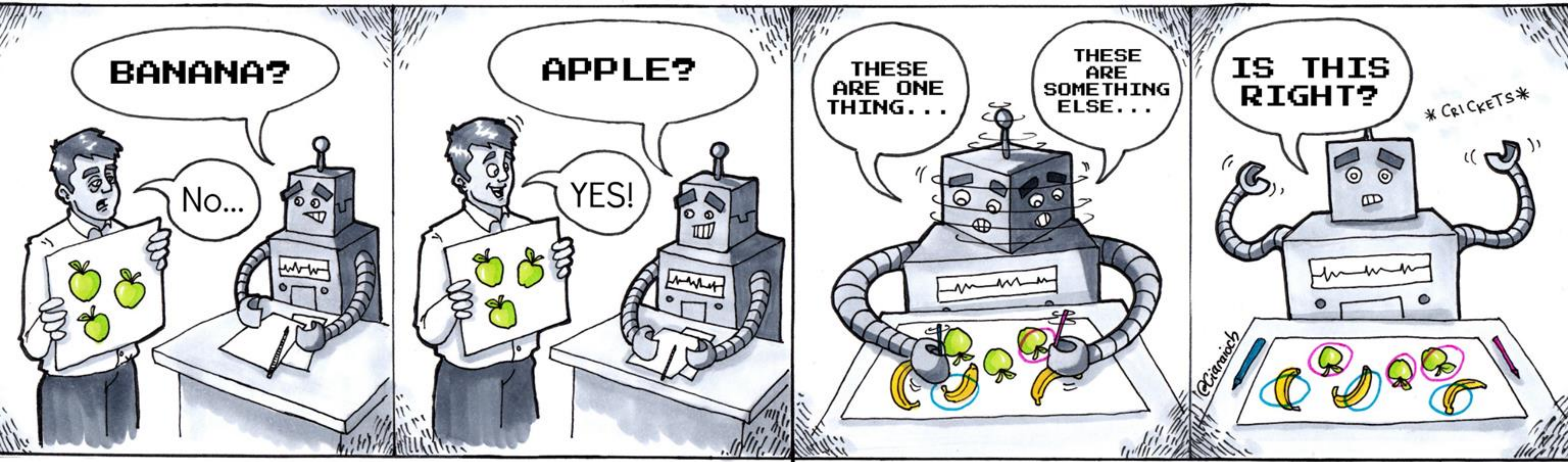
CSC 580 Principles of Machine Learning

Unsupervised learning

Chicheng Zhang

Department of Computer Science

THE UNIVERSITY OF ARIZONA



Supervised Learning

Unsupervised Learning

What is unsupervised learning?

- Uncovering structures in unlabeled data
- What can we expect to learn?
  - **Clustering**: obtain partition of the data that are well-separated.
    - can be viewed as a preliminary classification without predefined class labels.
  - **Component analysis**: extract common components that compose data points.
    - e.g., topic modeling given a set of articles: each article talks about a few topics => extract the set of topics that appears frequently.
- Usage
  - As a summary of the data
    - **Exploratory data analysis**: what are the **patterns** we can get even without labels?
  - Often used as ‘preprocessing techniques’
    - e.g., extract useful **representation** using principal component analysis (will be covered later)

Outline

- Clustering
  - K-means clustering revisited
  - Hierarchical clustering

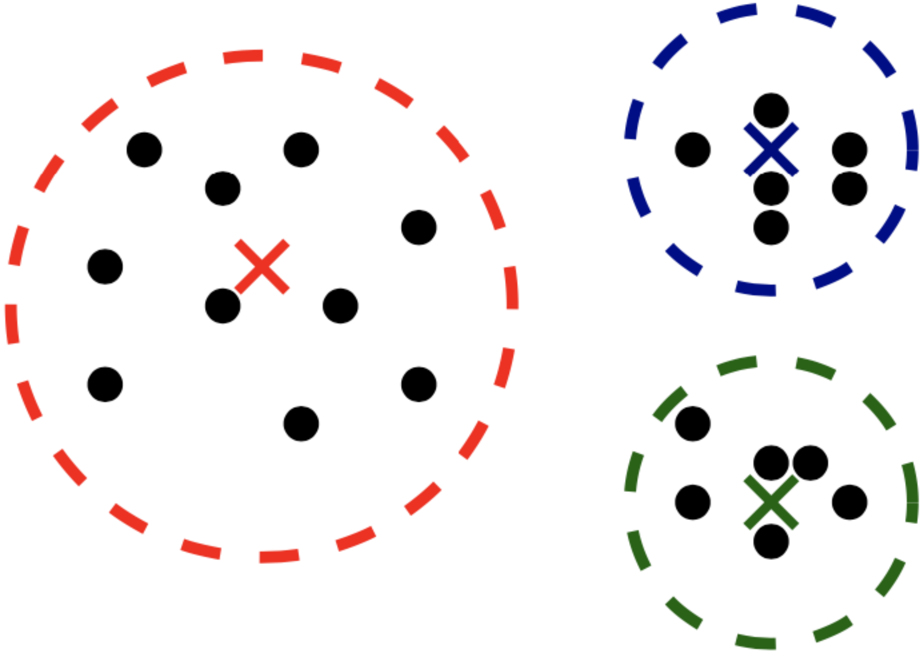
- Principal Component Analysis (PCA)

Outline

- Clustering
    - **K-means clustering revisited**
    - Hierarchical clustering

- Principal Component Analysis (PCA)

Clustering



- Input: k: the number of clusters (hyperparameter)

  $$S = \{x_1, \ldots, x_n\}$$

- Output
  - partition $\{G_i\}_{i=1}^{k}$ s.t. $S = \cup_i G_i$ (disjoint union).
  - often, we also obtain 'centroids'

- Recall: in k-means clustering, how did we define centroid of a cluster G?

- Answer: average point $\frac{\sum_{x \in G} x}{|G|}$

Recap: K-means clustering [Lloyd’82]

Input: k: num. of clusters, $S = \{x_1, \ldots, x_n\}$

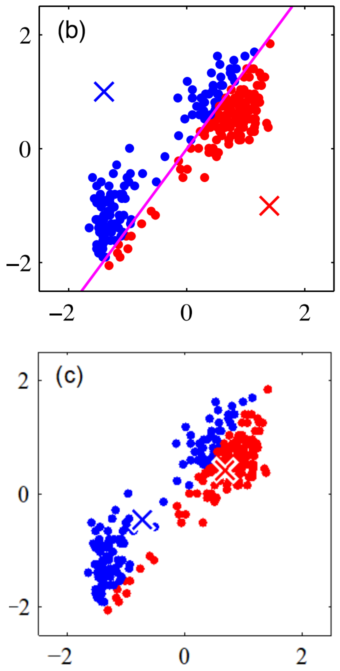
[Initialize] Pick $c_1, \ldots, c_k$ as randomly selected points from S (see next slides for alternatives)

Notation: $[k] = \{1,2, \ldots, k\}$

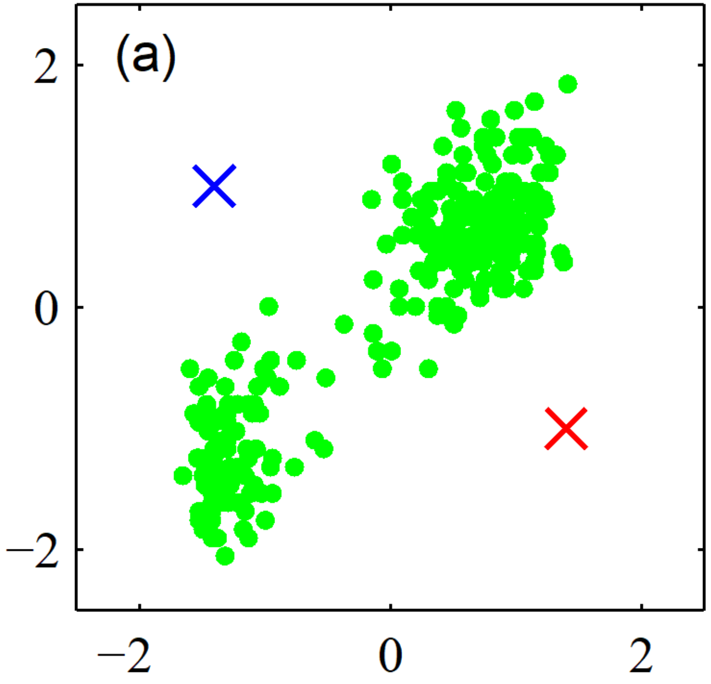
```
For $t = 1,2, \ldots, T$:
    • [Update cluster assignments]   $\forall x \in S, \quad z_t(x) = \arg\min_{j \in [k]} \left\| x - c_j \right\|_2$
    • [Update centroids]   $\forall j \in [k], \quad c_j \leftarrow \text{average}(\{x \in S: z_t(x) = j\})$

    • If t $\neq 1$ AND $z_t(x) = z_{t-1}(x), \forall x \in S$
        break
```

Output: $c_1, \ldots, c_k$ and $\{z_t(x_i)\}_{i \in [n]}$

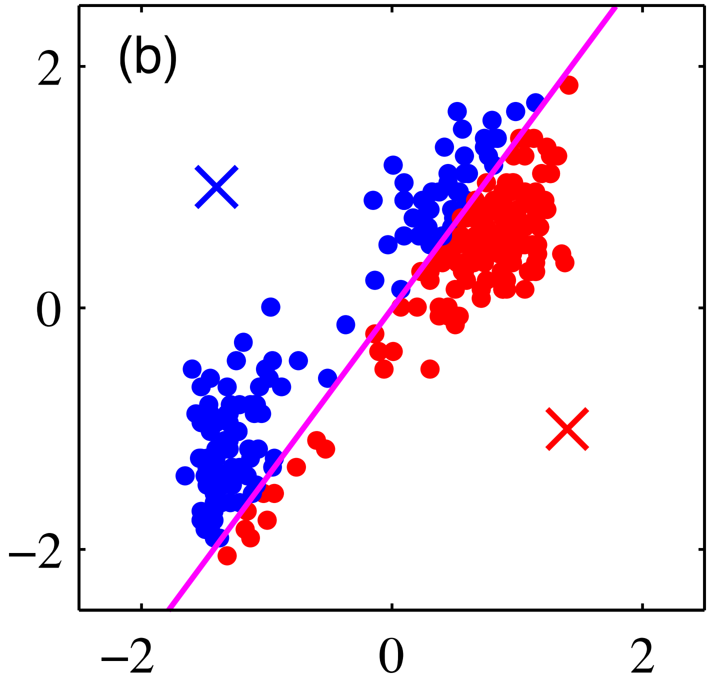


Initialization

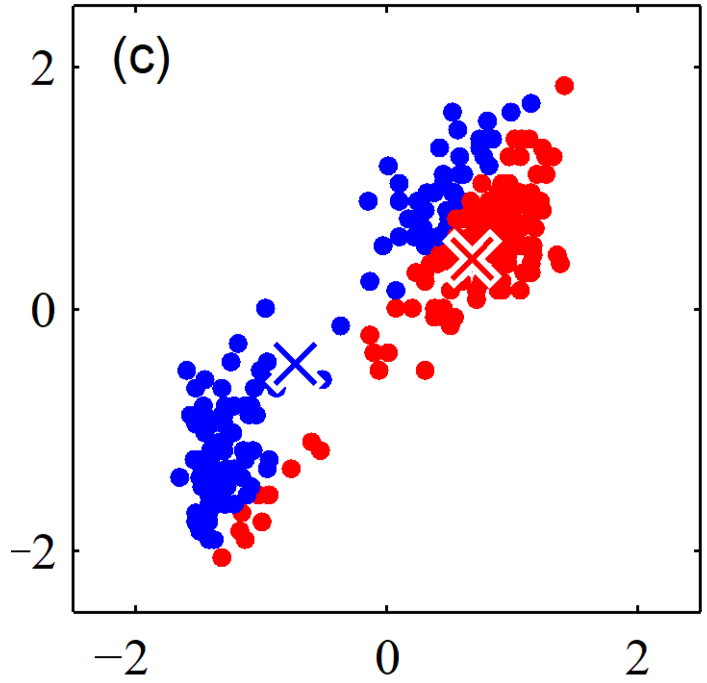


Arbitrary/random initialization of c_1 and c_2

Iteration 1

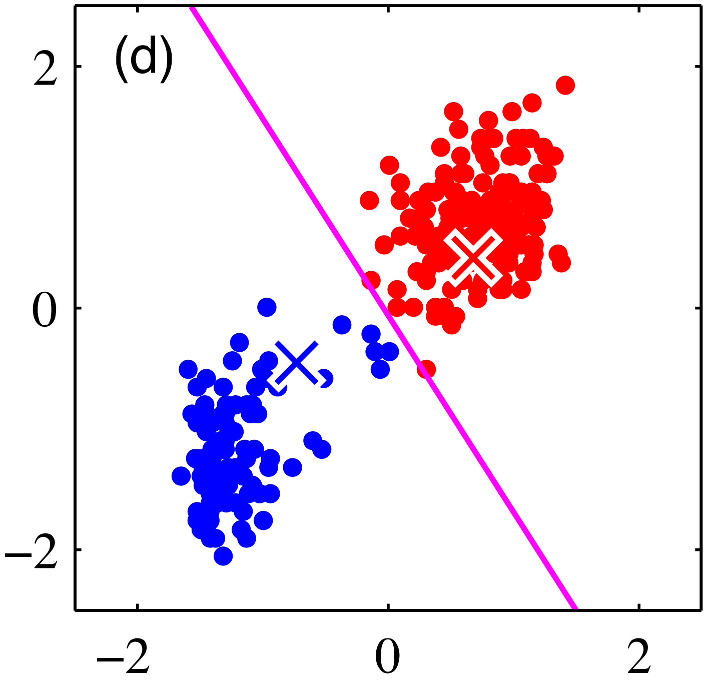


(A) update the cluster assignments.

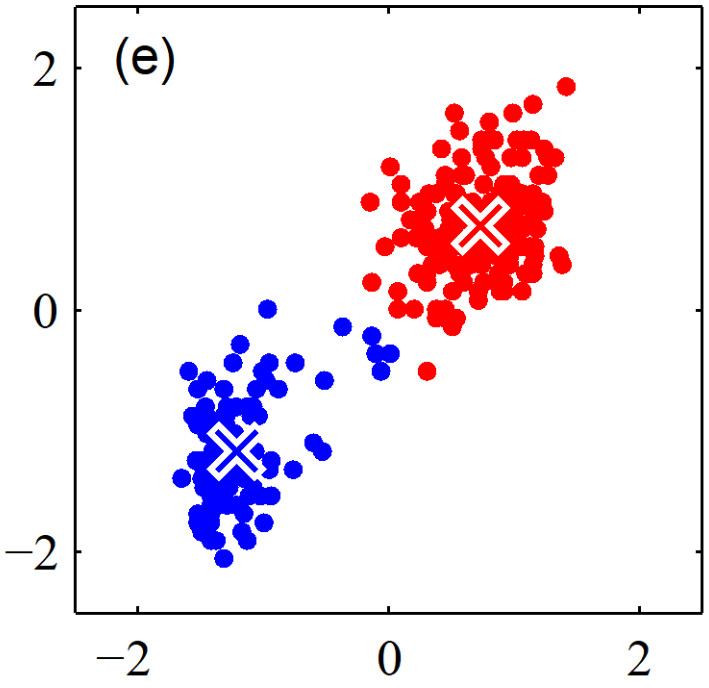


(B) Update the centroids $\{c_j\}$

Iteration 2

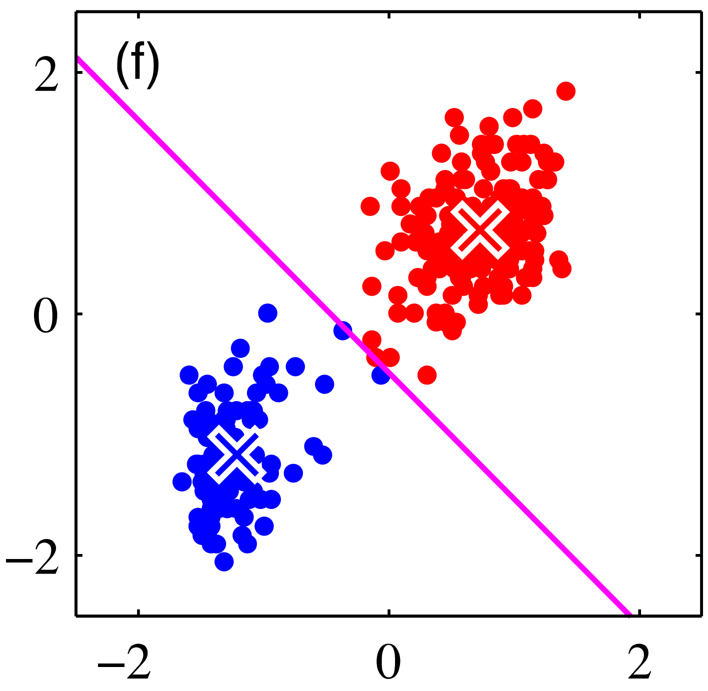


(A) update the cluster assignments.

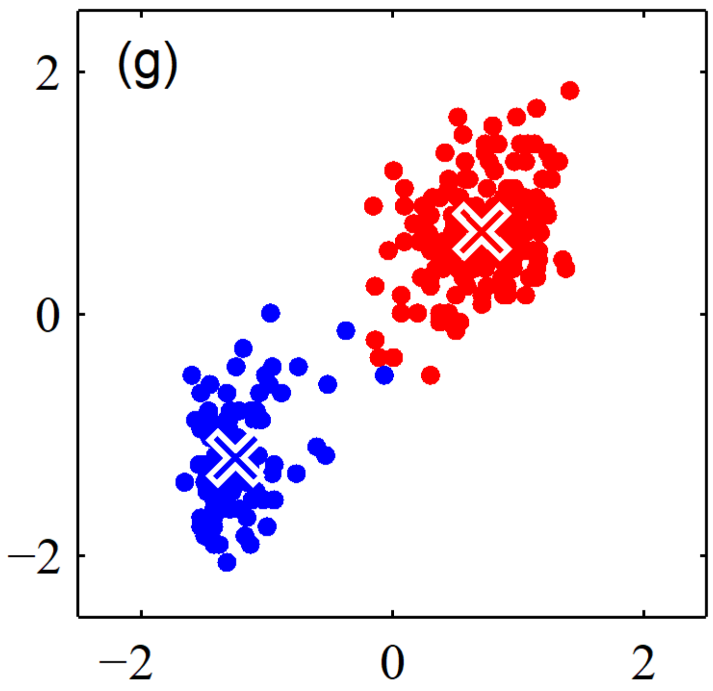


(B) Update the centroids $\{c_j\}$

Iteration 3

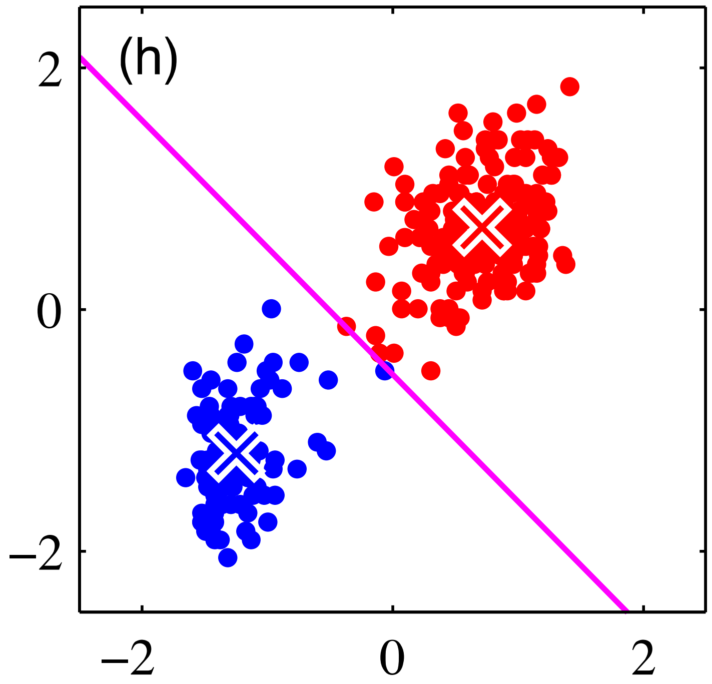


(A) update the cluster assignments.

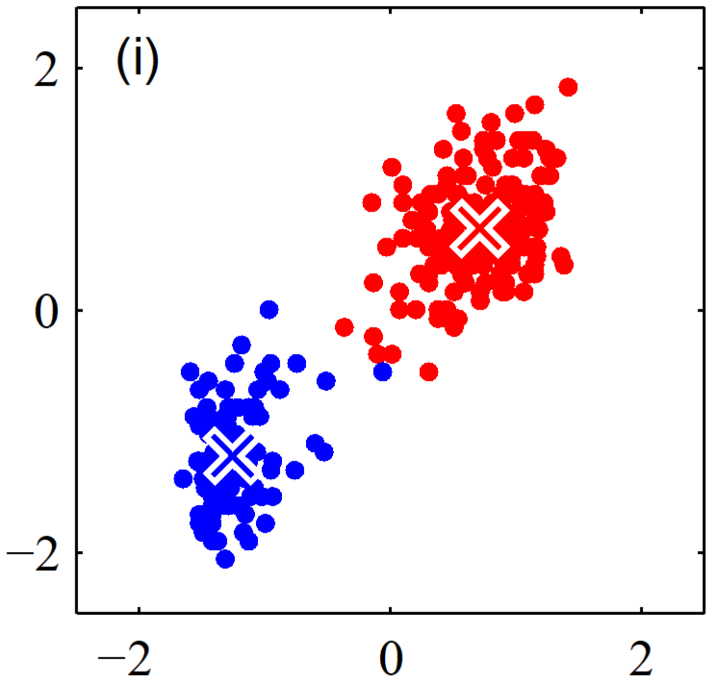


(B) Update the centroids $\{c_j\}$

Iteration 4



(A) update the cluster assignments.



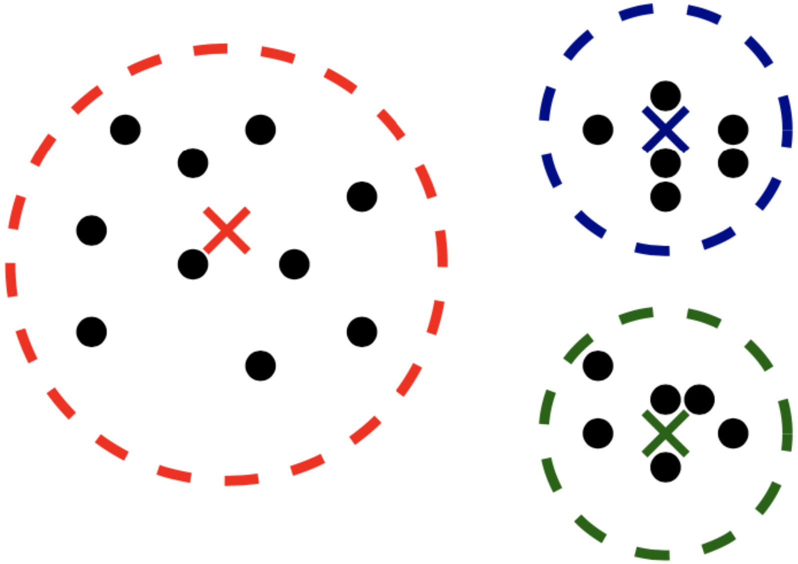
(B) Update the centroids $\{c_j\}$

Clustering as compression

- Clustering can be viewed as a way to do *compression*
  - Original dataset $x_1, \dots, x_n$
  - Compressed representation: $\vec{c} = (c_1, \dots, c_k)$, $\vec{z} = (z_1, \dots, z_n)$

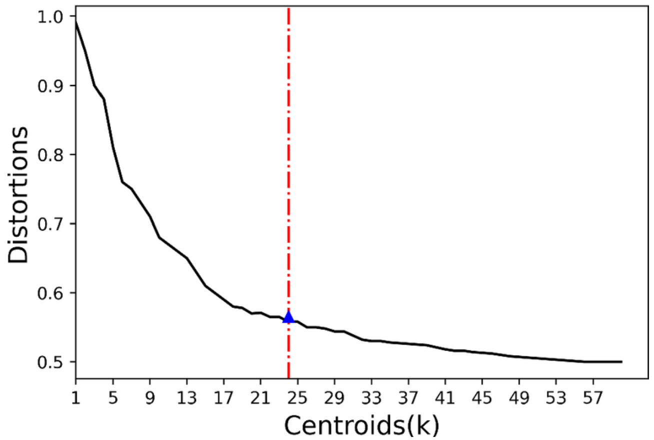
cluster centroids; each $c_i \in \mathbb{R}^d$

cluster assignments; each $z_i \in \{1, .., k\}$



  - Reconstructed dataset: $c_{z_1}, \dots, c_{z_n}$
  - Is the compression *lossless* or *lossy*?
  - Sometimes call clustering “quantization” -- rate-distortion theory in information theory

- Reconstruction error for example x_i: $\left\|c_{z_i} - x_i\right\|_2^2$
  - Total reconstruction error: $g(\vec{c}, \vec{z}) = \sum_{i=1}^{n}\left\|c_{z_i} - x_i\right\|_2^2$



K-means: cost minimization perspective

- Lloyd's algorithm minimizes reconstruction error by *alternating minimization*
- For variables

cluster centroids; each $c_i \in \mathbb{R}^d$ cluster assignments; each $z_i \in \{1, . . , k\}$

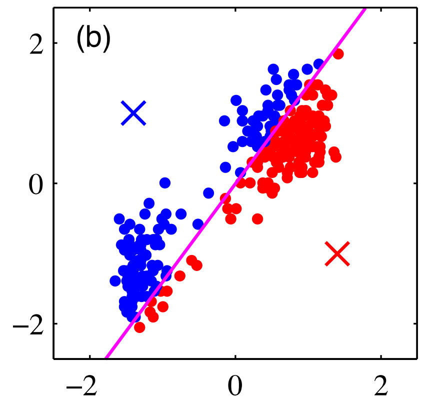
$\vec{c} := (c_1, \dots, c_k)$, $\vec{z} := (z_1, \dots, z_n)$

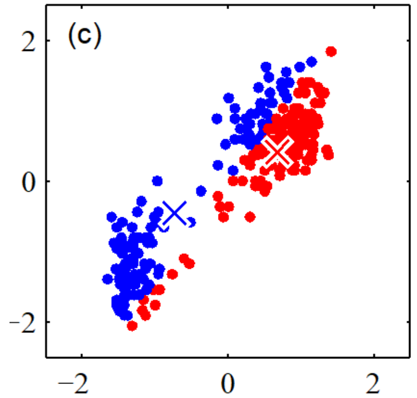
- Solve the following optimization problem:

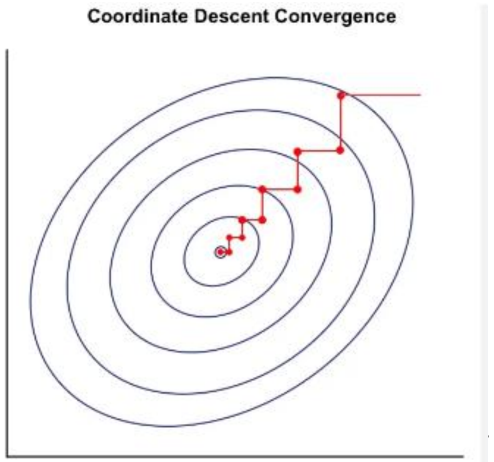
$$\min_{\vec{c},\vec{z}} g(\vec{c}, \vec{z}) := \sum_{i=1}^{n} \left\| x_i - c_{z_i} \right\|_2^2$$

- Lloyd's algorithm:
- For $t = 1,2, \dots, T$:
  - ***[Update cluster assignments]*** $\vec{z}_t \leftarrow \operatorname{argmin}_{\vec{z}}\, g(\vec{c}_{t-1}, \vec{z})$
  - ***[Update centroids]*** $\vec{c}_t \leftarrow \operatorname{argmin}_{\vec{c}}\, g(\vec{c}, \vec{z}_t)$

- Observation: objective function $g(\vec{c}_t, \vec{z}_t)$ decreases *monotonically in t*

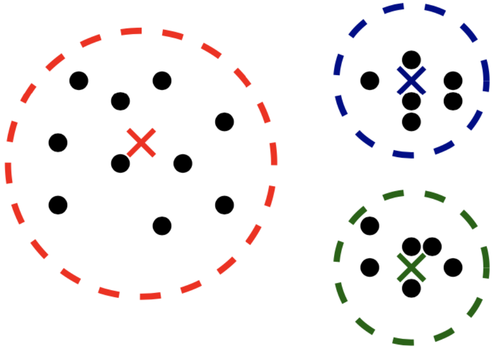






K-means: cost minimization perspective

- Observation: given any $\vec{c}$, the $\vec{z}$ that minimizes $g(\vec{c},\vec{z}) \coloneqq \sum_{i=1}^{n} \left\| x_i - c_{z_i} \right\|_2^2$ satisfies that
  - $z_i = \operatorname{argmin}_{j \in [k]} \left\| x_i - c_j \right\|_2$, for every i
  - which induces reconstruction error $\sum_{i=1}^{n} \min_{j \in [k]} \left\| x_i - c_j \right\|_2^2 =: f(\vec{c})$



- We can also view optimizing reconstruction error as just finding k “centers” $\vec{c} = (c_1, \ldots, c_k)$ that minimizes

$$f(\vec{c}) \coloneqq \sum_{i=1}^{n} \min_{j \in [k]} \left\| x_i - c_j \right\|_2^2$$

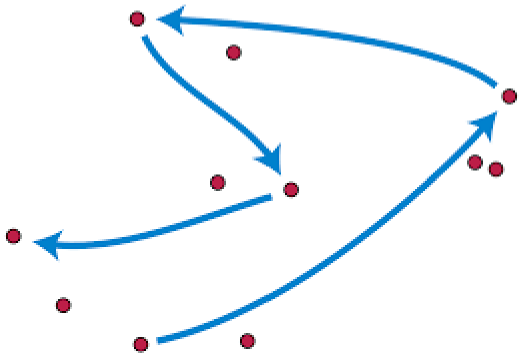
Issue 1: Unreliable solution

- You usually get different solutions every time you run.
- **Standard practice**: Run it 50 times and take the one that achieves the smallest objective function
  - Recall: $\underset{c_1,\ldots,c_k}{\text{minimize}}\, f(\vec{c})$, where $f(\vec{c}) = \sum_{i=1}^{n} \min_{j\in[k]} \left\| x - c_j \right\|_2^2$
- Or, change the initialization (next slide)
  - Idea: ensure that we pick a widespread $c_1, \ldots, c_k$

Two alternative initializations

- **Furthest-first traversal** ⇒ Sequentially choose c_j that are the farthest from the previously-chosen.
  - Pick $c_1 \in \{x_1, \ldots, x_n\}$ arbitrarily (or randomly)
  - For $j = 2, \ldots, k$
    - Pick $c_j \in \mathbb{R}^d$ as a point in $\{x_1, \ldots, x_n\}$ that maximizes the squared distances to $c_1, \ldots, c_{j-1}$.

$$c_j = \arg\max_{i \in [n]} \min_{j' \in [j-1]} \left\| x_i - c_{j'} \right\|_2$$



- ***k*-means++ (Arthur and Vassilvitskii, 2007)**
  - Pick $c_1 \in \{x_1, \ldots, x_n\}$ uniformly at random
  - For $j = 2, \ldots, k$
    - Define a distribution $\forall i \in [n],\ \mathbb{P}(c_j = x_i) \propto \min_{j' \in [j-1]} \| x_i - c_{j'} \|_2^2$
    - Draw c_j from the distribution above.

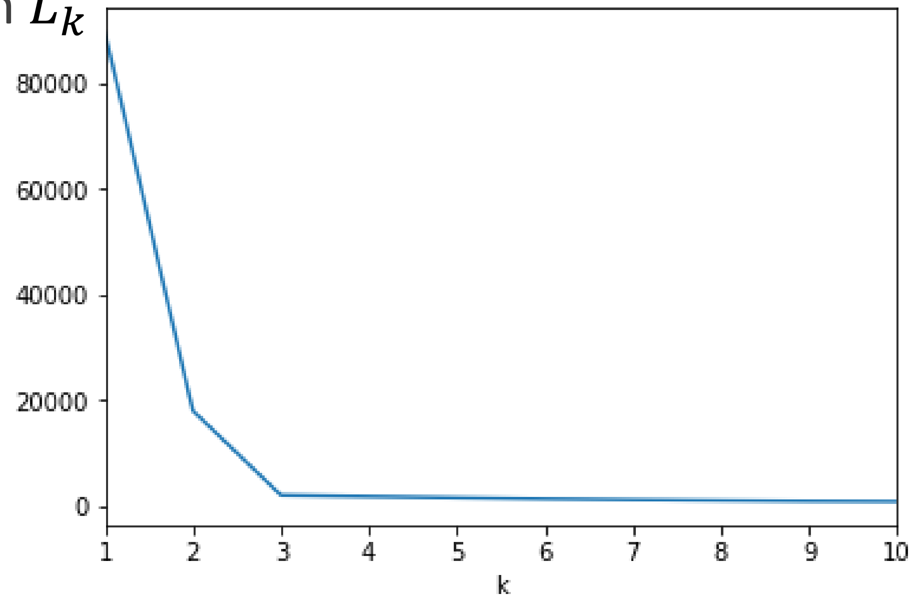
More likely to choose x_i that is farthest from already-chosen centroids.

=> has a mathematical guarantee that it will be better than an arbitrary starting point!

Issue 2: Choosing k

- $\hat{L}_k = f(c_1, \dots, c_k)$ for $c_1, \dots, c_k$ obtained by any k-means clustering algorithm

Objective function $\hat{L}_k$



- Elbow method: see where you get saturation.
- Akaike information criterion (AIC): $\mathrm{argmin}_k \left(\hat{L}_k + 2kd\right)$
- Bayesian information criterion (BIC): $\mathrm{argmin}_k \left(\hat{L}_k + kd \cdot \log n\right)$

https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb

Kernelizing K-means algorithm

How to perform clustering with feature transformations $\phi: \mathcal{X} \to \mathbb{R}^D$?

<u>Input</u>: k: num. of clusters, $S = \{x_1, \dots, x_n\}$, kernel function K with feature map ϕ

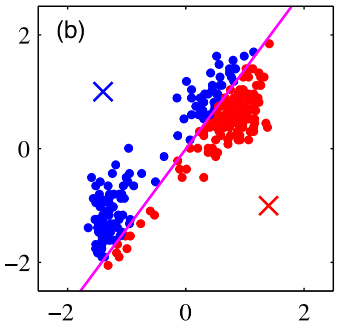
Idea: perform clustering over $\tilde{S} = \{\phi(x_1), \dots, \phi(x_n)\}$, without explicitly evaluating ϕ

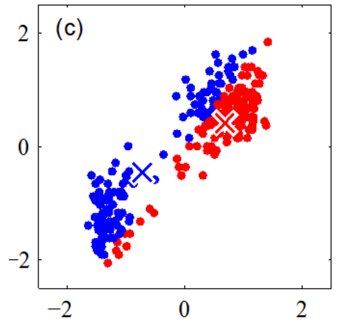
[Initialize] Pick $c_1, \dots, c_k$ as randomly selected points from $\tilde{S}$

For $t = 1,2, \dots, T$

- ***[Assignments]*** $\forall x \in S, \quad z_t(x) = \arg\min_{j \in [k]} \left\| \phi(x) - c_j \right\|_2^2$
- ***[Centroids]*** $\forall j \in [k], \quad c_j \leftarrow \text{average}(\{\phi(x) \colon x \in S, z_t(x) = j\})$

<u>Output</u>: $c_1, \dots, c_k$ and $\{a_t(x_i)\}_{i \in [n]}$





Kernelizing K-means algorithm (cont'd)

- How to calculate $\left\|\phi(x) - c_j\right\|_2^2$ without explicitly evaluating ϕ?

- Key observation: c_j always takes the form $c_j = \frac{1}{|U|}\sum_{i\in U} \phi(x_i)$ for some U, and therefore has the form $c_j = \sum_{i=1}^n \alpha_i \phi(x_i)$

- Therefore,

$$\begin{aligned}\left\|\phi(x) - c_j\right\|_2^2 &= \langle\phi(x), \phi(x)\rangle - 2\langle\phi(x), \textstyle\sum_{i=1}^n \alpha_i\phi(x_i)\rangle + \langle\textstyle\sum_{i=1}^n \alpha_i\phi(x_i), \textstyle\sum_{i=1}^n \alpha_i\phi(x_i)\rangle \\ &= K(x,x) - 2\textstyle\sum_{i=1}^n K(x, x_i) + \textstyle\sum_i \sum_j \alpha_i\alpha_j K(x_i, x_j)\end{aligned}$$

- Efficiently computable: only requires evaluating K now

Clustering as cost minimization: additional remarks

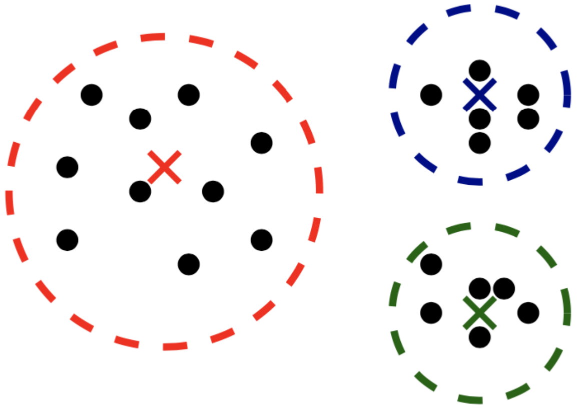
- The squared reconstruction error (aka k-means objective) is not the only criterion used:

$$f(c_1, \dots, c_k) = \sum_{i=1}^{n} \min_{j \in [k]} \left\| x - c_j \right\|_2^2$$

- Alternative popular cost functions:

k-median: $f(c_1, \dots, c_k) = \sum_{i=1}^{n} \min_{j \in [k]} \left\| x - c_j \right\|_2$

k-center: $f(c_1, \dots, c_k) = \max_{i} \min_{j \in [k]} \left\| x - c_j \right\|_2$



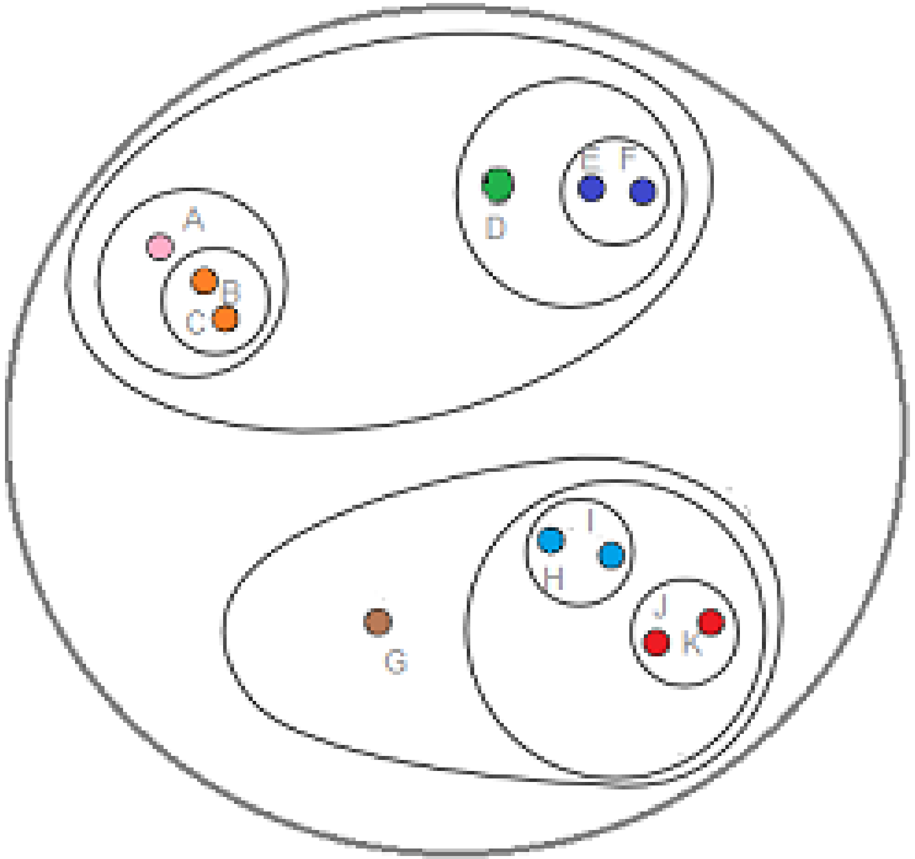
- Furthermore, we don't have to restrict to using ℓ_2 reconstruction error

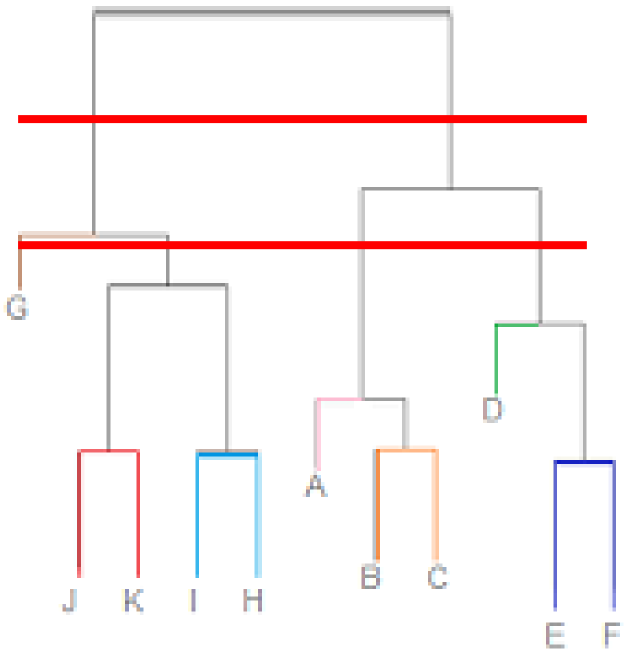
Outline

- Clustering
    - K-means clustering revisited
    - **Hierarchical clustering**
- Principal Component Analysis (PCA)

Hierarchical clustering – getting rid of tuning k

- Motivation: multiresolution data representation
- Idea: produce a tree structure over objects
- Can prune the tree appropriately to fit application needs (e.g. cluster radius / size requirements)

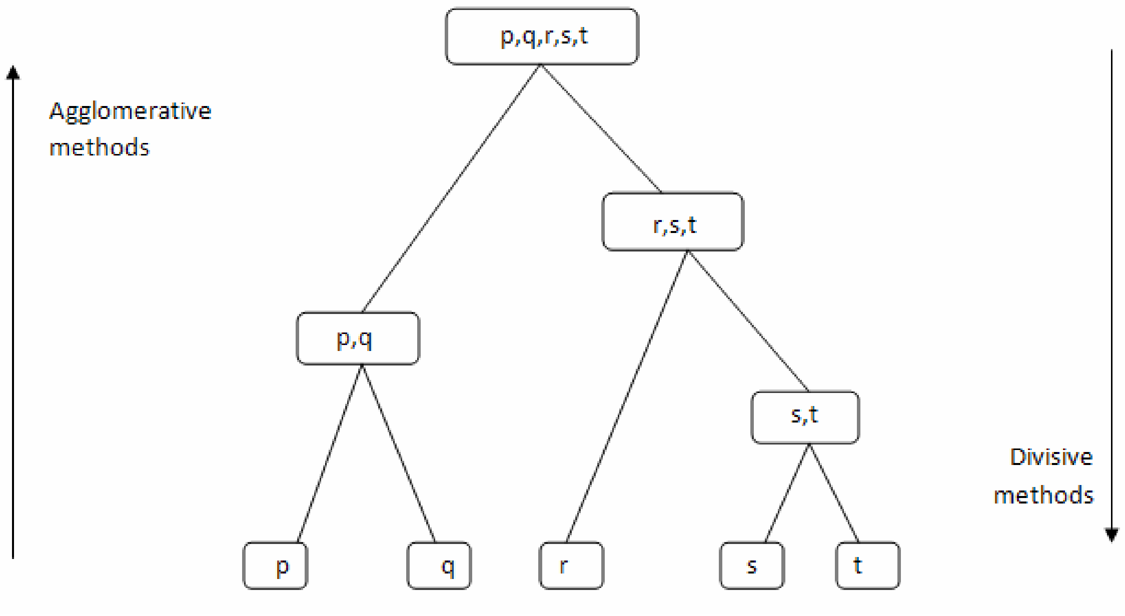




Hierarchical clustering

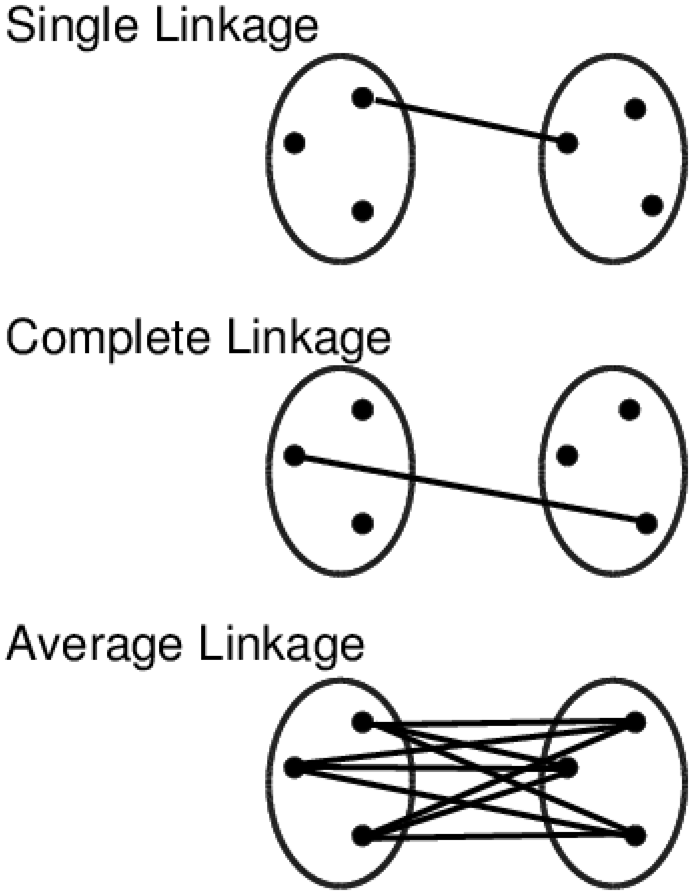
- Method 1: Top-down (divisive)
    - k-means clustering with k=2
    - Do this recursively on each resulting cluster (no more recursion when there is only one point in a cluster)
    - Conceptually similar to decision tree training

- Method 2: bottom-up (agglomerative, more popular)
    - Start with every point x_i being a singleton cluster
    - Repeatedly pick a pair of clusters with the smallest 'distance'
    - How do we define a distance between two clusters?



Agglomerative clustering: Distance between two clusters

- Single linkage
  - $\text{dist}(C, C') = \min_{x \in C, x' \in C'} \|x - x'\|_2$
- Complete linkage
  - $\text{dist}(C, C') = \max_{x \in C, x' \in C'} \|x - x'\|_2$
- Average linkage
  - $\text{dist}(C, C') = \frac{1}{|C| \cdot |C'|} \sum_{x \in C} \sum_{x' \in C} \|x - x'\|_2$



https://girke.bioinformatics.ucr.edu/GEN242/tutorials/rclustering/rclustering/

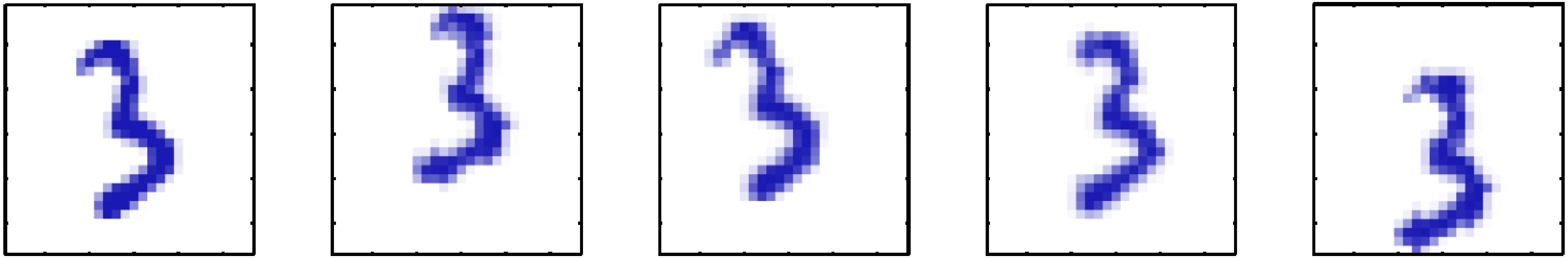
Outline

- Clustering
    - K-means clustering revisited
    - Hierarchical clustering

- **Principal Component Analysis (PCA)**

Motivation

Data often have a lot of redundant information…



Example A dataset consisting of a hand-drawn 3 at random locations and rotations in a 100x100 pixel image.

Data Dimension 100 x 100 = 10,000

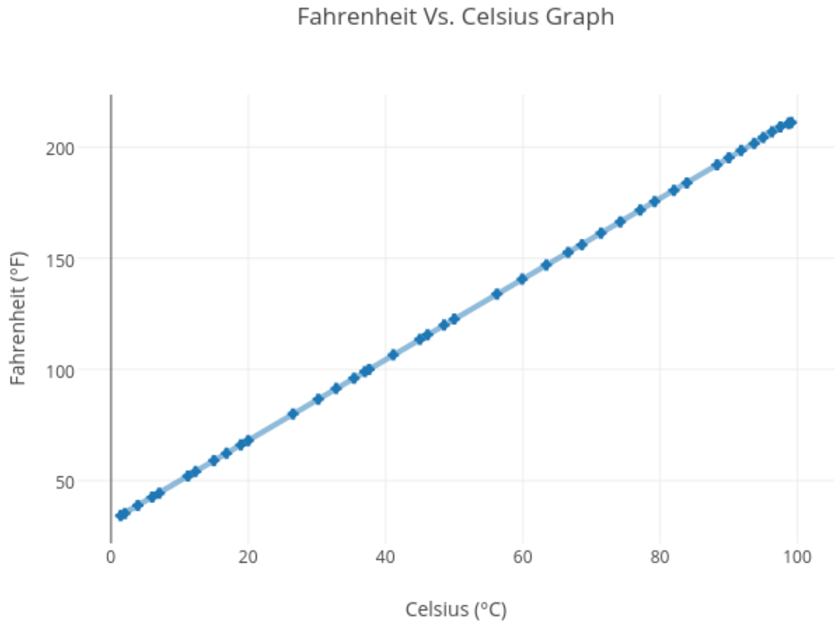
Intrinsic Dimension 3 (X-position, Y-position, Rotation)

[Source: Bishop, C.]

Motivation

…or data have strongly dependent features…

Fahrenheit	Celsius
3.1	-16.1
100.5	38.1
27.3	-2.6
18.1	-7.7
18.9	-7.3
21.7	-5.7
…	…

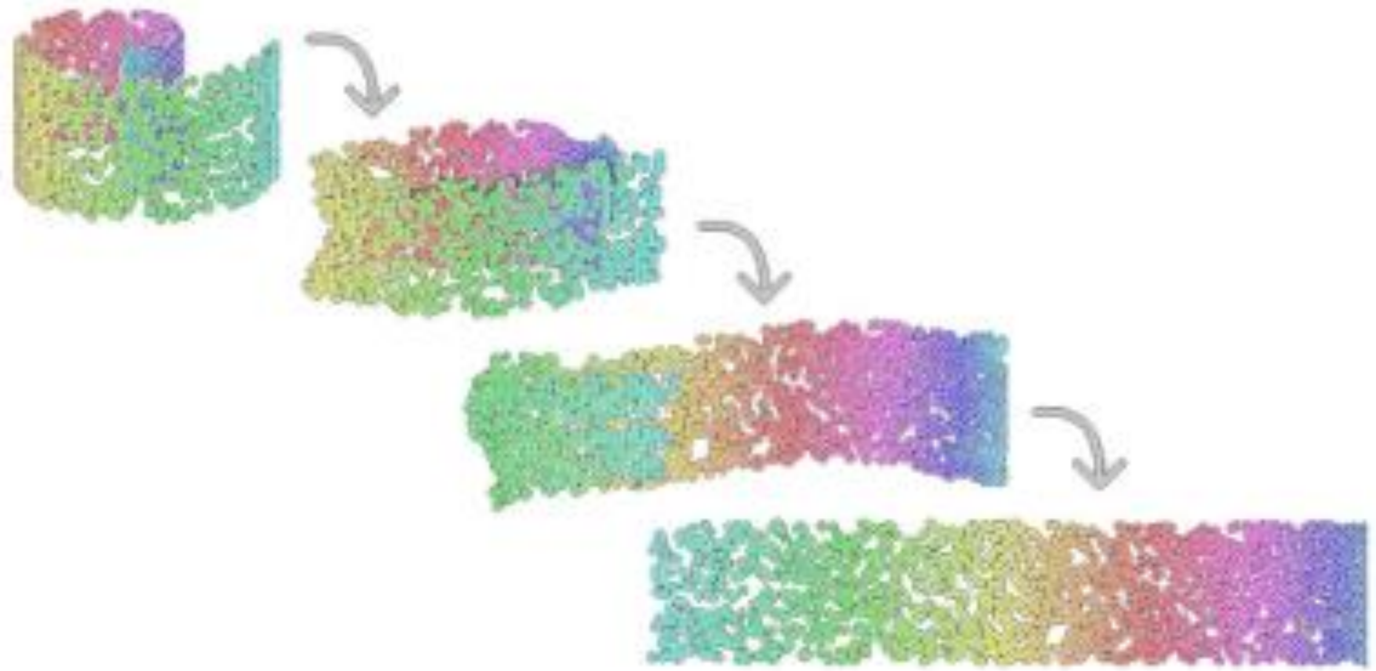


Linear Function

$$F = 1.8C + 32$$

Motivation

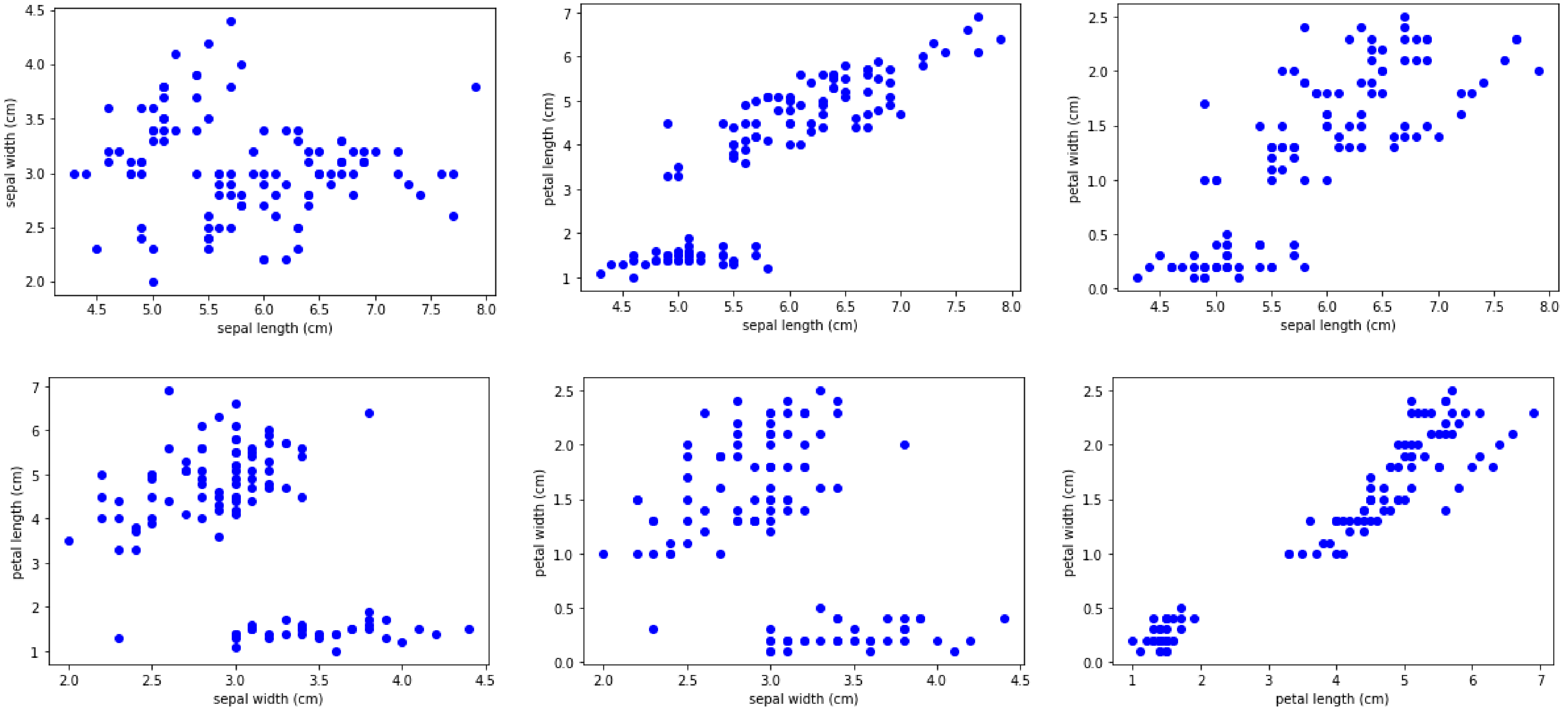
…or data are high-dimensional but have low *intrinsic dimension*…



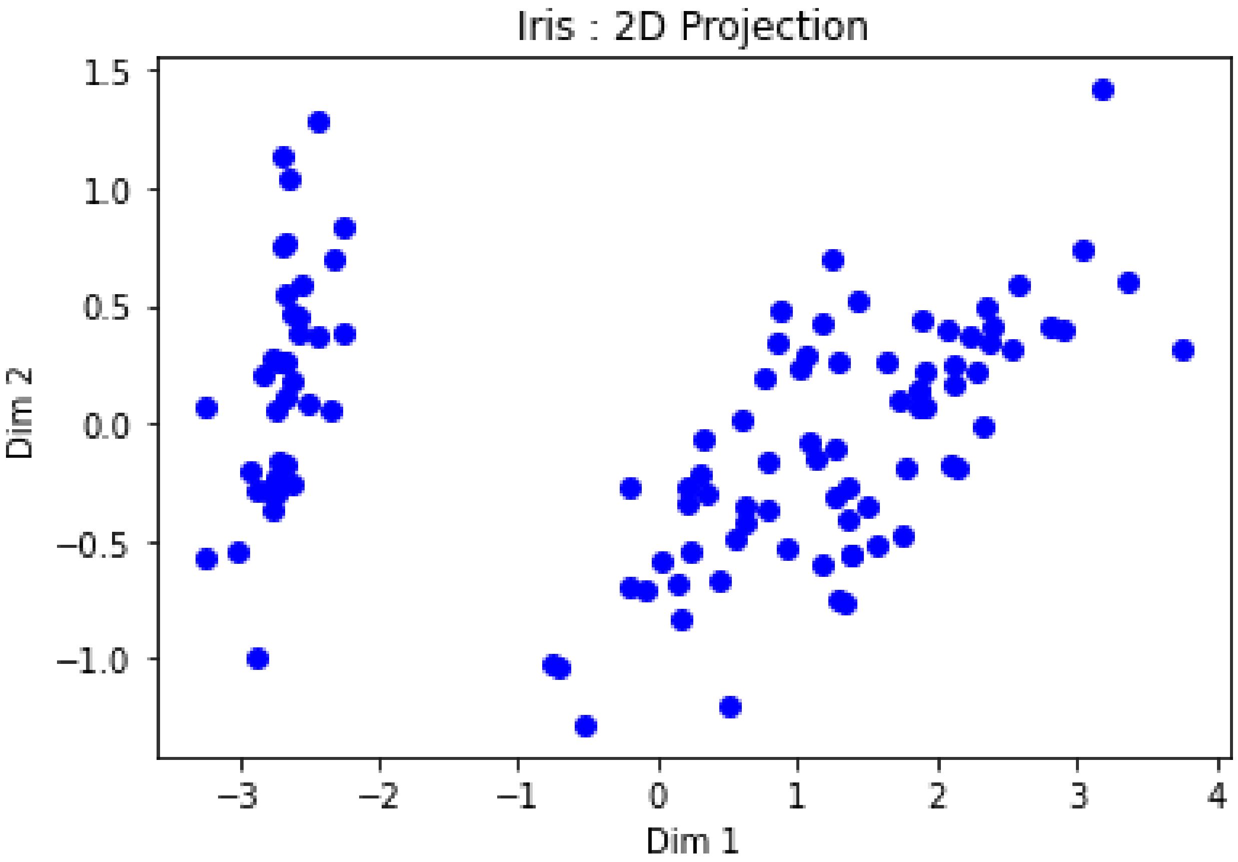
…in all cases, finding lower-dimensional representation is useful

Example: Iris Dataset

Recall that the Iris dataset has 4 features:
sepal length / width, petal length / width…



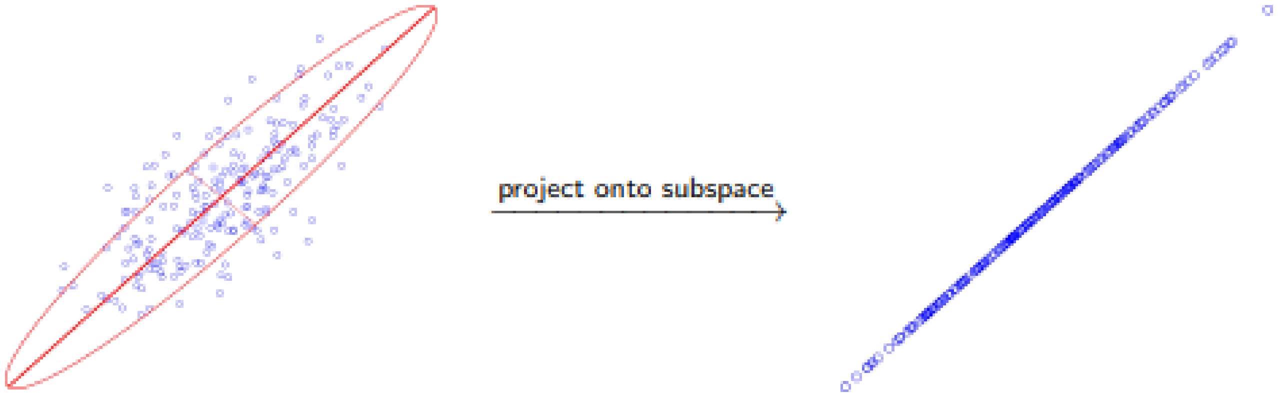
Example : Iris Dataset



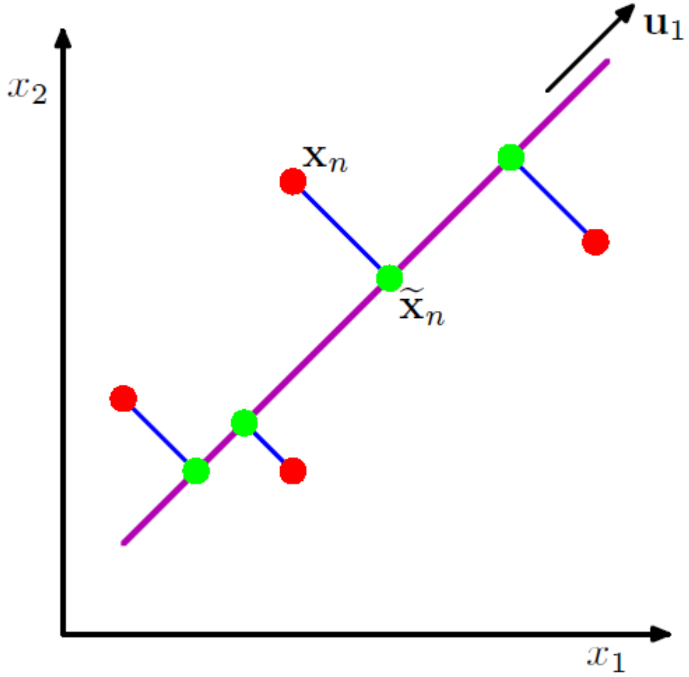
Data still cluster in a two-dimensional subspace

We can represent data in 2D to reduce complexity, visualize results, etc.

Linear Dimensionality Reduction



Project data onto a line or plane…

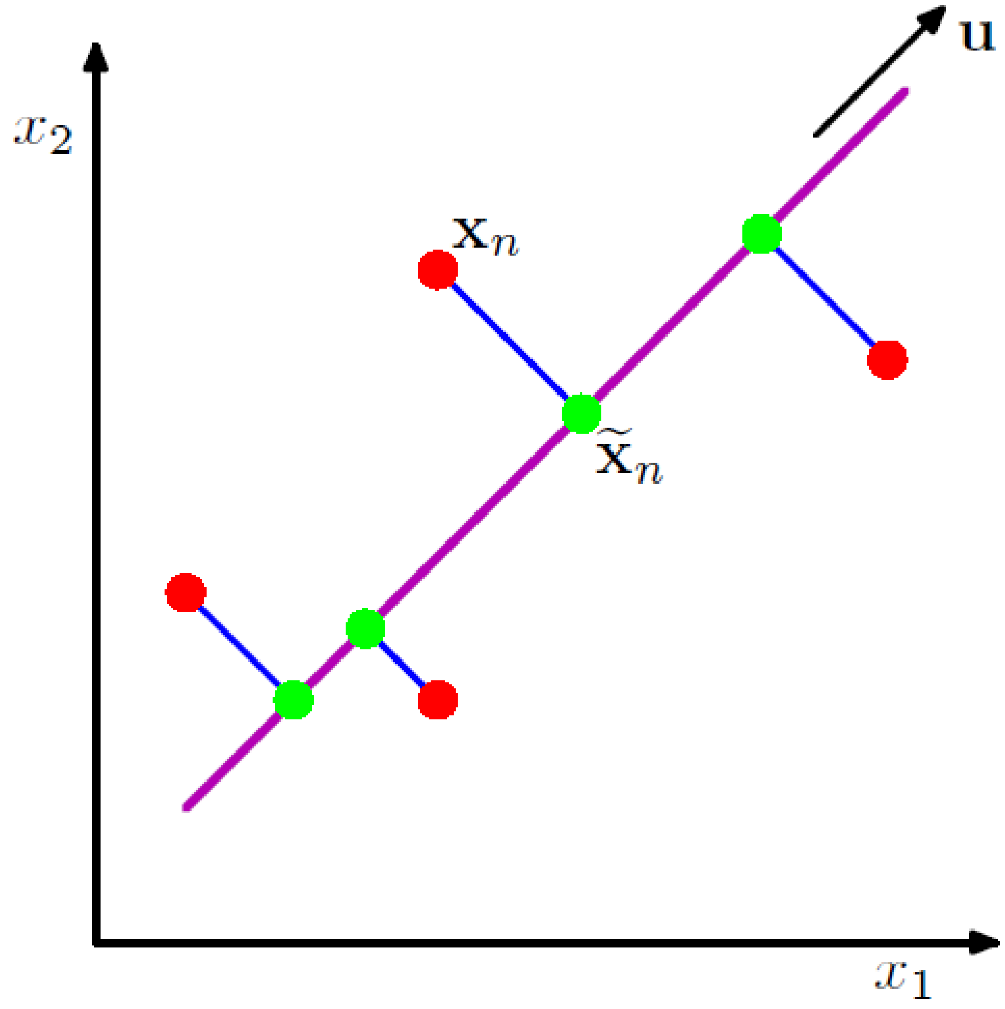


…one of the simplest dimensionality reduction approaches

First, let’s review some linear algebra…

[Source: Manfred Warmuth; Bishop, C.]

Linear Dimensionality Reduction



Projecting data onto a vector u is a simple inner product,

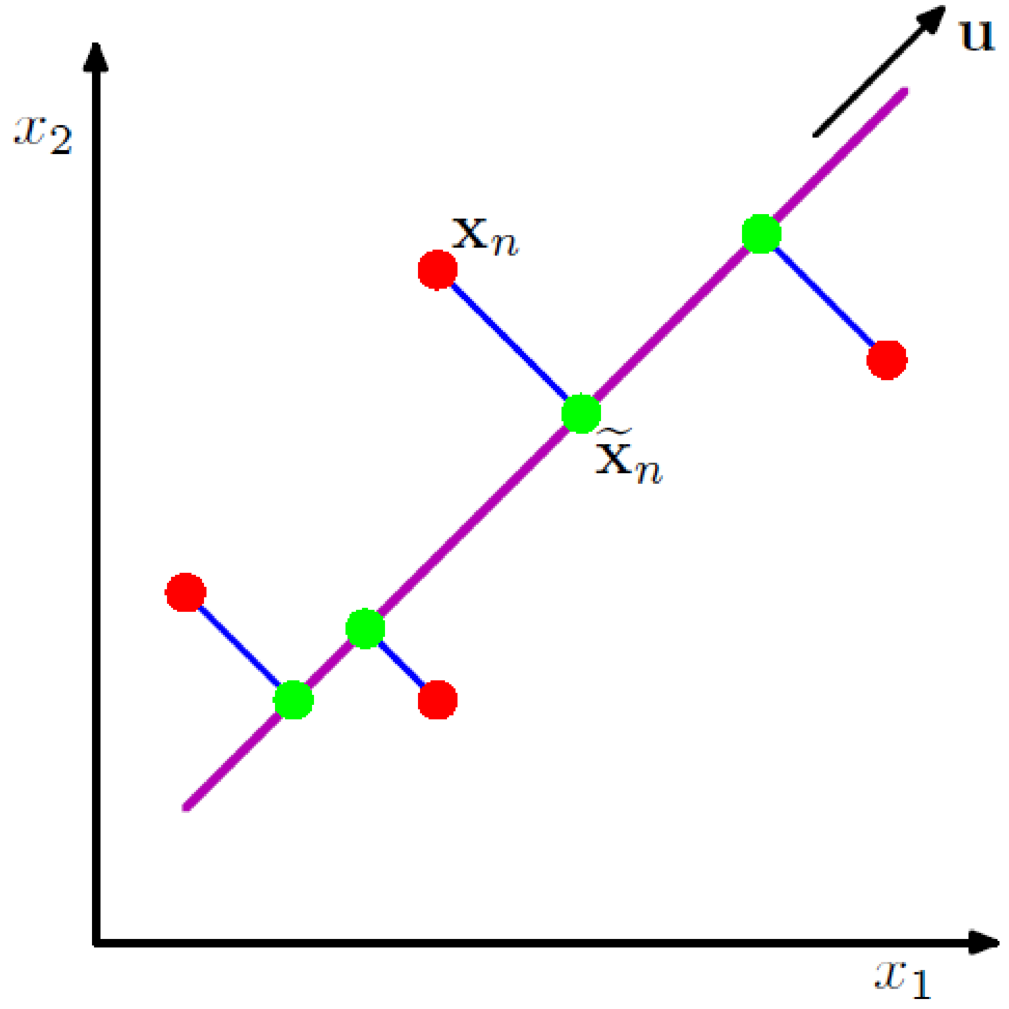
$$\widetilde{x}_n = u^T x_n$$

We call u the *linear subspace*

Question Why would dimensionality reduction be better than feature selection (e.g. choose 1-D features X1 or X2)?

[Source: Bishop, C.]

Linear Dimensionality Reduction



Projecting data onto a vector is a simple inner product,

$$\widetilde{x}_n = u^T x_n$$

We call *u* the *linear subspace*

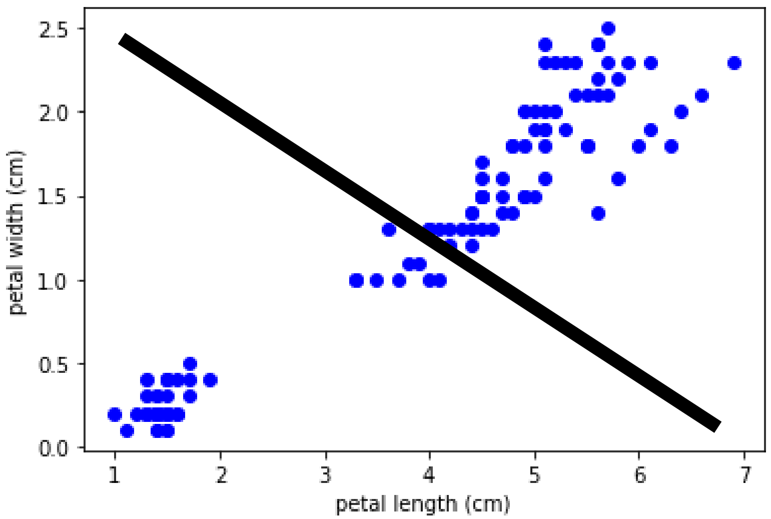
Answer No features are discarded (uses all the data),

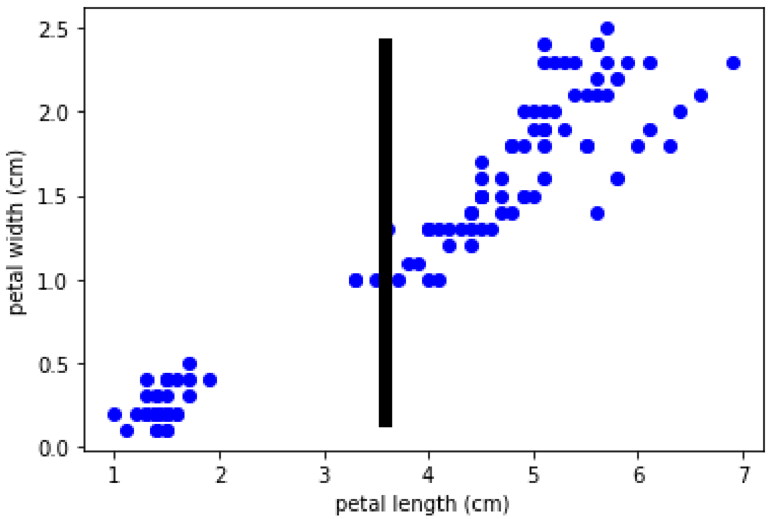
$$\widetilde{x}_n = u_1 x_{n1} + u_2 x_{n2}$$

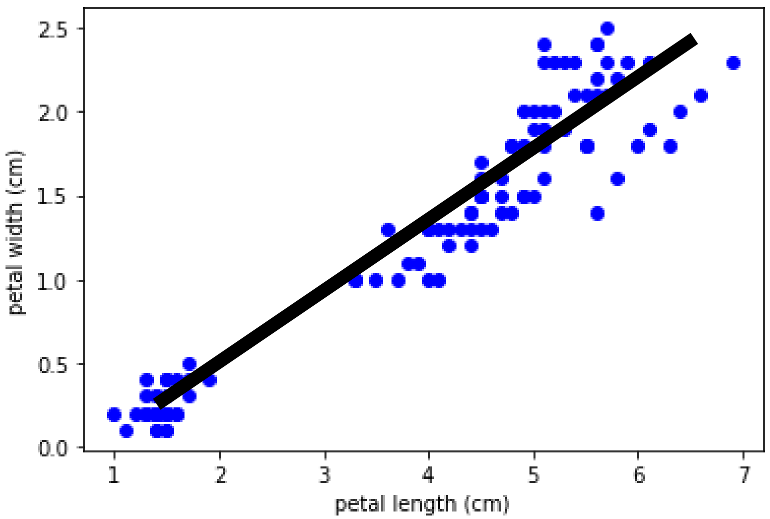
[Source: Bishop, C.]

Linear Dimensionality Reduction

Which choice of subspace is best? And why?

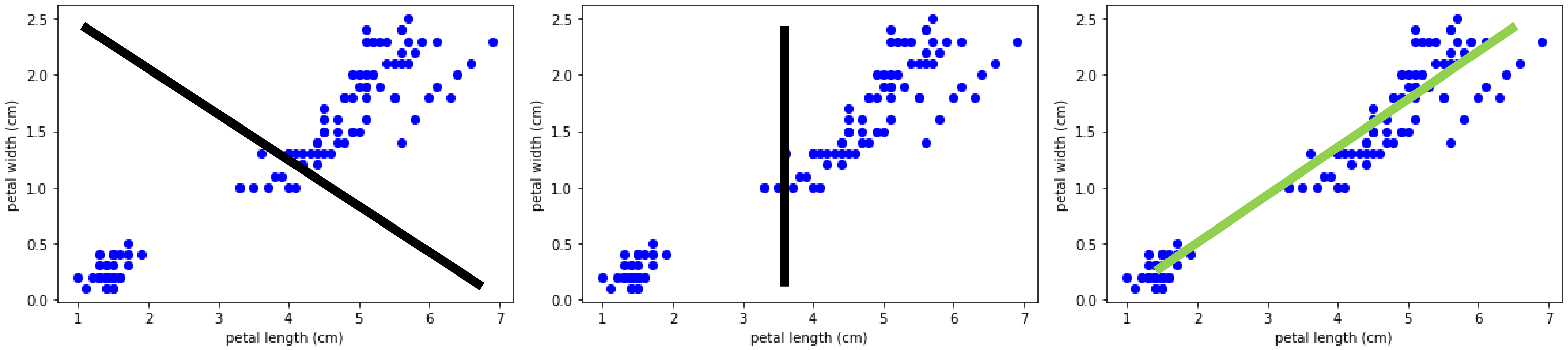






Linear Dimensionality Reduction

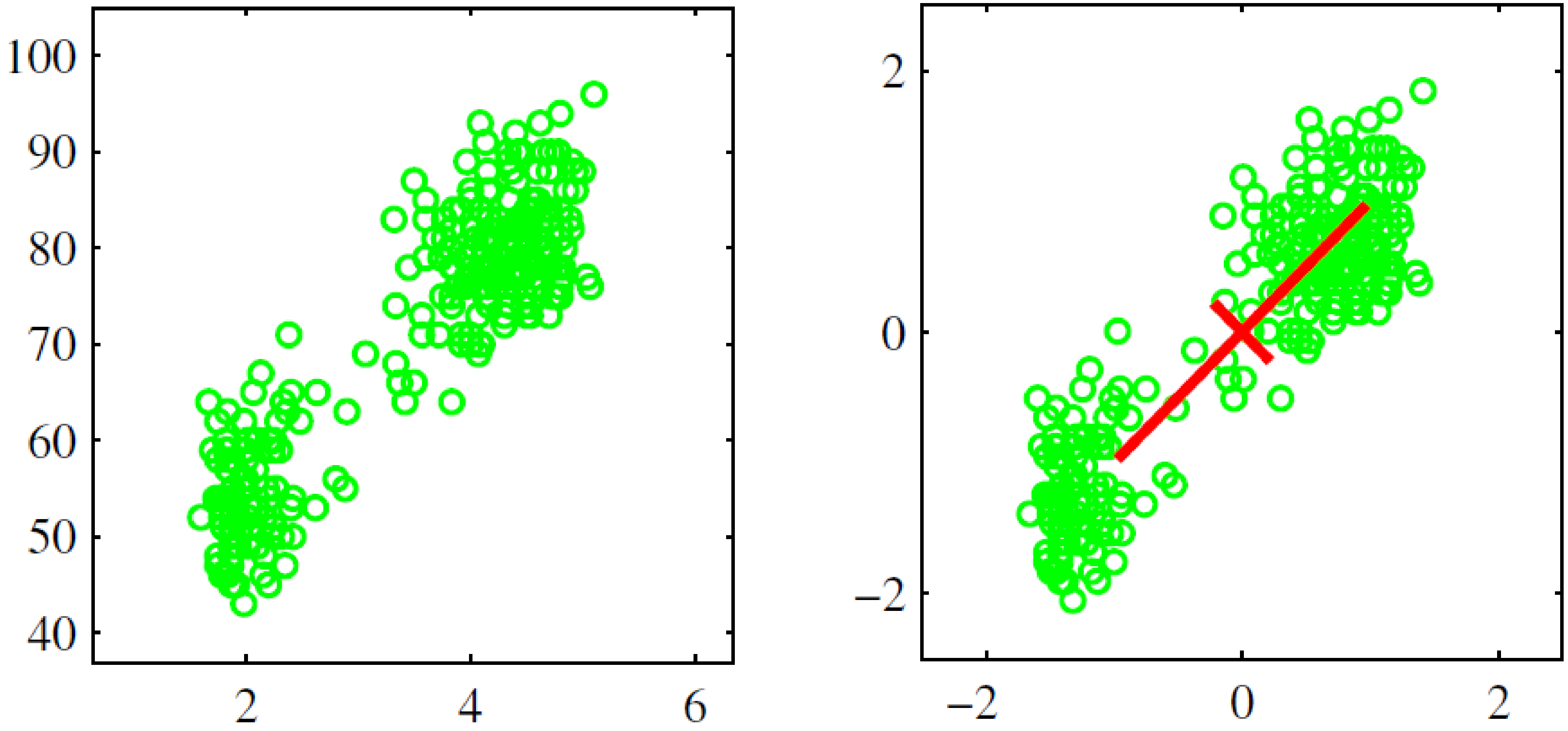
Which choice of subspace is best? And why?



Idea Choose the subspace that captures the most variation in the original data

Principal Component Analysis (PCA)

Identify directions of *maximum variation* as subspaces…



…we call each direction a *principal component*

[Source: Bishop, C.]

Principal Component Analysis (PCA)

First, center the data by subtracting the sample mean,

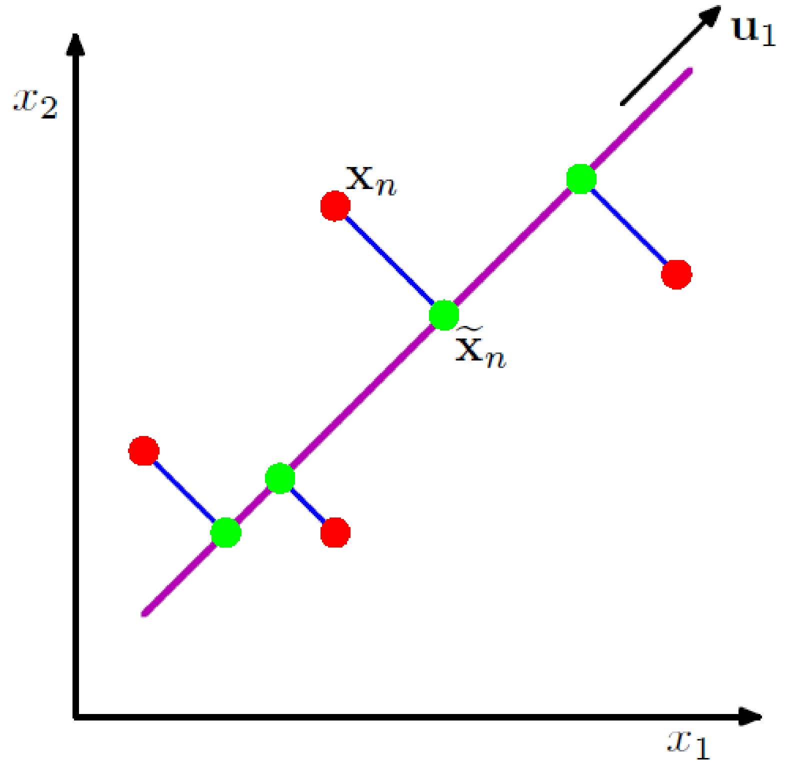
$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

Variance of projected data,

$$\frac{1}{N}\sum_{n=1}^{N}\left(u^T x_n - u^T \bar{x}\right)^2$$

Projection of n^{th} data point

Projection of mean



Maximum Variance Formulation

A little algebra…

$$\frac{1}{N}\sum_{n=1}^{N}\left(u^T x_n - u^T \bar{x}\right)^2 = \frac{1}{N}\sum_{n=1}^{N}\left\{u^T (x_n - \bar{x})\right\}^2$$

Pull out u

Quadratic form

$$= \frac{1}{N}\sum_{n=1}^{N} u^T (x_n - \bar{x})(x_n - \bar{x})^T u$$

Define: $S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T$

What is this? **Data covariance matrix**

Then: $\frac{1}{N}\sum_{n=1}^{N}\left(u^T x_n - u^T \bar{x}\right)^2 = u^T S u$ ←

This is what we will optimize over u

Maximum Variance Formulation

Find u so that projected variance is maximal…

$$\max_{u} u^T S u$$

Don’t want to *cheat* with large magnitude u, so we add penalty,

$$\max_{u} u^T S u - \lambda u^T u$$

Set the derivative (gradient) to zero and solve…

$$S u - \lambda u = 0$$

What does this equation mean?

$$S u = \lambda u$$

u **is an *eigenvector* with *eigenvalue* λ**

Recap of Concepts

- Learning a low-dimensional representation is useful
- The easiest approach is to find a *linear subspace*
- PCA chooses the linear subspace that maximizes variance of the projected data

$$\max_u u^T S u - \lambda u^T u$$

- Such subspaces are defined by the *eigenvectors,*

$$S u = \lambda u$$

But what is an eigenvector?

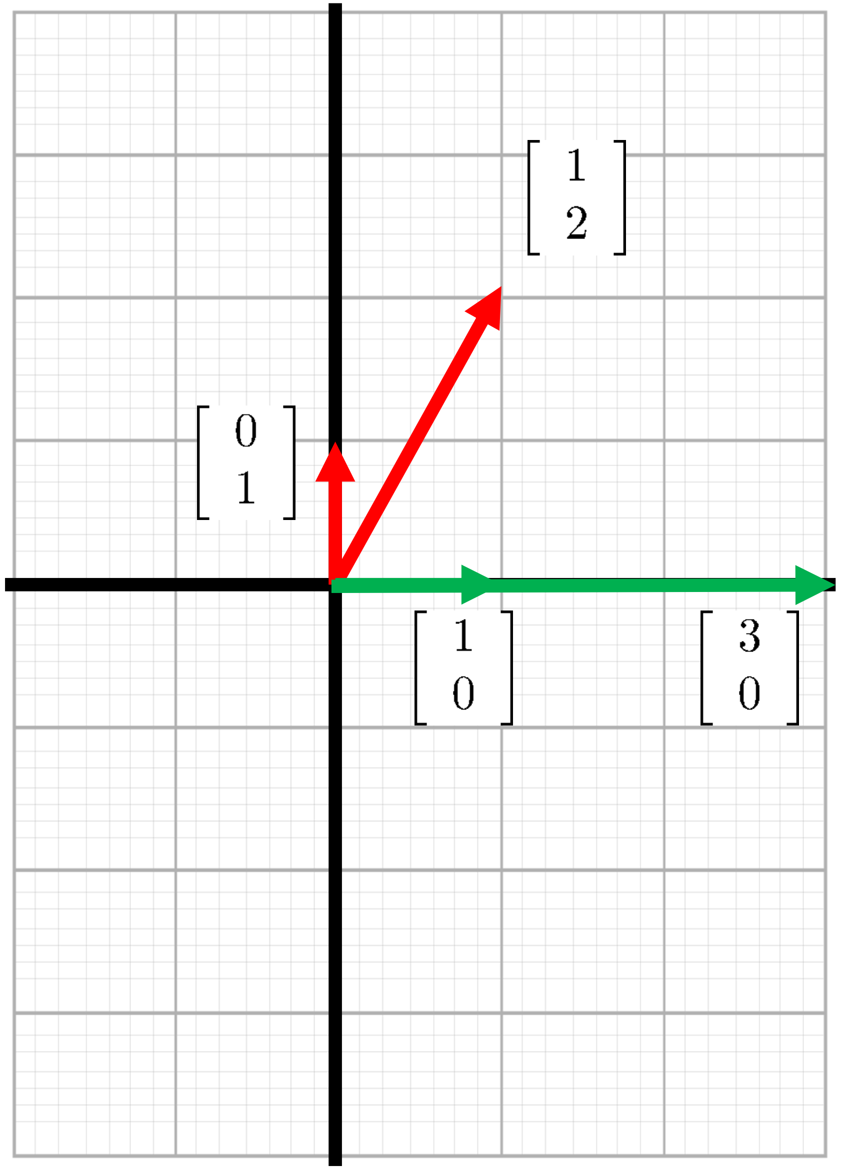
Linear Transformations

Consider the matrix: $\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$

Let’s multiply it with some vectors…

$$\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3\cdot 0 + 1\cdot 1 \\ 0\cdot 0 + 2\cdot 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3\cdot 1 + 1\cdot 0 \\ 0\cdot 1 + 2\cdot 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$


$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$
$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$

- Matrix transforms vectors from one basis to another
- Columns are transformation of standard basis

Eigenvalue & eigenvector

Observe that the X-axis vector just gets “stretched out”,

$$\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

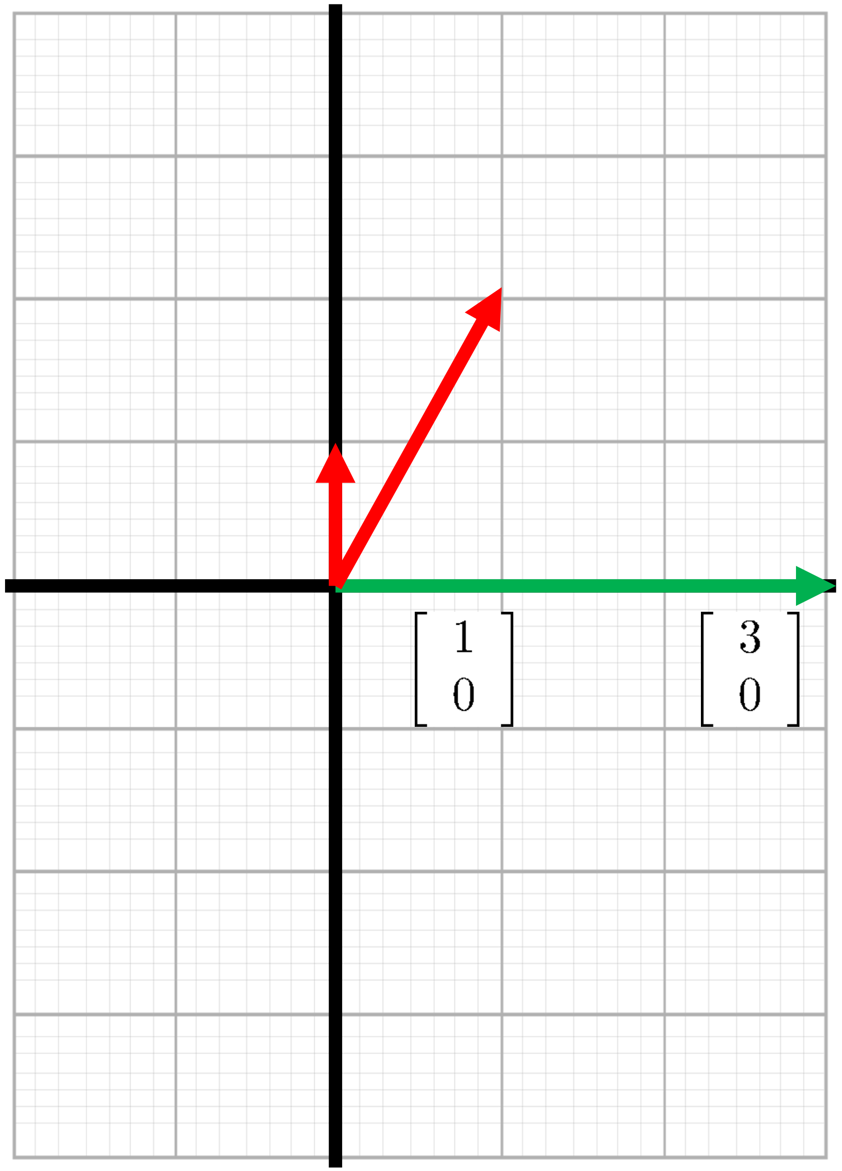
Factoring out the 3 we have,

$$\underset{\mathbf{S}}{\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}} \underset{\mathbf{u}}{\begin{bmatrix} 1 \\ 0 \end{bmatrix}} = \underset{\lambda}{3} \underset{\mathbf{u}}{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}$$

Define some variables and we have the equation,

$$Su = \lambda u$$

So $(1,0)^T$ is an *eigenvector* of S with *eigenvalue* 3


$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$

Eigenvalue & eigenvector

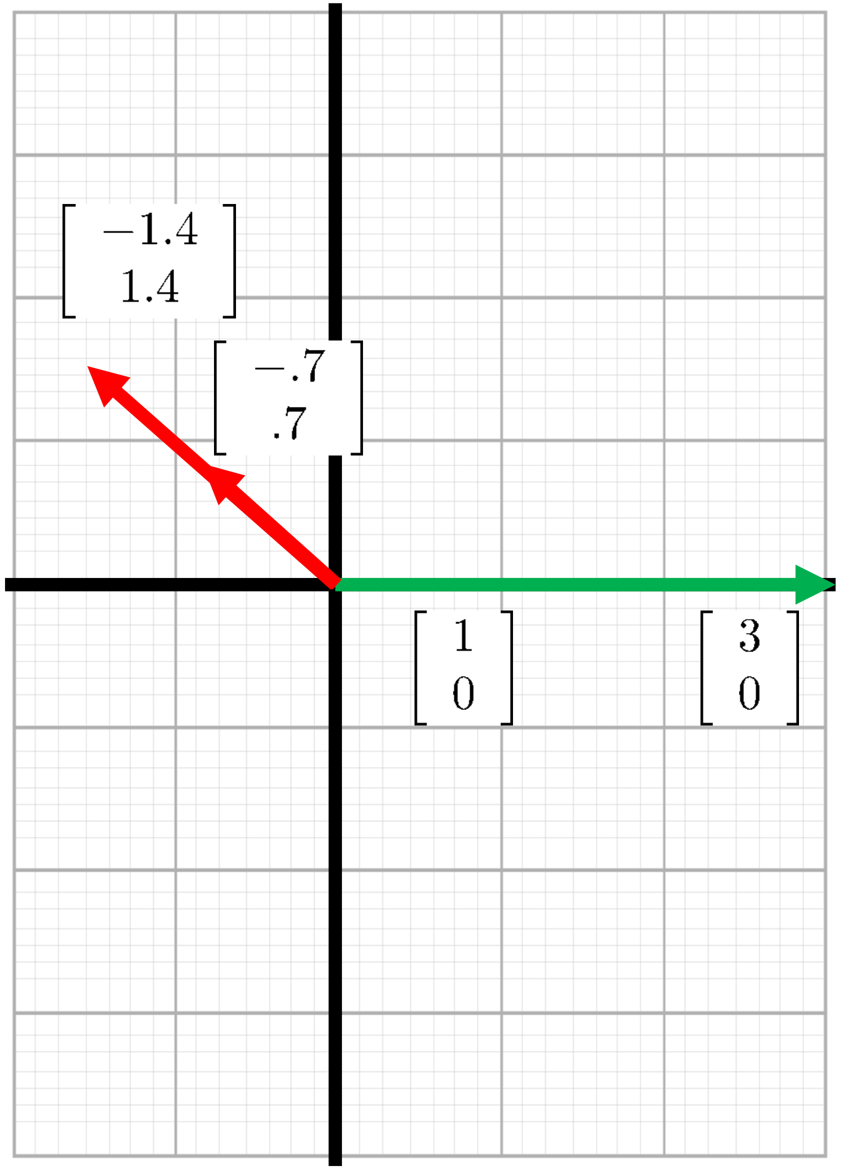
Transformation has one other eigenvector,

$$\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -0.7 \\ 0.7 \end{bmatrix} = \begin{bmatrix} -1.4 \\ 1.4 \end{bmatrix} = 2 \begin{bmatrix} -0.7 \\ 0.7 \end{bmatrix}$$

- Complete eigen-representation of S

Eigenvectors $\begin{bmatrix} 1 & -0.7 \\ 0 & 0.7 \end{bmatrix}$ Eigenvalues $\begin{bmatrix} 3 \\ 2 \end{bmatrix}$

- Eigenvectors of linear transformation S are only stretched / shrunk / flipped
- Eigenvalues tell how much they are stretched / shrunk / flipped

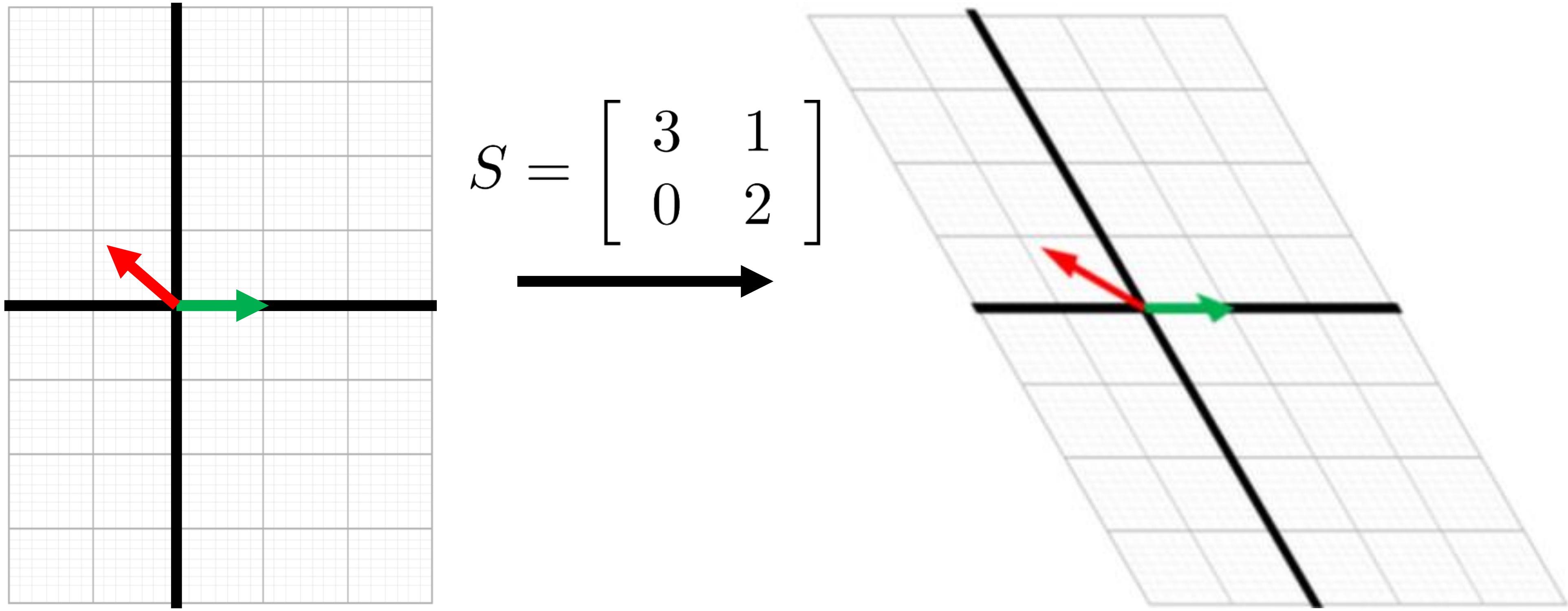


$\begin{bmatrix} -1.4 \\ 1.4 \end{bmatrix}$
$\begin{bmatrix} -.7 \\ .7 \end{bmatrix}$
$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$

Eigenvalue & eigenvector

Eigenvectors $\begin{bmatrix} 1 & -0.7 \\ 0 & 0.7 \end{bmatrix}$ Eigenvalues $\begin{bmatrix} 3 \\ 2 \end{bmatrix}$

Eigenvalue & eigenvector highlight what a linear transformation does by identifying directions that are *not* altered



$$S = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$$

Eigenvalue & eigenvector

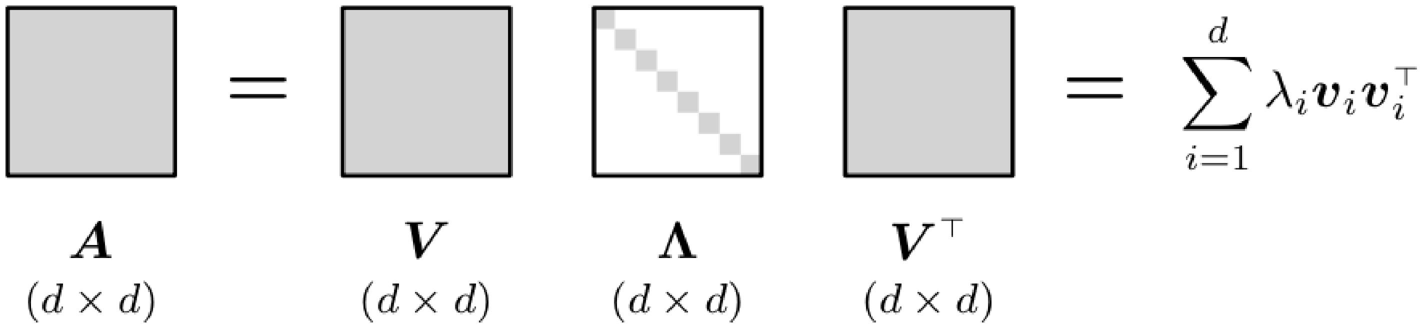
Eigenvectors / values of a matrix solve the equation

$$Su = \lambda u$$

- Matrix S may have *multiple* eigenvectors / values that solve the above equation

Eigendecomposition for symmetric real matrices

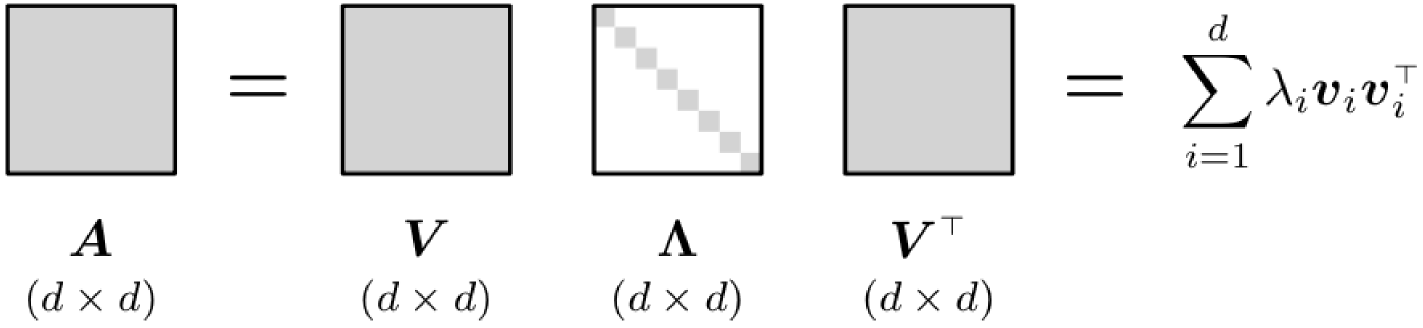
- **Fact:** Every **<u>symmetric</u>** **<u>real</u>** matrix $A \in \mathbb{R}^{d\times d}$ is guaranteed to have the following factorization:



- Convention: $\lambda_1 \geq \cdots \geq \lambda_d$
- For positive semi-definite (PSD) A, $\lambda_i \geq 0$ for all i
- Here, $V = \begin{pmatrix} | & \cdots & | \\ v_1 & \ldots & v_d \\ | & \cdots & | \end{pmatrix}$ has orthonormal columns, i.e. $v_i^\top v_j = I(i = j)$
- Why do we care?
  - Our data covariance matrix $S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^\top$ is symmetric real & PSD

Eigendecomposition for symmetric real matrices

- **Fact:** Every **symmetric real** matrix $A \in \mathbb{R}^{d \times d}$ is guaranteed to have the following factorization:



- Claim: A has exactly d (eigenvalue, eigenvector) pairs: $(v_1, \lambda_1), \ldots, (v_d, \lambda_d)$
- Why? Take v_1 for example:

$$\begin{aligned} Av_1 &= \lambda_1 v_1 v_1^\top v_1 + \cancel{\lambda_2 v_2 v_2^\top v_1} + \cdots \cancel{\lambda_d v_d v_d^\top v_1} \\ &= \lambda_1 v_1 \end{aligned}$$

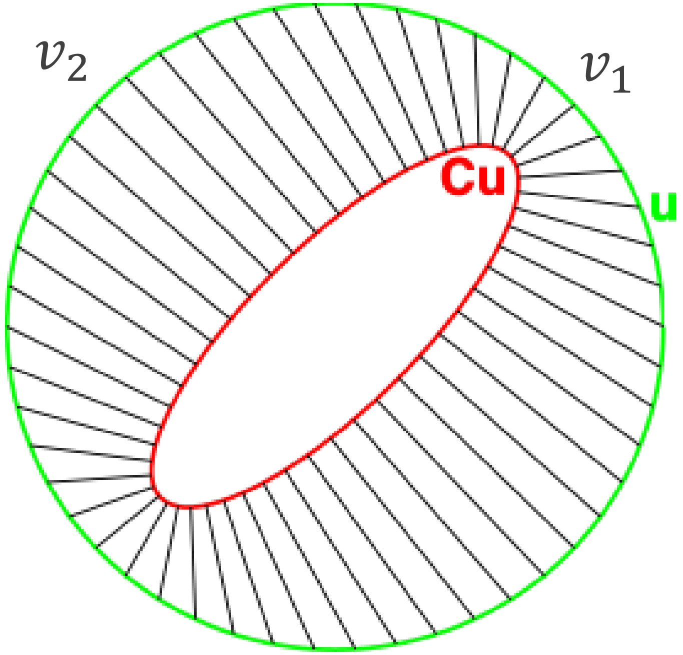
Eigenvectors and Ellipses

How does this connect to PCA?

Take all points on a unit circle and apply the linear transformation C

If C is a *covariance matrix,* then points will be transformed into an ellipse and…

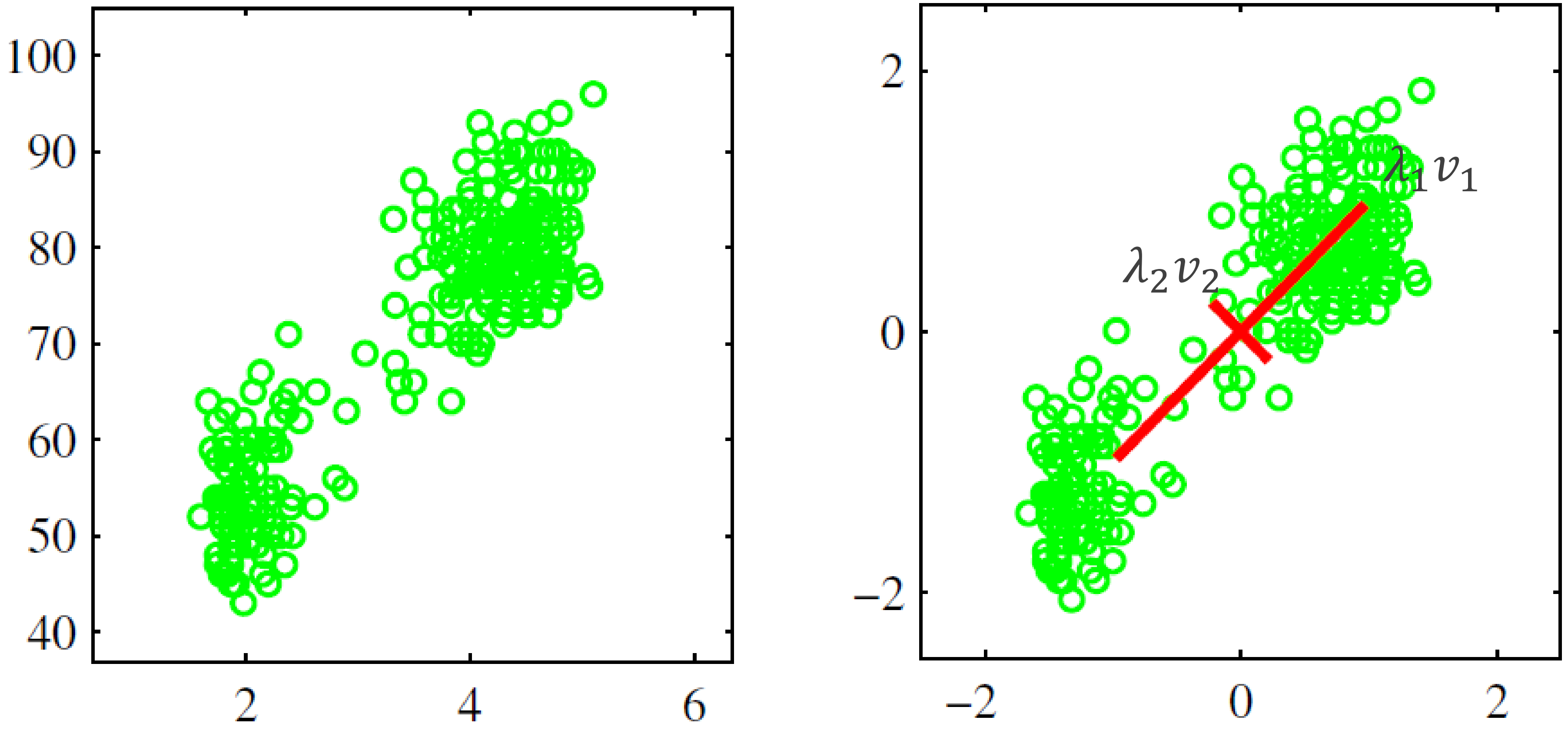
- Eigenvectors are axes of ellipse
- Eigenvalues are length of each axes
- Sort eigenvalues to get major / minor / etc. axes
- In the context of PCA, eigenvectors = **principal components**



https://mwarmuth.bitbucket.io/pubs/C77talk.pdf

Principal Component Analysis (PCA)

Sort eigenvectors by their eigenvalues…

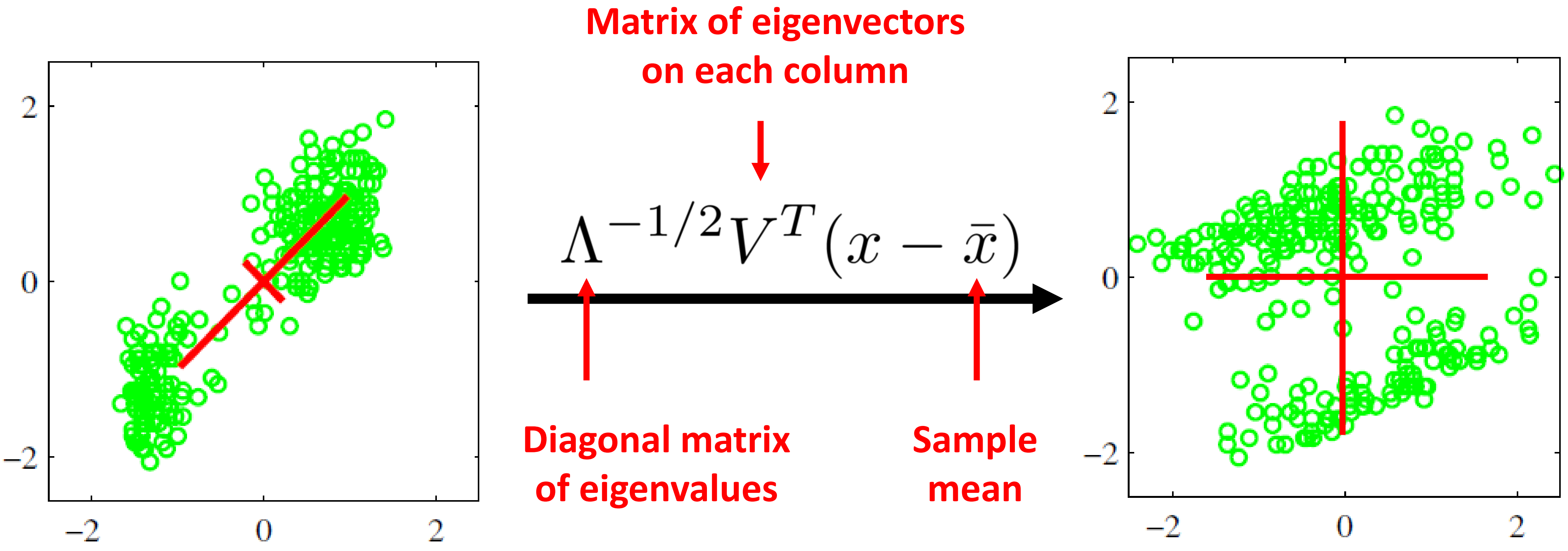


…amount of variance in each principal component decreases with eigenvalue

[Source: Bishop, C.]

Data “Whitening”

Multiplying data by eigenvectors transforms data so they are zero-mean and uncorrelated



Matrix of eigenvectors on each column

$$\Lambda^{-1/2}V^T(x-\bar{x})$$

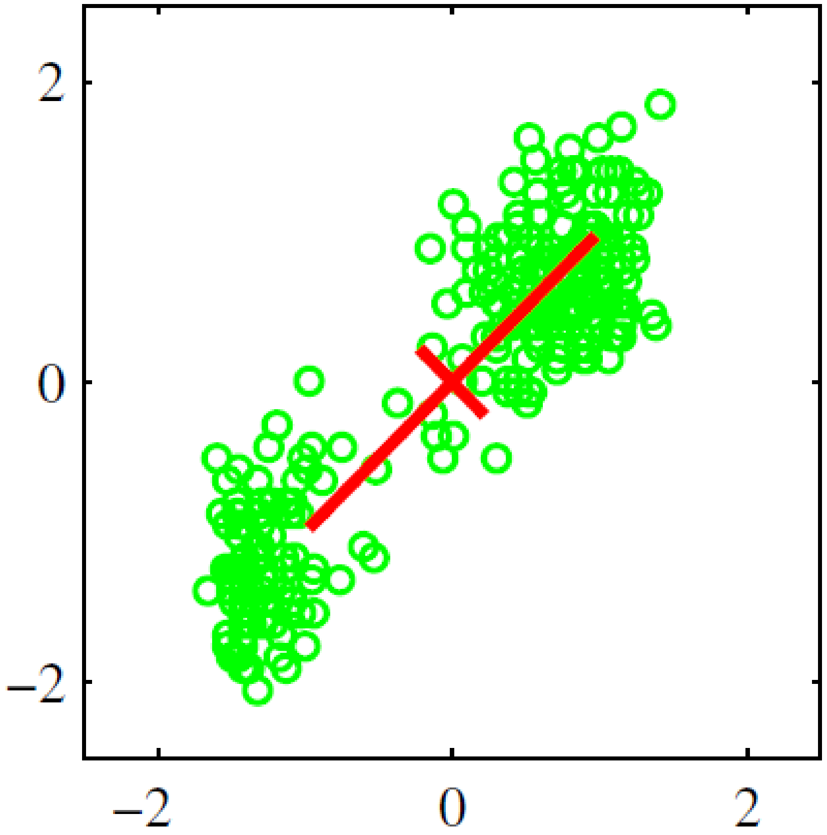
Diagonal matrix of eigenvalues

Sample mean

Data whitening can be an important preprocessing step for many data science applications (even if we don’t care about dimensionality reduction)

Principal Component Analysis (PCA)

How much variance is captured by just the first principal component (i.e. eigenvector with largest eigenvalue)?



Let v_1 be the first principal component, then variance of first PC is,

$$\frac{1}{N}\sum_n \left\{v_1^T(x_n - \bar{x})\right\}^2 = v_1^T S v_1 = \lambda_1$$

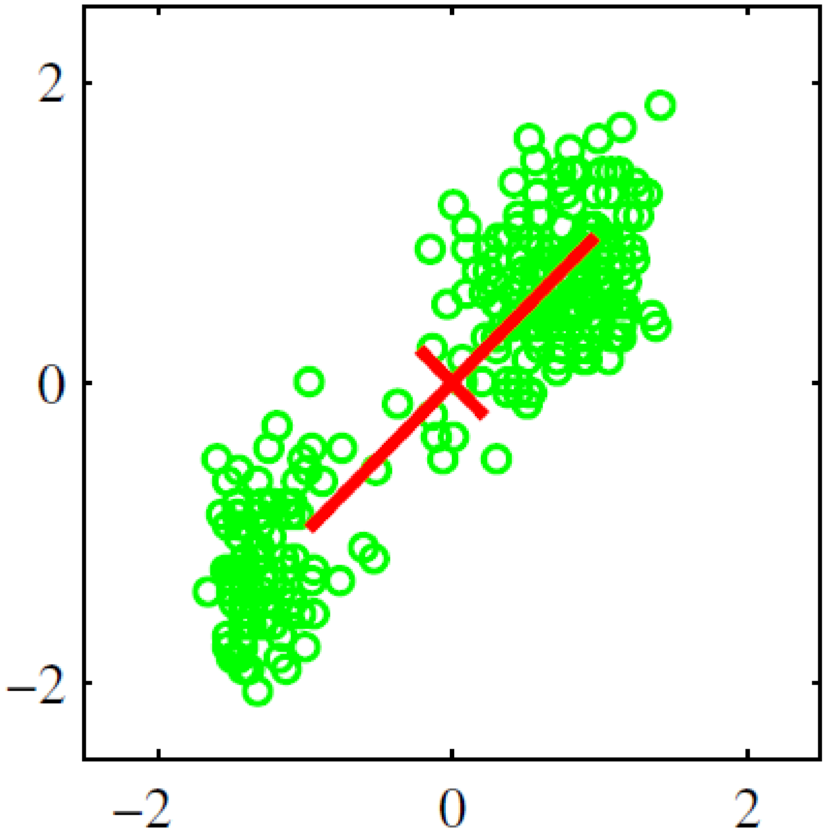
How much in the second PC?

$$\frac{1}{N}\sum_n \left\{v_2^T(x_n - \bar{x})\right\}^2 = v_2^T S v_2 = \lambda_2$$

[Source: Bishop, C.]

Explained Variance

How much variance is captured in M < D principal components?



$$\frac{1}{N}\sum_{m=1}^{M}\sum_{n}\left\{u_m^T(x_n-\bar{x})\right\}^2=\sum_{m=1}^{M}\lambda_m$$

We call this the *explained variance* of the first M principal components

Divide by total variance to find percentage of the total variance explained by the subspace

[Source: Bishop, C.]

Concept Recap

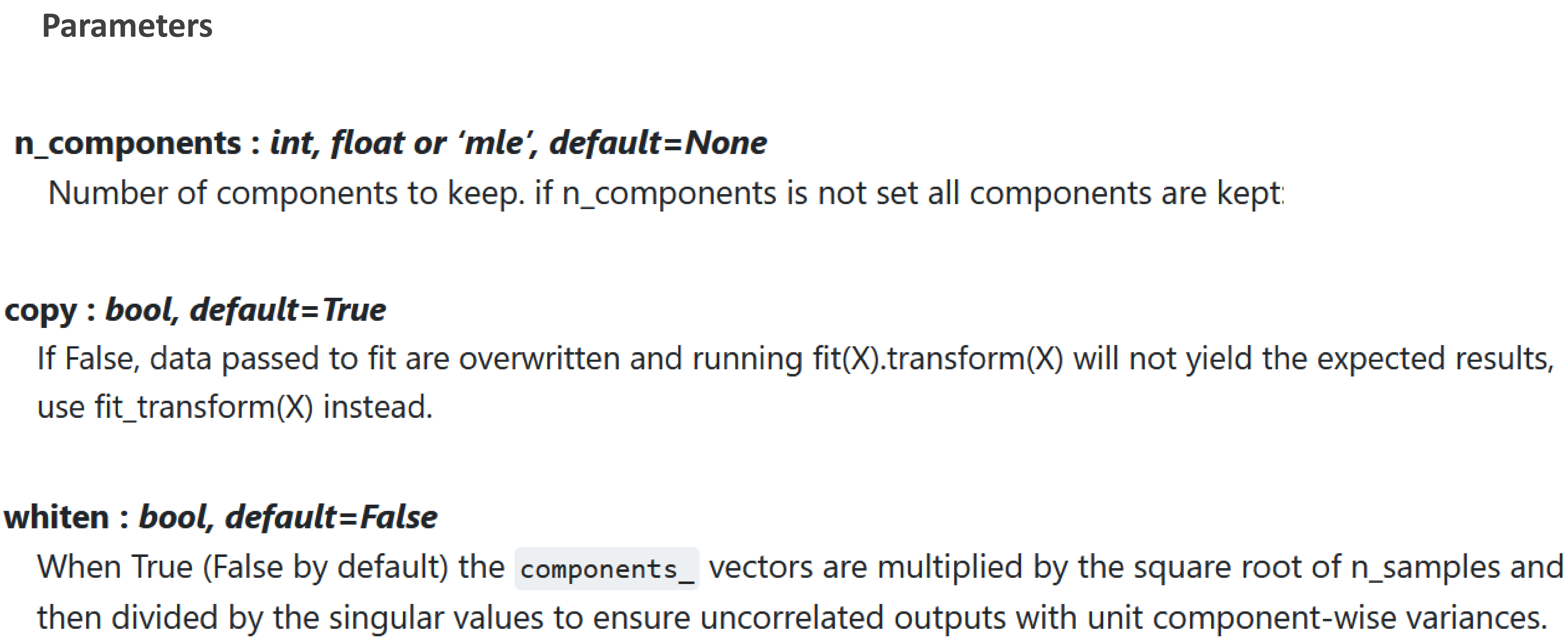
Eigenvectors

- For a general linear transform – identify directions that are only stretched / shrunk / flipped
- For a covariance matrix – identify axes of the ellipse that describes covariance

PCA

- Learns linear subspace as M < D principal components corresponding to M eigenvectors with largest eigenvalues
- Can be used to *whiten* (standardize, de-correlate) data
- Explained variance of M principal components easily calculated as percent of total explained variance in whitened data

sklearn.decomposition.PCA



Parameters

n_components : *int, float or 'mle', default=None*

Number of components to keep. if n_components is not set all components are kept:

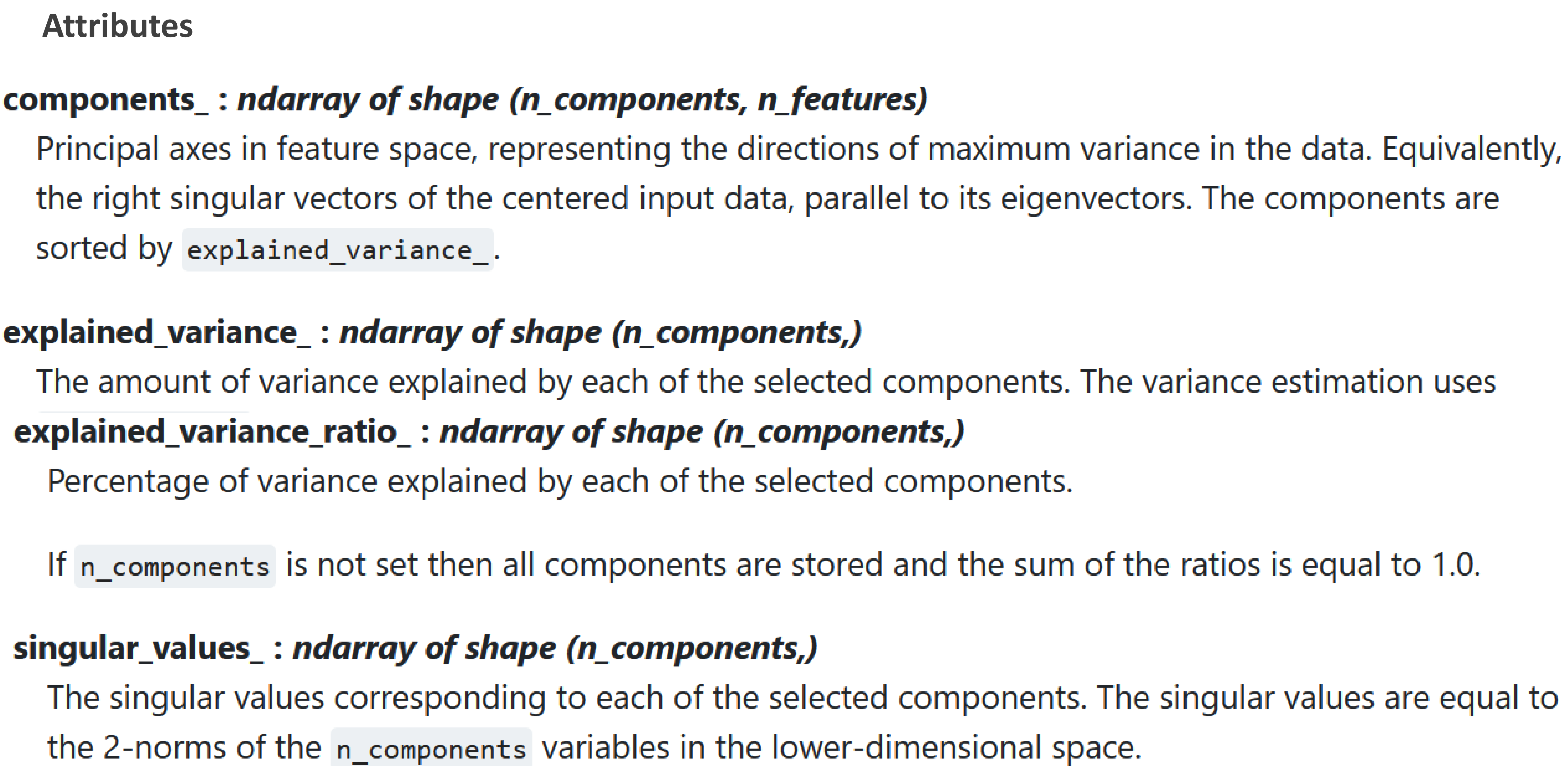
copy : *bool, default=True*

If False, data passed to fit are overwritten and running fit(X).transform(X) will not yield the expected results, use fit_transform(X) instead.

whiten : *bool, default=False*

When True (False by default) the `components_` vectors are multiplied by the square root of n_samples and then divided by the singular values to ensure uncorrelated outputs with unit component-wise variances.

`sklearn.decomposition`.PCA



Attributes

components_ : *ndarray of shape (n_components, n_features)*

Principal axes in feature space, representing the directions of maximum variance in the data. Equivalently, the right singular vectors of the centered input data, parallel to its eigenvectors. The components are sorted by `explained_variance_`.

explained_variance_ : *ndarray of shape (n_components,)*

The amount of variance explained by each of the selected components. The variance estimation uses

explained_variance_ratio_ : *ndarray of shape (n_components,)*

Percentage of variance explained by each of the selected components.

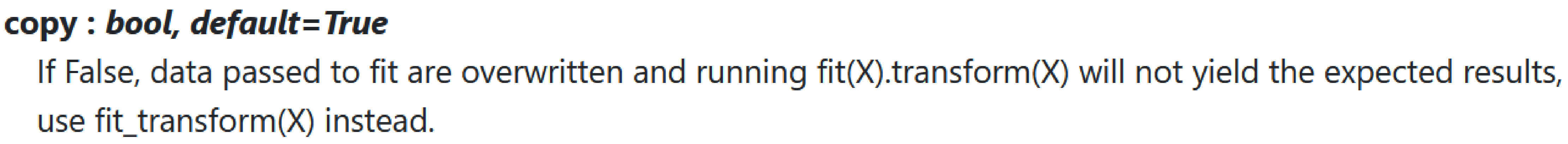
If `n_components` is not set then all components are stored and the sum of the ratios is equal to 1.0.

singular_values_ : *ndarray of shape (n_components,)*

The singular values corresponding to each of the selected components. The singular values are equal to the 2-norms of the `n_components` variables in the lower-dimensional space.

Caution

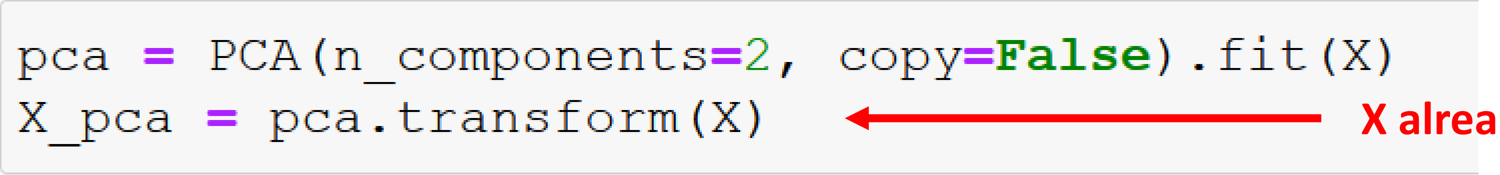
Careful with the following parameter,



copy : *bool, default=True*

If False, data passed to fit are overwritten and running fit(X).transform(X) will not yield the expected results, use fit_transform(X) instead.

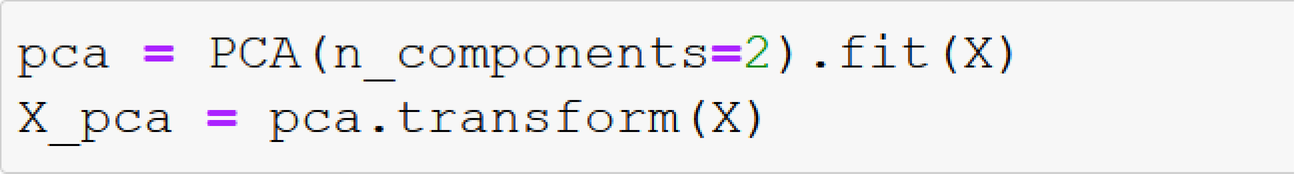
Wrong



```
pca = PCA(n_components=2, copy=False).fit(X)
X_pca = pca.transform(X)
```

X already modified

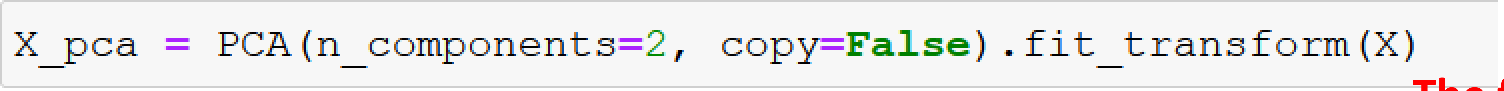
Right



```
pca = PCA(n_components=2).fit(X)
X_pca = pca.transform(X)
```

Why would you prefer one over the other?

Right

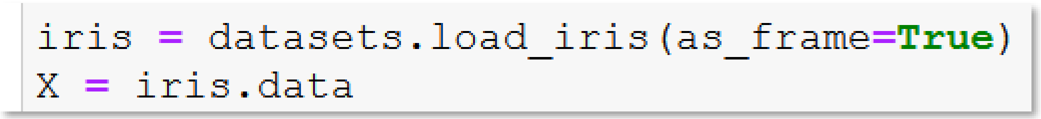


```
X_pca = PCA(n_components=2, copy=False).fit_transform(X)
```

The first approach keeps pca object which allows to interpret PCs

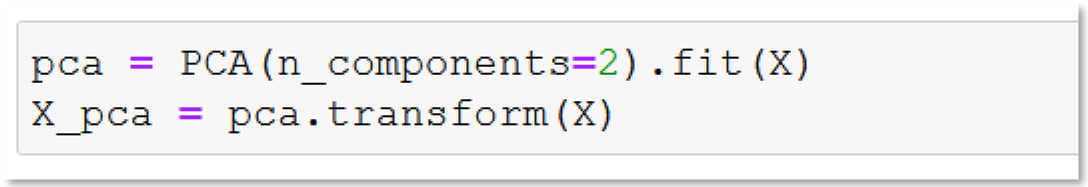
Example : PCA on Iris Data

Load Iris data without labels,



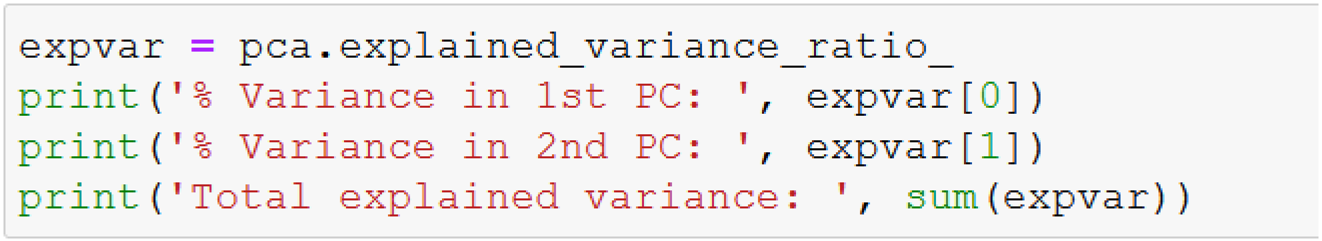
```python
iris = datasets.load_iris(as_frame=True)
X = iris.data
```

Find PCA with 2 principal components,

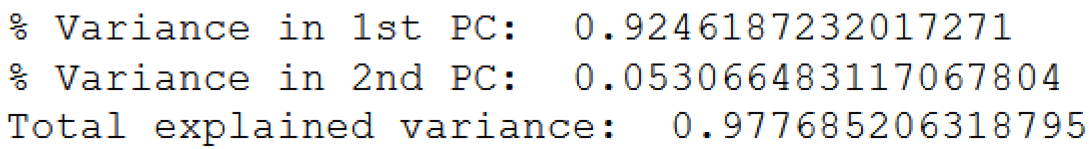


```python
pca = PCA(n_components=2).fit(X)
X_pca = pca.transform(X)
```

How much variance did we capture?



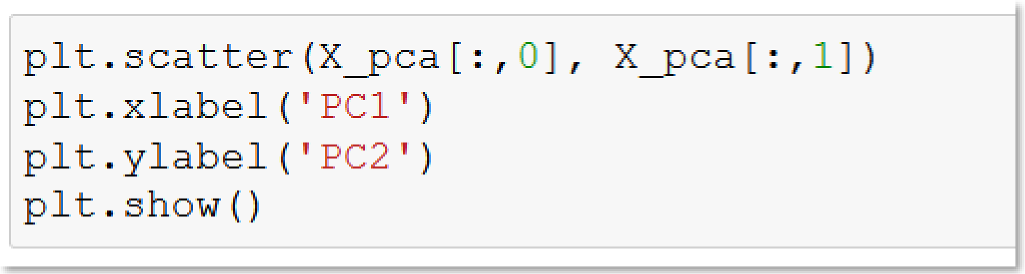
```python
expvar = pca.explained_variance_ratio_
print('% Variance in 1st PC: ', expvar[0])
print('% Variance in 2nd PC: ', expvar[1])
print('Total explained variance: ', sum(expvar))
```



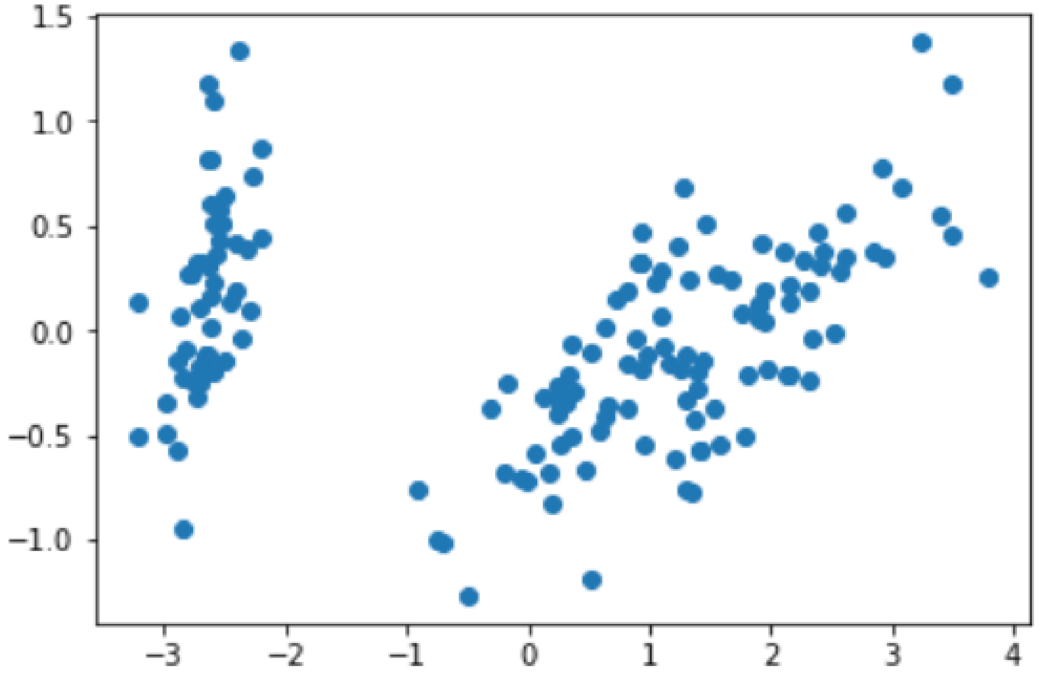
```
% Variance in 1st PC:  0.9246187232017271
% Variance in 2nd PC:  0.053066483117067804
Total explained variance:  0.977685206318795
```

Example : PCA on Iris Data

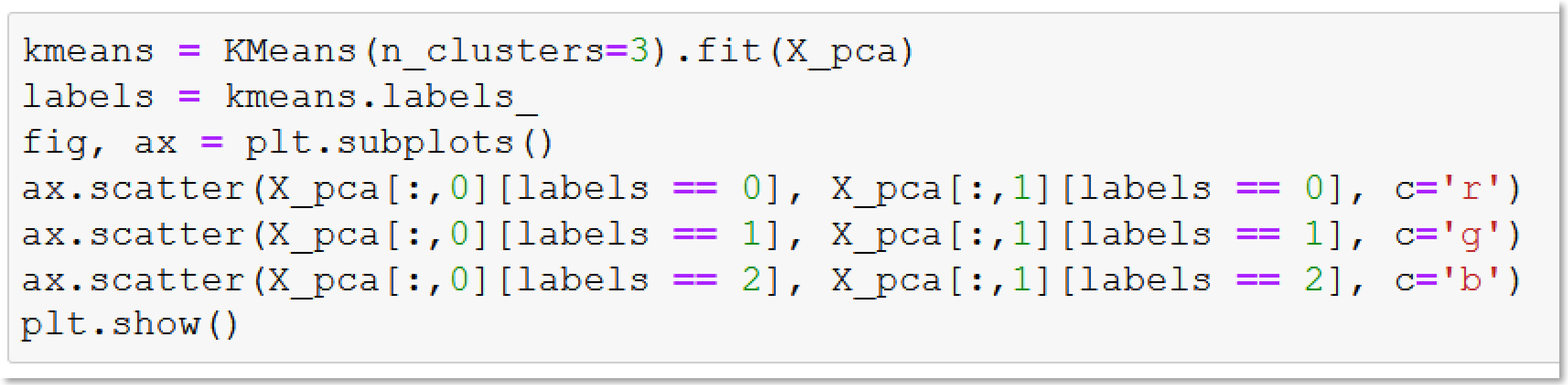
View data in 2-D subspace,



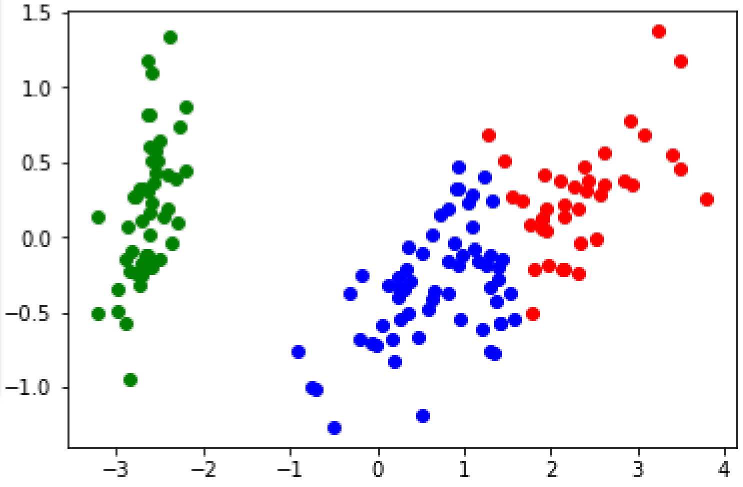
```
plt.scatter(X_pca[:,0], X_pca[:,1])
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```



Do K-means clustering in 2-D subspace,

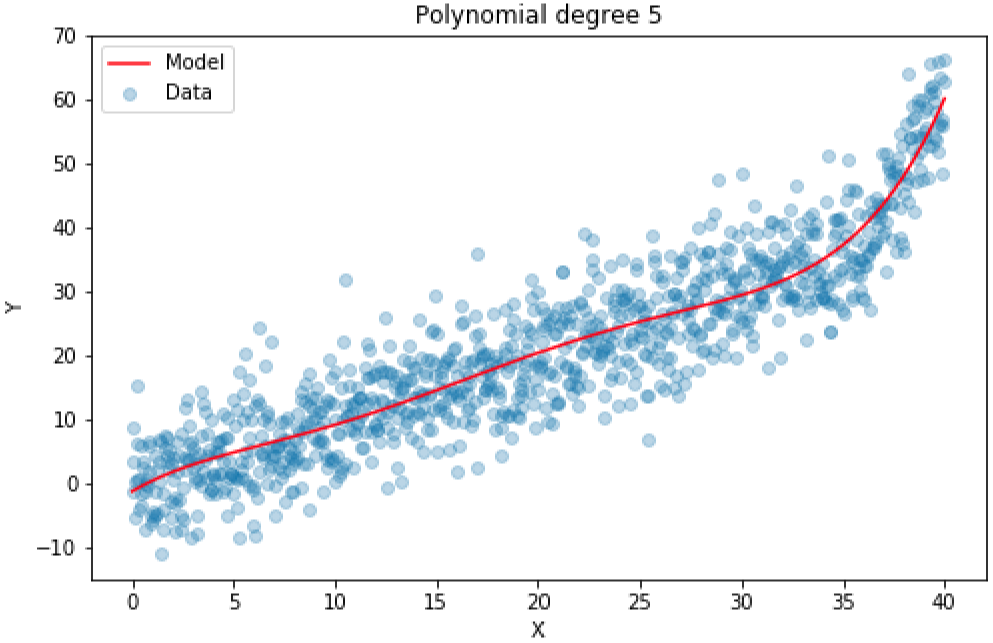


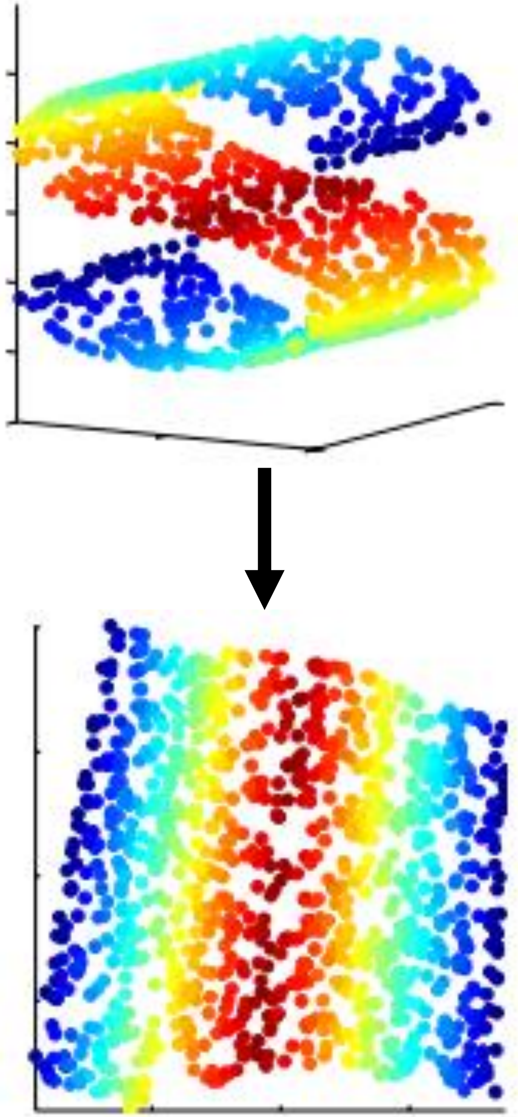
```
kmeans = KMeans(n_clusters=3).fit(X_pca)
labels = kmeans.labels_
fig, ax = plt.subplots()
ax.scatter(X_pca[:,0][labels == 0], X_pca[:,1][labels == 0], c='r')
ax.scatter(X_pca[:,0][labels == 1], X_pca[:,1][labels == 1], c='g')
ax.scatter(X_pca[:,0][labels == 2], X_pca[:,1][labels == 2], c='b')
plt.show()
```



Nonlinear Dimensionality Reduction

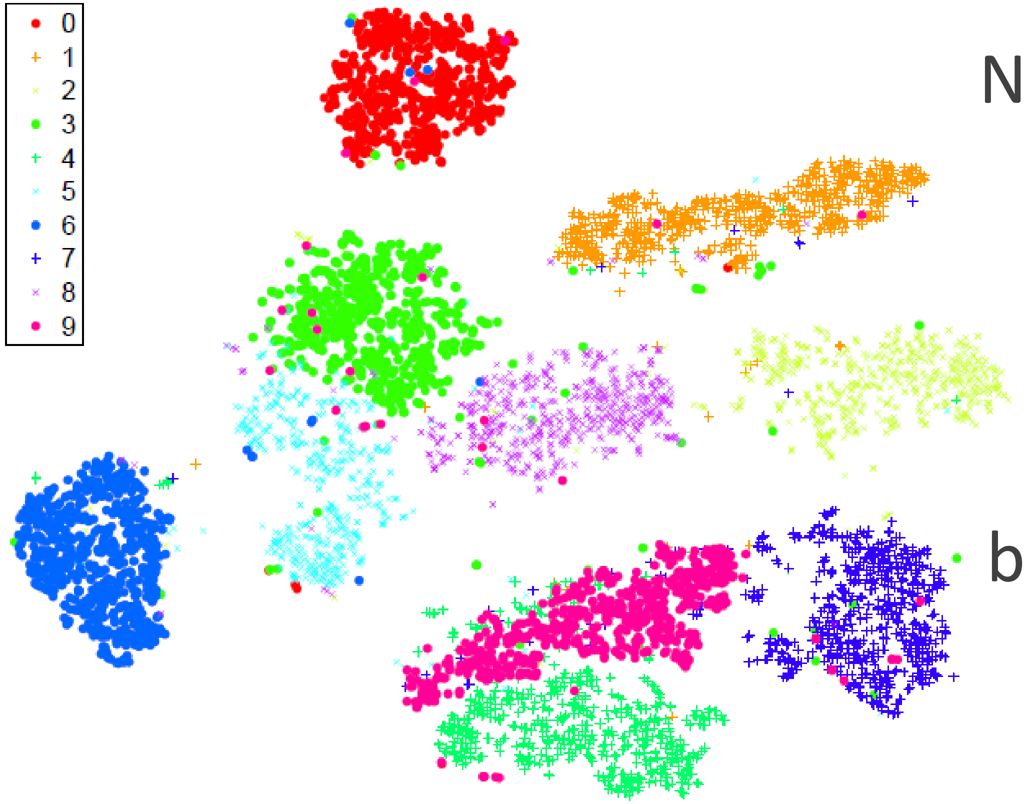
For general data, linear dimensionality reduction is not sufficient…





Many methods exist for nonlinear dimensionality reduction

t-SNE



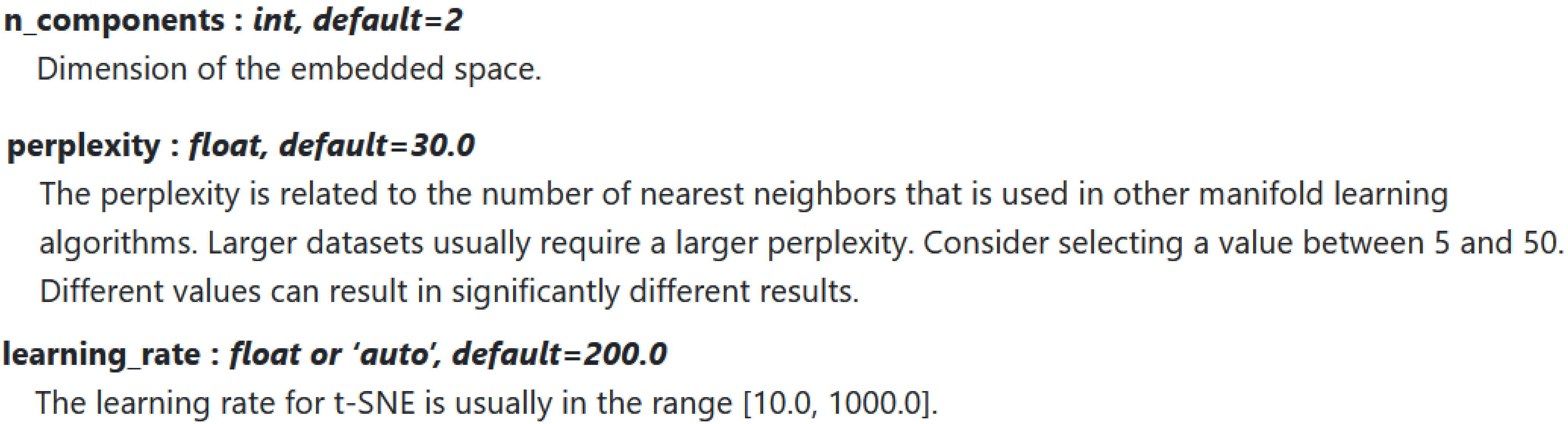
Nonlinear reduction can (potentially) amplify clustering properties

t-Distributed Stochastic Neighbor Embedding (t-SNE) Models similarity between data as a Student’s-t distribution in high / low dimensions and optimizes reduction to preserve similarity

Visualization shows MNIST digits projected to 2D

`sklearn.manifold`.TSNE

Parameters



n_components : *int, default=2*

Dimension of the embedded space.

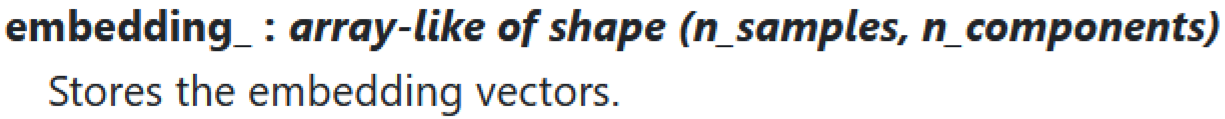
perplexity : *float, default=30.0*

The perplexity is related to the number of nearest neighbors that is used in other manifold learning algorithms. Larger datasets usually require a larger perplexity. Consider selecting a value between 5 and 50. Different values can result in significantly different results.

learning_rate : *float or 'auto', default=200.0*

The learning rate for t-SNE is usually in the range [10.0, 1000.0].

Attributes

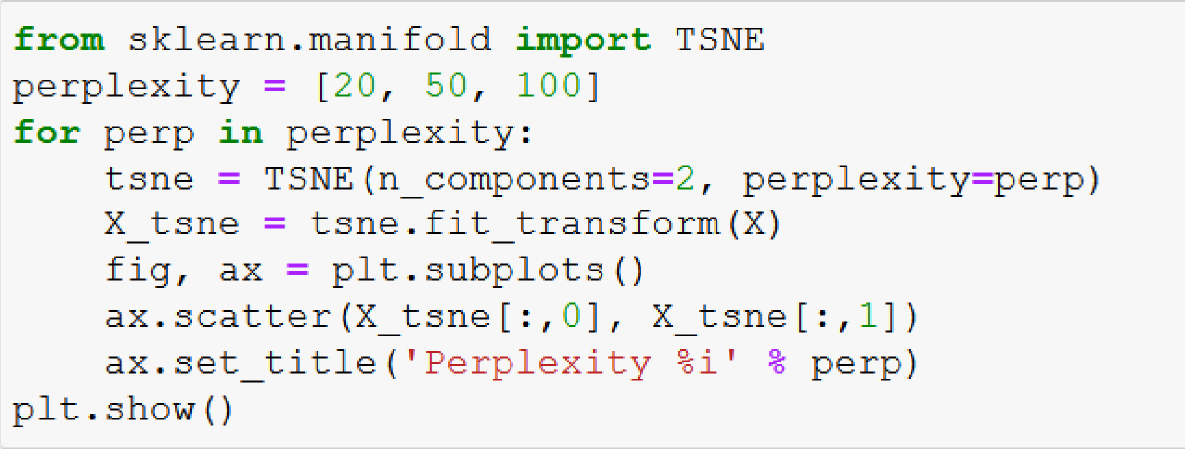


embedding_ : *array-like of shape (n_samples, n_components)*

Stores the embedding vectors.

Example : t-SNE on Iris Dataset

t-SNE can work surprisingly well…

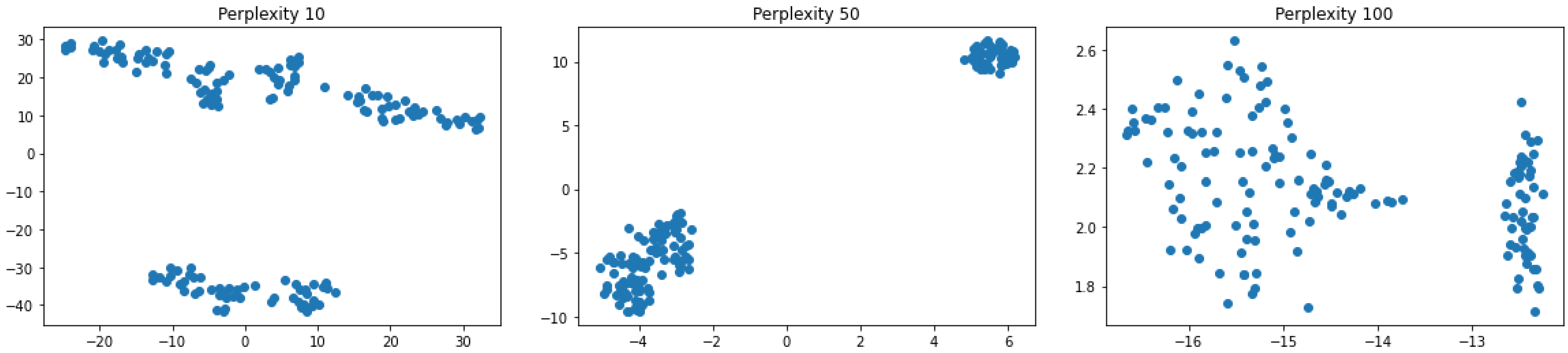


```
from sklearn.manifold import TSNE
perplexity = [20, 50, 100]
for perp in perplexity:
    tsne = TSNE(n_components=2, perplexity=perp)
    X_tsne = tsne.fit_transform(X)
    fig, ax = plt.subplots()
    ax.scatter(X_tsne[:,0], X_tsne[:,1])
    ax.set_title('Perplexity %i' % perp)
plt.show()
```

One advantage of PCA is that it has no parameters that need tuning (aside from number of PCs)

PCA is also much easier to interpret

…but can be a bit fussy about parameters and unreliable



Closing Comments

- Nonlinear methods in Scikit-Learn categorized under “manifold learning” in the *manifold* sub-package,
    - Isomap, Locally Linear Embedding, Spectral Embedding, Multidimensional scaling, and of course TSNE

- Other methods related to PCA (in *decomposition* sub-pkg):
    - Factor Analysis, Kernel PCA, Incremental PCA

- For multiple data sources, consider cross-decomposition
    - Canonical Correlation Analysis (CCA)
    - Learns same embedding for both spaces
    - Under *cross_decomposition* sub-package

Next time

- Probabilistic machine learning: Bayes networks

- Reading: CIML Chapter 9

Backup

Task 1 : Group These Set of Document into 3 Groups based on meaning

Doc1 : Health , Medicine, Doctor

Doc 2 : Machine Learning, Computer

Doc 3 : Environment, Planet

Doc 4 : Pollution, Climate Crisis

Doc 5 : Covid, Health , Doctor

Task 1 : Group These Set of Document into 3 Groups.

Doc1 : Health , Medicine, Doctor

Doc 2 : Machine Learning, Computer

Doc 3 : Environment, Planet

Doc 4 : Pollution, Climate Crisis

Doc 5 : Covid, Health , Doctor

Task 1 : Group These Set of Document into 3 Groups.

Doc1 : Health , Medicine, Doctor

Doc 5 : Covid, Health , Doctor

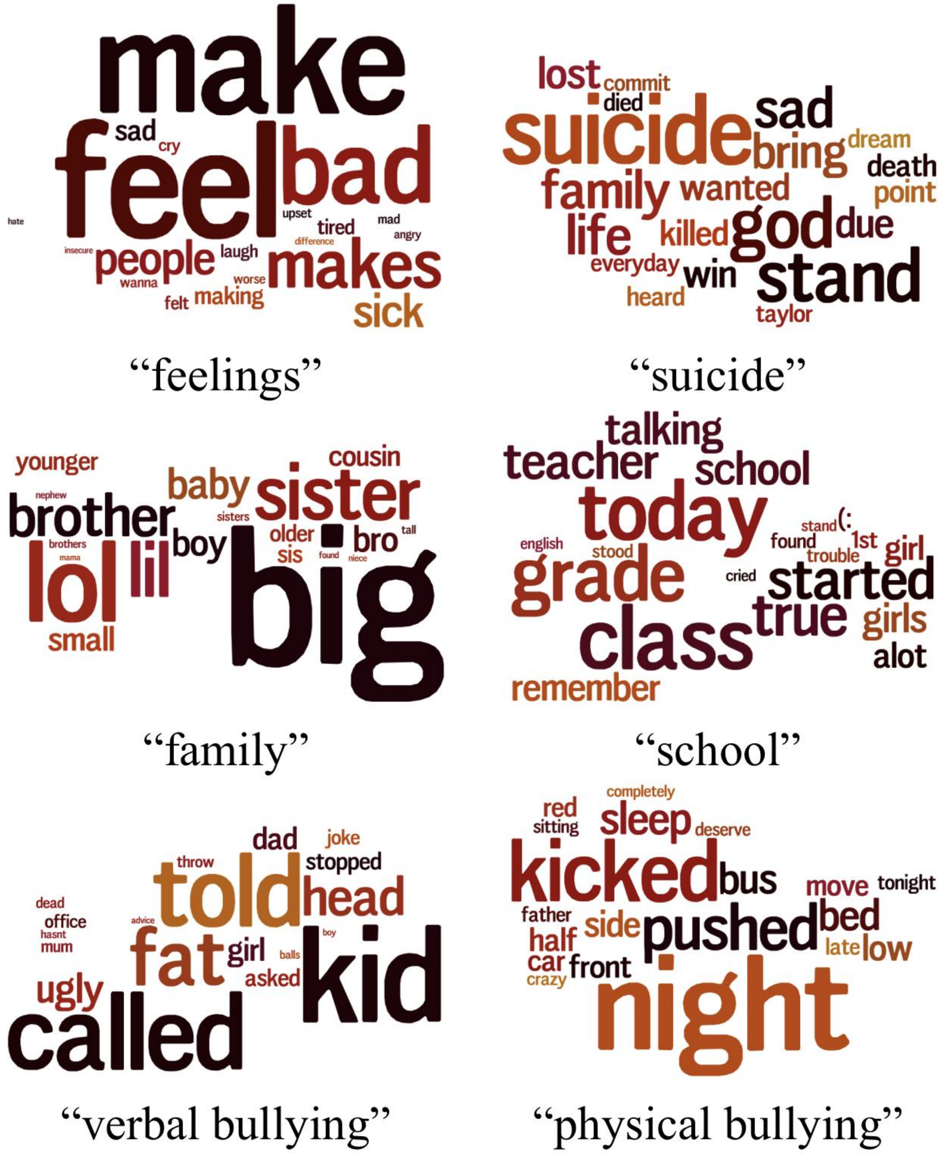
Doc 3 : Environment, Planet

Doc 4 : Pollution, Climate Crisis

Doc 2 : Machine Learning, Computer

Task 2: Topic modeling

- Provides a summary of a corpus.
- n tweets containing the keyword “bullying”, “bullied”, etc.
- Extracts k topics: each topic is a list of words with importance weights.
  - A set of words that co-occurs frequently throughout.



“feelings” “suicide”

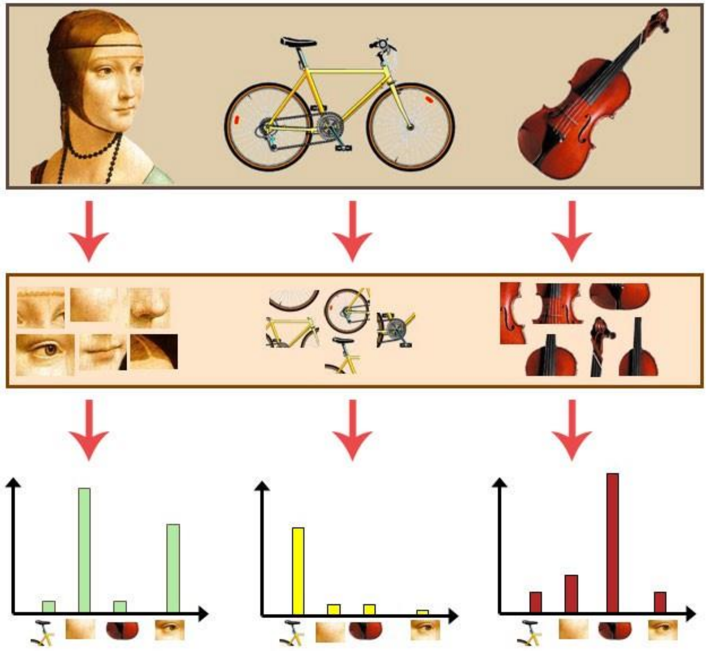
“family” “school”

“verbal bullying” “physical bullying”

Figure 4: Selected topics discovered by latent Dirichlet allocation.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, Amy Bellmore, “Learning from Bullying Traces in Social Media”

Application: Clustering for feature extraction



- Feature extraction: **histogram features (bag of visual words)**
- A set of images: $S = \{x_1, \dots, x_n\}$
- Cut up each $x_i \in \mathbb{R}^d$ into different parts $x_i^{(1)}, \dots, x_i^{(m)} \in \mathbb{R}^p$
  - e.g., small (overlapping) patches of an image
- Notation: $[n] \coloneqq \{1, \dots, n\}$
- Pool all the patches together: $P \coloneqq \left\{x_i^{(j)}\right\}_{i\in[n], j\in[m]}$
- Run clustering on P with #clusters=k $\Rightarrow$ for each $x_i^{(j)}$, we have a cluster assignment $A\left(x_i^{(j)}\right) \in [k]$
- Generate the feature vector of x_i as the histogram of $\left\{A\left(x_i^{(j)}\right)\right\}_{j\in[m]}$
  - i.e., $z = (z_1, \dots, z_k)$ where z_ℓ is the count of the cluster ℓ

https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb

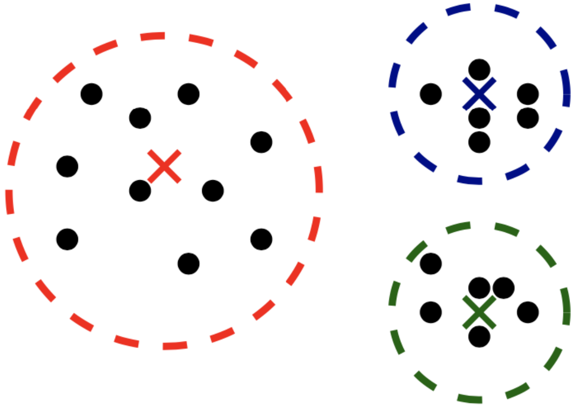
k-means clustering

- Idea: to partition the data, it would be great if someone gives us k reasonable centroids $c_1, \dots, c_k$, since then we can partition the data with them.

$$A(x) = \arg\min_{j\in[k]} \left\| x - c_j \right\|_2$$

- But we don’t have those centroids => Let’s find them with an optimization formulation.

$$\operatorname*{minimize}_{c_1,\dots,c_k} f(c_1, \dots, c_k), \text{ where } f(c_1, \dots, c_k) = \sum_{i=1}^{n} \min_{j\in[k]} \left\| x - c_j \right\|_2^2$$



Special case: k=1

- $\min_{c_1,\ldots,c_k} \sum_{i=1}^{n} \min_{j\in[k]} \left\|x_i - c_j\right\|_2^2$ => $\min_{c} \sum_{i=1}^{n} \|x_i - c\|_2^2$

- Let $F(c) = \sum_{i=1}^{n} \|x_i - c\|_2^2$ convex; minimizer c^* satisfies that $\nabla F(c^*) = 0$

  => $\sum_{i=1}^{n}(x_i - c^*) = 0$ => $c^* = \frac{1}{n}\sum_{i=1}^{n} x_i$

For $k \geq 2$

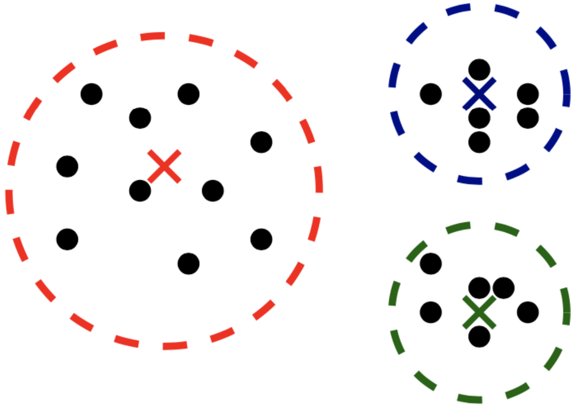
- $\underset{c_1,\dots,c_k}{\text{minimize}}\, f(c_1, \dots, c_k)$, where $f(c_1, \dots, c_k) = \sum_{i=1}^{n} \min_{j \in [k]} \left\| x - c_j \right\|_2^2$ => NP-hard even when $d = 2$

- **K-means algorithm**: solve it approximately (heuristic)

(Also called Lloyd's algorithm)

- Observation: The chicken-and-egg problem.
  - Cluster center location depends on the cluster assignment
  - Cluster assignment depends on cluster location

- Very common heuristic (that may or may not be the best thing to do)



Andrea Vattani, "the hardness of k-means clustering in the plane"

Clustering

EM for PCA

We can derive an *expectation maximization* (EM) algorithm for PCA…but why would we do this if PCA is closed-form?

For N data points of D-dimensions

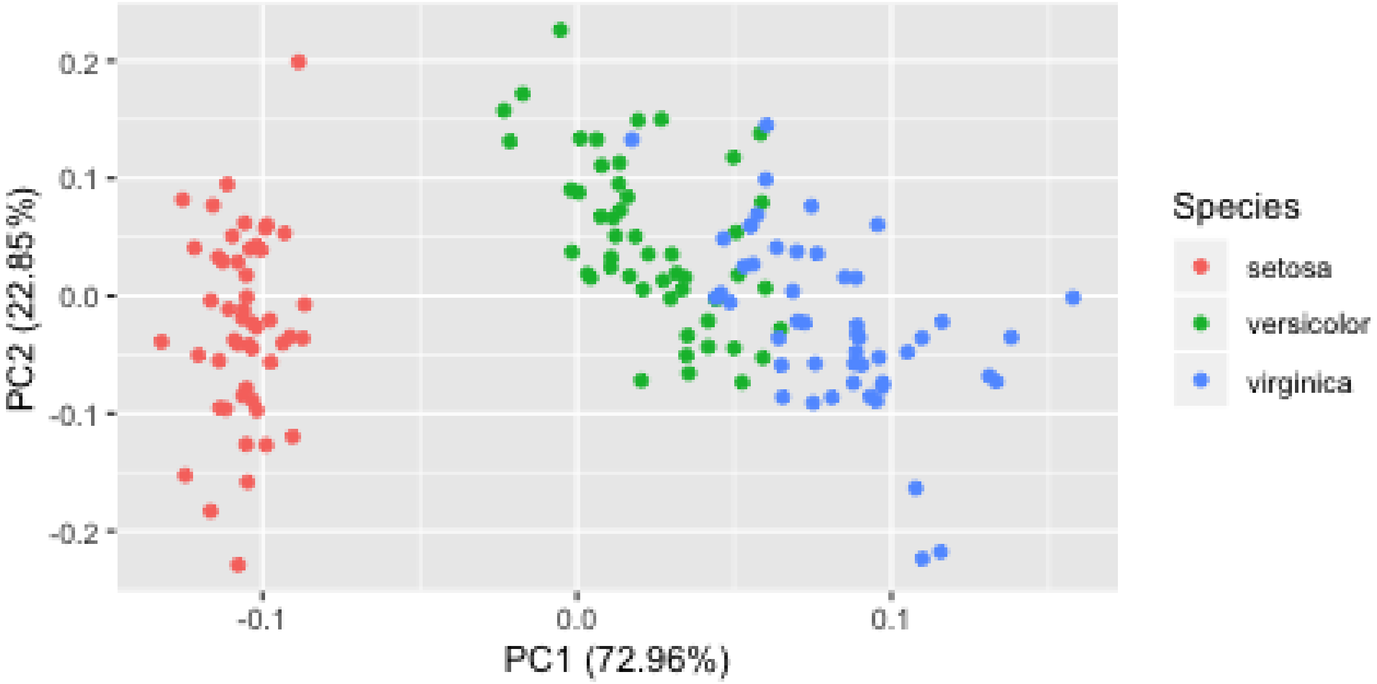
- Computing the first $M < D$ principal components takes $O(MD^2)$
- Evaluating the covariance needs $O(ND^2)$ time
- Most expensive step in EM is $O(NDM)$ time
- If D large and $M << D$ then $O(NDM) << O(ND^2)$

Dimensionality Reduction

and Principal Component Analysis (PCA)

Dimensionality reduction: motivation

- Data compression: Identifies important components that can reconstruct data points
- Identify informative feature transformations
- Visualization & visual analytics: high-dim data -> 2d => easy to plot



Iris flower dataset (4 features)

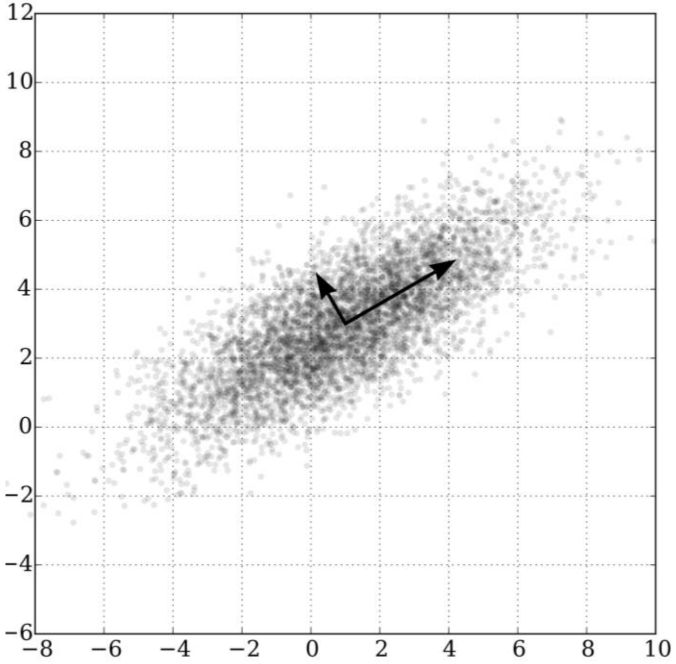
https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_pca.html

PCA: Introduction

- Task:
  - Given: raw feature vectors $x_1, \dots, x_n \in \mathbb{R}^d$, target dimension k
  - Output: a k-dimensional **subspace** represented by an *orthonormal* basis $q_1, \dots, q_k \in \mathbb{R}^d$ that the projections of datapoints with it would maximally preserve the ``spread''.

- Application: dimensionality reduction

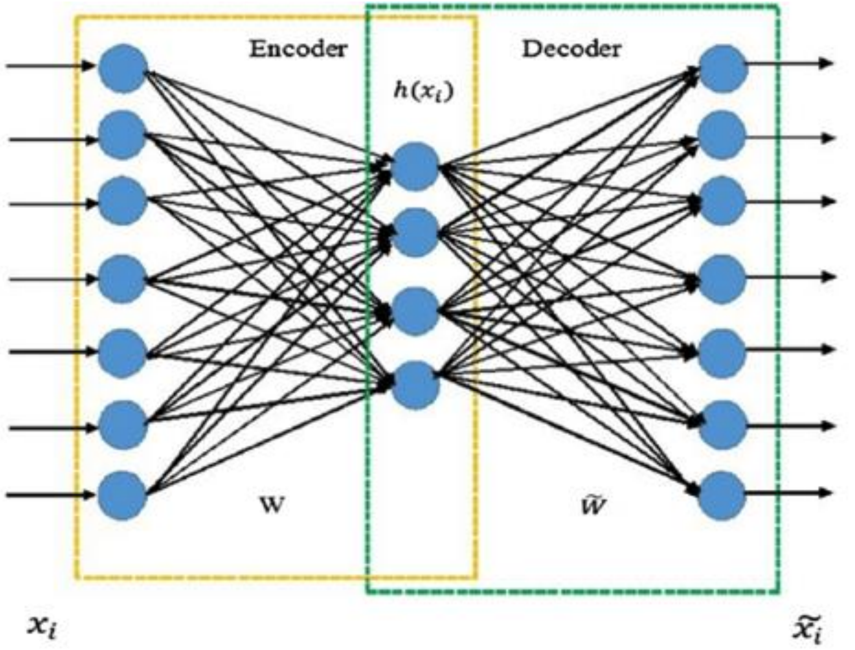
- Closely related to projections



if k=1, which basis should we choose?

Principal components: usage

- Compressing the data:
  - Let $Q = \begin{pmatrix} - \; q_1 \; - \\ \cdots \\ - \; q_k \; - \end{pmatrix} \in \mathbb{R}^{d \times k}$
  - $x_i \in \mathbb{R}^d$ mapped to 'encoding' $z_i = Qx_i = \begin{pmatrix} q_1^\top x_i \\ \ldots \\ q_k^\top x_i \end{pmatrix} \in \mathbb{R}^k$

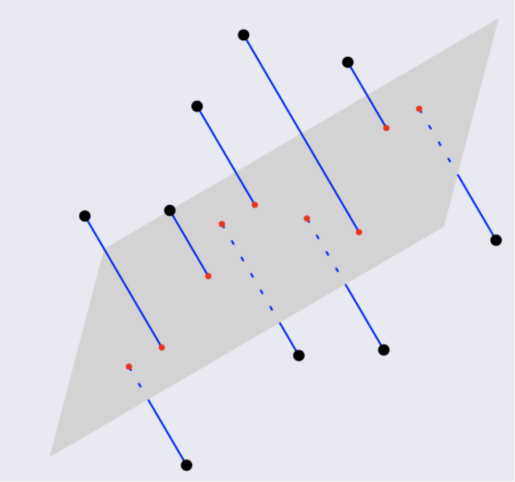


- Resconstructing the data ('decoding')
  - Given z_i, reconstruct x_i with $\widetilde{x_i} = \begin{pmatrix} | & \cdots & | \\ q_1 & \ldots & q_k \\ | & \cdots & | \end{pmatrix} z_i = Q^\top z_i$
  - Reconstruction error: $x_i - \widetilde{x_i} = x_i - Q^\top Q x_i$
  - If $k = d$, then perfect reconstruction ($\widetilde{x_i} = x_i$)

Projection

- Why reconstructing using $Q^\top z_i$?
- Given orthonormal $Q = \begin{pmatrix} - & q_1 & - \\ & \cdots & \\ - & q_k & - \end{pmatrix}$,

$$Q^\top Q x = \underbrace{\begin{pmatrix} | & \cdots & | \\ q_1 & \cdots & q_k \\ | & \cdots & | \end{pmatrix} \cdot \begin{pmatrix} - & q_1 & - \\ & \cdots & \\ - & q_k & - \end{pmatrix}}_{\text{projection matrix } \Pi = \sum_{i=1}^{k} q_i q_i^\top} x = \sum_i (q_i^\top x) q_i$$

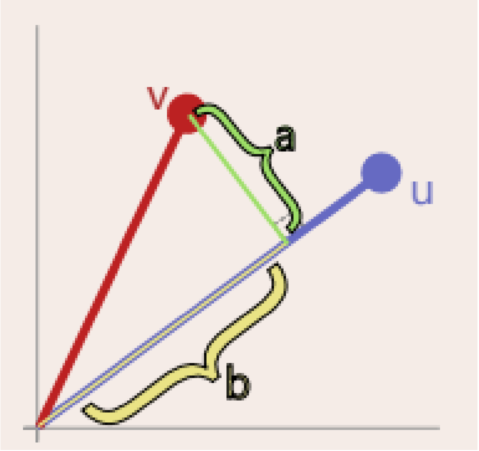


is also the *projection* of x to subspace $\mathrm{span}(q_1, \dots, q_k)$

- **Projection Objective**: find a k-dimensional **projection matrix** Π s.t. the average residual squared error (reconstruction error) is minimized:

$$\frac{1}{n}\left(\sum_{i=1}^{n} \|x_i - \Pi x_i\|_2^2\right)$$

Projection when k=1



- Objective:

$$\underset{q:\|q\|=1}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \|x_i - qq^\top x_i\|_2^2$$

- Observation: $qq^\top x_i$ and $x_i - qq^\top x_i$ are orthogonal, and sum to x_i
- Pythagorean theorem => $\|x_i - qq^\top x_i\|_2^2 = \|x_i\|_2^2 - \|qq^\top x_i\|_2^2 = \|x_i\|_2^2 - (q^\top x_i)^2$
- PCA optimization problem is thus equivalent to

$$\underset{q:\|q\|=1}{\operatorname{argmax}} \frac{1}{n} \sum_{i=1}^{n} (q^\top x_i)^2$$

- In matrix form, $\underset{q:\|q\|=1}{\operatorname{argmax}}\, q^\top \left(\frac{1}{n} X^\top X\right) q$

PCA as variance maximization

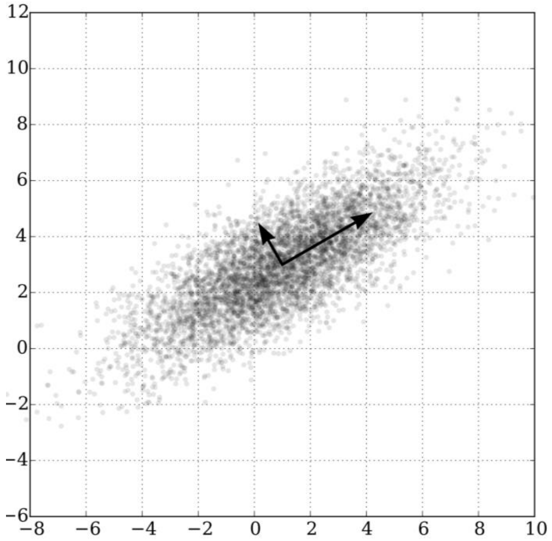
$$\underset{q:\|q\|=1}{\operatorname{argmax}} \frac{1}{n} \sum_{i=1}^{n} (q^\top x_i)^2$$

- $\frac{1}{n}\sum_{i=1}^{n}(q^\top x_i)^2 = \mathrm{E}_S[(q^\top x)^2]$

- If data is centered, i.e., $\mathrm{E}_S[x] = 0$

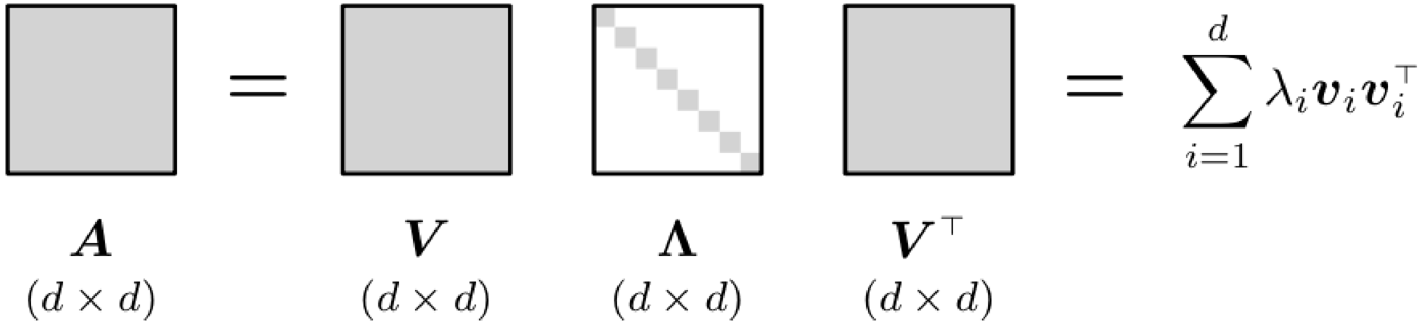
  $\Rightarrow$ the objective = $\mathrm{var}_S[q^\top x] = \mathrm{E}_S[(q^\top x - \mathrm{E}_S[q^\top x])^2]$

- PCA on centered data ⇔ Finding direction q, such that the projected data $\{q^\top x\}_{x\in S}$ has the maximum variance



Eigendecomposition for real symmetric matrices

- Fact: Every **Symmetric** **real** matrix A is guaranteed to have eigendecomposition with real eigenvalues:



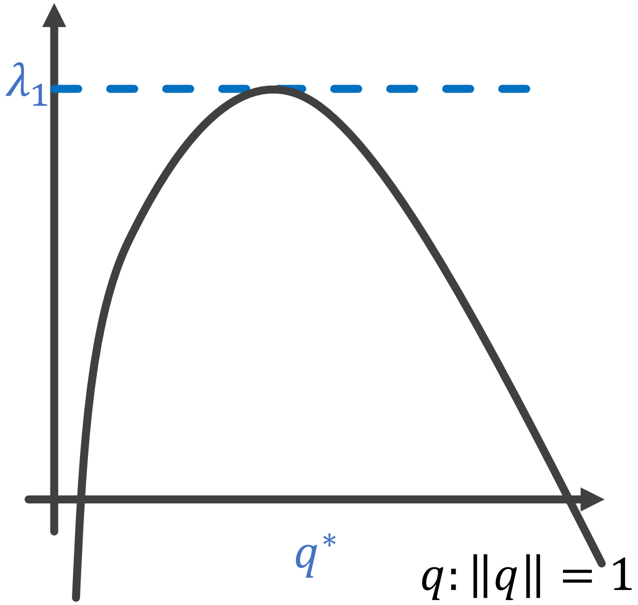
- Convention: $\lambda_1 \geq \cdots \geq \lambda_d$
- For positive semi-definite A, $\lambda_i \geq 0$ for all i
- Recall the definition of eigenvectors: $Av_i = \lambda_i v_i \ \forall i \in [d]$
- Here, $V = \begin{pmatrix} | & \cdots & | \\ v_1 & \dots & v_d \\ | & \cdots & | \end{pmatrix}$ has orthonormal columns, i.e. $v_i^\top v_j = I(i = j)$

Variational characterization of the top eigenvector

- Claim: $\max_{q:\|q\|=1} q^\top A q$ has a maximizer $q^* = v_1$, with maximum objective value λ_1
- Proof: recall $A = \sum_{i=1}^{n} \lambda_i v_i v_i^\top$
  - (Maximum objective upper bound): For any unit vector q,

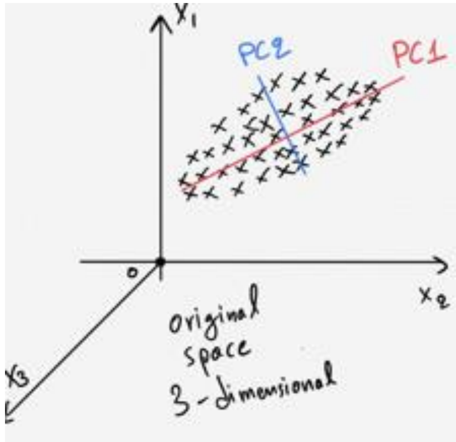
$$q^\top A q = \sum_{i=1}^{d} \lambda_i \left(v_i^\top q\right)^2 \leq \lambda_1,$$

since $\left(a_i = \left(v_i^\top q\right)^2\right)_{i=1}^{d}$ satisfies $\sum_{i=1}^{d} a_i = 1$ and $a_i \geq 0$ for all i



  - (The upper bound is achievable) $q^* = v_1$ satisfies that ${q^*}^\top A q^* = \lambda_1$

PCA with $k \geq 2$



$$\underset{Q \in \mathbb{R}^{d \times k}, Q^\top Q = I}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \|x_i - QQ^\top x_i\|_2^2$$

Equivalent to $\underset{Q \in \mathbb{R}^{d \times k}, Q^\top Q = I}{\operatorname{argmax}} \frac{1}{n} \sum_{i=1}^{n} \|Q^\top x_i\|_2^2$, i.e., $\underset{Q \in \mathbb{R}^{d \times k}, Q^\top Q = I}{\operatorname{argmax}} \operatorname{tr}\left(Q^\top \left(\frac{1}{n} X^\top X\right) Q\right)$,

where for $B \in \mathbb{R}^{d \times d}$, $\operatorname{tr}(B) = \sum_{i=1}^{d} B_{ii}$ is the *trace* of matrix B (Important property: $\operatorname{tr}(AB) = \operatorname{tr}(BA)$)

- Variance maximization interpretation:
  - For centered data, $Q^\top \left(\frac{1}{n} X^\top X\right) Q = \frac{1}{n} \sum_{i=1}^{n} (Q^\top x_i)(Q^\top x_i)^\top$ is the covariance matrix of $\{Q^\top x_i\}$'s
  - PCA chooses Q with the “largest” variance on projected data

PCA with $k \geq 2$

$$\underset{Q \in \mathbb{R}^{d \times k}, Q^\top Q = I}{\operatorname{argmax}} \quad \operatorname{tr}(Q^\top A\, Q)$$

- Fact: optimal Q has form $Q^* = \begin{pmatrix} | & \cdots & | \\ v_1 & \dots & v_k \\ | & \cdots & | \end{pmatrix}$, where A has eigendecomposition $A = \sum_i^d \lambda_i v_i v_i^\top$

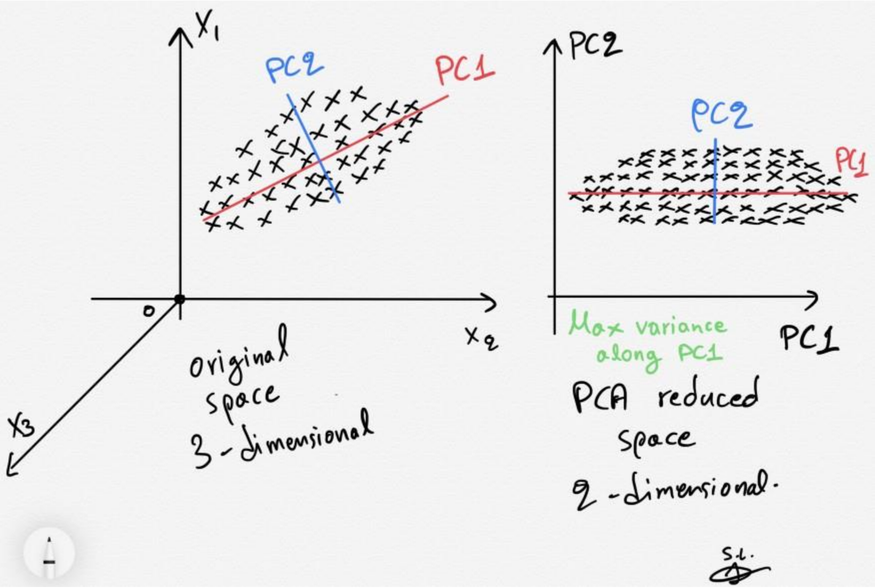
- In summary,

  k-dimensional subspace with smallest reconstruction error

= k-dimensional subspace with the maximum total variance

= top-k eigenvectors of $A = \frac{1}{n} X^\top X$

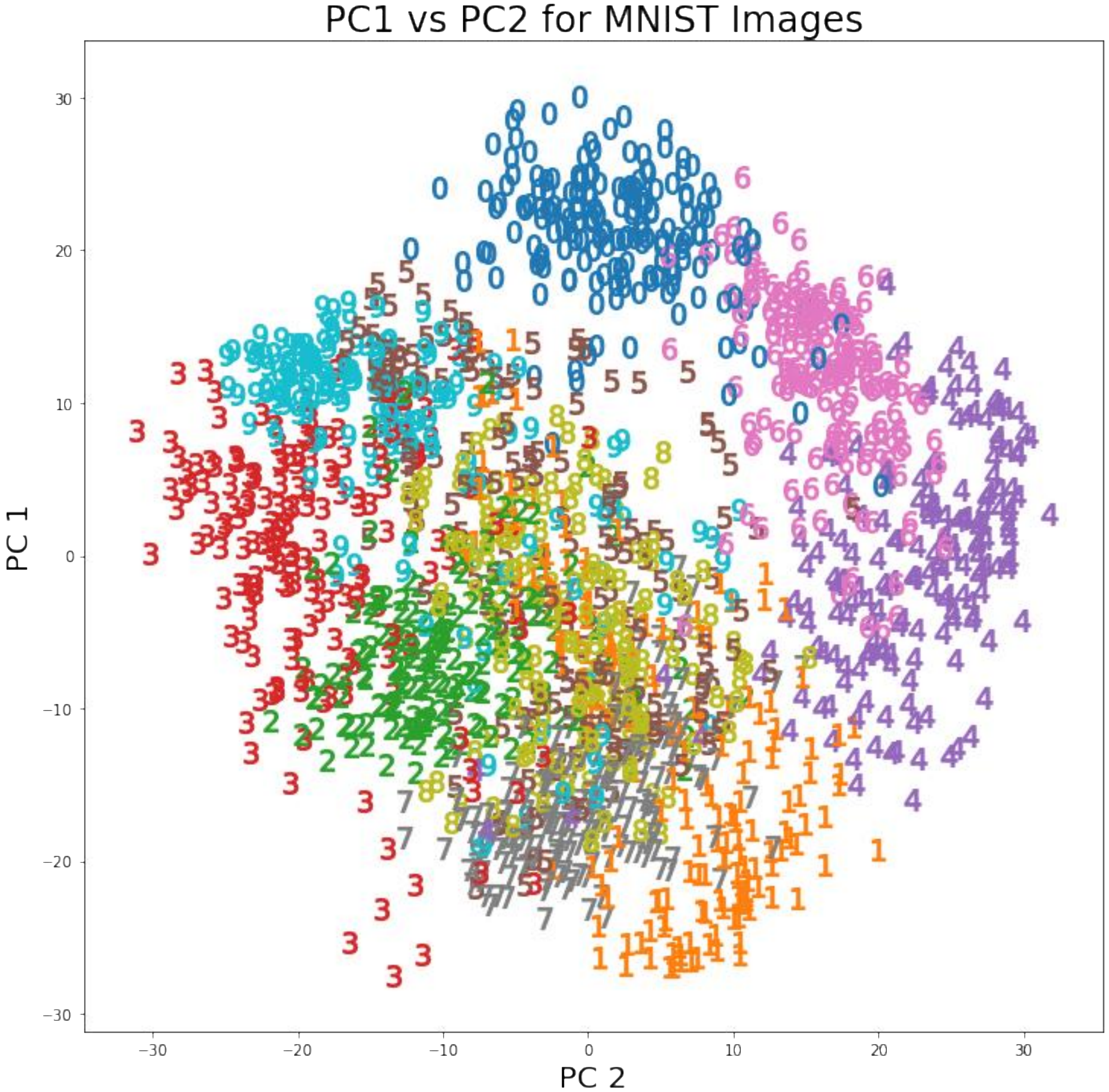
PCA pseudocode (with centering)



- Input: data matrix $X \in \mathbb{R}^{n \times d}$
- Centering: Let $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$. Compute $x_i' = x_i - \mu, \forall i \in [n]$
- Compute the top k eigenvectors $V = [v_1, \dots, v_k]$ of $\frac{1}{n}\sum_{i=1}^{n} x_i'(x_i')^\top$
- Feature map: $\phi(x) = \left(v_1^\top(x-\mu), \dots, v_k^\top(x-\mu)\right) \in \mathbb{R}^k$ (k-dimensional embedding)
- (thm) Decorrelating property (aka "whitening")
    - $\frac{1}{n}\sum_{i=1}^{n} \phi(x_i) = 0$
    - $\frac{1}{n}\sum_{i=1}^{n} \phi(x_i)\phi(x_i)^\top = \mathrm{diag}(\lambda_1, \dots, \lambda_k)$ λ_i is the eigen value (paired with v_i)
- (optional) Reconstruction (the actual projection): apply $\mu + V\phi(x) \in \mathbb{R}^d$
    - can be used as a ``denoising'' procedure.

https://towardsdatascience.com/pca-clearly-explained-how-when-why-to-use-it-and-feature-importance-a-guide-in-python-7c274582c37e

Example: MNIST dataset

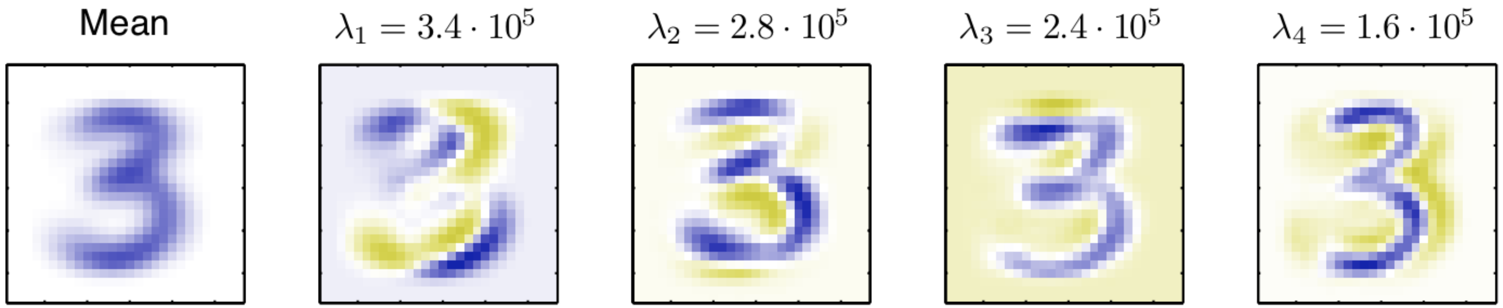


https://stats.stackexchange.com/questions/340175/why-is-t-sne-not-used-as-a-dimensionality-reduction-technique-for-clustering-or

Example: data compression

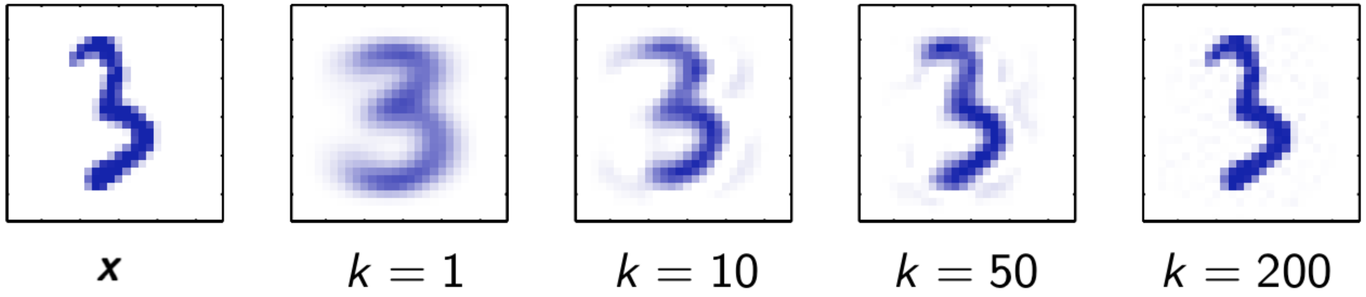
16×16 pixel images of handwritten 3s (as vectors in $\mathbb{R}^{256}$)

Mean $\boldsymbol{\mu}$ and eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4$



Mean $\lambda_1 = 3.4 \cdot 10^5$ $\lambda_2 = 2.8 \cdot 10^5$ $\lambda_3 = 2.4 \cdot 10^5$ $\lambda_4 = 1.6 \cdot 10^5$

Reconstructions:



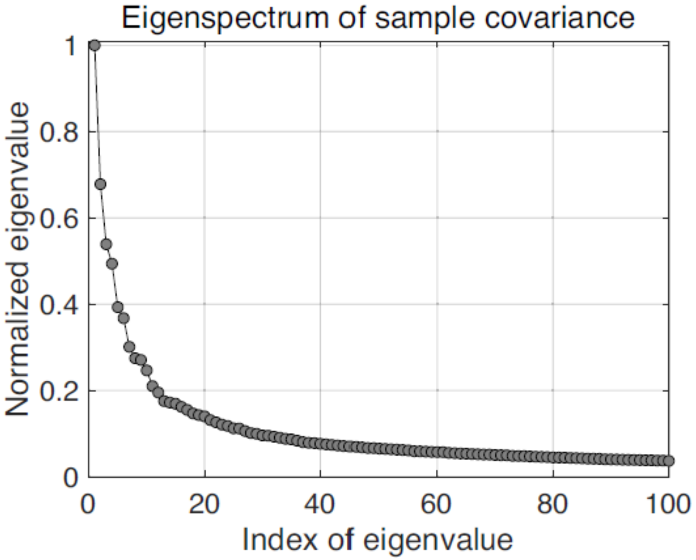
$\boldsymbol{x}$ $k = 1$ $k = 10$ $k = 50$ $k = 200$

Only have to store k numbers per image,
along with the mean $\boldsymbol{\mu}$ and k eigenvectors ($256(k+1)$ numbers)

Example: eigenfaces

The Yale Face Dataset; $n = 165, d = 243 \times 320 = 77760$

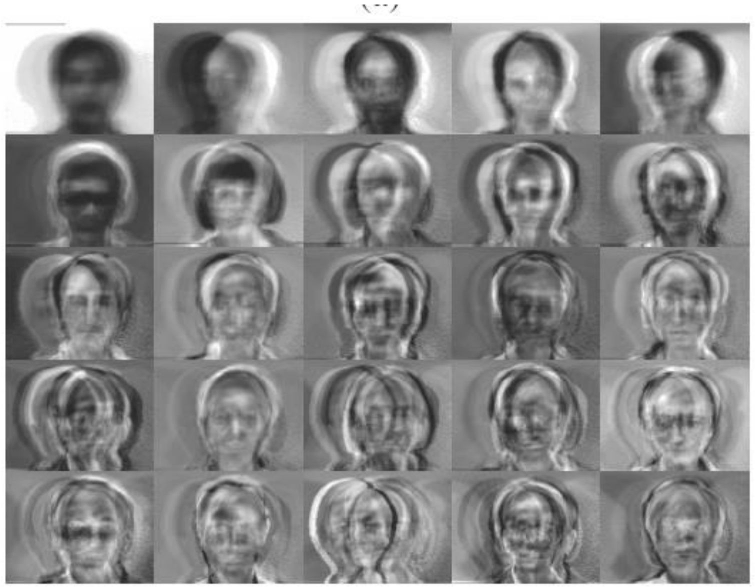
Eigenvalues of $A = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top$



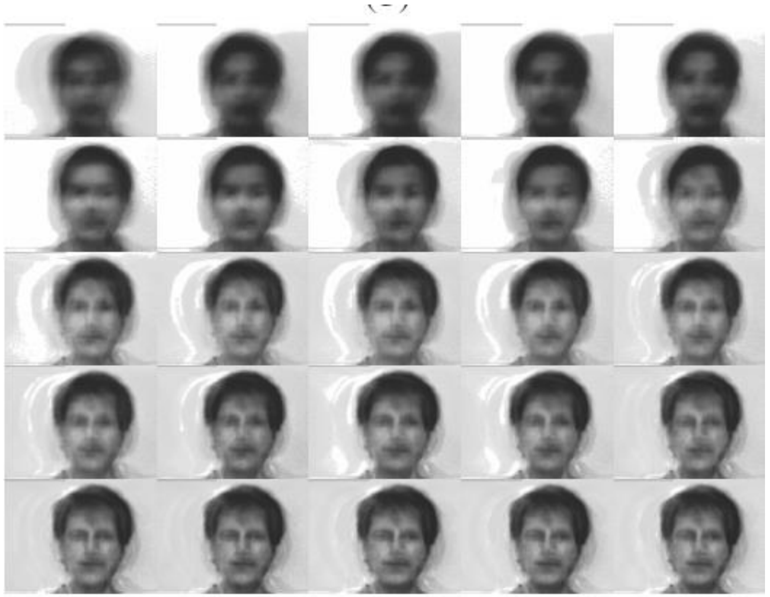
Wainwright, “High-dimensional statistics: an non-asymptotic viewpoint”; Yale Face Dataset

Example: eigenfaces (cont’d)

The average face, along with the top 24 PCs (eigenfaces)



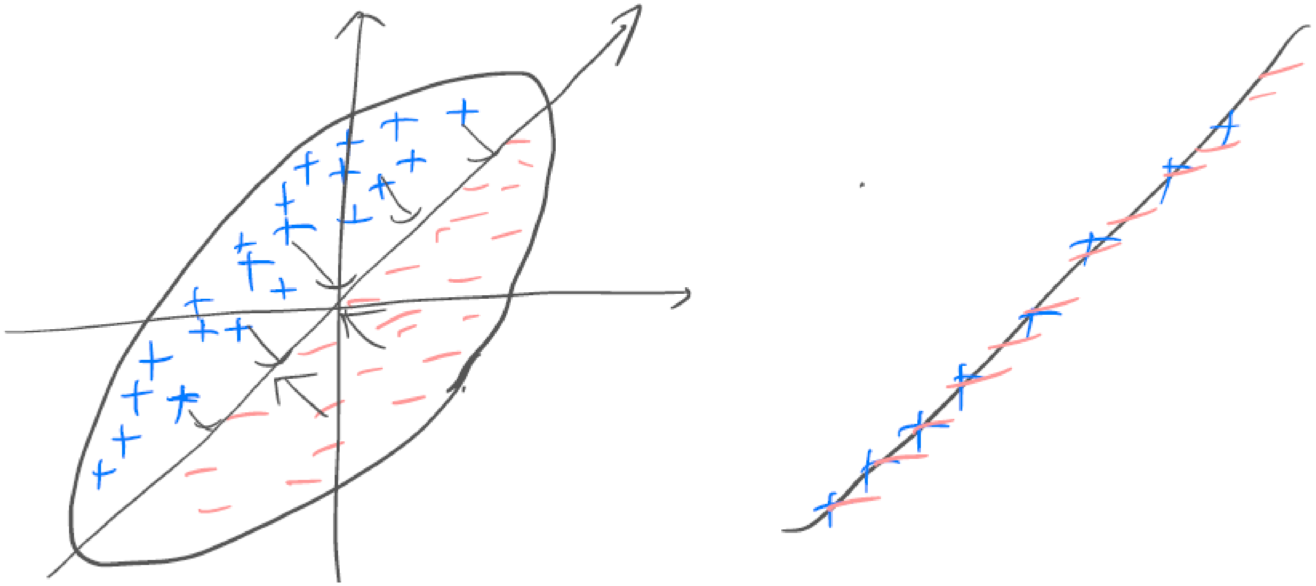
Reconstruction using the average face and the top PCs



Wainwright, “High-dimensional statistics: an non-asymptotic viewpoint”; Yale Face Dataset

PCA caveat

- The direction of maximizing variance is not necessarily useful for classification!



Next lecture (10/12)

- Probabilistic machine learning; naïve Bayes algorithm

- Assigned reading: CIML Sections 9.1-9.3

Eigendecomposition of real-symmetric matrices

Eigenvectors and Ellipses

How does this connect to PCA?

Take all points on a unit circle and apply the linear transformation S

If S is a *covariance matrix,* then points will be transformed into an ellipse and…

- Eigenvectors are axes of ellipse
- Eigenvalues are length of each axes
- Sort eigenvalues to get major / minor / etc. axes
- In the context of PCA eigenvectors = **principal components**