Machine Learning (Semester 1 2024)

Dimensionality Reduction & Unsupervised Learning

Nina Hernitschek
Centro de Astronomía CITEVA
Universidad de Antofagasta

July 23, 2024

Motivation

So far, we have seen mostly **supervised machine learning**.

Today in our final lecture, we will learn about **Unsupervised Learning**, which will complete the machine learning course. Unsupervised learning consists techniques such as dimensionality reduction and density estimation.

Those methods have in common that there is no associated response variable, but rather we want to find out interesting properties of the observations themselves.

Unsupervised Learning

So far, we have seen supervised learning methods such as regression and classification.

In supervised learning, we have access to a set of p features $X_1, X_2, ..., X_p$, measured on n observations, and a response Y also measured on those same n observations.

The goal is then to predict Y using $X_1, X_2, ..., X_p$.

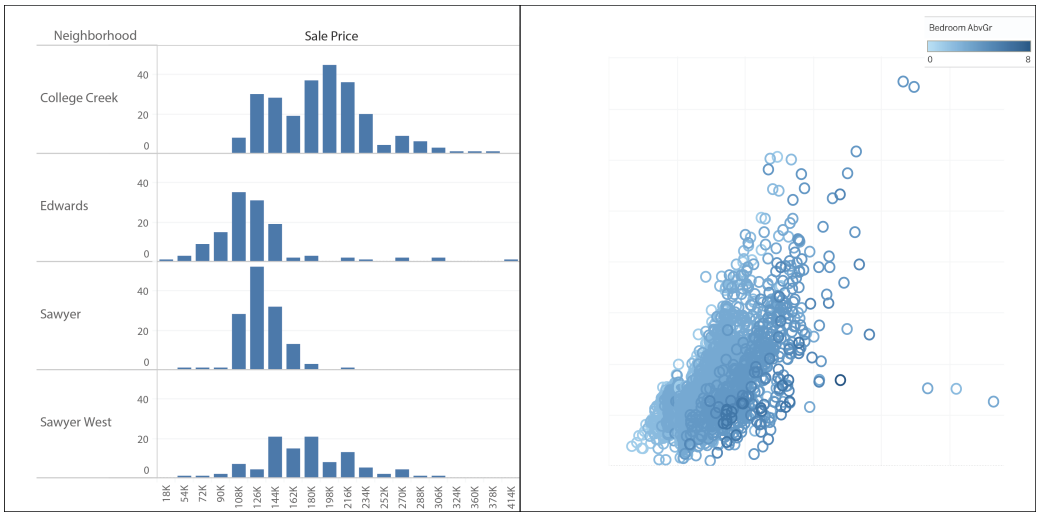
Now, instead, we will focus on unsupervised learning, a set of statistical tools intended for problems in which we have only a set of features $X_1, X_2, ..., X_p$ measured on n observations. We are not interested in prediction, because we do not have an associated response variable Y.

Unsupervised Learning

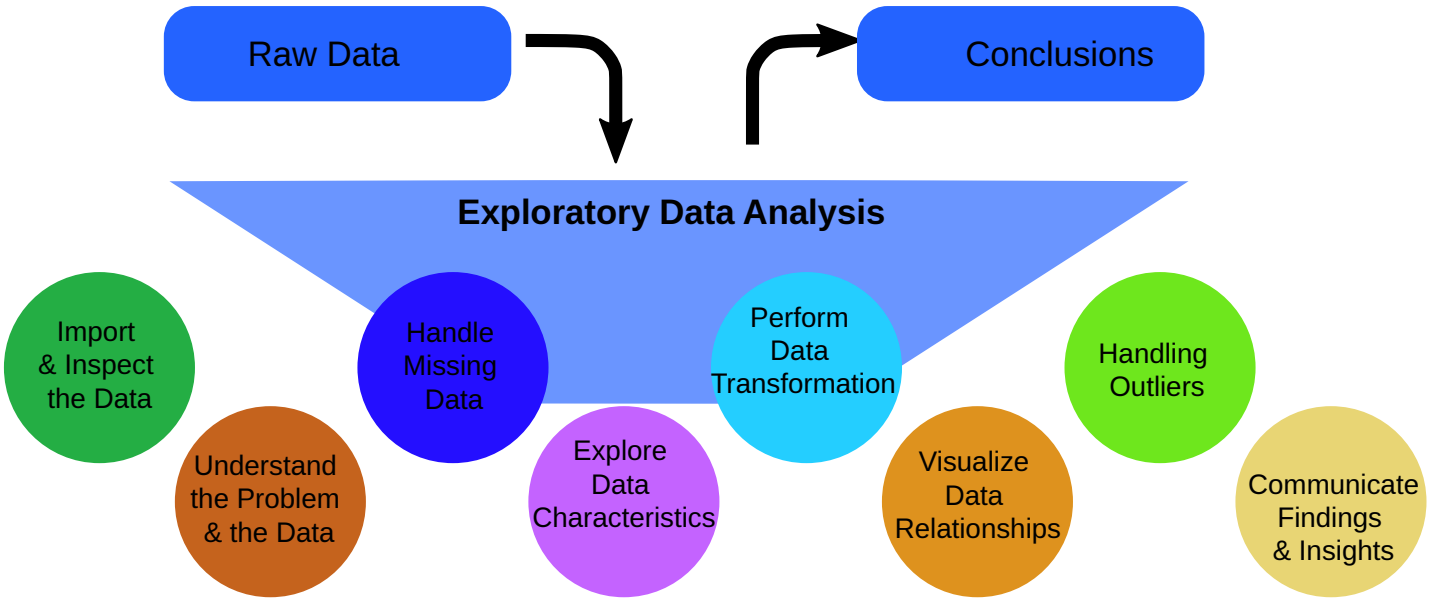
Goals of unsupervised learning are typically:

- Can we discover subgroups among the variables or among the observations?
- Is there an informative way to visualize the data?

Unsupervised learning is often performed as part of an **exploratory data analysis**.



Unsupervised Learning



Unsupervised Learning

In contrast to supervised learning, unsupervised learning is often much **more challenging**:

There is no simple goal for the analysis, such as prediction of a response.

In addition, it can be hard to assess the results obtained from unsupervised learning methods, since there is no universally accepted mechanism for performing cross- validation or validating results on an independent data set: we don't know the "true answer".

Unsupervised Learning

Applications in astronomy of unsupervised learning are such as:

- explorative analysis of (very) large data sets
- anomaly detection
- similarity search
- preprocessing as part of a machine-learning pipeline (that also contains supervised learning)

Unsupervised Learning

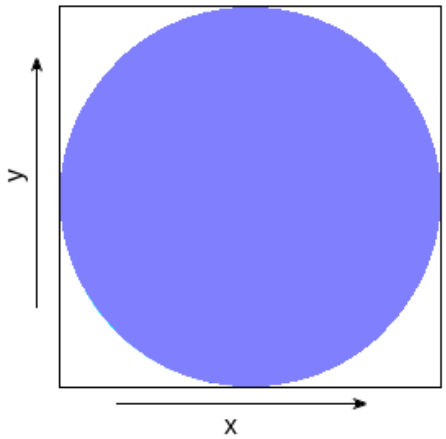
Applications in astronomy of unsupervised learning are such as:

- explorative analysis of (very) large data sets
- anomaly detection
- similarity search
- preprocessing as part of a machine-learning pipeline (that also contains supervised learning)

A review of unsupervised learning in astronomy is given in Sotiria Fotopoulou (2024).

Motivation: The *Curse of Dimensionality*

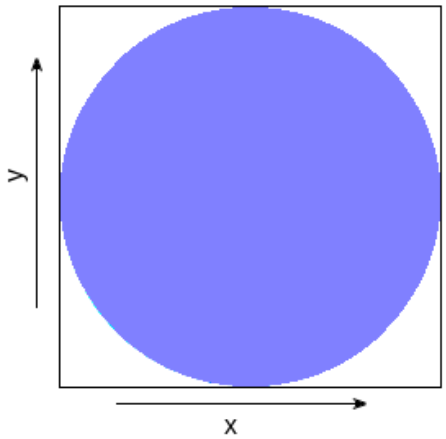
We inscribe a circle with radius r in a square.



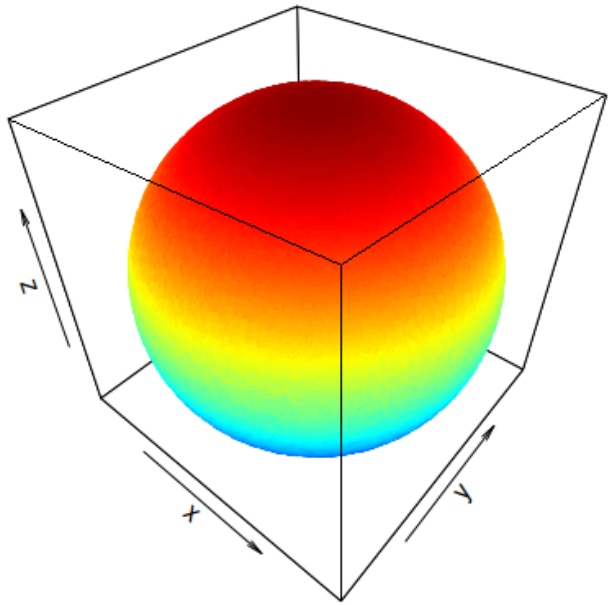
The fraction of area is: $\frac{V_{circle}}{V_{square}} = \frac{\pi r^2}{(2r)^2}$

Motivation: The *Curse of Dimensionality*

We inscribe a sphere with radius r in a cube.



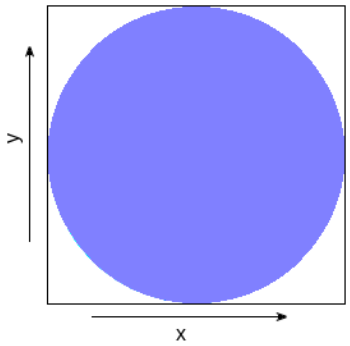
$$\frac{V_{circle}}{V_{square}} = \frac{\pi r^2}{(2r)^2}$$



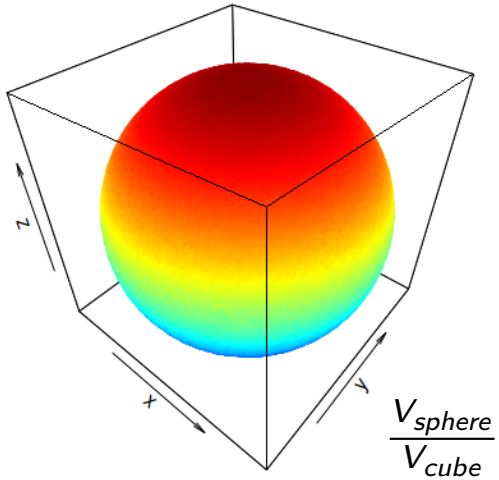
The fraction of volume is: $\frac{V_{sphere}}{V_{cube}} = \frac{4/3\pi r^3}{(2r)^3}$

Motivation: The *Curse of Dimensionality*

We **generalize** this:



$$\frac{V_{circle}}{V_{square}} = \frac{\pi r^2}{(2r)^2}$$



$$\frac{V_{sphere}}{V_{cube}} = \frac{4/3\pi r^3}{(2r)^3}$$

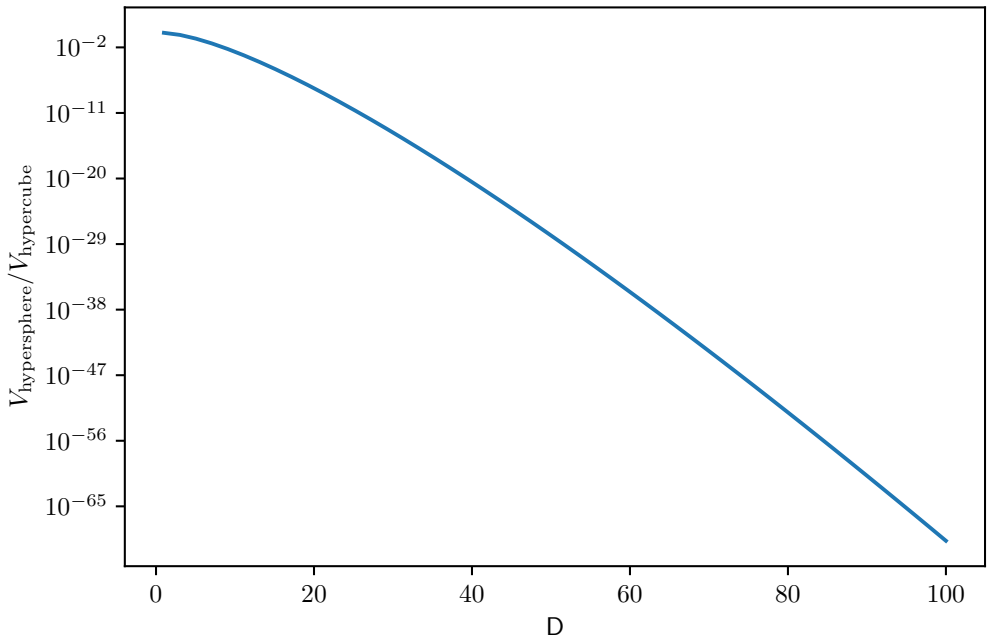
For D dimensions, the volume of hypersphere (with radius r) becomes $V_{hypersphere} = \frac{2r^D \pi^{D/2}}{D\,\Gamma(D/2)}$, with Γ being the Gamma function.

The volume of a hypercube becomes $V_{hypercube} = (2r)^D$. Thus the fraction becomes

$$\frac{V_{hypersphere}}{V_{hypercube}} = \frac{\pi^{D/2}}{D2^{D-1}\Gamma(d/2)}$$

Motivation: The *Curse of Dimensionality*

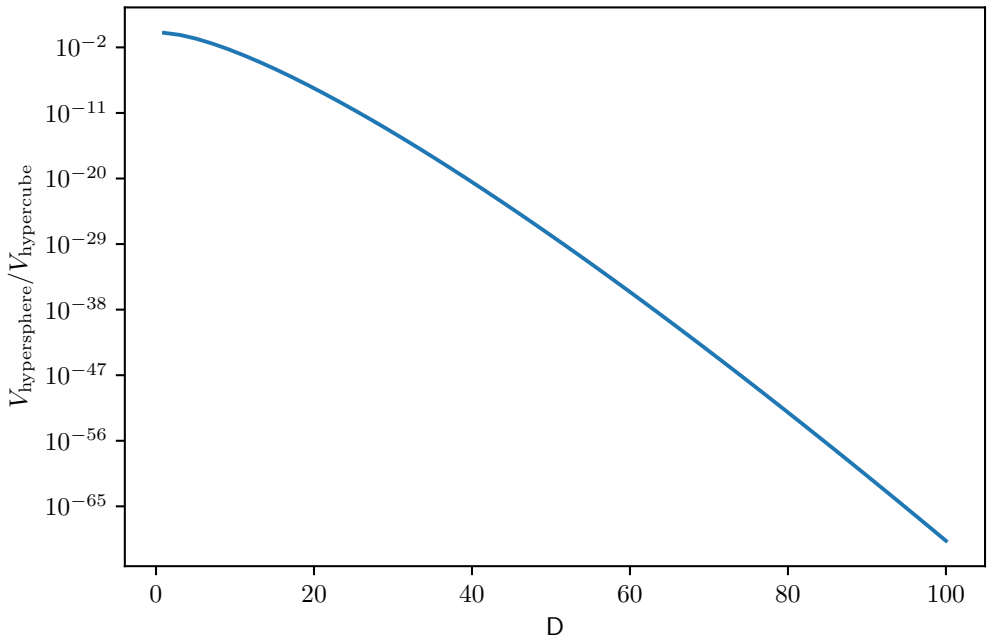
from numerical calculation:



$\frac{V_{hypersphere}}{V_{hypercube}}$ goes to 0 as D goes to infinity

Motivation: The *Curse of Dimensionality*

from numerical calculation:



$\frac{V_{hypersphere}}{V_{hypercube}}$ goes to 0 as D goes to infinity

The area outside of the circle (sphere, hypersphere...) grows larger and larger as the number of dimensions increases.

Motivation: The *Curse of Dimensionality*

Mathematically we can describe this as: the more **dimensions** that your data span, the **more points needed to uniformly sample the space**.

Motivation: The *Curse of Dimensionality*

Mathematically we can describe this as: the more **dimensions** that your data span, the **more points needed to uniformly sample the space**.

The **curse of dimensionality** refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces that do not occur in low-dimensional settings. The expression was coined by Richard E. Bellman when considering problems in dynamic programming.

Dimensionally cursed phenomena occur in domains such as numerical analysis, sampling, combinatorics, machine learning, data mining and databases.

Motivation: The *Curse of Dimensionality*

another ***example***:

The **Hughes Phenomenon** (also called the **peaking phenomenon**) states that for fixed number of training samples, the average (expected) predictive power of a classifier or regressor first increases as the number of dimensions or features used is increased but beyond a certain dimensionality it starts deteriorating instead of improving steadily.

Motivation: The *Curse of Dimensionality*

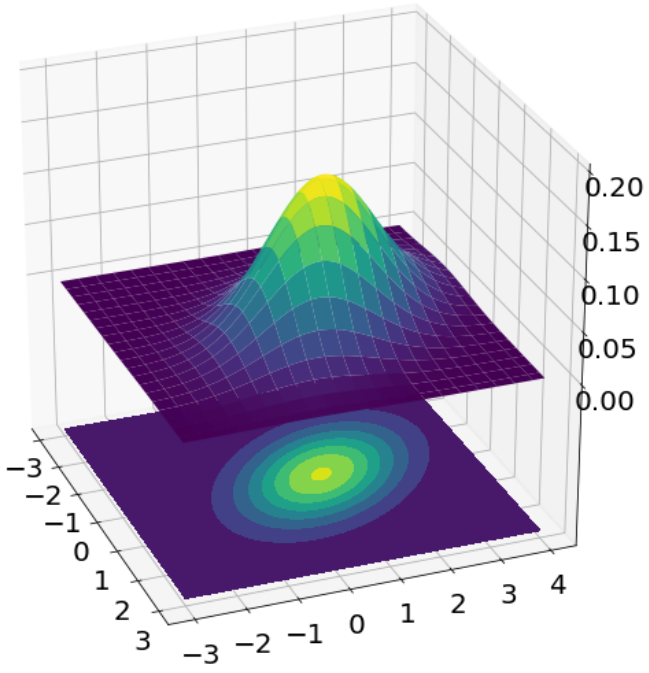
another ***example***:

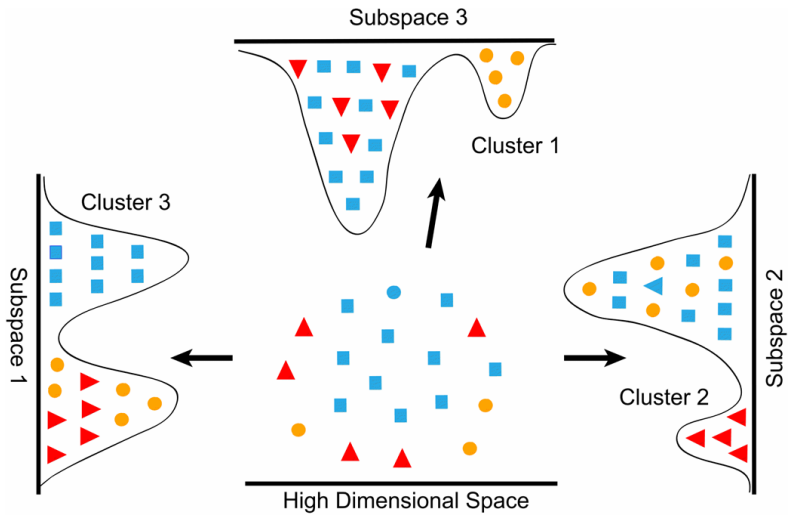
The **Hughes Phenomenon** (also called the **peaking phenomenon**) states that for fixed number of training samples, the average (expected) predictive power of a classifier or regressor first increases as the number of dimensions or features used is increased but beyond a certain dimensionality it starts deteriorating instead of improving steadily.

Issues that arise with high dimensional data are:

- Running a risk of overfitting the machine learning model.
- Difficulty in clustering similar features.
- Increased space and computational time complexity.

Dimensionality Reduction



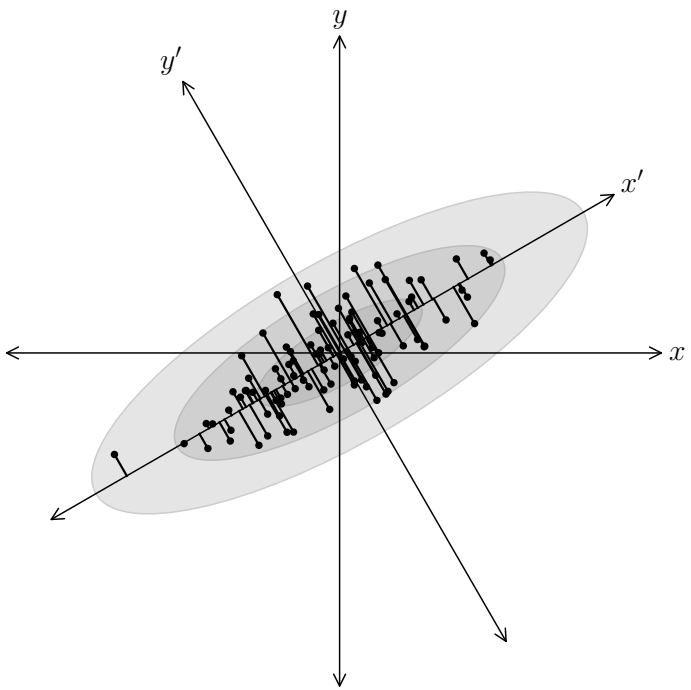


Principal Component Analysis

Principal Component Analysis (PCA) is an unsupervised method for **dimensionality reduction**. A transformation is applied to the data such that the new coordinate axes are aligned with the maximal variance of the data.

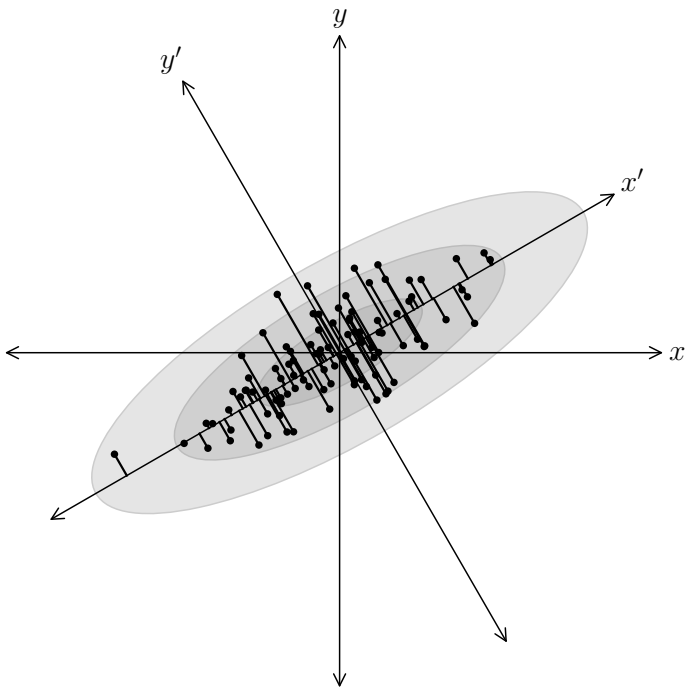
Principal Component Analysis

Principal Component Analysis (PCA) is an unsupervised method for **dimensionality reduction**. A transformation is applied to the data such that the new coordinate axes are aligned with the maximal variance of the data.



Technically, it involves the process of finding the principal components, i.e. the decomposition of the feature matrix into eigenvectors.
PCA transformations are **linear transformations**, thus it will not be

Principal Component Analysis



The rotation is chosen to maximize the ability to discriminate between the data points:

- **principal component** is the direction of maximal variance
- second principal component is orthogonal to the first component and maximizes the residual variance

Principal Component Analysis

We can define the whole process of PCA into just four steps:

1. Standardization: The data has to be transformed to a common scale by taking the difference between the original dataset with the mean of the whole dataset. This will make the distribution 0 centered.

Principal Component Analysis

We can define the whole process of PCA into just four steps:

1. Standardization: The data has to be transformed to a common scale by taking the difference between the original dataset with the mean of the whole dataset. This will make the distribution 0 centered.

2. Finding covariance: Covariance helps to understand the relationship between the mean and original data.

Principal Component Analysis

We can define the whole process of PCA into just four steps:

1. Standardization: The data has to be transformed to a common scale by taking the difference between the original dataset with the mean of the whole dataset. This will make the distribution 0 centered.

2. Finding covariance: Covariance helps to understand the relationship between the mean and original data.

3. Determining the principal components: Principal components can be determined by calculating the eigenvectors and eigenvalues. Eigenvectors are a special set of vectors that help us to understand the structure and the property of the data that would be principal components. The eigenvalues on the other hand help us to determine the principal components. The highest eigenvalues and their corresponding eigenvectors make the most important principal components.

Principal Component Analysis

We can define the whole process of PCA into just four steps:

1. Standardization: The data has to be transformed to a common scale by taking the difference between the original dataset with the mean of the whole dataset. This will make the distribution 0 centered.

2. Finding covariance: Covariance helps to understand the relationship between the mean and original data.

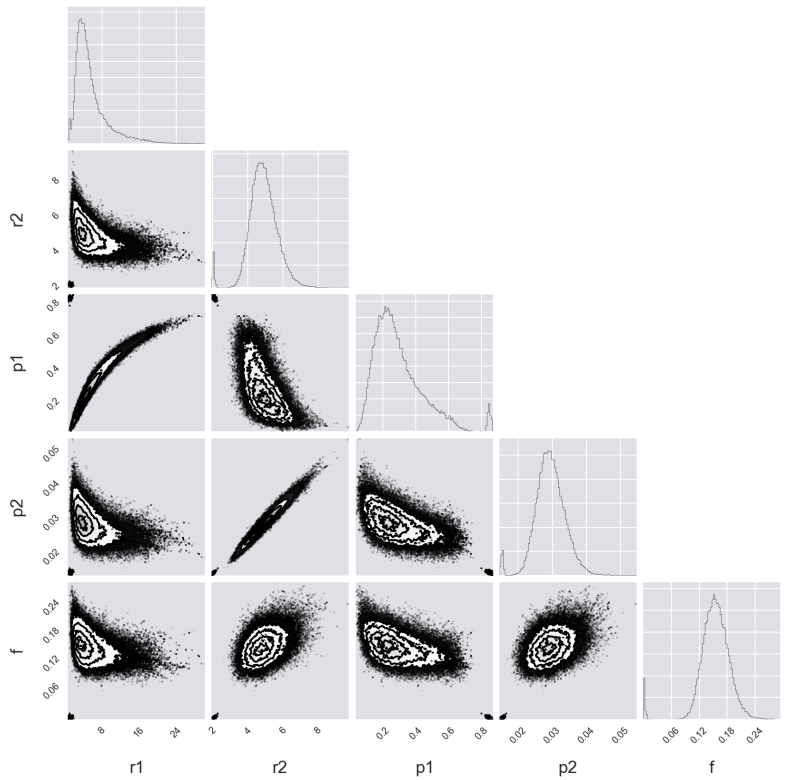
3. Determining the principal components: Principal components can be determined by calculating the eigenvectors and eigenvalues. Eigenvectors are a special set of vectors that help us to understand the structure and the property of the data that would be principal components. The eigenvalues on the other hand help us to determine the principal components. The highest eigenvalues and their corresponding eigenvectors make the most important principal components.

4. Final output: It is the dot product of the standardized matrix and the eigenvector. Note that the number of columns or features will be changed.

Principal Component Analysis

We will now see how we implement this approach **mathematically**.

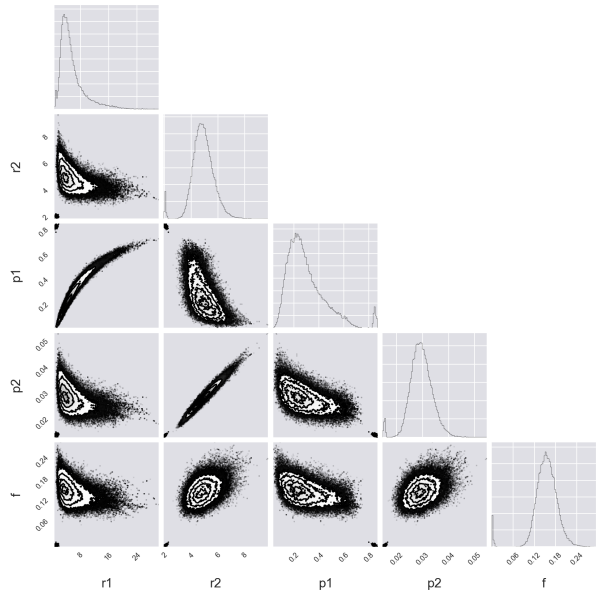
Assume we want to visualize n observations with measurements on a set of p features, $X_1, X_2, ..., X_p$, as part of an exploratory data analysis.
We could do this by examining two-dimensional scatterplots, each of which contains the n observations' measurements on two of the features.



Principal Component Analysis

We will now see how we implement this approach **mathematically**.

Assume we want to visualize n observations with measurements on a set of p features, $X_1, X_2, ..., X_p$, as part of an exploratory data analysis.
We could do this by examining two-dimensional scatterplots, each of which contains the n observations' measurements on two of the features.



However, there are $\binom{p}{2} = p(p-1)/2$ such scatterplots.
Example: for $p = 10$ there are 45 plots.

Principal Component Analysis

Especially for large p, a better method is required to visualize the observations.
For doing so, we have to find a **low-dimensional representation** of the data that captures as much of the information as possible.

The idea behind PCA is that each of the n observations lives in a p-dimensional space, but not all of these dimensions are equally interesting. PCA seeks a small number of dimensions that are as interesting as possible, where the concept of interesting is measured by the amount that the observations vary along each dimension.

Principal Component Analysis

Especially for large p, a better method is required to visualize the observations.
For doing so, we have to find a **low-dimensional representation** of the data that captures as much of the information as possible.

The idea behind PCA is that each of the n observations lives in a p-dimensional space, but not all of these dimensions are equally interesting. PCA seeks a small number of dimensions that are as interesting as possible, where the concept of interesting is measured by the amount that the observations vary along each dimension.

Each of the dimensions found by PCA is a **linear combination** of the p features.

Principal Component Analysis

The first **principal component** of a set of features $X_1, X_2, ..., X_p$ is the normalized linear combination of the features

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + ... + \phi_{p1} X_p$$

that has the largest variance. (Normalized: $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.)

We refer to the elements $\phi_{11}, ..., \pi_{p1}$ as the **loadings of the first principal component**; together, the loadings make up the principal component loading vector, $\phi_1 = (\phi_{11}\phi_{21}..., \pi_{p1})^T$.

Principal Component Analysis

The first **principal component** of a set of features $X_1, X_2, ..., X_p$ is the normalized linear combination of the features

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + ... + \phi_{p1} X_p$$

that has the largest variance. (Normalized: $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.)

We refer to the elements $\phi_{11}, ..., \pi_{p1}$ as the **loadings of the first principal component**; together, the loadings make up the principal component loading vector, $\phi_1 = (\phi_{11} \phi_{21} ..., \pi_{p1})^T$.

Given an $n \times p$ data set $\mathbf{X}$, how do we compute the first principal component?

Principal Component Analysis

As we are only interested in variance, we assume that each of the variables in X has been centered to have mean zero. We then look for the **linear combination** of the sample feature values of the form

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \ldots + \phi_{p1}x_{ip}$$

that has the largest sample variance, subject to the constraint that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.

The first principal component loading vector solves the optimization problem

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1}x_{ij} \right)^2 \right\}$$

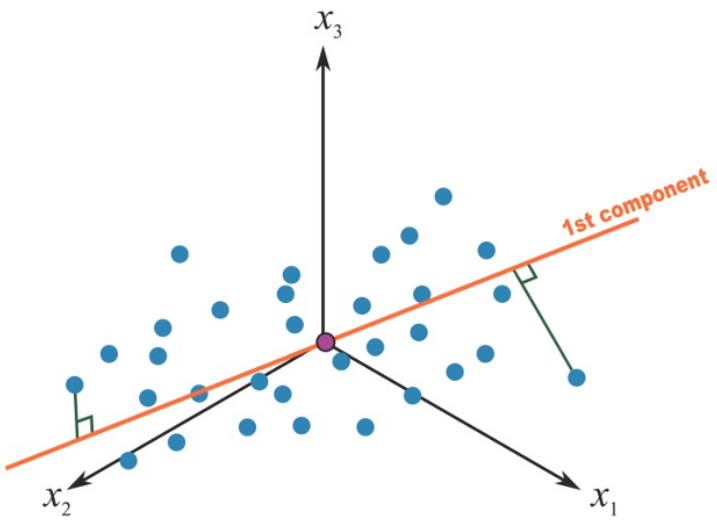
subject to $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.

We refer to $z_{11}, \ldots, z_{n1}$ as the **scores of the first principal component**.

Principal Component Analysis

There is a nice **geometric interpretation** of the first principal component.

The loading vector ϕ_1 with elements $\phi_{11}, \phi_{21}, \ldots, \phi_{p1}$ defines a direction in feature space along which the data vary the most.
If we project the n data points $x_1, \ldots, x_n$ onto this direction, the projected values are the principal component scores $z_{11}, \ldots, z_{n1}$ themselves.



Principal Component Analysis

After the first principal component Z_1 of the features has been determined, we can find the second principal component Z_2.

The second principal component is the linear combination of $X_1, ..., X_p$ that has maximal variance out of all linear combinations that are uncorrelated with Z_1.

The second principal component scores $z_{12}, z_{22}, ..., z_{n2}$ take the form

$$z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + ... + \phi_{p2}x_{ip}$$

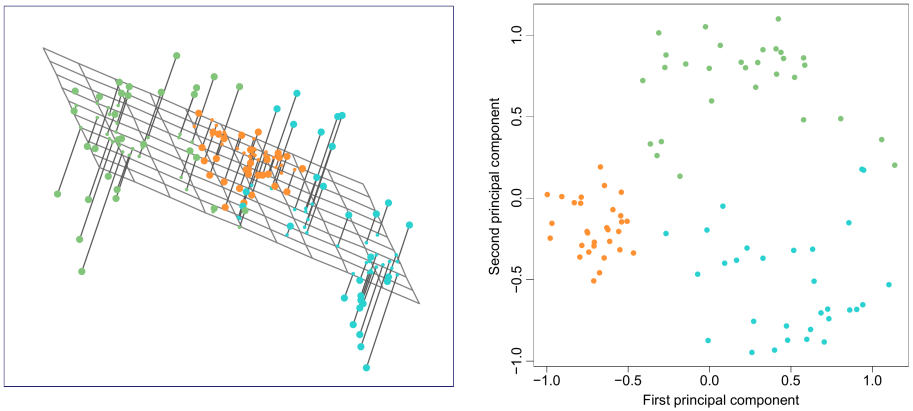
where ϕ_2 is the second principal component loading vector, with elements $\phi_{12}, \phi_{22}, ..., \phi_{p2}$.

It turns out that constraining Z_2 to be uncorrelated with Z_1 is equivalent to constraining the direction ϕ_2 to be **orthogonal** to the direction ϕ_1.

Principal Component Analysis

To find ϕ_2, we solve a similar problem, under the constraint $\phi_2 \perp \phi_1$.

The plot below shows the first two principal component loading vectors in a simulated three- dimensional data set. The two loading vectors span a plane along which the observations have the highest variance.
We see that this two-dimensional representation of the three-dimensional data successfully captures the major pattern in the data.



Simulated observations in three dimensions. Colors added to aid visualization. Left: The first two principal component directions span the plane that best fits the data. The plane minimizes the sum of squared distances to each point. Right: The first two principal component score vectors give the coordinates of the projection of the observations onto the plane.
Source: Fig. 12.2 from `https://www.statlearning.com/`

Principal Component Analysis

We have interpreted the principal component loading vectors as the directions in feature space along which the data vary the most, and the principal component scores as projections along these directions.

An **alternative interpretation** of principal components can also be useful: principal components provide **low-dimensional linear surfaces** that are closest to the observations.
The first principal component loading vector has a very special property: it is the line in p-dimensional space that is closest to the n observations (using average squared Euclidean distance as a measure of closeness).

From this interpretation: We seek a single dimension of the data that lies as close as possible to all of the data points, since such a line will likely provide a good summary of the data.

Using this interpretation, together the first M principal component score vectors and the first M principal component loading vectors provide the best M-dimensional approximation (in terms of Euclidean distance) to the ith observation x_{ij}.

Principal Component Analysis

How much of the information in a given data set is lost by projecting the observations onto the first few principal components?

That is, how much of the variance in the data is not contained in the first few principal components?

More generally, we are interested in knowing the proportion of variance explained (PVE) by each principal component. The total variance present in a data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

and the variance explained by the mth principal component is

$$\frac{1}{n} \sum_{i=1}^{n} z_{im}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2$$

Principal Component Analysis

Therefore, the PVE of the mth principal component is given by

$$\frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2} = \frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{p} \phi_{jm} x_{ij}\right)^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2}$$

The PVE of each principal component is a positive quantity.

In order to compute the cumulative PVE of the first M principal components, we can simply sum over each of the first M PVEs. In total, there are $\min(n-1, p)$ principal components, and their PVEs sum to one.

Principal Component Analysis

Scaling the Variables

Before PCA is performed, the variables should be centered to have mean zero. Furthermore, the results obtained when we perform PCA will also depend on whether the variables have been individually scaled (each multiplied by a different constant).

This is in contrast to some other supervised and unsupervised learning techniques, such as linear regression, in which scaling the variables has no effect.

(In linear regression, multiplying a variable by a factor of c will simply lead to multiplication of the corresponding coefficient estimate by a factor of $1/c$, and thus will have no substantive effect on the model obtained.)

Principal Component Analysis

Uniqueness of the Principal Components

In theory, the principal components need not be unique. In almost all practical settings they are (up to sign flips).

This means that two different software packages will yield the same principal component loading vectors, although the signs of those loading vectors may differ.

The signs may differ because each principal component loading vector specifies a direction in p- dimensional space: flipping the sign has no effect as the direction does not change.

Similarly, the score vectors are unique up to a sign flip, since the variance of Z is the same as the variance of $-Z$.

Principal Component Analysis

How Many Principal Components to Use?

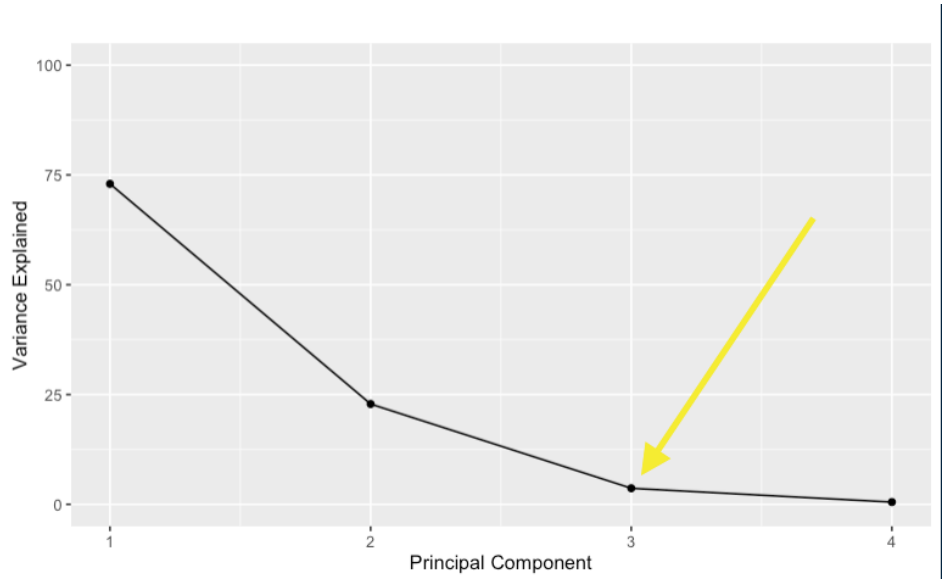
In general, a $n \times p$ data matrix $\mathbf{X}$ has $\min(n-1, p)$ distinct principal components.

However, we usually are not interested in all of them; rather, we would like to use just the first few principal components in order to visualize or interpret the data.
How many principal components are needed? Unfortunately, there is no single (or simple!) answer to this question.

Principal Component Analysis

How Many Principal Components to Use?

We typically decide on the number of principal components required to visualize the data by examining a **scree plot**:



We choose the smallest number of principal components required to explain a sizable amount of the variation in the data. We look for a point at which the proportion of variance explained by each subsequent principal component drops off. This drop is often referred to as an elbow in the scree plot.

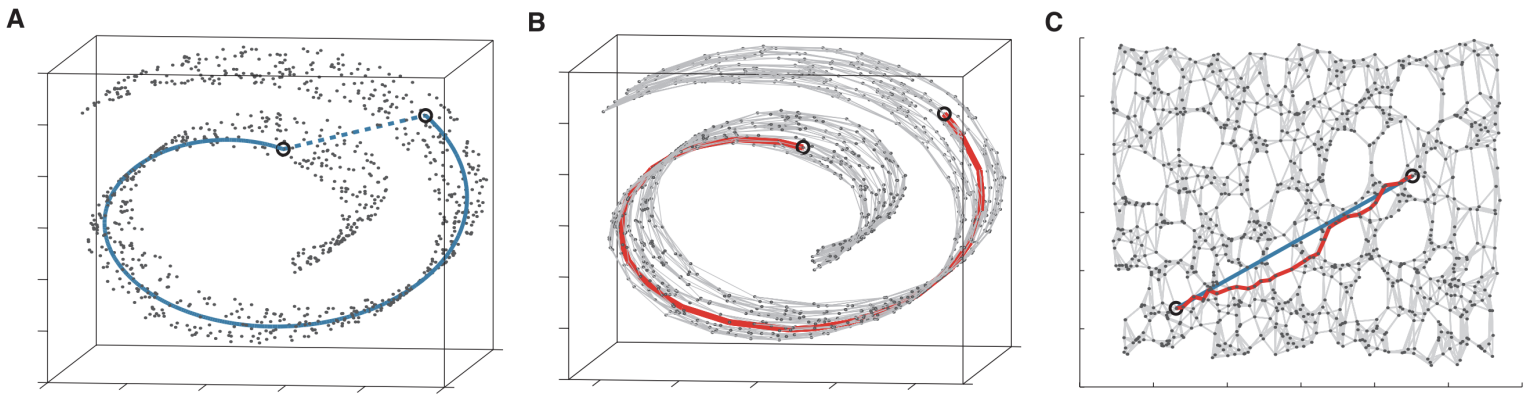
Principal Component Analysis

Caveats:

- PCA is a linear process, whereas the variations in the data may not be. So it may not always be appropriate to use and/or may require a relatively large number of components to fully describe any non-linearity.
- PCA can be very impractical for large data sets which exceed the memory per core as the computational requirement goes as $\mathscr{O}(D^3)$ and the memory requirement goes as $\mathscr{O}(2D^2)$.

Isometric Mapping

Isometric Mapping (IsoMap) is a dimensionality reduction technique based on the multi-dimensional scaling (MDS) framework. Rather than using Euclidean distances between points, IsoMap approximates geodesic curves within in the embedded manifold to compute the distances between each point in the data set.



left: the Euclidean distance (dashed line) not truely reflects how far apart they are on the embedded manifold (solid line)

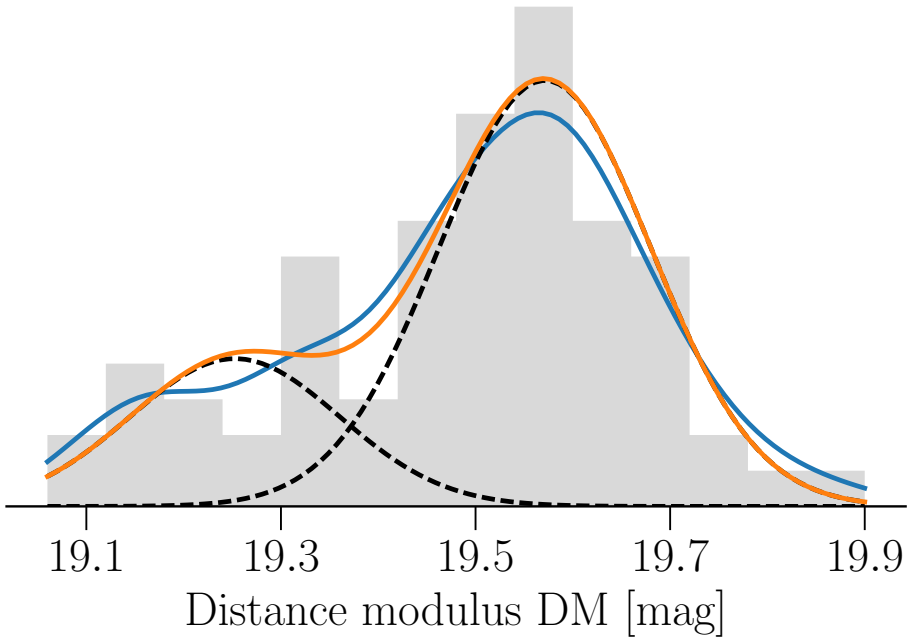
center: a neighboorhood graph is constructed, allowing the geodesic distance between points to be computed by finding the shortest path

right: a lower dimensional embedding of the manifold is recovered by IsoMap that preserves the relative geodesic distances between points

Density Estimation

Inferring the probability density function (pdf) of a sample of data is known as **density estimation**. Essentially, we are smoothing the data to correct for the finiteness of our sample and to better recover the underlying distribution.

Gaussian Mixture models are a case of **Parametric Density Estimation**.



Gaussian and Uniform Distribution

Usually distributions aren't exactly Gaussians, but can be modeled as superpositions of Gaussians, uniform distributions...

example: Gaussian distribution embedded in a uniform background distribution

such distributions are common in physics and astronomy:

spectral lines superimposed upon a background:

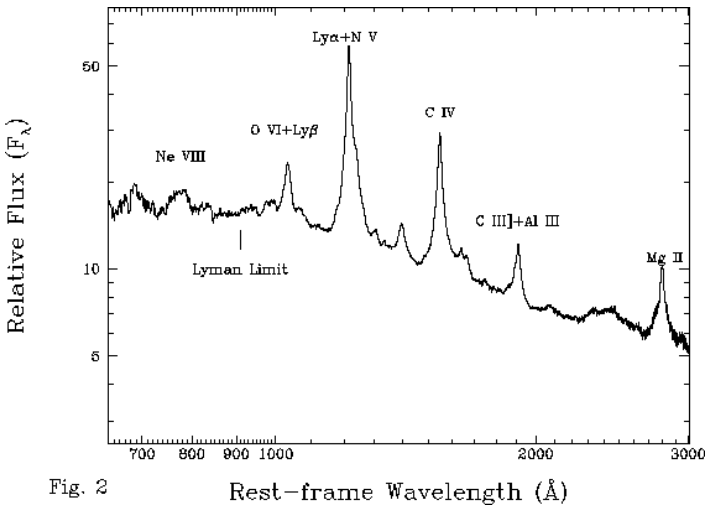
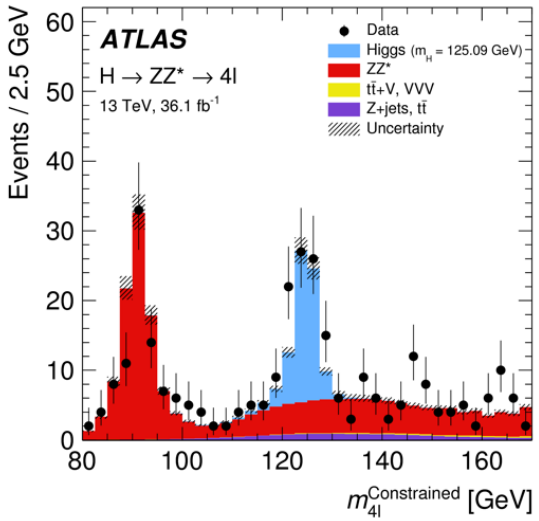


Fig. 2

Higgs boson peak embedded in background noise and other particles:



Gaussian and Uniform Distribution

We assume that

- the location parameter, μ, is known (say from theory) and
- the uncertainties in x_i are negligible compared to σ.

Gaussian and Uniform Distribution

We assume that

- the location parameter, μ, is known (say from theory) and
- the uncertainties in x_i are negligible compared to σ.

The **likelihood** of obtaining a single measurement, x_i, can be written as a probabilistic mixture of either the Gaussian or the uniform distribution:

$$p(x_i|A,\mu,\sigma,I) = \frac{A}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right) + \frac{1-A}{W}.$$

in detail:

- Here the background probability is taken to be $0 < x < W$ and 0 otherwise.
- The feature of interest lies between 0 and W.
- A and $1-A$ are the relative strengths of the two components, which are obviously anti-correlated.
- Note that there will be covariance between A and σ.

Gaussian and Uniform Distribution

The **likelihood** of obtaining a single measurement, x_i, can be written as a probabilistic mixture of either the Gaussian or the uniform distribution:

$$p(x_i|A,\mu,\sigma,I) = \frac{A}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right) + \frac{1-A}{W}.$$

We adopt a uniform **prior** in both A and σ:

$$p(A,\sigma|I) = C, \text{ for } 0 \leq A < A_{\max} \text{ and } 0 \leq \sigma \leq \sigma_{\max}.$$

Gaussian and Uniform Distribution

The **likelihood** of obtaining a single measurement, x_i, can be written as a probabilistic mixture of either the Gaussian or the uniform distribution:

$$p(x_i|A,\mu,\sigma,I) = \frac{A}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right) + \frac{1-A}{W}.$$

We adopt a uniform **prior** in both A and σ:

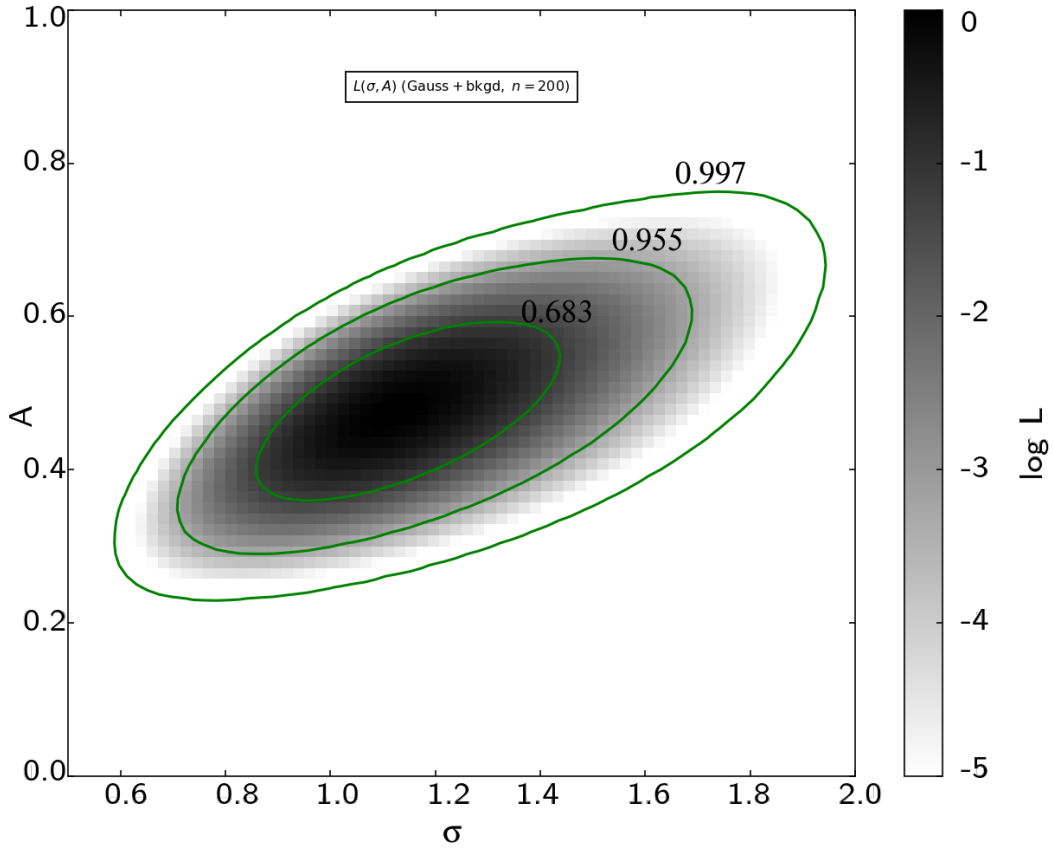
$$p(A,\sigma|I) = C, \text{ for } 0 \leq A < A_{\max} \text{ and } 0 \leq \sigma \leq \sigma_{\max}.$$

The **posterior pdf** is then given by

$$\begin{aligned}\log L &= \ln[p(A,\sigma|\{x_i\},\mu,W)] \\ &= \sum_{i=1}^{N} \ln\left[\frac{A}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right) + \frac{1-A}{W}\right].\end{aligned}$$

Gaussian and Uniform Distribution

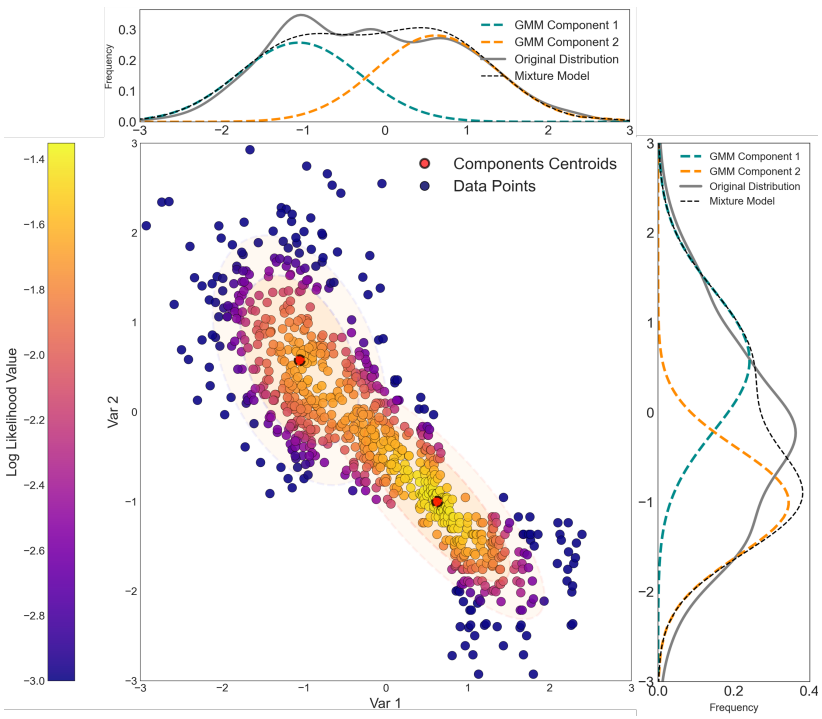
The example below is $\log L$ with $A = 0.5$, $\sigma = 1$, $\mu = 5$, $W = 10$, evaluated on a grid:



Gaussian Mixture Models

A **Gaussian Mixture** is a function that is comprised of several Gaussians, each identified by $k \in \{1, ..., K\}$ where K is the number of clusters of our dataset. Each Gaussian k in the mixture has the following parameters:

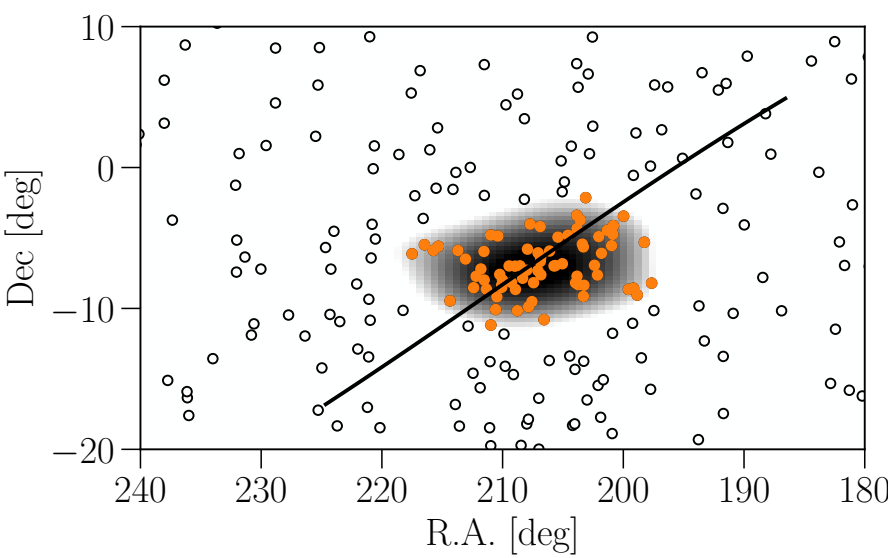
- A mean μ_k that defines its centre.
- A covariance Σ_k that defines its width.
- A mixing probability π_k that defines its amplitude.

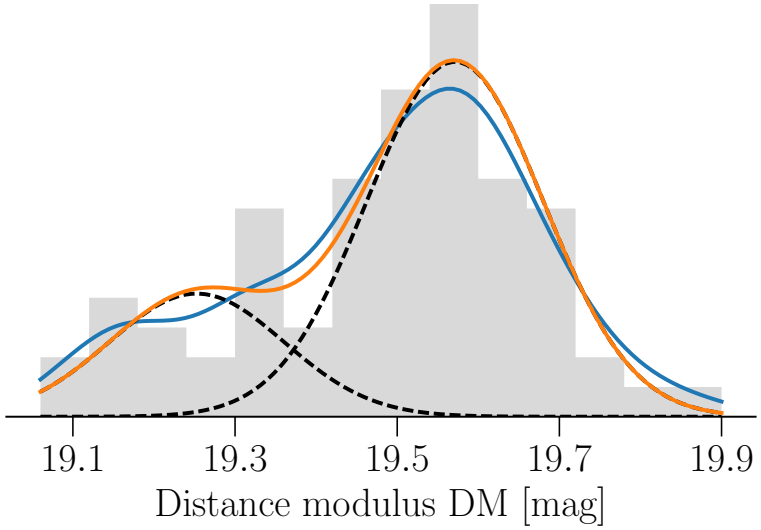


Source: `https://geostatisticslessons.com/lessons/gmm`

Gaussian Mixture Models

example:





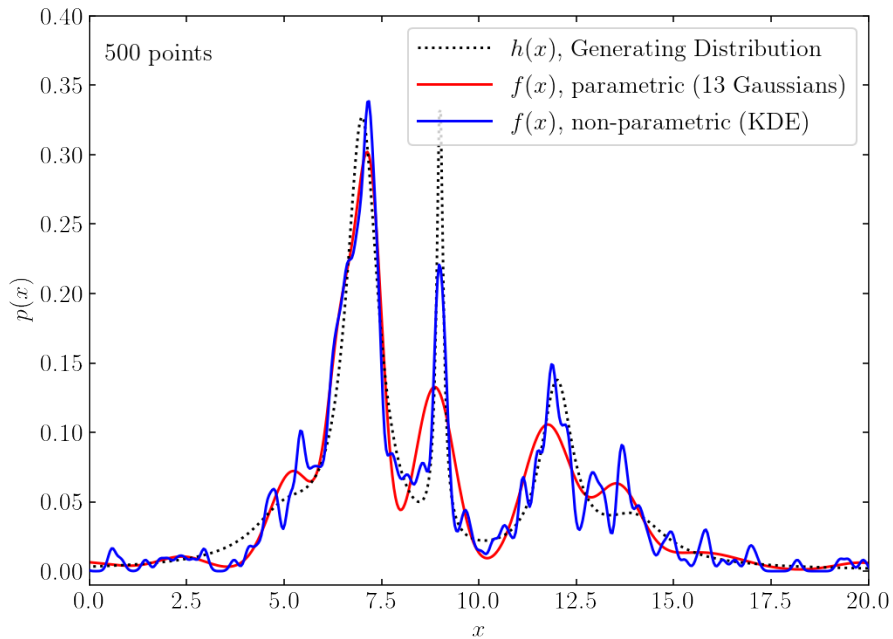
Spatial distribution of PS1 3π RRab stars in the vicinity of the newly discovered Outer Virgo Overdensity (orange solid circles in the top panel). Density was obtained by running a Gaussian mixture model on the data, optimization with Gaussian kernel density estimation from Python `scikit.learn`.

credit: B. Sesar, N. Hernitschek, M. I. P. Dierickx et al., 2017

Non-parametric Density Estimation

Nonparametric density estimation is useful when we know nothing about the underlying distribution of the data, since we don't have to specify a functional form.

Kernel Density Estimation (KDE) is the standard approach for non-parametric density estimation.



Kernel Density Estimation

Kernel density estimators belong to a class of estimators called **non-parametric density estimators**. In comparison to **parametric estimators** where the estimator has a fixed functional form (structure) and the parameters of this function are the only information we need to store, Non-parametric estimators have no fixed structure and depend upon all the data points to reach an estimate.

Kernel density estimation estimates an unknown probability density function using a **kernel function** K.

Kernel Density Estimation

Let $(x_1, x_2, ..., x_n)$ be independent and identically distributed samples drawn from some univariate distribution with an unknown density f. The **kernel density estimator** for f is

$$\widehat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where K is the **kernel function** and $h > 0$ is a smoothing parameter called the **bandwidth**.

Kernel Density Estimation

Let $(x_1, x_2, ..., x_n)$ be independent and identically distributed samples drawn from some univariate distribution with an unknown density f. The **kernel density estimator** for f is

$$\widehat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where K is the **kernel function** and $h > 0$ is a smoothing parameter called the **bandwidth**.

More generally, to measure the distance using a metric different than the Euclidean one, replacing $(x - x_i)$ with $d(x, x_i)$, and generalizing to D dimensions:

$$\widehat{f}_h(x) = \frac{1}{nh^D}\sum_{i=1}^{n} K\left(\frac{d(x, x_i)}{h}\right)$$

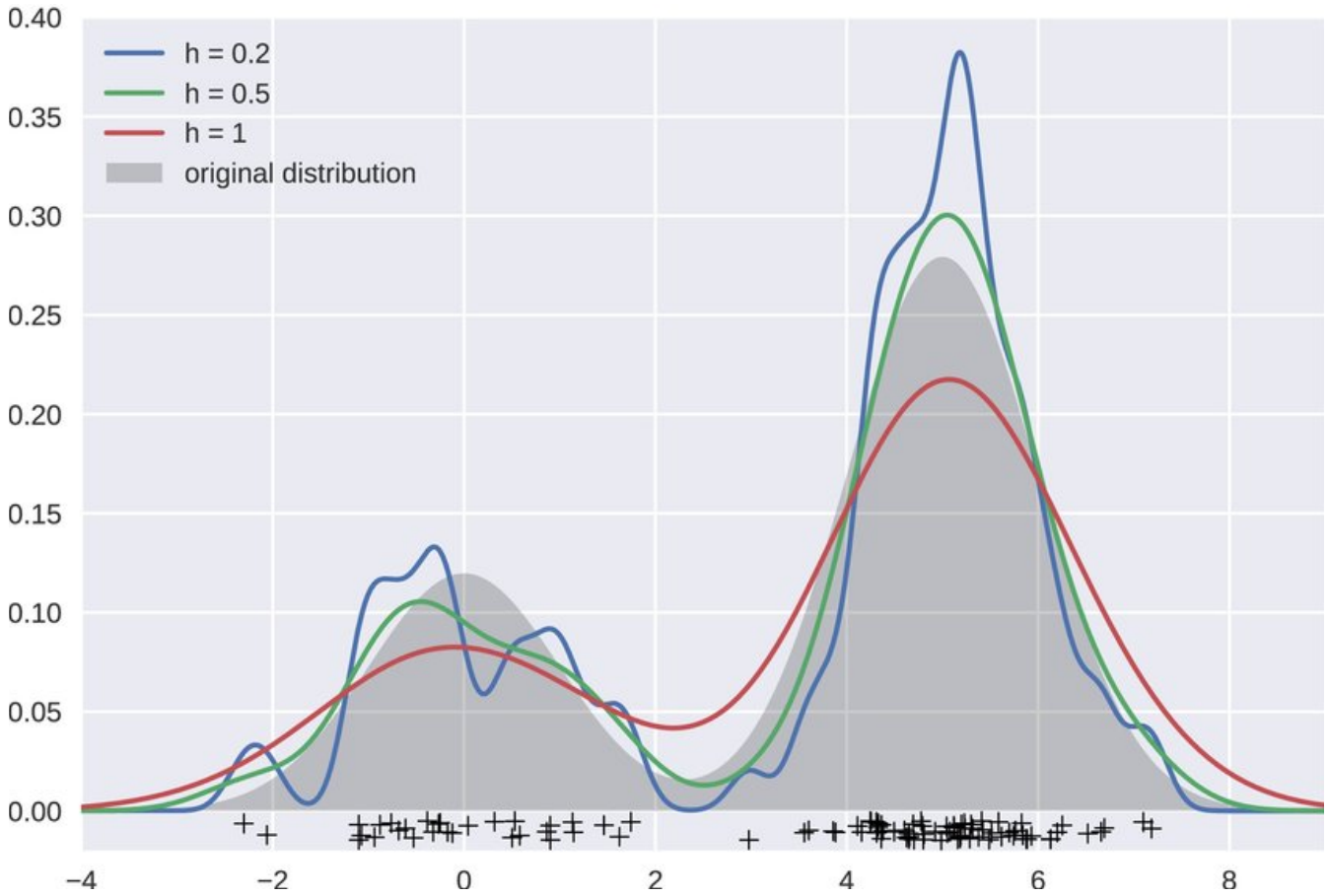
Kernel Density Estimation

The **kernel function** typically exhibits the following properties:

- Symmetry such that $K(u) = K(-u)$.
- Normalization such that $\int_{-\infty}^{\infty} K(u)du = 1$
- Monotonically decreasing such that $K'(u) < 0$ when $u > 0$
- expectation value equals zero such that $E[K] = 0$.

Kernel Density Estimation

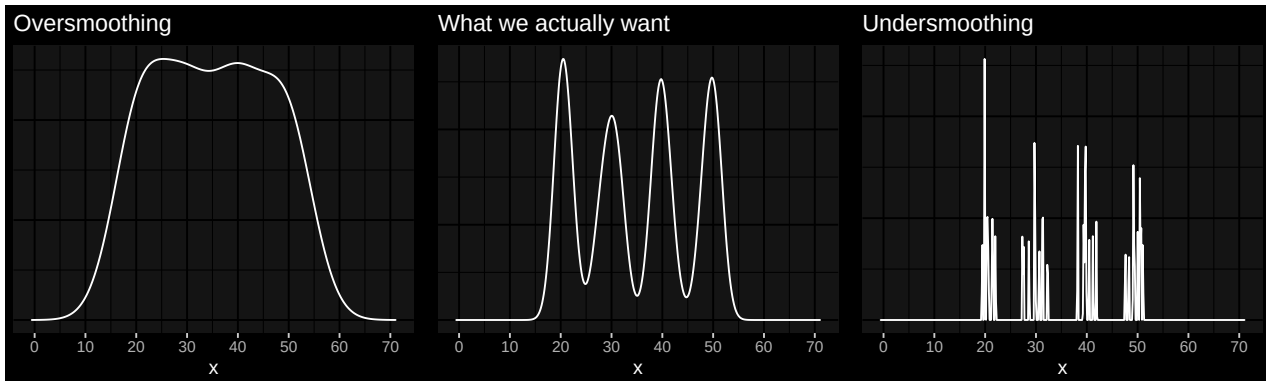
Choice of **bandwidth** h:



Kernel Density Estimation

Choice of **bandwidth** h:

Intuitively one wants to choose h as small as the data will allow; however, there is a trade-off between the bias of the estimator and its variance.



A poorly chosen bandwidth often leads to undesired transformations:

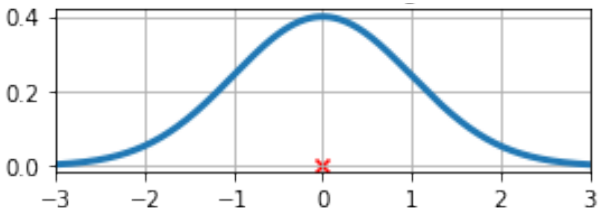
A too small bandwidth leads to undersmoothing:
The density plot will look like a combination of individual peeks (one peek per each sample element).

A too large bandwidth leads to oversmoothing:
The density plot will look like a unimodal distribution, hiding distribution properties (e.g.: distribution is multimodal, but that's invisible in the plot).

Kernel Density Estimation

A common kernel is the **Gaussian kernel**:

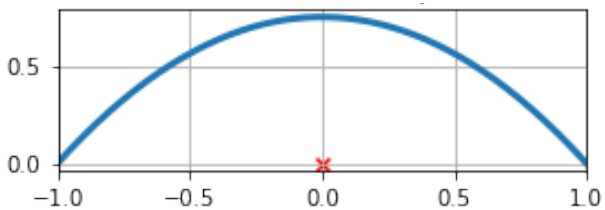
$$K(u) = \frac{1}{(2\pi)^{D/2}} \exp\left(-u^2/2\right)$$



where D denotes the dimensionality of the data.

The **Epanechnikov kernel** is 'optimal' because it minimizes the variance of the kernel density estimate:

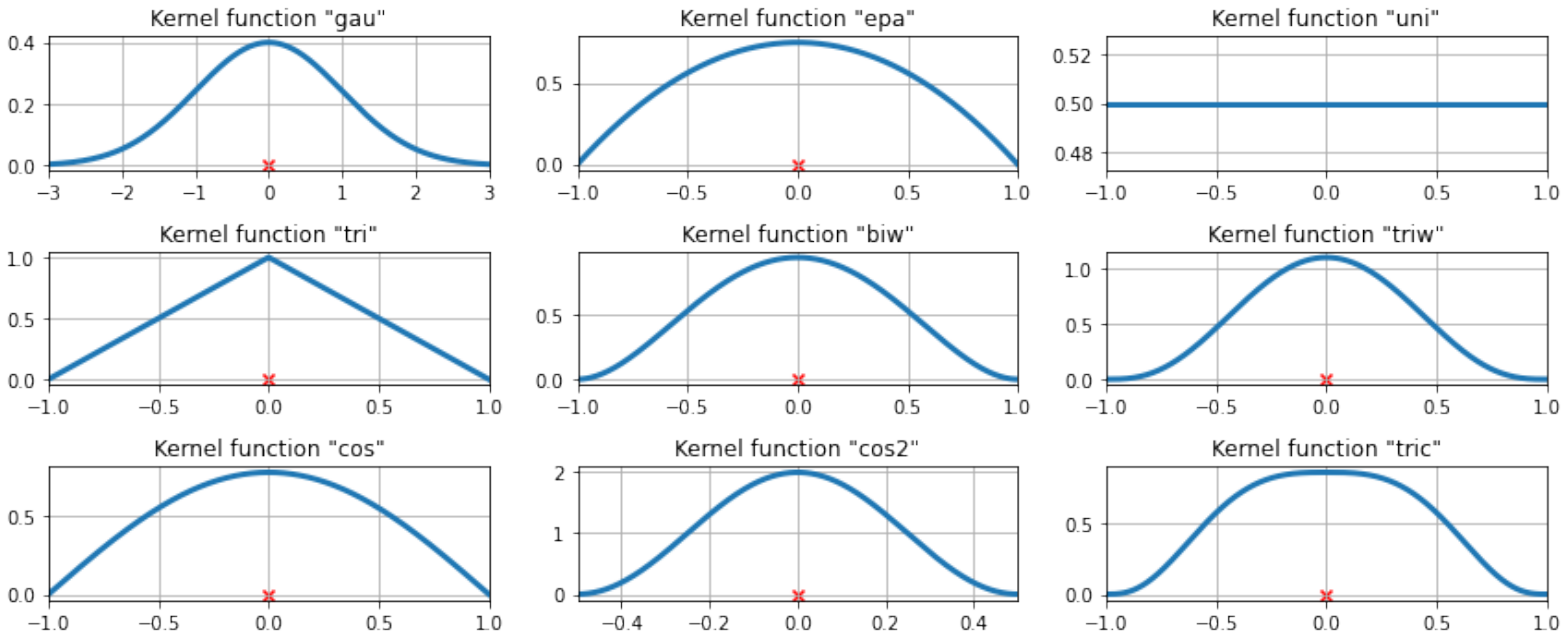
$$K(x) = \frac{3}{4}(1 - x^2)$$



for $|x| \leq 1$ and 0 otherwise.

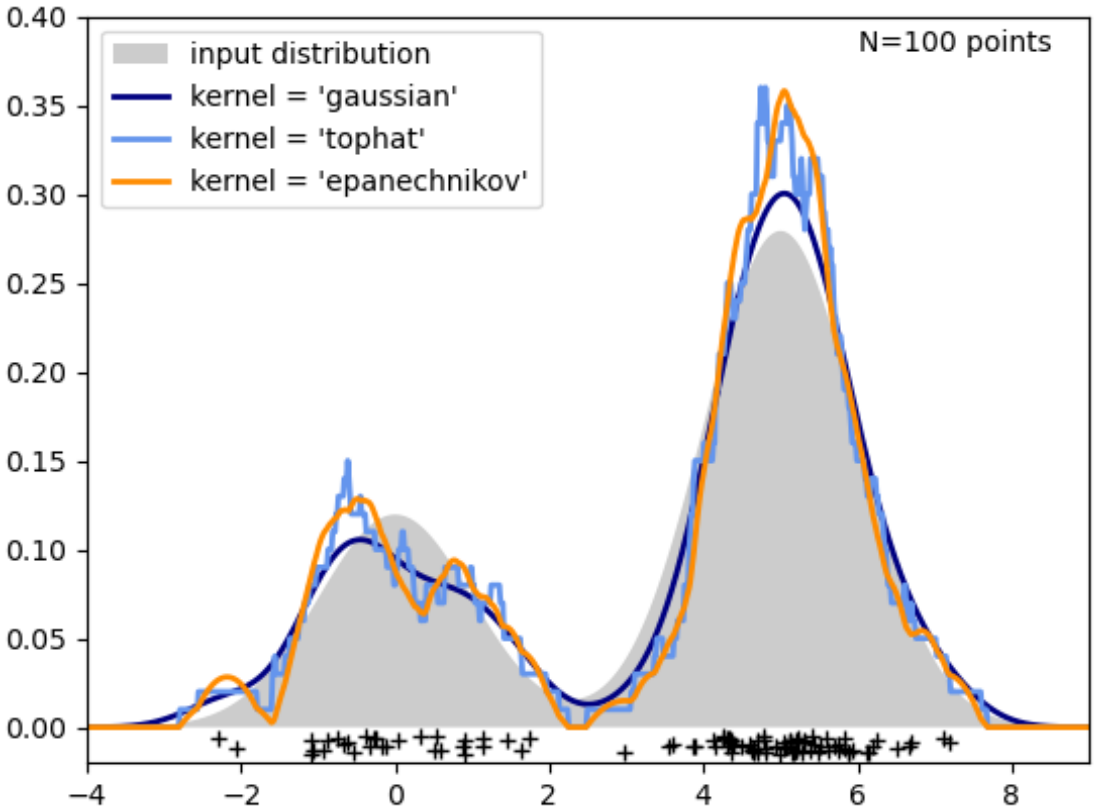
Kernel Density Estimation

several kernels are available from Python `statsmodels.nonparametric.kde`:



Kernel Density Estimation

In the following figure, 100 points are drawn from a bimodal distribution, and the kernel density estimates are shown for three choices of kernels:



Nearest-Neighbor Density Estimation

Another way to estimate the density of an N-dimensional distribution is to look to the K nearest objects and compute their distances, d_K. This is the K-**Nearest Neighbor algorithm**.

The density at a given point x is estimated as

$$\widehat{f}_K(x) = \frac{K}{V_D(d_K)},$$

where $V_D(d)$ is given generically by $\frac{2d^D\pi^{D/2}}{D\Gamma(D/2)}$ with Γ being the gamma function.

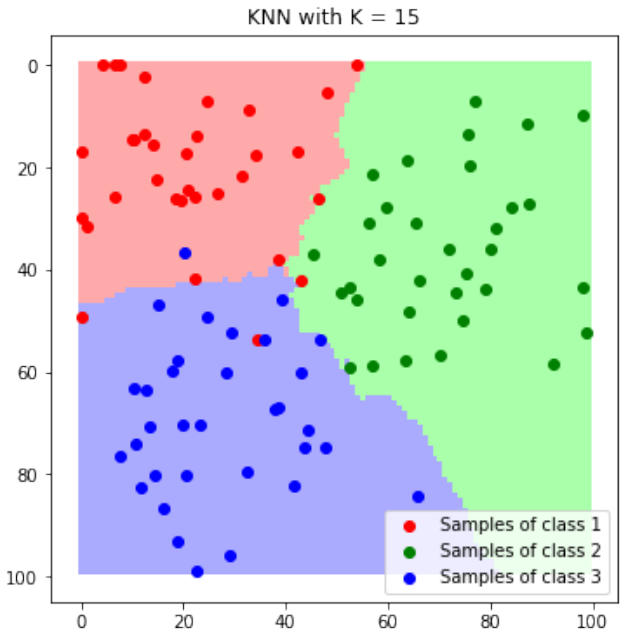
This estimator has some intrinsic bias, which can be reduced by considering all K nearest neighbors:

$$\hat{f}_K(x) = \frac{C}{\sum_{i=1}^{K} d_i^D}$$

Nearest-Neighbor Density Estimation

The KNN's steps are:

1. Receive an unclassified data.
2. Measure the distance from the new data to all other data that is already classified.
3. Get the K smaller distances.
4. Check the list of classes that had the shortest distance and count the amount of each class that appears.
5. Take as correct class the class that appeared the most times.
6. Classify the new data with the class from step 5.

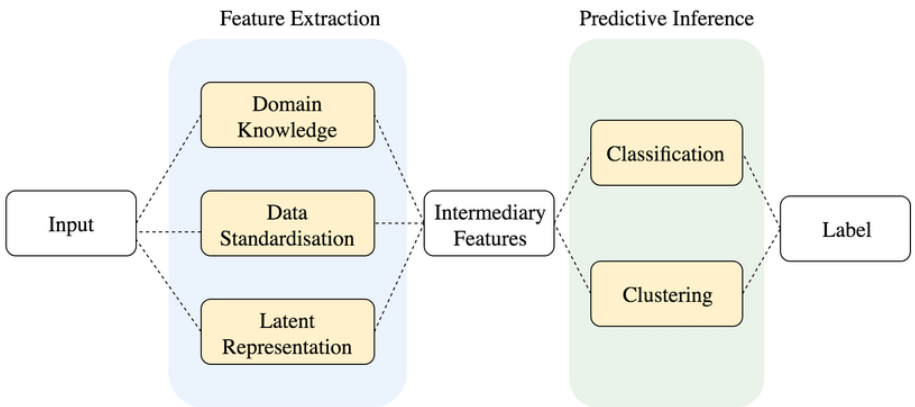


Summary

Unsupervised Learning methods can help us with structuring data to

- identify subgroups among the variables or among the observations
- visualize the data
- carry out explorative data analysis

Often, unsupervised learning methods are used together with supervised learning to form a **machine-learning pipeline**.



General machine learning pipeline that maps an input to a label. The two main steps of the pipeline are (i) extraction of an intermediary feature space and (ii) label prediction using a classification or clustering algorithm.

Summary

This course prepared you to apply machine learning techniques to a research project, and for more advanced courses like **Advanced Machine Learning** (extends the concept of machine learning to neural networks).