Principal Component Analysis

Idea

High dimensional data can often be represented using a much lower dimensional space. This happens when the data lives near a linear manifold in the high dimensional space.

If we find the manifold we can project the data in the manifold and using to represent the data without losing much information.

Manifold

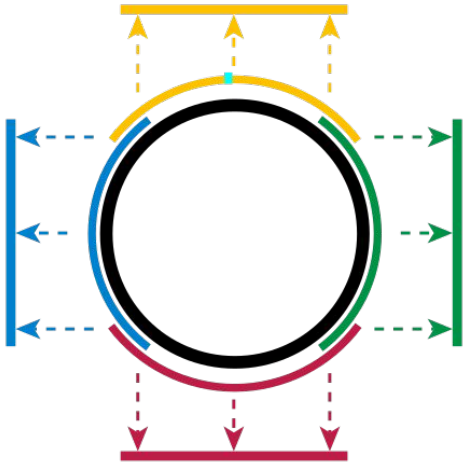
A Manifold is a topological space that **locally (near each point) resembles the Euclidean space.**

Consider the upper half of the unit circle, $x^2 + y^2 = 1$, where $y \geq 0$ (yellow arc).

Every point on this arc can be uniquely identified by its x-coordinate $\chi_{top}(x,y) = x$

The projection onto the first coordinate defines a **smooth** and **invertible mapping** from the upper arc to the open interval $(-1,1)$

Functions which provide a one-to-one correspondence between open regions of a surface and subsets of Euclidean space, are called *charts*.



Charts

Each chart can be seen as a mapping $\phi : R^1 \rightarrow S \subset R^2$. ϕ must be smooth and invertible (diffeomorphism). The key property is that both the function and its inverse are continuously differentiable.

The domain of ϕ is the parametric space and is Euclidean.

The image of ϕ is the embedding and is a surface.

Manifolds in the end are unions of charts.

Unsupervised Learning

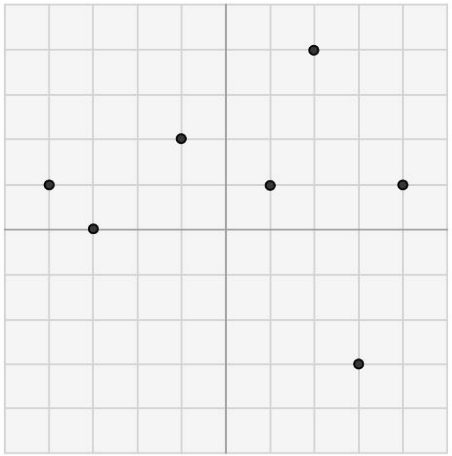
Unsupervised learning focuses on effectively representing datasets that lack labeled outputs, meaning we work solely with input data.

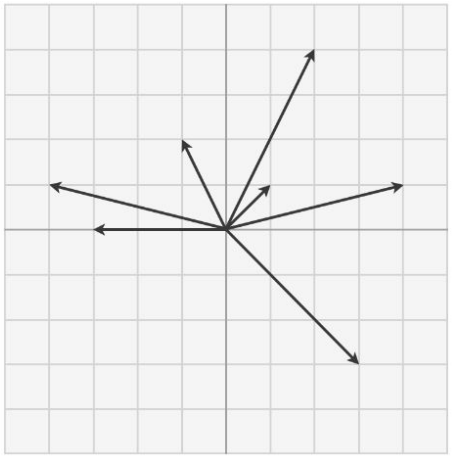
The primary goal is to uncover meaningful structures or representations within the data.

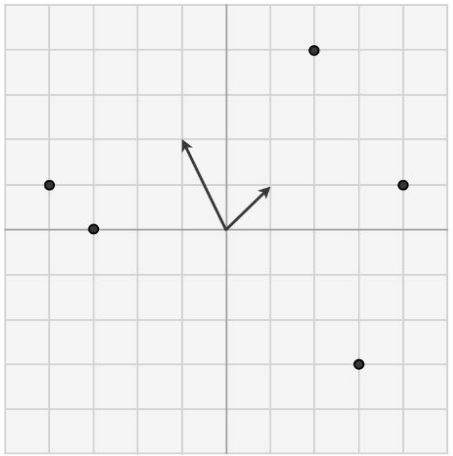
This concept is closely related to the idea of a spanning set of vectors in basic linear algebra, where the aim is to represent a space efficiently using a minimal and meaningful set of components.

Vector Space

When visualizing data points in a multi-dimensional vector space, we can represent them as either *dots* (left panel) or *arrows* (middle panel). To understand a basis, it's helpful to use both conventions simultaneously (right panel): some points as *arrows* (a basis or spanning set) and others as *dots* (the points to be represented). The basis vectors are used to efficiently reconstruct all other points in the space.







Basis representation

Suppose our dataset consists of N input points, $\{x_1, x_2, \ldots, x_N\}$, each living in D-dimensional space. For simplicity, we assume the dataset has been mean-centered (mean is subtracted along each dimension) ensuring the data is centered at the origin.

To perfectly represent all N points, the basis $\{c_1, c_2, \ldots, c_D\}$ must also reside in the same D-dimensional space. For any D-dimensional data point, there must exist a set of weights such that the basis, in a specific linear combination, reconstructs the data point:

$$\sum_{d=1}^{D} c_d w_{d,n} = x_n \quad \text{for } n = 1, \ldots, N.$$

This requires the basis vectors to be **linearly independent**, meaning they do not overlap and point in distinct directions, ensuring they span the entire D-dimensional space.

Standard basis

As a simple example, consider the spanning set as the D standard basis vectors. Each standard basis vector consists of zeros everywhere except for a 1 in the k-th position: c_k=[0,0,…,1,…,0] where the 1 is in the k-th slot.

Key properties $c_n^T c_m = 0 \quad \text{and} \quad \|c_n\|^2 = 1 \quad \text{for} \quad n \neq m.$

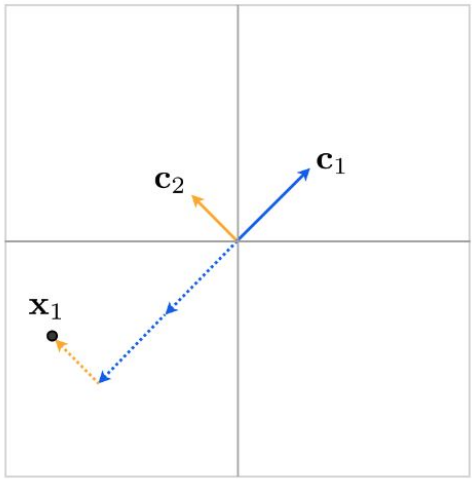
Or $C^T C = I,$

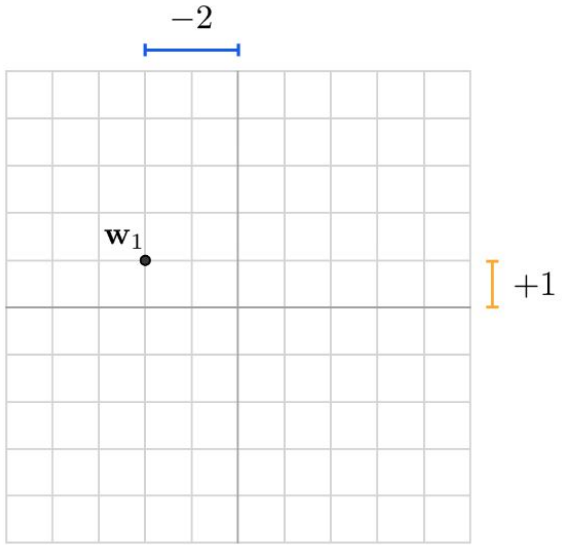
Representing a data point using the standard basis is straightforward. The weights are simply the values of the data point itself: $w_{d,n}=x_{d,n}$. Therefore the new representation for the point x_n is exactly w_n.

For most any other spanning set however these weights must be solved for numerically.

Example

Once tuned, the weight vector w_n provides the representation (or encoding or embedding) of x_n in terms of the spanning set $c_1,\ldots,c_D$.





How do we find the proper weights?

$$\min_{w_1, w_2, \ldots, w_N} \frac{1}{N} \sum_{n=1}^{N} \|Cw_n - x_n\|_2^2.$$

C is DxD matrix where the columns are the basis vectors, w_n is the learned weight and x_n the original input.

By setting the gradient of the cost function to zero and solving for w_n, we obtain a linear symmetric system of equations:

$$C^T C w_n = C^T x_n$$

Orthonormal basis?

When the spanning set is orthonormal, the algebraic formula for the weight vector or encoding w_p of a point x_p becomes straightforward:

$$w_p = C^T x_p$$

This equation demonstrates that, when the spanning set is orthonormal, the entire set of encodings for a dataset can be expressed directly in terms of the spanning set and the data itself.

$$CC^T x_p = x_p$$

The operation CC^T acts as a projection matrix, ensuring that each data point x_p is perfectly represented by the orthonormal basis, with no need for further adjustment or solving systems of equations.

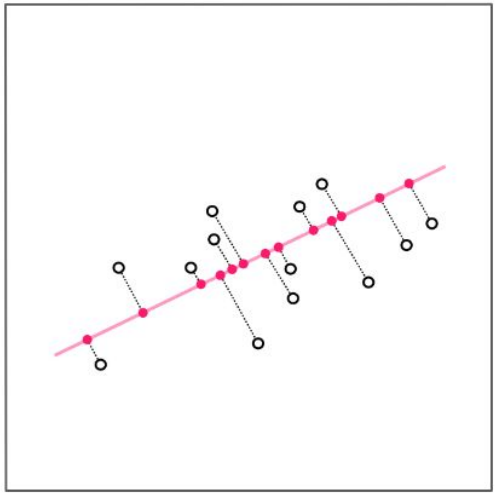
Into a Smaller Dimension

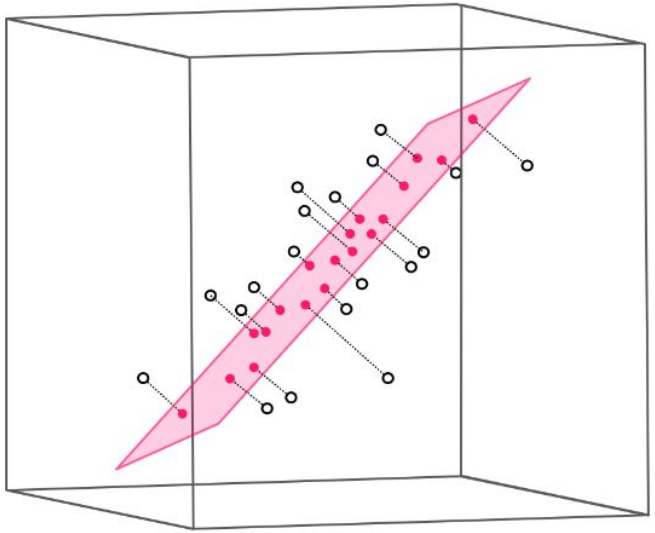
We previously discussed two key requirements for a spanning set or basis to perfectly represent points in a generic D-dimensional space:

1. The vectors must be linearly independent, meaning they point in different directions within the space.
2. The set must contain at least D-vectors to span the entire space.

But what happens if we relax the second condition and consider a case where we have fewer than D spanning vectors, specifically $K \leq D$?

Lower dimension

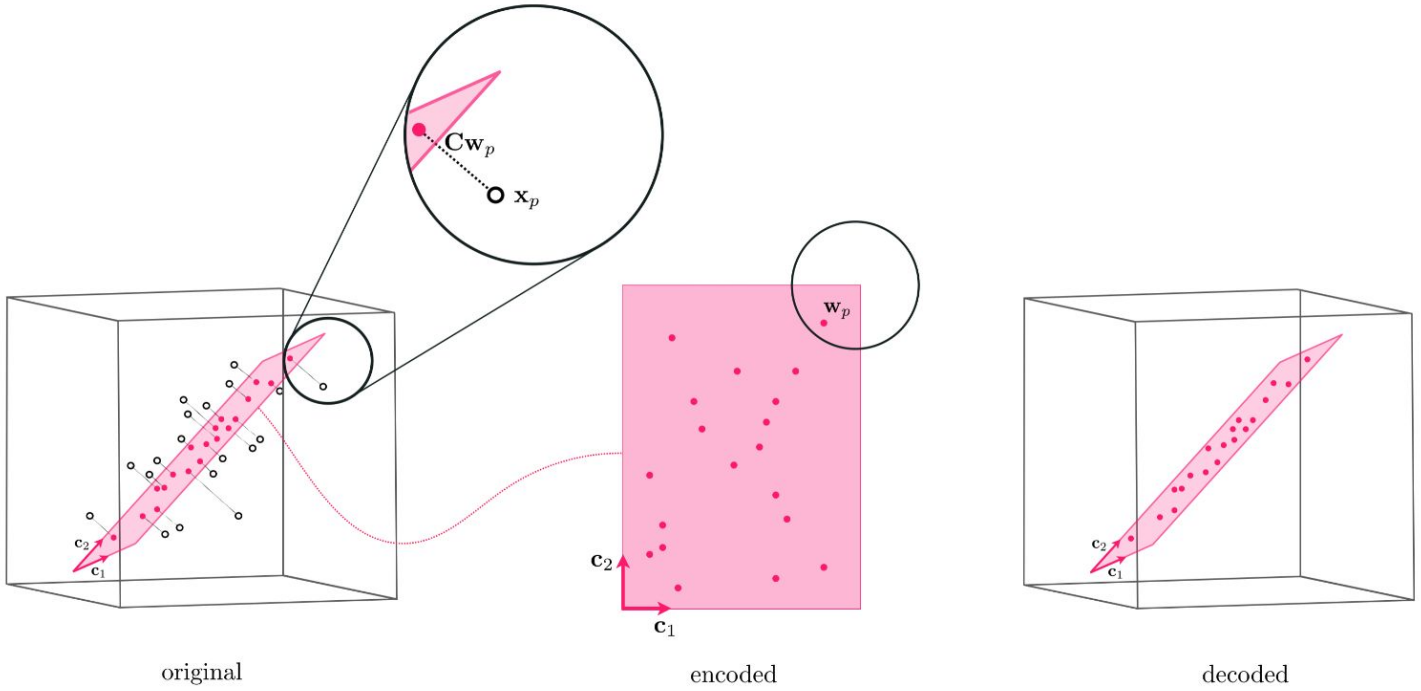




Doesn’t change much

While we may not be able to perfectly represent a given point or set of points in the space we can still approximate it very well using k spanning vectors C now is (DxK). Once the weight vectors w_p are computed, the projection of x_p (its representation in the subspace spanned by C) is given by Cw_p. This projection represents the 'dropping' of x_p perpendicularly onto the subspace formed by the K basis vectors. The weight vector w_p gives the encoded representation of x_p over the spanning set C. The decoded version of x_p is simply the projection of the original data point onto the subspace defined by the spanning vectors, which is given by Cw_p.

See?


$\mathbf{C}\mathbf{w}_p$
$\mathbf{x}_p$
$\mathbf{w}_p$
$\mathbf{c}_2$
$\mathbf{c}_1$
original
encoded
decoded

PCA

We will learn both an appropriate basis and the corresponding weights. This approach, where the basis is learned alongside the weights, is known as Principal Component Analysis (PCA).

The only change here is that, since we aim to learn the basis C as well, it has been added to the list of variables we wish to minimize in the original Least Squares cost function.

$$\min_{w_1, w_2, \ldots, w_N, C} \frac{1}{N} \sum_{n=1}^{N} \|Cw_n - x_n\|_2^2.$$

Learn orthonormal Basis

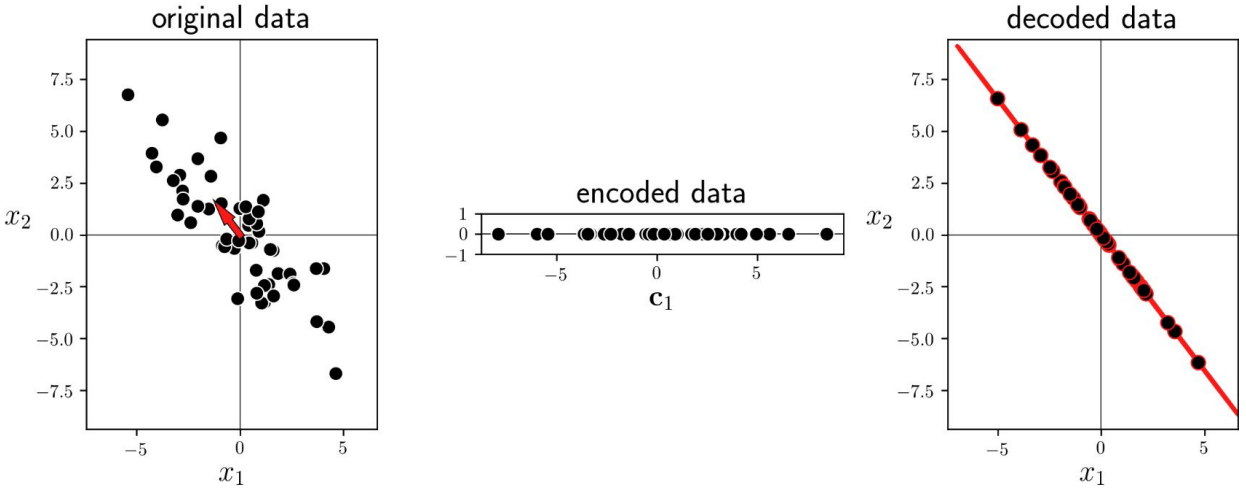
If we constrain our search to orthogonal matrices C such that $C^TC=I$ (K×K) , the PCA Least Squares cost function simplifies as follows:

$$g(C) = \frac{1}{N} \sum_{n=1}^{N} \|CC^T x_n - x_n\|_2^2.$$

This is significant because, under the orthogonality constraint, the cost function no longer depends on the weight vectors w_n, and is only a function of C.

Autoencoder

This simplified PCA Least Squares cost function is known as the autoencoder. The reason for this name is that, by minimizing the cost, we learn both the encoding (via the learned weights w_p) and the decoding (via the projection Cw_p) for each data point. In this form, we aim to encode and decode each point in terms of itself.

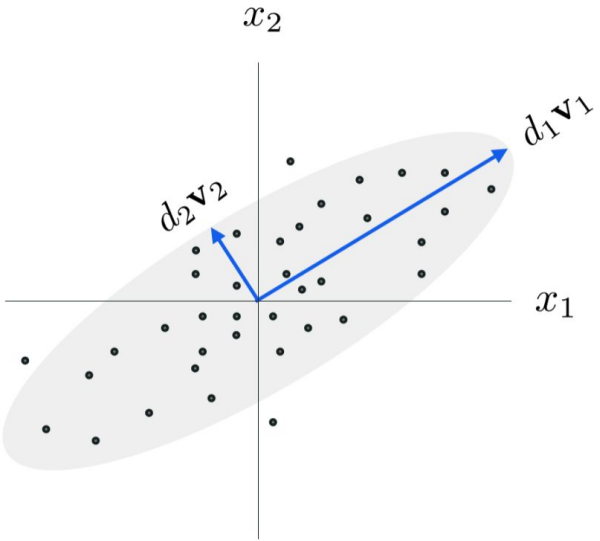


The solution?

The classic orthogonal PCA minimizer of the autoencoder cost function. The elements of this basis point in the orthogonal directions of variance of the dataset, that is the orthogonal directions in which the dataset is most spread out.

It is a closed form solution!

> The elements of this basis are so special they are given the formal name *principal components*



Analytical Solution

Given the data matrix X(NxD), the principal component basis can be computed (as a minimum of the autoencoder cost function) as the eigenvectors of the corresponding correlation matrix of this data

$$\mathrm{Cov}(X) := \frac{1}{N} X X^T$$

The eigenvector/eigenvalue decomposition of the covariance matrix is:

$$\frac{1}{N} X X^T = V D V^T,$$

where V contains the eigenvectors and D is the diagonal matrix of eigenvalues. The orthonormal basis we recover is precisely given by the eigenvectors, C=V (the principal components of the data). Additionally, the variance along each principal component direction is exactly the corresponding eigenvalue in D.

References

All the images were taken from:

- https://kenndanielso.github.io/mlrefined/blog_posts/11_Linear_unsupervised_learning/11_1_Spanning_sets_orthonormality_projections.html
- https://kenndanielso.github.io/mlrefined/blog_posts/11_Linear_unsupervised_learning/11_2_Principal_Component_Analysis.html