Data mining and Machine learning

Part 5. Kernel Methods

Input and Feature Space

For mining and analysis, it is important to find a suitable data representation. For example, for complex data such as text, sequences, images, and so on, we must typically extract or construct a set of attributes or features, so that we can represent the data instances as multivariate vectors.

Given a data instance $\boldsymbol{x}$ (e.g., a sequence), we need to find a mapping ϕ, so that $\phi(\boldsymbol{x})$ is the vector representation of $\boldsymbol{x}$.

Even when the input data is a numeric data matrix a nonlinear mapping ϕ may be used to discover nonlinear relationships.

The term *input space* refers to the data space for the input data $\boldsymbol{x}$ and *feature space* refers to the space of mapped vectors $\phi(\boldsymbol{x})$.

Sequence-based Features

Consider a dataset of DNA sequences over the alphabet $\Sigma = \{A, C, G, T\}$.

One simple feature space is to represent each sequence in terms of the probability distribution over symbols in Σ. That is, given a sequence $\boldsymbol{x}$ with length $|\boldsymbol{x}| = m$, the mapping into feature space is given as

$$\phi(\boldsymbol{x}) = \{P(A), P(C), P(G), P(T)\}$$

where $P(s) = \frac{n_s}{m}$ is the probability of observing symbol $s \in \Sigma$, and n_s is the number of times s appears in sequence $\boldsymbol{x}$.

For example, if $\boldsymbol{x} = ACAGCAGTA$, with $m = |\boldsymbol{x}| = 9$, since A occurs four times, C and G occur twice, and T occurs once, we have

$$\phi(\boldsymbol{x}) = (4/9, 2/9, 2/9, 1/9) = (0.44, 0.22, 0.22, 0.11)$$

We can compute larger feature spaces by considering, for example, the probability distribution over all substrings or words of size up to k over the alphabet Σ.

Nonlinear Features

Consider the mapping ϕ that takes as input a vector $\boldsymbol{x} = (x_1, x_2)^T \in \mathbb{R}^2$ and maps it to a "quadratic" feature space via the nonlinear mapping

$$\phi(\boldsymbol{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T \in \mathbb{R}^3$$

For example, the point $\boldsymbol{x} = (5.9, 3)^T$ is mapped to the vector

$$\phi(\boldsymbol{x}) = (5.9^2, 3^2, \sqrt{2} \cdot 5.9 \cdot 3)^T = (34.81, 9, 25.03)^T$$

We can then apply well-known linear analysis methods in the feature space.

Kernel Method

Let $\mathcal{I}$ denote the input space, which can comprise any arbitrary set of objects, and let $\boldsymbol{D} = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathcal{I}$ be a dataset comprising n objects in the input space. Let $\phi: \mathcal{I} \to \mathcal{F}$ be a mapping from the input space $\mathcal{I}$ to the feature space $\mathcal{F}$.

Kernel methods avoid explicitly transforming each point $\boldsymbol{x}$ in the input space into the mapped point $\phi(\boldsymbol{x})$ in the feature space. Instead, the input objects are represented via their pairwise similarity values comprising the $n \times n$ *kernel matrix*, defined as

$$\boldsymbol{K} = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & K(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_2, \boldsymbol{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{x}_1) & K(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix}$$

$K: \mathcal{I} \times \mathcal{I} \to \mathbb{R}$ is a *kernel function* on any two points in input space, which should satisfy the condition

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

Intuitively, we need to be able to compute the value of the dot product using the original input representation $\boldsymbol{x}$, without having recourse to the mapping $\phi(\boldsymbol{x})$.

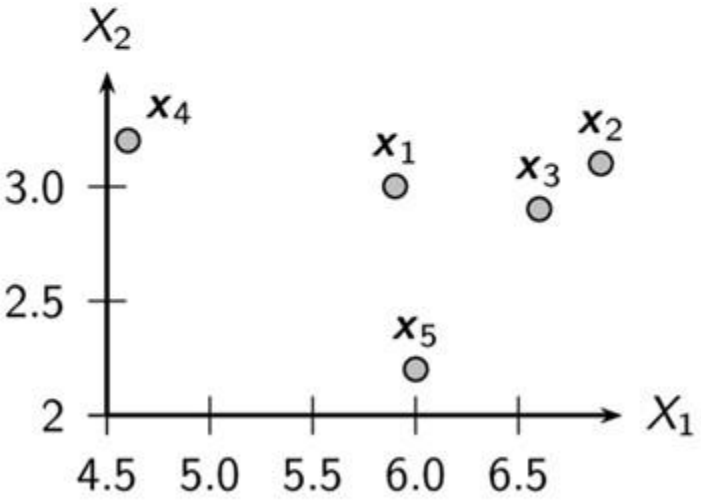
Linear Kernel

Let $\phi(\boldsymbol{x}) \rightarrow \boldsymbol{x}$ be the *identity kernel*. This leads to the *linear kernel*, which is simply the dot product between two input vectors:

$$\phi(\boldsymbol{x})^T \phi(\boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{y} = K(\boldsymbol{x}, \boldsymbol{y})$$

For example, if $\boldsymbol{x}_1 = \begin{pmatrix} 5.9 & 3 \end{pmatrix}^T$ and $\boldsymbol{x}_2 = \begin{pmatrix} 6.9 & 3.1 \end{pmatrix}^T$, then we have

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1^T \boldsymbol{x}_2 = 5.9 \times 6.9 + 3 \times 3.1 = 40.71 + 9.3 = 50.01$$



K	x_1	x_2	x_3	x_4	x_5
x_1	43.81	50.01	47.64	36.74	42.00
x_2	50.01	57.22	54.53	41.66	48.22
x_3	47.64	54.53	51.97	39.64	45.98
x_4	36.74	41.66	39.64	31.40	34.64
x_5	42.00	48.22	45.98	34.64	40.84

Kernel Trick

Many data mining methods can be *kernelized* that is, instead of mapping the input points into feature space, the data can be represented via the $n \times n$ kernel matrix $\boldsymbol{K}$, and all relevant analysis can be performed over $\boldsymbol{K}$.

This is done via the *kernel trick*, that is, show that the analysis task requires only dot products $\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$ in feature space, which can be replaced by the corresponding kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$ that can be computed efficiently in input space.

Once the kernel matrix has been computed, we no longer even need the input points $\boldsymbol{x}_i$, as all operations involving only dot products in the feature space can be performed over the $n \times n$ kernel matrix $\boldsymbol{K}$.

Kernel Matrix

A function K is called a **positive semidefinite kernel** if and only if it is symmetric:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = K(\boldsymbol{x}_j, \boldsymbol{x}_i)$$

and the corresponding kernel matrix $\boldsymbol{K}$ for any subset $\boldsymbol{D} \subset \mathcal{I}$ is positive semidefinite, that is,

$$\boldsymbol{a}^T \boldsymbol{K} \boldsymbol{a} \geq 0, \text{ for all vectors } \boldsymbol{a} \in \mathbb{R}^n$$

which implies that

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0, \text{ for all } a_i \in \mathbb{R}, i \in [1, n]$$

Dot Products and Positive Semi-definite Kernels

Positive Semidefinite Kernel

If $K(x_i, x_j)$ represents the dot product $\phi(x_i)^T \phi(x_j)$ in some feature space, then K is a positive semidefinite kernel.

First, K is symmetric since the dot product is symmetric, which also implies that $\boldsymbol{K}$ is symmetric.

Second, $\boldsymbol{K}$ is positive semidefinite because

$$\begin{aligned}
\boldsymbol{a}^T \boldsymbol{K} \boldsymbol{a} &= \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j K(x_i, x_j) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \phi(x_i)^T \phi(x_j) \\
&= \left(\sum_{i=1}^{n} a_i \phi(x_i) \right)^T \left(\sum_{j=1}^{n} a_j \phi(x_j) \right) \\
&= \left\| \sum_{i=1}^{n} a_i \phi(x_i) \right\|^2 \geq 0
\end{aligned}$$

Empirical Kernel Map

We now show that if we are given a positive semidefinite kernel $K: \mathcal{I} \times \mathcal{I} \to \mathbb{R}$, then it corresponds to a dot product in some feature space $\mathcal{F}$.

Define the map ϕ as follows:

$$\phi(\boldsymbol{x}) = \Big((K(\boldsymbol{x}_1, \boldsymbol{x}), K(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, K(\boldsymbol{x}_n, \boldsymbol{x}) \Big)^T \in \mathbb{R}^n$$

The *empirical kernel map* is defined as

$$\phi(\boldsymbol{x}) = \boldsymbol{K}^{-1/2} \cdot \Big((K(\boldsymbol{x}_1, \boldsymbol{x}), K(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, K(\boldsymbol{x}_n, \boldsymbol{x}) \Big)^T \in \mathbb{R}^n$$

so that the dot product yields

$$\begin{aligned} \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) &= \Big(\boldsymbol{K}^{-1/2} \, \boldsymbol{K}_i \Big)^T \Big(\boldsymbol{K}^{-1/2} \, \boldsymbol{K}_j \Big) \\ &= \boldsymbol{K}_i^T \left(\boldsymbol{K}^{-1/2} \boldsymbol{K}^{-1/2} \right) \boldsymbol{K}_j \\ &= \boldsymbol{K}_i^T \, \boldsymbol{K}^{-1} \, \boldsymbol{K}_j \end{aligned}$$

where $\boldsymbol{K}_i$ is the ith column of $\boldsymbol{K}$. Over all pairs of mapped points, we have

$$\left\{ \boldsymbol{K}_i^T \boldsymbol{K}^{-1} \, \boldsymbol{K}_j \right\}_{i,j=1}^n = \boldsymbol{K} \, \boldsymbol{K}^{-1} \, \boldsymbol{K} = \boldsymbol{K}$$

Data-specific Mercer Kernel Map

The Mercer kernel map also corresponds to a dot product in feature space.

Since $\boldsymbol{K}$ is a symmetric positive semidefinite matrix, it has real and non-negative eigenvalues. It can be decomposed as follows:

$$\boldsymbol{K} = \boldsymbol{U}\Lambda\boldsymbol{U}^T$$

where $\boldsymbol{U}$ is the orthonormal matrix of eigenvectors $\boldsymbol{u}_i = (u_{i1}, u_{i2}, \ldots, u_{in})^T \in \mathbb{R}^n$ (for $i = 1, \ldots, n$), and Λ is the diagonal matrix of eigenvalues, with both arranged in non-increasing order of the eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$:
The Mercer map ϕ is given as

$$\phi(\boldsymbol{x}_i) = \sqrt{\Lambda}\boldsymbol{U}_i$$

where $\boldsymbol{U}_i$ is the ith row of $\boldsymbol{U}$.
The kernel value is simply the dot product between scaled rows of $\boldsymbol{U}$:

$$\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j) = \left(\sqrt{\Lambda}\boldsymbol{U}_i\right)^T\left(\sqrt{\Lambda}\boldsymbol{U}_j\right) = \boldsymbol{U}_i^T\Lambda\boldsymbol{U}_j$$

Polynomial Kernel

Polynomial kernels are of two types: homogeneous or inhomogeneous.

Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. The (inhomogeneous) *polynomial kernel* is defined as

$$K_q(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y}) = (c + \boldsymbol{x}^T \boldsymbol{y})^q$$

where q is the degree of the polynomial, and $c \geq 0$ is some constant. When $c = 0$ we obtain the homogeneous kernel, comprising only degree q terms. When $c > 0$, the feature space is spanned by all products of at most q attributes.
This can be seen from the binomial expansion

$$K_q(\boldsymbol{x}, \boldsymbol{y}) = (c + \boldsymbol{x}^T \boldsymbol{y})^q = \sum_{k=1}^{q} \binom{q}{k} c^{q-k} \left(\boldsymbol{x}^T \boldsymbol{y}\right)^k$$

The most typical cases are the *linear* (with $q = 1$) and *quadratic* (with $q = 2$) kernels, given as

$$K_1(\boldsymbol{x}, \boldsymbol{y}) = c + \boldsymbol{x}^T \boldsymbol{y}$$
$$K_2(\boldsymbol{x}, \boldsymbol{y}) = (c + \boldsymbol{x}^T \boldsymbol{y})^2$$

Gaussian Kernel

The Gaussian kernel, also called the Gaussian radial basis function (RBF) kernel, is defined as

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left\{-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right\}$$

where $\sigma > 0$ is the spread parameter that plays the same role as the standard deviation in a normal density function.

Note that $K(\boldsymbol{x}, \boldsymbol{x}) = 1$, and further that the kernel value is inversely related to the distance between the two points $\boldsymbol{x}$ and $\boldsymbol{y}$.

A feature space for the Gaussian kernel has infinite dimensionality.

Basic Kernel Operations in Feature Space

Basic data analysis tasks that can be performed solely via kernels, without instantiating $\phi(\boldsymbol{x})$.

Norm of a Point: We can compute the norm of a point $\phi(\boldsymbol{x})$ in feature space as follows:

$$\|\phi(\boldsymbol{x})\|^2 = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x})$$

which implies that $\|\phi(\boldsymbol{x})\| = \sqrt{K(\boldsymbol{x}, \boldsymbol{x})}$.

Distance between Points: The distance between $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$ is

$$\begin{aligned} \|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|^2 &= \|\phi(\boldsymbol{x}_i)\|^2 + \|\phi(\boldsymbol{x}_j)\|^2 - 2\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) \\ &= K(\boldsymbol{x}_i, \boldsymbol{x}_i) + K(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned}$$

which implies that

$$\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\| = \sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i) + K(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j)}$$

Basic Kernel Operations in Feature Space

Kernel Value as Similarity: We can rearrange the terms in

$$\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|^2 = K(\boldsymbol{x}_i, \boldsymbol{x}_i) + K(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

to obtain

$$\frac{1}{2}\left(\|\phi(\boldsymbol{x}_i)\|^2 + \|\phi(\boldsymbol{x}_j)\|^2 - \|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|^2\right) = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

The more the distance $\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|$ between the two points in feature space, the less the kernel value, that is, the less the similarity.

Mean in Feature Space: The mean of the points in feature space is given as $\boldsymbol{\mu}_\phi = 1/n \sum_{i=1}^n \phi(\boldsymbol{x}_i)$. Thus, we cannot compute it explicitly. However, the the squared norm of the mean is:

$$\|\boldsymbol{\mu}_\phi\|^2 = \boldsymbol{\mu}_\phi^T \boldsymbol{\mu}_\phi = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

The squared norm of the mean in feature space is simply the average of the values in the kernel matrix $\boldsymbol{K}$.

Basic Kernel Operations in Feature Space

Total Variance in Feature Space: The total variance in feature space is obtained by taking the average squared deviation of points from the mean in feature space:

$$\sigma_\phi^2 = \frac{1}{n}\sum_{i=1}^{n}\|\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_\phi\|^2 = \frac{1}{n}\sum_{i=1}^{n}K(\boldsymbol{x}_i,\boldsymbol{x}_i) - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}K(\boldsymbol{x}_i,\boldsymbol{x}_j)$$

Centering in Feature Space We can center each point in feature space by subtracting the mean from it, as follows:

$$\hat{\phi}(\boldsymbol{x}_i) = \phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_\phi$$

The kernel between centered points is given as

$$\begin{aligned}\hat{K}(\boldsymbol{x}_i,\boldsymbol{x}_j) &= \hat{\phi}(\boldsymbol{x}_i)^T\hat{\phi}(\boldsymbol{x}_j)\\ &= K(\boldsymbol{x}_i,\boldsymbol{x}_j) - \frac{1}{n}\sum_{k=1}^{n}K(\boldsymbol{x}_i,\boldsymbol{x}_k) - \frac{1}{n}\sum_{k=1}^{n}K(\boldsymbol{x}_j,\boldsymbol{x}_k) + \frac{1}{n^2}\sum_{a=1}^{n}\sum_{b=1}^{n}K(\boldsymbol{x}_a,\boldsymbol{x}_b)\end{aligned}$$

More compactly, we have:

$$\hat{\boldsymbol{K}} = \left(\boldsymbol{I} - \frac{1}{n}1_{n\times n}\right)\boldsymbol{K}\left(\boldsymbol{I} - \frac{1}{n}1_{n\times n}\right)$$

where $1_{n\times n}$ is the $n \times n$ matrix of ones.

Basic Kernel Operations in Feature Space

Normalizing in Feature Space: The dot product between normalized points in feature space corresponds to the cosine of the angle between them

$$\phi_n(\boldsymbol{x}_i)^T \phi_n(\boldsymbol{x}_j) = \frac{\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)}{\|\phi(\boldsymbol{x}_i)\| \cdot \|\phi(\boldsymbol{x}_j)\|} = \cos\theta$$

If the mapped points are both centered and normalized, then a dot product corresponds to the correlation between the two points in feature space.

The normalized kernel matrix, $\boldsymbol{K}_n$, can be computed using only the kernel function K, as

$$\boldsymbol{K}_n(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)}{\|\phi(\boldsymbol{x}_i)\| \cdot \|\phi(\boldsymbol{x}_j)\|} = \frac{K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i) \cdot K(\boldsymbol{x}_j, \boldsymbol{x}_j)}}$$

$\boldsymbol{K}_n$ has all diagonal elements as 1.

Spectrum Kernel for Strings

Given alphabet Σ, the l-spectrum feature map is the mapping $\phi: \Sigma^* \to \mathbb{R}^{|\Sigma|^l}$ from the set of substrings over Σ to the $|\Sigma|^l$-dimensional space representing the number of occurrences of all possible substrings of length l, defined as

$$\phi(\boldsymbol{x}) = \left(\cdots, \#(\alpha), \cdots\right)^T_{\alpha \in \Sigma^l}$$

where $\#(\alpha)$ is the number of occurrences of the l-length string α in $\boldsymbol{x}$.

The (full) spectrum map considers all lengths from $l = 0$ to $l = \infty$, leading to an infinite dimensional feature map $\phi: \Sigma^* \to \mathbb{R}^\infty$:

$$\phi(\boldsymbol{x}) = \left(\cdots, \#(\alpha), \cdots\right)^T_{\alpha \in \Sigma^*}$$

where $\#(\alpha)$ is the number of occurrences of the string α in $\boldsymbol{x}$.

The (l-)spectrum kernel between two strings $\boldsymbol{x}_i, \boldsymbol{x}_j$ is simply the dot product between their (l-)spectrum maps:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

The (full) spectrum kernel can be computed efficiently via suffix trees in $O(n+m)$ time for two strings of length n and m.

Diffusion Kernels on Graph Nodes

Let $\boldsymbol{S}$ be some symmetric similarity matrix between nodes of a graph $G = (V, E)$. For instance, $\boldsymbol{S}$ can be the (weighted) adjacency matrix $\boldsymbol{A}$ or the Laplacian matrix $\boldsymbol{L} = \boldsymbol{A} - \Delta$ (or its negation), where Δ is the degree matrix for an undirected graph G, defined as $\Delta(i,i) = d_i$ and $\Delta(i,j) = 0$ for all $i \neq j$, and d_i is the degree of node i.

Power Kernels: Summing up the product of the base similarities over all l-length paths between two nodes, we obtain the l-length similarity matrix $\boldsymbol{S}^{(l)}$, which is simply the lth power of $\boldsymbol{S}$, that is,

$$\boldsymbol{S}^{(l)} = \boldsymbol{S}^l$$

Even path lengths lead to positive semidefinite kernels, but odd path lengths are not guaranteed to do so, unless the base matrix $\boldsymbol{S}$ is itself a positive semidefinite matrix.

Power kernel $\boldsymbol{K}$ can be obtained via the eigen-decomposition of $\boldsymbol{S}^l$:

$$\boldsymbol{K} = \boldsymbol{S}^l = \left(\boldsymbol{U}\Lambda\boldsymbol{U}^T\right)^l = \boldsymbol{U}\left(\Lambda^l\right)\boldsymbol{U}^T$$

Exponential Diffusion Kernel

The exponential diffusion kernel we can obtain a new kernel between nodes of a graph by paths of all possible lengths, but damps the contribution of longer paths

$$\begin{aligned}\boldsymbol{K} &= \sum_{l=0}^{\infty} \frac{1}{l!}\beta^l \boldsymbol{S}^l \\ &= \boldsymbol{I} + \beta\boldsymbol{S} + \frac{1}{2!}\beta^2\boldsymbol{S}^2 + \frac{1}{3!}\beta^3\boldsymbol{S}^3 + \cdots \\ &= \exp\{\beta\boldsymbol{S}\}\end{aligned}$$

where β is a damping factor, and $\exp\{\beta\boldsymbol{S}\}$ is the matrix exponential. The series on the right hand side above converges for all $\beta \geq 0$.

Substituting $\boldsymbol{S} = \boldsymbol{U}\Lambda\boldsymbol{U}^T$ the kernel can be computed as

$$\begin{aligned}\boldsymbol{K} &= \boldsymbol{I} + \beta\boldsymbol{S} + \frac{1}{2!}\beta^2\boldsymbol{S}^2 + \cdots \\ &= \boldsymbol{U}\begin{pmatrix} \exp\{\beta\lambda_1\} & 0 & \cdots & 0 \\ 0 & \exp\{\beta\lambda_2\} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \exp\{\beta\lambda_n\} \end{pmatrix}\boldsymbol{U}^T\end{aligned}$$

where λ_i is an eigenvalue of $\boldsymbol{S}$.

Von Neumann Diffusion Kernel

The *von Neumann diffusion kernel* is defined as

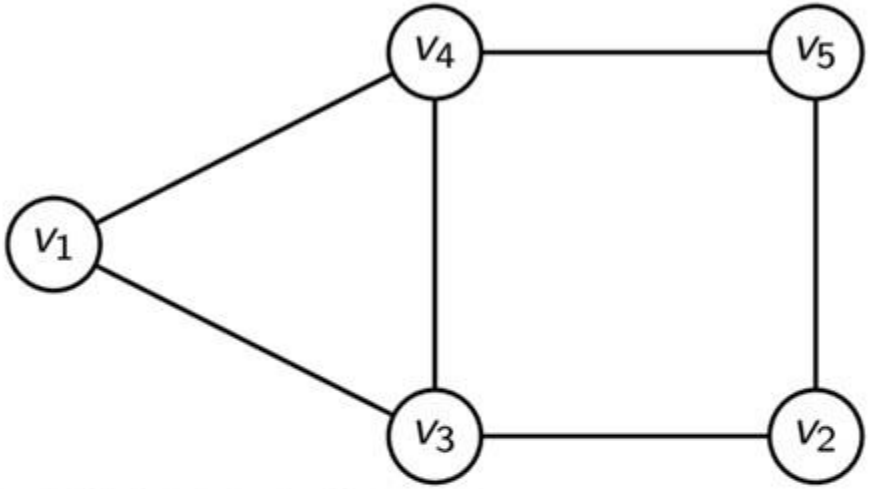
$$\boldsymbol{K} = \sum_{l=0}^{\infty} \beta^l \boldsymbol{S}^l$$

where $\beta \geq 0$. Expanding and rearranging the terms, we obtain

$$\boldsymbol{K} = (\boldsymbol{I} - \beta \boldsymbol{S})^{-1}$$

The kernel is guaranteed to be positive semidefinite if $|\beta| < 1/\rho(\boldsymbol{S})$, where $\rho(\boldsymbol{S}) = \max_i\{|\lambda_i|\}$ is called the *spectral radius* of $\boldsymbol{S}$, defined as the largest eigenvalue of $\boldsymbol{S}$ in absolute value.

Graph Diffusion Kernel: Example



Adjacency and degree matrices are given as

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad \Delta = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Graph Diffusion Kernel: Example

Let the base similarity matrix $\boldsymbol{S}$ be the negated Laplacian matrix

$$\boldsymbol{S} = -\boldsymbol{L} = \boldsymbol{A} - \boldsymbol{D} = \begin{pmatrix} -2 & 0 & 1 & 1 & 0 \\ 0 & -2 & 1 & 0 & 1 \\ 1 & 1 & -3 & 1 & 0 \\ 1 & 0 & 1 & -3 & 1 \\ 0 & 1 & 0 & 1 & -2 \end{pmatrix}$$

The eigenvalues of $\boldsymbol{S}$ are as follows:

$$\lambda_1 = 0 \qquad \lambda_2 = -1.38 \qquad \lambda_3 = -2.38 \qquad \lambda_4 = -3.62 \qquad \lambda_5 = -4.62$$

and the eigenvectors of $\boldsymbol{S}$ are

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 & \boldsymbol{u}_4 & \boldsymbol{u}_5 \\ \hline 0.45 & -0.63 & 0.00 & 0.63 & 0.00 \\ 0.45 & 0.51 & -0.60 & 0.20 & -0.37 \\ 0.45 & -0.20 & -0.37 & -0.51 & 0.60 \\ 0.45 & -0.20 & 0.37 & -0.51 & -0.60 \\ 0.45 & 0.51 & 0.60 & 0.20 & 0.37 \end{pmatrix}$$

Graph Diffusion Kernel: Example

Assuming $\beta = 0.2$, the exponential diffusion kernel matrix is given as

$$\boldsymbol{K} = \exp\{0.2\boldsymbol{S}\} = \boldsymbol{U} \begin{pmatrix} \exp\{0.2\lambda_1\} & 0 & \cdots & 0 \\ 0 & \exp\{0.2\lambda_2\} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \exp\{0.2\lambda_n\} \end{pmatrix} \boldsymbol{U}^T$$

$$= \begin{pmatrix} 0.70 & 0.01 & 0.14 & 0.14 & 0.01 \\ 0.01 & 0.70 & 0.13 & 0.03 & 0.14 \\ 0.14 & 0.13 & 0.59 & 0.13 & 0.03 \\ 0.14 & 0.03 & 0.13 & 0.59 & 0.13 \\ 0.01 & 0.14 & 0.03 & 0.13 & 0.70 \end{pmatrix}$$

Assuming $\beta = 0.2$, the von Neumann kernel is given as

$$\boldsymbol{K} = \boldsymbol{U}(\boldsymbol{I} - 0.2\Lambda)^{-1}\boldsymbol{U}^T = \begin{pmatrix} 0.75 & 0.02 & 0.11 & 0.11 & 0.02 \\ 0.02 & 0.74 & 0.10 & 0.03 & 0.11 \\ 0.11 & 0.10 & 0.66 & 0.10 & 0.03 \\ 0.11 & 0.03 & 0.10 & 0.66 & 0.10 \\ 0.02 & 0.11 & 0.03 & 0.10 & 0.74 \end{pmatrix}$$

Data mining and Machine learning

Part 6. High-Dimensional Data

High-dimensional Space

Let $\boldsymbol{D}$ be a $n \times d$ data matrix. In data mining typically the data is very high dimensional. Understanding the nature of high-dimensional space, or *hyperspace*, is very important, especially because it does not behave like the more familiar geometry in two or three dimensions.

Hyper-rectangle: The data space is a d-dimensional *hyper-rectangle*

$$R_d = \prod_{j=1}^{d} \Big[\min(X_j), \max(X_j)\Big]$$

where $\min(X_j)$ and $max(X_j)$ specify the range of X_j.

Hypercube: Assume the data is centered, and let m denote the maximum attribute value

$$m = \max_{j=1}^{d} \max_{i=1}^{n} \Big\{ |x_{ij}| \Big\}$$

The data hyperspace can be represented as a *hypercube*, centered at 0, with all sides of length $l = 2m$, given as

$$H_d(l) = \Big\{ \boldsymbol{x} = (x_1, x_2, \ldots, x_d)^T \mid \forall i,\ x_i \in [-l/2, l/2] \Big\}$$

The *unit hypercube* has all sides of length $l = 1$, and is denoted as $H_d(1)$.

Hypersphere

Assume that the data has been centered, so that $\boldsymbol{\mu} = 0$. Let r denote the largest magnitude among all points:

$$r = \max_i \left\{ \|\boldsymbol{x}_i\| \right\}$$

The data hyperspace can be represented as a d-dimensional *hyperball* centered at 0 with radius r, defined as

$$B_d(r) = \left\{ \boldsymbol{x} \mid \|\boldsymbol{x}\| \leq r \right\} \text{or } B_d(r) = \left\{ \boldsymbol{x} = (x_1, x_2, \ldots, x_d) \mid \sum_{j=1}^{d} x_j^2 \leq r^2 \right\}$$

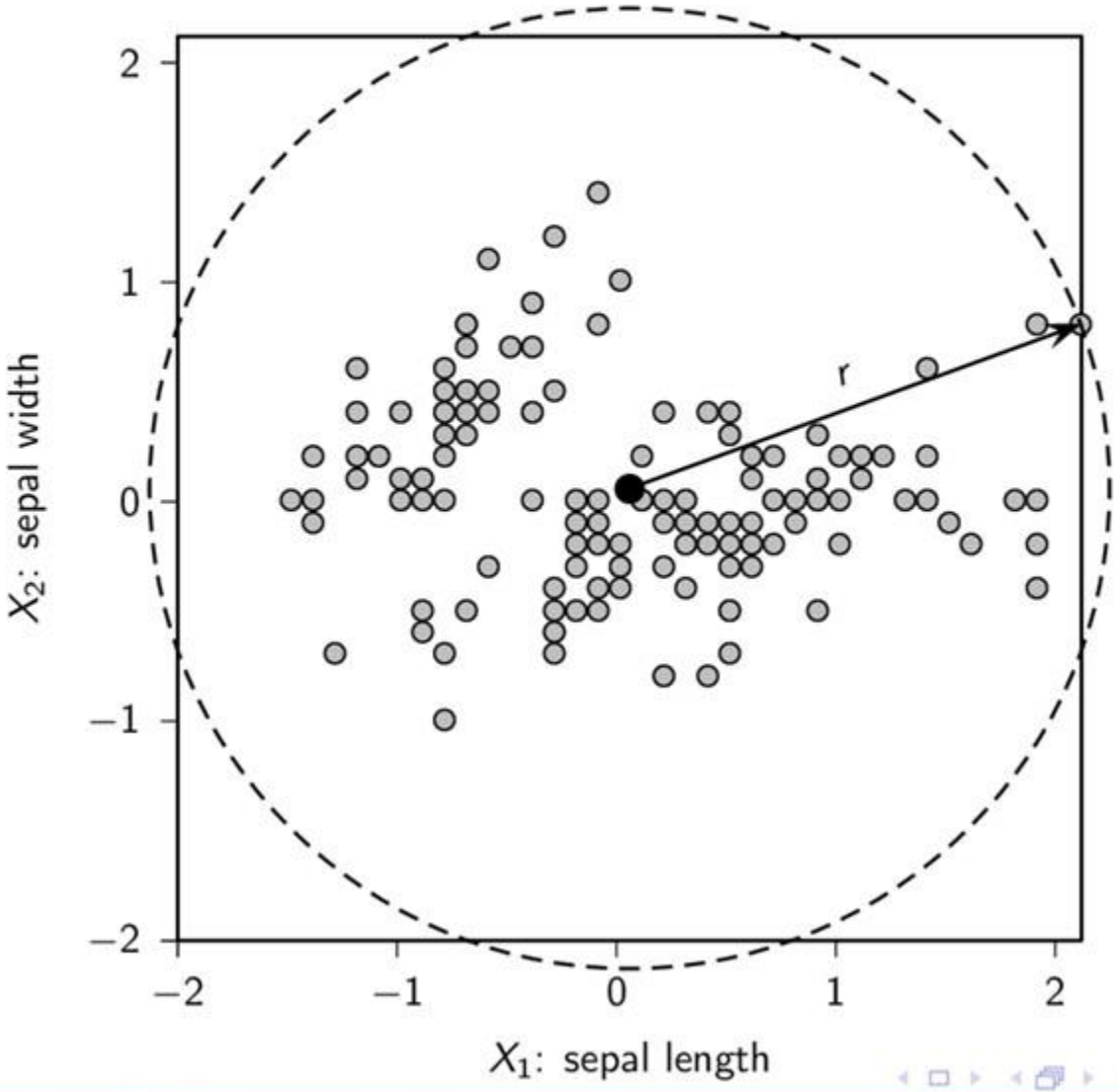
The surface of the hyperball is called a *hypersphere*, and it consists of all the points exactly at distance r from the center of the hyperball

$$S_d(r) = \left\{ \boldsymbol{x} \mid \|\boldsymbol{x}\| = r \right\}$$

$$\text{or } S_d(r) = \left\{ \boldsymbol{x} = (x_1, x_2, \ldots, x_d) \mid \sum_{j=1}^{d} (x_j)^2 = r^2 \right\}$$

Iris Data Hyperspace: Hypercube and Hypersphere

$l = 4.12$ and $r = 2.19$



High-dimensional Volumes

Hypercube: The volume of a hypercube with edge length l is given as

$$\text{vol}(H_d(l)) = l^d$$

HypersphereThe volume of a hyperball and its corresponding hypersphere is identical The volume of a hypersphere is given as

In 1D: $\text{vol}(S_1(r)) = 2r$ In 2D: $\text{vol}(S_2(r)) = \pi r^2$ In 3D: $\text{vol}(S_3(r)) = \frac{4}{3}\pi r^3$

In d-dimensions: $$\text{vol}(S_d(r)) = K_d r^d = \left(\frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}+1\right)}\right) r^d$$

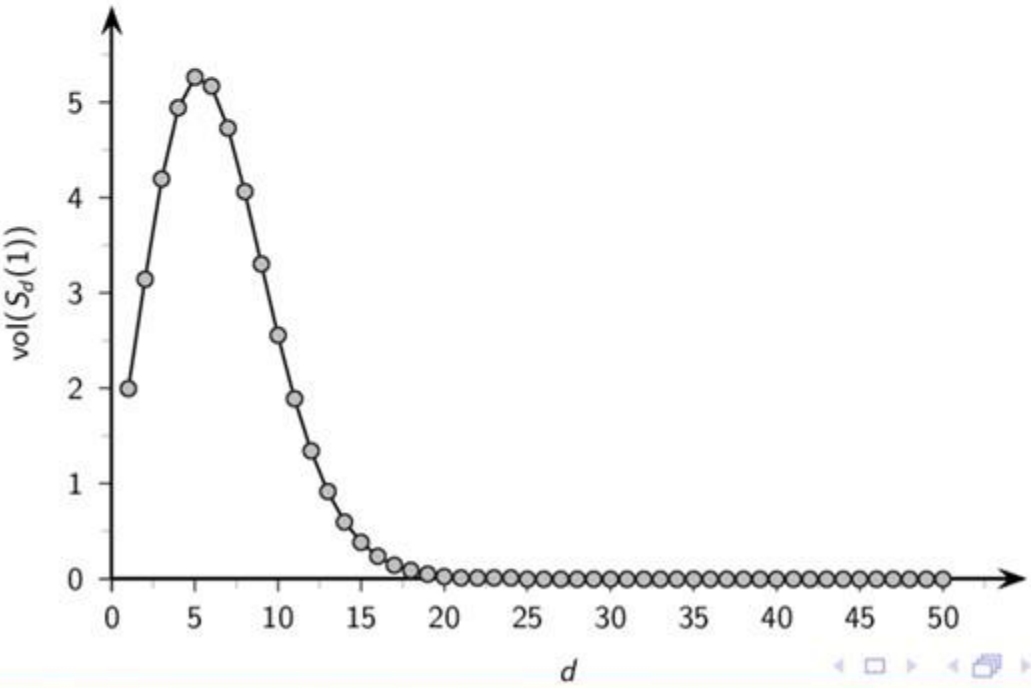
where

$$\Gamma\left(\frac{d}{2}+1\right) = \begin{cases} \left(\frac{d}{2}\right)! & \text{if } d \text{ is even} \\ \sqrt{\pi}\left(\frac{d!!}{2^{(d+1)/2}}\right) & \text{if } d \text{ is odd} \end{cases}$$

Volume of Unit Hypersphere

With increasing dimensionality the hypersphere volume first increases up to a point, and then starts to decrease, and ultimately vanishes. In particular, for the unit hypersphere with $r = 1$,

$$\lim_{d \to \infty} \text{vol}(S_d(1)) = \lim_{d \to \infty} \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2} + 1)} \to 0$$



Hypersphere Inscribed within Hypercube

Consider the space enclosed within the largest hypersphere that can be accommodated within a hypercube (which represents the dataspace).

The ratio of the volume of the hypersphere of radius r to the hypercube with side length $l = 2r$ is given as

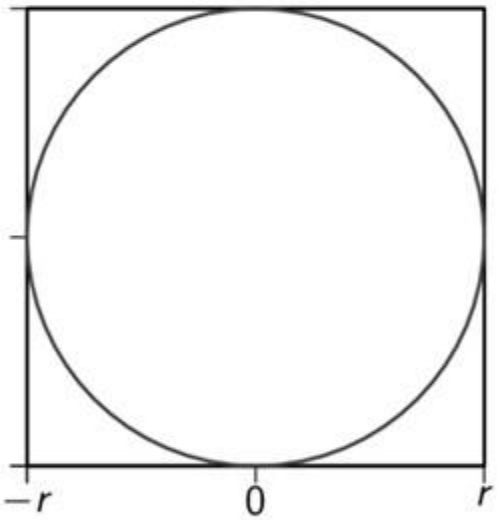
In 2 dimensions: $$\frac{\text{vol}(S_2(r))}{\text{vol}(H_2(2r))} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4} = 78.5\%$$

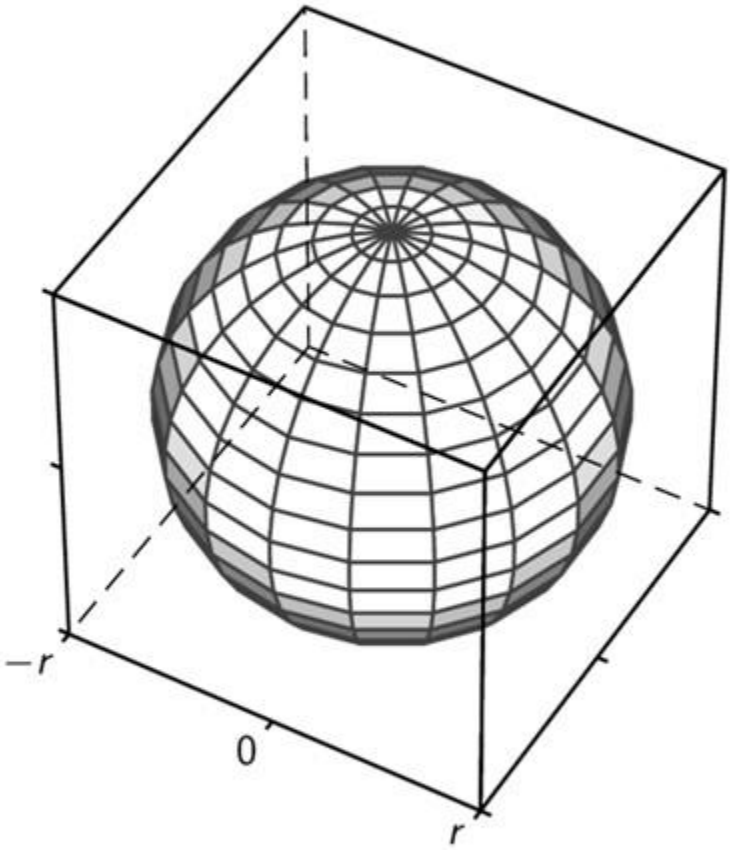
In 3 dimensions: $$\frac{\text{vol}(S_3(r))}{\text{vol}(H_3(2r))} = \frac{\frac{4}{3}\pi r^3}{8r^3} = \frac{\pi}{6} = 52.4\%$$

In d dimensions: $$\lim_{d\to\infty} \frac{\text{vol}(S_d(r))}{\text{vol}(H_d(2r))} = \lim_{d\to\infty} \frac{\pi^{d/2}}{2^d \Gamma(\frac{d}{2}+1)} \to 0$$

As the dimensionality increases, most of the volume of the hypercube is in the "corners," whereas the center is essentially empty.

Hypersphere Inscribed inside a Hypercube



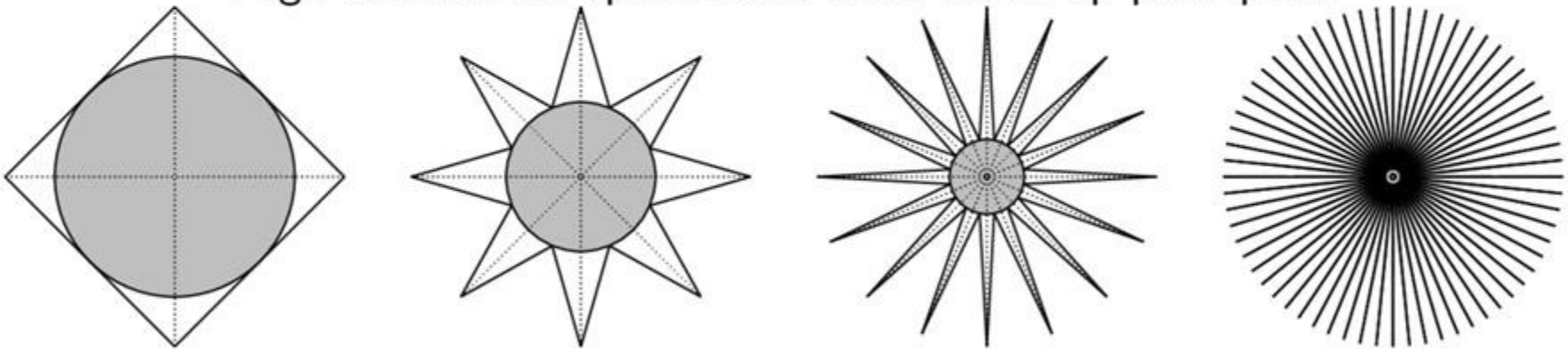


Conceptual View of High-dimensional Space

Two, three, four, and higher dimensions

All the volume of the hyperspace is in the corners, with the center being essentially empty.

High-dimensional space looks like a rolled-up porcupine!



(a) 2D (b) 3D (c) 4D (d) dD

Volume of a Thin Shell

The volume of a thin hypershell of width ϵ is given as

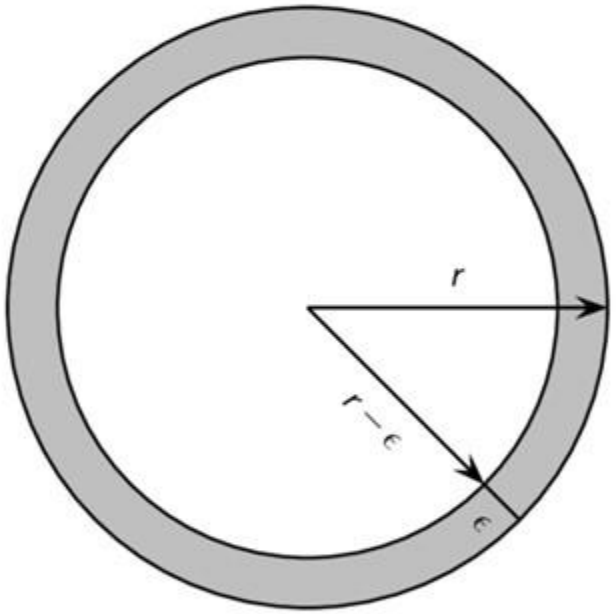
$$\begin{aligned}\text{vol}(S_d(r,\epsilon)) &= \text{vol}(S_d(r)) - \text{vol}(S_d(r-\epsilon)) \\ &= K_d r^d - K_d (r-\epsilon)^d.\end{aligned}$$

The ratio of volume of the thin shell to the volume of the outer sphere:

$$\frac{\text{vol}(S_d(r,\epsilon))}{\text{vol}(S_d(r))} = \frac{K_d r^d - K_d (r-\epsilon)^d}{K_d r^d} = 1 - \left(1 - \frac{\epsilon}{r}\right)^d$$

As d increases, we have

$$\lim_{d\to\infty} \frac{\text{vol}(S_d(r,\epsilon))}{\text{vol}(S_d(r))} = \lim_{d\to\infty} 1 - \left(1 - \frac{\epsilon}{r}\right)^d \to 1$$



Diagonals in Hyperspace

Consider a d-dimensional hypercube, with origin $0_d = (0_1, 0_2, \ldots, 0_d)$, and bounded in each dimension in the range $[-1, 1]$. Each "corner" of the hyperspace is a d-dimensional vector of the form $(\pm 1_1, \pm 1_2, \ldots, \pm 1_d)^T$.

Let $\boldsymbol{e}_i = (0_1, \ldots, 1_i, \ldots, 0_d)^T$ denote the d-dimensional canonical unit vector in dimension i, and let 1 denote the d-dimensional diagonal vector $(1_1, 1_2, \ldots, 1_d)^T$.

Consider the angle θ_d between the diagonal vector 1 and the first axis $\boldsymbol{e}_1$, in d dimensions:

$$\cos\theta_d = \frac{\boldsymbol{e}_1^T 1}{\|\boldsymbol{e}_1\| \, \|1\|} = \frac{\boldsymbol{e}_1^T 1}{\sqrt{\boldsymbol{e}_1^T \boldsymbol{e}_1}\sqrt{1^T 1}} = \frac{1}{\sqrt{1}\sqrt{d}} = \frac{1}{\sqrt{d}}$$

Diagonals in Hyperspace

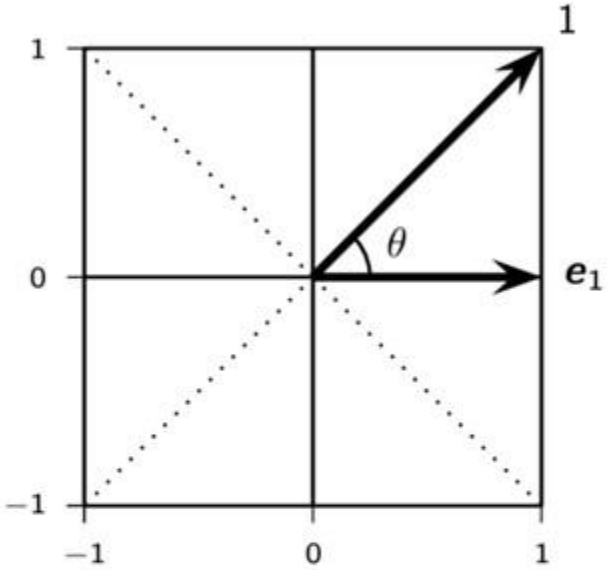
As d increases, we have

$$\lim_{d\to\infty} \cos\theta_d = \lim_{d\to\infty} \frac{1}{\sqrt{d}} \to 0$$

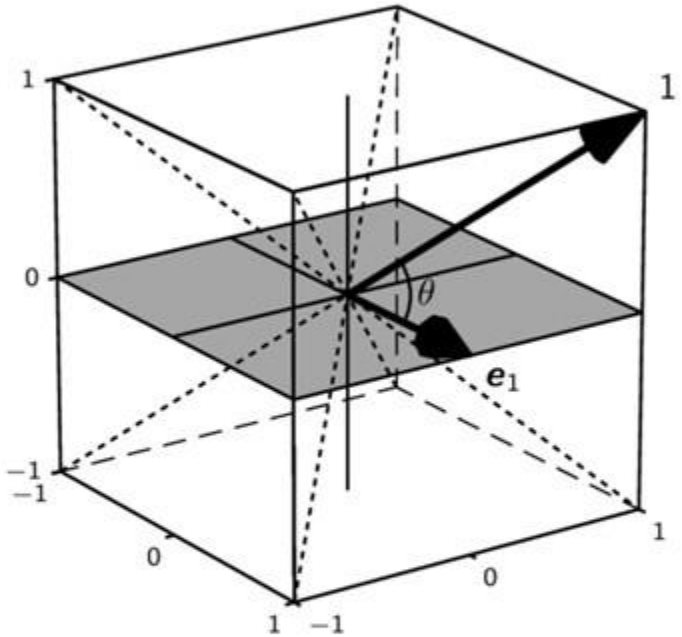
which implies that

$$\lim_{d\to\infty} \theta_d \to \frac{\pi}{2} = 90^\circ$$

Angle between Diagonal Vector 1 and $\boldsymbol{e}_1$



(a) In 2D



(b) In 3D

In high dimensions all of the diagonal vectors are perpendicular (or orthogonal) to all the coordinates axes! Each of the 2^{d-1} new axes connecting pairs of 2^d corners are essentially orthogonal to all of the d principal coordinate axes! Thus, in effect, high-dimensional space has an exponential number of orthogonal "axes."

Density of the Multivariate Normal

Consider the standard multivariate normal distribution with $\mu = 0$, and $\Sigma = I$

$$f(x) = \frac{1}{(\sqrt{2\pi})^d} \exp\left\{-\frac{x^T x}{2}\right\}$$

The peak of the density is at the mean. Consider the set of points x with density at least α fraction of the density at the mean

$$\frac{f(x)}{f(0)} \geq \alpha$$

$$\exp\left\{-\frac{x^T x}{2}\right\} \geq \alpha$$

$$x^T x \leq -2\ln(\alpha)$$

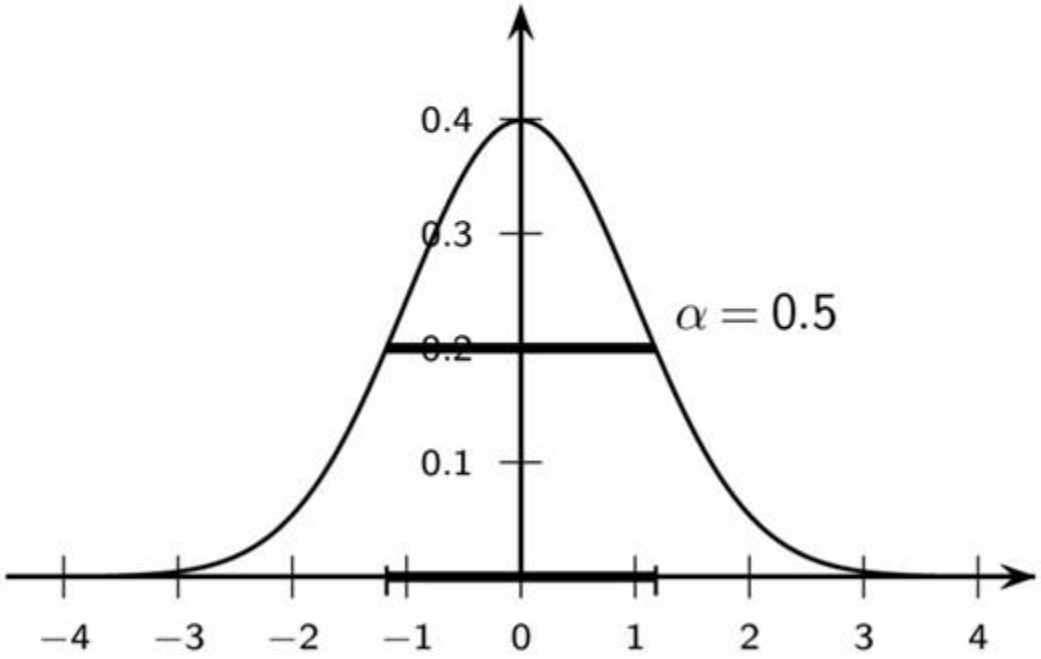
$$\sum_{i=1}^{d} (x_i)^2 \leq -2\ln(\alpha)$$

The sum of squared IID random variables follows a chi-squared distribution χ^2_d. Thus,

$$P\left(\frac{f(x)}{f(0)} \geq \alpha\right) = F_{\chi^2_d}(-2\ln(\alpha))$$

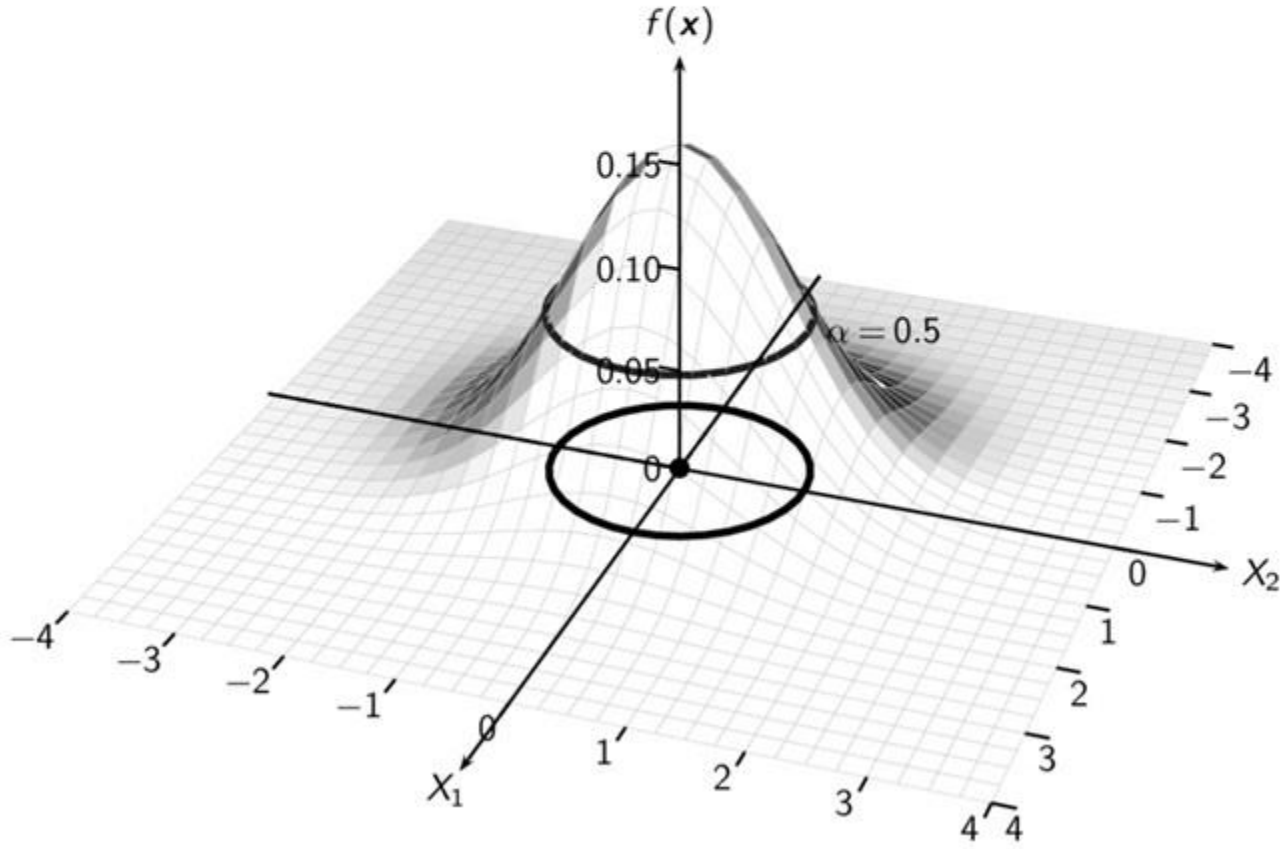
Density Contour for α Fraction of the Density at the Mean: One Dimension

Let $\alpha = 0.5$, then $-2\ln(0.5) = 1.386$ and $F_{\chi^2_1}(1.386) = 0.76$. Thus, 24% of the density is in the tail regions.



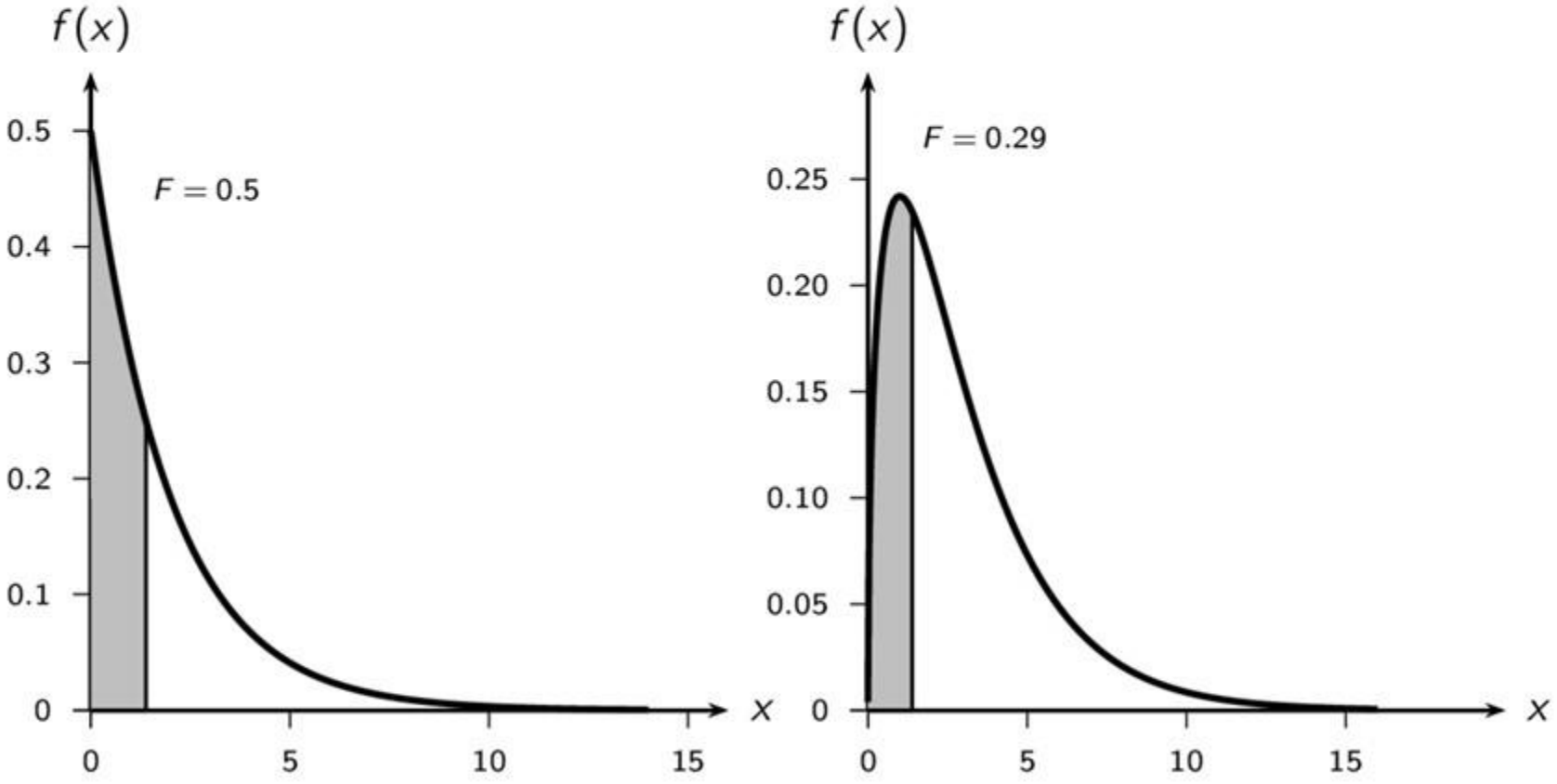
Density Contour for α Fraction of the Density at the Mean: Two Dimensions

Let $\alpha = 0.5$, then $-2\ln(0.5) = 1.386$ and $F_{\chi^2_2}(1.386) = 0.50$. Thus, 50% of the density is in the tail regions.

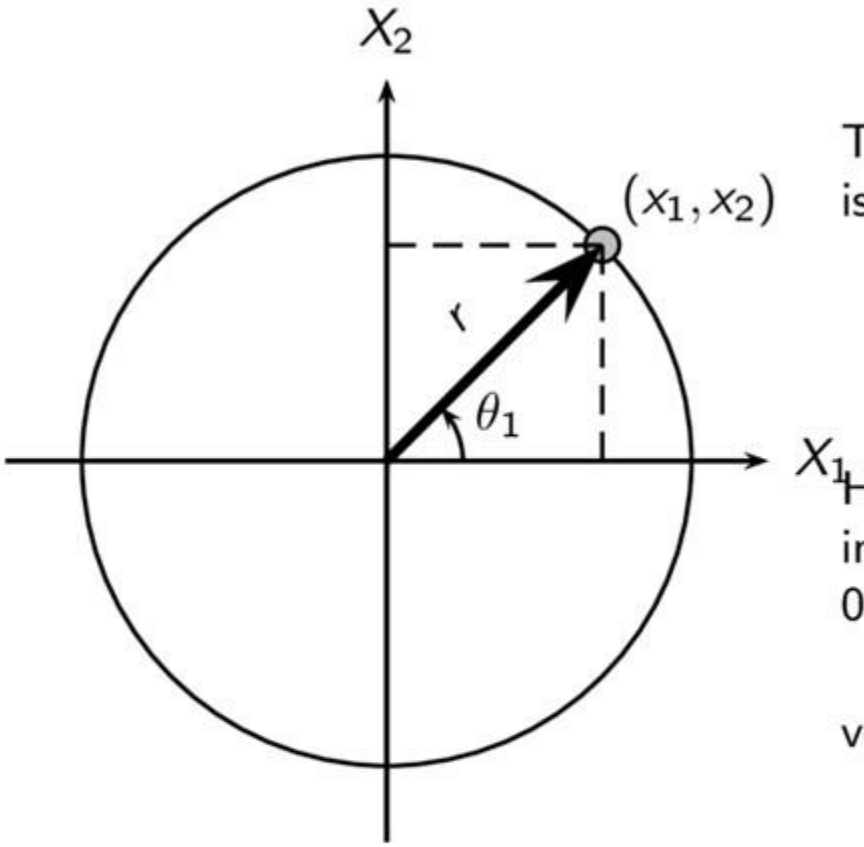


Chi-Squared Distribution: $P(f(\boldsymbol{x})/f(0) \geq \alpha)$

This probability decreases rapidly with dimensionality. For 2D, it is 0.5. For 3D it is 0.29, ie., 71% of the density is in the tails. By $d = 10$, it decreases to 0.075%, that is, 99.925% of the points lie in the extreme or tail regions.



Hypersphere Volume: Polar Coordinates in 2D



The point $\boldsymbol{x} = (x_1, x_2)$ in polar coordinates

$$x_1 = r\cos\theta_1 = rc_1$$

$$x_2 = r\sin\theta_1 = rs_1$$

where $r = \|\boldsymbol{x}\|$, and $\cos\theta_1 = c_1$ and $\sin\theta_1 = s_1$.

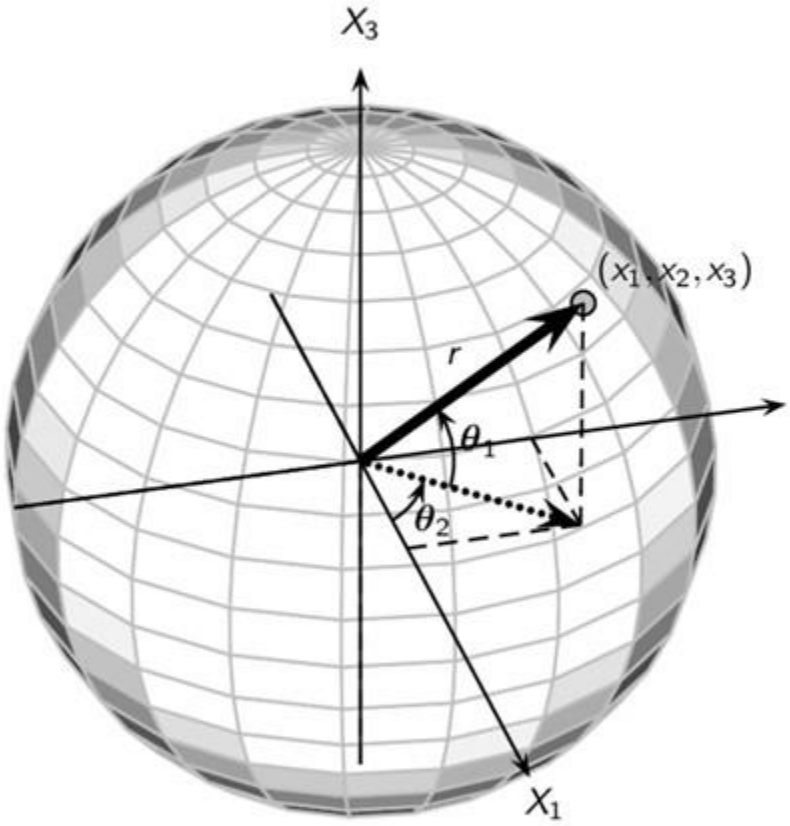
The *Jacobian matrix* for this transformation is given as

$$J(\theta_1) = \begin{pmatrix} \frac{\partial x_1}{\partial r} & \frac{\partial x_1}{\partial \theta_1} \\ \frac{\partial x_2}{\partial r} & \frac{\partial x_2}{\partial \theta_1} \end{pmatrix} = \begin{pmatrix} c_1 & -rs_1 \\ s_1 & rc_1 \end{pmatrix}$$

Hypersphere volume is obtained by integration over r and θ_1 (with $r > 0$, and $0 \leq \theta_1 \leq 2\pi$):

$$\begin{aligned} \text{vol}(S_2(r)) &= \int_r \int_{\theta_1} \left|\det(J(\theta_1))\right| \, dr \, d\theta_1 \\ &= \int_0^r \int_0^{2\pi} r \, dr \, d\theta_1 = \int_0^r r \, dr \int_0^{2\pi} d\theta_1 \\ &= \frac{r^2}{2}\Big|_0^r \cdot \theta_1 \Big|_0^{2\pi} = \pi r^2 \end{aligned}$$

Hypersphere Volume: Polar Coordinates in 3D



$\boldsymbol{x} = (x_1, x_2, x_2)$ in polar coordinates

$$x_1 = r\cos\theta_1\cos\theta_2 = rc_1c_2$$
$$x_2 = r\cos\theta_1\sin\theta_2 = rc_1s_2$$
$$x_3 = r\sin\theta_1 = rs_1$$

The Jacobian matrix is given as

$$J(\theta_1, \theta_2) = \begin{pmatrix} c_1c_2 & -rs_1c_2 & -rc_1s_2 \\ c_1s_2 & -rs_1s_2 & rc_1c_2 \\ s_1 & rc_1 & 0 \end{pmatrix}$$

The volume of the hypersphere for $d = 3$ is obtained via a triple integral with $r > 0$, $-\pi/2 \leq \theta_1 \leq \pi/2$, and $0 \leq \theta_2 \leq 2\pi$

$$\text{vol}(S_3(r)) = \int_r \int_{\theta_1} \int_{\theta_2} \left|\det(J(\theta_1, \theta_2))\right| \, dr \, d\theta_1 \, d\theta_2$$
$$= \frac{4}{3}\pi r^3$$

Hypersphere Volume in d Dimensions

The determinant of the d-dimensional Jacobian matrix is

$$\det(J(\theta_1, \theta_2, \ldots, \theta_{d-1})) = (-1)^d r^{d-1} c_1^{d-2} c_2^{d-3} \ldots c_{d-2}$$

The volume of the hypersphere is given by the d-dimensional integral with $r > 0$, $-\pi/2 \leq \theta_i \leq \pi/2$ for all $i = 1, \ldots, d-2$, and $0 \leq \theta_{d-1} \leq 2\pi$:

$$\begin{aligned}
\text{vol}(S_d(r)) &= \int_r \int_{\theta_1} \int_{\theta_2} \cdots \int_{\theta_{d-1}} \left|\det(J(\theta_1, \theta_2, \ldots, \theta_{d-1}))\right| \, dr \, d\theta_1 \, d\theta_2 \ldots d\theta_{d-1} \\
&= \int_0^r r^{d-1} dr \int_{-\pi/2}^{\pi/2} c_1^{d-2} d\theta_1 \cdots \int_{-\pi/2}^{\pi/2} c_{d-2} d\theta_{d-2} \int_0^{2\pi} d\theta_{d-1} \\
&= \frac{r^d}{d} \frac{\Gamma\left(\frac{d-1}{2}\right)\Gamma\left(\frac{1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)} \frac{\Gamma\left(\frac{d-2}{2}\right)\Gamma\left(\frac{1}{2}\right)}{\Gamma\left(\frac{d-1}{2}\right)} \cdots \frac{\Gamma(1)\Gamma\left(\frac{1}{2}\right)}{\Gamma\left(\frac{3}{2}\right)} 2\pi \\
&= \frac{\pi \Gamma\left(\frac{1}{2}\right)^{d/2-1} r^d}{\frac{d}{2}\Gamma\left(\frac{d}{2}\right)} \\
&= \left(\frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)}\right) r^d
\end{aligned}$$

Data mining and Machine learning

Part 7. Dimensionality Reduction

Dimensionality Reduction

The goal of dimensionality reduction is to find a lower dimensional representation of the data matrix $\boldsymbol{D}$ to avoid the curse of dimensionality.

Given $n \times d$ data matrix, each point $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})^T$ is a vector in the ambient d-dimensional vector space spanned by the d standard basis vectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d$.

Given any other set of d orthonormal vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_d$ we can re-express each point $\boldsymbol{x}$ as

$$\boldsymbol{x} = a_1\boldsymbol{u}_1 + a_2\boldsymbol{u}_2 + \cdots + a_d\boldsymbol{u}_d$$

where $\boldsymbol{a} = (a_1, a_2, \ldots, a_d)^T$ represents the coordinates of $\boldsymbol{x}$ in the new basis. More compactly:

$$\boldsymbol{x} = \boldsymbol{U}\boldsymbol{a}$$

where $\boldsymbol{U}$ is the $d \times d$ orthogonal matrix, whose ith column comprises the ith basis vector $\boldsymbol{u}_i$. Thus $\boldsymbol{U}^{-1} = \boldsymbol{U}^T$, and we have

$$\boldsymbol{a} = \boldsymbol{U}^T\boldsymbol{x}$$

Optimal Basis: Projection in Lower Dimensional Space

There are potentially infinite choices for the orthonormal basis vectors. Our goal is to choose an *optimal* basis that preserves essential information about $\boldsymbol{D}$.

We are interested in finding the optimal r-dimensional representation of $\boldsymbol{D}$, with $r \ll d$. Projection of $\boldsymbol{x}$ onto the first r basis vectors is given as

$$\boldsymbol{x}' = a_1\boldsymbol{u}_1 + a_2\boldsymbol{u}_2 + \cdots + a_r\boldsymbol{u}_r = \sum_{i=1}^{r} a_i\boldsymbol{u}_i = \boldsymbol{U}_r\boldsymbol{a}_r$$

where $\boldsymbol{U}_r$ and $\boldsymbol{a}_r$ comprises the r basis vectors and coordinates, respv. Also, restricting $\boldsymbol{a} = \boldsymbol{U}^T\boldsymbol{x}$ to r terms, we have

$$\boldsymbol{a}_r = \boldsymbol{U}_r^T\boldsymbol{x}$$

The r-dimensional projection of $\boldsymbol{x}$ is thus given as:

$$\boldsymbol{x}' = \boldsymbol{U}_r\boldsymbol{U}_r^T\boldsymbol{x} = \boldsymbol{P}_r\boldsymbol{x}$$

where $\boldsymbol{P}_r = \boldsymbol{U}_r\boldsymbol{U}_r^T = \sum_{i=1}^{r} \boldsymbol{u}_i\boldsymbol{u}_i^T$ is the *orthogonal projection matrix* for the subspace spanned by the first r basis vectors.

Optimal Basis: Error Vector

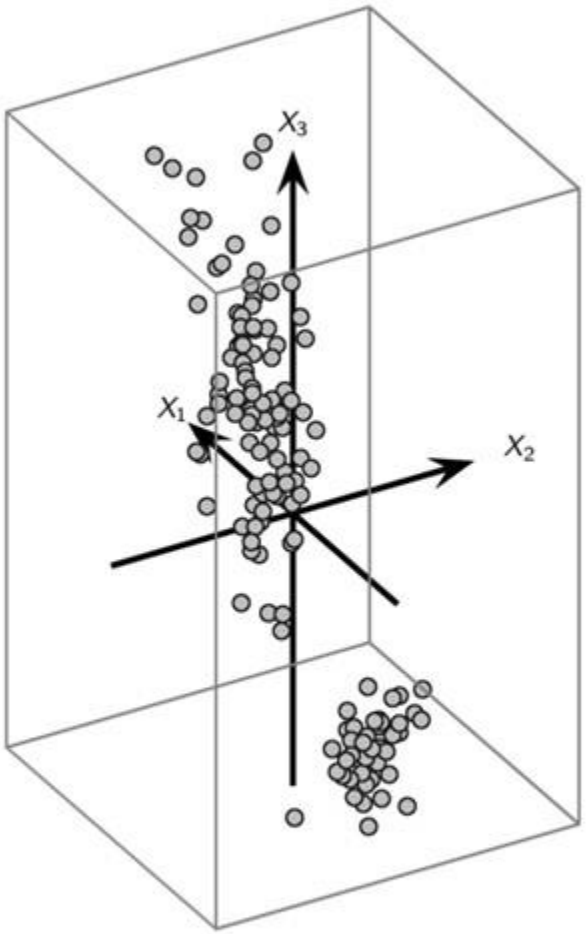
Given the projected vector $\boldsymbol{x}' = \boldsymbol{P}_r \boldsymbol{x}$, the corresponding *error vector*, is the projection onto the remaining $d - r$ basis vectors

$$\boldsymbol{\epsilon} = \sum_{i=r+1}^{d} a_i \boldsymbol{u}_i = \boldsymbol{x} - \boldsymbol{x}'$$

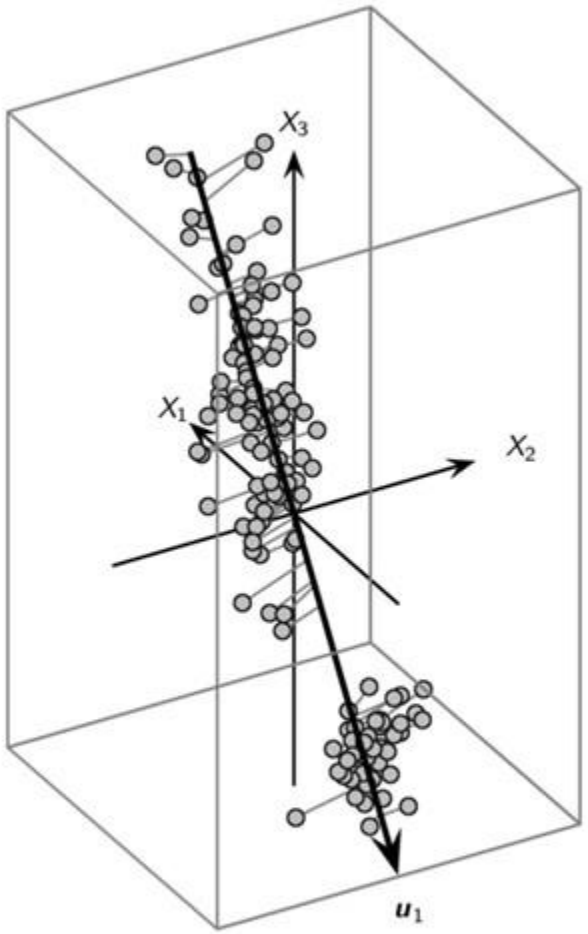
The error vector $\boldsymbol{\epsilon}$ is orthogonal to $\boldsymbol{x}'$.

The goal of dimensionality reduction is to seek an r-dimensional basis that gives the best possible approximation $\boldsymbol{x}'_i$ over all the points $\boldsymbol{x}_i \in \boldsymbol{D}$. Alternatively, we seek to minimize the error $\boldsymbol{\epsilon}_i = \boldsymbol{x}_i - \boldsymbol{x}'_i$ over all the points.

Iris Data: Optimal One-dimensional Basis

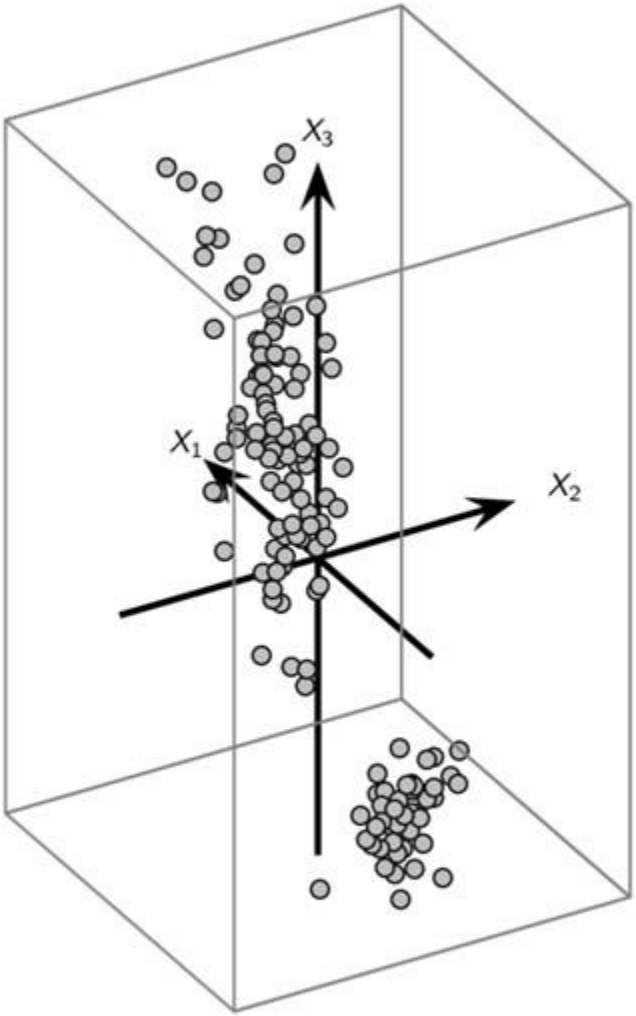


Iris Data: 3D

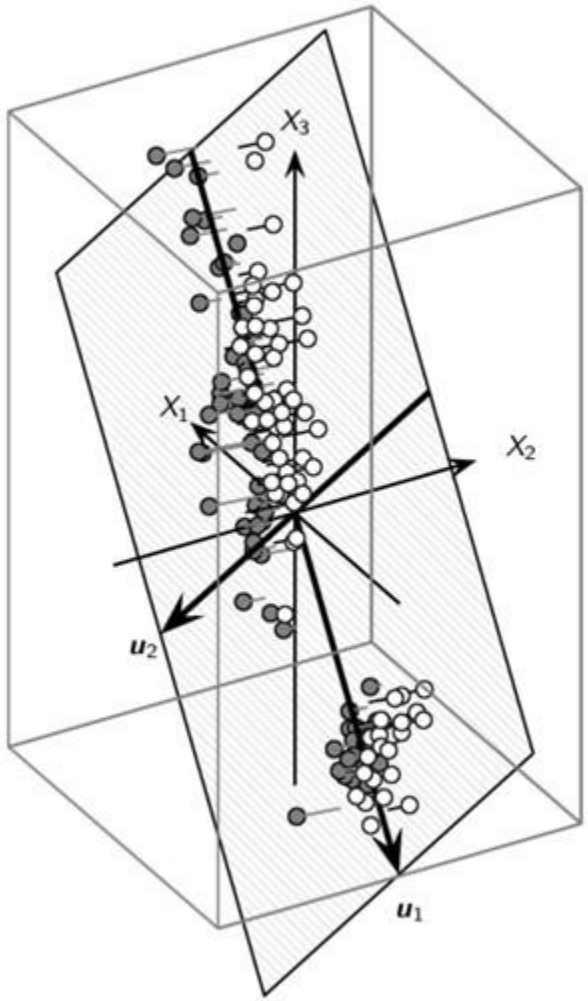


Optimal 1D Basis

Iris Data: Optimal 2D Basis



Iris Data (3D)



Optimal 2D Basis

Principal Component Analysis

Principal Component Analysis (PCA) is a technique that seeks a r-dimensional basis that best captures the variance in the data.

The direction with the largest projected variance is called the first principal component.

The orthogonal direction that captures the second largest projected variance is called the second principal component, and so on.

The direction that maximizes the variance is also the one that minimizes the mean squared error.

Principal Component: Direction of Most Variance

We seek to find the unit vector $\boldsymbol{u}$ that maximizes the projected variance of the points. Let $\boldsymbol{D}$ be centered, and let Σ be its covariance matrix.

The projection of $\boldsymbol{x}_i$ on $\boldsymbol{u}$ is given as

$$\boldsymbol{x}_i' = \left(\frac{\boldsymbol{u}^T \boldsymbol{x}_i}{\boldsymbol{u}^T \boldsymbol{u}}\right) \boldsymbol{u} = (\boldsymbol{u}^T \boldsymbol{x}_i)\boldsymbol{u} = a_i \boldsymbol{u}$$

Across all the points, the projected variance along $\boldsymbol{u}$ is

$$\sigma_{\boldsymbol{u}}^2 = \frac{1}{n}\sum_{i=1}^{n}(a_i - \mu_{\boldsymbol{u}})^2 = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{u}^T\left(\boldsymbol{x}_i \boldsymbol{x}_i^T\right)\boldsymbol{u} = \boldsymbol{u}^T\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i \boldsymbol{x}_i^T\right)\boldsymbol{u} = \boldsymbol{u}^T \Sigma \boldsymbol{u}$$

We have to find the optimal basis vector $\boldsymbol{u}$ that maximizes the projected variance $\sigma_{\boldsymbol{u}}^2 = \boldsymbol{u}^T \Sigma \boldsymbol{u}$, subject to the constraint that $\boldsymbol{u}^T \boldsymbol{u} = 1$. The maximization objective is given as

$$\max_{\boldsymbol{u}} J(\boldsymbol{u}) = \boldsymbol{u}^T \Sigma \boldsymbol{u} - \alpha(\boldsymbol{u}^T \boldsymbol{u} - 1)$$

Principal Component: Direction of Most Variance

Given the objective $\max_{\boldsymbol{u}} J(\boldsymbol{u}) = \boldsymbol{u}^T \Sigma \boldsymbol{u} - \alpha(\boldsymbol{u}^T \boldsymbol{u} - 1)$, we solve it by setting the derivative of $J(\boldsymbol{u})$ with respect to $\boldsymbol{u}$ to the zero vector, to obtain

$$\frac{\partial}{\partial \boldsymbol{u}} \left(\boldsymbol{u}^T \Sigma \boldsymbol{u} - \alpha(\boldsymbol{u}^T \boldsymbol{u} - 1) \right) = 0$$

$$\text{that is, } 2\Sigma \boldsymbol{u} - 2\alpha \boldsymbol{u} = 0$$

$$\text{which implies } \boxed{\Sigma \boldsymbol{u} = \alpha \boldsymbol{u}}$$

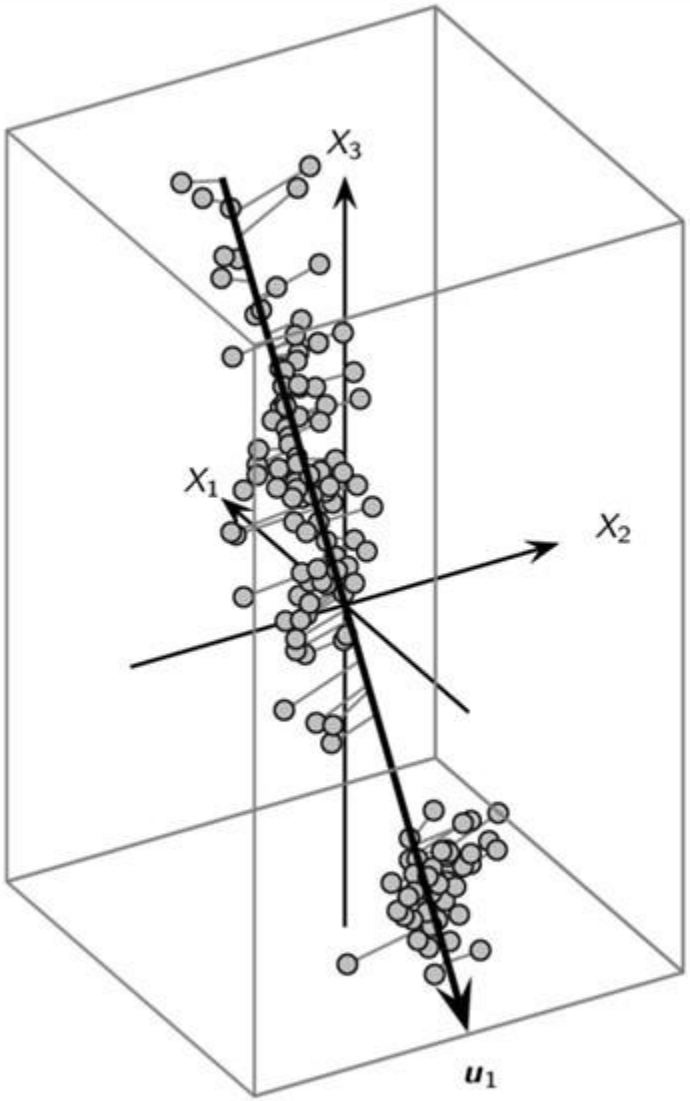
Thus α is an eigenvalue of the covariance matrix Σ, with the associated eigenvector $\boldsymbol{u}$.

Taking the dot product with $\boldsymbol{u}$ on both sides, we have

$$\sigma_{\boldsymbol{u}}^2 = \boldsymbol{u}^T \Sigma \boldsymbol{u} \boldsymbol{u}^T \alpha \boldsymbol{u} = \alpha \boldsymbol{u}^T \boldsymbol{u} = \alpha$$

To maximize the projected variance $\sigma_{\boldsymbol{u}}^2$, we thus choose the largest eigenvalue λ_1 of Σ, and the dominant eigenvector $\boldsymbol{u}_1$ specifies the direction of most variance, also called the *first principal component*.

Iris Data: First Principal Component



Minimum Squared Error Approach

The direction that maximizes the projected variance is also the one that minimizes the average squared error. The mean squared error (MSE) optimization condition is

$$MSE(\boldsymbol{u}) = \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{\epsilon}_i\|^2 = \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i - \boldsymbol{x}_i'\|^2 = \sum_{i=1}^{n}\frac{\|\boldsymbol{x}_i\|^2}{n} - \boldsymbol{u}^T\Sigma\boldsymbol{u}$$

Since the first term is fixed for a dataset $\boldsymbol{D}$, we see that the direction $\boldsymbol{u}_1$ that maximizes the variance is also the one that minimizes the MSE. Further,

$$\sum_{i=1}^{n}\frac{\|\boldsymbol{x}_i\|^2}{n} - \boldsymbol{u}^T\Sigma\boldsymbol{u} = var(\boldsymbol{D}) = tr(\Sigma) = \sum_{i=1}^{d}\sigma_i^2$$

Thus, for the direction $\boldsymbol{u}_1$ that minimizes MSE, we have

$$MSE(\boldsymbol{u}_1) = var(\boldsymbol{D}) - \boldsymbol{u}_1^T\Sigma\boldsymbol{u}_1 = var(\boldsymbol{D}) - \lambda_1$$

Best 2-dimensional Approximation

The best 2D subspace that captures the most variance in $\boldsymbol{D}$ comprises the eigenvectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ corresponding to the largest and second largest eigenvalues λ_1 and λ_2, respv.

Let $\boldsymbol{U}_2 = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 \end{pmatrix}$ be the matrix whose columns correspond to the two principal components. Given the point $\boldsymbol{x}_i \in \boldsymbol{D}$ its projected coordinates are computed as follows:

$$\boldsymbol{a}_i = \boldsymbol{U}_2^T \boldsymbol{x}_i$$

Let $\boldsymbol{A}$ denote the projected 2D dataset. The total projected variance for $\boldsymbol{A}$ is given as

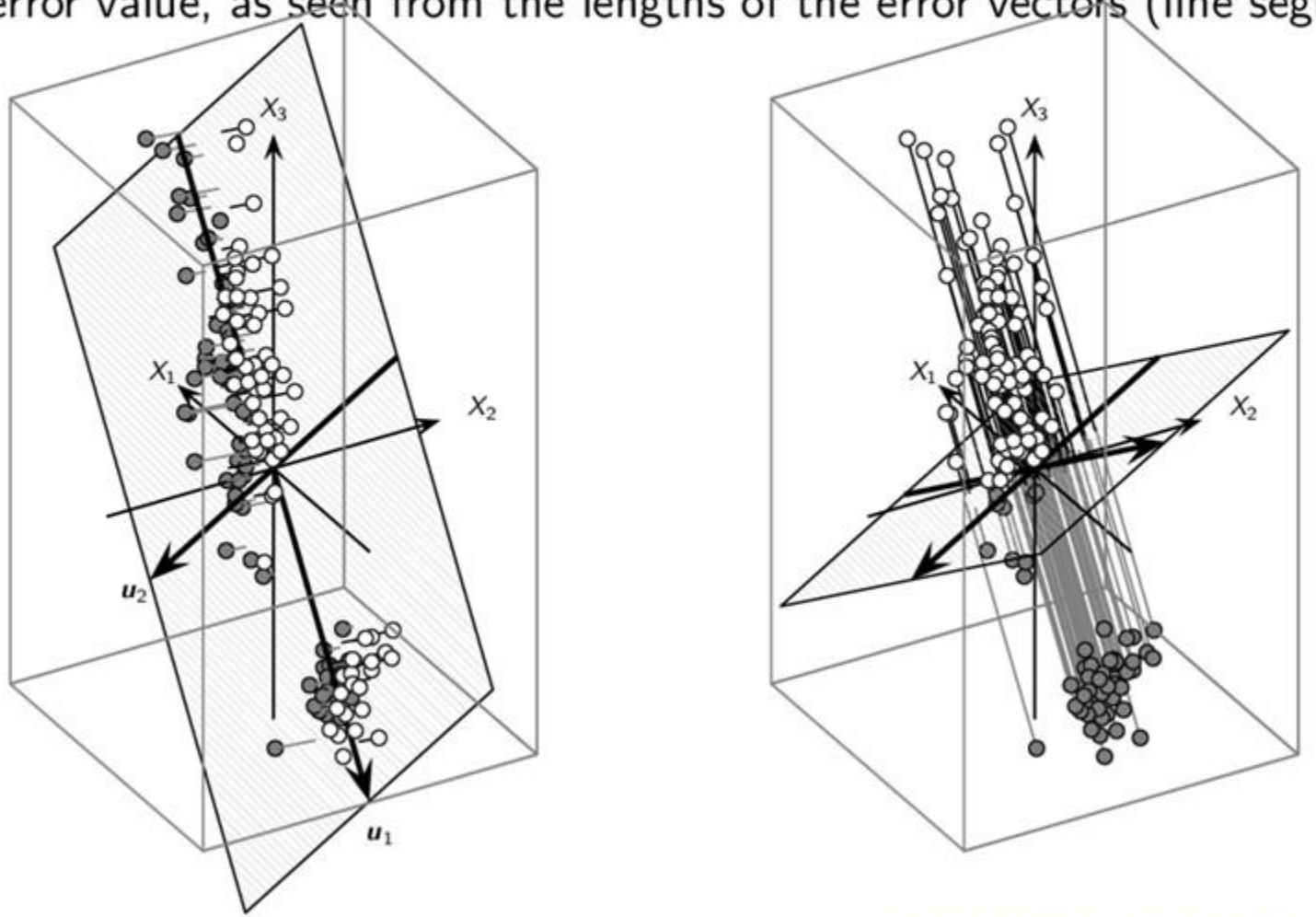
$$var(\boldsymbol{A}) = \boldsymbol{u}_1^T \Sigma \boldsymbol{u}_1 + \boldsymbol{u}_2^T \Sigma \boldsymbol{u}_2 = \boldsymbol{u}_1^T \lambda_1 \boldsymbol{u}_1 + \boldsymbol{u}_2^T \lambda_2 \boldsymbol{u}_2 = \lambda_1 + \lambda_2$$

The first two principal components also minimize the mean square error objective, since

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{x}_i'\|^2 = var(\boldsymbol{D}) - \frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{x}_i^T \boldsymbol{P}_2 \boldsymbol{x}_i\right) = var(\boldsymbol{D}) - var(\boldsymbol{A})$$

Optimal and Non-optimal 2D Approximations

The optimal subspace maximizes the variance, and minimizes the squared error, whereas the nonoptimal subspace captures less variance, and has a high mean squared error value, as seen from the lengths of the error vectors (line segments).



Best r-dimensional Approximation

To find the best r-dimensional approximation to $\boldsymbol{D}$, we compute the eigenvalues of Σ. Because Σ is positive semidefinite, its eigenvalues are non-negative

$$\lambda_1 \geq \lambda_2 \geq \cdots \lambda_r \geq \lambda_{r+1} \cdots \geq \lambda_d \geq 0$$

We select the r largest eigenvalues, and their corresponding eigenvectors to form the best r-dimensional approximation.

Total Projected Variance: Let $\boldsymbol{U}_r = \begin{pmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_r \end{pmatrix}$ be the r-dimensional basis vector matrix, withe the projection matrix given as $\boldsymbol{P}_r = \boldsymbol{U}_r \boldsymbol{U}_r^T = \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{u}_i^T$.

Let $\boldsymbol{A}$ denote the dataset formed by the coordinates of the projected points in the r-dimensional subspace. The projected variance is given as

$$var(\boldsymbol{A}) = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i^T \boldsymbol{P}_r \boldsymbol{x}_i = \sum_{i=1}^{r} \boldsymbol{u}_i^T \Sigma \boldsymbol{u}_i = \sum_{i=1}^{r} \lambda_i$$

Mean Squared Error: The mean squared error in r dimensions is

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left\| \boldsymbol{x}_i - \boldsymbol{x}_i' \right\|^2 = var(\boldsymbol{D}) - \sum_{i=1}^{r} \lambda_i = \sum_{i=1}^{d} \lambda_i - \sum_{i=1}^{r} \lambda_i$$

Choosing the Dimensionality

One criteria for choosing r is to compute the fraction of the total variance captured by the first r principal components, computed as

$$f(r) = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_r}{\lambda_1 + \lambda_2 + \cdots + \lambda_d} = \frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{d} \lambda_i} = \frac{\sum_{i=1}^{r} \lambda_i}{var(\boldsymbol{D})}$$

Given a certain desired variance threshold, say α, starting from the first principal component, we keep on adding additional components, and stop at the smallest value r, for which $f(r) \geq \alpha$. In other words, we select the fewest number of dimensions such that the subspace spanned by those r dimensions captures at least α fraction (say 0.9) of the total variance.

Principal Component Analysis: Algorithm

PCA $(\boldsymbol{D}, \alpha)$:

1 $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$ // compute mean

2 $\boldsymbol{Z} = \boldsymbol{D} - 1 \cdot \boldsymbol{\mu}^T$ // center the data

3 $\Sigma = \frac{1}{n}\left(\boldsymbol{Z}^T\boldsymbol{Z}\right)$ // compute covariance matrix

4 $(\lambda_1, \lambda_2, \ldots, \lambda_d) = \text{eigenvalues}(\Sigma)$ // compute eigenvalues

5 $\boldsymbol{U} = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_d \end{pmatrix} = \text{eigenvectors}(\Sigma)$ // compute eigenvectors

6 $f(r) = \frac{\sum_{i=1}^{r}\lambda_i}{\sum_{i=1}^{d}\lambda_i}$, for all $r = 1, 2, \ldots, d$ // fraction of total variance

7 Choose smallest r so that $f(r) \geq \alpha$ // choose dimensionality

8 $\boldsymbol{U}_r = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_r \end{pmatrix}$ // reduced basis

9 $\boldsymbol{A} = \{\boldsymbol{a}_i \mid \boldsymbol{a}_i = \boldsymbol{U}_r^T\boldsymbol{x}_i, \text{for } i = 1, \ldots, n\}$ // reduced dimensionality data

Iris Principal Components

Covariance matrix:

$$\Sigma = \begin{pmatrix} 0.681 & -0.039 & 1.265 \\ -0.039 & 0.187 & -0.320 \\ 1.265 & -0.32 & 3.092 \end{pmatrix}$$

The eigenvalues and eigenvectors of Σ

$$\lambda_1 = 3.662 \qquad \lambda_2 = 0.239 \qquad \lambda_3 = 0.059$$

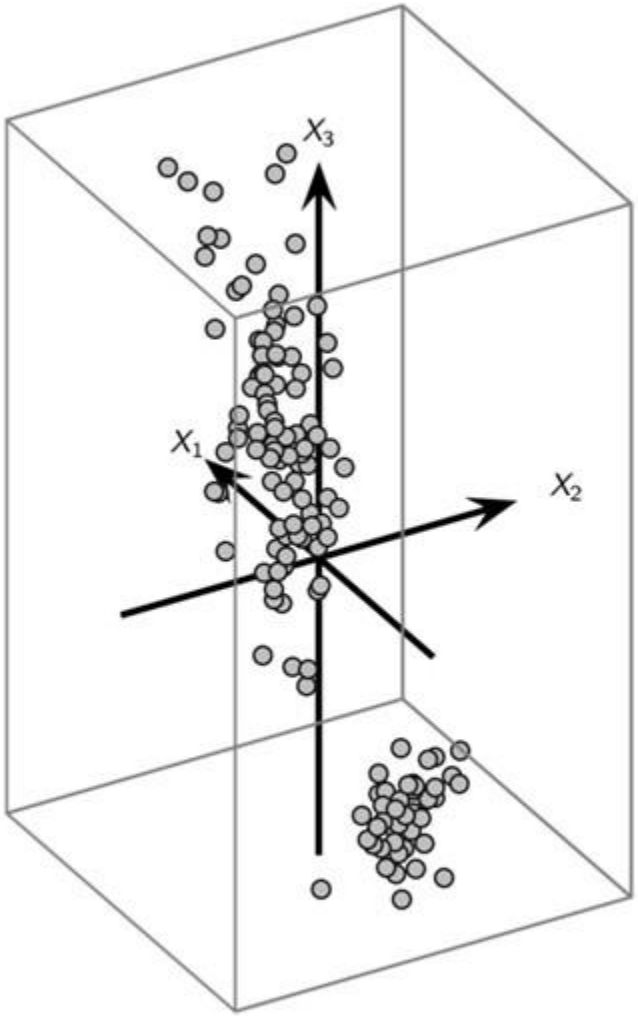
$$\boldsymbol{u}_1 = \begin{pmatrix} -0.390 \\ 0.089 \\ -0.916 \end{pmatrix} \qquad \boldsymbol{u}_2 = \begin{pmatrix} -0.639 \\ -0.742 \\ 0.200 \end{pmatrix} \qquad \boldsymbol{u}_3 = \begin{pmatrix} -0.663 \\ 0.664 \\ 0.346 \end{pmatrix}$$

The total variance is therefore $\lambda_1 + \lambda_2 + \lambda_3 = 3.662 + 0.239 + 0.059 = 3.96$.
The fraction of total variance for different values of r is given as

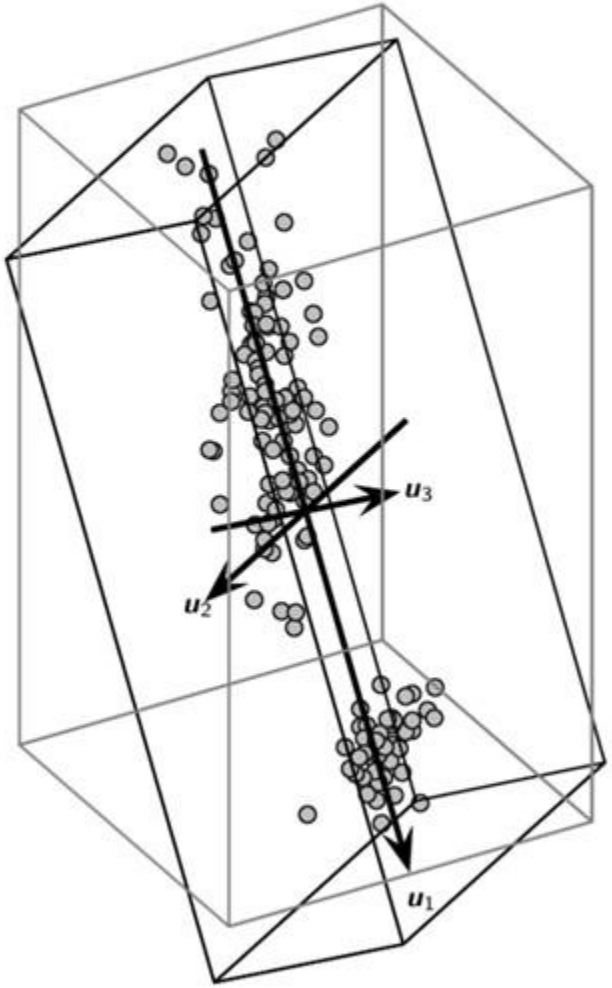
r	1	2	3
$f(r)$	0.925	0.985	1.0

This $r = 2$ PCs are need to capture $\alpha = 0.95$ fraction of variance.

Iris Data: Optimal 3D PC Basis

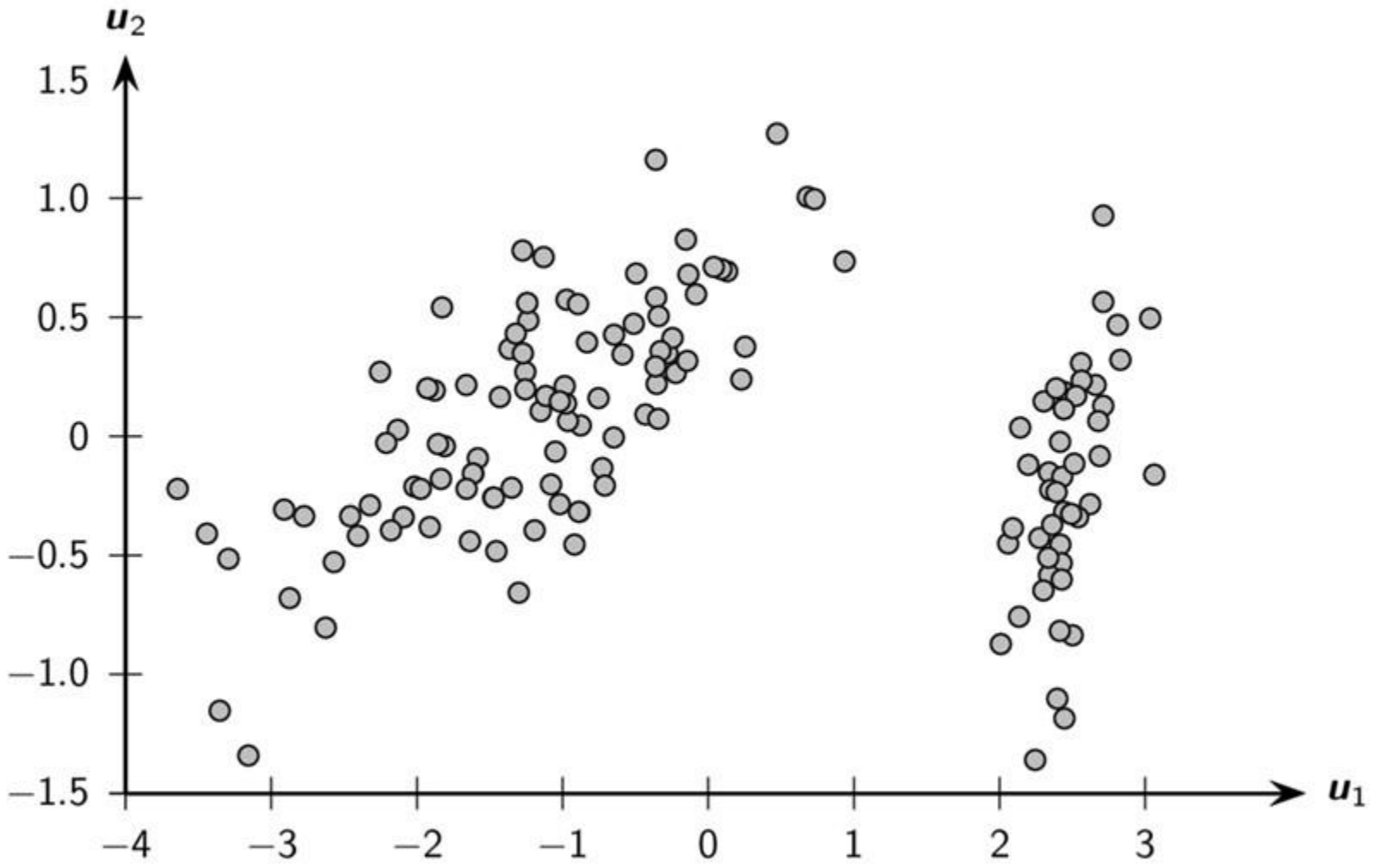


Iris Data (3D)



Optimal 3D Basis

Iris Principal Components: Projected Data (2D)



Geometry of PCA

Geometrically, when $r = d$, PCA corresponds to a orthogonal change of basis, so that the total variance is captured by the sum of the variances along each of the principal directions $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_d$, and further, all covariances are zero.

Let $\boldsymbol{U}$ be the $d \times d$ orthogonal matrix $\boldsymbol{U} = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_d \end{pmatrix}$, with $\boldsymbol{U}^{-1} = \boldsymbol{U}^T$. Let $\Lambda = diag(\lambda_1, \cdots, \lambda_d)$ be the diagonal matrix of eigenvalues. Each principal component $\boldsymbol{u}_i$ corresponds to an eigenvector of the covariance matrix Σ

$$\Sigma \boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \text{ for all } 1 \leq i \leq d$$

which can be written compactly in matrix notation:

$$\Sigma \boldsymbol{U} = \boldsymbol{U} \Lambda \text{ which implies } \Sigma = \boldsymbol{U} \Lambda \boldsymbol{U}^T$$

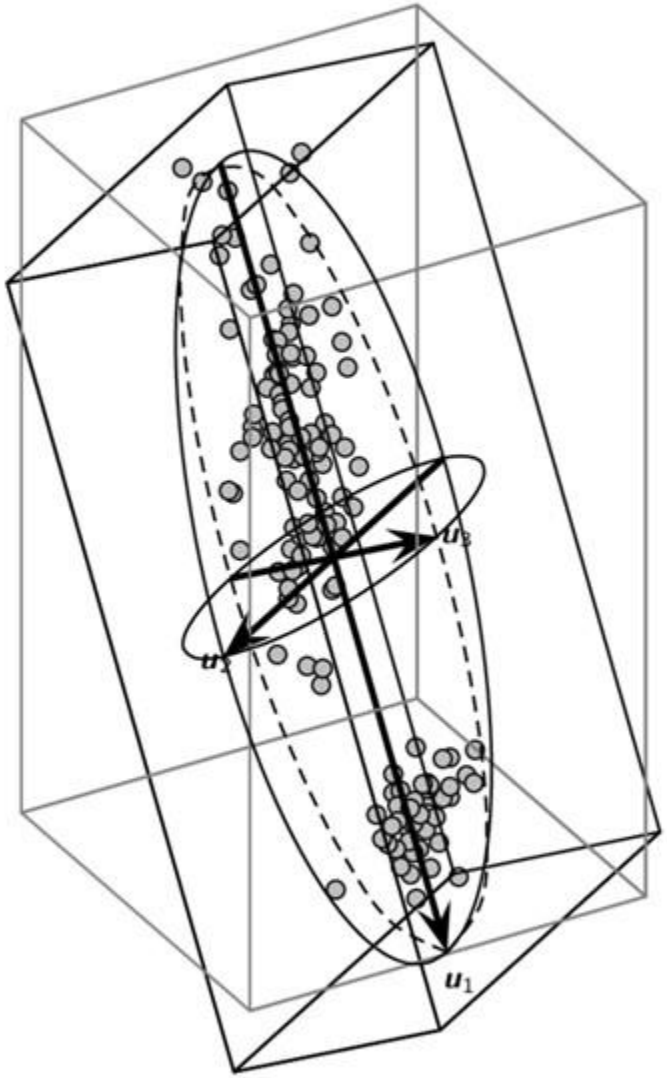
Thus, Λ represents the covariance matrix in the new PC basis.

In the new PC basis, the equation

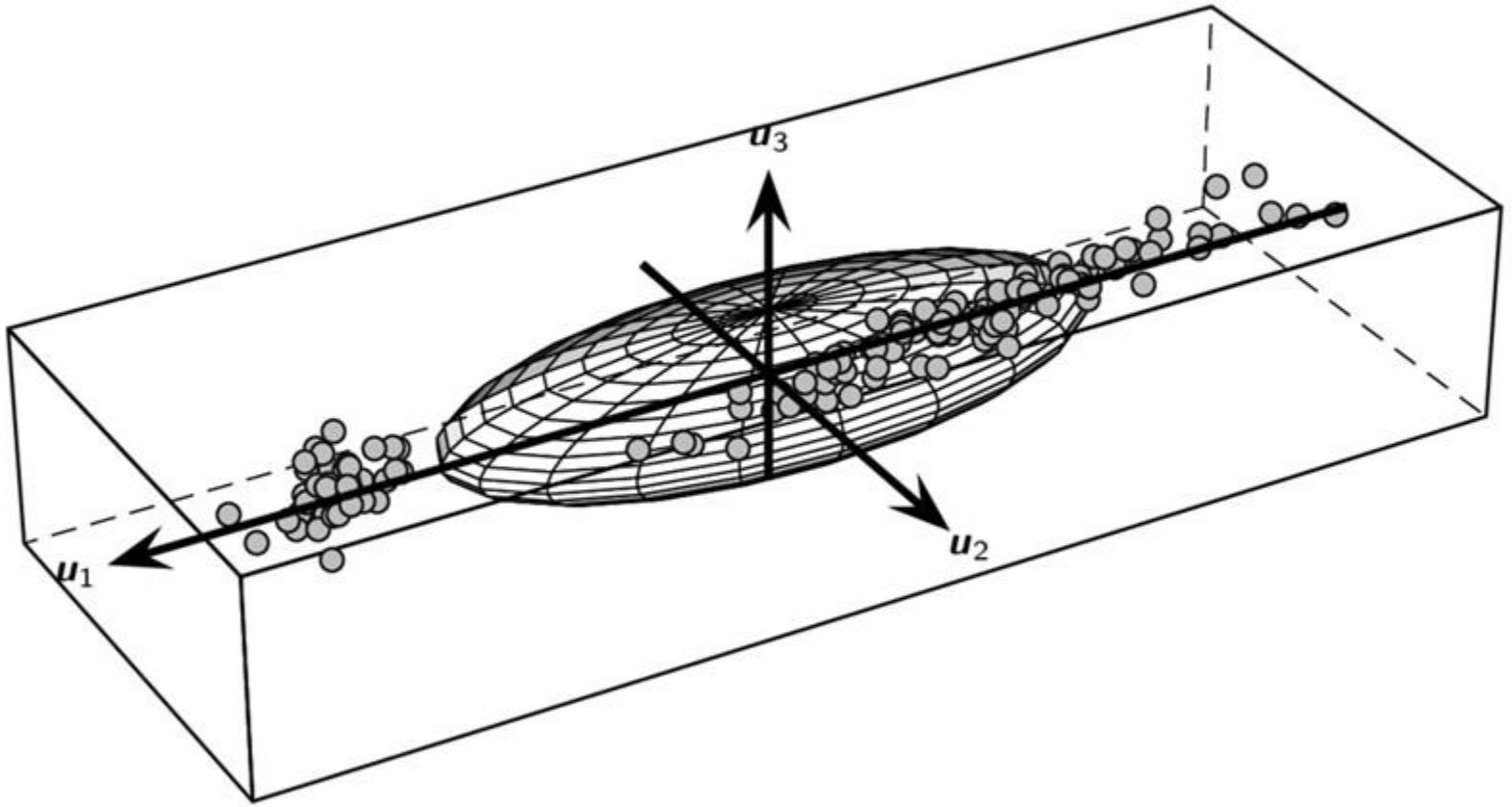
$$\boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x} = 1$$

defines a d-dimensional ellipsoid (or hyper-ellipse). The eigenvectors $\boldsymbol{u}_i$ of Σ, that is, the principal components, are the directions for the principal axes of the ellipsoid. The square roots of the eigenvalues, that is, $\sqrt{\lambda_i}$, give the lengths of the semi-axes.

Iris: Elliptic Contours in Standard Basis



Iris: Axis-Parallel Ellipsoid in PC Basis



Kernel Principal Component Analysis

Principal component analysis can be extended to find nonlinear "directions" in the data using kernel methods. Kernel PCA finds the directions of most variance in the feature space instead of the input space. Using the *kernel trick*, all PCA operations can be carried out in terms of the kernel function in input space, without having to transform the data into feature space.

Let ϕ be a function that maps a point $\boldsymbol{x}$ in input space to its image $\phi(\boldsymbol{x}_i)$ in feature space. Let the points in feature space be centered and let Σ_ϕ be the covariance matrix. The first PC in feature space correspond to the dominant eigenvector

$$\Sigma_\phi \boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1$$

where

$$\Sigma_\phi = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) \phi(\boldsymbol{x}_i)^T$$

Kernel Principal Component Analysis

It can be shown that $\boldsymbol{u}_1 = \sum_{i=1}^{n} c_i \phi(\boldsymbol{x}_i)$. That is, the PC direction in feature space is a linear combination of the transformed points.

The coefficients are captured in the weight vector

$$\boldsymbol{c} = \left(c_1, c_2, \cdots, c_n\right)^T$$

Substituting into the eigen-decomposition of Σ_ϕ and simplifying, we get:

$$\boldsymbol{K}\boldsymbol{c} = n\lambda_1 \boldsymbol{c} = \eta_1 \boldsymbol{c}$$

Thus, the weight vector $\boldsymbol{c}$ is the eigenvector corresponding to the largest eigenvalue η_1 of the kernel matrix $\boldsymbol{K}$.

Kernel Principal Component Analysis

The weight vector $\boldsymbol{c}$ can be used to then find $\boldsymbol{u}_1$ via $\boldsymbol{u}_1 = \sum_{i=1}^{n} c_i \phi(\boldsymbol{x}_i)$.

The only constraint we impose is that $\boldsymbol{u}_1$ should be normalized to be a unit vector, which implies $\|\boldsymbol{c}\|^2 = \frac{1}{\eta_1}$.

We cannot compute directly the principal direction, but we can project any point $\phi(\boldsymbol{x})$ onto the principal direction $\boldsymbol{u}_1$, as follows:

$$\boldsymbol{u}_1^T \phi(\boldsymbol{x}) = \sum_{i=1}^{n} c_i \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) = \sum_{i=1}^{n} c_i K(\boldsymbol{x}_i, \boldsymbol{x})$$

which requires only kernel operations.

We can obtain the additional principal components by solving for the other eigenvalues and eigenvectors of

$$\boldsymbol{K}\boldsymbol{c}_j = n\lambda_j \boldsymbol{c}_j = \eta_j \boldsymbol{c}_j$$

If we sort the eigenvalues of $\boldsymbol{K}$ in decreasing order $\eta_1 \geq \eta_2 \geq \cdots \geq \eta_n \geq 0$, we can obtain the jth principal component as the corresponding eigenvector $\boldsymbol{c}_j$. The variance along the jth principal component is given as $\lambda_j = \frac{\eta_j}{n}$.

Kernel PCA Algorithm

KernelPCA $(\boldsymbol{D}, K, \alpha)$**:**

1 $\boldsymbol{K} = \left\{K(\boldsymbol{x}_i, \boldsymbol{x}_j)\right\}_{i,j=1,\ldots,n}$ // compute $n \times n$ kernel matrix

2 $\boldsymbol{K} = (\boldsymbol{I} - \frac{1}{n}1_{n\times n})\boldsymbol{K}(\boldsymbol{I} - \frac{1}{n}1_{n\times n})$ // center the kernel matrix

3 $(\eta_1, \eta_2, \ldots, \eta_d) = \text{eigenvalues}(\boldsymbol{K})$ // compute eigenvalues

4 $\begin{pmatrix}\boldsymbol{c}_1 & \boldsymbol{c}_2 & \cdots & \boldsymbol{c}_n\end{pmatrix} = \text{eigenvectors}(\boldsymbol{K})$ // compute eigenvectors

5 $\lambda_i = \frac{\eta_i}{n}$ for all $i = 1, \ldots, n$ // compute variance for each component

6 $\boldsymbol{c}_i = \sqrt{\frac{1}{\eta_i}} \cdot \boldsymbol{c}_i$ for all $i = 1, \ldots, n$ // ensure that $\boldsymbol{u}_i^T \boldsymbol{u}_i = 1$

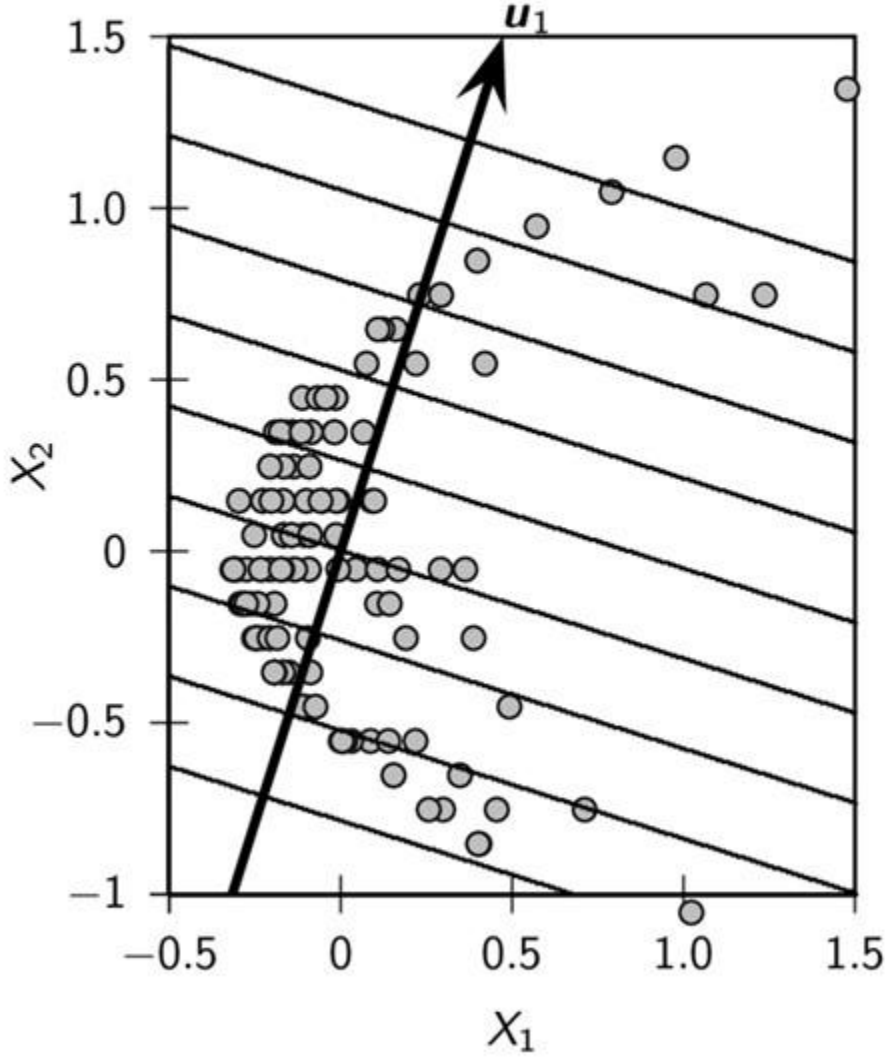
7 $f(r) = \frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$, for all $r = 1, 2, \ldots, d$ // fraction of total variance

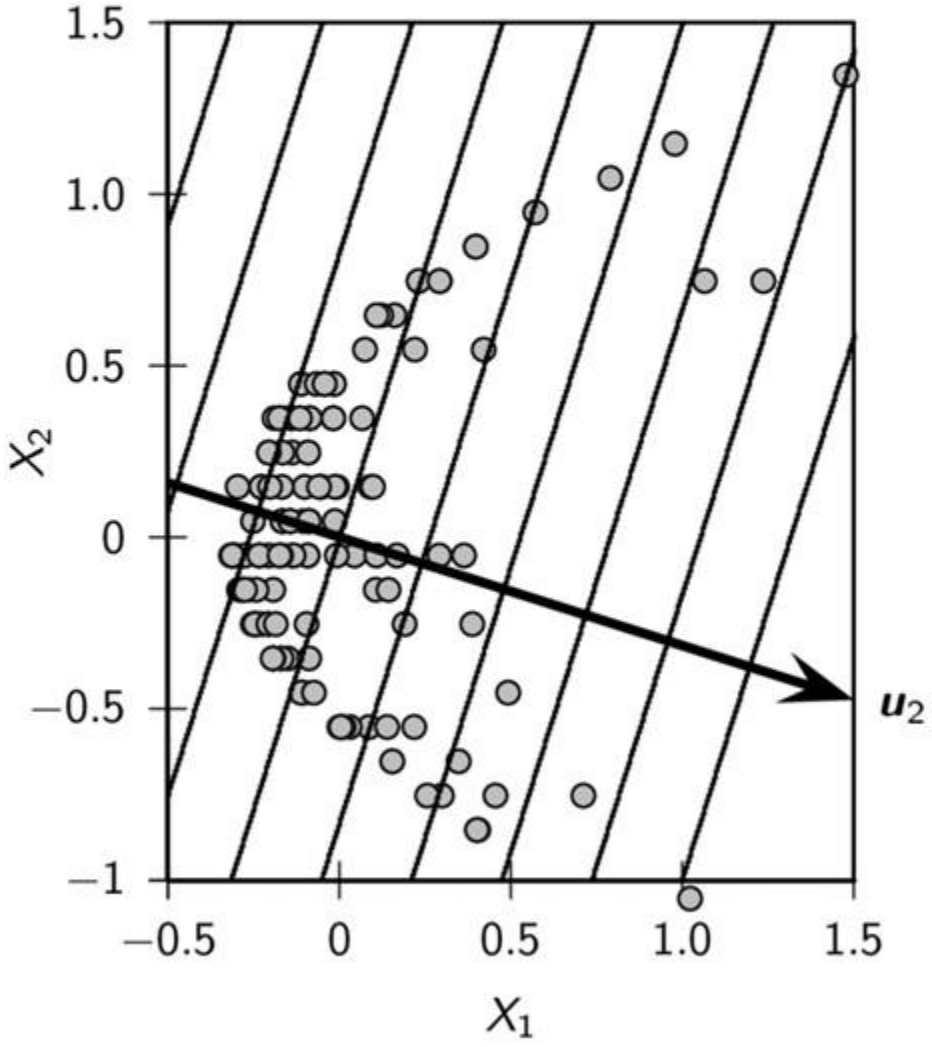
8 Choose smallest r so that $f(r) \geq \alpha$ // choose dimensionality

9 $\boldsymbol{C}_r = \begin{pmatrix}\boldsymbol{c}_1 & \boldsymbol{c}_2 & \cdots & \boldsymbol{c}_r\end{pmatrix}$ // reduced basis

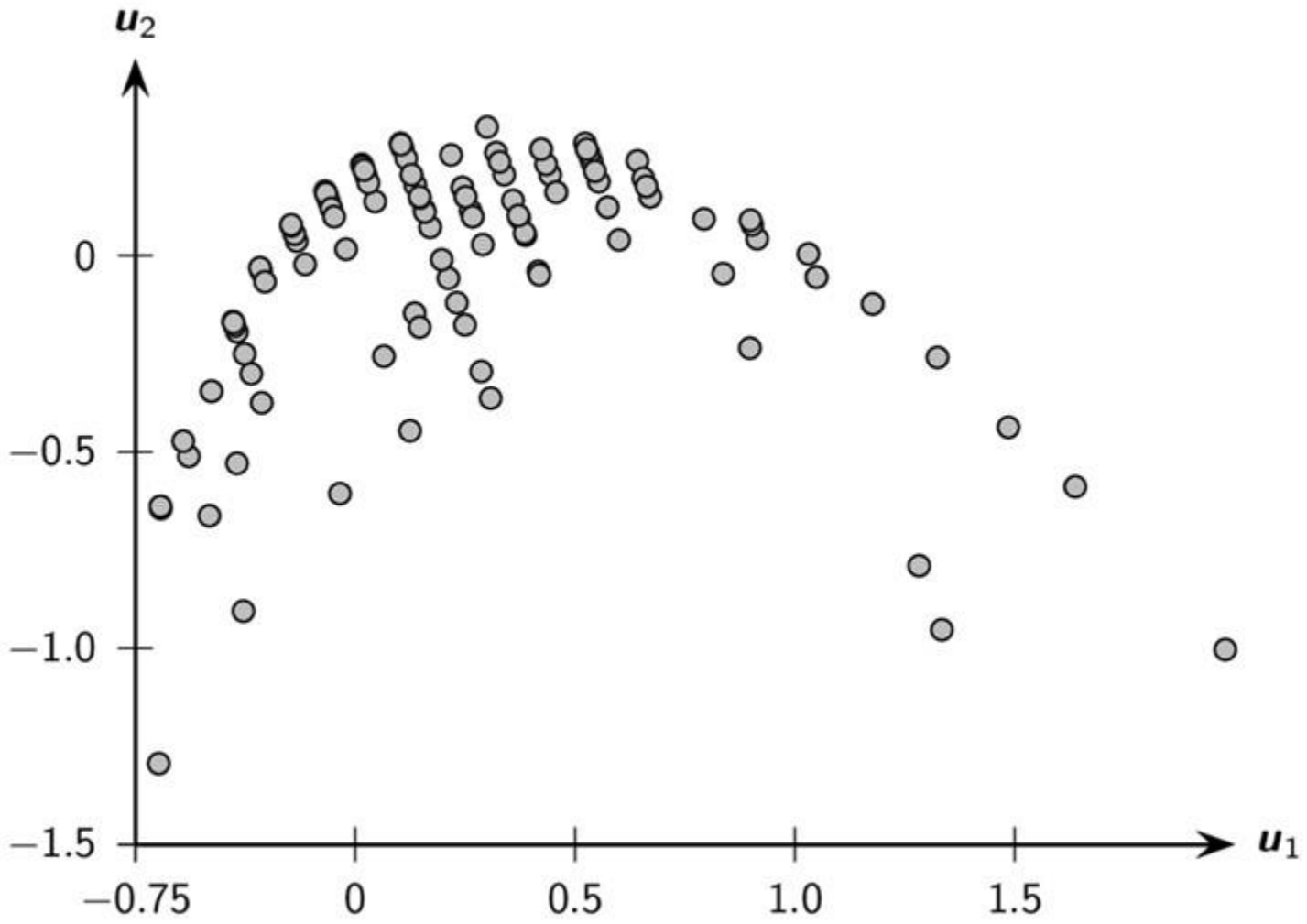
10 $\boldsymbol{A} = \{\boldsymbol{a}_i \mid \boldsymbol{a}_i = \boldsymbol{C}_r^T \boldsymbol{K}_i, \text{for } i = 1, \ldots, n\}$ // reduced dimensionality data

Nonlinear Iris Data: PCA in Input Space



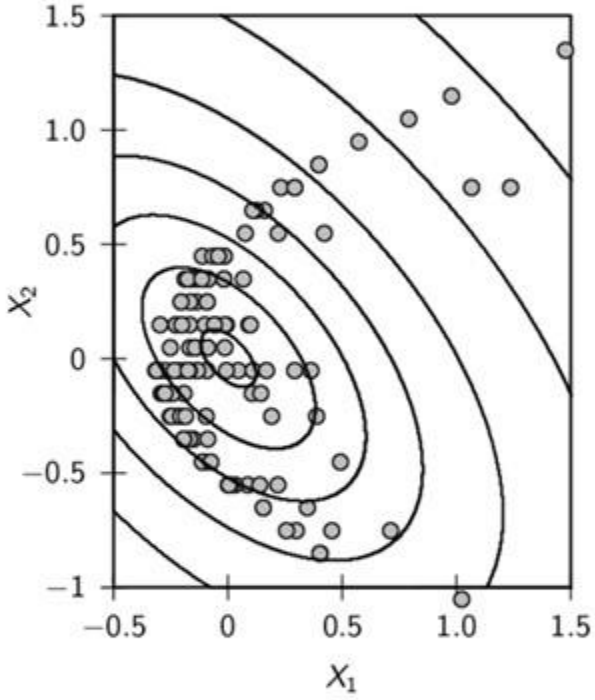


Nonlinear Iris Data: Projection onto PCs

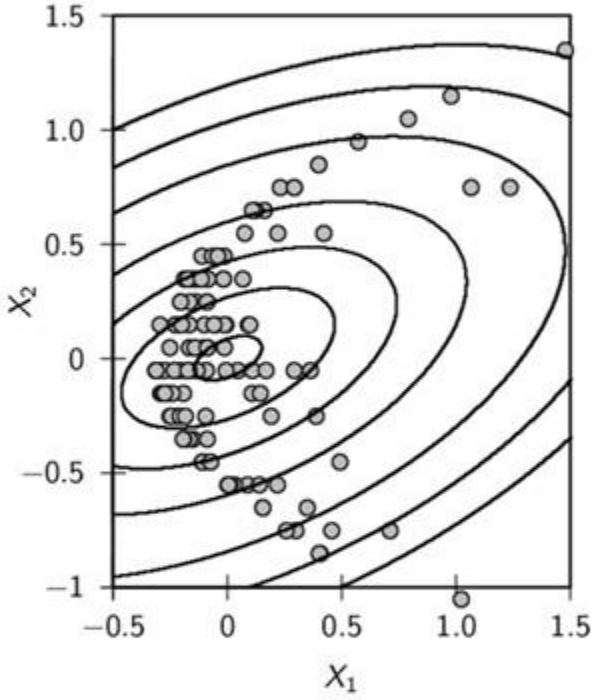


Kernel PCA: 3 PCs (Contours of Constant Projection)

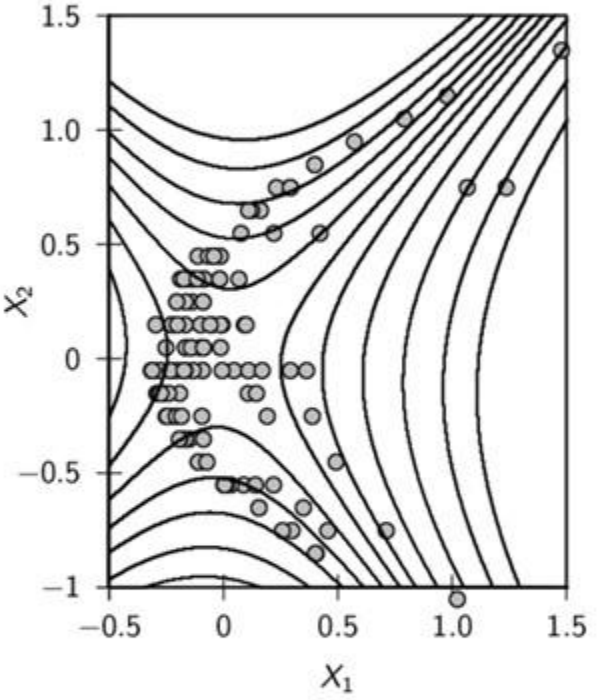
Homogeneous Quadratic Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j)^2$



(a) $\lambda_1 = 0.2067$



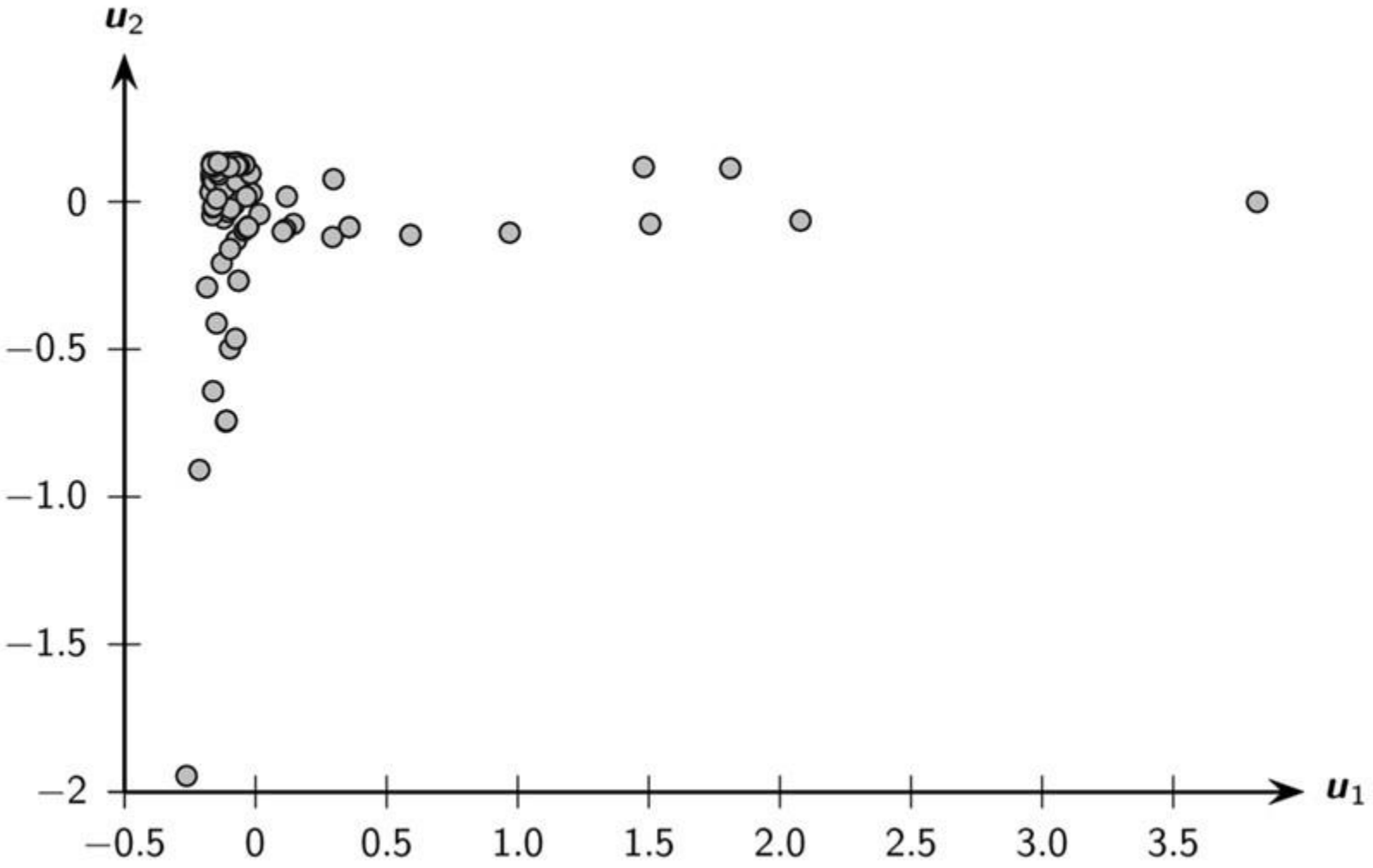
(b) $\lambda_2 = 0.0596$



(c) $\lambda_3 = 0.0184$

Kernel PCA: Projected Points onto 2 PCs

Homogeneous Quadratic Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j)^2$



Singular Value Decomposition

Principal components analysis is a special case of a more general matrix decomposition method called *Singular Value Decomposition (SVD)*. PCA yields the following decomposition of the covariance matrix:

$$\Sigma = \boldsymbol{U} \Lambda \boldsymbol{U}^T$$

where the covariance matrix has been factorized into the orthogonal matrix $\boldsymbol{U}$ containing its eigenvectors, and a diagonal matrix Λ containing its eigenvalues (sorted in decreasing order).

SVD generalizes the above factorization for any matrix. In particular for an $n \times d$ data matrix $\boldsymbol{D}$ with n points and d columns, SVD factorizes $\boldsymbol{D}$ as follows:

$$\boldsymbol{D} = \boldsymbol{L} \Delta \boldsymbol{R}^T$$

where $\boldsymbol{L}$ is a orthogonal $n \times n$ matrix, $\boldsymbol{R}$ is an orthogonal $d \times d$ matrix, and Δ is an $n \times d$ "diagonal" matrix, defined as $\Delta(i, i) = \delta_i$, and 0 otherwise. The columns of $\boldsymbol{L}$ are called the *left singular and the columns of* $\boldsymbol{R}$ *(or rows of* $\boldsymbol{R}^T$*) are called the* right singular vectors. The entries δ_i are called the *singular values* of $\boldsymbol{D}$, and they are all non-negative.

Reduced SVD

If the rank of $\boldsymbol{D}$ is $r \leq \min(n, d)$, then there are only r nonzero singular values, ordered as follows: $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_r > 0$.

We discard the left and right singular vectors that correspond to zero singular values, to obtain the *reduced SVD* as

$$\boldsymbol{D} = \boldsymbol{L}_r \Delta_r \boldsymbol{R}_r^T$$

where $\boldsymbol{L}_r$ is the $n \times r$ matrix of the left singular vectors, $\boldsymbol{R}_r$ is the $d \times r$ matrix of the right singular vectors, and Δ_r is the $r \times r$ diagonal matrix containing the positive singular vectors.

The reduced SVD leads directly to the *spectral decomposition* of $\boldsymbol{D}$ given as

$$\boldsymbol{D} = \sum_{i=1}^{r} \delta_i \boldsymbol{l}_i \boldsymbol{r}_i^T$$

The best rank q approximation to the original data $\boldsymbol{D}$ is the matrix $\boldsymbol{D}_q = \sum_{i=1}^{q} \delta_i \boldsymbol{l}_i \boldsymbol{r}_i^T$ that minimizes the expression $\|\boldsymbol{D} - \boldsymbol{D}_q\|_F$, where $\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{d} \boldsymbol{A}(i,j)^2}$ is called the *Frobenius Norm* of $\boldsymbol{A}$.

Connection Between SVD and PCA

Assume $\boldsymbol{D}$ has been centered, and let $\boldsymbol{D} = \boldsymbol{L}\Delta\boldsymbol{R}^T$ via SVD. Consider the *scatter matrix* for $\boldsymbol{D}$, given as $\boldsymbol{D}^T\boldsymbol{D}$. We have

$$\boldsymbol{D}^T\boldsymbol{D} = \left(\boldsymbol{L}\Delta\boldsymbol{R}^T\right)^T\left(\boldsymbol{L}\Delta\boldsymbol{R}^T\right) = \boldsymbol{R}\Delta^T\boldsymbol{L}^T\boldsymbol{L}\Delta\boldsymbol{R}^T = \boldsymbol{R}(\Delta^T\Delta)\boldsymbol{R}^T = \boldsymbol{R}\Delta_d^2\boldsymbol{R}^T$$

where Δ_d^2 is the $d \times d$ diagonal matrix defined as $\Delta_d^2(i,i) = \delta_i^2$, for $i = 1, \ldots, d$.

The covariance matrix of centered $\boldsymbol{D}$ is given as $\Sigma = \frac{1}{n}\boldsymbol{D}^T\boldsymbol{D}$; we get

$$\begin{aligned}\boldsymbol{D}^T\boldsymbol{D} &= n\Sigma \\ &= n\boldsymbol{U}\Lambda\boldsymbol{U}^T \\ &= \boldsymbol{U}(n\Lambda)\boldsymbol{U}^T\end{aligned}$$

The right singular vectors $\boldsymbol{R}$ are the same as the eigenvectors of Σ. The singular values of $\boldsymbol{D}$ are related to the eigenvalues of Σ as

$$n\lambda_i = \delta_i^2, \text{ which implies } \lambda_i = \frac{\delta_i^2}{n}, \text{ for } i = 1, \ldots, d$$

Likewise the left singular vectors in $\boldsymbol{L}$ are the eigenvectors of the matrix $n \times n$ matrix $\boldsymbol{D}\boldsymbol{D}^T$, and the corresponding eigenvalues are given as δ_i^2.

Q&A