Berkeley
MSSE
MOLECULAR SCIENCE & SOFTWARE ENGINEERING

Lecture 9:

Eigenvalues and Eigenvectors

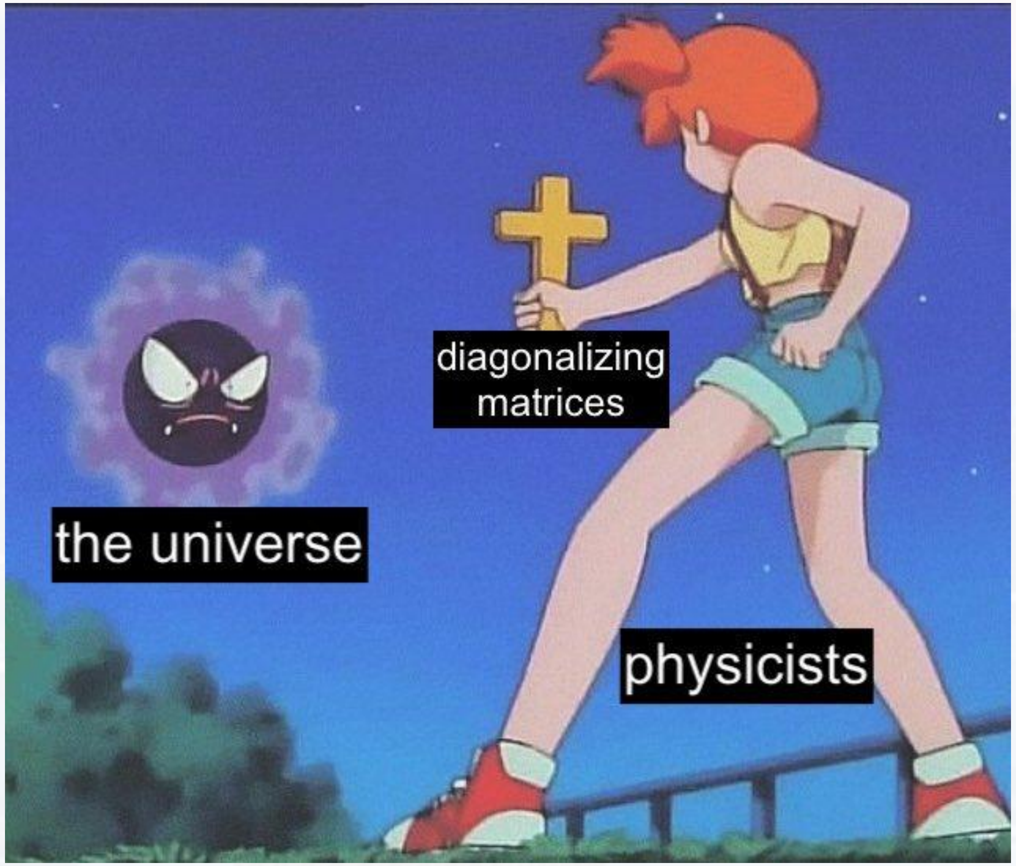
Markus Hohle

University California, Berkeley

Numerical Methods for Computational Science

Course Map

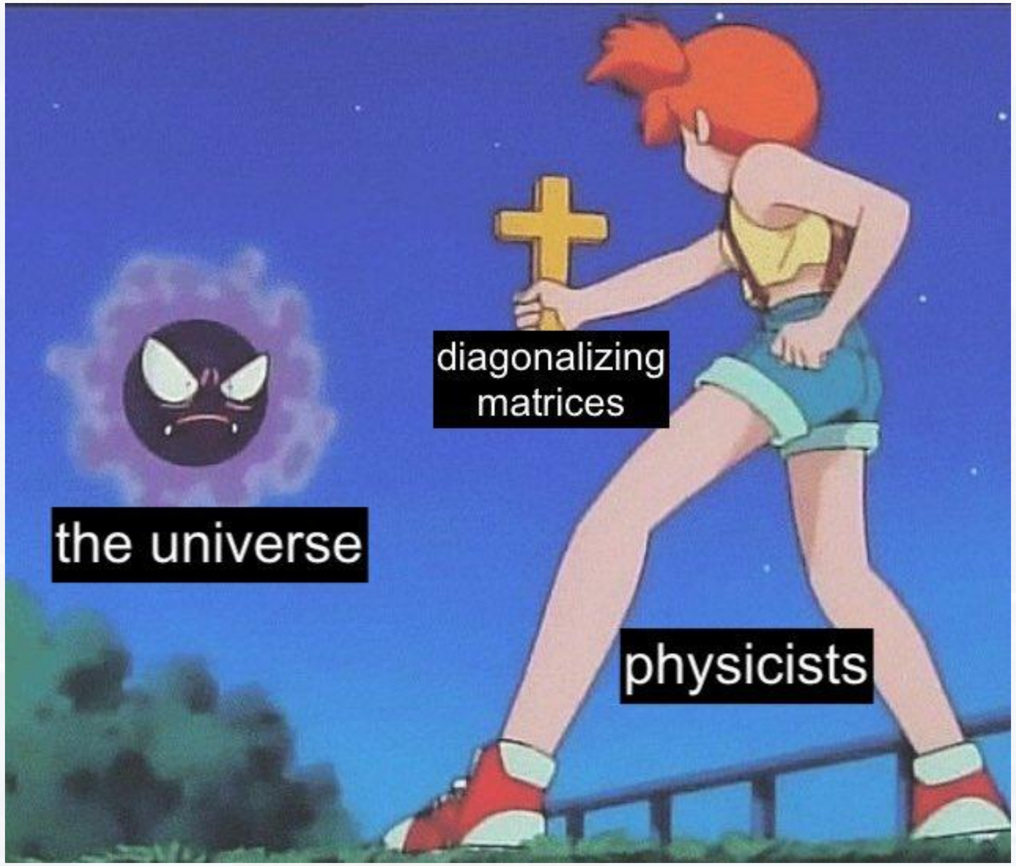
Week 1:	Introduction to Scientific Computing and Python Libraries
Week 2:	Linear Algebra Fundamentals
Week 3:	Vector Calculus
Week 4:	Numerical Differentiation and Integration
Week 5:	Solving Nonlinear Equations
Week 6:	Probability Theory Basics
Week 7:	Random Variables and Distributions
Week 8:	Statistics for Data Science
Week 9:	**Eigenvalues and Eigenvectors**
Week 10:	Simulation and Monte Carlo Method
Week 11:	Data Fitting and Regression
Week 12:	Optimization Techniques
Week 13:	Machine Learning Fundamentals



angryfermion

Outline

- A Geometrical Approach
- Finding Eigenvectors and Eigenvalues
- PCA
- Example I
- Example II



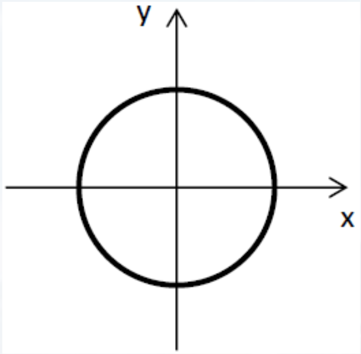
angryfermion

Outline

- **A Geometrical Approach**
- Finding Eigenvectors and Eigenvalues
- PCA
- Example I
- Example II

about quadratic forms

circle



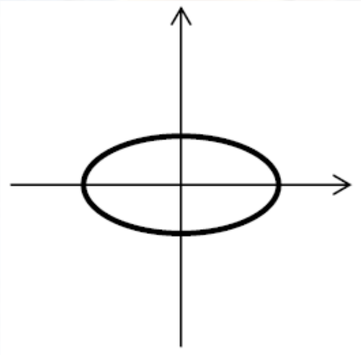
"distance to a reference point is constant"

$$x^2 + y^2 = const = r^2$$

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = const$$

$$a = b \rightarrow x^2 + y^2 = r^2$$

ellipse



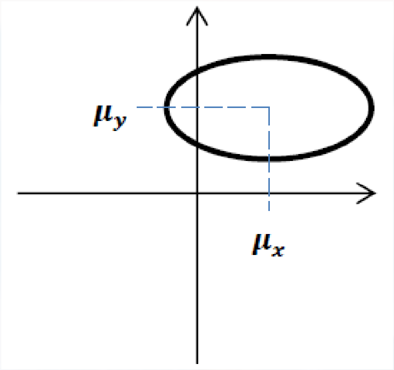
"stretching the coordinate system"

$$a \neq b$$

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = const$$

about quadratic forms

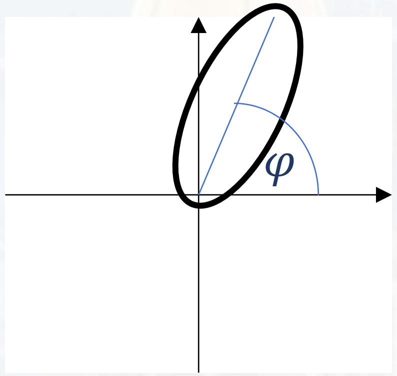
ellipse



"moving the center of the ellipse"

$$\frac{(x-\mu_x)^2}{a^2}+\frac{\left(y-\mu_y\right)^2}{b^2}=const$$

$$x_0=0, x_0\rightarrow\mu_x$$
$$y_0=0, y_0\rightarrow\mu_y$$



"turning the ellipse by an angle φ"

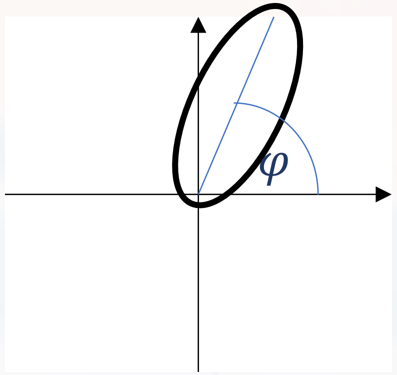
$$\varphi=\frac{1}{2}atan\left(\frac{c}{\frac{1}{a^2}-\frac{1}{b^2}}\right)$$

$$\frac{(x-\mu_x)^2}{a^2}+\frac{\left(y-\mu_y\right)^2}{b^2}+c(x-\mu_x)\left(y-\mu_y\right)=const$$

turning the form

about quadratic forms

$$\frac{(x-\mu_x)^2}{a^2}+\frac{\left(y-\mu_y\right)^2}{b^2}+c(x-\mu_x)\left(y-\mu_y\right)=const$$



matrix form:

$$const=\begin{pmatrix}x-\mu_x\\ y-\mu_y\end{pmatrix}^T\begin{pmatrix}1/a^2 & c/2\\ c/2 & 1/b^2\end{pmatrix}\begin{pmatrix}x-\mu_x\\ y-\mu_y\end{pmatrix}$$

often:

$$const=\begin{pmatrix}x-\mu_x\\ y-\mu_y\end{pmatrix}^T\begin{pmatrix}A & C/2\\ C/2 & B\end{pmatrix}\begin{pmatrix}x-\mu_x\\ y-\mu_y\end{pmatrix}$$

more general:

$$const=\begin{pmatrix}x-\mu_x\\ y-\mu_y\\ 1\end{pmatrix}^T\begin{pmatrix}A & C/2 & D/2\\ C/2 & B & E/2\\ D/2 & E/2 & F\end{pmatrix}\begin{pmatrix}x-\mu_x\\ y-\mu_y\\ 1\end{pmatrix}$$

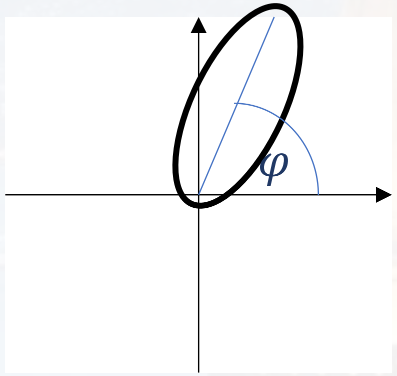
depending on A, B, C, D, E, F

- **circle**
- **ellipse**
- **parabola**
- **hyperbola**

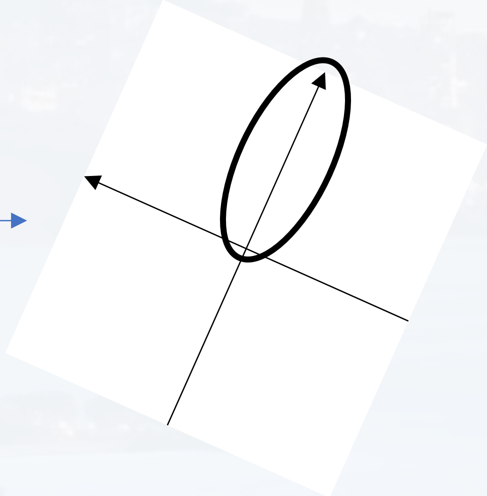
about quadratic forms

$$\frac{(x-\mu_x)^2}{a^2} + \frac{(y-\mu_y)^2}{b^2} + \boxed{c(x-\mu_x)(y-\mu_y)} = \begin{pmatrix} x-\mu_x \\ y-\mu_y \end{pmatrix}^T \begin{pmatrix} 1/a^2 & \boxed{c/2} \\ \boxed{c/2} & 1/b^2 \end{pmatrix} \begin{pmatrix} x-\mu_x \\ y-\mu_y \end{pmatrix} = const$$

turning the form



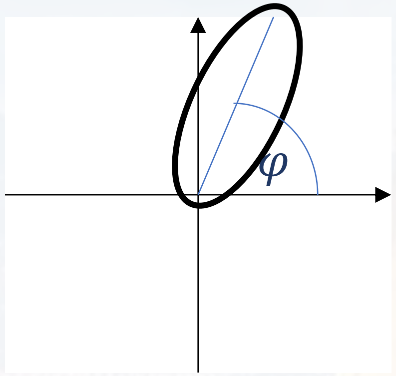
turning the coordinate system



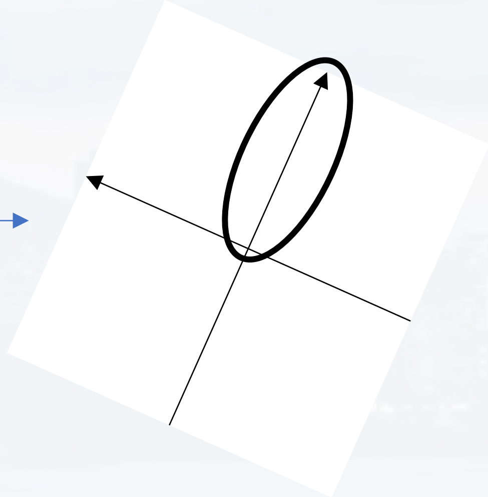
$$\frac{(x_{new}-\mu_{x(new)})^2}{a_{new}^2} + \frac{(y_{new}-\mu_{y(new)})^2}{b_{new}^2} = \begin{pmatrix} x_{new}-\mu_{x(new)} \\ y_{new}-\mu_{y(new)} \end{pmatrix}^T \boxed{\begin{pmatrix} 1/a_{new}^2 & 0 \\ 0 & 1/b_{new}^2 \end{pmatrix}} \begin{pmatrix} x_{new}-\mu_{x(new)} \\ y_{new}-\mu_{y(new)} \end{pmatrix} = const$$

about quadratic forms

non – diagonal elements: - turn/shear the object
diagonal elements: - stretches (or flips, if negative) the object



turning the coordinate system



not turned/sheared →principal axes of the object are **parallel to the coordinate** axes

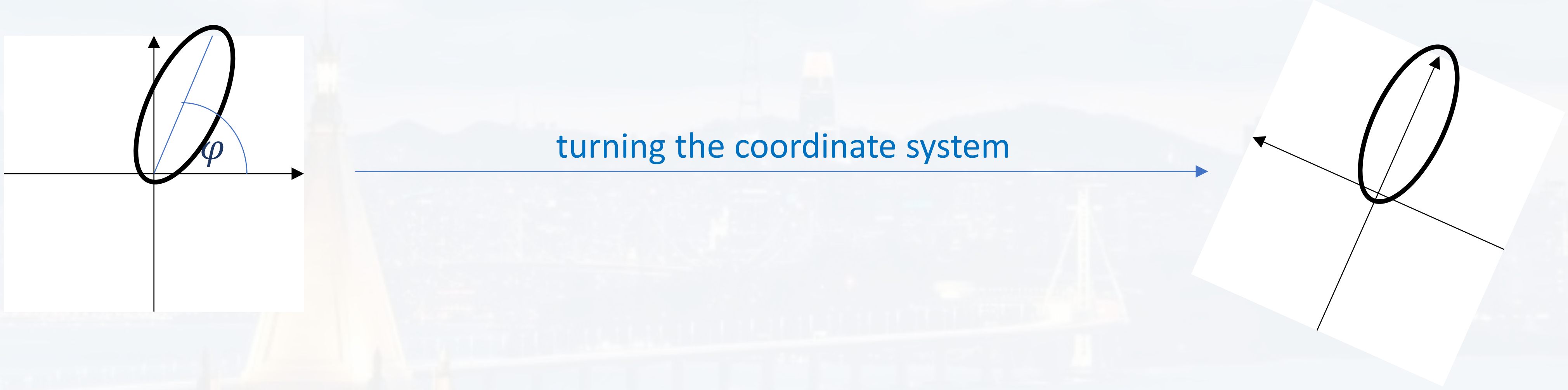
new coord. axis are called: **eigenvectors** $\vec{v}$ ("eigen", German for "proper")
→ they span the proper coordinate system!

$$\begin{pmatrix} \lambda_1 & 0 \ldots & 0 \\ 0 & \lambda_i & 0 \\ 0 & 0 & \lambda_N \end{pmatrix}$$

in the proper coordinate system: matrix is diagonal (entries are called **"eigenvalues" λ**)

about quadratic forms

non – diagonal elements: - turn/shear the object
diagonal elements: - stretches (or flips, if negative) the object

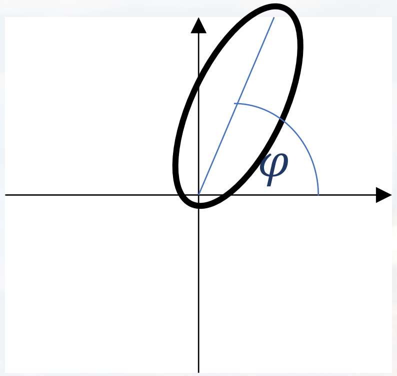


In a coordinate system in which the principal axes are **parallel to the coordinate** axes

- the matrix that defines the form is **diagonal**
- the **entries** of the now diagonal matrix are called **eigenvalues λ**
- the **axes** of this coordinates system are called **eigenvectors $\vec{v}$**
- eigen means **"proper"**, i. e. it is the **"most suitable"** coordinate system

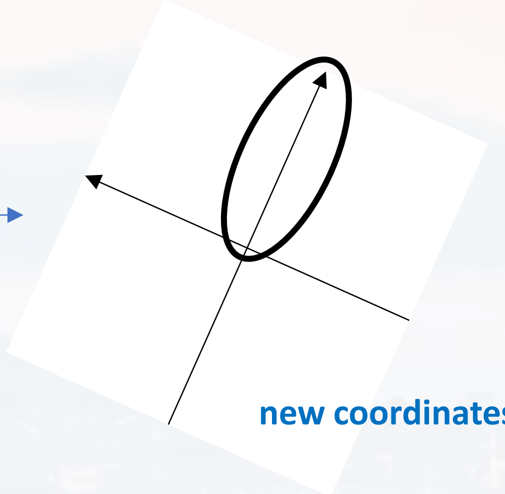
$$\begin{pmatrix} \lambda_1 & 0 \ldots & 0 \\ 0 & \lambda_i & 0 \\ 0 & 0 & \lambda_N \end{pmatrix}$$

about quadratic forms



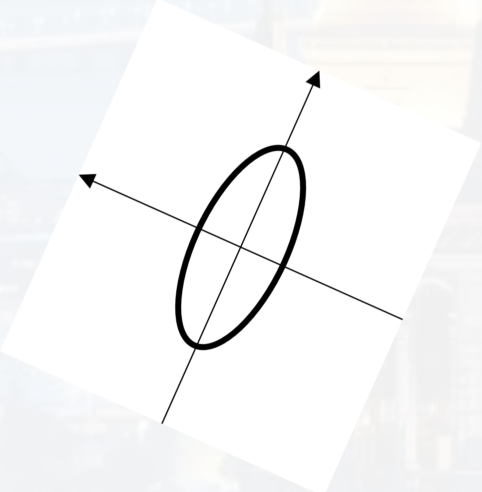
old coordinates

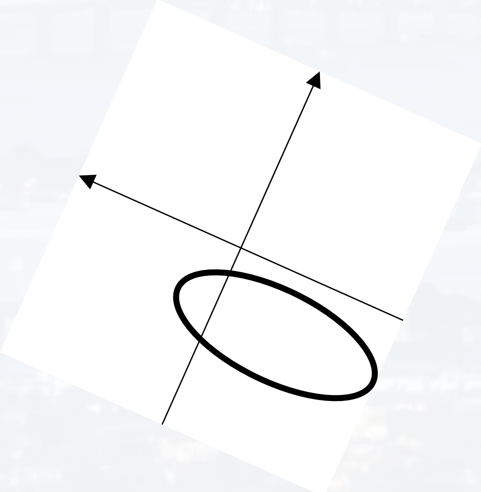
turning the coordinate system

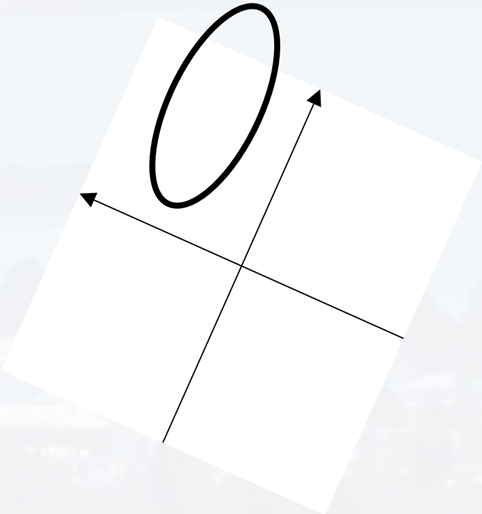


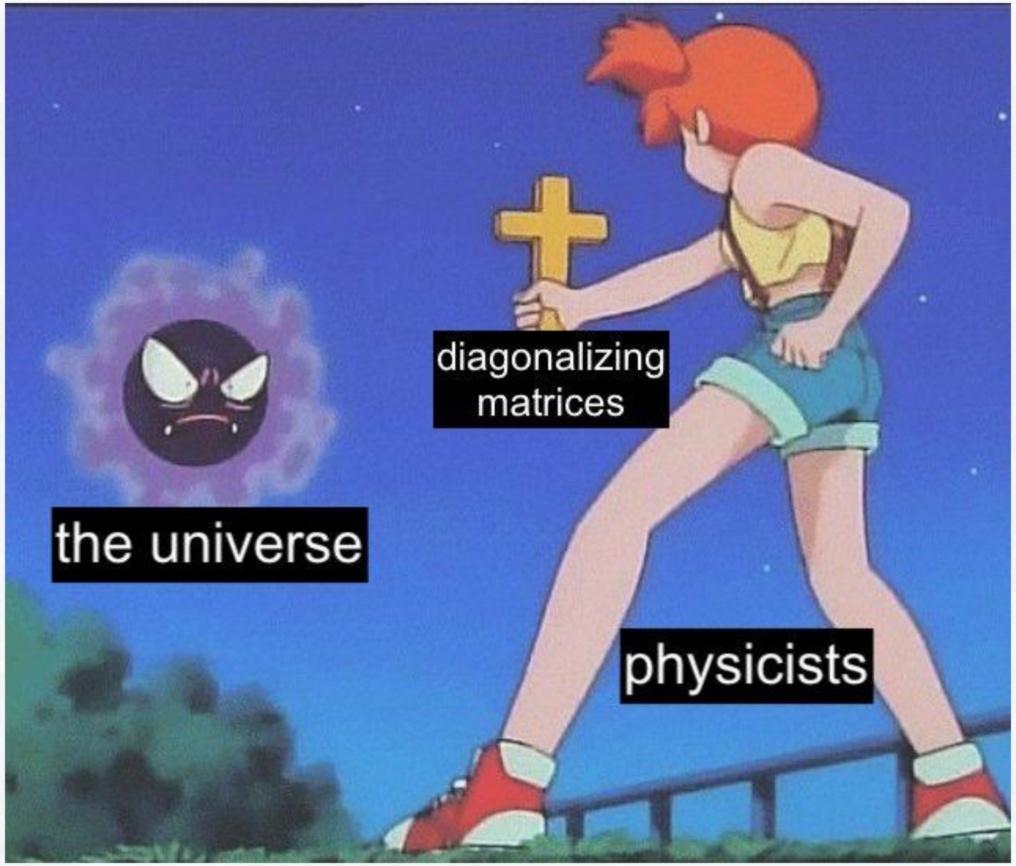
new coordinates

not turned/sheared → principal axes of the object are **parallel to the coordinate** axis









angryfermion

Outline

- A Geometrical Approach

- Finding Eigenvectors and Eigenvalues

- PCA

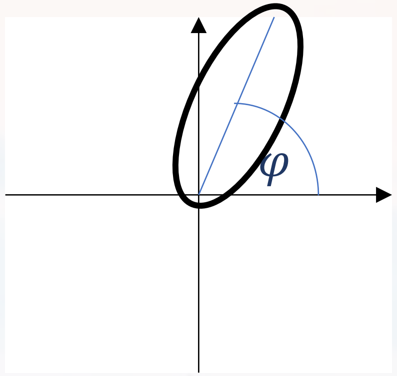
- Example I

- Example II

Eigenvalues and Eigenvectors:

1) turned ellipse

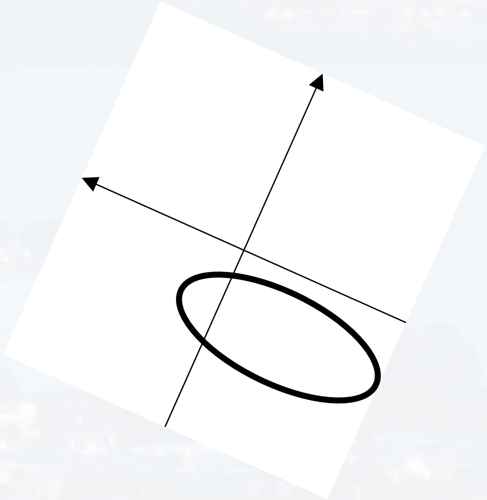
$$\begin{pmatrix} x \\ y \end{pmatrix}^T \underbrace{\begin{pmatrix} A & C/2 \\ C/2 & B \end{pmatrix}}_{M} \begin{pmatrix} x \\ y \end{pmatrix} = A\,x^2 + B\,y^2 + C\,xy$$



2) turning the coordinate system, such that principal axes of the are **parallel to the coordinate**

$$\begin{pmatrix} x_{new} \\ y_{new} \end{pmatrix}^T \underbrace{\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}}_{M_{new}} \begin{pmatrix} x_{new} \\ y_{new} \end{pmatrix} = \lambda_1 {x_{new}}^2 + \lambda_2 {y_{new}}^2$$

eigenvalues λ



How to turn M into M_{new}?

Eigenvalues and Eigenvectors:

How to turn *M* into M_{new}?

we assume, we have a set of eigenvectors $\vec{v}_i$

transforming *M* with *B* = ($\vec{v}_1, \vec{v}_2, \ldots \vec{v}_N$) should turn *M* into a **diagonal matrix** M_{new}

$$M_{new} = B^T M B$$

after some algebra: $M\vec{v}_i = \lambda_i \vec{v}_i$ which can be solved with: $\det(M - \lambda_i I) = 0$

$$\boxed{M\vec{v}_i = \lambda_i \vec{v}_i}$$

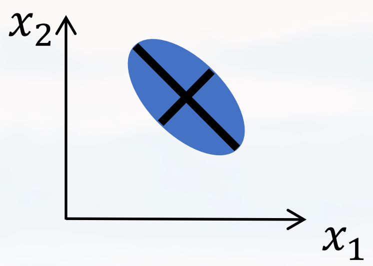
characteristic equation

Eigenvalues and Eigenvectors:

simple example:

$$\boxed{det(M - \lambda I) = 0}$$

$$x_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad x_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad M = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$



$$\det(M - \lambda_i I) = 0 = det\left[\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} - \begin{pmatrix} \lambda_i & 0 \\ 0 & \lambda_i \end{pmatrix}\right] = det\left[\begin{pmatrix} 2-\lambda_i & -1 \\ -1 & 2-\lambda_i \end{pmatrix}\right]$$

$$= \boxed{3 - 4\lambda_i + \lambda_i{}^2 = 0}$$

characteristic polynomial

N eigenvalues and N eigenvectors for N coordinates

$$\boxed{\lambda_1 = 1 \qquad \lambda_2 = 3}$$

$\lambda_1 = 1 \qquad \lambda_2 = 3$

$det(M - \lambda I) = 0$

calculating the **eigenvectors** $\vec{v}_i$:

$$M\vec{v}_i = \lambda_i \vec{v}_i$$

characteristic equation

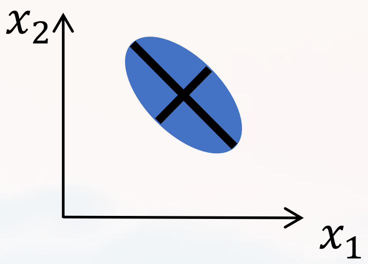
$(M - \lambda_i I)\vec{v}_i = 0$

for λ_1 $\left[\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right]\begin{pmatrix} v_{1x} \\ v_{1y} \end{pmatrix} = \begin{pmatrix} v_{1x} & -v_{1y} \\ -v_{1x} & v_{1y} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $v_{1x} = v_{1y}$ e.g. $\vec{v}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

for λ_2 $\left[\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} - \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}\right]\begin{pmatrix} v_{2x} \\ v_{2y} \end{pmatrix} = \begin{pmatrix} -v_{2x} & -v_{2y} \\ -v_{2x} & -v_{2y} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $v_{2x} = -v_{2y}$ e.g. $\vec{v}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

Eigenvalues and Eigenvectors:

$x_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ $x_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ $M = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$ $\boxed{\lambda_1 = 1 \quad \lambda_2 = 3}$



$\boxed{\vec{v_1} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}}$ $\boxed{\vec{v_2} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}}$

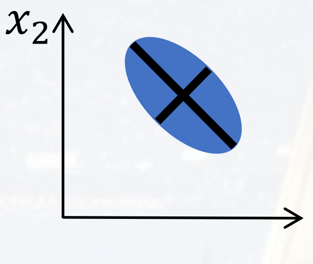
recall: $B = (\vec{v_1}\ \vec{v_2})$ and $M_{new} = B^T M B$

$\lambda_1 = 1$

M in the new coordinates is $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^T \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = 2 \boxed{\begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}}$ $\lambda_2 = 3$

$x_1 = \begin{pmatrix}1\\0\end{pmatrix} \quad x_2 = \begin{pmatrix}0\\1\end{pmatrix} \qquad M = \begin{pmatrix}2 & -1\\-1 & 2\end{pmatrix} \qquad \boxed{\lambda_1 = 1 \qquad \lambda_2 = 3} \quad \boxed{\vec{v_2} = \begin{pmatrix}1\\-1\end{pmatrix}} \quad \boxed{\vec{v_1} = \begin{pmatrix}1\\1\end{pmatrix}}$

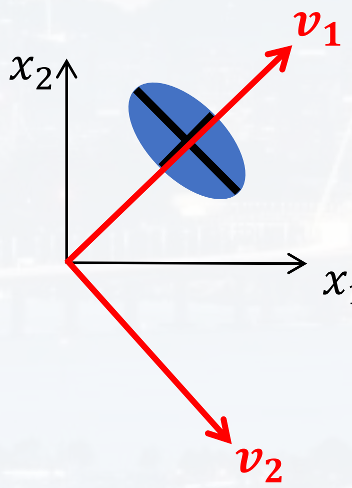
the old coordinates



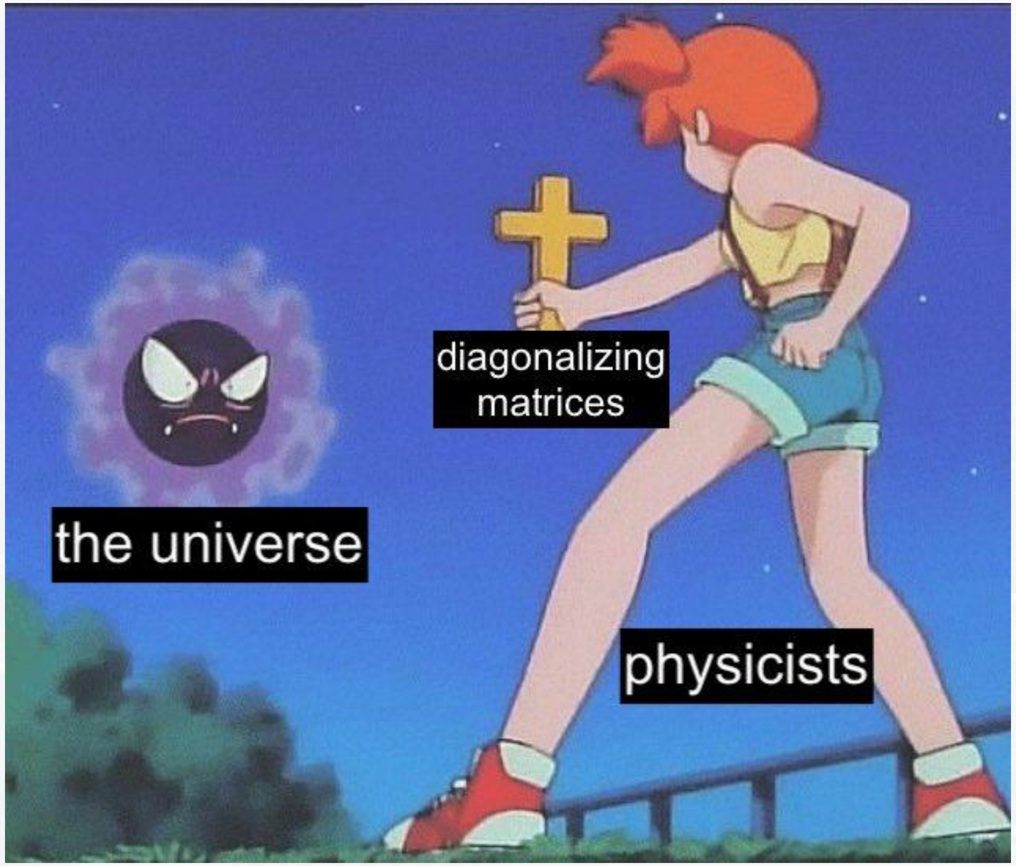
$M = \begin{pmatrix}2 & -1\\-1 & 2\end{pmatrix}$

$x_2 = \begin{pmatrix}0\\1\end{pmatrix} \quad x_1 = \begin{pmatrix}1\\0\end{pmatrix}$

the new coordinates



$M_{new} = 2\begin{pmatrix}1 & 0\\0 & 3\end{pmatrix}$



angryfermion

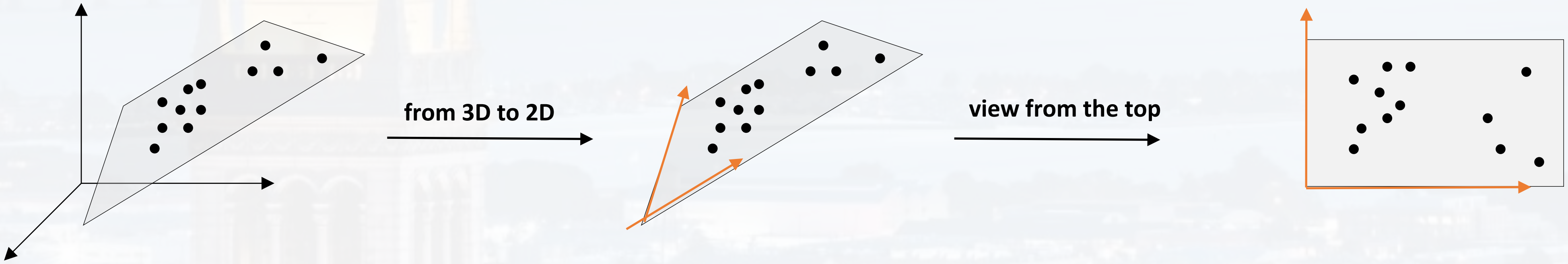
Outline

- A Geometrical Approach
- Finding Eigenvectors and Eigenvalues
- **PCA**
- Example I
- Example II

Goals:

- **Dimension reduction**!
- Reducing the complexity of the dataset **without loosing** information
- Removing redundancies
- Reducing the number of features

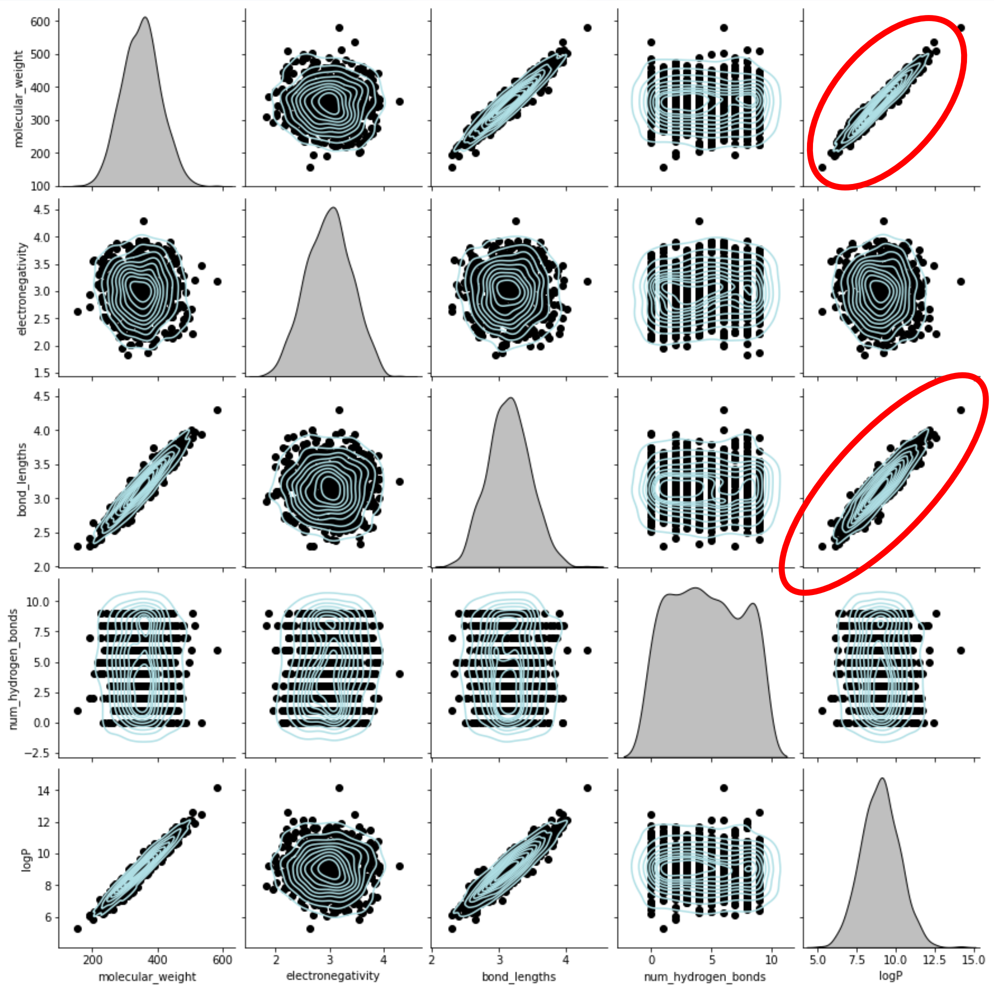
→ trick: using **correlation** between different features



Eigenvalues and Eigenvectors:

Lecture 8: correlation

label	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP
Toxic	382.602	2.00269	3.61153	3	9.82666
Toxic	408.961	2.93626	3.47904	6	9.85889
Non-Toxic	239.548	2.71413	2.63922	8	6.75962
Non-Toxic	315.58	2.85598	2.86034	9	8.70674
Non-Toxic	282.521	2.83877	2.9664	1	7.8173



$$corr(x,y) = \frac{cov(x,y)}{\sqrt{var(x)var(y)}}$$

Eigenvalues and Eigenvectors: PCA

Lecture 8: correlation

correlation means:

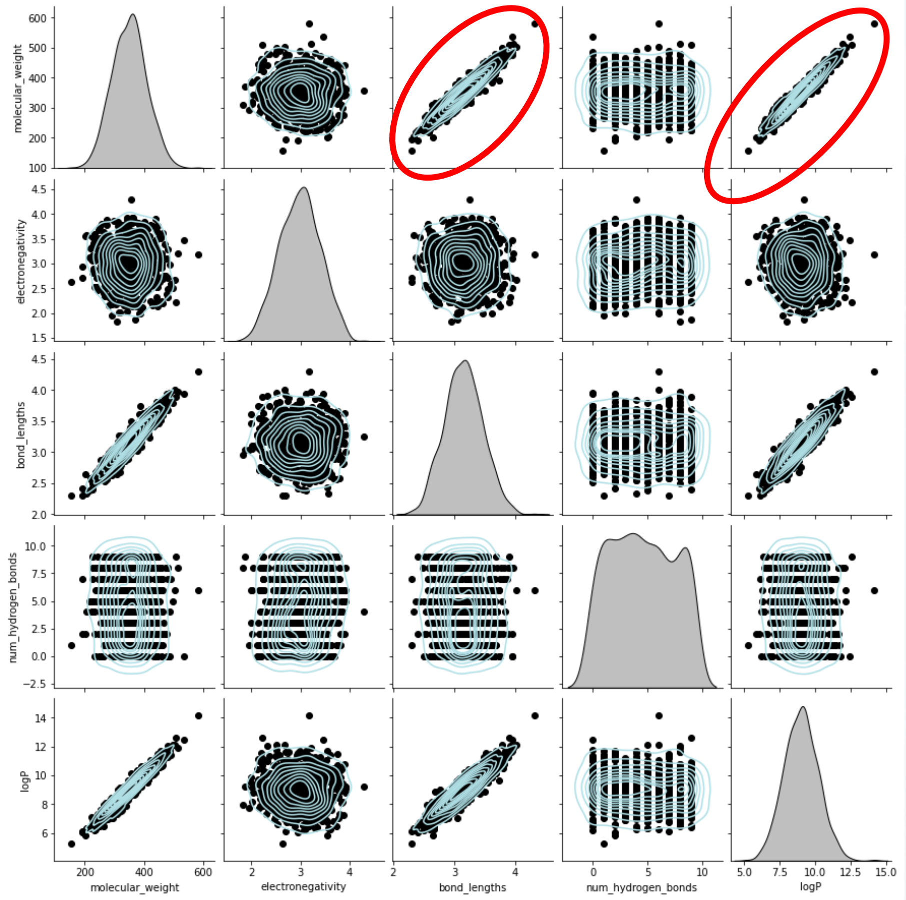
- features are **not mutually independent**

- we can predict feature ***a*** from feature ***b*** to some extend

- we don't need all features

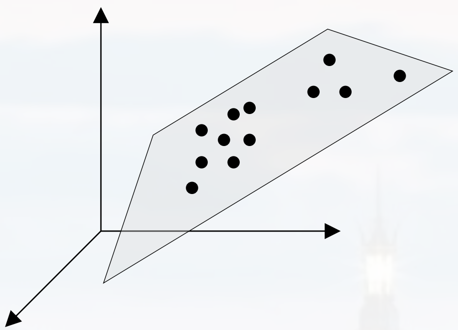
→ **reducing number of features** (dimensions) without losing information

label	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP
Toxic	382.602	2.00269	3.61153	3	9.82666
Toxic	408.961	2.93626	3.47904	6	9.85889
Non-Toxic	239.548	2.71413	2.63922	8	6.75962
Non-Toxic	315.58	2.85598	2.86034	9	8.70674
Non-Toxic	282.521	2.83877	2.9664	1	7.8173

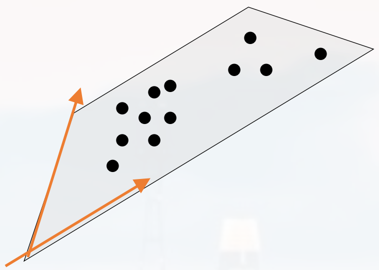


Eigenvalues and Eigenvectors: PCA

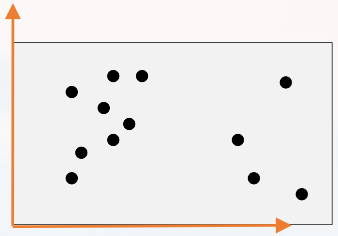
Lecture 8: correlation



from 3D to 2D



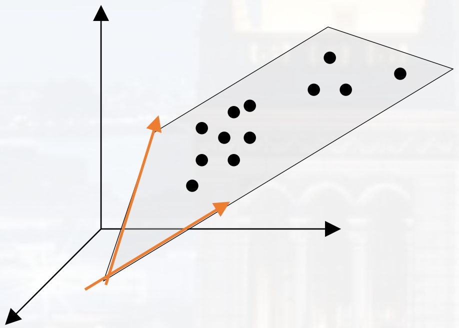
view from the top

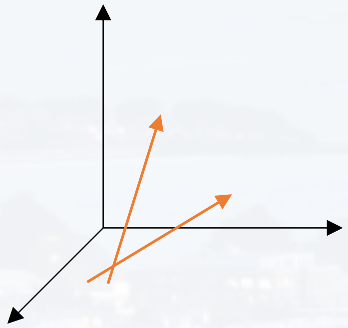


each data point is represented by **three** features...

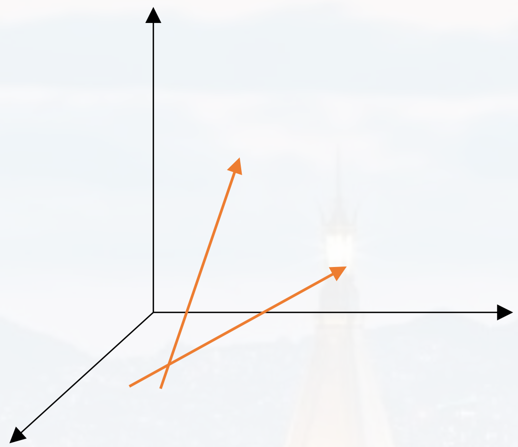
... but those features correlate
(x, y) → z

new coordinate system





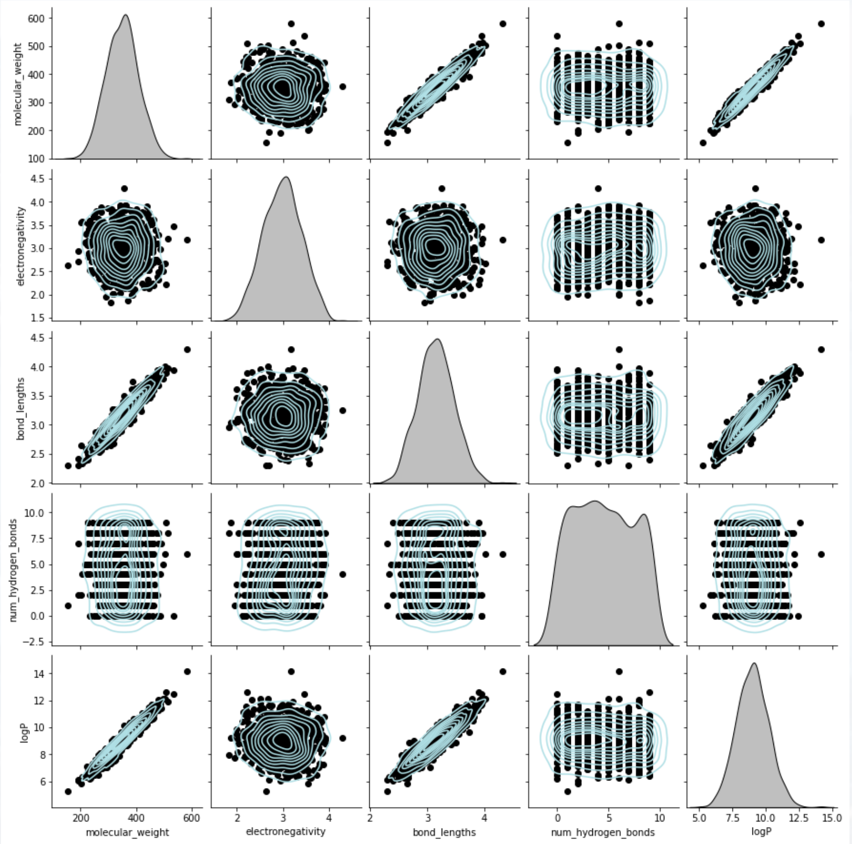
some features correlate!



Eigenvalues and Eigenvectors:

some features correlate!

How do we find these coordinates?



Eigenvalues and Eigenvectors:

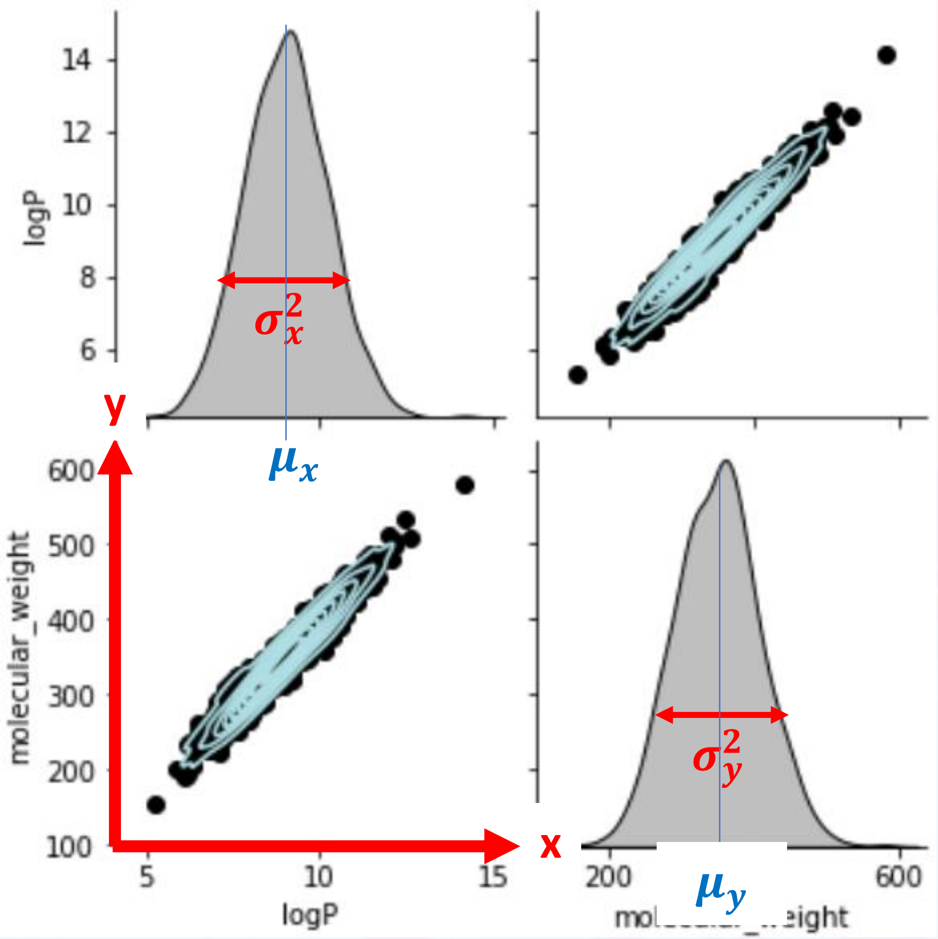
Lecture 8: correlation

$$corr(x,y) := \frac{cov(x,y)}{\sqrt{var(x)var(y)}}$$

$$var(x) \equiv \sigma_x^2 := \sum_i^N (x_i - \mu_x)^2$$

$$cov(x,y) := \sum_j^M \sum_i^N (x_i - \mu_x)(y_j - \mu_y)$$

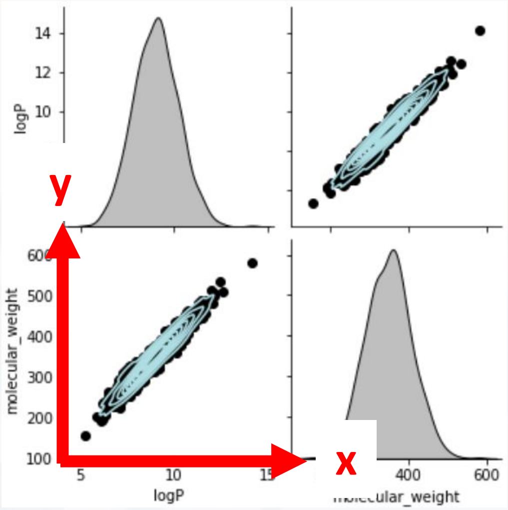
$$\sigma_{tot}^2 = \boxed{\sigma_x^2} + \boxed{\sigma_y^2} + \boxed{2\,cov(x,y)}$$



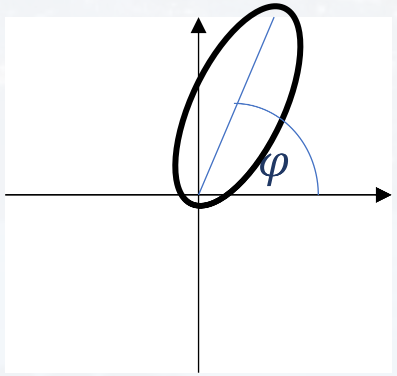
Eigenvalues and Eigenvectors:

$$\sigma_{tot}^2 = \boxed{\sigma_x^2} + \boxed{\sigma_y^2} + \boxed{2\, cov(x,y)}$$

$$= \boxed{\sum_i^N (x_i - \mu_x)^2} + \boxed{\sum_j^M (y_j - \mu_y)^2} + \boxed{2 \sum_j^M \sum_i^N (x_i - \mu_x)(y_j - \mu_y)}$$



$$const = \boxed{\frac{(x-\mu_x)^2}{a^2}} + \boxed{\frac{(y-\mu_y)^2}{b^2}} + \boxed{2\, c(x-\mu_x)(y-\mu_y)}$$



It is the same structure!

$$const = \begin{pmatrix} x-\mu_x \\ y-\mu_y \end{pmatrix}^T \begin{pmatrix} 1/a^2 & c \\ c & 1/b^2 \end{pmatrix} \begin{pmatrix} x-\mu_x \\ y-\mu_y \end{pmatrix}$$

$$C = \begin{pmatrix} x-\mu_x \\ y-\mu_y \end{pmatrix}^T \begin{pmatrix} \sigma_x^2 & cov(y,x) \\ cov(x,y) & \sigma_y^2 \end{pmatrix} \begin{pmatrix} x-\mu_x \\ y-\mu_y \end{pmatrix} \qquad cov(y,x) = cov(x,y)$$

Eigenvalues and Eigenvectors:

$$\sigma_{tot}^2 = \sigma_x^2 + \sigma_y^2 + \boxed{2\, cov(x,y)}$$

$$= \sum_i^N (x_i - \mu_x)^2 + \sum_j^M (y_j - \mu_y)^2 + \boxed{2 \sum_j^M \sum_i^N (x_i - \mu_x)(y_j - \mu_y)}$$

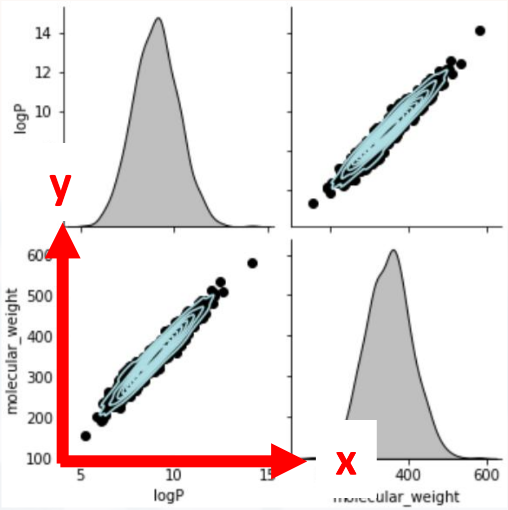
$$const = \frac{(x-\mu_x)^2}{a^2} + \frac{(y-\mu_y)^2}{b^2} + \boxed{2\, c(x-\mu_x)(y-\mu_y)}$$

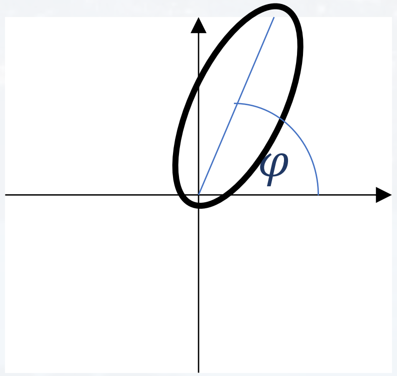
It is the same structure!

$$const = \begin{pmatrix} x-\mu_x \\ y-\mu_y \end{pmatrix}^T \begin{pmatrix} 1/a^2 & \boxed{c} \\ \boxed{c} & 1/b^2 \end{pmatrix} \begin{pmatrix} x-\mu_x \\ y-\mu_y \end{pmatrix}$$

$$C = \begin{pmatrix} x-\mu_x \\ y-\mu_y \end{pmatrix}^T \begin{pmatrix} \sigma_x^2 & \boxed{cov(y,x)} \\ \boxed{cov(x,y)} & \sigma_y^2 \end{pmatrix} \begin{pmatrix} x-\mu_x \\ y-\mu_y \end{pmatrix}$$

$$cov(y,x) = cov(x,y)$$





Eigenvalues and Eigenvectors:

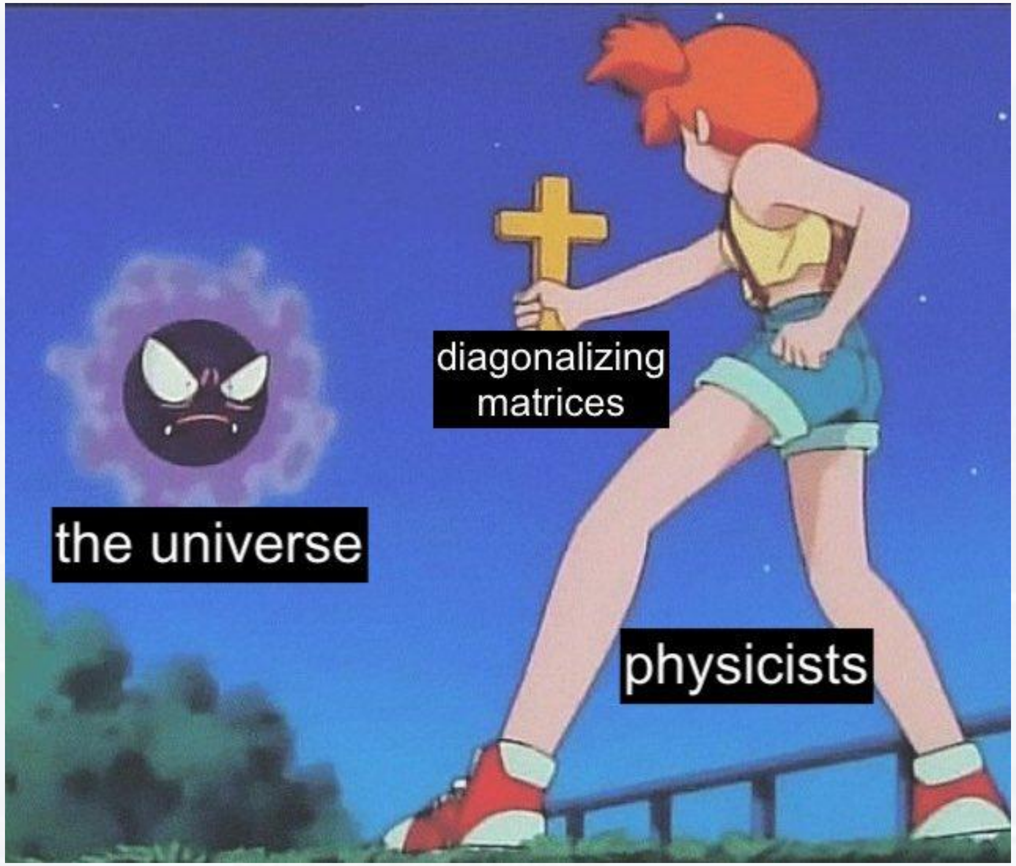
$$C = \begin{pmatrix} \boldsymbol{x}-\boldsymbol{\mu_x} \\ \boldsymbol{y}-\boldsymbol{\mu_y} \end{pmatrix}^T \underbrace{\begin{pmatrix} \sigma_x^2 & \boxed{cov(y,x)} \\ \boxed{cov(x,y)} & \sigma_y^2 \end{pmatrix}}_{\textit{covariance matrix } \Sigma} \begin{pmatrix} \boldsymbol{x}-\boldsymbol{\mu_x} \\ \boldsymbol{y}-\boldsymbol{\mu_y} \end{pmatrix} \qquad cov(y,x) = cov(x,y)$$

- geometrically, the covariance matrix can be interpreted as quadratic form
- the covariances are the **non-diagonal** elements of the covariance matrix
- aim: finding a coordinate transformation, where the covariance matrix is diagonal

$$\begin{pmatrix} \lambda_1 & \ddots\, 0 \ldots & 0 \\ 0 & \lambda_i \ \ddots & 0 \\ 0 & 0 & \lambda_N \end{pmatrix}$$

the diagonal entries are called eigenvalues (= variances in new coordinate system)

→ all variables **are independent**
→ principal components of the covariance matrix are **parallel** to the **new coordinate axes** **(=** eigenvectors**)**



angryfermion

Outline

- A Geometrical Approach
- Finding Eigenvectors and Eigenvalues
- PCA
- **Example I**
- Example II

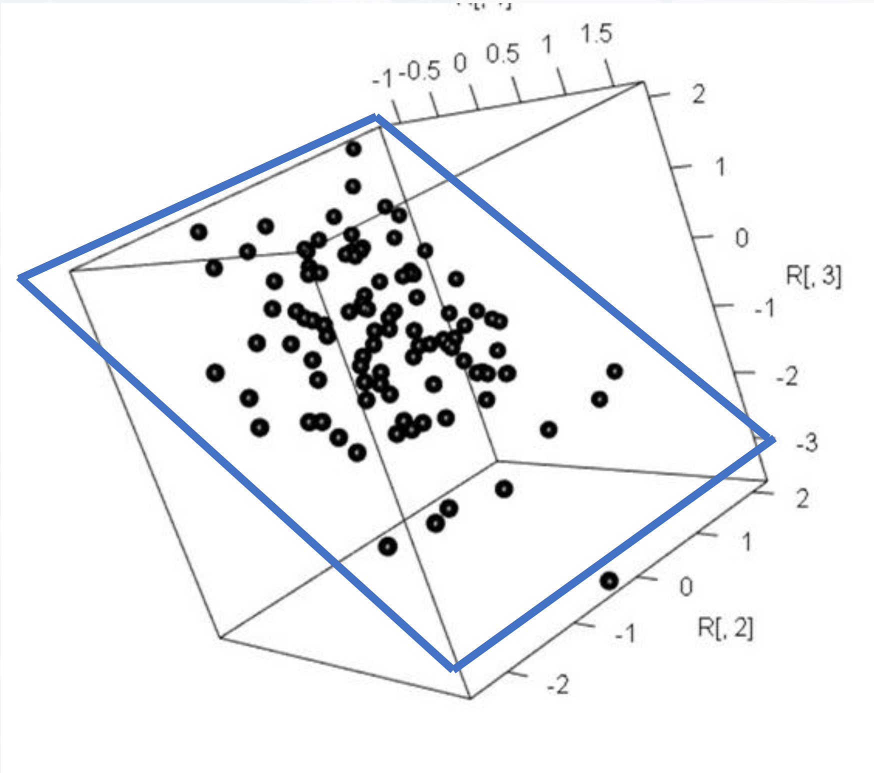
Eigenvalues and Eigenvectors:


```
from sklearn.decomposition import PCA
```

Let us take a look at some artificial data first:

see **`PCA_simple.ipynb`**

- 3D data cloud
- however, all data points seem to be located on **one plane**
- PCA should be able to **reduce dimensions**



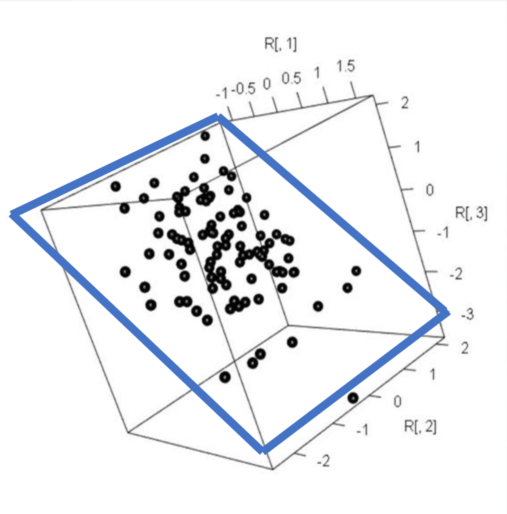
Eigenvalues and Eigenvectors: Example I

```
from sklearn.decomposition import PCA
```

Let us take a look at some artificial data first:

- 3D data cloud
- however, all data points seem to be located on **one plane**
- PCA should be able to **reduce dimensions**

```
XYZ = pd.read_csv('Rot.txt', delim_whitespace = True,\
                  header = None)
XYZ = np.array(XYZ)

fig = plt.figure(figsize = (12, 12))
ax = fig.add_subplot(projection = '3d')
ax.scatter(XYZ[:,0], XYZ[:,1], XYZ[:,2], c = 'black’,\
           marker = 'o', s = 40)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.tick_params(axis = 'both', which = 'major', labelsize = 30)
plt.show()
```



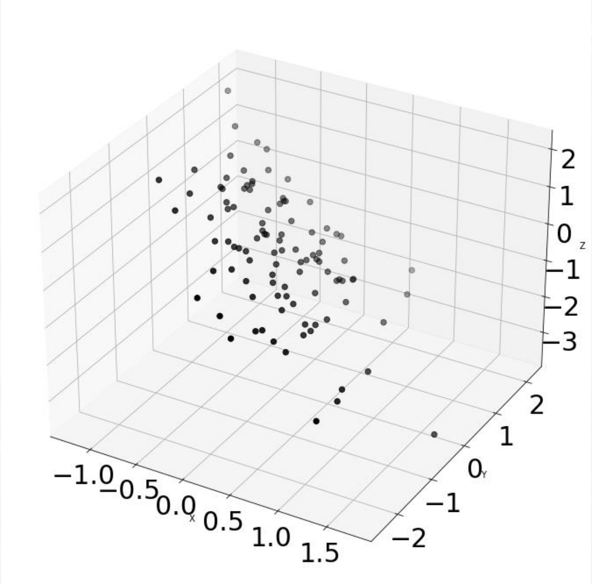
Eigenvalues and Eigenvectors:


```
from sklearn.decomposition import PCA
```

Let us take a look at some artificial data first:

- 3D data cloud
- however, all data points seem to be located on **one plane**
- PCA should be able to **reduce dimensions**

```
XYZ = pd.read_csv('Rot.txt', delim_whitespace = True,\
                  header = None)
XYZ = np.array(XYZ)

fig = plt.figure(figsize = (12, 12))
ax = fig.add_subplot(projection = '3d')
ax.scatter(XYZ[:,0], XYZ[:,1], XYZ[:,2], c = 'black',\
           marker = 'o', s = 40)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.tick_params(axis = 'both', which = 'major', labelsize = 30)
plt.show()
```



Eigenvalues and Eigenvectors:

- 3D data cloud
- however, all data points seem to be located on **one plane**
- PCA should be able to **reduce dimensions**

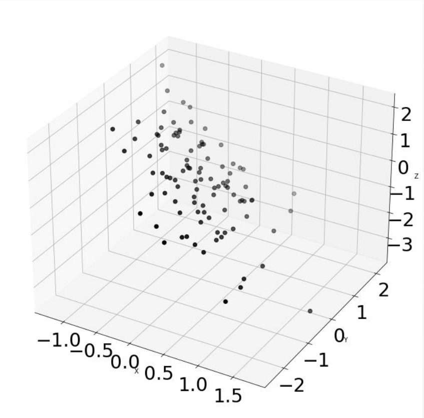
performing the actual PCA:

```
out = PCA(n_components = 3).fit(XYZ)

eigenVec = out.components_
eigenVal = out.explained_variance_
eigenXYZ = out.transform(XYZ)
```

plotting the eigenvalue spectrum:

```
xplot = np.arange(1,4)

plt.bar(xplot, eigenVal, color = (0.8, 0.8, 0.8), edgecolor = 'black')
plt.xlabel('dimension')
plt.ylabel('eigenvalue')
plt.yscale('log')
plt.xticks(xplot)
plt.show()
```

Eigenvalues and Eigenvectors:

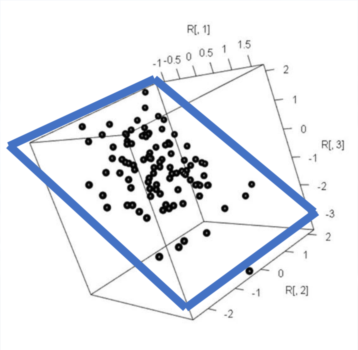


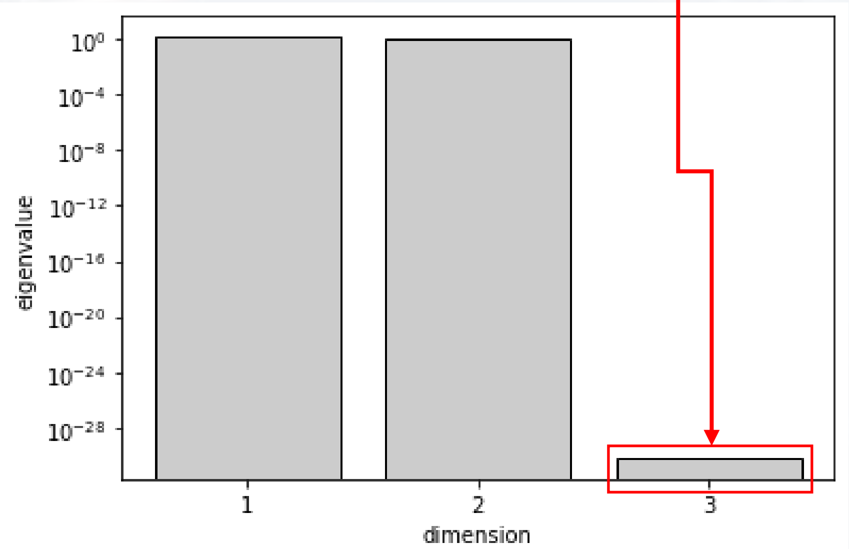

```
out = PCA(n_components = 3).fit(XYZ)

eigenVec = out.components_
eigenVal = out.explained_variance_
eigenXYZ = out.transform(XYZ)
```

plotting the eigenvalue spectrum:

```
xplot = np.arange(1,4)

plt.bar(xplot, eigenVal, color = (0.8, 0.8, 0.8), edgecolor = 'black')
plt.xlabel('dimension')
plt.ylabel('eigenvalue')
plt.yscale('log')
plt.xticks(xplot)
plt.show()
```

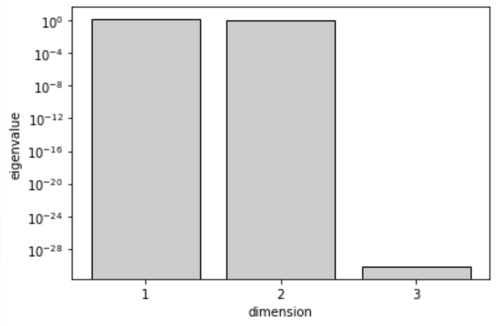
one eigenvalue is zero





Eigenvalues and Eigenvectors: Example I

plotting the eigenvalue spectrum:

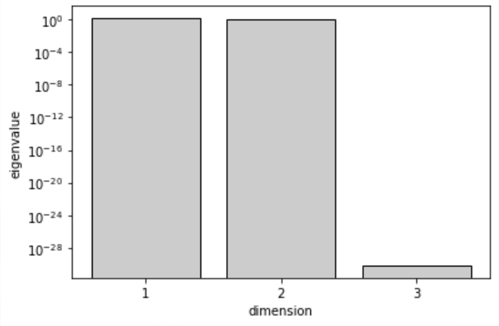



```
xplot = np.arange(1,4)

plt.bar(xplot, eigenVal, color = (0.8, 0.8, 0.8), edgecolor = 'black')
plt.xlabel('dimension')
plt.ylabel('eigenvalue')
plt.yscale('log')
plt.xticks(xplot)
plt.show()
```

```
fig = plt.figure(figsize = (12, 12))
ax = fig.add_subplot(projection = '3d')
ax.scatter(eigenXYZ[:,0], eigenXYZ[:,1], eigenXYZ[:,2], c = 'black', \
           marker = 'o', s = 40)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.tick_params(axis = 'both', which = 'major', labelsize = 30)
plt.show()
```

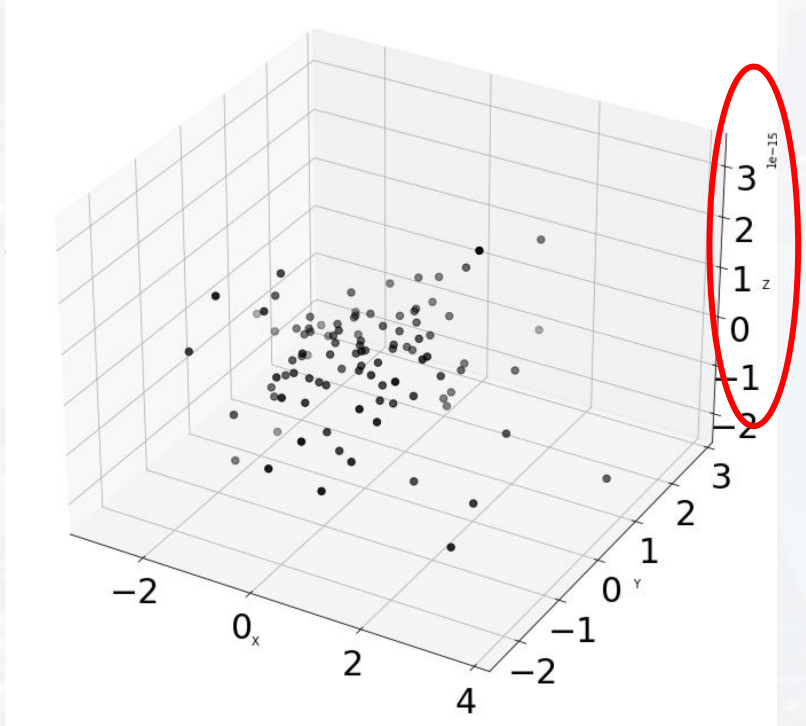
Eigenvalues and Eigenvectors:

plotting the eigenvalue spectrum:




```
xplot = np.arange(1,4)

plt.bar(xplot, eigenVal, color = (0.8, 0.8, 0.8), edgecolor = 'black')
plt.xlabel('dimension')
plt.ylabel('eigenvalue')
plt.yscale('log')
plt.xticks(xplot)
plt.show()


fig = plt.figure(figsize = (12, 12))
ax = fig.add_subplot(projection = '3d')
ax.scatter(eigenXYZ[:,0], eigenXYZ[:,1], eigenXYZ[:,2], c = 'bl
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.tick_params(axis = 'both', which = 'major', labelsize = 30)
plt.show()
```

almost no variance along new z-coord



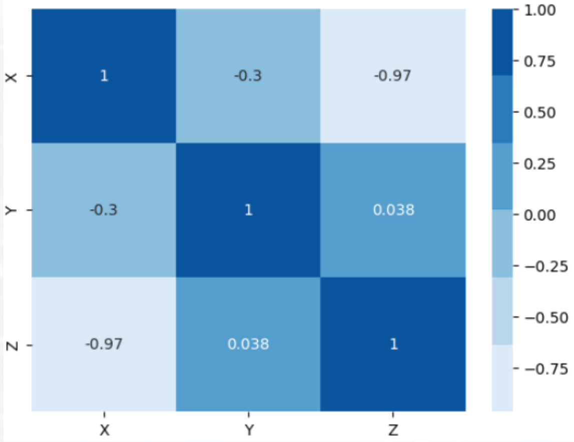
check also eg:

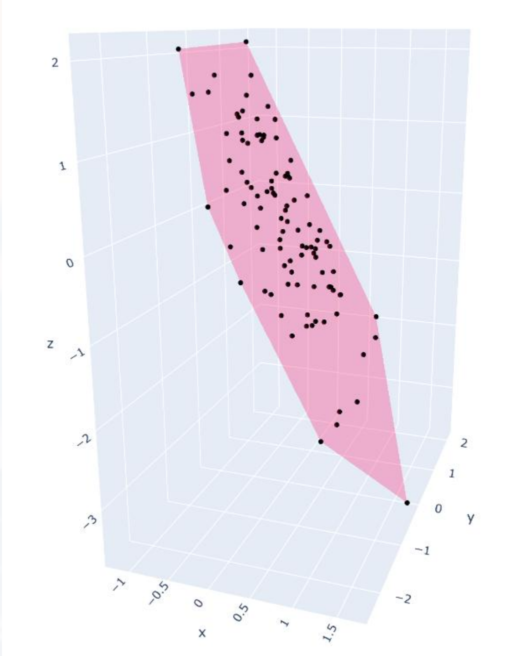
```
np.dot(eigenVec[:,0],eigenVec[:,1])
```

Eigenvalues and Eigenvectors:

Summary:

- We don’t need **three** coordinates in order to describe the data points
  → some of the directions (features) correlate

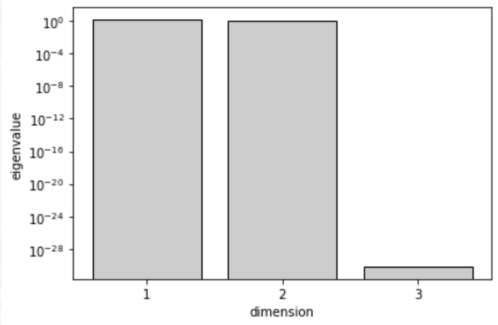


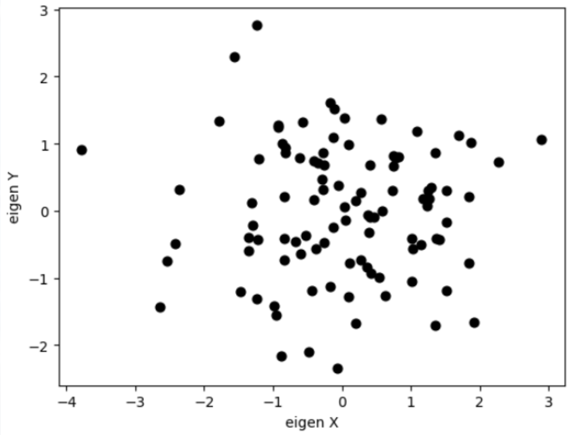


- running a PCA in order to find the proper coordinate system

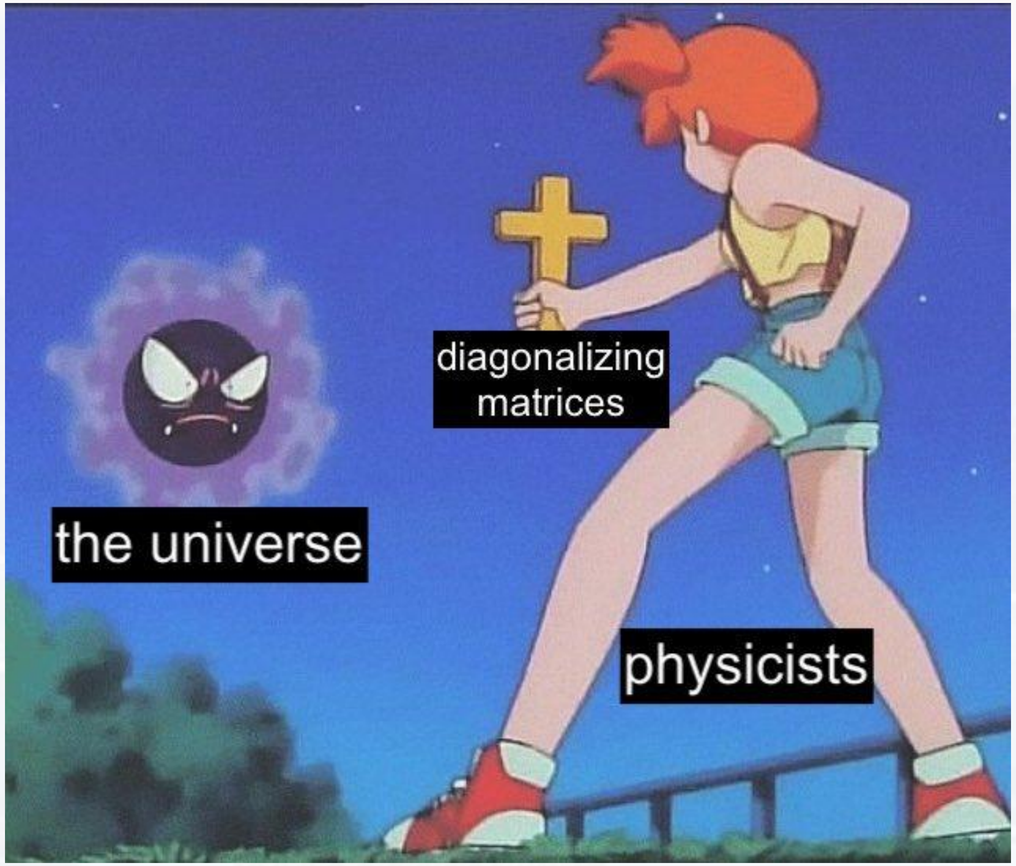
- **one** of **three** eigenvalues is a lot smaller than the other **two**

→ We only need **two** coordinates for the data set





We can reduce the complexity of the data set without loosing information



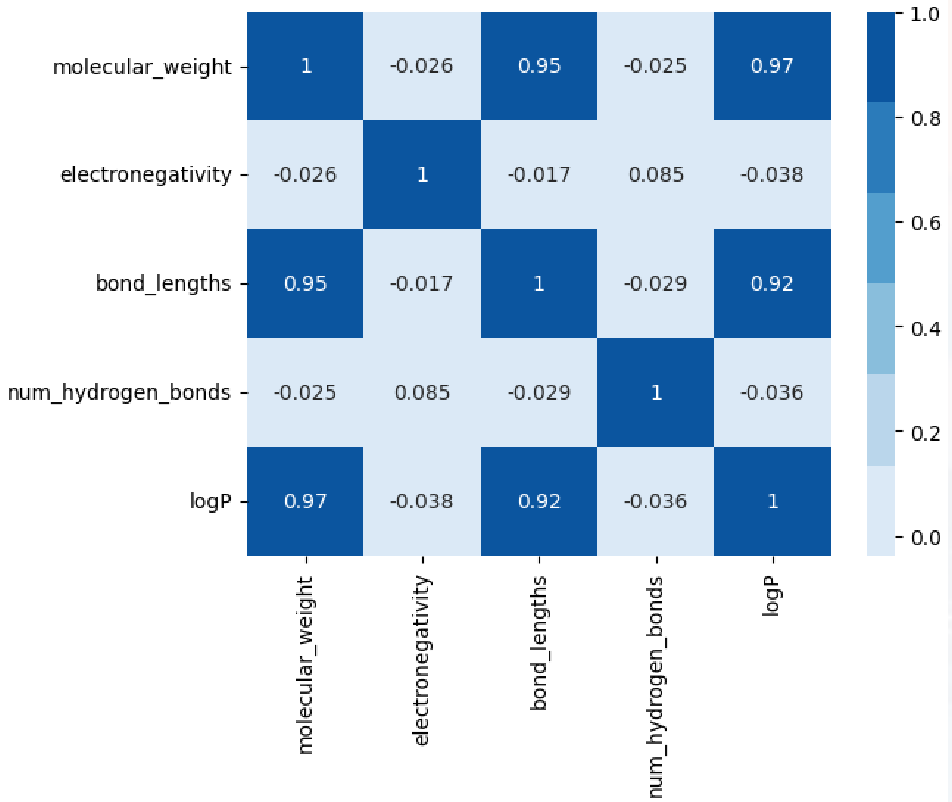
angryfermion

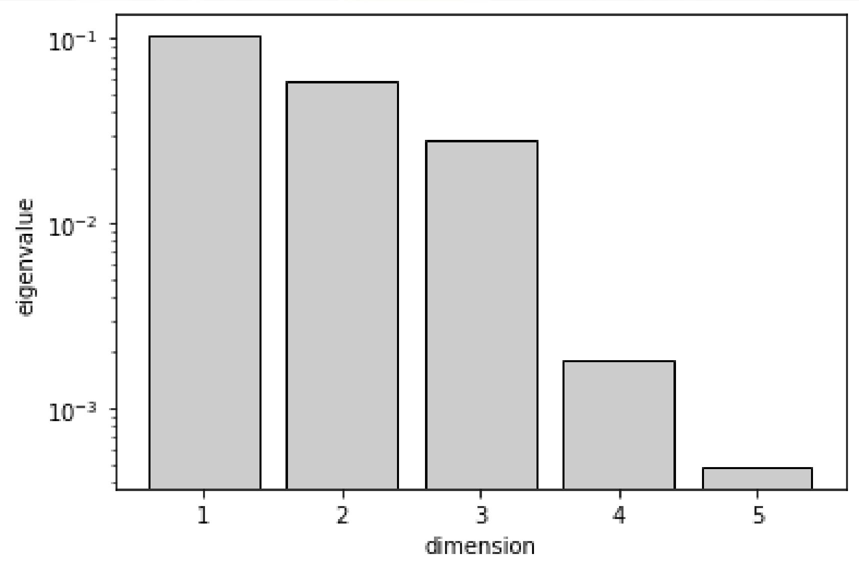
Outline

- A Geometrical Approach
- Finding Eigenvectors and Eigenvalues
- PCA
- Example I
- **Example II**

Eigenvalues and Eigenvectors: Example II

label	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP
Toxic	382.602	2.00269	3.61153	3	9.82666
Toxic	408.961	2.93626	3.47904	6	9.85889
Non-Toxic	239.548	2.71413	2.63922	8	6.75962
Non-Toxic	315.58	2.85598	2.86034	9	8.70674
Non-Toxic	282.521	2.83877	2.9664	1	7.8173





Lecture Exercise!

Thank you very much for your attention!