Data representation

UNIVERSITÀ DI PISA

Data representation

UNIVERSITÀ DI PISA

Original representations are defined in *raw* terms of the data, and not in terms of its intended *use*:

- Manipulation
- Exploration
- Visualization

How to best represent data?

How to represent data?

We will deal with two (out of many) approaches. Represent data...

By correlation

I want to represent data according to the correlation of the dataset

Algorithm: `PCA`

By neighborhood

I want to represent the data so that similar instances are similar

Algorithm: `t-SNE`.

By manifold

I want to represent the data so that its manifold is preserved

Algorithm: `UMAP`.

Principal Component Analysis (PCA)

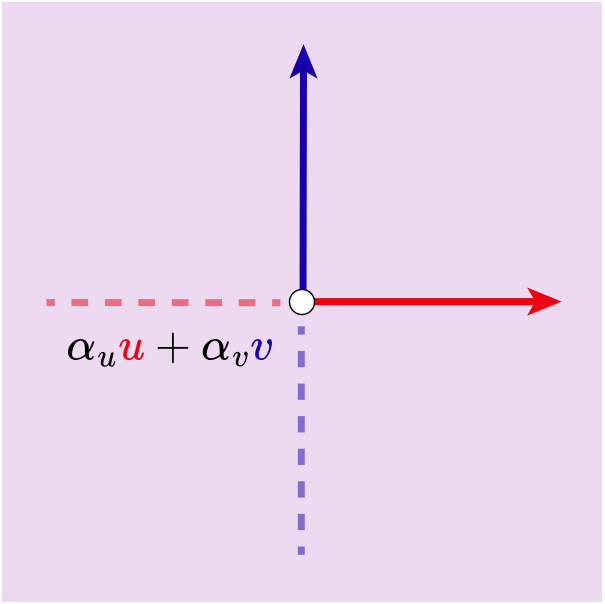
UNIVERSITÀ DI PISA

Vectors, linear combinations, and spaces

Università di Pisa

Vectors are m-dimensional elements in a field, and enjoy both addition and multiplication by scalar.

Composing these two, we can generate an infinite number of vectors: this is a **vector space**, and is defined by the *basis* vectors involved in the composition.



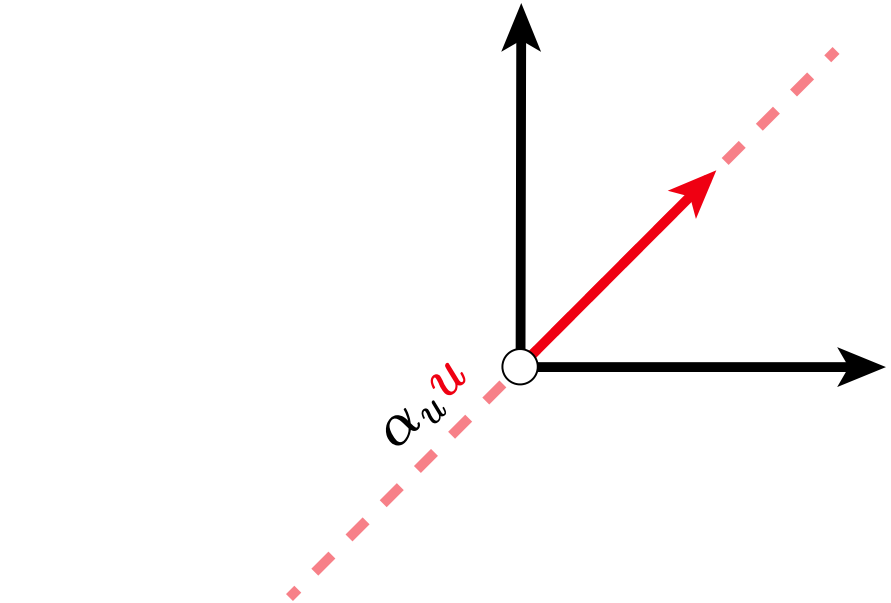
Two vectors u, v (in red and blue), and the plane spanned by all their linear combinations $\alpha_u u + \alpha_v v$ (in purple).

For simplicity, we consider Real fields.

Vectors, linear combinations, and spaces

Università di Pisa

The simplest linear combination: scaling.
Given a vector v, we have combination αv,
which defines a **direction** in the space.



A vector u, and the direction αu.

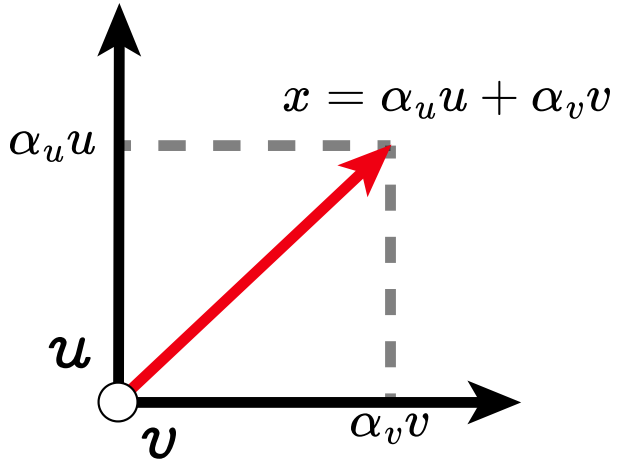
For simplicity, we consider Real fields.

Vectors as linear combinations

Università di Pisa

Given a suitable set of vectors, called *basis*, we can redefine every vector as a linear combination of the basis.
Protip: is it called **basis** because it defines the **basis coordinates** of the space!

Every vector $[x_1, \ldots, x_m]$ can be defined as a linear combination of the standard basis $[1, 0, \ldots, 0], [0, 1, \ldots, 0], \ldots, [0, 0, \ldots, 1]$, with coefficients $x_1, x_2, \ldots, x_m$.

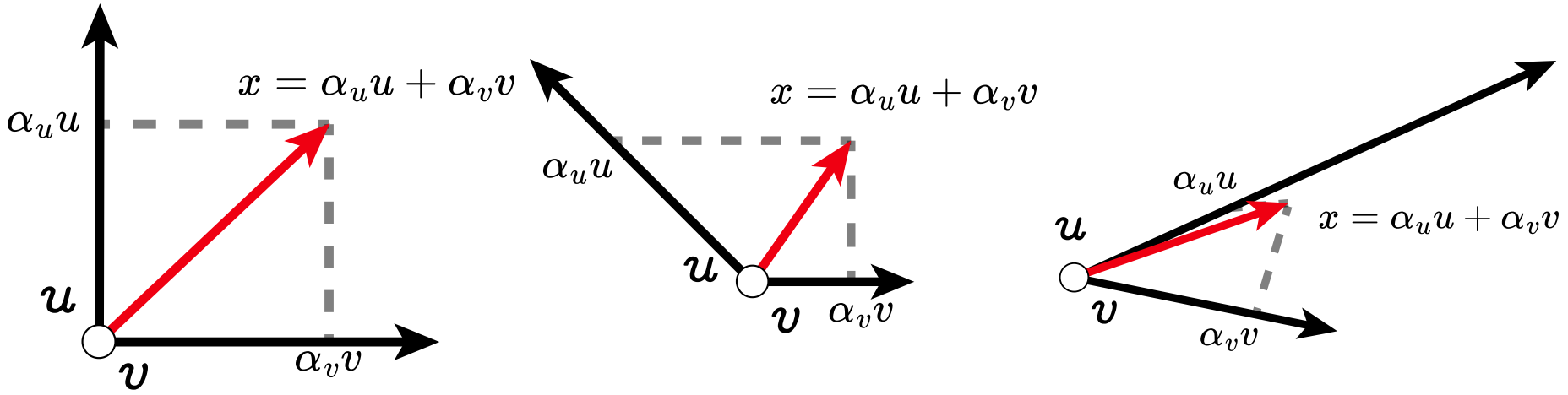


A vector x, defined as a linear combination $\alpha_u u + \alpha_v v$.

Changes of basis

UNIVERSITÀ DI PISA

If I change the space, where do the vectors end up? I simply redefine them... in terms of the new coordinates!

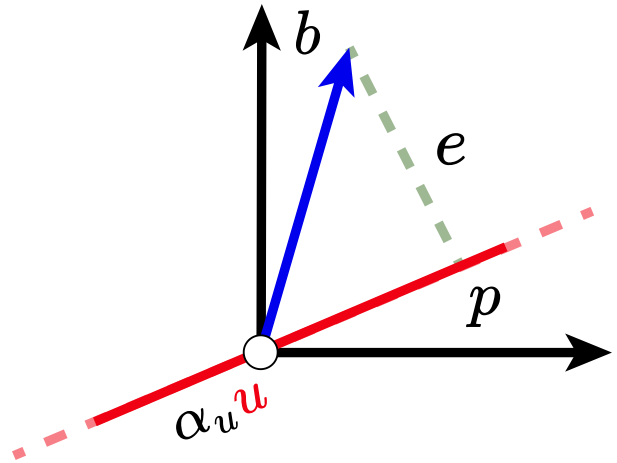


A vector x, defined as a linear combination $\alpha_u u + \alpha_v v$, on three different basis.

Subspaces and projections

Projections allows us to define a subspace-dependent view of a vector, mapping them in another subspace.

- How would this 3-dimensional vector look from this 2-dimensional plane?
- How would this 2-dimensional vector look from this 4-dimensional hyperplane?
- …



Projection p of a vector (in blue) on a subspace defined by vector u: the error e is perpendicular to the subspace.

They allows us to view the same object from different points of view at different dimensionionalities.

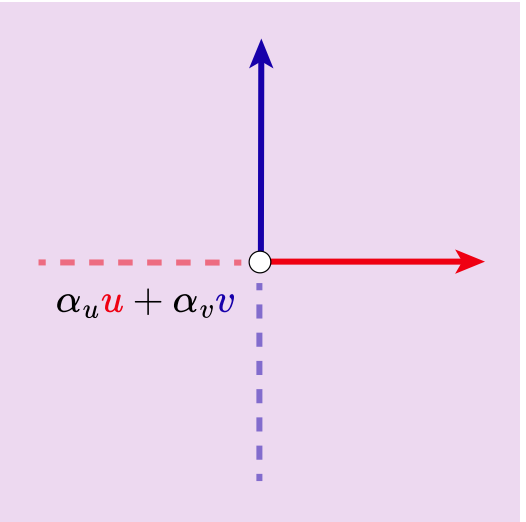
Vectors and matrices

Università di Pisa

We encode vectors in $(m \times n)$ matrices, easing computation of several interesting properties, including actual dimensionality. Since vectors define a space, **matrices define a space**!

We can define a vector in terms of a linear combination of the columns of a matrix through matrix-vector multiplication.

$$A = \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix}$$



A (2×2) matrix A, and the space spanned by it through linear combinations Ax.

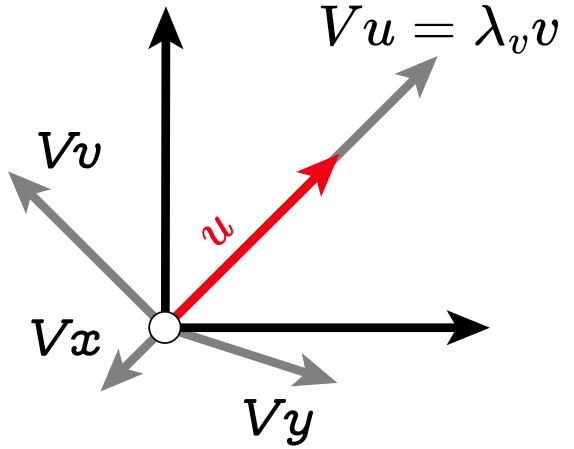
Matrices as linear transformations

A matrix A defines a space... and thus a linear transformation! Av linearly combines the columns of A with coefficients given by v, and yields:

- a rotation of v
- a scaling of v by some factor σ
- other more complex transformations.

With vectors... simply put them in a matrix too!

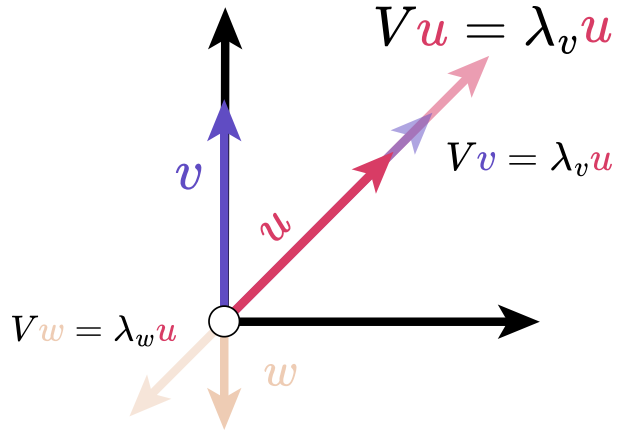
$$A \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$$



Linear transformations can rotate, scale, or otherwise *linearly* transform vectors.

Peculiar transformations: eigenpairs

Among all possible directions (subspaces defined by a single vector), some $v_1, \ldots, v_m$ are always scaled:
$Av_1 = [\lambda_1 v_1, \ldots, \lambda_m v_m]$.



A linear transformation V and its effects on vectors u, v, w: its eigenvector u is stretched by a factor of $\lambda_u u$.

Eigenvectors and eigenvalues

The eigenvectors $v_1, \ldots, v_m$ of a matrix A define the stretching of the space, and their eigenvalues $\lambda_1 > \cdots > \lambda_m$ define the stretching factor.

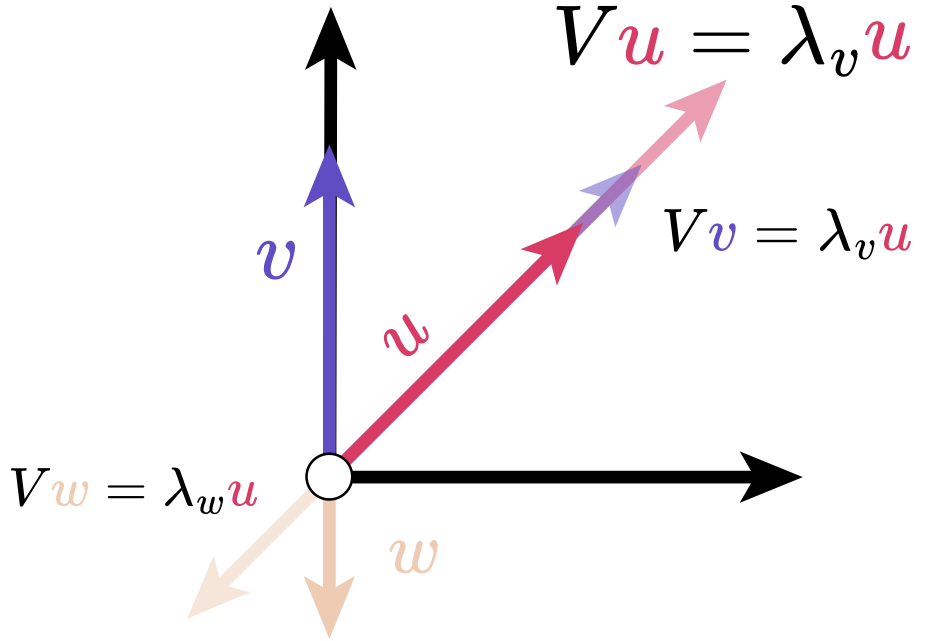
Symmetric matrices and eigenpairs

Among all possible matrices, symmetric matrices have a peculiar relationship with eigenpairs.

A symmetric matrix A of size $(m \times n)$:

- Always has $r = \min\{m, n\}$ *unique* eigenpairs
- The eigenvectors are orthogonal

Symmetric matrices may be rare... but given a matrix A, AA^T is always symmetric!



A linear transformation V and its effects on vectors u, v, w: its eigenvector u is stretched by a factor of $\lambda_u u$.

Symmetric matrices and eigenpairs

UNIVERSITÀ DI PISA

Spectral decomposition

A symmetric matrix A with eigenpairs $(v_1, \lambda_1), \ldots, (v_r, \lambda_r)$ admits a decomposition

$$A = V \begin{bmatrix} \lambda_1 & & \\ & \ldots & \\ & & \lambda_r \end{bmatrix} V^T,$$

where V is an orthogonal matrix.

We can redefine our data in terms of its directions!

Linking back to data representation

UNIVERSITÀ DI PISA

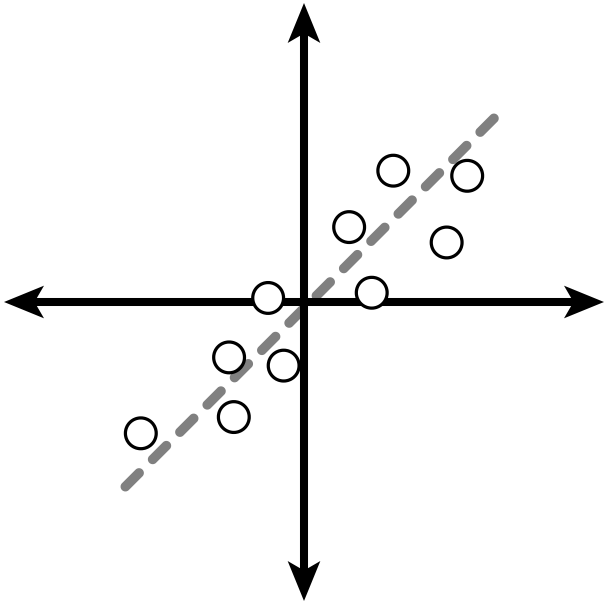
	allow us to...
Matrices	organize our data
Projection	map the data to another (more suitable) space
Eigenvectors	define the characteristic directions of the data
Eigenvalues	define the scaling across said directions

Principal component analysis (PCA)

Università di Pisa

Data can often be correlated, and linear dependencies can exist among variables, e.g.,

- Rent is linearly dependent on salary and food expenses
- Bank deposit is linearly dependent on salary and work
- Cardio is linearly dependent on hematocrit and VO_2max



A two-variables mean-centered dataset $\bar{X}$, and the slope between the variables.

Wouldn't it be nice to remove all such dependencies, and pack them together?

Principal component analysis (PCA)

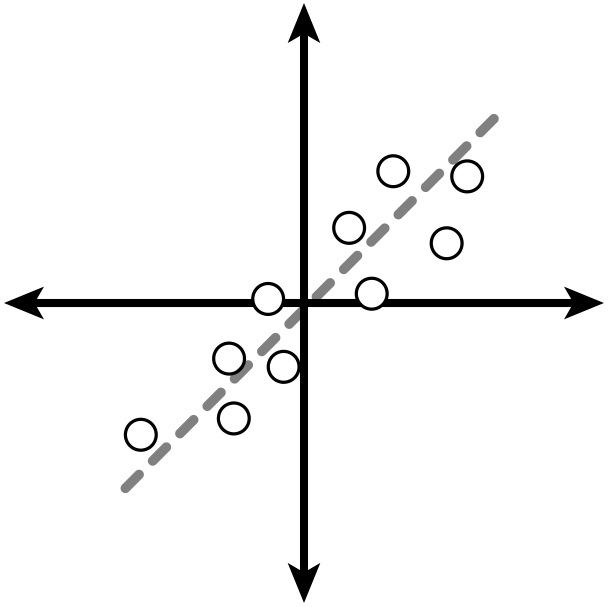
Università di Pisa

PCA projects some data X to $\hat{X}$ through a *linear* transformation A: $AX = \hat{X}$.

Fun fact #1: for a mean-centered $\bar{X}$, the slope is directly proportional to the covariance!

$$\Sigma = \begin{bmatrix} \sigma^2_{\bar{X}^1} & \dots & cov(\bar{X}^1, \bar{X}^n) \\ & \dots & \dots \\ & & \sigma^2_{\bar{X}^n} \end{bmatrix}$$

Fun fact #2: we can measure covariance (and thus slope) through matrix multiplication $\bar{X}\bar{X}^T$.



A two-variables mean-centered dataset $\bar{X}$, and the slope between the variables.

For those interested, see the `slopes and covariance.pdf` supplement.

PCA and the covariance matrix $\bar{\Sigma}$

PCA aims to embed collinearity in a set of novel features: each novel feature is defined in terms of linear combinations of other features.

With collinearity already embedded, are the novel features collinear?

PCA and the covariance matrix $\bar{\Sigma}$

Università di Pisa

PCA aims to embed collinearity in a set of novel features: each novel feature is defined in terms of linear combinations of other features.

With collinearity already embedded, are the novel features collinear? **No!**

PCA aims to transform $\bar{X}$ into $\hat{X}$ with zero covariances:

$$\bar{\Sigma} = \begin{bmatrix} \sigma^2_{\bar{X}^1} & \dots & \dots & cov(\bar{X}^1, \bar{X}^n) \\ & \dots & \dots & \dots \\ & & & \sigma^2_{\bar{X}^n} \end{bmatrix} \rightarrow \hat{\Sigma} = \begin{bmatrix} \sigma^2_{\hat{X}^1} & 0 & 0 & 0 \\ 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\hat{X}^n} \end{bmatrix}.$$

PCA: mathematical formulation

Università di Pisa

We start with defining our result $\hat{X} = A\bar{X}$, and its symmetric covariance matrix $\hat{\Sigma}$, which we wish to diagonalize:

$$(n-1)\hat{\Sigma} = \hat{X}\hat{X}^T.$$

With $\hat{X} = A\bar{X}$, we can rewrite this as

$$(n-1)\hat{\Sigma} = A\bar{X}(A\bar{X})^T = A\bar{X}\bar{X}^T A^T$$

Remember: the transpose of a product is the same product with inverted orders, and its components transposed: $(AB)^T = B^T A^T$.

PCA: mathematical formulation

We start with defining our result $\hat{X} = A\bar{X}$, and its symmetric covariance matrix $\hat{\Sigma}$, which we wish to diagonalize:

$$(n-1)\hat{\Sigma} = \hat{X}\hat{X}^T.$$

With $\hat{X} = A\bar{X}$, we can rewrite this as

$$(n-1)\hat{\Sigma} = A\bar{X}(A\bar{X})^T = A\bar{X}(A\bar{X})^T = A\underbrace{\bar{X}\bar{X}^T}_{=(n-1)\bar{\Sigma}}A^T = A\bar{\Sigma}A^T.$$

Remember: the transpose of a product is the same product with inverted orders, and its components transposed: $(AB)^T = B^T A^T$.

PCA: leveraging eigenvectors

As a symmetric matrix, $\bar{\Sigma}$ enjoys an orthogonal eigenvalue decomposition $\bar{\Sigma} = \bar{V}\bar{\Lambda}\bar{V}^T$.

Now, let us plug that back in $(n-1)\hat{\Sigma}$:

$$(n-1)\hat{\Sigma} = A\bar{\Sigma}A^T = A(\bar{V}\bar{\Lambda}\bar{V}^T)A^T.$$

Remember: our goal is to make this matrix **diagonal**! How do we do it?

PCA: leveraging eigenvectors

As a symmetric matrix, $\bar{\Sigma}$ enjoys an orthogonal eigenvalue decomposition $\bar{\Sigma} = \bar{V}\bar{\Lambda}\bar{V}^T$.
Now, let us plug that back in $(n-1)\hat{\Sigma}$:

$$(n-1)\hat{\Sigma} = A\bar{\Sigma}A^T = A(\bar{V}\bar{\Lambda}\bar{V}^T)A^T.$$

Remember: our goal is to make this matrix **diagonal**! We already have a diagonal matrix ($\bar{\Lambda}$), ideally we'd like to remove all the other factors. We can do so because A is our unknown! Let us set A to $\bar{V}^T$:

$$(n-1)\hat{\Sigma} = A(\bar{V}\bar{\Lambda}\bar{V}^T)A^T = \bar{V}^T\bar{V}\bar{\Lambda}\bar{V}^T\bar{V} \qquad \text{for } A = V^T$$

In the last step, $A^T = (\bar{V}^T)^T$ resolves to $\bar{V}$ because of double transpose.

PCA: leveraging eigenvectors

Again by property of symmetric matrices, the eigendecomposition yields orthogonal eigenvector matrices, that is, matrices V whose inverse is the transpose, that is, V is such that $V^{-1}V = V^TV = I$. This results in

$$
\begin{aligned}
(n-1)\hat{\Sigma} &= A(\bar{V}\bar{\Lambda}\bar{V}^T)A^T = \bar{V}^T\bar{V}\bar{\Lambda}\bar{V}^T\bar{V} && \text{for } A = V^T \\
(n-1)\hat{\Sigma} &= \underbrace{\bar{V}^{-1}\bar{V}}_{=I}\,\bar{\Lambda}\,\underbrace{\bar{V}^{-1}\bar{V}}_{=I} \\
(n-1)\hat{\Sigma} &= \bar{\Lambda},
\end{aligned}
$$

which gives us the diagonal covariance matrix $\hat{\Sigma}$ we were looking for. It follows that the linear PCA transformation is given by $A = V^T$, the transpose of the eigenvectors matrix of $\bar{X}$.

PCA algorithm: a summary

1. Mean-center your data X, obtaining $\bar{X}$
2. Compute its eigenvectors matrix $\bar{V}$.
3. Transpose V to obtain the transformation matrix V^T.
4. Project $\bar{X}$ through $V^T\bar{X}$, obtaining the PCA-transformed data $\hat{X}$.

Using the PCA

Università di Pisa

Observations

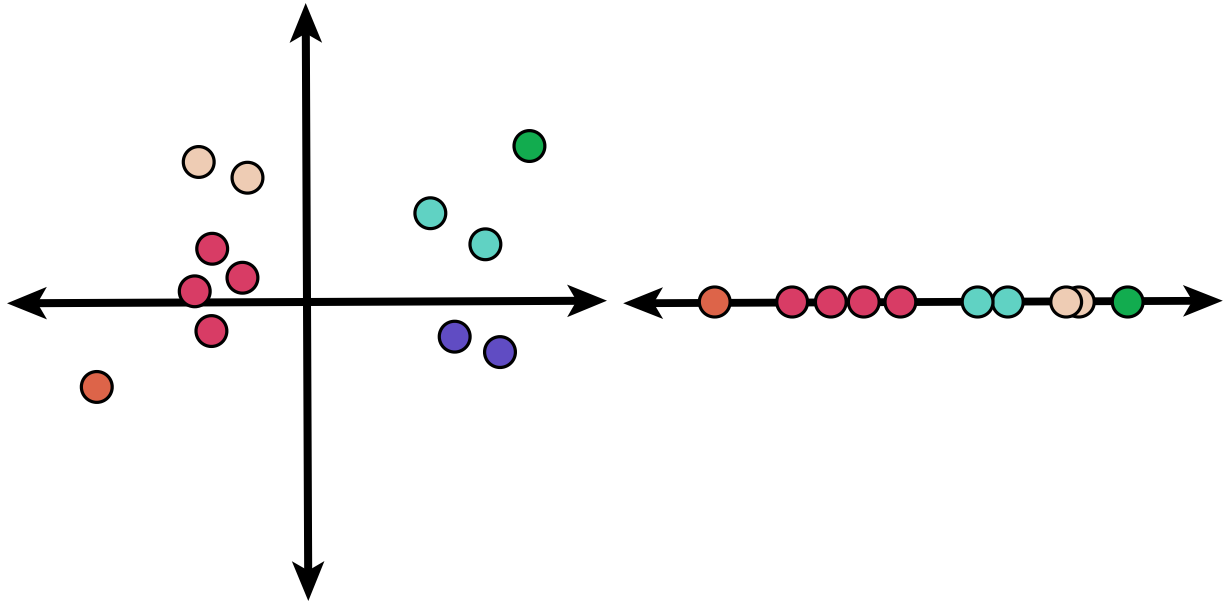
- PCA redefines data by removing collinearity: if your data has low covariance, the transformation will have minimal effect.
- PCA performs a *linear* transformation to tackle *linear* relationships between variables. Nonlinear relationships are not influenced.

Uses

- Feature selection: high covariance of a feature may indicate disposability.
- Dimensionality reduction: trimming columns of $\hat{X}$ lets us reduce the dimension of the resulting data.
- Clustering preprocessing: correlated features inflate object similarity.

Using the PCA (poorly)

By collapsing covariant variables, instances may collapse together.



2-dimensional instances, and a 1-dimensional mapping.

What if we instead represent by neighborhood?

Geoffrey Hinton, Stochastic neighbor embedding. 2002

t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE focuses on data clusters rather than subspace representation, and again maps the original data X to a representation $\hat{X}$.

t-SNE tackles this problem in two phases:

1. **Similarity phase** In the original space $\mathcal{X}$, how similar is x_i to x_j?
2. **Embedding phase** In the mapped space $\hat{\mathcal{X}}$, how similar is $\hat{x}_i$ to $\hat{x}_j$?

Similarity phase

UNIVERSITÀ DI PISA

How similar is x_i to x_j? Even better, **what is the probability that x_j is a neighbor of x_i?**

Neighboring phase

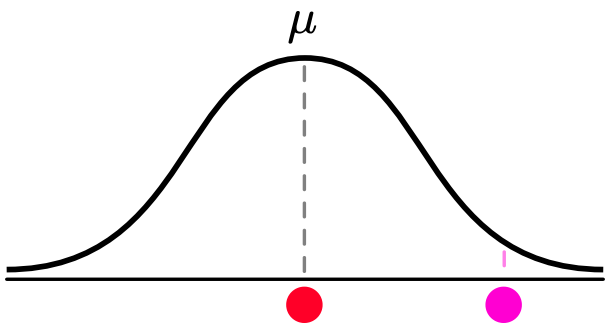
How similar is x_i to x_j? Even better, **what is the probability that x_j is a neighbor of x_i?**

Neighboring through distribution.

Every instance x_i defines a probability distribution $P_i = \mathcal{N}(x_i, \sigma_i^2)$ of neighboring.

$p_{j|i} = P_i(x_j)$ estimates the probability of x_j being a neighbor of x_i on the basis of their euclidean distance $|| x_i - x_j ||$.

In general, $\sigma_i^2 \neq \sigma_j^2$, hence $p_{i,j} = \frac{1}{2}(p_{j|i} + p_{i|j})$.

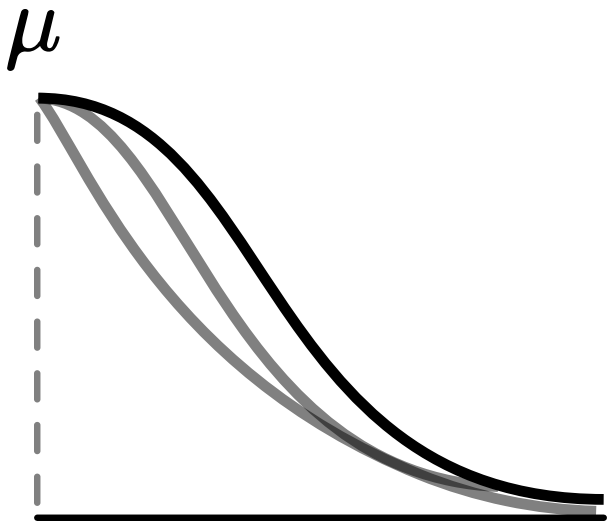


A Normal $\mathcal{N}(\mu, \sigma)$ distribution centered on μ (the red instance), and the relative density of another instance x_j (in pink).

Neighboring phase: locality

Università di Pisa

σ_i^2 defines the bandwidth of the Gaussian, and as such, the density of the cluster. We choose σ_i^2 so that the resulting local distribution, and thus clustering, has a controlled intra-cluster similarity.

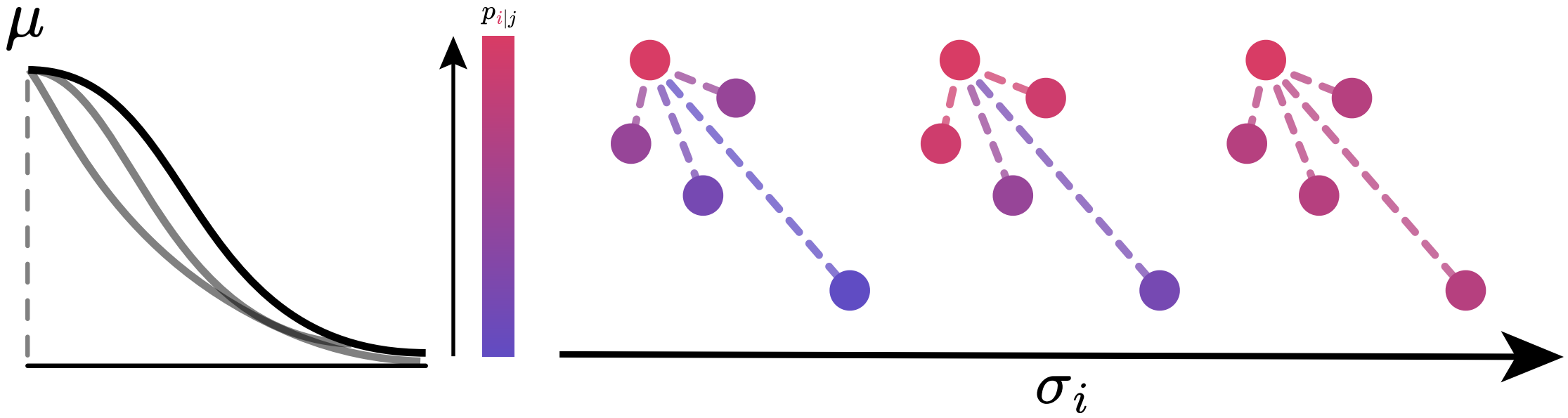


A Normal $\mathcal{N}(\mu, \cdot)$ distribution at different bandwidths $\sigma_1, \ldots, \sigma_k$.

Neighboring phase: defining locality

UNIVERSITÀ DI PISA

The *perplexity* hyperparameter h quantifies the heterogeneity of the distribution: the larger the perplexity, the more heterogeneous the cluster, and the farther the points included in the cluster: $h = Perp(P_i) = 2^{H(P_i)}$.

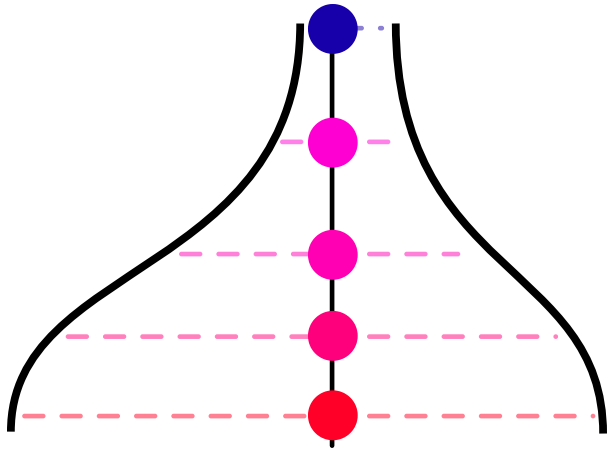


$H(P_i)$ measures the entropy $H(P_i) = \frac{1}{2}\log(2\pi\sigma_i^2) + \frac{1}{w}$ of the distribution.

Neighbors as a starting representation

UNIVERSITÀ DI PISA

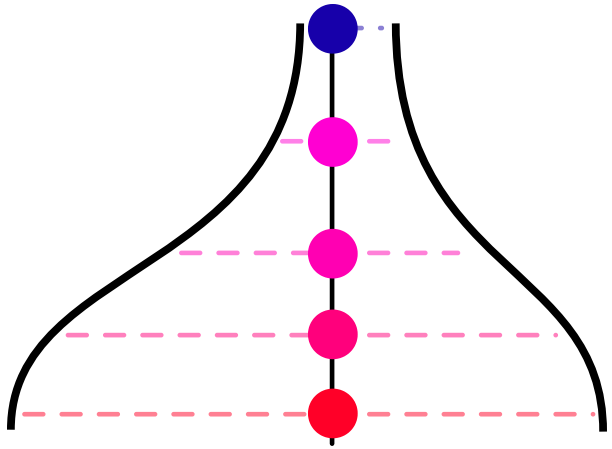
The neighboring step generates a (soft) neighboring matrix S akin to the one we use in clustering. The subsequent goal of t -SNE: to learn a representation $\hat{X}$ with as close a neighboring matrix $\hat{S}$ as possible.



Instances lied on a Normal (left) and t-student (right) distribution, induced by the original representation X, and by the t-SNE representation $\hat{X}$, respectively. t-SNE aims to make the densities on the two as similar as possible.

Searching for neighbors in $\hat{\mathcal{X}}$

$\hat{\mathcal{X}}$ is typically a lower dimensional space than $\mathcal{X}$, in which the tails Gaussian distributions decrease rapidly, and far-away instances end up crowding them. Rather, t-SNE employs a t-student distribution with 1 degree of freedom, which has much slacker tails.



Instances lied on a Normal (left) and t-student (right) distribution: t-SNE aims to make the densities on the two as similar as possible.

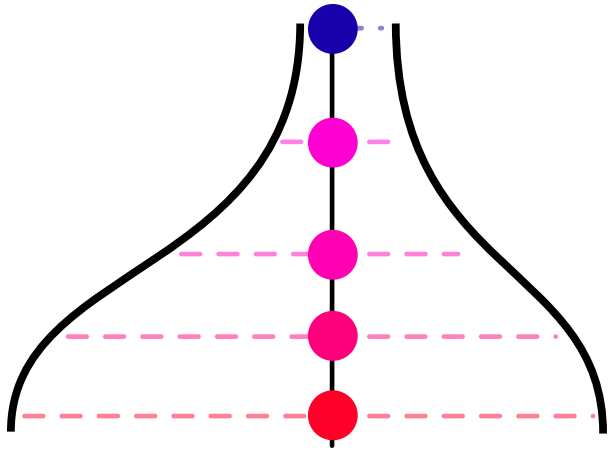
The low tail and consequent concentration of instances on the tails is called 'overcrowding'.

Searching for neighbors in $\hat{\mathcal{X}}$

UNIVERSITÀ DI PISA

Representation optimization.
t-SNE minimizes the distance between S and $\hat{S}$ through Kullback-Leiber divergence. Each P_i induces a minimization

$$KL(P_i \mid\mid \hat{P}_i) = \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{\hat{p}_{j|i}}$$



Densities of various points on the original normal distribution (left), and on the t-student distribution (right).

The low tail and consequent concentration of instances on the tails is called 'overcrowding'.

Using t-SNE

- t-SNE computes a full similarity matrix, thus it can be computationally expensive on extremely large datasets
- Unlike PCA, the transformation is not linear, and thus expressively more powerful (at the cost of interpretability)
- Perplexity is an hyperparameter that should be tuned
- The optimization algorithm hides another set of parameters, which can result in nondeterministic results
- Perplexity is, in some hard-to-define way, related to dataset size. The more points in the dataset, the higher the inherent heterogeneity of the cluster, the higher the required perplexity. How much higher?

Ridiculously good visualization at https://distill.pub/2016/misread-tsne/

PCA VS t-SNE

UNIVERSITÀ DI PISA

	PCA	**t-SNE**
Transformation	Linear	Nonlinear
Hyperparameters	None	Perplexity
Determinism	Deterministic	Nondeterministic
Interpretability	Interpretable	Noninterpretable
Locality	Global	Local
Computational cost	Low, $\mathcal{O}(n^3)$	High

A third way: Uniform Manifold Approximation and Projection

UNIVERSITÀ DI PISA

PCA and t-SNE tackle locality as a dichotomy: either global, or local, and are thus not *locally-adaptive*:

- PCA: we study spectral decomposition of the whole dataset
- t-SNE: we define perplexity over the whole dataset

What if we adapt the neighborhoods?

Leland McInnes, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. 2018

UMAP and the adaptive manifold

Università di Pisa

Both t-SNE and UMAP approximate the data manifold, only the former can only approximate it accurately for uniform manifolds! *Unlike* t-SNE, UMAP locally adapts the manifold to each instance, thus defining *adaptive* neighborhoods, **each instance** defining its neighborhood with **its own parameterization**.

A visualization of a possible adaptive neighborhood definition: distance to the k-th neighbor adaptively and locally determines the manifold. Each instance stretches the space so that its own neighborhood has a given volume.

UMAP and the connectivity graph

The computed distances induce a connectivity graph, and thus an adjacency matrix A, its edges measuring distances among instances. After turning distances into probabilities, UMAP optimizes a distance on A, to make it so that all and only the edges on the original manifold also appear in the transformed manifold with the same magnitude.

UMAP and the connectivity graph

For the set of edges E, UMAP minimizes

$$-\sum_{e \in E} (\underbrace{\Pr(e; X) \log(\Pr(e; Z))}_{all\ edges} + \underbrace{(1 - \Pr(e; Z)) \log(1 - \Pr(e; X))}_{only\ edges}),$$

where $\Pr(e; X), \Pr(e; Z)$ indicate the probability of edge e in the original and transformed representation, respectively.

PCA VS t-SNE VS UMAP

Università di Pisa

	PCA	**t-SNE**	**UMAP**
Transformation	Linear	Nonlinear	Nonlinear
Hyperparameters	None	Perplexity	Neighborhood size
Determinism	Deterministic	Nondeterministic	Nondeterministic
Interpretability	Interpretable	Noninterpretable	Noninterpretable
Locality	Global	Local	Adaptive
Computational cost	Low, $\mathcal{O}(n^3)$	High	Low, but higher than PCA

Data representation: which to choose?

Università di Pisa

PCA

- Strong mathematical foundation
- Interpretable results
- Extremely fast

- Global

t-SNE

- Powerful

- Slow
- Sensitive to initialization and parameters
- Parameters a bit obscure

UMAP

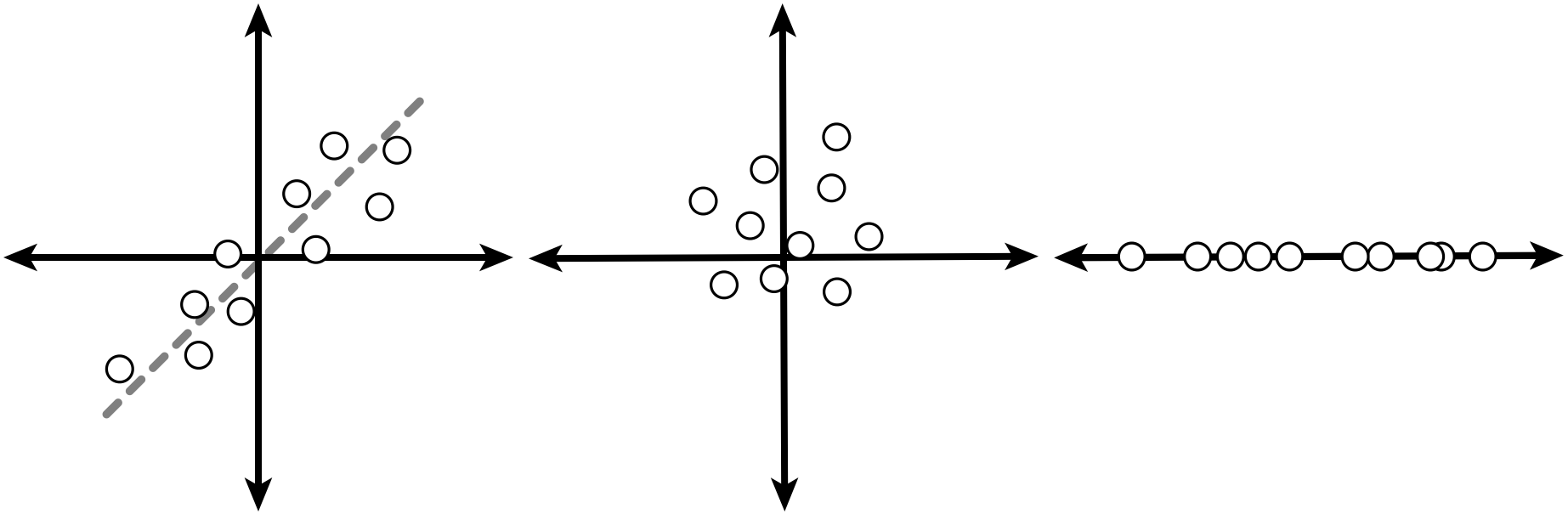
- Adaptive
- Interpretable parameters
- Strong mathematical foundation
- Fast

- Sensitive to initialization

Space representations

Università di Pisa

Representations allows us to map data into another space. What if we pick a space of smaller dimensionality?

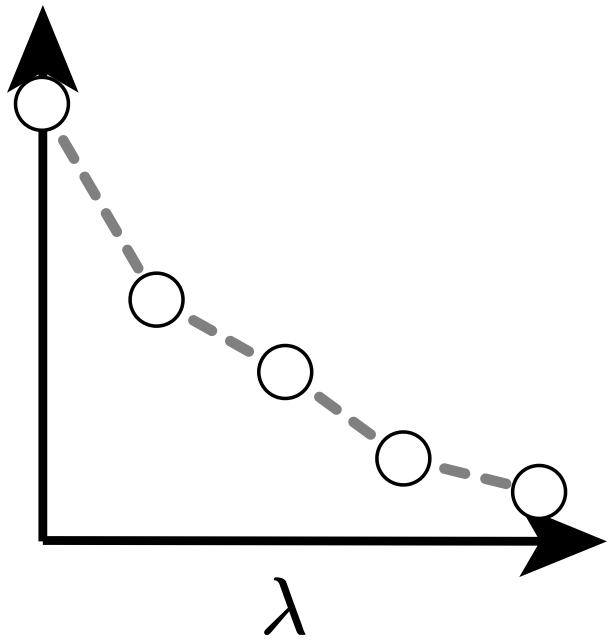


From the original dataset, to an alternative representation, to a smaller space representation.

Space representations in PCA

Università di Pisa

PCA is a linear transformation $A\bar{X} = \hat{X}$, and as such it follows standard dimension rules on matrix-* multiplication:
$(m \times n)(n \times k)$ yields a matrix $(m \times k)$.
We can trim columns off of $\hat{X}$... but at what cost?

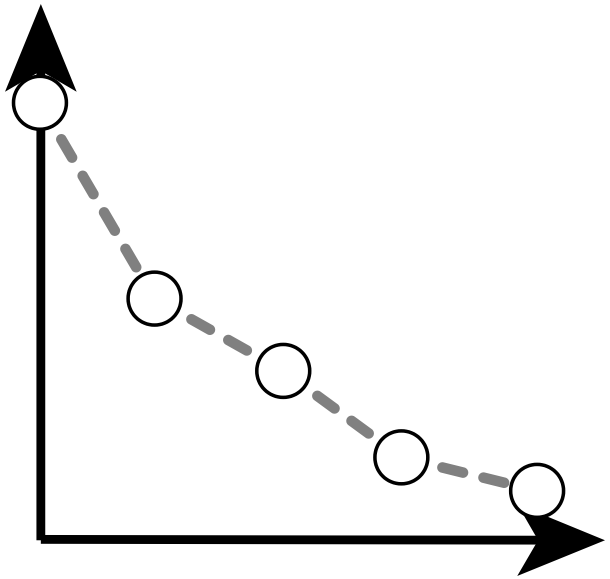


Eigenvalues in decreasing order of magnitude: larger eigenvalues increase the magnitude of instances, lower instead lower it. We can trim this with the elbow method: look for a value of maximum change in eigenvalues.

Space representations in PCA

Università di Pisa

PCA is a linear transformation $A\bar{X} = \hat{X}$, and as such it follows standard dimension rules on matrix-* multiplication:
$(m \times n)(n \times k)$ yields a matrix $(m \times k)$.
We can trim columns off of $\hat{X}$... but at what cost?



$|| \hat{X} ||$

PCA norm on an increasing number of trimmed dimensions. We can trim this with the elbow method: look for a value of maximum change in eigenvalues.