an $(m \times m)$ orthonormal matrix that is the same U in $Q = U \Sigma V^T$)

$$\vec{w}^* = \underset{\vec{w}}{\operatorname{argmax}} \left[\vec{w}^T S \vec{w} \right]$$

$\|\vec{w}\| = 1$

[S has e-values σ_i^2 and e-vectors $\vec{u}_i$: $S\vec{u}_i = \sigma_i^2 \vec{u}_i$]

Can be shown that $\vec{w}^* = \underset{\|\vec{w}\|=1}{\operatorname{argmax}} \vec{w}^T S \vec{w}$ is $\vec{w} = \vec{u}_1$, where $\vec{u}_1$ is the maximum eigenvector of $S = QQ^T$ corresponding to $\lambda_1 = \sigma_1^2$, where the e-vals of S are $\sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_m^2$.

"Proof":

Write $\vec{w} \in \mathbb{R}^m$ in eigenbasis $\{\vec{u}_i\}$ co-ordinate system:

$$\vec{w} = \alpha_1 \vec{u}_1 + \alpha_2 \vec{u}_2 + \cdots + \alpha_m \vec{u}_m = \sum_{i=1}^{m} \alpha_i \vec{u}_i$$

$$1 = \|\vec{w}\|^2 = \vec{w}^T\vec{w} = \left(\sum_{i=1}^{m} \alpha_i \vec{u}_i^T\right)\left(\sum_{j=1}^{m} \alpha_j \vec{u}_j\right)$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j \vec{u}_i^T\vec{u}_j \quad \vec{u}_i^T\vec{u}_j = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{else} \end{cases}$$

$$\Rightarrow \boxed{\sum_{i=1}^{m} \alpha_i^2 = 1} \Rightarrow \alpha_1^2 + \alpha_2^2 + \dots + \alpha_m^2 = 1$$

$$\begin{aligned} S\vec{u}_1 &= \lambda_1\vec{u}_1 \\ S\vec{u}_2 &= \lambda_2\vec{u}_2 \\ &\vdots \\ S\vec{u}_m &= \lambda_m\vec{u}_m \end{aligned}$$

eigenvalue equations for S

$(\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m)$

So, we want to find the $\vec{w}$ that maximizes $\vec{w}^T S\vec{w}$ s.t. $\vec{w} = \sum_{i=1}^{m} \alpha_i \vec{u}_i$

where $\alpha_1^2 + \alpha_2^2 + \dots + \alpha_m^2 = 1$

- If $\vec{w} = \vec{u}_1$, $\alpha_1 = 1, \alpha_2 = \alpha_3 = \dots = \alpha_m = 0$,

$$\vec{w}^T S\vec{w} = \vec{u}_1^T S\vec{u}_1 = \vec{u}_1^T \lambda_1\vec{u}_1 = \lambda_1 \underbrace{\|\vec{u}_1\|^2}_{1}$$

- If $\vec{w} = \vec{u}_2$, $\alpha_1 = 0, \alpha_2 = 1, \alpha_3 = \dots = \alpha_m = 0$

$$\vec{w}^T S\vec{w} = \vec{u}_2^T S\vec{u}_2 = \lambda_2$$

Similarly, If $\vec{w} = \vec{u}_i$, $\vec{w}^T S\vec{w} = \vec{u}_i^T S\vec{u}_i = \lambda_i$

We need to pick a weighted-combination of the $\vec{u}_i$'s (since $\vec{w} = \sum_{i=1}^{m} \alpha_i \vec{u}_i$) where $\alpha_1^2 + \alpha_2^2 + \dots + \alpha_m^2 = 1$.

Q/ What is the optimal choice for the α's?

A) $\vec{\alpha} = (\alpha_1, \alpha_2, \dots \alpha_m) = (1, 0, 0 \dots 0)$

$$\Rightarrow \vec{w}^* = \underset{\|\vec{w}\|=1}{\operatorname{argmin}} \; \vec{w}^T QQ^T\vec{w} = \vec{u}_1$$

□

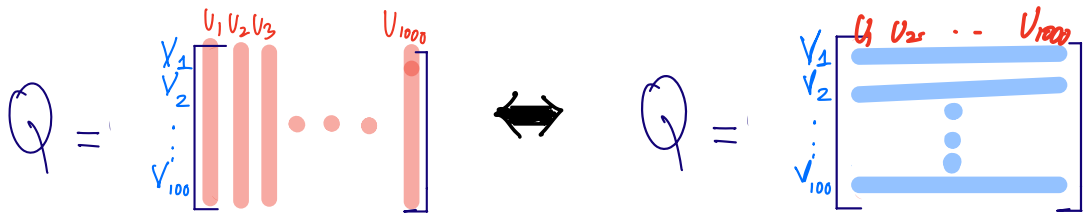
$$\underset{\vec{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \| \vec{q}_i - \langle \vec{q}_i, \vec{w} \rangle \vec{w} \|^2 = \vec{u}_1$$

Similarly, 2nd most important component? $\vec{u}_2$

More generally, if we want the l-dimensional low-rank approx. to Q, the solution will turn out to be $Q_l = \sum_{i=1}^{l} \sigma_i \vec{u}_i \vec{v}_i^T$

(Eckart-Young theorem)

If the data points of interest are organized by rows instead of by columns



$$\underset{\vec{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \| \vec{v}_i - \langle \vec{v}_i, \vec{w} \rangle \vec{w} \|^2 = \vec{v}_1 \quad (\|\vec{w}\| = 1)$$

$\vec{v}_i$: Row i of both terms

$\vec{v}_1$: first right singular vector

- $\vec{u}_i$'s are principal components along columns (U_i's)
- $\vec{v}_i^T$'s " " " " rows (V_i^T's)

SUMMARY: Given data pts. $\{\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n\}$, find the principal components of this data.

1) Arrange the data into a matrix:

$$X = \begin{bmatrix} | & | & & | \\ \vec{x}_1 & \vec{x}_2 & \cdots & \vec{x}_n \\ | & | & & | \end{bmatrix}$$

→ "Mean-removal": depending on application, "center" each column around 0 by subtracting the column means from each column

2) Compute $X = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i \vec{u}_i \vec{v}_i^T$

3) For k principal components along the columns, choose $\vec{u}_1, \vec{u}_2, \ldots \vec{u}_k$.

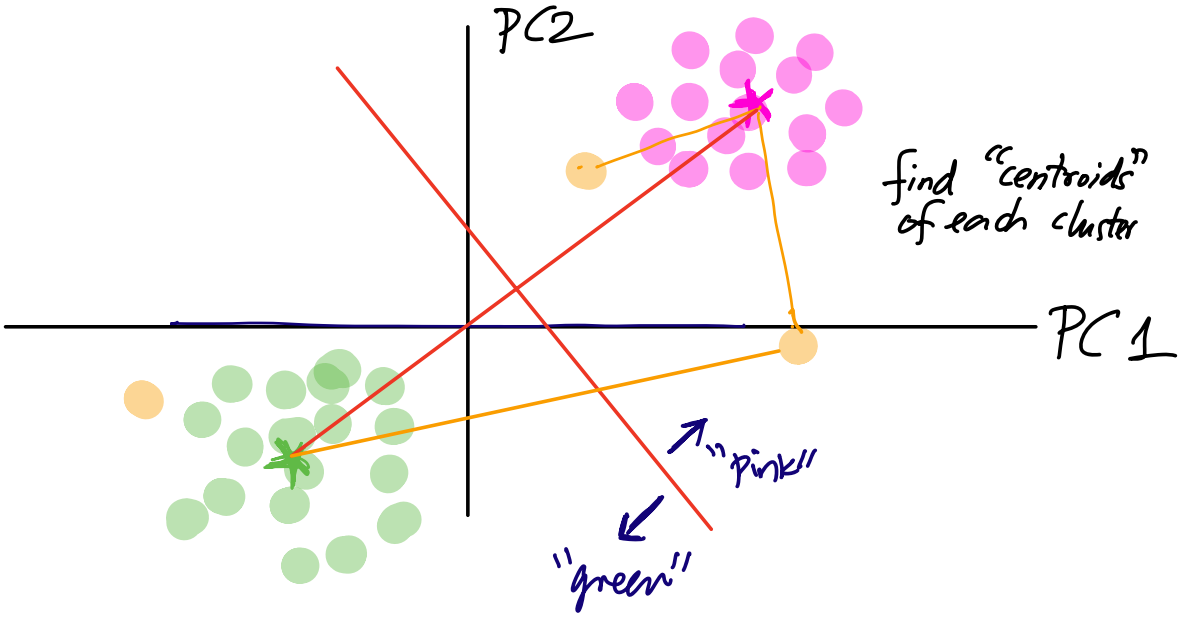
For k principal components along the rows, choose $\vec{v}_1, \vec{v}_2, \ldots \vec{v}_k$.

4) (OPTIONAL) Project data onto $\vec{u}_1, \vec{u}_2 \ldots \vec{u}_k$ to get low-dimensional structure

use this to perform clustering/classification

Classification:

After projecting data onto top 2 Principal Components (200-dim. dataset)



Linearization for control

So far, in this course, we have focused on LINEAR control systems, where in discrete-time,

$$\vec{x}[k+1] = A\,\vec{x}[k] + B\,\vec{u}[k]$$

or in continuous-time,

$$\frac{d\vec{x}(t)}{dt} = A\,\vec{x}(t) + B\,\vec{u}(t)$$

RECAP First, what is a linear system? (Holds for CT and DT systems)

If $x_1(t) \to \mathcal{L}\{\cdot\} \to y_1(t)$

and $x_2(t) \to \mathcal{L}\{\cdot\} \to y_2(t)$

then $\alpha x_1(t) \to \mathcal{L}\{\cdot\} \to \alpha y_1(t)$

and $x_1(t) + x_2(t) \to \mathcal{L}\{\cdot\} \to y_1(t) + y_2(t)$

$$(\alpha x_1[k] + \beta x_2[k]) \to \mathcal{L}\{\cdot\} \to (\alpha y_1[k] + \beta y_2[k])$$

$$\alpha x_1(t) + \beta x_2(t) \to \mathcal{L}\{\cdot\} \to \alpha y_1(t) + \beta y_2(t)$$

Ex.: (i) $\vec{y} = A\vec{x}$ (matrix multiplication)

is a linear system.

Why? If $\mathcal{L}\{\cdot\} = A$, and if $A\vec{x}_1 = \vec{y}_1$; $A\vec{x}_2 = \vec{y}_2$

then, $A[\alpha\vec{x}_1 + \beta\vec{x}_2] = \alpha \underbrace{A\vec{x}_1}_{\vec{y}_1} + \beta \underbrace{A\vec{x}_2}_{\vec{y}_2}$. ✓

(ii) $y(t) = \dfrac{dx(t)}{dt}$ (differential equation)

$y_1(t) = \dfrac{dx_1(t)}{dt}$; $y_2(t) = \dfrac{dx_2(t)}{dt}$

$$\frac{d}{dt}[\alpha x_1(t) + \beta x_2(t)] = \alpha \frac{dx_1(t)}{dt} + \beta \frac{dx_2(t)}{dt}$$

$$= \alpha y_1(t) + \beta y_2(t)$$

✓

However, in the real world, many systems exhibit NON-LINEAR behavior!

Ex.: — Transistors: not just "on-off" switches but work "continuously" with highly non-linear governing equations.

— Robotics & control: can have highly non-linear dynamics; e.g. applying feedback control in order to stabilize an inverted pendulum on a cart.

— Machine learning: can have highly non-linear systems. E.g. gradient descent can result in non-linear trajectories.

Generally, a nonlinear control system is characterized by:

Discrete-time:

$$\underbrace{\vec{x}[k+1]}_{\text{state vector}} = \vec{f}\left(\vec{x}[k], \underbrace{\vec{u}[k]}_{\text{input control vector}}\right)$$

$\vec{f}$: an arbitrary vector function

Continuous time:

$$\frac{d\vec{x}(t)}{dt} = \vec{f}\left(\vec{x}(t), \vec{u}(t)\right)$$

Key concept: How to deal w/ NL systems by approx. them locally using linear models by a process called LINEARIZATION of the function $\vec{f}$.

Ex: Linearization.

$$f(x) = x^2$$
$$f(x,u) = x^2 u^3$$

$$\frac{d\vec{x}(t)}{dt} = f(\vec{x},\vec{u}) = \underbrace{A\vec{x} + B\vec{u}}_{\text{Linear formulation}}$$

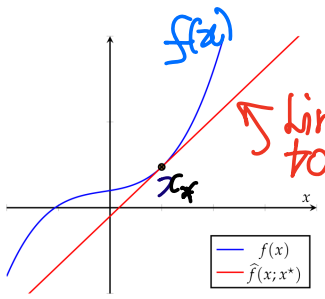
Scalar functions (One variable)

$$f(x) = f(x_*) + \text{stuff}$$

$f(x)$: general

$f(x_*)$: specific "operating" point

stuff: depend on how close x, x_* are



↖ Linear approx. to $f(x)$ about $x = x_*$

How does $f(x)$ change in the neighbourhood of x^*?

Derivative of $f(x)$ @ x_* $= \left.\frac{df}{dx}\right|_{x=x_*}$

Taylor Expansion

$$f(x) = \underbrace{f(x_*)}_{\text{constant}} + \overbrace{\left.\frac{df}{dx}\right|_{x=x_*} (x-x_*)}^{\text{linear}}$$

$$+ \left.\frac{d^2f}{dx^2}\right|_{x=x_*} \frac{(x-x_*)^2}{2} \quad \} \text{ quadratic}$$

$$+ \left.\frac{d^3f}{dx^3}\right|_{x=x_*} \frac{(x-x_*)^3}{6} \quad \} \text{ cubic}$$

$$+ \ldots\ldots$$

SCALAR FUNCTION CASE:

Linear approximation

$$f(x) \approx f(x_*) + f'(x_*)(x-x_*)$$

($\approx$: approx. equal)