BIOS 7747: Machine Learning for Biomedical Applications

Dimensionality reduction

Antonio R. Porras (antonio.porras@cuanschutz.edu)

Department of Biostatistics and Informatics
Colorado School of Public Health
University of Colorado Anschutz Medical Campus

Outline

- ❑ The curse of dimensionality
- ❑ Dimensionality reduction: principal component analysis
  - Linear geometric transformations
  - Eigenvectors and eigenvalues
  - Multivariate Gaussian transformations
  - Eigen-decomposition of the covariance matrix
- ❑ Supervised dimensionality reduction: linear discriminant analysis

The curse of dimensionality

- Hughes phenomenon or peaking paradox
  - Model performance (on test) increases with number of features up to an optimal performance, and decreases after that

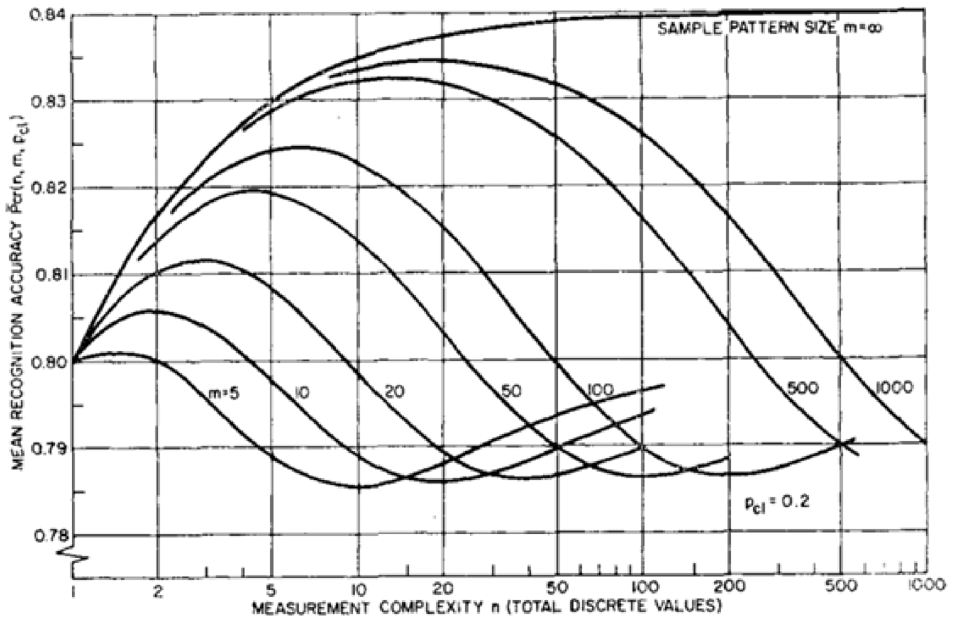
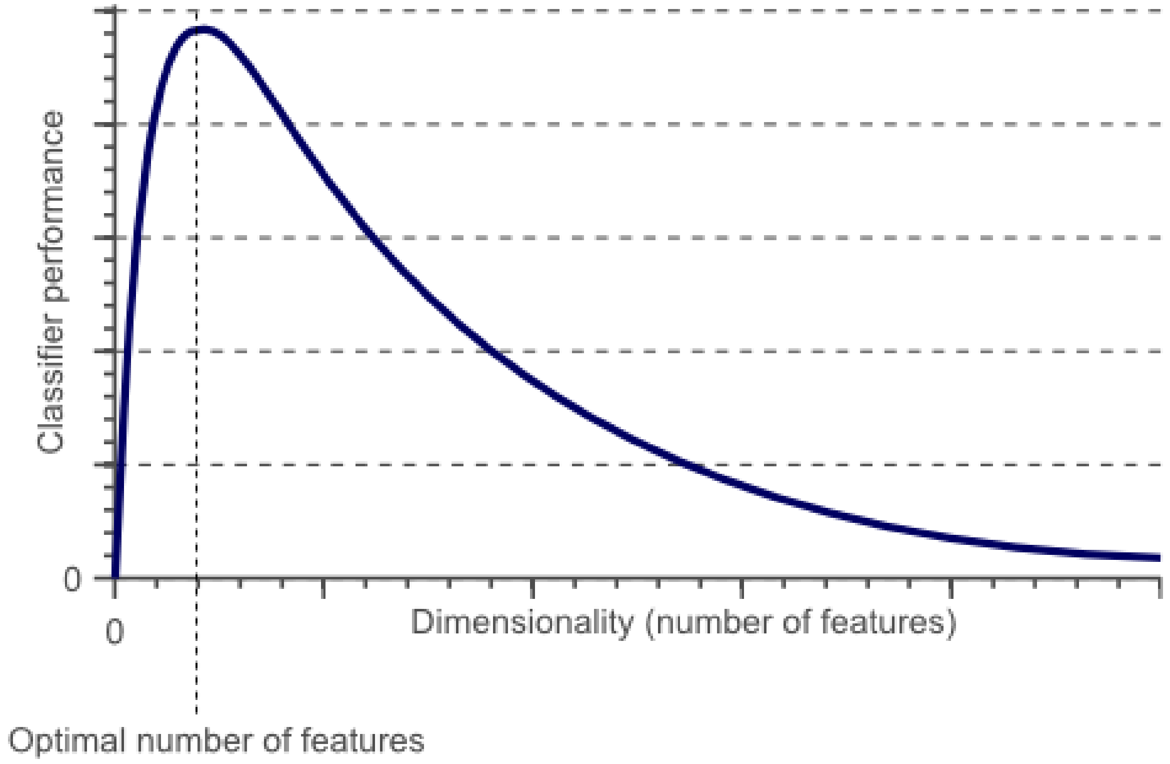


Fig. 4. Finite data set accuracy ($p_{c1} = \frac{1}{5}$).



G. Hughes, "On the mean accuracy of statistical pattern recognizers," in *IEEE Transactions on Information Theory*, vol. 14, no. 1, pp. 55-63, January 1968, doi: 10.1109/TIT.1968.1054102.

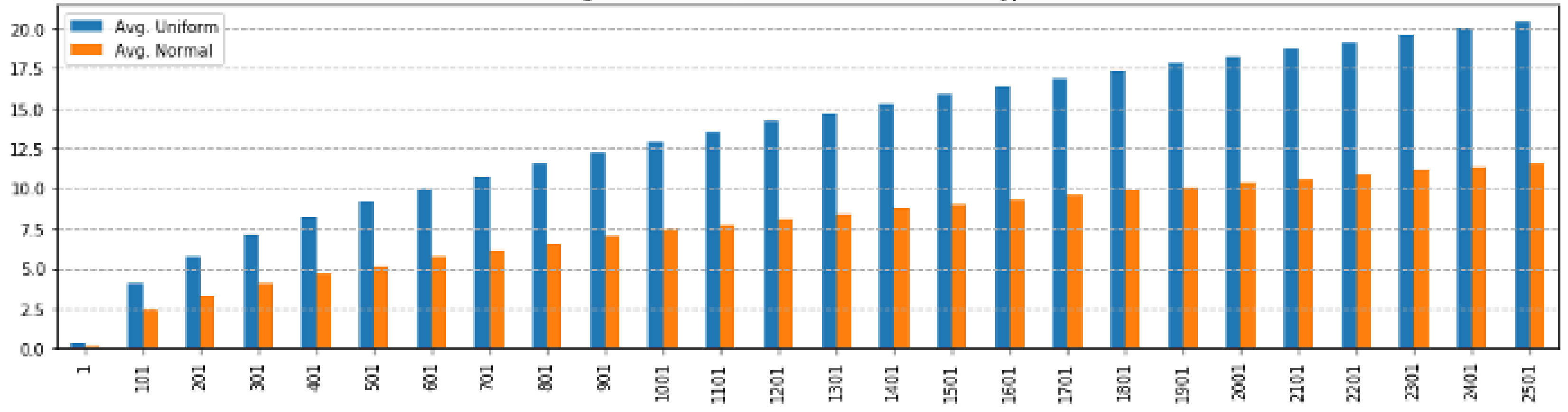
The curse of dimensionality

- Increasing the number of features increases data sparsity

Average Euclidean distance between two samples for M features

$$d_{p,q} = \sqrt{\sum_{i=1}^{M} (p_i - q_i)^2}$$

Average Distance of 1,000 Data Points in a Unit Hypercube



- As number of features increase, larger sample sizes are needed to preserve average distances (to fill in the empty space)

The curse of dimensionality

- High number of features usually mean higher likelihood of creating an overfit model

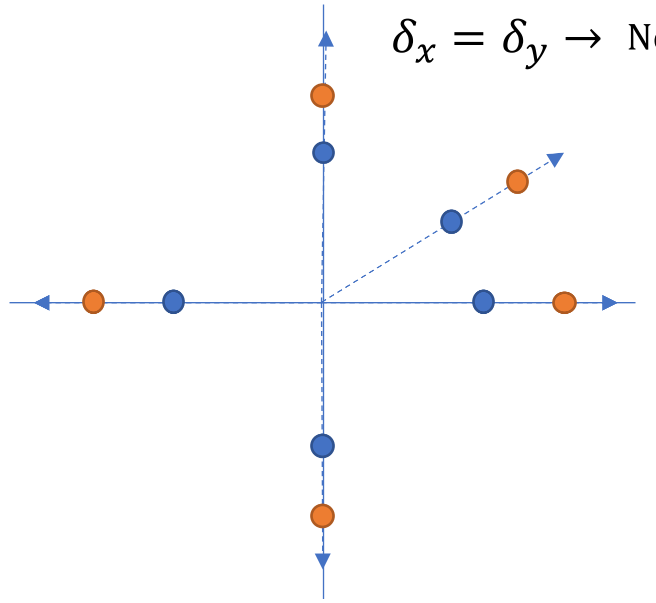
- Dimensionality reduction:

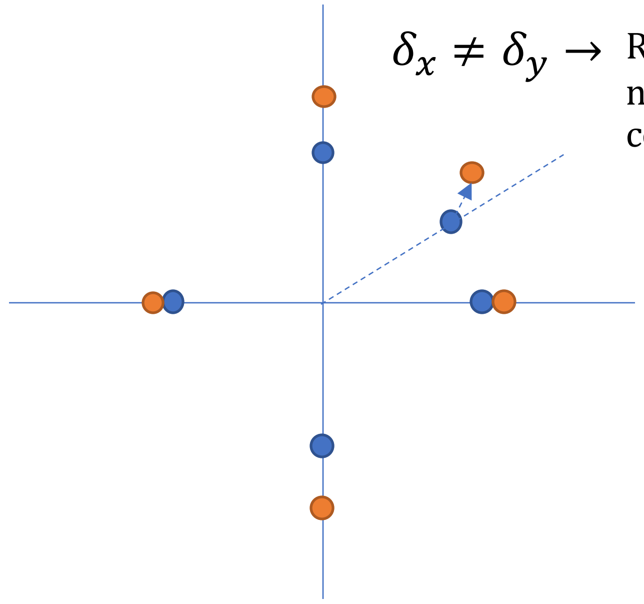
  - Feature selection: identify most relevant features $\boxed{\text{to a machine learning task}}$

  - Feature extraction: create a lower-dimensional representations of the available features that is meaningful $\boxed{\text{to a machine learning task}}$

Linear geometric transformations

- Linear geometric transformations $T(x) = Ax$ $A = \begin{pmatrix} 1+\delta_x & 0 \\ 0 & 1+\delta_y \end{pmatrix}$




$\delta_x \neq \delta_y \rightarrow$ Rotation only for points not laying in the coordinate axes

For any scaling-only matrix $\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$, points in the axes are scaled but not rotated

- Scale on axis 1: $\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \lambda_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \lambda_1 e_1 = \Lambda e_1$
- Scale on axis 2: $\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \lambda_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \lambda_2 e_2 = \Lambda e_2$

Any diagonal matrix maps any vector parallel to a basis vector into another vector that is also parallel to that basis vector

Eigenvectors and eigenvalues

- For any diagonal matrix $\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$, points in the axes are scaled but not rotated
  - Scale on axis 1: $\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \lambda_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \lambda_1 \boldsymbol{e}_1 = \boldsymbol{\Lambda e_1}$
  - Scale on axis 2: $\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \lambda_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \lambda_2 \boldsymbol{e}_2 = \boldsymbol{\Lambda e_2}$

$\Rightarrow$

Eigenvalue equation

$$\boxed{\boldsymbol{\Lambda} \underbrace{\boldsymbol{e_i}}_{\text{eigenvector}} = \underbrace{\boldsymbol{\lambda_i}}_{\text{eigenvalue}} \boldsymbol{e_i}}$$

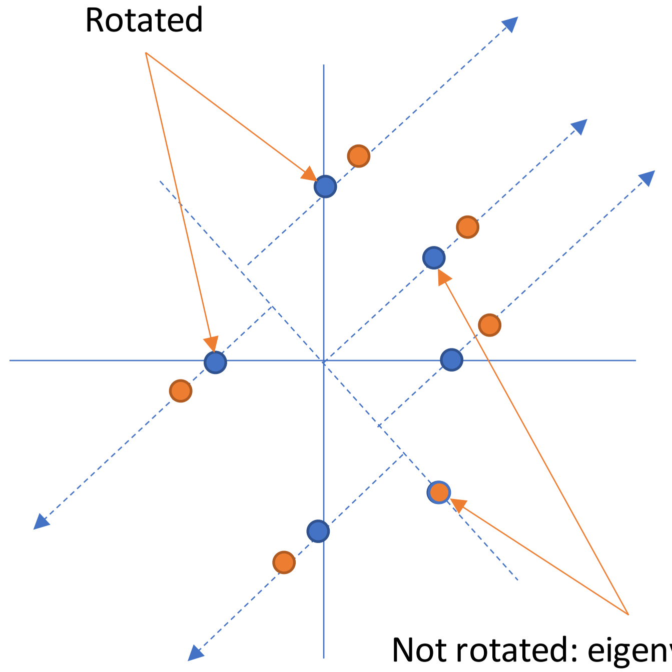
- Eigenvector of a matrix: any vector $\boldsymbol{v}$ that is mapped by the matrix into a parallel vector $\lambda \boldsymbol{v}$

- Eigenvalue of a matrix: the scale factor of an eigenvector

- A matrix in $\mathbb{R}^n$ has n pairs of eigenvectors and eigenvalues

Eigenvectors and eigenvalues

- Rotated scale transformations: $T(x) = Ax$ $\qquad A = \Phi\Lambda\Phi^T$



$$\Lambda = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\Phi = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \text{ [45 degrees]}$$

$$Ae_i = \lambda_i e_i \leftarrow \text{eigenvector}$$

$$\Phi\Lambda\Phi^T e_i = \lambda_i e_i$$

$$\Phi^T\Phi\Lambda\Phi^T e_i = \Phi^T\lambda_i e_i$$

$$\Lambda\Phi^T e_i = \Phi^T(\lambda_i e_i)$$

$$\Lambda\underbrace{\Phi^T e_i} = \lambda_i\underbrace{\Phi^T e_i}$$

Basis vector: $b_i = \Phi^T e_i$
$e_i = \Phi b_i$

- Eigenvectors are rotated standard basis vectors

Any symmetric transformation matrix A can be written as:

- $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 \\ 1 \end{pmatrix} = e_1$
- $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = e_2$

Each column is an eigenvector!

$$A = \Phi\Lambda\Phi^T = (e_1, e_2 \dots)\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}(e_1, e_2 \dots)^T$$

Eigenvectors and eigenvalues

- Calculating eigenvalues and eigenvectors: 2D example

$$\boldsymbol{Ae_i} = \lambda_i \boldsymbol{e_i}$$

Solutions for which $\boldsymbol{e_i} \neq \mathbf{0}$:

$$\det(\boldsymbol{A}) = \begin{vmatrix} a-\lambda_i & b \\ c & d-\lambda_i \end{vmatrix} = (a-\lambda_i)(d-\lambda_i) - cd = 0$$

$$\boldsymbol{Ae_i} = \lambda_i \boldsymbol{I} \boldsymbol{e_i}$$

$$\det(\boldsymbol{A} - \lambda_i \boldsymbol{I}) = 0$$

$$\underbrace{\lambda_i^2 - \lambda_i(a+d) + (ad - bc) = 0}_{\text{Characteristic equation of matrix A}}$$

$$(\boldsymbol{A} - \lambda_i \boldsymbol{I})\boldsymbol{e_i} = 0$$

$$\boxed{\lambda_i = \frac{1}{2}(\mathrm{a}+\mathrm{d}) \pm \frac{1}{2}\sqrt{(a-d)^2 + 4bc}} \longrightarrow$$

Eigenvalues can be real or complex!

In a symmetric matrix, since $b = \mathrm{c}$, then: $\lambda_i = \frac{1}{2}(\mathrm{a}+\mathrm{d}) \pm \frac{1}{2}\sqrt{(a-d)^2 + 4b^2}$. All eigenvalues are real.

- Eigen-decomposition: process of decomposing $\boldsymbol{A}$ in $\boldsymbol{A} = \boldsymbol{\Phi\Lambda\Phi}^{\mathrm{T}}$

$$\boldsymbol{A\Phi} = \boldsymbol{\Phi\Lambda\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$$

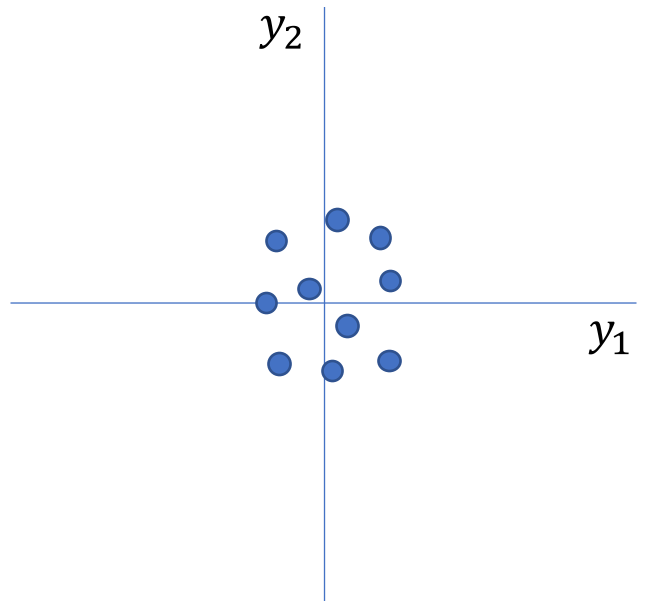
$$\boldsymbol{A\Phi} = \boldsymbol{\Phi\Lambda}$$

$$\Big\updownarrow \text{ Equivalent}$$

$$\boldsymbol{Ae_i} = \lambda_{\boldsymbol{i}} e_i$$

Multivariate Gaussian distributions

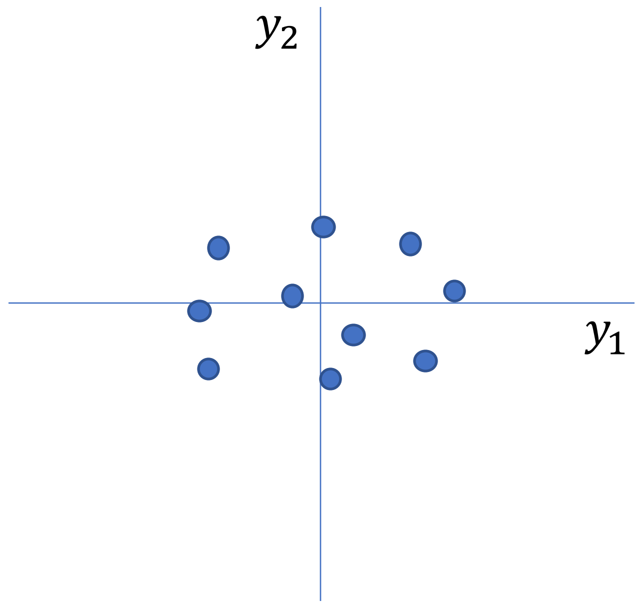
Isotropic



Variance: σ^2

$$p(y) \propto e^{-\frac{1}{2\sigma^2}y^T y}$$

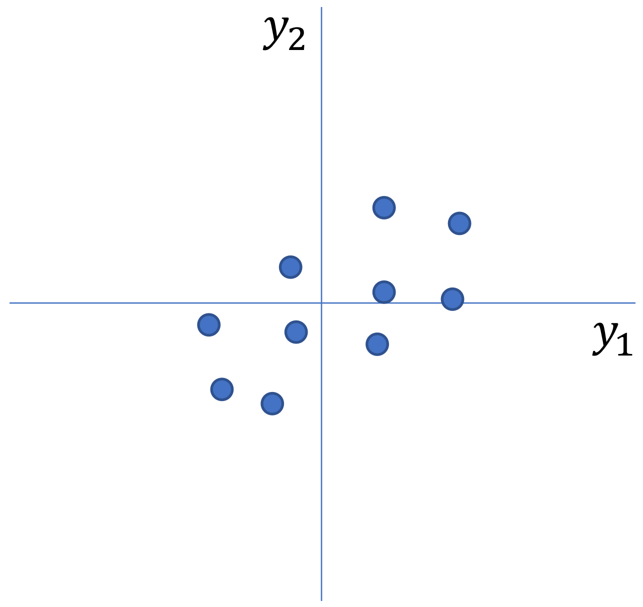
Anisotropic (no correlation)



Variance matrix: $S = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}$

$$p(y) \propto e^{-\frac{1}{2}y^T S^{-1} y}$$

Anisotropic (with correlation)

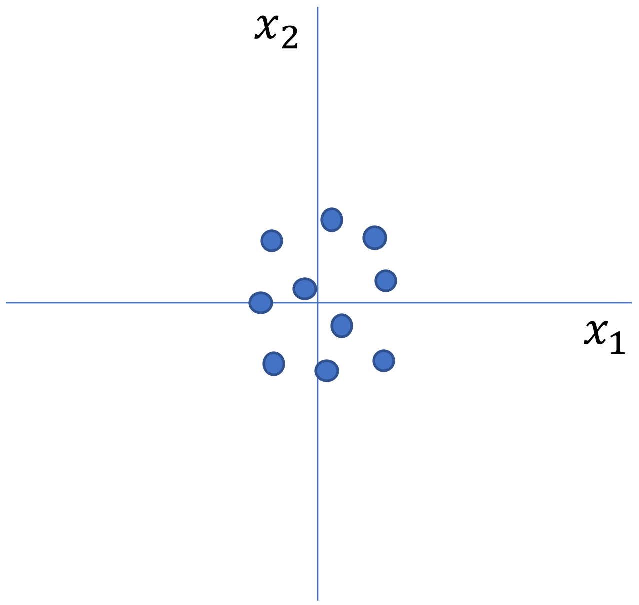


Co-variance matrix: $\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{1,2} \\ \sigma_{2,1} & \sigma_2^2 \end{pmatrix}$

$$p(y) \propto e^{-\frac{1}{2}y^T \Sigma^{-1} y}$$

Transforming multivariate Gaussian distributions

Isotropic



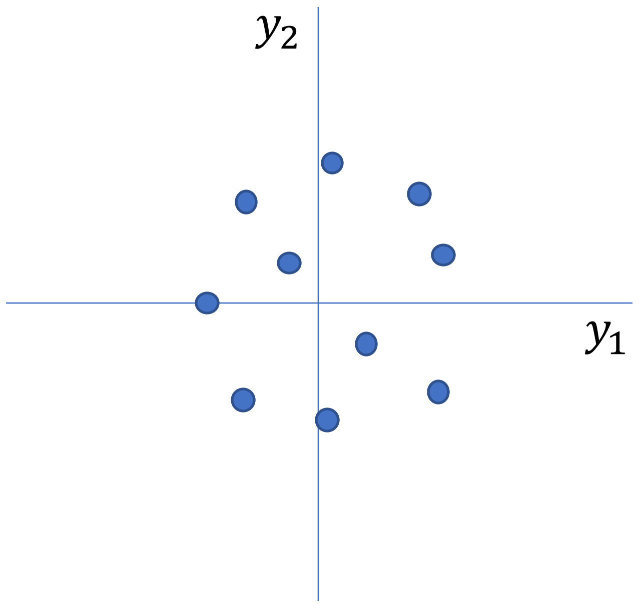
Variance: $\sigma_x^2 = 1$

$p(y) \propto e^{-\frac{1}{2}x^T x}$

$y = \sigma_y x$

$x = \sigma_y^{-1} y$

Isotropic

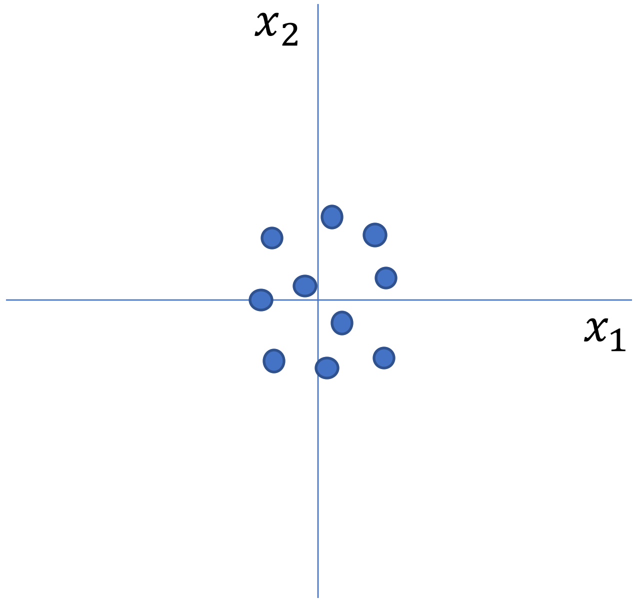


Variance: σ_y^2

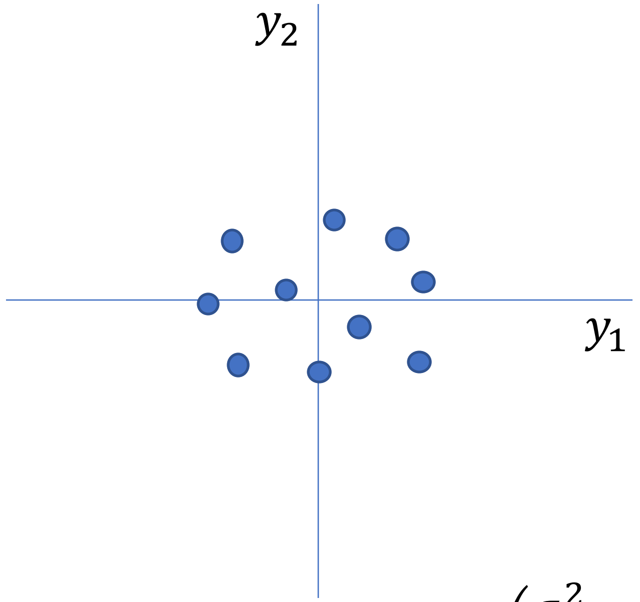
$p(y) \propto e^{-\frac{1}{2\sigma_y^2}y^T y}$

Transforming multivariate Gaussian distributions

Isotropic



Anisotropic (no correlation)



$$y = S_y x$$

$$x = S_y^{-1} y$$

Variance: $\sigma_x^2 = 1$

$$p(y) \propto e^{-\frac{1}{2}x^T x}$$

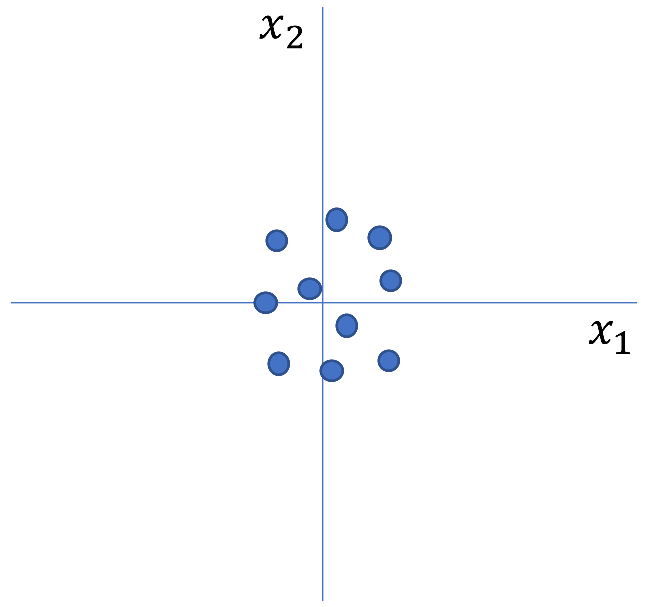
Variance matrix: $S_y^2 = \Lambda = \begin{pmatrix} \sigma_{y_1}^2 & 0 \\ 0 & \sigma_{y_2}^2 \end{pmatrix}$

$$p(y) \propto e^{-\frac{1}{2}y^T S_y^{-2} y} = e^{-\frac{1}{2}y^T \Lambda^{-1} y}$$

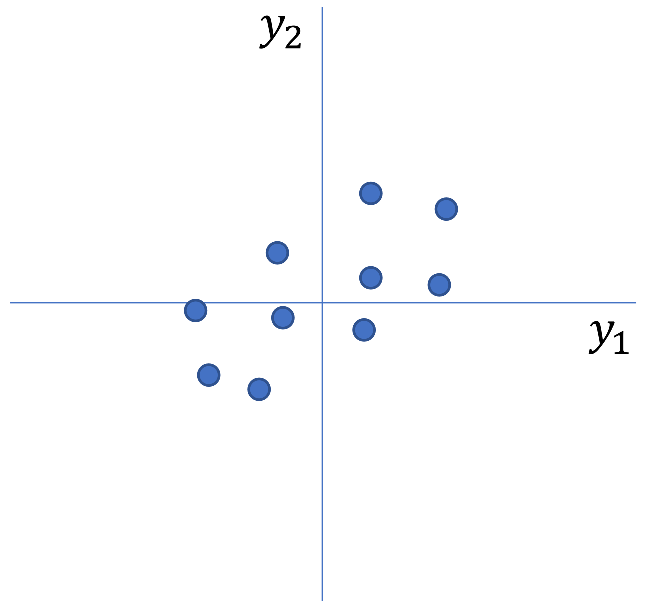
$$\Lambda^{-1} = \begin{pmatrix} \sigma_{y_1}^{-2} & 0 \\ 0 & \sigma_{y_2}^{-2} \end{pmatrix}$$

Transforming multivariate Gaussian distributions

Isotropic



Anisotropic (with correlation)



$$y = \Phi S_y \Phi^T x$$

$$x = \Phi S_y^{-1} \Phi^T y$$

Variance: $\sigma_x^2 = 1$

$$p(y) \propto e^{-\frac{1}{2}x^T x}$$

Covariance matrix: $\Sigma = \begin{pmatrix} \sigma_{y_1}^2 & \sigma_{y_1 y_2} \\ \sigma_{y_2 y_1} & \sigma_{y_2}^2 \end{pmatrix} = \Phi S_y^2 \Phi^T = \Phi \Lambda \Phi^T$

$$p(y) \propto e^{-\frac{1}{2}y^T \Sigma^{-1} y}$$

$$\Sigma^{-1} = \Phi \Lambda^{-1} \Phi^T$$

Eigen-decomposition of the covariance matrix

- A covariance matrix can be seen as a linear mapping between an isotropic Gaussian distribution and a non-isotropic Gaussian distribution with correlated features.
    - Note that all distributions are centered at the origin of coordinates (i.e., feature means are zero)

- Eigen-decomposition of the covariance matrix provides:
    - Eigenvectors: The basis vectors of the rotated space
    - Eigenvalues: The variance of the data is the direction of the basis vectors

Principal component analysis

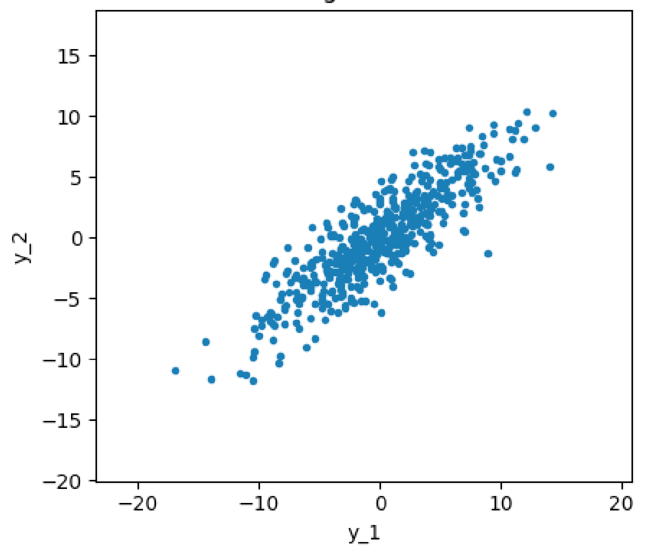
- PCA: data analysis in a rotated feature space (so the features are not correlated) calculated using the eigenvectors of the covariance matrix

Example: Data analysis in the rotated space using the eigenvectors of the covariance matrix

```
# Creating dataset with feature correlations
y1 = np.random.normal(0, 5, 500)
noise = np.random.normal(0, 2, 500)
y2 = 0.7 * y1 + noise
data = np.array([y1, y2]).T

covariance = np.cov(data.T)
print(f'Covariance of original data: {covariance}')
```

```
Covariance of original data:
[[28.07275458 20.33218298]
 [20.33218298 18.98284443]]
```

Original data

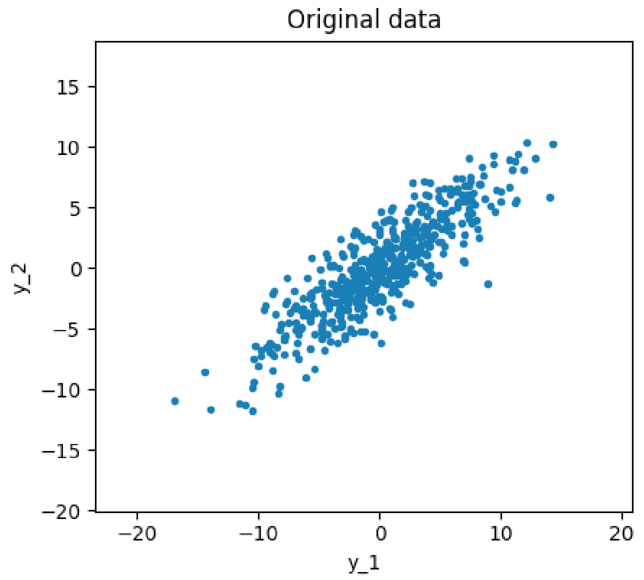


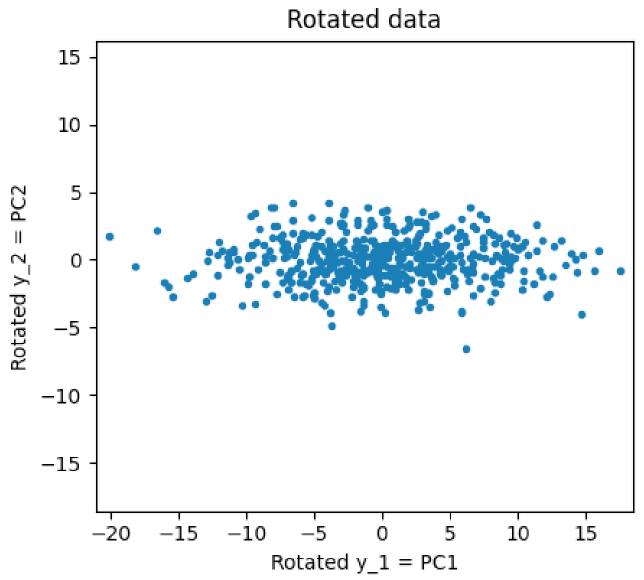
Principal component analysis

Example: Data analysis in the rotated space using the eigenvectors of the covariance matrix

```
# Calculating eigenvalues and eigenvectors
eigenValues, eigenVectors = np.linalg.eig(covariance)
eigenVectors = eigenVectors.T
print(f'Eigenvalues: {eigenValues}')
print(f'Eigenvectors: {eigenVectors}')

# Rotating the data
yRotated = data @ eigenVectors.T
yRotatedCovariance = np.cov(yRotated.T)
print(f'Covariance of y rotated: {yRotatedCovariance}')
```

```
Eigenvalues:
[44.36176891  2.6938301 ]

Eigenvectors:
[[ 0.78043295  0.62523948]
 [-0.62523948  0.78043295]]

Covariance of y rotated:
[[4.43617689e+01 9.45580732e-16]
 [9.45580732e-16 2.69383010e+00]]
```



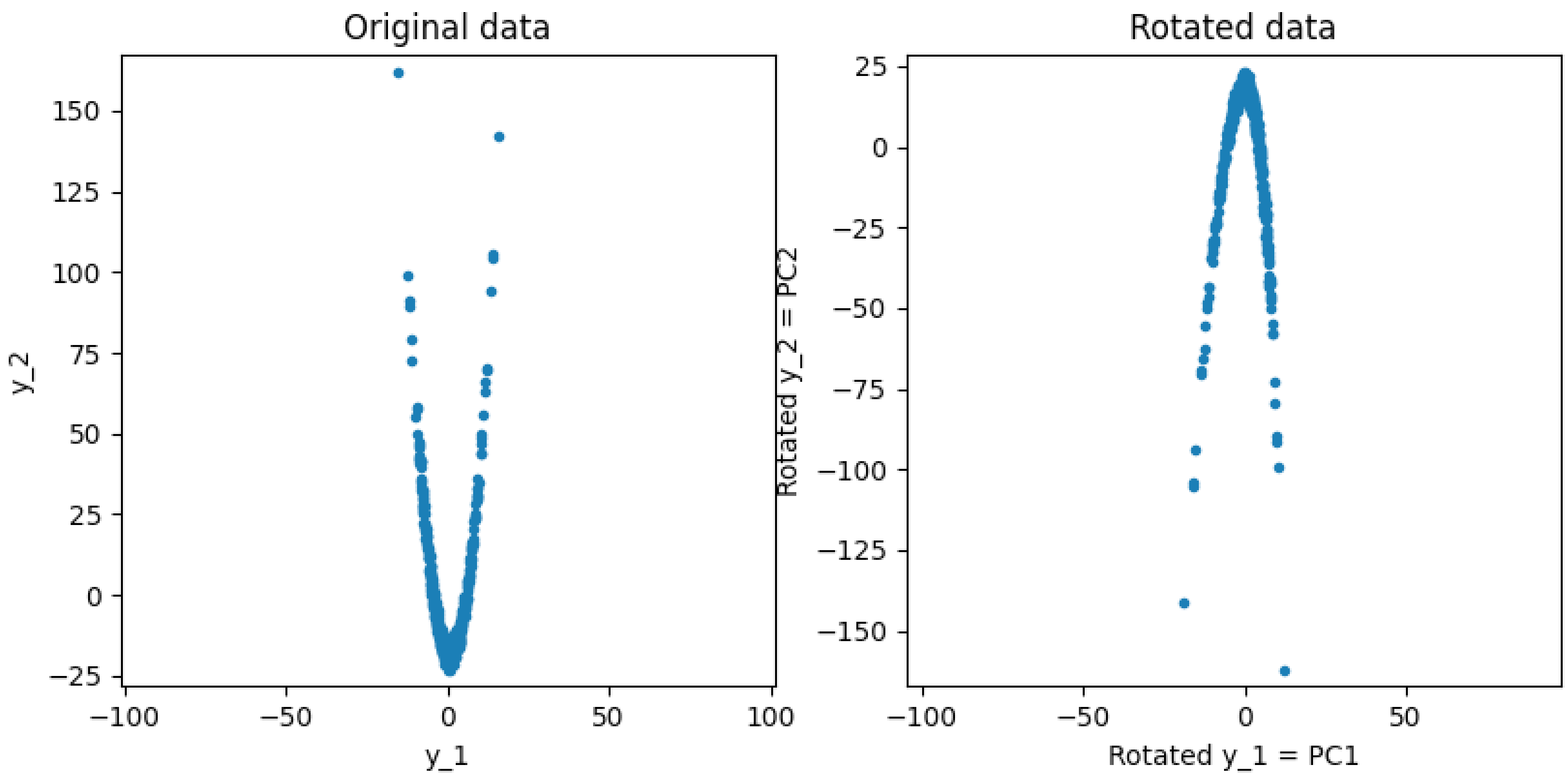


Principal component analysis

Data analysis in the rotated space using the eigenvectors of the covariance matrix

```
# Creating dataset with non-linear feature dependencies
x = np.random.normal(0, 5, 500)
noise = np.random.normal(0, 2, 500)
y = 0.7 * x**2 - x + noise
y -= np.mean(y)
data = np.array([x, y]).T
```

```
Covariance of original data:
[[ 24.39693751 -22.91257623]
 [-22.91257623 673.74145237]]
Eigenvalues:
[ 23.58945533 674.54893455]

Eigenvectors:
[[-0.99937958 -0.03522001]
 [ 0.03522001 -0.99937958]]

Covariance of y rotated:
[[ 2.35894553e+01 -1.27584427e-14]
 [-1.27584427e-14  6.74548935e+02]]
```



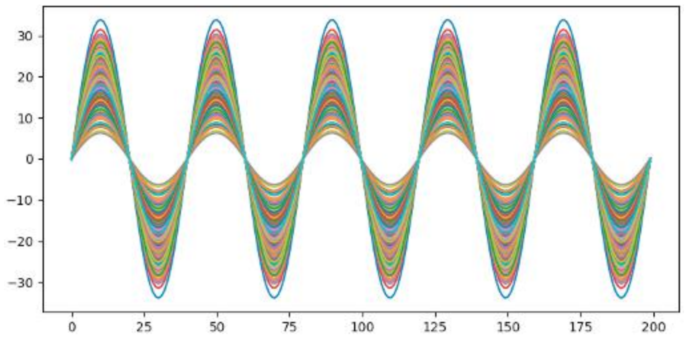
PCA is just a rotation…

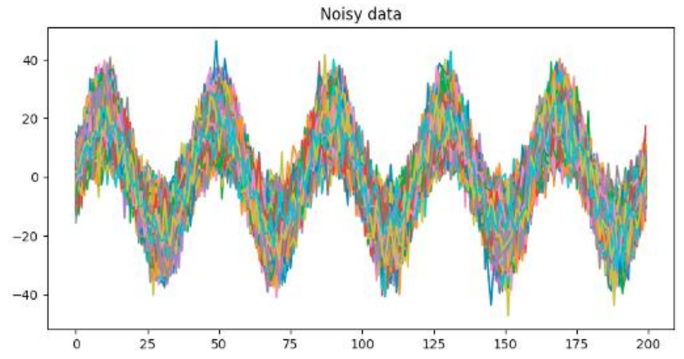
Principal component analysis

… but a very powerful rotation!

Example of noise elimination via dimensionality reduction

```
nSamples = 400
nFeatures = 200

time = np.linspace(0, 1, nFeatures).reshape(1,-1)
temporalSamples = np.repeat((2 * np.pi * time * 5), nSamples, axis=0)
magnitude = np.random.normal(20, 5, nSamples).reshape(-1, 1)
data = magnitude * np.sin(temporalSamples)

noisyData = data + np.random.normal(0, 5, (nSamples, nFeatures))

meanData = np.mean(noisyData, axis=0, keepdims=True)
covariance = np.cov((noisyData-meanData).T)

eigenValues, eigenVectors = np.linalg.eig(covariance)
eigenVectors = eigenVectors.T

rotatedData = (noisyData - meanData) @ eigenVectors.T
reconstructedData = rotatedData[:, :1] @ eigenVectors[:1, :] + meanData
```

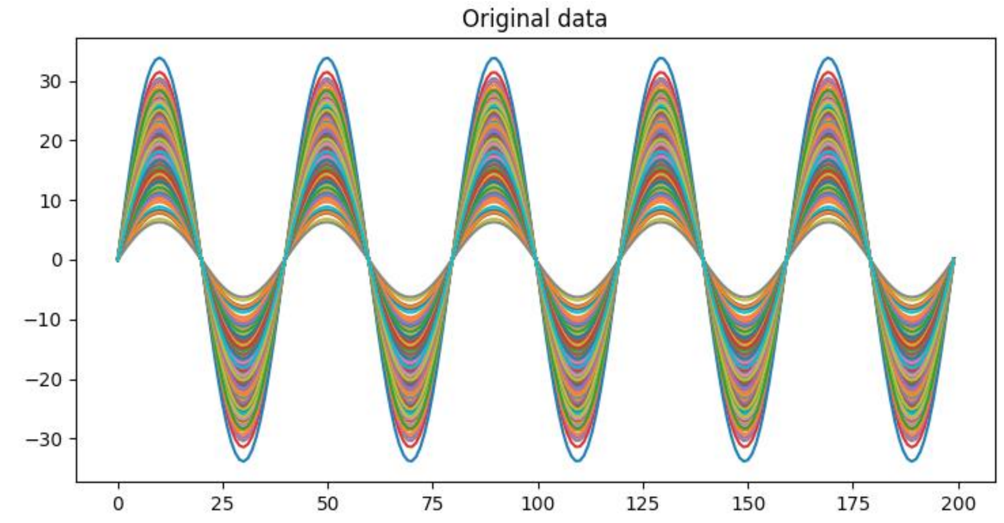


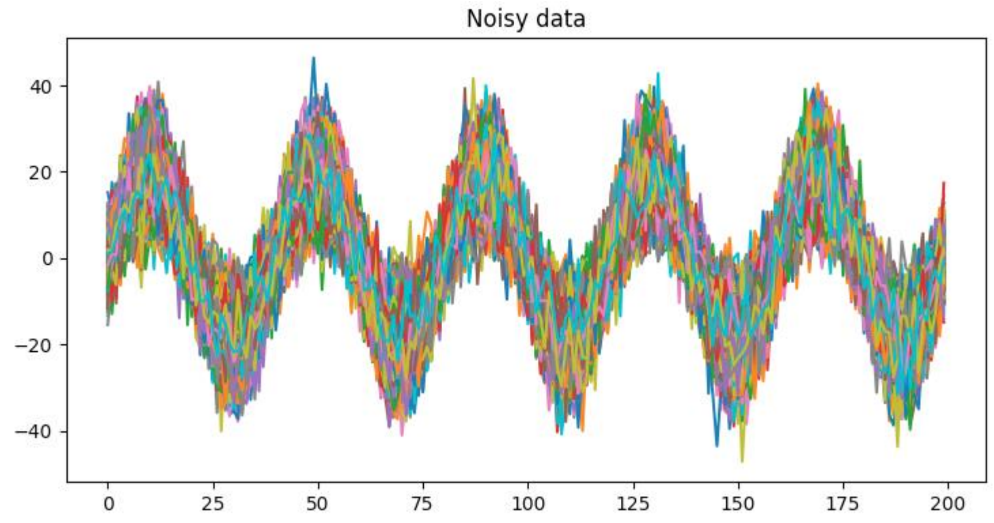


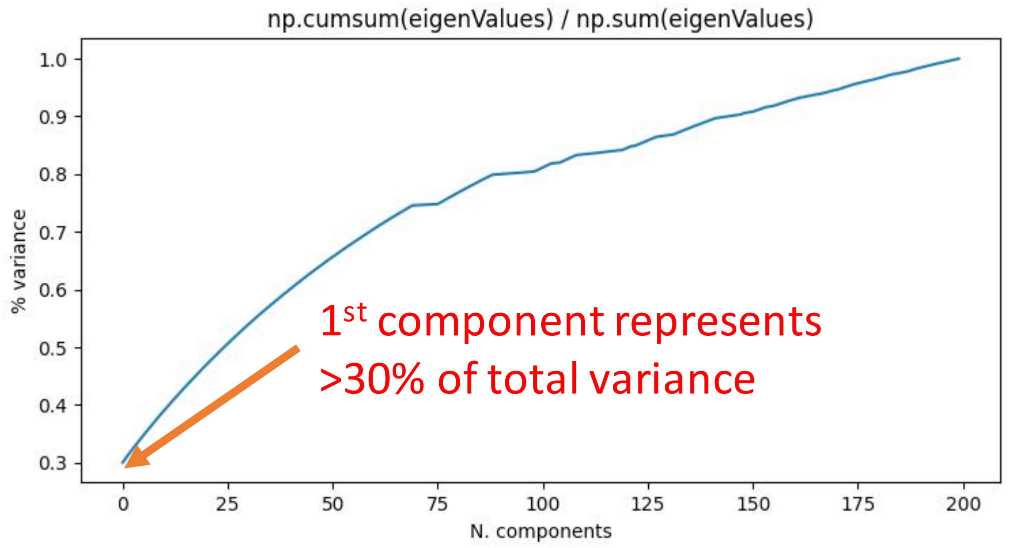
Remember that the data needs to be centered at the origin of coordinates

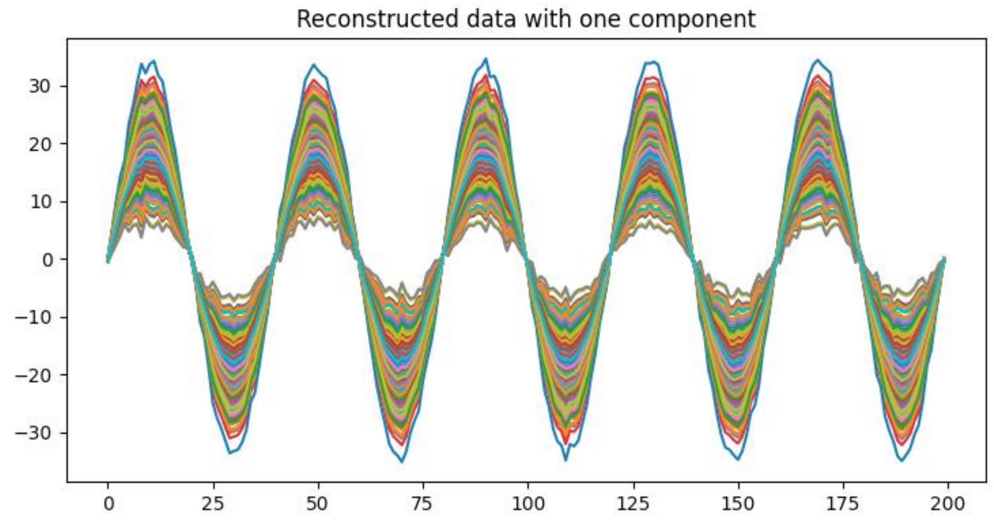
Principal component analysis

Example of noise elimination via dimensionality reduction



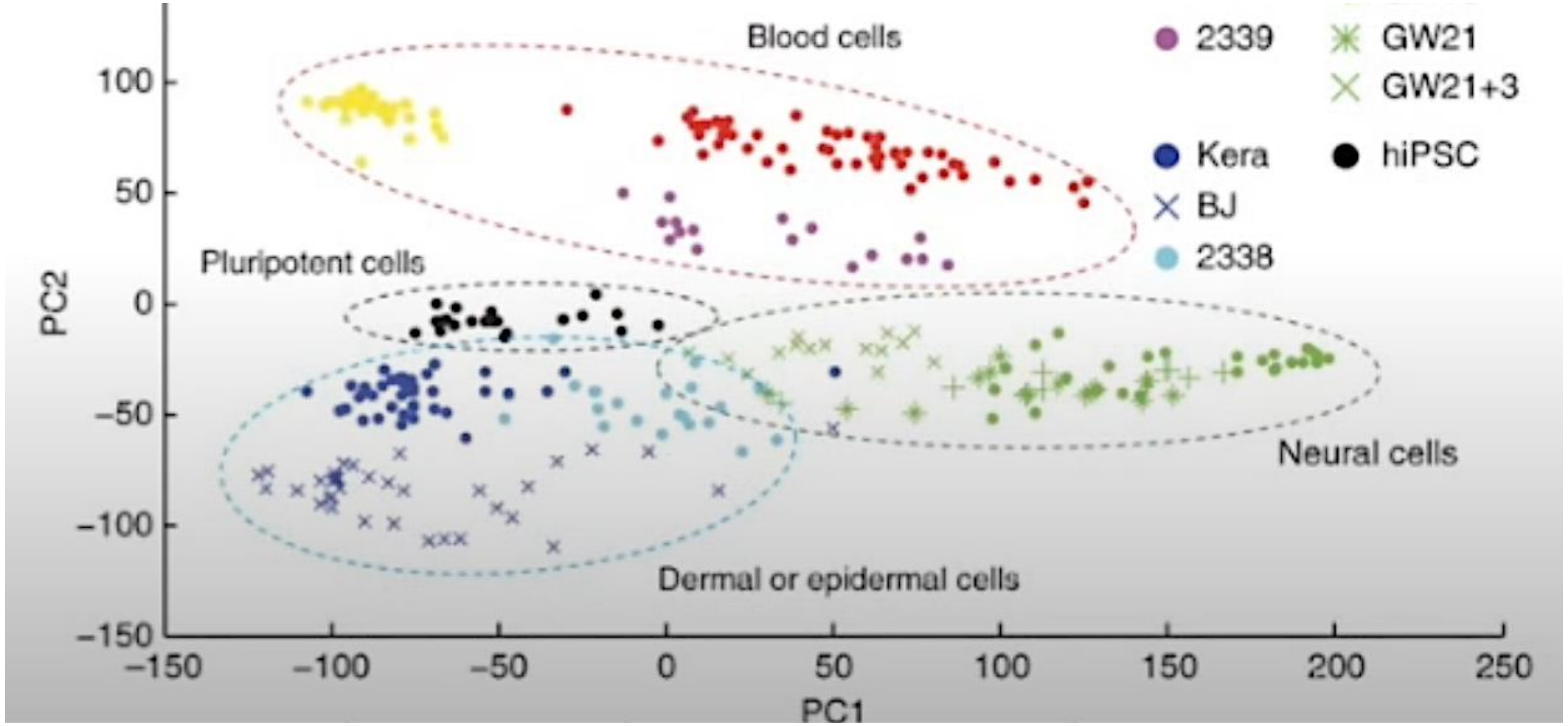






Principal component analysis

Dimensionality reduction is RNA sequencing data
(every cell is represented by ~10,000 transcribed genes)

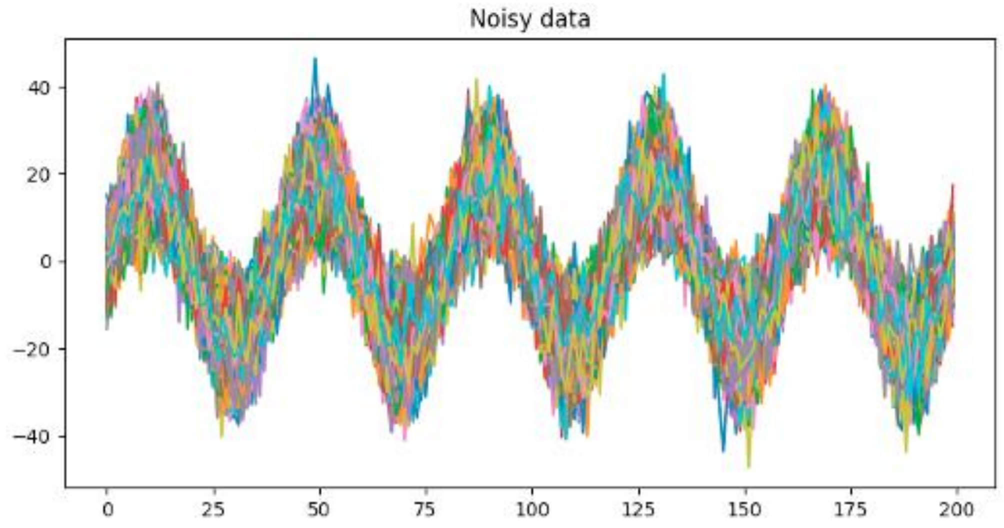


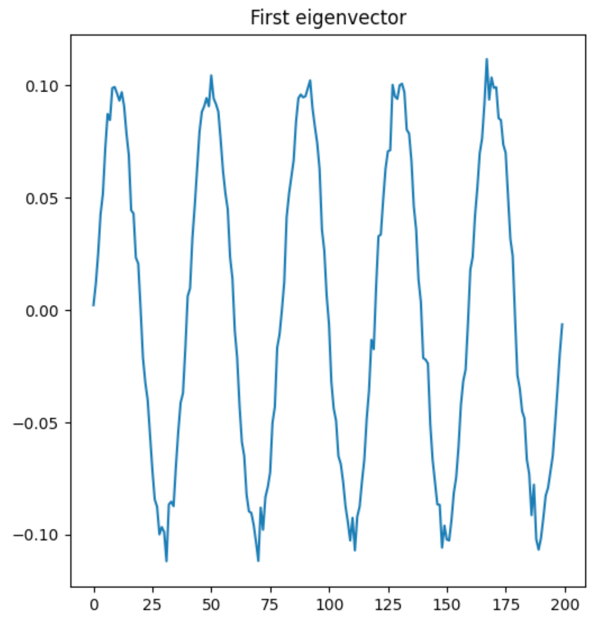
[Pollen et al., Nature Biotechnology, 2014]

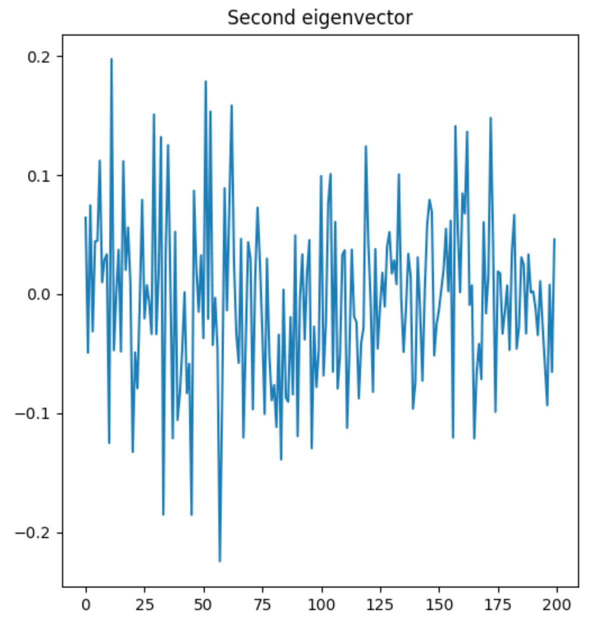
Note: not observing a specific structure in the first principal components does not mean that there is no structure. After all, PCA is only a linear rotation

Principal component analysis

The eigenvectors can be interpreted visually



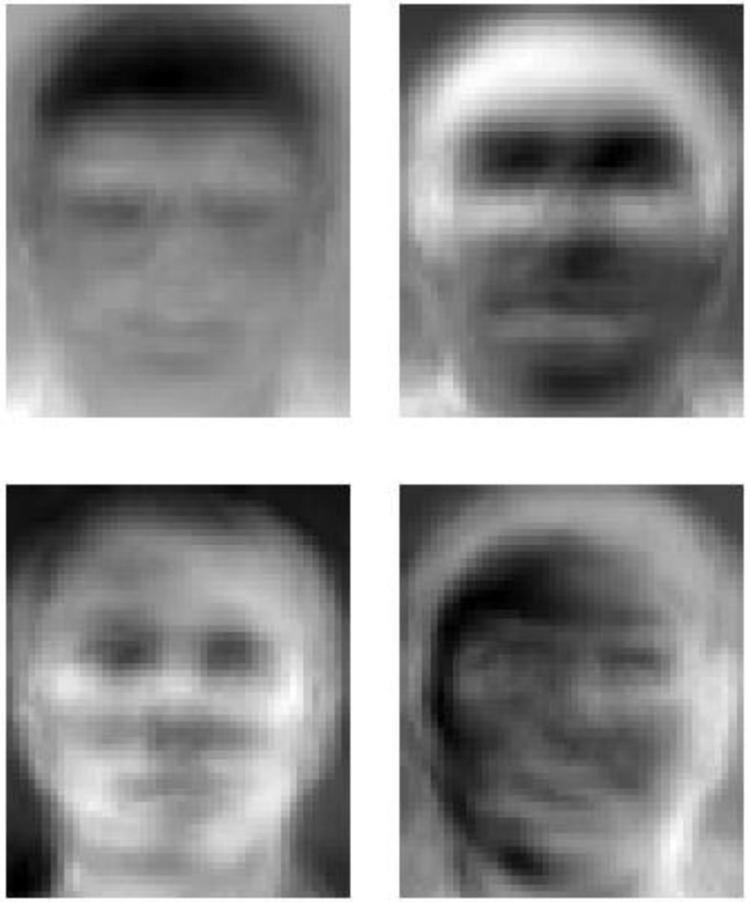




Principal component analysis

The eigenvectors can be interpreted visually

Eigenfaces

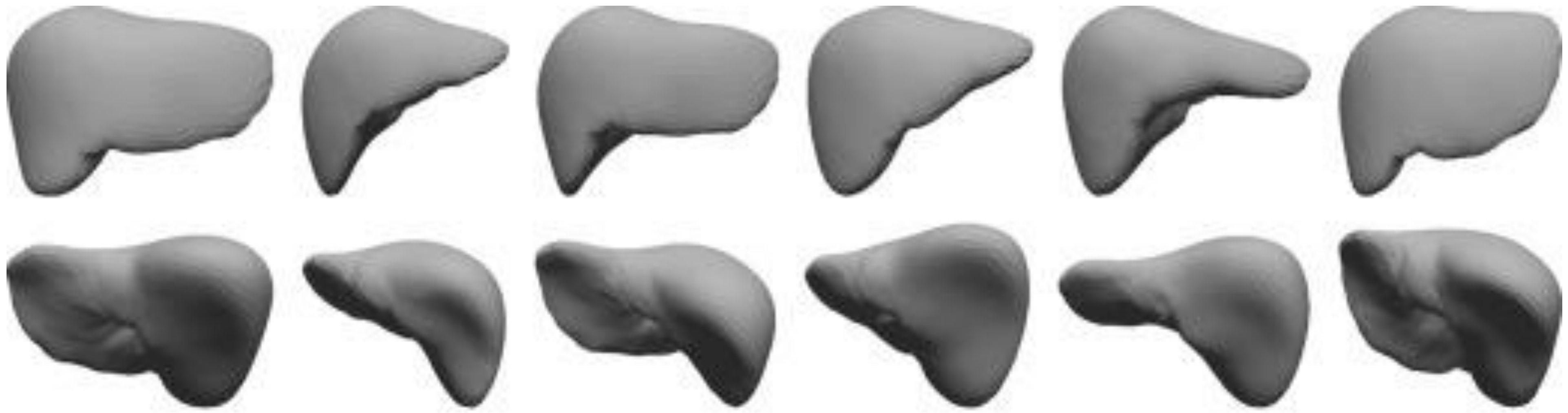


•*M. Kirby; L. Sirovich (1990). "Application of the Karhunen-Loeve procedure for the characterization of human faces". IEEE Transactions on Pattern Analysis and Machine Intelligence.* ***12*** *(1): 103–108. doi:10.1109/34.41390.*

Principal component analysis

- Since PCA features are orthogonal and normally distributed, and the eigenvalues represent the variance, PCA can be used to generate synthetic data

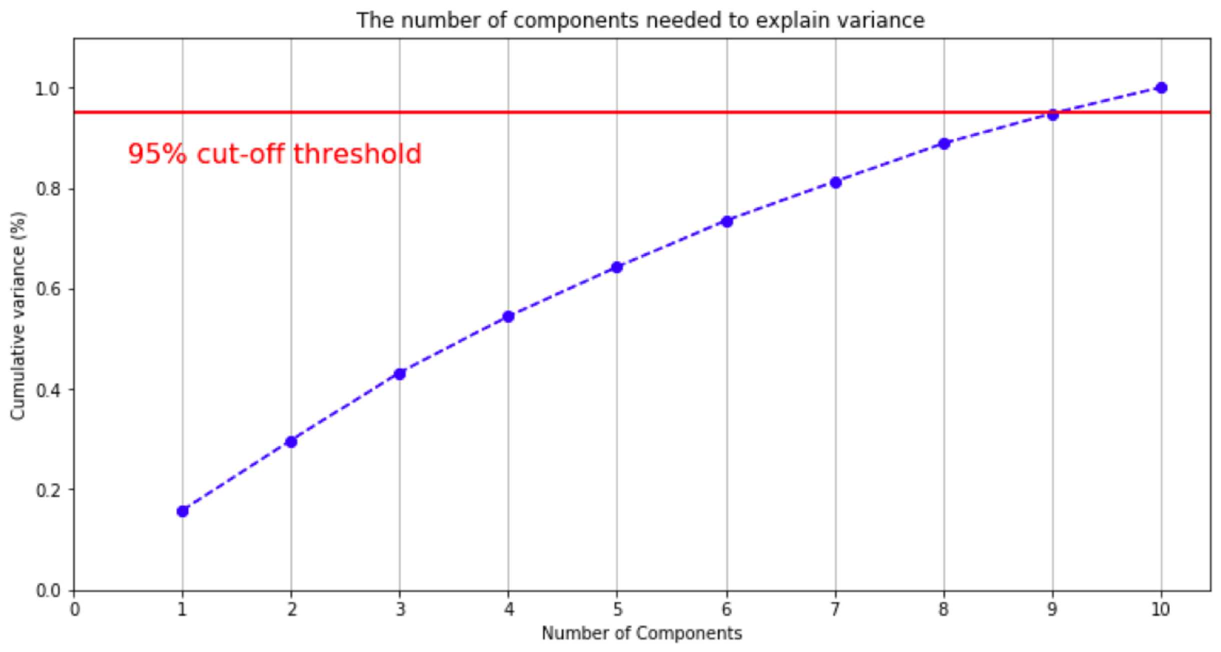
$$y = \bar{x} + p\Phi^T \ , p \in N(0, \Lambda)$$



[S. Kevin Zhou, D. Xu, Chapter 8 - A Probabilistic Framework for Multiple Organ Segmentation Using Learning Methods and Level Sets, The Elsevier and MICCAI Society Book Series, Medical Image Recognition, Segmentation and Parsing, Academic Press, 2016]

Principal component analysis

- Selecting number of components:
  - Thresholds of between 80% and 98% are common but there is no golden rule.

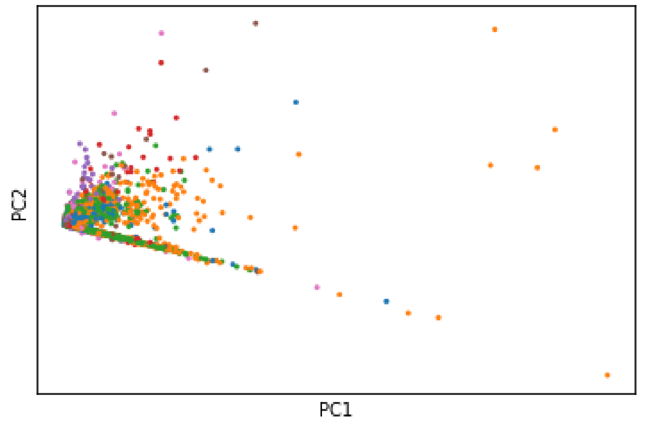


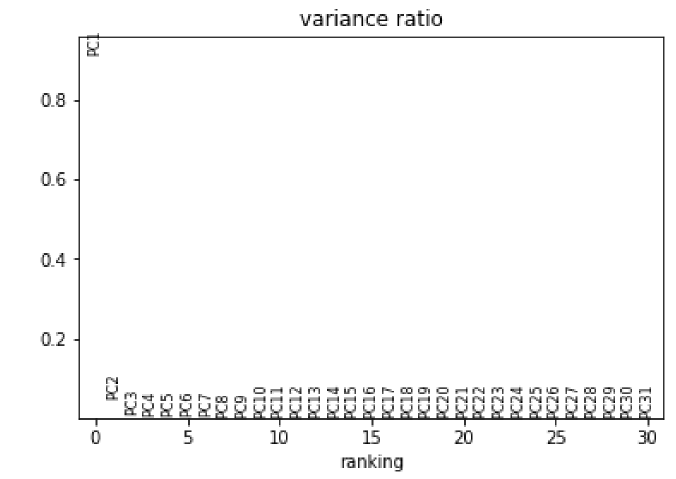
- Component selection is also important for applications such as modeling: The last few components represent data noise and tend to distort the generated data.

Principal component analysis

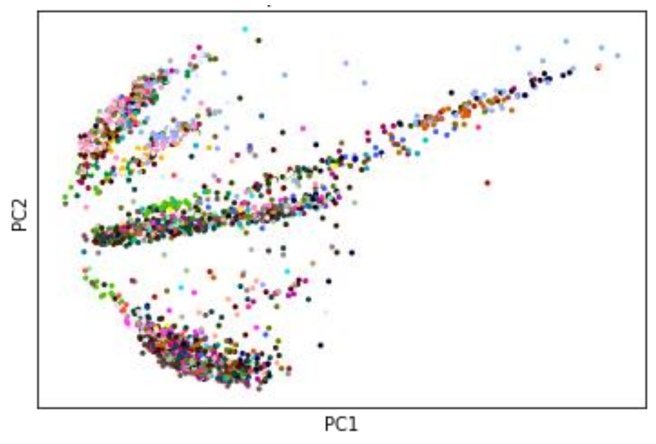
- ❑ Feature normalization is essential for adequate modeling.
  - Example: single cell RNA sequencing data to identify tissue types in mice

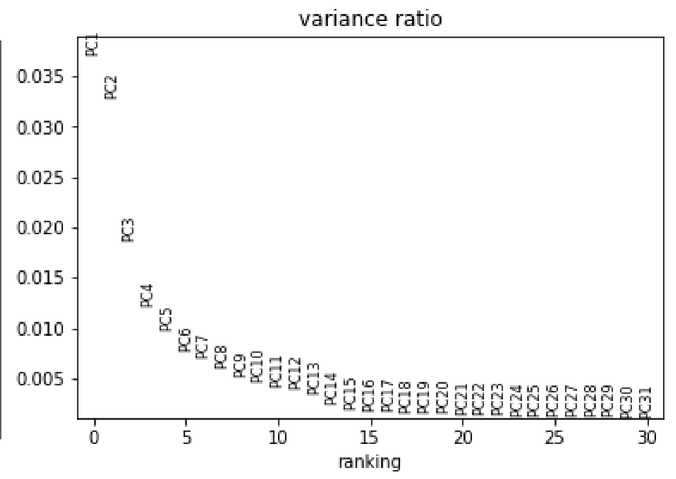
Before normalization





After normalization





https://tabula-muris-senis.ds.czbiohub.org

Principal component analysis

- Limitations
  - Not useful for categorical variables
  - Dimensionality reduction comes at the expense of accuracy
  - Variance is often not equivalent to feature importance
  - Assumes normal data distributions
    - Check: Generalized PCA
  - Feature distributions are often not orthogonal
    - Check: Independent component analysis
  - Feature correlations are often non-linear
    - Check: Kernel PCA (remember the kernel trick?)

Linear discriminant analysis (LDA)

- Supervised dimensionality reduction
  - The outcome of the machine learning task is known in the training dataset and it can be leveraged to extract a lower dimensional representation
  - Instead of finding a lower-dimensional representation that maximizes data variance, we seek the maximization of class separability
- Linear discriminant analysis (LDA)

…or Fisher’s linear discriminant

  - Goal: find the **linear** combination of features that maximizes the separability between different classes
  - Approach: maximization of between-class variance and minimization of within-class variance

Linear discriminant analysis (LDA)

- Linear Discriminant Analysis

  - Find a linear combination of features ($\boldsymbol{x}$): $y = w^T \boldsymbol{x}$

  - Objective: $\max \left(\frac{S_b}{S_{w_1} + S_{w_1}} \right)$

Between-class variance

Within-class variance

Between class variance:

Original space: $S_b = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$

Transformed space: $S_b^* = (w^T \mu_1 - w^T \mu_2)(w^T \mu_1 - w^T \mu_2)^T = w^T(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T w = w^T S_b w$

Within class variance:

Original space: $S_{w_i} = \sum_{\forall x_j \in l_i} (x_j - \mu_j)(x_j - \mu_j)^T$

Transformed space: $S_{w_i}^* = \sum_{\forall x_j \in l_i} (w^T x_j - w^T \mu_j)(w^T x_j - w^T \mu_j)^T = \sum_{\forall x_j \in l_i} w^T (x_j - \mu_j)(x_j - \mu_j)^T w = \sum_{\forall x_j \in l_i} w^T S_{w_i} w$

Linear discriminant analysis (LDA)

- Fisher's criterion:

$$J(w) = \frac{w^T S_b w}{w^T S_w w}$$

$$w = \arg\max\big(J(w)\big)$$

- Solving for w

$$\frac{dJ(w)}{dw} = S_w^{-1} S_b w - J(w)w = 0$$

$$\boxed{S_w^{-1} S_b w = J(w)w}$$

Generalized eigenvalue problem

$$\boldsymbol{w} = \boldsymbol{S_w^{-1}}(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})$$

<u>Fisher's discriminant</u>: not really a discriminant but a direction for data projections

Linear discriminant analysis (LDA)

- LDA generalizes to separating *C* classes using *C-1* projections

$$w \rightarrow W = [w_1|w_2 \dots |w_{C-1}]$$

$$y = W^T x$$

$$S_w = \sum_{i=1}^{C-1} S_{w_i}$$

$$S_b = \sum_{i=1}^{C-1} (\mu_1 - \mu)(\mu_i - \mu)^T$$

$$J(w) = \frac{W^T S_b W}{W^T S_w W} \rightarrow S_w^{-1} S_b w_i = \lambda_i w$$

w_i are the eigenvectors with the highest eigenvalues

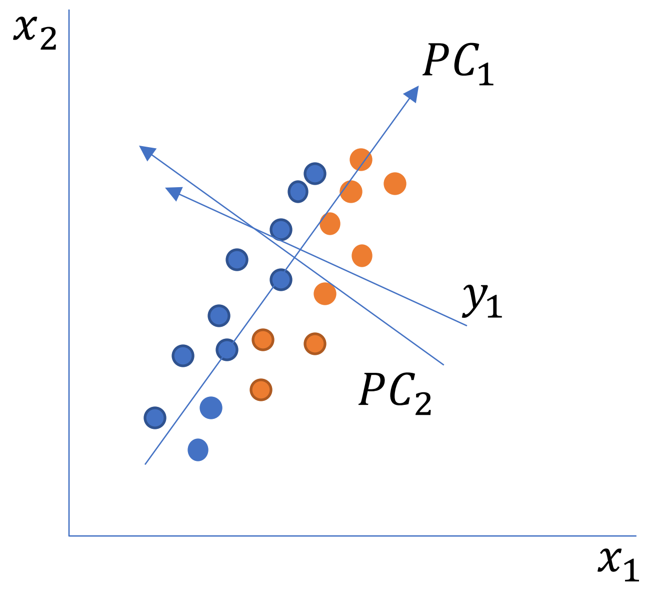
Linear discriminant analysis (LDA)

❑ Limitations

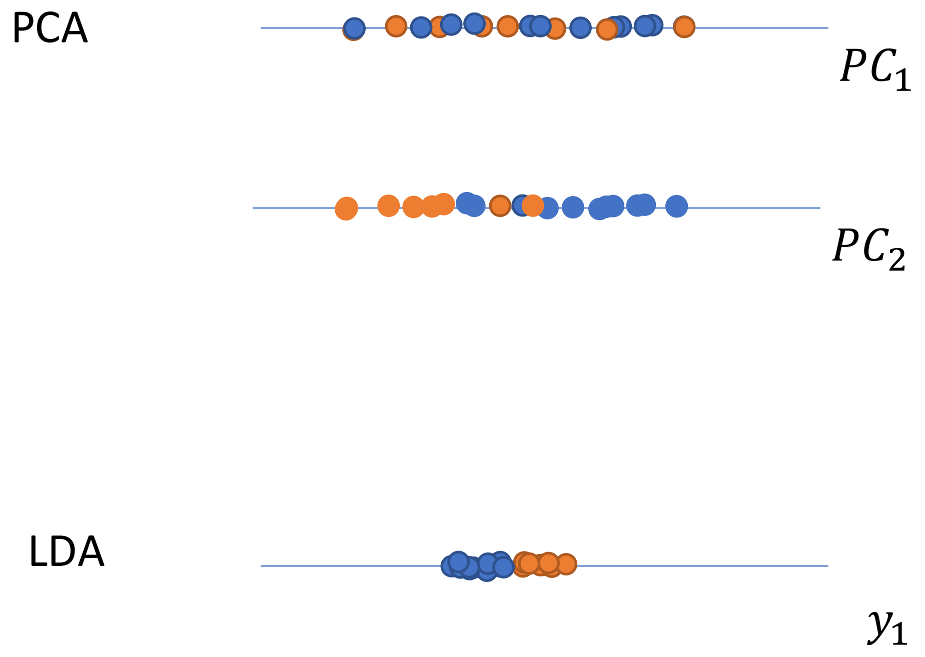
- Assumes linearly separable classes
    - Remember the kernel trick? Check: Generalized or kernel discriminant analysis
- Homoscedasticity: uniform variances
- Assumes normal distributions
- The maximum number of projections is limited by the number of classes

LDA vs PCA

- LDA vs PCA



x_2
PC_1
y_1
PC_2
x_1



PCA
PC_1
PC_2
LDA
y_1